The Book of Dojo

Welcome to the Book of Dojo. This book covers both versions 0.9 and 1.0, and all 1.0 extensions and changes are clearly marked for your enjoyment. Please use the forums for support questions, but if you see something missing, incomplete, or just plain wrong in this book, please leave a comment.

For an offline version, click on the Printer-Friendly Page link at the bottom. This assembles the entire book on one long HTML page, which you can then save offline. Caveats: it takes a long time to load, you will get approximately 150 JavaScript errors (which you can ignore) and the graphics don't appear. But all the text and code will be intact.

Table of Contents

Contributors

Simon Bates	David Bolter	Matt Bowen	Pete Brunet	Dipen Chaudhary
Jeff Chimene	Lance Duivenbode	Sam Foster	Becky Gibson	Peter Higgins
Martin Humphreys	Bill Keese	Carla Mott	Brad Neuberg	Shane O'Sullivan
Shelita Overton	Ashish Patil	Adam Peller	Leesa Payne	Craig Riecke, editor
Alex Russell	Gerwood Stewart	Peter Waegener		

Quick Installation

There are three main ways to install Dojo:

1. Install nothing! Use Dojo from AOL's Content Distribution Network (CDN).
2. Install the latest release on your server
3. Install directly from source control

Use Dojo from CDN

This method is quick and painless! You simply load Dojo through <script> tags pointing to the AOL CDN. You don't need to invest any of your own server disk space or resources nor will you need to install Dojo locally in many cases. Instructions are available on using Dojo from the CDN.

All of the Dijit examples in this book load Dojo through this method. You can literally copy and paste any example in Parts 1 and 2 onto your own web server and it will work as-is! In Part 3, where examples are most often code fragments, we'll tell you any modifications needed to run the example on CDN.

Use Dojo from your Own Server

For you traditionalists out there, you can download, install and use Dojo the old fashioned way.

1. Download the latest build from http://dojotoolkit.org/downloads.
2. Uncompress the file onto your web server. Assuming you install it under the directory /js, when you're done, the file system should look something like this:
   [inline:dir_list.png]
3. With your browser, open http://yoursite.com/js/dojo-0.9.0/dijit/themes/themeTester.html You should see a page like this:
   [inline:themeTester.png]

You've got a working Dojo!

Getting the Nightly Build

Finally, for those of you who live on the edge ... you can get the latest, greatest code directly from the Subversion code repository.

Hello World - Dojo for the Attention-Impaired

The purpose of this tutorial is to provide a starting point for users who are new to Dojo. Whilst every effort is made to introduce as much as possible about Dojo, it is impossible to include more of the detailed information since to do so would be counterproductive and confusing to new users. For more information on the concepts introduced here, please see the links to other resources at the end of this document (Finding More Resources).

Requirements

Obviously, you need Dojo first! You can get the latest stable build from http://download.dojotoolkit.org. Next you need a web server. Whether it's hosted offsite or onsite, on Linux or Windows or Mac ... matters naught. The Dojo JavaScript library is simply pulled from your web server to the browser as needed. However, the AJAX examples in this document require a server-side scripting language like PHP or ASP.

The Dojo and Dijit code, which runs on the client browser, is certified to run on IE 6 and 7, Firefox 2, and Safari.

Setting Up Dojo

First, you should create a directory on the web server. We'll call ours HelloWorldTutorial. Then create a directory called dojoroot underneath it. Finally, use your favorite unzipping tool to unzip Dojo into /HelloWorldTutorial/dojoroot. It'll look like this when you're done:

[inline:debugging9.png]

Getting Started

Once we have setup the directory and file structure for the tutorial, we will need to setup the JavaScript component of our HTML page. Have a look at the code below:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<html>
  <head>
    <title>Dojo: Hello World!</title>
    <!-- SECTION 1 -->
    <style type="text/css">
        @import "dojoroot/dijit/themes/tundra/tundra.css";
        @import "dojoroot/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="dojoroot/dojo/dojo.js"
      djConfig="parseOnLoad: true"></script>
  </head>
  <body class="tundra">
  </body>
</html>
```

As it can be seen above, the page is a just a standard HTML skeleton with three things:

- A couple of CSS style sheets. The one marked Tundra is the *theme* we will use from Dijit for this example. There are other themes available.
- A script element inserted into the head section. This script element is responsible for loading the base Dojo script that provides access to all the other Dojo functionality that we will use.
- Lastly, we place the tundra CSS class in the body tag.

Creating a Button Widget

Ok, now for the exciting part! In this example we're going to create a Button widget with the text 'Hello World!'. In the case of the Button widget, three visual states (mouseOut, mouseOver, and mouseDown) are available which means that we are able to enhance the user's experience somewhat.

The first step in creating the widget is telling Dojo to load the appropriate modules. In the header, add another section (hereafter referred to as section 2) below section 1 as follows: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!-- SECTION 2 -->
    <script type="text/javascript">
       // Load Dojo's code relating to the Button widget
       dojo.require("dijit.form.Button");
    </script>
```

The dojo.require line instructs Dojo to load the Button widget. If you were to omit this line, the markup code for the button would not be evaluated by Dojo upon loading, resulting in a plain HTML button instead of what you expect.

After making the changes, insert the following code into the body section of the HTML:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;}

.geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<button dojoType="dijit.form.Button" id="helloButton">Hello World!</button>
```

The key attribute of this HTML element to notice is the dojoType attribute. The dojoType attribute is responsible for instructing Dojo on how to process the element when the page is loading. In this case we've used a button element for the button though we could have used an input element - Dojo will work with either as long as the dojoType attribute is present. It is worth noting that if we did use an input element, we would have to specify the button's text by using adding a caption attribute that contained the desired text.

Connecting an Event to the Widget

A button is all well and good, but what about getting it to do something when it's clicked? We could just specify an onClick event handler for the button, but there's another, more efficient way - the Dojo event system!

The easiest way to attach an event to a button is through a script tag. But not just any script tag ... this one has a type of dojo/method, like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<button dojoType="dijit.form.Button" id="helloButton">
        Hello World!
        <script type="dojo/method" event="onClick">
            alert('You pressed the button');
        </script>
    </button>
    }
```

Pretty simple, eh? Putting the script inside the tag body makes a good deal of sense. And you can harness the full power of DOM Level 2 events inside the script. That means you can detect SHIFT and CTRL keys, get all sorts of event properties, and bubble events up through the HTML tree. If you've ever used Level 2 events, you know how IE and Firefox use different syntax. In Dojo, the same functions work in any supported browser. That's powerful stuff!

Reading Data from the Server

Having an alert pop up when we press the button is great, but what if we want to retrieve some data from the server? Again, Dojo comes to the rescue with an easy method of accomplishing this - dojo.xhrGet. For easy reference, the code for this section is available as HelloWorld-Section5.html and response.txt in the attachments section.

To get started, we first need a callback function to handle the data to be returned from the server. Insert the following code into the header:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script>
       function helloCallback(data,ioArgs) {
          alert(data);
       }
       function helloError(data, ioArgs) {
          alert('Error when retrieving data from the server!');
       }
</script>
```

The two arguments to the functions (data, and ioArgs) are important - don't leave any of them out! The first argument (data) contains the data sent back from the server, whilst the second argument contains a Dojo I/O Bind object. Only the first concerns us right now.

The next step is to link the click of the button to the server request. To do this, modify the following code:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<script type="dojo/method" event="onClick">
    alert('You pressed the button');
</script>
```

To this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<script type="dojo/method" event="onClick">
   dojo.xhrGet({
        url: 'response.txt',
        load: helloCallback,
        error: helloError
   });
```

```
</script>
```

The above code basically tells Dojo to query the URL specified by url and to use the function specified by handler to process the response from the server.

Finally, we need to create another file in the same directory as HelloWorld.html called response.txt. In this file, place the text 'Welcome to the Dojo Hello World Tutorial'.

Now, when the button is clicked, a JavaScript alert should display the text from the response.txt file. Dojo-Easy!

Next, we'll look at doing something meaningful with that server request.

Sending Data to the Server Using GET

It's all well and good retrieving static data from the server, but it is hardly a widely used situation in real life. So, instead of simply requesting data from the server we also will send it some information for it to process. In this section, we'll use the GET method whilst in the next section we'll use the POST method. For easy reference, the code for this section is available as HelloWorld-Section6.html in the attachments section. Server side code is also available as HelloWorldResponseGET. where type is ASP ('.asp'), PHP ('.php'), ColdFusion ('.cfm'), or Java ('.jsp').

Firstly, in the markup section of the HelloWorld.html file (i.e. the body section), we need to add another element - an input element. So, change the code in this section from:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<button dojoType="Button" widgetId="helloButton">
    <script type="dojo/method" event="onClick">
    dojo.xhrGet({
        url: 'response.txt',
        load: helloCallback,
        error: helloError
    });
    </script>
</button>
```

to:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<button dojoType="dijit.form.Button" id="helloButton">
        Hello World!
        <script type="dojo/method" event="onClick">
        dojo.xhrGet({
           url: 'HelloWorldResponseGET.php',
           load: helloCallback,
           error: helloError,
           content: {name: dojo.byId('name').value }
        });
        </script>
     </button>
     Please enter your name: <input type="text" id="name">
```

Before we go any further - it is important to mention that the url property in the dojo.xhrGet function call must be set to the file that is appropriate to your environment. If you are using an ASP server then the value must read 'HelloWorldResponseGET.asp' instead of 'HelloWorldResponseGET.php' Likewise, if you are using a ColdFusion server then the value must read 'HelloWorldResponseGET.cfm' instead of 'HelloWorldResponseGET.php'. Finally, if you are using a Java server (JSP) then the value must read 'HelloWorldResponseGET.jsp' instead of 'HelloWorldResponseGET.php', or if you are using a Perl server then the value must read 'HelloWorldResponseGET.pl' instead of 'HelloWorldResponseGET.pl'. The code for these files is in the sections below, and is also available as attachments to this tutorial.

In the code above, you will notice that there is a new property that has been passed to the dojo.xhrGet function. This property - content - allows the programmer to send arbitary values to the server as parameters. In this case, since we are using the default method of dojo.io.bind which is GET, the server side script will have the value of the textbox available to it as a the GET parameter 'name'. It is worth mentioning that if the script expected the parameter under a different name (such as 'myName'), we would simply change the content property to be (note the change of 'name' to 'myName' on the left of the assignment operator ':'):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
content: {myName: dojo.byId('name').value }
```

Since we've not used it before, it is also worth noting the call dojo.byId('name').value. Quite simply, this call is a shortcut for the standard document.getElementById(..) function.

Finally, if you enter your name into the text box and you click the 'Hello World' button, an alert box should appear with the message 'Hello , welcome to the world of Dojo!' where is the name you entered into the text box.

Here are the server side scripts. A few of them are downloadable at the bottom of this page (the website content management system doesn't allow .jsp or .cfm files).

Using a PHP Server

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #808080; font-style: italic;} .geshifilter .co2 {color: #808080; font-style: italic;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .me1 {color: #006600;} .geshifilter .me2 {color: #006600;} .geshifilter .re0 {color: #0000ff;} .geshifilter .re1 {color: #ff0000}

```
<?php
  /*
  * HelloWorldResponseGET.php
  * --------
  *
  * Print the name that is passed in the
  * 'name' $_GET parameter in a sentence
  */
  header('Content-type: text/plain');
  print "Hello {$_GET['name']}, welcome to the world of Dojo!\n";
?>
```

Using an ASP Server

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #990099; font-weight: bold;} .geshifilter .kw2 {color: #0000ff; font-weight: bold;} .geshifilter .kw3 {color: #330066;} .geshifilter .co1 {color: #008000;} .geshifilter .co2 {color: #ff6600;} .geshifilter .coMULTI {color: #008000;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #006600; font-weight: bold} .geshifilter .st0 {color: #cc0000;} .geshifilter .nu0 {color: #800000;} .geshifilter .me1 {color: #9900cc;}

```
<%
  '
  ' HelloWorldResponseGET.asp
  ' --------
  '
  ' Print the name that is passed in the
  ' 'name' GET parameter in a sentence
  '
  response.ContentType="text/plain"
  response.write("Hello " & request.querystring("name") & ", welcome to the world of Dojo!\n")
%>
```

Using a ColdFusion Server

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<!---
  /*
  * HelloWorldResponseGET.cfm
  * --------
  *
  * Print the name that is passed in the
  * 'name' GET parameter in a sentence
  */
--->
<cfsetting showDebugOutput="No">
Hello, #url.name#, welcome to the world of Dojo!
</cfsetting>
```

Using a Java Server (JSP)

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<%
  /*
  ' HelloWorldResponseGET.jsp
  ' --------
  '
  ' Print the name that is passed in the
  ' 'name' GET parameter in a sentence
  */
  response.setContentType("text/plain");
%>
Hello <%= request.getParameter("name") %> , welcome to the world of Dojo!
```

Using a Perl Server

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
#!/usr/bin/perl
#
#  ' HelloWorldResponseGET.pl
#  ' --------
```

```
#  '
#  ' Print the name that is passed in the
#  ' 'name' GET parameter in a sentence
#
use strict;
use CGI;
my $cgi = CGI::new();
print $cgi->header(-type => "text/html; charset=utf-8");
print "Hello " . $cgi->param('name') . ", welcome to the world of Dojo!\n";
```

Sending Data to the Server Using POST

Using GET data is all well and good, but sometimes you want to use Dojo to improve the user's experience when using a traditional HTML form. As usual, Dojo has a very nice way of making this easier. Again, the code for these files is in the sections below, and are also available as attachments to this tutorial. Additionally, as with the last section, you will need to change the 'url' property to point to the file that is appropriate to your environment.

First, we need to change the markup in the body of HelloWorld.html from:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<br>
    Please enter your name: <input type="text" id="name">
```

to:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<br>
    <form id="myForm" method="POST">
      Please enter your name: <input type="text" name="name">
    </form>
```

Next we need to change the dojo/method:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="dojo/method" event="onClick">
        dojo.xhrGet({
           url: 'HelloWorldResponseGET.php',
           load: helloCallback,
           error: helloError,
           content: {name: dojo.byId('name').value }
        });
</script>
```

to:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="dojo/method" event="onClick">
   // Don't forget to replace the value for 'url' with
   // the value of appropriate file for your server
  // (i.e. 'HelloWorldResponsePOST.asp') for an ASP server
    dojo.xhrPost({
        url: 'HelloWorldResponsePOST.php',
        load: helloCallback,
        error: helloError,
        form: 'myForm'
   });
</script>
```

As can be seen from the code above, we've changed dojo.xhrGet to dojo.xhrPost. We removed the 'content' property and replaced it with a new property 'form'. This basically informs the dojo.xhrPost function that it needs to use the form 'myForm' as the source for the data in the call.

As with the last section, entering your name and clicking 'Hello World!' should yield a message such as 'Hello , welcome to the world of Dojo!' where is the name you entered into the text box.

Using a PHP Server

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #808080; font-style: italic;} .geshifilter .co2 {color: #808080; font-style: italic;} .geshifilter .coMULTI {color: #808080;

font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .me1 {color: #006600;} .geshifilter .me2 {color: #006600;} .geshifilter .re0 {color: #0000ff;} .geshifilter .re1 {color: #ff0000}

```
<?php
  /*
  * HelloWorldResponsePOST.php
  * --------
  *
  * Print the name that is passed in the
  * 'name' $_POST parameter in a sentence
  */
  header('Content-type: text/plain');
  print "Hello {$_POST['name']}, welcome to the world of Dojo!\n";
?>
```

Using an ASP Server

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #990099; font-weight: bold;} .geshifilter .kw2 {color: #0000ff; font-weight: bold;} .geshifilter .kw3 {color: #330066;} .geshifilter .co1 {color: #008000;} .geshifilter .co2 {color: #ff6600;} .geshifilter .coMULTI {color: #008000;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #006600; font-weight: bold} .geshifilter .st0 {color: #cc0000;} .geshifilter .nu0 {color: #800000;} .geshifilter .me1 {color: #9900cc;}

```
<%
  '
  ' HelloWorldResponsePOST.asp
  ' --------
  '
  ' Print the name that is passed in the
  ' 'name' $_POST parameter in a sentence
  '
  response.ContentType="text/plain"
  response.write("Hello " & request.form("name") & ", welcome to the world of Dojo!\n")
%>
```

Using a ColdFusion Server

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<!---
  /*
  * HelloWorldResponsePOST.cfm
  * --------
  *
  * Print the name that is passed in the
  * 'name' POST parameter in a sentence
  */
--->
<cfsetting showDebugOutput="No">
Hello, #form.name#, welcome to the world of Dojo!
</cfsetting>
```

Using a Java Server (JSP)

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<%
  /*
  ' HelloWorldResponsePOST.jsp
  ' --------
  '
  ' Print the name that is passed in the
  ' 'name' POST parameter in a sentence
  */
  response.setContentType("text/plain");
%>
Hello <%= request.getParameter("name") %> , welcome to the world of Dojo!
```

Using a Perl Server

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
#!/usr/bin/perl
#
#  ' HelloWorldResponsePOST.pl
#  ' --------
#  '
#  ' Print the name that is passed in the
#  ' 'name' POST parameter in a sentence
#
use strict;
use CGI;
my $cgi = CGI::new();
print $cgi->header(-type => "text/html; charset=utf-8");
print "Hello " . $cgi->param('name') . ", welcome to the world of Dojo!\n";
```

Finding more resources

I hope you've enjoyed this tutorial and found it informative. No doubt though, you will need more information on Dojo and how it and it's widgets work. Below is a list of links that will point you in the right direction.

- The documentation page for Dojo. Has links to all documentation.
- The Dojo event system. How you can link functions to the normal JS events the Dojo way.
- Dojo XmlHttpRequest - The foundation for AJAX in Dojo.
- The File Upload Widget - Widgets! Components! Does the complete tour of the creation of a fully functional UI component and it's usage.
- Dojo Unit Tests from current nightly build (good for learning how things work)
- Dijit Unit Tests from the current nightly build (good for learning how widgets work)

Contacting the Author

Thinking of making modifications to this document? Want to make suggestions / constructive criticism?

If so, please contact me (Lance Duivenbode) at dojo AT duivenbode DOT id DOT au. Feedback is always welcome since it helps me improve my documentation - both now and in the future. Thanks!

Changelog

- 17th November 2007 - Pulled kicking and screaming into the Dojo 1.0 era (Craig Riecke)
- 28th June 2006 - Addition of Perl Server examples for GET and POST (courtesy of Gareth Tansey)
- 21th June 2006 - Modification for compatibility with 0.3.x release (Bill Keese)
- 22th May 2006 - Addition of Java Server (JSP) examples for GET and POST (courtesy of Kexi)
- 19th May 2006 - Addition of ColdFusion examples for GET and POST (courtesy of Matthew Reinbold)
- 8th May 2006 - Initial Public Release

Debugging Tutorial

Dojo is very lean and speedy, and uses some very clever tricks to save memory and time. The downside: Dojo does not contain much error trapping, which would bloat and slow down the code. If your code is not behaving, some of the resulting errors can look mighty puzzling at first.

No worries! Here are a few tips to make you a confident, successful bug finder. Knowing these ahead of time will make your Dojo learning curve less steep.

Use Firebug or Firebug Lite

We can't stress this enough. Firebug, an open source debugging extension for Firefox, is *essential* for JavaScript, HTML and CSS debugging. You can download it from the Firebug web site.

Not a Firefox user? You may want to consider switching, at least for development work. One of Dojo's virtues is its hiding of cross-platform differences. So the more you rely on it, the more your code will be portable, and the less difference your development browser choice matters.

Still not convinced? That's OK too. If you use Internet Explorer or Safari, you can use the Firebug Lite library, bundled with Dojo. This gives you some of the logging and command line features of Firebug. It's not a full emulation, but it's a fairly good alternative and is fully API-compatible.

To use Firebug Lite, you must include the isDebug config parameter like so:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0//dojo/dojo.js"
        djConfig="parseOnLoad: true, isDebug: true"></script>
```

This parameter has no effect on Firefox browsers with Firebug already present. So including this parameter makes your debugging code usable in IE, Safari, and Firefox with no changes.

Faulty dojo.require's and the Firebug Console

The following code has a subtle bug:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Fix me!</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
```

```
    </style>
        <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.js"
                djConfig="parseOnLoad: true, isDebug: true"></script>
        <script type="text/javascript">
                dojo.require("dojo.parser");
                dojo.require("dijit.form.Textbox");
    </script>
</head>
<body class="tundra">
<form>
What's the 411? <input type="text" size="20" name="info"  dojoType="dijit.form.TextBox"
           trim="true" propercase="true" /><br>
</body>
</html>
```

Without Firebug, this code may pop up an unhelpful dialog box (IE) or display nothing (Firefox). In either case the textbox doesn't look right. You check this by entering a lowercase name and tabbing out of the box ... the propercase attribute should capitalize the first letter. It doesn't.

With Firebug the error is easier to spot. First, there's an indicator at the bottom right hand corner.

[inline:debugging1.png]

You click on the "1 Error" message and the Firebug console pops up:

[inline:debugging2.png]

Looking back at your code, you notice the capitalization on "Textbox" in the dojo.require is wrong. That's a common mistake, but easily fixed. It's also common to forget the dojo.require, or to misspell the class in the dojoType attribute. In each case, Firebug will set you straight.

Firebug Lite will give you a similar console, but it will appear naturally at the bottom of your browser window. To turn it off, you must set the isDebug flag to false and run the page again.

Errors In Dojo/Method and Dojo/Event Code

The following code has an error:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Fix me!</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
           djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.form.Button");
        var newWidth = '200px';
    </script>
</head>
<body class="tundra">
<div dojoType="dijit.form.Button">
   Click to break!
   <script type="dojo/event" event="onClick">
      this.domNode.style.width = dojo.newWidth;
   </script>
</div>
</html>
```

But when you look at the console, there's no apparent error. If you were programming in other languages, you might use a debugger and set a breakpoint on the "this.domNode" line. Firebug's debugger let's you do that, but not on this particular code. Scripts of type "dojo/event" and "dojo/method" are compiled and interpreted differently than "text/javascript" ones. But there are a couple of alternate debugging methods.

Method 1: Logging

The first method is *logging*, and if you've used modern logging tools like log4j you'll find it familiar. The idea is to write trace messages to a log which you can then use to find variable values or the last executed bit of code.

Why not just use alert() ? The trusty JavaScript alert() is a favorite debugging tool, but it suffers from the following problems:

- If you have many alerts, it's annoying to keep clicking [OK]
- Too much text can make the dialog box overflow the screen
- You must remember to remove every alert() before release
- alert() in a tight loop might make it impossible to stop without killing the browser process
- You cannot easily print object contents or arrays

Clearly alert's just not powerful enough. In Dojo logging, you can associate messages with severity, just like in log4j. The following code illustrates the five severity levels:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:

bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
console.log("Nothing happening");
console.debug("Checking to make sure nothing happened");
console.info("Something might happen.");
console.warn("Something happened, but it's no big deal.");
console.error("Cough cough!");
```

In the Firebug console, the messages will appear like this:

[inline:firebug_logging.png]

In IE, they will appear like this:

[inline:firebug_ie_capture.png]

Another useful method, console.dir() dumps variable contents to the screen. While console.log works fine for strings and integers, console.dir prints more complex variables - objects, arrays, arrays of objects, or whatever. For example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
console.dir([
   {attribute: "last_name", sortDescending: true},
   {attribute: "last_name", sortDescending: true}
]);
```

produces:

[inline:firebug_logging2.png]

So in our example above, we write:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
console.debug("dojo.newWidth is" + dojo.newWidth);
this.domNode.style.width = dojo.newWidth;
```

Running this, we quickly find that dojo.newWidth is undefined. Maybe we spelled it wrong? To quickly find out, we change the debugging statement to:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
console.dir("dojo is" + dojo);
this.domNode.style.width = dojo.newWidth;
```

Nope, there's no property in dojo that looks like newWidth. Finally, we spot our error and change the right hand side to "newWidth." Case closed.

Method 2: The "debugger" Statement

Alternatively you can set a "poor person's breakpoint" in the code. Just insert the debugger; statement, which is a legal JavaScript reserved word.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
debugger;
this.domNode.style.width = dojo.newWidth;
```

This statement stops the code and brings you to a Firebug command prompt. It appears the code has stopped at ... huh?

[inline:debugging4.png]

That's a side effect of running dojo/event code. The breakpoints don't seem correct at all.

But just click the Console tab and now you can examine variables or execute just about any JavaScript you want. In this case, we look at the dojo.newWidth property, which has nothing in it. But "dojo" does and we examine it by console.dir(dojo). Basically all the logging features of method 1 are available to type here.

[inline:debugging3.png]

To Follow The dojo.require Trail, Use Dojo Locally

Since that code is now running, we try a minor variant which sets the button to blue:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Fix me!</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.form.Button");
    </script>
</head>
<body class="tundra">
<div dojoType="dijit.form.Button">
   Click to break!
   <script type="dojo/event" event="onClick">
      this.domNode.style.backgroundColor = dojo.Color.named.aliceblue;
   </script>
</div>
</html>
```

You check the console ... no errors there. But that dojo.Color.named.aliceblue is a little questionable. You know that dojo.colors needs to be included, but you thought dijit.form.Button already did that.

You can find out for sure by using a local copy of Dojo. CDN Dojo is very quiet about the modules it loads. Local Dojo is very noisy. So, assuming our local copy of Dojo is installed on the web server underneath /dojoroot, the following change:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
```

Yields the following on the console

[inline:debugging5.png]

You see every Dojo Core and Dijit Component loaded. Sure enough, dojo.colors is not in the list, so we add a dojo.require statement

That Doesn't Look Right ... DOM Inspection

Unfortunately, that doesn't fix the problem either. When styling errors occur, it's a good time to use Firebug's DOM Inspector. You can think of it as View Source on steroids.

- It displays the *current* DOM tree, not the one initially loaded (which is what View Source shows)
- You can examine the DOM properties of nodes by inspecting them - that is, pressing Inspect and pointing

So we click Inspect and point at the screen button

[inline:debugging6.png]

The right-hand side of the console tells what styles and style rules are applied to this class. Crossed-off lines are styles that have been overriden. Very nice!

Debugging External Classes With debugAtAllCosts

dojo/method and dojo/event scripts are good for short, non-reusable snippets of code. But when you start building reusable components, you'll be storing your code into Dojo-declared classes instead. The good news is the more you make this switch, the easier your debugging task will be.

So here's a piece of HTML code and a reusable Dojo-based widget:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Goolica Tax Form</title>
    <style type="text/css">
        @import "/dojoroot/dijit/themes/tundra/tundra.css";
        @import "/dojoroot/dojo/resources/dojo.css"
    </style>
```

```
    <script type="text/javascript" src="/dojoroot/dojo/dojo.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dojobook.online-book.debugging.BuggyWidget");
    </script>
</head>
<body class="tundra">
    <div dojoType="dojobook.online-book.debugging.BuggyWidget"></div>
</body>
</html>
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.provide("dojobook.online-book.debugging.BuggyWidget");
dojo.require("dijit._Widget");
dojo.declare(
"dojobook.online-book.debugging.BuggyWidget",
[dijit._Widget],
{
   postCreate: function() {
      dojo.nonExistentMethod();
   }
});
```

Running this code, you will see an error appear, but it's nowhere near the right location:

[inline:debugging8.png]

But by simply setting the debugAtAllCosts flag to true:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript" src="/dojoroot/dojo/dojo.js"
            djConfig="parseOnLoad: true, debugAtAllCosts: true"></script>
```

the displayed error location will now be correct:

[inline:debugging7.png]

Important! you should always remove debugAtAllCosts from production code. It slows down the client unnecessarily. Rather than manually inserting and removing them, I like to delegate that job to a server side language like PHP:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #808080; font-style: italic;} .geshifilter .co2 {color: #808080; font-style: italic;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .me1 {color: #006600;} .geshifilter .me2 {color: #006600;} .geshifilter .re0 {color: #0000ff;} .geshifilter .re1 {color: #ff0000}

```
<?php
$djConfig     = $inProduction ? "parseOnLoad: true" : "parseOnLoad: true, debugAtAllCosts: true";
$loadLocation = $inProduction ? "http://o.aolcdn.com/dojo/1.0.0" : "/dojoroot";
$useXd        = $inProduction ? ".xd" : "";
?>
    <style type="text/css">
        @import "<?= $loadLocation ?>/dijit/themes/tundra/tundra.css";
        @import "<?= $loadLocation ?>/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="<?= $loadLocation ?>/dojo/dojo<?= $useXd ?>.js"
            djConfig="<?= $djConfig ?>"></script>
```

Introduction

Dojo provides a lot of power and attempts to make it digestable in three major layers: Dojo Core, Dijit, and DojoX. This book serves as a guide to these layers, introducing concepts as you need them and working downward from high-level usage to getting your hands dirty in building your own widgets, custom namespaces, and unit tests.

As you'll see, Dijit and DojoX build on the solid foundation that Dojo Core provides for all Dojo applications. With Core and Dijit locked into stable APIs and a heavy QA, i18n, and accessibility process many edge-of-the-web features for which Dojo is known are developed in the looser, more organic DojoX project. Throughout the book we will show you how these parts build on each other and use the infrastructure each provides to elegantly solve user experience problems which until now have been difficult for browser-based UIs to address.

Thanks for checking out Dojo and the Dojo Book. If things aren't clear in the book, just comment on the book page in question and we'll try to improve it. It's your applications that have inspired us to build Dojo and the stories of people improving user experiences with the toolkit keep us going, and it's with your help that we are evolving this book.

Licensing

You may use Dojo in commercial software without obtaining a separate license or incurring other obligations.

The Dojo Toolkit is dual-licensed. The preferred license is the Academic Free License v2.1. It is extremely liberal, allows for commercial use, and provides for sub-licensing. All Dojo Foundation projects release their code under the terms of this license. It is almost never necessary to exercise the "dual" portion of the dual-licensing terms since the AFL is extremely permissive.

For users who face the problem of artificial ambiguity the FSF has created regarding the compatibility of the AFL and the (L)GPL, The Dojo Toolkit may alternately be used under the terms of the BSD License. Both the AFL and the BSD licenses meet the licensing goals of the Dojo Foundation.

Dojo's "dual licensing" is different than that of many Open Source projects in that the terms of both licenses are Open Source and extremely permissive. There are no royalties or commercial use clauses to complicate the matter. In almost every case, you will not need to choose anything other than the AFL and in the common case you need not do anything to denote this choice of license. If you have questions regarding Dojo licensing, please do not hesitate to contact Alex Russell, current President of the Dojo Foundation.

The Role of the Dojo Foundation

Dojo is Open Source software, distributed by a non-profit foundation which has been set up for the purpose of providing a vendor-neutral owner of Dojo intellectual property. In order to ensure to users of Foundation projects that there is no ambiguity or hidden liability regarding the use of Foundation code, all contributors are required to provide signed Contributor License Agreements.

All committers on Dojo Foundation projects have a vote in Foundation matters. The Foundation is run by contributors, operates in a transparent way, and is funded exclusively by donations. The licensing goals of the Foundations are briefly covered on the Foundation page.

Third-Party Licenses

Dojo uses code from other open source projects, subject to the terms of their licenses. Those licenses and software copyright notices are listed below:

- Firebug Lite (Dojo Base): BSD License
- Common Locale Data Repository, CLDR (Dojo Base): Unicode License
- dojo.css (Dojo Base): BSD License
- AES Encryption Algorithm (DojoX SQL): BSD License
- MD5 and SHA1 Encruption Algorithms (DojoX Cryptography): BSD License
- Dojo Offline Editor Server Demo (DojoX Offline): Apache License
- CLDR (Dojo Utilities): Apache and Mozilla Public Licenses
- Packer (Dojo Utilities): MIT License
- DOH Sounds (Dojo Utilities): Copyright Original Authors
- ShrinkSafe, Rhino Interpreter (Dojo Utilities): Mozilla Public License 1.1

History

In early 2004 Alex Russell (original creator of netWindows) began looking to hire a collaborator on DHTML projects at Informatica. In the process many members of the DHTML community were contacted, culminating in the April 25, 2004 email titled "Selling the future of DHTML". David Schontzler (Stilleye) spent a summer working at Informatica, and Dylan Schiemann also joined Informatica at that time. The first lines of code contributed to Dojo were done by Alex and Dylan with the support of Informatica. There were many other community members that were active participants in shaping the direction of Dojo, including Joyce Park, Tom Trenka, Mark Anderson, Leonard Lin (who suggested the name Dojo), Aaron Boodman, Simon Willison, Cal Henderson, and Dan Pupius.

After several months of discussions on the ng-dhtml (now dojo-developer) mailing list about licensing, choosing a name, coding conventions, build tools, server configuration, and requirements, work began and the Dojo Foundation was formed. The foundation is a 501(c)6 entity designed to house the code and IP rights and today hosts several other projects as well. By March 2005 contributions from the community began to outweigh those of the core development team and today 5 major releases have been made with over 300,000 downloads of Dojo 0.4.x. Contributions and code have come from more than 60 developers and companies and major users such as IBM, AOL, Sun, SitePen, Bloglines, and others continue to keep Dojo's quality high and the community vibrant.

Why Dojo?

Today there are several high-quality JavaScript toolkits available and several hundred other toolkits of varying quality and completeness. Why, out of that sea of possible options, should you chose Dojo?

- **Breadth and Depth:** Dojo is the "full stack". Instead of cobbling together components from several different sources, Dojo allows each component to build on a trusted set of high-quality building blocks by providing integrated infrastructure and a wide variety of optional modules. These components provide good solutions to common user experience problems and can be easily tweaked to meet your needs. From pane-based layouts to client-side charting and graphing to data binding to a time-tested module system, Dojo is solid infrastructure for delivering great experiences.
- **Quality:** Infrastructure for internationalization and accessibility is woven through the entire fabric of Dojo. Keystrokes are hinted correctly. The components all fit together as a cohesive whole. Everything is customizable easily with CSS, but very little needs to be tweaked to get a great looking UI nearly everywhere. These are the hallmarks of a system which has been designed and tested to deliver great experiences, not only to users, but also to designers and developers.
- **Performance:** Dojo is used on high-profile, high-traffic sites every day and Dojo's build tools are a key reason why. Dojo's package system makes it easy to manage large-scale UI development projects and the build system layers on top to make your applications scream; all without code changes. Dojo also packs high-performance implementations of common utilties into its core, and the rebuild of Dojo for 0.9 focuses heavily on performance and reduced code footprint. The result is a small, tight toolkit that is blazing fast. Dojo's performance alone makes it an ideal platform to extend and build on.
- **Community:** Dojo is an open community. As a result many individuals and companies have been able to come together on a level playing field to build tools that benefit everyone. The licensing of the toolkit is designed to be as apolitical as possible and we work hard to ensure that getting your itches scratched is easy if you are willing to get involved. All development happens in the open and the barriers to entry are intentionally very low. We don't care where you work or how "qualified" you are, all we care is that you want to build products that makes user experiences better. Designer or developer or doc writer, the community Dojo values contributions of every kind and position in the community is commensurate with the quality of the work you do, not political wrangling. We're working to change the notions of who can be contributors in Open Source and we invite you to join us in charting a new path. If you want to build a great product and think you can help us, we want to hear from you.

Dojo vs. Other Toolkits

Several other toolkits are often compared to Dojo. What follows is not a comprehensive comparison, but rather a high-level comparison of the features and design goals of these alternatives and how they compare to Dojo's features, development process, and philosophy.

- **MochiKit:** MochiKit is a high-quality JavaScript toolkit which makes writing JavaScript as pythonic as is reasonably possible while still achieving good performance. It follows many of the same conservative principles about packaging, naming, and global namespace usage as Dojo. Authored primarily by Bob Ippolito, it has great docs and tests but unlike Dojo does not ship with a widget system or extensive component set. Some code is shared between Mochi and Dojo (under CLA, of course). Mochi is not backed by a foundation and code lineage is not verified but it is very liberally licensed.
- **Prototype+Scriptaculous:** These pervasive libraries provide much of the same functionality as Dojo Core but do not share the same conservative philosophy regarding global namespace contention, favoring shorter naming of commonly used functions over other concerns. They feature good documentation and broad community support as well as tight integration with Ruby On Rails (among other frameworks). Scriptaculous provides some controls (auto-complete, sliders, etc.) but is not a widget toolkit and does not have support for building widgets easily. Many add-on libraries are available for Prototype+Scriptaculous but they are not distributed with the library as such. They do not feature a package or build system (aside from their own distributables). Prototype and Scriptaculous are not backed by a foundation and code lineage is not verified but they are very liberally licensed.
- **YUI:** YUI is developed in-house at Yahoo and features extensive, high-quality documentation and examples. Designed for speed and targeted at a population of professional PHP developers, YUI is designed with the needs of Yahoo-scale applications in mind. A growing list of controls is available with the toolkit as are useful CSS normalizing and layout style sheets. No package system is available, but "roll up" files of common functionality are distributed and documentation makes clear what order to load files in. No CSS query or markup-driven widget construction system is available in YUI. YUI has an active community and liberal licensing but external committers are not allowed on the project and Yahoo has not clarified the code lineage and other IP rights around the toolkit. No source control access of any sort is available. YUI is edge-cached on Yahoo's CDN for use by all.
- **JQuery:** A minimalist system focused primarily on operating on existing DOM structures, JQuery features a hybrid XPath/CSS query language (Dojo uses standard CSS 3 queries) and provides a rich set of options and operations on the results of these queries. JQuery packs Ajax, effects, and other utilities into a small core (beating all but MooTools). While there is no widget or package system in JQuery per sae, 3rd party component libraries are available which build on top of JQuery. The JQuery community is highly active, usually helpful, and external contributions and patches are accepted. Good docs for the system are readily available. JQuery is dual-licensed MIT and GPL with all copyrights resting with John Resig. It is not clear how IP rights are assigned to John by other contributors and under what terms. Several frameworks (notably Drupal) integrate JQuery.
- **EXT:** Like Dojo's Dijit system, EXT is a component library. It features a large number of consistent, good looking widgets with an emphasis on pixel-perfect layout and desktop-like UIs across browsers. Originally developed to run on top of YUI and later JQuery, EXT now has it's own low-level library, removing the need for 3rd party dependencies. The EXT community is very active and good documentation is available for the library. It is licensed under the terms of the LGPL and commercial licenses of various forms are available. It is not clear whether or not external contributions are accepted (and under what terms) and anonymous subversion access is limited to those who financially support the project in some way.
- **GWT:** Directly integrating server-side development and client-side development, GWT takes the perspective that JavaScript is a bug to be solved and uses advanced compiler technology to allow developers to write in Java and generate dynamic, JavaScript based UIs in the Google style. The default widget set is a strict subset of those provided by Dijit, but GWT takes great pains to optimize all generated code. A growing trove of add-on libraries are available to enhance the default components. Unlike YUI and EXT, GWT is being run as a real Open Source project, allowing external committers, and doing development in the open while managing IP issues in a very sophisticated manner (CLAs, code review, etc. Much like Apache or Dojo). GWT applications can only be written in Java and are most often deployed on Java containers. Good documentation is available and a thriving community is helpful.

For the sake of comparison, Dojo:

- Allows external committers and uses CLAs (like GWT or Apache) to ensure that there are no IP issues
- Is very liberally licensed and provides anonymous SVN access to everyone. Committer privs are earned
- Provides a relatively rich client-side component set but does not require tight binding to any server-side language ("protocols, not APIs")
- Attempts to provide a balance between on-the-wire size and common-case functionality. Dojo Base is similar in size to Prototype.
- Is very conservative about not stepping on the toes of other code in your pages and preserving the global namespace
- Is edge-cached on AOLs CDN for use by all
- Provides a package system which makes knowing which order to load things in a moot problem
- Allows for incremental enhancement via markup and provides a very easy-to-use widget system for building your own reusable components which can then be easily instantiated via markup.

Part 1: Life With Dojo - Dojo and Dijit Application Examples

Dojo applications look good, but their primary benefit is in helping real people solve real interaction problems in real web application. Dojo makes it easy to design a more usable web experience for the intended audience.

The following personas illustrate how people with different goals and skill sets can make Dojo work for them. We will follow them working through an example. The personas and examples are made-up, but broadly represent who the toolkit is built for and each section of the book is designed to help solve problems for each of them, sometimes more for one than the others, but always for their users:

John Walsh is a Web Developer. He's been out of college for 3 years and he works for a small company that creates web sites for clients. He lives and breathes HTML and CSS. He has some basic JavaScript experience, for example with click handlers. He knows a lot about Photoshop, but if you ask him most days, he doesn't really consider himself to be a Designer. Several of his older co-workers would call themselves Designers and only incidentally Web Developers. John and his co-workers care greatly about how an interface looks, they are completely sold on CSS, and they want their tools to work the way they think they should.

Andy Tso has been doing the "startup thing" for nearly a decade. He's seen it all and is a very discerning consumer of technology. He couldn't get enough of his CS and math courses when he was at Stanford. After graduation Andy didn't really know where he wanted to go, so he started on an advanced CS degree at MIT but dropped when some of his other friends left to found an e-commerce thing in '98. It imploded quickly but by that time he'd caught the startup bug. His current startup is pushing the edges of what you can (or should) do in a browser and when they started investigating Dojo, they saw it wasn't everything they needed, but certainly a good starting point. Andy is the kind of guy who could have written Dojo but is wise enough not to. He might contribute patches, though. Andy's main problem is getting through the gunk to the hard tech docs and giving his junior Developers something to work from.

Laura Allen is an Enterprise IT Developer. She has worked for the same (medium sized) company over the past 15 years as it has been bought out twice and renamed three times. She supports internal development sites and relies on tools and frameworks all day long. For her, Web2.0 is tremendously exciting. She didn't know you could do much of anything in a browser, but things that aren't Java/PHP scare her a bit. She's heard that Microsoft mentioned a toolkit too but her manager saw a Dojo demo at a conference and now he's pushing his teams to investigate Dojo.

Example 1: Why Doesn't Anyone Fill Out Their Tax Forms?

John Walsh lives in the country of Googolica. This country is the first in the world to require its citizens to file their taxes online. Here's the form they use:

2007 Tax Form

The Sovreign Nation of Googolica, In Search We Trust

First Name:
Last Name:
Email Address:

---

1. Please Enter Your 2007 Gross Income
2. Please enter the value from line 1. This is your *2007 tax*
3. Would you like to contribute an extra $3 to the Presidential Campaign Fund?

Submit

To the astonishment of the Googolican government, people have not been using the form. The Parliament investigated. The problem, they first thought, might be the brutal tax percentage - 100%. This theory was quickly dismissed.

The final report concluded the web page is faulty, thus discouraging it use. Among its problems:

- It's ugly. Really ugly.
- The form does nothing to help the user. It does not clean up mistakes. And it reports errors after the form has been submitted, leaving the user to stare at a blank form again.
- Googolica has two official languages, English and Snobol. Snobol-speakers are out of luck. And bifurcating the page into an English and a Snobol version makes development twice as difficult.
- People with special needs - low vision, motor problems that discourage mouse use - are out of luck.
- MORE...

Fortunately, the Googolican IT department employs John Walsh. And John has just downloaded Dojo and Dijit. Things are looking up!

Adding Dijit

Let's look at the HTML code for that form:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<head>
    <title>Taxes, The Surest Thing Next to Death!</title>
</head>
<body>
<h1>2007 Tax Form</h1>
    The Sovereign Nation of Googolica,  In Search We Trust
<form>
    First Name: <input type="text" length="20" name="first"><br>
    Last Name: <input type="text" length="20" name="last"><br>
    Email Address: <input type="text" length="20" name="email"><br>
<hr>
<ol>
    <li>
        Please Enter Your 2007 Gross Income
        <input type="text" length="10" name="grossIncome">
    </li>
    <li>
        Please enter the value from line 1.  This is your <em>2007 tax</em>
        <input type="text" length="10" name="tax">
   </li>
    <li>
        Would you like to contribute an extra $3 to the Presidential Campaign Fund?
        <input type="checkbox" name="campaign" value="Y">
    </li>
    <li>
        Filing Date:
        <input type="text" length="10" name="filingDate">
    </li>
</ol>
```

```
<input type="submit" value="Submit">
</form>
</body>
```

Pretty standard stuff. Using Dojo we can improve the UI just a few lines of Javascript and some extra attributes on our existing markup.

The magic is in *Dijit* - shorthand for "dojo widgets". Dijit widgets perform all sorts of tasks, from embelllishing a form control, to controlling the layout of sections and beyond.

Preliminaries for Dijit

You must add two snippets of code for every page using dijits:

- A HEAD snippet which loads the style sheet and Dojo libraries, then calls functions to load individual dijit types.
- A class to the BODY tag specifying the name of your theme. In our examples, we'll use the "Tundra" theme.

Our examples will use the America Online hosted version of Dojo, so you don't have to install one bit of Dojo code! Just copy and paste them to a file on your server, and they will work. So here's the HEAD snippet:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Taxes, The Surest Thing Next to Death!</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
    </script>
</head>
```

Alternatively, if you're a do-it-yourselfer running Dojo from your local site, you just make a few changes to the header:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Taxes, The Surest Thing Next to Death!</title>
    <style type="text/css">
        @import "/dojoroot/dijit/themes/tundra/tundra.css";
        @import "/dojoroot/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="/dojoroot/dojo/dojo.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
    </script>
</head>
```

All of the remaining examples will use the AOL version of the header. The important thing is: none of the rest of the code changes. You can use AOL CDN while you're trying out Dojo, then just change the first lines when/if you install you're own.

This is a fairly standard block. If you have access to a server-side language like PHP or ASP, it's handy to place the header in a separate file and include it.

Tundra is the default *theme* for dijit, a theme being a standard color and design scheme across elements. Themes are discussed at length in Themes, and dijit comes preloaded with a few very nice ones, or you can create your own.

The second snippet is trivial:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<body class="tundra">
```

But if you don't do it, your widgets will look very strange on screen. They rely almost entirely on CSS.

Turning Ordinary INPUT Tags into Widgets

Dijit introduces a new attribute "dojoType". You simply add that to the tag you wish to "dijit-ize". /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080;

font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<form>
First Name: <input type="text" length="20" name="first"  dojoType="dijit.form.TextBox"><br>
Last Name: <input type="text" length="20" name="last"  dojoType="dijit.form.TextBox"><br>
Email Address: <input type="text" length="20" name="email"  dojoType="dijit.form.TextBox">
Filing Date: <input type="text" length="10" name="filingDate" dojoType="dijit.form.DateTextBox">
<hr>
<ol>
<li>Please Enter Your 2007 Gross Income
<input type="text" length="10" name="grossIncome"  dojoType="dijit.form.TextBox"></li>
<li>Please enter the value from line 1.  This is your <em>2007 tax</em>
<input type="text" length="10" name="tax"  dojoType="dijit.form.TextBox"></li>
<li>Would you like to contribute an extra $3 to the Presidential Campaign Fund?
<input type="checkbox" name="campaign" value="Y"  dojoType="dijit.form.CheckBox"></li>
</ol>
```

We use the dojoTypes dijit.form.TextBox and dijit.form.Checkbox. Now that we know this, we can fill in some code in the head snippet:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<script type="text/javascript">
    dojo.require("dojo.parser");
    dojo.require("dijit.form.TextBox");
    dojo.require("dijit.form.CheckBox");
    dojo.require("dijit.form.DateTextBox");
</script>
```

What Did We Get?

The form is looking better already:

First Name:
Last Name:
Email Address:
Filing Date:

---

1. Please Enter Your 2007 Gross Income
2. Please enter the value from line 1. This is your *2007 tax*
3. Would you like to contribute an extra $3 to the Presidential Campaign Fund?

It is now clear which field has the focus: the border turns dark and the gradient fill turns upside down. A small thing, but very helpful to find the usually-tiny insertion point. The checkbox uses a snazzy checkbox icon. It just looks better.

But what is style without substance? Let's add some...

Validating and Assisting

To solve the data entry problems, we can use Dijit validating text boxes. They either make changes to the input for you - like trimming and casifying - or alert the user of invalid input. We'll do this in a few steps.

Trimming and Changing Case

A few extra attributes for the TextBox dijit help the user along:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<FORM>
First Name:
<input type="text" size="20" name="first"  dojoType="dijit.form.TextBox"
          trim="true" propercase="true" />
Last Name:
<input type="text" size="20" name="last"  dojoType="dijit.form.TextBox"
          trim="true" propercase="true"  />
Email Address:
<input type="text" size="20" name="email"  dojoType="dijit.form.TextBox"
          lowercase="true"/>
```

Setting trim="true" will make dijit.form.TextBox trim the spaces before and after the data value. When the TextBox loses the focus, dojo silently hacks the input and places it back in the box. Trimming is often essential in database work, where leading and trailing spaces often lead to search abnormalities down the road.

The propercase and lowercase attributes are similar. They make the input uppercase-first and lowercase, respectively, when the TextBox loses focus. The propercase attribute only changes lowercase letters - all existing uppercase letters are left alone. Similarly, lowercase changes only uppercase letters.

Currency Input

For valid number input, we employ dijit.form.CurrencyTextBox. This box does little things for you like adding the digit separators (comma and period in the US), and making sure only digits are typed.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<input type="text"
                maxlength="12"
                class="fillwidth currency"
                id="grossincome" name="grossincome"
                value="0.00"
                dojoType="dijit.form.CurrencyTextBox"
                required="true"
                onChange="updateTotals()"
                currency="USD"/>
```

And because we haven't added a require for CurrencyTextBox, we add one to the top: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.require("dijit.form.CurrencyText");
```

Adding Some Help

Googolican citizens complain that the form is too complex. John needs to add some directions, but it seems unfair to punish the intelligent citizens by cluttering up the form. So John decides to include some instructions, tucked away but in easy reach. The Dijit TitlePane is perfect for that.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.TitlePane" open="false"
     title="Directions (click to Expand)" style="width:400px;height:300px">
Proin risus. Nullam rhoncus purus id turpis. Praesent aliquam adipiscing ligula. Aenean lorem ante,
accumsan quis, elementum id, cursus eu, lorem. Fusce viverra. Ut tempor nisi at ipsum. Etiam sed nibh.
</div>
```

Of course, the real text can be filled in later. Latin is not exactly a popular language in Googolica.

For some extra help, John inserts a Dijit Tooltip next to the gross income. Tooltip's can be tied to any DOM node - that is, any HTML tag - with an id attribute:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<img src="symbol_help.gif" id="helpIncome"/>
<div dojoType="dijit.Tooltip" style="display:none" connectId="helpIncome">
    That's how much <b>money</b> you make.
</div>
```

The display:none style is not necessary, but it prevents the tooltip from flickering when the screen paints. Why does this happen? It's the way that Dijit handles pages:

1. First all the HTML is drawn. The dojoType attributes are ignored, since they're not standard HTML.
2. Once the page is drawn, the Dojo parser runs - that's what parseOnLoad does. The parser replaces all the dojoType'd tags with the widget HTML
3. The widget then applies styles - including display attributes and colors - to make it look "right".

You can easily hide the 1st step screen-junk by applying display:none to those tags.

And lastly, don't forget the dojo.require's:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.require("dijit.Tooltip");
dojo.require("dijit.TitlePane");
```

Accessibility (A11y)

The right to pay heavy taxes should not be denied to anyone, regardless of physical ability. John knows how important that is. His selection of Dijit was a good one here, because it handles a lot of the A11y details (A11y = A + 11 letters + y, a standard abbreviation).

One a11y consideration is low vision. Usually this is overcome by high-contrast color schemes. Dijit automatically detects this condition, and renders the page accordingly. You can simulate this by merely changing the body class from "tundra" to "a11y":

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<body class="a11y">
```

Graphic embellishments, like gradients, are eliminated so as not to blend into the important text. And all graphic symbols are placed with clearer text ones, as in the Title Pane:

Proin risus. Nullam rhoncus purus id turpis. Praesent aliquam adipiscing ligula. Aenean lorem ante, accumsan quis, elementum id, cursus eu, lorem. Fusce viverra. Ut tempor nisi at ipsum. Etiam sed nibh.

Internationalization (i18n)

Up until this point, we haven't had to use JavaScript beyond the dojo.require's. That's about to change. But as you'll see, a small amount of JavaScript goes a long way ... when hooked to Dijit and Dojo.

We're going to make the tax form accessible to readers of both languages: English and Swedish Chef. But first here's the **wrong** way to do it: make two pages with the same controls, the same styles, and the same wiring, but with different text on each. Very bad. When one page changes the other needs to change too. To add more languages, the job becomes even worse.

Since only the text changes, it be nice to have one copy of the page with textual placeholders. Dojo i18n is perfect for that.

Bundles and Resources

If you've used i18n in other languages like Java, the concepts are pretty similar here. Your application users are partitioned into "locales" - a language and a location and/or dialect. The ISO defines standard identifiers for locales, so that en-us is English in the U.S. and en-au is English in Australia. In our example, we'll be using a functional locale called "sw-chef". All other locales will map to the default locale, which in our case is just plain ol' English.

A *bundle* is a directory named after a locale, and a Dojo i18n *resource* is a JavaScript file underneath. As is common in other i18n setups, all of our bundles go under the directory "nls". For our i18n setup, we'll use the resource name "taxform". Here's the way it'll look:

[inline:taxform5.png]

The taxform.js underneath nls is the default resource, which looks like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
({
first: "First Name",
last: "Last Name",
email: "Email Address",
filingDate: "Filing Date",
grossincome: "Please Enter Your 2007 Gross Income",
deductibles: "Total Deductions",
netincome: "Taxable Income",
taxpaid: "Total Witholding",
refund: "Your Refund",
owed: "Amount You Owe",
campaign: "Would you like to contribute an extra $3 to the Presidential Campaign Fund?"
})
```

The names before the : are called *keys* and they are one-word abbreviations for each of the text pieces. The sw-chef resource sw-chef/taxform.js is similar:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0

{color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
({
first: "Furst Neme-a",
last: "Lest Neme-a ",
email: "Emeeel Eddress",
filingDate: "Feeling Dete-a",
grossincome: "Pleese-a Inter Yuoor 2007 Gruss Incume-a",
deductibles: "Tutel Dedoocshuns",
netincome: "Texeble-a Incume-a",
taxpaid: "Tutel Veethulding",
refund: "Yuoor Reffoond",
owed: "Emuoont Yuoo Oove-a",
campaign: "Vuoold yuoo leeke-a tu cuntreeboote-a un ixtra $3 tu zee Preseedentiel Cempeeegn Foond? Bork Bork Bork!"
})
```

Now we somehow need to pull these names into the right places on the form. For that, we'll build our own custom widget.

An i18n Field Label Widget

Widgets are a lot like macros. You're basically substituting one "fake" tag with a dojoType - called the *widget class* - for a set of "real" tags. To transform the fake tag to the real tag, you use a *template*. For example, suppose our goal is to get this tag:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<label for="first" id="first_label">First Name</label>
```

To construct the Dijit template, we insert place holders for the parts that will vary:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<label for="${fieldid}" id="${fieldid}_label"></label>
```

Then nestle it inside a dijit.Declaration tag. We can place this tag and template anywhere in our HTML file, in theory, but most people place them under the BODY tag.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<span dojoType="dijit.Declaration" widgetClass="TaxformI18n"
      defaults="{ fieldid: 'none' }">
    <label for="${fieldid}" id="${fieldid}_label"></label>
</span>
```

The widgetClass parameter gives the name TaxformI18n to our widget class. This will match the dojoType in our actual widgets. The defaults parameter defines all the placeholders and their default values. Any placeholder you use in the template must have a default here.

Having declared this, now the tag:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="TaxformI18n" fieldid="first"></div>
```

Will be replaced with

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<label for="first" id="first_label"></label>
```

Well, that's helpful but we still need our translated text. That's where we'll call Dojo.

The Dojo i18n API

The fieldid in our widget template can be used as a key into our resource. Ah ha! So you've got a label tag connected to first, which will be filled in with the text tied to "first" ... e.g. "First Name" or "Furst Neme-a". Given that the key is in "fieldid", here's the JavaScript to get the translation:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:

bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var labelNode = dojo.byId(this.fieldid + "_label");
var taxFormBundle = dojo.i18n.getLocalization("taxformI18n","taxform");
labelNode.innerHTML = taxFormBundle[this.fieldid];
```

The third line puts the translated text into the tag marked "fieldid_label".

So taxform is the resource name, but what is taxformI18n? That's a package name, which we define outside of the widget:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// This is necessary to i18n routines look in this directory for the nls folder
dojo.registerModulePath("taxformI18n","/online-book/taxform");
dojo.requireLocalization("taxformI18n", "taxform");
```

These two statements say "my resources are in http://yourserver/online-book/taxform/nls".

Now for the wiring. The dojo.Declaration must have a template, but it can also contain JavaScript code. You might be tempted to just plunk it in there with a <script type="text/javascript"> but that won't work. Instead you use the type "dojo/connect" and the event "startup." like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<span dojoType="dijit.Declaration" widgetClass="TaxformI18n"
    defaults="{ fieldid: 'none' }">
    <label for="${fieldid}" id="${fieldid}_label"></label>

    <script type='dojo/connect' event='startup'>
        var labelNode = dojo.byId(this.fieldid + "_label");
        var taxFormBundle = dojo.i18n.getLocalization("taxformI18n","taxform");
        labelNode.innerHTML = taxFormBundle[this.fieldid];
    </script>
</span>
```

And we're ready to go!

i18n in Action

So here's the complete listing:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Googolica Tax Form</title>
<style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
</style>
<script type="text/javascript" src="/dojoroot/dojo/dojo.js"
        djConfig="parseOnLoad: true"></script>
<script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.form.TextBox");
        dojo.require("dijit.form.CheckBox");
        dojo.require("dijit.form.DateTextBox");
        dojo.require("dijit.form.CurrencyTextBox");
        dojo.require("dijit.Declaration");
        dojo.require("dijit.TitlePane");
        dojo.require("dijit.Tooltip");

        // This is necessary to i18n routines look in this directory for the nls folder
        dojo.registerModulePath("taxformI18n","/online-book/taxform");
        dojo.requireLocalization("taxformI18n", "taxform");
    </script>
</head>
<body class="tundra">
<span dojoType="dijit.Declaration" widgetClass="TaxformI18n"
        defaults="{ fieldid: 'none' }">
        <label for="${fieldid}" id="${fieldid}_label"></label>

        <script type='dojo/connect'     event='startup'>
        var labelNode = dojo.byId(this.fieldid + "_label");
        var taxFormBundle = dojo.i18n.getLocalization("taxformI18n","taxform");
        labelNode.innerHTML = taxFormBundle[this.fieldid];
    </script>
</span>
<form>
<div dojoType="TaxformI18n" fieldid="first"></div>
<input type="text" size="20" name="first" dojoType="dijit.form.TextBox"
        trim="true" propercase="true" /><br>
<div dojoType="TaxformI18n" fieldid="last"></div>
```

```
<input type="text" size="20" name="last" dojoType="dijit.form.TextBox"
        trim="true" propercase="true" /><br>
<span dojoType="TaxformI18n" fieldid="email"></span>: <input type="text"
        length="20" name="email" dojoType="dijit.form.TextBox" lowercase="true" />
<div dojoType="TaxformI18n" fieldid="filingDate"></div>
<input type="text" length="10" id="filingDate" name="filingDate"
        dojoType="dijit.form.DateTextBox">
<hr />
<ol>
        <li>
        <div dojoType="TaxformI18n" fieldid="grossincome"></div>
        <input type="text" maxlength="12" id="grossincome" name="grossincome"
                value="0" dojoType="dijit.form.CurrencyTextBox" required="true"
                currency="USD" /> <img src="symbol_help.gif" id="helpIncome" />
        <div dojoType="dijit.Tooltip" connectId="helpIncome">That's how
        much <b>money</b> you make.</div>
        </li>
        <li>
        <div dojoType="TaxformI18n" fieldid="deductibles"></div>
        <input type="text" dojoType="dijit.form.CurrencyTextBox"
                id="deductibles" name="deductibles" value="0" required="false"
                currency="USD" /></li>
        <li>
        <div dojoType="TaxformI18n" fieldid="netincome"></div>
        <input type="text" value="0" dojoType="dijit.form.CurrencyTextBox"
                required="false" id="netincome" name="netincome" currency="USD" /></li>
        <li>
        <div dojoType="TaxformI18n" fieldid="taxpaid"></div>
        <input type="text" value="0" class="fillwidth currency"
                dojoType="dijit.form.CurrencyTextBox" required="false" readonly="true"
                id="taxpaid" name="taxpaid" currency="USD" /></li>
        <li>
        <div dojoType="TaxformI18n" fieldid="refund"></div>
        <input type="text" value="0" dojoType="dijit.form.CurrencyTextBox"
                required="false" readonly="true" id="refund" name="refund"
                currency="USD" /></li>
        <div dojoType="TaxformI18n" fieldid="owed"></div>
        <input type="text" value="0" dojoType="dijit.form.CurrencyTextBox"
                required="false" readonly="true" id="owed" name="owed" currency="USD" />
        </li>
        <li>
        <div dojoType="TaxformI18n" fieldid="campaign"></div>
        <input type="checkbox" name="campaign" value="Y"
                dojoType="dijit.form.CheckBox"></li>
</ol>
<div dojoType="dijit.TitlePane" open="false"
        title="Directions (click to Expand)" style="width:400px;height:300px">
Proin risus. Nullam rhoncus purus id turpis. Praesent aliquam adipiscing
ligula. Aenean lorem ante, accumsan quis, elementum id, cursus eu,
lorem. Fusce viverra. Ut tempor nisi at ipsum. Etiam sed nibh.</div>
</body>
</html>
```

The browser knows which locale to use based on the web browser installation, and it feeds this directly to Dojo, by default. So unless your locale is "sw-chef", your page will look exactly like the previous examples. But you can easily override this by changing the configuration when loading dojo.js:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<script type="text/javascript" src="/dojoroot/dojo/dojo.js"
        djConfig="parseOnLoad: true, locale: 'sw-chef'"></script>
```

And run the result.

[inline:taxform6.png]

It makes you want to throw kitchen utensils in the air with glee! OK, one more design run, and our form will be finished.

Example 2: The Postman Always Clicks Twice

Andy Tso gets a call from Regina Opulence, the head of the new Internet Startup clickdammit.com. "I need to have a mail web site," she said, "one like Google Mail. Can you make one up in a week?"

"Hmmmm." said Andy, "What's your business plan, if I may ask?"

"It's brilliant." she said. "Our backend servers will read the mail they send back and forth, then send them ads based on the things they talk about!"

Andy rolled his eyes to the ceiling. It would do no good to talk about privacy issues, legality, or all that other nonsense. He was just a tech guy, and Regina wanted him to do techie-type things. At least Clickdammit.com paid well. "Lemme give it a whirl. I think I can get you a pretty good demo in a week. "

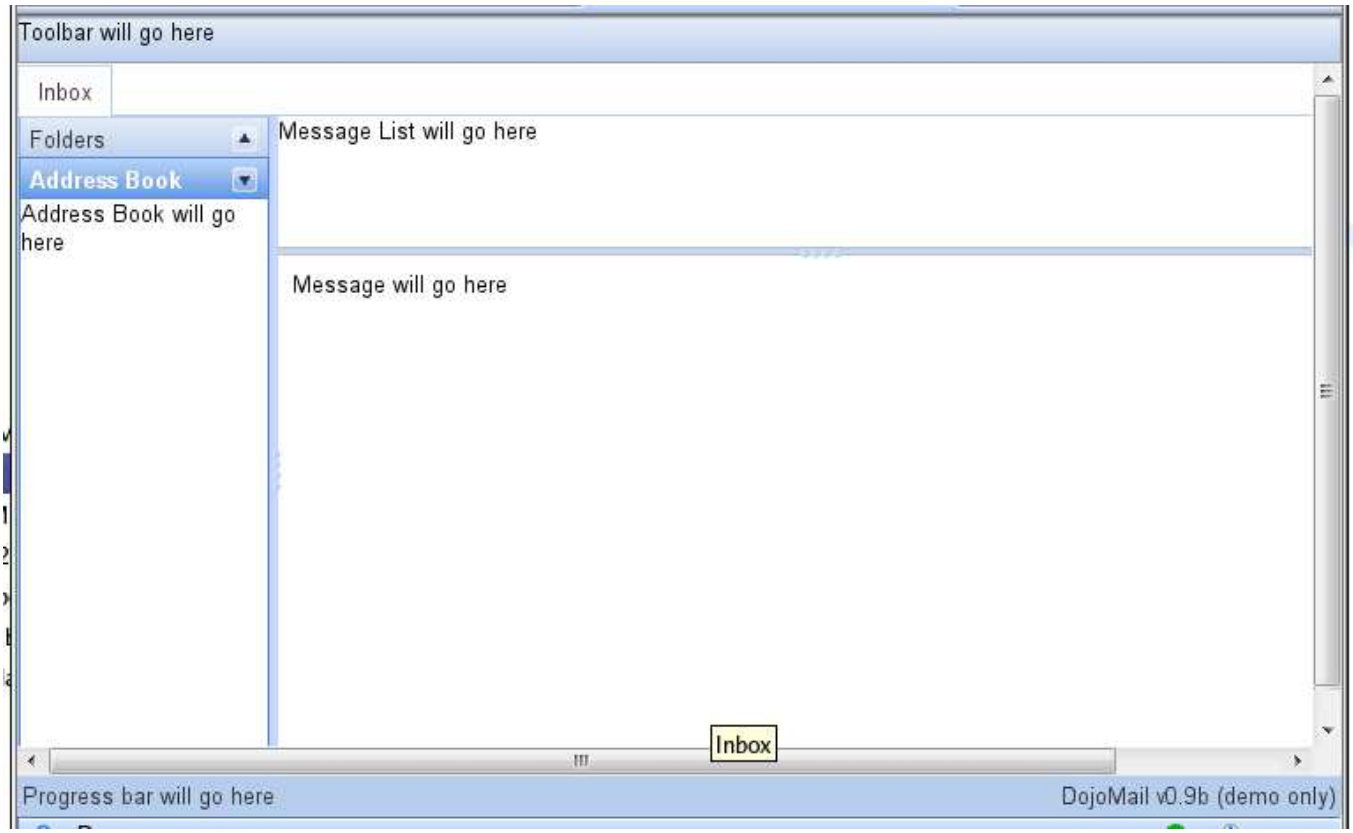
He hung up the phone and wondered ... how could he get this demo done *and* propose to his longtime girlfriend on Friday night? He still had to write the speech, after all.

Fortunately there's Dojo! *

* Also fortunately, Dojo has a demo email client that comes in every package! Written by Bill Keese with contributions from Alex Russell and Peter Higgins, this demo covers a lot of Dojo goodness.

Basic Layout

So here's the cocktail napkin view of the email client:



Like most Outlook-inspired email programs, it will have a three pane, split-screen layout with a toolbar on top and status bar on the bottom. The left hand side will use accordion panes that flip up and down like a window blinds. Fortunately, all the layout tools you need are already in Dijit.

Andy likes to build the layout from the inside-out best, so he'll tackle the steps in this order:

1. Divide the screen into list and messages areas
2. Add the navigation bars - address book, message list, etc. - to the left hand area
3. Add the message preview pane
4. Add the tabbed panes for Compose Mail
5. Glue on the toolbar and progress bar

Dividing the Screen with SplitContainer

start with the message list and message area first. In between these two is the movable divider bar, called the *sizer*. In Dijit, you can use the dijit.layout.SplitContainer widget to model this like so:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.SplitContainer" id="rightPane"
        orientation="vertical"  sizerWidth="5"  activeSizing="0">
        <div id="listPane" dojoType="dijit.layout.ContentPane" sizeMin="20" sizeShare="20">
             Message List will go here
        </div>

        <div id="message" dojoType="dijit.layout.ContentPane" sizeMin="20" sizeShare="80">
            Message will go here
        </div>
</div> <!--  End right hand side split container -->
```

Usually the innermost children in a layout are dijit.layout.ContentPane's. You can place arbitrary HTML in the pane, as we've done here, or lazy load the content from other URL's, like a server-side include.

The dijit.layout.SplitContainer itself can be oriented horiztonally or vertically, as the "orientation" attribute shows. A vertically oriented SplitContainer means the components are stacked vertically. Note that the orientation is *opposite* of the sizer bar orientation - i.e. if the SplitContainer is vertical, as it is in this case, the sizer orientation is horizontal. The sizer bar here is 5 pixels and changing it does not change the content immediately, as the activeSizing attribute shows.

The Navigation Bars: AccordionContainer

The left hand bar is an Accordion Container, which is easier to show than to describe. In Dijit, the dijit.layout.AccordionContainer holds dijit.layout.AccordionPane's, a special kind of ContentPane.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.AccordionContainer" sizeMin="20" sizeShare="20">
        <div dojoType="dijit.layout.AccordionPane" title="Folders">
            Folders will go here
        </div>
        <div dojoType="dijit.layout.AccordionPane" title="Address Book">
            Address Book will go here
        </div>
</div>
```

The title attributes give the label for the top of the AccordionPane. When the user clicks on this title or the arrow icon next to it, the pane will slide into view.

The Message Preview Pane - SplitContainer

Now we can glue these two pieces together with another SplitContainer, this one oriented horizontally:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!-- main section with tree, table, and preview -->
<div dojoType="dijit.layout.SplitContainer"
        orientation="horizontal" sizerWidth="5" activeSizing="0" title="Inbox">
    <div dojoType="dijit.layout.AccordionContainer" sizeMin="20" sizeShare="20">
     ...
    </div>
    <div dojoType="dijit.layout.SplitContainer" id="rightPane"
         orientation="vertical"  sizerWidth="5"  activeSizing="0">
    ...
    </div>
</div>
```

Tabbed Panes - TabContainer

The whole thing will be situated in a tab named Inbox. Though our cocktail napkin drawing doesn't show other tabs, the idea is to open them for new email composition. This way you can work on many emails at once, clicking on the tabs to switch between them. Dijit has a widget dijit.layout.TabContainer, and like the Split and Accordion containers, it's a container for subscontainers and panes.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.TabContainer"  id="tabs" jsId="tabs" layoutAlign="client">
    <div dojoType="dijit.layout.SplitContainer"
            orientation="horizontal" sizerWidth="5" activeSizing="0" title="Inbox">
    ...
    </div>
</div>
```

The dijit.Layout.TabContainer can hold as many tabs as you want, each marked with a title attribute used to title the tab. This illustrates a familiar pattern in Dijit. Sometimes a widget contains attributes that don't really deal with the widget itself - title, for instance, or sizeMin and sizeShare in earlier examples. They only make sense when that widget is used within other widgets. So an AccordionContainer ignores a sizeMin attribute, but because it's a child of a SplitContainer, the SplitContainer uses it for sizing. Title is similar. In a normal HTML tag, title would display a tooltip, but because the SpilitContainer is inside a TabContainer, the title is used specially.

What is that jsId attribute? jsId sets a global JavaScript variable for the widget, which will make it easy to script widget actions later. For example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// Note used for step 1 - this is a preview
tabs.closeChild(tabs.selectedChildWidget);
```

closes the currently selected tab in the tab container. Without even knowing the Dijit API's, it's easy to understand what's going on here. That's object-oriented magic at work.

Gluing On Toolbar and Status Areas - LayoutContainer

To top it all off, we smush these tabs into a dijit.layout.LayoutContainer. A LayoutContainer was pioneered by Borland Delphi and copied in Java AWT and other toolkits. The idea is to split a box into five areas: top, bottom, left, right, and client (the middle). If the box is resized, the client area takes most of the growth or shrinkage - it's the document part of the app.

In our email client, the toolbar will occupy the top, the status bar the bottom, and the TabContainer the client portion:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.LayoutContainer" id="main">
        <div dojoType="dijit.Toolbar" layoutAlign="top" style="height:25px;">
           Toolbar will go here
        </div>

        <div dojoType="dijit.layout.ContentPane" layoutAlign="bottom"
             id="footer" align="left">
                <span style="float:right;">DojoMail v1.0 (demo only)</span>
                Progress bar will go here
        </div>
        <div dojoType="dijit.layout.TabContainer"
            id="tabs" jsId="tabs" layoutAlign="client">
         ...
        </div>
</div>
```

The layoutAlign attributes tell LayoutContainer where they should be placed.

Now Andy just adds the standard Dojo headers and the skeleton is complete! /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Demo Mail Application</title>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
            dojo.require("dijit.Toolbar");
            dojo.require("dijit.layout.LayoutContainer");
            dojo.require("dijit.layout.SplitContainer");
            dojo.require("dijit.layout.AccordionContainer");
            dojo.require("dijit.layout.TabContainer");
            dojo.require("dijit.layout.ContentPane");
    </script>
        <style type="text/css">
            @import "http://o.aolcdn.com/dojo/1.0.0/resources/dojo.css";
            @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/soria/soria.css";
            @import "http://o.aolcdn.com/dojo/1.0.0/dijit/demos/mail/mail.css";
        </style>
</head>
<body class="soria">
```

There's no JavaScript at all beyond the dojo.require statements. And because he's using CDN, he didn't even have to install Dojo on the server. While Andy takes a moment to write more of his empassioned speech, you can download the code in step1.html below and try it out.

Ready? On to the next step...

The Address Book

Andy thinks, "That was easy. Now what's the next easiest thing to do?" The left-hand side panes, Address Book and Folders, will each have a tree. The Folders tree will have a heirarchy - folders within folders - but the Address Book is nice and flat. Andy figures he'll use the Address Book to learn Dijit Trees, then use his knowledge on the more complex Folders pane later. Here's the cocktail napkin view:

[inline:mail_step2.png]

Back in the old days (i.e. 5 minutes ago), Andy would have drawn up a PHP file that sent all the markup for the Address Book tree - the folder IMG tags, the labels, etc. - over the wire. While this approach worked fine in the old days, he has learned it doesn't work well for Web 2.0 applications like this one. Why?

- It doesn't scale well. For a few entries, assembling the markup on the server, downloading, parsing and inserting the HTML is no problem. But this Address Book could contain hundreds or thousands of entries. And it will need to be drawn frequently.
- It requires PHP to generate JavaScript. Since each Book entry needs an onlcik handler, Andy would need to embed onclick handlers into each tag. Mixing languages like this requires him to juggle syntax in his head. His mind is on other things (remember the girlfriend? the proposal?)

Address Book Data - dojo.Data and JSON

He decides to leave the architecture questions alone for a moment, and looks at the Dijit catalog entry for Tree. The easiest data-driven way to approach this uses dojo.data. Dojo.data reminds Andy of ODBC in the Windows world, JDBC in Java, and DBD in PHP (Andy gets around.) Though dojo.data can talk to many different kinds of backing stores formatted in CSV, XML or HTML, Andy picks JavaScript Object Notation, or JSON. A JSON representation of the address book would look like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;}

```
.geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
{
        identifier: 'id',
        label: 'label',
        items: [
                { type: 'address', id: 'adam', label: "Adam Arlen" },
                { type: 'address', id: 'bob', label: "Bob Baxter" },
                { type: 'address', id: 'carrie', label: "Carrie Crow" }
        ]
}
```

JSON nicely represents JavaScript objects, bracketed by { and }, and arrays bracketed by [and]. The attributes identifier and label conform to dojo.data's metadata specifications. Dojo.data uses the drivers dojo.data.ItemFileReadStore and dojo.data.ItemFileWriteStore to read JSON data in this format.

Andy thinks, "it'll be a snap to write a PHP for that format!" For the demo, he will simply use the above file as mail/mail.json. If the project moves past the demo phase, he can replace it with a PHP that reads mail on the database server and outputs JSON. Good deal.

To connect his demo to the data store, he inserts this below the BODY tag:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<div dojoType="dojo.data.ItemFileWriteStore" jsId="mailStore"
        url="mail/mail.json"></div>
```

He has decided to use the ItemFileWriteStore to write entries back later (maybe not in the demo, but later on.)

Displaying the Address Book - Tree

Now drawing the tree is a snap using dijit.Tree:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<div dojoType="dijit.Tree" id="addrTree" store="mailStore"
        labelAttr="label" query="{type:'address'}">
        <script type="dojo/method" event="getIconClass" args="item">
                var specifiedIcon = item && mailStore.getValue(item, "icon");
                return specifiedIcon || "mailIconFolderDocuments";
        </script>
</div>
```

The store attribute connects to the jsId attribute on the ItemFileWriteStore. LabelAttr tells which attribute is used for the labels. Finally, the query tells dojo.data which entries to retrieve. You can equate a dojo.data query with a SQL query - it works with any backing store that dojo.data understands. As you can see here, a dojo.data query is much simpler than a SQL query. Sorting and compound selection are available for later, but for now this simple query will do fine.

Tree Icons - Extension Points

But wow ... what's up with that SCRIPT tag? What is type "dojo/method"?

That came about when Andy learned about tree icons ... the hard way. His first stab at dijit.Tree drew a nice address book but no icons. That wouldn't do. Clickdammit.com really, really likes icons. The trouble is there might be different icons based on the address book entry type.

At first Andy thought he'd have to modify dijit.Tree ... good thing Dojo is open source, right? But he'd been down that road before. You start monkeying around with changing the source and it becomes a never-ending struggle with revision control. So he read over the Dijit catalog entry for Tree and found:

[inline:mail_step2a.png]

An *extension point* is a point at which you can add, remove or replace functionality. The Dijit designers have built lots of extension points so you can change Dijit instances to your heart's content. And that's what Andy did.

First he looked up the signature for the extension point in the API guide: dijit.Tree.getIconClass:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
getIconClass: function(/*dojo.data.Item*/ item){
                // summary: user overridable function to return CSS class name to display icon
        },
```

Since each of the address book entries was a dojo.data.Item, he had no problem passing something in. Now he can replace the guts of the SCRIPT tag with JavaScript code to read the icon attribute and turn it into a CSS class name. That class name points to the right icon specified in mail/mail.css. Problem solved!

One note about that "item &&" phrase. In Dojo 1.0, Trees always have one and only one root node. There is nothing in dojo.data that specifies items as root nodes, and thus the first item passed to getIconClass is always null. By saying "item &&" we take advantage of JavaScript's short-circuit && (and) operator. It says "if item is a false-y value (= false, 0, null), stop here and don't evaluate what's after the

&&."

"Cool!" thought Andy. Extension points made it easy to change widget behavior, and dojo/method calls did the nasty job of wiring JavaScript code.

And with a few more header entries:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.require("dojo.data.ItemFileWriteStore");
        dojo.require("dijit.Tree");
```

step2.html is now a working address book. If you don't believe it, download it and try it yourself! Then we'll move right along...

User Interaction

Pretty nice so far, but this app is long on displaying, short on interaction. So next Andy decides to add some toolbars, tooltips and dialog boxes. "Surely this will involve lots of JavaScript," he thinks.

Here's the cocktail napkin view of the toolbar:

[inline:mail_step3a.png]

We already have the dijit.Toolbar widget in the app as a placeholder. Dijit.toolbar expects its immediate children to be buttons:

- dijit.form.Button is a plain ol' command button, but better.
- dijit.form.DropDownButton, when clicked, displays a menu of commands
- dijit.form.ComboButton combines the two - acting like a button when clicked, or a menu when held-down.

Getting Mail - ComboButton and Tooltip

The New Mail button is a ComboButton:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<button id="getMail" dojoType="dijit.form.ComboButton"
        iconClass="mailIconGetMail">
        <script type="dojo/method" event="onClick">
                fakeDownload();
        </script>
        <span>Get Mail</span>
        <ul dojoType="dijit.Menu">
                <li dojoType="dijit.MenuItem" iconClass="mailIconGetMail">Yahoo</li>
                <li dojoType="dijit.MenuItem" iconClass="mailIconGetMail">GMail</li>
        </ul>
</button>
<span dojoType="dijit.Tooltip" connectId="getMail">Click to download new mail.</span>
```

The dijit.Tooltip does what you think it should - it displays a message when the user hovers over the button. It knows this by connecting the connectId to the id attribute of the widget.

At the heart of a ComboButton or DropDownButton is a dijit.Menu, itself composed of dijit.MenuItem objects. Andy chooses not to wire the items - Yahoo and GMail - to an event handler yet. But he does tie the onClick handler of the ComboButton itself.

Like for the Tree, Andy uses a dojo/method script to fire off a snippet of JavaScript. In this case, it's a stubbed out function called fakeDownload(). But wait! Can't you just do this? /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!-- It works, but uses DOM Level 0 event model.  Yuck. -->
<button id="getMail" dojoType="dijit.form.ComboButton"
        iconClass="mailIconGetMail"
        onclick="fakeDownload();">
```

Yes, but you wouldn't want to. Using dojo/method and connecting to the extension point gets you the DOM Level 2 event model for free. DOM Level 2 events pass much more information to the event handler. And before you ask ... yes, this even works in Internet Explorer, which doesn't natively support the DOM Level 2 event model. Dojo adds a nice layer which emulates the model on IE.

Sweet! That's cross-browser functionality for free. Dojo is really, really good at that.

New Mail and Options - Dialog Boxes

So let's look at the other buttons:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;}

```
<button
        id="newMsg" dojoType="dijit.form.Button"
        iconClass="mailIconNewMessage">
        New Message
        <script type="dojo/method" event="onClick">
                alert("New message functionality coming soon.");
        </script>
</button>
<span dojoType="dijit.Tooltip" connectId="newMsg">Click to compose new message.</span>
<button id="options" dojoType="dijit.form.Button" iconClass="mailIconOptions">
        Options
        <script type="dojo/method" event="onClick">
                dijit.byId('optionsDialog').show();
        </script>
</button>
<div dojoType="dijit.Tooltip" connectId="options">Set various options</div>
```

There's nothing here we haven't seen before ... except the call to optionsDialog. This displays a dialog box, which should look like this:

[inline:mail_step3b.png]

Now this looks like a regular HTML form, and in fact we'll code it as if it were. But ... we'll use a dijit.Dialog to do it. The good news is that unlike HTML forms, a dijit.Dialog acts modally, leaving all the information and page state alone while the user fills in the dialog details. So the server interaction happens in the background.

The code for the dialog box goes at the bottom of our app, just above the /BODY tag, separated away from all the other HTML:

```
<div dojoType="dijit.Dialog" id="optionsDialog" title="Options:">
        <table>
        <tr>
            <td style="text-align:right;"><label for="option1">Transport type:</label></td>
            <td>
                <select id="option1" dojoType="dijit.form.FilteringSelect">
                    <option value="pop3">POP3</option>
                    <option value="imap">IMAP</option>
                </select>
            </td>
        </tr>
        <tr>
            <td style="text-align:right;"><label for="option2">Server:</label></td>
            <td>
                <input id="option2" dojoType="dijit.form.TextBox" type="text">
            </td>
        </tr>
        <tr>
            <td style="text-align:right;">
                <input type="checkbox" id="fooCB" dojoType="dijit.form.CheckBox">
            </td>
            <td><label for="fooCB">Leave messages on Server</label></td>
        </tr>
        <tr>
             <td style="text-align:right;">
                 <input type="checkbox" id="fooCB2" dojoType="dijit.form.CheckBox">
             </td>
             <td><label for="fooCB2">Remember Password</label></td>
        </tr>
        <tr>
            <td colspan="2" style="text-align:center;">
                <button dojoType="dijit.form.Button" type="submit"
                            iconClass="mailIconOk">OK</button>
                <button dojoType="dijit.form.Button" type="submit"
                            iconClass="mailIconCancel">Abort</button>
             </td>
        </tr>
        </table>
</div>
```

You can look at dijit.Dialog as a mini-form, with Dijit form elements sprinkled into it. Note, we don't do anything with the data here, but a finished app would save the settings and communicate them back to the server.

Andy places the dojo.require's in the header

```
dojo.require("dijit.form.Button");
dojo.require("dijit.Menu");
dojo.require("dijit.Tooltip");
dojo.require("dijit.Dialog");
dojo.require("dijit.form.ComboBox");
dojo.require("dijit.form.CheckBox");
dojo.require("dijit.form.FilteringSelect");
dojo.require("dijit.form.Textarea");
```

And step 3 is done. "Wow," Andy thought, "that's a lot less JavaScript than I thought." To tell the truth, there *is* a lot of JavaScript behind it, but you didn't have to write it. Dojo has it all!

A Compose Mail Widget

Next, Andy needs a Compose Mail window that looks like this:

[inline:mail_step4.png]

Hmmmm, composing HTML email looks like it might be a problem. There's that huge toolbar and all those display stuff. Well, to be honest, with Dojo that's the easiest part of this step. Simply write a template like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<textarea dojoType="dijit.Editor" style="overflow:auto"
        extraPlugins="[{name:'dijit._editor.plugins.LinkDialog'}]"
>
<i> This is just a sample message. There is email-address auto-complete in the to: field.
<br><br> give it a whirl.
</i>
</textarea>
```

And save it in the file mail/newMail.html on your server. The dijit.Editor widget sets up the word-processor like environment for you, complete with toolbar, accelerator keys, and a fairly full array of HTML tools. The LinkDialog plugin adds an "Add Link..." button to the editor. Wow! It's a 5 line word processor. Andy mentally files this away - it could come in handy for composing his heartfelt speech later!

Using Dijit components is fun in a geeky sorta way. Still, Andy was one of those kids who got bored with Lego's ... until he made new ones with a band saw and a soldering iron.

If there were only one Compose Mail screen at a time, he could model it as a single tab and show it when necessary. But we need more than one. Andy thinks ... if one could just make a Compose Mail widget, then you can create as many instances as you want.

You can do that! And just like you can create widget instances declaratively or programmatically, so you can create new widget classes declratively or programmatically. As you might guess, the declarative way is easier. You simply pull out the dijit.Declaration widget.

Dijit.Declaration defines a widget class using a familiar macro-like template. Let's take a look:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.Declaration" widgetClass="mail.NewMessage">
    <div dojoType="dijit.layout.LayoutContainer" dojoAttachPoint="container"
         title="Composing..." closeable="true">
        <div dojoType="dijit.layout.LayoutContainer" layoutAlign="top"
             style="overflow: visible; z-index: 10; color:#666; ">
            <table width=100%>
                <tr style="padding-top:5px;">
                    <td style="padding-left:20px; padding-right: 8px; text-align:right;">
                        <label for="${id}_to">To:</label>
                    </td>
                    <td width=100%>
                        <select dojoType="dijit.form.ComboBox"
                                id="${id}_to" hasDownArrow="false">
                            <option></option>
                            <option>adam@yahoo.com</option>
                            <option>barry@yahoo.com</option>
                            <option>bob@yahoo.com</option>
                            <option>cal@yahoo.com</option>
                            <option>chris@yahoo.com</option>
                            <option>courtney@yahoo.com</option>
                        </select>
                    </td>
                </tr>
                <tr>
                    <td style="padding-left: 20px; padding-right:8px; text-align:right;">
                        <label for="${id}_subject">Subject:</label>
                    </td>
                    <td width=100%>
                        <select dojoType="dijit.form.ComboBox"
                                id="${id}_subject" hasDownArrow="false">
                            <option></option>
                            <option>progress meeting</option>
                            <option>reports</option>
                            <option>lunch</option>
                            <option>vacation</option>
                            <option>status meeting</option>
                        </select>
                    </td>
                </tr>
            </table>
            <hr noshade size="1">
        </div>

        <div dojoType="dijit.layout.LayoutContainer"
             layoutAlign="bottom" align=center>
            <button dojoType="dijit.form.Button"
                    iconClass="mailIconOk">Send</button>
            <button dojoType="dijit.form.Button"
                    iconClass="mailIconCancel">Cancel</button>
    </div>
        <!-- new messase part -->
        <div dojoType="dijit.layout.LayoutContainer" layoutAlign="client">
            <div dojoType="dijit.layout.ContentPane" href="mail/newMail.html"></div>
        </div>
```

```
    </div>
</div>
```

The HTML defined inside the dijit.Declaration tagis dropped in wherever the widget class "mail.newMessage" is requested. You could, in fact, drop one in like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="mail.newMessage" id="myNewMessage"></div>
```

You pass parameters to your new widget by using:

- **An attribute on the calling side**: in our case, we set id="myNewMessage" to pass the parameter "id".
- **A ${___} on the declaration side**: in our case, we have ${id} marked several places in the definition. These are substituted with myNewMessage in this call.

This declarative usage is neat, but not helpful in our case since we don't know how many newMessage components to create. Instead, we create instances programmatically using the Compose Mail button:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<button
        id="newMsg" dojoType="dijit.form.Button"
        iconClass="mailIconNewMessage">
        New Message
        <script type="dojo/method" event="onClick">
                /* make a new tab for composing the message */
                var newTab = new mail.NewMessage({id: "new"+paneId  }).container;
                dojo.mixin(newTab,
                        {
                                title: "New Message #" + paneId++,
                                closable: true,
                                onClose: testClose
                        }
                );
                tabs.addChild(newTab);
                tabs.selectChild(newTab);
        </script>
</button>
```

This new code creates a mail.NewMessage programmatically, then creates a Tab out of it and adds it to the TabContainer (specified by the JavaScript variable newTabs)

Add some declarations and a few supplemental variables to the header:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.require("dijit.Declaration");
dojo.require("dijit.Editor");
dojo.require("dijit._editor.plugins.LinkDialog");
// Global var for new mail pane
var paneId=1;

// for "new message" tab closing
function testClose(pane,tab){
  return confirm("Are you sure you want to leave your changes?");
}
```

And now we can compose mail to people.

Mail Progress Bar

The people who run clickdammit.com are cheapskates. Their servers are Pentium Pros bought off Ebay in exchange for some Beanie babies. So Andy knows that users will click "Get Mail" and wonder what's taking so long.

This is a common problem in Web 2.0 applications. Hopefully your servers are faster than Pentium Pros, but even then you can't always be sure when response times will be slow. Since you're not submitting the form, you can't rely on the browser's spinning logo to tell the user something's happening.

Fortunately, Dijit has a progress bar for those situations. Andy doesn't have access to clickdammit's mail server yet, but he decides to sketch out the progress bar and write some Wait loops behind them to simulate delays.

First he places the progress bar where the placeholder text was:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div id="fetchMail" style="opacity:0;visibility:hidden">
        <div annotate="true" id="fakeFetch" dojoType="dijit.ProgressBar"
```

```
                style="height:15px; width:275px;" indeterminate="true"
                report="fakeReport"></div>
</div>
```

We make this progress bar hidden so it can be called up when needed. At first the progress will be incalculable since we don't know the number of messages to download. That's what "indeterminate=true" shows.

We have already stubbed out fakeDownload() and connected it to the Get Mail button. Now we can actually add some code to it.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var fakeDownload = function(){
        dojo.byId('fetchMail').style.visibility='visible';
        numMails = Math.floor(Math.random()*10)+1;
        dijit.byId('fakeFetch').update({ maximum: numMails, progress:0 });
        dojo.fadeIn({ node: 'fetchMail', duration:300 }).play();
        for (var i=0; i<=numMails; i++){
            setTimeout(
                      "updateFetchStatus("+i+")",
                      ((i+1)*(Math.floor(Math.random()*100)+400))
            );
        }
}
```

First we turn on the progress bar, then pick a random number (1 to 10) of emails to download. Now to update the progress bar ... Andy looks up the dijit.ProgressBar widget in the Dijit catalog and finds under Methods:

Methods

update update progress information

Update looks good, and he consults the example before writing the update statement.

dojo.fadeIn does what you think. It fades-in an object but turning up the opacity from 0% to 100%. The "300" here is the number of milliseconds fadeIn should take to complete its job. 0.3 seconds sounds small, but it's enough to make the animation noticeable without stopping the proceedings.

Any adds the updateFetchStatus function, which will fire once for every email.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var numMails;
var updateFetchStatus = function(x){
        if (x == 0) {
                dijit.byId('fakeFetch').update({ indeterminate: false });
                return;
        }
        dijit.byId('fakeFetch').update({ progress: x });
        if (x == numMails){
                dojo.fadeOut({ node: 'fetchMail', duration:800,
                        // set progress back to indeterminate. we're cheating, because this
                        // doesn't actually have any data to "progress"
                        onEnd: function(){
                                dijit.byId('fakeFetch').update({ indeterminate: true });
                                // remove progress bar from tab order
                                dojo.byId('fetchMail').style.visibility='hidden';
                        }
                }).play();
        }
}
```

dojo.fadeOut looks a lot like fadeIn, but the onEnd property is new. It expects a function as its property. Here we add a function literal that removes the progress bar from the screen. But why put this in onEnd? Why not just put it after play()? Andy tried that first, and found the progress bar would blink out before it had completely faded. That's because fadeOut, like all Dojo animations, runs asynchronously. In other words, it kicks off the fadeOut and starts executing the next statement. FadeOut continues executing. So setting the duration to 800000000 would make the animation slow, but the browser would still be immediately usable.

Finally, we connect a function to ProgressBar's "report" extension point. Remember how we specified report="fakeReport" in the Progress Bar tag? Now we can fill it in. Report gets passed in a percent and expects a string to place on the progress bar.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var fakeReport = function(percent){
    return "Fetching: "+(percent*this.maximum) + " of " + this.maximum + " messages.";
}
```

Andy clicks on the Get Mail and sits back. It looks darn good! He clicks it a few more times just to make sure. Very nice. We're getting pretty close to a working demo to show off. Just a few more things need to go in place.

Mail Folders and dojo.Data

JSON and dojo.data stores worked pretty well for address book data. Andy wonders whether they would work to send mail and folders too.

[inline:mail_step6.png]

Not only is it possible ... you can pass all the data back in one fell swoop. This is possible because ItemFileReadStore and JSON handle heirarchical data with ease. In mail land, you have folders which contain other folders or mail items. In JSON notation, you can model folders like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{
identifier: 'id',
label: 'label',
items: [
        // Hierarchy of folders
        { type: 'folder', id: 'mailbox', label:'Folders', folders: [
                { type: 'folder', id: 'inbox', label:'Inbox', icon:'mailIconFolderInbox' },
                { type: 'folder', id: 'deleted', label:'Trash', icon:'mailIconTrashcanFull' },
                { type: 'folder', id: 'save', label:'Save',  folders:[
                        { id: 'work', label:'stuff for work'},
                        { id: 'fun', label:'stuff for fun'}
                ]}
        ]},
```

You can see how we nest objects within objects, like the "work" and "fun" folders underneath the "Save" folder. We can model mail like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// Flat list of messages (each message lists its folder)
        { type: 'message', id: 'node1.1',
                  folder: 'inbox', label: "today's meeting",
                  sender: "Adam Arlen", sent: "2005-12-19",
                  text: "Today's meeting is cancelled.<br>Let's do it tomorrow instead.<br><br>Adam"
        },
        { type: 'message', id: 'node1.2',
                  folder: 'inbox', label: "remaining work",
                  sender: "Bob Baxter", sent: "2005-12-18",
                  text: "Hey, we need to talk about who's gonna do all the left over work.  Pick a day you want to meet: <div
dojoType='dijit._Calendar'></div>"
        },
```

Now you have three types of objects, delineated by type: address, folder and message. Andy decides to stick them all in one JSON data message. That means only one trip ito the server grabs everything about the user's mail store. Gotta love that!

The actual folder listing looks a lot like the address book code. The getIconClass extension point draws the folder icons:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.AccordionPane" title="Folders">
        <div dojoType="dijit.Tree" id="mytree" store="mailStore"
                labelAttr="label" childrenAttr="folders"
                query="{type:'folder'}" label="Folders">
                <script type="dojo/method" event="getIconClass" args="item">
                        return (item && mailStore.getValue(item, "icon"))
                                || "mailIconFolderDocuments";
                </script>
        </div>
</div>
```

The "childrenAttr" in the Tree tag does all the sub-folder magic.

One more thing and he'd be ready to show it to Ms. Opulence: connect the folders to messages and the messages to a preview pane. Andy looks at his watch. He's got about 15 minutes before the el stops by his office. If he could catch it and get home early, that'd give him some serious writing and reflecting time. So let's make short work of this, shall we?

The Message List - dojox.Grid

Andy has been studying up on Dojo 1.0, and found one of the great new features - the Dijit grid. A grid is like a mini spreadsheet, or a maxi-HTML-table. You can sort the columns, edit cells, and enable all kinds of cool stuff. It sounds like the perfect container for a message list.

[inline:mail_step7.png]

Feeding the Grid

Once you've worked with Dojo awhile, you start to see Grand Unifying Themes. One of them is dojo.data, feeding server-pulled data to different kinds of widgets. Grid is no different. Fortunately, Andy already has the data store defined. And since that data store has messages in it, hierarchically tied to folders, it's a short step to display them. First he adds the separate Grid style sheet:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<style type="text/css">
    @import "http://o.aolcdn.com/dojo/1.0.0/dojox/grid/_grid/tundraGrid.css";
    @import "http://o.aolcdn.com/dojo/1.0.0/resources/dojo.css";
    ...
```

A grid has two main elements: the model and the structure. The model is the data behind the grid, while the structure is a set of display instructions.

Andy does the model first. We already havy the data in a dojo.data store called mailModel. To make the data source mailStore work with Grid, he declares a dojox.grid.data.DojoData adapter. You can think of it as a pipe where the data flows in, and the appropriately filtered data comes out for use in a grid. When the app initially starts, there is no folder selected. Hence there are no messages to display. Andy decides to initialize the adapter, but issues a query that will return no items. This has performance benefits, as we'll see in a minute.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dojox.grid.data.DojoData" jsId="mailModel"
     rowsPerPage="20" store="mailStore" query="{ type: 'NONE' }"
     clientSort="true">
</div>
```

The structure is a description of the Grid layout. Structures are composed from views, which are little independently scrollable subgrids called views. Our grid is simple, containing only one view. A cell is Grid's name for a column. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var mailView = {
    cells: [[
        {name: 'From', field:'sender',width:"25%"},
        {name: 'Subject', field:'label',width:"65%"},
        {name: 'Date', field:'sent',width:"10%"}
    ]]
};
var layout = [ mailView ];
```

The cells property is a two dimensional JavaScript array, hence the double square brackets. In the Grid section, you'll see how to split large amounts of data into subrows, all selectable as one unit. Here we have just one subrow with a small amount of data.

Wiring up the actual Grid is a snap. The Grid widget does all the work for you:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div id="listPane" dojoType="dijit.layout.ContentPane" sizeMin="20" sizeShare="20">
    <div id="grid" dojoType="dojox.Grid" jsId="mailGrid" structure="layout" model="mailModel"
         onRowClick="displayMailMessage">
    </div>
</div>
```

Clicking a row in this grid will display the appropriate message in the bottom area. So Andy stubs out the call to the onRowClick event, which he'll fill in later.

In the folder tree, Andy connects some Dojo code to the onClick event. Once you know the folder name, you simply build a new adapter, set that as the Grid model, and the grid will automatically populate. The nice thing here is the Mail messages are already loaded in the dojo.data store. Building a new Grid model on top of it re-uses the data with a new query without a server round-trip.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.Tree" id="mytree" store="mailStore"
    labelAttr="label" childrenAttr="folders" query="{type:'folder'}" label="Folders">
    <script type="dojo/method" event="onClick" args="item">
        if(!item){
            return; // top level node in tree doesn't correspond to any item
        }
        /* filter the message list to messages in this folder */
        var newMailModel = new dojox.grid.data.DojoData();
        newMailModel.rowsPerPage = 20;
        newMailModel.store = mailStore;
        newMailModel.query = {
            type: "message",
            folder: mailStore.getValue(item, "id")
        };
        newMailModel.clientSort = true;
        mailGrid.setModel(newMailModel);
    </script>
```

So now clicking a folder displays the message list. It's time to go back and fill in onRowClick for the message. This turns out to be fairly trivial:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function displayMailMessage(evt) {
     var msg = mailGrid.model.getDatum(evt.rowIndex,6);
     dijit.byId("message").setContent(msg);
}
```

So now he has a message list and a message. Andy looks at his watch. 5 minutes to spare. He can't resist adding one more item - displaying a widget inside a mail message. So he quickly adds a test case to the sample data:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ type: 'message', id: 'node4.4', folder: 'fun',
   label: "paint", sender: "Jack Jackson", sent: "2005-12-16",
   text: "what color is good for the new office?<div dojoType='dijit.ColorPalette'></div>Let me know soon"
},
```

And dojo.require's dijit.ColorPalette. Sure enough, displaying the message automagically displays the palette:

[inline:mail_step8.png]

Summary

Wow! That's a lot of functionality. In a short amount of code, about 1/2 JavaScript and 1/2 HTML, we have a working email client user interface with:

- Command buttons with icons
- An Options dialog box
- A tabbed interface where you can compose multiple emails at once and quickly switch between them.
- A full HTML-based email format. You can create, edit and read email messages with full, rich formatting.
- An address book and folder list, heirachically arranged with expansion and contraction buttons.
- A message list tied to the folders with a click
- Lots of little user cues - tooltips and a progress meter for downloads

Sure, the server part needs writing. But that can wait until tomorrow, and it shouldn't be too difficult. Find the right words to get a woman to marry you ... *now that's difficult*. Andy heads out the door for the train

Example 3: Chatting With Tech Support

Laura Allen has been an AIM user for years, and has been reading about the business uses of chat software. One use that intrigues her is "Live Support". Her company sells very sophisticated equipment which generates many sales questions. It be very nice to have a support person on the opposite end of a chat client. Maybe if they would all just use AIM ... but this is unrealistic.

She considers writing it in PHP. The trouble is the stateless nature of the web. It's easy to send a message to the server. But that generates a page refresh that loses most of the state. Plus the refresh makes it look primitive and ugly compared to AIM.

Enter Dojo. With it, she can write a mini chat client that polls the server via XHR and no page refreshes. The ASP server portion is easy - just a simple storage mechanism and a way to transmit the data via JSON. The event model of dojo is familiar like Visual Basic or Java Swing, and passes the chat messages easily. Not to mention (and this is wickedly cool), the rich text editor of dojo makes it easy to do italics, bold, sizes, etc. Finally, wrapping this all up in a collapsible title pane tucks it out of the way so a customer can pull it in when needed.

On the operator end, dojo helps as well. Operators can have several conversations at once, with each conversation in a separate tab. The tab will light up when there's activity, so the operator can keep chats on hold without losing them.

* The source code for this demo, written by Peter Higgins, is included in the dijit/demos folder of the Dojo distribution. For now, to make this demo work you will need your own Cometd server. We hope to have a public sandbox server available shortly for your demoing pleasure.

Architecture

Laura, being the agile programmer she is, wants to first build the smallest solution that solves the problem. She sighs, longing for the days of C and network sockets. But JavaScript's security model prohibits peer-to-peer networking, so there's nothing like that.

She looks on the Dojo site and finds the cometd (pronounced comet-d) server. Interesting! It's a lightweight, HTTP-based server with a small footprint. And it is supposed to have good Dojo integration through publish-subscribe events.

Publish-Subscribe

Publish-subscribe is an easy-to-understand model for cross-process communication, and Dojo has it baked in. It effectively promotes loose coupling between components. Suppose you have a tax form similar to Example 1. You have ten dijit.form.CurrencyTextBox's whose contents get added up to a Gross Income, another dijit.form.CurrencyTextBox. How do you keep Gross Income updated? One way is for the component CurrencyTextBox's to call a central procedure for updating:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript">
function reAdd() {
   newSubtotal = dijit.byId("wages").getValue() + ...;
   dojo.byId("grossIncome") = newSubtotal;
}
</script>
<div dojoType="dijit.form.CurrencyTextBox" id="wages">
   <script type="dojo/connect" event="onChange">
      reAdd();
   </script>
</div>
<div dojoType="dijit.form.CurrencyTextBox" id="tips">
   <script type="dojo/connect" event="onChange">
      reAdd();
   </script>
</div>
...
```

Because this system is tightly coupled, problems arise:

- If you add an 11th line to total up, you must change the reAdd() procedure.
- If half of the boxes need to send extra onChange events to computeDeductions(), those half need to be changed to call this method.

Granted, this example is contrived because the methods are so small. But the larger your system, the more nightmarish maintenance becomes. It would be nice to have a whiteboard where controls could sign up for events that interest them. This whiteboard could be maintained by a third party to keep the controls from worrying about communication details.

That's the essence of publish-subscribe. Dojo is the whiteboard maintainer. The GrossIncome box then says "I'm interested whenever a box changes." This is called a *topic*, and we say the GrossIncome box *subscribes* to that topic. The component boxes agree to *publish* information on this topic when they change. Here's how it looks like in code:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript">
function reAdd(args) {
   dojo.byId("grossIncome").setValue(
        dojo.byId("grossIncome").getValue()
        + args.newValue
        - args.oldValue
   );
}
</script>
<div dojoType="dijit.form.CurrencyTextBox" id="GrossIncome">
   <script type="dojo/method">
      dojo.subscribe("componentChange", reAdd);
   </script>
</div>
<div dojoType="dijit.form.CurrencyTextBox" id="wages">
   <script type="dojo/connect" event="onChange" args="e">
      dojo.publish("componentChange", { oldValue: e.oldValue, newValue: e.targetNode.value });
   </script>
</div>
...
```

So the topic "componentChange" is published when any of the component boxes change. Sent along with the topic are message-specific data elements that help the subscriber - oldValue and newValue in this case. Need to find which elements in the event object they are.. The lone subscriber, the GrossIncome box, uses these to adjust the current sum. Now to add a component box, you just add it to the form with the same dojo.publish statement. Very nice.

Publish-Subscribe With Cometd

Cometd effectively extends Dojo publish-subscribe past the browser. Cometd speaks the Bayeaux protocol which itself runs over HTTP. Laura immediately sees the usefulness of this for online support. The client can publish a topic to Cometd containing the message. The operator subscribes to that topic and displays messages on the console. It replies on that same topic. Both operator and client send the

userid with the message, so they can ignore the message they themselves send.

The protocol works like this. Assume the client's username is "alex.russell" and operator's username is "operator".

- **Operator**:
- (Time Marches On...)
- **Client**: Subscribe to "/chat/demo/alex.russell".
- **Client**: Publish a message on "/chat/demo/alex.russell": { user: "alex.russell", join: true, chat : "alex.russell has joined the room."}
- **Client**: Publish a message on "/chat/demo": { user: "alex.russell", joined: "alex.russell"}
- **Operator**: (Sees message on /chat/demo) Subscribe to "/chat/demo/alex.russell"
- **Client**: Publish a message on "/chat/demo/alex.russell": { user: "alex.russell", chat : "I have a question about Dojo."}
- **Operator**: (Sees message on /chat/demo/alex.russell) Publish a message on "/chat/demo/alex.russell": { user: "operator", chat : "Shoot."}
- **Client**: (Sees message on /chat/demo/alex.russell) Publish a message on "/chat/demo/alex.russell": { user: "alex.russell", chat : "Oh wait, I invented Dojo. Never mind."}
- **Client**: Unsubscribe to "/chat/demo/alex.russell"
- **Client**: Publish a message on "/chat/demo/alex.russell": { user: "alex.russell", leave: true, chat : "alex.russell has left the room."}

Although the protocol looks a little "chatty", Laura has designed it with public chat rooms in mind. When those come to pass, users will be able to join arbitrary public chat rooms using this protocol.

This upfront design will payoff - coding will go quite rapidly.

Details for the Impatient

- Cometd Server
- Publish-subscribe events
- dojox.cometd - cometd client support in Dojo
- dijit.form.CurrencyTextBox

The Room Widget

In previous examples, we have built a few widgets with the dojo.Declaration tag. This is a fast way to add functionality, but the problem is sharing it with other pages. Although Laura knows the Operator and Client pages will be different, she senses common elements in the way they chat. So she will define a widget class through JavaScript instead. That way, both pages can use it.

The Widget Skeleton

A widget is a way to tie Dojo API calls into one displayable element. The way Laura designs it, the display is pretty much the same on both sides. The only difference is the container. Here's a cocktail napkin view of the client side:

[inline:chat1.png]

And the operator side:

To be added once we get the public cometd server up.

The message display area, the input box for messages and the send button are exactly the same. So we can package those up as a widget, which we will call dijit.demos.chat.room. Using a little object oriented analysis, Laura stubs out the object code:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.provide("dijit.demos.chat.room");
dojo.require("dojox.cometd");
dojo.require("dijit._Widget");
dojo.require("dijit._Templated");
dojo.declare("dijit.demos.chat.Room",
    // All widgets inherit from _Widget, and all templated widgets mix in _Templated
    [dijit._Widget,dijit._Templated],
    {

        // Username of client
    _username: null,

        // Default room id.  Will become the client's username for our tech support line
    roomId: "public",

        // For future expansion into public chat rooms
    isPrivate: false,

        // Constant
    prompt: "Name:",

    join: function(name){
            // Join a room
    },

    _join: function(/* Event */e){
            // Respond to someone joining a room (only operator does this)
    },

    leave: function(){
            // leave a room
    },
```

```
    chat: function(text){
            // Send a message
    },

    _chat: function(message){
            // Receive a message
    },

    startup: function(){
            // Required function for a widget.  Called on startup
    },
});
```

The Widget Template

As we did in Example 1 for i18n, we build a template for the widget. Essentially this template will replace the tag with dojoType="dijit.demos.chat.room"

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div id="${id}" class="chatroom">
    <div dojoAttachPoint="chatNode" class="chat"></div>
    <div dojoAttachPoint="input" class="input">
        <div dojoAttachPoint="joining">
            <span>${prompt}</span>
            <input class="username" dojoAttachPoint="username" type="text" dojoAttachEvent="onkeyup: _join">
            <input dojoAttachPoint="joinB" class="button" type="submit" name="join"
                   value="Contact" dojoAttachEvent="onclick: _join"/>
        </div>
        <div dojoAttachPoint="joined" class="hidden">
            <input type="text" class="phrase" dojoAttachPoint="phrase" dojoAttachEvent="onkeyup: _cleanInput" />
            <input type="submit" class="button" value="Send" dojoAttachPoint="sendB"
                   dojoAttachEvent="onclick: _sendPhrase"/>
        </div>
    </div>
</div>
```

The placeholders ${id} and ${prompt} look familiar. These are replaced with the properties id and prompt from the widget instance. A few of the properties look unfamiliar, but they all start with the prefix "dojo".

- **dojoAttachPoint** creates a property in the widget containing the DOM node. For example, the DIV tag with dojoattachpoint="joining" creates a joining property, which you can pick up with "this.joining" in your widget code. You can do with anything you can do with a DOM node, for instance set its styles or CSS class.
- **dojoAttachEvent** connects an event and DOM node with a function. So dojoAttachEvent="onkeyup: _cleanInput" means "when the onkeyup event happens here, call _cleanInput."

Only one of the "joining" node:

[inline:chat2.png]

and the "joined" node:

[inline:chat3.png]

is visible at any one time. That gives the illusion that the "joined" and "joining" nodes toggle back and forth, occupying the same screen real estate. This is a fairly common trick in widget templates.

Starting the Widget

Every widget has extension points that get called during widget creation. The widget class designer can hook into these by providing methods, as we do here with startup():

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
startup: function(){
    this.joining.className='';
    this.joined.className='hidden';
    //this.username.focus();
    this.username.setAttribute("autocomplete","OFF");
    if (this.registeredAs) { this.join(this.registeredAs); }
    this.inherited("startup",arguments);
},
```

this.inherited() acts like Java's super() operator, but more generally. Here it calls dijit._Widget.startup(). This is a good practice to get into for the widget extension points.

Joining and Leaving a Room

Dojo calls the join method upon clicking the Submit button. Just as we outlined on the previous page, it establishes the chat connection with the operator (only the client calls this particular method). "_join" does the same thing in response to keystrokes.

Note how the dojoAttachPoints come in handy here: they make flipping the nodes on and off straightforward.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
join: function(name){
    if(name == null || name.length==0){
        alert('Please enter a username!');
    }else{
        if(this.isPrivate){ this.roomId = name; }
        this._username=name;
        this.joining.className='hidden';
        this.joined.className='';
        this.phrase.focus();
        dojox.cometd.subscribe("/chat/demo/" + this.roomId, this, "_chat");
        dojox.cometd.publish("/chat/demo/" + this.roomId,
           { user: this._username, join: true,
             chat : this._username+" has joined the room."}
        );
        dojox.cometd.publish("/chat/demo", { user: this._username, joined: this.roomId });
    }
},
_join: function(/* Event */e){
    var key = (e.charCode == dojo.keys.SPACE ? dojo.keys.SPACE : e.keyCode);
    if (key == dojo.keys.ENTER || e.type=="click"){
        this.join(this.username.value);
    }
},
```

Leaving, for the most part, just reverses the join sequence:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
leave: function(){
    dojox.cometd.unsubscribe("/chat/demo/" + this.roomId, this, "_chat");
    dojox.cometd.publish("/chat/demo/" + this.roomId,
       { user: this._username, leave: true,
         chat : this._username+" has left the chat."}
    );
    // switch the input form back to login mode
    this.joining.className='';
    this.joined.className='hidden';
    this.username.focus();
    this._username=null;
},
```

Sending and Receiving a Message

Finally, the meaty part of the Room widget. Sending is a fairly simple matter over our protocol:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
chat: function(text){
    // summary: publish a text message to the room
    if(text != null && text.length>0){
        // lame attempt to prevent markup
        text=text.replace(/</g,'<');
        text=text.replace(/>/g,'>');
        dojox.cometd.publish("/chat/demo/" + this.roomId, { user: this._username, chat: text});
    }
},
```

Receive is handled by the topic subscriptions, which we connected in join(). The event mediator sends receive() a message, which becomes the parameter "message" here. Then message.data contains the object that the publisher sent over.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
_chat: function(message){
    // summary: process an incoming message
    if (!message.data){
        console.warn("bad message format "+message);
        return;
    }
    var from=message.data.user;
    var special=message.data.join || message.data.leave;
    var text=message.data.chat;
    if(text!=null){
        if(!special && from == this._last ){ from="...";
        }else{
            this._last=from;
            from+=":";
        }
        if(special){
            this.chatNode.innerHTML +=
                "<span class=\"alert\"><span class=\"from\">"+from+
                " </span><span class=\"text\">"+text+"</span></span>";
            this._last="";
```

```
        }else{
            this.chatNode.innerHTML +=
                "<span class=\"from\">"+from+" </span><span class=\"text\">"+text+
                "</span>";
            this.chatNode.scrollTop = this.chatNode.scrollHeight - this.chatNode.clientHeight;
        }
    }
},
```

Entering and leaving messages use the alert CSS class to distinguish them from regular chat messages.

There's our bouncing baby widget! Now let's make use of it...

Details for the Impatient

- Creating a Widget Class
- Publish-subscribe events

The Client Page

The client page uses some eye candy from Dojo. (Clients love eye candy!)

Animation

The client chat window can be open and closed at will. Laura decides that the window should fade in and fade out - an easy thing to do with Dojo.

An *animation* in Dojo terms is the graduated movement of a DOM node from one state to another. A fade-in, for example, is the movement of opacity (the opposite of transparency) from 0% to 100%. Animations cover a set span of time, so you can fade in over the course of milliseconds, seconds, or minutes. In Dojo Animation, the most general form of an animation is a function. Some animations are so popular, like fades and slides, that Dojo packages those functions for you.

The animation for fading in, triggered by the Show button, looks like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<button dojoType="dijit.form.Button">
Show / Hide Tech Support Chat
<script type="dojo/method" event="onClick">
   var anim = dojo.fadeIn({ node: helpNode.domNode, duration: 400 });
   dojo.connect(anim,"beforeBegin",function(){
        dojo.style(helpNode.domNode,"display","block");
        helpNode.toggle();
        _positionIt();
   });
   anim.play();
</script>
</button>
```

The fade-in lasts for 400 ms. We hook a snippet of code in front of the animation to set the panel to display:block mode first, turn it on and position it to the top of the screen. This is necessary because opacity only works on an element that's actually displayed ... if it's display:node, as our node starts out as, the fade in won't work at all.

Because the corresponding fade-out for hiding is very similar, we rewrite the extension point to handle both jobs. helpNode.open is true if the chat window is currently open.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<button dojoType="dijit.form.Button">
Show / Hide Tech Support Chat
<script type="dojo/method" event="onClick">
    var anim = dojo[(helpNode.open ? "fadeOut" : "fadeIn")]({ node: helpNode.domNode, duration: 400 });
    dojo.connect(anim,(helpNode.open ? "onEnd" : "beforeBegin"),function(){
       dojo.style(helpNode.domNode,"display",(helpNode.open ? "none" : "block"));
       helpNode.toggle();
       _positionIt();
    });
    anim.play();
</script>
</button>
```

Keeping the Window Visible

Of course we don't want the chat window to scroll up as the user scrolls up. So we hook some code into the onScroll event of the window. onScroll is one of those handy Dojo Events. It is actually a front for the DOM Level 2 event of the same name, which Firefox implements in a standard way, and IE in a non-standard way. Well, we don't have to worry about that. That gives Laura more time to be creative.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;}

.geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
// this puts our help box in the top/right corner on scroll and show
function _positionIt(evt){
    if (helpNode.domNode.style.display == "block"){
        dojo.style(helpNode.domNode,"top",(dijit.getViewport().t + 4) + "px");
    }
}
var helpNode;
dojo.addOnLoad(function(){
    dojo.parser.parse(dojo.body());
    helpNode = dijit.byId('helpPane');
    dojo.connect(window,"onscroll","_positionIt");
    // this is a placeholder for the cometd server, once we get a public one.
    dojox.cometd.init("http://comet.yours.com:9000/cometd");
});
```

_positionIt repositions the window to be 4 pixels from the top of the viewport, which Dijit provides through its handy dijit.viewport function. The four viewport coordinates are kept in properties t, b, l and w (top, bottom, length, width).

dojo.addOnLoad specifies a code snippet to be run after all the DOM nodes have loaded and widgets have been drawn. It's a good place for connecting events, as we've done here with onscroll. Finally, we connect to the cometd server for the chat.

Lastly, Laura needs the operator page...

The Operator Page

The operator, unlike the client, can carry on multiple conversations. So they can subscribe to many chat topics, all called /chat/demo/username for uniqueness. Each conversation will have a different Room widget in a different ContentPane

When a message arrives, it may be a new chat request or an existing conversation. We keep track of this in objects that correspond to conversations. First, the addOnLoad starts subscriptions running:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.addOnLoad(function(){
        dojo.parser.parse(dojo.body());
        dojox.cometd.init("http://comet.sitepen.com:9000/cometd");
        dojox.cometd.subscribe("/chat/demo",control,"_getAlert");
});
```

The control object is an overall controller for operator conversations. Here's a skeleton:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var control = {
    _chats: {},
    _getAlert: function(e){
    },
    _privateChat: function(e){
    }
};
```

Coversation records are kept in the _chats object, which will be a hash of username keys with current conversations. Through our subscribe() above, _getAlert will get any messages over the public topic /chat/demo. It will then either create a new conversation and tabbed pane or simply return:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
_getAlert: function(e){
    // Ignore all control messages where we already have a conversation registered,
    // or messages bound for someone else.
    if (!this._chats[(e.data.user)] && (operator != e.data.user)){
        dojox.cometd.subscribe("/chat/demo/"+e.data.joined,this,"_privateChat");
        // Create a tab called chatWithUsername and insert it into the tabbed container
        var tabNode = document.createElement('div');
        tabNode.id = "chatWith" + e.data.user;
        var chatNode = document.createElement('div');
        chatNode.id = e.data.user + "Widget";
        tabNode.appendChild(chatNode);
        var newTab = new dijit.layout.ContentPane({
            title: e.data.user,
            closable: true
        },tabNode);
        dijit.byId('tabView').addChild(newTab);
        // Create an associated Room object
        var chat = new dijit.demos.chat.Room({
            roomId: e.data.joined,
            registeredAs: operator
        },chatNode);
        chat.startup();
        // And record this conversation
        this._chats[(e.data.user)]=true;
    }
},
```

The subscription to /chat/demo/username passes received messages to the proper ContentPane:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
_privateChat: function(e){
    var thisChat = dijit.byId(e.data.user+"Widget") || false;
    if (thisChat) { thisChat._chat(e); }
}
```

Laura calls over one of her team members and shows him the chat function. On separate PC's they play with it for about 15 minutes or so, trading war stories and jokes. It feels pretty nice to have a working system built this quickly and with so little code. Laura no longer misses C.

Part 2: Dijit - The Dojo Widget Library

[inline:themeTester.png]

Dijit is a widget system layered on top of dojo. If you are new to the whole Dojo experience, Dijit is a good place to start. You can build amazing Web 2.0 GUI's using very little, or no, JavaScript.

You can use Dijit in one of two ways: *declaratively* by using special attributes inside of regular HTML tags, and *programmatically* through JavaScript. You have the same options either way. As we did in Part 1, we'll use declarative Dijit for all the examples in this part. Part 3 will show how to make the same calls programmatically.

Dijit comes bundled with its own theme, tundra, which brings a common design and color scheme to all the widgets. You can override the theme by container or by element to add nuance and flair.

Everything in Dijit is designed to be globally accessible -- to accommodate users with different languages and cultures as well as those with different abilities. Language translations, bi-directional text, and cultural representation of things like numbers and dates are all encapsulated within the widgets. Server interactions are done in a way that makes no assumptions about local conventions. All widgets are keyboard accessible and using the standard Dijit theme, usable in high-contrast mode as well as by screen readers. These features are baked in so that, as much as possible, all users are treated equally.

Dijit at a Glance

Form, Validation, Specialized Input

CheckBox	[inline:checkbox.png]
ComboBox	[inline:combo_box.png]
CurrencyTextBox	[inline:currency_textbox.png]
DateTextBox	[inline:date_textbox.png]
FilteringSelect	[inline:filtering_select.png]
InlineEditBox 0.9 - 1.0	[inline:inline_edit.png]
NumberSpinner	[inline:number_spinner.png]
NumberTextBox	[inline:number_textbox.png]
Slider	[inline:slider.png]
Textarea Textarea expands to fit content	[inline:textarea.png]
TextBox	[inline:textbox.png]
TimeTextBox	[inline:time_textbox.png]
ValidationTextBox	[inline:validating_textbox.png]

Layout

AccordionContainer	[inline:accordion_pane.png]
ContentPane	[inline:content_pane.png]
LayoutContainer	[inline:layout_container.png]
SplitContainer	[inline:split_container.png]
StackContainer	[inline:stack_container.png]

TabContainer	[inline:tab_container.png]

Command Control

Button, ComboButton, DropDownButton	[inline:button.png]
Menu	[inline:menu.png]
Toolbar	[inline:toolbar.png]

User Assistance and Feedback

Dialog	[inline:dialog_box.png]
TooltipDialog	[inline:tooltipDialog.png]
ProgressBar	[inline:progress_bar.png]
TitlePane Click arrow to expand/contract	[inline:title_pane.png]
Tooltip	[inline:tooltip.png]

Advanced Editing and Display

ColorPalette	[inline:color_picker.png]
Editor	[inline:editor.png]
Grid (1.0)	[inline:grid_terms.gif]
InlineEditBox 0.9 - 1.0	[inline:inline_edit.png]
Tree	[inline:tree.png]

Common Features

Attributes, Methods, Extension Points

Each Dijit component has interaction points, so you can customize them for your app. In order of complexity:

- An **attribute** is a data element used for controlling display or behavior. For example, the label attribute of a ContentPane displays when that pane is part of a TabbedContainer.

  Attributes are settable only at creation time. That means you can set them in the declarative tag, or in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `new dijit.___()` call. After that, you may not set them directly. Some attributes can be changed by a method call, usually with the name /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `setAttributeName()`. We'll note these in the guide where applicable.

- A **method** is a function you call to control something. For example, in a NumberSpinner the user can click the Up and Down buttons to adjust a number. But you can call the method adjust() to do the same thing.
- An **extension point** is a function you provide to override behavior. For example, a ValidationTextBox has an isValid() extension point. If your validation involves several fields or needs more power than regular expressions provide, you can create a new subclass of ValidationTextBox and provide your own isValid function.

Common Attributes

These attributes may be specified in any widget, in addition to all the normal HTML attributes (which are simply passed through unchanged to the finished widget).

Attribute	Type or values	Default	Description

domNode	Node	None	this is our visible representation of the widget! Other DOM Nodes may by assigned to other properties, usually through the template system's dojoAttachPoint syntax, but the domNode property is the canonical "top level" node in widget UI. In non-templated widgets, this will equal the srcNodeRef.
id	String	None	a unique, opaque ID string that can be assigned by users or by the system. If the developer passes an ID which is known not to be unique, the specified ID is ignored and the system-generated ID is used instead.
lang	String	djConfig.locale or default provided by navigator object	overrides the page-wide setting to use a different language for this widget only. Must choose from bootstrapped languages in djConfig.extraLocale. It's usually best to default to the page settings so that the page can be localized with a single setting or the default. Use of this attribute is typically limited to demonstrations or pages with multiple languages besides the user's.
layoutAlign	String	Depends onorder within LayoutContainer	Only applies to Dijit components in dijit.layout.LayoutContainer. Possible values: "left", "right", "bottom", "top", and "client". The LayoutContainer takes its children marked as left/top/bottom/right, and lays them out along the edges of the box, and then it takes the child marked "client" and puts it into the remaining space in the middle.
srcNodeRef	Node	DOM Node which has the dojoType attribute.	Consider this attribute Read Only. In templated widgets, usually this node will disappear, replaced by a filled-in template.

Extension Points

Both methods and extension points are JavaScript functions, and the difference between them is a semantic one. We use the term method for "functions programmers normally call" and extension point for "functions the Dijit component normally calls". There's no technical reason you couldn't call an extension point yourself, or override a method yourself, but it makes little sense to do so.

You can easily provide an extension point function for declarative elements. As an example, here's how to extend a ValidationTextBox:

1. First, find the extension point on the appropriate Dijit page. In our case, it's dijit.form.ValidationTextBox.isValid
2. Copy the signature from the API documentation. For example, isValid() for ValidatingTextBox has signature function(/* Boolean*/ isFocused) and returns a boolean.
3. Then, use a dojo/method call directly in the markup, like this: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<input dojoType="dijit.form.ValidatingTextBox" isValid="my.form.isValid"  ...>
        <script type="dojo/method" event="isValid">
            // Flip a coin to determine validity.  *evil laughter!*
            return Math.round(Math.random() * 100) % 2 == 0;
        </script>
   </input>
```

Of course, if your isValid function is the same for many widgets, you don't want to define it over and over. So alternatively:

1. Follow first 2 steps above
2. Write your function: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
my.form.isValid = function(isFocused) {
   // Flip a coin to determine validity.  *evil laughter!*
   return Math.round(Math.random() * 100) % 2 == 0;
}
```
3. Connect the function to the extension point through an HTML attribute: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<input dojoType="dijit.form.ValidatingTextBox" isValid="my.form.isValid" .../>
```

Other Things You Can Do With Dijit

- You can change the style individual components, a Dijit class, or even create an entire theme spanning all Dijit components. The Themes and Design section tells you how.
- You can create all Dijit components programmatically as well as declaratively (through markup). All attributes, methods and extension points are available through JavaScript.
- You can create subclasses of existing Dijit classes, or create your own from scratch, described in the Writing Your Own Widget section.

Form, Validation, Specialized Input

The following widgets can be used in a FORM tag, in a dijit.form.Form widget, or outside of a form.

All form widgets implement the following attributes and methods. Many of them act as "shadow" attributes for their HTML counterparts, as in "name" or "id".

dijit.form._FormWidget
Base class for all form widgets - that is, all widgets beginning with "dijit.form"

Attributes

disabled	Boolean	Should this widget respond to user input? In markup, this is specified as "disabled='disabled'", or just "disabled". *Use setDisabled() to change after creation time.*
intermediateChanges	Boolean	If true, trigger an onChange event every time setValue is called. If false, trigger onChange only when asked by the callee. For example, on Slider a true value would fire onChange events at each point of the mouse drag. False would trigger onChange only on mouseUp.
tabIndex	Integer	Order fields are traversed when user hits the tab key

Methods

focus	Set the focus on this widget
getValue	get the value of the widget.
setDisabled	Set disabled state of widget.
setValue	set the value of the widget.
undo	restore the value to the last value passed to onChange

Extension Points

onChange	callback when value is changed

CheckBox, RadioButton	[inline:checkbox.png]
ComboBox	[inline:combo_box.png]
CurrencyTextBox	[inline:currency_textbox.png]
DateTextBox	[inline:date_textbox.png]
FilteringSelect	[inline:filtering_select.png]
InlineEditBox (0.9) For 1.0, see dijit.InlineEditBox	[inline:inline_edit.png]
NumberSpinner	[inline:number_spinner.png]
NumberTextBox	[inline:number_textbox.png]
Slider	[inline:slider.png]
Textarea Textarea expands to fit content	[inline:textarea.png]
TextBox	[inline:textbox.png]
TimeTextBox	[inline:time_textbox.png]
ValidationTextBox	[inline:validating_textbox.png]

Form Widget

Although you're not required to place Dijit form elements in a dijit.form.Form, doing so gets you some nice methods and extension points to use.

dijit.form.Form
Adds conveniences to regular HTML form.

Methods

getValues	generate JSON structure from form values get widget values
isValid	Return true if every widget's isValid method returns true.
setValues	fill in form values from a JSON structure generate map from name --> [list of widgets with that name]
submit	programatically submit form

Extension Points

execute	User defined function to do stuff when the user hits the submit button

CheckBox, RadioButton, ToggleButton

dijit.form.CheckBox, dijit.form.RadioButton, and dijit.form.ToggleButton capture binary user-choices. Unlike command buttons, which *do* some action on being pressed, these buttons are more for form data. ToggleButtons can be used on Toolbars - the Bold button being the classic example - so they act a little like a data button, a little like a command button.

Examples

CheckBoxes in dijit are very intuitive and easy to use. Markup constructs for check boxes is the same as html but dojo provides more control and styling options than a conventional check box. The following example creates a CheckBox:

Are you a Developer

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Checkbox Example</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.form.CheckBox");
     </script>
</head>
<body class="tundra">
    <input id="cb" dojotype="dijit.form.CheckBox"
           name="developer" checked="checked" value="on"
           type="checkbox" />
    <label for="cb"> Are you a Developer </label>
</body></html>
```

dijit.form.ToggleButton works very similarly to a checkbox, but requires including the "dijit.form.Button" module.

RadioButtons in dijit are also easy to create and use as the following example shows:

Metallica Extreme Slayer

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Radio Button Example</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.form.CheckBox");
    </script>
</head>
<body class="tundra">
    <input dojoType="dijit.form.RadioButton" id="sum1" name="album"
           checked="checked" value="metallica" type="radio" />
           <label for="sum1"> Metallica </label>
    <input dojotype="dijit.form.RadioButton" id="sum2"  name="album"
           value="Extreme" type="radio" />
           <label for="sum2"> Extreme </label>
    <input dojotype="dijit.form.RadioButton" id="sum3"  name="album"
           value="Slayer" type="radio" />
           <label for="sum3"> Slayer </label>
</body></html>
```

The RadioButton class is declared in the CheckBox.js file, hence you need to dojo.require() only dijit.form.CheckBox for RadioButtons to work.

dijit.form.CheckBox, dijit.form.RadioButton, dijit.form.ToggleButton
Checkbox, RadioButton and ToggleButton capture binary user choices. Checkbox and RadioButton are like their HTML counterparts, while ToggleButton can be pushed in or out (like the Bold button in word processors).

Attributes

checked	String	Corresponds to the native HTML input element's attribute. In markup, specified as "checked='checked'" or just "checked". True if the button is depressed, or the checkbox is checked, or the radio button is selected, etc. *Use setChecked() to change after creation time.*

Methods	
setChecked(/* Boolean */ checked)	If true, turn button or box on.
Extension Points	
onClick(/*Event*/ e)	Called when widget is clicked.

Accessibility

Keyboard

Widget	Keyboard Interaction
ToggleButton	Same as HTML button
Checkbox	Same as HTML input type checkbox
RadioButton	Same as HTML input type radio

ComboBox

The ComboBox is a hybrid between a SELECT combo box and a text field. Like a SELECT, you provide a list of acceptable values. Unlike SELECT, and like a text box, the user can ignore all the choices and type whatever they want. This is especially good for open-ended multiple choice questions. Rather than having two fields - a combo box and an Other text box - you can use just one field.

Note that SELECT's always have value/description pairs, e.g. the OPTION's value attribute and the OPTION's body text. ComboBoxes do not. They only pass their displayed text - just like a text box.

Examples

Using inlined data with the ComboBox is very much like using a native `<select>` tag:

California

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Simple ComboBox</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.ComboBox");
       function setVal1(value) {
           console.debug("Selected "+value);
       }
   </script>
</head>
<body class="tundra">
        <select name="state1"
                dojoType="dijit.form.ComboBox"
                autocomplete="false"
                value="California"
                onChange="setVal1">
                <option selected="selected">California</option>
                <option >Illinois</option>
                <option >New York</option>
                <option >Texas</option>
        </select>
</body></html>
```

Name and autocomplete are passed through to the HTML. The value attribute enables you to set the default value. The option tags do not have hidden submit values; to use a hidden value, change your ComboBoxes to FilteringSelects.

Important: IE7 only uses the selected attribute of an option tag and ignores the value attribute on the select tag. For best results, set both the selected attribute on the default option and the value attribute on the select.

dojo.Data-Enabled ComboBox

ComboBox, FilteringSelect, Tree and Grid are dojo.data-Enabled widgets. This means rather than specifying all the data in the page - the OPTION's or tree nodes - you can have dojo.data fetch them from a server-based store. The unified dojo.data architecture, can get its data from various places - databases, web services, etc.. See the dojo.data manual section for complete details.

The code:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<div dojoType="dojo.data.ItemFileReadStore" jsId="stateStore"
             url="states.txt"></div>
```

... looks like a widget, but it's not. (Only dojoTypes from the module dijit.___ are widgets). This tag takes advantage of dojo.parser, which creates JavaScript objects by using standard HTML. You can read about Dojo.parser at Understanding the Parser in Part 3.

For this example, we'll use a fixed JSON data store. You can download it from states.txt below. Here's an excerpt:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
{
identifier:"abbreviation",
items: [
        {name:"Alabama", label:"Alabama",abbreviation:"AL"},
        {name:"Alaska", label:"Alaska",abbreviation:"AK"},
        {name:"American Samoa", label:"American Samoa",abbreviation:"AS"},
        {name:"Arizona", label:"Arizona",abbreviation:"AZ"},
```

The following example makes use of our data store:

California

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Simple ComboBox</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.ComboBox");
       dojo.require("dojo.data.ItemFileReadStore");
       function setVal2(value) {
           console.debug("Selected "+value);
       }
   </script>
</head>
<body class="tundra">
        <div dojoType="dojo.data.ItemFileReadStore" jsId="stateStore"
             url="states.txt"></div>

     <input dojoType="dijit.form.ComboBox"
              store="stateStore"
              value="California"
              searchAttr="name"
              name="state2"
              onChange="setVal2" />

</body></html>
```

`jsId` is the name of the global variable that the store is assigned to. `url` can be used to suck data out of particular URL, perhaps a web service. Instead of using the `url` attribute, you can embed your data directly (as is common in traditional server-side programming) using the `data` attribute.

dijit.form.ComboBox
Auto-completing text box The drop down box's values are populated from an class called a data provider, which returns a list of values based on the characters that the user has typed into the input box. Some of the options to the ComboBox are actually arguments to the data provider.

Attributes

autoComplete	Boolean true	If you type in a partial string, and then tab out of the <input> box, automatically copy the first entry displayed in the drop down list to the <input> field
hasDownArrow	Boolean true	Set this textbox to have a down arrow button
ignoreCase	Boolean true	True if the ComboBox should ignore case when matching possible items.

pageSize	Integer Infinity	Argument to data provider. Specifies number of search results per page (before hitting "next" button)
query	Object {}	A query that can be passed to 'store' to initially filter the items, before doing further filtering based on searchAttr and the key.
searchAttr	String name	Searches pattern match against this field
searchDelay	Integer 100	Delay in milliseconds between when user types something and we start searching based on that value
store	Object	Reference to data provider object used by this ComboBox.

Accessibility

Keyboard

Action	Key
Open the menu of options (filtered by current input)	Down arrow
Navigate through the options	Up and down arrows
Pick an option	Enter
Close the menu of options without picking one	Esc

Known Issues

JAWS 8 and Window-Eyes 6 may fail to read an option when it becomes highlighted.

FilteringSelect

FilteringSelect is like an HTML SELECT tag, but is populated dynamically. It works very nicely for select's with potentially thousands of entries because it does not need all the options to be loaded at once. It also resembles ComboBox, but does not allow values outside of the provided ones.

When the user tries to submit invalid input (say if they choose an option, and the legal options change) the user gets a warning message, but the Select keeps text box input as is and also keeps the last valid submit value. If the user selects the text box and presses Escape on the keyboard, the text box reverts to the last valid value, corresponding to the hidden value. This change guarantees that you will always get a valid submit value.

Examples

First, here is a FilteringSelect with inlined data:


```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Filter Select Example 1</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.FilteringSelect");
     </script>
</head>
<body class="tundra">
        <select dojoType="dijit.form.FilteringSelect"
        name="state3"
        autocomplete="false"
        value="CA">
                <option value="CA" selected="selected">California</option>
                <option value="IL" >Illinois</option>
                <option value="NY" >New York</option>
                <option value="TX" >Texas</option>
        </select>
</body></html>
```

As with ComboBox, has a value attribute. Unlike ComboBox, this value refers to the value attribute of the <option> tag. For example, if you

set the value to AL, the text "Alabama" will appear in the text box on load. If you want to change the value attribute programmatically, use

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dijit.byId("yourwidgetid").setValue("yourhiddenvalue")
```

Like ComboBox FilteringSelect is dojo.data-enabled. As with all dojo.data stores, you can add an identifier field to the top level of your data. The value of the identifier field tells the store which field in your data contains the submit value. Here's an example from states.txt:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
{
identifier:"abbreviation",
items: [
        {name:"Alabama", label:"Alabama",abbreviation:"AL"},
        ...
```

This code shows an identifier set to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `abbreviation`. The identifier instructs the dojo.data store to set the submit value of the items in this store to the value of the attribute named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `abbreviation`. In this example, the first item has a field named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `abbreviation` with a value of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `AL`. If one of your users selected that item in your FilteringSelect, the form would submit /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `AL` to your server.

Here's the corresponding FilteringSelect

Go!

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Data-Enabled FilteringSelect</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
        <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.FilteringSelect");
       dojo.require("dojo.data.ItemFileReadStore");
   </script>
</head>
<body class="tundra">
        <div dojoType="dojo.data.ItemFileReadStore" jsId="stateStore"
             url="states.txt"></div>
        <form method="post">
               <input dojoType="dijit.form.FilteringSelect"
                   store="stateStore"
                       searchAttr="name"
                       name="state1"
                       autocomplete="true"
                       />
            <input type="submit" value="Go!" />
        </form>
</body></html>
```

The net result of the identifier is that you can easily set the submit attribute of any number of Selects using the same data without actually adding extra attributes to the Selects.

dijit.form.FilteringSelect		
Attributes		
autoComplete	Boolean true	If you type in a partial string, and then tab out of the <input> box, automatically copy the first entry displayed in the drop down list to the <input> field
hasDownArrow	Boolean true	Set this textbox to have a down arrow button/td>
ignoreCase	Boolean true	Does the ComboBox menu ignore case?
labelAttr	String searchAttr	String Searches pattern match against this field String Optional. The text that actually appears in the drop down. If not specified, the searchAttr text is used instead.
labelType	String text	"html" or "text"
pageSize	Integer Infinity	Argument to data provider. Specifies number of search results per page (before hitting "next" button)
query	Object {}	A query that can be passed to 'store' to initially filter the items, before doing further filtering based on searchAttr and the key.
searchAttr	String name	Searches pattern match against this field
searchDelay	Integer 100	Delay in milliseconds between when user types something and we start searching based on that value
store	String	Reference to data provider object used by this ComboBox.
Methods		
setDisplayedValue(/*String*/ label)		Set label (and corresponding value) to "label"
setValue(/*String*/ value)		Set value (and corresponding label) to "value"

Accessibility

Keyboard

Action	Key
Open the menu of options (filtered by current input)	Down arrow
Navigate through the options	Up and down arrows
Pick an option	Enter
Close the menu of options without picking one	Esc

Known Issues

JAWS 8 and Window-Eyes 6 may fail to read an option when it becomes highlighted.

InlineEditBox (0.9)

In 1.0, dijit.form.InlineEditBox has been deprecated in favor of dijit.InlineEditBox

InlineEditBox is better described as a widget wrapper, which takes a child widget declared in markup and makes it inline-editable. This widget acts like a tag when not in edit mode. When the contained rendered text is clicked, the widget enters an edit mode. In this mode, the previously displayed text is hidden, and another widget capable of editing text is made visible in its place.

Examples

Edit me - I trigger the on(

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
```

```
<head>
<title>InlineEdit Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/0.9.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/0.9.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.InlineEditBox");
       dojo.require("dijit.form.TextBox");
       function myHandler(idOfBox, value) {
           console.debug("Edited value from "+idOfBox+" is now "+value);
       }
     </script>
</head>
<body class="tundra">
        <span id="editable" style="font-size:larger;"
              dojoType="dijit.form.InlineEditBox"
              onChange="myHandler(this.id,arguments[0])">
        <input dojoType="dijit.form.TextBox" value="Edit me - I trigger the onChange callback">
    </span>
</body></html>
```

The outermost span is the InlineEditBox. The inner input tag is the TextBox widget. When a user loads the page, they see the text "Edit me - I trigger the onChange callback". If the user clicks the text, a TextBox widget containing the text "Edit me - I trigger the onChange callback" appears. When the user changes the value and clicks away, the TextBox disappears and the TextBox's contents appear inline.

InlineEditBox supports the textarea mode through the Textarea widget. By simply adding a Textarea inside the InlineEditBox HTML tag, you add inline-editing to the Textarea widget. Furthermore, by adding renderAsHtml=true, users can enter HTML into the Textarea and have it appear inline as rich text. :


```
                        I'm one
big paragraph.  Go ahead and
<i>edit</i> me.  <b>I dare
you.</b>
                        The
quick brown fox jumped over the
lazy dog.  Blah blah blah blah
blah blah blah ...
```

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>InlineEdit Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/0.9.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.InlineEditBox");
       dojo.require("dijit.form.Textarea");
     </script>
</head>
<body class="tundra">
    <span id="areaEditable" dojoType="dijit.form.InlineEditBox"
        renderAsHtml="true" autoSave="false">
            <textarea dojoType="dijit.form.Textarea">
                    I'm one big paragraph.  Go ahead and <i>edit</i> me.  <b>I dare you.</b>
                    The quick brown fox jumped over the lazy dog.  Blah blah blah blah blah blah blah ...
            </textarea>
        </span>
</body></html>
```

The outermost span in this code is the InlineEditBox. The inner textarea tag is the Textarea widget. When a user loads the page, they see the paragraph of rich text. If the user clicks the text, a Textarea widget containing the paragraph in plain text form appears. When the user changes the value and clicks away, the Textarea disappears and the Textarea's contents appear inline.

InlineEditBox can make any arbitrary widget that has a text value, or has the methods get/setDisplayedValue, inline. DateTextBox is an example of such a widget. This code shows a DateTextBox made inline in HTML:

2005-12-30

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>InlineEdit Demo</title>
    <style type="text/css">
```

```
        @import "http://o.aolcdn.com/dojo/0.9.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.InlineEditBox");
       dojo.require("dijit.form.DateTextBox");
     </script>
</head>
<body class="tundra">
    <p id="backgroundArea" dojoType="dijit.form.InlineEditBox" >
            <input name="date" value="2005-12-30"
                    dojoType="dijit.form.DateTextBox"
                    constraints={datePattern:'MM/dd/yy'}
                    lang="en-us"
                    required="true"
                    promptMessage="mm/dd/yy"
                    invalidMessage="Invalid date. Use mm/dd/yy format.">

</body></html>
```

The InlineEditBox can wrap around any widget that implements the following interface:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
/* void */ setTextValue(/*String*/ value) { ... }
String value = getTextValue() {... }
/* void */ focus() { ... }
```

The contained widget's setTextValue() method is called with the previously displayed text. When the Save button is pressed, the editing widget's getTextValue() method is called to retrieve the new text. After which, the editing widget is hidden, and the returned text is displayed. The focus method allows the editing widget to intelligently set focus to an appropriate node.

dijit.form.InlineEditBox
Wrapper widget for holding click-to-edit text.

Methods

addChild	Process the given child widget, inserting it's dom node as a child of our dom node
getChildren	returns array of children widgets
hasChildren	returns true if widget has children
removeChild	removes the passed widget instance from this widget but does not destroy it

Accessibility

General Behavior

InlineEditBoxes are wrappers around the various types of dojo TextBoxes and Textareas. When they are activated the underlying dojo widget is activated. When they are "closed" they appear as text but are tab stops in the keyboard focus ring and have an accessible role of button. They can have autoSave or non-autoSave behavior. When an non-autoSave InlineEditBox is open it has associated Save and Cancel buttons. An autoSave InlineEditBox does not have these buttons and they act like miniature forms or dialogs, i.e pressing the Esc key will close the widget and pressing the Enter key will close the widget, saving and displaying the text.

Note that since InlineEditBoxes may be used on the page without a traditional label element, the developer should add a title attribute in order to provide a description that is available to screen reader users. The title will also be displayed by the browser when the user places the mouse over the element.

Keyboard

If the widget is closed.

Action	Key
Navigate to the next widget in the tab order.	Tab
Navigate to the prior widget in the tab order.	Shift+Tab
Open the widget.	Enter or spacebar

Note: The Esc key is ignored.

TextBox with autoSave specified and the TextBox is open:

Action	Key	Comments
Navigate to the next widget in the tab order.	Tab	The data is saved and the widget closes.
Navigate to the prior widget in the tab order.	Shift+Tab	The data is saved and the widget closes.
Close the TextBox, saving changes.	Enter	Keyboard focus is on the closed InlineEditBox.
Revert the last entry.	Esc	If the user has not entered data, the TextBox is closed.
Close the Textarea, discarding changes.	Esc	If the user has entered data, the Esc must be pressed two times; the first time the data will be reverted; the second time the TextBox will close.

Textarea with autoSave specified and the Textarea is open:

Action	Key	Comments
Navigate to the next widget in the tab order.	Tab (press twice in Firefox - see the Known Issues below)	The data is saved and the widget closes.
Navigate to the prior widget in the tab order.	Shift+Tab	The data is saved and the widget closes.
Enter a newline into the text.	Enter	There is no equivalent to the Enter key behavior of TextBoxes. The user would have to use something like Tab and Shift + Tab.
Revert the last entry.	Esc	If the user has not entered data, the Textarea is closed.
Close the Textarea, discarding changes.	Esc	If the user has entered data, the Esc must be pressed two times; the first time the data will be reverted; the second time the Textarea will close.

TextBox without autoSave specified, the TextBox is open, keyboard focus is in the edit field:

Action	Key	Comments
Navigate to the Save or Cancel button.	Tab	Focus changes to the Save button if the data has been changed, otherwise it moves to the Cancel button.
Navigate to the prior widget in the tab order.	Shift+Tab	The TextBox remains open.
Close the TextBox, saving changes.	Tab to the Save button, then press the Enter key	Keyboard focus is on the closed InlineEditBox.
Revert the last entry.	Esc	If the user has not entered data, the Esc key is ignored.
Close the Text Box, discarding changes.	Tab to the Cancel button, then press the Enter key.	Keyboard focus is on the closed InlineEditBox.

Note: The Enter key is ignored when focus is in the edit field.

Textarea without autoSave specified, the Textarea is open, keyboard focus is in the edit field:

Action	Key	Comments
Navigate to the Save or Cancel button.	Tab (press twice in Firefox - see the Known Issues below)	Focus changes to the Save button if the data has been changed, otherwise it moves to the Cancel button.
Navigate to the prior widget in the tab order.	Shift+Tab	The Textarea remains open.
Close the Textarea, saving changes.	Tab to the Save button, then press the Enter key	Keyboard focus is on the closed InlineEditBox.
Revert the last entry.	Esc	If the user has not entered data, the Esc key is ignored.
Close the Textarea, discarding changes.	Tab to the Cancel button, then press the Enter key.	Keyboard focus is on the closed InlineEditBox.

Note: Pressing the Enter key results in a newline being inserted into the edit field.

Known Issues

- See the Comment for the Enter key in the information for autoSaving Text Areas above.

- Ticket 3910: When Inline Text Boxes are opened, all the text should be selected.
- On Firefox 2, the user must press the Tab key twice with focus in an textarea before keyboard focus moves to the next widget. This is a permanent restriction on Firefox 2. This is because the Dojo text area is implemented using the Firefox editor component in an iframe. This editor component implements usage of the tab key within the editor to indent text and shift-tab to outdent text. There is no keyboard mechanism in Firefox to move focus out of the editor. So, the dijit editor traps the tab key in the editor and sets focus to the editor iframe. From there pressing tab again will move to the next focusable item after the editor.

Screen Reader Issues

The InlineEditBox is implemented as a button. Since these are intended to be used "in-line" within text there is often no label element associated with the underlying control. For this reason, developers are encouraged to add a title attribute to InlineEditBoxes. The Window-Eyes screen reader will speak the title as part of the button description. JAWS has an option to speak different attributes on an button. A JAWS user may need to use the insert-v command to modify the behavior to speak the button title when working with Dojo InlineEditBoxes.

NumberSpinner

The Number Spinner, a familiar widget in GUI interfaces, makes integer entry easier when small adjustments are required. The down and up arrow buttons "spin" the number up and down. Furthermore, when you hold down the buttons, the spinning accelerates to make coarser adjustments easier.

Example

1000

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Number Spinner Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
        <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.NumberSpinner");
    </script>
</head>
<body class="tundra">
        <input dojoType="dijit.form.NumberSpinner"
                value="1000"
                smallDelta="10"
                constraints="{min:9,max:1550,places:0}"
                maxlength="20"
                id="integerspinner2">
</body></html>
```

dijit.form.NumberSpinner
Number Spinner

Attributes

constraints	Object {}	min and max properties are used for bounds. See ValidationTextBox for details.
defaultTimeout	Integer 500	number of milliseconds before a held key or button becomes typematic
invalidMessage	String locale dep.	The message to display if value is invalid. constraints: Object user-defined object needed to pass parameters to the validator functions
intermediateChanges	Boolean	If true, trigger an onChange event every time setValue is called. If false, trigger onChange only when asked by the callee. For example, on Slider a true value would fire onChange events at each point of the mouse drag. False would trigger onChange only on mouseUp.
largeDelta	Number 10	adjust the value by this much when spinning using the PgUp/Dn keys
promptMessage	String	Hint string
rangeMessage	String	The message to display if value is out-of-range
required	Boolean true	Defaults to true in NumberSpinner because the arrows won't work otherwise.
smallDelta	Number 1	adjust the value by this much when spinning using the arrow keys/buttons

timeoutChangeRate	Number 0.90	fraction of time used to change the typematic timer between events 1.0 means that each typematic event fires at defaultTimeout intervals < 1.0 means that each typematic event fires at an increasing faster rate
trim	Boolean false	Removes leading and trailing whitespace if true. Default is false.

Slider

A slider is a scale with a handle you can drag up/down or left/right to select a value. Calling dojo.require("dijit.form.Slider") provides dijit.form.HorizontalSlider, dijit.form.VerticalSlider and all the rule and label classes.

Examples

One way you could show the user the value of your Slider is to create a textbox that the Slider fills when the user moves the Slider. The following code fills in a simple textbox called `horizontalSliderValue`.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Slider Example 1</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Slider");
    </script>
</head>
<body class="tundra">
<div id="horizontalSlider" dojoType="dijit.form.HorizontalSlider"
   value="5" minimum="-10" maximum="10" discreteValues="11"
   intermediateChanges="true"
   onChange="dojo.byId('horizontalSlider').value = arguments[0];"
   handleSrc="http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/images/preciseSliderThumb.png"
></div>
</body></html>
```

You can point the handleSrc image to wherever you want. If you want to use the default handle image, just remove the handleSrc.

For a horizontal slider, you can use the HorizintalRule and HorizontalRuleLabels to create your ruler marks programmatically, reducing the amount of data being transferred over the wire:

1.
2. 20%
3. 40%
4. 60%
5. 80%
6. 100%

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Slider Example 2</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Slider");
    </script>
</head>
<body class="tundra">
<div id="horizontalSlider" dojoType="dijit.form.HorizontalSlider"
        value="5" minimum="-10" maximum="10" discreteValues="11"
        intermediateChanges="true"
        showButtons="true">
                <div dojoType="dijit.form.HorizontalRuleLabels" container="topDecoration"
```

```
             style="height:1.2em;font-size:75%;color:gray;"></div>
                <ol dojoType="dijit.form.HorizontalRuleLabels" container="topDecoration"
            style="height:1em;font-size:75%;color:gray;">
            <li> </li>
            <li>20%</li>
            <li>40%</li>
            <li>60%</li>
            <li>80%</li>
            <li> </li>
        </ol>
                <div dojoType="dijit.form.HorizontalRule" container="bottomDecoration"
             count=5 style="height:5px;"></div>
                <ol dojoType="dijit.form.HorizontalRuleLabels" container="bottomDecoration"
            style="height:1em;font-size:75%;color:gray;">
            <li>0%</li>
            <li>50%</li>
            <li>100%</li>
        </ol>
</div>
</div>
</body></html>
```

The VerticalSlider API is identical to the HorizontalSlider API. You can use the VerticalRule class to create vertical ruler marks.

dijit.form.HorizontalSlider, dijit.form.VerticalSlider
A form widget that allows one to select a value with a horizontally (or vertically) draggable image

Attributes

clickSelect	boolean true	If clicking the progress bar changes the value or not
discreteValues	integer Infinity	The maximum allowed values dispersed evenly between minimum and maximum (inclusive).
intermediateChanges	Boolean false	If true, trigger an onChange event every time setValue is called. If false, trigger onChange only when asked by the callee. For example, on Slider a true value would fire onChange events at each point of the mouse drag. False would trigger onChange only on mouseUp.
maximum	integer 100	The maximum allowed value.
minimum	integer 0	The minimum value allowed.
pageIncrement	integer 2	The amount of change with shift+arrow
showButtons	boolean true	Show increment/decrement buttons at the ends of the slider?

Methods

setValue((/*Number*/ value, /*Boolean, optional*/ priorityChange)	Regular setValue, but if priorityChange is true, then it is more likely to be animated.

dijit.form.HorizontalRule, dijit.form.VerticalRule
Create hash marks for the Horizontal/Vertical slider

Attributes

container	Node containerNode	If this is a child widget, connect it to this parent node
count	Integer 3	Number of hash marks to generate
ruleStyle	String	CSS style to apply to individual hash marks

dijit.form.HorizontalRuleLabels, dijit.form.VerticalRuleLabels
Create labels for the Slider

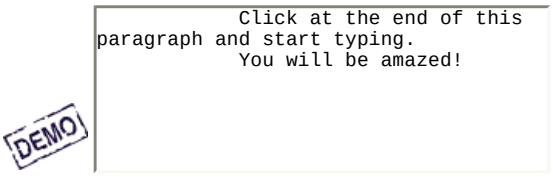
Attributes

labels	Array []	Array of text labels to render - evenly spaced from left-to-right or bottom-to-top
labelStyle	String	CSS style to apply to individual text labels

Textarea

A Textarea widget is like a regular HTML textarea, but it dynamically resizes to fit the content of the text inside. It takes nearly all the parameters (name, value, etc.) that a vanilla textarea takes. Cols is not supported and the width should be specified with style width. Rows is not supported since this widget adjusts the height. It is especially useful in an InlineEditBox. Note that when declaring a Textarea in markup you should use a <textarea> node to preserve the newline formatting.

Example



/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Textarea Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Textarea");
    </script>
</head>
<body class="tundra">
        <textarea dojoType="dijit.form.Textarea" style="width:300px">
            Click at the end of this paragraph and start typing.
            You will be amazed!
        </textarea>
</body>
</html>
```

Accessibility

Keyboard

Action	Key	Comments
Move focus to the next widget in the tab order.	Tab	
Move focus to the prior widget in the tab order.	Shift+Tab	
Enter a newline into the text.	Enter	
Revert the last entry.	Esc	If the user has not entered data, the Esc key is ignored.

Known Issues

- Ticket 3902: When the Text Area widget is first opened the caret is at the start; it should be at the end.
- On Firefox 2, the user must press the Tab key twice before keyboard focus moves to the next widget. (There is no problem when using Shift+Tab.) This is a permanent restriction on Firefox 2.

TextBox family: Validation, Currency, Number, Date, Time

These widgets augment the functionality of the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<INPUT type="text">` tag. The base widget dijit.form.Textbox by itself can trim, change case, and require input. dijit.form.ValidationTextbox extends this by validating the input when the box loses focus. The other widgets further extend the validation function with range and format checking. Internal to the MappedTextBox widget subclass are two INPUT elements. One interacts with the user obeying local customs, the other is hidden and represents the named form element to submit data to the server using a normalized serialization. By default, the widget will discover the appropriate locale and behavior as specified by Dojo.

For example, when using a NumberTextBox in the United States, an optional comma is used for the thousands separator and a period for a decimal separator when interacting with the user. For German users, a period is used for the thousands separator and a comma for the decimal separator. Other locales may have different conventions. When sending data to the server or interpreting the "value" attribute, numbers are represented simply as JavaScript formats them with a period for decimal and no thousands separators. This representation is unambiguous, so other applications may interact with this data without assuming any locale-specific behavior. With DateTextBox, a subset of the ISO-8601 format (e.g. 12-31-2006) is used for the value attribute. For CurrencyTextBox, a number is transmitted, and it is the responsibility of the developer to associate the ISO-4217 country code with the amount to qualify what type of currency is indicated. All of these behaviors are considered Dojo and JSON best practices, but may be customized as described below.

Constraints

To override the defaults, you can use the "constraints" attribute. "constraints" is an object passed to functions responsible for validating,

parsing, and formatting the data in the box, and various properties may be provided to override system or locale-specific defaults. Constraints are handled in Dojo low-level routines in dojo.date, dojo.currency and dojo.number, and you can refer to the API documentation for complete details. We summarize them here for convenience:

For CurrencyTextBox and NumberTextBox:

- **currency**: (currency only) the ISO-4217 currency code, a three letter sequence like "USD" See http://en.wikipedia.org/wiki/ISO_4217
- **fractional**: (currency only) where places are implied by pattern or explicit 'places' parameter, whether to include the fractional portion.
- **locale**: override the locale on this widget only, choosing from djConfig.extraLocale
- **pattern**: override localized convention with this pattern. As a result, all users will see the same behavior, regardless of locale, and your application may not be globalized. See http://www.unicode.org/reports/tr35/#Number_Format_Patterns.
- **places**: number of decimal places to accept.
- **strict**: strict parsing, false by default. When strict mode is false, certain allowances are made to be more tolerant of user input, such as 'am' instead of 'a.m.', some white space may be optional, etc.
- **symbol**: (currency only) override currency symbol. Normally, will be looked up in localized table of supported currencies (dojo.cldr) 3-letter ISO 4217 currency code will be used if not found.
- **type**: choose a format type based on the locale from the following: decimal, scientific (not yet supported), percent, currency. decimal by default.

For DateTextBox and TimeTextBox:

- **am,pm**: override strings for am/pm in times
- **datePattern,timePattern**: override localized convention with this pattern. As a result, all users will see the same behavior, regardless of locale, and your application may not be globalized. See http://www.unicode.org/reports/tr35/#Date_Format_Patterns
- **formatLength**: choose from formats appropriate to the locale -- long, short, medium or full (plus any custom additions). Defaults to 'short'
- **locale**: override the locale on this widget only, choosing from djConfig.extraLocale
- **selector**: choice of 'time', 'date' (default: date and time)
- **strict**: false by default. If true, parsing matches exactly by regular expression. If false, more tolerant matching is used for abbreviations and some white space.

Examples

dijit.form.TextBox

testing testing

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>TextBox Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.TextBox");
    </script>
</head>
<body class="tundra">
        <input type="text" name="firstname" value="testing testing"
                dojoType="dijit.form.TextBox"
                trim="true"
                propercase="true" />
</body></html>
```

dijit.form.DateTextBox

2005-12-30

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Textarea Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
```

```
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.form.DateTextBox");
    </script>
</head>
<body class="tundra">
        <input type="text" name="date1" value="2005-12-30"
                dojoType="dijit.form.DateTextBox"
                required="true" />
</body></html>
```

dojo.form.CurrencyTextBox

54775.53

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>CurrencyTextBox Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.CurrencyTextBox");
    </script>
</head>
<body class="tundra">
    <form method="post">
        <input type="text" name="income1" value="54775.53"
                dojoType="dijit.form.CurrencyTextBox"
                required="true"
                constraints="{fractional:true}"
                currency="USD"
                invalidMessage="Invalid amount.  Include dollar sign, commas, and cents." />
    <input type="submit" value="Go!"/>
    </form>
</body></html>
```

The `value` attribute is a floating point number. This means that you can easily build CurrencyTextBoxes for a wide range of currencies without having to set a different value for each currency format. `fractional` is still set to true, but it is set inside the constraints object instead of on the widget.

3000

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>NumberTextBox Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.NumberTextBox");
    </script>
</head>
<body class="tundra">
        <input id="q05" type="text"
                dojoType="dijit.form.NumberTextBox"
                name= "elevation"
                value="3000"
                constraints="{min:-20000,max:20000,places:0}"
                promptMessage= "Enter a value between -20000 and +20000"
                required= "true"
                invalidMessage= "Invalid elevation."
                />
</body></html>
```

dijit.form.ValidationTextBox

ValidationTextBoxes usually use Regular Expression validation, as in the following example:

someTestString

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ValidationTextBox Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.ValidationTextBox");
    </script>
</head>
<body class="tundra">
        <input type="text" name="phone" class="medium" value="someTestString"
                dojoType="dijit.form.ValidationTextBox"
                regExp="[\w]+"
                required="true"
                invalidMessage="Invalid Non-Space Text.">
</body></html>
```

The regular expression syntax comes directly from JavaScript. The start and ending qualifiers of the regular expression, ^ and $, are implicit - you do not need to include them. This code demonstrates a ValidationTextBox that only accepts a 5 digit zip code.

00000

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ValidationTextBox Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.ValidationTextBox");
    </script>
</head>
<body class="tundra">
        <input type="text" name="zip" value="00000"
                dojoType="dijit.form.ValidationTextBox"
                regExp="\d{5}"
                required="true"
                invalidMessage="Invalid zip code.">
</body></html>
```

ValidationTextBox also supports functions that generate regular expressions. Having a generating function enables you to write much more dynamic Web applications. ValidationTextBox passes its constraints object to the generating function. The following code demonstrates a dynamic ValidationTextBox that only accepts a 5 digit zip code after 5:00PM, and only accepts a county name before then.

00000

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ValidationTextBox Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
```

```
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.ValidationTextBox");
           function after5(constraints){
                   var date=new Date();
                   if(date.getHours() >= 17){
                          return "\\d{5}";
                   }else{
                          return "\\D+";
                   }
                }
    </script>
</head>
<body class="tundra">
        <input type="text" name="zip" value="00000"
                dojoType="dijit.form.ValidationTextBox"
                regExpGen="after5"
                required="true"
                invalidMessage="Zip codes after 5, county name before then." />
</body></html>
```

The attributes of dijit.form.TextBox apply to all types of text boxes:

dijit.form.TextBox
A generic textbox field. Serves as a base class to derive more specialized functionality in subclasses.

Attributes

lowercase	Boolean false	Converts all characters to lowercase if true. Default is false.
maxlength	String	HTML INPUT tag maxlength declaration. **Updated in 1.0:** Attribute is now called "maxLength" for greater browser compatability.
propercase	Boolean false	Converts the first character of each word to uppercase if true.
size	String	HTML INPUT tag size declaration. **Updated in 1.0:** size is simply copied to the widget. It's not used by TextBox.
trim	Boolean false	Removes leading and trailing whitespace if true.
uppercase	Boolean false	Converts all characters to uppercase if true.

Methods

String getDisplayedValue()	**New in 1.0:** the displayed value, as opposed to the Value parameter passed via the form.
setDisplayedValue(/*String*/value)	Set the displayed value and fire IncrementalChange (see dijit..form._FormWidget)
setValue(value, /*Boolean, optional*/ priorityChange, /*String, optional*/ formattedValue)	Sets the value

dijit.form.ValidationTextBox
Use this widget for validation when you write your own validation routine or regular expression.

Attributes

constraints	Object {}	user-defined object used to pass parameters to the validator functions. In markup, this attribute must be defined using JavaScript Object syntax with curly braces and quoting as appropriate for strings. See the section above for details.
invalidMessage	String locale dep.	The message to display if value is invalid.
promptMessage	String	Hint string
rangeMessage	String	The message to display if value is out-of-range **Removed in 1.0**
regExp	String .*	ValidationTextBox only - no subclasses. Regular expression string used to validate the input. Do not specify both regExp and regExpGen
required	Boolean false	Can be true or false, default is false.

Extension Points

displayMessage(/*String*/ message)	User overridable method to display validation errors/hints. By default uses a tooltip.
String getErrorMessage(/*Boolean*/ isFocused)	return an error message to show if appropriate

String getPromptMessage(/* Boolean*/ isFocused)	return a hint to show if appropriate
Boolean isValid(/* Boolean*/ isFocused)	Over-ride with your own validation code in subclasses. Use this.textbox.value to get the value.
String regExpGen(/* Object */constraints)	ValidationTextBox only - no subclasses. User replaceable function used to generate regExp when dependent on constraints. Do not specify both regExp and regExpGen currently displayed message

dijit.form.DateTextBox, dijit.form.TimeTextBox
Text boxes which allow input of dates and times, with popups to assist in making a selection. Local conventions are used when interacting with the user. ISO-8601 format is used when sending data to and from the server (notably the value attribute) Most attributes below apply only to TimePicker. See RangeTextBox, ValidationTextBox and TextBox above for additional attributes and methods. The TimeTextBox is still beta level (particularly there is no styling).

Attributes

clickableIncrement	String	ISO-8601 string representing the amount by which every clickable element in the time picker increases. Set in non-Zulu time, without a time zone Example: "T00:15:00" creates 15 minute increments Must divide visibleIncrement evenly.
value	String	Date to display. Defaults to current time and date. Can be a Date object or an ISO-8601 string. If you specify a time zone as an offset from GMT (e.g. "-01:00"), the time will be converted to the local time. Otherwise, the time is considered to be in the local time zone. If you specify the date and isDate is true, the date is used. Example: if your local time zone is GMT -05:00, "T10:00:00" becomes "T10:00:00-05:00" (considered to be local time), "T10:00:00-01:00" becomes "T06:00:00-05:00" (4 hour difference), "T10:00:00Z" becomes "T05:00:00-05:00" (5 hour difference between Zulu and local time) "yyyy-mm-ddThh:mm:ss" is the format to set the date and time Example: "2007-06-01T09:00:00"
visibleIncrement	String	ISO-8601-style string representing the amount by which every element with a visible time in the time picker increases. Set in non Zulu time, without a time zone or date. Example: "T01:00:00" creates text in every 1 hour increment
visibleRange	String	ISO-8601 string representing the range of this TimePicker The TimePicker will only display times in this range Example: "T05:00:00" displays 5 hours of options

Extension Points

Integer compare(/* Object */val1, /* Object */val2)	compare 2 values and return negative number for less than, 0 for equal, positive number for greater than
Boolean isInRange(/* Boolean*/ isFocused)	Need to over-ride with your own validation code in subclasses. Return true if current value is in range.

dijit.form.NumberTextBox
A validating, serializable, range-bound text box. Local conventions are used when interacting with the user. JavaScript Number.toString() is used to serialize data to the server ("." for decimal point, no other notation) See ValidationTextBox and TextBox above for additional attributes and methods.

Extension Points

Integer compare(/* Object */val1, /* Object */val2)	compare 2 values and return negative number for less than, 0 for equal, positive number for greater than
Boolean isInRange(/* Boolean*/ isFocused)	Need to over-ride with your own validation code in subclasses. Return true if current value is in range.

dijit.form.CurrencyTextBox
Like NumberTextBox but formats with currency. Data is serialized as a number only; developer is responsible for conveying currency information. It is recommended to pass ISO-4217 data with all currency transactions. See ValidationTextBox and TextBox above for additional attributes and methods.

Attributes

currency	String	the ISO4217 currency code, a three letter sequence like "USD" See http://en.wikipedia.org/wiki/ISO_4217

Extension Points

Integer compare(/* Object */val1, /* Object */val2)	compare 2 values and return negative number for less than, 0 for equal, positive number for greater than
Boolean isInRange(/* Boolean*/ isFocused)	Need to over-ride with your own validation code in subclasses. Return true if current value is in range.

Sending and Receiving Server Formats

Patterns given as constraints in a DateTextBox or NumberBox only apply to the on-screen value, not the value received or sent to the server. Dojo encourages the use of standard, locale-neutral formats when marshalling data as best practice. In some cases, the receiving

application may have special requirements. A shim on the server can do the necessary translation, but it is also possible to create a custom widget to use a different format. For example when Oracle database processes dates, by default it insists on dd-MMM-yyyy format in English, as in 01-APR-2006. If you wish to send it in this format, you can override the serialize method of DateTextBox. Here's an example:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
    <style type="text/css">
        @import "/dojoroot/dijit/themes/tundra/tundra.css";
        @import "/dojoroot/dojo/dojo.css"
    </style>
    <script type="text/javascript" src="/dojoroot/dojo/dojo.js"
                djConfig="parseOnLoad: true"></script>
   <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.DateTextBox");

       dojo.declare("OracleDateTextBox",[dijit.form.DateTextBox], {
          serialize: function(d, options) {
             return dojo.date.locale.format(d, {selector:'date', datePattern:'dd-MMM-yyyy'}).toLowerCase();
          }
       });
    </script>
</head>
<body class="tundra">
    <form>
    <input dojoType="OracleDateTextBox" name="mydate" value="2006-04-01"/>
    <input type="submit" value="go"/>
    </form>
</body></html>
```

You can also pull the OracleDateTextBox widget into a module and dojo.require it in your pages. Similar customization is possible with numbers, although the default Javascript number representation tends to be less of an issue.

Since Dojo is open source and the widgets are fully customizable, if you really want to use a custom protocol to communicate to and from a server, you can simply override the necessary methods. Here's an example of a DateTextBox subclass that uses a custom date format.

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>DateTextBox subclass Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript"
        src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="isDebug: false, parseOnLoad: false">
    </script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.form.DateTextBox");
        // subclass DateTextBox to allow the initial value to be specified
        // as MM/dd/y instead of yyyy-MM-dd in the markup
        dojo.addOnLoad(function(){
                dojo.declare("altDateTextBox", dijit.form.DateTextBox, {
                        serialize: function(value, constraints){
                                // overrides to send the date to the server with a format of constraints.datePattern
                                // instead of calling dojo.date.stamp.toISOString
                                return dojo.date.locale.format(value, constraints);
                        },
                        postMixInProperties: function(){
                                this.inherited(arguments);
                                this.constraints.datePattern = "MM/dd/y";
                                if(this.srcNodeRef){
                                        // reparse the value attribute using constraints.datePattern
                                        // instead of calling dojo.date.stamp.fromISOString
                                        var item = this.srcNodeRef.attributes.getNamedItem('value');
                                        if(item){
                                                this.value = dojo.date.locale.parse(item.value, this.constraints);
                                        }
                                }
                        }
                });
                dojo.parser.parse();
        });
    </script>
</head>
<body class="tundra">
        <input id="markup" dojoType="altDateTextBox" value="12/31/2007">
        <button onclick="alert('value serialized to ' + dijit.byId('markup').toString());return false">Serialize</button>
</body>
</html>
```

Accessibility

Keyboard

Action	Key	Comments
Move focus to the next widget in the tab order.	Tab	
Move focus to the prior widget in the tab order.	Shift+Tab	
Submit the form.	Enter	
Revert the last entry.	Esc	If the user has not entered data, the Esc key is ignored.

Screen Readers

If an invalid value is entered into a validating Text Box the "state" of the Text box changes, i.e. its background color changes. To accomodate users who are blind, the Text Box's ARIA state is changed to "invalid" so a screen reader can notify the screen reader user. In addition to the "state" change, a pop-up appears. When the pop-up appears screen readers should read the contents of the pop-up. The pop-up text comes from the "invalidMessage" parameter.

Known Issues

Sometimes the popup message supplied by invalidMessage attribute may be unnecessary. For example, omitting a required field already displays an icon when the cursor leaves the field. In these cases you can omit the "invalidMessage" parameter, but keep in mind that good labels and instructions are still necessary for accessibility, i.e. if the invalid popup will not be displayed then there must be clear instructional text indicating the field is required.

UPDATED for 1.0: Window-Eyes 6.1 speaks "read only" for fields that have been marked with the ARIA property invalid=true even though the field is still editable.

Layout

HTML and Layouts

Typically HTML has bottom-up sizing, where a container is as big as it's contents, so that given

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div id="outer">
    <div id="inner1">
        Part 1
    </div>
    <div id="inner2">
        Part 2
    </div>
</div>
```

inner1 is big enough to hold the text "Part 1", inner2 is big enough to hold the text "Part 2", and outer is big enough to hold the divs. And if outer is bigger than the browser's viewport, the browser window displays a scrollbar. The web page you're reading now uses that layout, and unless your monitor is 3 feet tall (in which case, we envy you!) you see the scrollbar on the right.

But for some web pages, you want them to work with the opposite pattern, where you start with a given size, typically the browser viewport, and then partition it into smaller sections. This is the way desktop application look, for example a mail program that has a tree on the left, a list of messages in the upper right, and the message preview on the lower right.

Note that in this scenario, there's no scrollbar on the browser window itself, but if any pane is too small to display all the text it contains then it gets a scrollbar.

[inline:maildemo.png]

Layout like the picture above can be done using tables or fancy CSS (see recent A List Apart article about CSS sizing), but that technique has it's limits... it doesn't allow things like tabs or accordions or split containers where the user can adjust the size of each pane.

Dijit Layout

Dijit has a number of layout widgets which can be combined in a hierarchy to achieve that. Every layout widget contains a list of other layout widgets, except for the "leaf" nodes in the hierarchy, which are typically ContentPanes.

How does this work in practice? You need to think about the application above in a top-down (or outside-in) way:

1. the screen is split into two parts. The top is a toolbar and
2. the bottom is split into a left section and right section
3. the left section has three panes one of which is shown at a time
4. the right section is split into two parts, a list of messages and a preview pane.

Conceptually it's a set of containers like this:

[inline:layoutblock.png]

There are three types of elements in that picture:

1. containers that display all their children side by side
2. containers that display one child at a time
3. leaf nodes containing content

#1 is either LayoutContainer or SplitContainer. A LayoutContainer is used when all but one of the elements is a constant size. (In this case, the toolbar is a constant size and the the bottom section takes the rest of the screen, so we will use a LayoutContainer for that, and SplitContainers for the other parts.

#2 is AccordionContainer, TabContainer, or StackContainer. They all do basically the same thing, but look different.

#3 is typically ContentPane but could be any widget. An important consideration is whether or not the widget's size is adjustable (like a ContentPane) or not (like a Toolbar). See #1 above.

So keeping those rules in mind and picking which widgets to use it will look like:

- LayoutContainer
  - Toolbar
  - Horizontal Split Container
    - Accordion Container
      - ContentPane #1
      - ContentPane #2
      - ContentPane #3
    - Vertical Split Container
      - Content Pane #4
      - Content Pane #5

And then from there it's easy to convert to HTML. Starting from the outside:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.LayoutContainer" id="mainDiv">
    <div dojoType="dijit.Toolbar" layoutAlign="top">...</div>
    <div dojoType="dijit.layout.SplitContainer" orientation="horizontal" layoutAlign="client">
    see below
    </div>
</div>
```

Note that the layoutAlign arguments on the child nodes are actually processed by the parent, but the other arguments are processed by the child. A bit confusing but that's the way it works.

The split container will look like:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.SplitContainer" orientation="horizontal">
    (left part)
    <div dojoType="dijit.layout.AccordionContainer">
        <div dojoType="dijit.layout.AccordionPane" title="Mail">...</div>
        <div dojoType="dijit.layout.AccordionPane" title="News">...</div>
        <div dojoType="dijit.layout.AccordionPane" title="Alerts">...</div>
    </div>
    (right part, see below)
</div>
```

And on the right... since you want to split the screen vertically the SplitContainer would actually contain another SplitContainer:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.SplitContainer" orientation="horizontal">
    <div dojoType="dijit.layout.AccordionContainer">
        <div dojoType="dijit.layout.AccordionPane" title="Mail">...</div>
        <div dojoType="dijit.layout.AccordionPane" title="News">...</div>
        <div dojoType="dijit.layout.AccordionPane" title="Alerts">...</div>
    </div>
    <div dojoType="dijit.layout.SplitContainer" orientation="vertical">
            <div dojoType="dijit.layout.ContentPane" title="Table">...</div>
        <div dojoType="dijit.layout.ContentPane" title="Preview">...</div>
    </div>
```

Tips

Sizing to browser viewport: To make the outermost layout widget size to the browser's viewport, in your page CSS you should have: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
html, body, #mainDiv {
    width: 100%; height: 100%;
```

```
    border: 0; padding: 0; margin: 0;
}
```

where mainDiv is the id of the outermost div.

Note that height=width=100% means different things depending on the browser when you have padding or border, so when using those tags it's best not to have either of those. Put your padding, border, and margin on elements inside the outer layout container.

Restrictions about visibility: none of the layout widgets work if they are inside a hidden element. This is very important and a mistake many people make.

Startup call: when building widgets programmatically, you create the parent first, then add the children, and grandchildren... and finally call startup(). Startup() is called once on the top element in the hierarchy, after the whole hierarchy has been setup and the element inserted.

Accordion Container In this container, panes are pulled up or down like window blinds by clicking the pane title. Only one shows in full at a time.	[inline:accordion_pane.png]
Content Pane the leaves in the hierarchy. Contains arbitrary HTML.	[inline:content_pane.png]
Layout Container Partitions children into top/left/bottom/right/center sections. Useful (for example) to reserve the top 100px of the screen for title and navigation, and then the rest of the viewport for the contents. Note how in this case there would be just one scrollbar on the contents, rather than on the browser window itself.	[inline:layout_container.png]
Split Container Splits the children into sections where user can adjust the size of each section (making one section bigger makes others smaller)	[inline:split_container.png]
Stack Container base class for TabContainer and AccordionContainer but allowing for user defined display to control which pane is shown	[inline:stack_container.png]
Tab Container Like tabbed folders in a desk drawer, you click on a title tab to bring the corresponding pane to the front.	[inline:tab_container.png]

AccordionContainer

Like StackContainer and TabContainer, an AccordionContainer holds a set of panes whose titles are all visible, but only one pane's content is visible at a time. Clicking on a pane title slides the currently-displayed one away, similar to a garage door.

Examples

For this example, we'll show the lazy-loading feature of panes. Lazy loading defers the loading process until the pane is actually displayed. Since Dojo does this with XHR, you can only load panes that reside on the server from which the original content came. We'll use a test pane provided with Dijit. You can download this file from the nightly build at http://svn.dojotoolkit.org/dojo/dijit/trunk/tests/layout/tab1.html. For the example to work verbatim, just place it in the same directory as this file on your web server.

Nunc consequat nisi vitae quam. Suspendisse sed nunc.
Proin ...

More content

- One
- Two
- Three

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Accordion Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.layout.AccordionContainer");
     </script>
</head>
<body class="tundra">
        <div dojoType="dijit.layout.AccordionContainer" duration="200"
                style="margin-right: 30px; width: 400px; height: 300px; overflow: hidden">
                <!-- content inline -->
                <div dojoType="dijit.layout.AccordionPane" selected="true" title="Pane 1">
                    <p >Nunc consequat nisi vitae quam. Suspendisse sed nunc. Proin ...</p >
                </div>

                <!-- loading by indirect reference -->
                <div dojoType="dijit.layout.AccordionPane" title="Pane 2 (lazy load)"
                    href="/dojoroot/dijit/tests/layout/tab1.html" >
                </div>

                <div dojoType="dijit.layout.AccordionPane" title="Pane 3">
                    More content
                    <ul>
                        <li>One</li>
                        <li>Two</li>
                        <li>Three</li>
                    </ul>
                </div>
        </div>
</body>
</html>
```

dijit.layout.AccordionContainer
Holds a set of AccordionPane widgets and displays the title of every pane, but only one pane is visible at a time. Switching between panes is visualized by sliding the other panes up/down.

Attributes		
duration	Integer 250	Amount of time (in ms) it takes for panes to slide

Methods	
addChild(/*Widget*/ child, /*Integer?*/ insertIndex)	Process the given child widget, inserting its dom node as a child of our dom node
back()	**New for 1.0**Select previous page.
closeChild(/*Widget*/ page)	Close the given widget, like clicking the X button
forward()	**New for 1.0**Select next page.
Widget[] getChildren()	returns array of children widgets
Widget getNextSibling()	returns the widget "to the right"
Widget getParent()	returns the parent widget of this widget, assuming the parent implements dijit._Container
Widget getPreviousSibling()	returns the widget "to the left"
Boolean hasChildren()	true if this widget has child widgets

removeChild(/*Widget*/ page)	removes the passed widget instance from this widget but does not destroy it
resize(/* Object */ args)	Explicitly set this widget size (in pixels), and then call layout() to resize contents (and maybe adjust child widgets). Args is of form {w: int, h: int, l: int, t: int}.
selectChild(/*Widget*/ page)	Show the given widget (which must be one of my children)

dijit.layout.AccordionPane
AccordionPane is a ContentPane with a title. Indirect loading with the href property is supported. Nested Dijit layout widgets inside AccordionPane, such as SplitContainer, are not supported at this time.

Accessibility

Keyboard

Action	Key
Navigate to next title	Right arrow
Navigate to previous title	Left arrow
Navigate into page	Tab
Navigate to next page	Ctrl + page down, ctrl + tab (except IE7)
Navigate to previous page	Ctrl + page up

BorderContainer (1.1)

New in 1.1

This widget replaces the functionality of BorderLayout and SplitContainer, providing the ability to place widgets in up to five positions: left (or leading), right (or trailing), top, bottom, and center, with optional splitters for any of the widgets on the sides. There is a choice of two "design" modes, headline or sidebar, where headline has top and bottom extending the entire width of the box, and sidebar lets the side panels extend the full length of the box. (need illustrations here)

ContentPane

A Content Pane is the most basic layout tile. Conceptually, it's like the content boxes in portals like MyYahoo. A content pane resembles an iframe, but contains extra design features, fits in with the current theme, and renders widgets properly.

You can use content panes by themselves, but usually you will place content panes inside of a layout container. For example, in a tabbed layout, content pane tags surround each tab of information.

Examples

Simple content panes have their content inside the tags Most of the time, these simple panes are used inside layout containers, so the examples in LayoutContainer, StackContainer and TabContainer illustrate their use.

Linked content panes, those with their content in a separate URL, are useful even without containers. To illustrate, we'll use the example data store countries.txt, which you can download below (after this page text, and right above the navigation bars). For the example to work verbatim, just place it in the same directory as these files on your web server. If the following is linked_content_pane.html:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!-- Note, you don't have to include Dojo scripts, CSS, or other things here ... -->
<div dojoType="dojo.data.ItemFileReadStore" jsId="continentStore"
        url="countries.txt"></div>
<div dojoType="dijit.Tree" id="mytree" store="continentStore" query="{type:'continent'}"
        labelAttr="name" typeAttr="type"></div>
```

You can include it within a page like this. Because the tree may take a long time to load, we display a loading message for the user.

This text be replaced with the tree
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Linked Content Pane Demo</title>
    <style type="text/css">
```

```
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
        <script type="text/javascript">
        dojo.require("dojo.parser");
                dojo.require("dijit.layout.ContentPane");
                dojo.require("dijit.Tree");
                dojo.require("dojo.data.ItemFileReadStore");
     </script>
</head>
<body class="tundra">
        <div preload="true" dojoType="dijit.layout.ContentPane" href="linked_content_pane.html">
                This text be replaced with the tree
        </div>
</body></html>
```

Note here that the dojo.require for the tree is placed in the *calling* html. SCRIPT tags of type "text/Javascript" in the included html file are ignored. **However** SCRIPT tags with type dojo/method and dojo/connect work fine. See Understanding the Parser for details.

Like all Dijit components, you can create a ContentPane dynamically. The following example adds a new ContentPane to an existing Tab Container.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Content Pane Programmatic Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
        <script type="text/javascript">
                dojo.require("dijit.layout.ContentPane");
                dojo.require("dijit.layout.TabContainer");
                dojo.require("dijit.Tree");
                dojo.require("dojo.data.ItemFileReadStore");

                dojo.addOnLoad(function() {
                        // create a ContentPane
                        var newChild =
                                new dijit.layout.ContentPane({
                                                href:'linked_content_pane.html',
                                                title:'newChild',
                                                refreshOnShow:true
                                        },
                                        dojo.doc.createElement('div')
                                );

                        // find my tabContainer and add our new ContentPane
                        dijit.byId('myTabContainer').addChild(newChild);

                        // find parent Container and subscribe to selectChild event
                        // without it refreshOnShow and href load wont work.
                        newChild.startup();
                });
     </script>
</head>
<body class="tundra">
        <div id="myTabContainer" dojoType="dijit.layout.TabContainer"
                 style="width:500px;height:300px"></div>
</body></html>
```

NOTE! You must include a sourcenode when creating a ContentPane. You must call .startup() after you have addChild() it to your parent

dijit.layout.ContentPane
A widget that acts as an all-purpose sizable pane without navigation, and includes a ajax interface

Attributes

errorMessage	String Locale dep.	Message that shows if an error occurs
extractContent	Boolean false	Extract visible content from inside of <body> .... </body>
href	String	The href of the content that displays now. Set this at construction if you want to load data externally when the pane is shown. (Set preload=true to load it immediately.) Changing href after creation does not have any effect; see setHref();
isLoaded	Boolean false	Tells loading status see onLoad|onUnload for event hooks

loadingMessage	String Locale dep.	Message that shows while downloading
parseOnLoad	Boolean true	parse content and create the widgets, if any
preload	Boolean false	Force load of data even if pane is hidden.
preventCache	Boolean false	Cache content retreived externally
refreshOnShow	Boolean false	Refresh (re-download) content when pane goes from hidden to shown
Methods		
cancel()	Cancels a inflight download of content	
refresh()	Force a refresh (re-download) of content, be sure to turn off cache we return result of _prepareLoad here to avoid code dup. in dojox.layout.ContentPane	
resize(/* String */size	Explicitly set this widget size (in pixels), and then call layout() to resize contents (and maybe adjust child widgets)	
setContent(/*String|DomNode|Nodelist*/data)	Replaces old content with data content, include style classes from old content	
setHref(/*String|Uri*/ href)	Reset the (external defined) content of this pane and replace with new url Note: It delays the download until widget is shown if preload is false	
Extension Points		
onContentError(/*Error*/ error)	called on DOM faults, require fault etc in content default is to display errormessage inside pane	
onDownloadEnd()	called when download is finished	
onDownloadError(/*Error*/ error)	Called when download error occurs, default is to display errormessage inside pane. Overide function to change that. The string returned by this function will be the html that tells the user a error happend	
onDownloadStart()	called before download starts the string returned by this function will be the html that tells the user we are loading something override with your own function if you want to change text	
onLoad(/* Event */e)	Event hook, is called after everything is loaded and widgetified	
onUnload(/* Event */e)	Event hook, is called before old content is cleared	

Accessibility - updated for 1.0

General Issues

The developer is responsible for determining if the ContentPane should be in the tab order of the page or not. If the ContentPane is **not** likely to have a focusable item within the contents, the developer may want to add tabindex=""0" onto the ContentPane element. This will put the ContentPane into the tab order so if someone is using the tab key to navigate through the elements on the page, the ContentPane itself will get focus. Having focus go to the ContentPane itself can be helpful for users of assistive technology to be able to navigate to an area that may not have any focusable elements within it such as a preview pane for mail messages or a page footer containing important information.

LayoutContainer

Similar to a layout pane in Java AWT and Delphi, a LayoutContainer is a box with a specified size (like style="width: 500px; height: 500px;"), that contains children widgets marked with "layoutAlign" of "left", "right", "bottom", "top", and "client". It takes it's children marked as left/top/bottom/right, and lays them out along the edges of the box, and then it takes the child marked "client" and puts it into the remaining space in the middle.

Left/right positioning is similar to CSS's "float: left" and "float: right", and top/bottom positioning would be similar to "float: top" and "float: bottom", if there were such CSS.

Note that there can only be one client element, but there can be multiple left, right, top, or bottom elements.

Examples

A Layout Container easily formats a table of contents:

The Dojo Book
Table of Contents

Introduction

1. Dojo: What is It?
2. History
3. What Dojo Gives You

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Layout Container Demo 2</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <style>
        /* NOTE: for a full screen layout, must set body size equal to the viewport. */
        html, body { height: 100%; width: 100%; margin: 0; padding: 0; }
    </style>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.layout.ContentPane");
        dojo.require("dijit.layout.LayoutContainer");
     </script>
</head>
<body class="tundra">
<div dojoType="dijit.layout.LayoutContainer" style="width: 100%; height: 100%; padding: 0; margin: 0; border: 0;">
   <div dojoType="dijit.layout.ContentPane" layoutAlign="top" style="background-color:red">
        The Dojo Book
   </div>
   <div dojoType="dijit.layout.ContentPane" layoutAlign="left"
        style="background-color:lightblue;width: 120px;">
        Table of Contents
    </div>
    <div dojoType="dijit.layout.ContentPane" layoutAlign="client"
        style="background-color:yellow">
            <blockquote><a href="../node/717">Introduction</a>
                <ol>
                    <li><a href="../node/718">Dojo: What is It?</a></li>
                    <li><a href="../node/719">History</a></li>
                    <li><a href="../node/733">What Dojo Gives You</a></li>
                </ol>
                </blockquote>
    </div>
</div>
</body></html>
```

To change the drawing order:

The LayoutContainer goes through the children as specified, and each child is laid out into the "remaining space", where "remaining space" is initially the content area of this widget, but is reduced to a smaller rectangle each time a child is added. So, changing the order of the children will change how they are laid out.

Table of Contents
The Dojo Book

Introduction

1. Dojo: What is It?
2. History
3. What Dojo Gives You

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Layout Container Demo 2</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
     <style>
        /* NOTE: for a full screen layout, must set body size equal to the viewport. */
        html, body { height: 100%; width: 100%; margin: 0; padding: 0; }
    </style>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.layout.ContentPane");
       dojo.require("dijit.layout.LayoutContainer");
     </script>
</head>
```

```
<body class="tundra">
<div dojoType="dijit.layout.LayoutContainer" style="width: 100%; height: 100%; padding: 0; margin: 0; border: 0;">
    <div dojoType="dijit.layout.ContentPane" layoutAlign="left"
         style="background-color:lightblue;width: 120px;">
         Table of Contents
     </div>
    <div dojoType="dijit.layout.ContentPane" layoutAlign="top" style="background-color:red">
         The Dojo Book
    </div>
     <div dojoType="dijit.layout.ContentPane" layoutAlign="client"
         style="background-color:yellow">
             <blockquote><a href="../node/717">Introduction</a>
                 <ol>
                     <li><a href="../node/718">Dojo: What is It?</a></li>
                     <li><a href="../node/719">History</a></li>
                     <li><a href="../node/733">What Dojo Gives You</a></li>
                 </ol>
                 </blockquote>
    </div>
</div>
</body></html>
```

dijit.layout.LayoutContainer
Provides Delphi-style panel layout semantics.

Methods

addChild(/*Widget*/ child, /*Integer?*/ insertIndex)	Process the given child widget, inserting its dom node as a child of our dom node
Widget[] getChildren()	returns array of children widgets
Widget getParent()	returns the parent widget of this widget, assuming the parent implements dijit._Container
removeChild(/*Widget*/ page)	removes the passed widget instance from this widget but does not destroy it
resize(/* Object */ args)	Explicitly set this widget size (in pixels), and then call layout() to resize contents (and maybe adjust child widgets). Args is of form {w: int, h: int, l: int, t: int}.

SplitContainer

Contains multiple children widgets, all of which are displayed side by side (either horizontally or vertically); there's a bar between each of the children, and you can adjust the relative size of each child by dragging the bars. You must specify a size (width and height) for the SplitContainer.

Example

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>SplitContainer Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.layout.SplitContainer");
       dojo.require("dijit.layout.ContentPane");
     </script>
</head>
<body class="tundra">
        <div dojoType="dijit.layout.SplitContainer"
                orientation="horizontal"
                sizerWidth="7"
                activeSizing="true"
                style="border: 1px solid #bfbfbf; width: 400px; height: 300px;">
                <div dojoType="dijit.layout.ContentPane" sizeMin="20" sizeShare="20">
                    <b>The Dojo Toolkit</b>

                        Dojo is an Open Source DHTML toolkit written in JavaScript. It
                        builds on several contributed code bases (nWidgets, Burstlib, f(m)),
                        which is
                        why we refer to it sometimes as a "unified" toolkit. Dojo aims to
                        solve some long-standing historical problems with DHTML which
                        prevented mass adoption of dynamic web application development.

                </div>
                <div dojoType="dijit.layout.ContentPane" sizeMin="50" sizeShare="50">
                    <b>Swedish Chef Translation</b>

                        Duju is un Oopee Suoorce-a DHTML tuulkeet vreettee in JefeScreept.
                        Um de hur de hur de hur. It booeelds oon seferel cuntreebooted
                        cude-a beses (nVeedgets, Boorstleeb, f(m)),
                        vheech is vhy ve-a reffer tu it sumeteemes es
                        a "uneeffied" tuulkeet. Um de hur de hur de hur.
                        Duju eeems tu sulfe-a sume-a lung-stundeeng heesturicel
                        prublems veet DHTML vheech prefented mess edupshun
```

```
                    ooff dynemeec veb eppleeceshun defelupment. Um de hur de
                    hur de hur.

                </div>
        </div>
</body></html>
```

dijit.layout.SplitContainer
Container with resizable dividers.

Attributes

activeSizing	Boolean false	If true, the children's size changes as you drag the bar; otherwise, the sizes don't change until you drop the bar (by mouse-up)
orientation	String horizontal	either 'horizontal' or vertical; indicates whether the children are arranged side-by-side or up/down.
persist	Boolean true	Save splitter positions in a cookie
sizerWidth	Integer 7	Size in pixels of the bar between each child

Methods

addChild(/*Widget*/ child, /*Integer?*/ insertIndex)	Process the given child widget, inserting its dom node as a child of our dom node
Widget[] getChildren()	returns array of children widgets
Widget getParent()	returns the parent widget of this widget, assuming the parent implements dijit._Container
removeChild(/*Widget*/ page)	removes the passed widget instance from this widget but does not destroy it
resize(/* Object */ args)	Explicitly set this widget size (in pixels), and then call layout() to resize contents (and maybe adjust child widgets). Args is of form {w: int, h: int, l: int, t: int}.

Sizing

sizeShare

Setting the **sizeShare** attribute on any child widgets of a SplitContainer sets the initial relative size of that widget, either its height or width depending on the layout of the SplitContainer. The value of **sizeShare** is not a percentage or a pixel measure. Its value is relative to other child widgets of the same SplitContainer. So, for example, given four child widgets and each having a **sizeShare** attribute of 25, would evenly divide the SplitContainer into four parts. However, giving each a **sizeShare** attribute of 10 or 1 would achieve the same result, as the values are computed relative to each other - they do not have to add up to 100.

sizeMin

Setting the **sizeMin** attribute on any child widget of a SplitContainer defines the smallest size, in pixels, that the child widget will be changed to. The value is specified as an integer, without "px" appended to it.

Accessibility (added for 1.0)

Keyboard

In Dojo 1.0 there is no keyboard mechanism to resize the split container. In Firefox the content panes within a split container will be in the tab order (this is default FF behavior) so if the data is not visible, the user can use the arrow keys to scroll the data into view. This allows keyboard access to all of the data and thus accessibility requirements are met by default in Firefox. IE does not include the content pane in the tab order. If there is a chance that all of the data will not being visible within a pane of a split container, a tabindex=0 should be included in the markup of the inner content pane to ensure keyboard accessibility in both Firefox and IE.

StackContainer

A container that has multiple children, but shows only one child at a time (like looking at the pages in a book one by one). This container is good for wizards, slide shows, and long lists or text blocks.

Examples

Here's a freely pageable document.

< >

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License. "The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Stack Container Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.require("dijit.layout.ContentPane");
        dojo.require("dijit.layout.StackContainer");
        dojo.require("dijit.form.Button");
     </script>
</head>
<body class="tundra">
        <button id="previous" onClick="dijit.byId('mainTabContainer').back()"><</button>
        <button id="next" onClick="dijit.byId('mainTabContainer').forward()">></button>
        <div id="mainTabContainer" dojoType="dijit.layout.StackContainer"
             style="width: 90%; border:1px solid #9b9b9b; height: 20em;
                    margin: 0.5em 0 0.5em 0; padding: 0.5em;">
        <p id="Page1" dojoType="dijit.layout.ContentPane" label="Intro">
                This version of the GNU Lesser General Public License incorporates
                the terms and conditions of version 3 of the GNU General Public
                License, supplemented by the additional permissions listed below.

        <p id="Page2" dojoType="dijit.layout.ContentPane">
                As used herein, "this License" refers to version 3 of the GNU Lesser
                General Public License, and the "GNU GPL" refers to version 3 of the GNU
                General Public License. "The Library" refers to a covered work governed by
                this License, other than an Application or a Combined Work as defined below.

        <p id="Page3" dojoType="dijit.layout.ContentPane" >
                You may convey a covered work under sections 3 and 4 of this License
                without being bound by section 3 of the GNU GPL.

   </div>
</body></html>
```

Indication of the current child

As standard, there are no styles on the buttons associated with the StackContainer to indicate which child is currently being shown. However, Dijit adds a "dijitToggleButtonChecked" class to the button for the child being shown and we can use this class to provide styling ourselves. For example, we could use the following rules to highlight the button with a white background and, in Windows high contrast mode, a dashed border: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
.dijitStackController .dijitToggleButtonChecked button {
    background-color: white;
    background-image: none;
}
.dijit_a11y .dijitStackController .dijitToggleButtonChecked button {
    border-style: dashed !important;
}
```

dijit.layout.StackContainer
A container of panes, one of which is always on top. Developer must provide navigation controls.

Attributes

doLayout	Boolean true	if true, change the size of my currently displayed child to match my size

Methods

addChild(/*Widget*/ child, /*Integer?*/ insertIndex)	Process the given child widget, inserting its dom node as a child of our dom node
back()	**New for 1.0**Select previous page.
forward()	**New for 1.0**Select next page.

Widget[] getChildren()	returns array of children widgets
Widget getNextSibling()	returns the widget "to the right"
Widget getParent()	returns the parent widget of this widget, assuming the parent implements dijit._Container
Widget getPreviousSibling()	returns the widget "to the left"
removeChild(/*Widget*/ page)	removes the passed widget instance from this widget but does not destroy it
resize(/* Object */ args)	Explicitly set this widget size (in pixels), and then call layout() to resize contents (and maybe adjust child widgets). Args is of form {w: int, h: int, l: int, t: int}.
selectChild(/*Widget*/ page)	Show the given widget (which must be one of my children)

Accessibility

Keyboard

Action	Key
Navigate to next tab button	Right arrow
Navigate to previous tab button	Left arrow
Navigate into page	Tab
Navigate to next page	Ctrl + page down, ctrl + tab (except IE7)
Navigate to previous page	Ctrl + page up
~~Delete a tab~~	~~Delete, ctrl + w~~ (updated for 1.0 - delete is not supported in stack container)

TabContainer

A TabContainer is a container that has multiple panes, but shows only one pane at a time. There are a set of tabs corresponding to each pane, where each tab has the title (aka label) of the pane, and optionally a close button.

Examples

Here's a Grimm set of tabs, indeed.

Once upon a time there was a dear little girl who was loved by every one who looked at her, but most of all by her grandmother, and there was nothing that she would not have given to the child.
Hard by a great forest dwelt a poor wood-cutter with his wife and his two children. The boy was called Hansel and the girl Gretel. He had little to bite and to break, and once when great dearth fell on the land, he could no longer

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>TabContainer Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
           dojo.require("dijit.layout.ContentPane");
           dojo.require("dijit.layout.TabContainer");
           dojo.require("dijit.form.Button");
     </script>
</head>
<body class="tundra">
   <div id="mainTabContainer" dojoType="dijit.layout.TabContainer"
        style="width:500px;height:100px">
      <div id="LittleRed" dojoType="dijit.layout.ContentPane" title="Little Red Cap">
      Once upon a time there was a dear little girl who was loved by
      every one who looked at her, but most of all by her grandmother,
      and there was nothing that she would not have given to the child.
      </div>
      <div id="HanselGretel" dojoType="dijit.layout.ContentPane"
           title="Hansel and Gretel" closable="true" selected="true">
      Hard by a great forest dwelt a poor wood-cutter with his wife
      and his two children. The boy was called Hansel and the girl Gretel.
      He had little to bite and to break, and once when great dearth fell
```

```
      on the land, he could no longer procure even daily bread.
      </div>
      <div id="GreenTwigs" dojoType="dijit.layout.ContentPane"
            title="The Three Green Twigs">
      There was once upon a time a hermit who lived in a forest at the foot
      of a mountain, and passed his time in prayer and good works,
      and every evening he carried, to the glory of God, two pails of water
      up the mountain.
      </div>
</div>
</body></html>
```

dijit.layout.TabContainer
A TabContainer is a container that has multiple panes, but shows only one pane at a time. There are a set of tabs corresponding to each pane, where each tab has the title (aka title) of the pane, and optionally a close button. Publishes topics <widgetId>-addChild, <widgetId>-removeChild, and <widgetId>-selectChild (where <widgetId> is the id of the TabContainer itself.

Attributes		
tabPosition	String	Defines where tabs go relative to tab content. "top", "bottom", "left-h", "right-h" override setting in StackContainer

Methods	
addChild(/*Widget*/ child, /*Integer?*/ insertIndex)	Process the given child widget, inserting its dom node as a child of our dom node
back()	**New for 1.0**Select previous page.
closeChild(/*Widget*/ page)	Close the given widget, like clicking the X button
forward()	**New for 1.0**Select next page.
Widget[] getChildren()	returns array of children widgets
Widget getNextSibling()	returns the widget "to the right"
Widget getParent()	returns the parent widget of this widget, assuming the parent implements dijit._Container
Widget getPreviousSibling()	returns the widget "to the left"
Boolean hasChildren()	true if this widget has child widgets
removeChild(/*Widget*/ page)	removes the passed widget instance from this widget but does not destroy it
resize(/* Object */ args)	Explicitly set this widget size (in pixels), and then call layout() to resize contents (and maybe adjust child widgets). Args is of form {w: int, h: int, l: int, t: int}.
selectChild(/*Widget*/ page)	Show the given widget (which must be one of my children)

Accessibility

Keyboard

Action	Key
Navigate to next tab button	Right or down arrow
Navigate to previous tab button	Left or up arrow
Navigate into page	Tab
Navigate to next page	Ctrl + page down, ctrl + tab (except IE7)
Navigate to previous page	Ctrl + page up
Delete a tab	With focus on the tab title of the tab to delete, press Delete or ctrl + w. Note that the tab must have been created with deletion enabled via the closable attribute.

Command Control

Button, ComboButton, DropDownButton	[inline:button.png]
Menu	[inline:menu.png]
Toolbar	[inline:toolbar.png]

Button, ComboButton, DropDownButton

Buttons are very popular UI elements, and Dijit has a bunch of them, from a simple click and go button to a DropDownButton, and much more.

Examples

Creating a Click and Go Button

A normal click-and-go button is simple to make and can be enhanced by using image icons, for example. The following markup creates a normal click-and-go button that calls a single JavaScript function:

Hi I am the One !!

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Click and Go Button Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Button");
        function call_function() {
           console.debug("Button was clicked.");
        }
     </script>
</head>
<body class="tundra">
        <button dojoType="dijit.form.Button" onclick="call_function">
                Hi I am the One !!
        </button>
</body></html>
```

You can set text on a button by using the label attribute. To display an image on the button, you use the iconClass attribute, then set up a CSS class which uses that icon as a background-image. See Toolbar for a description of sprites, a technique which can make multiple icons load faster.

Programatically Creating a Button

The following code creates a button programatically:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Click and Go Button Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/0.9.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
          dojo.require("dijit.form.Button");
          function call_function() {
             console.debug("Button was clicked.");
          }

               function create_button()
               {
                   var params = {
                       label: "Hi I am the Second one",
                       // Note here, when creating programmatically, this is a function, not a string
                       onClick: call_function
                   };

                   var button_dynamic = new dijit.form.Button(
                       params,dojo.byId("button-placeholder")
                   );
               }
      dojo.addOnLoad(create_button);
     </script>
</head>
<body class="tundra">
    <div id="button-placeholder"> </div>
</body></html>
```

The created button will take the place of the provided DIV node; in the example, this is the DIV with the ID "button-placeholder".

Alternatively, you can use the DOM's createElement() call to create the DIV programmatically, then pass it to the dijit.form.Button constructor.

Creating DropDownButtons

There are, of course, fancier buttons that can be used to make a Web application more like a desktop application, thus providing the power of web 2.0 UI. DropDownButtons provide a drop-down menu when clicked, which is rendered using the dijit.Menu widget of the dijit system and follows the Menu rules. A PopupMenuItem always has two child nodes: a tag with the displayed label (usually in a SPAN tag), and a widget to be popped up, typically a dijit.Menu widget. For example:

Edit

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>DropDownButton Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
        dojo.require("dijit.form.Button");
       dojo.require("dijit.Menu");
        function call_function(choice) {
              console.debug(choice+" was clicked.");
        }
     </script>
</head>
<body class="tundra">
        <div dojoType="dijit.form.DropDownButton">
            <span>Edit</span>
            <div dojoType="dijit.Menu" id="Edit">
                <div dojoType="dijit.MenuItem" label="Copy"
                        onclick="call_function('copy');"></div>
                <div dojoType="dijit.MenuItem" label="Cut"
                        onclick="call_function('cut');"></div>
                <div dojoType="dijit.MenuItem" label="Paste"
                        onclick="call_function('paste');"></div>
            </div>
        </div>
</body></html>
```

In the preceding example, you need to require dijit.Menu alone to source dijit.Menu and dijit.MenuItem. DropDownButtons are used in Web applications like the rich text editor where you need a menu.

Creating ComboButtons

ComboButtons are one step ahead of DropDownButtons as they can be clicked and also offer present menu options like a DropDownButton. The following example shows a ComboButton being created:

Save

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ComboButton Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Button");
       dojo.require("dijit.Menu");
           function save_function() {
              console.debug("Save was clicked.");
           }
           function save_as_function(choice) {
              console.debug("Save As was clicked. Dialog box should go here.");
           }
     </script>
</head>
<body class="tundra">
```

```
    <div dojoType="dijit.form.ComboButton" onclick="save_function">
    <span>Save</span>
            <div dojoType="dijit.Menu" id="saveMenu" toggle="fade" style="display: none;">
                <div dojoType="dijit.MenuItem"
                     iconClass="dijitEditorIcon dijitEditorIconSave" onclick="save_function">
                    Save
                </div>
                <div dojoType="dijit.MenuItem" onclick="save_as_function">
                    Save As
                </div>
            </div>
        </div>
</body></html>
```

The ComboButton class is declared in the Button.js file, so you only need to require the Button and Menu Classes.

dijit.form.Button, dijit.form.DropDownButton, dijit.form.ComboButton,
Basically the same thing as a normal HTML button, but with special styling.

Attributes		
iconClass	String	class to apply to div in button to make it display an icon
label	String	text to display in button. *Use setLabel() to change after creation time.*
optionsTitle	String	text that describes the options menu (accessibility)
showLabel	Boolean	whether or not to display the text label in button
Methods		
addChild	Process the given child widget, inserting it's dom node as a child of our dom node	
getChildren	returns array of children widgets	
setLabel	reset the label (text) of the button; takes an HTML string	
hasChildren	returns true if widget has children	
removeChild	removes the passed widget instance from this widget but does not destroy it	
Extension Points		
onClick	callback for when button is clicked; user can override this function for some reason type=submit buttons don't automatically submit the form; do it manually	

Accessibility

Keyboard

Action	Key
Navigate to a button	tab - all buttons are in the tab order
Activate the button	enter or space key
Close an open drop down	escape key - focus returns to button
With drop down open, navigate to the next element on page	tab will close drop down and set focus back to the button, tab again to navigate to next element
Navigate to a checkbox	tab - all checkboxes are in the tab order
toggle a checkbox	space
Navigate to a radio button or group of radio buttons	tab - see below for browser differences
select a radio in a group	tab to the radio group, up down arrow to the desired radio item

Checkboxes and Radio buttons are implemented using the standard input type=checkbox and type=radio elements respectively. CSS is used to overlay the unique theme over the actual input elements. Thus, the keyboard behavior of checkboxes and radio buttons mimics the behavior in the browser.

Radio buttons in Firefox 2: When radio buttons are grouped the selected radio button is in the tab order. If no radio is selected, each radio button in the group is in the tab order until one of the radios is selected by navigating to it using an arrow key or navigating to it using the tab key and pressing space.

Radio buttons in Internet Explorer 6 & 7: When radio buttons are grouped the selected radio button is in the tab order. If no radio is selected only one radio in the group is in the tab order. The first radio in the group is in the tab order when navigating forward to the radio group. The last radio in the group is in the tab order when navigating backward to the radio group. Once focus is in the radio group, use the arrow keys to select one of the radio buttons.

High Contrast Mode

All buttons should include a label parameter with text for the button even if the showLabel parameter is set to false. The label parameter is used to identify the button in high contrast mode when the icon for the button will no longer be displayed and is also used to identify the button to a screen reader.

Screen Reader

In order to identify the button description to the screen reader, all buttons should include a label parameter even if the showLabel parameter is set to false.

All Combo Buttons should include a optionsTitle parameter to identify the function of the drop down button. The optionsTitle parameter is used by the screen reader to speak the information about the drop down portion of the button. Note that the Window-Eyes screen reader will speak "question" and then the optionsTitle text when the drop down portion of the Combo button receives focus. The "question" is spoken because Window-Eyes does not recognize the html entity character that is used to provide the visual drop down arrow in the button.

Updates for 1.0

Accessibility - Keyboard

When navigating to a ComboButton using the keyboard the focus is not visible using Firefox 2. The current versions of Firefox 3, however, do show a visible focus rectangle for the pieces of the ComboButton so the focus problem in Firefox 2 was not addressed within the dijit button code.

Menu

The Menu widget models a context menu, otherwise known as a right-click or popup menu, and they also appear in ComboButton and DropDownButton widgets.

MenuItem widgets are the actual items in the menu. The PopupMenuItem is like a MenuItem, but displays a submenu or other widget to the right . A PopupMenuItem always has two child nodes: a tag with the displayed label (usually in a SPAN tag), and a widget to be popped up, typically a dijit.Menu widget.

Example

Right click anywhere in Grey Area

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Menu Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.Menu");
     </script>
</head>
<body class="tundra">
        <style>
        .myIcon {
           /*  Note: Drupal may add an anchor tag here.  Don't include it in your code */
           background-image:
              url(http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/images/checkmark.gif);
           background-position: -16px 0px;
           width: 16px;
           height: 16px;
        }
        </style>

        <div dojoType="dijit.Menu" id="submenu1"
              contextMenuForWindow="true" style="display: none;">
            <div dojoType="dijit.MenuItem" iconClass="myIcon"
                 onClick="alert('Hello world');">Enabled Item</div>
            <div dojoType="dijit.PopupMenuItem" id="submenu2">
                <span>Submenu</span>
                <div dojoType="dijit.Menu">
                        <div dojoType="dijit.MenuItem"
                             onClick="alert('Submenu 1!')">Submenu Item One</div>
                        <div dojoType="dijit.MenuItem"
                             onClick="alert('Submenu 2!')">Submenu Item Two</div>
                </div>
            </div>
        </div>
</body></html>
```

dijit.Menu
Define a drop down or context (right-click) menu.

Attributes

contextMenuForWindow	Boolean false	if true, right clicking anywhere on the window will cause this context menu to open; if false, must specify targetNodeIds

popupDelay	Integer 500	number of milliseconds before hovering (without clicking) causes the popup to automatically open
targetNodeIds	String[]	Array of dom node ids of nodes to attach to. Fill this with nodeIds upon widget creation and it becomes context menu for those nodes.
Methods		
Methods		
addChild(/*Widget*/ child, /*Integer?*/ insertIndex)	Process the given child widget, inserting its dom node as a child of our dom node	
bindDomNode(/*String|DomNode*/ node)	attach menu to given node	
Widget[] getChildren()	returns array of children widgets	
removeChild(/*Widget*/ page)	removes the passed widget instance from this widget but does not destroy it	
unBindDomNode(/*String|DomNode*/ nodeName)	detach menu from given node	
Extension Points		
onClose()	callback when this menu is closed	
onOpen(/*Event*/ e)	Open menu relative to the mouse	

dijit.MenuItem
One menu item, usable in a Menu

Attributes		
disabled	Boolean false	if true, the menu item is disabled if false, the menu item is enabled. *Use setDisabled() to change after creation time.*
iconClass	String	class to apply to div in button to make it display an icon
label	String	Menu text. Typically specified as innerHTML when declaring a Menu programmatically.
Methods		
setDisabled(/*Boolean*/ value)	enable or disable this menu item	
Extension Points		
getParent()	returns the parent widget of this widget, assuming the parent implements dijit._Container	
onClick()	User defined function to handle clicks	

dijit.MenuSeparator
A line between two menu items

Accessibility

Keyboard

Action	Key
Open a context menu	On Windows: shift-f10 or the Windows context menu key On Firefox on the Macintosh: ctrl-space
Navigate menu items	Up and down arrow keys
Activate a menu item	Spacebar or enter
Open a submenu	Spacebar, enter, or right arrow
Close a context menu or submenu	Esc or left arrow
Close a context menu and all open submenus	Tab

Known Issues

When reading a menu item on Firefox 2, JAWS 8 may say "submenu" for an item that does not have a submenu. This will be fixed in Firefox 3.

Toolbar

Just as dijit.Menu is a container for dijit.MenuItem's, so dijit.Toolbar is a container for buttons. Any button-based Dijit component can be placed on the toolbar, including ComboButtons and DropdownButtons.

Example

In this example, we borrow some of the toolbar buttons from the Editor

Cut
Copy
Paste
Bold

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<html>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<head>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Button");
       dojo.require("dijit.Toolbar");
       function bold() { console.debug("Bold!"); }
       function cut() { console.debug("Cut!"); }
       function copy() { console.debug("Copy!"); }
       function paste() { console.debug("Paste!"); }
       dojo.addOnLoad(function() {
           dojo.connect(dojo.byId("toolbar1.bold"),"onclick",bold);
           dojo.connect(dojo.byId("toolbar1.cut"),"onclick",cut);
           dojo.connect(dojo.byId("toolbar1.copy"),"onclick",copy);
           dojo.connect(dojo.byId("toolbar1.paste"),"onclick",paste);
       });
     </script>
</head>
<body class="tundra">
    <!-- Tags end on line afterwards to eliminate any whitespace -->
    <div id="toolbar1" dojoType="dijit.Toolbar"
        ><div dojoType="dijit.form.Button" id="toolbar1.cut" iconClass="dijitEditorIcon dijitEditorIconCut"
            showLabel="false">Cut</div
        ><div dojoType="dijit.form.Button" id="toolbar1.copy" iconClass="dijitEditorIcon dijitEditorIconCopy"
            showLabel="false">Copy</div
        ><div dojoType="dijit.form.Button" id="toolbar1.paste" iconClass="dijitEditorIcon dijitEditorIconPaste"
            showLabel="false">Paste</div
        ><!-- The following adds a line between toolbar sections
            --><span dojoType="dijit.ToolbarSeparator"></span
         ><div dojoType="dijit.form.ToggleButton" id="toolbar1.bold"
            iconClass="dijitEditorIcon dijitEditorIconBold" showLabel="false">Bold</div>
   </div>
</body></html>
```

Where do the icons come from? The theme defines one large image with all the buttons. Tundra's editor button image looks like the following:

[inline:editor.gif]

The particular icon is selected using the attribute "iconClass". The Cut button with class "dijitEditorIconCut" has the following definition in Tundra.css:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}

```
.tundra .dijitEditorIcon
/* All of the selectors for the icons go here */
{
        background-image: url('images/editor.gif'); /* mega-image */
        background-repeat: no-repeat;
        width: 18px;
        height: 18px;
        text-align: center;
}
.tundra .dijitEditorIconCut { background-position: -108px; }
```

The Cut icon starts 108 px from the right edge, and measures 18px by 18px. 108 equals 6 * 18, so it's the 6th image from the right. You can define your own buttons by setting up CSS selectors using code similar to the follwing, and wire up the iconClass.

dijit.Toolbar
Toolbar, which can be filled with ComboButton and DropDownButton instances

Methods

Methods

addChild(/*Widget*/ child, /*Integer?*/ insertIndex)	Process the given child widget, inserting its dom node as a child of our dom node
Widget[] getChildren()	returns array of children widgets

removeChild(/*Widget*/ page)	removes the passed widget instance from this widget but does not destroy it

Accessibility (applies to version 1.0 of Toolbar)

Keyboard

Action	Key
Move focus between widgets in the toolbar	Left and right arrow keys

Known Issues (updatd for 1.0 - toggle buttons did not display in high contrst mode in 0.9)

In hign contrast mode when a toggle button is checked an html entity charcter (?) is displayed since the CSS background image icon for the checked state is no longer visible. When the toggle button is part of a toolbar the checkmark character does not display properly in IE6. In IE6 with high contrast mode turned on, a checked toggle button in a toolbar displays as two vertical bars rather than the checkmark character.

User Assistance and Feedback

Dialog	[inline:dialog_box.png]
TooltipDialog	[inline:tooltipDialog.png]
ProgressBar	[inline:progress_bar.png]
TitlePane Click arrow to expand/contract	[inline:title_pane.png]
Tooltip	[inline:tooltip.png]

ProgressBar

A ProgressBar gives dynamic feedback on the progress of a long-running operation. The progress can be updated by JavaScript function calls. This method works best for long-running JavaScript operations, or a series of JavaScript XHR calls to the server.

Examples

This progress meter is updated using JavaScript

Go!

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Progress Bar Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
          djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dijit.ProgressBar");
        dojo.require("dojo.parser");

        function download(){
            // Split up bar into 7% segments
            numParts = Math.floor(100/7);
            jsProgress.update({ maximum: numParts, progress:0 });
            for (var i=0; i<=numParts; i++){
                // This plays update({progress:0}) at 1nn milliseconds,
                // update({progress:1}) at 2nn milliseconds, etc.
                setTimeout(
                   "jsProgress.update({ progress: " + i + " })",
                   (i+1)*100 + Math.floor(Math.random()*100)
                );
            }
        }
     </script>
</head>
<body class="tundra">
    <div dojoType="dijit.ProgressBar" style="width:300px"
         jsId="jsProgress" id="downloadProgress"></div>
    <input type="button" value="Go!" onclick="download();" />
</body></html>
```

See the API docs for more information on update().

Suppose you do not know the maximum up front. ProgressMeter has an indeterminate version, which does not display a percentage bar. Instead, an animation indicates that something is happening.

This code fragment displays the indeterminate progress meter while performing an XHR call. The time spent for the call is used to estimate the length of 10 calls, which will then be used as the maximum value. (Note: you will need to provide your own XHR url here.)

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.addOnLoad(function() {
            // This starts the indeterminate bar
            dijit.byId("setTestBar").update({indeterminate: true});
            // Do a call to see what the average response time is
            timeStart = new Date();
            var d = dojo.xhrGet({
                url: "../services/waste_time.jsp",
                handleAs: "text"
            });
            d.addCallback(function() {
                    // Stop the indeterminate bar
                    dijit.byId("setTestBar").update({indeterminate: false});
                    avgTime = new Date() - timeStart;
                    //Now you can estimate the time for 10 calls
                    avgTime10 = avgTime * 10;
                    dijit.byId("setTestBar").update({maximum: avgTime10});
                });
        });
```

dijit.ProgressBar
a progress widget

Attributes

indeterminate	Boolean false	Default is false. Show progress true: show that a process is underway but that the progress is unknown
maximum	Float 100	Maximum value possible
places	Number 0	number of places to show in values; 0 by default
progress	String "0"	(Percentage or Number) initial progress value. with "%": percentage value, 0% <= progress <= 100% or without "%": absolute value, 0 <= progress <= maximum

Methods

update(/* Object */prg)	update progress information - use progress, maximum and indeterminate properties in the object, just as in the attributes

Extension Points

onChange()	Callback for when progress changes

Accessibility

The progress bar is made accessible by providing a solid border around the visual progress indicator. This border is visible in high contrast mode as well as when images are turned off.

The internalProgress div is assigned the ARIA role of progressbar The valuenow attribute is updated as the progress is updated. No valuemin and valuemax values are provided since the valuenow attribute may be a string provided by the Web developer.

Note: The hot key for the Window-Eyes screen reader to speak progress bar information is ctrl-ins-b. JAWS provides the hot key ins-tab for announcing progress bar name and status.JAWS also has a setting to select the frequency of progress bar announcements. Go to the Configuration Manager, Select Set Options, then User Options and select the desired announcement frequency.

Changes to accessibility for version 1.0 of the ProgressBar

The ARIA valuenow property is set to the ProgressBar's progress value. The valuemin and valuemax properties are set to 0 and the ProgressBar's maximum, respectively.

Tooltip

Tooltip is similar to the title="" attribute in regular HTML, but is much more flexible. You can control the display timing, and specify arbitrary HTML for the tooltip text.

Example

Longanimity

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tooltip Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
        <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.Tooltip");
     </script>
</head>
<body class="tundra">
        <span id="wordOfTheDay">longanimity</span>
        <div dojoType="dijit.Tooltip"
                    connectId="wordOfTheDay"
                    label="a disposition to bear injuries patiently : forbearance">
                    </div>
    <div id="wordOfTheDay">Longanimity</div>
</body></html>
```

dijit.Tooltip
Internal widget that holds the actual tooltip markup, which occurs once per page. Called by Tooltip widgets which are just containers to hold the markup

Attributes

connectId	String	Id of domNode to attach the tooltip to. (When user hovers over specified dom node, the tooltip will appear.) **New in 1.0** You an also specify a comma-separated list of id to attach to multiple nodes. When using the programmatic interface, you can pass an array of Strings.
label	String	Text to display in the tooltip. Specified as innerHTML when creating the widget from markup.
showDelay	Integer 400	Number of milliseconds to wait after hovering over/focusing on the object, before the tooltip is displayed.

Accessibility - updated for 1.0

General Issues

Tooltips are displayed when the associated item receives focus or a mouseover event. Be careful when assigning tooltips to arbitrary elements such as spans of text which may not receive keyboard focus because users of assistive technology or keyboard only users will not benefit from the tooltip. If the tooltip information is important, make certain that the item which triggers display of the tooltip can receive focus via the keyboard. This can be accomplished by adding a tabindex="0" attribute onto the trigger element to put it into the tab order of the page.

Dialog and TooltipDialog

Dijit's modal Dialog Box simulates a regular GUI dialog box. The contents can be arbitrary HTML, but are most often a form or a short paragraph. The user can close the dialog box without acting by clicking on the X button in the top-right corner.

Example

The following dialog box shows a simple Change Password form:

Change Password

Old Password:

New Password:

Confirm New Password:

OK

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Dialog demo</title>
    <style type="text/css">
```

```
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Button");
       dojo.require("dijit.Dialog");
       dojo.require("dijit.form.TextBox");
       function checkPw(dialogFields) {
          if (dialogFields.confirmpw != dialogFields.newpw)
             alert("Confirmation password is different.  Password is unchanged.");
       }
     </script>
</head>
<body class="tundra">
<button dojoType="dijit.form.Button" onclick="dijit.byId('dialog1').show()">Change Password</button>
<div dojoType="dijit.Dialog" id="dialog1" title="First Dialog" execute="checkPw(arguments[0]);">
    <table>
        <tr>
        <td><label for="name">Old Password: </label></td>
                <td><input dojoType="dijit.form.TextBox" type="password" name="oldpw"></td>
        </tr>
        <tr>
                <td><label for="loc">New Password: </label></td>
                <td><input dojoType="dijit.form.TextBox" type="password" name="newpw"></td>
        </tr>
        <tr>
                <td><label for="desc">Confirm New Password: </label></td>
                <td><input dojoType="dijit.form.TextBox" type="password" name="confirmpw"></td>
        </tr>
        <tr>
                <td colspan="2" align="center">
                        <button dojoType=dijit.form.Button type="submit">OK</button></td>
        </tr>
    </table>
</div>
</body></html>
```

Everything in the dialog box is wrapped in an implicit form. The code in the execute() attribute is executed when the dialog is submitted normally.

Auto-focusing The First Text Box

Dialog, like most forms, does not automatically set the focus on the first form field. To do this with Dijit Dialog or TooltopDialog, simply define this code snippet once:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var focusFirstInput = function(e) {
   dijit.focus(dojo.query('input', this.domNode)[0]);
}
```

Then use dojo.connect to hook it to each dialog's onOpen event:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var c = dojo.connect(someDialogWidget, 'onOpen', dojo.hitch(someDialogWidget, focusFirstInput));
```

TooltipDialog

A variant on Dialog Box is dijit.TooltipDialog. This displays the dialog box contents in a Tooltip

Although both Dialog and TooltipDialog are modal, TooltipDialog can be closed by clicking anywhere on the screen, whereas for Dialog you must click on the [x] mark of the Dialog.

A TooltipDialog appears as a drop down from a button, similar to a drop down button.

Example

Change Password

Old Password:

New Password:

Confirm New Password:

OK

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;}

.geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>TooltipDialog demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Button");
       dojo.require("dijit.Dialog");
       dojo.require("dijit.form.TextBox");
       function checkPw(dialogFields) {
          if (dialogFields.confirmpw != dialogFields.newpw)
             alert("Confirmation password is different.  Password is unchanged.");
       }
     </script>
</head>
<body class="tundra">
<div dojoType="dijit.form.DropDownButton">
        <span>Change Password</span>
        <div dojoType="dijit.TooltipDialog" id="dialog1" title="First Dialog" execute="checkPw(arguments[0]);">
            <table>
                <tr>
                <td><label for="name">Old Password: </label></td>
                        <td><input dojoType="dijit.form.TextBox" type="password" name="oldpw"></td>
                </tr>
                <tr>
                        <td><label for="loc">New Password: </label></td>
                        <td><input dojoType="dijit.form.TextBox" type="password" name="newpw"></td>
                </tr>
                <tr>
                        <td><label for="desc">Confirm New Password: </label></td>
                        <td><input dojoType="dijit.form.TextBox" type="password" name="confirmpw"></td>
                </tr>
                <tr>
                        <td colspan="2" align="center">
                                <button dojoType=dijit.form.Button type="submit">OK</button></td>
                </tr>
            </table>
        </div>
</div>
</body></html>
```

dijit.Dialog, dijit.TooltipDialog
Simulates GUI Dialog box. Dialog displays forms overlaying the current page, TooltipDialog displays form in a tooltip.

Attributes

errorMessage	String Locale dep.	Message that shows if an error occurs
extractContent	Boolean false	Extract visible content from inside of <body> .... </body>
href	String	The href of the content that displays now. Set this at construction if you want to load data externally when the pane is shown. (Set preload=true to load it immediately.) Changing href after creation does not have any effect; see setHref();
isLoaded	Boolean false	Tells loading status see onLoad|onUnload for event hooks
loadingMessage	String Locale dep.	Message that shows while downloading
open	Boolean false	Is dialog box initially open
parseOnLoad	Boolean true	parse content and create the widgets, if any
preload	Boolean false	Force load of data even if pane is hidden.
preventCache	Boolean false	Cache content retreived externally
refreshOnShow	Boolean false	Refresh (re-download) content when pane goes from hidden to shown
title	String	**(TooltipDialog only)** Description of tooltip dialog (required for a11Y). For regular dialogs, you can set dijit.byId("dlg").titleNode.innerHTML to change the title.

Methods

cancel()	Cancels a inflight download of content
hide()	Hide the dialog if we haven't been initialized yet then we aren't showing and we can just return
layout()	Sets the background to the size of the viewport (rather than the size of the document) since we need to cover the whole browser window, even if the document is only a few lines long.
orient(/*Object*/ corner)	**(TooltipDialog only)** configure widget to be displayed in given position relative to the button. See dijit.Tooltip for notes on positioning
refresh()	Force a refresh (re-download) of content, be sure to turn off cache we return result of _prepareLoad here to avoid code dup. in dojox.layout.ContentPane
resize(/* String */size)	Explicitly set this widget's size (in pixels), and then call layout() to resize contents (and maybe adjust child widgets)
setContent(/*String|DomNode|Nodelist*/data)	Replaces old content with data content, include style classes from old content
setHref(/*String|Uri*/ href)	Reset the (external defined) content of this pane and replace with new url Note: It delays the download until widget is shown if preload is false
show()	**Dialog only** display the dialog first time we show the dialog, there's some initialization stuff to do
Extension Points	
onContentError(/*Error*/ error)	called on DOM faults, require fault etc in content default is to display errormessage inside pane
onDownloadEnd()	called when download is finished
onDownloadError(/*Error*/ error)	Called when download error occurs, default is to display errormessage inside pane. Overide function to change that. The string returned by this function will be the html that tells the user a error happend
onDownloadStart()	called before download starts the string returned by this function will be the html that tells the user we are loading something override with your own function if you want to change text
onLoad(/* Event */e)	Event hook, is called after everything is loaded and widgetified
onOpen(/*Object*/ pos)	**(TooltipDialog only)** called when dialog is displayed. See dijit.Tooltip for pos details.
onUnload(/* Event */e)	Event hook, is called before old content is cleared

Accessibility (added for 1.0)

General

When a dialog is opened focus goes to the title section of the dialog. This was implemented to provide screen reader support to speak the title of the dialog when it is opened. Likewise, when a tooltip dialog is opened, focus is placed on the container of the tooltip dialog. In future versions of the dialog and tooltip dialog widgets, focus will go to the first item in the dialog or tooltip dialog.

When focus is in a dialog, pressing the tab key will move focus forward to each focusable element within the dialog. When focus reaches the last focusable element in the dialog, pressing tab will cycle focus back to the dialog title. Pressing shift-tab will move focus backwards through focusable elements within the dialog until the dialog title is reached. If focus has previous cycled forward through all of the elements, pressing shift-tab with focus on the dialog title will move focus to the last element in the dialog. If focus has **not** previously been cycled through all of the focusable elements in the dialog using the tab key, pressing shift-tab with focus on the dialog title will leave focus in the title. The same focus cycling applies to the tooltip dialog as well with focus being set to the tooltip dialog container since there is no dialog title.

Known Issue: On Windows, In Firefox 2, when in High Contrast mode, the dialog with display correctly, but the underlying page will not be seen.

Keyboard

Action	Key
Navigate to next focusable element in the dialog/tooltip dialog	tab
Navigate to previous focusable element in the dialog/tooltip dialog	shift-tab (see general section above for additional details)
Close the dialog/tooltip dialog	escape

TitlePane

A TitlePane is a pane with a title on top that can be opened or collapsed. The visibility of the container is toggled by activating an arrow "button" on the title bar via the mouse or keyboard.

Examples

The rain in Spain falls mainly on the New York Stock Exchange.

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>TitlePane Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
        <script type="text/javascript">
           dojo.require("dojo.parser");
           dojo.require("dijit.TitlePane");
     </script>
</head>
<body class="tundra">
        <div dojoType="dijit.TitlePane" title="A Message from Thurston Howell III">
           The rain in Spain falls mainly on the New York Stock Exchange.
        </div>
</body></html>
```

dijit.TitlePane
A pane with a title on top, that can be opened or collapsed.

Attributes

duration	Integer 250	milliseconds to fade in/fade out
open	Boolean true	Whether pane is opened or closed.
title	String	Title of the pane. *Use setTitle() to change after creation time.*

Methods

setContent(/* String */content)	Typically called when an href is loaded. Our job is to make the animation smooth
setTitle(/* String */title)	sets the text of the title
toggle()	switches between opened and closed state

Accessibility (updated for 1.0)

Keyboard

Each title pane title is included in the tab order of the document.

Action	Key
toggle open/close of the title pane	enter or space key with focus on the title pane title
Navigate into an opened title pane	tab

Screen Reader Information

The title pane container will have an ARIA labelledby property which points to the id of the title pane title. The title pane title has the ARIA property of haspopup=true to indicate that it controls the display of the pane. The tilte pane container will have an ARIA role of region which will be fully supported in Firefox 3.

Advanced Editing and Display

ColorPalette	[inline:color_picker.png]
Editor	[inline:editor.png]
Grid (1.0)	[inline:grid_terms.gif]
InlineEditBox (1.0) For 0.9, see dijit.form.InlineEditBox	[inline:inline_edit.png]
Tree	[inline:tree.png]

ColorPalette

ColorPalette is a color picker for Web pages. You can do lots of stuff with this kind of widget like allowing your user to choose color for theming interactively. Color palette is an abstraction of popular hexa color codes by dijit.

Example

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ColorPalette Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
          dojo.require("dojo.parser");
          dojo.require("dijit.ColorPalette");
          function reportColor(selectedColor) {
              console.debug(selectedColor);
           }
     </script>
</head>
<body class="tundra">
        <div dojoType="dijit.ColorPalette" onChange="reportColor"></div>
</body></html>
```

dijit.ColorPalette
Grid showing various colors, so the user can pick a certain color

Attributes

defaultTimeout	Number 500	The number of milliseconds before a held key or button becomes typematic
timeoutChangeRate	Number 0.90	Fraction of time used to change the typematic timer between events; 1.0 means that each typematic event fires at defaultTimeout intervals < 1.0 means that each typematic event fires at an increasing faster rate
palette	String 7x10	Size of grid, either "7x10" or "3x4".

Extension Points

onChange(/* String */color)	Callback when a color is selected. Parameter is the hex color.

Accessibility

Keyboard

Action	Key
Navigate colors	Arrow keys
Pick a color	Spacebar or enter

Screen Reader

Screen readers will read the name of each color as it is highlighted. For example, "white", "seashell", "cornsilk", and so on.

Grid (1.0)

[inline:grid_terms.gif]

This widget is only available in 1.0. Grid is a DojoX project, but is documented here for user convenience.

Grids are familiar in the client/server development world. Basically a grid is a kind of mini spreadsheet, commonly used to display details on master-detail forms. From HTML terms, a grid is a "super-table" with its own scrollable viewport.

The Dojo grid is fast, robust, and very functional. In particular, grid has:

- High performance drawing. Rows are rendered "lazily" as the user moves down the grid.
- Addition or deletion of cells, rows and columns at will.
- Rows made of multiple sub rows, acting as one selectable unit.
- Summary rows.
- Adjustable row and column spans to fit data into different rectangular shapes in the row.
- Fixed rows and columns that stay still while the variable part of the grid scrolls.
- A rich event structure, so you can hook into selection and scrolling. Styles can be set from an arbitrary function through the onStyle hook.
- Support for "expandos" to show or hide detail.
- Automatic even/odd row coloring
- Ability to change the structure of rows on the fly.
- Support for rich in-cell editing of text or non-text data. All Dijit form wdigets can be used in cells. Cells can be set to edit with a single-click.
- Context menus settable for different cells
- Support for selecting cells, rows or columns.
- Option of automatic column sizing
- Grid nesting, so a grid can be housed in the cell of an outer grid.

A Simple Grid

To begin working with Grid, let's start with a simple example. The Dijit class directory is kept in /dijit/tests/_data/dijits.json. We'll base on a grid on it, and add embellishments throughout the next few sections.

Our First Grid

The Basics

Every page using Grid needs to import the basic grid style sheet. When used with Dijit and Dojo, your combined style loading block should look like:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<style type="text/css">
        /* tundraGrid.css matches Dijit Tundra style.  Others forthcoming.
            Use Grid.css on the same path for a more color-neutral theme */
        @import "http://o.aolcdn.com/dojo/1.0.0/dojox/grid/_grid/tundraGrid.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
</style>
```

The Model

Each Grid begins with data, and two DIV tags will define our data source. Because of the same-origin security rule, you will need to place the data file on your web server. You can download this file, dijits.json, at the bottom of this page. Just place it in the same directory as the example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;}

.geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojo.data.ItemFileReadStore"
                jsId="jsonStore" url="dijits.txt">
        </div>
        <div dojoType="dojox.grid.data.DojoData" jsId="model"
                rowsPerPage="20" store="jsonStore" query="{ namespace: '*' }">
        </div>
```

The first DIV should look familiar. It's the good ol' dojo.Data definition you use with the Dijit components ComboBox or Tree. The second is new - it's the dojox.grid.data.DojoData *adapter* that turns a data source into a Grid model. The model has options for which items to select. In this example, we turn the datasource jsonStore into the model named "model".

Models can also be created from JavaScript arrays, which we'll see in: Model Options.

The View

In standard spreadsheet and table terminology, a cell is the basic unit of displayed data. A row is a horizontally aligned contiguous group of cells, and a column is a vertically aligned contiguous group of cells. (Wow, that makes a simple concept sound complex!)

In grid-land, there's a distinction between rows and subrows. A subrow is what people normally think of as a row - it's exactly one cell tall. In Grid, a row may be more than one subrow - but it is selectable as a unit. So you'll notice in our demo grid that logical rows are exactly 2 subrows tall.

A View is a group of contiguous logical rows with the same inner and outer "shape". In our example above, each logical row is two subrows tall, with 2 columns on the top physical row and 1 column on the bottom physical row (the last cell spanning 2 columns). You specify this in JavaScript with an array of arrays. Each array element is an object literal. The most important property of the object is "name", which names the column. The column name always appear as the top logical row of the grid, and unlike other rows, it doesn't scroll up or down.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// a grid view is a group of columns
var view1 = {
        cells: [[
                {name: 'Namespace', field: "namespace"},
                {name: 'Class', width: "25em", field: "className"}
          ],
          [
                {name: 'Summary', colSpan:"2", field: "summary"}
          ]
        ]
};
```

Fields in the model are applied across each view cell in order. Property names are ignored. In our example, Namespace holds field 0, Class holds field 1, and so on.

As in good ol' HTML tables, you can specify:

- colSpan - note the capital "S" here, unlike standard HTML
- rowSpan
- width - all the usual CSS measurements are valid here

The Structure

Finally, views can be grouped together into a structure. You can think of a structure as a dijit.layout.LayoutContainer applied to views - you can place views in the top, bottom, left and/or right sides, plus one in the middle. Our simple example only has one view:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var layout = [ view1 ];
```

The Widget

The model and structure (which is composed of views) come together in the grid widget:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div id="grid" dojoType="dojox.Grid" model="model" structure="layout"></div>
```

The model and structure attributes point to our JavaScript variables for the model and structure. Nice! And with no other code, the grid is:

- scrollable
- sizable in the columns - point between columns on the top and drag left or right

- row-selectable - just click anywhere on a row

So, here's the entire program:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <title>Test dojox.Grid Basic</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"></meta>
        <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dojox/grid/_grid/tundraGrid.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
                body {
                        font-size: 0.9em;
                        font-family: Geneva, Arial, Helvetica, sans-serif;
                }
                .heading {
                        font-weight: bold;
                        padding-bottom: 0.25em;
                }

                #grid {
                        border: 1px solid #333;
                        width: 35em;
                        height: 30em;
                }
        </style>
        <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
                djConfig="isDebug:false, parseOnLoad: true"></script>
        <script type="text/javascript">
            dojo.require("dojo.data.ItemFileReadStore");
                dojo.require("dojox.grid.Grid");
                dojo.require("dojox.grid._data.model");
                dojo.require("dojo.parser");

                // a grid view is a group of columns.
                var view1 = {
                        cells: [[
                                {name: 'Namespace', field: "namespace"},
                                {name: 'Class', width: "25em", field: "className"}
                          ],
                          [
                                {name: 'Summary', colSpan:"2", field: "summary"}
                          ]
                        ]
                };
                // a grid layout is an array of views.
                var layout = [ view1 ];
</script>
</head>
<body class="tundra">
<div class="heading">Our First Grid</div>
        <div dojoType="dojo.data.ItemFileReadStore"
                jsId="jsonStore" url="dijits.txt">
        </div>
        <div dojoType="dojox.grid.data.DojoData" jsId="model"
                rowsPerPage="20" store="jsonStore" query="{ namespace: '*' }">
        </div>
        <div id="grid" dojoType="dojox.Grid" model="model" structure="layout"></div>
</body>
</html>
```

Note that normally with CDN, you would need to wrap the view1 initialization code like in a dojo.addOnLoad. But that's not necessary here because no DOM needs to be drawn yet. We're just setting up anonymous objects. Because these are available when the widgets are drawn, we can use the structure property in the Grid tag. The design is nice and clean.

Now let's cover the model, view and structure elements in more depth:

Combining Views: Row Selection and Independent Scrolling

That's nice so far. The column header stays in place while the user scrolls down, making them easy to identify. Can we apply to that rows as well?

Yup. You can make row headers that stay in place and act as selection points.. What's more, you can split your grid into arbitrary scrollable sections that can stay in sync or scroll independently. You do this by gluing more than one view into a structure.

Selectable Rows

By default, when you click on a cell, you select that cell *and* the entire containing row. A special view called dojox.grid.GridRowView draws a column of empty handles. When clicked, these handles select the row without selecting a particular cell. Unlike most other views, GridRowView has no "cells" property, only a width. So adding this to the structure on our last page:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var rowbar = {
     type: 'dojox.GridRowView', width: '20px'
};

// a grid layout is an array of views.
var layout = [ rowbar, view1 ];
```

The resulting page (downloadable below as grid1.html), yields:

As you would expect, CTRL+click selects non-contiguous rows and SHIFT+click selects contiguous ones. As it stands, the selection is only for display. In the Events section, we'll actually do something with the rows.

Linked Scrollable Views

You may have noticed that the Selection bar rows and the data rows scroll together. So it's no surprise that when you add another view, it too scrolls vertically in sync.

To see this, we'll split off the Dijit class name into its own view:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// a grid view is a group of columns
var view1 = {
        cells: [[
                {name: 'Namespace', field:0, width: "25em"}
          ],
          [
                {name: 'Summary', colSpan:"2", field:2}
          ]
        ]
};
var rowbar = {
   type: 'dojox.GridRowView', width: '20px'
   };
var fixedColumn = {
        cells: [[ {name: 'Class', field:1, width:"25em"} ]]
};
```

Since we're using fields in a different order, specifying the field numbers in each cell definition is mandatory. Now combine that into a layout like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var layout = [ rowbar, fixedColumn, view1 ];
```

And you get two grids with separate scroll bars. By default each scrollbar moves *both* views in sync with each other.

Admittedly, the extra scroll bar isn't very useful. But when you add fields to the right hand grid like so:

```
var view1 = {
        cells: [[
                {name: 'Namespace', field:0, width:"20em"},
                {name: 'Description', field:3, width:"20em"}
          ],
          [
                {name: 'Summary', field:2},
                {nane: 'Examples', field:4}
          ]
        ]
};
```

And run the example, you find the bottom scroll bar scrolls the views *independently*. We say the views are vertically dependent, but horizontally independent.

Now you turn off the scroll bar in the left hand view with the noscroll property:

```
var fixedColumn = {
    noscroll: true,
    cells: [[
```

Yields:

Sorting and Other Dojo.Data Considerations

By default, when dojo.data datasources feed a Grid, the columns are not user-sortable. That's easy to rectify. Just set clientSort="true" in the tag:

```
<div dojoType="dojox.grid.data.DojoData" jsId="model"
        rowsPerPage="20" store="jsonStore" query="{ namespace: '*' }"
        clientSort="true">
</div>
```

Now the user can click on any column to sort it. You may also set the sort order programmatically:

```
// Sort 4'th field in ascending order
myGrid.setSortIndex(3, true);
```

Filtering

To filter a list, you can create a new adapter with a different query then attach it to the grid.

```
var newModel = new dojox.grid.data.DojoData();
newModel.rowsPerPage = 20;
newModel.store = myStore;
newModel.query = {
      type: someOtherType
};
newModel.clientSort = true;
myGrid.setModel(newModel);
// Remember to call newModel.destroy() when you're done.
```

A popular use of filtering is to display a grid with everything, then let the user chop the list down incrementally. In this case, you can define an initial model with a minimal query. Save it, and then you can setModel back to it for quickly resetting all the filters. But do not keep unused models lying around! They take up memory.

Cell Editing

Up until now, we've showed grid contents in view mode. Now, let's add some interactivity.

An editable cell is handled by a *cell editor*. To make a cell editable, you simply specify the cell editor class in the column definition. Here is a part of the view definition code in /dojoroot/dojox/grid/tests/test_edit.html.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
gridLayout = [
        cells: [[
                { name: 'Priority', styles: 'text-align: center;',
                   editor: dojox.grid.editors.select,
                   options: ["normal", "note", "important"]
                },
                { name: 'Mark', width: 3, styles: 'text-align: center;',
                   editor: dojox.grid.editors.bool
                },
                { field: 2, name: 'Status', styles: 'text-align: center;',
                   editor: dojox.grid.editors.select,
                   options: [ "new", "read", "replied" ]
                }
```

In the actual grid, you double-click on a cell to begin editing. Here, we've double clicked on the status box, and a SELECT appears in place of the data:

[inline:gredit_edit1.png]

Some of the cell editors, like dojox.grid.editors.DateTextBox are just wrappers for their Dijit form widget counterparts. All the functionality and properties are available to you. This is an example from /dojoroot/dojox/grid/tests/test_edit_dijit.html.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
gridLayout = {
  cells: [[
        { name: 'Date', width: 10, field: 7,
           editor: dojox.grid.editors.DateTextBox,
           formatter: formatDate,
           constraint: {formatLength: 'long', selector: "date"}
        }
  ]]
};
```

Editing the Date cell brings up the familiar Dijit date box:

[inline:gredit_edit2.png]

Now, what do you actually do with an edited cell? That's really up to you, and you hook in your desired code by connecting to an event, which we'll cover next.

Available Cell Editors

Cell Editor Class	Attributes
dojox.grid.editors.CheckBox	See dijit.form.CheckBox
dojox.grid.editors.ComboBox	See dijit.form.ComboBox
dojox.grid.editors.DateTextBox	See dijit.form.DateTextBox
dojox.grid.editors.Editor	See dijit.Editor
dojox.grid.editors.select	Similar to dojox.grid.editors.ComboBox, but doesn't allow freeform values **String[] options**: text of each item **String[] values**: value for each item **Boolean returnIndex**: editor returns only the index of the selected option and not the value
dojox.grid.editors.TextBox	See dijit.form.TextBox
dojox.grid.editors.TimeTextBox	See dijit.form.TimeTextBox

Events

As we alluded to in the last few pages, selection and cell editing is pointless without some kind of background processing. So how do you hook code into these places? Through Dojo's event model, of course!

Editing Changes

If you're using a writable dojo.data datastore, you simply hook your procedures into the dojo.data Notification API. Suppose in our running

example, we make the Description field editable:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
cells: [[
    {name: 'Namespace', field:0, width:"30em"},
    {name: 'Description', field:3, width:"30em",
        editor: dojox.grid.editors.Dijit }
  ]
],
```

Then, we place the hook into dojo.data.Notification's onSet extension point:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dojo.data.ItemFileWriteStore"
        jsId="jsonStore" url="dijits.txt">
    <script type="dojo/connect" event="onSet" args="item,attr,oldVal,newVal">
       console.debug("About to change "+attr+" from "+oldVal+" to "+newVal);
       // Save the record with dojo.xhrPost or your favorite remote method
    </script>
</div>
```

Here is the entire source code. Note how we place the layout initialization code in a dojo/method block inside the Grid tag. That's due to the editor class dojox.grid.editors.Dijit in the view definition. When you're using CDN, the dojox.grid.editors package is not available directly after the dojo.require., so we can't place the initialization code after it (as we did in our previous examples). By using a dojo/method, we can place this code close to its use (the Grid tag) and it's guaranteed to run after all dojo.require'd modules have loaded. You can also use dojo.addOnLoad to accomplish this.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>Test dojox.Grid Editing</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"></meta>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dojox/grid/_grid/tundraGrid.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
        body {
            font-size: 0.9em;
            font-family: Geneva, Arial, Helvetica, sans-serif;
        }
        .heading {
            font-weight: bold;
            padding-bottom: 0.25em;
        }

        #grid {
```

```
            border: 1px solid #333;
            width: 40em;
            height: 30em;
        }
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="isDebug:false, parseOnLoad: true"></script>
    <script type="text/javascript">
        dojo.require("dojo.data.ItemFileWriteStore");
        dojo.require("dojox.grid.Grid");
        dojo.require("dojox.grid._data.model");
        dojo.require("dojox.grid.editors");
        dojo.require("dojo.parser");

</script>
</head>
<body class="tundra">
<div class="heading">Grid Events</div>
    <div dojoType="dojo.data.ItemFileWriteStore"
        jsId="jsonStore" url="dijits.txt">
        <script type="dojo/connect" event="onSet" args="item,attr,oldVal,newVal">
            console.debug("About to change "+attr+" from "+oldVal+" to "+newVal);
        </script>
    </div>
    <div dojoType="dojox.grid.data.DojoData" jsId="model"
        rowsPerPage="20" store="jsonStore" query="{ namespace: '*' }"
        clientSort="true">
    </div>
    <div id="grid" elasticView="2" dojoType="dojox.Grid" model="model"
        jsId="thisGrid">
        <script type="dojo/method">
            var view1 = {
                cells: [[
                    {name: 'Namespace', field:0, width:"30em"},
                    {name: 'Description', field:3, width:"30em"}
                  ],
                  [
                    {name: 'Summary', field:2, colspan:2,
                     editor: dojox.grid.editors.Dijit }
                  ]
                ]
            };
            var rowbar = {
               type: 'dojox.GridRowView', width: '20px'
            };
            var fixedColumn = {
                 noscroll: true,
                 cells: [[ {name: 'Class', field:1} ]]
            };

            // When you initialize inside the dojo/method script, you must set the
            // structure manually.
            var layout = [ rowbar, fixedColumn, view1 ];
            thisGrid.setStructure(layout);
        </script>
       </div>
</body>
</html>
```

You can read more on the dojo.data Notification API in Part 3 of the book, but here are the basics for your Grid needs:

- **onSet: function(/* item */ item, /* attribute-name-string */ attribute, /* object | array */ oldValue, /* object | array */ newValue)** - called after any cell is edited and saved.
- **onNew: function(/* item */ newItem,)** - called after a row is added to the grid.
- **onDelete: function(/* item */ deletedItem)** - called after a row is deleted

Low-Level Events

For more granular event processing, you can hook into Grid events. Each event calls the function you provide, passing back the event object. If e is the event, the interesting stuff is in:

- **e.rowIndex**: the row number
- **e.cell.index**: the column (cell) number. Taken with e.rowIndex, effectively gives you coordinates of a cell event.
- **e.keyCode**: keystroke value, only applicable to keydown event.

Entity	click/double click	mouse over/out	right click
Data Cell	onCellClick onCellDblClick	onCellMouseOver onCellMouseOut	onCellContextMenu
Hdr Cell	onHeaderCellClick onHeaderCellDblClick	onHeaderCellMouseOver onHeaderCellMouseOut	onHeaderCellContextMenu
Data Row	onRowClick onRowDblClick	onRowMouseOver onRowMouseOut	onRowContextMenu
Hdr Row	onHeaderClick onHeaderDblClick	onHeaderMouseOver onHeaderMouseOut	onHeaderContextMenu

Styles

Cell Styles

Fixed styles that apply to all cells of a column are settable in the view. The properties you need:

- **classes**: sets a CSS class for both header and contents
- **styles**: sets a CSS style for both header and contents
- **headerClasses**: sets a CSS class for column header. Combines with "classes".
- **headerStyles**: sets a CSS style for column header. Combines with "styles"
- **cellClasses**: sets a CSS class for the contents. Combines with "classes".
- **cellClasses**: sets a CSS class for the contents. Combines with "styles"

The onStyleRow Extension Point

The above properties make sweeping style changes across a column. But how do you change styles for individual cells? For example, suppose you want to color negative numbers red and positive numbers black.

The onStyleRow extension point can do this. Grid passes a Row object to your onStyleRow method. You set the properties customStyles and/or customClasses in this object, and Grid will restyle your row accordingly. The Row object has the following properties:

- **selected**, true if the row is selected;
- **over**: true of the mouse is over the row;
- **odd**: true if the row is odd.
- **customClasses**: you set this property for the CSS class
- **customStyles**: does the same with CSS styles

In the following example, we look at each row and color the row text red if the Dijit contains a description.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function colorDescriptions(inRow) {
   if (model.getRow(inRow.index) === undefined)
      return;
   if (model.getRow(inRow.index).description != '')
      inRow.customStyles = 'color:red';
}
```

You connect this function to the extension point in the Grid tag:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div id="grid" elasticView="2" dojoType="dojox.Grid" model="model"
            structure="layout" onStyleRow="colorDescriptions"></div>
```

And behold ... red text where the row has a non-blank description

[inline:grid_demo4.png]

Cell Formatters

A cell formatter alters the *text* in the cell, not the styles. As you might guess, this is useful for formatting numbers, currency, dates, percentages, etc. The cell formatter is specified with the formatter extension point in the cell object:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ name: 'Amount', formatter: formatCurrency, field: "moola"},
```

And define the formatter itself separately. (You can also use a function literal inside the cell definition).

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function formatCurrency(inDatum){
        return isNaN(inDatum) ? '...' : dojo.currency.format(inDatum, { currency: 'USD' });
}
```

Liberal use of Dojo i18n for formatting dates and numbers is strongly encouraged!

Summary Rows

Grouping data for summarization requires a simple strategy. We will calculate a summary subrow for *every row in the table*, then just hide the ones not on a group boundary. So here is some (boring!) numeric data:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{
        identifier: 'id',
        label: 'name',
        items: [
        { id:'Q1_06', name: 'Q1 2006', year:2006, quarter:1, sales:345436 },
        { id:'Q2_06', name: 'Q2 2006', year:2006, quarter:2, sales:234525 },
        { id:'Q3_06', name: 'Q3 2006', year:2006, quarter:3, sales:129104 },
        { id:'Q4_06', name: 'Q4 2006', year:2006, quarter:4, sales:-10000 },
        { id:'Q1_07', name: 'Q1 2007', year:2007, quarter:1, sales:-178775 },
        { id:'Q2_07', name: 'Q2 2007', year:2007, quarter:2, sales:286027 },
        { id:'Q3_07', name: 'Q3 2007', year:2007, quarter:3, sales:429546 },
        { id:'Q4_07', name: 'Q4 2007', year:2007, quarter:4, sales:946375 }
    ]
}
```

To implement our strategy, we first build two functions which supply the total and total label for each totalling subrow.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function getYearlyLabel(inRowIndex) {
            return model.getRow(inRowIndex) ? "Total for " + model.getRow(inRowIndex).year : 'None';
        }
        function getYearlyTotal(inRowIndex) {
            return model.getRow(inRowIndex) ? (yearlyTotal += model.getRow(inRowIndex).sales) : -1;
        }
```

Next, we make an onAfterRow procedure to hide all the rows that are not after Q4

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// inRow is an array of subRows. we hide the summary subRow except for every nth row
function onAfterRow(inDataIndex, inRow) {
    // note that header row inDataIndex == -1
    inRow[1].hidden = true;
    // Before rows 3, 7, 11 turn on display of the total
    if (inDataIndex != -1 && inDataIndex % 4 == 3) {
        yearlyTotal = 0;
        inRow[1].hidden = false;
    }
}
```

Then we wire it all up in the view definition:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var view1 = {
    onAfterRow: onAfterRow,
        cells: [[
                {name: 'Year/Quarter', field:'name'},
                {name: 'Sales', field:'sales'}
        ],[
            // The summary subrow, which will be hidden on most rows
                {name: 'Cell2', get: getYearlyLabel },
                {name: 'Yearly Sales', get:getYearlyTotal, styles:'font-weight:bold;'},
        ]]
};
```

And the summary rows are displayed. Currently this example shows the summary row for all rows. A question is pending on the forums about this, and the example will be fixed accordingly.

Model Options

Array Models

Instead of feeding dojo.data sources to the grid, you may feed it a two-dimensional array. This approach works well for grids with fixed data, e.g. static reference tables. Here's an example culled from the unit test /dojoroot/dojox/grid/tests/test_grid.html. Note the following:

- Fields are numbered, not named, and start at 0.
- If the cells are laid out in the same order as the data elements, you may omit the field: property in the cells.
- Placing the initialization code in addOnLoad is necessary when using CDN because of the dojox.grid.data.Table reference.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;}

```
.geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<script type="text/javascript">
        dojo.require("dojox.grid.Grid");
        dojo.require("dojox.grid._data.model");
        dojo.require("dojo.parser");
    // We need to place the initializations here because of CDN.  This way they
//    are run after dojo.require's are loaded (we need dojox.grid.data.Table
    //    in particular, but before grid is drawn.
        dojo.addOnLoad(function() {
                var data = [
                [ "normal", false, "new", 'But are not followed by two hexadecimal',
                        29.91, 10, false ],
                [ "important", false, "new", 'Because a % sign always indicates', 9.33, -5, false ],
                [ "important", false, "read", 'Signs can be selectively', 19.34, 0, true ],
                [ "note", false, "read", 'However the reserved characters', 15.63, 0, true ],
                [ "normal", false, "replied", 'It is therefore necessary', 24.22, 5.50, true ],
                [ "important", false, "replied", 'To problems of corruption by', 9.12, -3, true ],
                [ "note", false, "replied", 'Which would simply be awkward in', 12.15, -4, false ]
                ];
                // global var "model"
                var model = new dojox.grid.data.Table(null, data);
                var view1 = {
                        cells: [[
                                {name: 'Column 0'}, {name: 'Column 1'}, {name: 'Column 2'},
                                {name: 'Column 3', width: "150px"}, {name: 'Column 4'}
                        ],[
                                {name: 'Column 5'}, {name: 'Column 6'}, {name: 'Column 7'},
                                {name: 'Column 8', field: 3, colSpan: 2}
                        ]]
                };
                var layout = [ view1 ];
           // Now set the model and structure
           gridWidget.setModel(model);
           gridWidget.setStructure(layout);
    });
        </script>

</script>
</head>
<body class="tundra">
<div class="heading">dojox.Grid Basic Test</div>
<div id="grid" dojoType="dojox.Grid" jsId="gridWidget"></div>
</body>
</html>
</script>
```

Observers

You can watch for changes to cells at the model level. This is called setting an *observer* and it followed the familiar Observer Design Pattern. With array-fed Grids, these are your only hook-in points for sending XHR back to the server. As we saw in Events, you can use dojo.Data's notification API, or the Observer API for the same thing.

To observe a model, you first create an object literal with function properties. These functions must be named:

- **modelChange**: called when any cell data changes (due to editing or calling change methods on the model).
- **modelInsertion**: called when a row is added
- **modelRemoval**: called when a row is removed.
- **modelAllChange**: called when entire model needs to be re-rendered.
- **modelRowChange**: called when a row is changed
- **modelDatumChange**: called when cell data changes

For example, this observer watches all model changes and updates a visible row count.

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
var modelObservers = {
   modelChange:function(){
        dojo.byId("rowCount").innerHTML = 'Row count: ' + model.count;
   }
}
```

Then you register the observer with the model:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
model.observer(modelObservers);
```

dojox.grid.data.Model
The raw data behind a grid. Can be obtained from a grid with dojox.grid.Grid.getModel()

Properties

clientSort	Boolean	User is allowed to sort by clicking column headers (dojo.data models only)
count	Integer	Number of rows currently in the model
query	Object	dojo.data query for current store (dojo.data stores only)

rowsPerPage	Integer	Rows to scroll down before another set of rows is rendered.
store	String	dojo.data store variable (dojo.data stores only)
updating	Integer	Number of rows in an UPDATING state
Methods		
String getDatum(/* Integer */inRowIndex, /* Integer */inColIndex)		Return data at the location. Column indexes are id's for dojo.data models, integers for array models.
Integer getColCount()		Return number of columns
Object getRow(/* Integer */inRowIndex)		Returns item for dojo.data elements, array for array-based elements, at inRowIndex
Integer getRowCount()		Returns number of rows in model
notObserver(/* Object */ inObserver)		De-register an observer.
observer(/* Object */ inObserver, /* String */inPrefix)		Register an observer object with the model. inPrefix is added to each function name before calling, as in myPrefixModelChange. That way you can specify multiple observers in the same object.
Integer pageToRow(/* Integer */inPageIndex)		Starting row number on given page
Integer rowToPage(/* Integer */inRowIndex)		Return page number for this row
setDatum(/* Object */inDatum, /* Integer */inRowIndex, /* Integer */inColIndex)		Set the data at the given position. Normally called by cell editing.
setRow(/* Object */inRow,/* Integer */inRowIndex)		Overwrite the row at the given index. Row must be in same object format as other rows, e.g. an item for dojo.data models or array for array models.
swap(/* Integer */inIndexA, /* Integer */inIndexB)		Swap rows A and B in model.

Cell Options

Grid neatly separates the model from the view in its MVC implementation. We just covered the model. Now we'll cover the view - that is, everything used to display the model elements.

In the world of Grid, the structure is the largest unit. Structures are composed of views. Views are composed of cells (what we normally think of as a column). We'll start at this lowest level first. As we've seen a cell is defined by a JavaScript object like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ name: 'Apple', field: 'apple', width: '4.5em' }
```

This defines a column with the heading of Apple and initial width of 4.5em, and mapped to the field 'apple' in the model. The field index can be a string, as is the case for dojo.Data-fed grids, or a number, as in array-fed Grids.

Cells themselves are not directly addressable. You usually get them by starting with a grid variable and work downwards: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// get the cell in the third view, second subrow, fourth column (all indexes are 0-based).
var thisCell = mygrid.structure[2].cells[1][3];
```

From here, you can access these properties and call these methods on thisCell:

Attributes		
cellClasses	String	CSS class applied to data
cellStyles	String	CSS Styles applied to data
classes	String	CSS Classes applied to all column: data and header
colSpan	Integer	Like colspan in HTML, number of columns each cell occupies. Only meaningful if there are subrows in each row, otherwise ignored.
editor	Class	For editable cells, this can be either "dojox.grid.editors.Dijit" for a Dijit form control or "dojox.grid.editors.Editor" for the rich text editor. (The rich text editoris essentially Dijit's with some modifications to make it nicer in a grid - like a shared toolbar.) Grid also bundles its own editors like dojox.grid.editors.bool for Booleans, but they are redundant with the Dijit widgets.
editorClass	String	If editor="dojox.grid.editors.Dijit", this designates the Dijit form widget to use. Note: this is a string with the class name, not the class itself.

extraField	Integer	Index field like "field", but tacked on
field	Integer	Index of field data in from model
headerClasses	String	CSS class applied to column header
headerStyles	String	CSS Styles applied to column header
name	String	Name used for the column header
noresize	Boolean	If true, column cannot be resized.
rowSpan	Integer	Number of subrows that each cell in this column occupies.
styles	String	CSS styles applied to all column: data and header
value	String	Constant value to placed in each column cell. Can contain HTML.
width	Number	Initial width of the column in ems
Extension Points		
formatter(/* String */ inDatum)		Function which handles formatting the cell data.
get(/* Integer */ inRowIndex)		Name of the function called to get a value.

View Options

Attributes		
cells	dojox.grid.cell[]	Array of cell objects, each defining a column
defaultCell	dojox.grid.cell	Properties of this cell are used as defaults in all cells
defaultWidth	String	width of cells (columns), if not specified in the cell itself
noscroll	Boolean	If true, do not draw scroll bars on right or bottom.
rowPad	Integer	Space to use between rows, in pixels
type	String	Name of class for programmatically-generated views like GridRowView.
viewWidth	String	Width of entire view in valid CSS units.

Methods	
Node getCellNode(/* Integer */ inRowIndex, /* Integer */ inCellIndex)	Get cell at the specified location
Integer getColumnsWidth()	Width of data columns, in px
String getContentWidth()	Width of the containing box in CSS units, minus scrollbar (= getColumnsWidth + padding and borders, etc.)
Node getRowNode(/* Integer */ inRowIndex)	Get row at the specified location
Integer getScrollbarWidth()	Width of scrollbar in px, 0 for noscroll
String getWidth()	Total width of scrollbar and columns in CSS units
Boolean hasScrollbar()	True if there are enough rows to display a scrollbar.
resize()	resizeHeight(), then resizeWidth()
resizeHeight()	Resize to fit new height of containing box
resizeWidth()	Resize to fit new width of containing box
setColWidth(/* Integer */ inIndex, /* Integer */inWidth)	Resize cell (column) at inIndex to inWidth pixels
setSize(/* Integer */ w, /* Integer */ h)	Set size of the bounding box. Call resize() after.

Grid Tag

Grid Sizing

By default, the grid fits exactly in the parent DOM node provided for it. If all of the rows do not fit, a vertical scroll bar appears. Likewise, if all the columns don't fit, the horizontal scroll bar appears. Nothing surprising there.

The following properties resize the grid to fit all of the columns. In essence, setting either of these makes the appropriate scroll bar disappear.

- **autoHeight**: resize the grid height to fit all of the rows.
- **autoWidth**: resize the grid width to fit all the rows.

You may either set these on the Grid tag itself, or set the properties through JavaScript and call dojox.grid.Grid.update() to redraw. You can also resize the width and height of the container (dojo.contentBox is good for this) and call update().

dijit.Grid
A grid widget with virtual scrolling, cell editing, complex rows, sorting, fixed columns, sizeable columns, etc.

Attributes

autoHeight	Boolean	If autoHeight is true, grid height is automatically set to fit the data.
autoRender	Boolean	If autoRender is true, grid will render itself after initialization.
autoWidth	Boolean	If autoWidth is true, grid width is automatically set to fit the data.
defaultHeight	string	default height of the grid, measured in any valid css unit.
elasticView	Integer	One of the views in the grid may be "elastic", that is: expanding or contracting to fill the remaining size when all non-elastic elements are placed. By default, the middle view is elastic. Specifying this property makes the indexed grid view elastic.
fastScroll	Boolean	flag modifies vertical scrolling behavior. Defaults to true but set to false for slower scroll performance but more immediate scrolling feedback
keepRows	Integer	Number of rows to keep in the rendering cache.
model	String or Object	Grid data model
rowCount	Integer	Number of rows to display
rowsPerPage	Integer	Number of rows to render at a time.
singleClickEdit	Boolean	Single-click starts editing. Default is double-click
structure	Object or String	View layout defintion. Can be set to a layout object, or to the (string) name of a layout object.

Methods

addRow(/* Array */ inRowData, /* Integer */ inIndex)	Add row inRowData after row[inIndex] in both displayed grid and model
String get(/* Integer */ inRowIndex)	Get raw data at row inRowIndex in the current cell position.
dojox.grid.Cell getCell(/* Integer */ inIndex)	Get the cell object (the column definition) in column inIndex
String getCellName(/* Integer */ inIndex)	Get the column name of inIndex
Boolean canSort(/* Integer */ inSortInfo)	Sort information, in sortInfo is 1-based index of column on which to sort, positive for an ascending sort and negative for a descending sort returns true if grid can be sorted on the given column in the given direction
Boolean getSortAsc(/* Integer */ inSortInfo)	returns true if grid is sorted in an ascending direction.
Integer getSortIndex()	returns index of sorted field
refresh()	re-render the grid with the new data model
removeSelectedRows()	remove all selected rows in displayed grid and model
render()	Render the grid, headers, and views. Edit and scrolling states are reset. To retain edit and scrolling states, see Update.
renderAtIdle()	Same as render, but wait until all background processing has completed.
resize()	Call after setting width or height
resizeHeight()	Call after setting just the height
setCellWidth(/* Integer */ inIndex. /* String */ inUnitWidth)	Set column size to a given CSS unit width
setModel(/* dojox.grid.model */ inModel)	set the grid's data model
setSortIndex(/* Integer */ inIndex, /* Boolean */ inAsc)	Sets a sort column and direction (true=ascending, false=descending).
setStructure(/* dojox.grid.Structure */ inStructure)	Install a new structure and rebuild the grid.
scrollTo(/* Integer */ inTop)	Vertically scroll the grid to a given pixel position

scrollToRow(/* Integer */ inRowIndex)	Scroll the grid to a specific row.
sort()	sort on current sort field
update()	Update the grid, retaining edit and scrolling states.
updateRow(/* Integer */inRowIndex)	Change the number of rows.
updateRowCount(/* Integer */inRowCount)	Update row count property to inRowCount
updateRowStyles(/* Integer */inRowIndex)	Update the styles for a row after it's state has changed.

InlineEditBox (1.0)

dijit.InlineEditBox is a new widget in 1.0, similar to the dijit.form.InlineEditBox widget in 0.9

InlineEditBox is best described as a behavior on some text on the page, such that clicking that text brings up an editor, and when the text is saved, the screen is reverted to it's original state (but with the new text). The editor is created on-demand, so as to not slow down page load.

Examples

Edit me - I trigger the onChange callback

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>InlineEdit Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.InlineEditBox");
       dojo.require("dijit.form.TextBox");
       function myHandler(idOfBox, value) {
           console.debug("Edited value from "+idOfBox+" is now "+value);
       }
     </script>
</head>
<body class="tundra">
        <h3 id="editable"
              dojoType="dijit.InlineEditBox" title="h3 example"
              onChange="myHandler(this.id,arguments[0])">
Edit me - I trigger the onChange callback
    </h3>
</body></html>
```

When a user loads the page, they see the text "Edit me - I trigger the onChange callback". If the user clicks the text, a TextBox widget containing the text "Edit me - I trigger the onChange callback" appears. When the user changes the value and clicks away, the TextBox disappears and the TextBox's contents appear inline.

InlineEditBox supports the textarea mode through the Textarea widget. By simply saying editor=dijit.form.Textarea, you can use that editor. Furthermore, by adding renderAsHtml=true, users can enter HTML into the Textarea and have it appear inline as rich text. :

I'm one big paragraph. Go ahead and edit me. I dare you. The quick brown fox jumped over the lazy dog. Blah blah blah blah blah blah blah ...

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title>InlineEdit Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.InlineEditBox");
       dojo.require("dijit.form.Textarea");
     </script>
</head>
<body class="tundra">
    <p id="areaEditable" dojoType="dijit.InlineEditBox" title="p example"
         autoSave="false">
                        I'm one big paragraph.  Go ahead and edit me.  I dare you.
                        The quick brown fox jumped over the lazy dog.  Blah blah blah blah blah blah blah ...

</body></html>
```

When a user loads the page, they see the paragraph of rich text. If the user clicks the text, a Textarea widget containing the paragraph in plain text form appears. When the user changes the value and clicks away, the Textarea disappears and the Textarea's contents appear inline.

InlineEditBox can use any arbitrary widget that has a text value, or has the methods get/setDisplayedValue as an editor. DateTextBox is an example of such a widget. This code shows a DateTextBox as the editor:

1/1/2007

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>InlineEdit Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.InlineEditBox");
       dojo.require("dijit.form.DateTextBox");
     </script>
</head>
<body class="tundra">
   <span dojoType="dijit.InlineEditBox" editor="dijit.form.DateTextBox" title="date example"
              width="200px" title="purchase date as mm/dd/yy">
        1/1/2007
   </span>
</body></html>
```

Note that the originally displayed text is generated by the server, and thus must be in the correct locale for the client machine. Since the server is generating the text, that burden of localizing the text falls on the server.

The InlineEditBox can wrap around any widget that implements the following interface:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
/* void */ setTextValue(/*String*/ value) { ... }
String value = getTextValue() {... }
/* void */ focus() { ... }
```

The contained widget's setTextValue() method is called with the previously displayed text. When the Save button is pressed, the editing widget's getTextValue() method is called to retrieve the new text. After which, the editing widget is hidden, and the returned text is displayed. The focus method allows the editing widget to intelligently set focus to an appropriate node.

dijit.InlineEditBox
Edit behavior applied to a node

Attributes

autoSave	Boolean true	Changing the value automatically saves it; don't have to push save button (and save button isn't even displayed)
buttonSave	String	Save button label
buttonCancel	String	Cancel button label
editing	Boolean false	Is the node currently in edit mode?

editor	String dijit.form.TextBox	name of widget to use as editor
editorParams	Object	Parameters to pass to editor (in addition to the value being edited. ex: "{constraints: {places:0} }"
renderAsHTML	Boolean false	true if the editor widget and takes HTML for setValue(), and returns HTML from getValue(). Ex: dijit.Editor
value	String	Read-only value of box
Methods		
cancel(/*Boolean*/ focus)		evert to display mode, discarding any changes made in the editor
save(/*Boolean*/ focus)		Focus on the display mode text
Extension Points		
onChange(/* String */value)		User should set this handler to be notified of changes to value

Accessibility

General Behavior

When InlineEditBoxes are "closed" they appear as text but are tab stops in the keyboard focus ring and have an accessible role of button. They can have autoSave or non-autoSave behavior. When an non-autoSave InlineEditBox is open it has associated Save and Cancel buttons. An autoSave InlineEditBox does not have these buttons and they act like miniature forms or dialogs, i.e pressing the Esc key will close the widget and pressing the Enter key will close the widget, saving and displaying the text.

Note that since InlineEditBoxes may be used on the page without a traditional label element, the developer should add a title attribute in order to provide a description that is available to screen reader users. The title will also be displayed by the browser when the user places the mouse over the element.

Keyboard

If the widget is closed.

Action	Key
Navigate to the next widget in the tab order.	Tab
Navigate to the prior widget in the tab order.	Shift+Tab
Open the widget.	Enter or spacebar

Note: The Esc key is ignored.

TextBox with autoSave specified and the TextBox is open:

Action	Key	Comments
Navigate to the next widget in the tab order.	Tab	The data is saved and the widget closes.
Navigate to the prior widget in the tab order.	Shift+Tab	The data is saved and the widget closes.
Close the TextBox, saving changes.	Enter	Keyboard focus is on the closed InlineEditBox.
Revert the last entry.	Esc	If the user has not entered data, the TextBox is closed.
Close the Textarea, discarding changes.	Esc	If the user has entered data, the Esc must be pressed two times; the first time the data will be reverted; the second time the TextBox will close.

Textarea with autoSave specified and the Textarea is open:

Action	Key	Comments
Navigate to the next widget in the tab order.	Tab (press twice in Firefox - see the Known Issues below)	The data is saved and the widget closes.

Navigate to the prior widget in the tab order.	Shift+Tab	The data is saved and the widget closes.
Enter a newline into the text.	Enter	There is no equivalent to the Enter key behavior of TextBoxes. The user would have to use something like Tab and Shift + Tab.
Revert the last entry.	Esc	If the user has not entered data, the Textarea is closed.
Close the Textarea, discarding changes.	Esc	If the user has entered data, the Esc must be pressed two times; the first time the data will be reverted; the second time the Textarea will close.

TextBox without autoSave specified, the TextBox is open, keyboard focus is in the edit field:

Action	Key	Comments
Navigate to the Save or Cancel button.	Tab	Focus changes to the Save button if the data has been changed, otherwise it moves to the Cancel button.
Navigate to the prior widget in the tab order.	Shift+Tab	The TextBox remains open.
Close the TextBox, saving changes.	Tab to the Save button, then press the Enter key	Keyboard focus is on the closed InlineEditBox.
Revert the last entry.	Esc	If the user has not entered data, the Esc key is ignored.
Close the Text Box, discarding changes.	Tab to the Cancel button, then press the Enter key.	Keyboard focus is on the closed InlineEditBox.

Note: The Enter key is ignored when focus is in the edit field.

Textarea without autoSave specified, the Textarea is open, keyboard focus is in the edit field:

Action	Key	Comments
Navigate to the Save or Cancel button.	Tab (press twice in Firefox - see the Known Issues below)	Focus changes to the Save button if the data has been changed, otherwise it moves to the Cancel button.
Navigate to the prior widget in the tab order.	Shift+Tab	The Textarea remains open.
Close the Textarea, saving changes.	Tab to the Save button, then press the Enter key	Keyboard focus is on the closed InlineEditBox.
Revert the last entry.	Esc	If the user has not entered data, the Esc key is ignored.
Close the Textarea, discarding changes.	Tab to the Cancel button, then press the Enter key.	Keyboard focus is on the closed InlineEditBox.

Note: Pressing the Enter key results in a newline being inserted into the edit field.

Known Issues

- See the Comment for the Enter key in the information for autoSaving Text Areas above.
- Ticket 3910: When Inline Text Boxes are opened, all the text should be selected.
- On Firefox 2, the user must press the Tab key twice with focus in an textarea before keyboard focus moves to the next widget. This is a permanent restriction on Firefox 2. This is because the Dojo text area is implemented using the Firefox editor component in an iframe. This editor component implements usage of the tab key within the editor to indent text and shift-tab to outdent text. There is no keyboard mechanism in Firefox to move focus out of the editor. So, the dijit editor traps the tab key in the editor and sets focus to the editor iframe. From there pressing tab again will move to the next focusable item after the editor.

Screen Reader Issues

The InlineEditBox is implemented as a button. Since these are intended to be used "in-line" within text there is often no label element associated with the underlying control. For this reason, developers are encouraged to add a title attribute to InlineEditBoxes. The Window-Eyes screen reader will speak the title as part of the button description. JAWS has an option to speak different attributes on an button. A JAWS user may need to use the insert-v command to modify the behavior to speak the button title when working with Dojo InlineEditBoxes.

Editor

Dijit's Rich Text editor, Dijit.Editor, is a text box on steroids. Designed to look and work like a word processor. The editor features a toolbar, HTML output, and a plugin architecture that supports new commands, new buttons and other new features.

Note: Editor is currently in development. Since we had problems getting an example to work over CDN, this example requires locally-installed Dojo. See Quick Installation for details. These problems should be worked out in the 1.0 time frame. (We use /dojoroot here, but you can substitute your own Dojo root directory with no problems.)

Examples

The default configuration of Dijit.Editor has a toolbar along the top with icons for Copy, Cut, Paste, Bold, Italic, Underline, Numbered and Bulleted Lists, and Indent Left and Right.

It was a dark and stormy night. Your story belongs here!

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Editor Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
        <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.Editor");
     </script>
</head>
<body class="tundra">
<form method="post">
<textarea name="field" width="200px" dojoType="dijit.Editor">
        It was a dark and stormy night.  Your story belongs here!
</textarea>
<input type="submit" value="Save" />
</form>
</body>
</html>
```

Toolbar Customization

You may limit the commands people can perform, such as removing the Indent Left button. Each button corresponds to a plugin, hence you specify toolbar buttons with the plugins attribute. For example, the following toolbar includes copy, cut, paste and bold: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<textarea dojoType="dijit.Editor" height="200"
    plugins="['copy','cut','paste','|','bold']">
</textarea>
```

The plugins attribute is specified in JavaScript array literal format. Here is a list of built-in plugins that you can reference: (names are case sensitive and those *in italics* are not yet implemented)

- **Character Formatting**: bold, italic, underline, strikethrough, subscript, superscript, removeFormat, forecolor, hilitecolor
- **Paragraph Formatting:**: indent, outdent,justifyCenter, justifyFull, justifyLeft, justifyRight, delete, selectall
- **Inserting Objects:**insertOrderedList, insertUnorderedList, createlink (use LinkDialog plugin), inserthtml
- **Misc.**: undo, redo, cut, copy, paste

dijit.Editor (Rich Text Editor)

Attributes

captureEvents	String[] []	Events that should be connected to the underlying editing area, events in this array will be addListener with capture=true
customUndo	Boolean true for IE	Whether to use custom undo/redo support instead of the native browser support. By default, only enable customUndo for IE is available, as it has broken native undo/redo support. Note: the implementation does support other browsers which have W3C DOM2 Range API.
editActionInterval	Integer 3	When using customUndo, not every keystroke is saved as a step. Instead typing (including delete) will be grouped together: after a user stop typing for editActionInterval seconds, a step will be saved; if a user resume typing within editActionInterval seconds, the timeout will be restarted. By default, editActionInterval is 3 seconds.
events	String[] ["onKeyPress", "onKeyDown", "onKeyUp", "onClick"]	Events that should be connected to the underlying editing area

focusOnLoad	Boolean false	Whether focusing into this instance of rich text when page onload
height	String 300px	Set height to fix the editor at a specific height, with scrolling. By default, this is 300px. If you want to have the editor always resize to accommodate the content, use AlwaysShowToolbar plugin and set height=""
inheritWidth	Boolean false	Whether to inherit the parent's width or (if false) simply use 100%.
isClosed	Boolean true	
isLoaded	Boolean false	
minHeight	String 1em	The minimum height that the editor requires
name	String	If a save name is specified, the content is saved and restored when the user leaves the page and returns, or if the editor is not properly closed after editing has started.
onLoadDeferred	dojo.Deferred	Deferred can be used to connect to the onLoad function. This will only be set if dojo.Deferred is required
plugins		See plugin section
styleSheets	String	Semicolon (";") separated list of .css files for the editing area.
toolbar	dijit.Toolbar DOM Node	To use your own custom toolbar
updateInterval	Number 200	Number of ms between display changes for onNormalizedDisplayChanged

Methods

addKeyHandler(/*String*/key, /*Int*/modifiers, /*Function*/handler)	Add a handler for a keyboard shortcut
addPlugin(/*String||Object*/plugin, /*Integer?*/index)	Takes a plugin name as a string or a plugin instance and adds it to the toolbar and associates it with this editor instance. The resulting plugin is added to the Editor's plugins array. If index is passed, it's placed in the plugins array at that index. No big magic, but a nice helper for passing in plugin names via markup.
addStyleSheet(/*dojo._Url*/uri)	Add an external stylesheet for the editing area
beginEditing(/* String */cmd)	Set mark for capturing undo steps
close(/*Boolean*/save, /*Boolean*/force)	Ends the editor and optionally writes back the modified contents to the element from which it originated.
endEditing(/* Boolean */ignore_caret)	Clear the undo buffer, as in after a save
exec(/* String */cmd)	Execute given command
String escapeXml(/*String*/str, /*Boolean*/noSingleQuotes)	Adds escape sequences for special characters in XML: &<>"' Optionally skips escapes for single quotes
open(/*DomNode?*/element)	Transforms the node referenced in this.domNode into a rich text editing node. This results in the creation and replacement with an iframe if designMode(FF)/contentEditable(IE) is used.
placeCursorAtEnd()	Place the cursor at the end of the editing area
placeCursorAtStart()	Place the cursor at the start of the editing area
Boolean queryCommandAvailable(/*Boolean*/ disabled)	Tests whether a command is supported by the host. Clients should check whether a command is supported before attempting to use it, behaviour for unsupported commands is undefined.
Boolean queryCommandEnabled(/* String */cmd)	Checks whether a command is enabled or not
Boolean queryCommandState(/*Boolean*/ disabled)	Checks the state of a given command
String queryCommandValue(/* String */cmd)	Checks the value of a given command
redo()	Redoes last un-done command
removeStyleSheet(/*dojo._Url*/uri)	Remove an external stylesheet for the editing area
replaceValue/*String*/html	This function sets the content while trying to maintain the undo stack (currently, this only works okay with Moz, this is identical to setValue in all other browsers)

undo()	Undo last command
Extension Points	
onChange	This is initiated if the editor loses focus and the content is changed
onDisplayChanged	This event is initiated each time the display context changes and the result needs to be reflected in the UI. If you do not want to have the update too often, use onNormalizedDisplayChanged instead
onNormalizedDisplayChanged	This event is initiated on each updateInterval ms or more
setupDefaultShortcuts()	Override to setup your own handlers. The default implementation does not use Editor commands, but directly runs the builtin commands within the underlying browser support.

Accessibility (Applies to 1.0 version of editor)

Keyboard for Editor

Action	Key
Move focus to the next widget in the tab order.	Tab (must press tab twice in some situations - see Known Issues below)
Move focus to the prior widget in the tab order (the editor toolbar)	Shift+Tab (must press shift-tab twice in some situations - see Known Issues below)

Keyboard for Editor Toolbar

Action	Key
Move focus to the next enabled button in the toolbar.	arrow right in left to right locales, arrow left in right to left locales
Move focus to the previous widget in the toolbar	arrow left in left to right locales; arrow right in right to left locales.

The arrow keys will not work within any optional drop down lists such as ComboBox or FilteringSelect in the editor toolbar until the drop down list of choices has been activated. Use the backspace or escape key to clear the current selection in the textbox associated with the drop down. When the list of choices is not activated, the arrow keys will move between toolbar buttons rather than within the combobox or select.

Known Issues

- On Firefox 2, the user must press the Tab key twice before keyboard focus moves to the next widget. This is a permanent restriction on Firefox 2. The reason for this is because Firefox implements usage of the tab key within the editor to indent text and shift-tab to outdent text. There is no keyboard mechanism in Firefox to move focus out of the editor. So, the dijit editor traps the tab key in the editor and sets focus to the editor iframe. From there pressing tab again will move to the next focusable item after the editor. When shift-tab is pressed within the editor, focus is set to the toolbar associated with the editor (currently there is always a toolbar defined for a dijit editor). Some people are unhappy with the loss of the tab key functionality within the editor so future versions may have an option to allow the use of tab and shift-tab within the editor to indent and outdent text.
- In IE6 or 7 when the editor has been created from a textarea the user must press tab twice to set focus into the editor to begin inserting or editing text. Likewise, with focus within editor text the user must press shift-tab twice to set focus back to the toolbar.

Plugins

A dijit.Editor plugin adds more features to the Editor widget. It may correspond to one "verb", or command, or it may add other features, such as special enter key handling. The built-in verbs of dijit.Editor (see the list in , like copy/cut/paste and indent/outdent, are implemented inside the editor code. You can add your own by adding Dojo classes with some special properties.

Plugin	Description
AlwaysShowToolbar	Set to true to always display toolbar. No corresponding button.
EnterKeyHandling	Make enter key handling consistent across browsers. The blockNodeForEnter property decides the behavior of Enter key. It can be either P, DIV, BR, or empty (which means disable this feature). Anything else will trigger errors. No corresponding button.
FontChoice	Adds font drop down.
LinkDialog	Provides a dialog for inserting URLs
TextColor	Adds text color button

Setup Clipboard actions for FF

To protect users' private information, unprivileged scripts cannot invoke the Cut, Copy, and Paste commands in the Mozilla rich text editor, so the corresponding buttons on the dijit Editor widget will not work. To enable these functions for purposes of the demo, you must modify your browser preferences.

Configure Firefox

1. Quit Firefox. If you have Quick Launch running (in Windows, an icon in the toolbar), quit that too.
2. Find your Firefox profile directory. On Windows, this is often located in

   C:\Documents and Settings\<Windows login>\Application Data\Mozilla\Firefox\Profiles\<one folder>.

   (See also editing configuration files for more info on locating your profile folder.)

3. Open the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
   .javascript .imp {font-weight: bold; color: red;}
   .javascript .kw1 {color: #000066; font-weight: bold;}
   .javascript .kw2 {color: #003366; font-weight: bold;}
   .javascript .kw3 {color: #000066;}
   .javascript .co1 {color: #009900; font-style: italic;}
   .javascript .coMULTI {color: #009900; font-style: italic;}
   .javascript .es0 {color: #000099; font-weight: bold;}
   .javascript .br0 {color: #66cc66;}
   .javascript .st0 {color: #3366CC;}
   .javascript .nu0 {color: #CC0000;}
   .javascript .me1 {color: #006600;}
   .javascript .re0 {color: #0066FF;}
   `user.js` file from that directory in a text editor. If there's no /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
   .javascript .imp {font-weight: bold; color: red;}
   .javascript .kw1 {color: #000066; font-weight: bold;}
   .javascript .kw2 {color: #003366; font-weight: bold;}
   .javascript .kw3 {color: #000066;}
   .javascript .co1 {color: #009900; font-style: italic;}
   .javascript .coMULTI {color: #009900; font-style: italic;}
   .javascript .es0 {color: #000099; font-weight: bold;}
   .javascript .br0 {color: #66cc66;}
   .javascript .st0 {color: #3366CC;}
   .javascript .nu0 {color: #CC0000;}
   .javascript .me1 {color: #006600;}
   .javascript .re0 {color: #0066FF;}
   `user.js` file, create one.
4. Add these lines to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
   .javascript .imp {font-weight: bold; color: red;}
   .javascript .kw1 {color: #000066; font-weight: bold;}
   .javascript .kw2 {color: #003366; font-weight: bold;}
   .javascript .kw3 {color: #000066;}
   .javascript .co1 {color: #009900; font-style: italic;}
   .javascript .coMULTI {color: #009900; font-style: italic;}
   .javascript .es0 {color: #000099; font-weight: bold;}
   .javascript .br0 {color: #66cc66;}
   .javascript .st0 {color: #3366CC;}
   .javascript .nu0 {color: #CC0000;}
   .javascript .me1 {color: #006600;}
   .javascript .re0 {color: #0066FF;}
   `user.js`:

   ```
   user_pref("capability.policy.policynames", "allowclipboard");
   user_pref("capability.policy.allowclipboard.sites", "http://localhost file:// http://dojotoolkit.org");
   user_pref("capability.policy.allowclipboard.Clipboard.cutcopy", "allAccess");
   user_pref("capability.policy.allowclipboard.Clipboard.paste", "allAccess");
   ```

   *Change the value of capability.policy.allowclipboard.sites to where you want to enable this function. In the example above, this function is enabled for localhost, accessing local files directly (in your own hard drive, without a http server) and dojo website.

5. Save the file, and restart Firefox. The Clipboard buttons should now function.

Note: The preference is site as well as protocol specific. For example

```
user_pref("capability.policy.allowclipboard.sites", "http://dojotoolkit.org")
```

is not the same as

```
user_pref("capability.policy.allowclipboard.sites", "https://dojotoolkit.org")
```

(the first is http and the second is https)

For more information about security policies, see http://www.mozilla.org/projects/security/components/ConfigPolicy.html.

This page is modified from Setting Prefs for the Mozilla Rich Text Editing Demo.

Tree

[inline:tree.png]

The trees we see in User Interfaces help sort out long, heirarchical lists. A file system is the classic example, with Windows using it in Explorer and Macintoshes with its folder windows. The Dijit tree widget is like that. The Tree widget itself is simple, but the real power comes in the data you pass - this represents the heirarchical structure of the tree. This data is fed by the powerful dojo.data API.

Dojo makes easy trees easy, and hard trees possible. In particular, you can:

- Build rooted or rootless trees (forests)
- Nest trees to an arbitrary depth ... each branch is independently expandible
- Apply different icons to different leaf or branch classes
- Connect your tree to any dojo.data store implementing the Identity API.
- Attach code to events. Events fire when users expand, contract or click particular nodes.
- Programmatically build trees. Add, remove or disable nodes programatically.
- Manipulate the Dojo.data store directly, which manipulates the tree indirectly
- Allow nodes to be dragged and dropped through the familiar Dojo DnD API.

A Simple Tree

To start, here's a one level tree. We split the tree data off into a separate file, poptarts.txt, in JSON format. You will need to download this and save it in the same directory as the example HTML below:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ label: 'name',
  identifier: 'name',
  items: [
     { name:'Fruit', type:'category'},
     { name:'Cinammon', type: 'category'},
     { name:'Chocolate', type: 'category'}
  ]
}
```

Then here's the HTML that draws the actual tree:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/resources/dojo.css";
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script>
        dojo.require("dojo.data.ItemFileReadStore");
        dojo.require("dijit.Tree");
        dojo.require("dojo.parser");
   </script>
</head>
<body class="tundra">
        <div dojoType="dojo.data.ItemFileReadStore"
             url="poptarts.txt" jsid="popStore" />
        <div dojoType="dijit.Tree" store="popStore" labelAttr="name" label="Pop Tarts"></div>
</body>
</html>
```

This is a good start, but there's a lot more you can do with Tree. Read on!

Trees and Dojo.Data

To desconstruct the previous example, we need some dojo.data terminology:

- an *attribute* is a named data set, much like a field or a column of a database,
- A *value* is the data itself. So in "name: 'Fruit'", name is the attribute, Fruit is the value.
- an *item* is one set of related attributes, much like a record or row. Unlike pure relational tables, items can have items nested within.
- an *identifier*is an attribute that uniquely identifies the item, like a primary key

dijit.Tree, conforming to the dojo.data spec, expects the following in its data store:

- The store attribute must be the name of a dojo.data data store.
- Each item in the data store must have a label (as returned by getLabel())
- An identifier attribute must be specified, even though it's not displayed.

So now let's take a step further. There are two methods to nesting a tree, corresponding to the two heirarchical methods in dojo.data.

Heirarchical Data

The easiest method for fixed text is to nest the items in the data store. To add a subtree, you must add a children attribute to the parent item, then add the child items to the children attribute. So for our previous example, we can add nodes under the Cinammon node:

Method 1: Direct Nesting

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ label: 'name',
  identifier: 'name',
   items: [
     { name:'Fruit', type:'category'},
     { name:'Cinammon', type: 'category',
       children: [
          { name:'Cinammon Roll', type:'poptart' },
          { name:'Brown Sugar Cinnamon', type:'poptart' },
          { name:'French Toast', type:'poptart' }
       ]
     },
     { name:'Chocolate', type: 'category'}
  ]
}
```

By downloading this file into poptarts.txt, you can use the same HTML as our previous example. And voila!

Method 2: References

Direct Nesting is a little inconvenient for relational table data. So Tree supports references, where all the data nodes are at the same level, but nesting occurs with psuedo pointers to child nodes. So our example above is written: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ label: 'name',
  identifier: 'name',
   items: [
     { name:'Fruit', type:'category'},
     { name:'Cinammon', type: 'category',
        children: [
           {_reference: 'Cinammon Roll'},
           {_reference:'Brown Sugar Cinnamon'},
           {_reference:'French Toast'}
        ]
     },
     { name:'Cinammon Roll', type:'poptart' },
     { name:'Brown Sugar Cinnamon', type:'poptart' },
     { name:'French Toast', type:'poptart' },
     { name:'Chocolate', type: 'category'}
    ]
}
```

Like our nested children example, the parent node requires a children attribute. But instead of actual items, you place reference objects linking to the identifier of another object. This requires a small change to the Tree tag:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.Tree" store="popStore2"
     labelAttr="name" label="Pop Tarts"
     query="{type:'category'}"></div>
```

The query is necessary to choose only the top level items from the store. The menu produced is exactly the same:

Icon Classes

User Actions

The problem is our tree does nothing but stand around looking beautiful. Nothing wrong with that. Normally, though, you'd want some kind of actions.

Events

At the very least, you probably want to do something when a user clicks or press [ENTER] on a node. To do this, you can use the onClick event.

Pick a Pop Tart Please
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:

```
bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;}
.geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;}
.geshifilter .sc2 {color: #009900;}
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script>
        dojo.require("dojo.data.ItemFileReadStore");
        dojo.require("dijit.Tree");
        dojo.require("dojo.parser");
   </script>
</head>
<body class="tundra">
        <div id="response">Pick a Pop Tart Please</div>
        <div dojoType="dojo.data.ItemFileReadStore"
             url="poptarts_direct.txt" jsid="popStore"></div>
        <div dojoType="dijit.Tree" store="popStore" labelAttr="name"
             label="Pop Tarts">
             <script type="dojo/method" event="onClick" args="item">
                dojo.byId("response").innerHTML =
                    "You're a " + popStore.getLabel(item) + " fan, eh?";
            </script>
        </div>
</body>
</html>
```

Alternatively, you can use Dojo's publish/subscribe event system. When a node is clicked, the tree id is sent as the topic along with the message:

- tree: the actual tree widget
- event: "execute"
- item: the dojo.data item selected
- node: the DOM node selected

Drag and Drop

Dojo trees are great, and so is Dojo Drag And Drop (DnD). But together, they're unstoppable! Not being satisfied with our selection of Pop Tarts, we'll create a pool of Drag and Drop sources to add:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:
bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;}
.geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;}
.geshifilter .sc2 {color: #009900;}
<ul dojoType="dojo.dnd.Source">
    <li class="dojoDndItem" id="Hot Chocolate">Hot Chocolate</li>
    <li class="dojoDndItem" id="Blueberry">Blueberry</li>
</ul>
```

The user should only be allowed to drop a Pop Tart on its home category - for example, Blueberry should only go under Fruit. To handle this, we use JavaScript regular expressions to make an intelligent guess. We add these to the data store:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:
bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3
{color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0
{color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
{ label: 'name',
  identifier: 'name',
   items: [
     { name:'Fruit', type:'category', regexp:'.*erry$'},
     { name:'Cinammon', type: 'category', regexp:'[Cc]innamon',
       children: [
          { name:'Cinammon Roll', type:'poptart' },
          { name:'Brown Sugar Cinnamon', type:'poptart' },
          { name:'French Toast', type:'poptart' }
       ]
     },
     { name:'Chocolate', type: 'category', regexp:'([Cc]hocolate|Fudge)'}
  ]
}
```

You can download this file below. Next we wire in DnD to the Tree. This is as simple as specifying two attributes: the controller and the acceptance checker extension point, which we'll write shortly.

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:
bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;}
.geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;}
.geshifilter .sc2 {color: #009900;}
<div dojoType="dijit.Tree" store="popStore" labelAttr="name"
     label="Pop Tarts" jsid="ptTree"
     dndController="dijit._tree.dndSource"
     checkItemAcceptance="poptartCheckItemAcceptance">
```

The checkItemAcceptance extension point function is called each time a drop target is entered. In Tree's case, every node is a drop target.

How do we know it's valid? The function must check the dragged node to the regular expression of the drop node:

```
function poptartCheckItemAcceptance(node,source) {
     // Get the associated dojo.data item for the target
     item = dijit.getEnclosingWidget(node).item;
     // Need to check for item because when dropping on a root node,
     // item === null
     if (! item) return false;

     ptType = ptTree.store.getValue(item,"type");

     if (ptType == 'category') {
         // We make intelligent guesses about the correct folder
         re = new RegExp(ptTree.store.getValue(item,"regexp"));
         okToMove = true;
         for (var itemId in source.selection) {
              console.debug(itemId+" tested against "+re.toString());
              okToMove &= re.test(itemId);
         }
         return okToMove;
     }
     else
         return false;
}
```

So now all the pieces are in place, yielding:

Drag a Pop Tart to Its Category
Hot Chocolate
Blueberry

And here's the full source code, which is downloadable below:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script>
        dojo.require("dojo.data.ItemFileWriteStore");
        dojo.require("dijit.Tree");
        dojo.require("dojo.parser");
        dojo.require("dojo.dnd.Source");
        dojo.require("dijit._tree.dndSource");

        function poptartCheckItemAcceptance(node,source) {
            // Get the associated dojo.data item for the target
            item = dijit.getEnclosingWidget(node).item;
            // Need to check for item because when dropping on a root node,
            // item === null
            if (! item) return false;

            ptType = ptTree.store.getValue(item,"type");

            if (ptType == 'category') {
                // We make intelligent guesses about the correct folder
                re = new RegExp(ptTree.store.getValue(item,"regexp"));
                okToMove = true;
                for (var itemId in source.selection) {
                   console.debug(itemId+" tested against "+re.toString());
                   okToMove &= re.test(itemId);
                }
               return okToMove;
            }
            else
                return false;
        }
   </script>

</head>
<body class="tundra">
        <div>Drag a Pop Tart to Its Category</div>
        <div dojoType="dojo.dnd.Source">
           <div class="dojoDndItem" id="Hot Chocolate">Hot Chocolate</div>
           <div class="dojoDndItem" id="Blueberry">Blueberry</div>
        </div>
        <div dojoType="dojo.data.ItemFileWriteStore"
             url="poptarts_dnd.txt" jsid="popStore" />
        <div dojoType="dijit.Tree" store="popStore" labelAttr="name"
             label="Pop Tarts" jsid="ptTree"
             dndController="dijit._tree.dndSource"
             checkItemAcceptance="poptartCheckItemAcceptance">
        </div>
</body>
```

```
</html>
```

Scripting Trees

In Actions we saw how to hook a piece of code into onClick. That's not all. With JavaScript and a Tree, you can style and manipulate trees in all sorts of combinations.

Adding an Icon or Class to a Node

OnClick is an *extension point*, which we'll cover in detail in Part 3. Three other extension points are used in drawing the tree nodes:

- **String getIconClass(/* dojo.data.Item */ item)** takes in an item and returns a String specifiying a CSS class for the icon. The class should have a CSS style specifying a background url, pointing at the image. We saw this in Example 2 for styling mail icons.
- **String getLabelClass(/*dojo.data.Item*/ item)** returns a CSS class applied to a label.
- **String getLabel(/*dojo.data.Item*/ item)** returns an actual label. This is useful when the data source label is not sufficient for display - e.g. displaying "First Name Last Name" when the data source label is the SSN.

Here is an example of coloring our Pop Tarts labels. Notice the loose coupling here. If another category of Pop Tarts are introduced in the data source, you only need to add a corresponding class to poptarts.css: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}

```
.Cinammon {
    color:red;
}
.Chocolate {
   color:brown;
}
.Fruit {
   color: blue;
}
```

And the getLabelClass is straightforward:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.css";
        @import "poptarts.css";
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script>
        dojo.require("dojo.data.ItemFileReadStore");
        dojo.require("dijit.Tree");
        dojo.require("dojo.parser");
   </script>
</head>
<body class="tundra">
        <div dojoType="dojo.data.ItemFileReadStore"
             url="poptarts_direct.txt" jsid="popStore" />
        <div dojoType="dijit.Tree" store="popStore" labelAttr="name"
             label="Pop Tarts">
               <script type="dojo/method" event="getLabelClass" args="item">
               if (item != null && popStore.getValue(item, "type") == 'category') {
                   // For name=Chocolate, return class Chocolate etc.
                       return popStore.getValue(item, "name");
               }
            </script>
        </div>
</body>
</html>
```

Adding, Removing and Disabling Nodes

Tree Tag

dijit.Tree, dijit._TreeNode
dijit.Tree is a container for a hierarchical list with expandable and collapsible items. dijit._TreeNode's are the items themselves. _TreeNodes are almost never created with markup, and in general you don't deal with them.

Attributes

childrenAttr	String	name of attribute that holds children of a tree node consider this "root node" to be always expanded
label	String	**New in 1.0** label for the top node of the tree, if desired. Note there is no actual item associated with this node.
query	String	get top level node(s) of tree (ex: {type:'continent'})
store	dojo.data.Store	The store to get data to display in the tree
Methods		
isExpanded	if expandible, returns true if children are displayed	
isExpandible	returns true if node can be expanded (has an expando icon next to it)	
Extension Points		
getIconClass	user overridable class to return CSS class name to display icon	
getItemChildren	User overridable function that return array of child items of given parent item, or if parentItem==null then return top items in tree	
getItemParentIdentity	User overridable function, to return id of parent (or null if top level). It's called with args from dojo.store.onNew	
getLabel	user overridable function to get the label for a tree node (given the item)	
mayHaveChildren	**New in 1.0** User overridable function to tell if an item has or may have children. Controls whether or not +/- expando icon is shown. (For efficiency reasons we may not want to check if an element has children until user clicks the expando node)	
onClick(item, node)	Called when someone clicks a tree item	

Accessibility

Keyboard

Action	Key
Navigate to first tree item*	Tab
Navigate to the next sibling	Down arrow
Navigate to the previous sibling	Up arrow
Open a subtree	Right arrow
Close a subtree	Left arrow
Navigate to open subtree	Right arrow
Navigate to parent	Left arrow
Activate a tree item	Enter

* Note: The last tree item focused will be in the Tab order.

Themes and Design

Dijit Themes lend a consistent look and feel to widgets. We've been using the Tundra theme until this point, but Dijit comes bundled with others as well.

Tundra

[inline:theme_tundra.png]

Soria

[inline:theme_soria.png]

Noir

[inline:theme_noir.png]

In this section, we'll review some theme features and learn how to override them for those occasional little tweaks. But you're not limited to the bundled themes - you can write your own as well, and we've devoted a page to that.

Common Elements

If you look at /dijit/themes/tundra, you can see that a theme is just a bunch of CSS and images:

```
themes/
   tundra/
       tundra.css       <-- all the CSS for all the widgets
       images/
          checkbox.gif  <--- all the checkbox and radio button images
          fader.gif     <--- background image referenced by tundra.css
```

The foreground images are located in the images directory (along with background images) and are referenced from the widget via CSS rules (via the background-image property of a dummy node).

The tundra.css file has rules like:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}
```
.tundra .dojoButton { ... }
```

We've been using Tundra in all of our examples, but to use an alternate theme for all widgets on your page, do:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<style type="text/css">
        @import "http://o.aolcdn.com/dojo/0.9.0/dijit/themes/themename/themename.css";
        @import "http://o.aolcdn.com/dojo/0.9.0/dojo/resources/dojo.css"
    </style>
```

Then add a theme name to the <body> element, like:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<body class="themename">
```

The reason the tundra theme requires a class=tundra on the body tag (or some other tag) is because the rules all list a hierarchy like ".tundra .dojoButton".

Overriding and Combining Themes

Using multiple themes

Dijit includes Noir and Soria themes as well as Tundra. You will be able to include additional CSS files into your document, like:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<style type="text/css">
    @import "http://o.aolcdn.com/dojo/0.9.0/dijit/themes/noir/noir.css";
    @import "http://o.aolcdn.com/dojo/0.9.0/dojo/resources/dojo.css"
</style>
```

noir.css will define rules like:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}
```
.noir .dijitButton { ... }
```

so it won't conflict with tundra.css.

To have different sections of your document that are different themes, you just change the class of each section. For example, to make the main document tundra theme, but then have sections that are noir and soria theme, do:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<body class=tundra>
...
        <div dojoType="dijit.layout.TabContainer">
                <div dojoType="dijit.layout.ContentPane" label="Tab1" class="noir">
                        <input dojoType="dijit.form.TextBox">
                        <button dojoType="dijit.form.Button">Noir Button</button>
                        ...
                </div>
                <div dojoType="dijit.layout.ContentPane" label="Tab2" class="soria">
                        <input dojoType="dijit.form.TextBox">
                        <button dojoType="dijit.form.Button">Soria Button</button>
```

```
                        ...
                </div>
        </div>
...
</body>
```

All the widgets in the first tab will have the Noir theme and all the widgets in the second tab will have the Soria theme.

Overriding a theme

You can also define a variation on a theme (much like Handel). Let's say that you like the tundra theme but for each tab above, just want to change the background color of the form widgets. You would define yellowForm and blueForm to just change the background color:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}

```
.yellowForm .dijitButton, .yellowForm .dijitInputField { background-color: yellow; }
 .blueForm .dijitButton, .blueForm .dijitInputField { background-color: blue; }
```

Then you would reference the override class in a similar way to above:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.TabContainer">
        <div dojoType="dijit.layout.ContentPane" label="Tab1" class="yellowForm">
                <input dojoType="dijit.form.TextBox">
                <button dojoType="dijit.form.Button">Yellow Button</button>
                ...
        </div>
        <div dojoType="dijit.layout.ContentPane" label="Tab2" class="blueForm">
                <input dojoType="dijit.form.TextBox">
                <button dojoType="dijit.form.Button">Blue Button</button>
                ...
        </div>
</div>
```

The two tabs would then be tundra theme except for the background color on form fields

Writing Your Own Theme

If you want to develop your own theme just make rules like

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}

```
.myTheme .dojoButton { ... }
```

and include them into your page.

The dijit.form.Checkbox widget displays the checkbox image using an <img> tag. However, it grabs the image location from the CSS.

The class names used on widgets do not change based on the theme, although they will change based on the state of the widget. For example, an input field will have class="dojoInputField", but a disabled input field will have class= "dojoInputField dojoInputFieldDisabled"

Applying style directly to a widget

You can apply styles to plain dom nodes in various ways:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div style= "margin: 30px;">...
<style>
   #xyz { margin: 30px; }
</style>
<div id= "xyz">
<style>
   .myClass { margin: 30px; }
</style>
<div class= myClass>
```

The first two techniques should work for widgets also. However, the third technique is not likely to work, because some of style rules like ".tundra .dojoButton", will take precendence.

Accessibility

It is important that all users can access applications built using Dojo. People with disabilities access the Web in many manners. Some people rely on the keyboard only to navigate and make selections. Others use custom font and color settings, custom style sheets, screen

magnification or screen readers to interact with the computer. With the addition of multimedia to the Web, captioning is important for folks with hearing loss. Those with cognitive disabilities may use a variety of assistive technology to adapt the content to a particular learning style.

In addition, many countries require that Web content is accessible to all. Section 508 of the Disabilities Act requires the US Government to purchase accessible technology. Australia, Japan, Great Britain and the European Union all have accessibility regulations. Thus, it is important for the Dojo Toolkit to provide a way to build accessible applications.

Web Accessibility Issues

The main accessibility concern for the Dojo Toolkit is the widget set. The widgets need to be usable by all users to create accessible applications using Dojo. The main issues are with color, device independent interactions and providing semantic information about these user interface (UI) components. The core widgets in Dojo 0.9, those found in the dijit subsystem, will be fully accessible for the 1.0 release.

Color/CSS

There are several issues with color. Some people have vision problems, which prevent them from seeing certain colors or may need a high contrast between foreground and background colors. These people may adjust the operating system colors to meet their needs or use the high contrast settings provided by the operating system. In some cases, when a person can not distinguish the colors in an image, they may turn off images in the browser and rely on the description of the image provided through a text alternative. In HTML, the text alternative is provided by using the alt attribute of the img element.

In order to create high performance, easily styled and visually appealing widgets, dijit makes use of CSS background images to create a desktop-like look and feel. Without additional work, these widgets would not work in high contrast mode. Selecting high contrast mode in the Windows operating system forces all color and image related CSS information to be turned off in the browser. All positioning information is retained but the colors and background images are turned off. Dojo widgets that create the visual effects using CSS background-images without providing text alternatives are not visible in high contrast mode. To accommodate high contrast mode, the dijit widgets rely on CSS background images to convey information contain a hidden text alternative element, which is made visible when high contrast mode is detected. This text alternative also accommodates users who may turn images off in the browser.

Device Independent Interactions

Not all users are able to interact with the computer using a mouse, thus, device independent interaction is important. This means that, at a minimum, keyboard interactions must be supported. The dijit widgets must not work solely via the mouse. For HTML links and form controls, the keyboard is automatically supported. But, since dijit is creating custom widgets via DHTML and scripting of elements other than links and form controls, keyboard event handling must be added. Dojo widgets created using technologies other than HTML such as SVG, must also support the keyboard or provide an alternative interface that works with the keyboard.

Detailed information about role and state

Accessibility application programming interfaces (APIs) for desktop graphical user interface (GUI) frameworks define a standard contract between an application component and an assistive technology (AT). The information about the type of component and its current state is provided to the AT via the accessibility programming interface. Examples of accessibility APIs are Java Accessibility API , Microsoft Active Accessibility (MSAA), Apple Accessibility for COCOA, and the Gnome Accessibility Toolkit (ATK). In the browser environment, certain HTML elements have well defined roles and states. Examples include lists, links and form elements. The browser communicates the information about these elements and the current state such as checked, unchecked, readonly, disabled, visited, etc. to assistive technology via the accessibility APIs. Now that user interface components are being created via scripted HTML elements such as

and , the assistive technology needs additional information about the created component and its behaviors.

The W3C Web Accessibility Initiative

Accessible Rich Internet Applications (WAI-ARIA) Roadmap provides specifications that describe how to provide this additional information to the assistive technology. These specifications define a set of roles and states that can be added to the created DHTML user interface components. With the addition of this information and support by the browser and assistive technology, a user of AT can obtain detailed information about the user interface components created for the Web. For example, a tree component is identified as a tree and each tree item, its level in the tree, expanded and collapsed state and number of children is now available to a screen reader user. The current ARIA specifications are supported in Firefox 1.5 and later versions as well as the by the Window-Eyes screen reader beginning in version 5.5 and partial support by JAWS beginning in version 7.1. Support from other Windows browsers and assistive technology vendors are expected in the future. To provide the most comprehensive accessibility solution for Dojo, the ARIA techniques are being applied to the dijit widgets.

Dojo Accessibility Strategy

Support High Contrast /Images Off

Customizing the dijit Widgets look and feel is very important, as is performance. The Dojo team uses background images in the creation of the widgets because it allows the look and feel of the widgets to be easily customized by modifying the CSS. In addition, using CSS allows several images to be combined into one file and then the proper subset of the image to displayed using positioning. Thus all of the images for a particular widget element can be retrieved via one HTTP request rather than a separate request for each separate image file. For these reasons, Dojo did not want to require the use of image elements when creating widgets. This presents a problem for users with high contrast mode settings because in this mode, background images are not displayed.

This problem is solved in dijit by checking for high-contrast mode when the widgets get loaded. If high-contrast mode is detected, an additional style sheet is added to the body element of the page. This is NOT a style sheet to provide a set of high-contrast mode colors and styles – it is an enhancement to the existing style sheet to enable text equivalents for the items that rely on CSS images.

For example, in the default Dojo theme called tundra, the close icon for a dialog box is represented using a CSS background image of an x within a shaded circle which is displayed in the upper right hand corner of the dialog box.

Within the dialog template there is an additional span that contains the character 'x' to serve as the text alternative for the background image icon. This span has a style of .closeText which is defined as follows:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}

```
.dijitDialog .closeText {
        display:none;
        position:absolute;
}
```

Notice that the closeText style is encapsulated within .dijitDialog so this style applies when it is cascaded within an element which has the .dijitDialog style applied. The span with .closeText applied is set to display:none. The dijit.css style sheet contains an additional style:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}

```
.dijit_a11y .dijitDialog .closeText {
        display:inline;
}
```

When high-contrast mode is detected the .dijit_a11y class is applied to the body element of the page and now the span with class .closeText is made visible via the display:inline directive. In high-contrast mode, the background image of the close icon is no longer visible but the character 'x' is displayed.

In some cases, an HTML entity character such as, ▼ - the Unicode black down pointing triangle (▼) is used as the text alternative. This character is used to provide the down arrow character to indicate a popup is available. This provides some issues for screen readers that may not correctly speak the HTML entity character, but the ARIA describedby and labelledby attributes are used to provide the correct text description for the character when necessary.

The high contrast detection code is in dijit._base.wai.js in the onload function. This function is run before the widgets are created to detect if a Windows system running Internet Explorer or Firefox is in high contrast mode. This function can also detect if images are turned off in Firefox when running via http (images are not turned off in Firefox when running from the file system even if Tools Options Content Load images automatically has been unchecked).

The high contrast detection code uses scripting to create an element, set specified style attributes on the element and append it to the body element. The styles assigned are two different border colors to the top and side borders, a background image, and absolute positioning to render it off screen. Next the computed style for the element is obtained and the border colors are compared. If high contrast mode is turned on, the top and right border colors will both be the same color since high contrast mode overrides CSS specified colors for the high contrast theme color.

Image detection works by examining the background image for the element. If images are turned off the background image style will "none" in IE or "url(invalid-url:)" in Firefox. Note however, that image off is only detected in IE when it is already in high contrast mode. Additional work is underway to determine images off mode in Internet Explorer in all cases.

Support Device Independent Interaction

One way of providing keyboard support in HTML is to use form and list elements which can accept keyboard focus by default. The user can use the tab key to navigate to these types of elements. The problem is that building sophisticated widgets using these elements in not practical. And, navigating via only the tab key can be very tedious. The solution is to provide full keyboard support within the widgets using additional keystrokes such as the arrow keys to provide more intuitive navigation.

One of the keys to supporting the keyboard is to allow focus to be set to any element. The tabindex attribute can be used to include additional elements in the tab order and to set programmatic focus to them. This was a feature implemented in Internet Explorer that has been extended to Firefox and Mozilla. The following table outlines the use of the tabindex attribute:

Tabindex Attribute Value	Focusable via mouse or scripting via element.focus()	Tab Navigation

not present	Follow default behavior of element (only form controls and anchors receive focus)	Follows default behavior of element
zero - tabindex="0"	Yes	In tab order relative to element's position in document
positive - tabindex="x" (where x is a positive integer between 1 and 32768)	Yes	tabindex value directly specifies where this element is positioned in the tab order
negative - tabindex="-1"	Yes	Not in tab order, author must focus it with element.focus() as result of a key press.

Adding a tabindex of -1 to an element allows the element to receive focus via JavaScript using the element.focus() method. This is used to allow arrow key navigation to elements. Each element that can be navigated to via arrow keys must have a tabindex of -1 to allow it to receive focus. A keydown event handler can determine the next object to receive focus and call that element's focus() method. In addition, the style of the element may need to be updated in order to show the focus as brower's are inconsistent in displaying focus for items that receive focus programmatically.

In order to assist with key event handling, an onkey event has been added to Dojo to normalize key events. The appropriate key event, either onkeydown or onkeypress, will be used depending upon the browser. The key codes have been normalized as well. See dojo.event.browser class in dojo.event.browser.js. In addtion, there is a special onDijitClick event implemented in the dijit system to provide support for a mouse click, Enter key press or Spacebar key press to invoke an action. By subscribing to the onDijitClick dijit event, the provided handler will be called when a click, enter key or space key is received allowing the developer to easily support both mouse and keyboard. This event is utilized by the core dijit widget set and is available to developers building custom widgets.

Implement ARIA Specification

ARIA techniques (described in the Web Accessibilty section) allow creating sophisticated UI components using scripting which can be identified to assistive technology. In the future, user agents can also make use of this information to provide additional visual clues about components as well. For example, client side validation of a text entry component that was marked using the ARIA invalid attribute could be visually identified by the browser rather than requiring the developer to provide a specific style or text identification on the component.

The ARIA information is being added into the dijit widgets Methods have been added into dijit to enable setting the ARIA information. The roles and states for a widget can be set via the widget template or within the widget scripting code. The details of these methods are discussed later in this document. In addition to providing the roles and states for each component, there are some architectural considerations as well. For components that represent a hierarchy, such as a tree or menu, it is important to identify parent and child relationships. For items where position or count are important it may be necessary to hierarchically group elements or identify a set of related elements as a group. In some cases there are specific ARIA roles for grouping items such as treegroup for tree items within the same level. If no specific grouping role is provided the generic group role can be used.

Dojo Accessibility Resources

dijit._base.wai.js - 0.9 Version

Functions for dealing with accessibility are found in dijit._base.wai.js. The name of the file and object is derived from the W3C Web Accessibility Initiative which is hosting the ARIA specification. The dijit.wai APIs are used to manipulate ARIA roles and properties. In addition, the onload function to detect high contrast mode is part of dijit.wai.

The dijit.wai module is provided to normalize setting the roles and states. The ARIA techniques are designed to be used with XHTML via namespaces. Since a content-type of application/xhtml+xml is required to fully support namespaces an alternate solution is needed for the most commonly supported content-type of text/html. The roles and states can be manipulated using the DOM namespace apis: getAttributeNS, setAttributeNS, and removeAttributeNS. In browsers which do not support the namespace apis, the generic attribute apis, getAttribute, setAttribute, removeAttribute, are used and namespaces are simulated.

The dijit.wai module provides the necessary mapping of namespace information and attribute apis for each of the dijit.waiNames. It contains two submodules, waiRole and waiState, each with the following variables.

- name represents the ARIA type being set, waiRole or waiState.
- namespace contains the actual namespace for the role or state information.
- alias is the pseudo namespace to be used when true namespaces are not supported.
- prefix will be added to the beginning of the value being set.

The dijit.wai methods getAttr(), setAttr(), and removeAttr() are wrappers to the appropriate attribute apis for the browser in use. These apis are called with the following parameters:

- node – the DOM node on which to make the appropriate attribute api call
- ns –this selects the appropriate dijit.wai module to use; waiRole or waiState.
- attr – the attribute name. This will be "role" when setting a role value and it will be one of the possible ARIA state attributes when specifying a state.
- value – the actual value to be set; either an ARIA role value (tree, treeitem, checkbox, tab, etc) or the value for the previously specified state name (true, false, mixed, etc).

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.wai.setAttr(/*DomNode*/node, /*String*/ ns, /*String*/ attr, /*String|Boolean*/value)
```

is used to set an ARIA role or state.
The following will set a role of treeitem onto a DOM node:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dijit.wai.setAttr( focusNode, “waiRole”, “role”, “treeitem”);
```
This example sets the state of the treeitem to expanded:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dijit.wai.setAttr( focusNode, “waiState”, “expanded”, “true”);
```
There are also dijit.wai APIs to get or remove attribute values:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dijit.wai.getAttr(/*DomNode*/node, /*String*/ ns, /*String*/ attr, /*String|Boolean*/value);
dijit.wai.removeAttr(/*DomNode*/node, /*String*/ ns, /*String*/ attr, /*String|Boolean*/value);
```

The role and state can also be set via the widget template using the waiRole or waiState prefix. Setting the role in the template is the same as setting it via scripting – the dijit.wai.setAttr() method will be called during widget instantiation. Simply add the waiRole=”actualrole” or waiState=”state-value” parameters into the template markup for the element. The element will be passed as the nodeObj into the dijit.wai.setAttr() method. The state is specified as a state name and value pair, the state is separated from the value using the hyphen character (-): state-value. The state becomes the attribute name when dijit.wai.setAttr() is called. Multiple states can be set within the template by separating the state-value pairs with a comma. This mechanism is useful when templates are used to create the objects requiring a role value and when the state is known at creation time.

Here is an example of setting the role in the diijt tree template. The domNode is given the “tree” role.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<div class="dijitTreeContainer" style="" waiRole="tree"
dojoAttachEvent="onclick:_onClick,onkeypress:_onKeyPress"
></div>
```
The role or state can also be specified via variables. This example shows an excerpt from the dijit button template that sets the tabindex, role and state on the button element. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<div class="dijit dijitLeft dijitInline dijitButton" baseClass="${baseClass}"
dojoAttachEvent="onclick:_onButtonClick,onmouseover:_onMouse,onmouseout:_onMouse,onmousedown:_onMouse">
  <div class='dijitRight'>
    <button class="dijitStretch dijitButtonNode dijitButtonContents" dojoAttachPoint="focusNode,titleNode"
     tabIndex="${tabIndex}" type="${type}" id="${id}" name="${name}" waiRole="button"
      waiState="labelledby-${id}_label">
      <div class="dijitInline ${iconClass}"></div>
        <span class="dijitButtonText" id="${id}_label" dojoAttachPoint="containerNode">${label}</span>
    </button>
  </div>
 </div>
```

---

dijit._base.wai.js - Updated for 1.0

Functions for dealing with accessibility are found in dijit._base.wai.js. The name of the file and object is derived from the W3C Web Accessibility Initiative which is hosting the ARIA specification. The dijit wai APIs are used to manipulate ARIA roles and properties. In addition, the onload function to detect high contrast mode is part of dijit.wai.

The dijit wai module is provided to normalize setting the roles and states. The ARIA techniques are designed to be used with XHTML via namespaces. Since a content-type of application/xhtml+xml is required to fully support namespaces an alternate solution is needed for the most commonly supported content-type of text/html. The roles can be set directly by prefixing the role value with "wairole". States can be manipulated using the DOM namespace apis: getAttributeNS, setAttributeNS, and removeAttributeNS. In browsers which do not support the namespace apis, the generic attribute apis, getAttribute, setAttribute, removeAttribute, are used and namespaces are simulated.

There are separate dijit methods for querying, getting, setting, and removing ARIA roles and states.

Roles

The following dijit methods will set the role onto an element. The role parameter must be one of the possible ARIA role values.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.hasWaiRole(/*Element*/ elem)
```
is used to determine if a role has been set on the element. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.getWaiRole(/*Element*/ elem)
```
is used to obtain a role that has been set on the element. It returns an empty string if no role has been set. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
diijt.setWaiRole(/*Element*/ elem, /*String*/ role)
```
assigns an ARIA role to the element. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.removeWaiRole(/*Element*/ elem)
```
removes the role attribute from the element.

States

These dijit methods set the state values onto an element and are wrappers to the appropriate attribute apis for the browser in use. The state parameter must be a valid ARIA state name and the value the appropriate value for the specified state.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.hasWaiState(/*Element*/ elem, /*String*/ state)
```
is used to determine if a particular state has been set on the element. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.getWaiState(/*Element*/ elem, /*String*/ state)
```
is used to obtain the value of a state that has been set on the element. It returns an empty string if the element has no value for the requested state. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
diijt.setWaiState(/*Element*/ elem, /*String*/ state, /*String*/ value)
```
assigns an ARIA state and value to the element. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.removeWaiState(/*Element*/ elem, /*String*/ state)
```
removes the role attribute from the element.

Examples of setting role and state:

The following will set a role of treeitem onto a DOM node:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.setWaiRole( domNode, “treeitem”);
```
This example sets the state of the treeitem to expanded:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.setWaiState( focusNode, “expanded”, “true”);
```
This example removes the valuenow property from an indeterminate progress bar.
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.removeWaiState(internalProgress, "valuenow");
```

Setting Role and State in the Widget Template

The role and state can also be set via the widget template using the waiRole or waiState prefix. Setting the role in the template is the same as setting it via scripting – the dijit.setWaiRole() method will be called during widget instantiation. Simply add the waiRole=”actualrole” or

waiState="state-value" parameters into the template markup for the element. The element will be passed as the nodeObj into the dijit.setWaiRole() and dijit.setWaiState() methods. The state is specified as a state name and value pair, the state is separated from the value using the hyphen character (-): state-value. Multiple states can be set within the template by separating the state-value pairs with a comma. This mechanism is useful when templates are used to create the objects requiring a role value and when the state is known at creation time.

Here is an example of setting the role in the dijt tree template. The domNode is given the "tree" role.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div class="dijitTreeContainer" style="" waiRole="tree"
dojoAttachEvent="onclick:_onClick,onkeypress:_onKeyPress"
></div>
```

The role or state can also be specified via variables. This example shows an excerpt from the dijit button template that sets the role and state on the button element. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div class="dijit dijitLeft dijitInline dijitButton"
        dojoAttachEvent="onclick:_onButtonClick,onmouseenter:_onMouse,onmouseleave:_onMouse,onmousedown:_onMouse"
        ><div class='dijitRight'
                ><button class="dijitStretch dijitButtonNode dijitButtonContents" dojoAttachPoint="focusNode,titleNode"
                        type="${type}" waiRole="button" waiState="labelledby-${id}_label"
                        ><span class="dijitInline ${iconClass}" dojoAttachPoint="iconNode"
                             ><span class="dijitToggleButtonIconChar">&#10003</span
                        ></span
                        ><span class="dijitButtonText" id="${id}_label" dojoAttachPoint="containerNode">${label}</span
                ></button
        ></div
></div>
```

Part 3: JavaScript Programming With Dojo and Dijit

In addition to being useful in itself, Dijit is a testament to Dojo's solidness and performance. The fact that you can build a really nice, extensible widget framework on top of it shows how cool Dojo is.

But that's only the beginning. Other Dojo-based mini-frameworks like dojo.data and dojox.offline allow you to build data sources and offline storage, respectively, in object oriented fashion. With a bit of JavaScript, you can use these services and many others bundled with Dojo for your own Web 2.0 applications:

- **Object Oriented Class Helpers** - JavaScript uses prototype-based, not class-based, object orientation natively. Dojo essentially builds a class system on top, adding great features like inheritance, encapsulation, mixin classes, and more. This will make Java and C# programmers feel at home, and help build solid Enterprise-level applications.
- **Modules** - As your client code grows, it's harder to keep all functions and global variables from using the same name and wrecking each other. Dojo adds a module system on top of JavaScript as part of its class layer. Modules are like packages in Java.
- **Dijit Methods** - As we've seen, you can create Dijit components programmatically - perfect for when you need 1000 tree nodes, for example. You can also build your own Dijit widget classes, and extend the ones already there.
- **Events** - Dojo generalizes the event system already present in JavaScript. You can connect events to code, and the resulting app works in Firefox, IE, and Safari without modification. It also introduces "double-blind" publish/subscribe events - perfect for loosely-coupled code.
- **XHR (Ajax)** - Dojo adds a nice wrapper around native XmlHttpRequest services. Often you can boil an XHR message-pass with one function call. The data is given to you in text, XML, or in a JavaScript object via JSON.
- **Drag and Drop** - Often the bane of DHTML applications, drag and drop services are essential for easy user interaction. Dojo's DnD layer is fast, cross-platform, and very straightforward.
- **Dojo.data** - The Data module is an abstraction layer that makes gathering data from outside sources consistent. Reading from a database or web service uses the same base calls. You can write pluggable data modules for your own sources.
- **Dojo.query** - Finding and manipulating HTML fragments is difficult. But Dojo.query makes it as easy as CSS. You can write sophsticated selectors, then apply an operation to all of them in one step.
- **i18n** - Short for "Internationalization" ("i", "n" and the 18 characters between them), Dojo i18n services help make all of your code globally accessible with understandable date, number and currency formats, and translated resource bundles.
- **Back button handling** - dojo.back saves your application from nervous users trying to use the Back button. One-page apps can destroy data at one touch of the back button. dojo.back alters the behavior of Back to make it less likely.
- **And more, more more!** - As your apps get sophsiticated, you can also write them compactly using Dojo's language extension features, HTML style functions, cookie functions, and much more!

Functions Used Everywhere

Before diving headfirst into the dojo pool, you need some basics. The following functions are called in many of the other sections' examples. While none do anything important for the user, per se, they are very handy for the dojo programmer.

Note that some functions, like dojo.connect() are described in more detail in their respective sections. Here, we give you just enough to get started.

dojo.require

dojo.require(String module)

If you've followed through the previous sections, you already know how important dojo.require is. In general if you use a function

```
dojo.someModule.someFunction();
```

You need to include

```
dojo.require("dojo.someModule");
```

If you don't, your scripts will throw a "dojo.someModule not defined" or "dojo.someModule.someFunction not defined..

Dojo's code is split into modules which are similar to packages in Java except that in Dojo a module can contain both constructors (like classes in Java) and simple functions. For example, the "dojo.string" module contains a number of functions, such as dojo.string.substitute(). The "dojo.dnd" module contains a number of constructors such as dojo.dnd.Container dojo.dnd.Source in additon to top-level functions and properties on the dojo.dnd object. We'll learn a lot more about this in Modules

Note the naming convention - functions, properties, and namespace objects start with a lowercase letter, and constructors (which are technically functions but act more like classes) start with a capital letter.

It may seem painful to require all modules, but dojo rewards by:

- Loading any dependent scripts for you. If dijit.form.TextBox requires dojo.math, you still need only require dijit.form.TextBox.
- Preventing loading dojo packages twice. dojo.require will simply return if the package is already loaded.
- Allowing you to build streamlined versions of dojo. If you use dijit.form.TextBox a lot, you can build a custom version of dojo that loads dijit.form.TextBox quickly. Dojo.require() knows whether the function is already loaded, and so you don't have to change any of your code. See The Build System for a discussion.

So you might wonder "So, don't I have to require the dojo module itself to use dojo.require?" Nope. Any function in the top-level package "dojo" is loaded automatically (dojo.query(), dojo.byId(), etc.). These are dojo's *Core functions*, and represent the most used functions according to usage patterns in the community. This is similar to the Java package java.lang, which is automatically available to all Java programs.

dojo.byId and dijit.byId

dojo.byId(String id)

This function is a synonym for document.all.id in IE or document.getElementById(id) in standard DOM. So you can say:

```
dojo.byId("breadbox").style.fontSize = "72pt";
```

If document.getElementById() works in all modern browsers, why use this? Sadly, bugs persist in Internet Explorer which make getElementById() unstable in some common scenarios. Also, dojo.byId() is easier to type.

dijit.byId(String id)

Because of the almost-identical spelling, this method is commonly mixed up with dojo.byId(String id). Here's the difference: dijit.byId() returns a Dijit widget instance - technically, an instance of dijit._Widget or one of its subclasses like dijit.Toolbar or dijit.TreeNode.

So here's the way to remember it:

- If you need a DOM Node, use dojo.byId. You need a DOM node anytime you call a standard browser function, change a standard property, etc.
- If you need a widget, use dijit.byId. From it you can access all of the attributes, methods and extension points listed in Part 2.

Or for you who think in terms of functions (e.g. LISP programmers?):

```
var myDomNode = dojo.byId("foo");
var myWidget = dijit.byId("foo");
console.debug(myDomNode == myWidget.domNode);  // true
console.debug(dijit.byNode(myDomNode) == myWidget); // true
```

dojo.addOnLoad

dojo.addOnLoad(Function fn)

Sooner or later, every Javascript programmer tries something like this:

```
<script>
  if(dayOfWeek == "Sunday"){
    document.musicPrefs.other.value = "Afrobeat";
  }
</script>
<form name="musicPrefs">
<input type="text" name="other">
...
```

It doesn't work because the "other" control is not defined yet. You can move the code to the bottom of the page, but that removes the linear nature of HTML. If you're reading the code, you want to zero in on a control, and see the code that influences it close by.

dojo.addOnLoad(...) defers script execution until all the HTML is loaded. So this code:

```
function setAfrobeat(){
   document.musicPrefs.other.value="Afrobeat";
}
dojo.addOnLoad(setAfrobeat);
```

conveniently replaces the one above. When the function is small, you may prefer to write it inline:

```
dojo.addOnLoad(
    function(){
        document.musicPrefs.other.value="Afrobeat";
    }
);
```

This is the *function literal* or *anonymous function* construct of JavaScript. If it looks really, really wierd to you, take a peek ahead at Functions as Variables for an explanation.

dojo.connect

Events in JavaScript or Dojo based applications are essential to making applications work. Connecting an event handler (function) to an element or an object is one of the most common things you will do when developing applications using Dojo. Dojo provides a simple API for connecting events via the dojo.connect() function. One important thing to note here is that events can be mapped to any property or object or element. Using this API you can wire your user interfaces together or allow for your objects to communicate. The dojo.connnect() API does not require that the objects be Dojo based. In other words, you can use this API with your existing code and interfaces.

Below is the code in the tutorial handling events. Here we connected the event handler, helloPressed, to the onclick property of the hello button element. When the button is clicked the funtion helloPressed will be called.

```
function helloPressed(){
   alert('You pressed the button');
}
function init(){
   button = dojo.byId('helloButton');
    dojo.connect(button, 'onclick', 'helloPressed');
}
```

It is also possible to use the Dojo event model to connect simple objects. To demonstrate, lets define a simple object with a couple of methods:

```
var exampleObj = {
    counter: 0,
    foo: function(){
        alert("foo");
        this.counter++;
    },
    bar: function(){
        alert("bar");
```

```
        this.counter++;
    }
};
```

So lets say that I want *exampleObj.bar()* to get called whenever *exampleObj.foo()* is called. We can set this up the same way that we do with DOM events:

```
dojo.connect(exampleObj, "foo", exampleObj, "bar");
```

Now calling *foo()* will also call *bar()*, thereby incrementing the counter twice and alerting "foo" and then "bar". Any caller that was counting on getting the return value from *foo()* won't be disappointed. The source method should behave just as it always has. On the other hand, since there's no explicit caller for *bar()*, it's return value will be lost since there's no obvious place to put it.

In either case, each time dojo.connect is called with the same arguments it will result in multiple connections. Later we will discuss strategies on how to guard against this.

Notice that dojo.connect takes a different number of arguments in the examples above. dojo.connect determines the types of positional arguments based on usage.

The Dojo event system allows you to connect to DOM elements or nodes or plain JavaScript objects. The API is sophisticated enough that it allows you to connect multiple listeners to a single object so you can have multiple actions as a result of a single event such as a mouse click. Of course there is an API to disconnect the listeners too. The Events section describes the Dojo Event system in more detail.

dojo.forEach

JavaScript 1.6 has a forEach loop, where you can apply a certain function to each element of an array. Unfortunately at the time of this writing, only Firefox 2 has support for JS 1.6. But never fear! Dojo has defined one you can use in any Dojo-supported browser.

Foreach is syntactic sugar for a regular ol' for loop. So for example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
for(var i in queueEntries){
   console.debug(queueEntries[i]);
}
```

Can be written as:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.forEach(queueEntries,
    function(oneEntry) {
        console.debug(oneEntry);
    }
);
```

Like dojo.connect, we use an anonymous function here to define the operation. This function must accept one element, being the value of each value in the array in turn.

For this simple loop, forEach isn't anything exciting. But combined with other Dojo functions, especially dojo.query(), it becomes remarkably useful. Consider this snippet, which disables all SELECT tags on the page:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.forEach(
    dojo.query("select", document),
    function(selectTag) {
        selectTag.disabled = true;
    }
);
```

How cool is that? (Answer: very!) There's no monkeying around with DOM functions, no using tedious names or id's, and it continues to work even when you add new SELECT tags.

Running dojo.forEach on a dojo.query result is so common, that Dojo defines a shortcut. This snippet: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.query("select", document).forEach(
    function(selectTag) {
        selectTag.disabled = true;
    }
);
```

does the same thing. But that's not all! **New in 1.0** you can collapse the function down to its body, passed in as a string like so: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;}

.geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// 1.0 only.
dojo.query("select", document).forEach("item.disabled = true;");
```

Ay carumba! That's a lot of functionality in a tiny 1-line package. Once you get used to the syntax, you'll never want to go back.

There's more on dojo.query in Selecting DOM Nodes with dojo.query

Multiple Versions of Dojo in a Page

[More content coming: not ready for an editor's pass yet]

As of Dojo 1.1, it is possible to run multiple versions of Dojo in a page, or to even rename the global "dojo", "dijit" and "dojox" variables to some other variable. This capability is useful if:

- You want to provide your own custom, namespaced library that uses Dojo underneath. It will be possible to use your own private Dojo version without having another version of Dojo on the page interfering with your code.
- You want to experiment with Dojo 1.1 features with a project using Dojo 1.0 or earlier.

Provide custom namespaced library

You want to create a JavaScript library that you want to distribute to others to use in their pages. The users of your library may be using Dojo on their page, and you want your version of Dojo to not interact with their version.

You call your code "coolio". As you are developing your code, you are using a normal build of Dojo 1.1+ and have your scripts laid out like so:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
-scripts
  -dojo
  -dijit
  -dojox
  -coolio
    -actions.js
-tests
  -test.html
```

The interface you want to publish to others consists of just one function: coolio.actions.foo(), and it is defined inside the scripts/coolio/actions.js file:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.provide("coolio.actions");
dojo.require("dojo.dnd");
dojo.require("dijit.ProgressBar");
coolio.actions.foo = function() { /* Does something that uses dojo.dnd and dijit.ProgressBar */ }
```

You want to use Dojo, but you do not want it to conflict with other Dojo versions on the page. In this case you can use a djConfig.scopeMap to map dojo, dijit and dojox to different names. Your test page, tests/test.html, might be set up like so:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<head>
<script type="text/javascript">
djConfig = {
    scopeMap: [
        dojo: "coolio",
        dijit: "cooliow",
        dojox: "cooliox",
    ],
    modulePaths: {
        "coolio.actions": "../coolio/actions"
    }
}
</script>
<script type="text/javascript" src="../scripts/dojo/dojo.js"></script>
<script type="text/javascript">
    coolio.require("coolio.actions");
    coolio.addOnLoad(function(){
        //do something with coolio.actions.foo();
    });
</script>
</head>
```

For this test page, djConfig.scopeMap has mapped "dojo" to "coolio", "dijit" to "cooliow" and "dojox" to "cooliox". dojo, dijit and dojox will not be visible to this HTML page, just coolio, cooliow and cooliox.

It is not recommended that you map dojo, dijit and dojox all to coolio. Doing so may result in some inadvertent clobbering of variables or objects, since you are basically collapsing dojo, dijit and dojox on top of each other. For example, if dojo.bar was defined and dijit.bar was defined, then one of them would clobber the other.

Also notice the modulePaths configuration for coolio.actions. If we did not do that, then when we do coolio.require("coolio.actions"), the code would try to load it from scripts/dojo/actions.js.

You can now develop your coolio.actions library in this manner. You can even use djConfig.debugAtAllCosts to do debugging (note that using debugAtAllCosts exposes dojo, dijit and dojox as global variables, so only test with your code in the page, and not mixed with code from another Dojo version).

Once you are ready to deliver your code to other people, use the Dojo build system to wrap up your code into a deliverable to give to others.

Build time

The build system allows you to "burn in" the scopeMap into the built JS file, avoiding the need for others to do that work.

Define a build profile like the below and save it as coolio.profile.js:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dependencies = {
        layers: [
                {
                        name: "dojo.js",
                        dependencies: [
                                "coolio.actions"
                        ]
                }
        ],
        prefixes: [
                [ "dijit", "../dijit" ],
                [ "dojox", "../dojox" ]
                //The mapping below assumes you placed "coolio"
                //as a sibling to the dojo directory. If not, adjust
                //the path accordingly.
                [ "coolio", "../coolio" ]
        ]
}
```

Notice the layer name is "dojo.js". This will make it so that your code is merged in with the base dojo.js, so that there is just one layer file that end users have to use. Once the build is complete, feel free to rename the file to "coolio.js" if you like.

The build command to burn in the scope map will look something like this (For Windows, change build.sh to build.bat):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
./build.sh profile=coolio releaseName=coolio version=0.1.0coolio action=release
scopeMap=[[\"dojo\",\"coolio\"],[\"dijit\",\"cooliow\"][\"dojox\",\"cooliox\"]]
```

The build command above assumes you saved coolio.profile.js in util/buildscripts/profiles. If you have it saved somewhere else, then replace profile=coolio with profileFile=path/to/coolio.profile.js.

This build command will generate a release/coolio directory. At this point you can rename release/coolio/dojo.js to release/coolio/coolio.js if you like.

Deliver the contents of release/coolio to the developers using your library. You can try to deliver just release/coolio/coolio.js, but depending on which dojo/dijit/dojox modules you use, that file may not be enough. You will likely need to do testing to see exactly what files are needed. If you only used the Dojo Base functionality (what comes with the normal dojo.js file), then you are fine just delivering the release/coolio/coolio.js file.

XDomain Build

If you plan on making an XDomain Build, there are a couple of other build arguments you need to pass:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
./build.sh profile=coolio releaseName=coolio version=0.1.0coolio action=release loader=xdomain xdDojoScopeName=coolio
xdDojoPath=http://some.domain.com/path/to/coolio
scopeMap=[[\"dojo\",\"coolio\"],[\"dijit\",\"cooliow\"][\"dojox\",\"cooliox\"]]
```

This command assumes you will be hosting your code at http://some.domain.com/path/to/coolio/. Under that directory would be the dojo, dijit, dojox and coolio directories.

For xdomain builds, you will want to rename release/coolio/dojo.xd.js to release/coolio/coolio.xd.js, and tell developers to reference http://some.domain.com/path/to/coolio/dojo/coolio.xd.js in their pages.

Using Dojo 1.1 with previous versions of Dojo

You may have an application running Dojo 0.4.3, but you might want to start playing with Dojo 1.1 features. It is always more efficient for the browser if you only use one version of Dojo in your page, but you may not have the time to port your whole application to Dojo 1.1 yet.

Suppose you have your code laid out like so:

Object Orientation

As we saw in the introduction to Part 3, JavaScript doesn't have a class system like Java. But Dojo has functions to simulate one. For some background on JavaScript and prototype-based object orientation, chapter 9 of David Flanagan's *JavaScript: The Definitive Guide, 5th edition* is a good read.

This section has some pretty abstract stuff, and you may wish to skip it on the first read. Certainly you can do a lot with Dojo without using dojo.declare or the other object orientation functions. But a good knowledge of it will help you program faster and smarter.

Code Re-Use

JavaScript is at its heart an object oriented language, but it is a prototype based object oriented language which does not have the same structure as class based languages like Java.This concept can be hard for new programmers who are not familiar with its construct. Dojo brings the object orientedness into a more familiar domain by modeling concepts that can be followed from Java and letting the toolkit handle the prototyping, inheritance and odd procedures JavaScript requires to make it work. Because of this, it not only allows people to get programming in object oriented JavaScript quicker, but it makes it faster to program because you can let the toolkit handle all of the odd procedures JavaScript requires to make it work.

In dojo, object orientation is used primarily for inheritance. Think of inheritance as "some of the things you can do, I can do better." For example, think of a Progress Bar widget which always displays "nnn%" in the middle. You'd like to change that to "n out of m steps completed." You could, in theory, copy and paste the dijit Progress Bar code to your own file and fix the message. But this is a lot of duplicated code and more Javascript to load into the browser.

In object oriented land, you simply define an object, e.g. NOutOfMProgressBar, as a subclass of ProgressBar. We say NOutOfMProgressBar *inherits* the behavior of ProgressBar. Then you specify only the method that differs between the two - in this case report() - in NOutOfMProgressBar. Then you can effectively a variable of type NOutOfMProgressBar wherever you'd use a ProgressBar.

This promotes a more effective sharing of code. Dojo can upgrade its version of Progress Bar, and as long as report() is part of the contract, NOutOfMProgressBar will continue to work correctly. And, hey, let's face it. The less Javascript code you have to write, the better off you are.

Authors: David A (?), Craig Riecke

Functions as Variables, or "Here, Do This."

Before jumping into the details, one aspect of Javascript requires some explanation. Unlike Java or C#, functions are first class objects in Javascript. You can do everything with a function that you can do with an integer, including:

- Assign it to a variable
- Use it as a value in an array, or associative array
- Pass it as a parameter

Here's a simple example. Suppose you want a function that calculates either the maximum or the minimum of two numbers, depending on another parameter to choose. You can code it like this: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function applyMaxOrMin(a, b, fnName){
   if(fnName == 'max'){
      return Math.max(a,b);
   }else{
      return Math.min(a,b);
   }
}
console.debug(applyMaxOrMin(1, 2, 'max'));
```

But you can make it a line or two shorter, and more general, by passing a function like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function applyTwoParameterFn(a, b, fn){
   return fn(a,b);
}
console.debug(applyTwoParameterFn(1, 2, Math.max));
```

Interestingly, you can substitute any function that takes two numbers here, not just Max or Min.

"Ah," you might say, "that's like passing a class in Java and using reflection." Almost, but not quite. Java is still strict about type checking, even when calling functions through reflection. In our example, what if the function passed aren't compatible? In the example above, what if you pass a function with three variables? It's handled in The Javascript Way, i.e. sloppily. If the number of parameters doesn't quite fit, extra ones are lobbed off or null's are added to the end. Type conversion is applied like it should.

Note: *There is no concept of overloading in Javascript!* That's a fundamental difference between it and strongly-typed object languages like

Java.

One special function type used a lot in dojo is the *callback*. Callbacks are functions that are supposed to be called when processing has ended. They are especially useful for asynchronous operations like XHR. You call an asynchronous function and say "call this function when you're done." If you've used event classes in Java, it's the same concept but looser.

In Java you can define classes anonymously, on-the-fly, right in the middle of a method call. You can do that in Javascript too. Simply define the function in the parameter list and omit the name. For example, instead of defining and passing a new function:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
function myTwoParameterFn(a, b) {
   return max(a, -b);
}
console.debug(myTwoParameterFn(1, 2, Math.max));
```

We can shorten it to:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
console.debug(myTwoParameterFn(
    1, 2,
    function(a,b){
        return max(a, -b)
    }
));
```

This makes from some pretty strange syntax, especially if the anonymous function is large. But callbacks are often specified this way when calling dojo functions.

Declaring a Class

Classes in Dojo are declared with a declare statement with a class name, super-class information, and body. Within the body can be variables and methods.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.declare("ClassName", null, {/*class body*/});
```

(Note: ClassName is the basic name, but to avoid naming conflicts, use module names like com.coolness.ClassName. See modules for details. For simplicity sake, we will start out with using just the simple name.)

Let's add some more content to our class by giving it a name and showing what the constructor can do. Following is a Person class with a constructor and a moveToNewState() function:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.declare("Person", null, {
        constructor: function(name, age, currentResidence){
                this.name=name;
                this.age=age;
                this.currentResidence=currentResidence;
        },
        moveToNewState: function(newState){
                this.currentResidence=newState;
        }
});
```

Note the use of anonymous functions here. You are passing to dojo.declare an associative array of anonymous functions. "That's not an anonymous function," you might say, "their names are constructor and moveToNewState!" Strictly speaking, no they aren't. They are anonymous functions with the *keys* constructor and moveToNewState.

The function named "constructor" is special. It gets called when you create an object with the "new" operator of JavaScript.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
//create an instance of a new person
var matt= new Person('Matt', 25, 'New Mexico');
```

In pure JavaScript, this is handled by a prototype function named after the class - for example, Person.prototype. Dojo wires in your constructor as a part of the prototype, but then adds extra goodies like calling the superclass constructor and initializing extra properties.

Our Matt object who is 25 currently lives in New Mexico, but let's say he moves a little further west to California. We can set his new

currentResidence with the Person class method moveToNewState():

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
matt.moveToNewState('California');
```

Now the current value of matt.currentResidence shows that he now lives in California.

If you have used prototypes in Javascript, the above example corresponds roughly to:/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
function Person(name, age, currentResidence){
        this.name=name;
        this.age=age;
        this.currentResidence=currentResidence;
},
Person.prototype.moveToNewState= function(newState) {
        this.currentResidence=newState;
}
```
But the dojo version is more compact and has added value as we will see later.

Arrays and Objects as Member Variables

If your class contains arrays or other objects, they should be declared in the constructor so that each instance gets it's own copy. Simple types (literal strings and numbers) and are fine to declare in the class directly.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.declare("my.classes.bar", my.classes.foo, {
        someData: [1, 2, 3, 4], // doesn't do what I want: ends up being static
        numItem : 5, // one per bar
        strItem : "string", // one per bar
         constructor: function() {
                this.someData = [ ]; // better, each bar has it's own array
                this.expensiveResource = new expensiveResource(); // one per bar
        }
});
```

On the other hand, if you want an object or array to be static (shared between all instances of *my.classes.bar*), then you should do something like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.declare("my.classes.bar", my.classes.foo, {
        constructor: function() {
                dojo.debug("this is bar object # " + this.statics.counter++);
        },
        statics: { counter: 0, somethingElse: "hello" }
});
```

"Statics" is not a special dojo construct - you can use any name you want, like "constants". In this example, you'd refer to the variable as myInstance.statics.counter both inside and outside the class definition.

Why is this true for arrays and objects, but not primitives? It's because, like most OOP languages, JavaScript uses object references. For example, given:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
x = { fruit: "apple" };
y = x;
```

Now *x* and *y* both refer to the same object. Modifying *x.fruit* will also affect *y.fruit*.

Inheritance

A person can only do so much, so let's create an Employee class that extends the Person class.The second argument in the dojo.declare() function is for extending classes.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;}

.geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.declare("Employee", Person, {
        constructor: function(name, age, currentResidence, position){
                // remember, Person constructor is called automatically
                this.password="";
                this.position=position;
        },
        login: function(){
            if(this.password){
                alert('you have successfully logged in');
            }else{
                alert('please ask the administrator for your password');
            }
        }
});
```

Dojo handles all of the requirements for setting up the inheritance chain, including calling the superclass constructor automatically. Methods or variables can be overridden by setting the name to the same as it is in the parent class. The Employee class can override the Person class moveToNewState(), perhaps by letting the company pay for moving expenses.

You initialize the subclass the same as the Person class with the new keyword.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var kathryn=new Employee(' Kathryn ', 26, 'Minnesota', 'Designer');
```

The Employee class passes the first three arguments down to the Person class, and sets the position.Kathryn has access to the login() function found in the Employee class, and also the moveToNewState() function by calling kathryn.moveToNewState("Texas"); Matt on the other hand, does not have access to the Employee login() function.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
matt.login() // ERROR can't log in because he is not an Employee
```

Calling Superclass Methods

Often when you're overriding a method, you want to *add* something to the superclasses method, not totally replace it. Dojo has helper functions to make this easy.

But you don't have to worry in the constructor. As we said above, superclass constructors are **always** called automatically, and **always** before the subclass constructor. This convention reduces boilerplate in 90% of cases.

For all other methods, you can use /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `inherited(arguments)` to call the superclass method of the same name. Take for example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
someMethod: function() {
      // call base class someMethod
      this.inherited(arguments);
      // now do something else
    }
```

Inherited will climb up the scope chain, from superclass to superclass and through mixin classes as well, until it finds "someMethod", then it will invoke that method.

The argument is always literally /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `arguments`, a special Javascript array variable which holds all the arguments (like argv in C).

There are a few variations to inherited() for special cases. If you have a method that was put into your object outside of declare, you need to specify the name of the calling function like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
this.inherited("someMethod", arguments);
```

And you can send custom parameters to the ancestor function. Just place the extra arguments in array literal notation with brackets: /*

GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
this.inherited(arguments, [ customArg1, customArg2 ])
```

Mixins

Just as Dojo adds class-based inheritance to JavaScript, so it adds support for *multiple inheritance*. We do this through Dojo *mixins*. The methods and properties of a mixed-in class are simply added to each instance.

In pure object-oriented languages like Java, you must use typecasts to make an object "act like" its mixed-in class (in Java, this is through interfaces). Not in Dojo. You can use the mixed-in properties directly.

Suppose, for example, you have a class called VanillaSoftServe, and classes MandMs and CookieDough. Here's how to make a Blizzard:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.declare("VanillaSoftServe",null, {
    constructor: function() { console.debug ("mixing in Vanilla"); }
});
dojo.declare("MandMs",null, {
    constructor: function() { console.debug("mixing in MandM's"); },
    kind: "plain"
});
dojo.declare("CookieDough",null, {
    chunkSize: "medium"
});
dojo.declare("Blizzard", [VanillaSoftServe, MandMs, CookieDough], {
        constructor: function() {
             console.debug("A blizzard with "+
                 this.kind+" M and Ms and "+
                 this.chunkSize+" chunks of cookie dough."
             );
        }
});
```

Then the following:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
new Blizzard();
```

Will first print "mixing in Vanilla" on the debug console because VanillaSoftServe is the superclass of Blizzard. In fact, VanillaSoftServe is the *only* superclass of Blizzard - the first mixin is always the superclass. Next the constructors of the mixins are called, so "mixing in MandMs" will appear. Then "A blizzard with plain M and Ms and medium chunks of cookie dough." will appear.

Mixins are used a lot in defining Dijit classes, with most classes extending Dijit._Widget and mixing in Dijit._Templated.

Understanding The Parser

One of the most common things to see in a Dojo page is a djConfig block like /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .html4strict .imp {font-weight: bold; color: red;} .html4strict .kw1 {color: #b1b100;} .html4strict .kw2 {color: #000000; font-weight: bold;} .html4strict .kw3 {color: #000066;} .html4strict .coMULTI {color: #808080; font-style: italic;} .html4strict .es0 {color: #000099; font-weight: bold;} .html4strict .br0 {color: #66cc66;} .html4strict .st0 {color: #ff0000;} .html4strict .nu0 {color: #cc66cc;} .html4strict .sc0 {color: #00bbdd;} .html4strict .sc1 {color: #ddbb00;} .html4strict .sc2 {color: #009900;} `djConfig="parseOnLoad: true"` and later /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.require("dojo.parser");`. Together these lines include and enable Dojo's page parsing infrastructure. This machinery layers on top of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.query()` to provide a way to declare instances *of any class* via markup in your page. To understand what we mean by that, let's take a simple example page which consists of a single tree backed by a JSON data store:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
<script type="text/javascript">
        dojo.require("dojo.data.JsonItemStore");
```

```
        dojo.require("dijit.Tree");
        dojo.require("dojo.parser");
        var countries = new dojo.data.JsonItemStore({ url: "countries.json" });
</script>
</head>
<body class="tundra">
        <div dojoType="dijit.Tree" store="countries" labelAttr="name" typeAttr="type"
           query="{type:'continent'}" ></div>
```

In this case, we see the familiar use of the dojoType attribute to denote where an instance of our widget should be created. This is the functional equivalent of writing:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
       djConfig="parseOnLoad: true"></script>
<script type="text/javascript">
        dojo.require("dojo.data.JsonItemStore");
        dojo.require("dijit.Tree");
        dojo.require("dojo.parser");
    dojo.addOnLoad(
        var countries = new dojo.data.JsonItemStore({ url: "countries.json" });
        var tree = new dijit.Tree({
            store: countries,
            labelAttr: "name",
            typeAttr: "type",
            query: {type: "continent"}
        }, dojo.byId("treePlaceHolder"));
    });
</script>
</head>
<body class="tundra">
        <div id="treePlaceHolder"></div>
```

In fact, they're identical. The only difference is that in the first example, the work of locating and creating the widget instance is handed off to the Dojo parser. Fundamentally, this means that the parser is:

- locating the nodes with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojoType` attributes in the page
- taking the attributes assigned to them and passing them into the constructor as properties on the configuration object
- passing the source node for the widget as the second parameter to the constructor

There's nothing about this process (except perhaps the passing of the node) which should be specific to widgets, and indeed the Dojo parser is equipped to create instances of any class. That means that we can revise our first example to be fully markup-driven:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
<script type="text/javascript">
        dojo.require("dojo.data.JsonItemStore");
        dojo.require("dijit.Tree");
        dojo.require("dojo.parser");
</script>
</head>
<body class="tundra">
        <div dojoType="dojo.data.JsonItemStore" url="countries.json" jsId="countries"></div>
        <div dojoType="dijit.Tree" store="countries" labelAttr="name" typeAttr="type"
           query="{type:'continent'}" ></div>
```

So how does this work? What happens to the source node in the resulting page when what we're creating isn't a widget? And what about the seemingly magical properties /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojoType` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `jsId`?

The Parsing Algorithm

To fully understand the parser, it's important to understand it's operation in broad terms. The parser operates by:

- Searching the document for elements with a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojoType` attribute. This search is linear and nodes are returned in document order.
- Iterating over all matching nodes, attempting to match the declared type with an available class to instantiate.
    - If a class is found, the parser iterates over attributes of the class's prototype and populates the arguments object from the

values of attributes on the source node of the same name. Lightweight type conversion is performed.
    - If a markup factory is found for the class, it is used to create a new instance to return
    - If no markup factory is found, the class is constructed using the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `new` operator. The arguments to the constructor are assumed to be in the form /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `new someClass(argumentsObj, sourceNode);`
- Events from markup are attached (although some may have been handled earlier)

The above process alludes to some features of the parser which we haven't seen in action yet. Here's a more sophisticated example. We create a class called "example.Class". The parser runs and creates an instance of this class (the div dojoType="example.Class"), which you can then access through the global variable "thinger."

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
<script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.declare("example.Class", null, {
                constructor: function(args, node){
                        //  this class constructor is designed for the
                        // parser's "args, node" convention
                        dojo.mixin(this, args);
                },
        });
</script>
</head>
<body class="tundra">
        <div dojoType="example.Class" foo="bar" jsId="thinger">
                <script type="dojo/method">
                        // this block is executed as the class is created but
                        // after the class constructor is finished
                        console.debug(this.foo);   // Prints "bar"
                </script>
        </div>
```

Each script of type "dojo/method" is executed after the constructor runs. We saw examples of this in Part 1, Example 2.. Finally, the class constructor uses a mixin to copy the attributes from tag to properties in the instance. Thus thinger is created by calling the constructor with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `args = {foo: "bar"}` and node as the div node itself. Foo is created as a property by mixin.

Type Conversions

Lightweight type conversions are done on the attribute values. The types are based on the property types used in the definition of the class. For example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
<script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.declare("example.Class", null, {
                constructor: function(args, node){
                        //  this class constructor is designed for the
                        // parser's "args, node" convention
                        dojo.mixin(this, args);
                },
                title: "Hello",
                isEnabled: true,
                dayCount: 45,
                onClick: function(){},
                names: ["Monday", "Tuesday", "Wednesday"],
                startDate: new Date()
        });
</script>
</head>
<body class="tundra">
        <div dojoType="example.Class" title="Good Morning" isEnabled="false" dayCount="4" onClick="alert(thinger.dayCount)"
names="Thursday, Friday" startDate="2008-01-01" jsId="thinger">
                <script type="dojo/method">
                        // this block is executed as the class is created but
                        // after the class constructor is finished
                        console.debug(this.foo);   // Prints "bar"
                </script>
        </div>
```

In this example, the attributes will be converted to their correponding types that were used in the definition of example.Class:

- **title:** String
- **isEnabled:** Boolean
- **dayCount:** Number
- **onClick:** Function. Specify the function body in the attribute.
- **names:** Array. Separate the array members by commas. Array members are assumed to be strings.
- **startDate:** Date. "now" can be used to get the current date, otherwise, dojo.date.stamp.fromISOString() will be used to convert the text string to a Date object.

If the property type does not match one of the types listed above, then dojo.fromJson() will be used to convert the attribute value.

Markup Factory

If the class declares a method with the name markupFactory, that function will be used to create the object instance, instead of the constructor. This is useful if the class has special initialization for instances created via markup, versus instances created in script via the class constructor. An example class that defines a markupFactory method:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript" src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
<script type="text/javascript">
        dojo.require("dojo.parser");
        dojo.declare("example.Class", null, {
                constructor: function(args, node){
                        //  this class constructor is designed for the
                        // parser's "args, node" convention
                        dojo.mixin(this, args);
                },
                title: "Hello",
                isEnabled: true,
                dayCount: 45,
                onClick: function(){},
                names: ["Monday", "Tuesday", "Wednesday"],
                startDate: new Date(),
                markupFactory: function(params, domNode, constructorFunction){
                        //params: object that contains the markup attribute values,
                        //with type conversion already completed.
                        //domNode: the DOM node (the div in the code below)
                        //constructorFunction: The constructor function matching
                        //the dojoType in markup. In this example, example.Class
                        var instance = new constructorFunction(params, domNode);
                        //Do special initialization intialization here
                        return instance;
                }
        });
</script>
</head>
<body class="tundra">
        <div dojoType="example.Class" title="Good Morning" isEnabled="false" dayCount="4" onClick="alert(thinger.dayCount)"
names="Thursday, Friday" startDate="2008-01-01" jsId="thinger">
                <script type="dojo/method">
                        // this block is executed as the class is created but
                        // after the class constructor is finished
                        console.debug(this.foo);   // Prints "bar"
                </script>
        </div>
```

Modules

Once you get started writing Dojo code, you'll start writing more sophisticated client side code. And the more code you write, the more you will split into JavaScript modules. The more files, the more dangerous are the consequences of using JavaScript carelessly. For example, a harmless little function like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function avg(listOfNumbers) {
   sum = 0;
   for(i in listOfNumbers){
      sum += listOfNumbers[i];
   }
   return sum / listOfNumbers.length;
}
```

tucked away in a script will wreak havoc on the following in another script: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function sum(listOfNumbers) {
   var retval = 0;
   for(i in listOfNumbers){
       retval += listOfNumbers[i];
```

```
   }
   return retval;
}
document.print(avg(a));
document.print(sum(a));
```

Because the variable sum in the avg function is stored in the global namespace (i.e. it is not prefixed by "var"), it will overwrite the loaded sum() function in the second script. This leads to hard-to-diagnose errors

What you need is to keep these modules corralled in their own spaces. While modularization is not part of JavaScript, Dojo can simulate it for you through namespaces. And you've been using them all along! Whenever you execute a dojo.require, you are invoking a module loaded into its own namespace. So, for example, dijit.Toolbar is a namespace. Java and C# have similar constructs named packages.

While namespaces are optional, they tend to lessen hassles down the road as your code base grows.

What dojo.require Does

The modules bundled with Dojo correspond roughly to .js files underneath the Dojo root. Looking underneath your dojo root directory, you will see at least three top-level directories dojo. dijit. and dojox. Everything that you've dojo.require'd so far has begun with one of these three prefixes.

Like most modern languages, Dojo uses "." to separate modules from submodules from sub-submodules, etc. These correspond to directories, subdirectories and sub-subdirectories underneath the Dojo root. The name after the last period is the JavaScript file name. So for example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.require("dijit.form.Button");
// Esentially loads the script dijit/form/Button.js underneath the Dojo root
```

Why not just load the scripts with a SCRIPT tag? Well, besides being shorter, a dojo.require statement ensures that modules are not loaded twice. They also make the build system function better, which you'll see in The Build System chapter. You can, however, pull in any Dojo code after the initial dojo.js script tag with a script tag and the package system will ensure that dependencies for that package are still satisfied. This trick is particularly handy when you need line-number debugging information from some new chunk of code you're working on.

The script dijit/form/Button.js in turn starts like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.provide("dijit.form.Button");
dojo.require("dijit.form._FormWidget");
dojo.require("dijit._Container");
dojo.declare("dijit.form.Button", dijit.form._FormWidget, {
   // [snip]
}
dojo.declare("dijit.form.DropDownButton", [dijit.form.Button, dijit._Container], {
   // [snip]
```

All of the classes defined in this script are then available to you. By convention, most modules define one class named after the module itself, as in our dijit.form.Button example. You can also use dijit.form.DropDownButton, if you dojo.require("dijit.form.Button"). But dojo.require("dijit.form.DropDownButton") will not work. Each script provides only one package.

Note, however, that it's Dojo convention to prefix "suggested private" classes with a "_". You shouldn't use these classes, even though JavaScript won't throw an error if you do.

Creating Your Own Modules

Easy Way: A Dojo Peer Directory

So suppose you want to create your own module called explosive.space.Modulator. The most straightforward method involves creating dojoroot/explosive/space/Modulator. That way, "explosive" is at the same directory level as "dojo" and "dijit":

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw3 {color: #000066;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .re0 {color: #0000ff;} .geshifilter .re1 {color: #0000ff;} .geshifilter .re2 {color: #0000ff;} .geshifilter .re3 {color: #808080; font-style: italic;} .geshifilter .re4 {color: #0000ff;}

```
[criecke@smoochie js]$ ls
dijit  dojo  dojox  util
[criecke@smoochie js]$ mkdir --parents explosive/space
[criecke@smoochie js]$ ls
dijit  dojo  dojox  explosive  util
```

Now create the file "explosive/space/Modulator.js":

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0

{color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.provide("explosive.space.Modulator");
dojo.declare("explosive.space.Modulator",null,{
    // fil in the body here
});
```

And you're ready to include it like any other Dojo module:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.require("explosive.space.Modulator");
var eludiumFuel36 = new explosive.space.Modulator();
```

Sometimes instead of classes, you may want to define plain ol' functions in a module. That's fine too. For example: you could define explosive.space.utilities:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.provide("explosive.space.utilities");
explosive.space.utilities.shuffleOverToCannon = function(steps) {
   // body here
}
```

Then, after the dojo.require, you can call explosive.space.utilities.shuffleOverToCannon().

Cleaner Way: External Directories

The problem with this approach is your mucking up the dojo root directory. It's better to keep your development separate from Dojo itself. It tends to make source control easier to deal with.

So let's say you put "explosive/space/Modulator.js" underneath private_dojo at the same level as Dojo root. Then you only need to add the registerModulePath statement:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.registerModulePath("explosive","../../private_dojo/explosive");
dojo.require("explosive.space.Modulator");
```

The module path is specified relative to the /dojoroot/dojo directory.

Enterprise-Class Way: Custom Builds

An alternate method, which doesn't require the registerModulePath, is to have a build script (Ant, Makefile, or whatever) mix both Dojo and your custom stuff under the same root. That way your source code stays separate from Dojo's in development, but mixes together in a nice way for production. See Custom Builds for details.

Module Helpers

Modules for Non-JavaScript Resources

dojo.require() works fine for JavaScript, but what about images, CSS, and other resources? Do you need to access them through absolute paths? Fortunately no. The following code:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var imgNode = document.createElement("img");
imgNode.src = dojo.moduleUrl("explosive.space","images/kaboom.gif");
```
locates the kaboom.gif file in our module. The nice thing is ... this code will work in any page that has registered the module. It doesn't matter where the code snippet is located relative to kaboom.gif. Slick!

Conditional Inclusion

Sometime modules are dojo.require'd because they *may* be used. If they are not used, that's a small bit of wasted time. If you know at runtime whether modules need inclusion, you can use dojo.requireIf: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.requireIf(dojo.isIE, "explosive.space.BlueScreenOfDeathCatcher");
```

Creating and Scripting Widgets

Part 2: Dijit discussed how to work with widgets *declaratively*, using the "dojoType" attribute in HTML tags. This section explains how to create and interact with dojo widgets *programmatically*, via JavaScript.

This discussion spans the following topics:

Creating a Widget Programmatically

How to create dojo widgets from JavaScript, and swap/insert them into a page's DOM existing structure

Interacting With Widgets

How to change properties/behavior of a dojo widget throughout the lifecycle of a page

Writing Your Own Widget

How to write and instantiate your own widgets

Creating a Widget Programmatically

In Part 2: Dijit, you saw how to instantiate (create an instance of) a dojo widget *declaratively*, using the "dojoType" attribute: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.TitlePane" title="Outer Pane">
   This is a title pane containing another title pane
   <div dojoType="dijit.TitlePane" title="Inner Pane">
      And this is the inner title pane...
   </div>
</div>
```

While this method is very convenient, widgets declared in this way are instantiated only when the page first loads. What if, however, you want to create a widget at some later time, e.g. as the result of a user-action? This is one case where we would want to instantiate a widget *programmatically*, which looks like: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var button1 = new dijit.form.Button(params, srcNodeRef);
```

Let's pick that programmatic example apart:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
`var button1`

This is the name of the variable that will refer to the instantiated widget. You can use this variable to interact with the widget later (see Interacting With Widgets)

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
`dijit.form.Button`

This is the fully-qualified name of the widget you want to instantiate; freeing you to, perhaps, later instantiate /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `yourmodule.form.Button` (see Writing Your Own Widget Class)

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
`params`

An object whose properties map to the widget's properties. Dijit uses this object to initializes the widget's properties. Thus, if the "params" object in this case has a "label" property, dijit will set the dijit.form.Button's "label" attribute to the value of that property. See **Setting Properties** below for elaboration.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900;

font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
`srcNodeRef`

> This is either a reference to an existing DOM node, or the id of an existing DOM node. When the widget is successfully instantiated, this node will be replaced with the widget's node. **New in 1.0:** You can skip this parameter for dijit.Tooltip, dijit.TooltipDialog and dijit.Dialog. Since they require no specific place on the page, there's no sense in providing one.

Thus, the programmatic equivalent of: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.TitlePane" title="Inner Pane">
  And this is the inner title pane...
</div>
```

from the example above would be: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var innerPane = new dijit.TitlePane( {title:"Inner Pane"}, dojo.byId("someDiv"));
```

When that line executes, the **div** with the id "someDiv" will be replaced with a TitlePane widget, with title "Inner Pane".

Setting Properties

Programmatically creating widgets allows extra freedom in the parameters. The following is perfectly legal:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var innerPane2 =
    new dijit.TitlePane({
        title: 'Creating New '+docType,
        duration: 5 * 1000    /* 5 seconds, converted to ms */
    }, dojo.byId("someDiv"));
```

In declarative widgets, you may only pass strings. In programmatic ones, you can pass arrays, nested objects, Dates or Numbers as parameters. This isn't important for bundled Dijit components - they all work with strings - but it can make building your own widgets easier.

New in 1.0: When programmatically creating a widget class, style, and id now need to be specified as parameters to the constructor, not as attributes of the placeholder node. For example, the following is incorrect: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script>
// Doesn't work in 1.0.  Class and style will be overwritten
new dijit.form.Button({},,dojo.byId("someDiv"));
</script>
<div id="someDiv" class="large" style="color:red"></div>
```

The correct code:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// Works in 1.0
new dijit.form.Button({ "class": "large", style: "color: red" }, dojo.byId("someDiv"));
```

startup()

Certain widgets require a startup() method to be called. When building widgets programmatically, you create the parent first, then add the children, and grandchildren... and finally call startup(). Startup() is called once on the top element in the hierarchy, after the whole hierarchy has been setup and the element inserted.

It's good practice to include the startup() call, even for widgets that have no children or do not require it.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
accordion = new dijit.layout.AccordionContainer({}, dojo.byId("accordionShell"));
accordion.addChild(new dijit.layout.ContentPane());
accordion.addChild(new dijit.layout.ContentPane());
accordion.addChild(new dijit.layout.ContentPane());
accordion.startup();
```

Interacting With Widgets

Overview

This section discusses how to programmatically retrieve a reference to a widget, and how to use that reference to interact with the widget.

Getting the Widget Reference

In order to programmatically interact with a widget, you need a variable that references that widget.

Retaining a Widget Reference

If the widget was created programmatically, you can simply retain a reference to what you created: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var widgetReference = new dijit.form.Button(params, srcNodeRef);
```

Obtaining a Widget Reference

If the widget was created declaratively, or you didn't retain a reference to a programmatically-created widget, you can obtain a reference to the widget object if you know its widgetId. You do this using /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dijit.byId("idOfWidget")` /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.form.Button" label="Click" id="button1"></div>
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.addOnLoad(function() {
            var widgetReference = dijit.byId("button1");
            widgetReference.setLabel("Don't Click!");
          });
```

Note: *This is different than a DOM id.* In the above example, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dijit.byId("button1")` returns a reference to the widget, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.byId("button1")` returns a reference to the actual DOM node of the button.

Using the Reference

- **The DOM Node**

  You can access the (root) DOM node of the widget via the "domNode" property of the reference.

- **Common Dijit-Widget Interactions**

  Popups

  Popups can be opened/closed using open() and close()

  StackContainer widgets

  StackContainer widgets such as TabContainer and AccordionContainer use a transition() method to switch between two widgets (for example, you might do chained animation, fading out one widget before wiping in the second).

- **Animations**

  You can make an arbitrary animation to show/hide a widget using the dojo.fx code (slideIn, fadeIn, or something more complicated using animateProperty and/or combined animations on myWidget.domNode

- **Events**

Writing Your Own Widget Class

It's hard for you to leave well-enough alone. We give you widgets, and now you want to change them. Or you want to make your own.

No problem! Dijit components are extendible, so you can make changes without touching the source code. In a way, you already do this by specifying your own attributes - e.g. sliders that go from 0-100 look different than those going from 0-200. But sometimes you need to go further. Maybe you need to create different behavior for onClick, or substitute a custom validation routine. This kind of modification uses *extension points* described in Common Attributes. You can add your own code to extension points through markup or through pure JavaScript calls to dojo.declare.

You can also create Dijit classes from scratch. Again, you can do this either through markup - using the dijit.Declaration dojoType attribute - or through dojo.declare.

The Template

Often, you'll find a widget that does almost exactly what you want ... but needs just a bit of help. *The last thing* you'd want to do is hack the source code. Good thing you don't have to! The same widget construction techniques apply to both creating and extending widgets. So first, let's extending an existing stable widget: the AccordionContainer and AccordionPane. Right now the pane titles can only be text, but suppose you want images there as well.

Most Dijit components revolve around a *template*. A template looks like a macro, and can perform simple substitutions of ${...} variables and substitutions of DOM nodes.

To see how the template language works, let's look at a plain ol' Accordion Container instance:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.layout.AccordionContainer">
    <div dojoType="dijit.layout.AccordionPane" title="pane 1">
        Text of Pane 1
    </div>
    <div dojoType="dijit.layout.AccordionPane" title="pane 2">
        Text of Pane 2
    </div>
</div>
```

The DOM that ends of in memory and on the screen is quite different. Though [View Source] in your browser shows the HTML above, Firebug shows the actual DOM used by the AccordionPane:

[inline:simple_accordion_firebug.png]

Dojo replaces the simple nodes of our example with groups of HTML elements. That's one of the jobs of the Dojo parser. The parser consults the widget's template to construct this group. Where's the template kept? For AccordionPane, you can find it in the source version of Dojo at dijit/layout/templates/AccordionPane.html:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div class='dijitAccordionPane'
        ><div dojoAttachPoint='titleNode,focusNode' dojoAttachEvent='ondijitclick:_onTitleClick,onkeypress:_onKeyPress'
                class='dijitAccordionTitle' wairole="tab"
                ><div class='dijitAccordionArrow'></div
                ><div class='arrowTextUp' waiRole="presentation">▲</div
                ><div class='arrowTextDown' waiRole="presentation">▼</div
                ><span dojoAttachPoint='titleTextNode'>${title}</span></div
        ><div><div dojoAttachPoint='containerNode' style='overflow: hidden; height: 1px; display: none'
                dojoAttachEvent='onkeypress:_onKeyPress'
                class='dijitAccordionBody' waiRole="tabpanel"
        ></div></div>
</div>
```

That's a lot of HTML to replace the one DIV tag. Mostly this looks like garden variety HTML with lots of CSS class markers to fill in the styling. That makes the theme system work well. But a few attributes and constructs are alien to HTML - these are Dojo's template language elements. In particular:

${title}

is replaced with the title="..." attribute sent to the widget

dojoAttachPoint='containerNode'

An attachPoint is a tag merged with user-provided HTML. It's like a substitution variable for HTML.

dojoAttachEvent='onkeypress:_onKeyPress'

Connects an event to an event handler at that node.

Attach Points

The AttachPoint is a little bit complex, so let's look at that a bit more closely. Each widget has a special variable named containerNode which represents the unchanged HTML. The act of attaching involves three steps.

1. Start with containerNode: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<div dojoType="dijit.layout.AccordionPane" title="pane 1">
    Text of Pane 1
</div>
```
2. mix in any attributes in the same tag as the attachPoint section (except the attachPoint attribute itself): /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<div dojoType="dijit.layout.AccordionPane"
        title="pane 1" style='overflow: hidden; height: 1px; display: none'
        class='dijitAccordionBody' waiRole="tabpanel">
        Text of Pane 1
</div>
```
3. Pop the result into the template

This is a recursive procedure so the containerNode itself can contain widgets.

A Template for ImageAccordion

ImageAccordion only needs a few small changes to the AccordionPane template. All of the rest of the code for AccordionPane can be reused. So here is our new template: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div class='dijitAccordionPane'
        ><div dojoAttachPoint='titleNode,focusNode'
                dojoAttachEvent='ondijitclick:_onTitleClick,onkeypress:_onKeyPress'
                class='dojocAccordionTitle' wairole="tab"
                ><div class='dijitAccordionArrow'></div
                ><div class='arrowTextUp' waiRole="presentation">▲</div
                ><div class='arrowTextDown' waiRole="presentation">▼</div
                ><span dojoAttachPoint='titleTextNode'><img alt="${title}" src="${src}"
        ></span></div
        ><div><div dojoAttachPoint='containerNode'
                        style='overflow: hidden; height: 1px; display: none'
                dojoAttachEvent='onkeypress:_onKeyPress'
                class='dojocImageAccordionBody' waiRole="tabpanel"
        ></div></div>
</div>
```

What's with the > signs on different lines? That ensures extra whitespace is not included in the actual generated HTML. It makes the template slightly less readable, but pays big dividends in performance.

There are only a few changes from the original AccordionPane:

1. The most important change is the replacement of ${text} with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<img alt="${title}" src="${src}"/>`. This means the ImageAccordion must be sent an additional attribute - src - with the URL for the image.
2. The titleNode now has the CSS class "dojocAccordionTitle"
3. The containerNode now has the CSS class "dojocImageAccordionBody"

So now we have the heart of our new widget. Let's pop the template in and make things happen.

Widgets inside the Template

So what if we want the widget to have a widget inside of it, as in ...:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<div class="combinedDateTime">
   <div dojoType="dijit.form.DateTextBox"></div>
   <div dojoType="dijit.form.TimeTextBox"></div>
</div>
```

When using this template in a directly extended widget class, you will need to set the property /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `widgetsInTemplate: true`. Why? Because a widget inside a template requires some recursive parsing, which may be slow if you're drawing thousands of widgets ... especially if there is nothing extra to parse. Therefore, it is false by default.

dijit.Declaration-based widget classes automatically set widgetsInTemplate to true.

Dijit.Declaration

Just as there are two ways to create a widget instance - declarative and programmatic - so there are two ways to declare a Dijit class. As you might guess, the declarative way is slightly easier so we'll start there.

To declare your class declaratively, use dijit.Declaration. Uhhhh, OK, too many "declare"s in that sentence. It's easier to show than to tell. Here's the entire ImageAccordion spec. The code doesn't display anything itself, so it's best to place it right after the BODY tag before any other displayable code, or at the end.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.Declaration"
      widgetClass="dojoc.widget.ImageAccordion"
      mixins="dijit.layout.AccordionPane"
><div class='dijitAccordionPane'
     ><div dojoAttachPoint='titleNode,focusNode'
             dojoAttachEvent='ondijitclick:_onTitleClick,onkeypress:_onKeyPress'
             class='dojocAccordionTitle' wairole="tab"
             ><div class='dijitAccordionArrow'></div
             ><div class='arrowTextUp' waiRole="presentation">▲</div
             ><div class='arrowTextDown' waiRole="presentation">▼</div
             ><span dojoAttachPoint='titleTextNode'><img alt="${title}" src="${src}"
       ></span></div
       ><div><div dojoAttachPoint='containerNode'
             style='overflow: hidden; height: 1px; display: none'
             dojoAttachEvent='onkeypress:_onKeyPress'
             class='dojocImageAccordionBody' waiRole="tabpanel"
       ></div></div>
  </div>
</div>
```

Dijit.Declaration turns this markup into a templated widget class. The mixins attribute tells which classes ImageAccordion will be based on. You can include more than one class here - each separated by commas. The first class is not technically a mixin - it's the parent class of this new widget. So all of the methods in AccordionPane are inherited in ImageAccordion. Only the template changes.

Declaring extension point implementations, or connecting to events inside a dijit.Declaration is exactly the same as for a declaratively-defined individual widget. So:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dijit.Declaration" widgetClass="simpleConnectedWidget" >
   Just a plain ol' piece of text
   <script type="dojo/connect" event="dblclick">
      console.debug("Ouch!  I've been double-clicked");
   </script>
</div>
```

Every widget declared with class /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `simpleConnectedWidget` will have the handler connected to it.

Notes

- widgetsInTemplate is automatically set to true, so any widgets you place in the template will be automatically filled in.
- If you do not specify /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `mixin`, the widget class will be a subclass of dijit._Widget and mix in dijit._Templated. If you specify mixin, the first class listed must be a subclass of dijit._Widget. At least one of the mixins should itself mixin dijit._Templated, or you should supply dijit._Templated yourself as a mixin.
- Only one extension point implementation of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `preamble`.

Direct Extension

Every widget class is a Dojo class. So you should be able to extend a widget class just like you would any other Dojo class, right? Like with dojo.declare? Absolutely!

What do you gain by using dojo.declare instead of dijit.Declaration? Mostly you gain flexibility. For example, say you wanted to declare a widget class with the same functionality, but two completely different versions of a template based on a user preferences. No problem. You construct the template itself with JavaScript code, then pass it to dojo.declare in the template property.

But let's get back to ImageAccordion. Following the examples in Modules, we'll place this class dojoc.layout.ImageAccordion in a file named

dojoc/layout/ImageAccordion.js under the Dojo root. Here's how to write it in code:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
// all packages need to dojo.provide() _something_, and only one thing
dojo.provide("dojoc.widget.ImageAccordion");
// AccordionContainer is the module with dijit.layout.AccordionPane
dojo.require("dijit.layout.AccordionContainer");
// our declared class
dojo.declare("dojoc.widget.ImageAccordion",
        // we inherit from this class, which in turn mixes
        // in _Templated and _Layout
        [ dijit.layout.AccordionPane ],
        // class properties:
        {
        templatePath: dojo.moduleUrl("dojoc","layout/templates/ImageAccordion.html"),
        // Necessary to keep Dijit from using templateString in AccordionPane
        templateString: "",
        // src: String
        //      src url for AccordionPaneExtension header
        src: ""
});
```

The src string does pretty much what we think - unlike with dijit.Declaration, we must declare the new attributes explicitly. The templatePath requires some explanation.

Attaching the Template to a Direct Extension

With dijit.Declaration, the template is just the body of the tag. In dojo.declare land, there are three ways to specify a template:

- Use the templatePath attribute to point to a URL with a template in it.
- -OR- Specify the template directly in the templateString attribute
- -OR- Pass a DOM node with a parsed representation of the template

The first option is preferred because it separates the JavaScript and HTML code cleanly. With templateString, you must remember to escape all the quote marks, required in a JavaScript string. The Custom build system will convert the template in templateUrl to an inline templateString to help performance, so no need to worry there. **Note:** as in our above example, if you are overriding a widget with a templateString, and you want to use a templateUrl in your subclass, be sure to set templateString to "".

The templates are stored in the module along with all the JavaScript code. We place this template file into dojoc/widgets/template/ImageAccordion.html:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
<div class='dojocImageAccordion'
        ><div dojoAttachPoint='titleNode,focusNode'
dojoAttachEvent='ondijitclick:_onTitleClick,onkeypress:_onTitleKeyPress,onfocus:_handleFocus,onblur:_handleFocus'
                  class='dojocImageAccordionTitle' wairole="tab"
                ><img alt="${title}" src="${src}"
        ></div
        ><div><div dojoAttachPoint='containerNode'
                style='overflow: hidden; height: 1px; display: none'
                class='dojocImageAccordionBody' waiRole="tabpanel"
        ></div></div>
</div>
```

The Widget Life-cycle

The lifecycle of a widget decribes the phases of its creation and destruction which you can hook into. Its useful to understand exactly what happens when. Whether you are sub-classing an existing widget, using dojo/method script blocks, or passing in method overrides to the constructor, these are your entry points for making a widget do what *you* want it to do.

Note: some aspects of the widget lifecycle have changed post the 0.9 release; please consult the 0.9 to 1.0 porting guide.

Widgets are classes, created with dojo.declare. All widgets inherit from dijit._Widget, and most get the _Templated mixin. That provides you the following extension points (methods) you can override and provide implementation for:

constructor

Your constructor method will be called before the parameters are mixed into the widget, and can be used to initialize arrays, etc.

parameters are mixed into the widget instance

This is when attributes in the markup (ex: <button iconClass=...>) are mixed in or, if you are instantiating directly, the properties object you passed into the constructor (ex: new dijit.form.Button({label: "hi"})). This step itself is not overridable, but you can play with the result in...

postMixInProperties

If you provide a postMixInProperties method for your widget, it will be invoked before rendering occurs, and before any dom nodes are created. If you need to add or change the instance's properties before the widget is rendered - this is the place to do it.

buildRendering

_Templated provides an implementation of buildRendering that most times will do what you need. The template is fetched/read, nodes created and events hooked up during buildRendering. If you dont mixin _Templated (and most OOTB dijits do) and want to handle rendering yourself (e.g. to really streamline a simple widget, or even use a different templating system) this is where you'd do it.

postCreate

This is typically the workhorse of a custom widget. The widget has been rendered (but note that sub-widgets in the containerNode have not!)

startup

If you need to be sure parsing and creation of any child widgets is complete, use startup.

destroy

Implement destroy if you have special tear-down work to do (the superclasses will take care of most of it for you; be sure to call this.inherited(arguments);)

In all cases its good practice to assume that you are overriding a method that may do something important in a class up the inheritance chain. So, call the superclass' method before or after your own code. E.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
postCreate: function() {
   // do my stuff, then...
   this.inherited("postCreate", arguments);
}
```

Testing the Widget

Does this thing actually work? Here's a unit test page to find out. We simply added some src="image.png" attribs to the AccordionContainer children in Dijit's AccordionContainer test page. You will need to add your own image.png to dojoc/widgets/images, or change the path. Here there are three 3 simple ImageAccordion's all 200px x 26px. Add in some basic CSS for the classes we changed, and voila:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
        <title>dojoc.widget.ImageAccordion Test</title>
        <style type="text/css">
                /* several CSS files you probably want: */
                @import "dojo/dojo/resources/dojo.css";
                @import "dojo/dijit/themes/tundra/tundra.css";
                @import "dojo/dijit/themes/dijit.css";
                /* this one is just for Dojo tests, but we'll use it in
                    hopes they will accept our code */
                @import "dojo/dijit/tests/css/dijitTests.css";
                .dojocImageAccordion .dojocImageAccordionBody {
                     background: #fff; margin:0; padding:0;
                }
                .dojocImageAccordion { border-right:1px solid #ededed; }
                .dojocImageAccordionTitle { cursor:pointer; }
                p { padding:2px; margin-top:2px; }
        </style>
        <script type="text/javascript" src="dojo/dojo/dojo.js"
                    djConfig="isDebug:true, parseOnLoad: true"></script>
        <script type="text/javascript">
                // Comment this out to use the Declarative Version
                dojo.require("dojoc.widget.ImageAccordion");
                dojo.require("dijit.layout.AccordionContainer");
                dojo.require("dojo.parser");
        </script>
</head>
<body class="tundra">
        <div><!-- The Declarative version goes here.  --></div>
        <h1>dojoc.widget.ImageAccordion</h1>

        It's poorly named, I know. An "ImageAccordion" would be an
        AccordionContainer of "ImagePane"'s (but one does not exist).
        This is an example of an AccordionContainer, with Images
        exclusively as their Titles, named "dojoc.widget.ImageAccordion".

        <div dojoType="dijit.layout.AccordionContainer" duration="300"
                style="width:200px; height:400px; background:#fff; ">
                <div dojoType="dojoc.widget.ImageAccordion" title="Node 1"
                        selected="selected" src="images/title1.png">
                        Hello World
                </div>
                <div dojoType="dojoc.widget.ImageAccordion" title="Node 2"
                        src="images/title2.png">

                        Nunc consequat nisi vitae quam. Suspendisse sed nunc. Proin
                        suscipit porta magna. Duis accumsan nunc in velit. Nam et nibh.
                        Nulla facilisi. Cras venenatis urna et magna. Aenean magna mauris,
                        bibendum sit amet, semper quis, aliquet nec, sapien.  Aliquam
                        aliquam odio quis erat. Etiam est nisi, condimentum non, lacinia
                        ac, vehicula laoreet, elit. Sed interdum augue sit amet quam
```

```
                        dapibus semper. Nulla facilisi. Pellentesque lobortis erat nec
                        quam.


                        Sed arcu magna, molestie at, fringilla in, sodales eu, elit.
                        Curabitur mattis lorem et est. Quisque et tortor. Integer bibendum
                        vulputate odio. Nam nec ipsum. Vestibulum mollis eros feugiat
                        augue. Integer fermentum odio lobortis odio. Nullam mollis nisl non
                        metus. Maecenas nec nunc eget pede ultrices blandit. Ut non purus
                        ut elit convallis eleifend. Fusce tincidunt, justo quis tempus
                        euismod, magna nulla viverra libero, sit amet lacinia odio diam id
                        risus. Ut varius viverra turpis. Morbi urna elit, imperdiet eu,
                        porta ac, pharetra sed, nisi. Etiam ante libero, ultrices ac,
                        faucibus ac, cursus sodales, nisl. Praesent nisl sem, fermentum eu,
                        consequat quis, varius interdum, nulla. Donec neque tortor,
                        sollicitudin sed, consequat nec, facilisis sit amet, orci. Aenean
                        ut eros sit amet ante pharetra interdum.

                </div>
                <div dojoType="dojoc.widget.ImageAccordion" title="Third Pane"
                        src="images/title3.png">

                        The quick brown fox jumps over the lazy dog.  The quick brown
                        fox jumps over the lazy dog.  The quick brown fox jumps over the lazy dog.

                </div>
        </div>
        <!-- filler text to show HTML-after -->

                Sed arcu magna, molestie at, fringilla in, sodales eu, elit.
                Curabitur mattis lorem et est. Quisque et tortor. Integer bibendum
                consequat quis, varius interdum, nulla. Donec neque tortor,
                sollicitudin sed, consequat nec, facilisis sit amet, orci. Aenean
                ut eros sit amet ante pharetra interdum.

</body>
</html>
```

Example: File Upload Dialog Box

So now, we'll build a widget to solve a practical problem. Your site is all neatly designed and feng-shui'ed ... except this one /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .html4strict .imp {font-weight: bold; color: red;} .html4strict .kw1 {color: #b1b100;} .html4strict .kw2 {color: #000000; font-weight: bold;} .html4strict .kw3 {color: #000066;} .html4strict .coMULTI {color: #808080; font-style: italic;} .html4strict .es0 {color: #000099; font-weight: bold;} .html4strict .br0 {color: #66cc66;} .html4strict .st0 {color: #ff0000;} .html4strict .nu0 {color: #cc66cc;} .html4strict .sc0 {color: #00bbdd;} .html4strict .sc1 {color: #ddbb00;} .html4strict .sc2 {color: #009900;} `<input type="file">` tag. The user clicks on the button, and all of a sudden gets an operating system-styled box that doesn't look right, and is completely out of your control. Ugggh.

This is a perennial thorn-in-the-side for web developers. A Google search finds the ever-useful web site quirksmode, and they have a pretty simple solution: hide the file input, put a real input directly beneath where it would be, and make your own button off to the side. Logical enough. How hard would that be to make a widget?

Not hard at all, it turns out. Here's the plan:

1. Hide the input boxes and draw our own
2. Use Dojo to handle all my connections and fancy stuff
3. Wrap it all up using /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dijit.form._FormWidget` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `diijt._Templated` to take care of everything else

Hiding the Built-In File Upload Box

We'll use the programmatic method here. First we'll need a location on-disk for our widget. Since others have expressed interest in this widget, and we hope to contribute it to dojox, we'll place the widget in dojox/widget/FileInput.js. This means we'll be able to require() and provide() the module "dojox.widget.FileInput" as per the package system conventions. To get additional behavior we'll need to require() mixin classes, in case we don't already have them. So here's a stub for the class:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.provide("dojox.widget.FileInput");
dojo.require("dijit.form._FormWidget");
dojo.require("dijit._Templated");
dojo.declare("dojox.widget.FileInput",
        [dijit.form._FormWidget,dijit._Templated],
        {
        // summary: A styled input type="file"
        //
        // description: A input type="file" form widget, with a button for uploading to be styled via css,
        //      a cancel button to clear selection, and FormWidget mixin to provide standard
        //      dijit.form.Form
        //
});
```

Next we'll need a template, a label for our submit button, a label for our cancel button, and the name of the input:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// label: String
        //      the title text of the "Browse" button
        label: "Browse ...",
        // cancelText: String
        //      the title of the "Cancel" button
        cancelText: "Cancel",
        // name: String
        //      ugh, this should be pulled from this.domNode
        name: "uploadFile",
        templatePath: dojo.moduleUrl("dojox.widget","FileInput/FileInput.html"),
```

Also note that whenever we use template variables , it's good practice to supply a default, e.g. "Browse ..." for the label. Otherwise, if your widget user omits the label attribute, the parser will complain.

As we said earlier in Direct Extension, it's preferrable to separate the template out into a different file. Ours will look like the following, placed in dojox/widget/FileInput/FileInput.html: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div class="dijitFileInput">
        <input id="${id}" class="dijitFileInputReal" type="file"
                dojoAttachPoint="fileInput" name="${name}" />
        <div class="dijitFakeInput">
                <input class="dijitFileInputVisible" type="text"
                        dojoAttachPoint="focusNode, inputNode" />
                <span class="dijitFileInputText"
                        dojoAttachPoint="titleNode">${label}</span>
                <span class="dijitFileInputButton" dojoAttachPoint="cancelNode"
                        dojoAttachEvent="onclick:_onClick">
                        ${cancelText}
                </span>
        </div>
</div>
```

Note how we assign classes to each major part, so we can apply design as CSS styles. We're also using /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojoAttachPoint="aString"` so the parser makes these nodes available to us in code at the location /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `this.fileInput`.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojoAttachEvent="onclick: _onClick"` connects the onclick event of this.cancelNode to this._onClick, the method we are about to define (otherwise dojo.hitch will throw an error mentioning something about _onClick not having properties). We'll get to the onClick handler for the cancel button, and the reasons for having to do it later.

What is the following for?

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// nonRequired: String
//    this way, the build system knows about me
nonRequired: "",
```

for now, lets make sure our widget starts up, and looks right. lets make some simple css rules using the class names we set in our template, based on the relative / absolute, opacity:0 stuff we learned earlier:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}

```
.dijitFileInput {
        position:relative;
        height:1.3em;
        padding:2px;
}
.dijitFileInputReal {
        position:absolute;
        z-index:2;
        opacity:0;
        filter:alpha(opacity:0);
```

```
}
.dijitFileInputButton,
.dijitFileInputText {
        border:1px solid #333;
        padding:2px 12px 2px 12px;
        cursor:pointer;
}
.dijitFileInputButton {
        opacity:0;
        filter:alpha(opacity:0);
        z-index:3;
        visibility:hidden;

}
.dijitFakeInput { position:absolute; top:0; left:0; z-index:1; }
```

looks good!

Connecting the Elements with Events

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojoAttachEvent="onclick: _onClick"` connects the onclick event of this.cancelNode to this._onClick, the method we are about to define (otherwise dojo.hitch will throw an error mentioning something about _onClick not having properties). We'll get to the onClick handler for the cancel button, and the reasons for having to do it later.

We also need to implement a simple onchange listener, like the article hints, so that when our onchange is detected in on our real file input (this.fileInput), we will call this._matchValue() to steal the value from it, and populate our visible input:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
startup: function(){
                // summary: listen for changes on our real file input
                this.inherited("startup",arguments);
                this._listener = dojo.connect(this.fileInput,"onchange",this,"_matchValue");
                this._keyListener = dojo.connect(this.fileInput,"onkeyup",this,"_matchValue");
        },
        _matchValue: function(){
                // summary: set the content of the upper input based on the semi-hidden file input
                this.inputNode.value = this.fileInput.value;
                if(this.inputNode.value){
                        this.cancelNode.style.visibility = "visible";
                        dojo.fadeIn({ node: this.cancelNode, duration:275 }).play();
                }
        }
```

You've probably also noticed we added an onkeyup connection, running the same code. This way, if we type in the input, our changes will be reflected after each key press. Fortunately for this example, we'll ignore little nitpicks like "holding backspace doesn't fire onkeyup".

The _matchValue() function simply steals the file input value, sets it to the visible input value and fades in the cancel button (which we set earlier to visbility:hidden in FileInput.css).

So next, we need a reset button. Unfortunately because we're faking HTML out a bit, a plain old Reset button won't work. Since we aren't allowed write access to the file input, we can't just null the data. So our _onClick method actually destroys the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .html4strict .imp {font-weight: bold; color: red;} .html4strict .kw1 {color: #b1b100;} .html4strict .kw2 {color: #000000; font-weight: bold;} .html4strict .kw3 {color: #000066;} .html4strict .coMULTI {color: #808080; font-style: italic;} .html4strict .es0 {color: #000099; font-weight: bold;} .html4strict .br0 {color: #66cc66;} .html4strict .st0 {color: #ff0000;} .html4strict .nu0 {color: #cc66cc;} .html4strict .sc0 {color: #00bbdd;} .html4strict .sc1 {color: #ddbb00;} .html4strict .sc2 {color: #009900;} `<input type="file">` and reads it.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
_onClick: function(/* Event */e){
        // summary: on click of cancel button, since we can't clear the input because of
        //      security reasons, we destroy it, and add a new one in it's place.
        // Disconnect the listeners so they're not orphaned, and cleanly remove the tag
        dojo.disconnect(this._listener);
        dojo.disconnect(this._keyListener);
        this.domNode.removeChild(this.fileInput);
        // Fade our the cancel button so we no longer can press it
        dojo.fadeOut({ node: this.cancelNode, duration:275 }).play();
        // Create an identical input tag
        this.fileInput = document.createElement('input');
        this.fileInput.setAttribute("type","file");
        this.fileInput.setAttribute("id",this.id);
        this.fileInput.setAttribute("name",this.name);
        dojo.addClass(this.fileInput,"dijitFileInputReal");
        // this.domNode is the root DOM node of the widget
        this.domNode.appendChild(this.fileInput);
        // Finally, connect the listeners to this new node.
        this._keyListener = dojo.connect(this.fileInput,"onkeyup",this,"_matchValue");
        this._listener = dojo.connect(this.fileInput,"onchange",this,"_matchValue");
        this.inputNode.value = "";
}
```

Wiring It All Together

So now we have our widget and our basic styles. We include this widget in our page declaratively by:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<input type="file" name="uploadFile" dojoType="dojox.widget.FileInput">
```

The real file input tag in the DOM goes away our templated input gets put in it's place. But if JavaScript is not present, it stays a regular HTML input tag, so it degrades nicely. Here is a sample test page to work with:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
        <title>dojox.widget.FileInput | The Dojo Toolkit</title>
        <style type="text/css">
                @import "../../../dojo/resources/dojo.css";
                @import "../../../dijit/themes/dijit.css";
                @import "../FileInput/FileInput.css";
        </style>
        <script type="text/javascript" src="../../../dojo/dojo.js"
                djConfig="isDebug:true, parseOnLoad: true"></script>
        <script type="text/javascript">
                dojo.require("dojox.widget.FileInput");
                dojo.require("dojo.parser");    // scan page for widgets and instantiate them
        </script>
</head>
<body>
        <h3>A standard file input:</h3>
        <input type="file" id="normal" name="inputFile" />
        <h3>The default dojox.widget.FileInput:</h3>

        <input dojoType="dojox.widget.FileInput" id="default" name="uploadFile" />
```

In theory, it will work inside of a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<form>` tag just as the original element did.

Because this is dojox, and we're bigs fan of re-using code, we can steal some CSS stuff from tundra.css and soria.css to provide theme-specific styles, so our input nodes look like they would in with all the other dijit.form Widgets (like ComboBox, FilteringSelect, ValidationTextBox, etc) ...

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000000; font-weight: bold;} .geshifilter .kw2 {color: #993333;} .geshifilter .co1 {color: #a1a100;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #933;} .geshifilter .re0 {color: #cc00cc;} .geshifilter .re1 {color: #6666ff;} .geshifilter .re2 {color: #3333ff;} .geshifilter .re3 {color: #933;} .geshifilter .re4 {color: #933;}
```
/* tundra */
.tundra .dijitProgressOverlay {
        border:1px solid #84a3d1;
        background-color:#cad2de;
}
.tundra .dijitFakeInput input {
        font-size: inherit;
        background:#fff
            url("../../../dijit/themes/tundra/images/validationInputBg.png")
            repeat-x top left;
        border:1px solid #9b9b9b;
        line-height: normal;
        padding: 0.2em 0.3em;
}
.tundra .dijitFileInputButton,
.tundra .dijitFileInputText {
        border:1px solid #9b9b9b;
        padding:2px 12px 2px 12px; /* .3em .4em .2em .4em; */
        background:#e9e9e9
            url("../../../dijit/themes/tundra/images/buttonEnabled.png")
            repeat-x top;
}
/* Soria */
.soria .dijitProgressOverlay {
        border:1px solid #333;
        background-color:#cad2de;
}
.soria .dijitFakeInput input {
        border:1px solid #333;
        background:#fff
            url("../../../dijit/themes/soria/images/gradientInverseTopBg.png")
            repeat-x top left;
        line-height:normal;
        background-position:0 -30px;
        padding:0.2em 0.3em;
}
.soria .dijitFileInputButton,
.soria .dijitFileInputText {
        border:1px solid #333;
```

```
        padding:2px 12px 2px 12px;
        background:#b7cdee
            url('../../../dijit/themes/soria/images/gradientTopBg.png') repeat-x;
}
```

Future Plans

If you've been paying attention all this way, you probably noticed a class up there that we didn't define. .dijitProgressOverlay ... it doesn't exist anywhere in the dojox.widget.FileInput template or code.

The Dojo folks are building an extension to this widget called dojox.widget.FileInputAuto. It works like FileInput except it submits itself after a delay following a blur() on the element. For details (as yet undocumented), get the latest nightly build of Dojo and look for it in dojox.widget.

Creating Accessible Widgets

Device Independent behavior means more than just supporting the keyboard. Where ever possible use the most generic event handler available. For example, consider a widget where the down arrow key selects an element in the widget. The selection needs to be distinguished with a specific style. Rather than modifying the style of the element when processing the down arrow key event, focus the item from the down arrow key event handler and change the style via a focus event handler. This way, if focus is set from a means other than the keyboard such as a voice input system, the styling is properly set and does not depend solely on keyboard actions.

Determining Keyboard Behavior

When implementing keyboard navigation, the ideal solution is to mimic the behavior of the operating system. For example, the right and left arrow keys are used to expand and collapse nodes in a Windows tree control and the up and down arrow keys move between nodes in the control. Unfortunately it is not always possible to mimic the operating system or browser behavior because the widgets may not be able to capture the necessary keys. A group of industry representatives are working to create a style guide to describe the navigation and behaviors of Web widgets. When completed, this Style Guide will be provided to open source and dijit plans to implement the recommendations. Eventually, the Style Guide will attempt to normalize the differences between operating systems and provide a generalized solution for Web components

Within all widgets interaction with both the keyboard and the mouse is important – users may switch between using the mouse and using the keyboard at any time. A widget author can not assume only keyboard or only mouse interaction. Thus, the widget component will generally need to store information about the current item with focus. This can also be useful when the keyboard event handler is placed on an owning object in the component hierarchy rather than the actual element generating the event – for example on the table element rather than on each td element. Even though the event handler provides information on exactly what element generated the event, it is often necessary or easier to use the stored point of reference In order to support both the mouse and standard enter key and space key press a dijit widget will implement an onDijitClick event hander. This handler must include steps to update the point of regard so that any additional keyboard actions after the onDijitClick will continue to work. In addition, the point of regard is often needed in order to update the style on the element losing focus before updating the new item irregardless or whether the mouse or the keyboard generated the event that results in a focus change.

Trapping Key Events

When implementing keyboard navigation, first determine where in the hierarchy to trap the key events. It is generally best to trap the key events at as high a level as possible and use the event object to determine that actual source of the event and perform the necessary action. This method prevents having to add a key handler to each individual element thus conserving the amount of markup to be generated. However, there may be cases where the event needs to be trapped at the level of each individual element. The actual source of the event is needed in order to determine how to process the keystroke received.

Once the component handles an event, it will usually stop that event from being propagated to other elements. For example, if the down arrow key is captured and moves focus to the next item in a tree control, the event should not propagate up to the browser where it might be interpreted as a command to scroll the page. Use the dojo.stopEvent(event) method to stop the event.

In order to assist with key event handling, an onkeypress event has been added to Dojo to normalize key events. The appropriate key event, either onkeydown or onkeypress, will be used depending upon the browser. The key codes have been normalized as well. See dojo.keys class in dojo._base.event.js. Add the dojo onkeypress event into the widget template or via scripting using one of the dojo event connection apis.

Example using dojo.connect to connect the object represented by the node variable to the onKey handler function in the current object:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;}
```
dojo.connect(node, "onkeypress", this, "onKey");
```

Example from tab container template to demonstrate adding the onkey handler and waiRole within the template:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div class="dijitTabContainer">
        <div dojoAttachPoint="tablistNode"></div>
        <div class="dijitTabPaneWrapper" dojoAttachPoint="containerNode" dojoAttachEvent="onkeypress:_onKeyPress"
waiRole="tabpanel"></div>
</div>
```

Tabindex and Focus

When navigating via the keyboard is it essential that the element that is navigated to receives focus. The focus should NOT be simulated via CSS - call the focus() method on the element. Styling can be used to enhance the visual focus or selection but should not replace

actually setting focus on an element. A screen reader will only speak information about the element when it receives focus. Screen magnifiers rely on focus to move the zoomed viewport on the screen.

Use the tabindex value to provide direct or programmatic keyboard focus to an element. See the tabindex chart in the Dojo Accessibility Strategy Device Independence section.

When adding support for keyboard navigation, consider the widget as a component. The tab key can be used to navigate from component to component on a page and then the arrow and other keys should be used to navigate within the component. Only one element in a given component should have a tabindex equal to zero at any one time. This allows the user to navigate into and set focus within in the component using the tab key. Then, trap the onkey events and use the arrow keys to navigate within the elements of the component. All of the elements within the component which can receive focus must have a tabindex equal to -1. When an element is programmatically given focus, its tabindex value is changed from -1 to 0 and the tabindex of the previous element with focus will be changed from 0 to -1. This will insure that only one element within the component is in the tab order of the page and that the element with tabindex = 0 is the most recently focused element in the component.

For example, when creating a tree control, each tree item is represented by an element. The first tree item in the control will be given a tabindex of 0. All of the other elements which represent tree items and can receive focus programmatically will have a tabindex value of -1. An onkeypress handler will trap the keyboard events for the tree control. When a tree item element is given focus via element.focus(), the element's tabindex will be changed from -1 to 0. and it will be put into the tab order. Now if the user moves focus out of the tree control (either via a mouse click or by tabbing to the next component on the page), when the user sets focus back into the tree control using the tab key, the last focused tree item, which was given a tabindex of 0, will receive focus.

Add ARIA information

The Accessible Rich Internet Applications Roadmap is being developed by the W3C Web Accessibility Initiative (WAI) Protocols and Formats working group. The group is creating specifications for role and state information which can be added to markup to provide semantic information about user interface components. The browsers will translate this role and state information into the accessibility api for the platform in use. Currently Firefox 1.5 and later supports this additional semantic information on the Windows platform where it converts the information into the Microsoft Active Accessibility (MSAA) api. When recent versions of the Window-Eyes and JAWS screen readers are used with Firefox, this additional information is spoken to the user.

Some HTML elements such as links and form elements have well defined roles and behaviors. Interactive controls created from generic elements can now also be identified with roles and states. When an element receives focus the role and state information provided by the developer will be made available to assistive technologies. For example, as a screen reader traverses through a dijit tree control using the arrow keys, as each tree item receives focus the title of the tree item will be spoken as well as its expanded or collapsed state if it has children. Likewise, a dijit checkbox created using and tags can be identified as a checkbox and its checked or unchecked state can be reported. When creating a new Dojo widget, the role of the widget must be identified and the state of the widget must be set and updated as it changes.

As of August, 2007, the ARIA specifications are still under development; however a significant portion of the specification has been implemented in Firefox 1.5 and 2.0. The public drafts of the specifications can be found at http://www.w3.org/wai/pf

- Roles for Accessible Rich Internet Applications (ARIA Roles) In addition to the listing of roles, this includes valuable information and examples for creating accessible components.
- States and Properties Module for Accessible Rich Internet Applications (ARIA States and Properties)

Accessible DHML in the Mozilla Developer Center provides additional information about using the ARIA specifications and includes a table of roles and states supported in current versions of Firefox (http://developer.mozilla.org/en/docs/Accessible_DHTML#Supported_roles_and_properties)

Assigning Roles

Use the tabindex to provide keyboard focus or to allow programmatic focus to an object. By adding a tabindex to an element, the element will now be included in the accessibility hierarchy of the Firefox browser. Information about elements in the accessibility hierarchy will be provided to assistive technologies. If you use a tabindex attribute on a <DIV>, <SPAN>, <IMG> or any element which has no natural role of its own then you need to provide a role. Any element that can receive focus must have a role, either implied, such as input elements and anchors, or specified via a role attribute. For things with an implied role such as input fields and anchors, you can use tabindex="-1" to remove them from the tab order. You can also specify a different role for elements which already have an implied role.

The role and states are added to Dojo widgets within the widget template or via the dijit.wai APIs as described in the Accessibility Resources section.

Providing Hierarchical Information

In order for Firefox to determine the correct parent child relationships between objects, and to communicate this via an accessibility API to assistive technologies, it is best to create components in a hierarchical fashion. For example, when creating a menubar it is best to have the components that make up the menus and menuitems of the menubar be children of the menubar. Likewise, menuitems should be children of the owning menus. This hierarchy allows Firefox to provide menu information to the assistive technologies, and for a screen reader to speak more information about the menu such as, "menu open, File, item 1 of 5" when the user opens a menu. Here is a simple pseudo code example demonstrating a hierarchical layout of elements for a menu control. This exmaple only shows the addition of role attributes and does not represent a complete menu widget. (Note: Attributes are not quoted in pseudo-code examples to help improve the readability):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div role="menubar">
<div role="menuitem">A</div>
      <div role="menu">
            <div role="menuitem">A.1</div>
            <div role="menuitem">A.2</div>
      </div>
```

```
      <div role="menuitem">B</div>
      <div role="menu">
            <div role="menuitem">B.1</div>
            <div role="menuitem">B.2</div>
            <div role="menu">
                  <div role="menuitem">B.2.1</div>
            </div>
      </div>
</div>
```

It may not always be practical to create items via HTML in a hierarchical fashion. In that case the group role can help to associate the items properly. This is illustrated in the following simple pseudo code example of a tree hierarchy.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div role="tree">
  <div role="treeitem">Top </div>
  <div role="group">
     <div role="treeitem">1</div>
     <div role="group">
       <div role="treeitem">1.1 </div>
       <div role="treeitem">1.2</div>
       <div role="treeitem">1.3</div>
<div role="group">
<div role="treeitem">1.3.1</div>
         <div role="treeitem">1.3.2</div>
         <div role="treeitem">1.3.3</div>
         <div role="treeitem">1.3.4</div>
       </div>
       <div role="treeitem">1.4</div>
     </div>
     <div role="treeitem">2</div>
     <div role="treeitem">3</div>
     <div role="group">
       <div role="treeitem">3.1</div>
       <div role="treeitem">">3.2</div>
     </div>
  </div>
</div>
```

The tree items at the same level in the hierarchy are grouped together within a

element identified with role=group. With this organization, the assistive technologies can be provided with the information about what level and item number a particular treeitem represents. For example, in the above tree example, with focus on item 1.3.3 a screen reader might speak, “one dot three dot three item three of four, level four” or something similar.

Other items included in the hierarchy may not be essential to the component. These items can be marked with a role of presentation to eliminate them from consideration when determining information about the component.

Using the Presentation Role

While it is preferable to use CSS for layout, tables are still used to quickly and easily arrange elements on a page. This is especially true of existing widgets which were originally created to work in older browsers. Putting information in tables can easily confuse the hierarchy of the component. If tables must be used, they can be marked with a role of presentation to eliminate them from the hierarchy. Here is a pseudo code example where the presentation role was used on tables within a tree component:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div role="tree">
      <table role="presentation">
            <tr><td><div role="treeitem">Top</div></td></tr>
      </table>
      <div role="group">
            <table role="presentation">
                  <tr><td><span role="treeitem">1</span></td></tr>
            </table>
            <div role="group">
                  <table role="presentation">
                        <tr><td><span role="treeitem">1.1</span></td></tr>
                  </table>
                  <table role="presentation">
                        <tr><td><span role="treeitem">1.2</span></td></tr>
                  </table>
            </div>
            <table role="presentation">
                  <tr><td><span role="treeitem">2</span></td></tr>
            </table>
      </div>
</div>
```

Since the table is only used for layout it is identified with a role of presentation to remove if from the accessibility hierarchy so that information about the table is not provided to assistive technology. Other elements may need to be removed from the accessibility hierarchy as well. For example, when creating a DHTML checkbox, an image may be contained within a span element that is marked with a role of checkbox and an appropriate state of checked equals true or false. The image which represents the checkbox is contained within the span and should not contain any alt text since the role and state are managed by the surrounding span. Images are considered important elements and are normally included with the accessibility hierarchy of the browser. In order to ignore this image in the accessibility hierarchy, it is marked with a role of presentation. Here is a generic HTML representation:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:

bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<span role="checkbox" checked="true">
<img src="checked.gif" role="presentation">
</span>
```

Assigning States

In addition to identifying the role of a widget, the state of the widget must be identified and updated. The initial state can be set within the widget template or via scripting when the widget is created. As the state changes during user interaction with the widget, the state must be updated using the dijit.wai apis:

Using Dojo 0.9

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.wai.setAttr(/*DomNode*/node, /*String*/ ns, /*String*/ attr, /*String|Boolean*/value);
dijit.wai.getAttr(/*DomNode*/node, /*String*/ ns, /*String*/ attr, /*String|Boolean*/value);
dijit.wai.RemoveAttr(/*DomNode*/node, /*String*/ ns, /*String*/ attr, /*String|Boolean*/value);
```

The ns value passed into these functions is either “waiState” or “waiRole”. The dijit.wai functions above are wrappers to the DOM apis to set, get and remove attributes. In browsers where namespaces are supported the setAttributeNS, getAttributeNS, and removeAttributeNS, apis are called. In other browsers the setAttribute, getAttribute and removeAttribute apis are called and the namespace is simulated. The namespace information is stored in the dijit.wai class.

Using Dojo 1.0

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dijit.hasWaiState(/*Element*/ elem, /*String*/ state);
dijit.getWaiState(/*Element*/ elem, /*String*/ state);
dijit.setWaiState(/*Element*/ elem, /*String*/ state, /*String*/ value);
dijit.removeWaiState:(/*Element*/ elem, /*String*/ state);
```

It is important to update the state information as it changes so assistive technology users can be made aware of the change. For example, when a treeitem is expanded, the state for the element that has been assigned role="tree"item, must be set to expanded=true. Likewise, when a treeitem is collapsed, the state for the element with the role="tree"item must be updated to expanded=false. Be aware that some of the boolean states imply more than just a dual state. For the state attributes checked, selected and expanded a value of false indicates that the widget is capable of being checked, selected or expanded while no attribute indicates that the element is not capable of that state. For example, a tree node with children will have either a state of expanded=true or expanded=false depending upon whether the child nodes are visible or not. An end node, with no children will have no expanded state value set.

Generally only items which have a role can have a state value. The role may be explicitly set by the author such as a treeitem or may be implicitly defined such as a form element or link. Items which have been added into the accessibility hierarchy via a tabindex attribute may also have properties such as desribedby, labelledby, required, invalid and others.

Testing Widgets for Accessibility

Currently the Windows operating system provides the most resources and functionality for testing for accessibility in more than one browser. Also, the majority of assistive technologies run under the Windows operating system. Test with both Firefox and Internet Explorer on at least the Windows operating system to assure at least a minimal level of accessibility.

Test for full keyboard support

All widgets must be tested for use with the keyboard only. The most stringent way to test for keyboard support is to remove the mouse from the system and interact with the widget. This assures that only the keyboard can be used for navigation and interaction. Test that all functionality of the widget can be accomplished using the keyboard only. The functionality does not necessarily have to be performed in exactly the same manner as with the mouse. For example, it is preferred that a slider can be adjusted by dragging with the mouse or focusing the slider and adjusting the value using the arrow keys or plus and minus characters on the keyboard. But if the value and position of a slider can be adjusted by entering a new value in an associated text field, the slider is keyboard accessible. Drag and drop operations are another example of behavior which may need to be supported via an alternative mechanism such as a menu bar or context menu.

After determining that the widget is accessible using only the keyboard, test using both the keyboard and the mouse. There is no guarantee that a user will interact only using the keyboard or only using the mouse. Make certain that both types of interaction can be used within the widget and that the focus and styles are updated appropriately.

Since not all browsers support adding the tabindex to elements to make them focusable and thus keyboard accessible, it may not be possible to provide full keyboard support in all instances. As of August 2007, Opera and Safari do not support adding tabindex to additional elements.

Resources for Navigating via the keyboard

- Mozilla Keyboard Planning FAQ and Cross Reference
- Common keyboard shortcuts in Firefox, and the equivalents in Internet Explorer and Opera
- Windows Keyboard Interface Summary

- Keyboard Shortcuts for Internet Explorer 6
- The Keyboard Lover's Guide to IE7
- Use Opera without a Mouse

Test for focus and role and state

It is important that when an element is clicked on with the mouse or navigated to via the keyboard that the element receives actual focus. Focus must not be simulated via styles since assistive technology relies on the focus event to inform the user about the element and the role and state. Focus can be visually tested since browsers will indicate focus, however the focus border may be difficult to see or have been modified with a different style.

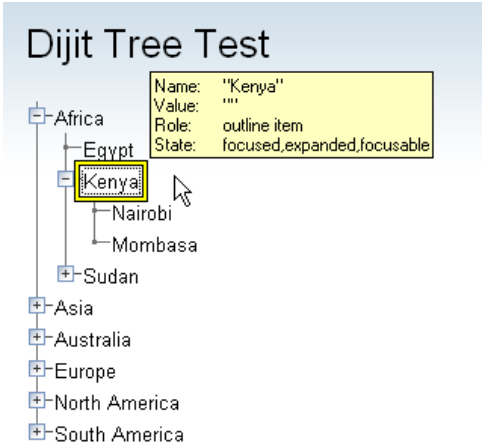
Testing using Microsoft Inspect Objects

The Microsoft Inspect Objects application from the Microsoft Active Accessibility SDK can be used on Windows to test for focus and also to verify the correct ARIA role and state has been set on the element. Test using Firefox version 2 or later since Firefox has the most comprehensive support for the ARIA role and state information.

Download Inspect32 from Active Accessibility 2.0 SDK Tools. Basic information on using the tool is provided at Using Inspect Objects.

Run Inspect Objects and test the widget for focus and role and state information. Turn on Highlight Focus tracking via the Inspect Objects Options menu. This mode will display a yellow rectangle around the currently focused element. Use this rectangle to verify that the expected element is actually receiving focus. The Show Information Tooltip option may also be helpful. As you set focus to an item, Inspect Objects will display information about the role and state of the object. You can verify that the object receives focus and that the role and state information is as expected.

Below is a picture of the diijt Tree with Inspect Objects running and the Show Information Tooltip and Turn On Highlight Focus options set. With focus on an expanded tree item the MS Inspect focus rectangle and role and state information is displayed within the tooltip. The role is "outline item" and the state is "focused, expanded, focusable". This verifies that the role of treeitem (interpreted as outline item by MS Inspect) has been properly set in the dijit tree item code. The state information verifies that the element is focused and the expanded state is set. With focus on an collapsed tree item node the state would display, "focused, collapsed, focusable", since the expanded property is set to false. Note that when the state of an element changes, focus must be removed from the element and then returned in order for Inspect Objects to update the information. Or, rather than change focus use the Inspect Objects Action Refresh command to update the information about the currently focused object.



Testing using a screen reader

For best results, testing with a screen reader is recommended. This section provides a brief introduction to using screen readers. Currently Window-Eyes 5.5 and JAWS 7.10 or later versions of both running with Firefox 2 or later will recognize ARIA role and state information. Unfortunately it takes some time and experience to become proficient using a screen reader and these programs are expensive. There are demo versions available which will run for a limited amount of time (usually 30 minutes) before a reboot of the system is required). But, note that often the demo versions are not to be used for commercial purposes.

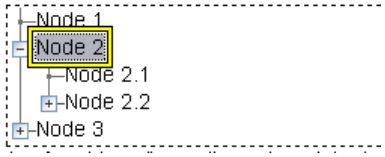
When testing with a screen reader be careful not to use the mouse when navigating through the widget. Often a tester will miss problems with keyboard navigation or focus by using the mouse to initiate or force interaction within a widget. Not all screen reader users are blind – some may have low vision or use the screen reader to assist with cognitive disabilities, but, in order to catch all problems it is important to interact with the widget in the same manner as someone with no vision.

Screen readers work by storing the contents of the entire page into a virtual buffer. The user can move through this buffer and have the contents spoken by the screen reader. There are many mechanisms to move through the virtual buffer, by character, word, line, or from object to object. Both JAWS and Window-Eyes have different terminology for this buffer. Window-Eyes refers to it as "browse mode" and JAWS as "virtual PC cursor mode". When interacting with a Web page, the screen reader can no longer rely on the virtual buffer. In Window-Eyes the user must turn browse mode off to interact with form elements and interactive widgets. In JAWS the user must enter forms mode on or turn virtual PC cursor mode off to interact with form controls and interactive widgets.

When testing fully keyboard accessible Dojo widgets the screen readers must be in the "interactive mode" to announce the role and state of an element as the element receives focus. In Window-Eyes this is "browse mode off" which is toggled using the ctrl-shift-a keys. In JAWS this is "forms mode on" which is turned on by pressing enter on a focusable element and is turned off by pressing the numpad plus key. JAWS also has a toggle to turn virtual PC cursor mode on and off, insert-z that can be used instead of forms mode.

If the role and state information have been applied correctly the screen reader will hear information about the role and stated of widget elements as well as information about child objects. The image below is a simple tree control with three main nodes with titles Node1 , Node 2, and Node 3. Node 1 has no children. Node 2 is expanded and has two child nodes titled Node 2.1 and Node 2.2. Node 3 has children but is not expanded. Here is a brief example of what is spoken by Window-Eyes with focus on the Node 2 tree item within the tree control

displayed in the image below, “Node 2 expanded two items, 2 of 3.”



The user knows which node currently has focus, Node 2; that the state of the node is expanded, it has two children and that Node 2 is the second node in a group of three nodes at the same level in the tree hierarchy. This type of information is critical for screen reader users to interact and navigate with complex widgets. The title of the Node is spoken when it receives focus. Because the tree control developer has properly set the state of Node 2 to expanded=true, that information is spoken by the screen reader. All of the nodes in the tree control have been properly organized into hierarchies and identified using the role=”group” so the browser is able to provide the screen reader with information about the number of children and the position of the current node within its siblings.

When testing widgets with a screen reader interact with each element and verify that the proper role and state and grouping information is spoken by the screen reader and all functionality is accessible via only the keyboard. In addition, make certain that elements which do not provide information to the screen reader user are marked with a role=”presentation” so they are ignored. In Figure 2, each tree item node which has children is preceded by an image of a plus or minus to indicate the expanded state of the item. This plus or minus image can not receive focus and it does not need focus since the keyboard user can expand and collapse the tree item using the arrow keys and the screen reader will speak the expanded or collapsed information based on the state value.

Test for high contrast mode support

The code in dijit._base.wai.js to check for high contrast mode currently only works with Firefox and Internet Explorer on the Windows Operating system. Windows comes configured with default high contrast mode settings. Turn on high contrast mode in Windows XP via the Accessibility Options dialog available from the Control Panel. From the Diplay panel check the high contrast checkbox. Press the settings button to modify the display colors and font sizes. Checking the Use shortcut checkbox from the settings dialog allows toggling high contrast mode on and off using the shift-alt-printscreen key combination. Press OK to confirm the settings and then OK again to close the dialog and turn on high contrast mode. Note, putting your system in high contrast mode will likely rearrange the desktop icons on the system due to the changes in font size.

After turning on high contrast mode, test the widget in Firefox or Internet Explorer. If the widget test page was already loaded in the browser, you may need to refresh the page for the high contrast mode to take affect. When the widget is reloaded high contrast mode will be detected and the accessible version of the widget will be loaded. This version should provide visible text alternatives for CSS background images to create the look and feel of the widget user interface. Verify that all components within the widget are visible. Any components or visual effects which are created via background images or color will no longer be visible in high contrast mode. Verify that the user is able to determine where current focus is within the widget. If focus is not visually evident the widget developer may have used a background color change to indicate focus (or the browser provided focus rectangle may be hard to distinguish). Fix this by using a different styled border or other mechanism to indicate focus if the browser differentiation is not sufficient. When testing in high contrast mode it is fairly evident where the problems occur since any visual effects which rely on color or images will no longer be visible.

Test for usable operation with images off

In addition to testing for support of high contrast mode, the widgets should also work with images are turned off in the browser. Currently this mode is only detected by the dijit.wai.onload function in Firefox on Windows. Test in Firefox by turning images off via the Tools Options dialog. Test in Internet Explorer by turning on high contrast mode (so the accessible version of the Dojo widgets will be loaded) and then also turning off images in the Advanced tab of the Tools Internet Options dialog.

With images turned off the widgets should still be usable. Since no images are loaded, the alt attribute of any real image elements and the text alternatives for any CSS background images should be displayed. Verify that the text alternatives provide sufficient information for the user to interact with the widget.

The Event System

Dojo's event system offers a refreshing alternative to the normal JavaScript events. With Dojo, you connect functions to one another, creating a link that calls one function when another fires. This means that you can connect a function of your own to

- a DOM event, such as when a link is clicked;
- an event of an object, such as an animation starting or stopping;
- a function call of your own, such as bar();
- a topic, which acts as a queue that other objects can publish objects to.

Your connected function is called when the event occurs. With simple events, when it calls your function, dojo passes your function a normalized event object, so that it can respond correctly, responding to keystrokes or stopping default behavior. With topics, Dojo passes any subscribed functions the object that was published. Dojo happily abstracts away all of the difficulty of cross-browser event systems, offering programmers a coherent event system that acts consistently across browsers.

Dojo's event system is flexible and gives you a few options for connecting your functions. In the core package, you have both simple events (which use a signal and slot system, similar to Qt's) and topics. In this section, you'll learn the following:

- how to connect functions to one another with *dojo.connect*,
- what comes with an event object
- how to connect functions with topics and even publish your own objects to the topic
- how to enjoy event-based programming

Simple Connections with dojo.connect

Dojo provides a pair of functions to handle many of your event-handling needs: *dojo.connect* and *dojo.disconnect*. With *dojo.connect*, you can link one function to fire when another does, handling DOM events and custom functions with a single mechanism. Additionally, *dojo.connect* allows you to connect as many functions to an event as necessary. With *dojo.disconnect*, you can cancel a connection, assuming you've kept some reference to it.

How does it work?

Imagine that you're hungry and have decided to cook a pizza in your oven. The pizza will take 17 minutes, so you set a timer. You have better things to do than sit around your kitchen hanging out by the timer though, so you get your brother and tell him, "When you hear the oven timer, take the pizza out of the oven and bring me a slice." Your brother can only keep track of one thing at a time, and you don't want your house to burn down, so you tell your sister, "When you hear the over timer, turn off the oven." Because you're a little worried that your dirty oven might start to smoke, you tell them both, "If you hear the smoke alarm, come get me and then go outside." After you get your pizza, you tell your brother and sister that they don't have to worry about the oven alarm now and that they can go play until you call for them again. You then set the oven alarm to wake you up from a nap.

In this example, your siblings are functions. Your telling them to respond to certain events, such as "onPizzaDone" and "onHouseOnFire" performs the same function as *dojo.connect* — it sets up your siblings (functions) to listen for an event and perform their tasks when they receive notice. The various alarms are similar to event objects; they inform your siblings of important details about the situation (such as what is beeping). Telling your siblings that they don't need to worry about the oven alarm anymore is similar to *dojo.disconnect*; the next time the oven alarm goes off, it means that you need to wake up, and you don't want your brother hunting for a pizza needlessly, so you've told him to stop listening to that event.

Syntax

Dojo.connect takes a variety of forms of arguments, depending on how you are planning to use it. This section will cover those various forms, based on use cases for them. You can think of it as a more in-depth version of the overview from Functions Used Everywhere.

Dojo.connect has the following signature (acceptable types in square brackets):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
handle = dojo.connect(Scope of Event [object or null], Event [string], Context of Linked Method [string or null], Linked Method [string or function], Don't Fix Flag [boolean])
```

All of the options for calling *dojo.event* are explored further below.

Example Code for Reference

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<head>
<title>Dojo Events are Great</title>
<script src="dojo/dojo.js" type="text/javascript"></script>
<script type="text/javascript">
function foo() { console.debug("A click upon your houses!"); }
function globalGuy() { console.debug("Global Guy fired!"); }
var someObject = {
   bar: function() { console.debug("Bar fired!"); return 7; },
   baz: function() { console.debug("Baz fired!"); return 14; }
}
var anotherObject = {
    afterBaz: function () { console.debug("afterBaz fired!"); }
}
</script>
<body>
<a id="firstLink" href="http://dojotoolkit.org/">Dojo</a> is an excellent tool kit.
</body>
```

Connecting to a DOM Event

To connect a function to a DOM event with Dojo, you first need to get the node that you want to connect to. Here, I'll use the venerable *dojo.byId*.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
firstLinkNode = dojo.byId("firstLink");
```
Now, to fire foo when a user clicks #firstLink, and I have the node, so I just need to use *dojo.connect* for the heavy lifting. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
firstLinkConnections = [];
firstLinkConnections[0] = dojo.connect(firstLinkNode, 'onclick', foo);
```

In this example, I passed *dojo.connect* the object I want my function to listen to (in this case, a DOM node), the name of the function that should trigger my function's call (in this case, the "onclick" event), and the name of my function. Note that I keep a reference to the connection by setting firstLinkConnections[0] to the return value of *dojo.connect*. This will allow me to disconnect the listener later, if I

desire. Now, when a user clicks "Dojo," a message appears in the log Because my function is global in scope, I can pass it directly to connect. The following, however, are equivalent:

```
firstLinkConnections[0] = dojo.connect(firstLinkNode, 'onclick', null, foo);
```

and

```
firstLinkConnections[0] = dojo.connect(firstLinkNode, 'onclick', null, "foo");
```

Now, if I also want to connect someObject.bar() to #firstLink, I can do that.

```
firstLinkConnections[1] = dojo.connect(firstLinkNode, 'onclick', someObject, "bar");
```

Because I've used Dojo's event handling, I can connect an arbitrary number of functions to fire on an event.

A note about the event names: In most cases, event names now are lower case, except in special cases (e.g., some Mozilla DOM events). Dojo will add "on" to your event name if you leave it off (e.g., 'click' and 'onclick' are the same thing to dojo). Dojo responds to all of the usual events (e.g., onclick, onmouseover, etc.) and the onkeypress event for capturing typing.

A note about return values: Any value returned by a function called by *dojo.connect* will be lost.

Connecting Functions to One Another

Connecting functions to one another is even simpler than connecting them to DOM events; because you already have a reference to the function, you don't need to do any byId or query work. To have anotherObject.afterBaz fire after someObject.baz fires, use the following:

```
objectConnections = [];
objectConnections[0] = dojo.connect(someObject, "baz", anotherObject, "afterBaz");
```

In the above code, the first argument is the context of "baz," the second argument is the event (in this case, when baz fires), the third argument is the context of your listener function, and the fourth argument is the listener function itself. Connecting two global functions is even easier:

```
objectConnections[1] = dojo.connect(foo, globalGuy);
```

Now, whenever foo is called, globalGuy will also fire. As you might expect, connecting a method to a global function, or vice versa, is logical and simple:

```
objectConnections[2] = dojo.connect(foo, anotherObject, "afterBaz");
objectConnections[3] = dojo.connect(someObject, "baz", globalGuy);
```

Disconnecting

To disconnect listeners from events, you simply pass the connection handle (the return value of *dojo.connect* to *dojo.disconnect*. To disconnect globalGuy from someObject.baz, I use the following code: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.disconnect(objectConnections[3]);
```

Event Object

When you connect a function to a DOM event with *dojo.connect*, dojo passes your function a **normalized** event object. This means that,

regardless of the client's browser, you can count on a set of standard attributes about the event and a set of methods to manipulate the event.

Syntax

Assume that your function has been called by dojo.connect and takes an argument named *event*

Dojo provides the following attributes with an event object:

- event.target — the element that generated the event
- event.currentTarget — the current target
- event.layerX — the *x* coordinate, relative to the *event.currentTarget*
- event.layerY — the *y* coordinate, relative to the *event.currentTarget*
- event.pageX — the *x* coordinate, relative to the view port
- event.pageY — the *y* coordinate, relative to the view port
- event.relatedTarget — For *onmouseover* and *onmouseout*, the object that the mouse pointer is moving to or out of
- event.charCode — For keypress events, the character code of the key pressed

Dojo provides the following methods with an event object:

- event.preventDefault — prevent an event's default behavior (e.g., a link from loading a new page)
- event.stopPropagation — prevent an event from triggering a parent node's event

Additionally, *dojo.stopEvent(event)* will prevent both default behavior any any propagation (bubbling) of an event.

Example Code for Reference

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<head>
<title>Dojo Events are Great</title>
<script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"></script>
<script type="text/javascript">
function echo(event) {
        key = event.charCode;
        console.debug(event.charCode);
}
function foo(event) {
        dojo.stopEvent(event);
        console.debug("The link was clicked");
}
dojo.addOnLoad(function() {
        interactiveNode = dojo.byId("interactive");
        linkNode = dojo.byId("link");
        dojo.connect(interactiveNode, 'onkeypress', echo);
        dojo.connect(linkNode, 'onclick', foo);
});
</script>
<body>
        <a href="http://dojotoolkit.org" id="link">Dojo</a> is great.
        <form>
                <label for="infield"> Type some text: </label>
                <input id="interactive" type="text" name="infield">
        </form>
</body>
```

Using a Dojo Event Object

In the example code, we have two functions that are connected to two different events. Echo sends the key code of any key typed in the text input field to the console. It does so by using the *charCode* property of the normalized event object. Foo is connected to the #link and cause it to send "The link was clicked" to the console instead of changing the browser's location; by using the preventDefault method of the normalized event object, connections to change the default behavior of DOM objects.

Now, imagine that you want to detect for the down arrow key in the text box. To do this, we just need to attach a new event listener to the text box and check to see if each keycode is the keycode for the down arrow. And how do you know what the keycode for the down arrow is, you may ask? Well, Dojo provides constants for every non-alpha-numeric key [[link to sub page to come]]. In our case, we are interested in dojo.keys.DOWN_ARROW. So, assuming that you want to just log when the down arrow is pressed, the following code should do the job: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.connect(interactiveNode, 'onkeypress', function (evt) {
        key = evt.keyCode;
        if(key == dojo.keys.DOWN_ARROW) {
                console.debug("The user pressed the down arrow!");
        }
});
```

Key Code Constants

Dojo provides the following keycode constants in the namespace dojo.keys:

- BACKSPACE
- TAB
- CLEAR
- ENTER
- SHIFT
- CTRL
- ALT
- PAUSE
- CAPS_LOCK
- ESCAPE
- SPACE
- PAGE_UP
- PAGE_DOWN
- END
- HOME
- LEFT_ARROW
- UP_ARROW
- RIGHT_ARROW
- DOWN_ARROW
- INSERT
- DELETE
- HELP
- LEFT_WINDOW
- RIGHT_WINDOW
- SELECT
- NUMPAD_0
- NUMPAD_1
- NUMPAD_2
- NUMPAD_3
- NUMPAD_4
- NUMPAD_5
- NUMPAD_6
- NUMPAD_7
- NUMPAD_8
- NUMPAD_9
- NUMPAD_MULTIPLY
- NUMPAD_PLUS
- NUMPAD_ENTER
- NUMPAD_MINUS
- NUMPAD_PERIOD
- NUMPAD_DIVIDE
- F1
- F2
- F3
- F4
- F5
- F6
- F7
- F8
- F9
- F10
- F11
- F12
- F13
- F14
- F15
- NUM_LOCK
- SCROLL_LOCK

Page Load / Unload

To perform any DOM scripting, one needs a DOM to work with. Otherwise, you don't have nodes to attach your events to or manipulate, and the results of your program can be undesirable (e.g., no output for no obvious reason, or events attached to the document root).

Dojo elegantly solves this problem with *dojo.addOnLoad*, which we introduced back in Functions Used Everywhere. Better yet, *dojo.addOnLoad* starts your scripts after the DOM has loaded but before all of the page elements have loaded. That means your script doesn't have to wait for images and other large resources before it manipulates page structure. This can significantly improve the perceived performance of your scripts.

Like other event handlers, you register a function with addOnLoad - either a function literal, or a function name as a string. You can also pass an object and a method name (as a string) to call an object-oriented method.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
// Runs tp.toggle() - see dijit.TitlePane - when the page is done loading.
   dojo.addOnLoad(tp, "toggle");
   // Function form
   dojo.addOnLoad(function() {
      dojo.byId("Status").innerHTML = "You may begin.";
```

```
   });
<script>
<body class="tundra">
   <div dojoType="dijit.TitlePane" jsId="tp"></div>
   <div id="Status"></div>
```

Acting in parallel to *dojo.addOnLoad* is *dojo.addOnUnload* , which runs functions when the page is being "unloaded" (e.g., when the user clicks a link off of the current page). This gives you the opportunity to send notifications to your web application or clean up unavoidable memory leaks.

dojo.addOnLoad and Cross-Domain Loading

Dojo.addOnLoad() is doubly-important when using cross-domain resources. Non-DOM-related Dojo functions can be called anytime from a locally-installed Dojo build, as in:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script>
   // Works in local Dojo.  Doesn't work in XDomain
   dojo.require("dojox.uuid.Uuid");
   dojo.require("dojox.uuid.generateTimeBasedUuid");
   var uniqueId = new dojox.uuid.Uuid(dojox.uuid.generateTimeBasedUuid());
   ...
```

In a local Dojo build, dojo.require loads the dojox.uuid code right away. In Cross-Domain Dojo, this is not guaranteed due to browser security rules. But this code works fine on either:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script>
   // Works in local and XDomain
   dojo.require("dojox.uuid.Uuid");
   dojo.require("dojox.uuid.generateTimeBasedUuid");
   dojo.addOnLoad(function() {
       var uniqueId = new dojox.uuid.Uuid(dojox.uuid.generateTimeBasedUuid());
       ...
   });
</script>
```

We highly recommend initializing your code with dojo.addOnLoad. Semantically, it's no different than bare code, and it allows you to switch from XDomain to local Dojo and back.

Publish and Subscribe Events

In addition to the simple event system created by *dojo.connect*, dojo offers support for anonymous publication and subscription of objects, via *dojo.publish* and *dojo.subcribe*. These methods allow a function to broadcast objects to any other function that has subscribed. This is dojo's topic system, and it makes it very easy to allow separate components to communicate without explicit knowledge of one another's internals.

There are three functions that you need to understand to use dojo's topic system: *dojo.publish*, *dojo.subscribe*, and *dojo.unsubscribe* *Dojo.publish* calls any functions that are connected to the topic via *dojo.subscribe*, passing to those subscribed functions arguments that are published (see syntax for details). As one might expect, *dojo.unsubscribe* will cause a previously subscribed function to no longer be called when *dojo.publish* is called in the future

How does it work?

Imagine that you run a running a conference, and there will be updates throughout the day. You could collect contact information for everyone at the beginning of the day, along with each person's interests. However, this would be a lot of logistical work. Instead, you decide to use your facility's Public Address System. When there is an update to the schedule, you announce "This is an update to the schedule: the Dojo training is full and we have added yet a third time slot for it tomorrow." When there is meal information, you announce "This is an update about food: we will be serving free ice cream in the main hall in five minutes." This way, anyone interested in your information can pay attention to any updates that could change their behavior. You don't need to know who is subscribing, and they don't need to fill out a bunch of paper work — it's a win-win.

Syntax

dojo.publish

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.publish(Topic Name [string], Arguments to Pass to Subscribed Function [array])`

dojo.subscribe

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color:

#009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `handle = dojo.subscribe(Topic Name [string], Context of Linked Method [string or null], Linked Method [string or function])`

dojo.unsubscribe

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.unsubscribe(Handle [handle object])`

Example Code for Reference

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function globalGuy(arg) { console.debug("Global Guy fired with arg " + arg); }
var someObject = {
   bar: function(first, second) { console.debug("Bar fired with first of "+first+" and second of "+second); return 7; },
}
}
```

Subscribing and Publishing Topics

To connect globalGuy to the topic "globalEvents" and someObject.bar to "fullNames", you simply use *dojo.subscribe*, as follows:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
topics = [];
topics[0] = dojo.subscribe("globalEvents", null, globalGuy);
topics[1] = dojo.subscribe("fullNames", "someObject", bar);
```

Note that the following alternative form would also work:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
topics = [];
topics[0] = dojo.subscribe("globalEvents", globalGuy);
topics[1] = dojo.subscribe("fullNames", "someObject", "bar");
```

To publish information to both of these topics, you pass *dojo.publish* the topic names and arrays of the arguments that you want to pass to subscribed functions, as follows

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.publish("globalEvents", ["data from an interesting source"]);
dojo.publish("fullNames", ["Alex", "Russell"]);
```

To disconnect someObject.bar from its topic, you use *dojo.disconnect*, as follows:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.unsubscribe(topics[1]);
```

XMLHttpRequest (XHR)

The XMLHTTP request object (XHR for short) is one of the basic building blocks for constructing responsive Ajax-drive interactions. By allowing you to retrieve data on the user's behalf without refreshing the whole page the XHR object provides tremendous, but cross-browser XHR usage is beset by memory leaks, divergent APIs, a lack of built-in form encoding from JavaScript, and painful corner cases when de-serializing response data.

Dojo provides a solid set of battle-tested XHR wrapper functions to allow you to build Ajax interactions with confidence, use a unified API, and handle forms with ease. These APIs are built into Dojo Base, so you can use them in any page that includes dojo.js. Read on to learn how easy it is to build powerful Ajax interactions with Dojo.

Hello, Ajax World!

Many programming tutorials contain a "Hello, World!" example, so it seems appropriate to have one for Dojo XHR. In this example, your web page will fetch a snippet of content via XHR and attach it directly to your page.

To setup the example:

1. Create a file named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ajax.txt`. It's your decision what to put in the file. "Hello, Ajax world!" is a good start.

2. Put /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ajax.txt` in the default /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `documents` directory on your web server. In many cases, that is the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `httpdocs` directory.

3. Open your web browser and navigate to the file. You should see the contents of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ajax.txt`.

4. Create a file named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `hello.html`. Copy and paste the contents of Example 1, “Hello, Ajax world!” into that file.

5. Set the */* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;}* `URL` argument to the value you used to test the server setup.

6. Put /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `hello.html` in the default /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `documents` directory on your web server. In many cases, this is the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `httpdocs` directory.

   Put the file in the same directory as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ajax.txt`.

7. Make the file read-only.

8. Open your web browser and navigate to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `hello.html`. You should see the contents of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ajax.txt` in your browser window.

Example 1. Hello, Ajax world!

```
<html>
 <head>
  <title>Hello, Ajax world!</title>
  <script type="text/javascript"
    src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"></script> <!--(1)-->
  <script type="text/javascript">
    function hello() { // (2)
      dojo.xhrGet( { // (3)
        // The following URL must match that used to test the server.
        url: "http:///* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
server//* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
ajax.txt",
        handleAs: "text",

        timeout: 5000, // Time in milliseconds

        // The LOAD function will be called on a successful response.
        load: function(response, ioArgs) { // (4)
          dojo.byId("cargo").innerHTML = response; // (5)
          return response; // (6)
        },

        // The ERROR function will be called in an error case.
        error: function(response, ioArgs) { // (4)
          console.error("HTTP status code: ", ioArgs.xhr.status); // (7)
          return response; // (6)
          }
        });
      }
  </script>
  <script type="text/javascript">
    dojo.addOnLoad(hello); // (8)
  </script>
 </head>
 <body>
   <div id="cargo" style="font-size: big"></div> <!--(9)-->
 </body>
</html>
```

(1) This JavaScript program bootstraps Dojo 1.0 via the AOL Content Distribution Network. If you choose to install Dojo locally, use the following script tag to bootstrap Dojo:

<script type="text/javascript" src="*/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo/dojo.js`"></script>

Modify the directory reference in the SRC attribute to match the location of your Dojo installation

(2) This function will be called *after* Dojo completes its initialization phase.

(3) The desired HTTP method call (GET, POST, PUT, DELETE) is part of the function name.

(4) Inside this function, the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `this` variable will be the object used as the argument to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.xhrGet()` call.

(5) This statement demonstrates one way of stuffing a server response into a document. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.xhrGet` call asks Dojo to treat data from the server as text (handleAs: "text"). Therefore /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `response` will be a text string.

(6) Dojo recommends that you always /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `return(response);` to propagate the response to other callback handlers. Otherwise, the error callbacks may be called in the success case.

(7) /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ioArgs` is an object with some useful properties on it. For instance, for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `XMLHttpRequest` calls, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ioArgs.xhr` is the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `XMLHttpRequest` that was used for the call.

(8) The Dojo team recommends that you always use /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.addOnLoad()` to call any startup code you write. Even for this simple example, unless we use /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.addOnLoad()`, we cannot reliably assume that the main function /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `hello()` will run after Dojo initialization completes.

Use /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.addOnLoad()` only to run start-up code. Do not use /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.addOnLoad()` to call JavaScript event handlers.

Two ways to abuse Dojo. The following techniques will produce tedious, head-scratching debugging sessions by not using /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.addOnLoad()`.

<body onload='hello'>

This technique is *never* a good idea when using Dojo. You have no assurance that /* GeSHi (C) 2004 - 2007 Nigel McNie

(http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `hello()` will run after Dojo initialization.

Put the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `hello()` body in a <script> without /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.addOnLoad()`

This technique usually fails with Dojo. Only the simplest JavaScript programs, like "Hello, Ajax world!" can employ this technique. It's a safe bet that yours isn't one of them.

(9) Holding area for the server response. A common use case is that the server returns an HTML fragment that the client will want to display. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `innerHTML` attribute of such placeholder tags is a convenient way to display HTML fragments.

Passing Data with JSON

JSON, or Javascript Object Notation, is a lightweight data interchange standard. It can, in theory, be used to pass data between any two programming languages, but it has special advantages when used with Javascript. JSON is fundementally just the Javascript array and object initializer syntax on its own. So this array of objects in a Javascript program:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var cobblers = [
   {"filling": "peach", "timeToBake": 30 },
   {"filling": "cherry", "timeToBake": 35 },
   {"filling": "blueberry", "timeToBake": 30}
];
```

Roughly everything after the "=" is JSON. Here's what the JSON packet would look like coming back from a web service:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ "cobblers": [
       {"filling": "peach", "timeToBake": 30 },
       {"filling": "cherry", "timeToBake": 35 },
       {"filling": "blueberry", "timeToBake": 30}
    ]
}
```

"Excuse me," you might say, "but we already have a data interchange format. You might have heard of it ... it's called XML." So why not just pass it as:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<cobblers>
   <cobbler filling="peach" timeToBake="30" />
</cobblers>
```

The answer is performance, performance, performance! JSON data is parsed up to 100 times faster than XML. That makes sense because the parser is just eval(). For small portions of data, this doesn't mean much, but for large ones it's indispensible. XML expressed in Javascript must carry the full weight of XML - including namespaces, DTD's and schemas. Furthermore, the DOM representation of XML is much more memory-intensive than native Javascript.

Add to that the fact that XML is interpreted a little differently in each browser, and JSON is the preferred method for data interchange in dojo. XML is supported, but you might not want to use it in simple scenarios. Where it really begins to make sense, however, is when large scale transformations are needed on the client side. Almost every modern browser today provides a client-side XSLT transform facility which can often outstrip JSON for speed in transforming large data sets from one structure to another since that transformation is handled in C or C++ and not in JavaScript. Whether your app chooses to use JSON or XML is often a foregone conclusion, but be aware that there

are tradeoffs for each.

Accepting JSON Data in dojo.xhrGet

Provided your web service sends JSON (as the above example shows) Dojo pretty much handles the rest for you, including parsing the JSON into a JavaScript object. All you have to do is specify a JSON content handler:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.xhrGet( {
        // The following URL must match that used to test the server.
        url: "http://server/ajax.txt",
        handleAs: "json",
        load: function(responseObject, ioArgs) {
          // Now you can just use the object
          console.dir(responseObject);  // Dump it to the console
          console.dir(responseObject.cobblers[0].filling);  // Prints "peach"
          return responseObject;
        }
        // More properties for xhrGet...
});
```

Errors and Timeouts

Regular web requests and Ajax requests with dojo.xhrGet/Post are much alike. Both use URL's and both use the HTTP protocol. But with browser requests, it is always clear to user when something goes wrong. You may get a 404 - Page Not Found, or a Server Unavailable, or at least something that says "Error". Ajax requests happen in the background, so when they error out the user won't know. Even worse, if the response *never* comes the browser may appear to lock up.

That's why it's extremely important to provide an error handler and a timeout handler with any dojo.xhrGet/Post calls. *You should consider these as critical as URL or the load function*

At the very least, you should alert the user that something went wrong. Here's an example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.xhrGet({
        url: "/cgi-bin/timeout.cgi",
        load: function(data){
                document.myForm.myBox.value = data;
                dojo.byId("boxLoadTime").innerHTML = new Date();
        },
        error: function(err){
                console.debug("Holy Bomb Box, Batman!  An error occurred: ", err);
        },
        timeout: 2000
});
```

The error() function takes the same arguments that load() does. But unlike load(), the only useful parameter is data, which contains the error message. You can also find out what kind of error was generated by looking at the error object's "dojoType" property. It will usually be "timeout" or "cancel", but other error types are possible.

The timeout, given in milliseconds, defaults to 0, which means "wait forever". Even if you expect the request will take a long time, you should set a high value here (e.g. 15000 = 15 seconds), not 0.

Sending Form Data

The Url of dojo.xhrGet may contain parameters, like so:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
url:  'myprogram.php?firstname=Chicken&lastname=Little&key=111111'
```

There are two problems: (1) it's difficult to URL encode everything, (2) it doesn't allow for dynamic parameters. The above works fine for everyone named Chicken Little, but ...

It's easier and more flexible to send an entire form of data. And you can do that with the form parameter of dojo.xhrGet

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var kw = {
        url: "myprogram.php",
        load: function(data){
                dojo.byId('myBox').value = data;
```

```
        },
        error: function(data){
                console.debug("An error occurred: ", data);
        },
        timeout: 2000,
        form: "myForm"
};
dojo.xhrGet(kw);
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<form id="myForm">
 <input type="hidden" name="key" value="111111" />
 <input type="text"  name="firstname" length="50" />
 <input type="text"  name="lastname" length="50" />

 <input type="text"  id="myBox" name="myBox" length="50" />
</form>
```

Alternate Transports

Script request/JSONP (dojo.io.script)

To make an IO call using a script tag (for instance, for cross-domain JSONP calls), dojo.require("dojo.io.script"), and use:

- dojo.io.script.get()

Specific arguments for dojo.io.script calls:

- callbackParamName: "". String. The URL parameter name that indicates the JSONP callback string. For instance, when using Yahoo JSONP calls it is normally, callbackParamName: "callback". For AOL JSONP calls it is normally callbackParamName: "c".
- checkString: "". String. A string of JavaScript that when evaluated like so: "typeof(" + checkString + ") != 'undefined'" being true means that the script fetched has been loaded. Do not use this if doing a JSONP type of call (use callbackParamName instead).

"handleAs" is NOT applicable to dojo.io.script.get() calls, since it is implied by the usage of "callbackParamName" (response will be a JSONP call returning JSON) or "checkString" (response is pure JavaScript defined in the body of the script that was attached).

IFrame requests (dojo.io.iframe)

To send an IO call using an IFrame (for instance, to upload files), dojo.require("dojo.io.iframe"), and use: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// gather all parameters from a form
dojo.io.iframe.send({
    form: "idOfForm",
    load: function(data){
         console.debug(data); // successful response
    }
});
// pass in all of the parameters manually
dojo.io.iframe.send({
    method: "GET",
    url: "iframeHandler.php",
    content: {
        param1: "la dee dah",
        param2: "my poor electrons!",
    },
    load: function(data){
         console.debug(data); // successful response
    }
});
```

Specific arguments for dojo.io.iframe calls:

- method: "POST". What type of HTTP method to use for the request. Valid values are "POST" (default) or "GET".
- handleAs: Valid values are text, html, javascript, and json. IMPORTANT: For all values EXCEPT html, The server response should be an HTML file with a textarea element. The response data should be inside the textarea element. Using an HTML document the only reliable, cross-browser way this transport can know when the response has loaded. For the text/html mimetype, just return a normal HTML document. NOTE: text/xml or any other XML type is NOT supported by this transport.

Remote Procedure Call (RPC)

Dojo provides low level I/O out of the box, but as applications grow in complexity it's natural to want something less one-off-ish. Dojo's Remote Procedure Calls (RPC) module aims to make this process less error prone, easier, and require less code.

Remote Procedure Calls allows you to invoke a method on a remote host. Dojo provides a basic RPC client class that has been extended to provide access to JSON-RPC services and Yahoo services. It was designed so that it is easy to implement custom RPC services.

For example, you have a little application that you want to use to make some server calls. For simplicity, the methods you want the server to do are add(x,y) and subtract(x,y). Without using anything special, like an RPC client, you could do something like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3

{color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
add = function(x,y) {
        request = {x: x, y: y};
        dojo.xhrGet({
            url: "add.php",
            load: onAddResults,
            handleAs: "text",
                content: request
        });
}
subtract = function(x,y) {
        request = {x: x, y: y};

        dojo.xhrGet({
            url: "subtract",
            load: onSubtractResults,
            handleAs: "text"
                content: request
        });
}
```

This isn't particularly difficult but it is repetitive and error prone. Dojo's RPC clients simplify this process by taking a simple definition of the remote methods and application needs and generating client side functions to call these methods. You need only write this definition, and initialize an RPC client object and then all of these remote methods are available for you to use as normal.

The definition file, called a Simple Method Description (SMD) file, is a simple JSON string that defines a URL to process the RPC requests, any methods available at that URL, and the parameters those methods take. The definition for the example above might look like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{
        "serviceType": "JSON-RPC",
        "serviceURL": "rpcProcessor.php",
        "methods":[
                {
                        "name": "add",
                        "parameters":[
                                {"name": "x"},
                                {"name": "y"}
                        ]
                },
                {
                        "name": "subtract",
                        "parameters":[
                                {"name": "x"},
                                {"name": "y"}
                        ]
                }
        ]
}
```

Once the definition has been created, the code is pretty simple. The definition can be supplied either as a URL to retrieve it, a JSON string, or a JavaScript object, as shown in the following example.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var myObject = new dojo.rpc.JsonService("http://localhost/definition.smd");
var myObject = new dojo.rpc.JsonService({smdStr: definitionJSON});
var myObject = new dojo.rpc.JsonService(definitionObj);
```

Thats it! Now all that's left is to call the method as shown in the following example.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `myObject.add(3,5);`

Of course this just calls the method and doesn't address what is returned from the call or how to get at the results. Just like the rest of the I/O system in dojo 0.9+, the RPC system returns a deferred to which you can attach callbacks.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var myDeferred = myObject.add(3,5);
myDeferred.addCallback(myCallbackMethod);
```

Or, it could be more succinctly done this way:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;}

.geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var myDeferred = myObject.add(3,5).addCallback(myCallbackMethod);
```

You just add /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `myCallbackMethod` as a callback for the Deferred returned from /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `myObject.add()`. In this case /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `myCallbackMethod` is called with a parameter with a value of 8. Likewise, you can attach an errback method to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `deferred` object to process any errors returned from the server. You can add as many callbacks and errbacks to your /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `deferred` object as you want and they will be called in the order that they were connected to the deferred object.

In addition to JsonService, Dojo offers a JsonpService client, which is used for services such as Yahoo which provide JSON-P style service. Most services of this style can be represented in an SMD and passed to the JsonpService and used in the same fashion as above in a crossplatform manner. This makes mashups a cinch. Take a look in the dojox project at the Yahoo.smd example provide. Pass that url to JsonPService and all of Yahoo's services are at your finger tips and others are easy to add.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var svc = new dojo.rpc.JsonpService(dojo.moduleUrl("dojox.rpc", "yahoo.smd"), {appid: "dojoApp"});
```

The resulting Yahoo Service object allows you to perform web searches, do term extraction, and search all sorts of local an contextual data. See the SMD file for details on the available methods, parameters, and documentation pointers.

While Dojo is currently limited to these two RPC clients, the design of the base classes allows you to easily customize and extend /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.rpc.RpcService` to create services that meet your specific needs. Have a look at the source for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `RpcService.js` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `JsonService.js` to see just how simple it is. If anyone is interested in contributing their own SMD files for their pet service or their company's services, please let us know!

Drag and Drop

Drag and Drop (DnD) in web applications is one thing which is gaining popularity these days. Already there is a handful of sites using drag and drop functionality coupled with AJAX to deliver some killer content. Suppose, one day your boss sees one such cool website . You are ordered to implement similar thing in the web application you are working on. You ask yourself if there is a solution that is easy, fast, tested, reusable, customizable and works with all major browsers !

The Dojo DnD API enables you to do all the cool(and complex) drag and drop stuff in a clean way with less pain.

Terminology

Source: A source is an HTML node that can be dragged around. It can be a simple node, or a node which contains other nodes - a DIV, a TABLE, etc. In Dojo, a source has type dojo.dnd.Source.

Target: Target is like a placeholder, onto which nodes can be dropped. A good example is a shopping cart application, where an item icon is a source while the shopping cart is a target.

Interestingly, all dojo.dnd.Source objects in Dojo DnD are also targets, so many applications just use dojo.dnd.Source for both. For a *pure target*, i.e. one that's not draggable, you define the target with type dojo.dnd.Target.

Avatar: Avatar is kind of ghost image of the object being dragged. This ghost image follows the mouse cursor, till the time it is dropped to a proper placeholder. In Dojo this functionality is available to every drag and drop operation by default. The 'Dojo.dnd.Avatar' class does all this for you and displays a nice tooltop with text 'moving' or 'copying', indicating the type of operation being performed.

DnD Manager: The DnD Manager is a singleton, which is responsible for handling all DnD operations on a page. It accomplishes many complex tasks required for successful DnD operations like detecting initialization of valid drag operation by user, creation of Avatar, allowing drop to only eligible targets and such things. Most of the time the default manager works fine, but you can extend or create your own DnD Manager class for more complex scenarios.

Mover: Mover is a mixin class which makes an object (HTML node) movable. Once a node is made movable using this class, user can freely drag and drop the node anywhere on the page, for example a popup window. There is no concept of source and target in this case.

Selector : Selector is the base class for Source in Dojo. This means that every Source is a Selector. A Selector controls how its child nodes can be selected. This also means that every node that participates in DnD operation is part of some selectable list.

Container: Container is the baseclass for Selector. It provides functionality of sensing mouse movement over its children.

To sum up: a container contains nodes, a selector makes it possible to select single or multiple nodes, a mover makes the selected nodes movable, an Avatar provides visual cue during DnD and the DnD Manager makes the overall operation possible.

A Simple Example

Let's make things interesting: a simple example of Dojo DnD in action ! In this example we will implement very basic DnD functionality in a web page. We will have a red and a blue rectangle that can be dragged and dropped onto a target.

First, we'll start with the standard header and some CSS:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<html>
<head>
<title>Simple DnD Example</title>
<style type="text/css">
        .target {border: 1px dotted gray; width: 300px; height: 300px;padding: 5px;}
        .source {border: 1px dotted skyblue;height: 200px; width: 300px;}
        .bluesquare {height:50px;width:100%;background-color:skyblue}
        .redsquare {height:50px;width:100%;background-color:red}
</style>
<script type="text/javascript" src="../../dojo.js" djConfig="parseOnLoad: true"></script>
<script type="text/javascript">
        dojo.require("dojo.dnd.source"); // dojo.dnd.Source in 1.0
        dojo.require("dojo.parser");
</script>
</head>
```

This is an important part. In this example we have only definied four classes. The 'target' and the 'source' class will be used by Dojo. The other two classes are used to create the red and blue rectangle.

Note: in 0.9, to use the class dojo.dnd.**S**ource, you must dojo.require the resource dojo.dnd.**s**ource. Note the capitalization here. In Dojo 1.0, both are capitalized.

Next, to hold the source and targets, we have a table with two cells. The first(left) cell hosts a source and the second(right) cell hosts a target:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<body style="font-size: 12px;">
<h1>A Simple Example</h1>
<table><tbody><tr>
<td>
<!-- Create a source with two nodes -->
<div dojoType="dojo.dnd.Source" jsId="c1" class="source">
        SOURCE
        <div class="dojoDndItem" dndType="blue">
                <div class="bluesquare">BLUE</div>
        </div>
        <div class="dojoDndItem" dndType="red,darkred">
                <div class="redsquare">RED</div>
        </div>
</div>
</td>
```

The outermost tag creates a source object. The inner tags with class 'dojoDnDItem' create nodes, that can participate in DnD. The 'dndType' attribute can be used to specify 'type' of a node. You can specify more than one type for a node. Our nodes contains a red and blue rectangle with text on them. You can replace this content with whatever you like.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<td>
<!-- Create a target that accepts nodes of type red and blue. -->
<div dojoType="dojo.dnd.Target" jsId="c2" class="target" accept="blue,darkred">
```

```
        TARGET
</div>
</td>
</tr><tbody/></table>
```

The above markup creates a pure target that can accept nodes with type 'blue' and 'darkred'. The accept attribute is comma separated list of 'types' which can be dropped onto this node. This means that only those nodes whose 'dndType' is present in list can be dropped onto this . You can restrict DnD operations easily by specifying appropriate 'types' for nodes and targets. With no effort on your part Dojo creates a default 'avatar' for each element when it is being dragged

As you can see, with pure markup and no code at all we added awesome drag and drop functionality to our page. This was simple example of DnD, in next chapters we will dwell into more details of the API. We will see how to customize DnD behaviour, use events and restricting DnD operations.

Beautification

Our last example looked pretty average. After playing with it for some time you may realize that visual indicators for DnD actions could be a great addition. Wouldn't it be nice if you could customize look of nodes so as to indicate important conditions or events in drag and drop functionality?

This is made very easy in Dojo. You don't have to intercept events and add your own code to modify the look of the nodes. Nodes that take part in DnD make use of certain predefined CSS classes. All you need is to supply a declaration to customize look and feel. We will making some changes to the last example to make it beautiful.

First, we will be using these two images in place of those ugly rectangles (BLUE.png and RED.png). [inline:BLUE.png=test] [inline:RED.png=test]

Then, we change our markup a bit. We will use <img> tags instead of . Finally, we add the magic <style> section. Many classes have been defined in this section. You may want to play around with them a bit. A list of such class can be found in ***'dndDefault.css'*** file located in ***dojotoolkit/dojo/tests/dnd*** directory. Given below is the complete source : /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<html>
<head>
<title>Beautification</title>
<style type="text/css">
        .target {border: 1px dotted gray; width: 300px; height: 300px;padding: 5px;
              -moz-border-radius:8pt 8pt;radius:8pt;}
        .source {border: 1px dotted skyblue;height: 200px; width: 300px;
               -moz-border-radius:8pt 8pt;radius:8pt;}
        .dojoDndItemOver {background: #feb;border: 1px dotted gray; }
        .dojoDndItemBefore {border-left: 2px dotted gray; }
        .dojoDndItemAfter {border-right: 2px dotted gray; }
        .target .dojoDndItemAnchor {border:1px solid gray;}
        .dojoDndAvatar {font-size: 75%; color: black;}
        .dojoDndAvatar td {padding-left: 20px; padding-right: 4px;height:20px}
        .dojoDndAvatarHeader {background: #ccc; background-repeat: no-repeat;}
        .dojoDndAvatarItem {background: #eee;}
        .dojoDndMove .dojoDndAvatarHeader {background-image: url(images/dndNoMove.png);}
        .dojoDndMove .dojoDndAvatarCanDrop .dojoDndAvatarHeader {background-image:
                url(images/dndMove.png);}
</style>
<script type="text/javascript" src="../../dojo.js" djConfig="parseOnLoad: true">
</script>
<script type="text/javascript">
        dojo.require("dojo.dnd.source"); // capital-S Source in 1.0
        dojo.require("dojo.parser");
</script>
</head>
<body style="font-size: 12px;">
<h1>A Beautification Example</h1>
<table><tbody><tr>
<td>
SOURCE
<div dojoType="dojo.dnd.Source" jsId="c1" class="source">
        <img src="BLUE.png" class="dojoDndItem" dndType="blue"/>
        <img src="RED.png" class="dojoDndItem" dndType="red"/>
        <img src="BLUE.png" class="dojoDndItem" dndType="blue"/>
        <img src="RED.png" class="dojoDndItem" dndType="red"/>
        <img src="BLUE.png" class="dojoDndItem" dndType="blue"/>
        <img src="RED.png" class="dojoDndItem" dndType="red"/>
        <img src="BLUE.png" class="dojoDndItem" dndType="blue"/>
        <img src="RED.png" class="dojoDndItem" dndType="red"/>
        <img src="BLUE.png" class="dojoDndItem" dndType="blue"/>
</div>
</td>
<td>
TARGET
<div dojoType="dojo.dnd.Target" jsId="c2" class="target" accept="red,blue">
</div>
</td>
</tr><tbody/></table>
</body>
</html>
```

Note that we haven't added any code yet. But now if you run this example you will agree that it certainly looks more beautiful. Some points worth noticing are:

- The target and source and clearly marked.
- Node gets highlighted when mouse hovers over it.

- The node that was last to be dropped shows a black gray border making it distinguishable from others.
- When you move the avatar onto the target, the left or right border of the node beneath get highlighted , indicating whether your node will be inserted before or after it.
- The avatar looks very pretty. Observe the nice little image on top-left of the avatar. It changes to a green arrow when you move the avatar over target. This indicates that it is possible to drop the node there. But if you move the avatar out of the target, it shows a red sign.

All this without writing any special code. With some imagination and creativity, you would be able to generate awesome looking pages, in no time.

Drag and Drop Actions

For a DnD application to be really useful, you would require mechanism to add user defined actions for different DnD events. This could be anything from updating a text box value to sending data back to server using AJAX. In this chapter we will see how to hook up actions to DnD events. This can be achieved with the publish/subscribe mechanism.

The following messages can be subscribed to :

- /dnd/source/over
- /dnd/start
- /dnd/drop
- /dnd/cancel

Given below is code snippet that attaches a callback function to a message: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.subscribe("/dnd/drop", function(source,nodes,iscopy){
  //code to perform some action
});
dojo.subscribe("/dnd/start", function(source,nodes,iscopy){
  //code to perform some action
});
dojo.subscribe("/dnd/source/over", function(source){
  //code to perform some action
});
dojo.subscribe("/dnd/cancel", function(){
  //code to perform some action
});
```

The parameters to the callback function are:

- *source* : Source of the nodes which are being dragged/dropped.
- *nodes*: Array of HTML node objects. To get corresponding DnDItem object use the following method: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
  ```
  var jsnode = source.getItem(nodes[0].id);
  var d = jsnode.data;
  var t  = jsnode.type;
  ```
- *iscopy:* This parameter is a boolean , which is true for a copy operation and false for drag operation.

If you are interested in getting the target object, then you can use following method: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var t = dojo.dnd.manager().target;
```

When we create a target or source via markup, we can specify a name for global level javascript variable for it, using the jsId attribute. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojo.dnd.Source" jsId="cart" class="target" accept="red,blue" id="target1">
</div>
```

This automatically creates a global variable 'cart' of the type "dojo.dnd.Source".

DnD Events Example

Now let us have a look at simple example of DnD events. In this example, I have created a simple shopping cart. Each blue and red sphere inside the source(shelf) has some price associated with it. This is assigned using the 'dndData' item. You can select multiple spheres from the source(shelf) and drop them onto the target (shopping cart). You will see the updated total price of your cart items. If you drop certain items back to the shelf, the corresponding amount is deducted from the total. I have attached a text file with this article which contains complete source. You will require BLUE.png and RED.png images to run the example.

Below is a excerpt from the file followed by explanation : /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<script type="text/javascript">
dojo.require("dojo.dnd.source"); // capital-S Source in Dojo 1.0
var item_price;
var total = 0;
function AddItems(target,nodes)
{
        for (var i=0;i<nodes.length;i++)
        total += parseFloat((target.getItem(nodes[i].id)).data);
        dojo.byId("cost").innerHTML = total;
}
function SubstractItems(target,nodes)
{
        for (var i=0;i<nodes.length;i++)
        total -= parseInt((target.getItem(nodes[i].id)).data);
        dojo.byId("cost").innerHTML = total;
}

function ShowPrice(target,nodes)
{       var sum =0;
        for (var i=0;i<nodes.length;i++)
        sum += parseInt((target.getItem(nodes[i].id)).data);
        dojo.byId("msg").innerHTML = "Selected Item Price is $"+ sum ;
}
function ClearMsg()
{   dojo.byId("msg").innerHTML = "";}
function init()
{

        dojo.subscribe("/dnd/drop", function(source,nodes,iscopy){
              var t = dojo.dnd.manager().target;
                ClearMsg();
                if(t == source) return;
                if(t == cart)AddItems(t,nodes);
                if(t == shelf)SubstractItems(t,nodes);});
        dojo.subscribe("/dnd/start", function(source,nodes,iscopy){
              var t = dojo.dnd.manager().target;
                ShowPrice(source,nodes);});
        dojo.subscribe("/dnd/cancel", function(){
              ClearMsg();});
}
dojo.addOnLoad( init );
</script>
<div dojoType="dojo.dnd.Source" jsId="shelf" class="source" id="source1" accept="red,blue" singular=false>
    <img src="BLUE.png" class="dojoDndItem" dndType="blue" dndData="10" title="$10"/>
    <img src="BLUE.png" class="dojoDndItem" dndType="blue" dndData="3" title="$3"/>
        .....
</div>
<div dojoType="dojo.dnd.Source" jsId="cart" class="target" accept="red,blue" id="target1"></div>
Total Price (USD): <span id="cost">0.00</span>
<b>Message: <span id="msg" style="color:blue"></span></b>
```

- A global level javascript variable 'total' is used to store the updated price.
- When the user starts dragging items, the 'ShowPrice' function gets called, which shows the total price of only the selected spheres.
- The callback function for "dnd\drop" message does the following :
    - If the target and source are the same, exit and don't modify the total. This is because item has been dropped on same source from where it was dragged.
    - If target is cart, then call 'AddItems()' to add the prices of dropped items to the total.
    - If target is shelf, it means that items are being moved from cart to shelf, so call the 'SubstractItems()' function to deduct prices of those items from total.

This was a simple example of handling the DnD events to execute custom defined actions. So far we have covered basics, looked at using CSS for rich user interface and seen how to handle events. In next chapters we will learn about the 'node creator' function and build a very simple web based application to demonstrate use of DnD.

New Events in Dojo 1.0

Moveable implements three new events:

- **onMove**, which implements the move itself
- **onMoving**, which is called before onMove, so you have a chance to change something, for example, the new position of the move to implement some restrictions.
- **onMoved**, which is called after the move, so you can update other objects after the move.

Events have better locality than topics: instead of getting called on every move and check if it is "the right" move, you can connect directly to events on the Moveable. Nevertheless topics are still supported.

Advanced Topics

So far we have seen how to use markup to accomplish drag and drop. In this chapter we will have a look at some features, we have not discussed. This chapter features a sample 'Shopping Cart Application'. You can download the related files and refer to the 'readme.txt' to try it out. Everything we will discuss in this chapter has gone into the making of this application.

Creating Source/Target with javascript
To do this we need to place two container tags on the page. The following javascript code will create a Source and Target object for us. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var node_creator = function(data, hint){
var types = [];
var node = dojo.doc().createElement("div");
```

```
if(data == 'RED'){ types.push("red"); dojo.addClass(node, "redsquare");}
if(data == 'BLUE'){ types.push("blue");dojo.addClass(node, "bluesquare"); }
node.innerHTML = data;
node.id = dojo.dnd.getUniqueId();
return {node: node, data: data, type: types};};
c1 = new dojo.dnd.Source("c1", {creator: node_creator,accept: ["item"],horizontal: false,copyOnly: false});
c1 = new dojo.dnd.Target("c2", { creator: targetnode_creator ,accept: ["item"]});
c2.insertNodes(false, ["RED","GREEN");
```

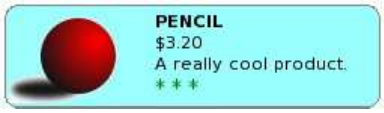
The following points should be noted:

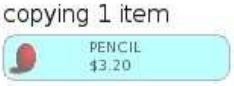
- 'c1' and 'c2' are ids of tags used to create source and target.
- 'node_creator' function needs to be specified. This is called for creation of each node. We will discuss this part in detail in the very next topic.
- 'copyOnly' set to true indicates that always a copy operation is performed (instead of moving nodes). If set to false 'CTRL' key needs to be used for copying.
- 'horizontal' set to true indicates horizontal alignment, else use vertical alignment.
- Nodes can only be inserted using call 'insertNodes'. First parameter if set to true will result in selection of the inserted nodes. Second parameter is an array. We need to understand the working of 'insertNodes' and 'node_creator' in more detail.

The Node Creator Function.

Typically, items that particpate in DnD would originate from server in one form or another. In that case they can't be hard-coded at design time, they have to be created dynamically at runtime. Also it makes sense to assume that the server will only give data (XML,JSON,CSV..). HTML elements (UI) for each of this data item will have to be created at client side using javascript. The node creator function is meant precisely to do this !

```
/* GeSHi (C) 2004 - 2007
Nigel McNie
(http://qbnz.com/highlighter)
*/ .geshifilter {font-family:
monospace;} .geshifilter
.imp {font-weight: bold;
color: red;} .geshifilter .kw1
{color: #000066;
font-weight: bold;}
.geshifilter .kw2 {color:
#003366; font-weight: bold;}
.geshifilter .kw3 {color:
#000066;} .geshifilter .co1
{color: #009900; font-style:
italic;} .geshifilter .coMULTI
{color: #009900; font-style:
italic;} .geshifilter .es0
{color: #000099;
font-weight: bold;}
.geshifilter .br0 {color:
#66cc66;} .geshifilter .st0
{color: #3366CC;}
.geshifilter .nu0 {color:
#CC0000;} .geshifilter .me1
{color: #006600;} .geshifilter
.re0 {color: #0066FF;}
{"id": "S001",
 "name": "PENCIL",
 "price": "3.20",
"rating": "3",
"description": "A really
cool product.",
"img_url":
"images/RED.png"}
```





Sample JSON from server | *HTML node created by the 'node_creator' function.* | *Customized avatar created using 'node_creator'*

Let us first have a look at the call to 'insertNodes()' function. The first parameter is boolean, if set to 'true' will result in marking the inserted nodes as selected. The second parameter is very important. It actually expects an array of javascript objects. Any object can be passed, but most probably you will want to pass data coming from server here. Important thing to know here is that the 'creator' function will be called for each object in the array. Every object in the array is passed to 'creator' function as the 'data' parameter. The creator function creates a HTML node based on our data object. Also, the creator function decides the 'type' and 'id' of node .You can pass any object in the array provided your creator function is capable of handling it.

What data would come from the server and how we would like to interpret is for us to decide. We need to write our 'node_creator' function accordingly. In this case, let us assume that using dojo.xhrGet(AJAX) call, we received the string 'RED,BLUE' from the server. So I have passed two string objects "RED" and "BLUE" in the call to 'insertNodes()'.

Our node creator function is a simple one. It creates a new tag with unique id for each node, based on the value of 'data'. It also assigns 'type' to the corresponding DnD item. The return statement is very important. The creator function returns a javascript object containing the node to be inserted, data related to the node and type assigned to the node. It is possible to assign multiple types to a single node.

The 'hint' parameter is used to customize creation of avatar. Dojo does create a default avatar for you (which is a carbon-copy of the node being dragged). But at times you will want your avatar to show different information or to show the same information in different a way.

Clearing Items from Source/Target

To clear all items you can use the following code: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter

.coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
c2.selectAll();
c2.deleteSelectedNodes();
c2.clearItems();
```

Iterating through all nodes in Source/Target

To iterate through all nodes in a source or target you can use the following code: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var x = c2.getAllNodes();
for(i =0;i<x.length;i++)
{//jsNode is javascript object for node
 //x[i] represents HTML element for the node
var jsnode = c2.getItem(x[i].id);}
```

Simple Shopping Cart Application

This is a simple application which contains a PHP script that acts like a service URL and HTML page, which is the main UI. The HTML page talks to service URL using AJAX for getting list of shop items and saving of cart items. You may want to spend some time looking at the source for this example, to see how web applications involving DnD could be designed.

With the 'Test.php' script following actions are supported:

Method	URL	Result
GET	Test.php?action=get_all_items	Returns JSON for all items in the shop. This is stored in a plain text file (data.txt). In real world scenario it would come from database.
POST	Test.php?action=save_cart_items	Sends JSON for all items in the shopping cart back to server, where they are stored in Session.
GET	Test.php?action=get_cart_items	Returns JSON for all shopping cart items, saved by user (with the intent of buying later) .These were saved in the Session.
GET	Test.php?action=reset	All saved items in shopping cart are deleted. Session is cleared.

If you open the 'shopping_cart.html' page, you will see three buttons 'Save', 'Logout' and 'Reset'. Try dragging and dropping some items from 'Today's Special' on 'Shopping Cart'. Click 'Save' and add some more items on Shopping Cart. Now click 'Logout'. On the logout page, click 'Login Again'. You should see only the saved items in your shopping cart. If you click 'Reset' your shopping cart will become empty.

Important: Since this is just a sample, in the file 'shopping_cart.html' , I have set /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
djConfig="usePlainJson: true"
```

In real life scenario, this could make your page vulnerable to 'JavaScript Hijacking', I haven't bothered about this. It is recommended you use **'text/json-comment-filtered'** as mime type while making XHR calls. This document contains good information on the security risks and ways to prevent it.

To try out the shopping cart application you need to download all the files and follow instructions in 'readme.txt'. These files are attached at the end. This is a very simple and rudimentary example, but still illustrates key concepts in making use of DnD in real world scenarios. This concludes our adventures with the DnD API in Dojo.

Shopping Cart Example Files

- shopping_cart.html
- Test.php
- data.txt
- logout.html
- readme.txt

Please Note:

I am still working on this example. This is my first attempt with PHP programing :-). Any suggestion/optimizations are welcomed. This message will be removed once the work is reviewed officially.

Using dojo.data

What is dojo.data?

Dojo.data is a uniform data access layer that removes the concepts of database drivers and unique data formats. All data is represented as an *item* or as an attribute of an item. With such a representation, data can be accessed in a standard fashion. Out of the box, dojo.data provides a basic /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` for reading a particular format of JSON data. The DojoX project provides more stores (for example, a simple XmlStore, a CsvStore, and an OpmlStore) for working with servers that can output data in such a format. In addition, dojo.data is an API that other users can write to, so you can write one for a custom data format, a specific subset of all the dojo.data APIs, or any other sort of customized data handling service you want to work with. After you have your custom format

accessible using a datastore, widgets that are aware of datastores, and other such code, can then access your data without having to learn new APIs specific to your data.

Ultimately, the goal of dojo.data is to provide a flexible set of APIs as interfaces that datastores can be written to conform to. Stores that conform to the standard interfaces should then be able to be used in a wide variety of applications, widgets, and so on interchangeably. In essence, the API hides the specific structure of the data, be it in JSON, XML, CSV, or some other data format, and provides one way to access items and attributes of items consistently. This also allows optimizations on data access to be placed where they are most appropriate -- the client or the server -- depending on the type, number, and structure the data sets being manipulated.

You can think of dojo.data as one layer above dojo.xhrGet(). Both operate asynchronously, and without refreshing the page. But, xhrGet will get almost any MIME type and return the data in a glob. It's your job to interpret it. With dojo.data, you call one set of APIs to access the data items and the attributes of the items, and it's up to the store to handle the interpretation of the native formats into a common access model.

dojo.data Terminology

dojo.data terminology is similar to relational database terminology. The following table compares and contrasts dojo.data terminology and relational database terminology:

Term	Equiv. Database Term	Description
datastore	cursor	A JavaScript object that reads data from a *data source* and makes that data available as *data items* using the dojo.data APIs.
data source	database	The place that the raw data comes from. For example, in a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `CsvStore`, the data source would be the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `.csv` formatted file that the store loaded. In general, the data source could be a file, a database server, a Web service, or something else completely. They can be as simple as flat, table-like rows, or as complex as a full heirarchical database with nested details.
item	row	A data item that has *attributes* with *attribute values*.
attribute	column	One of the fields or properties of an item.
value	-	The contents of an *attribute* for a given *item*.
reference	--	A value in an item that points to another *item*.
identity	primary key	An identifier that can be used to uniquely identify an item within the context of a single *datastore*.
query	WHERE clause of SQL Select	A specification or request that asks a *datastore* for some subset of the *items* it knows about. A query is often an object with a set of attribute/value pairs that define what attributes should be matched. It is possible, however, that the query could be a string or a number. **Note**: It is highly recommended that all stores use an object structure of attribute name/value pairs as the query format for consistency between stores.
dojo.data APIs	JDBC or ODBC	The standard set of functions that *datastore* implements. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api` module includes a set of APIs (such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `Read` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `Write`) and a *datastore* can implement one or more of the APIs.

Term	Equiv. Database Term	Description
internal data representation	-	The private data structures that a *datastore* uses to cache data in local memory (for example XML DOM nodes, anonymous JSON objects, or arrays of arrays).
request	SQL Select	The parameters used to limit and sort a set of items. This includes the query, sorting attributes, upper and lower limits, and callbacks.

dojo.data Design and APIs

Before diving directly into the APIs of dojo.data, the basic concepts behind the APIs need to be explored because some design descisions that were made might seem odd without an explanation as to why they were chosen. Therefore, read this page in its entirety before moving onto the individual APIs.

Concept 1: Data access is broken down into separate APIs that stores can choose to implement

Data access is broken down into separate APIs because not every service or data backend is able to provide complete access and functions. So not all datastores could possibly implement functions such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `read`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `write`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `identify`, or /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `notifications`. To make it simple to see what features a store provides, each store must provide the 'getFeatures()' function. This function reports which APIs the store implements. The following list of basic APIs are defined:

dojo.data.api.Read

The ability to read data items and attributes of those data items. This also includes the ability to search, sort, and filter data items.

dojo.data.api.Write

The ability to create, delete, and update data items and attributes of those data items. Not all back end services allow for modification of data items. In fact, most public services like Flikr, Delicious, GoogleMaps, for example are primarily read-based data providers.

dojo.data.api.Identity

The ability to locate and look up an item based on its unique identifier, if it has one. Not all data formats have unique identifiers that can be used to look up data items.

dojo.data.api.Notification

The ability to notify listeners for change events on data items in a store. The basic change events for an item are create, delete, and update. These are particularly useful for cases such as a datastore that periodically polls a back end service for data refresh.

Future Features

There are some further functions that the Dojo development community would like to define as additional features stores which might be implemented. However, they have not been completely specified yet and are a work in progress. As such, they are not currently provided in the Dojo Toolkit. Note that the list can change at any time as decisions evolve about what capabilities the dojo.data APIs should provide. The following features are functions that the Dojo development community would like to define as additional features stores to implement:

dojo.data.api.Schema

dojo.data.api.Attribution

Creation and modification of timestamps, author, and other functions of data items.

dojo.data.api.Versioning

Tracking and accessing old versions of data items.

dojo.data.api.Derivation

Attributes derived from other attributes and calculated values

Concept 2: Items and item attributes are always accessed, modified, created, and

deleted through store functions. Attributes are never directly accessed from the item object.

This concept is likely one of the aspects of dojo.data that might seem confusing at first. The following code snippet shows this concept:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new some.data.Store();
  var items;
   ... //Assume in this time items is now an array populated by a call to store.fetch();
  //To iterate over the items and print out the value of a 'foo' attribute, you would do the following:
  for (var i = 0; i < items.length; i++){
      var item = items[i];
      console.log("For attribute 'foo' value was: [" + store.getValue(item, "foo") + "]");
  }
```

This example might make you wonder why attributes are not accessed as shown in one of the following examples:

- `var value = item["foo"];`
- `var value = item.foo;`
- `var value = item.getValue("foo");`

Why is this a requirement of dojo.data?

The following list presents the reasons for this requirement:

- **Efficiency in accessing the values on the items:** By requiring access to go through store functions, the store can hide the internal structure of the item. This allows the item to remain in a format that is most efficient for representing the datatype for a particular situation. For example, the items could be XML DOM elements and, in that case, the store would access the values using DOM APIs when /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.getValue()` is called.

  As a second example, the item might be a simple JavaScript structure and the store can then access the values through normal JavaScript accessor notation. From the end-users perspective, the access is exactly the same: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.getValue(item, "attribute")`. This provides a consistent look and feel to accessing a variety of data types. This also provides efficiency in accessing items by reducing item load times by avoiding conversion to a defined internal format that all stores would have to use.

- **The store could use a very compact internal structure:** This lessens the amount of memory required by a particular store to represent some item and its attribute values.
- **Going through store accessor function provides the possibility of lazy-loading in of values as well as lazy reference resolution.**

The Read API

The most fundamental API of dojo.data is the Read API. All stores will implement this API because all stores need the ability to retrieve and process data items. The Read API is designed to be extremely flexible in how items are handled. The Read API provides the ability to:

- Introspect any datastore to determine the APIs the datastore implements through the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `getFeatures()` call.
- Instrospect, On an item by item basis, what attributes each item has in a way that is agnostic to the data format.
- Get values of attributes in a way that is agnostic to the data format.
- Test attributes of items to see if they contain a specific value.
- Test any JavaScript object to see if it is an item from the store.
- Test to see if an item has been fully loaded from its source or if it is just the stub of an item that needs to be fully loaded.
- Load stub items (lazy-loading).
- Search for items that match a query.
- Sort items in a search.
- Page across items in a search.
- Filter items by the query and wildcard matching.

The following examples, guidelines, and complete API documentation provide further information on the Read API. For more complete examples, review the Using Datastores section.

Example Usage

The following sections provide examples of the Read API in use, as described by each example heading:

Example 1: Listing the APIs supported by a datastore

```
var store = new some.Datastore();
var features = store.getFeatures();
for(var i in features){
    console.log("Store supports feature: " + i);
}
```

Example 2: Testing if an object is a store item

```
var store = new some.Datastore();
if(store.isItem(someObject)){
    console.log("Object was an item.");
}else{
    console.log("Object was NOT an item.");
}
```

Example 3: Listing the attributes of an item

```
var store = new some.Datastore();
...
//Assume that someItem is an item we got from a load.
var attributes = store.getAttributes(someItem);
for(var i = 0; i < attributes.length; i++){
    console.log("Item has attribute; " + attributes[i]);
}
```

Example 4: Testing an item for an attribute

```
var store = new some.Datastore();
...
//Assume that someItem is an item we got from a load.
if(store.hasAttribute(someItem, "foo")){
    console.log("item has attribute foo.");
}else{
    console.log("item DOES NOT have attribute foo.");
}
```

Example 5: Getting the label of an item

```
var store = new some.Datastore();
...
//Assume that someItem is an item we got from a load.
var label = store.getLabel(someItem);
console.log("item has label: " + label);
```

Example 6: Fetching all the items from the store

```
var store = new some.Datastore();
var gotItems = function(items, request){
    console.log("Number of items located: " + items.length);
```

```
};
store.fetch({onComplete: gotItems});
```

Further examples

Further examples of the API usage are covered in the Using Datastores section. Refer to it for examples on paging, sorting, selecting, and so forth.

The complete ReadAPI

The following ReadAPI was taken directly from /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo/data/api/Read.js`:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
getValue: function(    /* item */ item,
                       /* attribute-name-string */ attribute,
                       /* value? */ defaultValue)
        //    summary:
        //        Returns a single attribute value.
        //        Returns defaultValue if and only if *item* does not have a value for *attribute*.
        //        Returns null if and only if null was explicitly set as the attribute value.
        //        Returns undefined if and only if the item does not have a value for the given
        //        attribute (which is the same as saying the item does not have the attribute).
        // description:
        //        Saying that an "item x does not have a value for an attribute y"
        //        is identical to saying that an "item x does not have attribute y".
        //        It is an oxymoron to say "that attribute is present but has no values"
        //        or "the item has that attribute but does not have any attribute values".
        //        If store.hasAttribute(item, attribute) returns false, then
        //        store.getValue(item, attribute) will return undefined.
        //
        //    item:
        //        The item to access values on.
        //    attribute:
        //        The attribute to access represented as a string.
        //    defaultValue:
        //        Optional.  A default value to use for the getValue return in the attribute does not exist or has no value.
        //
        //    exceptions:
        //        Throws an exception if *item* is not an item, or *attribute* is not a string
        //    examples:
        //        var darthVader = store.getValue(lukeSkywalker, "father");
    getValues: function(/* item */ item,
                        /* attribute-name-string */ attribute)
        //    summary:
        //         This getValues() method works just like the getValue() method, but getValues()
        //        always returns an array rather than a single attribute value.  The array
        //        may be empty, may contain a single attribute value, or may contain many
        //        attribute values.
        //        If the item does not have a value for the given attribute, then getValues()
        //        will return an empty array: [].  (So, if store.hasAttribute(item, attribute)
        //        returns false, then store.getValues(item, attribute) will return [].)
        //
        //    item:
        //        The item to access values on.
        //    attribute:
        //        The attribute to access represented as a string.
        //
        //    exceptions:
        //        Throws an exception if *item* is not an item, or *attribute* is not a string
    getAttributes: function(/* item */ item)
        //    summary:
        //        Returns an array with all the attributes that this item has.  This
        //        method will always return an array; if the item has no attributes
        //        at all, getAttributes() will return an empty array: [].
        //
        //    item:
        //        The item to access attributes on.
        //
        //    exceptions:
        //        Throws an exception if *item* is not an item, or *attribute* is not a string
    hasAttribute: function(    /* item */ item,
                           /* attribute-name-string */ attribute)
        //    summary:
        //        Returns true if the given *item* has a value for the given *attribute*.
        //
        //    item:
        //        The item to access attributes on.
        //    attribute:
        //        The attribute to access represented as a string.
        //
        //    exceptions:
        //        Throws an exception if *item* is not an item, or *attribute* is not a string
    containsValue: function(/* item */ item,
                            /* attribute-name-string */ attribute,
                            /* anything */ value)
        //    summary:
        //        Returns true if the given *value* is one of the values that getValues()
        //        would return.
        //
        //    item:
```

```
    //        The item to access values on.
    //    attribute:
    //        The attribute to access represented as a string.
    //    value:
    //        The value to match as a value for the attribute.
    //
    //    exceptions:
    //        Throws an exception if *item* is not an item, or *attribute* is not a string
isItem: function(/* anything */ something)
    //    summary:
    //        Returns true if *something* is an item and came from the store instance.
    //        Returns false if *something* is a literal, an item from another store instance,
    //        or is any object other than an item.
    //
    //    something:
    //        Can be anything.
    //
isItemLoaded: function(/* anything */ something)
    //    summary:
    //        Returns false if isItem(something) is false.  Returns false if
    //        if isItem(something) is true but the the item is not yet loaded
    //        in local memory (for example, if the item has not yet been read
    //        from the server).
    //
    //    something:
    //        Can be anything.
    //

loadItem: function(/* object */ keywordArgs)
    //    summary:
    //        Given an item, this method loads the item so that a subsequent call
    //        to store.isItemLoaded(item) will return true.  If a call to
    //        isItemLoaded() returns true before loadItem() is even called,
    //        then loadItem() need not do any work at all and will not even invoke
    //        the callback handlers.  So, before invoking this method, check that
    //        the item has not already been loaded.
    //     keywordArgs:
    //        An anonymous object that defines the item to load and callbacks to invoke when the
    //        load has completed.  The format of the object is as follows:
    //        {
    //            item: object,
    //            onItem: Function,
    //            onError: Function,
    //            scope: object
    //        }
    //    The *item* parameter.
    //        The item parameter is an object that represents the item in question that should be
    //        contained by the store.  This attribute is required.

    //    The *onItem* parameter.
    //        Function(item)
    //        The onItem parameter is the callback to invoke when the item has been loaded.  It takes only one
    //        parameter, the fully loaded item.
    //
    //    The *onError* parameter.
    //        Function(error)
    //        The onError parameter is the callback to invoke when the item load encountered an error.  It takes only one
    //        parameter, the error object
    //
    //    The *scope* parameter.
    //        If a scope object is provided, all of the callback functions (onItem,
    //        onError, etc) will be invoked in the context of the scope object.
    //        In the body of the callback function, the value of the "this"
    //        keyword will be the scope object.   If no scope object is provided,
    //        the callback functions will be called in the context of dojo.global().
    //        For example, onItem.call(scope, item, request) vs.
    //        onItem.call(dojo.global(), item, request)
fetch: function(/* Object */ keywordArgs)
    //    summary:
    //        Given a query and set of defined options, such as a start and count of items to return,
    //        this method executes the query and makes the results available as data items.
    //        The format and expectations of stores is that they operate in a generally asynchronous
    //        manner, therefore callbacks are always used to return items located by the fetch parameters.
    //
    //    description:
    //        A Request object will always be returned and is returned immediately.
    //        The basic request is nothing more than the keyword args passed to fetch and
    //        an additional function attached, abort().  The returned request object may then be used
    //        to cancel a fetch.  All data items returns are passed through the callbacks defined in the
    //        fetch parameters and are not present on the 'request' object.
    //
    //        This does not mean that custom stores can not add methods and properties to the request object
    //        returned, only that the API does not require it.  For more info about the Request API,
    //        see dojo.data.api.Request
    //
    //    keywordArgs:
    //        The keywordArgs parameter may either be an instance of
    //        conforming to dojo.data.api.Request or may be a simple anonymous object
    //        that may contain any of the following:
    //        {
    //            query: query-string or query-object,
    //            queryOptions: object,
    //            onBegin: Function,
    //            onItem: Function,
    //            onComplete: Function,
    //            onError: Function,
    //            scope: object,
    //            start: int
    //            count: int
    //            sort: array
    //        }
    //        All implementations should accept keywordArgs objects with any of
    //        the 9 standard properties: query, onBegin, onItem, onComplete, onError
    //        scope, sort, start, and count.  Some implementations may accept additional
    //        properties in the keywordArgs object as valid parameters, such as
    //        {includeOutliers:true}.
```

```
        //
        //    The *query* parameter.
        //        The query may be optional in some data store implementations.
        //        The dojo.data.api.Read API does not specify the syntax or semantics
        //        of the query itself -- each different data store implementation
        //        may have its own notion of what a query should look like.
        //        In most implementations the query will probably be a string, but
        //        in some implementations the query might be a Date, or a number,
        //        or some complex keyword parameter object.  The dojo.data.api.Read
        //        API is completely agnostic about what the query actually is.
        //        In general for query objects that accept strings as attribute value matches,
        //        the store should support basic filtering capability, such as * (match any character)
        //        and ? (match single character).
        //
        //    The *queryOptions* parameter
        //        The queryOptions parameter is an optional parameter used to specify optiosn that may modify
        //        the query in some fashion, such as doing a case insensitive search, or doing a deep search
        //        where all items in a hierarchical representation of data are scanned instead of just the root
        //        items.  It currently defines two options that all datastores should attempt to honor if possible:
        //        {
        //            ignoreCase: boolean, //Whether or not the query should match case sensitively or not.  Default behaviour
is false.
        //            deep: boolean     //Whether or not a fetch should do a deep search of items and all child
        //                              //items instead of just root-level items in a datastore.  Default is false.
        //        }
        //
        //    The *onBegin* parameter.
        //        function(size, request);
        //        If an onBegin callback function is provided, the callback function
        //        will be called just once, before the first onItem callback is called.
        //        The onBegin callback function will be passed two arguments, the
        //        the total number of items identified and the Request object.  If the total number is
        //        unknown, then size will be -1.  Note that size is not necessarily the size of the
        //        collection of items returned from the query, as the request may have specified to return only a
        //        subset of the total set of items through the use of the start and count parameters.
        //
        //    The *onItem* parameter.
        //        function(item, request);
        //        If an onItem callback function is provided, the callback function
        //        will be called as each item in the result is received. The callback
        //        function will be passed two arguments: the item itself, and the
        //        Request object.
        //
        //    The *onComplete* parameter.
        //        function(items, request);
        //
        //        If an onComplete callback function is provided, the callback function
        //        will be called just once, after the last onItem callback is called.
        //        Note that if the onItem callback is not present, then onComplete will be passed
        //        an array containing all items which matched the query and the request object.
        //        If the onItem callback is present, then onComplete is called as:
        //        onComplete(null, request).
        //
        //    The *onError* parameter.
        //        function(errorData, request);
        //        If an onError callback function is provided, the callback function
        //        will be called if there is any sort of error while attempting to
        //        execute the query.
        //        The onError callback function will be passed two arguments:
        //        an Error object and the Request object.
        //
        //    The *scope* parameter.
        //        If a scope object is provided, all of the callback functions (onItem,
        //        onComplete, onError, etc) will be invoked in the context of the scope
        //        object.  In the body of the callback function, the value of the "this"
        //        keyword will be the scope object.   If no scope object is provided,
        //        the callback functions will be called in the context of dojo.global().
        //        For example, onItem.call(scope, item, request) vs.
        //        onItem.call(dojo.global(), item, request)
        //
        //    The *start* parameter.
        //        If a start parameter is specified, this is a indication to the datastore to
        //        only start returning items once the start number of items have been located and
        //        skipped.  When this parameter is paired withh 'count', the store should be able
        //        to page across queries with millions of hits by only returning subsets of the
        //        hits for each query
        //
        //    The *count* parameter.
        //        If a count parameter is specified, this is a indication to the datastore to
        //        only return up to that many items.  This allows a fetch call that may have
        //        millions of item matches to be paired down to something reasonable.
        //
        //    The *sort* parameter.
        //        If a sort parameter is specified, this is a indication to the datastore to
        //        sort the items in some manner before returning the items.  The array is an array of
        //        javascript objects that must conform to the following format to be applied to the
        //        fetching of items:
        //        {
        //            attribute: attribute || attribute-name-string,
        //            descending: true|false;   // Optional.  Default is false.
        //        }
        //        Note that when comparing attributes, if an item contains no value for the attribute
        //        (undefined), then it the default ascending sort logic should push it to the bottom
        //        of the list.  In the descending order case, it such items should appear at the top of the list.
        //
        //    returns:
        //        The fetch() method will return a javascript object conforming to the API
        //        defined in dojo.data.api.Request.  In general, it will be the keywordArgs
        //        object returned with the required functions in Request.js attached.
        //        Its general purpose is to provide a convenient way for a caller to abort an
        //        ongoing fetch.
        //
        //        The Request object may also have additional properties when it is returned
        //        such as request.store property, which is a pointer to the datastore object that
        //        fetch() is a method of.
        //
```

```
        //    exceptions:
        //        Throws an exception if the query is not valid, or if the query
        //        is required but was not supplied.
    getFeatures: function()
        //    summary:
        //        The getFeatures() method returns an simple keyword values object
        //        that specifies what interface features the datastore implements.
        //        A simple CsvStore may be read-only, and the only feature it
        //        implements will be the 'dojo.data.api.Read' interface, so the
        //        getFeatures() method will return an object like this one:
        //        {'dojo.data.api.Read': true}.
        //        A more sophisticated datastore might implement a variety of
        //        interface features, like 'dojo.data.api.Read', 'dojo.data.api.Write',
        //        'dojo.data.api.Identity', and 'dojo.data.api.Attribution'.
    close: function(/*dojo.data.api.Request || keywordArgs || null */ request)
        //    summary:
        //        The close() method is intended for instructing the store to 'close' out
        //        any information associated with a particular request.
        //
        //    description:
        //        The close() method is intended for instructing the store to 'close' out
        //        any information associated with a particular request.  In general, this API
        //        expects to recieve as a parameter a request object returned from a fetch.
        //        It will then close out anything associated with that request, such as
        //        clearing any internal datastore caches and closing any 'open' connections.
        //        For some store implementations, this call may be a no-op.
        //
        //    request:
        //        An instance of a request for the store to use to identify what to close out.
        //        If no request is passed, then the store should clear all internal caches (if any)
        //        and close out all 'open' connections.  It does not render the store unusable from
        //        there on, it merely cleans out any current data and resets the store to initial
        //        state.
    getLabel: function(/* item */ item)
        //    summary:
        //        Method to inspect the item and return a user-readable 'label' for the item
        //        that provides a general/adequate description of what the item is.
        //
        //    description:
        //        Method to inspect the item and return a user-readable 'label' for the item
        //        that provides a general/adequate description of what the item is.  In general
        //        most labels will be a specific attribute value or collection of the attribute
        //        values that combine to label the item in some manner.  For example for an item
        //        that represents a person it may return the label as:  "firstname lastlame" where
        //        the firstname and lastname are attributes on the item.  If the store is unable
        //        to determine an adequate human readable label, it should return undefined.  Users that wish
        //        to customize how a store instance labels items should replace the getLabel() function on
        //        their instance of the store, or extend the store and replace the function in
        //        the extension class.
        //
        //    item:
        //        The item to return the label for.
        //
        //    returns:
        //        A user-readable string representing the item or undefined if no user-readable label can
        //        be generated.
    getLabelAttributes: function(/* item */ item)
        //    summary:
        //        Method to inspect the item and return an array of what attributes of the item were used
        //        to generate its label, if any.
        //
        //    description:
        //        Method to inspect the item and return an array of what attributes of the item were used
        //        to generate its label, if any.  This function is to assist UI developers in knowing what
        //        attributes can be ignored out of the attributes an item has when displaying it, in cases
        //        where the UI is using the label as an overall identifer should they wish to hide
        //        redundant information.
        //
        //    item:
        //        The item to return the list of label attributes for.
        //
        //    returns:
        //        An array of attribute names that were used to generate the label, or null if public attributes
        //        were not used to generate the label.
```

The Write API

Some datastores provide the ability to create new items and save those items back to a service, in addition to simply reading items from a service. Stores with this capability will implement the Write API, which provides standard functions for creating new items, modifing existing items, and deleting existing items. Review the following examples, guidelines, and complete API documentation for further information on the Write API.

Example Usage

The following sections provide examples of the Read API in use, as described by each example heading:

Example 1: Simple attribute modification and save

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
//Instantiate some write implementing store.
var store = some.DataWriteStore(<init params>);
//Set our load completed hander up...
var onCompleteFetch = function(items, request) {
    //Define the save callbacks to use
```

```
    var onSave = function(){
       alert("Save done.");
    }
    var onSaveError = function(error) {
       alert("Error occurred: " + error)
    }
    // Process the items and update attribute 'foo'
    for (int i = 0; i < items.length(); i++){
       var item = items[i];
       store.setValue(item, "foo", ("bar" + 1));
    }

    // If the store has modified items (it should), call save with the handlers above.
    if (store.isDirty()){
       store.save({onComplete: onSave, onError: onSaveError});
    }
}
//Define a fetch error handler, just in case.
var onFetchError = function(error, request){
    alert("Fetch failed.  " + error);
}
// Fetch some data...  All items with a foo attribute, any value.
store.fetch({query: {foo:"*"}, onComplete: onCompleteFetch});
```

Example 2: Simple emit of all modified items (before a save has been called)

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
//Instantiate some write implementing store.
var store = some.DataWriteStore(<init params>);
//Set our load completed hander up...
var onCompleteFetch = function(items, request) {
    // Process the items test for modification
    for (int i = 0; i < items.length(); i++){
       var item = items[i];
       if (store.isDirty(item){
          alert("Item with label: " + store.getLabel(item) + " is dirty.");
       }
    }
}
//Define a fetch error handler, just in case.
var onFetchError = function(error, request){
    alert("Fetch failed.  " + error);
}
// Fetch some data...  All items, in fact (no query should return everything)
store.fetch({onComplete: onCompleteFetch});
```

Write API requirements

The following list provides the requirements for the Write API:

- Datastores that implement the Write interface act as a two-phase intermediary between the client and the ultimate provider or service that handles the data. This allows for the batching of operations, such as creating a set of new items and then saving them all back to the persistent store with one function call.
- The save API is defined as asynchronous. This is because most datastores will be talking to a server and not all I/O methods for server communication can perform synchronous operations.
- Datastores track all /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `newItem`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `deleteItem`, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `setAttribute` calls on items so that the store can both save the items to the persistent store in one chunk and have the ability to revert out all the current changes and return to a pristine (unmodified) data set.
- Revert should only revert the store items on the client side back to the point the last save was called.
- Datastores, in their /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `Save` function, account for any copying of items and generation of save format required by the back end service before it enters into the asynchronous I/O with the server. This is to avoid any contention issues with modifications that are occurring while the datastore is is waiting for the server I/O to complete.
- The parameter to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `newItem` is a /* GeSHi (C) 2004 - 2007 Nigel McNie

(http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `keywordArgs` object. For ease of interoperability, this parameter should be constructed as a JavaScrpt object with attribute names and values that match the conceptual structure of the attribute list the item would return. For example, if the source store is an XML backed store, a call to create a new XML Element in that store with attributes /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `foo`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `bar`, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `bit`, should look like this: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
store.newItem({foo: "fooValue", bar: "barValue", bit: "bitValue"});
```

The store will then handle constructing the actual DOMElement with the appropriate DOM attributes.

- Items returned from /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.newItem()` are valid items. In other words, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.isItem(item)` returns true.
- Items returned from /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.newItem()` are dirty items until the next save. In other words, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.isDirty(item)` returns true.
- Items deleted by /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.deleteItem()` are no longer valid items. In other words, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.isItem(item)` returns false unless /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `store.revert()` is called and the delete is undone.

The complete Write API

The following Write API was taken directly from dojo/data/api/Write.js):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
newItem: function(/* Object? */ keywordArgs, /*Object?*/ parentInfo){
        //   summary:
        //        Returns a newly created item.  Sets the attributes of the new
        //        item based on the *keywordArgs* provided.  In general, the attribute
        //        names in the keywords become the attributes in the new item and as for
        //        the attribute values in keywordArgs, they become the values of the attributes
        //        in the new item.  In addition, for stores that support hierarchical item
        //        creation, an optional second parameter is accepted that defines what item is the parent
        //        of the new item and what attribute of that item should the new item be assigned to.
```

```
        //        In general, this will assume that the attribute targetted is multi-valued and a new item
        //        is appended onto the list of values for that attribute.
        //
        //    keywordArgs:
        //        A javascript object defining the initial content of the item as a set of JavaScript 'property name: value'
pairs.
        //    parentInfo:
        //        An optional javascript object defining what item is the parent of this item (in a hierarchical store.  Not
all stores do hierarchical items),
        //        and what attribute of that parent to assign the new item to.  If this is present, and the attribute specified
        //        is a multi-valued attribute, it will append this item into the array of values for that attribute.  The
structure
        //        of the object is as follows:
        //        {
        //            parent: someItem,
        //            attribute: "attribute-name-string"
        //        }
        //
        //    exceptions:
        //        Throws an exception if *keywordArgs* is a string or a number or
        //        anything other than a simple anonymous object.
        //        Throws an exception if the item in parentInfo is not an item from the store
        //        or if the attribute isn't an attribute name string.
        //    examples:
        //        var kermit = store.newItem({name: "Kermit", color:[blue, green]});
    deleteItem: function(/* item */ item)
        //    summary:
        //        Deletes an item from the store.
        //
        //    item:
        //        The item to delete.
        //
        //    exceptions:
        //        Throws an exception if the argument *item* is not an item
        //        (if store.isItem(item) returns false).
        //    examples:
        //        var success = store.deleteItem(kermit);
    setValue: function(    /* item */ item,
                           /* string */ attribute,
                           /* almost anything */ value)
        //    summary:
        //        Sets the value of an attribute on an item.
        //        Replaces any previous value or values.
        //
        //    item:
        //        The item to modify.
        //    attribute:
        //        The attribute of the item to change represented as a string name.
        //    value:
        //        The value to assign to the item.
        //
        //    exceptions:
        //        Throws an exception if *item* is not an item, or if *attribute*
        //        is neither an attribute object or a string.
        //        Throws an exception if *value* is undefined.
        //    examples:
        //        var success = store.set(kermit, "color", "green");
    setValues: function(/* item */ item,
                           /* string */ attribute,
                           /* array */ values)
        //    summary:
        //        Adds each value in the *values* array as a value of the given
        //        attribute on the given item.
        //        Replaces any previous value or values.
        //        Calling store.setValues(x, y, []) (with *values* as an empty array) has
        //        the same effect as calling store.unsetAttribute(x, y).
        //
        //    item:
        //        The item to modify.
        //    attribute:
        //        The attribute of the item to change represented as a string name.
        //    values:
        //        An array of values to assign to the attribute..
        //
        //    exceptions:
        //        Throws an exception if *values* is not an array, if *item* is not an
        //        item, or if *attribute* is neither an attribute object or a string.
        //    examples:
        //        var success = store.setValues(kermit, "color", ["green", "aqua"]);
        //        success = store.setValues(kermit, "color", []);
        //        if (success) {assert(!store.hasAttribute(kermit, "color"));}
    unsetAttribute: function(    /* item */ item,
                                 /* string */ attribute)
        //    summary:
        //        Deletes all the values of an attribute on an item.
        //
        //    item:
        //        The item to modify.
        //    attribute:
        //        The attribute of the item to unset represented as a string.
        //
        //    exceptions:
        //        Throws an exception if *item* is not an item, or if *attribute*
        //        is neither an attribute object or a string.
        //    examples:
        //        var success = store.unsetAttribute(kermit, "color");
        //        if (success) {assert(!store.hasAttribute(kermit, "color"));}
    save: function(/* object */ keywordArgs)
        //    summary:
        //        Saves to the server all the changes that have been made locally.
        //        The save operation may take some time and is generally performed
        //        in an asynchronous fashion.  The outcome of the save action is
        //        is passed into the set of supported callbacks for the save.
        //
        //    keywordArgs:
        //        {
```

```
        //            onComplete: function
        //            onError: function
        //            scope: object
        //        }
        //
        //    The *onComplete* parameter.
        //        function();
        //
        //        If an onComplete callback function is provided, the callback function
        //        will be called just once, after the save has completed.  No parameters
        //        are generally passed to the onComplete.
        //
        //    The *onError* parameter.
        //        function(errorData);
        //
        //        If an onError callback function is provided, the callback function
        //        will be called if there is any sort of error while attempting to
        //        execute the save.  The onError function will be based one parameter, the
        //        error.
        //
        //    The *scope* parameter.
        //        If a scope object is provided, all of the callback function (
        //        onComplete, onError, etc) will be invoked in the context of the scope
        //        object.  In the body of the callback function, the value of the "this"
        //        keyword will be the scope object.   If no scope object is provided,
        //        the callback functions will be called in the context of dojo.global.
        //        For example, onComplete.call(scope) vs.
        //        onComplete.call(dojo.global)
        //
        //    returns:
        //        Nothing.  Since the saves are generally asynchronous, there is
        //        no need to return anything.  All results are passed via callbacks.
        //    examples:
        //        store.save({onComplete: onSave});
        //        store.save({scope: fooObj, onComplete: onSave, onError: saveFailed});
    revert: function()
        //    summary:
        //        Discards any unsaved changes.
        //    description:
        //        Discards any unsaved changes.
        //
        //    examples:
        //        var success = store.revert();
    isDirty: function(/* item? */ item)
        //    summary:
        //        Given an item, isDirty() returns true if the item has been modified
        //        since the last save().  If isDirty() is called with no *item* argument,
        //        then this method returns true if any item has been modified since
        //        the last save().
        //
        //    item:
        //        The item to check.
        //
        //    exceptions:
        //        Throws an exception if isDirty() is passed an argument and the
        //        argument is not an item.
        //    examples:
        //        var trueOrFalse = store.isDirty(kermit); // true if kermit is dirty
        //        var trueOrFalse = store.isDirty();       // true if any item is dirty
```

The Identity API

The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Identity` interface defines the set of APIs that are implemented by a datastore if a data source provides a method by which to uniquely identify each item. This API then allows users of that datastore to request a specific item without searching for an item that matches specific attributes. Review the following examples, guidelines, and complete API documentation for futher information on the Identity API.

Example usage

For all of the examples in the following sections, assume that there is a simple /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` instantiation from the following data in the countries.json file:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ identifier: 'abbr',
  label: 'name',
  items: [
    { abbr:'ec', name:'Ecuador',          capital:'Quito' },
    { abbr:'eg', name:'Egypt',            capital:'Cairo' },
    { abbr:'sv', name:'El Salvador',      capital:'San Salvador' },
    { abbr:'gq', name:'Equatorial Guinea', capital:'Malabo' },
    { abbr:'er', name:'Eritrea',          capital:'Asmara' },
    { abbr:'ee', name:'Estonia',          capital:'Tallinn' },
    { abbr:'et', name:'Ethiopia',         capital:'Addis Ababa' }
]}
```

Example 1: Basic lookup of an item by identity

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var itemStore = new dojo.data.ItemFileReadStore({url: countries.json});
function failed(error) {
         ... //Do something with the provided error.
}
function gotItem(item) {
    if (itemStore.isItem(item)){
        if(!(itemStore.getValue(item,"name") === "El Salvador")){{
           failed(new Error("The item loaded does not have the attribute value for attribute [name] expected."));
        }else{
           ... //Do something with it.
        }
    }else{
        //This should never occur.
        console.log("Unable to locate the item with identity [sv]");
    }
}
//Invoke the lookup.  This is an async call as it may have to call back to a server to get data.
itemStore.fetchItemByIdentity({identity: "sv" onItem: gotItem, onError: failed});
```

Example 2: Obtaining the value of an item's identity

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var itemStore = new dojo.data.ItemFileReadStore({url: countries.json});
...
function onError(error, request){
    ... //Do something with the provided error.
}
function onComplete(items, request) {
    if(items.length === 1){
       var identifier = itemStore.getIdentity(items[0]);
       if(identifier !== null && identifier === "er"){
          ... //Do something with the located identity.
       }else{
          onError(new Error("The identifier returned does not match what was expected."), request);
       }
    }else{
       onError(new Error("Too many matches found."), request);
    }
}
//Search the store and find the item with the name Eritrea
itemStore.fetch({query: {name:"Eritrea"}, onComplete: onComplete, onError: onError});
```

Example 3: Obtaining the list of attributes that comprise the identity of an item

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var itemStore = new dojo.data.ItemFileReadStore({url: countries.json});
function failed(error) {
         ... //Do something with the provided error.
}
function gotItem(item) {
    if (itemStore.isItem(item)){
        if(!(itemStore .getValue(item,"name") === "El Salvador")){{
           failed(new Error("The item loaded does not have the attribute value for attribute [name] expected."));
        }else{
           var identityAttributes = itemStore .gettityAttributes(item);
           if(identityAttributes !== null){
              for(var i = 0; i < identityAttributes.length; i++){
                  var identifier = identityAttributes[i];
                  ... //Do something with 'identifier'.
              }
           }else{
              failed(new Error("Unable to locate the list of attributes comprising the identity."));
           }
        }
    }else{
        //This should never occur.
        throw new Error("Unable to locate the item with identity [sv]");
        }
}
//Invoke the lookup.  This is an async call as it may have to call back to a server to get data.
itemStore.fetchItemByIdentity({identity: "sv" onItem: gotItem, onError: failed});
```

Futher Examples

To see more examples of the Identity API, refer to the test cases defined in the /* GeSHi (C) 2004 - 2007 Nigel McNie

(http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo/tests/data/ItemFileReadStore.js` file of your dojo source tree.

Identity API Requirements

The following list provides the requirements for the Identity API:

- The identifiers must always be an object that can be converted to a string using the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `toString()` JavaScript API.

  Note: This does not keep identities from being compound keys; they just must be able to be represented in a string fashion.

- Stores that implement the Identity API may expose the identity as a publicly accessible *attribute* on the item, or they may implement the identity as a private *attribute*.
- Stores that expose the identity of the store as a public attribute (or set of attributes), must return the attribute(s) identifier(s) from the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `getIdentityAttributes()` method. If they are not exposed as public attributes, then the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `getIdentifyAttributes()` method must return a null value.
- All identifier values must be unique and address only one item within the store.
- The store's /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `getFeatures()` function will return, as part of its associative map, a property with the key name of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Identity`. The value of the property can be anything reasonable, such as the boolean value *true*, the name of the attribute that represents the identity, an array of attributes, or even an object. By the mere presence of this key in the map, the store declares that it implements this API.

The Complete Identity API

The following Identity API was taken directly from /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo/data/api/Identity.js`:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
getIdentity: function(/* item */ item)
        //    summary:
        //        Returns a unique identifier for an item.  The return value will be
        //        either a string or something that has a toString() method.
        //    item:
        //        The item from the store from which to obtain its identifier.
        //    exceptions:
        //        Conforming implementations may throw an exception or return null if
        //        item is not an item.
    getIdentityAttributes: function(/* item */ item)
        //    summary:
        //        Returns an array of attribute names that are used to generate the identity.
        //        For most stores, this is a single attribute, but for some complex stores
        //        such as RDB backed stores that use compound (multi-attribute) identifiers
        //        it can be more than one.  If the identity is not composed of attributes
        //        on the item, it will return null.  This function is intended to identify
        //        the attributes that comprise the identity so that so that during a render
        //        of all attributes, the UI can hide the the identity information if it
        //        chooses.
        //    item:
        //        The item from the store from which to obtain the array of public attributes that
        //        compose the identifier, if any.
    fetchItemByIdentity: function(/* object */ keywordArgs){
        //    summary:
        //        Given the identity of an item, this method returns the item that has
        //        that identity through the onItem callback.  Conforming implementations
```

```
//        should return null if there is no item with the given identity.
//        Implementations of fetchItemByIdentity() may sometimes return an item
//        from a local cache and may sometimes fetch an item from a remote server,
//
//    keywordArgs:
//        An anonymous object that defines the item to locate and callbacks to invoke when the
//        item has been located and load has completed.  The format of the object is as follows:
//        {
//            identity: string|object,
//            onItem: Function,
//            onError: Function,
//            scope: object
//        }
//    The *identity* parameter.
//        The identity parameter is the identity of the item you wish to locate and load
//        This attribute is required.  It should be a string or an object that toString()
//        can be called on.
//
//    The *onItem* parameter.
//        Function(item)
//        The onItem parameter is the callback to invoke when the item has been loaded.  It takes only one
//        parameter, the item located, or null if none found.
//
//    The *onError* parameter.
//        Function(error)
//        The onError parameter is the callback to invoke when the item load encountered an error.  It takes only one
//        parameter, the error object
//
//    The *scope* parameter.
//        If a scope object is provided, all of the callback functions (onItem,
//        onError, etc) will be invoked in the context of the scope object.
//        In the body of the callback function, the value of the "this"
//        keyword will be the scope object.   If no scope object is provided,
//        the callback functions will be called in the context of dojo.global.
//        For example, onItem.call(scope, item, request) vs.
//        onItem.call(dojo.global, item, request)
```

The Notification API

When working with data and items, sometimes it is useful to be notified when items are created, deleted, or modified within a given dojo.data datastore. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Notification` feature is implemented by stores to expose such a capability. This set of functions defines monitoring for the main change events a store can see on an item: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `create`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `modify`, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `delete`. Review the following examples, guidelines, and complete API documentation for further information on the Notification API.

Example Usage

There are two general patterns of listening on these functions for change events. The first pattern is to use the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.connect()` event model to bind to the function on the store and have one of your functions called whenever the store calls the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onSet`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onNew`, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onDelete` functions. The second pattern is to replace the implementation of the notification functions on the store with custom logic to do something each time the store calls the function. Example usage of such functions are provided in the following examples.

Example 1: Basic dojo.connect to a DataStore onNew

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:

bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = some.NotifyWriteStore();
var alertOnNew = function(item) {
    var label = store.getLabel(item);
    alert("New item was created: [" + label + "]");
};
dojo.connect(store, "onNew", alertOnNew);
//An alert should be thrown when this completes
var newItem = store.newItem({foo:"bar"});
```

Example 2: Replacing the onNew function of the store with a custom one

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = some.NotifyWriteStore();
store.onNew = function(item) {
    var label = this.getLabel(item);
    alert("New item was created: [" + label + "]");
};
//An alert should be thrown when this completes
var newItem = store.newItem({foo:"bar"});
```

Further Information

Note that the previous examples can be applied to any of the store Notification APIs. Also note that, by doing either the connect or the function override on the store, you can implement a variety of item event handling. This includes performing actions only when desired items are effected, as needed for your particular notification scenario. As an example, you could look at a list of items your store has specifically kept track of as items of interest and only perform an operation when the item passed into the on*() function matches one in the list.

Notification API Requirements

As with all DataStores, not all stores will implement this API. For stores that implement this API, the following assumptions should be made:

- All change events on items (/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `create`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `set attribute` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `delete`) will trigger a call to the appropriate store notification function.
- Notifications occur at a store level and for all items. The Notifications API does not cover registering to listen for only particular items being modified. That does not mean it cannot be done in a custom store with an extended set of functions, only that such behavior is not defined by the specification. This is because, after careful analysis, it was determined by the dojo community that a per-item registration scheme could get extremely costly in terms of CPU time or object construction and was therefore avoided.
- Most store implementations of Notification should implement these functions as simple no-op bind points for efficiency and performance. This also provides safety, should you want to replace the stub function with a more complex implementation or notification (or both) scheme.

The Complete Notification API

The following Notification API was taken directly from /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo/data/api/Notification.js`:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
onSet: function(/* item */ item,
                    /* attribute-name-string */ attribute,
                    /* object | array */ oldValue,
                    /* object | array */ newValue)
        //    summary:
        //        This function is called any time an item is modified via setValue, setValues, unsetAttribute, etc.
        //    description:
        //        This function is called any time an item is modified via setValue, setValues, unsetAttribute, etc.
```

```
        //        Its purpose is to provide a hook point for those who wish to monitor actions on items in the store
        //        in a simple manner.  The general expected usage is to dojo.connect() to the store's
        //        implementation and be called after the store function is called.
        //
        //    item:
        //        The item being modified.
        //    attribute:
        //        The attribute being changed represented as a string name.
        //    oldValue:
        //        The old value of the attribute.  In the case of single value calls, such as setValue, unsetAttribute, etc,
        //        this value will be generally be an atomic value of some sort (string, int, etc, object).  In the case of
        //        multi-valued attributes, it will be an array.
        //    newValue:
        //        The new value of the attribute.  In the case of single value calls, such as setValue, this value will be
        //        generally be an atomic value of some sort (string, int, etc, object).  In the case of multi-valued
attributes,
        //        it will be an array.  In the case of unsetAttribute, the new value will be 'undefined'.
        //
        //    returns:
        //        Nothing.
    onNew: function(/* item */ newItem, /*object?*/ parentInfo){
        //    summary:
        //        This function is called any time a new item is created in the store.
        //        It is called immediately after the store newItem processing has completed.
        //    description:
        //        This function is called any time a new item is created in the store.
        //        It is called immediately after the store newItem processing has completed.
        //
        //    newItem:
        //        The item created.
        //    parentInfo:
        //        An optional javascript object that is passed when the item created was placed in the store
        //        hierarchy as a value f another item's attribute, instead of a root level item.  Note that if this
        //        function is invoked with a value for parentInfo, then onSet is not invoked stating the attribute of
        //        the parent item was modified.  This is to avoid getting two notification  events occurring when a new item
        //        with a parent is created.  The structure passed in is as follows:
        //        {
        //            item: someItem,                    //The parent item
        //            attribute:        "attribute-name-string",        //The attribute the new item was assigned to.
        //            oldValue: something       //Whatever was the previous value for the attribute.
        //                                      //If it is a single-value attribute only, then this value will be a single
value.
        //                                      //If it was a multi-valued attribute, then this will be an array of all the
values minues the new one.
        //            newValue: something       //The new value of the attribute.  In the case of single value calls, such as
setValue, this value will be
        //                                      //generally be an atomic value of some sort (string, int, etc, object).  In the
case of multi-valued attributes,
        //                                      //it will be an array.
        //        }
        //
        //    returns:
        //        Nothing.
    onDelete: function(/* item */ deletedItem)
        //    summary:
        //        This function is called any time an item is deleted from the store.
        //        It is called immediately after the store deleteItem processing has completed.
        //    description:
        //        This function is called any time an item is deleted from the store.
        //        It is called immediately after the store deleteItem processing has completed.
        //
        //    deletedItem:
        //        The item deleted.
        //
        //    returns:
        //        Nothing.
```

Using Datastores

This section contains documentation on the operations that are most common to data access: Searching, Sorting, Pagination, and Lazy Loading.

A Simple Data Source

The easiest data store is a static one, so let's begin with that. The file in the following example has the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `/pantry_spices.json` URL: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ identifier: 'name',
   items: [
        { name: 'Adobo', aisle: 'Mexican' },
        { name: 'Balsamic vinegar', aisle: 'Condiments' },
        { name: 'Basil', aisle: 'Spices' },
        { name: 'Bay leaf', aisle: 'Spices' },
        { name: 'Beef Bouillon Granules', aisle: 'Soup' },
...
        { name: 'Vinegar', aisle: 'Condiments' },
        { name: 'White cooking wine', aisle: 'Condiments' },
        { name: 'Worcestershire Sauce', aisle: 'Condiments' }]}
```

In this example:

- The data source is /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `/pantry_spices.json`.
- Each item is an ingredient, and each item has two attributes, name and aisle.
- The name attribute is an identifier. Each ingredient has a different name to prevent confusion.

This is a simple but useful technique. Because this data source is a separate file, you can share the data among many pages. But what can you do with it? The following simple sample displays a select list of pantry items:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojo.data.ItemFileReadStore" jsId="pantryStore"
        url="pantry_items.json"></div>
<div
     name="pantry_item" dojoType="dijit.form.FilteringSelect"
     store="pantryStore"
     searchAttr="name" value="Vinegar" autoComplete="true"
></div>
```

The class for the PantryStore, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.ItemFileReadStore`, tells dojo.data to expect the data in a specific format that uses JSON structure to store the data. Our static file is in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `pantry_items.json` so this is the URL. The target could also be a dynamic, server-run script that returns the specific data in the defined JSON format. Other widgets, like Tree and Grid, also use the dojo.data objects and URL to load data into them.

Fetching Single Items and Values

For this example, we'll assume the following simple data source:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ identifier: 'name',
  items: [
        { name: 'Adobo', aisle: 'Mexican' },
        { name: 'Balsamic vinegar', aisle: 'Condiments' },
        { name: 'Basil', aisle: 'Spices' },
        { name: 'Bay leaf', aisle: 'Spices' },
        { name: 'Beef Bouillon Granules', aisle: 'Soup' },
...
        { name: 'Vinegar', aisle: 'Condiments' },
        { name: 'White cooking wine', aisle: 'Condiments' },
        { name: 'Worcestershire Sauce', aisle: 'Condiments' }
        { name: 'pepper', aisle: 'Spices' }
]}
```

Working with One Item

You might want to access items directly and work with one item at a time. Stores that implement the identity interface allow you to do this quite easily. In this example of accessing an item by its unique identifier, the following APIs are used:

Identity

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fetchItemByIdentity()` Fetches an item by its key value. Because the identity value of each item is unique, you are guaranteed at most one answer back.

Read

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `getValue()` Takes an item and an attribute and returns the associated value

The following code fragment returns the aisle pepper is in:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:

bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var pantryStore = new dojo.data.ItemFileReadStore({url: "pantry_items.json" } );
    function onItem(item) {
      console.debug('Pepper is in aisle ' + pantryStore.getValue(item,"aisle"));
    }
   pantryStore.fetchItemByIdentity({identity: "pepper", onItem: onItem});
```

Note: In the previous example, the fetch is asynchronous. This is because many Datastores will need to go back to a server to actually look up the data and some I/O methods do not readily allow for a synchronous call.

Fetching Multiple Items and Values

For this example, we'll assume the simple data source detailed on the previous page

Working with Multiple Items

You will likely want to access multiple items from such a data source as in the preceding example. No problem! Dojo.data Read API provides a mechanism for loading a set of items. All you have to do is provide the following information to the fetch function of the Read API:

- This is what I want (if I don't tell you something, get everything)
- Do this if there is an error
- Do that when everything is loaded

If this sounds like it might be event-driven, that's because it is. The prime method to call, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Read.fetch()`, is asynchronous. This is because, in a wide variety of data access cases, the datastore will have to make a request to a server service to get the data. Many I/O methods for doing server requests only behave in an asynchronous manner. Therefore, the primary search function of dojo.data is asynchronous by definition.

In this example, the **Read** API is used with the following values:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fetch()`

> Asynchronous API that fetches a set of items which match a list of attributes.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `getValue()`

> Takes an item and an attribute and returns the associated value.

The following code fragment returns all items:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var pantryStore = new dojo.data.ItemFileReadStore({url: "pantry_items.json" } );
//Define a callback that fires when all the items are returned.
var gotList = function(items, request){
    var itemsList = "";
    for (var i = 0; i < items.length; i++){
       itemsList += pantryStore .getValue(items[i], "name") + " ";
    }
    alert("All items are: " + itemsList);
}
var gotError = function(error, request){
    alert("The request to the store failed. " +  error);
}
//Invoke the search
pantryStore.fetch({onComplete: gotList, onError: gotError});
```

Working with *Lots* of Items

Now that we've looked at dealing with getting a list of items in one batch, what if the list is huge? It could take a long time to get all the items, push them into an array, and then call the callback with the array of items. Wouldn't it be nice if you could stream the items in, one at a time, and do something each time a new item is available? Well, with dojo.data, you can do that! There is an alternate callback you can pass to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;}

.javascript .re0 {color: #0066FF;} `fetch()` that is called on an item by item basis. It is the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onItem` callback.

In the following example, the code will request that all items be returned (an empty query). As each item gets returned, it will add a textnode to the document. In this example, the **Read** API is used with the following values:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fetch()`

Asynchronous API that fetches a set of items which match a list of attributes.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `getValue()`

Takes an item and an attribute and returns the associated value.

The following code fragment loads all items and streams them back into the page:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var pantryStore = new dojo.data.ItemFileReadStore({url: "pantry_items.json" } );
//Define the onComplete callback to write COMPLETED to the page when the fetch has finished returning items.
var done = function(items, request){
    document.appendChild(document.createTextNode("COMPLETED"));
}
//Define the callback that appends a textnode into the document each time an item is returned.
gotItem = function(item, request) {
    document.appendChild(document.createTextNode(pantryStore.getValue(item, "name")));
    document.appendChild(document.createElement("br"));
}
//Define a simple error handler.
var gotError = function(error, request){
    alert("The request to the store failed. " +  error);
}
//Invoke the search
pantryStore.fetch({onComplete: done, onItem: gotItem, onError: gotError});
```

Note: If the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onItem` callback is present in the parameters to fetch, then the first parameter to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onComplete` callback, the items array, will always be null. Therefore, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onItem` is streaming only mode and does not rely on onComplete for anything other than a signal that the streaming has ended.

Selecting Items

There are many times when you might not want an entire item list. Though you could fetch the entire list, and loop through to select elements, dojo.data's API definition has facilities to do the tough work for you.

Selecting subsets of items requires a *query*. A query is a JavaScript object that has attributes which look a lot like the attributes of the data store. It's a kind of query-by-example. So for our pantry, selecting the items from the Spice aisle involves this query:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ aisle: "Spice" }
```

Each type of data store can have its own query syntax. With JsomItemStore, you can use wildcards: * to mean any characters. and ? to mean one character. This notation will be familiar to you if you've worked with Perl, Java, UNIX shell regular expressions, or even old BATCH scripts. And in general, the dojo.data community would highly recommend that all stores try to follow this method of specifying the query for consistency. Even datastores that are backed by an SQL database should be able to handle such character matching, because * maps to %, and ? maps to _ in SQL syntax.

The following query will pick up items in the Condiments aisle:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ aisle: "Condiment*" }
```

Multiple attributes assume an "and" between the terms. So a query like the following one will match spices with the word pepper inside them, but not "green peppers" in the vegetable aisle:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ name: "*pepper*", aisle: "Spices" }
```

Once we have constructed the query, we pass it to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fetch()` along with the other parameters as shown in the following example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<script>
    var itemStore = new dojo.data.ItemFileReadStore({ url: "pantry_items.json"});
    itemStore.fetch({
        query: { name: "*pepper*", aisle: "Spices" },
        onComplete:
                ...
   });
</script>
```

Finally, it's important to note that searches are case-sensitive by default. If you want to make a case-insensitive query, just add the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ignoreCase` option to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `queryOptions` object as shown in the following example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
itemStore .fetch({
        queryOptions: {ignoreCase: true},
        query: { name: "*pepper*", aisle: "Spices" },
        onComplete:
                ...
   });
```

This example will match both "Black Pepper" and "white pepper."

In general, any option that would affect the behavior of a query, such as making it case insensitive or doing a deep scan where it scans a hierarchy of items instead of just the top level items (the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `deep:true` option), in a store belongs in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `queryOptions` argument.

Why isn't it just SQL for a query?

The simple and short answer to this question is that not all datastores are backed directly by a database that handles SQL. An immediate example is ItemFileReadStore, which just uses a structured JSON list for its data. Other examples would be datastores that wrap on top of services like Flickr and Delicious, because neither of those take SQL as the syntax for their services. Therefore, the dojo.data API defines basic guidelines and syntax stores that can be easily mapped to a service (for example, attribute names can map directly to parameters in a query string). The same is true for an SQL backed datastore. The attributes become substitutions in a prepared statement that the stores use (when they pass back the query to the server) and a simple common pattern matching syntax, the * and ?, which also map easily across a wide variety of datasource query syntax.

Nested Items and Lazy Loading

Items can be handled in chunks, as well as streamed in. The other articles about datastores show items as a flat list with no hierarchy. So, what if you want a datastore to represent hierarchical data? And how do you walk across the hierarchy? Walking the hierarchy is, in fact, quite easy to do. The following example shows how to do this using /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `JsonItemStore`.

```
Assume a datasource of:
  { identifier: 'name',
     label: 'name',
     items: [
       { name:'Africa', type:'continent',
          children:[{_reference:'Egypt'}, {_reference:'Kenya'}, {_reference:'Sudan'}] },
       { name:'Egypt', type:'country' },
       { name:'Kenya', type:'country',
          children:[{_reference:'Nairobi'}, {_reference:'Mombasa'}] },
       { name:'Nairobi', type:'city' },
       { name:'Mombasa', type:'city' },
       { name:'Sudan', type:'country',
          children:{_reference:'Khartoum'} },
       { name:'Khartoum', type:'city' },
       { name:'Asia', type:'continent',
          children:[{_reference:'China'}, {_reference:'India'}, {_reference:'Russia'},          {_reference:'Mongolia'}] },
       { name:'China', type:'country' },
       { name:'India', type:'country' },
       { name:'Russia', type:'country' },
       { name:'Mongolia', type:'country' },
       { name:'Australia', type:'continent', population:'21 million',
          children:{_reference:'Commonwealth of Australia'}},
       { name:'Commonwealth of Australia', type:'country', population:'21 million'},
       { name:'Europe', type:'continent',
          children:[{_reference:'Germany'}, {_reference:'France'}, {_reference:'Spain'}, {_reference:'Italy'}] },
       { name:'Germany', type:'country' },
       { name:'France', type:'country' },
       { name:'Spain', type:'country' },
       { name:'Italy', type:'country' },
       { name:'North America', type:'continent',
          children:[{_reference:'Mexico'}, {_reference:'Canada'}, {_reference:'United States of America'}] },
       { name:'Mexico', type:'country',  population:'108 million', area:'1,972,550 sq km',
          children:[{_reference:'Mexico City'}, {_reference:'Guadalajara'}] },
       { name:'Mexico City', type:'city', population:'19 million', timezone:'-6 UTC'},
       { name:'Guadalajara', type:'city', population:'4 million', timezone:'-6 UTC' },
       { name:'Canada', type:'country',  population:'33 million', area:'9,984,670 sq km',
          children:[{_reference:'Ottawa'}, {_reference:'Toronto'}] },
       { name:'Ottawa', type:'city', population:'0.9 million', timezone:'-5 UTC'},
       { name:'Toronto', type:'city', population:'2.5 million', timezone:'-5 UTC' },
       { name:'United States of America', type:'country' },
       { name:'South America', type:'continent',
          children:[{_reference:'Brazil'}, {_reference:'Argentina'}] },
       { name:'Brazil', type:'country', population:'186 million' },
       { name:'Argentina', type:'country', population:'40 million' }
  ]}
```

The above datasource for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `JsonItemStore` uses references to other items to build the hierarchy. Other datasources and datastores might use different internal representations for hierarchy. But, in this example, notice that the continent type items have children that are countries, which in turn have children that are cities.

The following code snippet walks across this hierarchy and displays all country items contained by the continent items:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojo.data.ItemFileReadStore({url: "countries.json"});
//Load completed function for walking across the attributes and child items of the
//located items.
var gotContinents = function(items, request){
    //Cycle over all the matches.
    for(var i = 0; i < items.length; i++){
        var item = items[i];
        //Cycle over all the attributes.
        var attributes = store.getAttributes(item);
        for (var j = 0; j < attributes.length; j++){
            //Assume all attributes are multi-valued and loop over the values ...
            var values = store.getValues(item, attributes[j]);
            for(var k = 0; k < values.length; k++){
                var value = values[k];

                if(store.isItem(value)){
                    //Test to see if the items data is fully loaded or needs to be demand-loaded in (the item in question is
just a stub).
                    if(store.isItemLoaded(value)){
                        console.log("Located a child item with label: [" + store.getLabel(value) + "]");
                    }else{
                        //Asynchronously load in the child item using the stub data to get the real data.
                        var lazyLoadComplete = function(item){
                            console.log("Lazy-Load of item complete.  Located child item with label: [" + store.getLabel(item)
+ "]");
                        }
                        store.loadItem({item: value, onItem: lazyLoadComplete});
```

```
                }
            }else{
                console.log("Attribute: [" + attributes[j] + "] has value: [" + value + "]");
            }
        }
    }
}
//Call the fetch of the toplevel continent items.
store.fetch({query: {type: "continent"}, onComplete: gotContinents});
```

Note: The previous sample also demonstrates how lazy-loading (on-demand) of items can be done through the combination of the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `isItemLoaded()` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `loadItem()` functions. For a demonstration of a lazy-loading approach that uses an extended version of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`, see the demo in dojox/data/demos/demo_LazyLoad.html.

Pagination

As shown in the other datastore sections, the fetch method of the Read API can query across and return sets of items in a variety of ways. However, there is generally only so much space on a display to list all the data returned. Certainly, an application could implement its own custom display logic for just displaying subsets of the data, but that would be inefficient because the application would have had to load all the data in the first place. And, if the data set is huge, it could severely increase the memory usage of the browser. Therefore, dojo.data provides a mechanism by which the store itself can do the paging for you. When you use the paging options of fetch, all that is returned in the callbacks for fetch is the page of data you wanted, no more. This allows the application to deal with data in small chunks, the parts currently visible to you.

The paging mechanism is used by specifying a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `start` parameter in the fetch arguments. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `start` parameter says where, in the full list of items, to start returning items. The index 0 is the first item in the collection. The second argument you specify is the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `count` argument. This option tells dojo.data how many items, starting at /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `start`, to return in a request. If /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `start` isn't specified, it is assumed to be 0. If /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `count` isn't specified, it is assumed to return all the items starting at /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `start` until it reaches the end of the collection. With this mechanism, you can implement simple paging easily.

To demonstrate the paging function, we'll assume an ItemFileReadStore with the following datasource:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{
    identifier: 'name',
    items: [
        { name: 'Adobo', aisle: 'Mexican' },
```

```
        { name: 'Balsamic vinegar', aisle: 'Condiments' },
        { name: 'Basil', aisle: 'Spices' },
        { name: 'Bay leaf', aisle: 'Spices' },
        { name: 'Beef Bouillon Granules', aisle: 'Soup' },
        ...
        { name: 'Vinegar', aisle: 'Condiments' },
        { name: 'White cooking wine', aisle: 'Condiments' },
        { name: 'Worcestershire Sauce', aisle: 'Condiments' }
        { name: 'pepper', aisle: 'Spices' }
    ]
}
```

The following code snippet would allow for paging over the items, 10 at a time:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojo.data.ItemFileReadStore({url: "pantryStore.json" });
    var pageSize = 10;
    var request = null;
    var outOfItems = false;
    //Define a function that will be connected to a 'next' button
    var onNext = function(){
       if(!outOfItems){
           request.start += pageSize;
           store.fetch(request);
       }
    };
    //Connect this function to the onClick event of the 'next' button
    //Done through dojo.connect() generally.
    //Define a function will be connected to a 'previous' button.
    var onPrevious = function(){
       if (request.start > 0){
          request.start -= pageSize;
          store.fetch(request);
       }
    }
    //Connect this function to the onClick event of the 'previous' button
    //Done through dojo.connect() generally.
    //Define how we handle the items when we get it
    var itemsLoaded = function(items, request){
       if (items.length < pageSize){
          //We have found all the items and are at the end of our set.
          outOfItems = true;
       }else{
          outOfItems = false;
       }
       //Display the items in a table through the use of store.getValue() on the items and attributes desired.
       ...
    }
    //Do the initial request.  Without a query, it should just select all items.  The start and count limit the number
returned.
    request = store.fetch({onComplete: itemsLoaded, start: 0, count: pageSize});
```

Note: The previous sample shows how the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fetch()` function returns a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `request` object. According to the dojo.data specification, the request object contains the query parameters passed in plus an /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `abort()` function appended to it. In general, the abort function is intended for cases in which a request might take too much time to process or, in using the streaming callback of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fetch()`, a way to stop the streaming.

The request object also serves another purpose for datastores. It is a location where the store can cache hidden information about the request in process, such as a cache entry key for boosting performance through specifying exactly what internal cache might be in use for this particular query. Therefore, datastores can avoid calls to the server if possible. And, in the paging case, it becomes important to reuse the request object returned from /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fetch()`. Also note that not all stores will append cache information to the request, but some might. Therefore, when in doubt, reuse the request object when paging.

Sorting

Items are, in general, returned in an indeterminate order. This isn't always what you want to happen; there are definite cases where sorting items based on specific attributes is important. Fortunately, you do not have to do the sorting yourself because dojo.data provides a

mechanism to do it for you. The mechanism is just another option passed to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fetch`, the sort array.

The sort array will look something like the following example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var sortKeys = [{attribute: "aisle", descending: true}];
```

Each sort key has an attribute, which should be an attribute in the data store item, and a *descending* boolean flag. If an attribute passed isn't an actual attribute of the item, it will generally be ignored by the sorting; it is treated as an undefined comparison.

For compound sorts, you can define as many sort keys as you want. The order in the array defines which keys take priority over other keys when sorting. The following example shows this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var sortKeys = [
    {attribute: "aisle", descending: true},
    {attribute: "name", descending: false}
];
```

A Complete Example

For this example, we'll use the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` data source:

```
{   identifier: 'name',
    items: [
        { name: 'Adobo', aisle: 'Mexican' },
        { name: 'Balsamic vinegar', aisle: 'Condiments' },
        { name: 'Basil', aisle: 'Spices' },
        { name: 'Bay leaf', aisle: 'Spices' },
        { name: 'Beef Bouillon Granules', aisle: 'Soup' },
        ...
        { name: 'Vinegar', aisle: 'Condiments' },
        { name: 'White cooking wine', aisle: 'Condiments' },
        { name: 'Worcestershire Sauce', aisle: 'Condiments' }
        { name: 'pepper', aisle: 'Spices' }
    ]
}
```

Now, assume you want to sort the items in the store by aisle first, then by name. The following code snippet would do this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojo.data.ItemFileReadStore({url: "pantryStore.json"});
var sortKeys = [
    {attribute: "aisle", descending: true},
    {attribute: "name", descending: false}
];
store.fetch({
       sort: sortKeys;
       onComplete:
           ...
});
//When onComplete is called, the array of items passed into it
//should be sorted according to the denoted sort array.
```

Available Stores

The following dojo.data stores are pre-packaged with Dojo:

dojo.data.ItemFileReadStore	read-only store for JSON data
dojo.data.ItemFileWriteStore	read/write store for JSON data
dojox.data.CsvStore	read-only store for comma-separated variable (CSV) formatted data
dojox.data.OpmlStore	read-only store for Outline Processor Markup Language (OPML)
dojox.data.HtmlTableStore	read-only store for data kept in HTML-formatted tables

dojox.data.XmlStore	read/write store for basic XML data
dojox.data.FlickrStore	read store for queries on flickr.com, and a good example data store for web services
dojox.data.FlickrRestStore	read store for queries on flickr.com, and a good example data store for web services. More advanced version of FlickrStore
dojox.data.QueryReadStore	like ItemFileReadStore, read-only store for JSON data, but queries servers on each request

dojo.data.ItemFileReadStore

Summary:
Dojo core provides a basic implementation of a read-only datastore, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. This store reads the JSON structured contents from an http endpoint (service or URL), or from an in-memory JavaScript object, and stores all the items in-memory for simple and quick access. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` is designed to allow for flexibility in how it represents item hierarchy, references, and custom data types. It also provides options for which attribute can act as the unique identifier (for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Identity`), and which attribute can be used as a general label for an item. This store has an expectation that data is provided to in in a specific though very flexible, format. All of the examples on this page demonstrate the general format expected.

The following dojo.data APIs are implemented by ItemFileReadStore

- dojo.data.api.Read
- dojo.data.api.Identity

Format Examples: The following examples of data conform to the format the store requires for item input.

JSON with References: The following geography example uses references (items referencing another item declared in the data):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ 'identifier': 'name',
  'label': 'name',
  'items': [
     { 'name':'Africa', 'type':'continent',
         'children':[{'_reference':'Egypt'}, {'_reference':'Kenya'}, {'_reference':'Sudan'}] },
     { 'name':'Egypt', 'type':'country' },
     { 'name':'Kenya', 'type':'country',
         'children':[{'_reference':'Nairobi'}, {'_reference':'Mombasa'}] },
     { 'name':'Nairobi', 'type':'city' },
     { 'name':'Mombasa', 'type':'city' },
     { 'name':'Sudan', 'type':'country',
         'children':{'_reference':'Khartoum'} },
     { 'name':'Khartoum', type:'city' },
     { 'name':'Asia', 'type':'continent',
         'children':[{'_reference':'China'}, {'_reference':'India'}, {'_reference':'Russia'}, {'_reference':'Mongolia'}] },
     { 'name':'China', 'type':'country' },
     { 'name':'India', 'type':'country' },
     { 'name':'Russia', 'type':'country' },
     { 'name':'Mongolia', 'type':'country' },
     { 'name':'Australia', 'type':'continent', 'population':'21 million',
         'children':{'_reference':'Commonwealth of Australia'}},
     { 'name':'Commonwealth of Australia', 'type':'country', 'population':'21 million'},
     { 'name':'Europe', 'type':'continent',
         'children':[{'_reference':'Germany'}, {'_reference':'France'}, {'_reference':'Spain'}, {'_reference':'Italy'}] },
     { 'name':'Germany', 'type':'country' },
     { 'name':'France', 'type':'country' },
     { 'name':'Spain', 'type':'country' },
     { 'name':'Italy', 'type':'country' },
     { 'name':'North America', 'type':'continent',
         'children':[{'_reference':'Mexico'}, {'_reference':'Canada'}, {'_reference':'United States of America'}] },
     { 'name':'Mexico', 'type':'country', 'population':'108 million', 'area':'1,972,550 sq km',
         'children':[{'_reference':'Mexico City'}, {'_reference':'Guadalajara'}] },
     { 'name':'Mexico City', 'type':'city', 'population':'19 million', 'timezone':'-6 UTC'},
     { 'name':'Guadalajara', 'type':'city', 'population':'4 million', 'timezone':'-6 UTC' },
     { 'name':'Canada', 'type':'country',  'population':'33 million', 'area':'9,984,670 sq km',
         'children':[{'_reference':'Ottawa'}, {'_reference':'Toronto'}] },
     { 'name':'Ottawa', 'type':'city', 'population':'0.9 million', 'timezone':'-5 UTC'},
     { 'name':'Toronto', 'type':'city', 'population':'2.5 million', 'timezone':'-5 UTC' },
     { 'name':'United States of America', 'type':'country' },
     { 'name':'South America', 'type':'continent',
         'children':[{'_reference':'Brazil'}, {'_reference':'Argentina'}] },
     { 'name':'Brazil', 'type':'country', 'population':'186 million' },
```

```
    { 'name':'Argentina', 'type':'country', 'population':'40 million' }
]}
```

JSON with Hierarchy: The following geography example uses hierarchical items (items that contain definitions of other items):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ 'identifier': 'name',
  'items': [
    { 'name':'Africa', 'type':'continent', children:[
        { 'name':'Egypt', 'type':'country' },
        { 'name':'Kenya', 'type':'country', children:[
            { 'name':'Nairobi', 'type':'city' },
            { 'name':'Mombasa', 'type':'city' } ]
        },
        { 'name':'Sudan', 'type':'country', 'children':
            { 'name':'Khartoum', 'type':'city' }
        } ]
    },
    { 'name':'Asia', 'type':'continent', 'children':[
        { 'name':'China', 'type':'country' },
        { 'name':'India', 'type':'country' },
        { 'name':'Russia', 'type':'country' },
        { 'name':'Mongolia', 'type':'country' } ]
    },
    { 'name':'Australia', 'type':'continent', 'population':'21 million', 'children':
        { 'name':'Commonwealth of Australia', 'type':'country', 'population':'21 million'}
    },
    { 'name':'Europe', 'type':'continent', 'children':[
        { 'name':'Germany', 'type':'country' },
        { 'name':'France', 'type':'country' },
        { 'name':'Spain', 'type':'country' },
        { 'name':'Italy', 'type':'country' } ]
    },
    { 'name':'North America', 'type':'continent', 'children':[
        { 'name':'Mexico', 'type':'country',  'population':'108 million', 'area':'1,972,550 sq km', 'children':[
            { 'name':'Mexico City', 'type':'city', 'population':'19 million', 'timezone':'-6 UTC'},
            { 'name':'Guadalajara', 'type':'city', 'population':'4 million', 'timezone':'-6 UTC' } ]
        },
        { 'name':'Canada', 'type':'country', 'population':'33 million', 'area':'9,984,670 sq km', 'children':[
            { 'name':'Ottawa', 'type':'city', 'population':'0.9 million', 'timezone':'-5 UTC'},
            { 'name':'Toronto', 'type':'city', 'population':'2.5 million', 'timezone':'-5 UTC' }]
        },
        { 'name':'United States of America', 'type':'country' } ]
    },
    { 'name':'South America', 'type':'continent', children:[
        { 'name':'Brazil', 'type':'country', 'population':'186 million' },
        { 'name':'Argentina', 'type':'country', 'population':'40 million' } ]
    } ]
}
```

Custom Data Types: The following example uses custom data types (items with custom data types):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ 'identifier': 'abbr',
  'label': 'name',
  'items': [
    { 'abbr':'ec', 'name':'Ecuador',           'capital':'Quito' },
    { 'abbr':'eg', 'name':'Egypt',             'capital':'Cairo' },
    { 'abbr':'sv', 'name':'El Salvador',       'capital':'San Salvador' },
    { 'abbr':'gq', 'name':'Equatorial Guinea', 'capital':'Malabo' },
    { 'abbr':'er',
      'name':'Eritrea',
      'capital':'Asmara',
      'independence':{'_type':'Date', '_value':"1993-05-24T00:00:00Z"} // May 24, 1993 in ISO-8601 standard
      },
    { 'abbr':'ee',
      'name':'Estonia',
      'capital':'Tallinn',
      'independence':{'_type':'Date', '_value':"1991-08-20T00:00:00Z"} // August 20, 1991 in ISO-8601 standard
      },
    { 'abbr':'et',
      'name':'Ethiopia',
      'capital':'Addis Ababa' }
]}
```

Note: For custom data types, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileStore` looks for attributes that have an object format value and contains the following two specific attributes:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `_type`

> Attribute used to look up in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript

.kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `typeMap` which constructor should be used to instantiate the custom data type.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `_value`

Parameter that is the data to be passed to the constructor of the type. In this case, the custom type is a JavaScript Date object, and its value is the ISO-8601 string format of the date.

NOTE: The top level wrapper of the input for each of these examples is a JavaScript object. It has the following basic attributes that define the list of items that constitute the data and meta information about them:

identifier

Optional Metadata. The attribute in each item to act as the unique identifier for that item. This parameter is optional.

label

Optional Metadata. The attribute in each item to act as the human-readable label for that item. This parameter is optional.

items

An array of JavaScript objects that act as the items of the store. The attributes that are part of the items can be any valid JavaScript attribute name. Note that there is a special way for items to reference other items in the store when dealing with hierarchical data.

Constructor parameters

The constructor for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` takes the following possible parameters in its keyword arguments:

url

The URL from which to load the JSON data. This is optional. If it isn't specified, then you should specify the 'data' parameter to identify the in-memory javascript object that constitutes the basis for the store content.

data

The JavaScript object that represents the stores contents, as defined by the structure displayed in the examples. This is optional.

typeMap

A JavaScript associative map of data types to deserialize them. This is optional. See the Custom Data Type Mapping for more details.

Custom Data Type Mappings

Custom data types are a way to specify how the store should interpret a value of an attribute on an 'item' when it is parsing and loading the store from the ItemFile format. The ItemFileReadStore has one built in custom data type; the 'Date' object. It, by default, maps attribute values of {_type: "Date" _value: "some ISO string"} to a new Date instance instead of treating that JS object as a child item with attributes of '_type' and '_value'. The ItemFileReadStore also allows users to define their own custom types so that they can control how the information in the ItemFile format is interpreted. There are a couple of ways to map custom data types. There is a simple mapping method and general purpose mapping method that are described in the following sections.

Simple mapping: Direct Constructor

The direct constructor approach is the simplest way to map a type. This example assumes that the value stored in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `_value` can be used as the parameter to the constructor for an object:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var typeMap = {
                "Color": dojo.Color,
                ...
            };
```

General Purpose Mapping

General purpose mapping is intended for cases where you cannot directly pass the value of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript

.st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `_value` into a constructor. A good example of this is how /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `Date` is mapped internally in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. This is used because /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` reads in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `_value` for a date as a serialized ISO string, which is not a format the general JavaScript Date object understands.

The following example shows the Date serialization load from an ISOString format:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var typeMap = {
                    "Date": {
                               type: Date,
                               deserialize: function(value){
                                   return dojo.date.stamp.fromISOString(value);
                               }
                           }
               };
```

Query Syntax

The fetch method query syntax for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` is simple and straightforward. It allows a list of attributes to match against in an AND fashion. For example, a query object that looks like the following example will locate all items that have attributes of those names that match both of those values:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar", bit:"bite"}
```

Note that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` supports the use of wild cards (multi-character * and single character ?) in its attribute value matching.

Examples

To find all items with attribute *foo* that start with *bar*, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar*"}
```

To find all items with attribute *foo* the value of which ends with *ar* and ignoring only the first character, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"?ar"}
```

NOTE: Other stores **should** follow the same semantics in defining queries for consistency.

Usage Examples

For these examples, we'll assume a data source as defined by the example data format in this page unless otherwise specified.

Example 1: Query for all continents

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojo.data.ItemFileReadStore({url: "geography.json"});
var gotContinents = function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located continent: " + store.getLabel(item));
    }
}
var request = store.fetch({query: {type:"continent"}, onComplete: gotContinents});
```

Example 2: Query for names that start with A, case insensitively

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojo.data.ItemFileReadStore({url: "geography.json"});
var gotNames= function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located name that started with A: " + store.getLabel(item));
    }
}
var request = store.fetch({query: {name:"A*"}, queryOptions: {ignoreCase: true}, onComplete: gotNames});
```

Example 3: Programmatic creation of ItemFileReadStore without a URL. Then query for names that start with J, case insensitively.

IMPORTANT NOTE: *In this example the object 'dataItems' is modified and used directly by the store to generate the internal store representation of all the data. This is for efficiency of use and code size. If you want to repeatedly reuse the dataItems object you will need to clone the object and pass the clone into the store. If you do not clone the object and try to reuse dataItems, you will likely encounter errors due to the modifications the ItemFileReadStore applies to the data set so that it can perform store operations efficiently.* /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var dataItems = {
   identifier: 'name',
   label: 'name',
   items: [
      {name: 'John Smith'},
      {name: 'Bob Smith'},
      {name: 'Nancy Smith}',
      {name: 'John Doe'}
   ]
};
var store = new dojo.data.ItemFileReadStore({data: dataItems});
var gotNames= function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located name that started with J: " + store.getLabel(item));
    }
}
var request = store.fetch({query: {name:"J*"}, queryOptions: {ignoreCase: true}, onComplete: gotNames});
```

Further Examples

For further examples refer to the Using Datastores section of the Dojo book or refer to the test cases for dojo.data provided in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `tests` sub-directory of your dojo distribution.

dojo.data.ItemFileWriteStore

Dojo core provides the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileWriteStore` store as an extension to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript

.st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` that adds on the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Write` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Notification` API support to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. It was specifically created as a separate class so that if you only need read capability, you do not incur the download penalty of the write and notification API support if you won't use it. If your application needs to write to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileStore` instead of just Read, then /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileWriteStore` is the store you should instantiate. The input data format is identical to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`.

The following dojo.data APIs are implemented by ItemFileWriteStore

- dojo.data.api.Read
- dojo.data.api.Write
- dojo.data.api.Identity
- dojo.data.api.Notification

Constructor Parameters

The constructor for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileWriteStore` takes the same parameters as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` these are the following possible parameters in its keyword arguments:

url

The URL from which to load the JSON data. This is optional.

data

The JavaScript object that represents the stores contents, as defined by the previous structure. This is optional.

typeMap

A JavaScript associative map of data types to deserialize them. This is optional. See the Custom Data Type Mapping for more details.

Custom Data Type Mappings

The custom type mapping for the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileWriteStore` follows the same conventions as the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. The only caveat is, that for general purpose mappings, you must also provide a serialize function for mapping so the data can be rendered back out appropriately. For simple mapping, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `object.toString()` is sufficient.

Simple Mapping: Direct Constructor

The direct constructor approach is the simplest way to map a type. This one assumes that the value stored in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `_value` can be used as the parameter to the constructor for an object. When serializing this back to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileFormat`, this assumes that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `object.toString()` is sufficient for the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `_value` value as shown in the following example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var typeMap = {
                "Color": dojo.Color,
                ...
            };
```

General Purpose Mapping

The general purpose mapping is intended for cases in which you cannot directly pass the value of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `_value` into a constructor. A good example of this is how /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `Date` is mapped internally in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. This is used because /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` reads in the _value for a date as a serialized ISO string, which is not a format the general JavasScript /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `Date` object understands.

The following example shows date serialization and deserialization mapping:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var typeMap = {
                    "Date": {
                                type: Date,
                                deserialize: function(value){
                                    return dojo.date.stamp.fromISOString(value);
                                },
                                serialize: function(object){
                                    return dojo.date.stamp.toISOString(object);
                                }
                            }
            };
```

Query Syntax

The query syntax for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1

{color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileWriteStore` is identical to the query syntax of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore` so see that section for more information.

Usage Examples

For these examples, we'll assume a datasource as defined by the following example data:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ identifier: 'abbr',
  label: 'name',
  items: [
    { abbr:'ec', name:'Ecuador',          capital:'Quito' },
    { abbr:'eg', name:'Egypt',            capital:'Cairo' },
    { abbr:'sv', name:'El Salvador',      capital:'San Salvador' },
    { abbr:'gq', name:'Equatorial Guinea', capital:'Malabo' },
    { abbr:'er', name:'Eritrea',          capital:'Asmara' },
    { abbr:'ee', name:'Estonia',          capital:'Tallinn' },
    { abbr:'et', name:'Ethiopia',         capital:'Addis Ababa' }
]}
```

Example 1: Add in a new country

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojo.data.ItemFileWriteStore({url: "countries.json"});
var usa = store.newItem({abbr: 'us', name: 'United States of America', capital: 'Washington DC'});
function saveDone(){
    alert("Done saving.");
}
function saveFailed(){
    alert("Save failed.");
}
store.save({onComplete: saveDone, onError: saveFailed});
```

Example 2: Delete a country

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojo.data.ItemFileWriteStore({url: "countries.json"});
function saveDone(){
    alert("Done saving.");
}
function saveFailed(){
    alert("Save failed.");
}
var gotNames= function(items, request){
    for (var i = 0; i < items.length; i++){
       console.log("Deleted country: " + store.getLabel(item);
       store.deleteItem(items[i]);
    }
    store.save({onComplete: saveDone, onError: saveFailed});
}
var request = store.fetch({query: {name:"Egypt"}, queryOptions: {ignoreCase: true}, onComplete: gotNames});
```

Custom Saving

The save method by itself only updates the in-memory copy. To write the store back to the server, you need to override the extension point "_saveCustom". In markup language, it'd look something like:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojo.data.ItemFileWriteStore" jsId="mailStore"
        url="mail/mail.json">
        <script type="dojo/method" event="_saveCustom" args="saveCompleteCallback, saveFailedCallback">
                //  xhrPost/xhrPut your data back to your server (convert it to the server format first if need be)
                // 'this' keyword refers to the ItemFileWriteStore instance being enhanced
                // dojo/method replaces the _saveCustom method of the ItemFileWriteStore (currently undefined)
                saveCompleteCallback();
        </script>
</div>
```

You could also extend ItemFileWriteStore using Dojo's inheritance facilities:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.declare("CustomItemFileWriteStore", ItemFileWriteStore, {
        _saveCustom: function(saveCompleteCallback, saveFailedCallback){
                //  xhrPost/xhrPut your data back to your server (convert it to the server format first if need be)
                // 'this' keyword refers to the ItemFileWriteStore instance being extended
                saveCompleteCallback();
        }
});
```

Further examples

For further examples refer to the Using Datastores section of the Dojo book or refer to the test cases for dojo.data provided in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `tests` sub-directory of your dojo distribution.

dojox.data.CsvStore

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `CsvStore` is a read interface that works with CSV formated data files. The CSV file format is commonly known to folks who work regularly with spread sheet data. Like /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `CsvStore` reads the contents from an http endpoint or a JavaScript Data object that contains CSV formatted data. The following is an example of a CSV data source:

```
Title, Year, Producer
City of God, 2002, Katia Lund
Rain,, Christine Jeffs
2001: A Space Odyssey, , Stanley Kubrick
"This is a ""fake"" movie title", 1957, Sidney Lumet
Alien, 1979   , Ridley Scott
"The Sequel to ""Dances With Wolves.""", 1982, Ridley Scott
"Caine Mutiny, The", 1954, "Dymtryk ""the King"", Edward"
```

Note that in the above data, the first row is always assumed to be the column names. Those are what get assigned as the attribute names for the CSV items. Each row in the CSV data is treated as a single item.

The following dojo.data APIs are implemented by CsvStore

- dojo.data.api.Read
- dojo.data.api.Identity

Constructor Parameters

The constructor for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `CsvStore` takes three possible parameters in its keyword arguments as defined in the following list:

url

The URL from which to load the CSV data. This is optional.

data

The JavaScript object which represents the stores contents as defined by the structure in the previous example. This is optional.

label

A string that identifies which column to treat as the human-readable label. It must match one of the column labels in the file for it to be effective.

Query Syntax:

The fetch method query syntax for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `CsvStore` is simple and straightforward. It allows a list of attributes to match against in an AND fashion, just like /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. For example, a query object that looks like the following example will locate all items that have attributes of those names that match both of those values:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar", bit:"bite"}
```

Note that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `CsvStore` supports the use of wild cards (multi-character * and single character ?) in its attribute value matching.

Examples

To find all items with attribute *foo* that start with *bar*, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar*"}
```

To find all items with attribute *foo* the value of which ends with *ar* and ignoring only the first character, the query would be: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"?ar"}
```

NOTE: Other stores should follow the same semantics in how it defines its queries for consistency.

Usage Examples

Assume a data source as defined by the example data format in this page.

Example 1: Query for all movies by producer /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `Ridley Scott`

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojox.data.CsvStore({url: "movies.csv", label: "Title"});
var gotMovies = function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located movie: " + store.getLabel(item);
    }
}
var request = store.fetch({query: {Producer:"Ridley Scott"}, onComplete: gotMovies});
```

Example 2: Query for titles that that start with A, case insensitively

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;}

.geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var store = new dojo.data.ItemFileReadStore({url: "movies.csv", label: "Title"});
var gotTitles= function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located name that started with A: " + store.getLabel(item);
    }
}
var request = store.fetch({query: {Title:"A*"}, queryOptions: {ignoreCase: true}, onComplete: gotTitles});
```

Further Examples

For further examples refer to the test cases provided in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.data.tests`.

dojox.data.OpmlStore

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `OpmlStore` is a simple read-only store provided by Dojo and contained in the DojoX project. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `OpmlStore` is a read interface to work with Opml formatted XML files. Similar to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `OpmlStore` reads the contents from an http endpoint or a browser DOM object that contains Opml formatted data.

The following dojo.data APIs are implemented by OpmlStore

- dojo.data.api.Read
- dojo.data.api.Identity

The following example shows an Opml data source:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<opml version="1.0">
    <head>
        <title>geography.opml</title>
        <dateCreated>2006-11-10</dateCreated>
        <dateModified>2006-11-13</dateModified>
        <ownerName>Magellan, Ferdinand</ownerName>
    </head>
    <body>
        <outline text="Africa" type="continent">
            <outline text="Egypt" type="country"/>
            <outline text="Kenya" type="country">
                <outline text="Nairobi" type="city"/>
                <outline text="Mombasa" type="city"/>
            </outline>
            <outline text="Sudan" type="country">
                <outline text="Khartoum" type="city"/>
            </outline>
        </outline>
        <outline text="Asia" type="continent">
            <outline text="China" type="country"/>
            <outline text="India" type="country"/>
            <outline text="Russia" type="country"/>
            <outline text="Mongolia" type="country"/>
        </outline>
        <outline text="Australia" type="continent" population="21 million">
            <outline text="Australia" type="country" population="21 million"/>
        </outline>
        <outline text="Europe" type="continent">
            <outline text="Germany" type="country"/>
            <outline text="France" type="country"/>
            <outline text="Spain" type="country"/>
            <outline text="Italy" type="country"/>
        </outline>
        <outline text="North America" type="continent">
            <outline text="Mexico" type="country" population="108 million" area="1,972,550 sq km">
                <outline text="Mexico City" type="city" population="19 million" timezone="-6 UTC"/>
                <outline text="Guadalajara" type="city" population="4 million" timezone="-6 UTC"/>
            </outline>
```

```
            <outline text="Canada" type="country" population="33 million" area="9,984,670 sq km">
                <outline text="Ottawa" type="city" population="0.9 million" timezone="-5 UTC"/>
                <outline text="Toronto" type="city" population="2.5 million" timezone="-5 UTC"/>
            </outline>
            <outline text="United States of America" type="country"/>
        </outline>
        <outline text="South America" type="continent">
            <outline text="Brazil" type="country" population="186 million"/>
            <outline text="Argentina" type="country" population="40 million"/>
        </outline>
    </body>
</opml>
```

Note: An item in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `OpmlStore` is an /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<outline/>` entry. The atributes defined in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<outline/>` tag make up the attributes that are exposed for that item. Any child items (nested /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<outline/>` tags) are accessable using the special attribute /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `children`.

Constructor Parameters

The constructor for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `OpmlStore` takes the following possible parameters in its keyword args:

url

The URL from which to load the Opmldata. This is optional.

data

The raw Opml DOM that represents the stores contents as defined by the previous structure. This is optional.

label

A string that identifies which column to treat as the human-readable label. It must match one of the attribute names in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<outline/>` ags for it to be effective.

Query Syntax

The fetch method query syntax for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `OpmlStore` is simple and straightforward. It allows a list of attributes to match against in an AND fashion, just like /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. For example, the following query object locates all items that have attributes of those names that match both of those values: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar", bit:"bite"}
```

Note that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color:

#006600;} .javascript .re0 {color: #0066FF;} `OpmlStore` supports the use of wild cards (multi-character * and single character ?) in its attribute value matching.

Examples

To find all items with attribute /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `foo`/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `bar`, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar*"}
```

To find all items with attribute /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `foo` that value ends with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ar`, and ignoring only the first character, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"?ar"}
```

NOTE: Other stores **should** follow the same semantics in defining queries for consistency.

Usage Examples

For these examples, we'll assume a datasource as defined by the example data format in this page.

Example 1: Query for all continents starting with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} A

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojox.data.OpmlStore({url: "grography.xml", label: "Text"});
var gotContinents = function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located continent: " + store.getLabel(item);
    }
}
var request = store.fetch({query: {text: "A*", type: "continent"}, onComplete: gotContinents});
```

Example 2: Query for all continents starting with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} A, case insensitively

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;}

.geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var store = new dojox.data.OpmlStore({url: "grography.xml", label: "Text"});
var gotContinents = function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located continent: " + store.getLabel(item);
    }
}
var request = store.fetch({query: {text: "a*", type: "continent"}, queryOptions: {ignoreCase: true}, onComplete:
gotContinents});
```

Further Examples

For further examples refer to the test cases provided in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.data.tests`.

dojox.data.FlickrRestStore

FlickrRestStore is an implementation of the dojo.data API provides access to the Flickr photo sharing site's REST API. Many advanced features are available, including tag search, sorting on numerous attributes, full text search, support for simultaneous clients, result caching and more.

Dojo has several examples of browser in-memory stores, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.ItemFileReadStore`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.data.CsvStore`, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.data.OmplStore`. While these stores are useful and great examples of how data stores can be used to wrapper accessing data, they are not the only way data is served and processed. In many cases, data stores can wrapper external services. It is those services that perform the querying and filtering of data, and then provide only that as a subset back to the data store for presentation as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `items`.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`FlickrRestStore` is one such store. The purpose of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`FlickrRestStore` is to wrapper the public photo feed of the Flickr service. Then by simply using the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`FlickrRestStore` store, as you would any data store in Dojo, you now have access to querying the vast repository of public photos made available by others on the Web. Look at http://archive.dojotoolkit.org/nightly/dojotoolkit/dojox/data/demos/demo... to see it in action, or look here for some example usages.

The following dojo.data APIs are implemented by FlickrRestStore

- dojo.data.api.Read

FlickrRestStore is built upon FlickrStore, a store which reads from Flickrs more simplistic public API. However, FlickrRestStore provides many more features:

- **Build on Flickr REST API**. The Flickr REST API, a much more flexible API than the one used by FlickrStore. This opens up the possibility of having a very comprehensive data store that can have a very rich querying interface.
- **Results caching**. If a request is repeated, or a subset of one or more previous requests is requested, it is returned immediately.
- **Supports multiple simultaneous clients**. If an identical request is received before the first request has completed, a second remote request is not made. Instead, the second caller is notified when the first request completes. An example of this would be two widgets sharing the same FlickrRestStore, and individually paging through the photos.
- **Result sorting**. The available sort attributes are *date-taken, date-published* and *interestingness*. See the examples below for their usage.
- **Full text search**. The Flickr API supports full text searching, which you can do by passing a *text* parameter to the request query. See the examples below for its usage.
- **Filter by Photo Set**. By specifying a *setid* parameter on the query, you can retrieve photos only from a single set. Due to limitations of the Flickr API, using this parameter negates the use of the full text search.
- **Tag search**. Passing a *tags* parameter to the request query searches by the given tags. This parameter can be either a comma separated string, or an array of strings. See the examples below.

Flickr Terms and Conditions

Note: While this store wraps making calls to the Flickr service, as a user, you should still verify that you agree to the terms and conditions by which you are using the public Flickr photo service. Review their terms and conditions, and the API terms and conditions, here.

API Key

Another difference between FlickrRestStore and FlickrStore is that, due to the fact that FlickrRestStore works with the Flickr REST APIs, you will need to get a free API key from Flickr. You can do so at http://www.flickr.com/services/api/keys/apply/.

The Flickr service provides its data back in a wide variety of formats (for example, ATOM, RSS, and JSON) but /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}

.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`FlickrRestStore` only makes use of the JSON format. The following example shows a query that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`FlickrRestStore` processes and the response:

Example

Query the first three photos from a user:

URL: http://www.flickr.com/services/rest/?format=json&method=flickr.people.ge...

Response:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}

jsonFlickrApi({
"photos":{
"page":1,
"pages":98,
"perpage":3,
"total":"489",
"photo":[
{
"id":"1352049918",
"owner":"44153025@N00",
"secret":"5636009306",
"server":"1111",
"farm":2,
"title":"The Liffey Panorama",
"ispublic":1,
"isfriend":0,
"isfamily":0
},
{
"id":"1351120079",
"owner":"44153025@N00",
"secret":"880bf6a003",
"server":"1027",
"farm":2,
"title":"Many Hands make pretty flowers",
"ispublic":1,
"isfriend":0,
"isfamily":0
},
{
"id":"1322051485",
"owner":"44153025@N00",
"secret":"b7c529335d",

```
            "server":"1110",
            "farm":2,
            "title":"Wok'n'Roll baby!",
            "ispublic":1,
            "isfriend":0,
            "isfamily":0
         }
      ]
   },
   "stat":"ok"}
)
```

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
```

`FlickrRestStore`'s role is to process the query parameters passed to the

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
```

`dojo.data.api.Read` API and generate the appropriate service URL. It then processes the response from the service and handles accessing the items returned from the query. It also provides simple attribute access to all the values.

Constructor Parameters

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
```

`FlickrRestStore` does not have any constructor parameters.

Item Attributes

All items returned from

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
```

`FlickrRestStore` have the following attributes that can be accessed using the

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
```

.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`dojo.data.api.Read` API to retrieve data about the item:

title

The title of the photo.

author

The person who published the photo to Flickr.

imageUrl

A URL to the full resolution photo image.

imageUrlSmall

A URL to the small (icon sized) resolution photo image.

imageUrlMedium

A URL to the mid resolution photo image.

imageUrlThumb

A URL to the thumbnail sized resolution photo image.

link

A URL linking to the Flickr page displaying the image.

dateTaken

The date the photo was taken.

datePublished

The date the photo was published.

Query Syntax

The fetch method query syntax for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`FlickrRestStore` is simple and straightforward. It allows the following attributes to be set and queried against:

userid

A Flickr userid to use to narrow the search scope, e.g. '*44153025@N00*'. This is required.

apikey

A Flickr API key. Flickr requires clients of their REST APIs to register for an API key. This is free, and can be done at http://www.flickr.com/services/api/keys/apply/. **Note: do not reuse the API key used in Dojo examples, register your own.** This is required.

setid

The id of a photo set to use to narrow the result data. This is optional. If not specified, photos from the users primary stream are returned.

page

Specifies the page of results to use. If not used, then the standard /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}

.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`start` parameter is used. This is optional.

lang

Specifies the language to return the results in. This is optional.

tags

Specifies the tags to search for. This can be either a comma separated list, or an array of strings. This is optional.

text

The text to use in a full text search. This matches any text in the title or description of a photo. This is optional.

sort

The order to sort the results in. This is a JSON object with two fields, as specified by the dojo.data API.

- **attribute**: This specifies the name of the attribute to sort on. The supported attribute names are
  - date-posted
  - date-taken
  - interestingness

  If an attribute is not specified, the default is *date-posted*

- **descending**: If set to *true*, the photos are sorted in descending order. If set to false, or not specified, the photos are sorted in ascending order.

. This is optional.

Note: Unlike many of the other example stores, the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`FlickrRestStore` store cannot do wild-card matching of the attributes. This is because the Flickr public photo feed service cannot do it. In an ideal service implementation, the Flickr service would provide a mechanism by with to pass in wild cards as part of its query parameters.

Usage Examples

The following example shows how you would query for the first ten images belonging to a single user, then emit the title, author, and image URL to the console:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}

```
var flickrRestStore = new dojox.data.FlickrRestStore();

function onBegin(totalCount, requestObj){
   console.log("TOTAL IMAGE COUNT:" + totalCount);
}
function onItem(item, requestObj){
   console.log("TITLE: " + flickrStore.getValue(item, "title");
   console.log("AUTHOR: " + flickrStore.getValue(item, "author");
   console.log("LINK: " + flickrStore.getValue(item, "link");
   console.log("IMAGE URL: " + flickrStore.getValue(item, "imageUrl");
   console.log("IMAGE URL Small: " + flickrStore.getValue(item, "imageUrlSmall");
   console.log("IMAGE URL Medium: " + flickrStore.getValue(item, "imageUrlMedium");
   console.log("IMAGE URL Thumbnail: " + flickrStore.getValue(item, "imageUrlThumb");
   console.log("DATE TAKEN: " + flickrStore.getValue(item, "dateTaken");
   console.log("DATE PUBLISHED: " + flickrStore.getValue(item, "datePublished");
```

```
}
function onComplete(items, request){
   console.log("DONE!")
}
function onError(error, request){
   console.log("FAILED!")
}

var request = {
   query: {
     userid: "44153025@N00",
     apikey: "8c6803164dbc395fb7131c9d54843627"
   },

   onBegin: onBegin,
   onItem: onItem,
   onComplete: onComplete,
   onError: onError
 };

//Get ten photos from user "44153025@N00".
request.start = 0;     //start at the beginning
request.count = 10;  //Retrieve ten images
flickrRestStore.fetch(request);

//Get ten photos from user "44153025@N00",
//matching the tags volleyball or dublin.
request.start = 0;     //start at the beginning
request.count = 10;  //Retrieve ten images
request.query.tags = ["volleyball", "dublin"];

//This causes the search to match
//"volleyball" OR "dublin".  To do an AND query,
//use request.query.tagmode = "all"
request.query.tagmode = "any";
flickrRestStore.fetch(request);

//Get fifty photos from user "44153025@N00",
//sorting descending on interestingness.
request.start = 0;     //start at the beginning
request.count = 50;  //Retrieve fifty images
request.query.tags = null; //delete the tags from the previous request

//The sort parameter is an array, as the
//dojo.data API specifies that a READ store should
//support multiple sort attributes.
//However, the Flickr API only supports a single
//sort parameter.
request.query.sort = [
  {
     //'attribute' could also be "date-taken" or "date-posted"
     attribute: "interestingness",
     descending: true
  }
];
flickrRestStore.fetch(request);

//Get the second page of twenty photos from a given set
request.start = 20;
request.count = 20;
request.query.setid = "72157600959797470";
request.query.sort = null; //Clean up after the last request
flickrRestStore.fetch(request);

//Perform a full text search, retrieving 50 photos
request.start = 0;
request.count = 50;
//clean up after the previous request
request.query.setid = null;
//Finds all photos with "kinsale" in the title or description
request.query.text = "kinsale";
flickrRestStore.fetch(request);
```

Further Examples

For further examples refer to the test cases provided in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}

.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`dojox/data/tests/stores/FlickrRestStore.js`.

dojox.data.HtmlTableStore

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `HtmlTableStore` is a simple read-only store provided by Dojo and contained in the DojoX project. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `HtmlTableStore` is a read interface to work with HTML tables with a generally set format. HTML tables are common ways for Web data to be displayed and they can be extremely useful as an alternate representation of data that is displayed in a charting or gauge widget. This store was created so that widgets, that can use dojo.data data stores, can read their input from HTML table data islands in the current page or in a remote page URL. This store implements both /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Read` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Identity`.

The following example shows an HTML table that this store can read:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<html>
<head>
    <title>Books2.html</title>
</head>
<body>
<table id="books2">
    <thead>
        <tr>
            <th>isbn</th>
            <th>title</th>
            <th>author</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>A9B57C</td>
            <td>Title of 1</td>
            <td>Author of 1</td>
        </tr>
        <tr>
            <td>A9B57F</td>
            <td>Title of 2</td>
            <td>Author of 2</td>
        </tr>
        <tr>
            <td>A9B577</td>
            <td>Title of 3</td>
            <td>Author of 3</td>
        </tr>
        <tr>
            <td>A9B574</td>
            <td>Title of 4</td>
            <td>Author of 4</td>
        </tr>
        <tr>
            <td>A9B5CC</td>
            <td>Title of 5</td>
            <td>Author of 5</td>
        </tr>
    </tbody>
</table>
</body>
</html>
```

Note that the table rows in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<tbody>` tag are the items. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript

.coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<thead>` tag is used for defining the attribute name for each column in the table row for an item.

Constructor Parameters

The constructor for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `HtmlTableStore` takes the following possible parameters in its keyword arguments:

url

The URL from which to load the HTML file containing the HTML table. This is optional.

tableId

The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `id` of the HTML tag that contains the table to read from, in either a remote page (if the URL was passed) or in the current HTML DOM if the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `url` parameter is null. This is required.

Query Syntax

The fetch method query syntax for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `HtmlTableStore` is simple and straightforward. It allows for a list of attributes to match against in an AND fashion, just like /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. For example, the following query object will locate all items that have attributes of those names that match both values:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar", bit:"bite"}
```

Note that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `HtmlTableStore` supports the use of wild cards (multi-character * and single character ?) in its attribute value matching.

Examples

To find all items with attribute /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `foo` that start with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `bar`, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar*"}
```

To find all items with attribute /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript

.me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `foo` that value ends with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ar` and ignoring only the first character, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
{ foo:"?ar"}
```

NOTE: Other stores **should** follow the same semantics in defining queries for consistency.

Usage Examples

For these examples, we'll assume a data source as defined by the example data format in this page.

Example 1: Query for all books that start with ISBN: A9B57

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var store = new dojox.data.HtmlTableStore({tableId: "books2"});
var gotBooks = function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located book: " + store.getValue(item, "title"));
    }
}
var request = store.fetch({query: {isbn:"A9B57*"}, onComplete: gotBooks});
```

Example 2: Query for all books that start with ISBN: A9B57 Case insensitively

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var store = new dojox.data.HtmlTableStore({tableId: "books2"});
var gotBooks = function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located book: " + store.getValue(item, "title");
    }
}
var request = store.fetch({query: {isbn:"a9b57*"}, queryOptions: {ignoreCase: true}, onComplete: gotBooks});
```

Further Examples

For further examples, refer to the test cases provided in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

`dojox.data.tests`

.

dojox.data.XmlStore

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `XmlStore` is a store provided by Dojo that is contained in the DojoX project. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `XmlStore` is a read and write interface to basic XML data. XML is a common data interchange format and a store that can work with fairly generic XML documents is useful. The store is designed so that you can over-ride certain functions to get specific behaviors to occur when performing reads and saves.

The following dojo.data APIs are implemented by XmlStore

- dojo.data.api.Read
- dojo.data.api.Write

The following is an example of an XML document that this store can read:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<books>
    <book>
        <isbn>A9B57C</isbn>
        <title>Title of 1</title>
        <author>Author of 1</author>
    </book>
    <book>
        <isbn>A9B57F</isbn>
        <title>Title of 2</title>
        <author>Author of 2</author>
    </book>
    <book>
        <isbn>A9B577</isbn>
        <title>Title of 3</title>
        <author>Author of 3</author>
    </book>
    <book>
        <isbn>A9B574</isbn>
        <title>Title of 4</title>
        <author>Author of 4</author>
    </book>
    <book>
        <isbn>A9B5CC</isbn>
        <title>Title of 5</title>
        <author>Author of 5</author>
    </book>
</books>
```

Constructor Parameters

The constructor for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `XmlStore` takes the following possible parameters in its keyword arguments:

url

The URL from which to load the XML file containing the data. This URL is also the end point used for posting data base in a save. This is optional.

sendQuery

Boolean option whether or not to send the query to a server for processing. The default is false.

false

It is assumed the server sends back the entire store dataset and the filerting and sorting must occur on the client side.

true

It is assumed the server is handling the filtering and is only sending back XML nodes that match the query. No filtering occurs clientside.

rootItem

A tag name for root items. This is optional. If it is not provided, then the XmlStore assumes the tags under the root element of the document are the root items.

keyAttribute

An attribute name for a key or an indentity. This is optional.

attributeMap

An anonymous object that contains properties for attribute mapping, for example /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `{"tag_name.item_attribute_name": "@xml_attribute_name", ...}` This is optional. This is done so that attributes which are actual XML tag attributes (and not sub-tags of an XML tag), can be referenced.

label

The attribute of an item to use for the return of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `getLabel()`. This is optional.

Functions intended to be over-ridden to alter save behavior

The following functions can be over-ridden to alter save behavior, as described:

_getPostUrl(item)

Function that can be over-ridden to alter the way the store POSTs new items to the service. Note that this follows the REST convention in which an HTTP POST is a creation of a new resource.

_getPutUrl(item)

Function that can be over-ridden to alter the way the store PUTs updated items to the service. Note that this follows the REST convention in which an HTTP PUT is an update of an existing resource.

_getDeleteUrl(item)

Function that can be over-ridden to alter the way the store sends a DELETE item to the service. Note that this follows the REST convention in which an HTTP DELETE is used to remove a resource.

Query Syntax

The fetch method query syntax for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `XmlStore` is simple and straightforward. It allows for a list of attributes to match against in an AND fashion, just like /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ItemFileReadStore`. For example, the following query object will locate all items that have attributes of those names that match both of those values:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar", bit:"bite"}
```

Note that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `XmlStore` supports the use of wild cards (multi-character * and single character ?) in its attribute value matching.

Examples

To find all items with attribute /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `foo` that start with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `bar`, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"bar*"}
```

To find all items with attribute /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `foo` the value of which ends with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ar` and ignoring only the first character, the query would be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{ foo:"?ar"}
```

NOTE: Other stores **should** follow the same query definition semantics for consistency.

Usage Examples

For these examples, we'll assume a data source as defined by the example data format in this page.

Example 1: Query for all books that start with ISBN: A9B57

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojox.data.XmlStore({url: "books.xml", rootItem: "book"});
var gotBooks = function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located book: " + store.getValue(item, "title");
    }
}
var request = store.fetch({query: {isbn:"A9B57*"}, onComplete: gotBooks});
```

Example 2: Query for all books that start with ISBN: A9B57 Case insensitively

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var store = new dojox.data.XmlStore({url: "books.xml", rootItem: "book"});
var gotBooks = function(items, request){
    for (var i = 0; i < items.length; i++){
       var item = items[i];
       console.log("Located book: " + store.getValue(item, "title");
    }
}
var request = store.fetch({query: {isbn:"a9b57*"}, queryOptions: {ignoreCase: true}, onComplete: gotBooks});
```

Further Examples

For further examples see the test cases provided in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.data.tests`.

dojox.data.FlickrStore

Dojo has several examples of browser in-memory stores, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.ItemFileReadStore`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.data.CsvStore`, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.data.OmplStore`. While these stores are useful and great examples of how data stores can be used to wrapper accessing data, they are not the only way data is served and processed. In many cases, data stores can wrapper external services. It is those services that perform the querying and filtering of data, and then provide only that as a subset back to the data store for presentation as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `items`.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` is one such store. The purpose of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` is to wrapper the public photo feed of the Flickr service. Then by simply using the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript

.st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` store, as you would any data store in Dojo, you now have access to querying the vast repository of public photos made available by others on the Web.

Note: While this store wraps making calls to the Flickr service, as a user, you should still verify that you agree to the terms and conditions by which you are using the public flickr photo service. Review their terms and conditions, and the API terms and conditions.

The Flickr service provides its data back in a wide variety of formats (for example, ATOM, RSS, and JSON) but /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` only makes use of the JSON format. The following example shows a query that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` processes and the response:

Query (all pictures with tags /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `animals`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `foxes`, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `cute`):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
http://api.flickr.com/services/feeds/photos_public.gne?tags=animals,bats,cute&format=json&tagmode=all
```

Response:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
jsonFlickrFeed({
                "title": "Photos from everyone tagged animals, foxes and cute",
                "link": "http://www.flickr.com/photos/",
                "description": "",
                "modified": "2007-05-24T09:35:27Z",
                "generator": "http://www.flickr.com/",
                "items": [
          {
                        "title": "Ceramic Figures",
                        "link": "http://www.flickr.com/photos/36362445@N00/511998141/",
                        "media": {"m":"http://farm1.static.flickr.com/228/511998141_7b8398c3eb_m.jpg"},
                        "date_taken": "2006-04-04T10:21:43-08:00",
                        "description": "<a href="http://www.flickr.com/people/36362445@N00/">travellingcharl</a> posted a
photo: <a href="http://www.flickr.com/photos/36362445@N00/511998141/" title="Ceramic Figures"><img
src="http://farm1.static.flickr.com/228/511998141_7b8398c3eb_m.jpg" width="240" height="180" alt="Ceramic Figures" /></a>
Ceramic figurines inside the Shinto shrine.",
                        "published": "2007-05-24T09:35:27Z",
                        "author": "nobody@flickr.com (travellingcharl)",
                        "tags": "cute animals japan ceramic geocaching coins toyko foxes naritatbstation3"
          },
          {
                        "title": "Red Fox pup",
                        "link": "http://www.flickr.com/photos/norwick/301289990/",
                        "media": {"m":"http://farm1.static.flickr.com/104/301289990_da7413890b_m.jpg"},
                        "date_taken": "2005-06-09T16:17:49-08:00",
                        "description": "<a href="http://www.flickr.com/people/norwick/">bluebird's</a> posted a photo: <a
href="http://www.flickr.com/photos/norwick/301289990/" title="Red Fox pup"><img
src="http://farm1.static.flickr.com/104/301289990_da7413890b_m.jpg" width="240" height="180" alt="Red Fox pup" /></a> Are you
sure it's save to come out!",
                        "published": "2006-11-19T22:14:47Z",
                        "author": "nobody@flickr.com (bluebird's)",
                        "tags": "wild summer canada cute nature beauty field animals landscape tiere scenery jung wilde sommer
wildlife natur young feld felder adorable peaceful canadian alberta summertime prairie aussicht prairies landschaft foxes alert
tier countrylife predators kanada redfox perky okotoks junger rotfuchs kanadische foxpups roterfuchs"
          }
        ]
})
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore`'s role is to process the query parameters passed to the /* GeSHi (C) 2004 - 2007 Nigel McNie

(http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Read` API and generate the appropriate service URL. It then processes the response from the service and handles accessing the items returned from the query. It also provides simple attribute access to all the values.

Constructor Parameters

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` does not have any constructor parameters.

Item Attributes

All items returned from /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` have the following attributes that can be accessed using the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.data.api.Read` API to retrieve data about the item:

title

The title of the photo.

author

The person who published the photo to Flickr.

description

A description of the photo. This will generally contain HTML formatted text.

dateTaken

A JavaScript date object representing the date the photo was taken.

datePublished

A JavaScript date object representing the date the photo was published to Flickr.

tags

The tags that are assigned to this photo. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `flickrStore.getValue(item, "tags")` returns the first tag, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `flickrStore.getValues(item, "tags")` returns all tags.

imageUrl

A URL to the full resolution photo image.

imageUrlSmall

A URL to the small (icon sized) resolution photo image.

imageUrlMedium

A URL to the mid resolution photo image.

link

A URL linking to the Flickr page displaying the image.

Query Syntax

The fetch method query syntax for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` is simple and straightforward. It allows the following attributes to be set and queried against:

id

A Flickr userid to use to narrow the search scope. This is optional.

ids

A comma separated list of IDs used to narrow search scope. This is optional.

tags

A comma separated list of tags to search for matches on. This is optional.

tagmode

Indicates whether all tags must match from the list or any can match from the list. Valid values are /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `all` or /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `any` and the default is /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `any`.

lang

Specifies the language to return the results in. This is optional.

Note: Unlike all the other example stores, the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `FlickrStore` store cannot do wild-card matching of the attributes. This is because the Flickr public photo feed service cannot do it. In an ideal service implementation, the Flickr service would provide a mechanism by with to pass in wild cards as part of its query parameters. Also, the Flickr public feed API limits the number of returned photos to a maximum of twenty.

Usage Examples

The following example shows how you would query for all images with the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `3dny` tag, then emit the title, author, and image URL to the console:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var flickrStore = new FlickrStore();
                function onItem(item, requestObj){
                    console.log("TITLE: " + flickrStore(item, "title");
                    console.log("AUTHOR: " + flickrStore(item, "author");
                    console.log("IMAGE URL: " + flickrStore(item, "imageUrl");
                }
                function onComplete(items, request){
                    console.log("DONE!")
                }
                function onError(error, request){
                    console.log("FAILED!")
                }
                //Get the photos tagged 3dny...
                flickrStore.fetch({
                                    query: {tags:"3dny"},
                                    onItem: onItem,
                                    onComplete: onComplete,
                                    onError: onError
                                });
```

Further Examples

For further examples refer to the test cases provided in `dojox/data/tests/stores/FlickrStore.js<.code>. In addition, there are demos using the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} FlickrStore store available in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} dojox/data/demos/.`

dojox.data.QueryReadStore (1.0)

New in 1.0, QueryReadStore is very similar to ItemReadStore. They both use JSON as their exchange format. The difference is in the way they query data. ItemReadStore makes one fetch from the server, and handles all sorting and filtering in the client. That's fine for hundreds of records, even thousands. But for hundreds of thousands of records or slow Internet connections, that's less feasible.

QueryReadStore makes a request to the server for each sorting or query. This makes it ideal for large datasets with small windows of data, as in dijit.FilteringSelect.

Query Translation

A dojo.data request follows a specific JSON format. As an example, suppose we have a FilteringSelect which looks up states. When the user presses "A", the dojo.data request is:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{
   query: {name: "A*"},
   queryOptions: {ignoreCase: true},
   sort: [{attribute:"name", descending:false}],
   start: 0,
   count: 10
}
```

Now we want to hand this off to the server. Odds are, your server doesn't recognize incoming JSON, and asking it to do so is too restrictive. Instead, most server queries follow a REST pattern like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
states.php?q=A*
```

Fortunately, it's easy to translate between the two. You simply subclass QueryReadStore like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.provide("custom.ComboBoxReadStore");
dojo.declare("custom.ComboBoxReadStore", dojox.data.QueryReadStore, {
    fetch:function(request) {
        request.serverQuery = {q:request.query.name};
        // Call superclasses' fetch
        return this.inherited("fetch", arguments);
    }
});
```

We can place this file into a folder "custom" at the same level as the dojo, dijit and dojox directories of the distribution. (See Creating Your Own Modules for a discussion and alternatives.

You can download QueryReadStore.php below (it's also in /dojox/data/tests/stores/QueryReadStore.php) to run this example on a PHP server. The server portion hands over a portion of the states that fits the query. The full client program:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>QueryReadStore Demo</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"
            djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dojox.data.QueryReadStore");
       dojo.require("dijit.form.FilteringSelect");
       dojo.require("custom.ComboBoxReadStore");
    </script>
</head>
<body class="tundra">
    <div dojoType="ComboBoxReadStore" jsId="store"
         url="QueryReadStore.php"
         requestMethod="get">
    </div>
    State: <input id="fs" dojoType="dijit.form.FilteringSelect" store="store" pageSize="5" />
</body></html>
```

Selecting DOM Nodes with dojo.query

XHR is half of the Web 2.0 story. Once you make a request for data and receive it via XHR, you must change the page - display the new data in a panel, turn an indicator from red to green, or whatever . Changing HTML is, in turn, dependent on locating nodes.

To select HTML elements in JavaScript, you can use the DOM API. If your request is simple, the DOM API is pretty straightforward. For example, in like "get me a list of all SELECT boxes", we could use getElementsByTag(). But the DOM API is very low level, and awkward to use for more sophisticated queries. For example, retrieving all nodes with the class "progressIndicator" uses this code:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var progressIndicatorList = [];
function getProgressIndicators(n, progressIndicatorList) {
   if(n.className !== undefined){
      progressIndicatorList.push(n);  // Add n to result list
   }
   // Call the function recursively for all children
   for(var m=n.firstChild; m != null; m = m.nextSibling){
      getProgressIndicators(m, progressIndicatorList);
   }
}
// Print list of all progress Indicators
console.dir(getProgressIndicators(document, progressIndicatorList);
```

Oy! That's a lot of code. It's too bad this weren't a style sheet, and we could just say:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// Not legal!  But we can dream...
console.dir( <STYLE>{
   .progressIndicator
}</STYLE>);
```

Amazingly, dojo.query allows you to do just that. It uses CSS selectors to return node lists which you can use for traversing, manipulating, and gathering data. The following is exactly equivalent to our first example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
console.dir(dojo.query(".progressIndicator"));
```

Wow! The use of CSS means you don't have to learn a whole new query language. To top it all off, it's fast too - see the original article on dojo.query for details.

Selecting Nodes by Tag, ID or Class

The most popular methods for selecting nodes are by tag name, or the class and id attributes of the tag. Those are the most popular selectors for CSS files too. So, as you'd expect, dojo.query makes these easy:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// Query by tag.  Equivalent to DOM document.getElementsByTagName("IMG");
console.dir(dojo.query("img"));   // Non-case sensitive, so IMG would work too.
// Query by class.  Equivalent to big loop on previous page
console.dir(dojo.query(".offToSeeTheWij"));
// Query by id.  Equivalent to DOM document.getElementById("widget123");
// or dojo.byId("widget123")
console.dir(dojo.query("#widget123"));
```

As with CSS, you can combine selectors in dojo.query to create Compound Selectors. By smooshing the selectors "a" and "b" against each other, you say "select nodes that have both properties a and b", as in: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// Select just image tags with the class offToSeeTheWij
console.dir(dojo.query("img.offToSeeTheWij"));
```

Things to Make and Do With Queries

dojo.query, like CSS3, can do very complex selecting. But first ... what can you do with the retrieved nodes?

dojo.query returns a standard DOM NodeList, which can be traversed with regular JavaScript array techniques. It has a .length property like

any array, and you can loop through it with a for loop. But Dojo provides some more convenient methods:

Applying a Dojo Function to Each Element

Since dojo.query returns a NodeList object, you can feed the results into any NodeList method. Some popular Dojo methods can be tacked right onto a dojo.query. Take for example **style()** which applies a style to each element queried. The following turns the background color of each element in class disableAble to gray:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.query(".disableAble").style("backgroundColor","gray");
```

Here is a complete list of Dojo methods that can be used in this manner.

- indexOf()
- lastIndexOf()
- coords()
- style()
- styles()
- addClass()
- removeClass()
- place()
- connect()
- orphan()
- adopt()
- addContent()

For Dojo 1.0: if you do a **dojo.require("dojo.NodeList-fx")**, the following methods will be added to the NodeList object (see API documentation for more information):

- wipeIn()
- wipeOut()
- slideTo()
- fadeIn()
- fadeOut()
- animateProperty()

More General Functions - forEach

*New for 1.0***forEach()** applies a snippet of JavaScript to each node, as described in Functions Used Everywhere. This snippet can use the following "magic" variables:

- **item** is the DOM node currently being examined
- **index** is its position in the NodeList, starting from 0
- **arr** is the entire array of nodes. You can use this to perform lookahead or look-behind. arr[index] === item is always true.

The following code disables all INPUT tags

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.query("input").forEach("item.disabled = true;");
```

This call style is supported by forEach(), map(), every(), and some(). dojo.filter() also supports it but the NodeList filter() method does not since it accepts a CSS string expression to filter by instead.

The above only works for Dojo 1.0, but in either 0.9 or 1.0 you can pass a function name or a function literal. The function must take at least one parameter, the first of which is the element being examined. (The rest of the parameters are ignored). So the following code is equivalent to the first example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.query("input").forEach(
    function(inputElement) {
        inputElement.disabled = true;
   }
);
```

Filter

filter() uses a JavaScript function to pare down a list of nodes. The function provided to filter must take a node as input (similar to forEach) and return true if the node is to be kept. It returns a nodeList, just like query. This is especially useful for element selections you cannot express in CSS selector language. For example, odd/even row coloring can be performed this way:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:

bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var nodeCount = 0;
dojo.query("tr").filter(
   function(thisRow) {
      return (nodeCount++) % 2 == 0;
   }
).style("backgroundColor",evenRowColor);
```

Note that, as we said before, NodelList filter cannot use the new 1.0 snippet syntax. You must pass in a function literal or function name.

Event Handlers

You can attach **event handlers** to all the nodes of a query. For example, to bind an onclick handler to all elements with the "deadLink" class:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.query(".deadLink").onclick(function(evt){
    dojo.stopEvent(evt);
});
```

NodeList methods are chainable:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.query(".deadLink").addClass("disabledLink").onclick(function(evt){
    dojo.stopEvent(evt);
});
```

The following are event handler binders available on a NodeList object (the object returned from a dojo.query() call -- see dojo.NodeList API documentation for more information):

- onblur()
- onclick()
- onkeydown()
- onkeypress()
- onkeyup()
- onmousedown()
- onmouseenter()
- onmouseleave()
- onmousemove()
- onmouseout()
- onmouseover()
- onmouseup()

'

Relative Position Query

An optional second parameter to dojo.query indicates the root node for searching. If a string is passed, dojo.query() assumes it to be the ID of the element to use as the search root, otherwise it will expect a DOM node. In all of our examples so far, the second parameter has been left out, and defaults to document, the root of the HTML page. (Actually, it's dojo.doc, which is "document", but can be modified with dojo.withDocument).

The following shows how the second parameter affects the result:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<html>
<head>
 <script type="text/javascript" src="../js/dojo/dojo.js"></script>
 <script type="text/javascript">
     dojo.addOnLoad(function() {
         console.debug(dojo.query("button").length);  // Outputs "3"
         console.debug(dojo.query("button", "thisForm").length);  // outputs 1
    });
</script>
</head>
<body>
   <button id="b1" />
   <button id="b2" />
   <form id="thisForm" >
      <button id="formB" />
   </form>
</body>
</html>
```

Standard CSS3 Selectors

Because dojo.query adopts the CSS3 standard for selecting nodes, you can use any CSS reference guide for help on choosing the right queries. Eric Meyer's *CSS: The Definitive Guide* is a good resource. For convenience, here's a chart of the standard CSS3 selectors, taken from the current working draft RFC.

Pattern	Meaning
*	any element
E	an element of type E
E[foo]	an E element with a "foo" attribute
E[foo="bar"]	an E element whose "foo" attribute value is exactly equal to "bar"
E[foo~="bar"]	an E element whose "foo" attribute value is a list of space-separated values, one of which is exactly equal to "bar"
E[foo^="bar"]	an E element whose "foo" attribute value begins exactly with the string "bar"
E[foo$="bar"]	an E element whose "foo" attribute value ends exactly with the string "bar"
E[foo*="bar"]	an E element whose "foo" attribute value contains the substring "bar"
E[hreflang|="en"]	an E element whose "hreflang" attribute has a hyphen-separated list of values beginning (from the left) with "en"
E:root	an E element, root of the document
E:nth-child(n)	an E element, the n-th child of its parent
E:nth-last-child(n)	an E element, the n-th child of its parent, counting from the last one
E:nth-of-type(n)	an E element, the n-th sibling of its type
E:nth-last-of-type(n)	an E element, the n-th sibling of its type, counting from the last one
E:first-child	an E element, first child of its parent
E:last-child	an E element, last child of its parent
E:first-of-type	an E element, first sibling of its type
E:last-of-type	an E element, last sibling of its type
E:only-child	an E element, only child of its parent
E:only-of-type	an E element, only sibling of its type
E:empty	an E element that has no children (including text nodes)
E:link E:visited	an E element being the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)
E:active E:hover E:focus	an E element during certain user actions
E:target	an E element being the target of the referring URI
E:lang(fr)	an element of type E in language "fr" (the document language specifies how language is determined)
E:enabled E:disabled	a user interface element E which is enabled or disabled
E:checked	a user interface element E which is checked (for instance a radio-button or checkbox)
E::first-line	the first formatted line of an E element
E::first-letter	the first formatted letter of an E element
E::selection	the portion of an E element that is currently selected/highlighted by the user
E::before	generated content before an E element
E::after	generated content after an E element
E.warning	an E element whose class is "warning" (the document language specifies how class is determined).
E#myid	an E element with ID equal to "myid".
E:not(s)	an E element that does not match simple selector s
E F	an F element descendant of an E element
E > F	an F element child of an E element
E + F	an F element immediately preceded by an E element
E ~ F	an F element preceded by an E element

Internationalization (i18n)

Internationalization, or i18n, is the process of making an application flexible to work in different languages and respect different conventions and customs. An identifier called a *locale* represents a language code with an optional list of variants, which may include geographic or other information. Based on the locale, different data and logic may be applied when rendering text or dealing with numbers or dates. This section will cover the basic concepts of internationalization and show how these features are implemented in Dojo.

Dojo aims to make internationalization a basic feature of the toolkit, such that the presentation to the user never assumes one particular language or culture. Developer-facing resources, such as console output and API documentation do not yet have an internationalization framework.

Localization in the form of translated resources, mostly in the Dijit project, are generally available starting with the 1.0 release in over a dozen languages, with more to come.

Globalization Guidelines

Overview

Dojo addressed the globalization features at the very beginning of its development. This document presents the rules and describes how to use these features to globalize your Web applications based on Dojo version 1.0. Each of the rules use one of the following directive words:

* **Must:** You must always follow the rules; otherwise your application cannot be globalized.
* **Should:** You are recommended to follow the rules in some situations. Your application can be globalized if you do not follow these rules, but more effort might be needed.
* **May:** These rules do not affect the capability of globalization. You can choose whether to follow them or not.

Use the following guidelines to implement internationalization.

Encoding guidelines

You should always use UTF-8 for encoding settings wherever applicable.
You should encode all text files in UTF-8.
You must specify the UTF-8 encoding in every HTML file before any non-English characters.
You must use the BOM header for UTF-16 files.
You must use UTF-8 to decode XHR request parameters.
You must use UTF-8 encoding when using a non-English string in a URL.
You must set Content-Type in an HTTP response header if the response is not encoded in UTF-8.

Locale and Resource Bundle Guidelines

You must set Content-Type in an HTTP response header if the response is not encoded in UTF-8.
You must set djConfig.locale in all files to the same as the locale used by the server code.
You must always use resource bundle to store the strings displayed to users.
You should use djConfig.locale to set the default locale and extra locales, and use only dojo.requireLocalization without the locale parameter.
You may make a build to include resource bundles in the locales that you use.

String Manipulation Guidelines

You should use the Js2Xlf tool to convert JSON files into XLIFF files for translation.
You should deal with free text using ICU library at the server side.
You should use only casing functions for locale neutral situation.
You should not use locale sensitive casing functions provided by JavaScript.
You should always escape a string as a whole rather than character by character.
You should not use any comparing, searching, or replacing functions for strings that might contain combining character sequences.
You should not use inserting, removing, or splitting functions for strings that might contain special characters.
You should not use trimming functions for strings that might contain special characters.
You should not use counting functions for strings that might contain special characters.
You must check the return value of String.charAt() when the string contains surrogates.

Formatting and Validation Guidelines

You must use dojo.string.substitute() to generate text output rather than simply use "+" between strings.
You must use Dojo format functions to convert locale sensitive data into text.
You must use Dojo validating and parsing functions to convert text from the users' input into data.
You should not hard-code patterns and locales when formatting data.

Dijit Widgets Guidelines

You should not specify both the height and the width of a widget to be translated by numeric units.
You must ensure that all resources used in widgets are localizable.
You should consider BiDi support in development and customization.

Encoding Guidelines

Encoding Guidelines

The following guidelines should be used to implement internationalization in encoding.

The UTF-8 Encoding

You should always use UTF-8 for encoding settings wherever applicable.

This is a general rule for Web application design and development. You should manually set the encoding for files or I/O streams wherever applicable, because the default encoding (usually ISO-8859-1) cannot handle all Unicode characters. UTF-8 is the best choice when you use Dojo in your application, since Dojo only uses UTF-8.

The rest of this chapter describes the details of setting encodings in a Web application.

UTF-8 File Encoding

You should encode all text files in UTF-8.

You should encode all text files in UTF-8, including HTML files, CSS files, JavaScript files, etc.

You must specify the UTF-8 encoding in every HTML file before any non-English characters.

You must specify the encodings of all HTML files as early as possible. Ideally, this occurs on the server such that the server applies HTTP encoding headers to mark the document, otherwise this must be achieved in the browser using the meta tag. For example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Hello World!</title>
    </head>
    ...
</html>
```

This encoding declaration must appear before any non-English characters in a file; otherwise a browser might fail to read it correctly. For example, IE 6.0/7.0 cannot render the following content (encoded in UTF-8):

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
<html>
    <head>
        <title>???</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>Hello!</body>
</html>
```

By default, browsers assume that all files referred by an HTML file use the same encoding as the referring HTML file. So if you have the encoding of every HTML file specified, you do not need to declare the encoding setting in each CSS or JavaScript file again, but you can override the encoding anyway when some files are not in the same encoding as the HTML file. For example,

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
...
<link rel="stylesheet" href="foo.css" type="text/css" charset="utf-8">
...
<script src="bar.js" type="text/javascript" charset="utf-8"></script>
...
```

Sending and Receiving Requests

You must use the BOM header for UTF-16 files.

A BOM header consists of 2, 3, or 4 bytes at the very beginning of a text file to indicate its encoding. For example, 0xFF 0xFE means that the file is encoded in UTF-16LE, while 0xEF 0xBB 0xBF means that the encoding is UTF-8. The BOM header can override the encoding settings mentioned above in a browser.

Using UTF-16 is not recommended, but if you choose it for some reason, the BOM header is required. Because UTF-16 is not compatible with ASCII, a browser even does not have a chance to read the encoding setting of the file content.

You must use UTF-8 to decode XHR request parameters.

The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.xhr*` functions are the most common way in Dojo to enable Ajax features -- sending an asynchronous request to the server by an /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `XMLHttpRequest` object. The typical call to one of these functions can be:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.xhrGet({
    url: "foo.jsp",
    content: {"name": "\u4e00"} // \u4e00 ("?") is the Chinese character for "one"
});
```

The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `url` is where this request will be sent to. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `content` is the JSON object that will be sent in the request. In Dojo's implementation, the key and value pairs in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `content` are encoded by the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `encodeURIComponent` function first, and then converted to a query string like "key=value&key=value&...". The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `xhrPost` function puts the query string into the request content, and other functions like /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `xhrGet` append the query string to the end of the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `url`, so the previous code is equal to the following code:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.xhrGet({
    url: "foo.jsp?name=%e4%b8%80", // %e4%b8%80 are the UTF-8 bytes for \u4e00
});
```

Because the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `encodeURIComponent` function always uses UTF-8, you must use UTF-8 at the server side to decode the request parameters both in the URL (/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `xhrGet`) and in the request content (/* GeSHi (C) 2004

- 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `xhrPost`).

For example, in Tomcat, you can set the encoding of URL by the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `URIEncoding` attribute in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `server.xml`:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8080" maxHttpHeaderSize="8192"
           maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
           enableLookups="false" redirectPort="8443" acceptCount="100"
           connectionTimeout="20000" disableUploadTimeout="true"
           URIEncoding="UTF-8" />
```

You can set the encoding of the request content (/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `xhrPost`) by simply calling /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `request.setCharacterEncoding` before using the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `request` object:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<%@page contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<%request.setCharacterEncoding("utf-8");%>
...
name=<%=request.getParameter("name")%>
```

You MUST manually set the encoding on your server, because almost no Web server uses UTF-8 to decode URLs and request content by default. For example, Tomcat always uses ISO-8859-1 to deal with requests if you do not set the encoding. WebSphere uses a locale-encoding map to determine the request encoding from the client's language, but no locale is mapped to UTF-8 by default.

You must use UTF-8 encoding when using a non-English string in a URL.

Some browsers like IE always send URLs using the default system encoding. For example, in a Simplified Chinese Windows XP operating system, IE sends a URL encoded in GB2312. If you need to put some non-English parameters in a URL, make sure that you have encoded it first using the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `encodeURIComponent` function. For example, in a Simplified Chinese Windows XP, if you run the following script in IE:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.xhrPost({
    url: "foo.jsp?name1=\u4e00",
    content: {"name2": "\u4e00"}
});
```

You might get different results for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `name1` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900;

font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `name2` at the server side:

```
name1 --> 0xD2 0xBB (in GB2312, Wrong!)
name2 --> 0xE4 0xB8 0x80 (in UTF-8, Right!)
```

The right way is to encode /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `name1` first:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.xhrPost({
    url: "foo.jsp?name1=" + encodeURIComponent("\u4e00"),
    content: {"name2": "\u4e00"}
});
```

Sending Responses

You must set Content-Type in an HTTP response header if the response is not encoded in UTF-8.

An XMLHttpRequest object first checks the HTTP header of a response to see if there is a Content-Type property that sets the encoding of the response; otherwise, it always uses UTF-8 to decode the response into a string. Web servers usually set the Content-Type property automatically for dynamic files like JSP. However, for static files, Web servers probably do not know the encoding of the files and also do not set the Content-Type property for them.

Locale and Resource Bundle

You must set djConfig.locale in all files to the same as the locale used by the server code.
You must always use resource bundles to store the strings displayed to users.
You may use djConfig.locale to set the default locale and extra locales, and use only dojo.requireLocalization without the locale parameter.
You should make a build to include resource bundles in the locales that you use.

Locale Setting in Dojo

There is a slight difference in the locale naming conventions between Dojo and Java. Dojo uses "-" (hyphen) as the separator for concatenate language code, country code, and variants, whereas Java uses an "_" (underline). For example, "zh_CN" in Java is similar to "zh-cn" in Dojo.

Like the default locale in Java, Dojo has a global locale value that is stored in a global variable: dojo.locale. This default locale value affects the behavior of several locale-related functions and widgets. The value of dojo.locale is not supposed to be changed. You should use djConfig.locale to initialize this value.

Must set djConfig.locale in all files to achieve server-based personalization

If djConfig.locale is undefined, Dojo will consult the browser's navigator object for the setting chosen at browser install time. Note that this is unrelated to the locale setting in the preferences dialog, which is for interaction with the server only. To provide personalization from the server to control locale settings in an application, you must set djConfig.locale in the page at the server side, prior to loading dojo.js. For example, here is a JSP page that sets the default locale for Dojo: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
...
<%
String actualLocale = ResourceBundle.getBundle("my.app.test",
    request.getLocale()).getLocale().toString().replace('_', '-');
%>
<script type="text/javascript" src="../dojo/dojo.js"
    djConfig="locale: '<%=userLocale%>'"></script>
...
```

Resource Bundle Files

You must always use resource bundles to store the strings displayed to users.

Dojo introduces resource bundle into JavaScript. If you are familiar with Java resource bundle, you can find that Dojo resource bundle is very similar to Java resource bundle. The following table shows a summary of the differences between Java and Dojo:

	Java	Dojo
File Format	Properties file	JSON file
Locale Identifier	Suffix of file name	Directory name

Locale Naming	Use "_" (underline) as separator	Use "-" (hyphen) as separator
Get Bundle	ResourceBundle.getBundle	dojo.requireLocalization, dojo.i18n.getLocalization
Get Message	ResourceBundle.getString	JSON object

For example, there are two resource bundles named "bar" and "foo" in a package named "my.app" with some of their localized versions:

In Java (6 files with different names): /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
my/
   app/
       bar.properties
       bar_zh.properties
       bar_zh_CN.properties
       bar_zh_TW.properties
       foo.properties
       foo_zh_CN.properties
```

And in Dojo (4 directories and 6 files): /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
my/
   app/
       nls/
           bar.js
           foo.js
           zh/
              bar.js
           zh-cn/
                 bar.js
                 foo.js
           zh-tw/
                bar.js
```

The fallback strategy in Dojo is the same as that in Java.

Using Resource Bundle

First, you should use the dojo.registerModulePath function to define the directory where resource bundles are as a registered module. The module name needs to be used in later callings to the dojo.requireLocalization and dojo.i18n.getLocalization functions. For the previous example, you can use the following line to define the module "my.app":

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.registerModulePath("my.app", "../../my/app");`

Note: Here, the "../../my/app" path is relative to the directory that contains "dojo.js".

Then you can use the dojo.requireLocalization function to load resource bundles from files. After a resource bundle is loaded, the dojo.i18n.getLocalization function returns a copy of the bundle object.

When you get the bundle object, you can use it as a normal JSON object (a hash) to get messages. If you modify values in the bundle object, the original global bundle object will not be affected.

You may use djConfig.locale to set the default locale and extra locales, and use only dojo.requireLocalization without the locale parameter.

djConfig.locale overrides the browser's default locale as specified by the navigator Javascript object. This setting is effective for the entire page and must be declared prior to loading dojo.js. djConfig.extraLocale establishes additional locales whose resource bundles will be made available. This is used rarely to accomodate multiple languages on a single page. No other locales may be used on the page.

If you omit the locale parameter when calling the dojo.requireLocalization function, the function will load the resource bundles for locales in djConfig.locale as well as for all the locales in djConfig.extraLocale.

For example, if you define: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<script type="text/javascript" src="../dojo/dojo.js"
    djConfig="locale: 'zh-cn', extraLocale: ['zh-tw', 'fr']"></script>
```

then the following two code blocks are equal:

Code block A: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.requireLocalization("my.app", "bar");
var bar = dojo.i18n.getLocalization("my.app", "bar");
```

Code block B: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.requireLocalization("my.app", "bar", "zh-cn"); // default locale
dojo.requireLocalization("my.app", "bar", "zh-tw"); // extra locale
dojo.requireLocalization("my.app", "bar", "fr");    // extra locale
var bar = dojo.i18n.getLocalization("my.app", "bar", "zh-cn"); // default locale
```

The first method is preferred as it is less brittle.

Builds

Before you deploy your Web application using Dojo, you should consider building the Dojo layers that are used by your application into a single JavaScript file. Using such a build brings you many advantages. The unused scripts, white spaces, comments, and overridden string values can be removed to make smaller downloads, and the need to search by locale can be skipped such that extra server requests and 404 responses are avoided. In general, the build reduces the request time from the browser to the server to avoid latency issues.

Should make a build to include resource bundles in the locales that you use

Resource bundles can either be included in a build or be used without a build. If you use resource bundles without a build, the first request for each resource bundle will generate N+1 HTTP requests when it searches the server for values, where N is the number of segments in the target locale. For example, a call of dojo.requireLocalization("my.app", "bar") in the "zh-cn" locale looks for "bar.js" first in the "zh-cn", then in "zh", and finally in the root. Without optimization, some of these requests might result in harmless HTTP 404 errors (page not found) if a variant does not need to override any definitions from its parent.

Translation

JSON is a convenient and efficient format for resource bundles in JavaScript, but the JSON format is not well supported by many professional translation centers. XLIFF is the industry standard file format for localization and translation. Among other things, XLIFF will ease in declaration of encoding and hide details from the translator such as JavaScript character entities. Tools will be developed to support round-trip transforms between JSON and XLIFF. Support for gettext PO files in the future is also possible.

Translators must also be aware of the substitution syntax of Dojo — `${x}`

String Manipulation

Although the string data in JavaScript is specified in UTF-16 internally, JavaScript does not provide enough manipulation functions for Unicode strings. It is not practicable to develop a full Unicode compatible JavaScript library like ICU, since some Unicode algorithms, like normalization and collation, are too complicated to run in a script language. This chapter shows what you can do and what you should not do with strings in JavaScript. If you need full support for Unicode compatible string manipulations, you must use ICU library (ICU4C or ICU4J) at the server side instead of in JavaScript.

I18N Guidelines for String Manipulation

You should use the Js2Xlf tool to convert JSON files into XLIFF files for translation.
You should deal with free text using ICU library at the server side.
You should use only casing functions for locale neutral situation.
You should not use locale sensitive casing functions provided by JavaScript.
You should always escape a string as a whole rather than character by character.
You should not use any comparing, searching, or replacing functions for strings that might contain combining character sequences.
You should not use inserting, removing, or splitting functions for strings that might contain special characters.
You should not use trimming functions for strings that might contain special characters.
You should not use counting functions for strings that might contain special characters.
You must check the return value of String.charAt() when the string contains surrogates.

Special Kinds of Characters

Some special characters in Unicode can probably cause problems in a string to be manipulated:

Combining Character Sequence: a combining character sequence starts with a normal character followed by one or more combining marks. For example, "A" followed by a U+0768 (an accent) becomes "Ȧ". A combining character sequence should be treated as one character.
Surrogate Character: surrogate characters must appear in pairs. For example, "U+DB40 U+DC00" represents an ancient Chinese character. A surrogate pair must be treated as one character.

Basic String Manipulations

There are several basic string manipulations. Some of them are locale-sensitive, which means that they might cause different results in different locales:

Transformation: to transform a string into a specified form, for example, casing and escaping. Casing is a locale-sensitive operation, because some characters might have different casing results in different locales. For example, the Latin letter "i" is uppercased to "İ" (a dotted "I") in the Turkish language.
Recognition: to recognize special characters in strings, for example, white-space characters.
Join: to concatenate two strings into one.
Division: to separate one string into two substrings from a certain position.

Enumeration: to count characters in a string, for example, getting the string's length.
Comparison: to compare one string with another, for example, searching and sorting. Comparison is locale-sensitive.

Possible Problems

The following table shows the possible problems that might occur when you perform a specific manipulation on strings that contain special characters:

	Ordinary Character	Combining Character Sequence	Surrogate Character
Conversion	Locale Sensitive **(Unsolvable)**	Locale Sensitive **(Unsolvable)**	Locale Sensitive **(Unsolvable)**
Recognition	Unicode Specific **(Unsolvable)**	Unicode Specific **(Unsolvable)**	Unicode Specific **(Unsolvable)**
Join	(No Problem)	(No Problem)	(No Problem)
Division	(No Problem)	Broken Combining Character **(Unsolvable)**	Invalid Code Point (Solvable)
Enumeration	(No Problem)	Wrong Character Number **(Unsolvable)**	Wrong Character Number (Solvable)
Comparison	(No Problem)	No Canonical Comparison **(Unsolvable)**	(No Problem)

Locale Sensitive: JavaScript has a pair of locale sensitive casing functions -- String.toLocaleUpperCase and String.toLocaleLowerCase, but they only honor the default locale of the client's operating system and cannot be controlled by Web applications.

Unicode Specific: JavaScript might not recognize all Unicode characters that have a specific property. For example, some white-space characters (e.g., the no-break space -- U+00A0) cannot be removed when the string is trimmed in JavaScript: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
alert("\u00A0".replace(/\s/g, "").length); // 0 (in Firefox), 1 (in IE)
```

Show Me (THIS STILL NEEDS TO BE FIXED)

Broken Combining Character: a combining character sequence might be broken into meaningless characters. For example, when you replace all "A" in a string to "B", you might get all "À" replaced by "B̀ ", which is obviously not what you want. This kind of problems cannot be solved in JavaScript.
Invalid Code Point: a pair of surrogates can be separated into unpaired surrogates, which are invalid code points in Unicode. This problem could be a very serious problem, because invalid code points might crash some browsers (Safari 2.0.3 loses response when you try to select such invalid characters). Fortunately, unlike other kinds of problems, this kind of problems can be solved in JavaScript, because the surrogate scope is quite simple to be checked (from U+D800 to U+DFFF). You must make sure that no unpaired surrogates appear in the result string.
Wrong Character Number: because of combining character sequence and surrogate character, one meaningful character can consist of several characters in JavaScript that are 16-bit integers. You want to get the number of meaningful characters, but only get the number of 16-bit integers. This kind of problems cannot be solved in JavaScript, unless it is caused by surrogate characters.
No Canonical Comparison: JavaScript cannot make canonical comparison on the strings in different Normal Form. Although the ECMAScript Specification v3 recommends that canonical comparison be supported by the String.localeCompare(str) function, no browser actually implements this feature.

You should deal with free text using ICU library at the server side.

The only safe operation for all strings is joining. To manipulate a string, first you need to know what kinds of characters are allowed in it and whether the manipulation is locale sensitive. If the string accepts combining character sequences and surrogate characters or if the manipulation is locale sensitive like casing, you are recommended to send the string to the server and deal with it by using ICU library at the server side. For example, your application might need to handle the following kinds of strings: **Restricted Identifier:** most identifiers, such as the user's login name, are restricted to letters and numbers, so you can manipulate them in JavaScript freely and safely. **Program Information:** program information is a string that is only used for your application code, for example, enumeration item names, program commands, and statements. Make sure that no combining character sequence or surrogate character is included in your program information. Then you can manipulate them without potential problems. **URL or E-mail Address:** URLs and e-mail addresses might contain almost any Unicode characters. You should deal with them at the server side. **Free Input Text:** the text entered freely by users should be manipulated at the server side. String Manipulating Functions in JavaScript and Dojo

You should use only casing functions for locale neutral situation.

Some casing functions perform only locale neutral casing conversion. You should use them only for program information that is not displayed to end users directly. These functions include:

- String.toLowerCase()
- String.toUpperCase()

You should not use locale sensitive casing functions provided by JavaScript.

Other casing functions perform locale sensitive casing conversion with the default locale of the client's operating system. You should never use them. The reasons are: first, it is not guaranteed that browsers implement these functions as the exact behavior defined in the latest

Unicode Standard; second, you cannot control the output by your Web application. It is almost unacceptable that an end user needs to change the default locale of the operating system so as to change the locale of your Web application. These functions include:

- String.toLocaleLowerCase()
- String.toLocaleUpperCase()

You should always escape a string as a whole rather than character by character.

- window.escape()
- window.unescape()
- window.encodeURI()
- window.decodeURI()
- window.encodeURIComponent()
- window.decodeURIComponent()

You should not use any comparing, searching, or replacing functions for strings that might contain combining character sequences.

Functions in this section might encounter problems from the manipulation of comparison and division on free text. They might replace wrong characters, return invalid strings or characters, and they cannot perform canonical comparison. Use them only for program information without special characters. These functions include:

- String.indexOf()
- String.lastIndexOf()
- String.localeCompare()
- String.match()
- String.replace()
- String.search()
- dojo.string.substitute()

Should not use inserting, removing, or splitting functions for strings that might contain special characters.

Functions in this section might have the problem from division on free text. They might return invalid string or characters. Use them only for program information without special characters. These functions include:

- String.slice()
- String.split()
- String.substring()

You should not use trimming functions for strings that might contain special characters.

Trimming functions might have the problem of Recognition. They cannot recognize all white-space characters defined by Unicode. Use them only for program information without special characters. These functions include:

- dojo.string.trim()
- dojo.trim()

You should not use counting functions for strings that might contain special characters.

Functions for counting characters can only return the number of UTF-16 units rather than the real meaningful characters. These functions include:

- String.length
- dojo.string.pad()

You must check the return value of String.charAt() when the string contains surrogates.

The String.charAt() function might return an unpaired surrogate that is invalid. You should always check the returned value, for example: view plaincopy to clipboardprint? /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
function codePointAt(s, pos) {
    var c = s.charAt(pos);
    if (c == "") {
        return c;
    }
    var code = c.charCodeAt(0);
    if (code >= 0xD800 && code <= 0xDBFF) {
        // lower surrogate detected, we need to fetch more char
        if (pos < s.length - 1) {
            code = s.charCodeAt(pos + 1);
            return code < 0xDC00 || code > 0xDFFF ? "" : s.substring(pos, pos + 2);
        } else {
            return ""; // error in the string
        }
    } else if (code >= 0xDC00 && code <= 0xDFFF) {
        // higher surrogate detected, we need to fetch more char backward
        if (pos > 0) {
            code = s.charCodeAt(pos - 1);
            return code < 0xD800 || code > 0xDBFF ? "" : s.substring(pos - 1, pos + 1);
        } else {
            return ""; // error in the string
        }
    }
}
```

Formatting and Validation

Dojo has the capability of formatting messages like what Java does. Message formatting functions are very important to the translatability of an application -- separating code and translatable materials and supporting application translation without any modification to the code. Furthermore, Dojo provides full functions for data formatting and validation based on Unicode CLDR data, just like what ICU library does. You can easily format your output and validate users' input in Dojo without invoking code at the server side.

Must use dojo.string.substitute() to generate text output rather than simply use "+" between strings.

You can use dojo.string.substitute() to substitute place holders in the message string. This function provides a function similar to the java.text.MessageFormat.format() function in Java. You must use this function to get your text output instead of simply using the "+" operator; otherwise your application loses its translatability. For example, an English message "File 'foo.txt' is not found in directory '/root/bar'." can be translated as "?'/root/bar'??????'foo.txt'???" in Chinese. If you write your code like msg = pieceA + fileName + pieceB + dirName + pieceC, you cannot translate this message into Chinese without modifying your code, because the positions of fileName and dirName are swapped. Therefore, the right approach is to write the code like this: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.requireLocalization("my.app", "message");
   var message = dojo.i18n.getLocalization("my.app", "message");
   msg = dojo.string.substitute(message.FILE_NOT_FOUND_IN_DIR, ["foo.txt", "/root/bar"]);
```

And the resource bundle "message.js" is like this: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{
    FILE_NOT_FOUND_IN_DIR: "File '${0}' is not found in directory '${1}'."
}
```

Now you can translate the message into Chinese by only providing a translated resource bundle in the locale "zh-cn": /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
{
   FILE_NOT_FOUND_IN_DIR: "?'${1}'??????'${0}'???"    FIXME!
}
```

Here are more examples of using dojo.string.substitute(): /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.require('dojo.string');
    dojo.require('dojo.number');
// Use format function.
// "dojo.number.format" is a format function defined by Dojo.
// It uses the default locale in Dojo, as defined by the user's environment
 console.debug(dojo.string.substitute(
     "The number of '${1}' is '${0:dojo.number.format}'.",["saved files", "123456"]));
// Output: The number of saved files is 123,456.

// Use named substitutions.
 console.debug(dojo.string.substitute(
     "The number of '${item}' is '${number:dojo.number.format}'.",
     {item: "saved files", number: 123456}));
// Output: The number of saved files is 123,456.
```

Cultural Formatting and Validation

You must use Dojo format functions to convert locale sensitive data into text

Dojo utilizes the Unicode CLDR data to format and validate locale sensitive information, such as time, number, and currency. The built-in conversion functions in JavaScript like Date.toString() are not fully Unicode- compatible, and you should not use them in a globalized Web application. For dates and numbers, Dojo format functions are not needed to be called explicitly in your code, and the dojo.string.substitute() function mentioned above can combine the format capabilities with format functions. You only need to specify the format function name in the template string, for example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.require('dojo.string');
    dojo.require('dojo.number');
    dojo.require('dojo.date.locale');
    var msg = "Number of processed file number before ${1:dojo.date.locale.format}:"
        + "${0:dojo.number.format}";
    console.debug(dojo.string.substitute(msg, [123456, new Date()]);
```

If you want more control over the output format and locale, you can use the specific format functions respectively. For details, refer to the API documents of the following functions:

- dojo.number.format
- dojo.date.locale.format

You must use Dojo validating and parsing functions to convert text from the users' input into data

Although JavaScript provides some methods to convert text to data, it does not fully support localized text, for example, view plaincopy to clipboardprint? /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
console.debug(new Number("123456"); // output: 123456
console.debug(new Number("123,456"); // output: NaN
```

You must use Dojo's functions to handle validation and parsing. For details, refer to the API documents of the following functions:

- dojo.number.regexp
- dojo.number.parse
- dojo.date.locale.regexp
- dojo.date.locale.parse

You should not hard-code patterns and locales when formatting data

Usually, you only need to set the selector and the formatLength properties for date (type for number and currency for currency), and use the default values of other properties when calling formatting, validating, or parsing functions. Dojo looks for the correct pattern based on the current default locale. If you specify the pattern by hard coding, the output format cannot be changed with the user's locale.

Generating files for cultural support in other locales

Dojo uses the Unicode Common Locale Data Repository (CLDR) to format and parse locale-specific data. A subset of the locales supported by the CLDR is transformed to JavaScript and checked into the dojo.cldr package as resource bundles. Although only a handful of locales ship in dojo.cldr, any or all of the hundreds of locales supported by the CLDR may be generated easily by invoking a script. Dojo provides an Ant task to perform the transformation from LDML, an XML-based markup, to JSON files. The entire CLDR source is available with these tools in the util/buildscripts/cldr directory. To invoke this script, you need to check out the current Dojo source code or download the buildscripts archive file and overlay it with your Dojo build, and consult the README file in util/buildscripts/cldr for details. It is as simple as invoking 'ant' in that directory. After the script is executed, all transformed JSON files are added to the "dojo/cldr/nls" directory.

Dijit Widgets

All Dijit Widgets are implemented according to these Globalization guidelines except for right-to-left bi-directional (BiDi) support, which is not complete as of the 0.9 release.

You should not specify both the height and the width of a widget to be translated by numeric units.
You must ensure that all resources used in widgets are localizable.
You should consider BiDi support in development and customization.

Translation Spread

Should not specify both the height and the width of a widget to be translated by numeric units

When the text content in a widget is translated into another language, the translated string might be longer than the original one, i.e., the one in English. If you specify the size of the widget by numeric units, the widget may not render properly in other languages. For example, some text might be truncated or unexpected text wrapping may distort the layout. Therefore, keeping the translation spread in mind at the beginning of development might be a better practice. You can use the following approaches to set the size of a widget:

- Use percentage rather than numeric units.
- Set only one of the dimensions (height or width) in numeric and leave the other as auto. You can also use the white-space style to control the text-wrapping style. This can make the widget expand on the right.

Using BiDi Support

Eventually, all Dijit widgets will be consistent with the direction of the HTML document. [TODO: the following text needs to be changed; I think we're only checking for the .dijitRtl class which is based on the document's direction -AP] By default, the display direction is inherited from the parent node, and the direction of the whole page is left-to-right. There are two ways to change the display direction for an HTML node: one is to set the DIR attribute, and the other is to set the direction style. The direction style can override the setting to the DIR attribute, and the computed value of the direction style is the actual way in which the node is displayed. To make the pop-up menu mirrored, for example, you can just set DIR="rtl" on the HTML node to mirror the whole page.

If a page contains a right-to-left layout, you must also import the RTL style sheet of the current Dojo theme and make sure that it is the last one to be imported. The RTL style sheet has no effect on the left-to-right layout.
view plaincopy to clipboardprint?

```
<style>
@import "../../../dojo/resources/dojo.css";
@import "../../themes/tundra/tundra.css";
@import "../../themes/tundra/tundra_rtl.css";
</style>
```

If the whole page is in a right-to-left layout, you only need to import the RTL style sheet. To make the whole page right-to-left, you can set

DIR="rtl" on either the HTML node or the BODY node.

If you want to mix both left-to-right and right-to-left layouts in a page, you must add a dijitRtl class to each right-to-left layout container besides importing the RTL style sheet.
view plaincopy to clipboardprint?

```
<div dir="rtl" class="dijitRtl">
<div dojoType="dijit.Tree">
...
</div>
</div>
```

Using Localized Widgets

Benefited from the CLDR support of the Dojo core library, Dijit provides plenty of localized widgets that are related to date, number, and currency data processing. You can find their introduction and manual in the corresponding chapters in the Dojo Book 0.9. These widgets include:

- dijit.form.NumberTextBox
- dijit.form.CurrencyTextBox
- dijit.form.TimeTextBox
- dijit.form.DateTextBox

Widgets Development and Customization

You must ensure that all resources used in widgets are localizable.

The resources that are used in widgets include texts, images, audio and video clips, etc. Only HTML code and style names can be hard-coded in the widget code. You can use the resource bundles in Dojo to store localized resources. See "Locale and Resource Bundle" for more information.

Should consider BiDi support in development and customization .

Most browsers and HTML itself support the mirrored display in the BiDi environment, but it does not mean that Dijit widgets can support BiDi by nature. There still are some specific codes in Dijit to handle the mirror feature. For example, when displayed in right-to-left direction, the popup menu should appear from left rather than from right, and the arrow that indicates sub menus should also be flipped.

[TODO: strike this para? I think dir is gone? -AP]
Dijit widgets have a dir attribute that is defined in the dijit._Widget class, the base class of all widgets. The dir attribute can be set to indicate the direction of the widget when it is being created. To know the actual direction in the code, you should use the _Widget.isLeftToRight() function. The returned value of this function is not supposed to be changed after the widget is created. That is, currently, Dijit widgets do not support dynamic change of the display direction.

To develop or customize the Dijit widget with BiDi enabled, you should use different styles for the left-to-right and right-to-left directions. For example, it is always required to change the styles like float: left and background-image: url(images/left-arrow.png) for the right-to-left layout for some portion of a widget. You can write all right-to-left styles in an RTL style sheet, and all style classes should begin with the dijitRtl class:
view plaincopy to clipboardprint?

```
.dijitRtl .TreeContainer .TreeExpando {
float:right;
}
```

Better still, try to avoid any gestures or visual cues which imply left or right direction, so that styling and coding for BiDi becomes unnecessary. For example, dijit.Tree uses the space key to open a tree instead of a left or right arrow. A "+" symbol might be used for an indicator for extended data, rather than a left or right arrow.

Encoding considerations

The two most important aspects of internationalization are insuring that the inputs and the outputs are in the proper encoding. Thankfully, UTF-8 encoding can be used exclusively on the wire in modern web applications to make this interaction extremely simple. Once read into memory, JavaScript treats all strings as a series of double-byte characters and encoding is irrelevant. Dojo makes no attempt to implement any encoding or decoding algorithms in JavaScript; this is the responsibility of the browser.

Other encodings should be used with great care. A user agent such as one of the current generation browsers infers the encoding of a page using the content-type header provided by a server or it may be picked up from a meta tag in the head of a document, such as the following:

```
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
```

The most common means of specifying the encoding in a page is to use the META tag. Note that the META tag only works in pages loaded directly by browsers or IFRAMEs and may not function when used in other situations, such as in a document referenced by HREF in dijit.layout.ContentPane. UTF-8 is the detault encoding used by XML documents exchanged by the XmlHttpRequest object and also is the encoding that is used internally by Dojo APIs such as dojo.io.bind. We recommend using UTF-8 as the encoding for all of your applications.

It is important to properly specify the encoding of the page when forms or form widgets are present as the page's encoding is used to properly encode data that is sent back to a server.

Specifying a Locale

By default, the locale of this browser is "".

To override the locale to make German the default for Dojo, use

```
<SCRIPT TYPE="text/javascript" SRC="dojo.js" djConfig="locale: 'de'"></SCRIPT>
```

What is a Locale?

Localization is driven by a locale, a short string often supplied by the host environment, which conforms to RFC 3066 used in the HTML specification. It consists of short identifiers, typically two characters long which are case-insensitive. Note that Dojo uses dash separators like the RFC, not underscores like Java (e.g. "en-us", not "en_US"). Typically country codes are used in the optional second identifier, and additional variants may be specified. For example, Japanese is "ja"; Japanese in Japan is "ja-jp". Notice that the lower case is intentional -- while Dojo will often convert all locales to lowercase to normalize them, it is the lowercase that must be used when defining your resources.

How does Dojo find the locale?

By default, Dojo derives the user locale setting from the navigator browser object, the only locale information available from Javascript. The browser locale is determined during browser installation and is not easily configurable. Note that this is not the same as the locale in the preferences dialog which can be used to accompany HTTP requests; there is unfortunately no way to access that locale from the client without a server round-trip. The user's locale may easily be overridden on a page prior to the Dojo bootstrap by setting the djConfig.locale property. Of course, setting this property in a static way defeats internationalization for other users. This setting may be established by a server to achieve personalization of web applications, where a user may be able to select their locale or this information may be available through some other means. Once Dojo is loaded, it is not possible to change the locale for the page.

Using many locales at the same time

In the unusual case where multiple locales are used on a single page, the djConfig.extraLocale property must be set, prior to bootstrap, listing the additional locales as elements in an array, otherwise they will not work at runtime. Optionally, one of these extra locales may be passed into routines like dojo.date.format or Dijit widgets using the 'lang' attribute, but such use cases are rare. Typically, the one locale is sufficient to localize the entire page and the locale should not be applied to any one specific widget or API.

Translatable Resource bundles

Purpose

dojo.i18n solves a problem in Javascript: now that we've implemented a significant amount of logic on the client, there are i18n issues which cannot easily be solved by server substitution. Furthermore, server substitution scales poorly and exposes the client-side toolkit to a particular server architecture. With a client-side internationalization framework, the integration point moves to the browser where simple or complex logic can be applied without becoming a bottleneck. This architecture also encourages encapsulation and efficient caching both at edge servers and in the browser. And, while all translations are present on the server, rest assured that only those which match the user's locale are requested by the client and sent over the wire.

The methods used in Dojo to substitute localized resources are intended for Dojo-generated content. There are trade-offs to this approach, such as trading server-load for client-side response time. It is usually best to continue using the same mechanism to localize the rest of the page, which is typically a webapp server, rather than trying to force everything on the page through Dojo and Javascript. This does mean that there will usually be two different sets of resources to manage, translate, and deploy. Dojo and Dijit will soon provide translations for many major languages, and additional translations may be provided by the community. Those augmenting Dojo or writing their own widgets will need to create and translate their own set of resources, as needed.

The translation task in a Dojo application is limited to anything which appears in the DOM, that is anything which is visible on the web page. It's unacceptable to hard-code an English string and have that appear to users, no matter how unlikely it is to appear. Debug or console message however, for the time being, are not localized as a matter of policy. There are no guidelines on console messages at the present time. They are generally discouraged in production code.

Localizing Strings

```
dojo.requireLocalization() / dojo.i18n.getLocalization()
```

these methods leverage the DojoPackageSystem (link?) to load localized resources. Each translated resource is implemented as a file containing a Javascript Object (see JSON notation) where each property may be a string or any other Javascript type. Resources are located within the directory structure beneath a specially named "nls" directory (short for native language support). Each translation is made available in a subdirectory named by locale.

dojo.requireLocalization() is used to declare usage of these resources and load them in the same way that dojo.requires() pulls in Javascript packages, but using the translation appropriate to the caller. The location of the bundle is specified using two arguments: the first is the directory structure containing the nls directory; the second is the name of the file in that directory containing the localized resources. The locale used is discovered at runtime from the browser, or specified by an override in djConfig (see "Specifying a locale") If djConfig.extraLocale is set, the localizations in that list will be loaded also.

Use dojo.i18n.getLocalization() to get a reference to the object representing the localized resources. The resources loaded by dojo.requireLocalization() are searched and one best matching the user's locale are used. The localized values will be available as properties on the returned object. For example:

```
//TODO: replace this example with the strings from dojo.color when translations are available
```

```
dojo.require("dojo.i18n");
dojo.requireLocalization("dijit.form", "validate");
var validate = dojo.i18n.getLocalization("dijit.form", "validate");
console.log(validate.invalidMessage);
```

For an English-speaking user, the example above will display the value for invalidMessage from dijit/form/validate.js:

```
"* The value entered is not valid."
```

The root happens to have the English translation, which also acts as a fallback for any unsupported locales (English was an arbitrary choice, but the one commonly used in Dojo). Therefore, no translations were found in the en or en-us directories as they would have been redundant. Meanwhile, a Japanese user in the ja-jp locale will see the value in dijit/form/nls/ja, which is the best match for that locale:

```
"* ??????????????????????"
```

Translation subdirectories are searched and mixed in such a way that variants can specify overrides for some or all of their parent locale. Because the search requires looking for translations under both the language as well as variants, sometimes a 404 will occur; this is normal and can be optimized at build time.

Cultural conventions: Date, Number and Currency

Dates and Times

Unlike standard Javascript, Dojo is capable of formatting and parsing date formats for many locales, using the CLDR repository at unicode.org. Both the date and time portion of a JavaScript Date object may be converted to or from a String representation using these routines. For example, look at the following date formatted using the default locale for the user (in this case, English - United States) and also with a specific locale override of Chinese - PRC China:

```
// the page must specify djConfig.extraLocale: 'zh-cn' to bootstrap the environment with support for an extra locale
dojo.require("dojo.date.locale");
var d = new Date(2006,9,29,12,30);
// to format a date, simply pass the date to the format function
dojo.date.locale.format(d);
=> "10/29/06 12:30 PM"
// the second argument may contain a list of options in Object syntax, such as overriding the default locale
dojo.date.locale.format(d, {locale:'zh-cn'})
=> "06-10-29 ??12:30"
```

Note that the positioning of month, day, and year are all different, as well as the "PM" symbol and its placement. Use of a locale override in this API is limited to examples like this one; usually the correct thing to do is to assume the user's default, or override the locale for the entire page (see "Setting a locale") Dojo.date offers a variety of formatting choices, such as the option to a different format "length" -- a choice of "short", "medium", "long", or "full" -- or to print only the date or time portion of the Date object:

```
dojo.date.locale.format(d, {selector:'date', formatLength:'full'});
=> "Sunday, October 29, 2006"
dojo.date.locale.format(d, {selector:'time', formatLength:'long', locale:'zh-cn'});
=> "??12?30?00?"
```

Also, it is possible to reverse the process and parse String objects into Dates. For a user running in a Dutch locale like "nl-nl", the following would produce a valid Date object:

```
dojo.date.locale.parse("maandag 30 oktober 2006", {formatLength: "full"});
```

Special patterns may be specified may be used to provide custom formats, however using such a pattern overrides the locale-specific behavior and may result in an application that is not properly localized. The patterns used follow the specification and are similar to those used by the Java dateformat class (e.g. MMddyyyy).

Also available under dojo.cldr.supplemental are routines to provide the first day of the week and the start and end of the weekend, according to local custom.

Numbers and currencies

The formatting and parsing of numbers is handled in much the same way. Conventions vary around the world for the decimal and thousands separator, placement of the sign, and symbols used to indicate exponential numbers or percentages. There are other exceptions, such as in India, where the thousands separator is used at the thousands place, then again after every two digits instead of three. Dojo provides the facilities to properly format and parse numbers on a localized basis using the methods in dojo.number:

```
dojo.require("dojo.number");
// in the United States
dojo.number.format(1234567.89);
=> "1,234,567.89"

// in France
dojo.number.format(1234567.89);
=> "1 234 567,89"
```

Other options may be specified to limit output to a certain number of decimal places or use rounding. And again, custom formats may be specified, overriding the local customs.

dojo.currency combines the functionality of dojo.number to use the appropriate syntax with knowledge of the conventions associated with a particular currency -- this includes the number of decimal places typically used with a currency, rounding conventions, and the currency symbol which itself may be rendered differently according to locale, any of these may be overridden. When calling dojo.currency APIs, be sure to specify a currency according to its 3-letter ISO-4217 symbol.

```
dojo.require("dojo.currency");
// in the United States
dojo.currency.format(1234.567, {currency: "USD"});
=> "$1,234.57"
dojo.currency.format(1234.567, {currency: "EUR"});
```

```
=> "€1,234.57"

// a French-speaking Swiss user would see
dojo.currency.format(-1234.567, {currency: "EUR"});
=> "-1 234,57 €"
// while a German-speaking Swiss user would see
dojo.currency.format(-1234.567, {currency: "EUR"});
=> "-€ 1,234.57"
```

Note: handling of Hindi and Arabic style numerals is planned for 1.0, but not yet implemented.

Locale support

It is not necessary to craft translated files to support these conventions in your locale. Dojo supports the above cultural conventions and currency types in pretty much every locale available through the CLDR, which is included with the Dojo build tools. However, by default, only a subset of these locales and currencies are built as Javascript objects in the Dojo repository under dojo.cldr. A script is available to build a custom or more complete set -- look for instructions at util/buildscripts/cldr/README.

Bi-Directional Text

Bi-directional Text (BiDi)

Some languages, mostly middle-eastern in origin, have text flow from the right to the left (e.g. Hebrew and Arabic) Again, the web browser generally takes care of this for us, provided the appropriate HTML attributes are set. The HTML elements comprising the widgets on the page are flipped from right to left, but if a graphic indicated a horizontal direction, it may also have to be flipped or swapped to follow the new right-to-left logic. Wherever possible, this is achieved using CSS rules. Sometimes the logic of an application or widget must change to accommodate Bi-Di. For example, in right-to-left mode, menu widgets drop down and to the left and contents are right-justified.

Bi-Di issues are largely related to widgets, and are therefore addressed mostly within the Dijit project. To support right-to-left users, some extra CSS is required. At the present time, these rules reside in a separate stylesheet and must be explicitly loaded for all users to support Bi-Di usage. (e.g. themes/tundra/tundra_rtl.css, which imports themes/tundra/tundra.css for you)

As of the 1.0 release, the following the widgets have limited BiDi support using the Tundra theme, but more testing is needed and known bugs exist, especially in Internet Explorer:

- dijit.ColorPalette
- dijit.Declaration
- dijit.Dialog
- dijit.Menu
- dijit.ProgressBar
- dijit.TitlePane
- dijit.ToolBar
- dijit.Tooltip
- dijit.Tree
- dijit.form.Button
- dijit.form.CheckBox
- dijit.form.ComboBox
- dijit.form.CurrencyTextBox
- dijit.form.DateTextBox
- dijit.form.FilteringSelect
- dijit.form.Form
- dijit.form.InlineEditBox
- dijit.form.NumberSpinner
- dijit.form.NumberTextBox
- dijit.form.Textarea
- dijit.form.TextBox
- dijit.form.TimeTextBox
- dijit.form.ValidationTextBox
- dijit.layout.ContentPane
- dijit.layout.LinkPane
- dijit.layout.StackContainer
- dijit.layout.AccordionContainer

Globalization Tutorial

This tutorial focuses on the Dojo globalization features that are fundamental to help developers construct a globalized AJAX application. Based on the Dojo globalization basics (introduced in Dojo Globalization Guidelines), this tutorial first introduces a Dojo Globalization Hello World application step by step, and then implements a more integrated MVC global Web application, the Dojo Car Store. At the end of this tutorial, some Dojo globalization usage samples are provided for your reference. Before you start with this tutorial, it is suggested that you first read the Dojo Globalization Guidelines, which introduces the technical details mentioned in this tutorial.

Implementing the Dojo car store application

Dojo Car Store is a globalized fictional MVC Web application demo. It’s designed to illustrate how to integrate dojo with typical web technology and construct a globalized AJAX Web application.

As a pure JavaScript library, dojo can be used with a wide variety of server side technologies, including JSP, PHP, Ruby, WebService, etc. This tutorial chooses JSP for demonstration and focuses on how to integrate dojo globalization features with the existing J2EE globalization best practices (JSTL, JSF, etc.).

Dojo Car Store mainly provides the following functions:
1. Locale switching
2. Car exhibiting (including detailed information)
3. Shopping cart
4. Order and shipment form

4.2 Step by step implementation
4.2.1. Locale (language) switching
You have two options for switching locales (languages) in Dojo Car Store.
- Option 1: Browser language option
Step1:
Open the language configuration tab of your browser:
- For Firefox: Tools-Options-Advanced-Language options
- For IE: Tools-Internet Options-General-Languages
Step2:
Add a locale (language) to or remove a locale (language) from the list, and set your preferred language as the first priority option.
Step3:
Click Confirm, close the tab, and refresh the current page. Please make sure that there is no locale parameter like ……
index.jsp?locale=en_US in the address field of the browser, because the demo will deduce locale information from your preferred language list first (see Option2), and then from the browser language configuration.

- Option2: User's preferred language list on the entry page – index.jsp
Select your preferred language from the dropdown list of languages. When you select another locale, the page will be refreshed automatically.

To ensure a consistent locale deducing strategy for both the client side (dojo) and the server side (JSTL /JSF tag),
the following steps are used:
Step 1: Get the locale from HttpRequest on the server side (for the above mentioned Option1):
<% Locale locale = request.getLocale(); %>
or
Get the locale parameter from HttpRequest URL on the server side (for the above mentioned Option2):
<% String localeStr = request.getParameter("locale");
……
Locale locale = new Locale(……) %>
or
Get the locale parameter from Http session on the server side:
<% Object sessionLocale = session.getAttribute("locale"); %>

Step 2: Set the retained locale in the session scope for later use.
<% session.setAttribute("locale", locale); %>
Note: Here the parameter set in the session is the uniform locale for both the client and the server side.

Step3: Initialize the dojo global parameter dojo.locale as:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}

">

Please also note that dojo only accepts locale string with the format "en-us", but the locale string from Servlet API has the format "en_US". Therefore, the above modification is required.

Please refer to index.jsp and checkout.jsp (under DojoGlobalizationDemo.war/DojoCarStore/) for more details.

4.2.2. Literal translation
The literal translations of Dojo Car Store are implemented in the following ways:
1. JSTL i18n tags
Three JSTL i18n tags are used for loading resource bundle: setLocale, setBundle, and message.
Step1: Set the resource bundle locale:

Here, ${sessionScope.locale} is the locale parameter set in session previously.
Step2: Set the base resource bundle:

Step3: Display the literal translation using the resource bundle:

2. Localized JSF tags
Three JSF tags are used for loading the resource bundle: locale, loadBundle, outputText
Step1: Set the locale for the JSF view tag:

Here, #{locale} is the locale parameter set in session previously.
Step2: Set the base resource bundle:

Step3: Display the literal translation using the resource bundle:

3. JSON resource bundle
The literal translations of all the localized dojo widgets come from the JSON resource bundle, for example, DateTextbox, ValidationTextbox, etc. Actually, you can get any JSON resource bundle in the dojo package including CLDR data (please refer to Dojo resource bundle usage sample in Appendix A).

4. Java resource bundle
Some literal translations are first generated on the server side and then sent back to the client side.
Therefore, these literal translations are loaded from the resource bundle using the normal Java method like:
ResourceBundle resBundle = ResourceBundle.getBundle(……, this.locale);
String pattern = resBundle.getString(……);
Object[] msgArgs = {String.valueOf(this.currentOrderId)};
String confirmJson = …… + MessageFormat.format(pattern, msgArgs) + ……;

Please refer to dojo.g11n.demo.servlet.DojoServlet and dojo.g11n.demo.db.DBManager for more details.

5. DB
Car profiles are stored in separate tables in DB with different language versions (the name of a table contains locale information like "CAR_en"), so when a user loads car profiles, the locale parameter should match the corresponding table.

Please refer to dojo.g11n.demo.servlet.DojoServlet and dojo.g11n.demo.db.DBManager for more details.

4.2.3. UI Implementation
1. Car inventory list
Car inventory list exhibits the images of all the cars in the inventory. You can switch between pages and click the image of a certain car to see its detailed information (or drag it to the Car detail view area directly)

This part mainly includes: Dojo animate transition (fadeIn / fadeOut), Dojo drag&drop, Dojo xhr transport, Dojo Fisheye widget etc.
Please refer to index.jsp and indexController.js(imgController) for more details.

2. Car detail view
Car detail view displays detailed information about the selected car, including car name, price, description, index, etc. Car detail view also provides a slide show that displays different car images. To add the current car to the shopping cart, you can either click the "Add to Cart" button or drag the corresponding image of the car to the shopping cart directly

This part mainly includes: Dojo ContentPane, Dojo animate transition (fadeIn / fadeOut), Dojo drag&drop, Dojo currency API, Dojo JSON, etc.

Please refer to index.jsp and indexController.js(detailController) for more details.

3. Shopping cart
Shopping cart contains information about the cars that you have selected, including car index, name, price, quantity, and total price. To add a car to the shopping cart, you can drag the corresponding image of the car from the Car inventory list or the Car detail view into the shopping cart. Another way to add a car to the shopping cart is to click the Add To Cart button in the Car detail view.

This part mainly includes: Dojo ContentPane, Dojo Currency/Number API, Dojo JSON, Dojo Style, Dojo xhr transport, etc.

Please refer to index.jsp and indexController.js(shoppingCartController) for more details.

4. Order and shipment form completion
When checking out, you need to fill in the shipment and Credit card information forms. In this part, dijit.form.DateTextbox / ValidationTextbox are used because they provide customized validation rules as well as localized warning messages.

After completing the forms, you can click the Checkout button to submit the order. Here, Dojo Dialog widget is used to promote the transaction confirmation and error information.

Please refer to checkout.jsp and checkOutController.js for more details.

4.2.4. Server side XMLHttpRequest processor
DojoServlet processes all the XMLHttpRequest sent through Dojo xhr from client side, such as:
1. Setting CharacterEncoding for both the request and the response as UTF-8;
2. Deducing locale from the request session:
locale = request.getSession().getAttribute("locale")
Note:
This demo provides two ways for locale switching. Therefore, simply deducing locale like
Locale locale = request.getLocale(); is not enough because this way only deduces the best matched locale between the client browser and the web container, without considering the user selected language (from the dropdown list). This is simplified by getting the locale parameter from the session (see 4.2.1) because this parameter is the uniform locale on both the server side and the client side.

Please refer to dojo.g11n.demo.servlet.DojoServlet for more details.

4.2.5. Persistence controller & DB
DBManager manages DB connection, data query and updating. Car profiles are stored in DB with different language versions, so car profile query is locale specific.

Dojo Car Store uses embedded Derby 10.1.3.1 by default (DojoGlobalizationDemo.war/DB/DojoCarStoreDB). You can also choose Derby client mode or DB2 instead, please refer to Appendix B for detailed steps.
Note: Derby database uses Unicode by default. When creating databases, tables, and inserting data, please make sure that they are encoded as UTF-8. Please refer to dojo.g11n.demo.db.DBManager for more details.

4.2.6. Web Server g11n configuration.
Dojo Car Store uses POST XMLHttpRequest (only UTF-8 encoding) to submit client data that contains locale specific literal, and then uses request.setCharacterEncoding("UTF-8") to get the submitted data correctly.

Note: If you use GET XMLHttpRequest and append the data parameters to URL, please make sure that the Web server parses the URL using the correct encoding, because request.setCharacterEncoding("UTF-8") only deals with the request body but not the URL itself. For example, in Tomcat server.xml, there will be a configuration item named URIEncoding. Please set its value to UTF-8, so that Web server will use UTF-8 to parse URL parameters.

5. Limitations
The Dojo Car Store is for demo purpose only, so all the data is fictional. If occasionally you find the page is not rendered correctly, it may be caused by a browser cache problem. Please restart your browser and try again. If the problem still exists, please check your server and database.

Installing the globalization sample applications

To install the Hello World sample and demo application, first download dojo-g11n-helloworld.zip and DojoGlobalizationDemo.war from http://TODO. The dojo-g11n-helloworld.zip file contains the Dojo G11N Hello World sample, and the DojoGlobalizationDemo.war file containes the Dojo Car Store and Dojo G11N samples.

Prerequisites

1. Browsers supported:
- Internet Explorer (Windows) 6.0+
- Firefox 1.5 – 2.x+
- Opera 9.x
- Safari 2.x +
2. Web server
- Apache Tomcat 5.5 and Java Development Kit 5.0
(Or - WebSphere Application Server Community Edition 1.1)

Installation (Only needed for Dojo Car Store demo app)

1. Apache Tomcat 5.5 + Java Development Kit 5.0 (Sun or IBM)
a. Make sure that your Java Development Kit 5.0 environment is ready.
b. Download Apache Tomcat 5.5 from Apache Tomcat Home and install it
(Suppose the installation directory of Tomcat is {TOMCAT_HOME}).
c. Copy DojoGlobalizationDemo.war to the directory {TOMCAT_HOME}/webapps.

2. (Optional) WebSphere Application Server Community Edition 1.1
1. DB2
a. Create a database in UTF-8 encoding.
b. Create tables and insert data using db.sql under DojoGlobalizationDemo.war/DB/.
c. Get java DB2 driver(db2jcc.jar, db2jcc_license_cu.jar), like in windows, they will be under
C:\Program Files\IBM\SQLLIB\java, and add them to DojoGlobalizationDemo.war/ WEB-INF/lib
d. Update the configuration in db.properties under DojoGlobalizationDemo.war/DB/, for example,
#DB2 configuration
driver=com.ibm.db2.jcc.DB2Driver
url=jdbc:db2://9.181.106.125:50000/DOJODEMO
user=db2admin
password=passw0rd

Or
Derby client mode 10.1.3.1
a. Add derbyclient.jar to DojoGlobalizationDemo.war/ WEB-INF/lib (you can get derbyclient.jar from Derby Home Please choose version 10.1.3.1)
b. Update the configuration in db.properties under DojoGlobalizationDemo.war/DB/, for example,
#client mode derby configuration
driver=org.apache.derby.jdbc.ClientDriver
url=jdbc:derby://localhost:1527/DojoCarStoreDB;create=true
user=user
password=password

2. WebSphere Application Server Community Edition 1.1
a. Download WAS CE from IBM WAS CE Home and install it (suppose the WAS CE installation directory is {WAS_CE_HOME}. In Windows, it may be C:\Program Files\IBM\WebSphere\AppServerCommunityEdition).
b. Launch the WAS CE server from the Windows start menu or {WAS_CE_HOME}\bin\startup.bat.
c. Launch the administrator console at https://localhost:8443/console
d. Click Deploy New on the left Console Navigation pane, locate the DojoGlobalizationDemo.war, leave all the options with default values, and install it.
e. Switch to Web App WARs on the left Console Navigation pane, and make sure that the newly installed DojoGlobalizationDemo/DojoCarStore/0.9/war application is running.

Accessing the sample applications

1. Access Dojo Globalization Hello World by extracting dojo-g11n-helloworld.zip, for example, in to the {Dojo_G11N_HW_DIR} directory, and open the Dojo_G11N_HW_DIR}/HelloWorld.html file in your browser offline.

This section introduces the first Hello World for Dojo Globalization. This Hello World application is pure HTML plus Dojo, therefore you can access it either offline or online. Please refer to Section 2.4 for detailed steps of accessing it.

After you have accessed the Hello World application for Dojo Globalization, you can switch to different locales, you will see the literal changes of Hello World, for example,
en-us:

Screen shot goes here

zh-cn:

Screen shot goes here

2. Access the Dojo Car Store application through http://localhost:8080/DojoGlobalizationDemo/DojoCarStore

3. Access the Dojo Globalization Sample through http://localhost:8080/DojoGlobalizationDemo/DojoG11NSample.html or extract the DojoGlobalizationDemo.war file, for example into the {DojoGlobalizationDemo_DIR}directory, and open the {DojoGlobalizationDemo_DIR}/DojoG11NSample.html file in your browser offline.

Creating the Dojo globalization Hello World sample application

This section introduces the first Hello World for Dojo Globalization. This Hello World application is pure HTML plus Dojo, therefore you can access it either offline or online. Please refer to Section 2.4 for detailed steps of accessing it.

Use the following steps to create an application similar to the Hello World application for Dojo globalization:

1. Include dojo bootstrap – dojo.js, and declare the locales that will be used later.
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}

2. Declare packages for later use.
dojo.require("dojo.date.locale");
dojo.require("dojo.string");
……

3. Register a resource bundle module for the hello world application.
dojo.registerModulePath("my.helloworld","../../helloWorldResource");

4. Actions to take when you switch locales:
- Set global dojo locale.
dojo.locale = lastIndexOf("locale=") >= 0 ?
substring(lastIndexOf("locale=") + "locale=".length) : "en-us"; // default locale is en-us

- Load JSON resource bundle for hello world.
dojo.requireLocalization("my.helloworld", "helloworld");
resourceBundle = dojo.i18n.getLocalization("my.helloworld", "helloworld");

- Format date using dojo.date.locale api, and format template strings using dojo.string api.
dojo.string.substitute(resourceBundle.contentStr,
[dojo.date.locale.format(new Date(), {selector:'date', formatLength:'long'})]);

- Update UI.
dojo.byId('content').innerHTML = ……

Back Button

Dynamic web applications that use things like XMLHTTPRequest and DOM updates instead of page refreshes do not update the browser history, and they do not change the URL of the page. That means if the user clicks the Back button, they will likely jump all the way out of the web application, losing any state that they were in. It is also hard to allow a user to bookmark the web application at a certain state.

Dojo's dojo.back module introduces browser history so that it is possible for the user to click Back and Forward without leaving the web application, and the developer can get notification of these Back and Forward events and update the web application appropriately. Browser history is generated by using a hidden IFRAME and/or adding a unique value to the fragment identifier portion of the page URL. The fragment identifier is the #value thing in a URL. For example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
http://some.domain.com/my/path/to/page.html#fragmentIdentifier
```

Since changing the fragment identifier does not cause the page to refresh, it is ideal for maintaining the state of the application. The

developer can specify a more meaningful value for the fragment identifier to allow bookmarking.

dojo.back allows setting a state object that represents the state of the page. This state object will get callbacks when the Back or Forward button is pressed.

Adding to Your Page

To use dojo.back:

- Define **preventBackButtonFix: false** in your djConfig like this:. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
```
<script type="text/javascript"  src="../dojo.js"
    djConfig="preventBackButtonFix: false">
</script>
```
This allows the hidden iframe to be added to the current page via a document.write(). If you do not do this, dojo.back will not work correctly.
- Add the appropriate require statement: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.require("dojo.back");
```

Register the initial state of the page by calling:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.back.setInitialState(state);
```

This state object will be called when the user clicks Back all the way back to the start of the web application. If the user clicks Back once more, they will go back in the browser to wherever they were before loading the web application.

The state object should have the following functions defined:

- For receiving Back notifications: **back()**, **backButton()** or **handle(type)**, where type will either be the string "back" or "forward".
- For receiving Forward notifications: **forward()**, **forwardButton()** or **handle(type)**, where type will either be the string "back" or "forward".

Example of the a very simple state object:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var state = {
        back: function() { alert("Back was clicked!"); },
        forward: function() { alert("Forward was clicked!"); }
};
```

To register a state object that represents the result of a user action, use the following call:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.back.addToHistory(state);
```

To change the URL in the browser's location bar, include a **changeUrl** property on the state object. If this property is set to true, dojo.back will generate a unique value for the fragment identifier. If it is set to any other value (except undefined, null, 0 or empty string), then that value will be used as the fragment identifier. This will allow users to bookmark the page.

Browser Compatability

- Do not mix dojo.back.addToHistory() calls that use changeUrl with ones that do not use changeUrl. Always use one or the other. If you are using changeUrl, you don't always have to provide a string for the value, but at least set it to **true** to get an auto-generated URL hash.
- Don't test this page using local disk for MSIE. MSIE will not create a history list for iframe_history.html if served from a file: URL. The XML served back from the XHR tests will also not be properly created if served from local disk. Serve the test pages from a web server to test in that browser.
- Safari 2.0.3+ (and probably 1.3.2+): Only the back button works OK (not the forward button), and only if changeUrl is NOT used. When changeUrl is used, Safari jumps all the way back to whatever page was shown before the page that uses dojo.undo.browser support.
- Opera 8.5.3: Does not work.
- Konqueror: Unknown. The latest may have Safari's behavior.

For complete examples, see the unit tests in /dojo/tests/back.html.

Other Functions

In a toolkit as comprehensive as Dojo, the designers had to be vigilant about not duplicating functionality. Virtually all facets of the toolkit need low-level functions - like JSON support, array features, node list processing, etc. And because these functions do not exist in standard JavaScript, or are available only in incompatible calls across browsers, Dojo programmers wrote a lot of low-level code themselves.

Luckily for us Dojo users, they expose these low-level functions. Instead of writing a lot of the annoying little JavaScript utility methods ourselves, we can use Dojo base functions instead. These API's are fast, flexible, and well-thought out. And they are consistent with the higher layers of Dojo's design. Can you dig it?

High-Level Data Format Conversion

Dojo relies on JavaScript objects and arrays for most of its internal communication. Unfortunately, the world doesn't work solely in JavaScript and you must convert data formats into JavaScript before Dojo can use them. Specifically:

- **HTML Forms** keep data for display and entry for the browser. Its contents live in a DOM tree.
- **URL Query Parameters** are used to pass form data to a server via HTTP GET.
- **JSON** is a popular XML alternative optimized for performance, and is often used where you can control the web service.

Dojo, being the diplomat that it is, can convert to and from most of these formats. Why not all? Because strictly speaking, the formats are not equivalent. JavaScript and JSON model arrays with ordered indexes, while HTML forms and URL queries model them as unordered collections. To illustrate:

Source format	Comments	Example

HTML Form	The array b is not ordered, so we don't know if 2 or 3 appears first. Therefore, converting from an ordered array like in JavaScript loses information. Converting to JavaScript is impossible unless we make a "guess" on the ordering. Also, the heirarchy is restricted to two levels - the form "x" at the root, and all other data elements below it, with arrays adding at most one level. You cannot nest any further.	/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
		`<form name="x">`
		`  <input type="text" name="a" value="1">`
		`  <input type="text" name="b" value="2">`
		`  <input type="text" name="b" value="3">`
		`</form>`

URL Query	Same limitations as HTML form.	?a=1&b=2&b=3
JavaScript	Allows arbitrary nesting and grouping. Arrays are ordered.	/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;} `var x = {a: 1, b: [2, 3] };`
JSON	Same limitations as JavaScript.	/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;} `{"x": {"a": 1, "b": [2, 3] }}`

Given those limitations, here are the dojo conversion functions. All are available in Dojo Base, so no required's are necessary.

Source format	To HTML Form	To URL Query	To JavaScript	To JSON
HTML Form	N/A	dojo.formToQuery()	dojo.formToObject()	dojo.formToJson()
URL Query	None	N/A	dojo.queryToObject()	None
JavaScript	None	dojo.objectToQuery()	N/A	dojo.toJson()
JSON	None	None	dojo.fromJson()	N/A

Communication Between Threads - dojo.Deferred

JavaScript has no threads, and even if it did, threads are hard. Deferreds are a way of abstracting non-blocking events, such as the final

response to an XMLHttpRequest. Deferreds create a promise to return a response a some point in the future and an easy way to register your interest in receiving that response.

How Does it Work?

Imagine a whiteboard in a math classroom. One professor knows that the maintenance person for this particular classroom is a math genius. So the professor writes on the board a question:

> "What is the Shortest Proof for Fermat's Last Theorem? Call x.2591 when you have the answer."

That night, the maintenance worker finds the question on the whiteboard. It's an interesting problem. So he continues to mop the floors thinking about it. (Uhh, does this sound like a movie with Ben Affleck and Matt Damon? Could be...)

Now suppose students would also like the answer. (It'd make a helluva term paper!) They scribble notes underneath like "Also call 555-8244. Also email mathlunkhead@aol.com," etc.

Finally, 8 nights later, the maintenance worker writes down the answer on the whiteboard and calls x.2591, 555-8244, and emails "mathlunkhead@aol.com".

dojo.Deferred is like the whiteboard. At least one person has a question that needs answering by some other entity. In Web 2.0 applications, this is often a server process called by XmlHTTPRequest. Unfortunately, we don't know when the answer will come back. We can either call the process synchronously (we wait by the whiteboard for the answer) or asynchronously (we leave the room and asked to be called).

The email addresses and phone numbers symbolize *handlers.* A handler is simply a Javascript function called when the answer is complete. These are split between *callbacks* and *errbacks* which handle normal completion and errors respectively.

Using Deferreds

The most important methods for Deferred users are:

- addCallback(handler)
- addErrback(handler)
- callback(result)
- errback(result)

In general, when a function returns a Deferred, users then "fill in" the second half of the contract by registering callbacks and error handlers. You may register as many callback and errback handlers as you like and they will be executed in the order registered when a result is provided. Usually this result is provided as the result of an asynchronous operation. The code "managing" the Deferred (the code that made the promise to provide an answer later) will use the callback() and errback() methods to communicate with registered listeners about the result of the operation. At this time, all registered result handlers are called **with the most recent result value**.

Deferred callback handlers are treated as a chain, and each item in the chain is required to return a value that will be fed into successive handlers. The most minimal callback may be registered like this:

```
var d = new dojo.Deferred();
d.addCallback(function(result){ return result; });
```

Perhaps the most common mistake when first using Deferreds is to forget to return a value (in most cases, the value you were passed).

The Deferred also keeps track of its current status, which may be one of three states:

- -1: no value yet (initial condition)
- 0: success
- 1: error

A Deferred will be in the error state if one of the following three conditions are met:

1. The result given to callback or errback is an object whose class or superclass is Error, e.g. "instanceof Error" is true.,
2. The previous callback or errback raised an exception while executing
3. The previous callback or errback returned a value whose class or superclass is "Error"

Otherwise, the Deferred will be in the success state. The state of the Deferred determines the next element in the callback sequence to run.

When a callback or errback occurs with the example deferred chain, something equivalent to the following will happen (imagine that exceptions are caught and returned):

```
 d.callback(result) or d.errback(result)
if(!(result instanceof Error)){
    result = myCallback(result);
}
if(result instanceof Error){
    result = myErrback(result);
}
result = myBoth(result);
if(result instanceof Error){
    result = myErrback(result);
}else{
    result = myCallback(result);

}
```

The result is then stored away in case another step is added to the callback sequence. Since the Deferred already has a value available, any new callbacks added will be called immediately.

There are two other "advanced" details about this implementation that are useful:

- Callbacks are allowed to return Deferred instances themselves, so you can build complicated sequences of events with ease.
- The creator of the Deferred may specify a canceller. The canceller is a function that will be called if Deferred.cancel is called before the Deferred fires. You can use this to implement clean aborting of an XMLHttpRequest, etc. Note that cancel will fire the deferred with a CancelledError (unless your canceller returns another kind of error), so the errbacks should be prepared to handle that error for cancellable Deferreds.

Deferred objects are often used when making code asynchronous. It may be easiest to write functions in a synchronous manner and then split code using a deferred to trigger a response to a long-lived operation. For example, instead of register a callback function to denote when a rendering operation completes, the function can simply return a deferred:

callback style:

```
function renderLotsOfData(data, callback) {
   var success = false;
   try{
      for(var x in data) {
         renderDataitem(data[x]);
      }
      success = true;
   }catch(e){ }
   if(callback){
      callback(success);
   }
}
```

using callback style:

```
renderLotsOfData(someDataObj, function(success){
   //handles success or failure
   if (!success){
      promptUserToRecover();
   }
});
```

NOTE: there's no way to add another callback here!!

Using a Deferred doesn't simplify the sending code any, but it provides a standard interface for callers and senders alike, providing both with a simple way to service multiple callbacks for an operation and freeing both sides from worrying about details such as "did this get called already?". With Deferreds, new callbacks can be added at any time.

Deferred style:

```
function renderLotsOfData(data, callback){
   var d = new dojo.Deferred();
   try {
      for (var x in data) {
         renderDataitem(data[x]);
      }
      d.callback(true);
   } catch(e) {
      d.errback(new Error("rendering failed"));
   }
   return d;
}
```

using Deferred style

```
renderLotsOfData(someDataObj).addErrback(function(){
   promptUserToRecover();
});
```

NOTE: addErrback and addCallback both return the Deferred again, so we could chain adding callbacks or save the deferred for later should we need to be notified again.

In this example, renderLotsOfData is syncrhonous and so both versions are pretty artificial. Putting the data display on a timeout helps show why Deferreds rock:

Deferred style and async func:

```
 Deferred style
function renderLotsOfData(data, callback){
    var d = new dojo.Deferred();
    setTimeout(function(){
        try{
            for(var x in data){
                renderDataitem(data[x]);
            }
            d.callback(true);
        }catch(e){
            d.errback(new Error("rendering failed"));
        }
    }, 100);
```

```
    return d;
}

 using Deferred style
renderLotsOfData(someDataObj).addErrback(function(){
    promptUserToRecover();
});
```

Note that the caller doesn't have to change his code at all to handle the asynchronous case.

Author: Alex, with Good Will Hunting analogy provided by Craig Riecke (and Ben and Matt)

Cookies

The dojo.cookie module has one method, dojo.cookie, a one-stop shop for all your HTML cookie needs.

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}
<head>
<title>Cookie Demo</title>
    <script type="text/javascript"
            src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"></script>
    <script type="text/javascript">
        dojo.require("dojo.cookie");
        dojo.addOnLoad(function() {
            // Calling dojo.cookie with two parameters sets a cookie
            dojo.cookie("online-book.cookie","oatmeal");

           // With one parameter, reads the cookie
           console.debug("cookie is "+dojo.cookie("online-book.cookie"));

           // Third parameter is an object with other cookie options.  Here, setting the expires
           // property to -1 deletes the cookie
           dojo.cookie("online-book.cookie","Value Doesn't matter",{expires: -1});
           console.debug("cookie should now be null: "+dojo.cookie("online-book.cookie"));
        });
    </script>
</head>
```

Rumor has is that Brad Neuberg calls cookies "Dojo Offline Lite."

Array Functions

Along with dojo.forEach, Dojo provides other array functions which mimic JavaScript 1.6 functionality. And that's good news for IE users, who don't have the luxury of JavaScript 1.6 yet.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Array Examples</title>
        <script type="text/javascript"
            src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"></script>
        <script type="text/javascript">
       dojo.addOnLoad(function(){
           // JavaScript is so important, it's listed twice!
           var webLanguages = ["JavaScript", "PHP", "Rails", "JavaScript", "JSP", "ASP.NET"];

           // indexOf and lastIndexOf are like the String functions
           console.debug(dojo.indexOf(webLanguages, "JavaScript")); // = 0
           console.debug(dojo.indexOf(webLanguages, "Javascript")); // = -1 - case sensitive!
           console.debug(dojo.lastIndexOf(webLanguages, "JavaScript")); // = 3

           // Apply function to each element and return a similarly shaped array
           console.dir(
               dojo.map(webLanguages,
                  function(elem) { return elem.length; }
               )
           );   // Returns [10, 3, 5, 10, 3, 7]

           // Filter applies a boolean function and returns array of elements that match
           console.dir(
               dojo.filter(webLanguages,
                  function(elem) { return elem.substring(0,1) == 'J'; }
               )
           );   // Returns ["JavaScript", "JavaScript", "PHP"]

           // Every and some apply a boolean function to each element and returns true
           // if function is true for all (every) or at least one (some)
           console.debug(
               dojo.every(webLanguages,
```

```
                function(elem) { return elem > 'BASIC'; }
            )
        );   // Returns false because ASP.NET is less than
             // BASIC (technical arguments aside...)
        console.debug(
            dojo.some(webLanguages,
                function(elem) { return elem.length < 4; }
            )
        );   // Returns true because length of JSP and PHP are less than  4
    });
  </script>
</head>
</html>
```

Checking Object Types

JavaScript's "instanceof" operator checks an object for class information. But there are nuances that make it difficult in practice. For example, a DOM NodeList can be accessed like an array with [] subscripts, but "instanceof Array" applied to a NodeList returns false nonetheless. These Dojo base functions shield those nasty details from you.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Type Checking Demo</title>
        <script type="text/javascript"
                src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"></script>
        <script type="text/javascript">
            //
            dojo.addOnLoad(function() {
                var allScripts = dojo.query("script");
                var iceCreamObject = { flavor: "vanilla", scoops: 2};

                    // NodeLists are like arrays
                    console.debug(dojo.isArray(allScripts));  // True
                    // An arbitrary object.  Arrays are objects too.
                    console.debug(dojo.isObject(iceCreamObject));  // True
                    console.debug(dojo.isObject(allScripts));  // True

                    console.debug(dojo.isFunction(dojo.query)); // True
                    console.debug(dojo.isFunction(dojo)); // False

                    //  This next example hurts my head!
                    console.debug(dojo.isFunction(dojo.isFunction));  // True, but weird

                    // This is a common idiom for doing "nullable" arguments in the Dojo source code.
                    // Here, you can pass either myFn(var, function, var) or myFn(var, var);
                    var myFn = function() {
                       if (dojo.isFunction(arguments[1])) {
                          return arguments[1](arguments[0], arguments[2]);
                       } else {
                          return arguments[0]+arguments[1];
                       }
                    }
                    console.debug(myFn(1,2));  // 3
                    console.debug(myFn(1,Math.max,2));  // 2 = Math.max(1,2);
        });

    </script>
</head>
</html>
```

Date Functions

The date functions in Dojo are like those little wrapped mints on your pillow. Surprising, little and fabulous!

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Date Examples</title>
        <script type="text/javascript"
                src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"></script>
        <script type="text/javascript">
           dojo.require("dojo.date");
           dojo.require("dojo.date.stamp");

       dojo.addOnLoad(function(){
           // We'll demonstrate these functions on today's date, so you'll have to think
           // for a minute to verify the functions work!
           var today = new Date();
           console.debug("today is "+today);

           // Is this year a leap year?
           console.debug("leap year: "+dojo.date.isLeapYear(today));

           // How many days in this month?
```

```
            console.debug("days in month: "+dojo.date.getDaysInMonth(today));

            // Convert to/from ISO format
            var dojo0_9Release = dojo.date.stamp.fromISOString("2007-08-20");

            // Do some arithmetic.
            console.debug(
                dojo.date.difference(dojo0_9Release, today)+
                " days since Dojo 0.9 release"
            );

            var oneYearAnniversary = dojo.date.add(dojo0_9Release,"year",1);
            var isOneYearYet = dojo.date.compare(oneYearAnniversary, today);
            if (isOneYearYet > 0) {
                    console.debug(
                        dojo.date.difference(today, oneYearAnniversary)+
                        " days until one year anniversary of 0.9 release"
                    );
                }

        });
    </script>
</head>
</html>
```

String Functions

Dojo includes some string operations. Here are a few examples:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Array Examples</title>
        <script type="text/javascript"
                src="http://o.aolcdn.com/dojo/1.0.0/dojo/dojo.xd.js"></script>
        <script type="text/javascript">
            dojo.require("dojo.string");

        dojo.addOnLoad(function(){
            // Pads * onto the left hand side
            console.debug(dojo.string.pad("9.99",10,"*"));

            // With true on the end, pads on the right
            console.debug(dojo.string.pad("Ninety-Nine and 75/100",25,"-",true));

            // Dojo has two versions of trim, functionally equivalent.  The first
            // is in base so you don't have to dojo.require anything.
            console.debug("[" + dojo.trim("    spaced    ") + "]");

            // The second is dojo.string you need to dojo.require("dojo.string"), but
            // it's faster.
            console.debug("[" + dojo.string.trim("    spaced    ") + "]");

            // Substitute does simple ${var} template substitution, much like you see in
            // _Templated widgets.  You can use it with arrays:
            console.debug(
                dojo.string.substitute(
                    "${2}ft ${0} cable - ${1}",
                    [ "red", "fiber", 7]
                )
            );

            // Or, even better, with objects:
            console.debug(
                dojo.string.substitute(
                    "${length}ft ${color} cable - ${media}",
                    {color: "red", media: "fiber", length: 7}
                )
            );

            // Even nested objects, using dot notation
            console.debug(
                dojo.string.substitute(
                    "Ooops, spilled ${condiment.ketchup}",
                    {burger: "boca",
                        condiment: { ketchup: "heinz", mustard: "french's"},
                     bun: "wonder"}
                )
            );
        });
    </script>
</head>
</html>
```

Part 4: Testing, Tuning and Debugging

Up until now, we've been looking directly at Dojo and Dijit. Now we'll shift gears for a bit and talk about stuff outside the code itself. When you work with a rich toolkit like Dojo to build Enterprise-grade apps, the old JavaScript development paradigms aren't so hot:

- Debugging with alert() is tedious and inflexible
- Testing by eyeballing pages requires too much manual recordkeeping
- Notepad just doesn't cut it as an IDE!

- Development "frills" like whitepace and comments now get in the way of performance. The more you do, the more bytes to download to the client, and the slower the load time

In Part 4, we'll look at ways to combat all of these issues. The Dojo people aren't content with dropping the API in your lap and running. Oh no! In the course of developing Dojo itself, they have built tools and techniques to help you, and they're very generous in sharing them. So let's have a look.

Getting the Code from Source Control

The Dojo team uses Subversion for source control. Those familiar with CVS will find the command line syntax for subversion to be very similar. Regardless, the following instructions are geared to those not familiar with CVS, or even source control in general.

Authoritative documentation on Subversion is available here.

Why Source Control?

Source control is one of those things that one rarely notices they need until it's far too late, usually when you accidentally delete part of your source tree instead of simply moving it to a different location, or when you make a set of complex changes that leave you worse off than were you started (but you can't go back). A source control system solves these problems by keeping copies of each revision of a set of files on a server, while giving you access to a local copy of those files to make changes on. Good source control systems allow multiple people to modify a single file at once, and will try to automatically merge changes between differing sets of modifications. A good source control system will also let you browse the history of a file or set of files (allowing you to "go back in time") and allow you to have acccess to your code from as many systems as you like. Subversion is one of those good source control systems.

General Information

Across the gamut of source control systems, there is quite a bit of confusing (and non-portable) nomenclature surrounding the common actions that you as a developer will preform with the source control system. We will use the terms here that are commonly accepted by CVS and Subversion users and administrators. So what are those terms?

- **Checkout**: a "checkout" is a local working copy of a "repository". This is the set of files that you will be working with when making modifications to Dojo.
- **Repository**: the logical grouping of project-related files on the server. A subversion server may host multiple repositories, but it is quite likely that your changes will be constrained to a single repository.
- **Checkin**: transmitting a set of changes from your local checkout to the server. Your changes will then be available to everyone else who has a checkout of that repository when they update their view.
- **Head**: (also called the "main line") the most up-to-date version of the source control tree. Most of your checkins will be to the "head" of the tree, although in some more complex situations, you may be checking in to a "branch".
- **Branch**: A clone of the source control repository from some point in time which contains a set of changes which are not shared with the "head" or main line. Branches are one way for a developer to work on a particular feature (usually a large feature) and have intermediate changes versioned without having to worry about whether or not his or her changes will break someone else's code. Changes can then be "merged" back back to the main line when the developer thinks they are stable.
- **Merge**: merging is the process of taking several versions of a single file and turning them into one authoritative version. Merging in is often an automated process with Subversion, but you may at times be called upon to merge a set of files manually (when the server cannot automatically take care of it).

Unlike some other source control systems, Subversion manages files on your disk without interjecting itself obtrusively into your workflow. You can change large sets of files
without worrying if anyone else is also modifying those files.

Browsing the repository

You may browser our subversion repository (http://svn.dojotoolkit.org/dojo/) directly using a web browser or browse using the trac interface.

Directory structure

The code is structured with each subproject (dojo, dijit, dojox, util) as a separate directory. Within each subproject you will find the "trunk" as well as "branches" and "tags". You may pull these subprojects separately, or use a special svn "view" which links the subprojects and checks them out with a single command.

Note that the "trunk" directory at the top-level is obsolete. This was used prior to the 0.9 release, when the code was reorganized into the various subprojects.

To do an anonymous, read-only checkout of the Dojo development trunk:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .bash .imp {font-weight: bold; color: red;} .bash .kw1 {color: #b1b100;} .bash .kw3 {color: #000066;} .bash .es0 {color: #000099; font-weight: bold;} .bash .br0 {color: #66cc66;} .bash .st0 {color: #ff0000;} .bash .nu0 {color: #cc66cc;} .bash .re0 {color: #0000ff;} .bash .re1 {color: #0000ff;} .bash .re2 {color: #0000ff;} .bash .re3 {color: #808080; font-style: italic;} .bash .re4 {color: #0000ff;} `svn co http://svn.dojotoolkit.org/dojo/view/anon/all/trunk dojodev`

Or, to pull a particular release, such as Dojo 1.0.2:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .bash .imp {font-weight: bold; color: red;} .bash .kw1 {color: #b1b100;} .bash .kw3 {color: #000066;} .bash .es0 {color: #000099; font-weight: bold;} .bash .br0 {color: #66cc66;} .bash .st0 {color: #ff0000;} .bash .nu0 {color: #cc66cc;} .bash .re0 {color: #0000ff;} .bash .re1 {color: #0000ff;} .bash .re2 {color: #0000ff;} .bash .re3 {color: #808080; font-style: italic;} .bash .re4 {color: #0000ff;} `svn co http://svn.dojotoolkit.org/dojo/tags/release-1.0.2 dojo102`

Branches in the Dojo repository

Most Dojo development takes place on the trunk. Branches may be used for development of experimental features or for code migration before being merged back into the trunk. Branches are also used to stabilize major releases.

Making changes to the repository

Anyone may access the Dojo Subversion server. Contributors are encouraged to access code directly from the repository and submit patches using the bug tracker. To submit changes, however, you must have committer status and have already received a system account from the administrator. Once your system account is created, please test your account by logging in over SSH. The Dojo team uses SSH (the Secure Shell) to encrypt access to our source control system because Unsecure Network Logins Suck (TM).

To access the repository as a committer, a different url scheme must be used with svn to provide authentication:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .bash .imp {font-weight: bold; color: red;} .bash .kw1 {color: #b1b100;} .bash .kw3 {color: #000066;} .bash .es0 {color: #000099; font-weight: bold;} .bash .br0 {color: #66cc66;} .bash .st0 {color: #ff0000;} .bash .nu0 {color: #cc66cc;} .bash .re0 {color: #0000ff;} .bash .re1 {color: #0000ff;} .bash .re2 {color: #0000ff;} .bash .re3 {color: #808080; font-style: italic;} .bash .re4 {color: #0000ff;} `svn co svn+ssh://username@svn.dojotoolkit.org/var/src/dojo/view/committer/all/trunk dojodev`

Using Development Tools

Setup for Eclipse

If you use Eclipse, we recommend the Subclipse plugin. Simply provide a URL for the Dojo repository as discussed above.

Setup for Windows

To access our subversion repository on Windows, please install TortoiseSVN, available at: http://tortoisesvn.tigris.org/ Installing Tortoise requires a reboot since it installs itself as a Windows Explorer shell extension.

Welcome back! Now that you have rebooted, open up a Windows Explorer window (Win-E, or right-click on the "Start" menu and select "Explore"), navigate to whatever directory you would like your Dojo source code to be placed under. Create a new directory there titled "dojo", and then navigate to it. Once inside the directory, right click on the emply file list and select TortoiseSVN->Settings. In the resulting dialogue box, got to the "Network" tab.

Using TortoiseSVN with ssh is a right pain because it repeatedly asks for the password again and again. If you can use an anonymous checkout that may be better.

NOTE: in the latest version (1.3.3) leaving the ssh client field blank worked, and setting it to TortoisePlink.exe didn't (Even in double quotes). Previously: As of version 1.1.x, you must manually configure Tortoise to be aware of it's SSH client program, which is usually located in:

C:\Program Files\TortoiseSVN\bin\TortoisePlink.exe

Use the "Browse..." button at the bottom of the "Network" tab in order to locate and select "TortoisePlink.exe". With that done, you can hit "OK" in the configuration tab and return to the "dojo" directory we created earlier.

Right-click in the directory and select "SVN Checkout..." from the context menu. In the resulting dialog box, you will be prompted for a URL for the repository you want to check out. Subversion supports multiple access methods, but we will use Subversion over SSH to get to our repo. In this dialog box, place the following URL (replacing your unix system login for "username"):

Dojo development trunk
svn+ssh://username@svn.dojotoolkit.org/var/src/dojo/view/committer/all/trunk

TortoiseSVN

Using TortoiseSVN under Windows, here's how to do it:

1. If you aren't using PuTTY, go grab it.
2. Create a new session in PuTTY:
    a. in the Host Name, type "username@dojotoolkit.org", where username is your username.
    b. check the SSH radio button; the port number should be 22.
    c. In the saved sessions box, type "svn.dojotoolkit.org" and save the session.
3. Create your checkout using that view url above.

The saved session in PuTTY will be picked up by Plink, and you'll be asked for your password just like with a regular checkout.

Click "ok" in the dialog box if a dialog box comes up discussing a host key. You will then be asked (two or three times) for your unix user login password at dojotoolkit.org. Provide it. You will then see a list of files being retreived from the server, and when it's finished, you will have your very own checkout!

Next, take a minute to update your svn config settings.

You can then check in your changes by right-clicking on the file(s) you want to check in, selecting providing a checkin comment (strongly encouraged). Other operations, including diffing and merging are also available from the context menu.

For more information about TortiseSVN or how to use it, see the documentation at:

http://tortoisesvn.tigris.org/docs/TortoiseSVN_en/index.html

Setup for Linux

Since you're running Linux, it is assumed that you're comfy with your systems package management system and the command line.

Firstly, you will need to install an SSH client (preferably OpenSSH) and a Subversion client library (through apt, yast, or whatever package management system your system supports).

Making a checkout is straightforward from the command line. Provided you already have Subversion installed, simply run:

Dojo development trunk

```
svn co svn+ssh://username@svn.dojotoolkit.org/var/src/dojo/view/committer/all/trunk dojodev
```

Next, take a minute to update your svn config settings.

This will create a local copy ("checkout") of the source tree (under a new directory called "dojo") in the current directory. Make edits to the files you want to change, then commit them back to the repository with:

```
svn commit -m "Commit message here" names/of/files
```

Setup for OS X

We assume that you are on at least OS 10.3 (Panther).

Developers on OS X may already have an SSH client installed, but may need to install the Developer Tools package in order to get the most up-to-date JDK and Ant packages. It is assumed that operations will be preformed at the command line, and Project Builder/XCode configuration is not covered here. It is, however, recommended that you download the latest set of developer tools from http://connect.apple.com (free registration required).

Once you have the OS X developer tools installed, download the latest 1.x Subversion package (1.1.3 as of this writing) from: http://metissian.com/projects/macosx/subversion/

The downloads are an OS X installer package. Install it, at which point the instructions from the Linux section will be sufficient to get you up and running.

Next, take a minute to update your svn config settings.

SVN Config Settings:

You'll need to add a couple config settings to your SVN config file. If you are on Windows, that's located at:

C:\Documents and Settings\YourUserName\Application Data\Subversion\config

And on UNIX/Mac OS X:

~/.subversion/config

Open it up in your favorite text editor. Most configs have some default settings, so locate **[miscellany]** and **enable-auto-props** in the file. If they exist, make sure that they are uncommented (remove **#** from beginning of line), otherwise add them. They line should look like:

```
[miscellany]
enable-auto-props = yes
```

Next, located **[auto-props]**. If it doesn't exist, add it, otherwise you'll probably have to uncomment it. Add the following entries below **[auto-props]**:

```
*.js = svn:eol-style=native
*.htm = svn:eol-style=native
*.html = svn:eol-style=native
*.svg = svn:eol-style=native
*.txt = svn:eol-style=native
*.xml = svn:eol-style=native
*.css = svn:eol-style=native
*.rest = svn:eol-style=native
Makefile = svn:eol-style=native
README = svn:eol-style=native
CHANGELOG = svn:eol-style=native
LICENSE = svn:eol-style=native
```

Save that and you should be set! Continue on to committing directions above.

Development Tools

No matter what your development environment, several tools are indispensable when building client-side applications:

- FireFox

- FireBug is a web developer's best friend. Also see it's associated performance tuning plugin YSlow.
- Virtualization software such as VMWare or Parallels can make your job significantly easier by allowing you to run multiple versions of IE side-by-side
- Safari (for Windows and Mac) and the WebKit nightly builds are great debugging aids.

Integrated Development Environments

At the high end of the spectrum, Integrated Development Environments, or IDEs support a wide range of web development activity, including editing JavaScript and HTML files, deploying code to servers, and integration with existing features like source control. In addition, some include runtime tools and browser integration to assist in debugging.

Cross-Platform, Eclipse-Based

- Aptana - Extremely strong JavaScript support, good support for many common server-side languages, and integration for Dojo.
- MyEclipse - MyEclipse Professional contains a JavaScript editor and debugger. The debugger can set breakpoints, inspect variables, and single step within its own Web 2.0 Browser, based on Mozilla/Firefox.
- ATF - The Ajax Tools Framework contains similar tools as MyEclipse, and adds AJAX toolkit "personalities". These are essentially snippets for the common idioms for a toolkit. The Dojo personality is included, but is not yet updated for 0.9.
- Adobe JSEclipse, formerly a product of Interakt, is a free JavaScript editor. Like ATF, it includes code completion idioms for Dojo, but does not include the debugger.

Windows

- Visual Studio .NET bundles the JavaScript editor and debugger with the other languages. This is especially useful when writing data services in ASP.NET and client-side scripts with Dojo.
- Visual Web Developer Express is a free, scaled down version of the web portion of Visual Studio. JavaScript editing and debugging is supported.
- Microsoft Script DebuggerAn extremely scaled-down and outdated debugger, but familiar to many.

Browser Addons

Mozilla/Firefox

- The Firebug is the debugger of choice for Dojo and Firefox. Consult Cheap Debugging for details.
- JavaScript Debugger, a/k/a Venkman has been around for years, and has adherents. Venkman seems to be dormant, gaining no new features but being ported verbatim to new versions of Firefox.
- Live HTTP Headers is useful for debugging HTTP traffic . Although Firebug is good for analyzing HTTP traffic within the page, as in XHR, Live HTTP Headers excels at between-page HTTP requests.
- The Web Developer Toolbar is a favorite of Web Designers, and includes rulers and positioning features that Firebug does not.

Internet Explorer

- IE Web Developer is a commercial product with Firebug-like features such as JavaScript debugging.
- IE Developer Toolbar, a free download from Microsoft, is useful for page design.
- The Microsoft Script Editor integrates with Internet Explorer and is available as part of office 2003. It is not included in the default install but you can get it from the installation cd's.
- Web Developer Helper - HTTP logging, script debugging and a DOM inspector.

Safari

Drosera is the "native" JavaScript debugger for Safari, and is written by the folks from Webkit, on which Safari is based. The Safari Developer FAQ has some general information about developing with Safari, as well as instructions on how to turn on a debug menu that allows showing a JavaScript console.

Traffic Analyzers

Many problems can be resolved by watching the traffic between the browser and the server. While IDE's and web add-ons analyze traffic, the following standalone programs are more heavy-duty. If you are having any problems with XHR or with js/html/css/jpg files not loading as you would expect, this is often the best way to diagnose them quickly. Also if you are having problems with required files not loading then this is a good starting point to find out why.

- Fiddler is a fantastic HTTP header and content inspector for Windows. It understands HTTP and presents the information very clearly (once you get used to its slightly quirky interface). Well integrated with IE, but work s with any browser. With FF a useful extension to help use Fiddler is the Switch Proxy extension (scroll down that page to find it).
- Wireshark (formerly Ethereal) is great for sniffing all kinds of network traffic.

Debugging Facilities

A great toolkit like Dojo deserves great tools alongside it. If you're used to chasing down problems with alert() and trial-and-error, it's time to give your development environment an upgrade!

Dojo itself has facilities for easing debugging tasks: an isDebug flag for better tracking down internals and console.debug() statements for logging. We'll cover these in the next sections, plus give you some hints on general JavaScript debugging.

Basic JavaScript Gotchas

Developing AJAX based applications, whether you use the Dojo Toolkit or develop everything the hard way from scratch, can pose quite a challenge to most developers. The difficulties typically arise from two main issues:

- JavaScript(TM) is an interpreted language and is not compiled into machine code. Therefore, you do not have a compiler to catch common syntax errors.
- Various Web browsers report errors in different ways. For example, Mozilla Firefox(TM) logs JavaScript errors into its JavaScript console. Microsoft(TM) Internet Explorer might bring up an alert window with the line and error message.

This article presents common problems and how currently available tools might be used to debug them. This section is not intended as an all-encompassing debugging guide for JavaScript problems.

Debugging Issue 1: No Compiler Syntax Checking

Developers who have worked primarily with compiled languages will miss the compiler when working with a completely interpreted language like JavaScript. More often than not, the browser's error message is far from helpful in determining the actual root cause of the error. In addition, the errors reported by many AJAX toolkits are often not very helpful in determining the root cause of the error. For example, if you have written a custom widget based on the Dojo Toolkit that contains a syntax error and attempt to load it, you might get the following error: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
FATAL exception raised: Could not load
'some.package.name.WidgetName'; last tried '__package__.js'
```

Unfortunately, the message does not tell you what the problem was, only that it could not load a particular widget. This error can be caused by many programming errors, but the most common ones are syntax errors in the widget JavaScript. The most common syntax errors that cause the above error are as follows:

- Using ';' instead of ',' when separating attributes of a JavaScript object definition. This error is extremely common for novice JavaScript developers who have a background in Java or C++.

  Note: All Dojo Toolkit widgets and extension widgets are JavaScript *objects* and therefore any custom widgets must use the proper comma separator between properties of the object (these include function declarations as properties).

- Incorrectly terminating the last attribute definition in a JavaScript object definition with ','. The last property in any JavaScript object definition must never end with a comma.
- Malformed (unmatched) braces {} .
- Malformed function definitions.

You can avoid syntax errors by running a lint processor that understands JavaScript syntax on your code as you develop it. The lint processor will check for syntax errors before the code is run. Some of the JavaScript lint processors available on the web include:

- http://www.jslint.com/
- http://www.aptana.org/

Most lint processors are excellent at catching the errors mentioned previously and more. Many of the lint processors also enforce that JavaScript program statements always line terminate with ';' . While terminating with ';' is considered optional in JavaScript, it is highly recommended to do it for strictness and uniformity.

In summary, using lint syntax checking will save you hours of development time and frustration when trying to find the misplaced comma, malformed function, or missing closure.

Debugging Issue 2: Runtime Problems.

Assuming that all the syntactical errors were caught, by running a lint processor on the JavaScript code, and corrected the next set of problems are all related to runtime problems. These issues tend to be tricker to debug, as the various browsers do not report errors in the same way. Unfortunately, this means that, as a JavaScript developer, you must become familiar with debugging tools and browser error reporting mechanisms that are specific to each of the major supported browsers on which your application will be used. The list below outlines how some of the more well-known browsers report errors:

Microsoft Internet Explorer

Errors, such as accessing an undefined reference, are reported as a pop up from the browser with the line number and a simple description of what it believes the error is.

Mozilla 1.7.X and Firefox 1.5.X

The Mozilla and Firefox browsers report all JavaScript errors into a 'JavaScript Console'. So, often on a page that encountered JavaScript problems, the Web application just won't work and it's not always obvious why. So, while developing and testing using these browsers, you should regularly open and check the JavaScript console. This console can be located from the following menu locations:

- **Mozilla 1.7.X:** Tools->Web Development->JavaScript Console.
- **Firefox 1.5.X:** Tools->JavaScript Console.

The most common problems encountered at runtime tend to be:

- Referencing undefined variables. For example, trying to access an attribute on a currently undefined object, such as jsObject.someAttribute, where the jsObject variable has not been assigned.

- Functions defined on a JavaScript object accessing properties on that JavaScript object and failing to use the 'this' reference identifier. For example, the following is incorrect:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
function testOnload() {
       var fooObject = {
            fooAttribute: "This is foo",
            fooFunction: function() {
                alert("The value of foo is: [" + fooAttribute + "]");
            }
        };
        fooObject.fooFunction();
    }
dojo.addOnLoad(testOnload);
```

  This is the correct way:

  /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
function testOnload() {
       var fooObject = {
            fooAttribute: "This is foo",
            fooFunction: function() {
                alert("The value of foo is: [" + this.fooAttribute + "]");
            }
        };
        fooObject.fooFunction();
    }
dojo.addOnLoad(testOnload);
```

Using JavaScript Debuggers

Debugging with the Dojo loader

The dojo.require() method of the Dojo loader uses a combination of XHR plus the eval statement to load JavaScript. This is one of the most powerful features in Dojo, however, it raises a few problems with some tools. First, most tools have no way to identify the source of the code. The information about the file or URL source of the buffer passed to eval in the Dojo loader is lost. (Note: the loader puts this metadata at the very bottom of the buffer, which may be inspected or parsed by some tools) Second, the Mozilla/Firefox console as well as most debuggers based on Mozilla's Spidermonkey engine suffer from a serious bug which makes debugging Dojo nearly impossible: all code loaded through an eval statement are identified at the location of the eval statement itself instead of the eval buffer. Not only are you presented with the wrong buffer for the stack frame, but the line number is also incorrect -- it is calculated as the offset of the eval command in the code plus the offset of the line within the eval buffer itself.

Neither Firebug nor Venkman work very well given the Spidermonkey limitation. Thankfully, there is an experimental patch available to Firebug which addresses both of these problems. Unitl then, setting debugAtAllCosts in djConfig can help better identify the error. Alternatively, you can [SCRIPT] include the JavaScript files directly on the page, but that can be very tedious.

Firefox Safe Mode

If you are having weird problems with Firefox, it is often worthwhile running Firefox in safe mode. This is because installed extensions can interfere with the DOM tree, CSS, or even with javascript. In Windows there is a shortcut to start Firefox in safe mode from within the Mozilla Firefox folder, from the Start button.

Javascript debugger statement

Javascript has a debugger keyword that forces a breakpoint to occur. Just insert debugger; and if you have a debugger for your browser, then it will stop at the debugger keyword.

This is especially useful when using Venkman, because otherwise it can be difficult to get Venkman into debugging mode (e.g. you try to click on a line of source and it puts in a [F] future breakpoint).

It also works with Firebug and most other JavaScript debuggers.

There's also a lint program which catches a variety of problems in source files, but also flags certain patterns which are valid JavaScript, but considered by the author to be bad style.

Profiling

Performance problems are bugs too, and Dojo has built-in facilities for identifying slow-running code segments. Some JavaScript debuggers like Firebug give file-loading performance, but often you need information at the code level.

You can cheaply calculate function execution time by taking the difference between two JavaScript dates. It's not a great technique, because you have to make educated guesses as to where the inefficient areas are, and it may take quite a lot of work to home in on the problem area. But it is easy to do if you are measuring a single function:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0

{color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var startTime = new Date();
// call your function here
console.debug("Total time: " + (new Date() - startTime));
```

Another example of code to do this is on this page.

D.O.H. Unit Testing

Dojo (from 0.9 onward) features a from-scratch unit testing harness, D.O.H., that's both flexible, easy to use, and portable. D.O.H. is designed to run in both browser and command line environments and unlike it's predecessor doesn't need to be run from the build system. D.O.H. has no Dojo dependencies, but can use the Dojo package system if it's available, thereby allowing tests to be run without need of the build tools at all. D.O.H. also includes a browser-based test runner which can execute command-line tests as well as browser-specific tests. When run from the command line, only pure-JS test are run.

Test Registration

D.O.H. defines one global object (doh.*) and code building tests can use APIs defined on it to pass in a test, a group of tests, or a URL to pull a test group from:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
doh.registerTest(group, testFuncOrObj);
doh.registerTests(group, testFuncOrObjArr);
doh.registerTestNs(nsObj, objName);
doh.registerTestUrl(url);
doh.register(...);
```

The tests.register() method accepts the function signatures of any of the other registration functions and determines the correct underlying function to dispatch registration to.

The contents of a typical, command-line-only, test file might look something like: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// file located in core at:
//      tests/moduleToBeTested.js
dojo.provide("tests.moduleToBeTested");
dojo.require("doh.runner");
doh.register("tests.moduleToBeTested",
        [
                // single test, no test fixture
                function assertTrueTest(){
                        doh.assertTrue(true);
                        doh.assertTrue(1);
                        doh.assertTrue(!false);
                },
                // a test fixture
                {
                        name: "thingerTest",
                        setUp: function(){
                                this.thingerToTest = new Thinger();
                                this.thingerToTest.doStuffToInit();
                        },
                        runTest: function(){
                                doh.assertEqual("blah", this.thingerToTest.blahProp);
                                doh.assertFalse(this.thingerToTest.falseProp);
                                // ...
                        },
                        tearDown: function(){
                        }
                },
                // ...
        ]
);
```

In this example, we see a variant of the test system that uses the doh.addTests() style of add() and registers both independent tests and fixture-driven tests. Note that we give the functions that are registered names even though we could easily provide anonymous functions. This allows the system to more correctly report on what went wrong and where, allowing you to talk more intelligently about your tests passing or failing. The fixture-based example uses the "name" property to provide the same information.

Assertions

D.O.H. exposes a small but adequate number of assertion APIs:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
doh.assertEqual(expected, actual); // aliased to: doh.is(e, a)
doh.assertTrue(condition); // aliased to: doh.t(condition)
doh.assertFalse(condition); // aliased to: doh.f(condition)
```

Asynchronous Tests

D.O.H. provides direct support for asynchronous test cases. Writing asynchronous tests depends on a script context that "knows about" asynchronous execution (aka, a browser but not Rhino) and a slightly modified test authoring syntax:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
doh.register("tests.moduleToBeTested",
        // the async  text fixture
        {
                name: "thingerTest",
                timeout: 2000, // 2 seconds, defaults to half a second
                setUp: function(){
                        this.thingerToTest = new Thinger();
                        this.thingerToTest.doStuffToInit();
                },
                runTest: function(){
                        var testCondition = true;
                        var d = new doh.Deferred();
                        setTimeout(function(){
                                try{
                                        if(testCondition){
                                                d.callback(true);
                                        }else{
                                                d.errback(new Error("we got a failure"));
                                        }
                                }catch(e){
                                        d.errback(e);
                                }
                        }, 100);
                        return d;
                }
        }
);
```

Note that in the above example, the runTest function *explicitly* returns a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `doh.Deferred` object. This is how the system knows that you are going to be testing potentially asynchronous conditions. Also, in our delayed call (see the setTimeout), we explicitly catch errors and pass them to the Deferred's /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `errback()` function. It is expected that your code will do this if you are testing asynchronous conditions. Lastly, you may specify a timeout in milliseconds as part of the fixture object.

As with the previous examples, you can specify an anonymous or single function in place of a the full fixture used here to handle asynchronous cases. The only caveat is that it must return a Deferred object in order to be treated as an async test. We can also simplify the above by using the tests.Deferred classes getTestCallback() method. Here's a simplified async test case:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
doh.register("tests.moduleToBeTested", function simplerAsyncTest(){
        var testCondtition = true;
        var d = new doh.Deferred();
        var checkCondition = function(){
                doh.assertTrue(testCondition);
        };
        setTimeout(d.getTestCallback(checkCondition), 100);
        return d;
});
```

While the test case still needs to pass back a Deferred object, the use of the getTestCallback() to wrap the success or failure test function allows us to stop manually handling exceptions that might be thrown in the callback function, specifically from assertTrue(), assertEqual(), or assertFalse().

Group Registration

Many times, it's advantageous to register an entire group of tests at once. D.O.H. provides a method for doing this as well as for registering group-level /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `setUp` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `tearDown` methods. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color:

#0066FF;} `tests.registerGroup(name, tests, setUp, tearDown)` method lets you handle this in a single call:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
// file located in core at:
//      tests/fullGroupTest.js
dojo.provide("tests.fullGroupTest");
dojo.require("tests.runner");
doh.registerGroup("tests.fullGroupTest",
        [
                // single test, no test fixture
                function assertTrueTest(t){ t.t(true); },
                // string variant of the same:
                "doh.t(true);",
                // test that uses variable set up by group
                function assertTrueTest(t){
                        t.t(tests.fullGroupTest._localVariable);
                },
                // ...
        ],
        function(){ // setUp
                tests.fullGroupTest._localVariable = true;
        },
        function(){ // tearDown
                tests.fullGroupTest._localVariable = false;
        }
);
```

Note that when using /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `registerGroup`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `setUp` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `tearDown` replace existing group-level handlers, but the registered tests are additive to any pre-existing tests registered for the group.

The above example also introduces yet another shorthand for writing tests, the string-only test. This style of test authoring is particularly terse. Tests written this way do not provide explicit fixture names and so the test code itself is used as the test name in reporting. In these tests, there is also always a variable /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `t` which is an alias to the global /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `tests` variable. This allows for very compact tests to be written in the form:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
[
        "doh.t(true);",
        "doh.f(!true);",
        "doh.is('thinger', 'thing'+'er')",
        // ...
]
```

With group registration, this style of test authoring requires very little typing. Just mind your string quotes!

URL-based Testing

Being developed explicitly to test JavaScript applications, D.O.H. includes features for browser-based test harnesses to load sub-documents which may run a set of tests explicitly on a browser-provided DOM. This lets you automate UI testing and isolate browser-specific bugs by writing tests once and quickly running them through the unified test harness UI. To support this, browser runtimes for D.O.H. provide an implementation for /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `tests.registerUrl(groupName, url)`. On other environments, this may be a no-op.

A real example from Dojo Core:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight:

bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
doh.registerUrl("tests._base.NodeList", dojo.moduleUrl("tests", "_base/NodeList.html"));
```

This example uses Dojo to normalize the tested URL with relationship to the loading code, but you can just as easily specify a full URL manually. Just be aware that in order for D.O.H. to be able to record the results of tests from this page, it must be hosted on the same domain as the hosting test harness.

But what does the page itself look like? Here's a snapshot of the page referenced above:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<html>
	<head>
		<title>testing dojo.NodeList</title>
		<script type="text/javascript" src="../../dojo.js"
			djConfig="isDebug: true"></script>
		<script type="text/javascript">
			<b>dojo.require("doh.runner");</b>
			dojo.addOnLoad(function(){
				doh.register("t",
					[
						function ctor(){
							var nl = new dojo.NodeList();
							nl.push(dojo.byId("c1"));
							doh.assertEqual(1, nl.length);
						},
						// ...
					]
				);
				<b>doh.run();</b>
			});
		</script>
	</head>
	<body>
		<h1>testing dojo.NodeList</h1>
		<div id="t">
			<span id="c1">c1</span>
		</div>
	</body>
</html>
```

The above code has several important features (in bold). First, we ensure that the test system itself is loaded into the tested page. Without this, the tests won't run. D.O.H. is smart enough to know if it's being loaded into a child frame or as a parent document. If you load this file into a normal browser window, the tests will still run, but you won't get the pretty D.O.H. chrome or audio feedback. Instead, the results of *only the tests from this page* will be sent to whatever console facility is available.

The second important feature of our tested URL is that it manually calls /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `tests.run()`, in this case after the page has been loaded and tests have been registered (a good time to do it). There are Dojo-isms in the test page, but they don't affect the important bits of the system. You can still load the test system with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `<script>` tags and hard-wired URLs and this file would participate in the larger test group correctly.

Since testing on loaded pages may take a long time (relatively), a default timeout of 10 seconds per URL is provided. If your tested page requires more (or less) time, you can pass an explicit timeout parameter to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `tests.registerUrl` method:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
doh.registerUrl(
	"tests._base.NodeList",
	dojo.moduleUrl("tests", "_base/NodeList.html"),
	5000); // 5000ms, or 5 seconds
```

Running the tests in Rhino

Here are some instructions for running the tests in Rhino. Ideally they should run in any Rhino version 1.6R4 or later, however it has only be verified to work with the custom_rhino.jar in Dojo's util repository (it is in util/buildscripts/lib). Steps:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3

{color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
> svn svn co http://svn.dojotoolkit.org/dojo/view/anon/all/trunk dojo_0.9
> cd dojo_0.9
> cd util/doh
> java -jar ../buildscripts/lib/custom_rhino.jar runner.js
```

NOTE: Do not use a built version of dojo when trying to run DOH through the Rhino command line. The built versions of Dojo that are available for download are optimized for use in the browser, and they do not have the auto-detection logic to load the right environment code for Rhino. So be sure to use a source distribution when running the DOH tests in Rhino.

Including Tests From Custom Modules

FIXME: TODOC

Including Tests Without Dojo

FIXME: TODOC

Performance Optimization

So you've built an application using Dojo and it "feels slow". Now what? Unfortunately, you can't send the latest Dual Core processors to your users But fortunately, experience has taught us that in JavaScript, smaller = faster, no matter what the client. So basically, you want to concentrate on the size of your pages. The smaller they are, the less time it takes to download, parse and execute them.

By default, all dojo.require() statements try to find the module in memory first. If it's not present, it will ask the server for a "full sized" version of the code in a synchronous manner , and incurring network I/O overhead. Synchronous network requests from a browser are necessary for loading external code because it's the only way to block execution of JavaScript while dependencies are satisfied. Once the code is fecthed it is then eval()'d and execution picks up after the dojo.require() statement.

Unfortunately synchronous IO requests also have the effect of "locking" the UI of the browser, preventing loading other resources affecting the layout of the page. Each request is serial. The package system doesn't know enough to try to request multiple files at once. To be fair, scripts included in a page via the <script> tag also suffer from the same serial behavior.

But there are still plenty of things you can do:

- Use Dojo's custom build system. A custom build of Dojo will improve the download of the page by grouping related modules into one script, and optimizing it for fast parsing. See The Package System and Custom Builds for more information.
- Configure the web server to cache JavaScript files aggressively and send status information quickly. Servers that don't send adequate cache information may find that UIs are very slow even when the script content isn't sent. Instead, browsers may be checking to see if the file has changed since it was last seen. This "has it changed?" check is synchronous and serial and so we want to eliminate it if possible.
- Use data compression and application program structure as your server permits. See the article Improving performance of Dojo-based web applications which contains useful information on these topics.
- If you find IE is repeatedly downloading the same image, read this article for advice. This also helps get rid of ugly image flickering in IE.
- Reduce then number of tags on a page.

  It sounds obvious, but many web pages have a lot of unnecessary markup. For example, instead of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

  ```
  <table ...><BR></BR>  <tr>
  <td>Hello World</td>
  </tr>
  </table>
  ```

  just do: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

  ```
  <div class="foo">Hello World</div>
  ```

  To achieve this, you definitely want to leverage CSS. The web standards movement has championed this approach, and Dijit uses it for reducing the widget download time.
- Instead of downloading your whole page at once, defer portions until the user needs them. The judicious use of ContentPanes nested inside containers /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

  ```
  <div dojoType="TabContainer">
      <a dojoType="LinkPane" href="tab1.jsp">Tab #1</a>
      <a dojoType="LinkPane" href="tab2.jsp">Tab #2</a>
  </div>
  ```

The Package System and Custom Builds

A Dojo custom build speeds performance by doing the following:

1. First, it groups together modules into *layers*. A layer, which is one big .js file, loads faster than the individual .js modules that comprise it
2. Second, it *interns* external non-JavaScript files. This is most important for Dijit templates, which are kept in a separate HTML file. Interning pulls the entire file in and assigns it to a string.
3. Third, it smooshes the layer down with ShrinkSafe. ShrinkSafe removes unneeded whitepsace and comments, and compacts variable names down to smaller ones. This file downloads and parses faster than the original.
4. Finally, it copies all non-layered scripts to the appropriate places. While this doesn't speed anything up, it ensures that all Dojo modules can be loaded, even if not present in a layer. If you use a particular module only once or twice, keeping it out of the layers makes those layers load faster.

The catch? You have to designate the modules in each layer with a *profile*, which is something like a Makefile or Ant script. But that's not too hard if you know your app well.

So the input of the build system is the Dojo source tree, plus any source trees for custom stuff you wish to include ... plus the profile. The output is a Dojo distribution tree which you can copy to your web server. Sweet!

Prerequisites

You need the following installed on your computer to run Dojo's build system:

- Java 1.4.2 or later (Java 1.5 recommended).
- A source build of Dojo, which you can obtain at http://download.dojotoolkit.org/release-1.0.0. The source builds are suffixed with "-src". If you want to download the latest code from the Subversion code repository, see the Use Subversion page.

Creating a Custom Profile

In the src/buildscripts/profiles directory, you will create a profile build file called foo.profile.js like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dependencies ={
    layers:  [
        {
        name: "mydojo.js",
        dependencies: [
            "dijit.Button",
            "dojox.wire.Wire",
            "dojox.wire.XmlWire",
            "explosive.space.Modulator"
        ]
        }
    ],
    prefixes: [
        [ "dijit", "../dijit" ],
        [ "dojox", "../dojox" ],
        [ "explosive", "../../explosive" ]
    ]
};
```

The dependencies section within the layer lists all the modules you call *directly*. Any referenced modules will also be included, so you don't have to trace back the dependency tree. Also, Dojo base modules are an implicit dependency, so you don't need to list things like "dojo.query". (Dojo core modules, however, do need to be listed.)

The modules for that layer are gathered together to make the "layer" file, in our example: "mydojo.js". Then you just load this layer file in your pages with a SCRIPT tag. Easy!

The prefixes section list any modules that need inclusion. Note our "explosive" module, which is located away from the Dojo tree. You need to list these if you use them, even if you don't want any modules from it in your layer file.

For the 1.0+: If you choose to optimize the JS files in a prefix directory (via the optimize= build parameter), you can choose to have a custom copyright text prepended to the optimized file. To do this, specify the path to a file tha contains the copyright info as the third array item in the prefixes array. For instance:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
prefixes: [
    [ "explosive", "../../explosive", "../../explosive/copyright.txt"]
]
```

If no copyright is specified in this optimize case, then by default, the dojo copyright will be used.

Running The Build

After specifying a profile file as shown above that statically specifies the resources you want to include, and saving it as /buildscripts/profiles/foo.profile.js, you run the Rhino interpreter on it and specify the profile name as a parameter. For example, from the buildscripts directory:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw3 {color: #000066;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .re0 {color: #0000ff;} .geshifilter .re1 {color: #0000ff;} .geshifilter .re2 {color: #0000ff;} .geshifilter .re3 {color: #808080; font-style: italic;} .geshifilter .re4 {color:

#0000ff;}
```
$ cd util/buildscripts
$ build.sh profile=foo action=release
```

On Windows PC's, substitute build.bat for build.sh. For both platforms, you may also specify additional build options. Run build.sh to see a list of all supported options. Here is a sample of the supported options:

profile	The name of the profile to use for the build. It must be the first part of the profile file name in the profiles/ directory. For instance, to use base.profile.js, specify profile=base. Default: base
profileFile	A file path to the the profile file. Use this if your profile is outside of the profiles directory. Do not specify the "profile" build option if you use "profileFile" Default: "",
action	The build action(s) to run. Can be a comma-separated list, like action=clean,release. The possible build actions are: clean, release Default: "help",
version	The build will be stamped with this version string Default: "0.0.0.dev",
localeList	The set of locales to use when flattening i18n bundles Default: "en-gb,en-us,de-de,es-es,fr-fr,it-it,pt-br,ko-kr,zh-tw,zh-cn,ja-jp",
releaseName	The name of the release. A directory inside 'releaseDir' will be created with this name Default: "dojo",
releaseDir	The top level release directory where builds end up. The 'releaseName' directories will be placed inside this directory Default: "../../release/",
loader	The type of dojo loader to use. "default" or "xdomain" are acceptable values." defaultValue: "default",
internStrings	Turn on or off widget template/dojo.uri.cache() file interning Default: true,
optimize	Specifies how to optimize module files. If "comments" is specified, then code comments are stripped. If "shrinksafe" is specified, then the Dojo compressor will be used on the files, and line returns will be removed. If "shrinksafe.keepLines" is specified, then the Dojo compressor will be used on the files, and line returns will be preserved. If "packer" is specified, Then Dean Edwards' Packer will be used Default: "",
layerOptimize	Specifies how to optimize the layer files. If "comments" is specified, then code comments are stripped. If "shrinksafe" is specified, then the Dojo compressor will be used on the files, and line returns will be removed. If "shrinksafe.keepLines" is specified, then the Dojo compressor will be used on the layer files, and line returns will be preserved. If "packer" is specified, Then Dean Edwards' Packer will be used Default: "shrinksafe",
copyTests	Turn on or off copying of test files Default: true,
log	Sets the logging verbosity. See jslib/logger.js for possible integer values Default: logger.TRACE,
xdDojoPath	If the loader=xdomain build option is used, then the value of this option will be used for the path to Dojo modules. The dijit and dojox paths will be assumed to be sibilings of this path. The xdDojoPath should end in '/dojo' Default: "",

Cross Domain (XDomain) Builds

Doing an xdomain build allows you to load Dojo and your custom modules from another domain. The benefits to doing this are listed in the 0.4 Book page about Cross Domain Resource Loading.

To do an xdomain build:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw3 {color: #000066;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .re0 {color: #0000ff;} .geshifilter .re1 {color: #0000ff;} .geshifilter .re2 {color: #0000ff;} .geshifilter .re3 {color: #808080; font-style: italic;} .geshifilter .re4 {color: #0000ff;}
```
$ cd util/buildscripts
$ build.sh profile=foo loader=xdomain xdDojoPath=http://my.server.com/path/to/buildoutputdir action=release
```

xdDojoPath is optional. It just burns in the location of dojo, dijit and dojox into the built dojo.js. If you do not specify that option, then you will need to use djConfig.modulePaths/dojo.registerModulePath() in your HTML page to set the xdomain locations for dojo, dijit and dojox. For your own custom modules, you will have to set djConfig.modulePaths/dojo.registerModulePath() even if you us the xdDojoPath build option.

For 0.9+ there is a bug about loading dojox.gfx with an xdomain build.. If you want to use dojox.gfx with an xdomain build, there are some workarounds until the bug gets fixed:

1. Include dojox/gfx.js directly in your page with a script tag in the HTML source, after the dojo.js script tag (do not use gfx.xd.js, use gfx.js).
2. Include dojox.gfx in a layer file that you load via a script tag in the HTML source (load the .js layer file, not the .xd.js layer file).

Part 5: DojoX - Experimental and Specialized Extensions

DojoX is an area for development of extensions to the Dojo toolkit. It acts as an incubator for new ideas, a testbed for experimental additions to the main toolkit, as well as a repository for more stable and mature extensions. Unlike Dojo and Dijit, DojoX is managed by subprojects, each of which has at least one module, a sponsor and a mission statement. [Release cycle policy TBD] The subprojects may have dependencies on Dojo and Dijit code or other subprojects in DojoX. Other projects may choose to keep their dependencies on Dojo minimal, perhaps only depending on Dojo Base, and remain largely toolkit agnostic.

Some Dojox projects directly extend Dojo components, like the Flickr data store. These are documented in Part 3.

Some caveats of using DojoX:

- The condition and level of support of DojoX code will vary, from experimental through release. See README for subproject status. DojoX subprojects may disappear entirely if unsuccessful.
- Unlike Dojo and Dijit, DojoX modules are **not** guaranteed to be fully accessible or internationalized
- DojoX subprojects may be moved to Dijit or Dojo Core, subject to the needs of the toolkit and the capacity of those teams to absorb additional code.
- Not all modules in DojoX will be documented here, since they are lower priority than base and core. Browse the API documentation and repository directly for a more complete list.

Cometd (client)

Version

Version 0.4

Release Date

Release date: May 29, 2007

Project state: beta

Project authors

Alex Russell Greg Wilkins

Project description

Low-latency data transfer from servers to clients. dojox.cometd implements a Bayeux protocol client for use with most Bayeux servers. See cometd.com for details on Cometd or on the Bayeux protocol.

Dependencies:

Needs a cooperating Bayeux server

Documentation

See http://cometd.com

Installation instructions

Use this library with (preferably through) an existing Cometd server.

DojoX Charting

Version

Version 0.800

Release Date

Release date: 10/31/2007

Project state:

beta

Credits

Tom Trenka

Eugene Lazutkin

Project description

Implementation of simple charting library based on dojox.gfx/dojox.gfx3d.

Dependencies:

Dojo Core

dojox.gfx

dojox.gfx3d

dojox.lang

Documentation

Needs documentation

Installation instructions

Needs documentation

DojoX Collections

Version

Version 0.9

Release Date

Release date: 05/27/2007

Project state: stable

Project authors

Tom Trenka

Project description

DojoX Collections is the port of the original Dojo 0.4.x collection classes. It is intended for use by people who are looking for a little bit more functionality out of common collections, like ArrayLists or Dictionaries.

Included are the Iterator and DictionaryIterator classes, both of which can operate on standard arrays and objects (respectively).

Dependencies:

DojoX Collections has no dependencies, outside of Dojo Core.

Documentation

See the API documentation for Dojo (http://dojotoolkit.org/api).

Installation instructions

Grab the following from the Dojo SVN Repository: http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/collections.js http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/collections/*

Install into the following directory structure: /dojox/collections/

...which should be at the same level as your Dojo checkout.

DojoX Cryptography

Version

Version 0.9

Release Date

Release date: 05/27/2007

Project state: beta

Project authors

Tom Trenka

Project description

The DojoX Cryptography project is a set of implementations of public crypto algorithms. At the time of writing, only MD5 and Blowfish are complete and tested; at least 5 other algorithms on the way.

DojoX Cryptography is comprised of both symmetric (Blowfish) and asymmetric (MD5) algorithms. Symmetric algs always implement encrypt() and decrypt() methods; asymmetric algs implement compute().

Dependencies:

DojoX Cryptography has no dependencies, outside of the Dojo package system and DOH.

Installation instructions

Grab the following from the Dojo SVN Repository: http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/crypto.js http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/crypto/*

Install into the following directory structure: /dojox/crypto/

...which should be at the same level as your Dojo checkout.

Documentation

See the Dojo API tool (http://dojotoolkit.org/api)

DojoX Data

Version

Version 0.9

Release Date

Release date: 05/29/2007

Project state: beta

Project authors

Jared Jurkiewicz

Shane O'Sullivan

Project description

The DojoX Data project is a container for extensions and extra example stores that implement the dojo.data APIs. It may also contain utility functions for working with specific types of data.

Dependencies:

DojoX Data has dependencies on core dojo (dojo.data) and the D.O.H. unit test framework.

Documentation:

Much of the Dojox Data package is documented in Part 3's Using dojo.dataSee the Dojo API tool (http://dojotoolkit.org/api) for further docs.

Installation instructions

Grab the following from the Dojo SVN Repository: http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/data/*

Install into the following directory structure: /dojox/data/

...which should be at the same level as your Dojo checkout.

/dojox/data/*

Require in the dojox.data stores you wish to use.

DojoX DTL (Django Template Language)

The Django Template Language is one part of Django, a "high-level Python Web framework that encourages rapid development and clean, pragmatic design." Django is the preferred web framework for several of the Dojo commiters.

The DojoX implementation implements the full *infrastructure* of the Django Template Language. This language, as implemented in the Django Project is limited to text, since it only deals with page serving. While Dojo's implementation also works with text, it has an additional layer that allows us to dynamically render blocks of HTML.

Existing templates should work without fuss using Dojo's implementation. There are some additional abilities in an HTML environment, obviously, but you can add those as you go.

Learning the Markup

Since Dojo implements markup just as it is in Django's implementation, the best place to visit would be their excellent book or their excellent documentation

But in case you are just itching to know what it looks like, it's made up of some simple parts: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `{% tags %}`, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `{{ variables }}`, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `{{ variables|filtered }}` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `{{ variables|more:"advanced"|filtering }}`. Sometimes tags have a start and an end tag, sometimes they work alone.

Creating and Rendering a Template in Plain Text

As in the version that you'd use within Django, rendering a Template involves a few steps. You need to create a Template, a Context, and then call the render function on the Template object.

Dojo's implementation works in the same way, the only thing that is important for you to know is what namespaces these objects are in.

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}


dojo.require("dojox.dtl");

var template = new dojox.dtl.Template("Hello {{ place }}!");
var context = new dojox.dtl.Context({
  place: "World"
});
console.debug(template.render(context)); //1
```

Like Django, Dojo's implementation allows you to render the template as many times as you like with as many versions of the context as you'd like

Writing a Widget

We'll bypass how to use the raw HtmlTemplate object (we'll get into that later) and explain how to write a widget using Dojo's implementation of the Django Template Language.

Unlike most Dijit widgets, a widget using this template language isn't tied specifically to a node. Instead, we tie the widgets to another widget that comes as part of dojox.dtl.widget, the AttachPoint. The reason for this is to both make rendering a template easier for the widget author, and to allow the flexibility of more than one Widget to render in the same place. Looking back at the page serving model of Django, the AttachPoint is a screen on which to draw.

A better way to explain it is this: If we have two different widgets that extend the same base template, then even though some portions of the template are changing, the nodes that are unique to the parent template should stay exactly where they are in the DOM.

There are two ways to use an AttachPoint with your widget. The easiest way is to simply nest it inside of an AttachPoint in your HTML, but you can also use the setAttachPoint function to accomplish the same thing

Extending dojox.dtl._Widget adds not only the setAttachPoint function, but also the render function, which you'll need to use in a dojox.dtl widget in order to display it

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}


dojo.require("dojox.dtl.widget");

dojo.declare("demo", dojox.dtl._Widget, {
  postCreate: function(){
    var template = new dojox.dtl.HtmlTemplate("<div>Hello {{ place }}!</div>");
    var context = new dojox.dtl.Context({ place: "World" });
    this.render(template, context);
  }
});

...

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #b1b100;}
.geshifilter .kw2 {color: #000000; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .coMULTI {color: #808080; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
```

```
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #ff0000;}
.geshifilter .nu0 {color: #cc66cc;}
.geshifilter .sc0 {color: #00bbdd;}
.geshifilter .sc1 {color: #ddbb00;}
.geshifilter .sc2 {color: #009900;}
```

```
<div dojoType="dojox.dtl.AttachPoint">
  <div dojoType="demo"></div>
</div>
```

New Tags and Filters for an HTML Environment

Tags

- **html**: If you want to render a variable as HTML, you have to use {% html varName %}

Attribute Tags

Before we get to the list, it's important to note that HtmlTemplate considers attributes to be a full-fledged part of the template system. What this means is that there are new tags that have been introduced that use named attributes in order perform new actions. For the curious tag author, this is accomplished by registering a named tag with "attr:" in front of it

- **onclick/onmouseover/etc**: The string you have in its value is the name of the function to call when that action is performed. eg onclick="goHome"
- **tstyle**: Use if you want to change style attributes during rendering. eg tstyle="top: {{ y }}px;"
- **attach**: The string you have in its value is the name of the variable to attach the node to. eg attach="myNode"

Using the Extends Tag

In Django, the extends tag looks through the installed applications until it finds the named template. In a browser environment, we don't want to have to go searching for templates, so there has to be a way to reference a specific file, while not changing the markup style of the extends tag.

The "easiest" way to do this is to put an explicit reference to the template. This means that you need to specify a URL in relation to your root page. But doing it like this creates a problem if you want to move around your directory structure, or if a page in a different directory wants to use the template.

Django helps us out by allowing a variable name to be used in the extends tag. What we can do with this, then, is set a variable in the Context using dojox.moduleUrl.

If we're using the extends tag in an HTML environment, there's another factor to consider. Let's say we have a blog and there are two ways of viewing the page: a list view, and a detail view. Both of these views use a parent node that contains the page header, a menu, and a sidebar. We don't want the template system to have to redraw the DOM for their parent template, but how do we indicate that? There are two ways, one which is significantly better than the other.

The first is to use a string in the extends tag, outlined in the "easiest" way at the top. Putting "shared:" at the beginning of the string tells the extends tag to reuse the nodes between all other children that also want to share the parent.

The significantly better way is partly outlined in the section above on moduleUrl. You can use a variable containing a moduleUrl, but how do you tell the extends tag that you want to share the parent? Instead of just passing a moduleUrl call, when we have an extends tag that looks like {% extends parent %}, we can use an object that looks like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
new dojox.dtl.Context({
  parent: {
    url: dojo.moduleUrl("mymodule", "templates/template.html"),
    shared: true
  }
});
```

New Context Object Abilities

Unlike the page serving model of Django, we can keep our Context objects around between each template render. What we want to be able to do is quickly clone an existing context, and either reduce, or add to, the data in the object. To do this, there are two new functions:

- **filter**: Just put the keys you want to key in its arguments to get only those keys back in a cloned Context.
- **extend**: Just pass it an object to get a cloned Context containing all of the old keys, plus the new ones.

Some new functions are added to allow tags to communicate with the rendering object.

- **setThis**: Sets the object on which to perform operations. Used by the attach attribute tag, for example.
- **getThis**: Used by tags, gets the currently set *this* object.

Advanced Topics

While Django provides some documentation about how their compiler works, which you should definitely read before continuing this section, there are some areas in which things are done differently, such as tag and filter registration.

The other major difference in Dojo's implementation is the new HTML compiler. This section will go over how much of what we've done was accomplished, and will provide information to those curious about how the normal text-based compiler works.

Writing and Registering Tags and Filters

The actual filter functions are identical in terms of structure to the way they are implemented in Django. What this means is that Django's documentation is great for learning how to write a filter, and doesn't need to be rewritten here

Tags work in the following way: a registered tag is passed the parser object, and the tag string, and is tasked with returning an object that obeys a specific interface. This is how they are implemented in Django, so once again, their documentation will go a long way toward understanding how to write a tag.

The one distinct difference between the two system is rendering a NodeList. The expected behavior would be to pass it the context and buffer objects that the render function accepts, but you must also pass its own instance (this) as the third parameter. Without this, there could be unexpected behavior when rendering in an HTML environment. You should also clone nodelists and then render them, rather than taking the re-rending approach of Django's normal system. Because the rendering in HTML results in a DOM tree, re-rendering without cloning would change the same nodes over and over.

As mentioned earlier, there is a new tag type in the HTML version of the compiler. If you've registered an attribute-based tag, you will be passed a null parser object, and the string contents of the attribute tag are prefixed with the attribute name. What this means is that, for example, in the case of an attach attribute that looks like attach="varName", the string that is passed is "attach varName". This way, you can register your function with a regular expression, but see exactly what was matched.

The only piece of the interface you have to implement for the text-based version is the render function. If your tag will appear in an HTML environment, you must add an unrender and clone function. The unrender function is tasked with removing a node from the DOM if one was previously added, or to tell the "swallowed" nodes to unrender. The clone function should duplicate the object. Note that the same tags are used in **both** the text and HTML versions of the compiler, so your functions should be written with this in mind.

Registering tags/filters looks like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}

dojox.dtl.register.filter("my.module", "my.module.obj", ["fn", "anotherFn", [/RegExp/, "fnForRegex"]]);

Obviously, registering a tag would use dojox.dtl.register.tag. The first parameter is the require statement to call if we encounter one of these tags or filters. The second parameter is the base object on which these functions are stored. And the final parameter is an array of function names, or RegExp/function name pairs. Since some words are "reserved words", if we can't find the function on your object, we check for the function name with an underscore on the end of it.

If you write a custom set of tags and filters, the way to use them is exactly the same as in Django: the load tag. At the top of your template, add /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.javascript .imp {font-weight: bold; color: red;}
.javascript .kw1 {color: #000066; font-weight: bold;}
.javascript .kw2 {color: #003366; font-weight: bold;}
.javascript .kw3 {color: #000066;}
.javascript .co1 {color: #009900; font-style: italic;}
.javascript .coMULTI {color: #009900; font-style: italic;}
.javascript .es0 {color: #000099; font-weight: bold;}
.javascript .br0 {color: #66cc66;}
.javascript .st0 {color: #3366CC;}
.javascript .nu0 {color: #CC0000;}
.javascript .me1 {color: #006600;}
.javascript .re0 {color: #0066FF;}
`{% load namespace.tags namespace.filters etc %}`. You should register your tags and filters in the same file as you declare them, and where you put the registration code doesn't matter, it can be before or after your object declarations.

How HTML Templates are Compiled

A quick note before we begin: If tags and filters within code are wrapped in comment tags, as in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .html4strict .imp {font-weight: bold; color: red;} .html4strict .kw1 {color: #b1b100;} .html4strict .kw2 {color:

#000000; font-weight: bold;} .html4strict .kw3 {color: #000066;} .html4strict .coMULTI {color: #808080; font-style: italic;} .html4strict .es0 {color: #000099; font-weight: bold;} .html4strict .br0 {color: #66cc66;} .html4strict .st0 {color: #ff0000;} .html4strict .nu0 {color: #cc66cc;} .html4strict .sc0 {color: #00bbdd;} .html4strict .sc1 {color: #ddbb00;} .html4strict .sc2 {color: #009900;} `<!--{% tag %}-->` then some browsers treat them as full nodes and will help a lot with node traversal. Just something worth knowing.

There are two steps to compiling a template: tokenizing a template, and parsing the tokens. The types of tokens can be found in dojox.dtl.text.types as well as dojox.dtl.html.types. They are: tag, variable, text, change, attribute, and element. Tags and variables are of course the foundation of our markup, text is just plain text that has no real bearing on markup, a change is when the parent node changes, an attribute is an HTML attribute, and an element is a DOMNode. During tokenizing, we actually disassemble the tree structure so that when the tokenization is done, none of the Nodes have a parent Node. The first Node encountered is the "root" Node. This is what we will use to insert our template into our document.

We build these tokens by traversing the Node tree, and when we reach a text node, tokenizing the contents if it has any tags or filters (note: this is where, if you use comments, this will never have to happen). After we're done tokenizing, we move on to parsing.

During parsing, all of the tokens become objects. Many of the tokens are always replaced with the same object. For example, the change type always becomes a dojox.dtl.html.ChangeNode, the element type always becomes a dojox.dtl.html.Node, a variable always becomes a dojox.dtl.html.VarNode, and text always becomes a dojox.dtl.html.TextNode. This leaves us with the attribute and tag types. An attribute token can go in two directions: If there is a tag registered for the name of that attribute, we call that function and the returned object replaces that token. If not, the token becomes a dojox.dtl.html.AttributeNode.

The major player here is the tag token. When the registered tag function is called, the tag doesn't always simply replace the token, it can actually "swallow" tokens until it finds a tag that it's looking for. A good example of this is the if/else/endif structure, where the function parses tokens until it reaches and endif tag. This is how nesting tags is possible. In the case of the if tag, while the first if tag is parsing tokens, looking for an else or endif tag, the parser might run into another if tag. This if tag now the one parsing tokens, looking until it finds the endif tag. Once it's done, that object replaces a whole range of tokens in the content that first if tag has been building. The parent if tag, when finished, returns an object to the parser that replaces a range of tokens, all of which are now in either one of the new objects returned by the if tags. No tokens ever disappear, they can all be accounted for in the various objects.

What we end up with is a very linear set of instructions. We have an array of objects, all our objects have a render function, and many of the objects contain lists of objects, and when their render function is called, they will call the render function on the objects they have swallowed. We have what's basically, a tree that always executes linearly, and is always called in the same order.

The Importance of our Buffer Objects

The render/unrender/clone functions used in registered tag objects all accept a buffer. Since we are using the same tag objects for both text and HTML, and we absolutely need a buffer to render an HTML template (we'll get to that in a second), a buffer will be both accepted and returned by every render/unrender function. When we render a text-based template, we use dojox.string.Builder to ensure the fastest string concatenation possible. But in the HTML version of the compiler, we have a very special Buffer.

Let's propose a situation that happens all the time when writing a template. It looks something like this:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div>
  <h1>Title</h1>
  {% if name %}
  <div>Hello {{ name }}</div>
  {% else %}
  <div><a onclick="login">Please log in </a></div>
  {% endif %}
  < p>Welcome to our wonderful site</ p>
</div>
```

Now when someone uses this page, we have a situation where, when they first use the application, they are given a link to log in, but once their name is set, they're given a "Hello" message. While we didn't have to write it this way, we've created a situation now where only one of two sibling divs can be shown at a time. For those of you familiar with DOM, you know how complicated it is to insert a node if it's at neither the beginning nor the end of a parent. The HTML buffer allows us to overcome this problem with ease.

As a template is rendered, the setParent function is called on the buffer object, which changes the Node that is considered to be the current parent. Calling concat on the buffer uses this parent to insert the passed node beneath the current parent. Where to place it involves a little bit of magic. As shown in the example above, if we just use appendChild, the second time we render the template, the div in the else tag will end up after the paragraph tag, which is not where it should go.

What we do, then, is check to see if the node we're inserting exists in DOM (by looking at its parentNode). If it doesn't exist in DOM, we check to see if the buffer's currently set parent has any childNodes. If the parent has no child nodes, we can just use appendChild. If there are existing child nodes, we store the child in a cache. When a node arrives that has the same parentNode as the buffer's currently set parent, we take the cache and append it **before** this node.

Finally, when setParent is called, we get the cache from the old parent, and make sure all the nodes are appended to the end of it, before changing the parent.

The reason this works is that every time we render, every node calls the concat function. While this may seem like a speed hit, it's not only necessary to allow us to fill in those cached nodes, the function does no other work. So in the example above we start out with the opening div set as the current parent, and no value for *name*. The buffer's concat function is called with the h1, the div in the else block, and finally the p tag. The h1 is added using appendChild, since it's the first node, and the div and p are cached until setParent is called at the end of the render (in code, we actually call setParent twice at the end on the root node so that the cache gets flushed).

But now, if we re-render after giving *name* a value, the following happens: The h1 is passed to concat and nothing happens since it has a parentNode, and the cache is empty, then the div in the if block gets added to the cache since it has no parentNode, the div in the else block gets removed from the DOM by the unrender function of the if tag, and when the p tag is passed to concat, we see that it has a

parentNode, and use insertBefore to place the items in the cache (our div) before the p tag.

Using this method of buffering, we don't need to worry **at all** about how the DOM is organized. Simply switching the parent and calling concat manages all the complex interactions we would otherwise have to account for.

Using the HtmlTemplate

Another quick note: This page is more to explain how this object works. Using the dojox.dtl.render.html.Render object to display your templates is a cleaner and more error-proof way of rendering and inserting your template into DOM.

Really the only thing you'll be doing with this object is creating an instance of it. You can either pass it a string to use as your template, or an object that has a toString method (we encourage the use of dojo.moduleUrl). Once you have an instance, you should pass it to the dojox.dtl._Widget.render function or the dojox.dtl.render.html.Render.render function.

Ultimately, even the widget's render function utilizes dojox.dtl.render.html.Render, so looking at the code for that will give you a great idea of how these functions are used. Let's actually display some of that code here:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
buffer = buffer || tpl.getBuffer();
```

You can pass whatever buffer you want to the HtmlTemplate's render function. What this means is that you could either subclass the current HtmlBuffer, or implement your own (not recommended) if you really want to do something crazy with the template architecture. But the HtmlTemplate object provides the **getBuffer** function so that you really don't need to worry about what Buffer object you should use.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
if(this._tpl && this._tpl !== tpl){
  this._tpl.unrender(context, buffer);
}
this._tpl = tpl;
```

The HtmlTemplate object also provides the **unrender** function to unrender the template. If we want to replace a template, we should call this function on the current template before we call render on the new one.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
var frag = tpl.render(context, buffer).getParent();
```

The **render** function builds a Node tree that's not necessarily in DOM. The getParent call here is actually on the HtmlBuffer, not on the HtmlTemplate. The HtmlTemplate provides a **getRootNode** function, which returns the first (and should be the only) top-level node in the rendered template. It is unlikely you'll need to use it, and one of the reasons we use it is to batch DOM changes (basically, checking to see if what's in the DOM is the root node, or whether we've removed it from DOM.

The HTML Renderer

This is a component that is completely unique to the Dojo implementation of the Django Template Language and is basically unrelated to the language itself. It is an object that is designed to render templates and make sure they are inserted into the DOM properly. It also add an important component for large templates, the ability to make sure that changes in the DOM are done in batches, though this ability is turned off by default, and even when turned on, can be done at a variety of levels

This is the object to use to make sure that rendering/unrendering happens properly, and what you want to use if you want to render a template, but don't want to use a widget to do it. All users are **strongly** encourage to use this to render a template if they are not using dojox.dtl._Widget to render their templates.

The object can be found at dojox.dtl.render.html.Render. It has two functions: setAttachPoint and render. It must have an attach point, which can be set in the constructor, or through the setAttachPoint function.

The **setAttachPoint** function simply sets the node we want to use to append the rendered template to.

The **render** function does several nice things:

- Unrenders the previous template if we're now using a new template.
- Creates a buffer if one hasn't been passed.
- Manages the batch change functionality discussed above.
- Inserts the rendered template into the document.

DojoX FX

Version

Version 0.1

Release Date

Release date: 08/02/2007

Project state:

prototype / expermental

Credits

Peter Higgins

Project description

dojox.fx provides a class of animation effects to use.

Dependencies:

dojox.fx requires dojo (core) and the dojo.fx package dojox.fx.easing requires only dojo core.

Documentation

existing API thus far:

dojox.fx.sizeTo - size a node about it's center to a new width/height

dojox.fx.addClass - animate the effects of applying a class to a node dojox.fx.removeClass - " " " " removing a class from a node
dojox.fx.toggleClass - wrapper for addClass/removeClass

dojox.fx.easing - a collection of easing functions to use this is a "stand alone" class, and can be used via:
dojo.require("dojox.fx.easing"); and provides: dojo.fx.easing.easeIn/easeOut/easeInOut to use in an _Animation easing: property

Installation instructions

checkout dojox/fx* from SVN and place into your sibling dojox folder.

see dojox/fx/tests/* for examples, and API tool for details.

DojoX GFX

Version

Version 1.100

Release Date

Release date: 08/01/2006

Project state:

beta

Credits

Eugene Lazutkin Kun Xi

Project description

Implementation of simple portable 2D graphics library.

Dependencies:

Dojo Core

Documentation

Currently it can be found here: http://docs.google.com/Doc?id=d764479_1hnb2tn

Installation instructions

Just include this line in your code:

dojo.require("dojox.gfx");

Drawing with GFX

dojox.gfx

What is it?

dojox.gfx is a cross-platform declarative interactive graphics package. It follows SVG loosely as the underlying model. At present time SVG and VML back-ends are implemented.

What does it do?

The dojox.gfx package provides graphics rendering and manipulation. Under Firefox, Opera and Safari dojox.gfx renders the final product as SVG, under IE it renders as VML.

Potentially the gfx package can allow you to create live and interactive graphing, a web based vector drawing program, view svg files in IE.

Underlying concepts

On conceptual level dojo.gfx has a simple declarative model: All basic primitives required for 2D graphics are defined:

- 2D coordinates
- 2D linear transformation matrices
- colors

There is a surface, which serves as a visual rectangular container for shapes:

- A web page can have several surfaces defined.
- Each surface has its own local coordinate system:
  - (0, 0) point is in the left-top corner, where the X axis is horizontal pointing right, and the Y axis is vertical pointing down.
  - Positive direction of rotation is defined as counter-clockwise (CCW).

There is a notion of a shape description object, which represent a simple description of geometry, with corresponding shape objects. dojox.gfx supports following shapes:

- Rectangle (optionally with rounded corners).
- Circle.
- Ellipse.
- Line.
- Polyline/polygon.
- Path (the most versatile shape). It implements the full SVG path language.
- Image.
- Text.
- TextPath (highly experimental at the moment).

Shapes support following set of properties:

- Geometric properties:
  - Shape description (shape-specific).
  - Linear transformation specified by 3 x 3 2D matrix.
  - Font (only for text shapes).
- Visual properties (not supported by the image shape):
  - Stroke (outline of a shape).
  - Fill (interior of a shape).

Shapes are stacked from bottom to top in an order of their definition. This z-order can be changed dynamically.

There is a group, which is a pseudo-shape. It combines other shapes (and possibly groups), and can be used to apply transformation to a group. All group members share a single z-order, but can be re-arranged within a group.

In order to draw a picture a programmer constructs a pseudo-DOM from a surface object, shapes, and groups, sets appropriate attributes, and picture is drawn automatically by a browser. Modifications of shapes change picture automatically. It is possible to attach event handlers to shapes using dojo.event.connect(). Following conventions are used:

While a path is the most universal geometric shape, which can emulate almost all other shapes (exceptions: image, and text shapes), all frequently-used shapes are provided as a convenience: rectangle (with optionally rounded corners), circle, ellipse, line, polyline/polygon.

All shape description properties are defined using a duck-typing technique. Incomplete shape description definitions are supported. All missing members will be taken from corresponding default shape definitions listed in common.js or from the current shape description object. Example: rect.setShape({width: 200}) - all missing members will be taken from dojo.gfx.defaultRect object making it equivalent to rect.setShape({x: 0, y: 0, width: 200, height: 100, r: 0}).

All shape description objects and visual property objects have a member called "type", which uniquely identifies a property type. This is a provision for a possible serialization.

Certain shape properties can be defined using shortcuts:

- Path can be defined as a string of path commands (more on path later):
  - path.setShape("m 0,0 l 100, 100 e") is equivalent to path.setShape({path: "m 0,0 l 100, 100 e"}).
- Polylines/polygons can be defined as an array of points:
  - poly.setShape([{x: 0, y: 0}, {x: 100, y: 100}]) is equivalent to poly.setShape({points: [{x: 0, y: 0}, {x: 100, y: 100}]}).
- Stroke can be defined by specifying a color as a string:
  - shape.setStroke("black") is equivalent to shape.setStroke({color: "black"}).

All methods without an apparent return type return their object itself. It is used for chaining multiple operations:

- surface.createRect({x: 100, y: 50}).setFill("red").setStroke("blue");

DojoX Grid

For user convenience, Grid documentation is located in Part 2 - Dijit.

DojoX I/O

Version

Version 0.1

Release Date

Release date: 08/29/2007

Project state:

experimental

Credits

Bryan Forbes

Project description

Dependencies:

DojoX IO has no dependencies, outside of Dojo Core.

Documentation

Installation instructions

Grab the following from the Dojo SVN Repository: http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/io/*

Install into the following directory structure: /dojox/io/

...which should be at the same level as your Dojo checkout.

Additional Notes

The information contained in this README does not pertain to DojoX XHR IFrame Proxy. For that information see proxy/README.

DojoX Image

Version

Version 0.9

Release Date

Release date: 08/29/07

Project state:

prototype | expermental

Credits

Peter Higgins

Shane O'Sullivan

Project description

A class to provide a common API for images, and home for image related Widgets.

Dependencies:

LightBox: dojo core, dojox.fx and optionally dojox.data. uses either tundra or soria theme, no standalone icons.

ThumbnailPicker: dojo core, dojox.fx and optionally dojox.data. Uses standalone icons.

SlideShow: dojo core, dojox.fx and optionally dojox.data. Uses standalone icons.

Gallery: dojo core, dojox.fx and optionally dojox.data. Uses SlideShow and ThumbnailPicker. Uses standalone icons.

Documentation

Installation instructions

there is no rollup for dojox.image, so you simply need the dojox/image folder and all of it's children to live in your dojox root.

Additional Notes

LightBox: currently works as individual items, and grouped items, but usage of dojox.data.FlickrStore is broken because all images are grouped in one master group, the API is subject to change, and is marked accordingly.

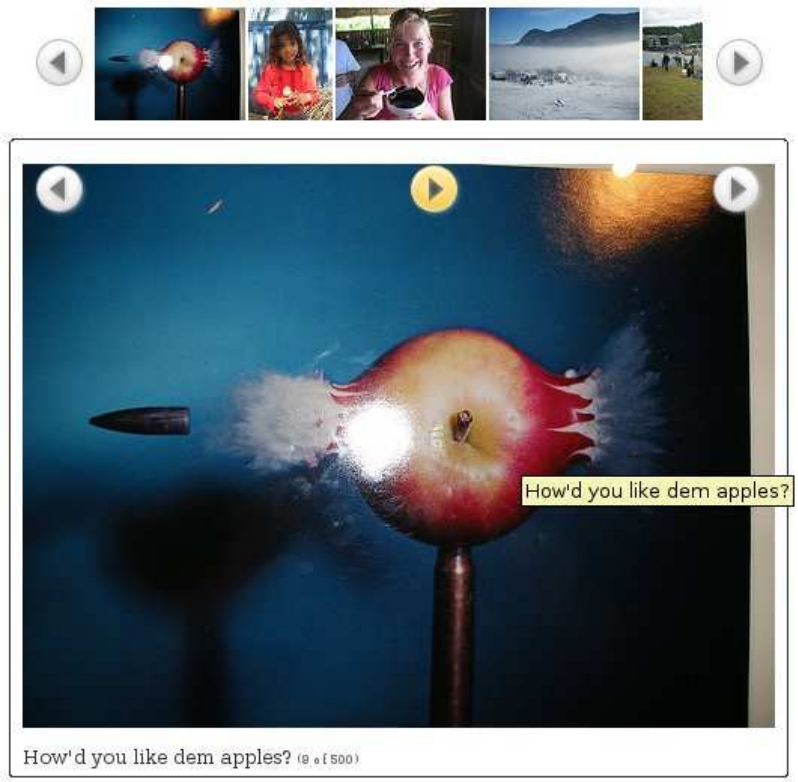
ThumbnailPicker: displays images in either a horizontal or vertical strip of small thumbnail sized icons. Image data is read from a dojo.data.api.Read data store.

SlideShow: displays a single image at a time, and provides controls to move between them, as well as running a slideshow where images change at a defined interval. Image data is read from a dojo.data.api.Read data store.

Gallery: wraps the ThumbnailPicker and SlideShow widgets, linking them together, so that a click on a thumbnail image results in the SlideShow changing its image. Image data is read from a dojo.data.api.Read data store.

Hoping to implement: Carossel, using a common API provided by dojox.image.Pane (?)

Gallery



dojox.image.Gallery is a widget that displays a series of thumbnail sized images, for quick browsing and selection, and a single large image. It provides a number of navigation controls for moving between images, and for playing an automated slideshow.

The Gallery works as a wrapper around two other widgets, dojox.image.ThumbnailPicker and dojox.image.SlideShow. It provides the following features:

- A row of thumbnail sized images. Clicking on a thumbnail image shows a large version of that picture in the main image pane. Navigation controls are placed to the left and right of the thumbnails, to move between them.
- A large image pane, displaying one image at a time. Navigation controls when the mouse pointer is over the image pane, allowing the user to move backwards and forwards between the images, and to start an automated slideshow. The size of the image pane is configurable using the **imageWidth** and **imageHeight** attributes.
- Reads image data from an implementation of the dojo.data.api.Read interface. This means it is completely decoupled from any particular data source, and can be used with any data source at all as long as it is a valid dojo.data store. For example, the Gallery can display images based on data in a simple text file using dojo.data.ItemFileReadStore, can show Flickr images using dojox.data.FlickrRestStore, or images from Picasa Web Albums using dojox.data.PicasaStore.
- Displays the title of each image, using a template that can be simply overridden.
- Links an image to a URL, so that clicking the image navigates to a specified web address. This is optional.

Examples

Creating a Gallery and setting a simple data source

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<script type="text/javascript">
  //Define the attribute names used to access the items in the data store
  var itemNameMap = {imageThumbAttr: "thumb", imageLargeAttr: "large"};
  //Define the request, with no query, and a count of 20, so 20 items will be
  //requested with each request
  var request = {query: {}, count: 20};
  dijit.byId('gallery1').setDataStore('imageItemStore', request, itemNameMap);
</script>
<div id="gallery1" dojoType="dojox.image.Gallery"></div>
<div jsId="imageItemStore" dojoType="dojo.data.ItemFileReadStore" url="images.json"></div>
```

Creating a Gallery with a FlickrRestStore

```
<script type="text/javascript">
  //Declare a FlickrRestStore data store.  This is used to access images from the
  //Flickr (www.flickr.com) photo sharing website.
  var flickrRestStore = new dojox.data.FlickrRestStore();
  //Define the request, with a count of 20, so 20 items will be requested with
  //each request. The query specifies information used to access Flickr,
  //including a user ID (optional) and API key (required).
  //You can also specify a sort order, tags to search for, and the matching
  //mode for the tags, which can be "any" or "all", which equate to boolean "or"
  //and a boolean "and" respectively
  var request = {
        query: {
            userid: "44153025@N00",//The Flickr user id to use
            apikey: "8c6803164dbc395fb7131c9d54843627",//An API key is required.
            sort: [{
                    descending: true //Use descending sort order, ascending is default.
                }
                ],
                tags: ["superhorse", "redbones", "beachvolleyball","dublin","croatia"],
                tag_mode: "any" //Match any of the tags
         },
         count: 20
  };

  dijit.byId('gallery1').setDataStore('imageItemStore', request, itemNameMap);
</script>
  <div id="gallery1" dojoType="dojox.image.Gallery"></div>
```

Setting the image width and height

```
<div id="gallery1" dojoType="dojox.image.Gallery" imageHeight="400" imageWidth="600"></div>
```

Setting Page Size and AutoLoad

It is possible to define how many images are requested from the data store with each request. This affects the performance. The larger the page size, the slower a request may be, but there will be fewer requests. The smaller the page size, the quicker a request may be, but there will be more requests. It is specified by altering the **pageSize** attribute.

By default, the Gallery will preload one page of images at a time. This gives a better user experience, as the user will have to wait less time to view an image. However, it may download more images than the user wishes to view. The autoloading of images can be disabled by setting the **autoLoad** attribute to "false".

```
<div id="gallery1" dojoType="dojox.image.Gallery" pageSize="50" autoLoad="false" ></div>
```

Changing the time interval in a SlideShow

The images in the large pane of the Gallery can be made to run a slide show by clicking its "Play" button. The amount of time between changing images can be configured by setting the **slideshowInterval** attribute to the number of seconds required.

```
<!--Set the interval to 5 seconds-->
<div id="gallery1" dojoType="dojox.image.Gallery" slideshowInterval="5" ></div>
```

Lightbox

TODO: Fill in info on dojox.image.Lightbox / finish / elaborate more

dojox.image.Lightbox

A clone of Lightbox2 for dojo. A Lightbox displays images in a modal dialog. It can display single images, or image groups, with smooth transition animations in between. You can populate the Lightbox group from a dojo.data datastore, or from markup.

Basic Usage

Being a clone, it is almost completely backwards compatible with Lightbox2. The Lightbox will display the image linked to in an anchor tag.

Using a Lightbox is as easy as adding a dojoType to a link tag:

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}
```

```
<a href="images/image1.jpg" dojoType="dojox.image.Lightbox">show image1</a>
```

Grouping images together is as simple as adding a group="" attribute. In this example, the first image is a single view, the other two popup a Lightbox with [previous] and [next] navigation, and 2 images in a group.

```
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */
.geshifilter {font-family: monospace;}
.geshifilter .imp {font-weight: bold; color: red;}
.geshifilter .kw1 {color: #000066; font-weight: bold;}
.geshifilter .kw2 {color: #003366; font-weight: bold;}
.geshifilter .kw3 {color: #000066;}
.geshifilter .co1 {color: #009900; font-style: italic;}
.geshifilter .coMULTI {color: #009900; font-style: italic;}
.geshifilter .es0 {color: #000099; font-weight: bold;}
.geshifilter .br0 {color: #66cc66;}
.geshifilter .st0 {color: #3366CC;}
.geshifilter .nu0 {color: #CC0000;}
.geshifilter .me1 {color: #006600;}
.geshifilter .re0 {color: #0066FF;}
```

```
<a href="images/image1.jpg" dojoType="dojox.image.Lightbox">show image1</a>
<a href="images/image2.jpg" dojoType="dojox.image.Lightbox" group="myGroup">show image2</a>
<a href="images/image3.jpg" dojoType="dojox.image.Lightbox" group="myGroup">show image3</a>
```

Your anchor text can be anything: a thumnail of the image to show, text, etc. It can also be anywhere on a page.

With JavaScript disabled, the natural link to the image will pass through, and display the image natively

Slideshow

[inline:slideshow.jpg]

The dojox.image.SlideShow widget displays a series of images, one at a time. It provides controls to move from image to the next or the previous image. It can also run an automated slideshow, moving from one image to the next every specified number of seconds.

Features

The Slideshow widget has the following features.

- Displays a single image at a time
- Provides navigation controls to move between images, and to start/stop the automated slideshow.
- Can optionally pre-load images in the background so that there is no visible delay when viewing images. While this uses more bandwidth, it provides a far better user experience.
- Links an image to a URL, so that clicking the image navigates to a specified web address. This is optional.
- Reads image data from an implementation of the dojo.data.api.Read interface. This means it is completely decoupled from any particular data source, and can be used with any data source at all as long as it is a valid dojo.data store. For example, the Slideshow can display images based on data in a simple text file using dojo.data.ItemFileReadStore, can show Flickr images using dojox.data.FlickrRestStore, or images from Picasa Web Albums using dojox.data.PicasaStore.
- Displays the title of each image, using a template that can be simply overridden.

Examples

Setting the Display Interval

It is possible to define the number of seconds between image transitions when running an automated slideshow. To do this, set the **slideshowInterval** attribute. For example, to set a three second interval between changing the displayed image, use the code below

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.SlideShow" id="slideshow1" slideshowInterval="3"> </div>
```

Overriding the Title Template

The Slideshow widget has a default title template that is used to display the title of the current image, as well as it's relative position. To do this, set the **titleTemplate** attribute. The supported place holders are :

- **@title** - The title of the image is placed here
- **@current** - The current index of the image is placed here
- **@total** - The total number of images is placed here.

For example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.SlideShow" id="slideshow1"
    titleTemplate="My title is '@title', this is image @current out of @total"
> </div>
```

Setting the Maximum Image Dimensions

To override the default height and width of the widget, set the **imageHeight** and **imageWidth** attributes. Images are automatically scaled to fit either the max height or width, depending on which of their dimensions is greater. E.g

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.SlideShow" id="slideshow1"
    imageWidth="600" imageHeight="300"
> </div>
```

Disabling the AutoLoad

The Slideshow widget automatically preloads a number of images in the background. While this generally provides a better user experience, it uses more bandwidth, so some users may want to disable it. To do so, set the **autoLoad** parameter to "false". e.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.SlideShow" id="slideshow1"
    autoLoad="false"
> </div>
```

This causes a delay when the user attempts to view an image, since it must wait to be loaded.

Disabling Resizing to Fit the Image

By default, if an image is less tall than the Slideshow widget, the widget resizes itself to fit to the image. In some circumstances this may be undesireable, such as when using an inflexible, fixed page layout. To disable this resizing behaviour, set the **fixedHeight** attribute to "true", e.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.SlideShow" id="slideshow1"
    fixedHeight="true"
> </div>
```

Setting the Data Store on the Slideshow

The Slideshow widget reads the image information from dojo.data objects. To set the data source for the Slideshow widget, first create one of the available data stores, such as the dojo.data.ItemFileReadStore or dojox.data.FlickrRestStore. Next, create a request object, which optionally contains a query. e.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.SlideShow" id="slideshow1" > </div>
<div jsId="imageItemStore" dojoType="dojo.data.ItemFileReadStore" url="images.json"></div>
<script type="text/javascript">
dojo.addOnLoad(function() {
  //Define the request, saying that 20 records should be fetched at a time,
  //and to start at record 0
  var request= {count:20, start:0};
  //Tell the widget to request the "large" parameter, as different
  //stores may use different parameter names
```

```
  var itemNameMap = {imageLargeAttr: "large"};
  //Call the setDataStore function, passing it the data store, the request object,
  //and the name map.
  dijit.byId('slideshow1').setDataStore(imageItemStore, request, itemNameMap);
});
</script>
```

Subscribing to Slideshow Events

The Slideshow publishes information about its state, that can be subscribed to using Dojo's Publish/Subscribe system. Two pieces of information are published to a named *topic*:

- Current image - whenever the displayed image changes, a JSON object with the following attributes:
  - **index** - The current numeric index of the image, that is, it's index in the data store
  - **title** - The string title of the image, if any
  - **url** - The URL of the image

  The name of the topic is retrieved by calling the **getShowTopicName** method on the widget /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
  ```
  dojo.subscribe(
      dijit.byId('slideshow1').getShowTopicName(),
      function(packet) {
        alert("Got index: " + packet.index
                    + ", url: " + packet.url
                    + ", and title: " + packet.title);
  });
  ```
- Loaded Image - when an image finishes loading, whether in the background, or the currently displayed image, information is published about it. The name of the topic is retrieved by calling the **getLoadTopicName** on the widget. A Number is published, which is the index of the image in the data store. E.g. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
  ```
  dojo.subscribe(
      dijit.byId('slideshow1').getLoadTopicName(),
      function(index) {
        alert("Got index: " +index);
  });
  ```

Further Examples

For a full example of the Slideshow, see the test file at http://archive.dojotoolkit.org/nightly/dojotoolkit/dojox/image/tests/tes...

ThumbnailPicker

[inline:thumbpicker.gif]

The ThumbnailPicker is a widget that displays a series of images either horizontally or vertically, with controls to page through the images. It reads its image data from data stores, that is, implementations of the dojo.data.api.Read API.

When an image is clicked by the user, information regarding that image is published to a dojo topic, which can be used to integrate the ThumbnailPicker with other objects on the page.

The ThumbnailPicker can be configured in a number of ways:

- Number of visible images
- Data source
- Can be horizontal or vertical
- Enabling/disabling following hyperlinks when an image is selected
- Notification of load status for images

Examples

Number of Visible Images

To set the number of visible images, and thereby the width or height of horizontal and vertical widgets respectively, set the numberThumbs attribute, e.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
<div dojoType="dojox.image.ThumbnailPicker" id="picker1" numberThumbs="4"> </div>
```

Setting the Data Source

To set the data source for the ThumnailPicker widget, first create one of the available data stores, such as the dojo.data.ItemFileReadStore or dojox.data.FlickrRestStore. Next, create a request object, which optionally contains a query. e.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.ThumbnailPicker" id="picker1" > </div>
<div jsId="imageItemStore" dojoType="dojo.data.ItemFileReadStore" url="images.json"></div>
<script type="text/javascript">
dojo.addOnLoad(function() {
  //Define the request, saying that 20 records should be fetched at a time,
  //and to start at record 0
  var request= {count:20, start:0};
  //Tell the widget to request the "thumb" parameter, as different
  //stores may use different parameter names
  var itemNameMap = {imageThumbAttr: "thumb"};
  dijit.byId('picker1').setDataStore(imageItemStore, request, itemNameMap);
});
</script>
```

Using a Vertical Layout

To make the ThumbnailPicker display itself vertically, set the **isHorizontal** attribute to "false". To leave it as horizontal, either omit the **isHorizontal** attribute, or set it to "true", e.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.ThumbnailPicker" id="picker1" isHorizontal="false"> </div>
```

Enabling/disabling following hyperlinks

To enable following a hyperlink when a thumbnail image is clicked, set the **useHyperlink** attribute to "true". By default it is false. When hyperlinks are enabled, by default the URL is opened is a new window. To open the link in the current window, set the **hyperlinkTarget** attribute to "this". e.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.ThumbnailPicker" id="picker1" useHyperlink="true" hyperlinkTarget="this"> </div>
```

Notification of load status for images

The ThumbnailPicker can display a notification for each image stating whether another version of it has loaded or not, for example when it is combined with the dojox.image.Slideshow widget. When this is enabled, the ThumbnailPicker relies on other code calling it's **markImageLoaded** method to change the notification from its *loading* state to *loaded* state.

To enable the load state notifier, set the **useLoadNotifier** to "true". By default, it is disabled, since it only really makes sense to use it in combination with other widgets or elements on a page. e.g.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.image.ThumbnailPicker" id="picker1" useLoadNotifier="true"> </div>
```

Full Example

This example will put a horizontal and a vertical dojox.image.ThumbnailPicker widget on a page, with a variety of settings, and using separate data stores. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<html>
<head>
  <style type="text/css">
    @import "../resources/ThumbnailPicker.css";
  </style>

  <script type="text/javascript" src="dojo/dojo.js" djconfig="parseOnLoad:true, isDebug: true,
defaultTestTheme:'soria'"></script>
  <script type="text/javascript" src="/dijit/tests/_testCommon.js"></script>
  <script type="text/javascript" src="dojox/image/ThumbnailPicker.js"></script>

  <script type="text/javascript">
    /*
      The FlickrRestStore and the ItemFileReadStore data stores will be used to provide data to the widgets
    */
    dojo.require("dojox.data.FlickrRestStore");
    dojo.require("dojo.data.ItemFileReadStore");
    dojo.require("dojo.parser");  // find widgets

    /*
      Initializes the ThumbnailPicker with a data store that reads from the Flickr REST APIs.
    */
```

```
    function initFlickrWidget() {
      //Create a new FlickrRestStore
      var flickrRestStore = new dojox.data.FlickrRestStore();

      //Create a request object, containing a query with the
      //userid, apikey and (optional) sort data.
      //Extra query parameters 'tags' and 'tag_mode' are also
      //used to further filter the results
      var req = {
        query: {
          userid: "44153025@N00",//The Flickr user id to use
          apikey: "8c6803164dbc395fb7131c9d54843627",//An API key is required.
          sort: [
               {
              descending: true //Use descending sort order, ascending is default.
              }
               ],
          tags: ["superhorse", "redbones", "beachvolleyball","dublin","croatia"],
          tag_mode: "any" //Match any of the tags
        },
        start: 0, //start at record 0
        count: 20 //request 20 records each time a request is made
      };

      //Set the flickr data store on two of the dojox.image.ThumbnailPicker widgets
      dijit.byId('thumbPicker1').setDataStore(flickrRestStore, req);
    }

    /*
      Initializes the second ThumbnailPicker widget with a data store that
      reads information from a JSON URL.  This also tells the ThumbnailPicker
      the name of the JSON attributes to read from each data item retrieved
      from the JSON URL.
    */
    function initItemStoreWidget(){
      var itemRequest = {
        query: {},
        count: 20
      };
      var itemNameMap = {
        imageThumbAttr: "thumb",
        imageLargeAttr: "large"
      };

      //Set the dojo.data.ItemFileReadStore on two of the dojox.image.ThumbnailPicker widgets
      //Note the use of the 'itemNameMap', which tells the widget what attributes to
      //read from the store.  Look in the 'images.json' file in the same folder as this
      //file to see the data being read by the widget.
      dijit.byId('thumbPicker2').setDataStore(imageItemStore, itemRequest, itemNameMap);
    }

    //Subscribe to clicks on the thumbnails, and print out the information provided
    function doSubscribe(){
      function updateDiv(packet){
        alert("You selected the thumbnail:"
            + "Index: " + packet.index
            + "Url: " + packet.url
            + "Large Url: " + packet.largeUrl
            + "Title: " + packet.title
            + "Link: " + packet.link)
            ;
      }

      //When an image in the ThumbnailPicker is clicked on, it publishes
      //information on the image to a topic, whose name is found by calling
      //the 'getTopicName' function on the widget.
      dojo.subscribe(dijit.byId('thumbPicker1').getTopicName(), updateDiv);
      dojo.subscribe(dijit.byId('thumbPicker2').getTopicName(), updateDiv);
    }

    dojo.addOnLoad(initFlickrWidget);
    dojo.addOnLoad(initItemStoreWidget);
    dojo.addOnLoad(doSubscribe);
  </script>
</head>
<body>
  <h2>From FlickrRestStore:</h2>
  This ThumbnailPicker should have 8 thumbnails, with each of them linking
  to a URL when clicked on, changing the current page.  The cursor should also change when over an image.
  The widget is laid out in the default horizontal layout.
  <div id="thumbPicker1" dojoType="dojox.image.ThumbnailPicker" numberThumbs="8" useHyperlink="true"
        hyperlinkTarget="this"></div>
  <h2>From ItemFileReadStore:</h2>
  This ThumbnailPicker should have 5 thumbnails. Clicking on a thumbnail should NOT
  open a URL, and the cursor should not change when over an image that is not an arrow.
  The widget is laid out in a vertical layout.

  <div id="thumbPicker2" dojoType="dojox.image.ThumbnailPicker" numberThumbs="5" isClickable="false"
        isHorizontal="false"></div>

  Create an ItemFileReadStore that reads data from the file "images.json".  This is used by the ThumbnailPicker "thumbPicker2"
  <div jsId="imageItemStore" dojoType="dojo.data.ItemFileReadStore" url="images.json"></div>
</body>
</html>
```

The code above would result in a page that looks like the image below:

[inline:testPage.gif] Further examples can be found in the test file at http://archive.dojotoolkit.org/nightly/dojotoolkit/dojox/image/tests/tes...

TODO: put info on dojox.image.ThumbnailPicker

DojoX Layout

Version

Version 0.003

Release Date

Release date: 2007-06-01

Project state:

[expermental]

Credits

Pete Higgins Fredrik Johansson

Project description

placeholder for dijit.layout extensions. Currently only:

dojox.layout.FloatingPane - an extension on TitlePane for drag/drop operation, "docking" [minimize/maximize], and [soon] resizing.

dojox.layout.ResizeHandle - resize handle to attach to a domNode. works well on normal domNodes, but will require adding a resizeTo(w,h) method to any widget you wish to use it on. [experimental]

dojox.layout.ContentPane - an extension on dijit ContentPane. Supports inline scripts, inline styles, relative path adjustments and having a table tag as domNode.

dojox.layout.StretchPane -- an extension on dojox ContentPane Supports animated resizing and calculated sizes for nodes see test files for examples.

Dependencies

require Dojo Core, Dojo Base (fx), and Dijit.TitlePane, Dijit.layout.ContentPane

Installation:

checkout:

http://svn.dojotoolkit.org/var/src/dojo/dojox/layout/* http://svn.dojotoolkit.org/var/src/dojo/dijit/*

and require via: dojo.require("dojox.layout.FloatingPane"); or: dojo.require("dojox.layout.ContentPane");

Basic Usage:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dojox.layout.FloatingPane" title="my title">
                Content To be Floated
        </div>
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div dojoType="dojox.layout.ContentPane"
                adjustPaths="true"
                renderStyles="true"
                executeScripts="true"
                href="my/page/containing/scripts/and/styles/in/a/sub/folder.html"
        >
                Initial content, will be replace by href.
                paths in folder.html will be adjusted to match this page
        </div>
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<div id="panex" dojoType="dojox.layout.StretchPane"
                style="border: solid blue 1px; overflow:hidden"
                x="90" y="60" w="40+0.2*w" h="40"
     >Stretch horiz</div>
```

DojoX Offline

Note: This document is a copy of the Dojo Offline Tutorial hosted on Google Docs as of August 18, 2007. Every effort is made to keep this copy in sync with the original

Dojo Offline is an open-source toolkit that makes it easy to create sophisticated, offline web applications. It sits on top of Google Gears, a plugin from Google that helps extend web browsers with new functionality. Dojo Offline makes working with Google Gears easier; extends it with important functionality; creates a higher-level API than Google Gears provides; and exposes developer productivity features. In particular, Dojo Offline provides the following functionality:

- An offline widget that you can easily embed in your web page with just a few lines of code, automatically providing the user with network feedback, sync messages, offline instructions, and more
- A sync framework to help you store actions done while offline and sync them with a server once back on the network
- Automatic network and application-availability detection to determine when your application is on- or off-line so that you can take appropriate action
- A slurp() method that automatically scans the page and figures out all the resources that you need offline, including images, stylesheets, scripts, etc.; this is much easier than having to manually maintain which resources should be available offline, especially during development.
- Dojo Storage, an easy to use hashtable abstraction for storing offline data for when you don't need the heaviness of Google Gear's SQL abstraction; under the covers Dojo Storage saves its data into Google Gears
- Dojo SQL, an easy to use SQL layer that executes SQL statements and returns them as ordinary JavaScript objects
- New ENCRYPT() and DECRYPT() SQL keywords that you can mix in when using Dojo SQL, to get transparent cryptography for columns of data. Cryptography is done on a Google Worker Pool thread, so that the browser UI is responsive.
- Integration with the rest of Dojo, such as the Dojo Event system

Dojo Offline is built to work with the 0.9 release of Dojo, and will not work with older versions of Dojo, such as 0.4. It also requires the Google Gears plugin to function; if users do not have it installed Dojo Offline will prompt users to download it.

Using Dojo Offline

Let's dive in and start using Dojo Offline.

Download Dojo Offline SDK

First download the Dojo SDK and unzip it.

SDK Layout

When you unzip the SDK, you will find the following directories:

- dojo - The core of Dojo
- dijit - The Dojo widget system, named Dijit
- dojox - Optional Dojo extensions

Dojo Offline is an optional Dojo extension, and is therefore located in the dojox directory.

If you are looking to track down Dojo Offline's source code, most of it is in dojox/off/. The Dojo SQL layer is in dojox/_sql/, while Dojo Storage is in dojox/storage/. An autogenerated, JavaDoc-like API is available. When looking at the API docs or source code, many advanced options are available to deeply customize Dojo Offline; you can almost always safely ignore these unless you are an advanced user, and they usually say "*For advanced usage; most developers can ignore this*" in their documentation.

Demos

Dojo Offline has three main demos, a Hello World example, a more complicated web-based editor named Moxie that includes an example server-side written in Java, and a demo of Dojo Offline's SQL cryptography. You can play with hosted versions of the Hello World example here; a hosted version of the Moxie editor here; and the SQL cryptography demo here.

If you want to study the demo examples' source code, the Hello World example is located in dojox/off/demos/helloworld/, while the Moxie editor is located in dojox/off/demos/editor/. You can see the SQL cryptography demo source in dojox/_sql/demos/customers/customers.html.

The Hello World example has no server-side requirement; Moxie, however, includes a full Java server-side that you can use as a template and scaffolding. Running the server-side of Moxie is simple. Make sure you have Java 1.5+ installed, and then just type: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `java -jar editor.jar`
while inside the directory dojox/off/demos/editor/server/, and the Moxie server-side will start running, with an embedded web-server (Jetty) and relational database (Derby) already set up for you.

In a web-browser, you can now go to the following URL:

http://localhost:8000/dojox/off/demos/editor/editor.html
to run Moxie against the local server you just started.

For more details on the server-side portion of Moxie and how to build see the README file at /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox/off/demos/editor/server/README`.

Creating an Offline Application

Now that you have awareness of the SDK, it's file layout, and the provided demos, let's get down to illustrating what you need to do to create an offline-aware application using Dojo Offline.

Bring in Dojo and Dojo Offline

For our example source code we will pretend that you are creating your application with the Dojo Offline SDK in a subdirectory named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `offline-sdk`.

First, bring in Dojo and Dojo Offline: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script type="text/javascript" src="offline-sdk/dojo/dojo.js" djConfig="isDebug: true"></script>
<script type="text/javascript" src="offline-sdk/dojox/off/offline.js"></script>
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `isDebug` is a useful flag that when set to true will print out debug messages, and when set to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `false` will hide them. In your own code you can add /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `console.debug("some message");` to have these printed out to help with debugging. If you are using Firefox in conjunction with Firebug then these messages will print out to the Firebug console; otherwise they will print on to the web page itself, such as in Internet Explorer.

Notice that we do not bring in Dijit; Dojo Offline has no dependencies on Dijit, the Dojo Widget system, helping to keep your code size smaller. We only include it in the SDK because Moxie, the example editor, uses the Dijit rich text editor.

Next, bring in Dojo and Dojo Offline's style sheets:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<style type="text/css">
        @import "offline-sdk/dojo/resources/dojo.css";

        /* Bring in the CSS for the default Dojo Offline UI widget */
        @import "offline-sdk/dojox/off/resources/offline-widget.css";
</style>
```

Dojo Offline Widget

Dojo Offline includes a default offline widget that does much of the hard work of providing a good offline UI to your end-users. It updates the user with online and offline feedback; provides sync messages; and delivers help and instructions on using your application offline. If Dojo Offline did not provide these you would have to roll them yourself, so providing a default UI makes developing offline applications easier.

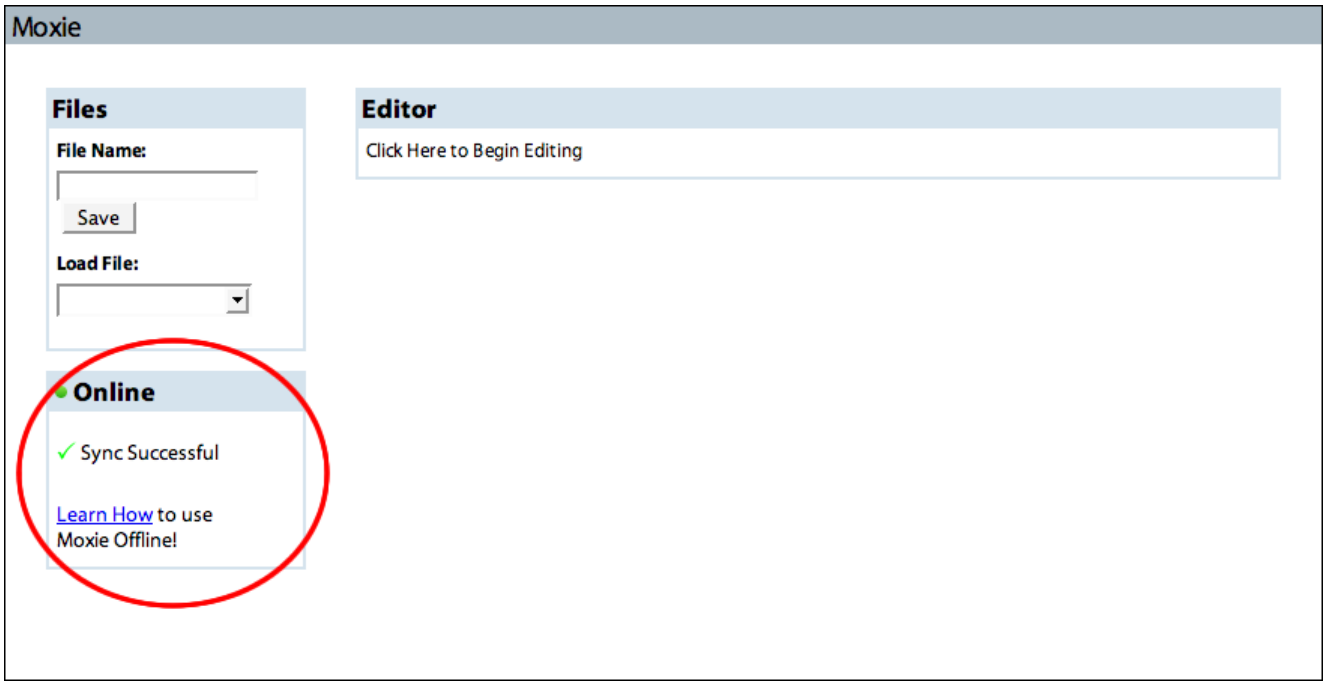
Getting this widget is as easy as adding a bit of HTML to your page, with a special, predefined ID:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<!--
        Place the Dojo Offline widget here; it will be automagically
        filled into any element with ID "dot-widget".
-->
<div id="dot-widget"></div>
```

When the page loads, Dojo Offline will automatically find an element with the ID dot-widget and automagically embed the offline widget.

The offline widget is a small, self-contained unit that provides feedback to the user, giving you a bunch of great functionality for free.

Here is a screenshot of Moxie, with the offline widget circled:



For more screenshots of the offline widget and a full description of what it gives you for free, see here. It is recommended that you use the offline widget in your own offline applications unless you are an advanced developer; information on customizing its look and feel, including dropping it, can be found here.

JavaScript

Let's take a look at our JavaScript now. All you have to do is set your application name:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// set our application name
dojox.off.ui.appName = "Example Application";
```

The Offline Widget will use your application name to customize it's user-interface; that is how the offline widget, for example, can insert "*Learn How to Use Hello World Offline*" into its instructions without you having to manually edit the offline widget's HTML.

Slurp! Bringing Your Application's Files Offline

For your application to be able to work offline, you must make sure that all of your JavaScript, HTML, CSS, etc. are available even when away from the network. While the browser may have these in its cache, you can't depend on this. The underlying Google Gears plugin that Dojo Offline uses makes it possible to specify what files you want to have available when offline, such as your JavaScript.

It can be tedious, however, to build up a full list of all your files, especially if you have many images and supporting libraries and are doing rapid development. Dojo Offline provides a single method, named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `slurp()`, that will slurp the page and automatically figure out all of the resources you need to have available offline:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// automatically "slurp" the page and
// capture the resources we need offline
dojox.off.files.slurp();
```

The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `slurp()` method is awesome; it will automatically scan your page and quickly figure out all your JavaScript and CSS files; grab any IMG tags in your source; and even grab any background URLs you might have in your attached CSS. The only thing it doesn't do is look at inline styles or scan your JavaScript for dynamic additions.

You can also manually add files if you want or need to; more information here.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color:

#009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `slurp()` has a nice debug method if you want to see all the files that have been slurped; make sure that Dojo's /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `isDebug` is /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `true`, and call /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.files.printURLs()` after the page and Dojo Offline are finished loading (more details on how to know when Dojo Offline is done loading in the next section).

During development, as you hit your offline application, it will always be pulled from the Google Gears file cache first, even if you are online. This can sometimes make development tricky and tedious. If you just made a change locally, hitting your web application, even if you are offline, will cause the older version to be pulled in first.

To make this process easier, I have created a bookmarklet that you can drag to your links toolbar in your browser. View this page to get the bookmarklets for Firefox and Internet Explorer.

These bookmarklets clear the Google Gears cache, removing all of the files that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `slurp()` added offline. Press it during development when you have made a new change that you want to show up; you must press it when you are at your web application (i.e. it won't work if you just press it when you are at a different or blank page). Then, refresh the page to see your change.

Knowing When Dojo Offline Can Be Used

Dojo Offline can't be used until the page is finished loading and Dojo Offline itself is finished initializing (such as the offline widget finishing being placed into the page). You should wait until the page and Dojo Offline are finished loading, and then your application can start doing its own custom thing:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var myApp = {
        initialize: function(){
                alert("Dojo Offline and the page are finished loading!");
        }
}
// Wait until Dojo Offline is ready
// before we initialize ourselves. When this gets called the page
// is also finished loading.
dojo.connect(dojox.off.ui, "onLoad", myApp, "initialize");

// tell Dojo Offline we are ready for it to initialize itself now
// that we have finished configuring it for our application
dojox.off.initialize();
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `myApp` is some object that will hold all of the methods for our application; /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `myApp.initialize()` is the method in particular that would initialize our application on page and Dojo Offline load in some way. We use /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.connect` to connect to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.ui.onLoad` event; when this fires, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `myApp.initialize()` is called. At this point we could begin to manipulate the DOM on the page, since it is loaded, or do further actions specific to your application; in this case we just print an alert message.

The final call to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.initialize()` is there to tell Dojo Offline that we are done configuring it, and that it can now initialize itself; we don't want Dojo Offline to initialize itself before we have set our /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `appName` and called /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `slurp()`, for example, so this final call tells Dojo Offline that we are finished doing configuration.

Let's look at all of our code so far in this tutorial:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<html>
	<head>
		<script type="text/javascript" src="offline-sdk/dojo/dojo.js" djConfig="isDebug: true"></script>
		<script type="text/javascript" src="offline-sdk/dojox/off/offline.js"></script>
		<style type="text/css">
			@import "offline-sdk/dojo/resources/dojo.css";

			/* Bring in the CSS for the default
			   Dojo Offline UI widget */
			@import "offline-sdk/dojox/off/resources/offline-widget.css";
		</style>
		<script>
			// set our application name
			dojox.off.ui.appName = "Example Application";
			// automatically "slurp" the page and
			// capture the resources we need offline
			dojox.off.files.slurp();
			var myApp = {
				initialize: function(){
					alert("Dojo Offline and the page are finished loading!");
				}
			}
			// Wait until Dojo Offline is ready
			// before we initialize ourselves. When this gets called the page
			// is also finished loading.
			dojo.connect(dojox.off.ui, "onLoad", myApp, "initialize");

			// tell Dojo Offline we are ready for it to initialize itself now
			// that we have finished configuring it for our application
			dojox.off.initialize();
		</script>
	</head>

	<body>
		<!--
			Place the Dojo Offline widget here; it will be automagically
			filled into any element with ID "dot-widget".
		-->
		<div id="dot-widget"></div>

	</body>
</html>
```

Important: I've noticed over and over in the frameworks I have created, such as Dojo Storage and the Really Simple History library, that developers get confused about an important point. Notice that configuring Dojo Offline must be done ***before the page loads***:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script>
	// set our application name
	dojox.off.ui.appName = "Example Application";
	// automatically "slurp" the page and
	// capture the resources we need offline
	dojox.off.files.slurp();
	// ...
	// Wait until Dojo Offline is ready
	// before we initialize ourselves. When this gets called the page
	// is also finished loading.
	dojo.connect(dojox.off.ui, "onLoad", myApp, "initialize");

	// tell Dojo Offline we are ready for it to initialize itself now
	// that we have finished configuring it for our application
	dojox.off.initialize();
</script>
```

Notice that our calls to things like /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.ui.appName` and the /* GeSHi (C) 2004 - 2007 Nigel McNie

(http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.connect` to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.ui` are *at the top-level* of the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `script` tag done before the page loads; if you were to put these into a function and call that after the page has loaded, Dojo Offline ***will not work***:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
<script>
// this is wrong -- don't do this because it will not work
window.onload = function(){
```

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// set our application name
        dojox.off.ui.appName = "Example Application";
        // automatically "slurp" the page and
        // capture the resources we need offline
        dojox.off.files.slurp();
        // etc.
}
</script>
```

At this point we have the shell of an offline application using Dojo Offline and Google Gears. Let's delve into common issues that come up with offline applications now, such as toggling your user-interface in different ways if you are on- or off-line; knowing the network status; syncing; and more.

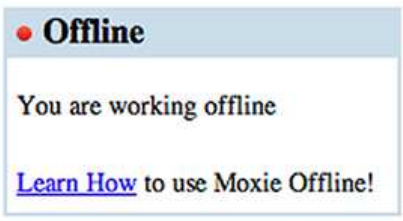
Network Status

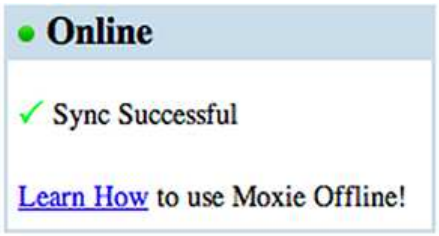
In the background Dojo Offline is checking to make sure that your web application is available on the network (for technical details on what it is doing see here). If your web application disappears or appears within a few seconds (five to thirty seconds), Dojo Offline will detect this and automatically inform your application so that you can move on- or offline, such as enabling or disabling UI elements that might not be available offline. Dojo Offline fires an event when the network status changes, which you can easily subscribe to using Dojo Connect:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
dojo.connect(dojox.off, "onNetwork", function(status){
   if(status == "online"){
      alert("We are online!");
   }else if(status == "offline"){
      alert("We are offline!");
   }
});
```

By default, the Dojo Offline UI widget also subscribes to these events, handling most of the work of informing the user when they are on- or off-line so you don't have to:





As you will see in more detail when we get to syncing later on in this tutorial, Dojo Offline also does an automatic sync when the network reappears:

Another useful property is /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.isOnline`. At any time in your code you can check /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.isOnline` to see if you are on- or off-line and change your applications behavior appropriately. This will be /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `true` if we are online and the application is available on the network, and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `false` otherwise.

A common pattern in offline applications is the need to have a UI element do different things when you are on- or off-line. For example, you could imagine a SEARCH button that when you are online will shoot off a network request to the server to do a search, while when offline will do a SQL search on the local Google Gears datastore. A common way to handle this is to use Dojo Connect and the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.isOnline` property:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #b1b100;} .geshifilter .kw2 {color: #000000; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .coMULTI {color: #808080; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #ff0000;} .geshifilter .nu0 {color: #cc66cc;} .geshifilter .sc0 {color: #00bbdd;} .geshifilter .sc1 {color: #ddbb00;} .geshifilter .sc2 {color: #009900;}

```
var searchButton = dojo.byId("searchButton");
dojox.connect(searchButton, "onclick", function(evt){
   if(dojox.off.isOnline){ searchOnline(); }
   else{ searchOffline(); }
});
```

This can be a nice pattern to separate on- and off-line code and easily toggle between them without littering lower level code with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.isOnline` network checks; just do it at the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.connect` level instead, which is cleaner.

Storing Offline Data

An offline application is useless if it does not have access to data. Dojo Offline provides two ways to store data, depending on your needs. The first is Dojo Storage, which provides a simple, persistent hash table abstraction; and Dojo SQL, which provides SQL based access to your data.

Dojo Storage

Dojo Storage contains a very simple hash table abstraction, providing methods such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `put()` and /*

GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `get()`, which you can use to quickly store data without having to delve into SQL. Under the covers it saves this data into the Google Gears persistent storage system.

Saving data is easy. You can store simple strings:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojox.storage.put("someKey", "someValue");
```

or more complicated JavaScript objects:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var car = {type: "Nissan", color: "white", price: 20000, optional:"air-conditioner, stereo"};
dojox.storage.put("complexKey", car);
```

JavaScript objects are automatically saved as they are. Note, however, that browser objects will not get serialized, such as references to the DOM or an XMLHttpRequest object.

Also note that key names can only be letters, numbers, and the under score character. Spaces are not allowed.

Loading data is just as easy; if you stored a JavaScript object, it will be returned to you as a JavaScript object:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var value = dojox.storage.get("someKey");
  var car = dojox.storage.get("complexKey");
```

If the object is not found, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `null` is returned.

Further info and advanced usage on Dojo Storage can be found here.

Dojo SQL

Dojo Storage can be great, but some times you need the full power of a relational data store. Enter Dojo SQL.

Dojo SQL is an easy to use, thin layer over Google Gear's relational storage layer. Executing SQL is easy:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var results = dojox.sql("SELECT * FROM DOCUMENTS");
```

Results are returned as ordinary JavaScript objects, unlike Google Gears, which makes working with SQL much easier. For example, if you have created a table named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `DOCUMENTS`:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojox.sql("CREATE TABLE IF NOT EXISTS DOCUMENTS ("
                        + "fileName          TEXT NOT NULL PRIMARY KEY UNIQUE, "
                        + "content            TEXT NOT NULL) ");
```

that has five rows in it (i.e. five documents), and you call /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojox.sql("SELECT *
```

```
FROM DOCUMENTS")
```
, an array will be returned that has five rows, one for each document. Dojo SQL creates an object for each row, automatically taking each of the column names, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `fileName` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `content`, and using those as the object literals: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var results = dojox.sql("SELECT * FROM DOCUMENTS");
// document 1
alert("The first documents file name is " + results[0].fileName + " and it's content is " + results[0].content);
// document 2
alert("The second documents file name is " + results[1].fileName + " and it's content is " + results[1].content);
```

Inserting data is just as easy:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojox.sql("INSERT INTO DOCUMENTS (fileName, content) VALUES (?, ?)",fileName,
contents);
```

Simply put a question mark for each parameter in the SQL, and then provide the actual variables that will fill them in as variable length arguments at the end. You can use this for any SQL statement where you want to provide a parameter:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojox.sql("SELECT * FROM DOCUMENTS WHERE fileName = ?", someFileName);
```

For a full list of SQL statements that can be executed by SQLite, see here. The SQL to do full-text indexing can be found here.

Dojo SQL automatically handles opening and closing the database; for advanced usage information on how to manually do this yourself, which you normally shouldn't need to do, see here.

Encrypting Offline Data

A common issue with offline web applications is that they must download data to be stored on the local machine. If the machine is a laptop, sensitive data could be stored that would be at risk if the laptop is stolen. For example, imagine you are dealing with an application that is downloading student records with social security numbers; you don't want to have these stored in the clear if the laptop is stolen.

To address this use-case, Dojo SQL includes a cool feature that makes it easy to encrypt and decrypt specific columns of data. For example, imagine we have a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `CUSTOMERS` table with three columns, a last name, a first name, and a social security number:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojox.sql("CREATE TABLE CUSTOMERS ("
                                        + "last_name TEXT, "
                                        + "first_name TEXT, "
                                        + "social_security TEXT"
                                + ")"
        );
```

For the first and last names, we don't care if they are stored in the clear. However, for the social security column we would like to encrypt it.

Dojo SQL adds two new SQL keywords that make this easy, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ENCRYPT()` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `DECRYPT()`. Let's see how to use them.

To encrypt the SS number, we would do the following:
/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var password = "foobar";
dojox.sql("INSERT INTO CUSTOMERS VALUES (?, ?, ENCRYPT(?))", "Neuberg", "Brad", "555-34-8962",
                password,
                function(results, error, errorMsg){
                        if(error){ alert(errorMsg); return; }
                });
```

In the example above, we provide our three /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `INSERT` parameters as usual; however, for the last one, the social security number, we put the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ENCRYPT(?)` keyword around it. After providing these as variable arguments at the end, we provide a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `password`. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `password` is used for encryption rather than a key -- it is passed into the underlying cryptographic system; you should **not** store this password as a cookie or into Dojo Storage or Dojo SQL. Instead, the user should be prompted to enter it when they start using your application. If you store it then a laptop thief could simply use it to unlock the local data store.

The final argument is a callback. If you use the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ENCRYPT` or /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `DECRYPT` keywords then the call to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.sql` becomes asychronous. This is because under the covers we spawn Google Gears Worker Pool threads to do the actual cryptography in such a way that the browser doesn't screech to a halt. When they are done the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `callback` will be called. If are doing decryption, as you will see below, then the results will have a normal Dojo SQL JavaScript results object, but with the values decrypted. If there was an error then /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `error` will be /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `true` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `errorMsg` will have a reason for the error.

Decryption is just as simple; simply put the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `DECRYPT` keyword around the result columns you want decrypted:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var password = "foobar";
dojox.sql("SELECT last_name, first_name, DECRYPT(social_security) FROM CUSTOMERS",
                password,
                function(results, error, errorMsg){
                        if(error){ alert(errorMsg); return; }
                        // go through decrypted results
                        alert("First customer's info: "
                                        + results[0].first_name + " "
                                        + results[0].last_name ", "
                                        + results[0].social_security);
                });
```

In this example we simply print out the decrypted results for the first row. If the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `password` is wrong then the decrypted results will still be decrypted; there is no way to detect an incorrect password attempt programatically.

You can combine several columns at once into a single /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ENCRYPT` statement, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ENCRYPT(?, ?, ?, ?)`, and you can do the same with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `DECRYPT` statements, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `DECRYPT(first_name, last_name, social_security)`.

Under the covers Dojo SQL's cryptography is powered by 256-bit AES, using the passphrase you provide to derive the key. Specifically, we use the JavaScript AES implementation given here if you would like to study how it works; special thanks to Chris Veness for contributing the AES encryption code to Dojo.

Loading Offline Data On Page Load

Whether you use Dojo Storage or Dojo SQL, there is an important scenario you need to think about. If the user loads your application while offline, you must initialize your application using your stored data rather than making a network call. You should do this after Dojo Offline has loaded:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
initialize: function(){
        // initialize our UI using offline data if necessary
        var documents = null;
        if (dojox.off.isOnline){
                // make a network call to get our data
                // ...
        }else{ // we are offline
                documents = dojox.storage.get("documents");
        }
}
```

In the example above, our /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `initialize()` method is called when Dojo Offline is done loading. Our UI must then get a list of documents to make available; if we are online (/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.isOnline`), then we just make a network call. If we are offline, then we just load our documents from offline storage, in this case Dojo Storage.

We cover how to get your downloaded data for offline use in the next section, covering syncing.

Syncing

As soon as you start creating offline applications syncing becomes an issue. However, if you are not careful, trying to figure out syncing can become an endless black hole that causes your project to spiral and fail. Syncing can either be relatively easy or infinitely complex, depending on how you approach the issue. In order to help, Dojo Offline includes both a syncing framework, named Dojo Sync, as well as a set of syncing guidelines to help you avoid sync land mines. We've thought through many of the hard issues and created a path for you to

navigate through the sync process.

Lets begin with the guidelines.

Sync Guidelines

First, **user's view syncing as a bug, not a feature**. They don't want to toy with sync interfaces -- they just want syncing to happen automatically.

Along these lines, **user's don't understand merge interfaces**. Developers are pretty comfortable with merging files, such as using Subversion or CVS. Users, however, don't care and don't understand merge interfaces. If you are popping up a complicated diff and merge UI as you do syncing, then you have failed.

Therefore, Dojo Sync recommends that **rather than showing merge interfaces when syncing, the client- or server-side should automatically make a best guess, without UI intervention by the user**. If a document was modified locally while offline, and also modified on the server by someone else, for example, they should be automatically merged without user intervention. If there is a conflict, you should make a best decision on which to keep based on your application. You could choose to keep the newest one, for example, and could also make a backup in the document's history list of the other change in case the user needs to retrieve something from it. **Do merging automatically, and decide the best way your application can support this**.

With all this said, **users still need basic UI feedback that syncing is occurring and where in the sync process they are**. Syncing should not be completely invisible; just as a browser has a basic network throbber to show you network activity is happening without showing the entire intricacy of the process, syncing is the same way. Users want to know that syncing is occurring; see from a high-level where the syncing is at (uploading, downloading, etc.); whether syncing succeeded or failed; and want the option of finding out why syncing failed or any automatic merges that might have happened. Even though users don't want to do merging, they some times want the option to look at a sync log to see what was merged. This should be done in a way that doesn't clutter the main sync UI (i.e. it should be a sync log link, for example, that would display a pop up window with a sync log of what was merged and what conflicted, and how the server did automatic merging).

Finally, **user's don't want to have to manually always kick off syncing -- they just want the application to work correctly offline and automatically start syncing at correct times**. In light of this, an application should always sync when it first loads and the network is present, bringing down a subset of data to make available offline. Users rarely know when they will be offline, so if the application simply always syncs then there should be some set of data available offline when the user loses the network. Likewise, when the network reappears, the application should automatically sync any local changes back to the server, with the user not needing to kick this off.

Dojo Sync Overview

Let's look at working with Dojo Sync and how it implements the guidelines laid out above.

The first thing to know about Dojo Sync is that it works in conjunction with the default Dojo Offline UI widget you embedded into your page earlier, doing much of the hard work of communicating the sync process to the user in a lightweight way. When you embed the offline widget you get a sync user-interface for free.

Second, syncing happens automatically, without the user having to initiate it. Dojo Sync automatically kicks off syncing when the web application first loads and we are on the network, and also if we are offline and we detect that network has come back online.

When syncing starts, three things happen in the following order:

1. Dojo Sync downloads and caches our web user-interface -- any files that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.files.slurp()` detected are now brought offline
2. We upload any data that was changed or created while offline
3. We download data to make available offline

As you will see in a bit, Dojo Sync does much of the work for you, and provides some specific places you hook into to help it do its job.

Because Dojo Sync hooks into the offline widget, the user-interface for these three steps is completely automated. Let's quickly see what this default sync UI looks like from an end-user's perspective, since this can help you as a programmer understand how to tie into Dojo Sync.

First, Dojo Sync downloads our web user-interface to make it available offline:

Next, our web application uploads any changes that were made while offline:

Finally, we download data to make it available offline:

If everything is successful, the user is notified:

If there was an error, the user is also notified:

The user can find more details about the error by pressing the *details* link; this will cause a pop up window to appear with details on why the sync error occurred:

>

Now that you have an overview of Dojo Sync and what its user-interface looks like, let's see how your application code actually hooks in.

Hooking Into Dojo Sync

As stated before, Dojo Sync does much of the heavy lifting for you around user-interface and syncing. Your application only has to hook into two parts of Dojo Sync to help out: downloading data to make available when offline, and uploading any changes that were done while away from the network. Downloading is the simplest.

Sync Downloading

During the syncing process, Dojo Sync fires sync events that you can subscribe to:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.connect(dojox.off.sync, "onSync", this, function(type){});
```

There are many different /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onSync` events, given by the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `type` variable, but you only need to be concerned with a few unless you are doing something unusual (you can see a full list of the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onSync` events here).

When the sync process reaches the downloading stage, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onSync` will fire with /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `type` equal to the string /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `download`. You should listen for the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `download` event and proceed to download your data. In the example code below, when the sync framework wants us to download, we make a network call to some web service that fetches new documents, for example:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojo.connect(dojox.off.sync, "onSync", this, function(type){
        if(type == "download"){
                // download new documents
                dojo.xhrGet({
                        url:        "/documents/download",
                        handleAs:       "javascript",
                        load:      function(data){
                                dojox.storage.put("documents", data);
                                dojox.off.sync.finishedDownloading();
                        },
                        error:    function(err){
                                err = err.message||err;
                                var message = "Unable to download our documents from server: "
                                                                        + err;
                        dojox.off.sync.finishedDownloading(false, message);
                        }
                });
        }
});
```

In the example code above, we first subscribe to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onSync` events. When a sync event fires, if it is the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color:

#009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `download` event, we then proceed to make an XMLHttpRequest call to some web service that downloads new documents, located at /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `"/documents/download"`, which returns JavaScript JSON as its results. We use Dojo's easy to use /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.xhrGet()` method to do so. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `xhrGet()` can take an /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `error` handler and a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `load` handler, as well as allowing us to specify how the return results should be handled. In the example above, our /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `handleAs: "javascript"` parameter says that the results will just be handed to us as a JavaScript object.

Let's look at the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `load()` method first. If the web service returned correctly, we simply take the data returned, which would be our list of documents, and store it right into Dojo Storage with a /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `put()` call; we could also have used Dojo SQL here and iterated over the results to write into a relational store if we wanted.

The next line is important:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojox.off.sync.finishedDownloading()
```

You **must** call this method when you are done downloading. Since downloading data tends to be an asychronous process that involves talking to the network, Dojo Sync has no way to know when it can continue the syncing process after downloading is complete. When you call /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.finishedDownloading()` from your network callback, it tells Dojo Sync to continue. If you do not, Dojo Sync will get stuck at the downloading phase and your user-interface will just show "*Downloading new data...*" forever.

Let's look at the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `error` callback now. If an error occurs during downloading, you must also call /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.finishedDownloading()`, but with two arguments. The first should be /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `false` to indicate that downloading was unsuccessful; the second argument should be the reason that downloading failed, which will be reported in the sync log at the end of the sync session.

Sync Finished

We will see how to handle sync uploading in the next section, but first there is one more sync event that you will find useful in your code. When syncing is finished a sync event is fired with type /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `"finished"`:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.connect(dojox.off.sync, "onSync", this, function(type){
        if(type == "download"){
                // download data
                // ...
        }else if(type == "finished"){
                // refresh our UI when we are finished syncing
                var documents = dojox.storage.get("documents");
                dojo.forEach(documents, function(i){
                        // update the UI list of documents with this new document
                        // ...
                });
        }
});
```

In the example code above, when the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `finished` event is fired, we load all of our new documents from Dojo Storage and then use each one to update our user-interface. The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `finished` event can be useful for updating your UI once syncing is done.

Note that this event is fired whether an error ocurred or not during syncing; check /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.successful` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.error` for sync details. These are boolean properties that will tell you whether an error occurred or not.

Sync Uploading

Sync uploading is the trickiest part of syncing in general. Before we can tackle that, let's take a quick look at how Dojo Sync recommends you record user actions when they are done offline.

Offline User Actions

What do you do when a user is offline, but starts doing user actions that would normally cause a network call or which updates data?

Dojo Sync adopts what is known as a *transactional model* to solve this problem. When a user works offline, creating a new business contact or modifying an existing one, for example, we **create a record of this action** and update our local data. These actions are added to an **action log**.

For example, imagine that a user is working offline with a web application that has sales contacts in it. The user first modifies the contact record for *"Brad Neuberg,"* changing the phone number. Since we are not on the network we can't write this change to the server; instead, we create an action object to indicate that *"Brad Neuberg"* has been updated, and store it in an action log. We also update our local offline data with the new phone number. The user next creates a new contact, named *"John Doe"*; again, we create an action object to represent this offline action, and store the new contact.

When we go back online, all we have to do is replay the action log, which will simply execute each of the actions while online in the exact same order they were performed when offline. When we replay actions while online, all we are doing is essentially "simulating" that the user just did them right then, but very fast. The great thing is you can call existing web services that you might have to handle these actions once online.

In the example above, when the network first appears and we start syncing and reach the Sync Upload phase, we first replay the user action of updating the *"Brad Neuberg"* contact. Replaying this action executes doing what we normally do while online, which is to call some network service to update some contact. In our code example above, we might do an XMLHttpRequest call to POST the updated contact data to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `"/contacts/Neuberg/Brad"`. We then "replay" the user action to create the *"John Doe"* contact, which might do an HTTP POST to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;}

.javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `"/contacts/Doe/John"` with the new contact.

As recommended in the Sync Guidelines section above, these web-services should do **automatic merge and conflict** detection so that the user is not inundated with complicated merge UIs. If you insist on having a merge UI, see the Advanced Syncing section.

Let's see what all of this looks like from a code level now.

If the user is offline and does some action, we create action objects and add them to our action log:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
// some new client added by sales person
var client = {lastName: “Doe”, firstName: "John", phoneNumber: “555-222-1212”);
// create an Action object to represent this action --
// these can have anything you want in them -- as you will
// see later, they should have enough data to help you replay them
// when we go back online
var action = {name: "new contact", data: client};
// add this to our offline action log
dojox.off.sync.actions.add(action);
// persist the contact into offline storage
dojox.storage.put("John Doe", client);
```

In the example code above, we first have an object that represents the new client that was created, *John Doe*. Next, we create an /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `action` object to represent this offline action; the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `action` object can have any values you want -- it must simply have enough data for you to be able to replay this action when we go back online. In general, you will have an action /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `name`, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `"new contact"`, and some /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `data` that goes along with this action, which in this case is the newly created /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `client`.

Next, we add this action to our offline action log. The offline action log is automatically persisted, and records any actions that were done while offline, in the order in which they were performed by the user.

Finally, we persist this new or updated action into local offline storage; even though we can't communicate on the network, we want to keep our local data up to date with any offline changes that were made by the user.

Replaying

When the network reappears, Dojo Sync kicks off and starts automatic syncing. After refreshing the offline UI, Dojo Sync then attempts to upload any changes that were made while offline. Basically, Dojo Sync tries to replay the action log you built up while the user was offline, executing each one one at a time.

However, Dojo Sync doesn't know how to replay actions -- thats the job of your application. You register yourself to know when replaying has occurred, and Dojo Sync calls your replay function for each action, simply handing it the action object you created earlier while offline. You can now use this action object to execute the action, but this time while online.

Let's build this code segment up one piece at a time. First, we register to know when replaying has occurred:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.connect(dojox.off.sync.actions, "onReplay", this,
               function(action, actionLog){
               }
);
```

The function we give will be called over and over for each action that was added to the offline log. The offline log is just a persistent array, or list of actions in the order they were done by the user and added by you to the action log. This function is given each of the actions, one at a time, as the first argument /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `action`, and a reference to the action log itself, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `actionLog`. Your code should then look at the action and try to replay it:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.connect(dojox.off.sync.actions, "onReplay", this,
                        function(action, actionLog){
                                if(action.name == "new contact"){
                                }
                        }
);
```

In this case we just check the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `action.name` property, which we set earlier when the user actually performed this action. If it is the value /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `"new contact"`, then we try to simply create this new contact:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojo.connect(dojox.off.sync.actions, "onReplay", this,
        function(action, actionLog){
                if(action.name == "new contact"){
                        var contact = action.data;

                        // create this new contact on the network
                        dojo.xhrPost({
                                url:        "/contact/" + contact.lastName + "/"
                                                + contact.firstName,
                                content:        { "content": dojo.toJson(contact) },
                                error:          function(err){
                                                        var msg = "Unable to create contact "
                                                                +  contact.firstName + " "
                                                                + contact.lastName + ": " + err;
                                                        actionLog.haltReplay(msg);
                                                }
                                },
                                load:      function(data){
                                                        actionLog.continueReplay();
                                                }
                        });
                }
        }
);
```

There's alot going on in the code block above. First, if we see that the action was to create a new contact, /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `"new contact"`, then we actually shoot off an HTTP POST to create this new contact. We get the contact to create from the data:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `var contact = action.data;`

And then do the actual HTTP POST using this data. Our URL becomes the first and last name of the contact, or /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `/contact/Doe/John` in this case. Dojo's /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `xhrPost()` method takes the content of the post as an attribute named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;}

.javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `content`, which we provide. We transform our contact into a JSON string to easily send to the server with the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojo.toJson(contact)` method call.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `xhrPost()` calls /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `error()` or /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `load()` based on whether things were successful or not.

Notice that we call something called /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `actionLog.haltReplay(msg)` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `actionLog.continueReplay()`. Since most of the things we will do while replaying are asychronous, we must tell Dojo Sync when to continue replaying and whether an error occurred. If the action was replayed successfully, you should call /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `actionLog.continueReplay()`, such as we do above in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `load()` method. If an error occurred, you should tell Dojo Sync to halt replaying by calling /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `actionLog.haltReplay(msg)`, providing a message string on what the error was -- this error message will be written into the sync log for users to see if they want.

Remember that these web-services should do automatic merge and conflict detection so that the user is not inundated with complicated merge UIs.

One final property that is useful is /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.actions.isReplaying`. Sometimes you want to share the same network code whether you are online or replaying an offline action, since you don't want to have to rewrite the network call. In this case in your /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `load` or /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `error` callbacks you can see if replaying is occurring, and if so tell the action log to continue replaying or to halt. If you are not replaying, you would just handle the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `load` or /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `error` as normal.

Speeding Up Syncing: /* GeSHi (C) 2004 - 2007 Nigel McNie

(http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `version.js`

During syncing, we always refresh the list of offline files. This is somewhat due to limitations in the underlying Google Gears APIs we use (for the technically inclined, we use Google Gear's /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ResourceStore` instead of it's /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `ManagedResourceStore`). This can slow down syncing considerably, however, since every time we sync we also pull down the offline files.

Dojo Offline has an optional feature to speed this up. Your application can have an optional /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `version.js` file bundled in the same directory as it. During syncing, Dojo Offline will read this file to see if the version of the application has changed. If it has then we go ahead and refresh all the offline files; if not, then we skip this step.

The /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `version.js` file is very simple; it should simply have the version inside of it:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `"07-12-2007.9"`

This can be a string or number; it does not matter. We simply look to see if the value has changed. We also force an offline file refresh if you are in debug mode (i.e. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `djConfig.isDebug` is /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `true`), or if you just debugged the prior time the page was loaded.

Simply place this file in the same directory as your main HTML file.

This is all your should need to know to work with Dojo Sync.

Further Resources

The following resources are available to help you while developing with Dojo Offline. In addition, this tutorial has extensive reference information below this section in an Addendum that provides further technical information and advanced usage of the Dojo Offline library.

- Dojo Offline Home Page
- Google Gears Home Page
- Google Gears Developer Documentation
- Dojo Home Page
- Download Dojo (bundled Dojo Offline)
- Dojo Offline Source Code
  - dojox.off
  - dojox.sql
  - dojox.storage
- API Documentation
- Available SQL statements
- SQL for Full Text Indexing
- Utility Bookmarklets
- Run Demos:
  - Hello World
  - Moxie

    - Customers Encryption Demo
- View Demo Source Code:
    - Hello World
    - Moxie
    - Customers Encryption Demo
- Moxie Server-Side README
- Moxie Server Side Source Code

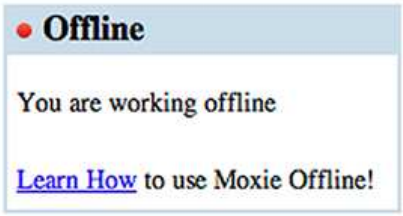
Addendum

The following sections provide further information and advanced usage information useful for curious developers. You can safely ignore these.

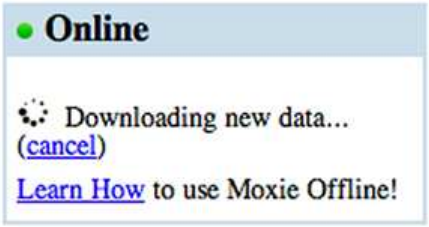
More Info: Offline Widget Walkthrough

Let's take a look at what the offline widget looks like, to see all the stuff it does for you without lifting a finger - you can skip this section if you want, but it can be helpful to see what the offline widget looks like and what it does.

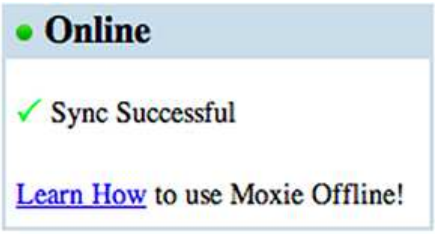
Let's peek at the offline widget a bit more, showing you its various states. If the user is offline, the offline widget will look like the following:



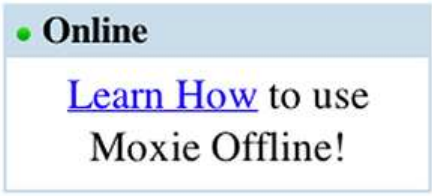
When the user goes back online, Dojo Offline automatically detects that the network and application have reappeared, and begins syncing:



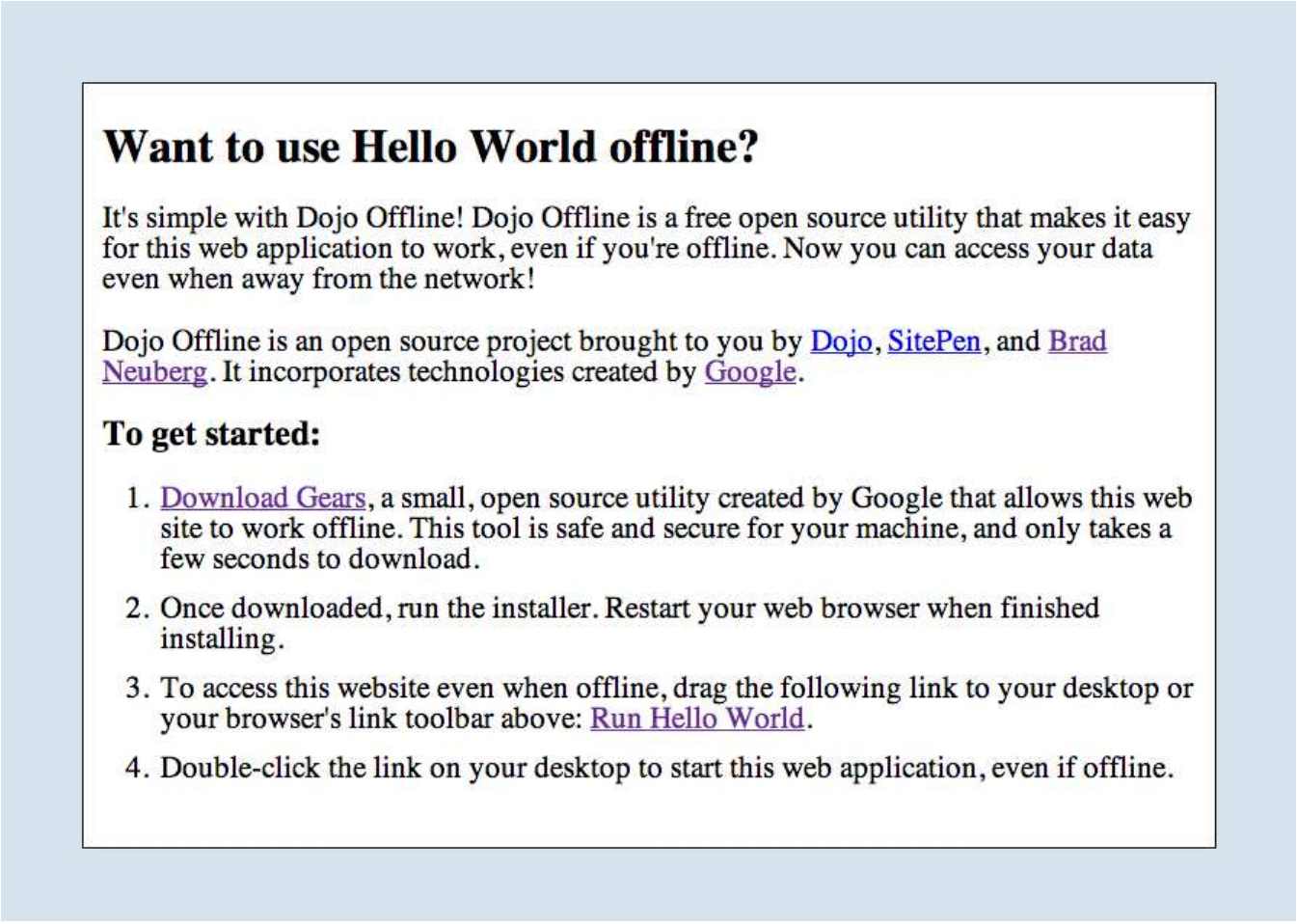
When done, the offline widget provides automatic feedback to the end-user:



If the user does not have Google Gears installed, then we provide a prominent Learn How link:



If the user clicks the *Learn How* link, a popup-window will appear that provides them instructions on how to install Google Gears and use your application offline:



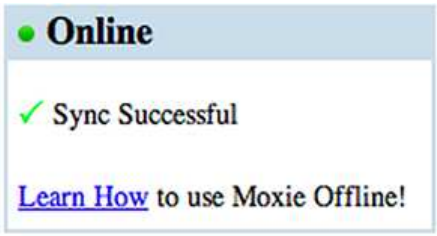
Notice that your application name is inserted into this *Learn How* page; the *Learn How* page automatically inserts your application's name, defined by setting /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.appName`, into this page, making it customized for your end-users. You do not have to edit it's HTML source.

The *Learn How* page also provides an easy hyperlink for your end-users to run your application when offline. When offline, a user would open their browser as normal and type your web application's URL into the location field, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `www.mywebapp.com`. However, Google Gears is sensitive to how a user types this; for example, if they type /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `mywebapp.com` instead of /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `www.mywebapp.com`, they might not be able to pull up the application offline and will get an error. Also, if your web app has a long name, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `http://mywebapp.com/foobar/something` then this is tedious and error-prone for an end user to type when offline. To solve this, the *Learn How* page automatically provides a hyperlink to your web application for a user to drag to their desktop or browser link toolbar:

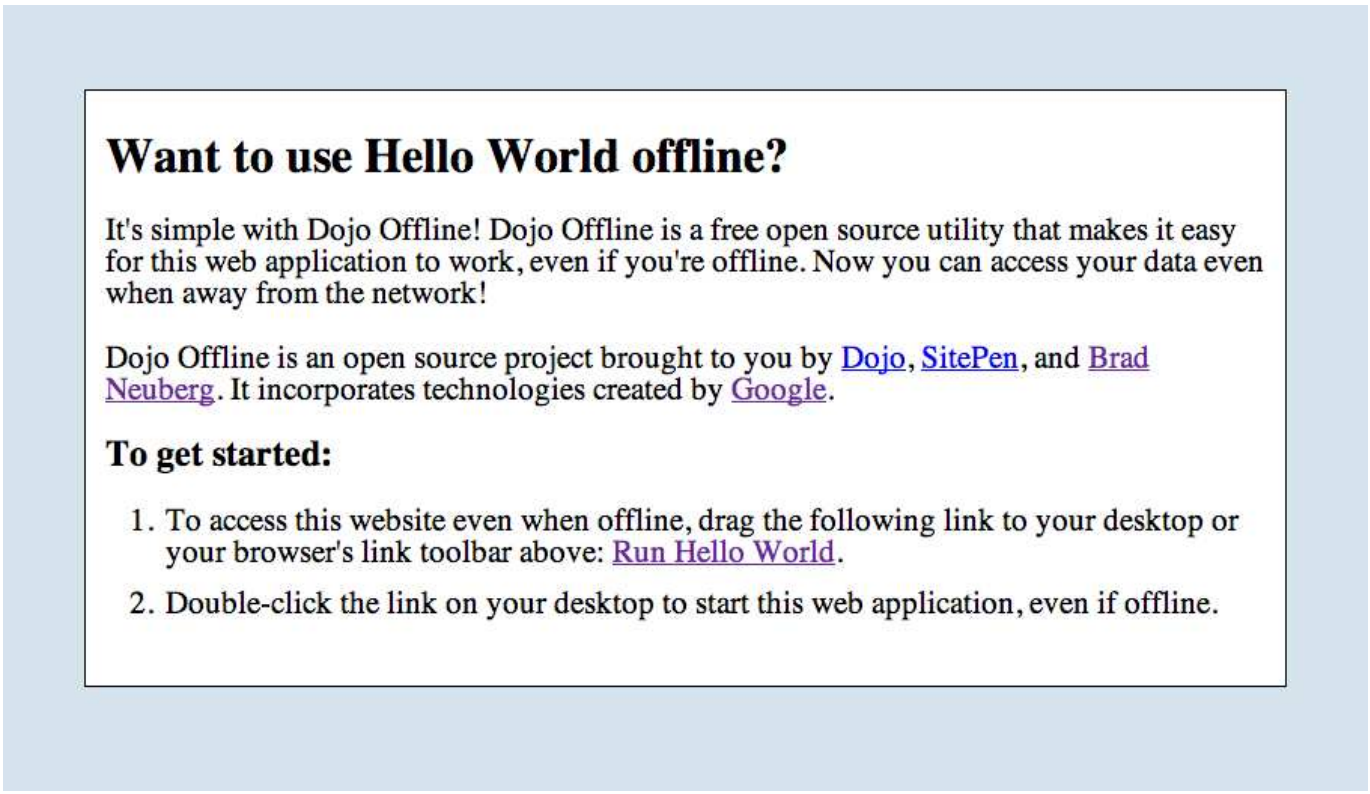
When offline, they can double-click this link to open a web browser and easily access your app when offline. Dojo Offline automatically

figures out what the URL of this link should be (instructions on how to change this default URL are available here).

If Google Gears is already installed, perhaps from working with another offline web application such as Google Reader, the offline widget still has the *Learn How* link at the bottom of the widget:



When clicked, though, the *Learn How* page automatically adapts to show less instructions, basically just providing the user with some minimal instructions on how to use things offline and provides the same draggable hyperlink:



Again, all of this is the kind of fit-and-finish that is hard to get right that Dojo Offline provides out of the box - it all comes for free.

If you want to customize the look and feel of the offline widget then learn more here.

More Info: Manually Adding Offline Files

Manually adding offline files is easy; you can even mix this with the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `slurp()` method if you want, so you can slurp the page for most of your resources, then perhaps manually add a few that weren't picked up because they are dynamically brought in through JavaScript.

To add a single file:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojox.off.files.cache("images/some_image_added_by_javascript.png");
```

To add many files at once:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0

{color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
```
dojox.off.files.cache([
                        "images/file1.png",
                        "images/file2.png",
                        "scripts/some_dynamic_script_tag.js"
                        ]);
```

You can call /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `cache()` as many times as you want; duplicates are automatically removed, and new cached additions are simply added to the end of our internal cached file list.

More Info: How Network Checking Works

Dojo Offline has a timer that every fifteen seconds is checking to make sure that your web application is available. By default, it looks for a simple static text file on your web server bundled with the Dojo Offline framework, located at /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox/off/network_check.txt`. This file is very short, with just the number /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `1` in it. We use a static text file rather than hitting your main web page since the main web application is probably dynamically generated and hitting it every 15 seconds would impose scalability limits, while a static text file is simple to serve.

You shouldn't need to change this part of the framework, but if you need to the interval for checking can be set on /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.NET_CHECK` and the availability URL can be changed from the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `network_check.txt` file to your own custom page by setting /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.availabilityURL`. If you want to completely disable this feature, you can set /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.doNetChecking` to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `false`.

Advanced Usage

This section describes various advanced usage scenarios that most users will never need to use; you can safely ignore these unless you are doing various strange, advanced things.

Advanced Usage: Learn How URL

On the *Learn How* page is a URL that a user can drag to their desktop to run your web application when offline. This URL is set automatically to the main page of your web app, but you can manually set it for unusual situations by assigning a URL to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.runLink` and change the title of the link by setting /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.runLinkTitle`; note that /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `runLinkTitle` should be normal text and not HTML.

Advanced Usage: Change The Dojo Offline Widget's Look and Feel

If you want to customize the look and feel of the offline widget you can. When we automagically embed the widget on page load, we actually grab a template file located at dojox/off/resources/offline-widget.html. This file simply has a bunch of HTML elements that we inline into the page. Every element has an ID, such as /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dot-widget-network-indicator`, that you can attach to with CSS to change. There is also a CSS file with defaults in dojox/off/resources/offline-widget.css.

You have several options for customization. The easiest is to just create new CSS rules for the element IDs in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `offline-widget.html` file; look up the name of the ID that you want to change, and then create a CSS rule. More advanced is to cut and paste the HTML from the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `offline-widget.html` file right into your HTML page, getting rid of the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dot-widget` automagic /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `DIV` you used in the past to have the template file inserted. Then, you can selectively drop any elements you want from the HTML, including adding new ones -- Dojo Offline is designed to be able to work even if you have dropped parts of the template.

If you want to change the Offline Widget's image icons for some reason, such as for branding or color palette reasons, the following properties can be set through JavaScript to a new URL:

- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.ui.onlineImagePath` - The green online ball
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.ui.offlineImagePath` - The red offline ball
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.ui.rollerImagePath` - The syncing spinner
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.ui.checkmarkImagePath` - The green success checkmark after syncing

For example, if you wanted to change the green online ball to a different icon, you would do the following:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
dojox.off.ui.onlineImagePath = "images/myonlineball.png";
```

If you want to brand the *Learn How* page in some special way, its HTML is at dojox/off/resources/learnhow.html /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} and its CSS is inside dojox/off/resources/offline-widget.css, near the bottom of the file.

To completely throw the Offline Widget away, you can simply set /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.ui.autoEmbed` to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `false`. Note that while Dojo Offline is designed to not use the offline widget, dropping the offline widget and doing

everything manually is currently not well tested yet. It is recommended that you use the offline widget in general.

Advanced Usage: Dojo Storage

This section provides further details on aspects of Dojo Storage that can help with advanced usage.

In addition to the standard /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `put()` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `get()`, there are some other methods on /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage` you might find useful:

- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.hasKey(key, namespace)` - Determines if some /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `key` is available; /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `namespace` is optional.
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.getKeys(namespace)` - Gets all of the keys available, as an array. namespace is optional. Example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.getKeys()`
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.clear(namespace)` - Clears all of the keys in the given /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `namespace`. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `namespace` is optional. Example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;}
  `dojox.storage.clear()`
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.remove(key, namespace)` - Removes the given key. /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `namespace` is optional. Example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}
  `dojox.storage.remove("someKey",`

"someValue")

- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.getNamespaces()` - Returns an array of all of the namespaces that are in use.
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.isValidKey(key)` - Will return /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `true` or /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `false` if the given /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `key` is valid (i.e. it will make sure that it has only letters, numbers, or under score characters).
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.put(key, value, resultsHandler, namespace)` - You can use an optional /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `namespace` to partition the keys you store -- useful for advanced usage when you have several different sets of data that might have similar keys. Set /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `resultsHandler` to /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `null` when doing this: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.put("someKey", "someValue", null, "someNamespace")`
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.get(key, namespace)` - You can also use an optional /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `namespace` here when retrieving keys. Example: /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.get("someKey", "someNamespace")`

Some readers of this tutorial might have used Dojo Storage in the past, before the new Google Gears and Dojo Offline integration. Some important notes. First, Dojo Offline provides a new Dojo Storage Provider named /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `GearsStorageProvider`, which stores it's hash table data into Google Gears. Second, we do not bundle any of the other storage providers with the Dojo Offline build, such as the Flash and Firefox storage providers, since they are not needed when used in conjunction with Google Gears and take up valuable space. Finally, you will notice that we did not give a callback to the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `put()` example above like in the past; when using Dojo Storage in conjunction with Dojo Offline and Google Gears, /* GeSHi (C) 2004 -

2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `put()` calls happen immediately and won't be denied, so you do not need to provide a callback. This makes doing /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `put()` calls much easier.

You can safely ignore knowing this, but it can be useful for some applications. The default namespace, if none is specified for the various Dojo Storage methods that take an optional namespace, is in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.DEFAULT_NAMESPACE`, which has the value /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `default`. The underlying Google Gears database is used to store the Dojo Storage values; the table name is in /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.storage.TABLE_NAME` (it is actually defined in the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox/storage/GearsStorageProvider.js` file), and it's value is /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `__DOJO_STORAGE`. Knowing these values can help with debugging in some cases so you can peer at them in the underlying Google Gears data store if necessary.

Advanced Usage: Dojo SQL Open/Close

The underlying Google Gears SQL data store must be opened before SQL can be executed, and should be closed when it is not used, though it will recover if you do not close it. Dojo SQL automatically does this open/close routine to simplify your code. When you execute a SQL statement with Dojo SQL, it will automatically open the database, execute your SQL, and close it. If you are working on performance sensitive code, where you have a for loop for example that is rapidly executing hundreds of inserts, this behavior could slow your app down considerably. In this case, you can just manually call /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.sql.open()` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.sql.close()` yourself, and Dojo SQL will automatically detect this and leave the database connection alone and open:

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
var bigData = getBigData(); // get an array that has a bunch of data in it
dojox.sql.open();
for(var i = 0; i < bigData.length; i++){
    dojox.sql("INSERT INTO MYTABLE (LAST_NAME, FIRST_NAME) VALUES (?, ?)", bigData[i].lastName, bigData[i].firstName);
}
dojox.sql.close();
```

Google Gears also supports having multiple named relational data stores for a single application; Dojo SQL does not currently expose this functionality, and stores its dataset as a Google Gears relational data store under the name given by /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.sql.dbName`, which is currently /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `PersistentStorage`. Note that if you have multiple applications running on the same web site they will create their tables in the same data store, the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `PersistentStorage` one, which can

create conflicts; this is a known issue that will be fixed in the future.

Advanced Usage: Dojo Sync

The Dojo Sync /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `onSync` event fires more than just the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `download` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `finished` events touched on in the tutorial above. Here is the full list of event types if you want to do deep customization, such as updating your UI to display the progress of syncing (note that the default Dojo Offline UI widget does this for you if you choose to pull that in). Most of these are only appropriate for advanced usage and can be safely ignored:

- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `start`- syncing has started
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `refreshFiles` - syncing will begin refreshing our offline file cache
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `upload` - syncing will begin uploading any local data changes we have on the client. This event is fired before we fire the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.actions.onReplay` event for each action to replay; use it to completely over-ride the replaying behavior and prevent it entirely, perhaps rolling your own sync protocol if needed.
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `download` - syncing will begin downloading any new data that is needed into persistent storage. Applications are required to implement this themselves, storing the required data into persistent local storage using Dojo Storage or Dojo SQL.
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `finished` - syncing is finished; this will be called whether an error ocurred or not; check /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.successful` and /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.error` for sync details
- /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `cancel` - Fired when canceling has been initiated; canceling will be attempted, followed by the sync event /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `finished`.

If you want to pop up a merge UI, you can create your own and choose to pop it up during your custom /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;}

`onReplay()` callback during replaying.

If you don't want to use the transactional action log system for syncing, register for the /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `upload` event and simply do uploading however you want.

If you want to completely override the sync process in general and roll your own sync infrastructure, override /* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .javascript .imp {font-weight: bold; color: red;} .javascript .kw1 {color: #000066; font-weight: bold;} .javascript .kw2 {color: #003366; font-weight: bold;} .javascript .kw3 {color: #000066;} .javascript .co1 {color: #009900; font-style: italic;} .javascript .coMULTI {color: #009900; font-style: italic;} .javascript .es0 {color: #000099; font-weight: bold;} .javascript .br0 {color: #66cc66;} .javascript .st0 {color: #3366CC;} .javascript .nu0 {color: #CC0000;} .javascript .me1 {color: #006600;} .javascript .re0 {color: #0066FF;} `dojox.off.sync.synchronize()` with your own logic; make sure to refresh the offline files as a minimum, though. Rolling your own sync infrastructure is not well supported, however, and is not recommended.

DojoX Presentation

Version

Version 0.1

Release Date

Release date: 06/25/2007

Project state:

prototype / experimental

Credits

pete higgins

Project description

This is the presentation base class. It provides a mechanism for various display-oriented tasks. It includes a powerpoint-esque engine [prototype] and an image SlideShow [experimental].

Dependencies:

dojox.presentation requires both Dojo Base, Dojo FX Core, and Dijit system(s).

Documentation

See the Dojo API tool (http://dojotoolkit.org/api)

Installation instructions

This package is self-contained, but needs Dijit sytem.

Grab the following from the Dojo SVN Repository:

svn co http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/presentation* svn co http://svn.dojotoolkit.org/var/src/dojo/dijit/*

into your: /dojo root folder [checkout/release root]

and require in dependancies via dojo.require('dojox.presentation');

if you are using SlideShow, you must:

dojo.require("dojox.presentation.SlideShow");

see /dojox/presentation/tests/test_presentation.html for example usage, but basically the structure is this:

```
presentation />
        Slide />
        Slide />
                Text Outside of Part is Static
                Part />
                Part />
                Action forPart/>
                Action forPart/>
        Slide href="remote.html" />
        Slide />
                Part />
                Action forPart/>
/presentation>
```

DojoX String Utilities

Version

0.9

Release date

05/08/2007

Project state

stable

Project authors

Ben Lowery, Tom Trenka

Project description

The DojoX String utilties project is a placeholder for miscellaneous string utility functions. At the time of writing, only the Builder object has been added; but we anticipate other string utilities may end up living here as well.

Dependencies

Dojo Core (package loader).

Documentation

See the Dojo Toolkit API docs (http://dojotooLkit.org/api), dojo.string.Builder.

Installation instructions

Grab the following from the Dojo SVN Repository: http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/string/*

Install into the following directory structure: /dojox/string/

...which should be at the same level as your Dojo checkout.

DojoX Timing

Version

Version 0.1.0

Release Date

Release date: 08/08/2007

Project state:

expermental

Credits

Tom Trenka (ttrenka AT gmail.com): original Timer, Streamer, Thread and ThreadPool Wolfram Kriesing (http://wolfram.kriesing.de/blog/): Sequence Jonathan Bond-Caron (jbondc AT gmail.com): port of Timer and Streamer Pete Higgins (phiggins AT gmail.com): port of Sequence

Project description

DojoX Timing is a project that deals with any kind of advanced use of timing constructs. The central object, dojox.timing.Timer (included by default), is a simple object that fires a callback on each tick of the timer, as well as when starting or stopping it. The interval of each tick is settable, but the default is 1 second--useful for driving something such as a clock.

dojox.timing.Streamer is an object designed to facilitate streaming/buffer-type scenarios; it takes an input and an output function, will execute the output function onTick, and run the input function when the internal buffer gets beneath a certain threshold of items. This can be useful for something timed-- such as updating a data plot at every N interval, and getting new data from a source when there's less than X data points in the internal buffer (think real-time data updating).

dojox.timing.Sequencer is an object, similar to Streamer, that will allow you to set up a set of functions to be executed in a specific order, at specific intervals.

The DojoX Timing ThreadPool is a port from the original implementation in the f(m) library. It allows a user to feed a set of callback functions (wrapped in a Thread constructor) to a pool for background processing.

Dependencies:

DojoX Timing only relies on the Dojo Base.

Documentation

TBD.

Installation instructions

Grab the following from the Dojo SVN Repository: http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/timing.js http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/timing/*

Install into the following directory structure: /dojox/timing.js /dojox/timing/

...which should be at the same level as your Dojo checkout.

DojoX UUID

Version

Version 0.9

Release Date

Release date: 06/21/2007

Project state: beta

Project authors

Brian Douglas Skinner

Project description

DojoX UUID is the port of the original Dojo 0.4.x UUID classes. The UUID classes can be used to represent Universally Unique IDentifiers (UUIDs), as described in the IETF's RFC 4122: http://tools.ietf.org/html/rfc4122

The DojoX UUID classes provide support for generating both "time-based" UUIDs and lightweight "random" UUIDs. DojoX UUID does not yet have support for generating new "name-based" UUIDs, but the dojo.uuid.Uuid class can represent existing name-based UUIDs, such as UUIDs read from a file or from a server.

Dependencies:

DojoX UUID has no dependencies, outside of Dojo Core.

Documentation

See the API documentation for Dojo (http://dojotoolkit.org/api).

Installation instructions

Grab the following from the Dojo SVN Repository: http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/uuid.js http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/uuid/*

Install into the following directory structure: /dojox/uuid/

...which should be at the same level as your Dojo checkout.

DojoX Validate

Version

Version 0.01

Release Date

Release date: 07/12/2007

Project state: experimental / beta

Credits

Peter Higgins

Project description

Provides a set of validation functions to match values against known constants for use in form validation.

Dependencies:

Requires dojo base and dojo.regexp.

Installation instructions

get a current checkout of the class via:

svn co http://svn.dojotoolkit.org/dojo/trunk/dojox/validate*

install into your sibling-dojox folder to /dojox/validate, require into your dojo app via dojo.require("dojox.validate");

and you are on your way.

see the API tool for full description of methods provided.

DojoX Widgets

Version

Version 0.2

Release Date

Release date: 7/27/2007

Project state:

expermental | beta

Credits

Peter Higgins Karl Tiedt

Project description

This is a collection of standalone widgets for use in your website. Each individual widget is independant of the others.

Dependencies:

Each widget has it's own requirements and dependencies. Most inherit from dijit base-classes such as dijit._Widget, dijit._Templated, etc ... So we will assume the availablility of dojo (core), and dijit packages.

Documentation

Please refer to the API-tool, or in-line documentation.

Installation instructions

These are standalone Widgets, so putting the [widget].js file in your dojox/widget folder, and copying any files in the /dojox/widget/[widget]/ folder as supplements/templates/etc should be all you need to do.

eg: FisheyeList: /dojox/widget/FisheyeList.js /dojox/widget/FisheyeList/blank.gif /dojox/widget/FisheyeList/FisheyeList.css

should be all you need to use the Fisheye widget.

Toaster

[inline:Toaster.png]

A *toaster* is an inobtrusive mechanism for displaying messages, and has become popular in recent years. Like toast, the message "pops up" in the window corner, temporarily overlaying any content there. The message stays up for a certain duration, or until the user clicks on it.

Toasters are preferrable to alert() boxes. Alert() must always be modal, meaning all action on the page stops until the user clicks "OK". Toasters are non-modal, so the user can continue working and finish their thought before responding.

You can either set the message programmatically, or use dojo's publish/subscribe event feature. Publish/subscribe allows you to have several toasters, or several controls besides toasters, respond to a particular event.

Examples

The first example uses setContent() and show() to vary the message and display it.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<script type="text/javascript">
        dojo.require("dojox.widget.Toaster");
        dojo.require("dojo.parser");
</script>
...
<div dojoType="dojox.widget.Toaster" id="toast" positionDirection="br-left" duration="0"
           style="display:hide"></div>
<script>
   function surpriseMe() {
     dijit.byId('toast').setContent('Twinkies are now being served in the vending machine!','fatal',500);
     dijit.byId('toast').show();
   }
</script>
<input type="button" onclick="surpriseMe()" value="Click here"/>
```

The next example does the same thing, but uses the publish/subscribe model. The message coming over the topic must be of the form:

- "message" : a single string with the message text
- -OR- {message: "...", type: "...", duration: "..."}: where message is the message text, duration is as the attribute, and type is either:
  - fatal
  - error
  - warning
  - message
- in decreasing order of severity.

/* GeSHi (C) 2004 - 2007 Nigel McNie (http://qbnz.com/highlighter) */ .geshifilter {font-family: monospace;} .geshifilter .imp {font-weight: bold; color: red;} .geshifilter .kw1 {color: #000066; font-weight: bold;} .geshifilter .kw2 {color: #003366; font-weight: bold;} .geshifilter .kw3 {color: #000066;} .geshifilter .co1 {color: #009900; font-style: italic;} .geshifilter .coMULTI {color: #009900; font-style: italic;} .geshifilter .es0 {color: #000099; font-weight: bold;} .geshifilter .br0 {color: #66cc66;} .geshifilter .st0 {color: #3366CC;} .geshifilter .nu0 {color: #CC0000;} .geshifilter .me1 {color: #006600;} .geshifilter .re0 {color: #0066FF;}

```
<div dojoType="dojox.widget.Toaster" id="toast" positionDirection="br-left" duration="0"
           messageTopic = "testMessageTopic"
></div>
<script>
```

```
   function surpriseMe() {
      dojo.publish("testMessageTopic",
        [
                {
                message: "Twinkies are now being served in the vending machine!",
                type: "fatal",
                duration: 500
                }
        ]
      );
   }
</script>
<input type="button" onclick="surpriseMe()" value="Click here"/>
```

Dijit Types, Attributes, Events and Methods

dojox.widget.Toaster
Message that slides in from the corner of the screen, used for notifications like "new email".

Attributes

duration	integer	2000	Duration of the message, in ms. 0 means the user must acknowledge the message by clicking on it.
messageTopic	String or object	-	Subscription topic to monitor. When a page publishes a message on this topic, the toaster will pop up.
positionDirection	br-up br-left bl-up bl-right tr-down tr-left tl-down tl-right	br-up	Corner from which message slides into screen (e.g. br=bottom right) and direction of slide.
separator	html	<hr>	If more than one unacknowledged message is present separate them with this.

Events

onSelect	called when user clicks the message to acknowledge it

Methods

hide()	Hide the toaster
setContent(String message, String type, int duration)	Set the content of the message to message. type can be "message" ,"warning", "error", "fatal" and determines the color (hence the importance)of the message. Duration is in ms, like the duration attribute.
show()	Display the toaster

CSS Classes I'm not quite sure how to represent these yet. In this case, you set styles on dojoProgressBarColorLayer and set the class of the outer div to dojoProgressBarColor.

Accessibility

DojoX Wire

Version

Version 0.9

Release Date

Release date: 05/29/2007

Project state: beta

Project authors

Jared Jurkiewicz

Project description

The DojoX Wire project is a set of functions that build a generic data binding and service invocation library to simplify how data values across a wide variety of widget and non-widget JavaScript constructs are accessed, updated, and passed to and from services. It also provides a set of widgets within the dojox.wire.ml package to allow for declarative data binding definitions in addition to the programmatic APIs.

In essense, this project is an API to provide a simplified way of doing MVC patterns in the client.

Dependencies:

DojoX Wire has dependencies on core dojo, the dijit widget system (for classes in the dojox.wire.ml package), dojox.data, and the D.O.H. unit test framework.

Documentation:

See the Dojo API tool (http://dojotoolkit.org/api)

Installation instructions

Grab the following from the Dojo SVN Repository:
http://svn.dojotoolkit.org/dojo/dojox/trunk/wire.js
http://svn.dojotoolkit.org/dojo/dojox/trunk/wire/*

Install into the following directory structure: /dojox/wire/

...which should be at the same level as your Dojo checkout.

It should look like: /dojox/wire.js /dojox/wire/*

Require in dojox.wire for all baseline functions (dojox.wire.connect, dojox.wire.register, etc). For specific Wire classes, require in the appropriate dojox.wire..

DojoX XML Utilities

Version

Version 0.1

Release Date

Release date: 05/30/2007

Project state:

expermental

Credits

Tom Trenka : DomParser

Project description

The goal of DojoX XML Utilities is provide differing XML utilities for use in various places. Currently this includes a native JS DomParser, but will most likely be expanded to include things as dealing with x-browser forks (like the Sarissa project), various DOM utilites, and more.

Dependencies:

DojoX XML relies only on the Dojo Base package system.

Documentation

None at the time of writing. The only object is dojox.xml.DomParser (a singleton), which has one method: parse:

dojox.xml.DomParser.parse(xmlString)

Installation instructions

Grab the following from the Dojo SVN Repository: http://svn.dojotoolkit.org/var/src/dojo/dojox/trunk/xml/*

Install into the following directory structure: /dojox/xml/

...which should be at the same level as your Dojo checkout.

DocX - Documentation Under Development

Here is where documents come to be born. Ah, sweet mystery of life!

Example 3: Many Happy Returns

Problem Statement

Andy is consulting for a company, No Problem Corporation, who needs a Customer RMA (Return Materials Authorization) application. This company had hired another consulting firm before Andy. But their web solutions were ... well, honestly, they sucked. So now Andy is on the line to deliver something really, really good.

The application helps a customer get their RMA'ed equipment properly tagged and shipped. To do this, they use a web interface to assign RMA'ed serial numbers to boxes. The application must interact with WPS (World Parcel Service) and enforce WPS requirements. The boxes must:

1. Weigh 150 lbs or less;
2. Contain parts whose total worth is $5,000 or less.

Once the customer assigns parts to boxes, they can click "Print" and the request is sent to WPS'es web service, which returns nice neat set of labels. The labels have No Problem Corporation's WPS billing number, so the customer essentially ships them for free. They like that. And No Problem Corp. likes it because the labels have item information clearly printed on them. Makes it very easy to unpack.

Lessons Learned

First Revision

Design

The earlier consulting firm built a prototype web application. It presented the process in a wizard-like format. You pick a serial number and a box number from a drop down box and click Next. Then you do the second and third.

Problems

The test users liked the application fine ... until they made mistakes. They tried going backwards in the wizard, and instantly lost track of their progress. Many times they closed their browser and started over.

Second Revision

Design

They built a second one. This one had a grid of serial numbers and box numbers in the middle. They filled them out one row at a time. If they caught a mistake it was easy to go back and correct.

Problems

The trouble was, they never knew they had overfilled a box (more than 150 lbs.) until they clicked OK. Then they had to split boxes. But since they had already assigned boxes 1, 2, and 3, they needed to fill in box 4's in the middle of the grid. It looked weird, and test users usually ended up renumbering them all to make it look right. This application was released, but so many customers complained it was quickly retracted.

Dojo Awareness

Design

Andy, having read the Dojo book, instantly saw a way out. You could list the serial numbers in a tree on the left hand side. You could have little box icons on the right. Then you just drag and drop the items into the boxes. You get instant feedback on weight and value as you drop things in. To split a box, you simply create new one and drag the items from one box to another. Because the UI doesn't let you make a mistake, the Print button always works flawlessly.

Implementation

To create this application, Andy needs:

- Events
- RPC (Remote Procedure Call) to communicate with WPS
- Drag and Drop
- Trees
- Custom-made widgets to represent boxes

Hierarchical Lists using DnD and Data

Trees aren't quite ready yet.

So, here is a first pass at using the data and DnD services.

This post is a placeholder so others can refer to the code.

Building a Custom Widget Box

TBW

Drag and Drop

TBW

Saving the Box Contents on the Client

TBW

Calling Web Services

TBW

Alternate Table of Contents

Learn Dojo

Quick Installation
Example 1: Why Doesn't Anyone Fill Out Their Tax Forms?: Adding Dijit to a page, Dijit Themes, Field Validation, Trimming and Casifying Input, A11y Basics
Example 2: The Postman Always Clicks Twice: Using Layout Widgets for Split Screens and Tabbed Panes, Dijit Trees for Heirarchical Display, Dojo.data for File Format-Independent Data Retrieval The Dijit Grid,
Functions Used Everywhere

Use a Dijit Widget

Dijit at a Glance: Listing and Contact Sheet of Every Bundled Widget
Common Features
Form Widget
CheckBox, RadioButton, ToggleButton
ComboBox
FilteringSelect
InlineEditBox (0.9)
NumberSpinner
Slider
Textarea
TextBox family: Validation, Currency, Number, Date, Time
AccordionContainer
ContentPane
LayoutContainer
SplitContainer
StackContainer
TabContainer
Command Control
Button, ComboButton, DropDownButton
Menu
Toolbar
User Assistance and Feedback
ProgressBar
Tooltip
Dialog and TooltipDialog
TitlePane
Advanced Editing and Display
ColorPalette
InlineEditBox (1.0)
Editor
Tree

Make a Widget Do What You Want

Manipulating Widgets Through Code
Writing Your Own Widget Class

Change the Widget Look and Feel

Themes and Design
Common Elements
Overriding and Combining Themes
Writing Your Own Theme

Organize Your Code

Object Orientation
Modules

Locate Web Page Fragments with JavaScript

Selecting DOM Nodes with dojo.query

Enable Drag and Drop (DnD)

Drag and Drop

Connect to Server Data

Using dojo.data
XMLHttpRequest (XHR)

Globalize Your Web Page

i18n

Make Your Web Page Accessible

Accessibility
Web Accessibility Issues
Dojo Accessibility Strategy
Dojo Accessibility Resources

Debug and Tune Up Your Application

Get the Code from Subversion
Development Tools
Debugging Facilities
D.O.H. Unit Testing
Performance Optimization
The Package System and Custom Builds

Other Things

The Event System
Back Button
Other Functions
Cometd (client)
Dojo Offline
DojoX Collections
DojoX Cryptography
DojoX Data
DojoX DTL (Django Template Language)
DojoX FX
DojoX GFX
DojoX Grid
DojoX I/O
DojoX Image
DojoX Layout
DojoX Presentation
DojoX String Utilities
DojoX Timing
DojoX UUID
DojoX Validate
DojoX Widgets
DojoX Wire
DojoX XML Utilities

Localizing custom widgets with Dijit Templates

not sure if this belongs here or in another section

Showing, Hiding, Moving Text

TBW

Example 3: Many Happy Returns

Problem Statement

Andy is consulting for a company, No Problem Corporation, who needs a Customer RMA (Return Materials Authorization) application. This company had hired another consulting firm before Andy. But their web solutions were ... well, honestly, they sucked. So now Andy is on the line to deliver something really, really good.

The application helps a customer get their RMA'ed equipment properly tagged and shipped. To do this, they use a web interface to assign RMA'ed serial numbers to boxes. The application must interact with WPS (World Parcel Service) and enforce WPS requirements. The boxes must:

1. Weigh 150 lbs or less;
2. Contain parts whose total worth is $5,000 or less.

Once the customer assigns parts to boxes, they can click "Print" and the request is sent to WPS'es web service, which returns nice neat set of labels. The labels have No Problem Corporation's WPS billing number, so the customer essentially ships them for free. They like that. And No Problem Corp. likes it because the labels have item information clearly printed on them. Makes it very easy to unpack.

The earlier consulting firm built a prototype web application. It presented the process in a wizard-like format. You pick a serial number and a box number from a drop down box and click Next. Then you do the second and third.

The test users liked the application fine ... until they made mistakes. They tried going backwards in the wizard, and instantly lost track of their progress. Many times they closed their browser and started over.

They built a second one. This one had a grid of serial numbers and box numbers in the middle. They filled them out one row at a time. If

they caught a mistake it was easy to go back and correct.

The trouble was, they never knew they had overfilled a box (more than 150 lbs.) until they clicked OK. Then they had to split boxes. But since they had already assigned boxes 1, 2, and 3, they needed to fill in box 4's in the middle of the grid. It looked weird, and test users usually ended up renumbering them all to make it look right. This application was released, but so many customers complained it was quickly retracted.

Andy, having read the Dojo book, instantly saw a way out. You could list the serial numbers in a tree on the left hand side. You could have a tree of boxes on the right. Then you just drag and drop the items into the boxes. You get instant feedback on weight and value as you drop things in. To split a box, you simply create new one and drag the items from one box to another. Because the UI doesn't let you make a mistake, the Print button always works flawlessly.

Small Touches

TBW

Build-time optimizations

TBW: peller

The 404 problem
Like Dojo's main build system (link?) address the problem of multiple, often small HTTP hits by combining them into a single file.