Web Performance Boot Camp

OmniTI / Speed It Up

Who am I? @postwait on twitter

- Author of "Scalable Internet Architectures"
  Pearson, ISBN: 067232699X
  (and "Web Operations" by O'Reilly)

- CEO of OmniTI
  We build scalable and secure web applications

- I am an Engineer
  A practitioner of academic computing.
  IEEE member and Senior ACM member.
  On the Editorial Board of ACM's Queue magazine.

- I work on/with a lot of Open Source software:
  Apache, perl, Linux, Solaris, PostgreSQL,
  Varnish, Apache Traffic Server, Spread, Reconnoiter, etc.

- I have experience.
  I've had the unique opportunity to watch a great many catastrophes.
  I enjoy immersing myself in the pathology of architecture failures.

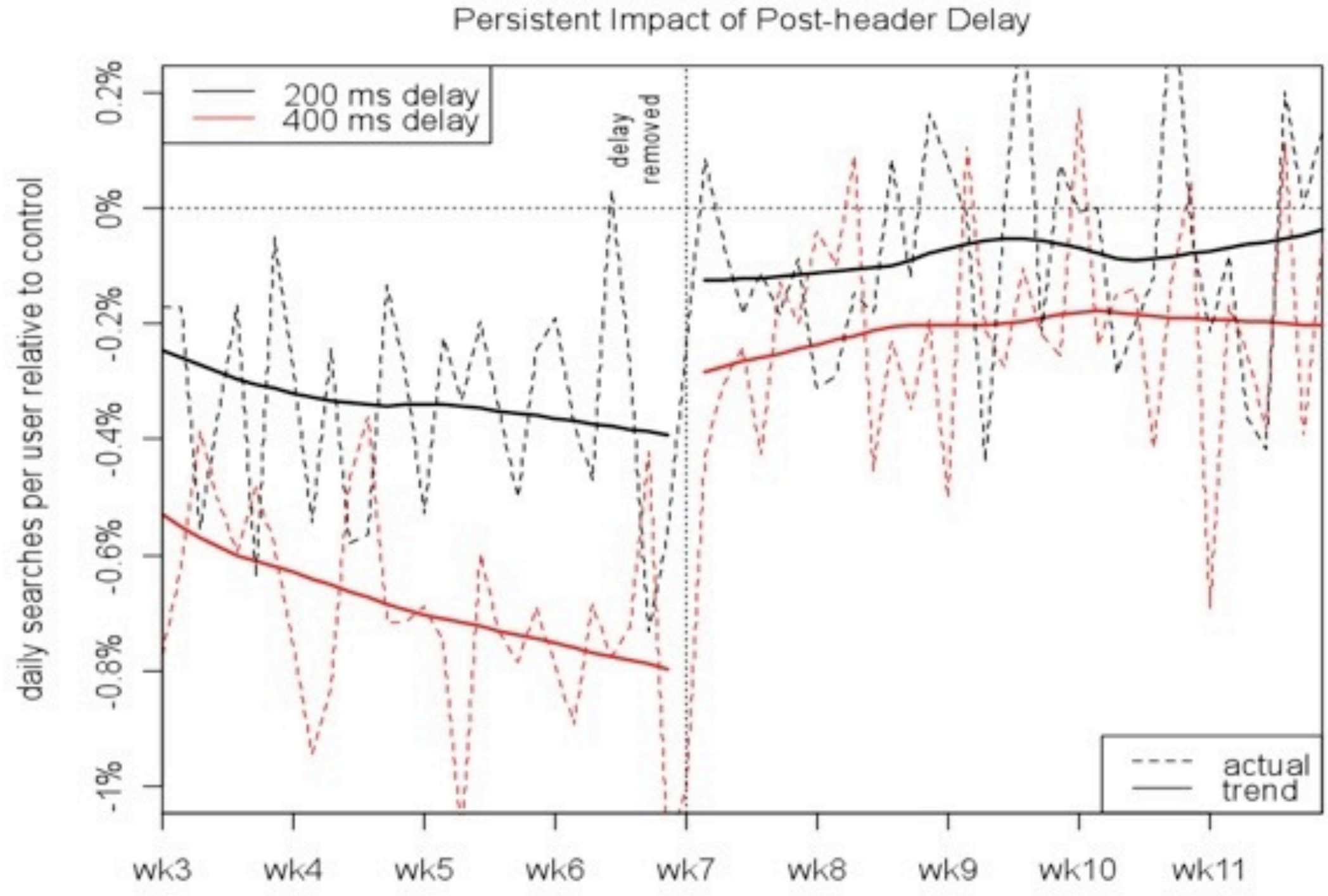
Why speed things up?

- Slow page loads cause user attrition… permanent user attrition.
- Users choose sites that are fast.

Bing:

	Distinct Queries/User	Query Refinement	Revenue/User	Any Clicks	Satisfaction	Time to Click (increase in ms)
50ms	-	-	-	-	-	-
200ms	-	-	-	-0.3%	-0.4%	500
500ms	-	-0.6%	-1.2%	-1.0%	-0.9%	1200
1000ms	-0.7%	-0.9%	-2.8%	-1.9%	-1.6%	1900
2000ms	-1.8%	-2.1%	-4.3%	-4.4%	-3.8%	3100

Bing:


Persistent Impact of Post-header Delay
200 ms delay
400 ms delay
delay removed
daily searches per user relative to control
0.2%
0%
-0.2%
-0.4%
-0.6%
-0.8%
-1%
actual
trend
wk3
wk4
wk5
wk6
wk7
wk8
wk9
wk10
wk11

Google:

Type of Delay	Delay (ms)	Experiment Duration (weeks)	Impact on Average Daily Searches Per User
Pre-header	50	4	Not measurable
Pre-header	100	4	-0.20%
Post-header	200	6	-0.29%
Post-header	400	6	-0.59%
Post-ads	200	4	-0.30%

Measuring

- A quick tour of tech:
  - Firefox + Firebug + YSlow! or PageSpeed
    - http://getfirebug.com/
    - http://developer.yahoo.com/yslow/
    - http://code.google.com/speed/page-speed/
  - Safari
  - Chrome

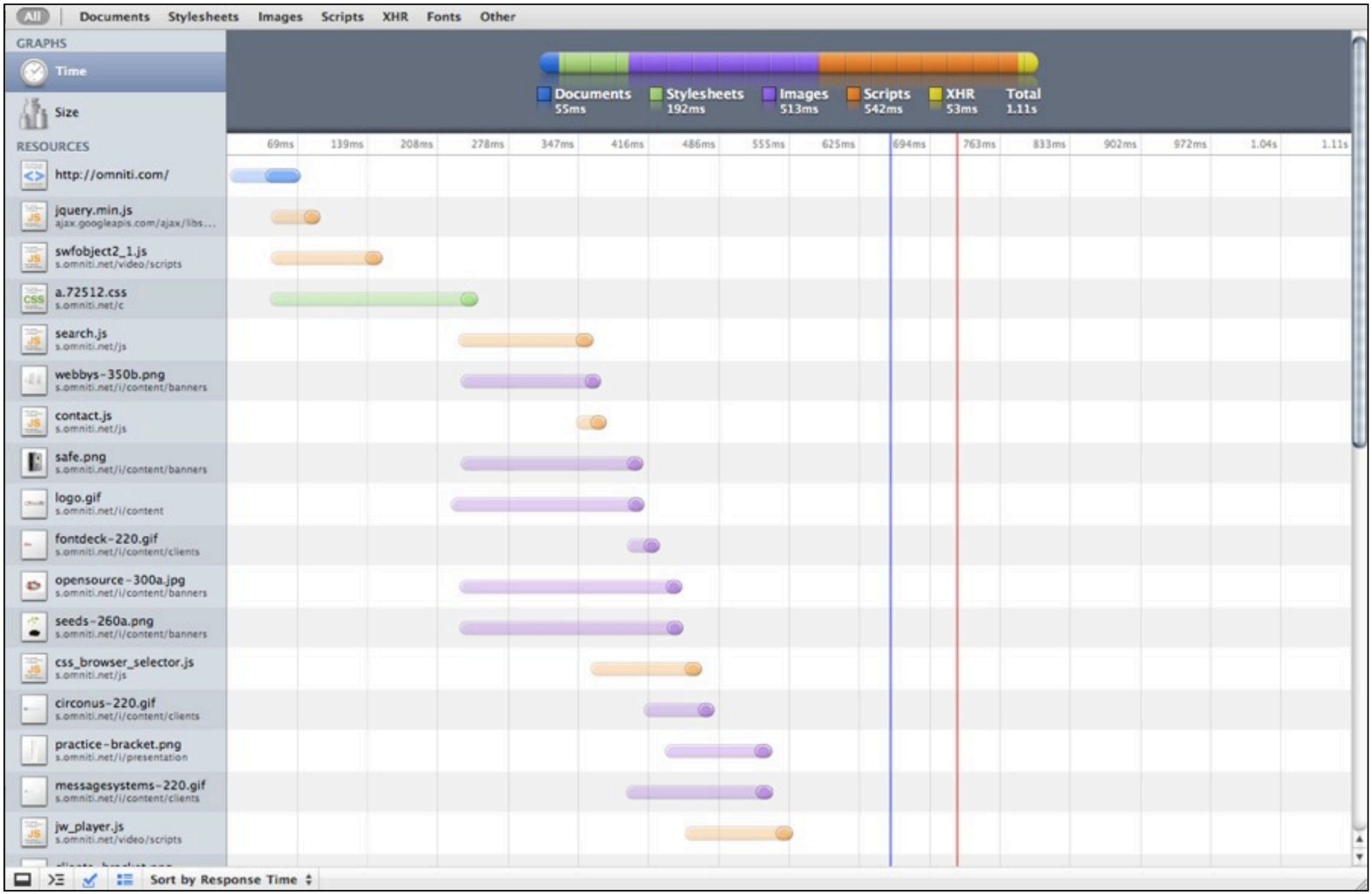
Measuring: Firefox w/ Firebug

Clear | Persist | All | HTML | CSS | JS | XHR | Images | Flash

URL	Status	Domain	Size	Timeline
Net panel activated. Any requests while the net panel is inactive are not shown.				
GET omniti.com	200 OK	omniti.com	5.8 KB	55ms
GET a.72512.css	200 OK	s.omniti.net	13.7 KB	172ms
GET jquery.min.js	200 OK	ajax.googleapis.com	24.1 KB	72ms
GET swfobject2_1.js	200 OK	s.omniti.net	3.8 KB	109ms
GET logo.gif	200 OK	s.omniti.net	2.8 KB	107ms
GET opensource-300a.jp	200 OK	s.omniti.net	18.3 KB	137ms
GET seeds-260a.png	200 OK	s.omniti.net	52.4 KB	251ms
GET webbys-350b.png	200 OK	s.omniti.net	70 KB	216ms
GET safe.png	200 OK	s.omniti.net	77.5 KB	200ms
GET messagesystems-22	200 OK	s.omniti.net	3.6 KB	149ms
GET fontdeck-220.gif	200 OK	s.omniti.net	1.2 KB	148ms
GET circonus-220.gif	200 OK	s.omniti.net	2.6 KB	156ms
GET natgeo-220.gif	200 OK	s.omniti.net	2.8 KB	159ms
GET etsy-220.gif	200 OK	s.omniti.net	3.5 KB	159ms
GET friendster-220.gif	200 OK	s.omniti.net	1.1 KB	163ms
GET search.js	200 OK	s.omniti.net	413 B	106ms
GET contact.js	200 OK	s.omniti.net	425 B	108ms
GET css_browser_selecto	200 OK	s.omniti.net	604 B	119ms
GET jw_player.js	200 OK	s.omniti.net	1.1 KB	119ms
GET omni-video-4.js	200 OK	s.omniti.net	1.7 KB	130ms
GET products-bracket.pn	200 OK	s.omniti.net	4 KB	21ms
GET practice-bracket.png	200 OK	s.omniti.net	2.5 KB	19ms
GET copy-bracket.png	200 OK	s.omniti.net	1 KB	18ms
GET clients-bracket.png	200 OK	s.omniti.net	1.9 KB	16ms
GET ico30-dquotes-l.gif	200 OK	s.omniti.net	208 B	13ms
GET ico30-dquotes-r.gif	200 OK	s.omniti.net	209 B	12ms
GET logo-bg-footer.gif	200 OK	s.omniti.net	1.4 KB	20ms
GET ico20-sprite-angle-	200 OK	s.omniti.net	392 B	20ms
GET ga.js	200 OK	google-analytics.com	10.4 KB	16ms
GET search	200 OK	omniti.com	731 B	1.17s
GET contact	200 OK	omniti.com	573 B	40ms
GET devopsday-playlist.x	200 OK	omniti.com	374 B	34ms
GET __utm.gif?utmwv=4.l	200 OK	google-analytics.com	35 B	14ms
GET ico21-search-close.(	200 OK	s.omniti.net	586 B	17ms
GET omniti-maryland.pn(	200 OK	s.omniti.net	28 KB	24ms
GET omniti-new-york.pn	200 OK	s.omniti.net	34.8 KB	34ms
GET ico21-search-insert.	200 OK	s.omniti.net	1 KB	19ms
37 requests			375.5 KB	2.49s (onload: 1.36s)

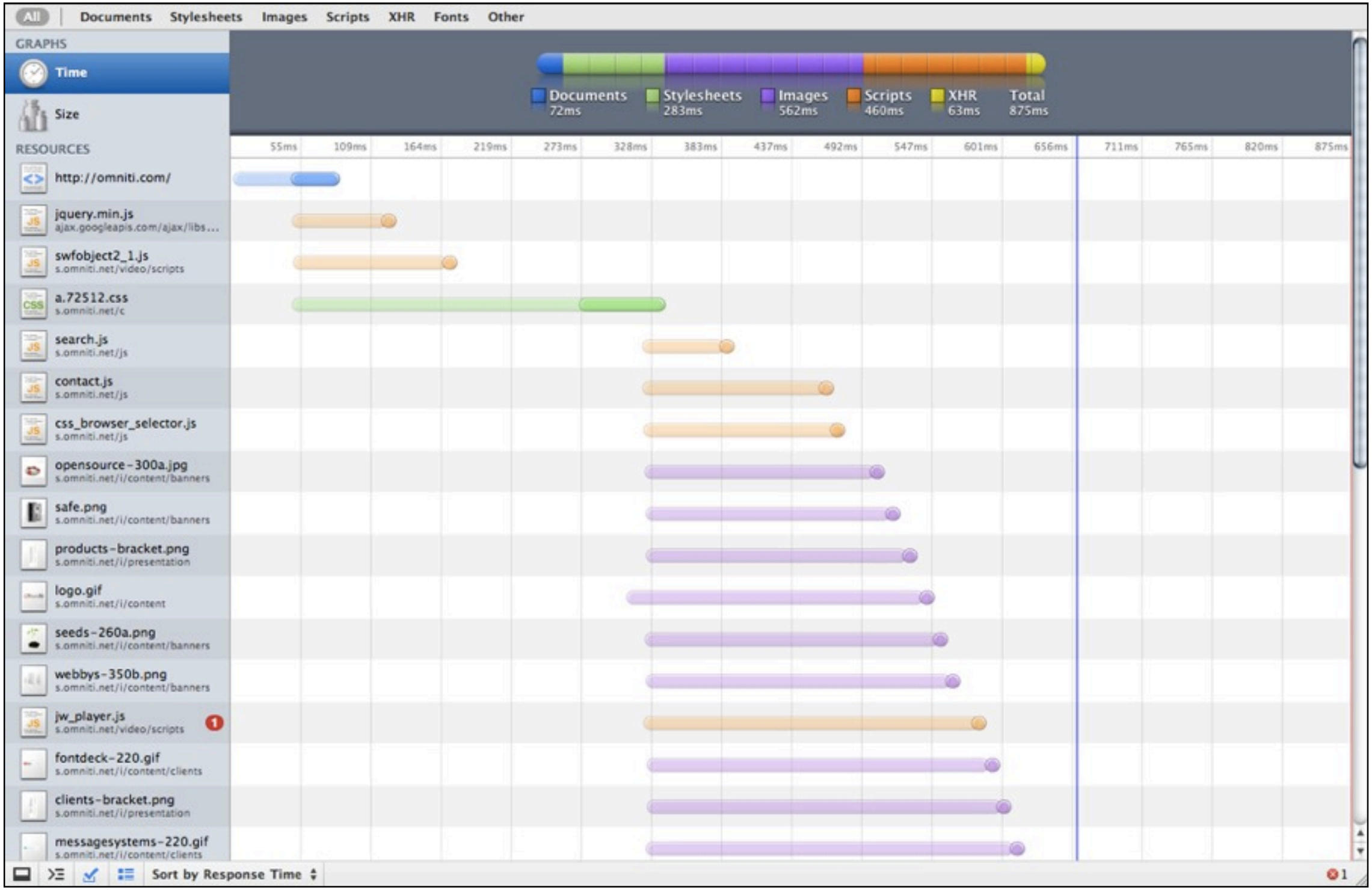
OmniTI

Measuring: Safari



OmniTI

Measuring: Chrome



OmniTI

Which technology?

- My personal preference for a browser is Chrome.
- My personal preference for a performance tool is Firefox.

- Firefox is more concise and easier to read on a single screen.

Waterfall ain't enough.

- The waterfall tells you on a layer 7 level what is going on…
- There is more:
  - use a packet capture and analysis tool:
  - Wireshark: http://www.wireshark.org/

Painfully Obvious

- Multiple assets are slow.
  - rm -f all.css && cat *.css > all.css
  - rm -f all.js && cat *.js > all.js
- One css include and one javascript include.

Don’t ask people to reload docs

- If the doc doesn’t change, or you control change well:
  - set infinite expire times and
    change the URL when the content is altered.
  - turn off ETags

Don’t ask people to reload docs

```
   # Images live here
<Directory "/www/sites/omniti.com/www/i">
    ExpiresActive On
    ExpiresDefault "access plus 1 year"
</Directory>

# CSS lives here
<Directory "/www/sites/omniti.com/www/c">
    ExpiresActive On
    ExpiresDefault "access plus 1 year"
</Directory>

# Javascript lives here
<Directory "/www/sites/omniti.com/www/s">
    ExpiresActive On
    ExpiresDefault "access plus 1 year”
</Directory>
```

Don’t ask people to reload docs

```
  # Javascript, CSS and images
<FilesMatch "\.(js|css|gif|png|jpe?g)$">
  FileETag None
</FilesMatch>
```

Painfully Obvious

- What happens when I change my CSS or images or javascript?
  - New images get new URLs. period. cope.
  - CSS and JS is included from a template, include a revision.
    - instead of all.css, reference all.12345.css
    - instead of all.js, reference all.12345.js
  - Each time you change the css or js, just bump the number in the template.

```
RewriteRule   ^(/c/.+)\.[0-9]+(\.css)$   $1$2
RewriteRule   ^(/s/.+)\.[0-9]+(\.js)$    $1$2
```

Static content on a CDN

- If you don't use a commercial CDN, you can still gain a lot...
- If your site is "company.com",
  get a different domain: "company.net"
- use "s.company.net" as your CDN domain
- Why?
  - Cookie domains.

Cookies

- A quick note on cookies...
- cookies are awesome...

  the most *powerful* and *accurate* distributed system available to you as a web engineer.

- Do not abuse the cookie
  - they are not free
  - they are sent with every request in their domain
  - many apps try to send them back on every request

Static content on a CDN

- All sites should *always* prepare for CDNized static content:
- A function in any language to ‘cdnize’ a uri:
  - <img src=”<?php echo CDN(‘/i/left-menu-background.gif’) ?>”...
  - <img src=”<%= $r->CDN(‘/i/left-menu-background.gif’) ?>”...

- even if it leaves it on the same host:

  sub CDN { return @_[1]; }

- Later you could change it to:

  sub CDN { return ‘http://s.company.net’ . @_[1]; }

Example HTML head

```
   <!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="charset=utf-8" />
<title>OmniTI</title>
<link type="text/css" rel="stylesheet"
      href="/c/all.css" />
<script type="text/javascript"
        src="/s/all.js">
</script>
</head>
```

A better HTML head

```
   <!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="charset=utf-8" />
<title>OmniTI</title>
<link type="text/css" rel="stylesheet"
      href="<?php echo $this->CDN("c/".$this->css_file) ?>" />
<script type="text/javascript"
        src="<?php echo $this->CDN("s/".$this->js_file) ?>">
</script>
</head>
```

A better HTML head: differences

```
   <!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="charset=utf-8" />
<title>OmniTI</title>
<link type="text/css" rel="stylesheet"
      href="<?php echo $this->CDN("c/".$this->css_file) ?>" />
<script type="text/javascript"
        src="<?php echo $this->CDN("s/".$this->js_file) ?>">
</script>
</head>
```

Domain sharding

- If you have lots of static assets all on one domain...
  - each asset is loaded one after another on a connection
  - browsers limit the number of concurrent connections by domain
    - IE 6 and 7 open only two connections per domain

- If you split you domains: s1.company.net and s2.company.net
  - you get twice the concurrency
  - and better performance

Compress Data

- Compress your payloads.
  - mod_deflate

Compression is easy

```
    # compress text-ish content

AddOutputFilterByType DEFLATE \
        text/html text/plain text/xml \
        application/javascript text/css
```

Image sprites

- Each image is a new HTTP request.
- Each request is more packets and round-trips.
- The further away your users, the more painful this is.
- Many images can be combined together (like a contact sheet)
- Then CSS tricks can be used to display only the bit you wanted.
- Each image on the contact sheet is called a “sprite”

Image sprites: cheating

```
javascript:(function() {
  spritemejs=document.createElement('SCRIPT');
  spritemejs.type='text/javascript';
  spritemejs.src='http://spriteme.org/spriteme.js';
  document.getElementsByTagName('head')[0]
          .appendChild(spritemejs);
})();
```

Make Javascript Asynchronous

- If you can (which you mostly can) make Javascript run after the page loads, then make Javascript run after the page loads.
- Javascript all loads serially and blocks page execution, display and interaction. (yes, Javascript is lame)
- It should (if it can) go immediately before the closing body tag.
- Put Javascript in the <body>, not the <head>

Javascript: simple

```
<html>
<head>
  <script type="text/javascript" src="/menu.js"></script>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```

Javascript: simple (in body)

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript" src="/menu.js"></script>
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```

Javascript: simple (defer)

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script defer type="text/javascript" src="/menu.js"></script>
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```

Javascript: simple (defer)

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script defer type="text/javascript" src="/menu.js"></script>
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```


init() will fire before menu.js is loaded

Javascript: simple (defer via iframe)

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <iframe src="/menu.html" width=0 height=0 frameborder=0 ></iframe>
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```

Javascript: simple (defer via iframe)

menu.html has menu.js as inline javascript

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <iframe src="/menu.html" width=0 height=0 frameborder=0 ></iframe>
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```

Javascript: simple (defer via iframe)

menu.html has menu.js as inline javascript

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <iframe src="/menu.html" width=0 height=0 frameborder=0 ></iframe>
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```

init() will fire before menu.js is loaded

Javascript: simple (defer via iframe)

menu.html has menu.js as inline javascript

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <iframe src="/menu.html" width=0 height=0 frameBorder=0 ></iframe>
  <script type="text/javascript">
    function init() { StuffFromMenu(); }
    init();
  </script>
</body>
</html>
```

init() will fire before menu.js is loaded

Javascript: hardcoded callback

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
  <script type="text/javascript" src="/menu.js"></script>
</body>
</html>
```

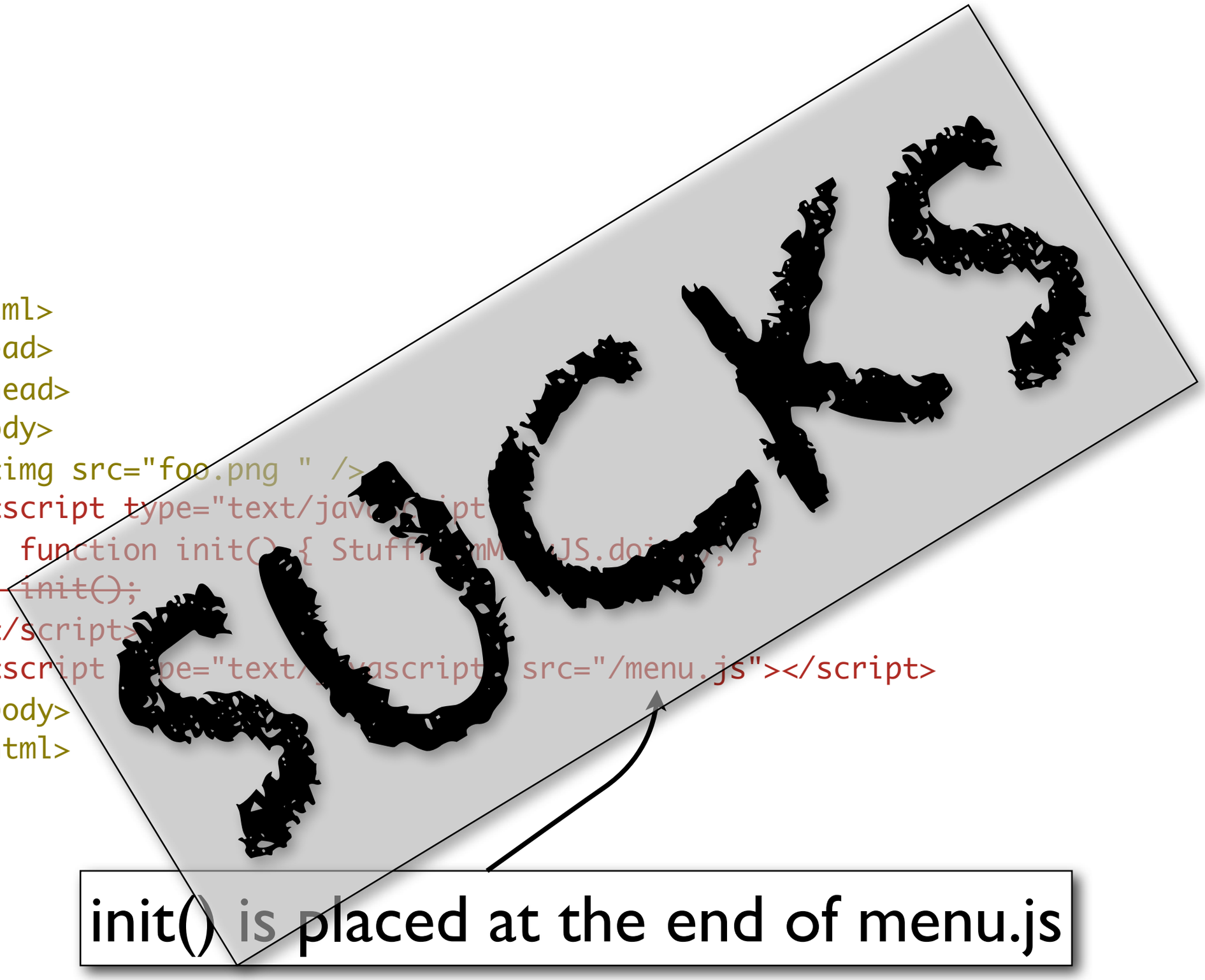
Javascript: hardcoded callback

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
  <script type="text/javascript" src="/menu.js"></script>
</body>
</html>
```


init() is placed at the end of menu.js

Javascript: hardcoded callback

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    function init() { StuffFromMyJS.doIt(); }
    init();
  </script>
  <script type="text/javascript" src="/menu.js"></script>
</body>
</html>
```



Javascript: DOM append

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    var domscript = document.createElement('script');
    domscript.src = "/menu.js";
    document.getElementsByTagName('head')[0].appendChild(domscript);

    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```

Javascript: DOM append

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    var domscript = document.createElement('script');
    domscript.src = "/menu.js";
    document.getElementsByTagName('head')[0].appendChild(domscript);

    function init() { StuffFromMenuJS.doit(); }
    init();
  </script>
</body>
</html>
```

init() will fire before menu.js is loaded

Javascript: via onload

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    var domscript = document.createElement('script');
    domscript.src = "/menu.js";
    document.getElementsByTagName('head')[0].appendChild(domscript);

    function init() { StuffFromMenuJS.doit(); }

    if ( window.addEventListener )
        window.addEventListener("load", init, false);
    else if ( window.attachEvent )
        window.attachEvent("onload", init);
  </script>
</body>
</html>
```

Javascript: via onload

init() will fire on page load
which can be long after the script load

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    var domscript = document.createElement('script');
    domscript.src = "/menu.js";
    document.getElementsByTagName('head')[0].appendChild(domscript);

    function init() { StuffFromMenuJS.doit(); }

    if ( window.addEventListener )
        window.addEventListener("load", init, false);
    else if ( window.attachEvent )
        window.attachEvent("onload", init);
  </script>
</body>
</html>
```

Javascript: via Timer

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    var domscript = document.createElement('script');
    domscript.src = "/menu.js";
    document.getElementsByTagName('head')[0].appendChild(domscript);

    function init() { StuffFromMenuJS.doit(); }
    function initTimer() {
        if ( "undefined" === typeof(StuffFromMenuJS) )
           setTimeout(initTimer, 300);
        else init();
    }
    initTimer();
  </script>
</body>
</html>
```

Javascript: via Timer

init() will fire will fire within 300ms of script load

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    var domscript = document.createElement('script');
    domscript.src = "/menu.js";
    document.getElementsByTagName('head')[0].appendChild(domscript);

    function init() { StuffFromMenuJS.doit(); }
    function initTimer() {
        if ( "undefined" === typeof(StuffFromMenuJS) )
           setTimeout(initTimer, 300);
        else init();
    }
    initTimer();
  </script>
</body>
</html>
```

Javascript: via on (script) load

init() will fire on script load!!!

```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }

    var ds = document.createElement('script');
    ds.src = "/menu.js";
    ds.onloadDone = false;
    ds.onload = function()
      { if (!ds.onloadDone) { ds.onloadDone = true; init(); } };
    ds.onreadystatechange = function()
      { if (("loaded" === ds.readyState || "complete" === ds.readyState) &&
            !ds.onloadDone )
          { ds.onloadDone = true; init(); } };
    document.getElementsByTagName('head')[0].appendChild(ds);
  </script>
</body>
</html>
```

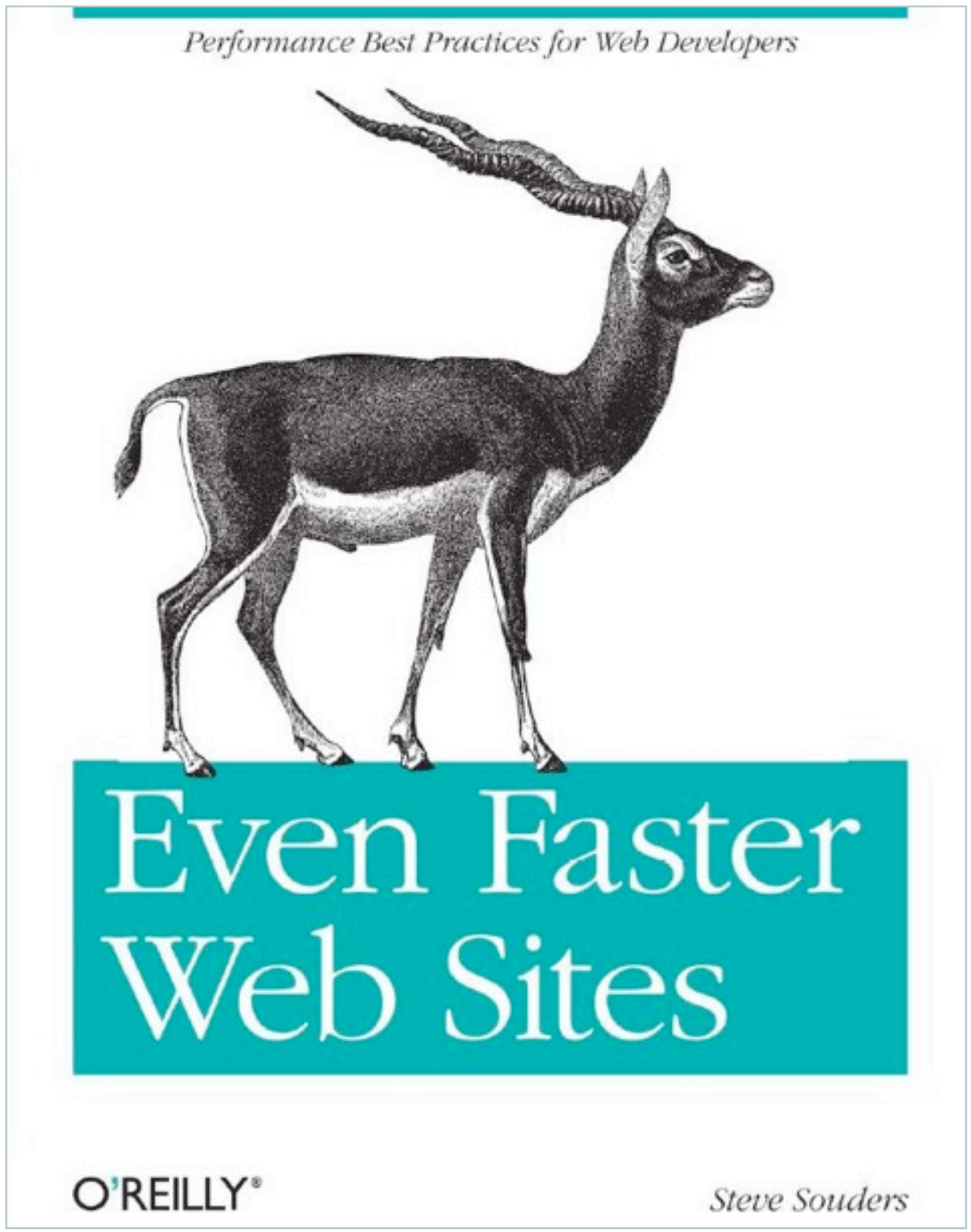
Javascript: XHR helper

```
<script type="text/javascript">

  function getXHRObject() {
    var xhrObj = false;
    try { xhrObj = new XMLHttpRequest(); }
    catch(e) {
      var aTypes = ["Msxml2.XMLHTTP.6.0", "Msxml2.XMLHTTP.3.0",
                    "Msxml2.XMLHTTP", "Microsoft.XMLHTTP"];
      var len = aTypes.length;
      for ( var i=0; i < len; i++ ) {
        try { xhrObj = new ActiveXObject(aTypes[i]); }
        catch(e) { continue; }
        break;
      }
    }
    finally { return xhrObj; }
  }

</script>
```

Javascript: via XHR Injection

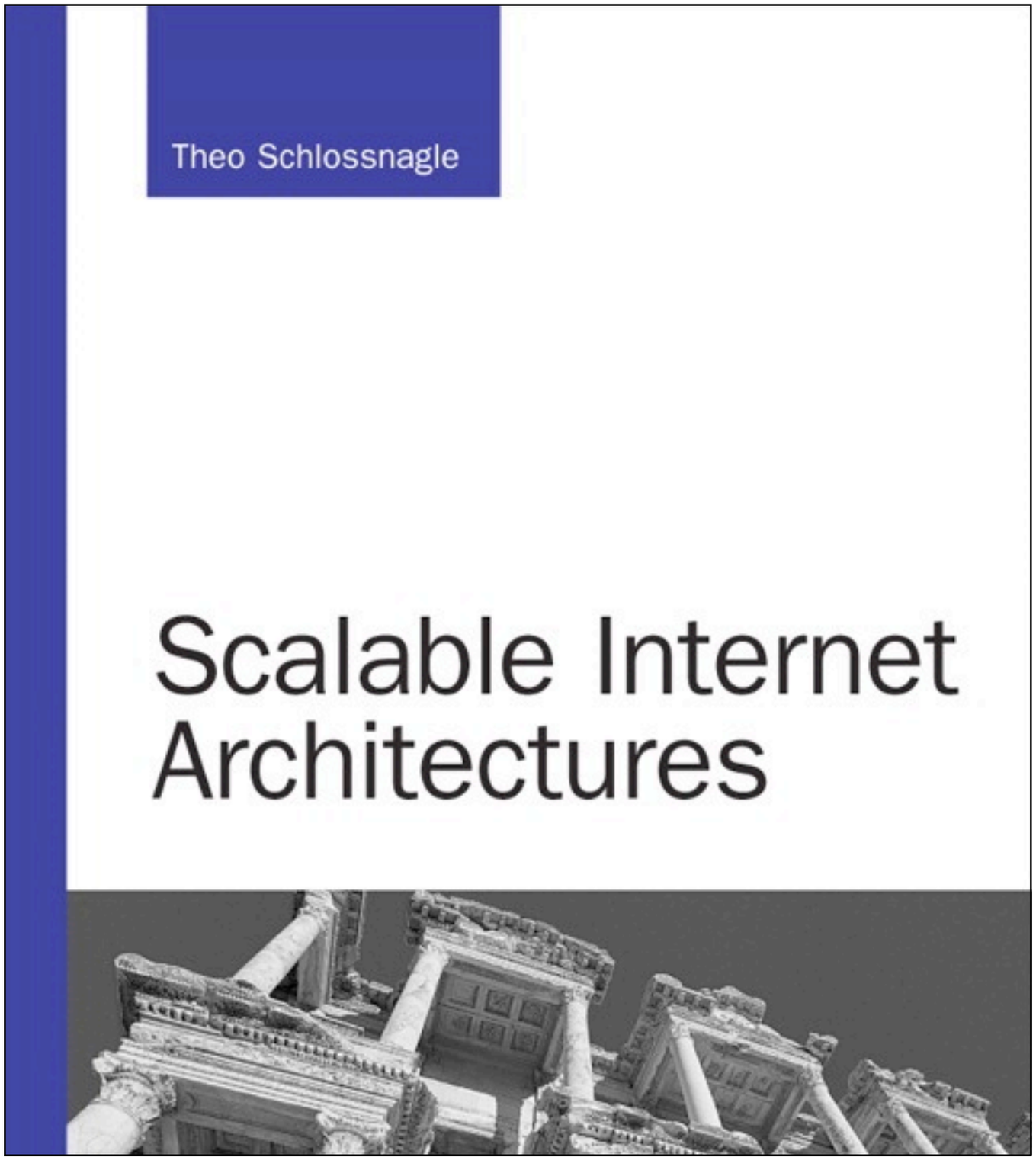
```
<html>
<head>
</head>
<body>
  <img src="foo.png " />
  <script type="text/javascript">
    function init() { StuffFromMenuJS.doit(); }
    xhrObj = getXHRObject();
    xhrObj.onreadystatechange = function() {
      if ( xhrObj.readyState != 4 ) return;
      var se = document.createElement('script');
      document.getElementsByTagName('head')[0].appendChild(se);
      se.text = xhrObj.responseText;
      init();
    };
    xhrObj.open('GET', '/menu.js', true);
    xhrObj.send('');
  </script>
</body>
</html>
```

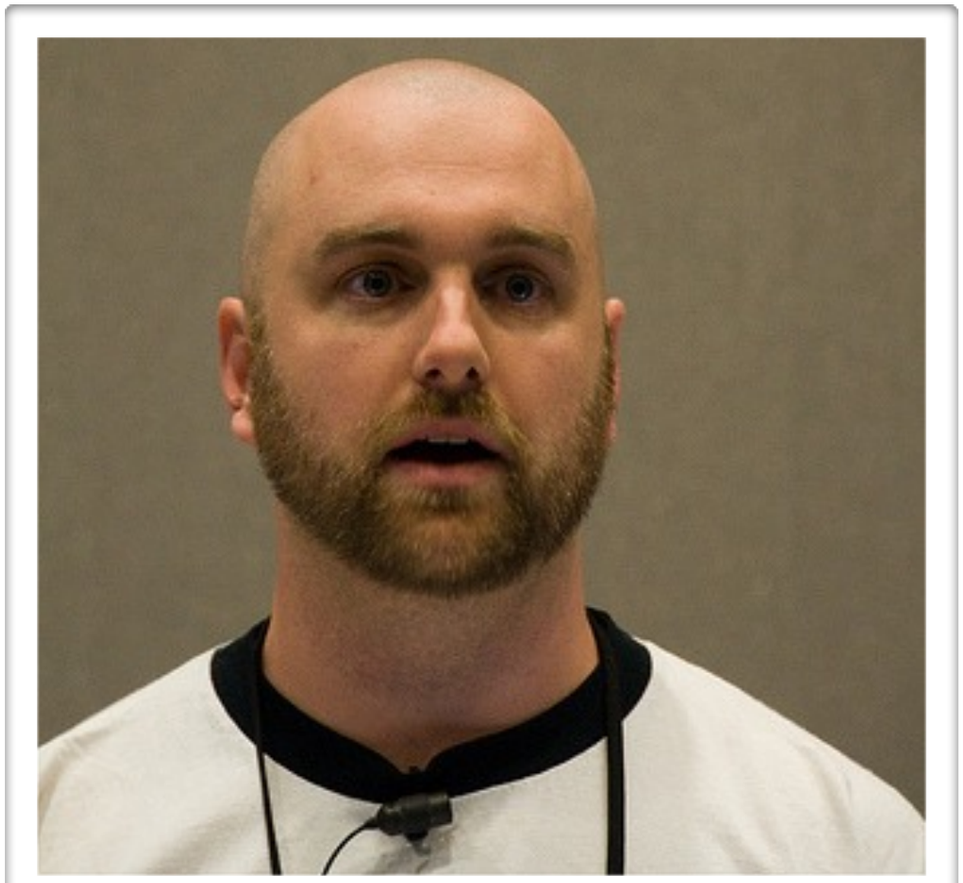
Web Performance Super Hero



OmniTI

Thanks!





OmniTI