web punch list:

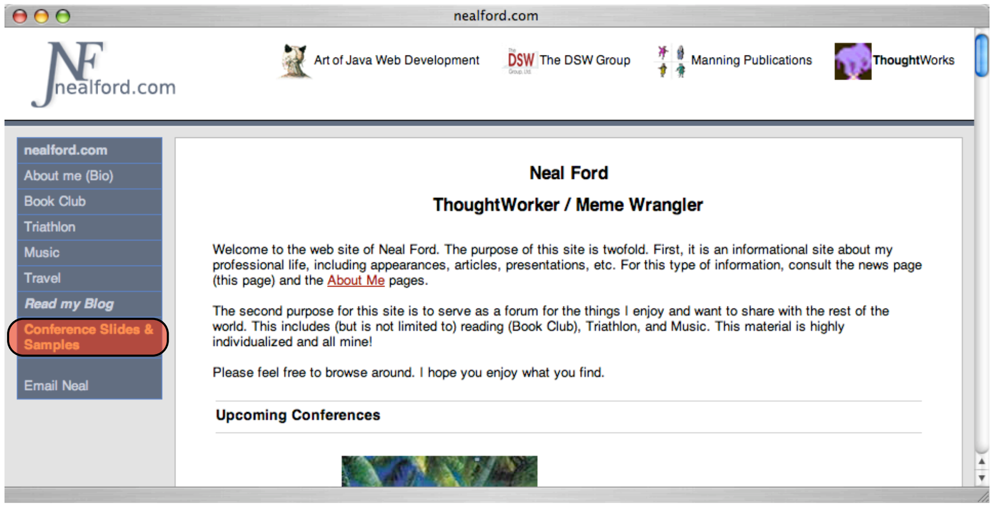
making your web applications suck less

NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

NF



Neal Ford

ThoughtWorker / Meme Wrangler

Welcome to the web site of Neal Ford. The purpose of this site is twofold. First, it is an informational site about my professional life, including appearances, articles, presentations, etc. For this type of information, consult the news page (this page) and the About Me pages.

The second purpose for this site is to serve as a forum for the things I enjoy and want to share with the rest of the world. This includes (but is not limited to) reading (Book Club), Triathlon, and Music. This material is highly individualized and all mine!

Please feel free to browse around. I hope you enjoy what you find.

Upcoming Conferences

☑ serve valid HTML (not XHTML or XHTML+XML)

HTML vs. XHTML

XHTML elements must be properly nested

XHTML elements must always be closed

XHTML elements must be in lowercase

XHTML documents must have one root element

html, xml, and html+xml

when XML and XHTML were first standardized, no browser supported them natively

W3C's HTML-compatible XHTML

allowed `<img src="funfun.jpg" />` to work

hack!

HTML parser viewed this as a "/" attribute and ignored it

whither XHTML?

what determines your document type?

MIME type:

- text/html => it’s HTML
- application/xhtml+xml or text/xml => XML

none of these will do it

Using an XHTML doctype declaration

Putting an XML declaration at the top

Using XHTML-specific syntax like self-closing tags

Validating it as XHTML

choosing...

your beautiful XHTML document is really just invalid HTML

what are your choices?

- **`application/xhtml+xml`**
- **`text/html`** to IE; **`application/xhtml+xml`** otherwise
- status quo

`application/xhtml+xml`

Internet Explorer won’t handle it

maybe not the best option!

`text/html` to IE; `application/xhtml+xml` otherwise

- your content has a chance of working on IE
- uses HTML compatible XML as originally intended
- documents parsed differently between browsers
- some browsers don’t support incremental rendering
- fundamental parser differences

status quo

generate XHTML but serve it has HTML

lose out on HTML validators (and Tidy)

subtle breakages if it ever is rendered as XHTML

what is it buying you?

serve valid HTML

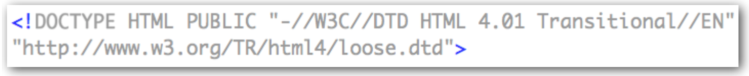
with the **`text/html`** mime type

still...it just works

the infrastructure is there

best way to ensure HTML

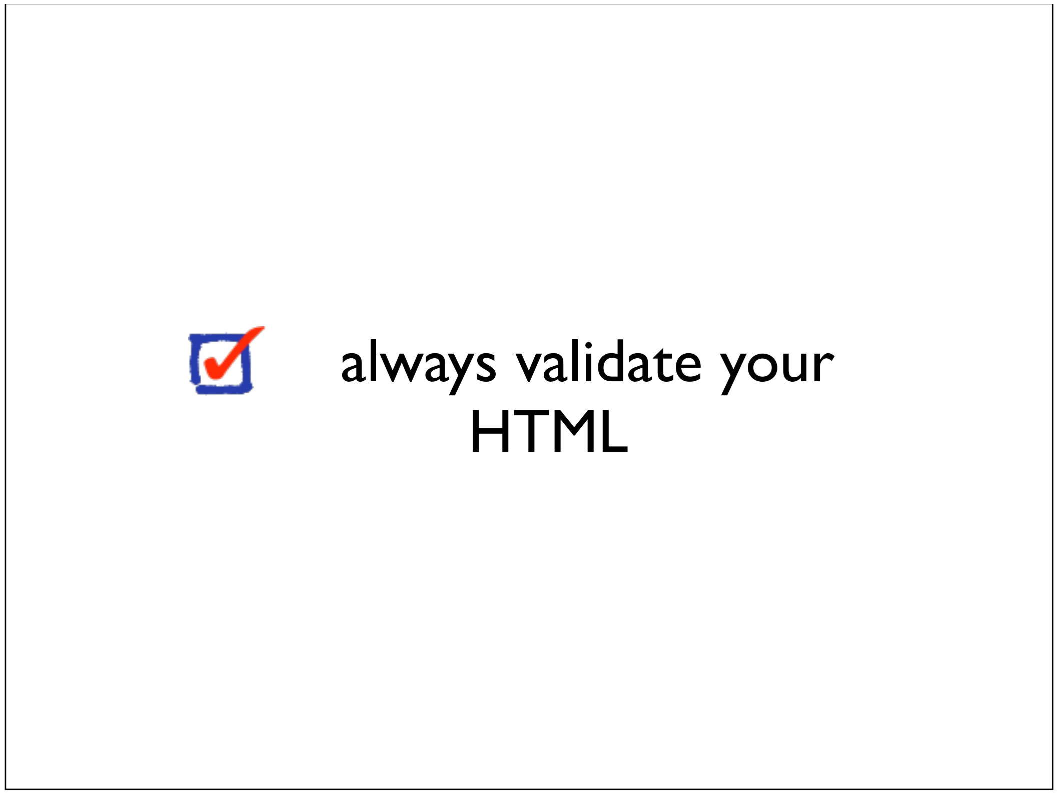
use HTML doctype that will trigger “standards” mode



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

serve content with **`text/html`** mime type (and name your files .html or .htm

validate content as HTML



☑ always validate your HTML

Always validate your HTML or markup. Its a great way to catch badly closed tags, block elements where they do not belong and the like. I don't use W3c much because I prefer Firefox's HTML validator extension running of TIDY.

validating HTML

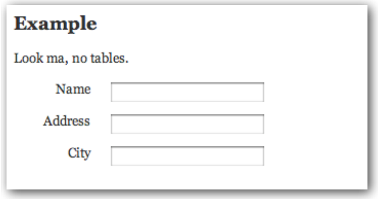
use the W3C’s validator

lots of Firefox add-ons

wire it into your build process

☑ use CSS for layout and design

forms with CSS



Example

Look ma, no tables.

Name	
Address	
City	

for older browsers

both id and name

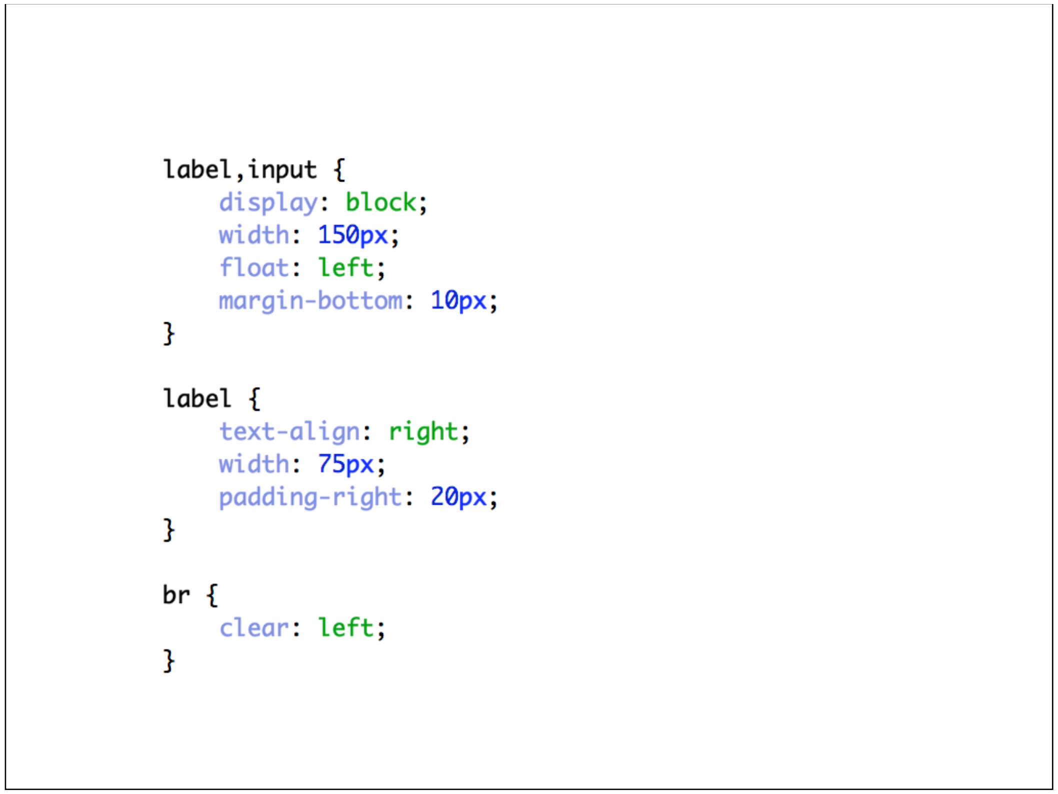
browsers only send form fields with name

clicking on a label only works if the form has id

```
<form>
  <label for="name">Name</label>
  <input id="name" name="name"><br>

  <label for="address">Address</label>
  <input id="address" name="address"><br>

  <label for="city">City</label>
  <input id="city" name="city"><br>
</form>
```



```
label,input {
    display: block;
    width: 150px;
    float: left;
    margin-bottom: 10px;
}

label {
    text-align: right;
    width: 75px;
    padding-right: 20px;
}

br {
    clear: left;
}
```

Finally, as the keystone, we give the `br` tag a `clear: left`, that is: any previously defined `float` is canceled. We have to insert a `clear` somewhere, or all labels and inputs would line up next to each other, which is not what we want.

I elected to declare the `clear` on the `br`, because the only other option (declaring it on the labels themselves) didn't work properly in Opera.

☑ do not use tables for layout - use them only for tabular data

tables!

using tables for layout confuses screen readers

biggest culprits of tag spam

technical debt

just because Google does it doesn't make it right!

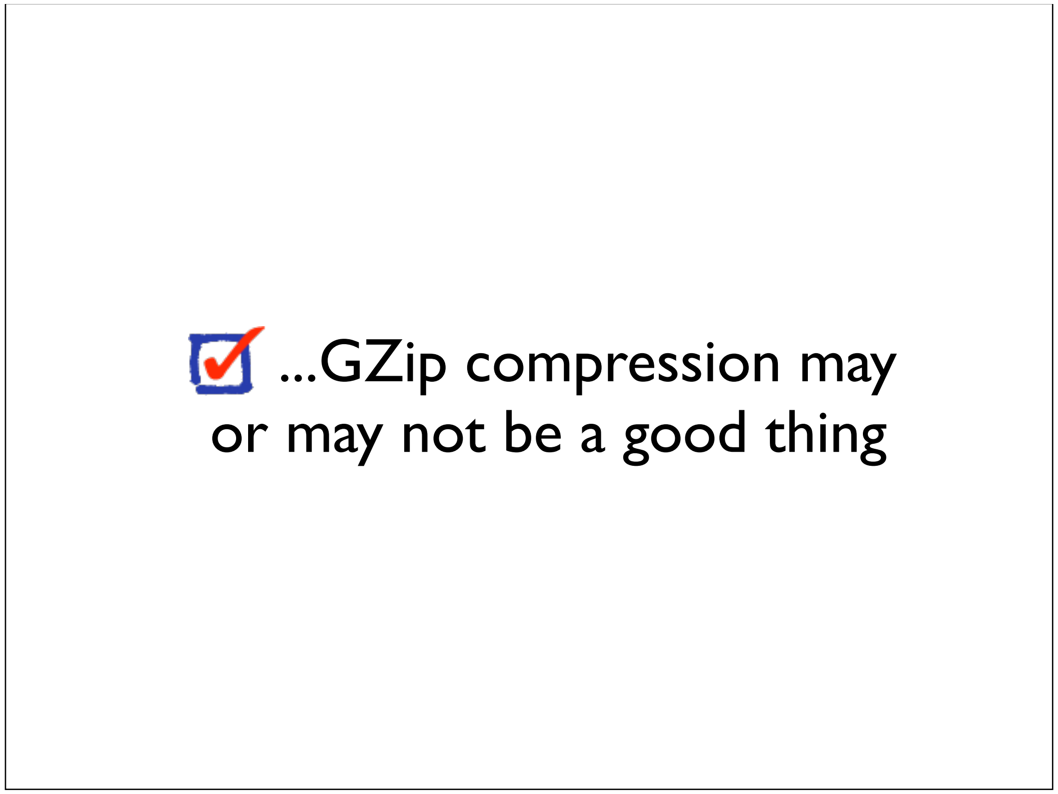
>How does that make anyone's life easier or more fulfilling?
>Google apps are using tables all over the place - because it's often
>better
>than fiddling with div positioning. Especially once you take your
>layout
>to the browser #1.

Harsh! Misusing tables disrupts the experience for visually impaired users.

Tables are also the biggest culprits of HTML tag spam. That will hit your bandwidth, your server load if you're using HTTP compression, your user's bandwidth, and their browser load to both decompress and render all that crap.

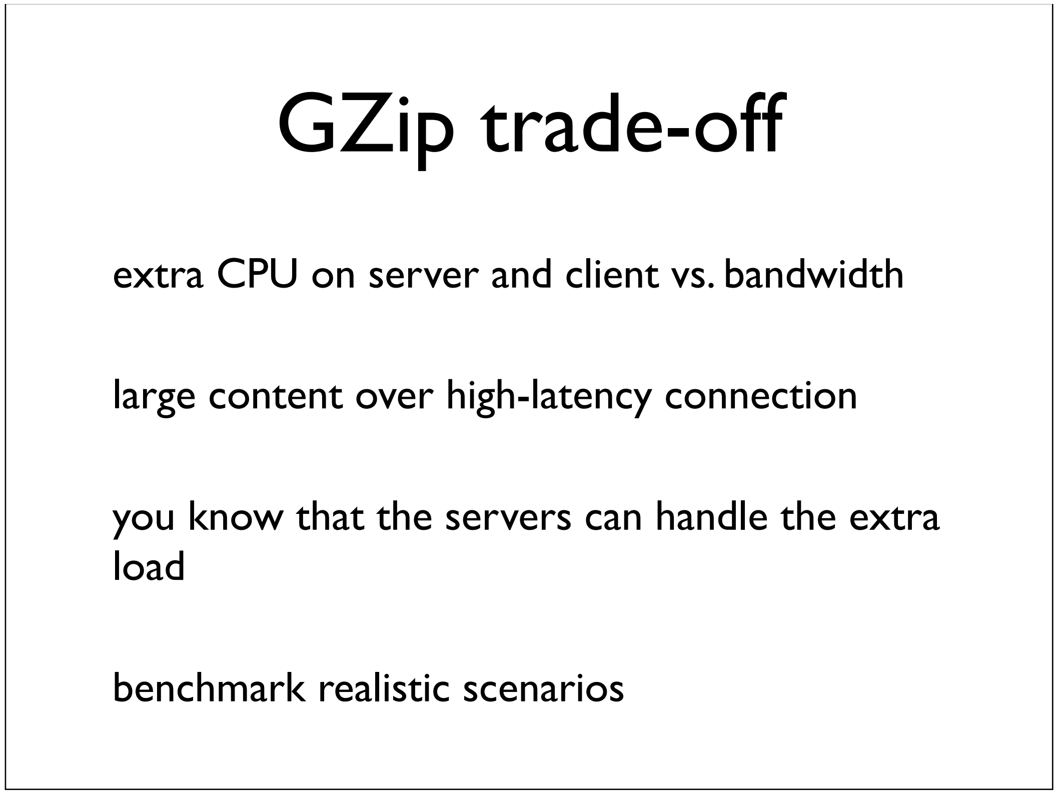
Tabular layout is better than fiddling with div positioning if you're lazy or in a rush. It's technical debt that you and your users will pay down later.

☑ use compression (like GZIP) on your HTTP server...



☑ ...GZip compression may or may not be a good thing

GZip compression may or may not be a good thing(its a good thing for areas with a slow internet, whereas its a bad thing for people with very fast internet), as the apparent load time may go noticeably up, browser takes time to uncompress response.....



GZip trade-off

extra CPU on server and client vs. bandwidth

large content over high-latency connection

you know that the servers can handle the extra load

benchmark realistic scenarios

> > 4. Use compression (like gzip) on your HTTP server
> As obvious as this may seem, I know at least one pretty big hosting
> shop that doesn't do it - the calculated that the cost of extra CPU
> power *for them* is more than the cost of extra bandwidth.

I'd say this is a last-resort kind of thing if you're not a hosting shop
and can control the content being served.

Good HTML, JS and CSS should cache nicely, and your load balancer or
reverse proxy can be configured to serve that with compression at a very
low cost. The trick here is working with all the HTTP headers that control
this sort of stuff - which is not trivial at all, especially since so many
proxies and HTTP agents get it wrong (Microsoft ISA Server and IE, I'm
looking at you).

For more info, see http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html

☑ use expires header to guide browser caching behavior

expires header

```
Expires: Thu, 01 Dec 1994 16:00:00 GMT
```

obviously for images...

...but consider using it for stylesheets & scripts

could reduce response time by 50% or more

much more information on the Yahoo Performace blog

☑ use asset tagging to let browsers know when a cached image has changed

caching servers

use different servers for cacheable information

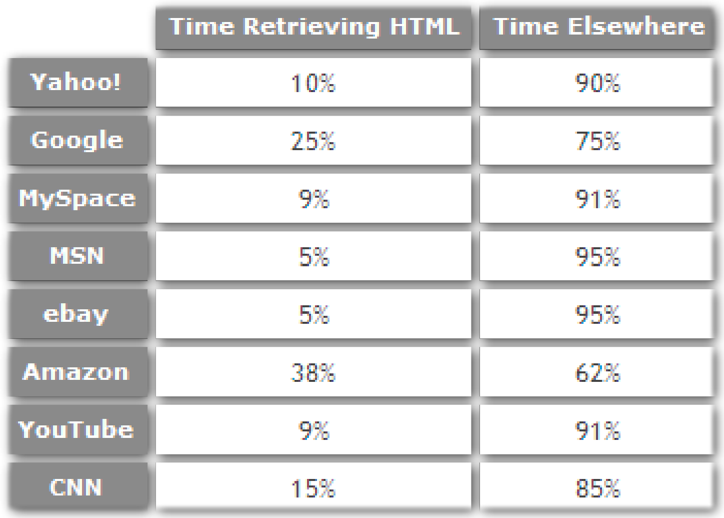
set far-future expires headers

if you need to replace a cached asset, change its name

forces the browsers to fetch and cache the new one

✓ use CSS sprites for commonly used images

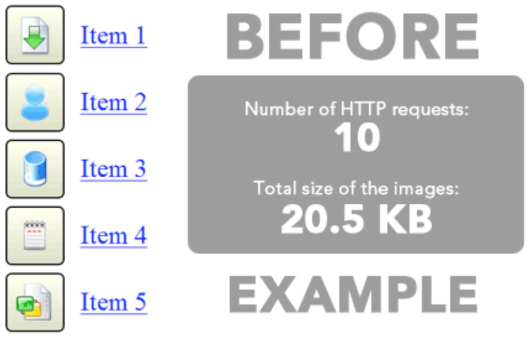
loading pages



	Time Retrieving HTML	Time Elsewhere
Yahoo!	10%	90%
Google	25%	75%
MySpace	9%	91%
MSN	5%	95%
ebay	5%	95%
Amazon	38%	62%
YouTube	9%	91%
CNN	15%	85%

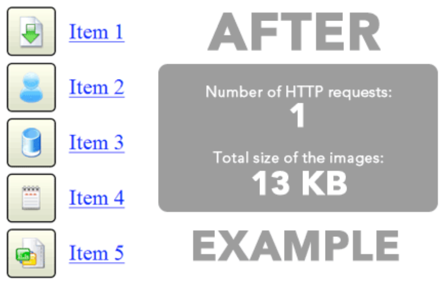
```
#nav li a {background:none no-repeat left center}
#nav li a.item1 {background-image:url('../img/image1.gif')}
#nav li a:hover.item1 {background-image:url('../img/image1_over.gif')}
#nav li a.item2 {background-image:url('../img/image2.gif')}
#nav li a:hover.item2 {background-image:url('../img/image2_over.gif')}
...
```

non-sprite CSS



```css
#nav li a {background-image:url('../img/image_nav.gif')}
#nav li a.item1 {background-position:0px 0px}
#nav li a:hover.item1 {background-position:0px -72px}
#nav li a.item2 {background-position:0px -143px;}
#nav li a:hover.item2 {background-position:0px -215px;}
...
```

sprite CSS



☑ consolidate commonly used javascript into a single file

☑ consolidate commonly used CSS into a single file

✓ load javascript as late as possible

JS as late as possible

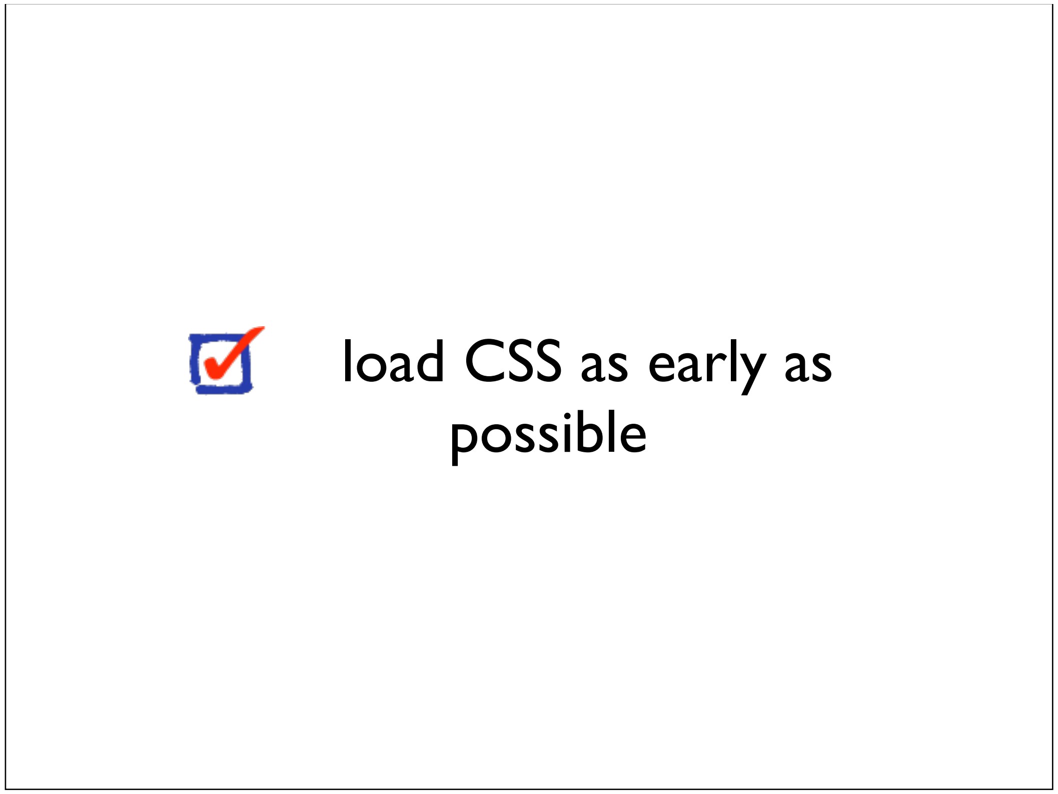
scripts block parallel downloads

while a script is loading, the browser won't start any other downloads (even from different hosts)

alternative: use the DEFER attribute

clue to the browser that you aren't going to do any **`document.write`** calls in your script

not yet supported in Firefox



☑ load CSS as early as possible

While researching performance at Yahoo!, we discovered that moving stylesheets to the document HEAD makes pages *apprear* to be loading faster. This is because putting stylesheets in the HEAD allows the page to render progressively.

☑ keep your CSS in separate stylesheets and link to them in the head of the page

externals

browser caches JavaScript and CSS

the only exception: home pages

home pages generally have only one page view per session

inline externals on the home page but...

...download external files at the bottom

☑ better yet - use JQuery. It's quite careful about not running any JavaScript until the document is really, really, really loaded.

☑ server-side caching of static resources pushes response time down

☑ accessibility, accessibility, accessibility

☑ clicking on the name of the company should take you to the home page

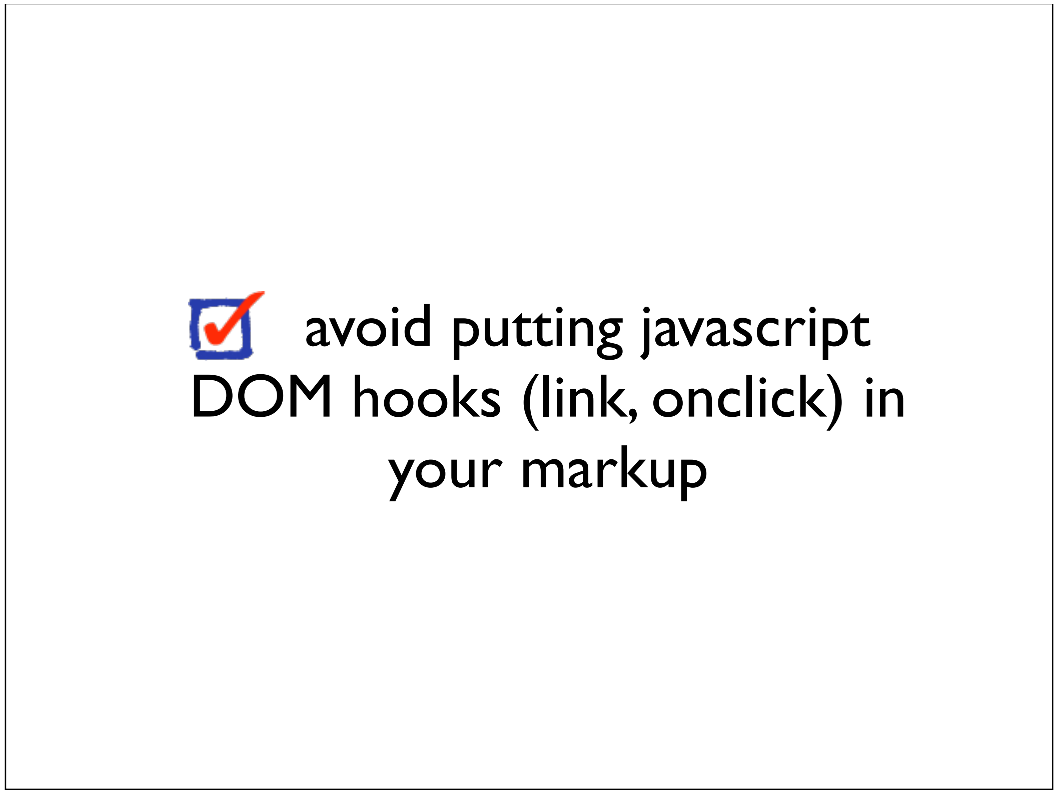
☑ degrade gracefully

☑ don’t break the back button / redirect after a post

☑ choose URLs that can last forever

(http://www.w3.org/Provider/Style/URI)

☑ stay stateless as long as possible



☑ avoid putting javascript DOM hooks (link, onclick) in your markup

Give the element and ID instead and hook an event hadler with JS. (YUI has really handy utilities for this stuff)

DOM hooks

give the element ID and name and hook the event with JavaScript

keeps behavior and view separate

YUI has handy utilities for this

☑ use a library of handy javascript functions

☑ prototype rapidly with Firefox, using tools like Web Developer toolbar, Firebug, etc...

☑ ...fix for IE6, IE 7 and IE8...

☑ ...test last on Opera and Safari

☑ make it run on IE ***and*** Firefox (at least)

☑ get your character set right

☑ always tidy your HTML

auto-tidying (in rails)

```ruby
if RAILS_ENV == 'test' and ENV['VALIDATE_HTML']
  require File.join(File.dirname(__FILE__),
      '/../app', '/../controllers', '/../application')
  require 'tidy'

  class ApplicationController
    after_filter :assert_valid_markup

    def status_code
      response.headers['Status'][0,3].to_i
    end

    def assert_valid_markup
      return unless RAILS_ENV == 'test'
      return unless(status_code == 200 &&
        response.headers['Content-Type'] =~ /text\/html/i &&
          response.body =~ /<html/i)
      assert_tidy
    end
```

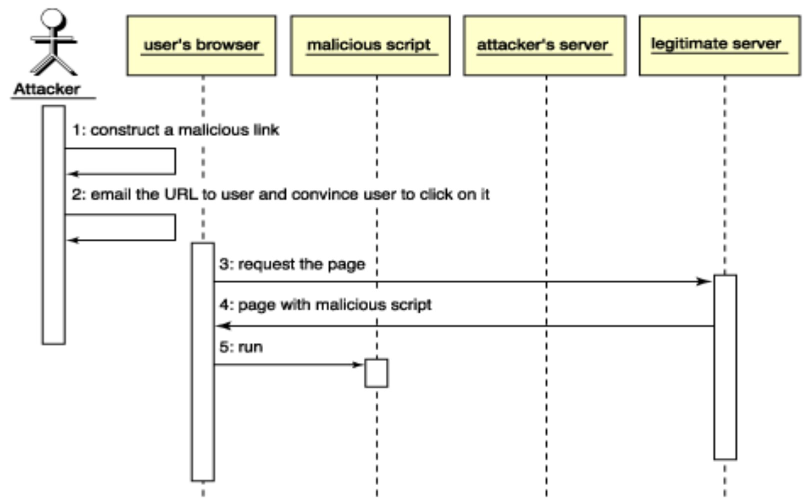
```ruby
def assert_tidy
  Tidy.path = '/usr/lib/tidylib.so'
  xml = Tidy.open(:show_warnings=>true) do |tidy|
    tidy.options.output_xml = true
    puts tidy.options.show_warnings
    xml = tidy.clean(response.body)
    puts tidy.errors
    puts tidy.diagnostics
    xml
  end
  puts xml

  raise "Tidy failed: #{$/} #{message}" unless tidy.errors.size.zero?
  tidy.release
end
```

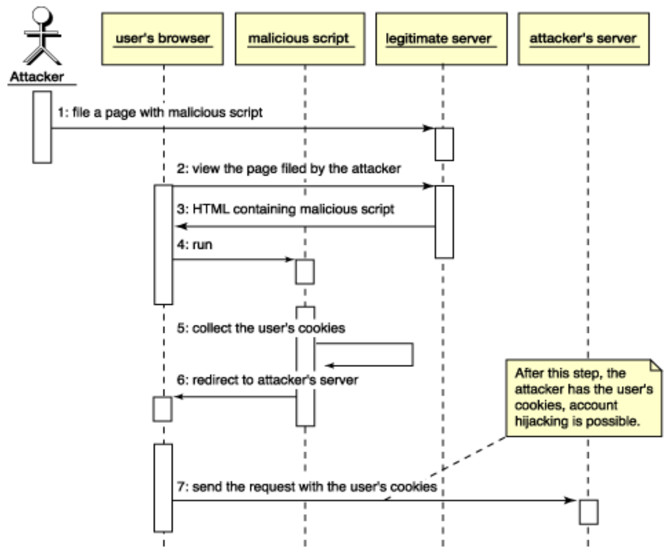
☑ code up the elements of your site like mashable components

- ✓ always keep XSS issues in the back of your mind

scripting via a malicious link

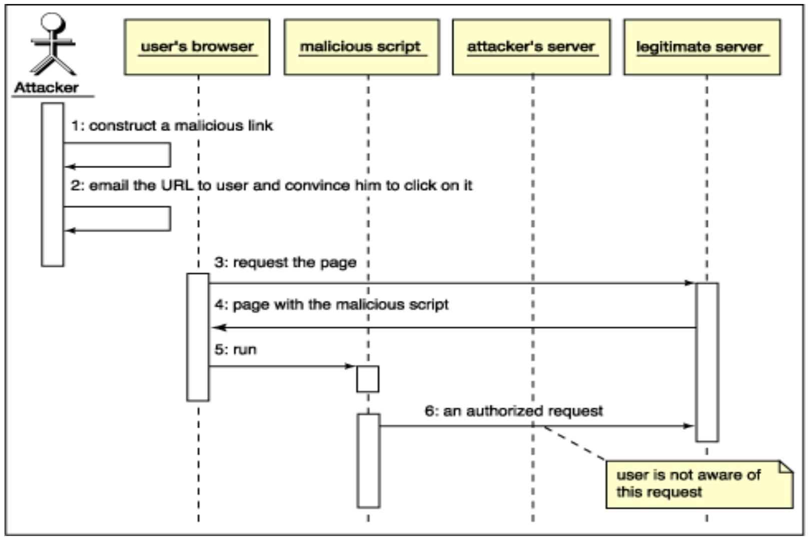

Attacker
user's browser
malicious script
attacker's server
legitimate server
1: construct a malicious link
2: email the URL to user and convince user to click on it
3: request the page
4: page with malicious script
5: run

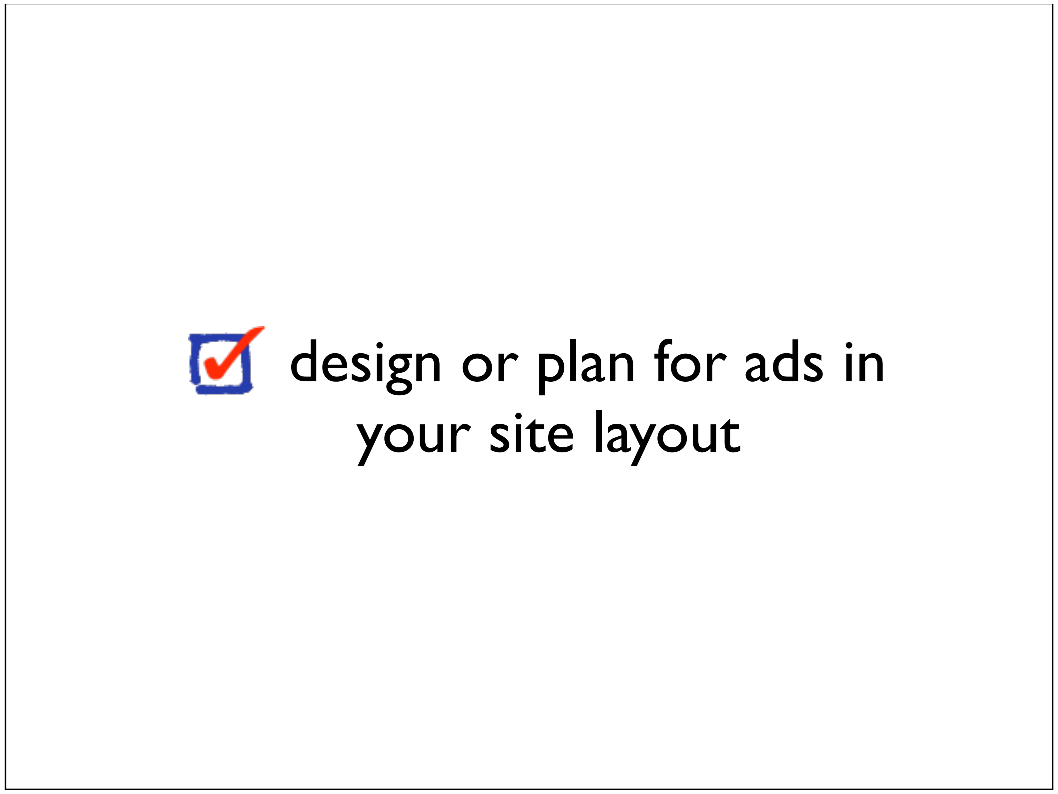
stealing cookies



Attacker
user's browser
malicious script
legitimate server
attacker's server
1: file a page with malicious script
2: view the page filed by the attacker
3: HTML containing malicious script
4: run
5: collect the user's cookies
6: redirect to attacker's server
After this step, the attacker has the user's cookies, account hijacking is possible.
7: send the request with the user's cookies

sending unauthorized request


Attacker
user's browser
malicious script
attacker's server
legitimate server
1: construct a malicious link
2: email the URL to user and convince him to click on it
3: request the page
4: page with the malicious script
5: run
6: an authorized request
user is not aware of this request



☑ design or plan for ads in your site layout

Its always a good idea to design or plan for Ads in your site layout and application code. Most commercial sites turn to ads at some point in time. Ads come in fixed sizes. Know them.

☑ plan in your application code for javascript-based analytics

☑ think about small things like sitemap.xml & robots.txt

☑ plan your site/code/templates with search engine optimization guidelines in the back of your mind (make full use of the H1-H6 headers, the title tag, alt text, title text, rel=no follow, etc.

☑ worry about localization

✓ when it comes to a choice between speed and perfection, choose speed first, perfection should follow

☑ when doing fancy stuff with javascript, watch the CPU cycles on older browsers.

☑ use WebDeveloper's linearize page option (under Miscellaneous) to figure out how a screen reader might see your page.

☑ remember that Javascript's string functions do not play nice with Unicode.

☑ always test your application at lower resolutions

☑ have someone else proofread your copy

☑ keep your URL's clean, semantic and book markable. Book markable URLs can be deep linked, resulting in better search engine optimization...

☑ ...make sure you put in redirects if these URLs change...

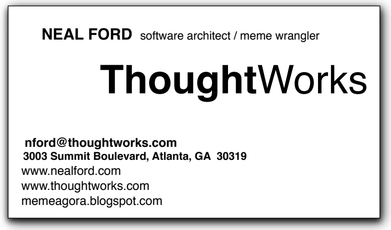
☑ ...404's are not nice

☑ always program your error messages with the user in mind (you never know when one may slip out into production)

☑ make sure your site doesn’t suck (or at least sucks less than the competition)

questions?

please fill out the session evaluations
slides & samples available at nealford.com



NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

resources

HTML vs. XHTML
http://www.w3schools.com/XHTML/xhtml_html.asp

CSS Sprites example
http://css-tricks.com/css-sprites-what-they-are-why-theyre-cool-and-how-to-use-them/

CSS Sprites: Imaging Slicing's Kiss of Death
http://www.alistapart.com/articles/sprites

Text

Text