

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

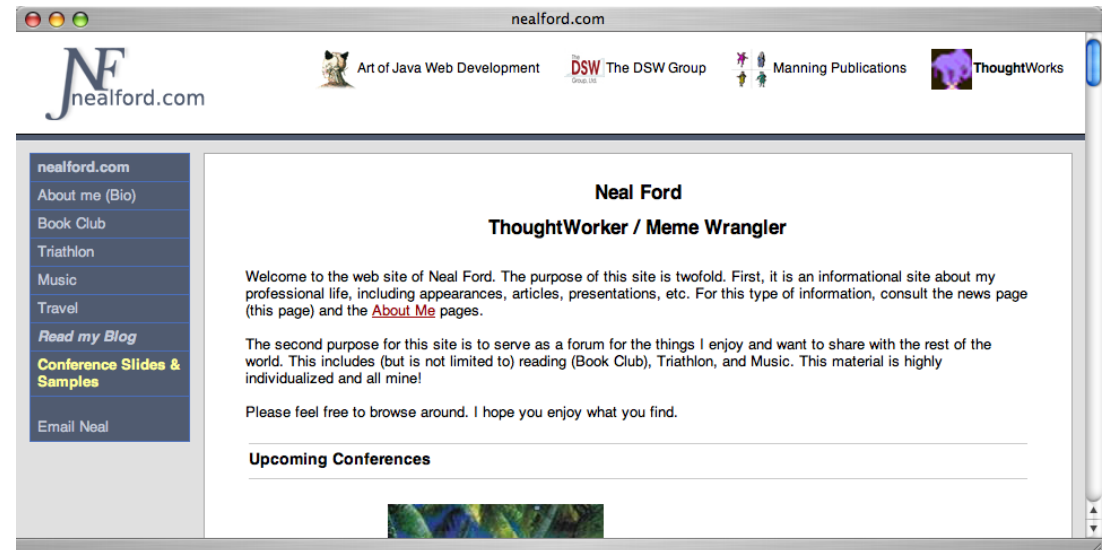
blog: memeagora.blogspot.com

twitter: [neal4d](https://twitter.com/neal4d)

housekeeping

ask questions anytime

download slides from
nealford.com →



download samples from `github.com/nealford`



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 `` 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 (even 8) won't handle it properly

maybe not the best option!

<http://blogs.msdn.com/ie/archive/2008/03/03/microsoft-s-interoperability-principles-and-ie8.aspx>

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 as 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



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">

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

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

</form>

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

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

```
br {  
    clear: left;  
}
```




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!



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

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

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



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



[Item 1](#)



[Item 2](#)



[Item 3](#)



[Item 4](#)



[Item 5](#)

BEFORE

Number of HTTP requests:

10

Total size of the images:

20.5 KB

EXAMPLE

```
#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



AFTER

Number of HTTP requests:

1

Total size of the images:

13 KB


EXAMPLE



consolidate commonly
used CSS into a single file



load CSS as early as
possible

 consolidate commonly
used javascript 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

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.



**accessibility, accessibility,
accessibility**

you'll eventually need it

all government sites in the US require section 508 compliance

most large corporations as well

why eliminate part of your audience?

freely available validators exist


<http://www.section508.info/>



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



little things

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



don't break the back
button / redirect after a post



choose URLs that can
last forever



stay stateless as long as
possible



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

“traditional” event hooks

```
referenceToElement.onclick = function () { alert('here'); };
```

problems:

adding additional functionality is cumbersome

what happens when 2 controls compete for the same event?

IE and Netscape handled collisions differently

DOM event listeners

```
refToElement.addEventListener('nameOfEvent',referenceToFunction,phase)
```

multiple events per element

phase can be either **false**(**bubble**,the default)
or **capture**

more flexibility

separates user interface from behavior

DOM events

```
<div id="thediv"><a href="/" id="thelink">test</a></div>
```

```
...
```

```
var oDiv = document.getElementById('thediv');
```

```
var oLink = document.getElementById('thelink');
```

```
oDiv.addEventListener('click',function (e) {
```

```
    alert('1. Div capture ran');
```

```
},true);
```

```
oDiv.addEventListener('click',function (e) {
```

```
    alert('3. Div bubble ran');
```

```
},false);
```

```
oLink.addEventListener('click',function (e) {
```

```
    alert('Link capture ran - browser does not follow the specification');
```

```
},true);
```

```
oLink.addEventListener('click',function (e) {
```

```
    alert('2. Link bubble ran (first listener)');
```

```
},false);
```

```
oLink.addEventListener('click',function (e) {
```

```
    alert('2. Link bubble ran (second listener)');
```

```
},false);
```




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)



always validate your
HTML

validating HTML

use the W3C's validator

lots of Firefox add-ons

wire it into your build process



always tidy your HTML

HTML Tidy

open source project to verify cleanliness of
html

reports incompatibilities between browsers

HTML Tidy (<http://sourceforge.net/projects/tidy>)

Java version (<http://sourceforge.net/projects/jtidy>)

Windows (<http://int64.sourceforge.net/tidy.html>)

auto-tidying (in rails)

```
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
end
```

```
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



future proof



design or plan for ads in
your site layout



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.



degrade gracefully



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



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...

routes

abstract actual locations from logical locations

preserves URLs forever

plays nice with tinyURL and other shorteners

build your own (ala Rails)

urlrewritefilter (<http://code.google.com/p/urlrewritefilter/>)



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

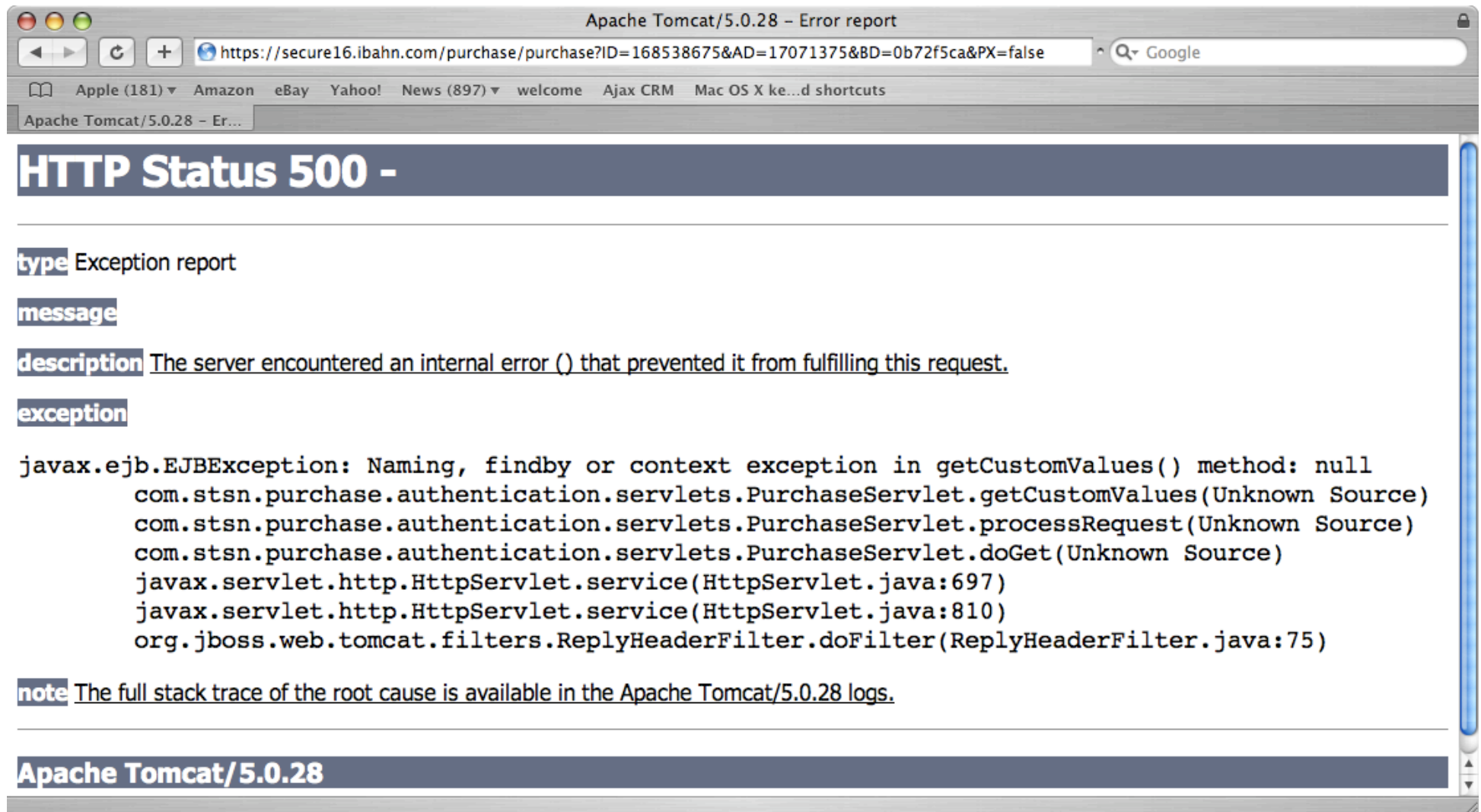


...404's are not nice



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

don't let this happen





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

?'S

please fill out the session evaluations
samples at github.com/nealford



This work is licensed under the Creative Commons
Attribution-Share Alike 3.0 License.

<http://creativecommons.org/licenses/by-sa/3.0/us/>

NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
blog: memeagora.blogspot.com
twitter: neal4d

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