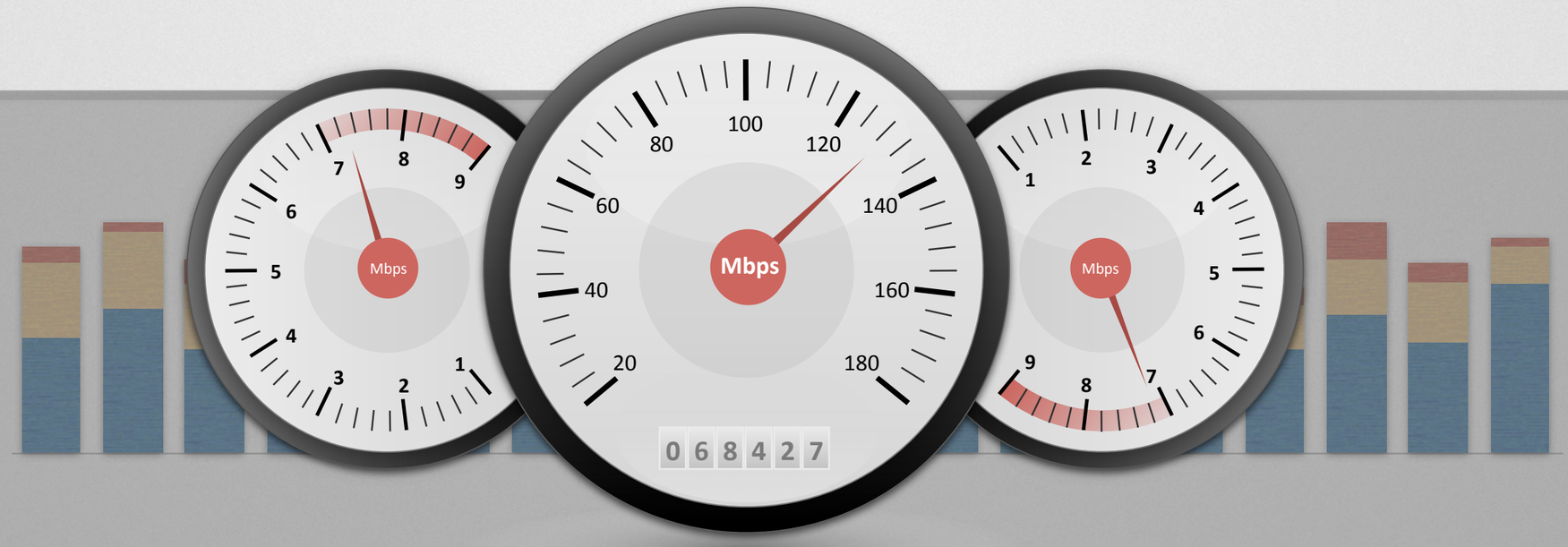
Dev vs. Prod

Optimizing Your Website Without Making Your Build Process Suck



Tuesday, June 26, 12

This workshop is about optimizing your website, and that “optimization” is two fold:

1. optimizing the performance of the production site
2. optimizing the build process itself

These two things seem t be at odds with one another, but we’ll see how they are more harmonious than you might expect.

First, the production side of things, which deals with server-to-client performance...

Tuesday, June 26, 12

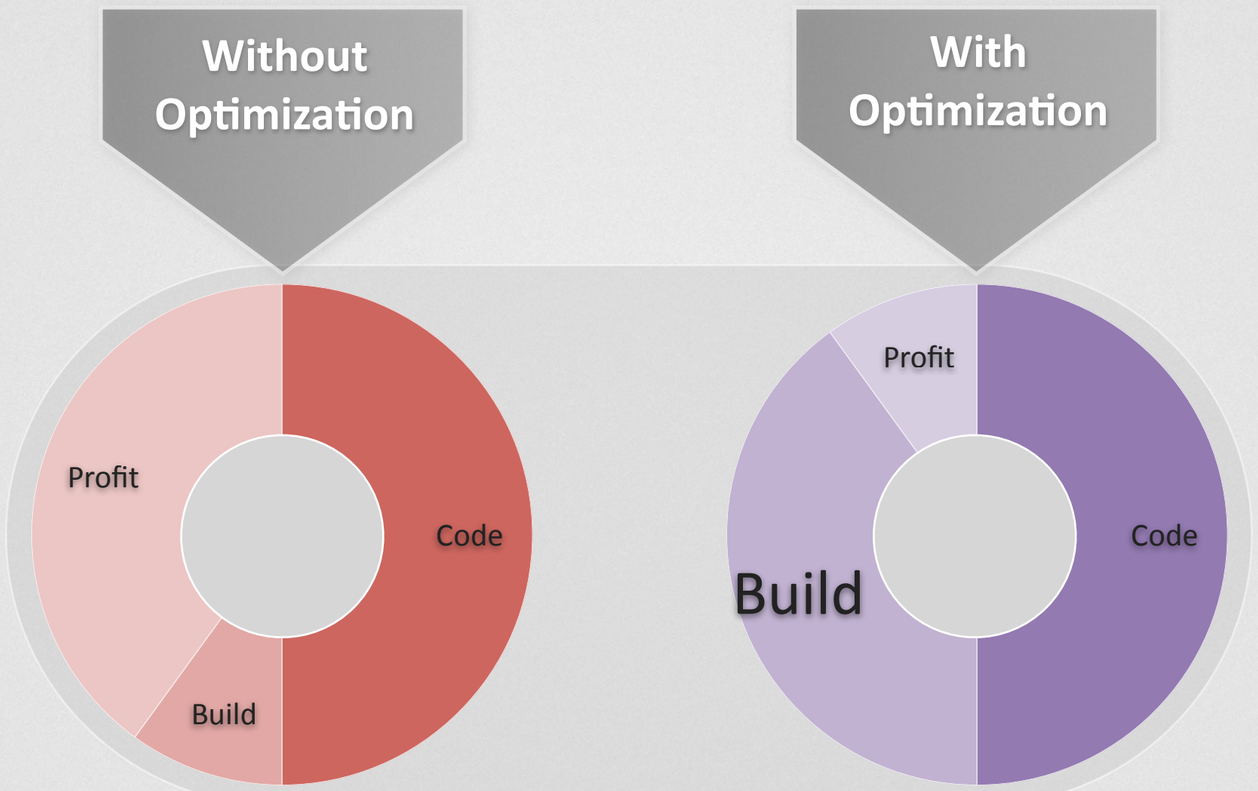
We'll go over a few big-ticket items, but for a ton more suggestions visit both:
- Google PageSpeed: https://developers.google.com/speed/docs/mod_pagespeed/filters
- Yahoo! YSlow!: http://developer.yahoo.com/performance/rules.html

this is a "production" driven endeavor
- not a lot of value in optimizing the development environment this way

if you've met engineers like my friends, this kind of optimization is always six months off, no mater what month you ask them.

=====
Organize CSS & JS
- Place CSS above JS to prevent blocking
- Avoid CSS @import
  - Blocks subsequent css files from loading
- Place JS as low (in the load order) as possible

Multiple domains
- increase the number of simultaneous downloads
- reduce overhead
  - no cookies on cross-domain requests
    - subdomains are cross-domain (so are ports)
- don't go nuts: 2-4 domains is plenty
  - each one adds additional DNS lookup time
- serve bootstrap scripts from main domain



Tuesday, June 26, 12

Dev: Keep the code-compile-test loop as short as possible
- new developers can get set up quickly
- no time for optimization, we'll do that later

Prod: Make the site as fast as possible

Benefits of a fast code-compile-test loop
- Lest waiting = more coding
- Gets new developers up and running quickly
- Encourages experimentation

Benefits of an optimized production environment
- The faster your site loads, the more users will stay for a *long* time
  - Jakob Nielsen (according to a research paper I have to pay to read - I didn't)
    - You have 10-20 seconds to make your value proposition before users leae
    - Once they're hooked, they'll stick around for much longer (2 min+)

This is the kind of thing that generates the dev vs. prod divergence problems:
- a fast development cycle is seen at-odds with production optimization
  - developers have no time for optimization, they're too busy creating whiz-bang features
- optimization generates files only a computer could love (unreadable)
  - introduces extra steps to the build process, slowing the code-compile-test loop

But, you can achieve both a fast development cycle and an optimized production environment without compromise

The Solution

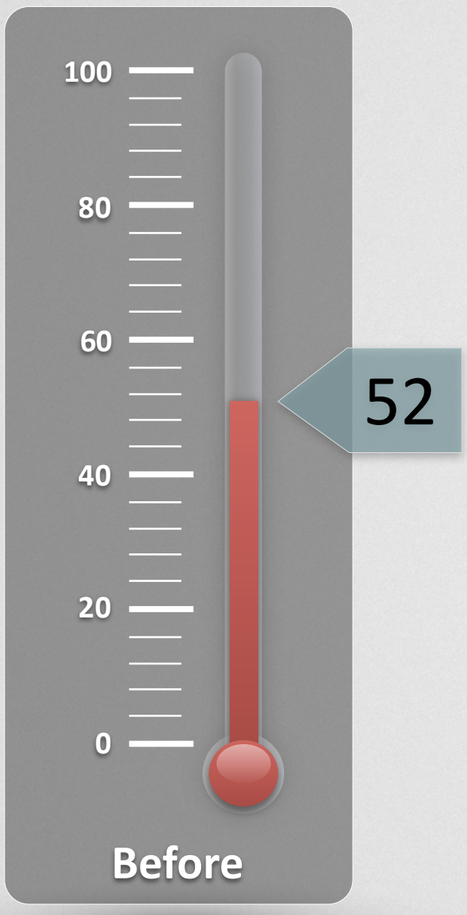
mod_pagespeed

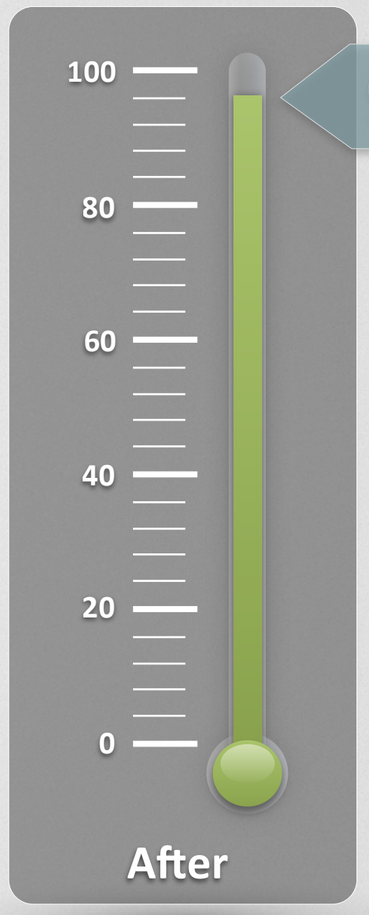
Tuesday, June 26, 12

(ice breaker)

mod_pagespeed
- easy to set up
- very flexible
- great for server-generated content
- handles multiple domains with shards (load balancing)

mod_pagespeed





Tuesday, June 26, 12

- Basic WordPress installation
  - Common Plugins
  - Added a single image
  - Added Google WebFont
- mod_pagespeed set up with generic options, nothing fancy to target this specific example
- took 5 minutes to get it up and running
- nearly 2x improvement, not bad

On that bombshell...

THANK YOU!

Let’s go get some free drinks

GitHub: @impressiver/velocity2012
Twitter: @impressiver
Email: ian@impressiver.com

NOT QUITE...

What?!

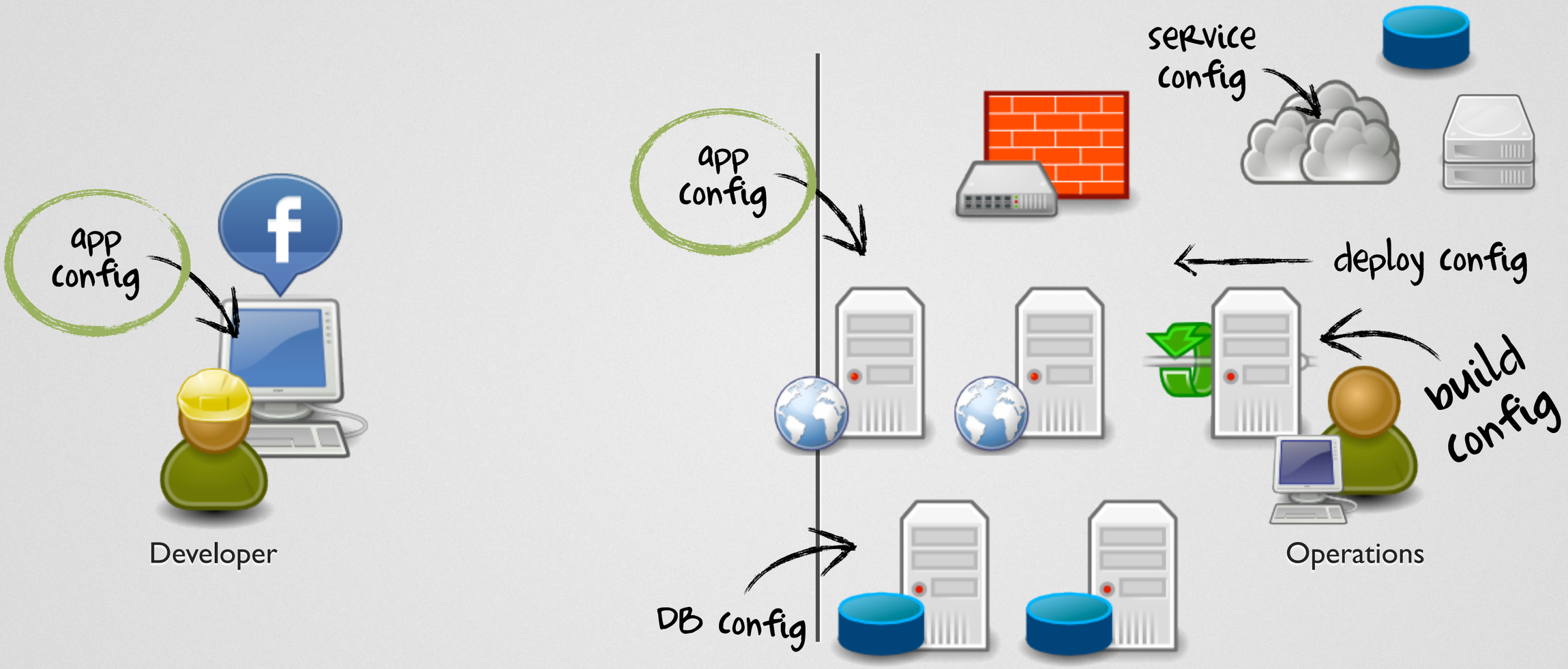
GitHub: @impressiver/velocity2012
Twitter: @impressiver
Email: ian@impressiver.com

Not always that simple
- varied topography
- need more control

mod_pagespeed
- requires Apache
- can't optimize client-heavy apps
- not good for dynamic scaling infrastructure
- no dynamic configuration
- you end up designing your application to fit the optimizer

So, we're going to need to get our hands dirty
- first: the easy stuff (server configuration)
- then: kind-of easy stuff (working with open-source libraries)
- then: set up dynamic configuration for multiple environments
- finally: what you can do with all of this

The Dev vs Prod Problem



Tuesday, June 26, 12

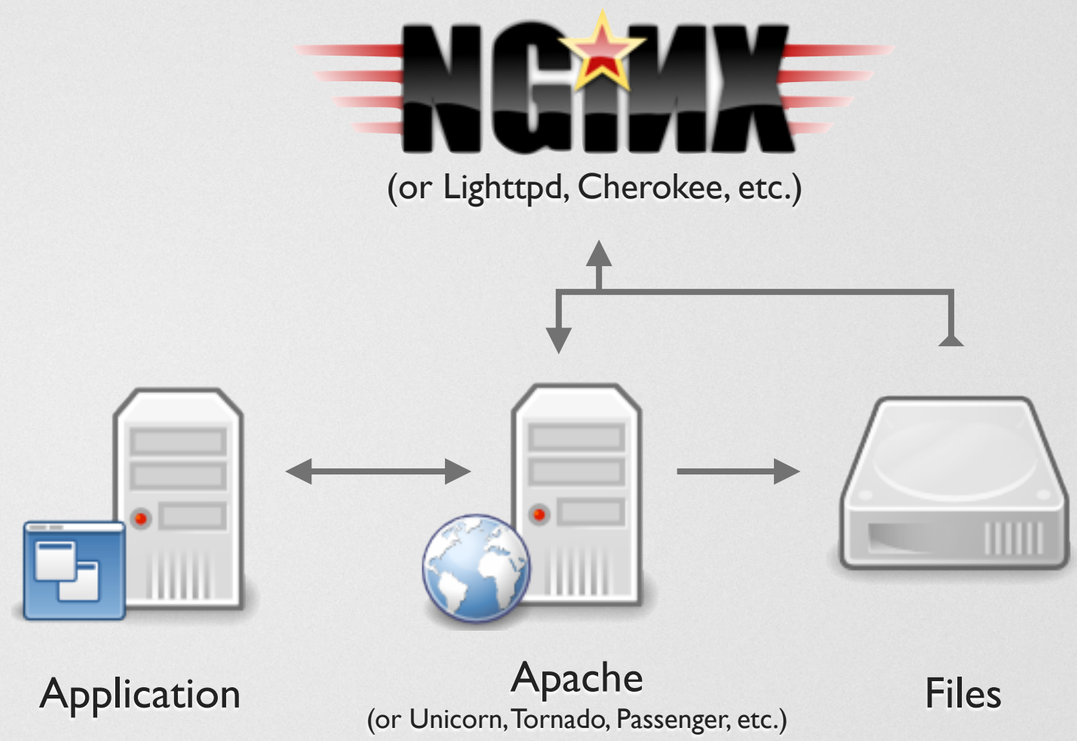
1. Build/CI server
2. Web servers
3. Databases
4. Cloud services (and web services)
5. Probably a firewall and load balancer...

- Developers not usually paying close attention to what Ops has set up
  - Dev changes config, forgets to tell ops, breaks when pushed to production
    ? How many people have had this happen at least once in their career? in the last month?
    - Fix: Parameterization (keep the configuration in the app)

- Developers have to build and push to a staging environment to test
  - Slow, not likely to happen very often
  - Fix: Build it into the app

Serving Static Resources

- Simple
- Fast
- Low overhead
- Gzip & Caching

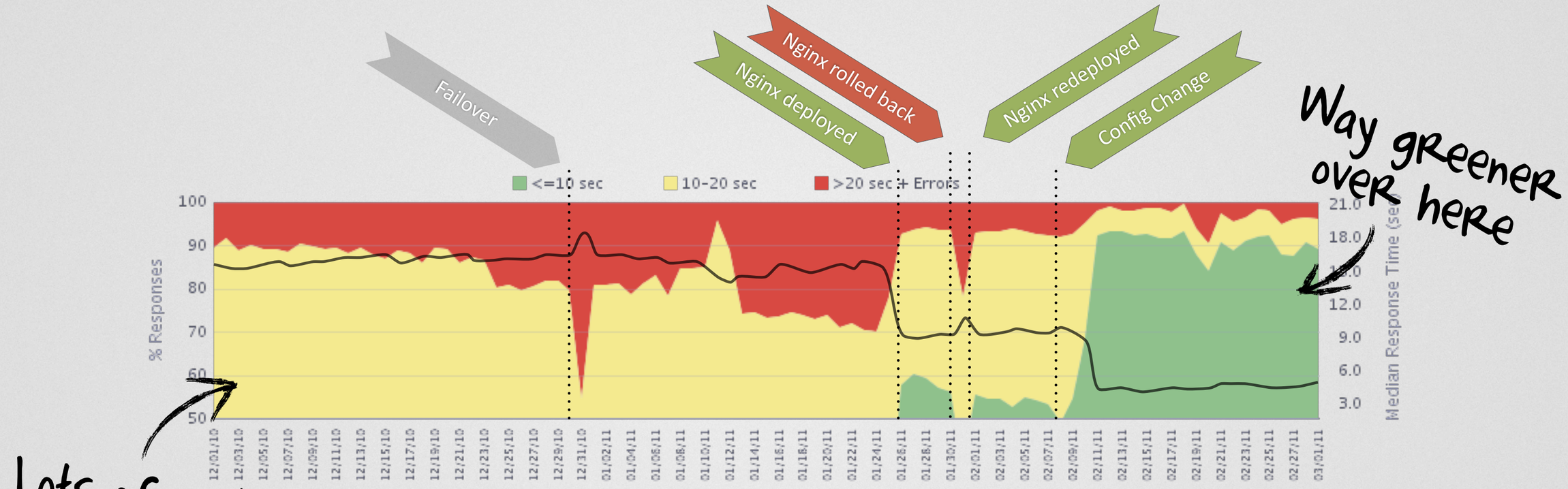


Tuesday, June 26, 12

- Works *with* Apache
- Lightweight
- Handles more connections
- Better memory management
- Proxy requests to Apache (and anything else)

- You could replace Nginx with anything you like (lighttpd, Cherokee, ...)
- Point is to keep it light weight, leave the big guns for the apps

The Nginx Effect



Webmetrics Production Monitoring

(full page load time using real browsers)

- Before:
  - Apache with mod_wsgi
  - Serving Python app + all files

- After
  - Nginx reverse-proxy
    - serving static files
    - proxy to Apache (same setup as before)

- If you can't see the labels, don't worry...
- Stats aren't important
  - your setup is different
  - test it out

Install Nginx

- apt-get nginx
- config to listen to port 80
- configure static servers (nginx)
- configure proxy to apache
- bonus: add default fallback
- config apache to listen to 127.0.0.1:80
- hope it works

Web Optimization

- ~~Organize CSS & JS~~
- ~~Multiple Domains~~
- ~~Gzip Compression~~
- ~~Resource Caching~~
- Far-Future Expiry
- Minification
- Image Optimization
- CSS Spriting

Tuesday, June 26, 12

(walk through these examples - left column only)

“Far Future” Expiry

- Good for (semi-)static resources
- Tells the browser to cache *forever**
- Browser won’t ask for the same thing (URL) again
- Need to change the url to refresh (“version”)

Tuesday, June 26, 12

* The official max is 1 year, but browsers don’t care
  - potential for breakage in proxy clients

Reduces request overhead significantly

(walk through setup)

Configure Far Future Expires

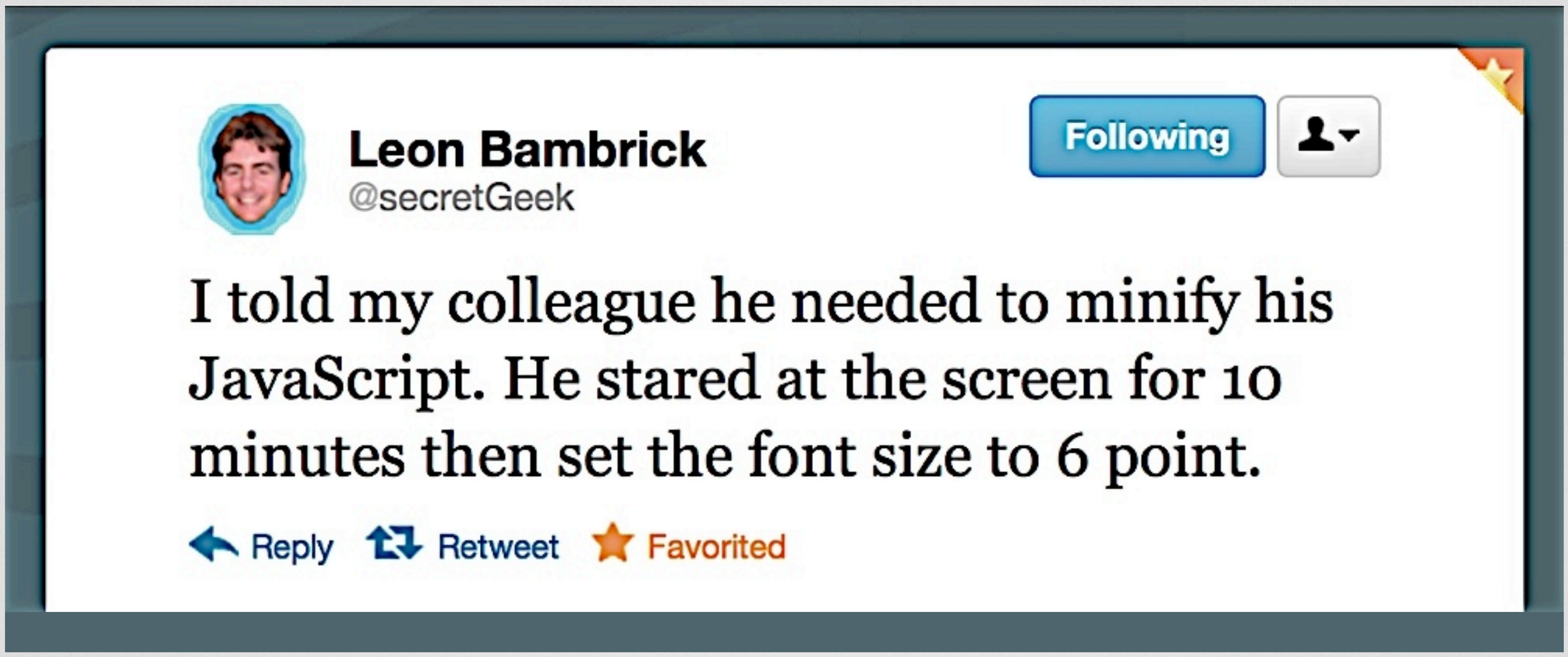
Web Optimization

- ~~Organize CSS & JS~~
- ~~Multiple Domains~~
- ~~Gzip Compression~~
- ~~Resource Caching~~
- Far Future Expiry
- Minification
- Image Optimization
- CSS Spriting

Tuesday, June 26, 12

But now we need to version our URLs

Minification



Minification

Don’t do this.

```
<?php
if ($config['env'] != 'prod') {
?>
    <script src="jquery-1.7.2.min.js"></script>
    <script src="jquery-ui-1.8.21.custom.min.js"></script>
    <script src="jquery.masonry.min.js"></script>
    <script src="bootstrap-modal.js"></script>
    <script src="bootstrap-twipsy.js"></script>
    <script src="bootstrap-alerts.js"></script>
    <script src="application.js"></script>
<?php
} else {
?>
    <script src="min.js?{{timestamp}}"</script>
<?php
}
?>
```


(real example, modified without altering the semantic content to keep the author anonymous)

So many things…
- Using individual minified scripts in development, why?
- Condition inside template, should be abstracted
- Blind assumption that the minified file really exists
- No way to “move” the scripts without global search-replace

Minification

Do this.

```
{% assets "boot.js" %}
    <script type="text/javascript" src="{{ ASSET_URL }}"></script>
{% endassets %}
{% block scripts %}{% endblock %}
```


Python w/ Flask-Assets

Minification

Or this.

```
# Load JS required to boot app
Config.loadScripts("boot").done =>
  # Start the app as soon as we have a user object
  if (@user)
    @start()
  else
    User.one "refresh change", =>
      @user = User.first()
      @start()
```


Tuesday, June 26, 12

Client-side non-blocking bundles

- CoffeeScript w/ jQuery Deferred objects
- There's also RequireJS

Minification

Along with this.

```
# ASSETS
ASSETS_DEBUG = False
ASSETS_VERSIONS = 'build'
ASSETS_URL = '//static.gooderlooking.com'
ASSETS_SCRIPTS = {
    'boot': [
        'scripts/jquery-1.7.2.js',
        'lib/jquery-ui-1.8.21/development-bundle/ui/jquery.ui.core.js',
        'lib/jquery-plugins/jquery.iecors.js',
        'scripts/spin.js',
        'scripts/site.js'
    ],
    'timelines': [
        'scripts/timelines.js'
    ]
}
```


Tuesday, June 26, 12

- The configuration is all in one place, makes it easy to toggle features
    - No need to manually rebuild when assets are added

- “bundles” allow for additional content to be cached and loaded as-needed.
    - users are like

Configure Minification

Versioning Static URL's

- md5 hashes
- Build numb
- Let's "cache

```
// Custom JavaScript app
'.../v147/scripts/core.js'
'.../v147/scripts/qualitycontrol.js'
'.../v147/scripts/application.js'


// Flask-Assets w/ custom versioning
'.../min/scripts/boot.73.js'
'.../min/scripts/timelines.73.js'
'.../min/scripts/admin.73.js'
```


Tuesday, June 26, 12

- good time to talk about how examples in this presentation are not perfected. might spark debate, which is a good thing
- possibly better:
    - localStorage
    - application manifest
        - "...honor HTTP caching semantics (such as expiration, ETags, and so forth) with respect to that cache." (http://www.whatwg.org/specs/web-apps/current-work/#application-cache-download-process)

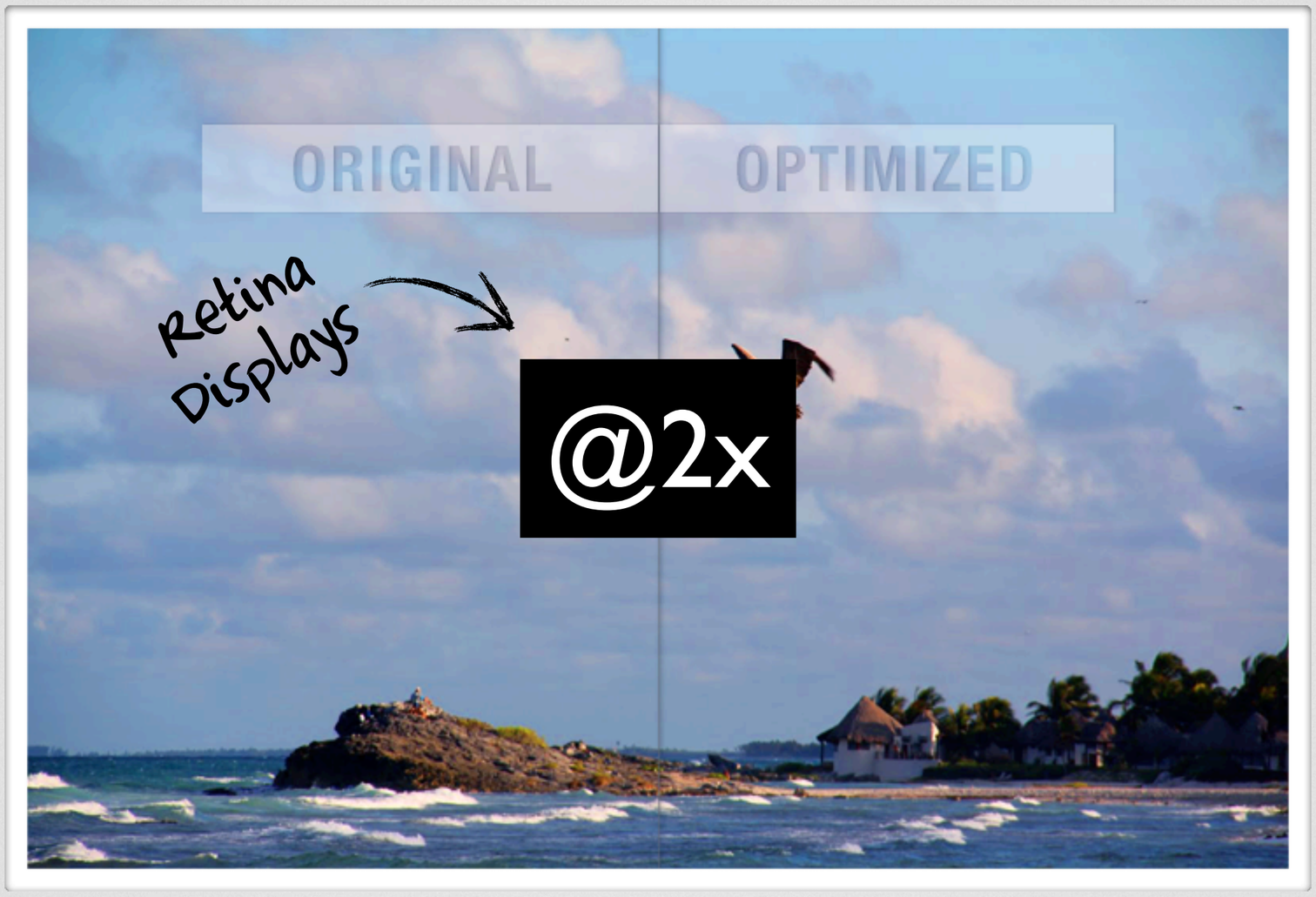
Versioning Walkthrough

- App appends current build to ‘base’ URL, which points to the ‘static’ subdomain
- Nginx server block receives the request
  - sets cache and gzip headers
  - try_files: awesome little thing

Web Optimization

- ~~Organize CSS & JS~~
- ~~Multiple Domains~~
- ~~Gzip Compression~~
- ~~Far Future Expiry~~
- ~~Minification~~
- Image Optimization
- ~~Resource Caching~~
- CSS Spriting

Image Optimization



Tuesday, June 26, 12

Libraries
↳ CLI tools
↳ GUI applications
↳ Web services

Images account for more than half of the overall payload.

Automation in this department is lacking.

Engineers are allergic to Photoshop
Unless you have a good reason, you want to optimize and cache before you serve an image (for the first time)

Apple just released retina MacBook Pro. Every site you've ever designed looks shit right now.
<link rel="stylesheet" type="text/css" href="/css/retina.css" media="only screen and (-webkit-min-device-pixel-ratio: 2)"/>

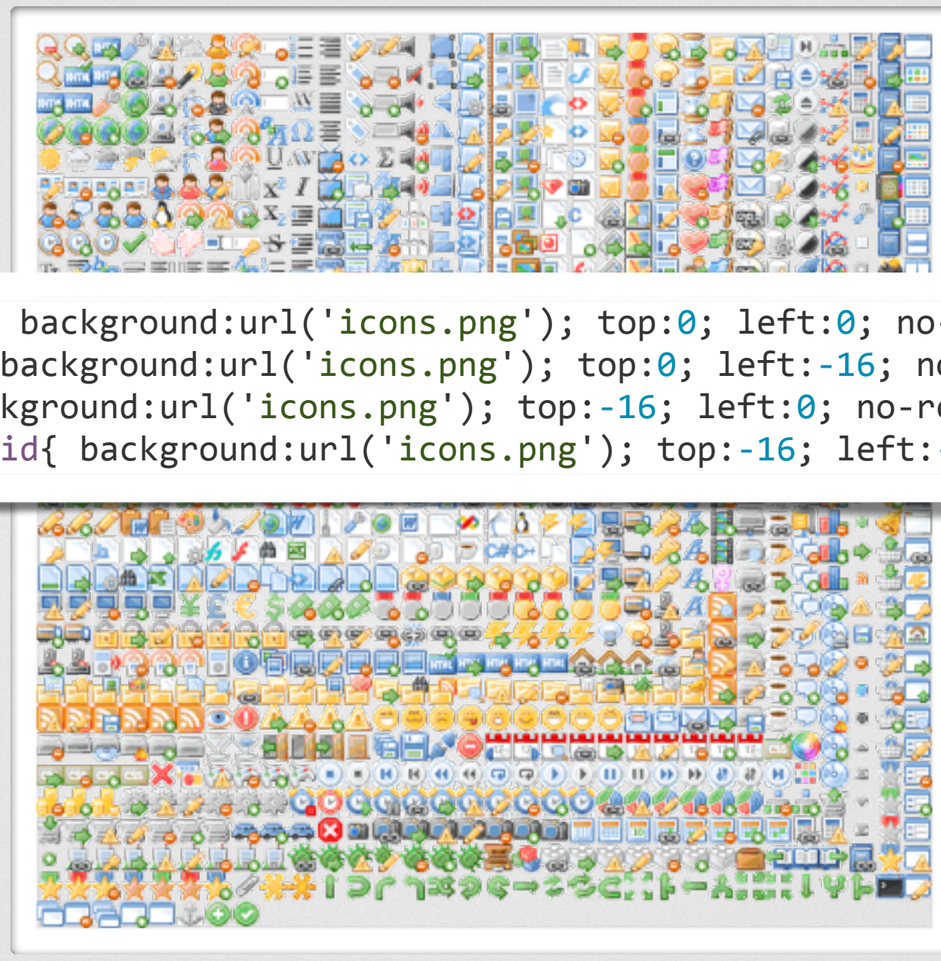
=====
Services
- JPEGmini (JPEG)
  - Have to email to get pricing, that means it's expensive
- kraken.io
- ImgMin (JPEG)
  - Open Source
- YSlow Smush.it™
  - Dying?

Desktop Apps

Configure Image Optimization

CSS Sprites

```
____icons
| |____accept.png
| |____add.png
| |____anchor.png
| |____application.png
| |____application_add.png
| |____application_cascade.png
| |____application_delete.png
| |____application_double.png
| |____application_go.png
| |____application_home.png
| |____application_key.png
| |____application_lightning.png
| |____application_link.png
...
```



```
.sprite-icons-zoom_out{ background:url('icons.png'); top:0; left:0; no-repeat;}
.sprite-icons-zoom_in{ background:url('icons.png'); top:0; left:-16; no-repeat;}
.sprite-icons-zoom{ background:url('icons.png'); top:-16; left:0; no-repeat;}
.sprite-icons-xhtml_valid{ background:url('icons.png'); top:-16; left:-16; no-repeat;}
```

Glue: https://github.com/jorgebastida/glue

Tuesday, June 26, 12

glue: https://github.com/jorgebastida/glue

- reduce request overhead
- use css clipping to show a single image
- really annoying to make by hand

- I will be turning it into a webassets filter

Web Optimization

- ~~Organize CSS & JS~~
- ~~Multiple Domains~~
- ~~Gzip Compression~~
- ~~Far Future Expiry~~
- ~~Minification~~
- ~~Image Optimization~~
- ~~Resource Caching~~
- ~~CSS Sprites~~

“Checkout-And-Go”

- Default configuration to dev settings
- Make it as easy as possible for new developers to get set up
- Override base for individual/stage/prod environments
- Offload as much as possible from the build process (“do it live!”)
- If you decide to pre-generate, create tasks in the application

Tuesday, June 26, 12

- Bad things happen when production values sit, ready to go, in a shared repo
- Especially necessary in Open Source, where people aren’t paid to set up your environment
- Parameterize config values to prevent config rot

THANK YOU!

For reals this time.

GitHub: @impressiver/velocity2012
Twitter: @impressiver
Email: ian@impressiver.com