DRUPAL – FRONTEND PERFORMANCE

Ashok Modi – DrupalCampLA 2011

About me

- Computer Systems Analyst at the California Institute of the Arts (http://calarts.edu)
  - See my profile at http://drupal.org/user/60422
  - Read my thoughts at http://btmash.com/
- Strong interest in Server Optimization.
  - Frontend and Backend

About this presentation

- "Backend" optimizations to speed up the frontend.
  - Show settings to tune on server for faster downloads by user
  - Useful modules.
  - Unless stated, assuming backend server is apache
- CSS & Javascript
  - Things to help make it faster.
- There will be some level of back and forth between the two.
- Have a question? Ask!
- Have something to share? Come on up!

Resources

- Konstantin Kaefer
  - http://kkaefer.com
- Steve Souders
  - http://stevesouders.com/blog
- Wim Leers
  - http://wimleers.com
- Addy Osmani
  - http://addyosmani.com

Goals

- Define your objectives and goals first
  - Do you want a faster response to the end user per page?
  - Do you want to handle more page views?
  - Do you want to minimize downtime?
- Related…but different
- Sometimes, there are 'low hanging fruit' that are easy to see and provide noticeable improvements
- Gets hard to achieve more performance (more effort, fewer gains)
  - More infrastructure?
  - Revisions at the JS/CSS Layer?
  - Revisions at the theme layer?

Diagnosis

- Proper diagnosis is essential before proposing and implementing a solution, or you're running blind.
- Based on proper data.
  - Analyze.
- Can lead to a few possible paths of optimization.

Points of optimization.

1. Introduction
2. Tools to measure and diagnose
3. Speed optimizations

‘Media’

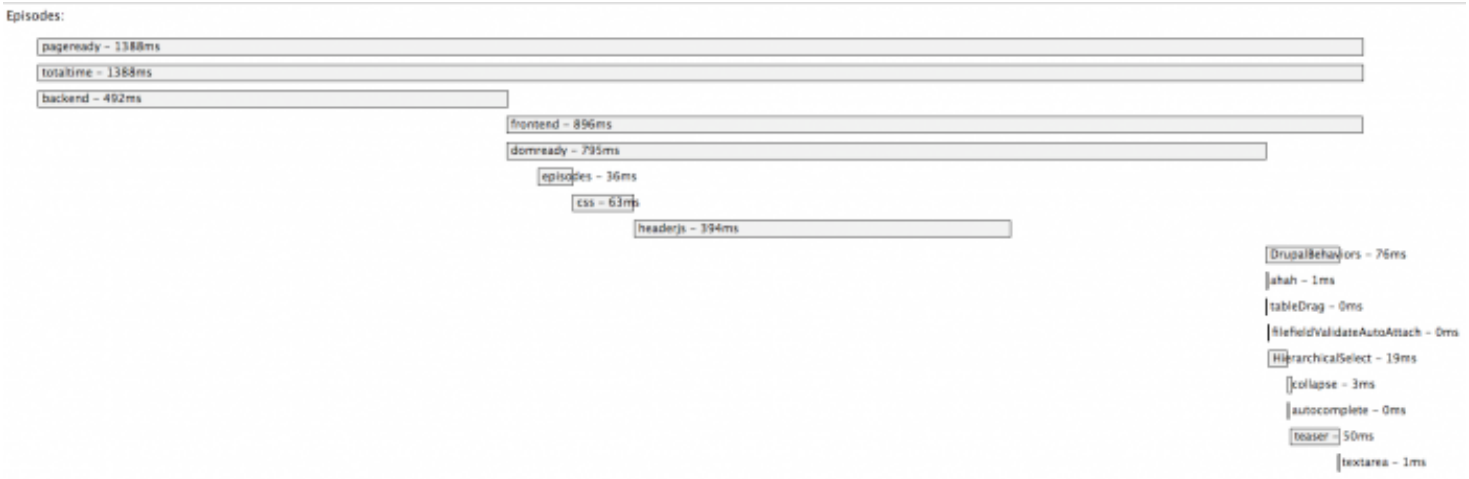
- Loading images, stylesheets, javascript can account for more than 80% of total load time.
  - Overall load time?
  - Page size?
  - Time until DOM is loaded?
  - Time until page is rendered?
  - Time until page is functional?

Tools to measure and diagnose

- Firebug's Net Panel
  - Track loading time for stylesheets, javascript, images, and total page load time.
- Yslow
  - Rates a webpage based on many criteria.
  - Determines overall load time.
  - Provides optimization suggestions.
  - Provides page statistics.
- Google Page Speed
  - Similar to Yslow.
  - Provides suggestions on improving CSS, optimizing image files, and minimizing javascript.
- Google Chrome Developer Panel
  - Very powerful.
  - Auditor helps track speed of js.

Tools to measure and diagnose (cont'd)

- Episodes
  - Aimed to measure timing of web 2.0 applications.
  - Granular measurements on how sections perform.
  - Drupal module (http://drupal.org/project/episodes)

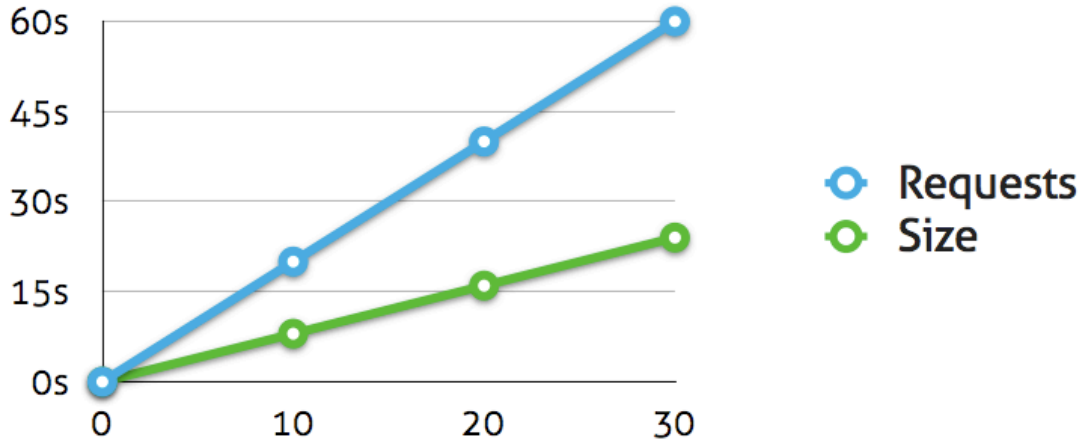


Non-Browser Tools

- AOL Page Test
  - http://webpagetest.org
- Pingdom
  - http://tools.pingdom.com
  - Waterfall diagram
- JSPerf
  - http://jsperf.com
  - Test out snippets of javascript
- CSSLint
  - http://csslint.com
  - Check and see suggestions on improving your CSS.

Backend Suggestion 1: Reduce Requests

- Every file produces an HTTP Request
- Fewer requests of large files are better than many smaller files (fewer server resources per user).
- Current browsers can download atleast *4* components per host in parallel.



Reduce Requests (cont'd)

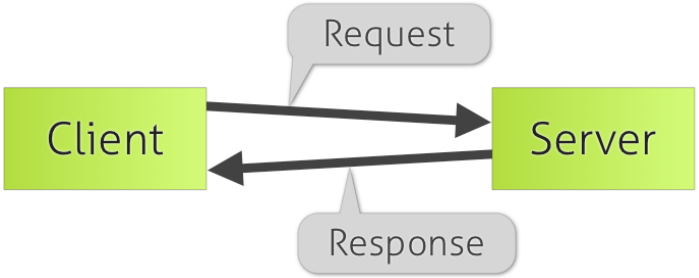
- Combine sprites together
  - Many images in one file.
  - Shift into view with CSS via background-position.
- Aggregate scripts and styles
  - Built into Drupal
  - NOTE: Drupal 7 splits up into many more files than before.
    - Not as 'efficient' in some ways as D6.
    - Pay attention to what happens with Bundle Cache at WPO (http://drupal.org/project/wpo)

Reduce Requests (cont'd)

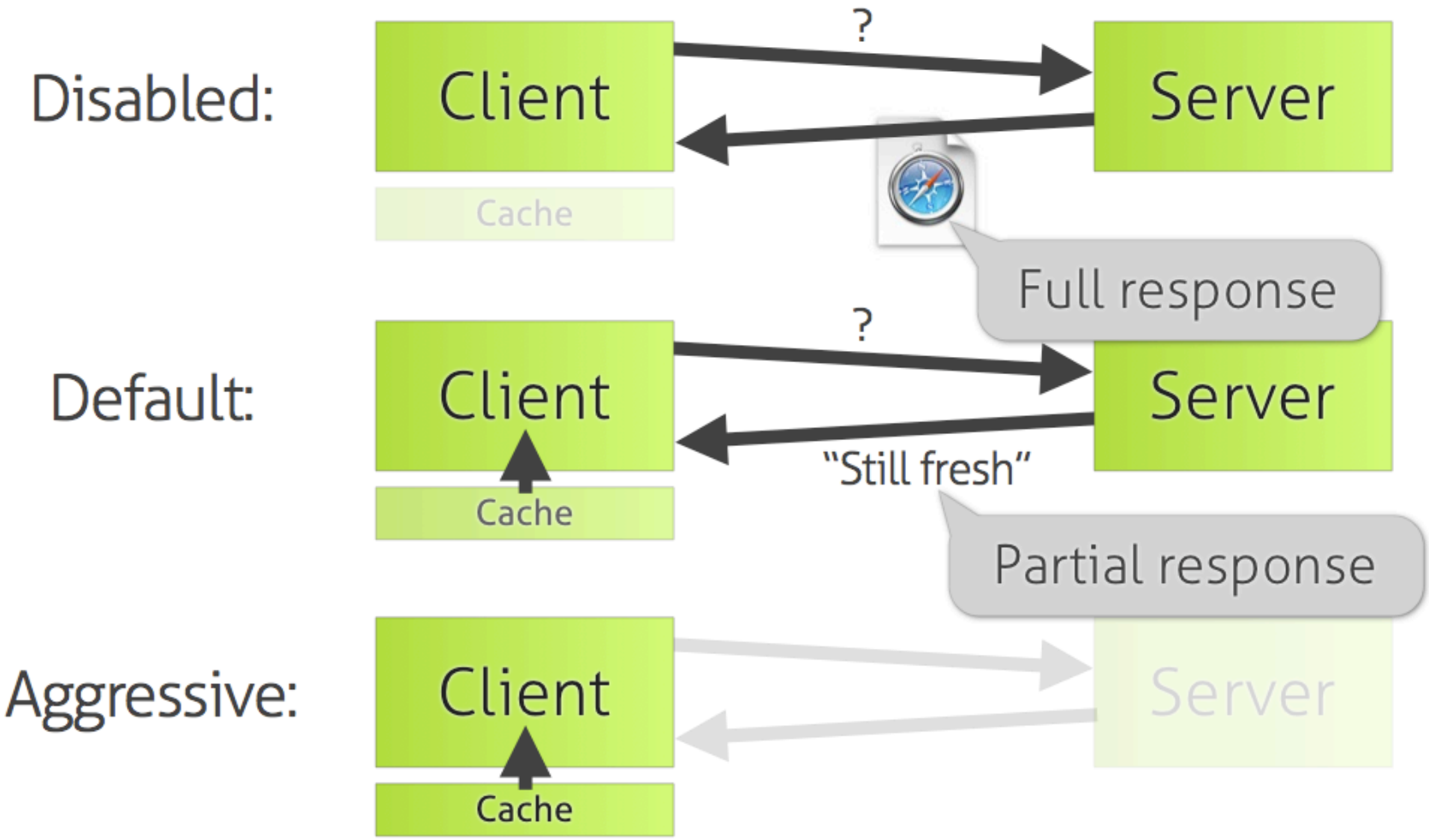
- Use CSS instead of images
  - border-radius, -moz-border-radius, -webkit-border-radius for rounded corners
- Use image data in CSS
  - background:url(/ge8WSLf/rhf/3kdbW1mxsbP//mf///yH5BAAAAAAALAAAAAAQAA4AAARe8L1Ekyky67QZ1hLnjM5UUde0ECwLJoExKcppV0aCcGCmTIHEIUEqjgaORCMxIC6e0CcguWw6aFjsVMkkIr7g77ZKPJjPZqIyd7sJAgVGoEGv2xsBxqNgYPj/gAwXEQA7) top left no-repeat;)
- Similarly, use font-data in CSS

Backend Suggestion 2: CDN

- Content Delivery Network.
- Lots of servers scattered around the world.
- Reduces roundtrip times.
- Example: using a CDN on one of my sites increased the requests/second from 150 r/s to 200 r/s.
- Can be inexpensive for hosting static files (7 cents/ GB)
- Akamai
- MaxCDN
- Cachefly
- Amazon Cloudfront
- Softlayer Cloudlayer



Backend Suggestion 3: Caching



Caching (cont'd)

- Controlled by HTTP Headers
- Browsers check if the content is 'fresh'
- Set expires header to a date in the future
- <location /css>
  ExpiresActive on
  ExpiresDefault "access plus 2 weeks"
  </location>
  ExpiresByType "access plus 1 month"
- For nginx: location ~* ^.+\.(css|js)$ { expires 30d; }
- Change filenames / url when updating.

Backend Suggestion #4: GZIP

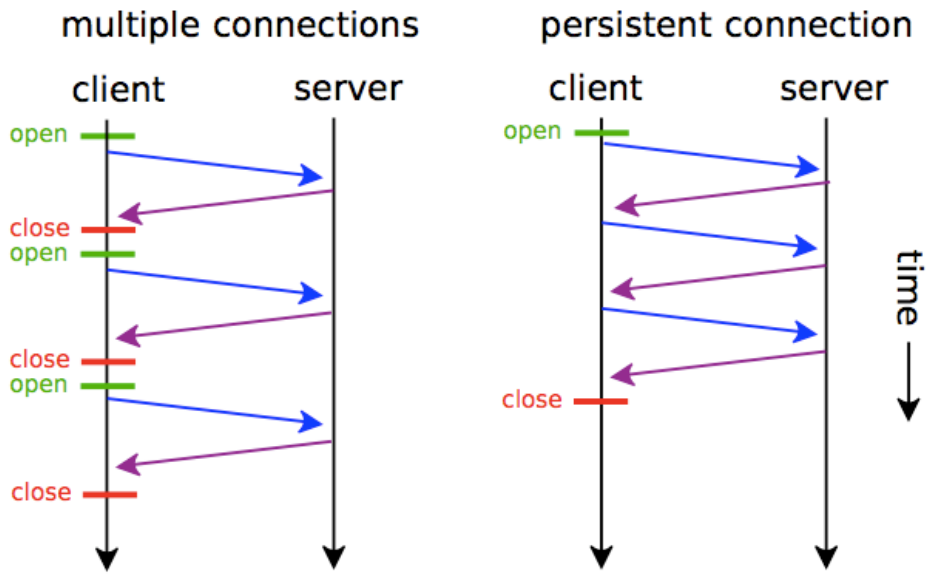
- Compress text content
  - <IfModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/css application/
    x-javascript
    </IfModule>
  - For nginx: http://goo.gl/PeDp
- Vastly reduces page size
  - http://redcat.org: 700kb -> 380kb (CSS went down from 120kb to 25kb)

Backend Suggestion #5: Parallelization

- Most browsers allow for more than 2 connections per host
  - IE8+: 6
  - FF3+: 6
  - Chrome: 6
  - Safari: 4
  - Opera: 4
- More hosts = more parallel downloads at same time.
  - Don't overdo it!
- CDN (http://drupal.org/project/cdn)
  - Doesn't have to be tied to a CDN
  - Point to your server with a different domain and users can download more of page at same time.

Backend Suggestion #6: Persistent HTTP

- Apache / NGINX support persistent connections



- Reconnecting over and over again takes time.
- Make sure KeepAlive is not turned off.

Backend Suggestion #7: Remove unnecessary cruft

- Apache enables 'modules' like mod_cgi, mod_proxy, mod_dav by default.
  - More likely than not, you do not need them.
  - Disabling will lower the memory usage by apache on a per connection basis.
  - More users.
- NginX starts off fairly lightweight and you have to add modules.
- Varnish makes no difference.

Theme Suggestion #1: CSS on TOP

- Placed in <head>
- Page is rendered when all css in header is loaded
- Loading CSS later causes re-rendering and user will see flash of unstyled content.
- <link> is faster than @import
  - But @import allows to use more than 32 files from IE restriction.
  - Drupal automatically handles itself between the two ☺

Theme Suggestion #2

- Placed right before </body>.
- Loading scripts blocks page rendering.
  - Downloading JS files will block downloads of other file types.
- Scripts are downloaded sequentially.
- Use graceful degradation.
  - Don't have *onfoo* event handlers in the code.

JS at the BOTTOM

Theme Suggestion #3: Minify JS/CSS

- Remove comments / whitespace.
- Great savings, even more with gzip
- No real modules for D7
  - Simple solution: Create a regular version of your file(s). Use a 3rd party tool (Google Closure Compiler) to minify/pack and use that.
  - CSS is slightly minified. But use Page Speed to get rid of unused CSS.

Theme Suggestion #4: Reduce image size

- OptiPNG, PNGCrush
  - Removes invisible content
  - Lossless compression
- JPEGTran/Imagemagick
  - Remove color profiles / metadata
  - Lossless JPEG operations
- http://smushit.com
- Google Page Speed also reports how much image can be compressed.
- User uploaded images can be compress for size by imagecache (D6) or in core (D7)

Theme suggestion #4: Lazy initialization

- Javascript takes time to initialize.
  - Libraries such as jQuery also count.
  - Defer setup work.
- Load images only when they need to be displayed.
  - Initial load on old calarts.edu website: 650kb.
  - After js lazy load: 250kb.
- Only load content above the fold.
  - Find out more at http://goo.gl/lpH4r
  - Very useful on image heavy sites.

CSS Improvements

- For the most part…fairly small gains
- Recent recommendations say to try and use classes as opposed to IDs
  - Performance difference is tiny.
  - Classes are much more flexible.
- Try to limit yourself to a second level selector (.class-one .child-class-two) especially if you use CSS3 selectors (.class:not(a), etc)
- Google Page Speed helps show which of your selectors could be improved.
- DEMO

jQuery improvements

- Newer releases of jQuery are generally faster.
  - jQuery Update module might be worthwhile to look at.
- Small tip: If you are going to call on the same selector(s) over and over, save them to a variable.
  - Should be a nice speed boost.
- Large area
  - Not all selectors are created equal ☺
  - ID and Element selectors are the fastest.
  - Class selectors are slightly slower.
  - Pseudo/Attribute selectors are the slowest
- Can test out a chunk of this at http://jsperf.com (used by jQuery team)

jQuery improvements (cont'd)

- Context vs. Find vs. Children vs…
- Find ($parent.find('.child')) is slightly faster than context ($('.child', $parent)) since the latter has to get parsed.
- $parent.children() – 50% slower.
- $('#parent > .child') – 70% slower.
- $('#parent .child) – **~80% slower.**
- $('#parent').find('.child') is about 16% slower than the fastest (since the #parent has to be fetched from the DOM)
  - As said before, best to get the $('#parent') object stored into a variable if it will be reused later.

jQuery improvements (cont'd)

- You don't always have to use jQuery.
  - $(item).attr('id') is **80 – 90% slower** than item.id
    - Apparently, item.id is also more up-to-date.
- Did I mention cache (store the result of) your result to a variable? ☺
  - http://jsperf.com/ns-jq-cached – This uncached is 56% slower on my computer.
- Chaining your methods on a selector ($('#item').doSomething().doSomethingElse()) is generally faster than separate calls.
- Try and keep your append() calls to a minimum if you absolutely need them.
  - Can be very costly (up to **90% decrease** in performance)

jQuery optimizations (cont'd)

- Try and bind event to as high of a parent as you possibly can.
  - Works with child elements at runtime and all those injected later.
  - .bind() doesn't automatically work with dynamically inserted elements
    - .live() allows for this though you cannot chain methods and really only for simple tasks.
- Working with modifying content by using detach() could provide a big boost in performance.
- while loops are faster than each() and $.each()

Questions?

- Interested in talking more? I sure am!
  - Let's talk after ☺
- What other presentations might be related and/or useful?
  - Drupal Design Skills 2012
  - Drupal Theme Design Q&A Panel
  - Designing Future Proof Websites
- Thank you! ☺