Front End Performance

Konstantin Käfer
Front End Performance?

- Load time of a web page
- Performance after page load
- Small devices?
Importance

- 80-90% is spent on loading components
- Absolute load time vs. Perception
- Render page as early as possible
Anatomy of a web page

- HTML
- CSS
- Scripts
- Background images
- Images
- Other media
Firebug + YSlow

- Rates a webpage based on 13 criteria
- Determines overall load time
- Provides suggestions
- Statistics
AOL Pagetest

- Creating waterfall diagrams
- Online version: http://webpagetest.org
- Grabs data from Internet Explorer
- Website not working in this venue :(
IBM Page Detailer

- Detailed waterfalls directly from IE
- Extremely detailed statistics
IBM Page Detailer

--- WD_CV_WS2_HTTP_HEADER_REQUEST(1057) CSTRING(7) Length=678
GET /sites/all/themes/szeged2008/images/calendar.png HTTP/1.1
Accept: */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .
Host: szeged2008.drupalcon.org
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: SESSd140531078345406c5c; __utma=37076706.4146334446790

--- WD_CV_WS2_HTTP_HEADER_REPLY(1056) CSTRING(7) Length=450
HTTP/1.0 200 OK
Date: Wed, 27 Aug 2008 13:15:34 GMT
Server: Apache
Last-Modified: Tue, 19 Aug 2008 14:05:09 GMT
ETag: "1cfdcc-a3fd-454d0935b2f40"
Accept-Ranges: bytes
Content-Length: 41979
Cache-Control: max-age=1209600
Expires: Wed, 10 Sep 2008 13:15:34 GMT
Content-Type: image/png
X-Cache: MISS from www3.drupal.org
X-Cache-Lookup: HIT from www3.drupal.org:80
Via: 1.0 www3.drupal.org:80 (squid/2.6.STABLE17)
Pingdom

- [Link](http://tools.pingdom.com/)
- Waterfall diagram
- Mimics web browser load order
Waterfall diagrams

- Start
- Connect
- Dispatch request
- Data transfer
- First byte
- Last byte

- Lower all three phases
WebKit’s Web Inspector

- [http://webkit.org](http://webkit.org)
Loading components

- HTTP 1.1: 2 components per host in parallel
- Waterfall diagrams show load order
- Ideally:
  - Short connect times
  - Narrow waterfall
TCP

- Transmission Control Protocol
- Stateful: Three Way Handshakes
- Round trip time has high effect
HTTP

- Stateless protocol on top of TCP
- Request/Response mechanism
- Header and Body have separate TCP packets
- POST involves $\geq 2$ packets, GET only 1
- Lots of different headers

http://www.w3.org/Protocols/rfc2616/rfc2616.html
Cookies are sent in HTTP header on each request
- Reduce cookie size
- Cookies are set per host name
  - Move components to a cookieless host

Keep-Alive/Persistent connections
- Reuse TCP connections
- Supported by most browsers/servers
HTTP Persistent connections

- Multiple connections:
  - Client: open → close → open → close
  - Server: open → close → open → close

- Persistent connection:
  - Client: open → close
  - Server: open → close
HTTPS

- Complex and time-consuming handshake

---

Phase 1:
- Client: generate random number (client_hello, crypto information)
- Server: generate random number
- Server: server_hello (crypto information)
- Server: server certificate
- Client: demand client certificate

Phase 2:
- Client: check server certificate
- Server: check server certificate
- Client: client certificate
- Server: client certificate
- Client: check client certificate
- Server: check client certificate

Phase 3:
- Client: check encrypted client certificate
- Server: check encrypted client certificate
- Server: generate random number pre-master-secret
- Server: calculate Master-Secret
- Client: change to encrypted connection

Phase 4:
- End SSL handshake
DNS

- Resolves host names to IP addresses
- Each host name has to be looked up
- Fewer host names ➔ fewer lookups
- `dig` command
Now what?
1. Reduce HTTP requests

- **Sprites**
  - Many images into one file
  - Shift into view with `background-position`

- **Aggregate** scripts and styles
  - Built into Drupal
  - Sophisticated: [http://drupal.org/project/sf_cache](http://drupal.org/project/sf_cache)

- No redirects
2. Use a CDN

- Content Delivery Network
- Lots of servers scattered around the world
- Reduces roundtrip times (ping)
- Comparably cheap: $0.07 - $0.80 per GB
  - http://www.simplecdn.com
  - http://pantherexpress.com/
  - http://cachefly.com/
3. Caching

- Controlled by HTTP headers
- Browsers check whether content is fresh
- Set Expires header to a date in the far future
- Change filenames/URLs when updating
4. GZip

› Compress text content (don’t use for images!)
› Vastly reduces page size
› NowPublic.com: 700 KB ➔ 300 KB
› Compress scripts and styles as well
5. CSS to the top

- == in <head>
- Page renders when all header CSS is loaded
- Loading CSS later causes re-rendering and FOUC
6. Scripts on the bottom

- == right before </body>
- Scripts block page rendering
- Scripts are loaded sequentially!
- Don’t use onfoo handlers in HTML code
- Graceful degradation
7. Minify CSS and JS

- Remove comments and whitespace
- Still savings, even with GZip
- Drupal’s aggregator or sf_cache.module
8. Parallelization + DNS

- HTTP 1.1: 2 requests per hostname in parallel
- Use multiple host names ➔ higher parallelization
- Don’t use too many hosts (DNS lookup time)
HTTP connections

“"A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy."" (RFC 2616, 8.1.4)
9. Reduce image weight

- OptiPNG, PNGCrush, ...
  - Removes non-visible content
  - Lossless recompression

- JPEGtran/ImageMagick
  - Remove color profiles, meta data, ...
  - Lossless JPEG operations
Resources

- http://yuiblog.com/blog/category/performance