Making Your Site Faster

And helping out those with bad internet

Dan Barrett — Digital Developer, LivingBrand

Current Statistics

- As of the 01/01/2015, the average page size for the Top 100 websites is 1448 KB[1]
- Top 1000 is 1889 KB[2]
- Related to JPEGs or Flash

[1] HTTP Archive: Top 100

[2] HTTP Archive: Top 1000

What do these results say about pagesize?

It tells us

- That the Top 100 sites have good developers

We don't all do the same

- All sites on HTTP Archive on the 1st of January of this year average 1931 KB in size[3]

[3] HTTP Archive: All

What should be done?

- Minimise the amount of HTTP requests
- Perform image compression
- Minify content where possible
- Strategic DOM manipulation

1. Minimise HTTP Requests

- Each additional request adds downtimes due to DNS lookups and initiating a GET request for the file
- Most browsers allow a maximum of 8 concurrent requests per unique domain name (not IP address, so use those CNAMEs)
- Concatenate, but do it wisely

2. Compress Images

- Images store unneeded comments, extra metadata colour profiles
- Use tools like ImageOptim[4], JPEGmini[5], and ImageAlpha[6]
- Or use a cloud service like Kraken[7] or EWWW IO[8]

---

[4] ImageOptim

[5] JPEGmini

[6] ImageAlpha

[7] Kraken

[8] EWWW IO

3. Minify Content

- Comments are great for dev team, but not necessary for the world to see
- Change variables from **`aVeryImportantVarName`** to **`a`** automatically
- Concatenate source files, but use CDNs for common frameworks (i.e. jQuery[9])

[9] jQuery on Google CDN

4. DOM Manipulation

- Writing to the DOM is slow!
- Ideally search using ID or tag selectors[10] [11]
- Use **`<canvas>`** xor React for crazy-fast performance[12]
- Combine alterations to a node into one task (if possible)[13]

---

[10] Selector optimisation with 24 Ways

[11] 10 performance tips from Paul Irish

[12] Flipboard goes to 60

[13] DOM node alterations

5. The Easy Stuff (Surprise Slide)

- Put your **`<script>`** tags in the footer (or use magic)[14]
- Load CSS asynchronously (e.g. Enhance.js[15], Yepnope[16], RequireJS[17], etc) to stop it blocking your page load

---

[14] The murky waters of script loading

[15] Enhance.js on GitHub

[16] Yepnope

[17] RequireJS

Is this practical to do in the real world?

Yes!

Personal Case Study #1

- Client with products page list - weighed in at **12.2 MB**, very slow to render
- Due to: no concatenation & minification, bad use of images (600x600 scaled down to 200x200), dead/poorly written code
- After refactor: **2.5 MB** with optimised images and minified JS/ CSS (with no dead code)

Personal Case Study #2

- JavaScript function polled every 100ms on **`scroll`** and **`resize`** events
- Before optimisation took **~7.9ms** to complete and wrote to the DOM every time
- After optimisation… **~0.2ms** to complete and only touches the DOM when absolutely necessary