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]
- Mainly related to images 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?

1. Minimise the amount of HTTP requests
2. Perform image compression
3. Minify content where possible
4. Strategic DOM manipulation
5. ???
6. Profit

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 the dev team, but not necessary for the world to see
- Changes variables from **`aVeryImportantVarName`** to **`a`** automatically
- Concatenate source files, but use CDNs for common frameworks/libraries (i.e. jQuery[9])[10]

---

[9] jQuery on Google CDN

[10] Letting Google host jQuery for you

4. DOM Manipulation

- Writing to the DOM is slow!
- Ideally search using ID or tag selectors[11] [12]
- Use `<canvas>` xor React for crazy-fast performance[13]
- Combine alterations to a node into one task (if possible)[14]

---

[11] Selector optimisation with 24 Ways

[12] 10 performance tips from Paul Irish

[13] Flipboard goes to 60

[14] DOM node alterations

5. The Easy Stuff

- Put your **`<script>`** tags in the footer (or use magic)[15]
- Load CSS asynchronously (e.g. Enhance.js[16], Yepnope[17], RequireJS[18], etc) to stop it blocking your page load
- Lazy load images so only images near the viewport are loaded[19]

---

[15] The murky waters of script loading

[16] Enhance.js on GitHub

[17] Yepnope

[18] RequireJS

[19] lazysizes image loader

5. The Easy Stuff (*cont*…)

- Use JPEGs for photos, not PNGs (surprising how often people stuff this up)
- Better yet, use the power of responsive images[20] with **`<picture>`** or **`<img>`** on steroids (polyfill available[21])
- Use CSS sprites[22] for icons (or SVG sprites or icon fonts[23])

---

[20] Responsive Images

[21] Picturefill - Responsive images polyfill

[22] _.throttle by Underscore JS

[23] Chrome DevTools Page

Care About Your Users

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[22] 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

---

[22] _.throttle by Underscore JS

The Takeaway

We can all make a difference

- Take the time to ensure your code isn't writing to the DOM unnecessarily
- Use the Chrome DevTools[23] to run tests and see how your code performs[24]

---

[23] Chrome DevTools Page

[24] Discover DevTools course