Performance

and PWAs

Why Performance Matters

- Pinterest increased search traffic and sign-ups by 15% when they reduced *perceived* wait times by 40%
- BBC found they lost an additional 10% of users for every additional second their site took to load
- 53% of mobile site visits were abandoned if a page took longer than 3 seconds to load

Performance is about the user experience

- As a site begins to load, there’s a period of time where users wait for content to appear
- Until this happens, there’s no user experience to speak of
- Performance is a foundational aspect of good user experience

Perceived Performance

- Perceived performance refers to how fast a user *thinks* your website is, not necessarily how fast your technical stats say it is
- In addition to improving your website according to PageSpeed scored you should make your website *feel* faster

Response Times

- 0.1 seconds
  - Operations that are completed in 100ms or faster will feel instant for users
- 1 second
  - Generally OK, but might make website feel a bit sluggish
- 3-10 seconds
  - When you lose the users attention

Improve perceived performance

- Loading dialogs
  - Progress bars
  - Animations

- Mirrors in elevators

How to improve performance

- Mind what resources you send
- Mind how you send resources
- Mind how much data you send

Mind what resources you send

- If you use Bootstrap or Foundation to build your UI, ask yourself if they're necessary. Such abstractions add heaps of CSS the browser must download, parse, and apply to a page, all before your site-specific CSS enters the picture.Flexbox and Grid are superb at creating both simple and complex layouts with relatively little code.
- JavaScript libraries are convenient, but not always necessary. For example, Zepto is a smaller jQuery alternative, and Preact is a much smaller alternative to React.
- Not all websites need to be single page applications (SPAs), as they often make extensive use of JavaScript. JavaScript is the most expensive resource we serve on the web byte for byte, as it must not only be downloaded, but parsed, compiled and executed as well. For example, news and blog sites with optimized front end architecture can perform well as traditional multipage experiences. Particularly if HTTP caching is configured properly, and optionally, if a service worker is used.

Mind how you send resources

Efficient delivery is vital to building fast user experiences.

- Migrate to HTTP/2. HTTP/2 addresses many performance problems inherent in HTTP/1.1, such as concurrent request limits and the lack of header compression.
- Download resources earlier using resource hints like rel=preload
- Consider using code splitting in webpack to limit the amount of scripts downloaded to only what is needed by the current page or view. Separate your CSS into smaller template or component-specific files, and only include those resources where they're likely to be used.

Mind how much data you send

Here are some suggestions for limiting *how much* data you send:

- Minify text assets. Minification is the removal of unnecessary whitespace, comments and other content in text-based resources.
- Configure your server to compress resources. Compression drastically reduces the amount of data you send to users, *especially* text assets.
- Optimize images to ensure your site sends as little image data as possible. Since images make up a large portion of the average per-page payload on the web, image optimization represents a uniquely large opportunity to boost performance.
  If you have time, consider serving alternative image formats.
- Use video instead of animated GIFs. Animated GIFs are *massive*. Videos of similar quality are *far* smaller, often by 80% or so. If your site makes heavy use of animated GIFs, this is probably the most impactful thing you can do to improve loading performance.