Better Site Speed: 4 Outside-the-Box Ideas

Most of us have done site speed audits, or seen audits done by others. These can be really helpful for businesses, but I often find they're quite narrow in focus. Typically we use well-known tools that throw up a bunch of things to look at, and then we dive into things from there.

However, if we dig deeper, there are often other ideas on how site speed can be improved. I often see plenty of opportunities that are never covered in site speed audits. Most site speed improvements are the result of a bunch of small changes, and so in this post I'm going to cover a few ideas that I've never seen in any site speed audit, all of which can make a difference.

A different angle on image optimization

Consider optimized SVGs over PNGs

I was recently looking to book some tickets to see Frozen 2 (because of, erm, my kids…) and so landed on this page. It makes use of three SVG images for transport icons:

SVG images are vector images, so they're well-suited for things like icons; if you have images displayed as PNGs you

may want to ask your designers for the original SVGs, as there can be considerable savings. Though not always better, using an SVG can save 60% of the filesize.

In this case, these icons come in at about 1.2k each, so they are quite small. They would probably fly under the radar of site speed audits (and neither Page Speed Insights or GTMetrix mention these images at all for this page).

So you may be thinking, “They’re less than 5k combined – you should look for bigger issues!”, but let’s take a look. Firstly, we can run them all through Jake Archibald’s SVG compression tool; this is a great free tool and on larger SVGs it can make a big difference.

In this case the files are small, so you may still be thinking “Why bother?” The tool compresses them without any loss in quality from ~1240 bytes to ~630 bytes – a good ratio but not much of an overall saving.

However… now that we’ve compressed them, we can think differently about delivering them…

Inline images

GTMetrix makes recommendations around inlining small bits of CSS or JS, but doesn’t mention inlining images. Images can also be inlined, and sometimes this can be the right approach.

If you consider that even a very small image file requires a complete round trip (which can have a very real impact on speed), even for small files this can take a long time. In the case of the Cineworld transport images above, I simulated a “Fast 3G” connection and saw:

The site is not using HTTP2 so there is a long wait period, and then the image (which is 1.2kb) takes almost 600ms to load (no HTTP2 also means this is blocking other requests). There are three of these images, so between them they can be having a real impact on page speed.

However, we've now compressed them to only a few hundred bytes each, and SVG images are actually made up of markup in a similar fashion to HTML:

You can actually put SVG markup directly into an HTML document!

If we do this with all three of the transport images, the compressed HTML for this page that is sent from the server to our browser increases from 31,182 bytes to 31,532 bytes – an increase of only 350 bytes for all 3 images!

So to recap:

- Our HTML request has increased 350 bytes, which is barely anything
- We can discard three round trips to the server, which we

can see were taking considerable time

Some of you may have realized that if the images were not inline they could be cached separately, so future page requests wouldn't need to refetch them. But if we consider:

- Each image was originally about 1.5kb over the network (they aren't gzipping the SVGs), with about 350 bytes of HTTP headers on top for a total of about 5.5kb transferred. So, overall we've reduced the amount of content over the network.
- This also means that it would take over 20 pageviews to benefit from having them cached.

Takeaway: Consider where there are opportunities to use SVGs instead of PNGs.

Takeaway: Make sure you optimize the SVG images, use the free tool I linked to.

Takeaway: Inlining small images can make sense and bring outsized performance gains.

Note: You can also inline PNGs – see this guide.

Note: For optimized PNG/JPG images, try Kraken.

Back off, JavaScript! HTML can handle this…

So often nowadays, thanks to the prevalence of JavaScript libraries that offer an off-the-shelf solution, I find JavaScript being used for functionality that could be achieved without it. More JS libraries means more to download, maybe more round trips for additional files from the server, and then the JavaScript execution time and costs themselves.

I have a lot of sympathy for how you get to this point. Developers are often given poor briefs/specs that fail to

specify anything about performance, only function. They are often time-poor and so it’s easy to end up just dropping something in.

However, a lot of progress has been made in terms of the functionality that can be achieved with HTML and or CSS. Let’s look at some examples.

Combo box with search

Dropdown boxes that have a text search option are a fairly common interface element nowadays. One recent article I came across described how to use the Select2 Javascript library to make such a list:

It is a useful UI element, and can help your users. However, in the Select2 library is a JavaScript library, which in turn relies on some CSS and the JQuery library. This means three round trips to collect a bunch of files of varying sizes:

- JQuery – 101kb
- Select2 JavaScript – 24kb
- Select2 CSS – 3kb

This is not ideal for site speed, but we could certainly make the case it is worth it in order to have a streamlined interface for users.

However, it is actually possible to have this functionality out of the box with the HTML datalist element:

This allows the user to search through the list or to free type their own response, so provides the same functionality. Furthermore, it has a native interface on smartphones!

You can see this in action in this codepen.

Details/Summary

LonelyPlanet has a beautiful website, and I was looking at this page about Spain, which has a ‘Read More’ link that most web users will be familiar with:

Like almost every implementation of this that I see, they have used a JavaScript library to implement this, and once again this comes with a bunch of overheads.

However, HTML has a pair of built-in tags called *details* and *summary*, which are designed to implement this functionality exactly. For free and natively in HTML. No overheads, and more accessible for users needing a screen reader, while also

conveying semantic meaning to Google.

These tags can be styled in various flexible ways with CSS and recreate most of the JS versions I have seen out there.

Check out a simple demo here: https://codepen.io/TomAnthony/pen/GRRLrmm

…and more

For more examples of functionality that you can achieve with HTML instead of JS, check out these links:

- http://youmightnotneedjs.com/
- https://dev.to/ananyaneogi/html-can-do-that-c0n

Takeaway: Examine the functionality of your sites and see where there may be opportunities to reduce your reliance on large Javascript libraries where there are native HTML/CSS options.

Takeaway: Remember that it isn't only the size of the JS files that is problematic, but the number of round trips that are required.

Note: There are cases where you should use the JS solution, but it is important to weigh up the pros and cons.

Networking tune-ups

Every time the browser has to collect resources from a server, it has to send a message across the internet and back; the speed of this is limited by the speed of light. This may sound like a ridiculous thing to concern ourselves with, but it means that even small requests add time to the page load. If you didn't catch the link above, my post explaining HTTP2 discusses this issue in more detail.

There are some things we can do to help either reduce the

distance of these requests or to reduce the number of round trips needed. These are a little bit more technical, but can achieve some real wins.

TLS 1.3

TLS (or SSL) is the encryption technology used to secure HTTPS connections. Historically it has taken two round trips between the browser and the server to setup that encryption – if the user is 50ms away from the server, then this means 200ms per connection. Keep in mind that Google historically recommends aiming for 200ms to deliver the HTML (this seems slightly relaxed in more recent updates); you're losing a lot of that time here.

The recently defined TLS 1.3 standard reduces this from two round trips to just one, which can shave some precious time off the users initial connection to your website.

Speak to your tech team about migrating to TLS 1.3; browsers that don't support it will fallback to TLS 1.2 without issue. All of this is behind the scenes and is not a migration of any sort. There is no reason not to do this.

If you are using a CDN, then it can be as simple as just turning it on.

You can use this tool to check which versions of TLS you have enabled.

QUIC / HTTP 3

Over the last 2-3 years we have seen a number of sites move from HTTP 1.1 to HTTP 2, which is a behind-the-scenes upgrade which can make a real improvement to speed (see my link above if you want to read more).

Right off the back of that, there is an emerging pair of standards known as QUIC + HTTP/3, which further optimize the

connection between the browser and the server, further reducing the round trips required.

Support for these is only just beginning to become viable, but if you are a CloudFlare customer you can enable that today and over the coming 6 months as Chrome and Firefox roll support out, your users will get a speed boost.

Read more here: https://blog.cloudflare.com/http3-the-past-present-and-future/

Super routing

When users connect to your website, they have to open network connections from wherever they are to your servers (or your CDN). If you imagine the internet as a series of roads, then you could imagine they need to ‘drive’ to your server across these roads. However, that means congestion and traffic jams.

As it turns out, some of the large cloud companies have their own private roads which have fewer potholes, less traffic, and improved speed limits. If only your website visitors could get access to these roads, they could ‘drive’ to you faster!

Well, guess what? They can!

For CloudFlare, they provide this access via their Argo product, whereas if you are on AWS at all then you can use their Global Accelerator. This allows requests to your website to make use of their private networks and get a potential speed boost. Both are very cheap if you are already customers.

Takeaway: A lot of these sorts of benefits are considerably easier to get if you’re using a CDN. If you’re not already using a CDN, then you probably should be. CloudFlare is a great choice, as is CloudFront if you are using AWS. Fastly is the most configurable of them if you’re more of a pro.

Takeaway: TLS 1.3 is now very widely supported and offers a

significant speed improvement for new connections.

Takeaway: QUIC / HTTP3 are only just starting to get support, but over the coming months this will roll out more widely. QUIC includes the benefits of TLS 1.3 as well as more. A typical HTTP2 connection nowadays needs 3 round trips to open; QUIC needs just one!

Takeaway: If you're on CloudFlare or AWS, then there is potential to get speed ups just from flipping a switch to turn on smart routing features.

Let CSS do more

Above I talked about how HTML has built-in functionality that you can leverage to save relying on solutions that are 'home-rolled' and thus require more code (and processing on the browsers side) to implement. Here I'll talk about some examples where CSS can do the same for you.

Reuse images

Often you find pages that are using similar images throughout the page in several places. For example, variations on a logo in different colors, or arrows that point in both directions. As unique assets (however similar they may be), each of these needs to be downloaded separately.

Returning to my hunt for cinema tickets above, where I was looking at this page, we can see a carousel that has left and right arrows:

Similarly to the logic used above, while these image files are small, they still require a round trip to fetch from the server.

However, the arrows are identical – just pointing in opposite directions! It's easy for us to use CSS's *transform* functionality to use one image for both directions:

You can check out this codepen for an example.

Another example is when the same logo appears in different styles on different parts of the page; often they will load multiple variations, which is not necessary. CSS can re-color logos for you in a variety of ways:

There is a codepen here showing this technique in action. If you want to calculate the CSS filter value required to reach an arbitrary color, then check out this amazing color calculator.

Interactions (e.g. menus & tabs)

Often navigation elements such as menus and tabs are implemented in JavaScript, but these too can be done in pure CSS. Check out this codepen for an example:

Animations

CSS3 introduced a lot of powerful animation capability into CSS. Often these are not only faster than JavaScript versions, but can also be smoother too as they can run in the native

code of the operating system rather than having to execute relatively slower Javascript.

Check out Dozing Bird as one example:

You can find plenty more in this article. CSS animations can add a lot of character to pages at a relatively small performance cost.

…and more

For more examples of functionality that you can achieve using pure CSS solutions, take a look at:

- http://youmightnotneedjs.com/
- https://dev.to/ananyaneogi/css-can-do-that-18g7m

Takeaway: Use CSS to optimize how many files you have to load using rotations or filters.

Takeaway: CSS animations can add character to pages, and often require less resources than JavaScript.

Takeaway: CSS is perfectly capable of implementing many interactive UI elements.

Wrap up

Hopefully you've found these examples useful in themselves, but the broader point I want to make is that we should all try to think a bit more out of the box with regards to site speed. Of particular importance is reducing the number of round trips needed to the server; even small assets take some time to fetch and can have an appreciable impact on performance (especially mobile).

There are plenty more ideas than we've covered here, so please

do jump into the comments if you have other things you have come across.

http://tracking.feedpress.it/link/9375/13107756