

Should you self-host Google Fonts?

Category:

1. [Blog](#)

This page was originally created on 12-Jan-2020 and last edited on 23-Feb-2020.

Introduction

In the last few weeks, because of a combination of various things at work, and in side-projects, I've been learning a lot about web fonts and also a lot more about Google Fonts specifically. Through that I've come up with a more nuanced answer to the question, that in the past I thought was easy: should you self-host Google Fonts?

Now, to be totally up front, I'll admit that fonts are not my strong point. I'm much more practical than design-y (look at this website for evidence of that!) and have never totally got the need for fonts. Sure they look a bit nicer, and can understand they make a message seem more on-brand, but for the main body of text at least they seem more of a nice to have - I've never read an article more or less (or treated the contents any differently) because it had a pretty font. However, I've also been acutely aware of the performance implications of them so maybe that's clouded my view of them.

Still, many feel differently, and fonts are here, whether I appreciate them or not, and many developers aren't given a choice whether to use them or not. So let's look at what we can do to minimise the performance impact, but also give the designers what they want - win win!

Self-Hosting assets versus Third-Party hosting

A few years back it was all the rage to use a CDN to serve common assets (e.g. jQuery from <https://code.jquery.com> - and yes [jQuery is still very much a thing!](#)). To be clear when I say CDN here, I mean where you are loading some assets from someone else's domain, rather than a CDN fronting your domain.

The theory behind this was that browsers limited the number of connections to each domain (typically to 6 connections) so using another domain gave you 6 more connections. While that may have been true in the past (particularly when browsers limited it to less than 6 domains) and before [HTTPS](#) became the norm, now [connections are expensive to create](#). Additionally [HTTP/2](#) actually benefits from one connection (mostly!) so using other domains is often a performance cost rather than gain.

Another way of doing this was by sharding your domain with one or more assets subdomain (e.g. assets.example.com) so again the fonts are not hosted on your main domain where your web page is loaded from. However, it has the same connection issues so again this is not the performance benefits it may once have had.

The other supposed benefit of using a public CDN, was from leveraging the fact that visitors might already have that version of jQuery loaded in their HTTP cache, but again I'm convinced that's over-egged. There are so many libraries and versions, and [browser caches are smaller than you think](#), so for you to be lucky enough to gain from this seems unlikely. Additionally [Safari has a unique HTTP cache per domain visited](#), for privacy reasons (called a double-keyed cache) and [Chrome soon will have too](#), ending any argument there.

That leads nicely into the privacy implications of using third-party CDNs. You have no idea what sort of tracking they are doing to your users by using them, rather than self-hosting. And recent legislation means a lot of sites have to explicitly list all the cookies used on the site, which gets more complicated when using a third-party.

I've been convinced for a while now that third-party CDNs, or even sharding your own domains, are not the performance boast they are thought to be. All too often you see the main domain serving the index.html, and then that connection not being used for anything else [as time is instead wasted setting up new connections](#).

This is not to mention the security implications of loading assets from another domain. Yes there is [SRI](#) but that can [cause unexpected issues](#), and I honestly don't see the point. If it is a static asset (where you can use SRI) then self-host, and if it is not static (because the contents are liable to change) then you can't use SRI.

On a related point, using a third-party, also introduces the risk of them becoming a single point of failure (SPOF) and taking down your website if it goes offline for any reason. This has been [recognised for a long time](#) and while it may seem unlikely that Google Fonts will go down, it can be blocked by company proxies or [whole countries](#). All in all, more and more have been advising to [self-host your static assets](#), ideally on the domain you serve the web pages from. Fonts are static assets, so they should also be self-hosted right? Well it turns out it is not quite as simple as that because fonts have their own peculiarities and performance optimisations that might make self-hosting a little trickier...

Google Fonts and how they work.

[Google Fonts](#) is an amazing resource for those of you that are into your fonts. It has 977 open-source fonts for anyone to use completely for free. Commercial fonts are ridiculously expensive for those of you that have ever looked into them and they are also usually licenced rather than bought, and are charged based on expected number of page views - like they will run out through use! To have so many free fonts in one collection and so easy to use is therefore very useful.

Google Fonts, however, takes it one step further. Like many providers of website assets (see jQuery example above), they also provide a CDN and host the fonts for you to use directly from them. This means you can start using fonts just by adding one line of code to your website to pull in the style sheet, like this:

```
<link href="https://fonts.googleapis.com/css?family=Lato&display=swap"
rel="stylesheet">
```

You can also add more weights and fonts to that one line to load several fonts and variations of each font:

```
<link
href="https://fonts.googleapis.com/css?family=Lato:400,400i,700,700i,900%
7CPoppins:300,400,700,900&display=swap" rel="stylesheet">
```

The downside to this is in performance (the upside is also in performance but that side is not as obvious - we'll get to that). The problem is that your website (say www.example.com) loads the stylesheet from fonts.googleapis.com, which returns some CSS made up of `font-face` declarations. Using the first example above, returns this then I view that URL in Chrome:

```
/* latin-ext */
@font-face {
  font-family: 'Lato';
  font-style: normal;
  font-weight: 400;
  font-display: swap;
  src: local('Lato Regular'), local('Lato-Regular'),
url (https://fonts.gstatic.com/s/lato/v16/S6uyw4BMUTPHjxAwXiWtFCfQ7A.woff2
) format('woff2');
  unicode-range: U+0100-024F, U+0259, U+1E00-1EFF, U+2020, U+20A0-20AB,
U+20AD-20CF, U+2113, U+2C60-2C7F, U+A720-A7FF;
}

/* latin */
@font-face {
  font-family: 'Lato';
  font-style: normal;
  font-weight: 400;
  font-display: swap;
```

```

    src: local('Lato Regular'), local('Lato-Regular'),
    url(https://fonts.gstatic.com/s/lato/v16/S6uyw4BMUTPHjx4wXiWtFCc.woff2)
    format('woff2');

    unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02BB-02BC, U+02C6,
    U+02DA, U+02DC, U+2000-206F, U+2074, U+20AC, U+2122, U+2191, U+2193,
    U+2212, U+2215, U+FEFF, U+FFFD;
}

```

We'll get into what some of those settings mean later (and also why there are two `font-face` declarations), but for now this means you can use this font in your style like the following:

```

body {

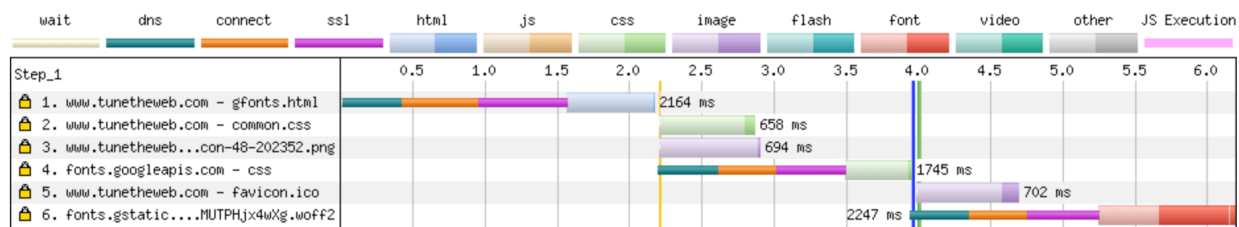
    font-family: 'Lato', sans-serif;

}

```

However this means you have to connect to `fonts.googleapis.com`, download the CSS, then connect to `fonts.gstatic.com` to download the actual fonts (why Google can't host both the CSS and the fonts on the one domain I really don't know!).

Fonts are often discovered late by the browser when loading a page (as you need to download the CSS to see the font references) but Google Fonts are discovered extra late, as you need to download the CSS from *another domain*, then the fonts from a *3rd domain* and, as discussed above, making an HTTPS connection takes time. This can be seen in the following waterfall diagram generated by [WebPageTest](#) (note all tests were run with Chrome - 3GSlow):



You can see on line 1 we are downloading the HTML then, once that's downloaded and read at just under 2 seconds, the browser sees the need for the Google Fonts CSS and downloads it on line 4. This takes a second just to make the connection, and then at 3.5 seconds we download the stylesheet, and we see the actual font we need and download that on line 6 - which takes about another second and a quarter to make the connection to `fonts.gstatic.com`, before we can actually start downloading the font.

So we can see using this font from Google Fonts is costing us a full 3 seconds of performance from the HTML being available, before we even start downloading the font!

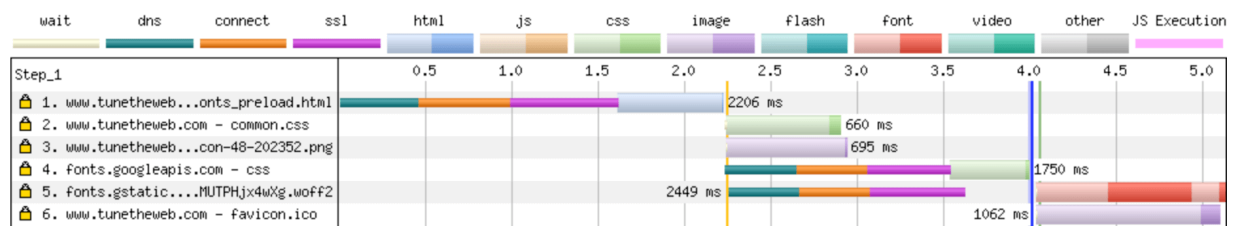
Improving Google Fonts with resource hints.

We can mitigate some of this performance hit of downloading the CSS and then the fonts from two different domains. The first domain (for the CSS) should be fairly high up your index.html so hopefully will be seen early enough, but the second domain is not seen until later. However we *know* what that domain will be (`fonts.gstatic.com`), so we can use a `preconnect` resource hint to ask the browser to open the connection in advance to save some of that second connection time later:

```
<link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>

<link href="https://fonts.googleapis.com/css?family=Lato&display=swap"
rel="stylesheet">
```


Rerunning our test produces the waterfall below:




Here we can see the connection on line 5 is set up in advance, before we download the CSS. This leads to over a second of improvement (downloading the fonts at 4 seconds, rather 5.25 seconds) as we do not pay that connection set up penalty, but instead can download the fonts as soon as we've read the Google Fonts CSS.

You might think you could take it to the next stage and `preload` the whole font, rather than settle for just `preconnect`-ing to the domain, but Google Fonts creates unique hashes for their font names. In above example the font downloaded is `S6uyw4BMUTPHjx4wXg.woff2` rather than `lato.woff2`, so preloading is not possible, unless you want to leave your site open to breakage if they ever change that hash. Anyway, if you are using Google Fonts and do nothing else after reading this post, at least add that `preconnect` hint if it's not there already - it's one line of code and should improve the performance of your page.


In fact Google Fonts actually returns that `preconnect` hint as a `link` HTTP Header when it returns the CSS as shown below:




gfonts.html
/experiments/gfonts




css?family=Lato&display=swap
fonts.googleapis.com




common.css
/experiments/gfonts



wrench-icon-48-202352.png
/assets/images/wrench-icon



data:image/svg+xml;...



data:image/svg+xml;...

▼ Response Headers

access-control-allow-origin: *

alt-svc: quic=":443"; ma=2592000; v="46,43",h3-Q050=":443"; ma=2

cache-control: private, max-age=86400, stale-while-revalidate=60

content-encoding: br

content-type: text/css; charset=utf-8

date: Mon, 13 Jan 2020 20:26:35 GMT

expires: Mon, 13 Jan 2020 20:26:35 GMT

last-modified: Mon, 13 Jan 2020 20:26:35 GMT

link: <https://fonts.gstatic.com>; rel=preconnect; crossorigin

In a lot of cases, this won't really help as by this point the browser now knows you want to connect to this domain to download the fonts, so you are still better specifying this in your HTML to get the `preconnect` started earlier (it doesn't matter that it's in both and the second hint will just be ignored). However if your page is still processing by the time this comes in, and the DOM is not ready (a sign of too much JavaScript on you page maybe?), then this can help improve the performance when it finally does figure out which of those fonts it needs.

Font Display Swap

In the above `font-face` code you can see a `font-display: swap;` line. This is a relatively new instruction you can add to your `font-face` declaration which tells the browser to use the fallback, system font (`sans-serif` in this example) initially and then swap-in the real font once it has been downloaded. This means the content is not delayed waiting on the font, and so is seen as a good performance improvement. This does lead to a flash of unstyled text (FOUT), which can be jarring and some don't like (personally I'm on the fence - I agree the content is more important than the styling but the jolt as the font steps in is often jarring and while mitigating it is possible by tweaking the fallback font, it is tricky). The alternative is the flash of invisible text (FOIT) where the text is hidden until the font comes in, which obviously delays the load and can cause other problems [if some text loads and other text doesn't!](#)

Anyway, prior to its introduction, different browsers handled this differently - some like IE and Edge used FOUT, others used FOIT, and they had various timeouts as to when giving up waiting on the fonts. This could leave your content invisible for a long time if the font failed to download. The introduction of `font-display: swap` put that choice in the website owners control. It also has [wide browser support](#) except for IE and Edge but, as per above, they use this by default anyway. Google fonts also [supports different font-display options](#), and suggests `font-display: swap` by default.

So, another tip, if you are using Google fonts is to check you have that `&display=swap` parameter to your loading URL, and if not (as it's only recently become supported), then add it:

For example, change from this:

```
<link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>

<link href="https://fonts.googleapis.com/css?family=Lato"
rel="stylesheet">
```

to this:

```
<link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>

<link href="https://fonts.googleapis.com/css?family=Lato&display=swap"
rel="stylesheet">
```


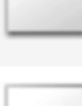









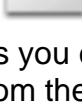
You can also specify [one of the other font-display values](#), like [optional if you prefer](#). Unfortunately, this only solves half the problem. This instruction is in the CSS that Google Fonts returns, so it is only useful *after* you have downloaded the CSS file. So this helps deal with the delay while the fonts themselves are downloading but does not help while you are waiting for that CSS to download. So a good improvement (at least for those who prefer FOUT), but still not the whole solution.

Self-hosting Google Fonts.

I've been [helping to create the Web Almanac](#) (a fantastic look at the state of the web - [check it out](#) if you've not seen it yet), and the slow loading of the Google Fonts on our site annoyed me and was something I wanted to look at. Especially with the jarring `font-display: swap` behaviour. I wanted to see if we could reduce the impact of this, and the natural solution seemed to be self-hosting, possibly with the use of `preload`.

We had already used the above performance improvements (`preload` and `font-display: swap`), but surely it would be better to not have to make that annoying CSS call at all - and then remove the need for the `preconnect` completely? We know what will be in that CSS so surely self-hosting is better? Well that's where it gets interesting... I found a handy script ([Google Font Download](#)) on GitHub to help me download all the various font variations (as we had a lot of them - up to 9 depending on the page), and then I copied the CSS it outputted into our main style sheet and added the fonts to our host directory. There are also [online tools that do the same](#). This all seemed to work and we got rid of that annoying CSS download and two domains - self-hosting for the win!

However, on closer inspection I noticed the fonts were bigger:

Name	Size ▼	Size ▼
 Lato_400i.woff2	25.4 KB	14.6 KB
 /static/fonts	23.9 KB	14.5 KB
 Lato_700i.woff2	25.4 KB	14.6 KB
 /static/fonts	23.9 KB	14.5 KB
 Lato_400.woff2	24.4 KB	14.0 KB
 /static/fonts	22.9 KB	13.7 KB
 Lato_700.woff2	24.0 KB	13.9 KB
 /static/fonts	22.5 KB	13.8 KB
 Lato_900.woff2	23.6 KB	13.5 KB
 /static/fonts	22.0 KB	13.4 KB
 Poppins_700.woff2	9.2 KB	7.9 KB
 /static/fonts	7.7 KB	7.8 KB

As you can see there was a hefty size increase (up to 74% extra for some of them!) from the Google Font loaded fonts (on the right) and the locally hosted fonts (on the left). I initially thought this was due to my local development web-server, presumably due to compression, but WOFF2 fonts are served uncompressed by the web server - or at least should be - since the format includes Brotli compression. Also the screenshot above shows the downloaded bytes (in black above) and also the uncompressed bytes (in slightly lighter colour on the bottom) for each column (they are similar enough in both columns as the fonts are being served without further compression by the web server, but the downloaded bytes include HTTP headers so are slightly larger), and there were differences in both compressed and uncompressed bytes, so it wasn't that.

Comparing the `font-face` declarations produced by the tool highlighted one difference:

From Google Fonts:

```
/* latin-ext */
@font-face {
  font-family: 'Lato';
  font-style: normal;
```



```

    font-weight: 400;

    font-display: swap;

    src: local('Lato Regular'), local('Lato-Regular'),
url(https://fonts.gstatic.com/s/lato/v16/S6uyw4BMUTPHjxAwXiWtFCfQ7A.woff2
) format('woff2');

    unicode-range: U+0100-024F, U+0259, U+1E00-1EFF, U+2020, U+20A0-20AB,
U+20AD-20CF, U+2113, U+2C60-2C7F, U+A720-A7FF;
}

/* latin */

@font-face {

    font-family: 'Lato';

    font-style: normal;

    font-weight: 400;

    font-display: swap;

    src: local('Lato Regular'), local('Lato-Regular'),
url(https://fonts.gstatic.com/s/lato/v16/S6uyw4BMUTPHjx4wXiWtFCc.woff2)
format('woff2');

    unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02BB-02BC, U+02C6,
U+02DA, U+02DC, U+2000-206F, U+2074, U+20AC, U+2122, U+2191, U+2193,
U+2212, U+2215, U+FEFF, U+FFFD;
}

```

From the download tool:

```

@font-face {

    font-family: 'Lato';

    font-style: normal;

    font-weight: 400;

    src:

        local('Lato Regular'),

        local('Lato-Regular'),

        /* from
https://fonts.gstatic.com/s/lato/v16/S6uyw4BMUTPHjx4wXg.woff2 */

        url('Lato_400.woff2') format('woff2'),

```

```
/* from
https://fonts.gstatic.com/s/lato/v16/S6uyw4BMUTPHjx4wWA.woff */

url('Lato_400.woff') format('woff');
}
```

The first difference is we've lost our `font-display: swap` line, so we can add that back in easily enough, but more interestingly is the fact that Google Fonts is serving two fonts - and including a different `unicode-range`, in them. This is due to font-subsetting and it reduces font files.

Font Subsetting

Font subsetting involves removing the characters you are not going to use to reduce the size of the font file. Typically most western users will just use the Latin characters, and downloading a font with all the characters you probably won't use is wasteful. I'd heard of it before but had never realised the dramatic impact it can have! Google Fonts will automatically provide a `font-face` with subset fonts for Latin language and will also, where available, provide a second font for the missing extended Latin characters (e.g. Å) which will only be downloaded if needed.

In fact you can take it a step further and also ask for a special font containing only the characters you want with the `text` parameter:

```
https://fonts.googleapis.com/css?family=Lato&text=ABC
```

Upon further reading, the font download tool I used apparently does support font subsetting, but only at a whole "language" (Latin or Latin-ext) and it merged both subsetted fonts into one file. So I ended up reverting to using what my browser was using, and stopped using this tool. This gave downloads of a similar size (only slightly difference being due to slightly different HTTP headers on my development environment), but I was soon to discover that it was not just subsetting that made the difference.

Google Font is clever about how it serves the fonts

Google Font does *not* serve the same CSS every time but instead [basis it on the user-agent provided](#). So for Internet Explorer 11 it provides the following:

```
@font-face {

  font-family: 'Lato';

  font-style: normal;
```

```
font-weight: 400;

font-display: swap;

src: local('Lato Regular'), local('Lato-Regular'),
url(https://fonts.gstatic.com/s/lato/v16/S6uyw4BMUTPHjx4wWA.woff)
format('woff');
}
```

Here you can see it is only providing the WOFF format, as IE 11 [does not support WOFF2](#), and it is not providing the `unicode-range` for the same reason. It does supply `font-display: swap` (as I specified that in the URL for the CSS) despite it not supporting that, but it does no harm.

It's not just browser make and version either. [Font hinting](#) involves extra instructions in the font file which are then used to ensure the font is displayed the best - especially on low resolution screens or for really small sizes. Font hinting is used by Windows, but not MacOS so depending which you use to get your Google Fonts if using the browser (even if using Chrome on each platform), you'll get font files with hints or without hints. If you then download the Windows version and serve those locally, you'll actually make your MacOS users suffer with larger font files full of hints that will not be used, and if you do the opposite you'll make Windows users potentially suffer with worse fonts as they will get no hints, when previously they did.

When I use Google Fonts, and also when I downloaded the fonts locally, I was doing it on my Mac, so got the smaller, un-hinted fonts. When I used the tool I got the full, hinted fonts. So this was another reason for the large size difference!

[Whether hinting is still relevant](#) as high definition screens have become more common is a good question, and many fonts do not come with hints, because they are so time consuming to create. When they are present they can dramatically increase the size of the fonts - Lato is doubled in size with hints. Whether they are worth that extra load is another decision you'll need to take if moving off Google Fonts to self-hosting.

So the Google Fonts CDN is backed by a clever script to serve the most appropriate fonts, and optimise for performance. By moving to self-hosted fonts you are taking on the responsibility to set this all up correctly, and may even be losing support for your fonts in some browsers if you don't.

For the Web Almanac we looked at the browsers that visited us and made the decision to only support WOFF2, and also use the MacOS versions without hinting since they were half the size. This made the CSS simpler (especially with the fact that [unicode-range support](#) is basically supported by all [browsers that support WOFF2](#)). IE 11 users, and other older browsers, default back to `sans-serif` which does not look quite as nice, but we are seeing the fonts as a progressive enhancement as the site is still more than useable without them. Additionally older browsers, on older machines may even benefit from using default, system fonts as they are more likely to be on older, under-powered machines.

You can also [make further adjustments to the fonts](#) if you so please (check the font licences first though!) and have more advanced knowledge, but copying the Google Fonts defaults used by Chrome on MacOS is probably sufficient for the most of us if you're willing to only support WOFF2 browsers and not have hinting. If you want the hinting, the fonts returned for Chrome on Windows is probably your best option, and then you can add other formats like WOFF if you want.

Still if we'd stuck with Google Fonts, then we'd have got WOFF (and even older formats) and font hinting or not as appropriate, without even having to think about it, and without any hassle of checking browser support to help decide which versions to serve locally, or thinking about font-subsetting or anything like that. So there are definite benefits to using the Google Fonts site to serve the fonts, and you are giving these up by self-hosting. This includes benefiting from any [future enhancements to Google Fonts](#).

Future Font Improvements

Since I was delving deep into fonts, and since I warned that you might lose any future benefits from Google Fonts by self-hosting, I'm going to take a slight side-step into other things I've discovered while working on this. What are the big upcoming changes in the font world? Well, there are two changes which are getting a lot of buzz at the moment and which may impact fonts in future (and therefore may be supported by Google Fonts, perhaps by default, if you stayed with them): *variable fonts* and *progressive font enrichment*.

Variable Fonts

[Variable fonts](#), allow different styles of a font to be used without having to download separate fonts. I mentioned earlier that the Web Almanac uses up to 9 different font files, but it is really only for 2 fonts. The reasons we have 9 font files, is we also have a bold, italic, bold-italic, black (basically super bold!) and even light versions of one or both of them. That might seem ridiculous to have so many variations of the same font (it surprised me!) and you'd think the browser could make fonts a little thicker for bold, or slate them for italics. Well they can, but each browser does it slightly differently, and the results can be [completely wrong](#), hence why many people call them "faux fonts".

The only way to ensure you display what you want to, is to use a "real font" specifically for each weight and style you need. Variable fonts however [standardises how to apply variations of a font](#), so you no longer need to download completely separate font files. This means, in theory, we could replace those 9 fonts, with just 2 when variable font versions of them become available.

This opens up a load of possibilities in font usage on the web. While I might think it is ridiculous to have 9 variations of two fonts, this is actually quite limited whereas variable fonts allows endless variations. On mobile you might want a slightly different thickness for "bold" than on desktop and on tablet, you might want somewhere in between - variable fonts allows that option, with some [simple CSS instructions](#), without the expense of another font. [Jason Pamental](#) gave a [fantastic talk on variable fonts at](#)

[DotCSS recently](#) and has a [demo page](#) showing how you can have a beautifully styled page with apparently many different fonts - all from one font file!

Using variable fonts also means that all variations of the font can load at the same time [avoiding confusion caused by font loading issues mentioned previously](#). This also results in less layout repaints: without variable fonts the page will redraw as each individual font variation is downloaded, which can result in many [reflows](#).

[Variable fonts have very good support in modern browsers](#), but the downside to variable fonts is that the font files are bigger to accommodate for the extra instructions (exactly like hints do too). It depends on the font, but typically they are up to twice as big after compression so you'd want to be making use of at least two styles to make them worthwhile. Even then you might be better prioritising one, smaller file being downloaded to display your critical text, rather than one larger file to display all your text. Then again with `font-display: swap`, maybe that's less of an issue? That's for each site owner to decide!

Progressive Font Enrichment

[Progressive font enrichment](#), basically takes the subsetting to the next level and allows downloading of additional character definitions as they are needed as a stream of extra information which supplements the currently downloaded font, rather than adding an additional one. This may seem like small gains for a lot of effort for us westerners, but for other languages - particularly in the far east - font files can be *massive* (like [2Mb massive](#)), given all the characters in those languages. For this reason web fonts are not used as much on those countries and progressive font enrichment may open web fonts up to sites in those languages.

Progressive font enrichment seems to be at a much earlier stage than variable fonts from my understanding, but there is [an online demo available](#). However they are another potentially exciting change coming to fonts.

Will Google Fonts Continue to be Enhanced?

Because of the way Google Fonts works, you can imagine it could drop in enhancements like these (or many others!) just by changing the CSS it returns to you. And it could be clever about when to do this. Obviously only when your browser supports it (like it does now with the font format - WOFF or WOFF2), or for other ways. For example if you ask more than two fonts, and it is cheaper to return the variable font, then it could do that automatically and reference the same font file in multiple `font-face` declarations, but if you request only one variant then reference the smaller, traditional font. Sound far-fetched? They [already did this with one font \(Oswald\)](#)! I'll be honest and admit I don't know if the same is possible with progressive font enrichment as I don't fully understand how they work, but it will be interesting to see.

This is not to mention that when fonts are updated, perhaps to add new character sets or correct errors in the glyphs, you get the new ones automatically when using Google Fonts directly, but not when self-hosting - at least not without the manual effort of updating them. Perhaps you could look at proxying the requests via your domain for the

best of both worlds, but this would likely still be slower and requires extra setup and management.

On the flip side, self-hosting provides stability as some updates may affect your design, for example if a one line title starts stretching to two lines because of a font change. There are a few instances of people getting *very* upset about this but will save linking to them to hide the rage from you.

Benefits of Self-Hosting

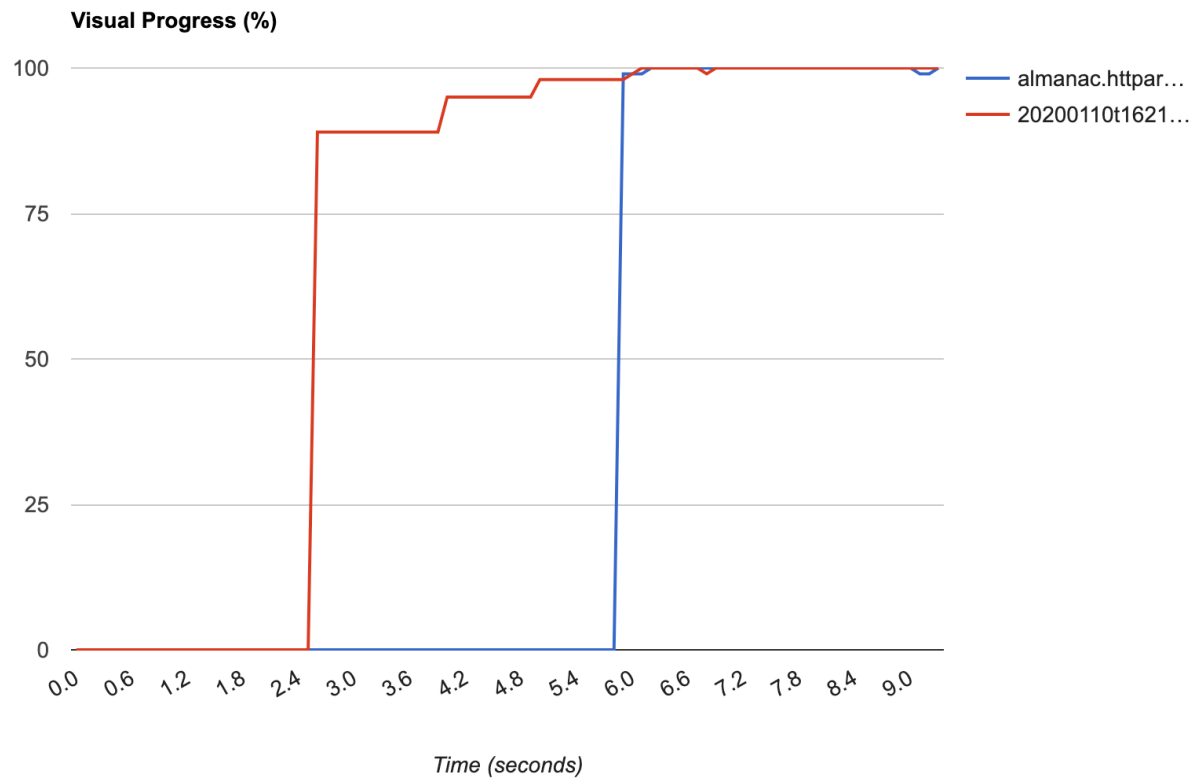
OK so we've a lot of background and theory here, and while there are clear potential performance benefits from self-hosting, there are also complexities to consider and so using the Google Fonts site has some clear benefits too. So is self-hosting still worth it? Well that depends on the actual gains, so you can make that judgement call. Small performance improvements maybe would mean you'd stick with Google Fonts, large improvements would be a different story.

For the Web Almanac we recently [switched to self-hosting our Google Fonts](#), and with testing we saw a *dramatic* change, as can be seen below:



The bottom image is with the locally hosted fonts on our test server and you can see the load time has halved from 6 seconds to 3 seconds! In fact when I drilled down I saw it was even better than half (3.3 seconds saving)!

What is not as apparent (but which we'll see later), unless you've very good eyes, is the fonts have not actually loaded at this point in either image - they happen by 7.5 seconds on locally hosted version and by 10.5 seconds on the Google Fonts hosted version. However the fonts are similar enough to the fall backs that there is only a small reflow in this case so it's less jarring than it is on some sites. This can be seen in the visual progress chart below:



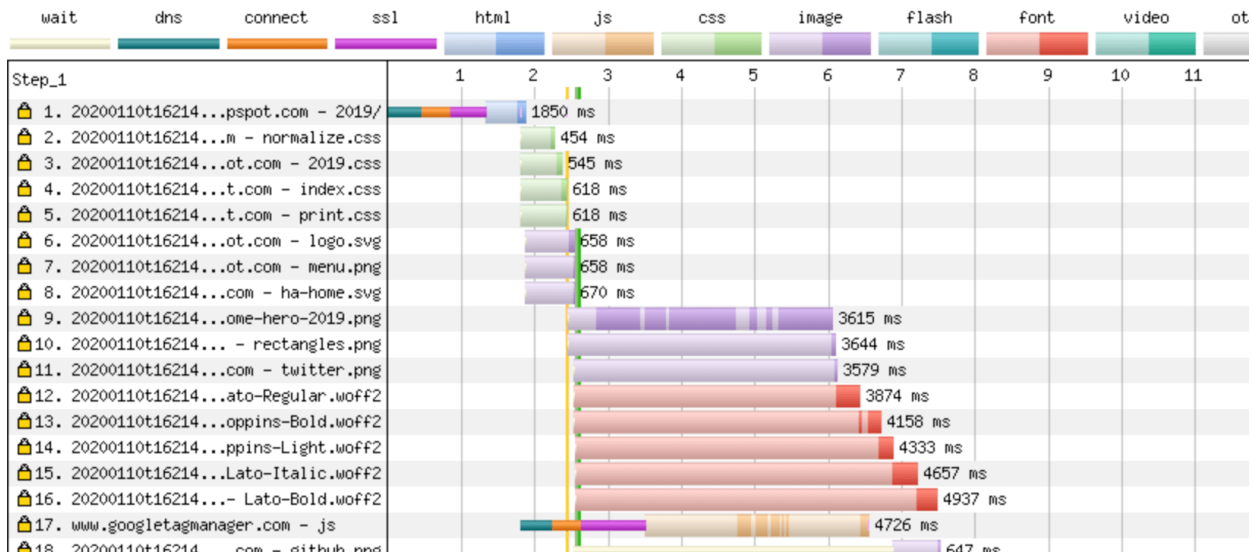
The chart shows the page for the locally hosted version (in red) is almost fully loaded at 2.4 seconds, and then has a few further enhancements as image downloads, and then the fonts drop into play. Yeah I'd still like to avoid that reflow completely, but hey there is still a performance cost to fonts that we can't optimise for!

The reason why the improvements are so dramatic, are due to one more thing I did not consider: CSS is render blocking. So the fact you included a Google Fonts CSS link, doesn't just hold up displaying the text - it holds up displaying *everything!* We know the font is a progressive enhancement and we can safely draw the rest of the page, and even the text itself (with fallback fonts) but the browser doesn't - it just sees that there is some CSS to download so holds up the whole page, while we connect to that domain, and then download it. [There is no async for CSS](#) - but perhaps there should be for cases like this? This also makes the risk of Google Fonts being unavailable, mentioned above, even larger as your whole site will be render blocked in this case until the browser times out!



In the above waterfall, you can see the vertical green Start Render line only happens at 6 seconds - as it has to wait for the Google Fonts CSS to be downloaded on line 12, which spends half the time connecting to the domain, and the other half doing the actual download.

Compare this to the locally hosted version:



Here we are able to start rendering as soon as the site's CSS is downloaded and processed at 2.5 seconds. There is no delay while it connects to the Google Fonts domain to get that CSS.

In both cases we can see the start render happens before the fonts are downloaded, and thanks to the magic of `font-display: swap`, the text is still visible. Therefore, at least when using `font-display: swap`, it would be better if Google Fonts didn't load via

render-blocking CSS, and instead loaded via some async JavaScript which then injects the CSS line into the page only when they've downloaded. If they did this, they'd not have any start render delays, but on the flip side, there would still be the connection delays so a longer time when unstyled text is displayed. Edit: I raised a [suggestion to have a non-render blocking way to load Google Fonts](#) on their GitHub issues list - upvote it if you want that too!

[Zach Leatherman](#) advocates [this approach to reduce reflows](#), and shows it is also possible with just JavaScript and without needing a CSS file. He then shows some other benefits of your fonts being managed by JavaScript, whereby you can choose to remove fonts completely if on a slow network (using [Network Information API](#)) or if the user has [Save-Data](#) or [Prefers-Reduced-Motion](#) settings enabled. Interesting possibilities!

Preloading fonts

Once you self-host, and move away from Google's hashed URLs you have the option of preloading the font for further performance gains. This is one that has some potential downsides however, as discussed by [Andy Davies](#) in his [Preloading Fonts and the Puzzle of Priorities blog post](#). Basically by preloading fonts, and moving them up the priority queue, you're implicitly deprioritising other critical loads (e.g. CSS), and due to a [bug in Chrome, the fonts may even jump ahead of some of those](#).

Additionally preloading when the font won't be used will cause a wasteful download - for example if a local version exists and can be used in preference, as described in our `font-face` declaration above (though I didn't go into this!), or if a browser doesn't support that font format (though all browsers that support `preload` also support WOFF2). While they do recommend `preload`, Google [explicitly warns you to consider this](#), and it is good to see them calling this out.

With `font-display: swap`, the need for `preload` may not be as high, though you may still benefit from preloading a small number of fonts to avoid FOUT and redraws. At the moment we haven't turned `preload` on in the Web Almanac as part of the move to locally-hosting our fonts and will need to do some more testing if we want to do this but for now I'm happy with it as is.

Conclusion

To answer the question in the title of this post: yes it's better to self-host as the performance gains are substantial. Of course your mileage may vary as it will depend on your exact site so test, test, test, but I would imagine it would be the better option, from a performance perspective, for most sites.

However, Google Fonts is not just a repository of hundreds of free fonts - it is also a clever delivery mechanism utilising many of the latest web performance techniques to try to deliver the most appropriate fonts, with the minimal effort to the website owner. In moving to self-hosting you ideally want to re-implement as many of those improvements like `font-display: swap`, subsetting, and removal of hinting (at least for some browsers), or you may actually negatively impact performance by having larger fonts files.

There is some complexity involved in this (which Google Fonts handles for you) but with WOFF2-only being a realistic option now, and good support of most techniques with inbuilt fallbacks, this is easier than it used to be. Still it pays to have some web font knowledge when setting this all up (which hopefully this post has helped with!), and you may want to keep an ear out for upcoming changes in the font world as there will undoubtedly be further improvements to come.

Alternatively maybe we're over optimising here, and using the full font, with hints and no subsetting, is sufficient when serving locally? There is a download option on the Google Fonts website in the Family Selected pop up which makes that relatively easy, though weirdly [it does not include WOFF and WOFF2 formats when using this](#), for reasons that I don't understand! However if you are really that interested in web performance (and you should be!), then you should try to do this properly. This post just shows this is a bit more complex than downloading and hosting on your site as I always presumed it was. Wow, that ended up being even longer than I thought it would be! But hopefully you found it as interesting as I did researching all this, and hopefully it gives you some ideas on improving your font loading time. I'd strongly encourage you to follow [Jason](#) and [Zach](#) on Twitter for more font goodness, and also [Andy](#) for general web performance advice.

Want to read more?

More Resources on the Font Loading

- [The Web Almanac chapter on Fonts.](#)
- [Zach's awesome presentation](#) at [2018 performance.now\(\)](#).
- [Jason's talk at 2019 DotCSS](#)
- [Responsive Web Typography](#) by [Jason Pamental](#)
- [Blog](#) by [Zach Leatherman](#)
- [Blog](#) by [Blog](#)

Related Posts on TuneTheWeb.com

- [Helping to Create the Web Almanac](#) - a blog post detailing my involvement in the Web Almanac.
- [Performance](#)

- [HTTP Performance Headers.](#)
- [Performance Measuring Tools.](#)
- [Inlining CSS is Not for Me.](#)

This page was originally created on 12-Jan-2020 and last edited on 23-Feb-2020.

How useful was this page?

- Great post, love the level of detail.

edit: Never mind, you're already discussing this below!

- 1.
- •
- Reply
- •
- Share ›

○

-
-
-



[Christian Engel](#) • 4 months ago

Id like to give some additional informationa and ask a question:

You can go to the google fonts website, select wich chars and weights should be included in your font and the download it directly as a zip archive. No need for third party tools or grabbing the files from your browser.

Google DOES offer alternative ways for loading the fonts. They describe that as well on the google fonts website. There is a section about how to self-host, use the google CDN and/or use a project called "google font loader" which is a library for downloading the fonts in non-blocking manner and even providing helper css classes that get added/removed to the body while fonts are loading or have been loaded.

And here is my question:

why would I remove other formats from the fontface definition and only host the woff format? Browsers will ignore formats which they dont understand or treat inferior so IE11 will not load

woff2 and modern browsers will ONLY load the modern font formats. Leaving some formats out of the css gains you nothing except broken support.

kudos

-
- •
- Reply
- •
- Share ›

○

■

■



Barry Pollard Mod [Christian Engel](#) • 4 months ago

You can go to the google fonts website, select which chars and weights should be included in your font and download it directly as a zip archive. No need for third party tools or grabbing the files from your browser.

Yes you can. Though annoyingly that don't download WOFF and WOFF2 formats and only the TTF formats. Not sure why to be honest.

Google DOES offer alternative ways for loading the fonts. They describe that as well on the google fonts website. There is a section about how to self-host, use the google CDN and/or use a project called "google font loader" which is a library for downloading the fonts in non-blocking manner and even providing helper css classes that get added/removed to the body while fonts are loading or have been loaded.

Can you give me the link where you see this? I see some talk of this in their FAQ (<https://developers.google.c...> but not to the detail I've gone in here nor mention of "google font loader" - which [I understand is no longer supported](#).

why would I remove other formats from the fontface definition and only host the woff format? Browsers will ignore formats which they don't understand or treat inferior so IE11 will not load woff2 and modern browsers will ONLY load the modern font formats. Leaving some formats out of the css gains you nothing except broken support.

It's a bit more complicated than that if you want to combine different settings. For example if I want to use `unicode-range` with the WOFF2 fonts (because all browsers than support WOFF2 support `unicode-range`), but not with WOFF (because those browsers do not support `unicode-range`) then that's more complex, and may not even be possible - I

dunno I didn't spend a lot of time on it to be honest because I was comfortable dropping WOFF support, for the reasons given in the post:

This made the CSS simpler (especially with the fact that unicode-range support is basically supported by all browsers that support WOFF2). IE 11 users, and other older browsers, default back to sans-serif which does not look quite as nice, but we are seeing the fonts as a progressive enhancement as the site is still more than useable without them. Additionally older browsers, on older machines may even benefit from using default, system fonts as they are more likely to be on older, under-powered machines.

Google Fonts avoids this complexity by providing different CSS (with `unicode-range` when returning WOFF2 and without when returning WOFF). You could add this dynamic loading to your own site if it's not possible to write the font-face declarations to support both (as I say I didn't give it a lot of effort) but that's quite complicated to do too.

Plus it was a lot less effort to download just the WOFF2 versions too and I'm lazy- only joking! Somewhat... 😊

- [Reply](#)
- [Share](#) ›



[thetomester13](#) • [4 months ago](#)

Very cool and in depth writing about self hosting fonts! I currently use Google Fonts and have been hoping to transition off of it soon and self host. Really glad I came across this write up as I didn't realize the intricacies that Google goes through for hosting fonts. Definitely some food for thought here. Thanks!

- 1_
- ●
- Reply
- ●
- Share ›

- ○
○



[mrangry](#) • 4 months ago

Google get themselves involved in politics, so I 'self host' them so I don't have to deal with the situation that their servers are not available for my users. I wonder if Microsoft or Amazon has a similar service, since they aren't so 'woke'.

- .3
- •
- Reply
- •
- Share ›

○

-
-
-



[Jonathan Weckerle](#) • 5 months ago

The google stylesheets contain the path to the reduced font files, which of course means you can just copy the path in the browser bar and then download the file. So what I just did: 1. Get woff and woff2-files plus CSS from <https://google-webfonts-helper.herokuapp.com/fonts/roboto/> 2. Go to google fonts and select the same fonts 3. open the link to the stylesheet in the browser 4. Download the woff2-Version of one font 5. rename the downloaded filename to the filename of the according file from google-webfonts-helper and then replace the file from google-webfonts-helper with the renamed file from Google 6. copy the unicode-range part from the Google stylesheet to the styles I got from google-webfonts-helper.

Short version: Just download the optimized (unicode-reduced) files from Google, so you can host these yourself. It takes a few minutes of stupid copy & pasting, but works fine.

- 4.
- •
- Reply
- •
- Share ›

○

-
-
-



■
■
■
■

●
●
●
●

Reply

Share ›



Does Google Fonts not provide an alternative way of loading their web fonts that doesn't result in a delay of rendering?

- 2_
- •
- Reply
- •
- Share ›





Barry Pollard Mod Sime Vidas • 5 months ago

I'm not aware that they do. I've raised this issue for them to tell me differently or considering an enhancement to add this: <https://github.com/google/f...>

- 1_
- •
- Reply
- •
- Share ›