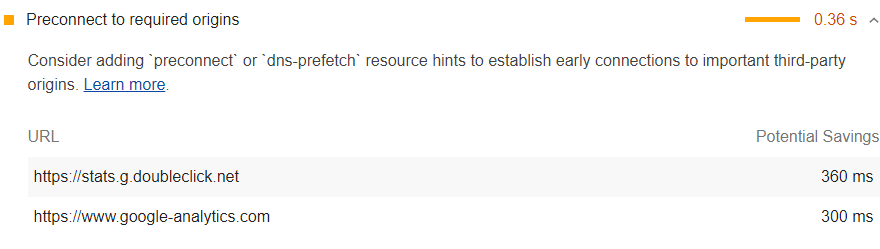
**Preconnect to required origins**

May 2, 2019 • Updated May 6, 2020

Appears in: [Performance audits](https://web.dev/lighthouse-performance)

The Opportunities section of your Lighthouse report lists all key requests that aren't yet prioritizing fetch requests with <link rel=preconnect>:



**Browser compatibility** [**#**](https://web.dev/uses-rel-preconnect/?utm_source=lighthouse&utm_medium=devtools#browser-compatibility)

<link rel=preconnect> is supported on most browsers. See [Browser compatibility](https://developer.mozilla.org/docs/Web/HTML/Link_types/preconnect#Browser_compatibility).

**Improve page load speed with preconnect** [**#**](https://web.dev/uses-rel-preconnect/?utm_source=lighthouse&utm_medium=devtools#improve-page-load-speed-with-preconnect)

Consider adding preconnect or dns-prefetch resource hints to establish early connections to important third-party origins.

<link rel="preconnect"> informs the browser that your page intends to establish a connection to another origin, and that you'd like the process to start as soon as possible.

Establishing connections often involves significant time in slow networks, particularly when it comes to secure connections, as it may involve DNS lookups, redirects, and several round trips to the final server that handles the user's request.

Taking care of all this ahead of time can make your application feel much snappier to the user without negatively affecting the use of bandwidth. Most of the time in establishing a connection is spent waiting, rather than exchanging data.

Informing the browser of your intention is as simple as adding a link tag to your page:

<link rel="preconnect" href="https://example.com">

This lets the browser know that the page intends to connect to example.com and retrieve content from there.

Bear in mind that while <link rel="preconnect"> is pretty cheap, it can still take up valuable CPU time, particularly on secure connections. This is especially bad if the connection isn't used within 10 seconds, as the browser closes it, wasting all of that early connection work.

In general, try to use <link rel="preload">, as it's a more comprehensive performance tweak, but do keep <link rel="preconnect"> in your toolbelt for the edge cases like:

* [Use-case: Knowing Where From, but not What You're Fetching](https://developers.google.com/web/fundamentals/performance/resource-prioritization#use-case_knowing_where_from_but_not_what_youre_fetching)
* [Use-case: Streaming Media](https://developers.google.com/web/fundamentals/performance/resource-prioritization#use-case_knowing_where_from_but_not_what_youre_fetching)

<link rel="dns-prefetch"> is another <link> type related to connections. This handles the DNS lookup only, but it's got wider browser support, so it may serve as a nice fallback. You use it the exact same way:

<link rel="dns-prefetch" href="https://example.com">.

# increase performance

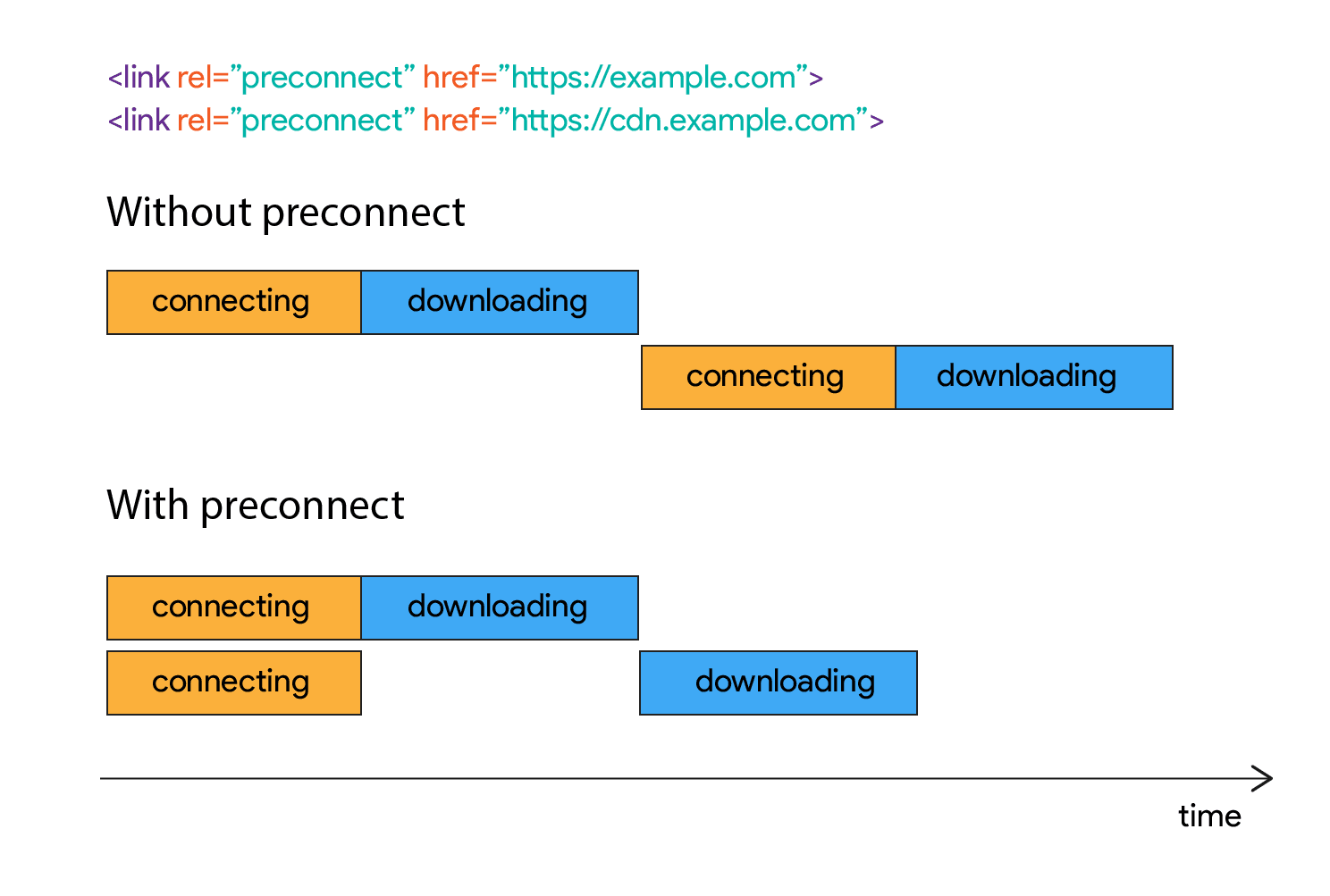
[](https://css-tricks.com/author/chriscoyier/)

[Chris Coyier](https://css-tricks.com/author/chriscoyier/) on Aug 19, 2019

[Milica Mihajlija](https://web.dev/preconnect-and-dns-prefetch/):

Adding rel=preconnect to a <link> informs the browser that your page intends to establish a connection to another domain, and that you’d like the process to start as soon as possible. Resources will load more quickly because the setup process has already been completed by the time the browser requests them.

The graphic in the post does a good job of making this an obviously good choice for performance:



Robin did [a good job of rounding up information](https://css-tricks.com/prefetching-preloading-prebrowsing/) on all this type of stuff a few years back. Looks like the best practice right now is using these two:

<link rel="preconnect" href="http://example.com">

<link rel="dns-prefetch" href="http://example.com">

For all domains that aren’t the main domain you’re loading the document from.

A quick look at CSS-Tricks resources right now, I get:

secure.gravatar.com

static.codepen.io

res.cloudinary.com

ad.doubleclick.com

s3.buysellads.com

srv.buysellads.com

www.google-analytics.com

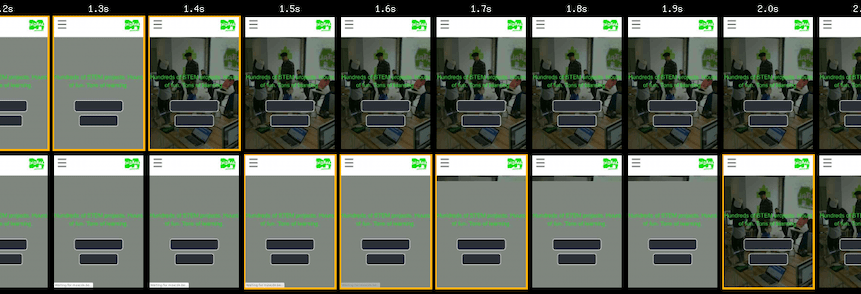
That’d be 14 extra <link> tags in the first few packets of data on every request on this site. It sounds like a perf win, but I’d want to test that before no-brainer chucking it in there.

[Andy Davies did some recent experimentation](https://andydavies.me/blog/2019/08/07/experimenting-with-link-rel-equals-preconnect-using-custom-script-injection-in-webpagetest/):

So what difference can preconnect make?

I used the HTTP Archive to find a couple of sites that use Cloudinary for their images, and tested them unchanged, and then with the preconnect script injected. Each test consisted of nine runs, using Chrome emulating a mobile device, and the Cable network profile.

There’s a noticeable visual improvement in the [first site](https://www.digitaladventures.com/), with the main background image loading over half a second sooner (top) than on the unchanged site (bottom).



This stuff makes me think of [instant.page](https://instant.page) (which [just went v2](https://instant.page/2)), which is a fancy little script that preloads things based on interactions. It’s now a browser extension ([FasterChrome](https://chrome.google.com/webstore/detail/fasterchrome/nmgpnfccjfjhdenioncabecepjcmdnjg)) that I’ve been trying out. I can’t say I notice a huge difference, but I’m almost always on fast internet connections.