Web Performance Calendar

The speed geek's favorite time of year
2025 Edition

Fixing the URL params performance penalty

23rdDec 2025 by Barry Pollard
ABOUT THE AUTHOR

Barry Pollard (@tunetheweb@webperf.social) is a Web Performance Developer Advocate in the Google Chrome team, working on Core Web Vitals and the Chrome User Experience Report (CrUX). He is one of the maintainers of the HTTP Archive and the author of HTTP/2 in Action from Manning Publications.

Tales of two pages…

What's the difference between these two pages?:

- `https://www.example.com/`
- `https://www.example.com/?utm_source=email`

I mean they've got different URLs, but many of us would probably guess that that `utm_source` URL query parameters (or "URL params" or "search params" as it's sometimes known) is not meaningful to the page contents and is more used for analytics tracking. So these pages more than likely have the same content.

What about this two pages?:

- `https://www.example.com/products?productid=1234`
- `https://www.example.com/products?productid=5678`

Here, I think it's safe to assume the page is going to contain details of different products.

So it's not always obvious when query params result in different content, and when they don't.

In fact it's even more complicated than that, because the initial HTML for the above two pages might in fact be the exact same! For client-side rendered (CSR) applications, very little unique content is often included in the initial HTML and JavaScript is then used to fetch and display the different content.

So why's this a problem?

URL params are a cache killer

The main problem is that, because you can't know if the contents of a URL may change based on the URL params, by default caches have to assume the content will be different. And this really matters because caching is one of the most important things you can do to improve web performance. So much so that various authors have written about caching on this web performance calendar too many times to count!

And you might think "you shouldn't need to visit what is basically the same page twice", so what's the issue? Well here's a few scenarios:

Scenario 1

- You visit a home page from a link with a site with a tracking UTM param added (for example `https://www.example.com/?utm_source=email`). This could be from an email, an ad, a social network… whatever! The owner of that medium has decided to add a UTM param (or params!) to allow them to measure the effectiveness of their various sources/campaigns.
- You browse around the site, visiting various pages.
- At some point you click a link to go back to the home page of the site (maybe that one that's usually in the top left with the icon of the site?).
- You've already been to the home page. But with the `utm_source=email` param so the browser thinks you've not seen THAT version of the page. So you have to wait for a pointless load of the page ☹ Not to mention filling up your hard disk with multiple copies of that page.
- You cry a little inside.

Scenario 2

- You run a little blog that gets 1-2 visitors today.
- It's served from a little under-powered server cause you're not an enterprise business with money to throw around needlessly.
- But you're clever and have put a free/cheap CDN in front of it just in case you get DoSed.
- One day you write something that's actually really, really interesting.
- It blows up and is spread around many places: HackerNews, all the social media sites, lots of newsletters. Well done you!
- But each source has different utm params, which bypasses your CDN since that version of your site is not cached and hammers your little server.
- Your site goes down.
- You cry a little inside.

Scenario 3

- You run an enterprise eCommerce platform.
- This time you've bigger servers and a CDN configured to handle this better (more on this later).
- It's Black Friday or Cyber Monday.
- Traffic is WILD!!! Even higher than expected! From all sorts of places!
- They have millions of tracking and affiliate UTM params you've never even heard of that for some reason are not stripped when it's sent on to your server.
- This bypasses your CDN since that version of your site is not cached and hammers your origin servers at a scale never seen before.
- Your site goes down.
- You cry a little inside.

Even if nothing as major as a site going down happens, the sheer waste of network bandwidth and compute from needlessly refetching, and reserving the same content over and over again is staggering to think about!

Many, many, many URL params serve the same content

There are many, many, many client-side tracking parameters services in content. From UTM params from analytics, to ad tracking params. These are read by JavaScript loaded on the page and used to log and measure whatever needs to be measured.

But in terms of the page content, they do not matter! The same content is served each time, and it's only when the analytics/ad/remarketing JavaScript is run, that it reads it from the URL.

It's such a common pattern. Here's a quick list of some of the most common I can think of off the top of my head:

- `utm_source`
- `utm_medium`
- `utm_campaign`
- `utm_term`
- `utm_content`
- `gclid`
- `fbclid`
- `msclkid`
- Actually no, I'm bored of this. Trust me — there are hundreds, if not thousands!

It's impossible to keep track of (pun 100% intended!).

And new ones are added all the time, sometimes causing visible performance regressions.

But it's not all hopeless.

CDN's have long realised this is an issue

And many CDNs will allow you to configure it to ignore query params and serve the same content from their edge cache, rather than hitting your origin server.

But… this is often not done by default, requires configuring and support varies by CDN. You can either ignore all query params (simple, but likely wrong if your app needs them at all), commonly used ones (basically that list I gave previously), or more complex rules. It's all kinda custom and not standardised.

Additionally this is only part of the solution. Users will still needlessly hit the CDN to re-fetch what is effectively the same resource. While it's good that it will hopefully be refetched quickly from the CDN edge cache, it's still slower than ideal and fills up browser caches with pointless duplicated entries.

And let's not get into the complexity of multiple CDNs, proxies…etc.

A standard solution

There has to be a better way than these custom, non-standard solutions? Well, some clever people at Google* thought the same and decided to tackle this.

This is not actually a new problem, and it's very similar to another problem.

Servers often serve different content to different browsers based on those browsers capabilities. One of the most common examples of this is image CDNs, where you can request `https://images.example.com/hero.png?auto=format` and it will return one of the newer formats (WebP, AVIF, JXL) based on the browser support advertised in the `Accept` HTTP request header. Because the URL does not specify the format (well it specifies the original file is in png format, but not the format that will actually be delivered based on that file) this causes its own caching problems—but in reverse to the problem we've talked about: content may be incorrectly cached. To avoid that, servers can include a `Vary` HTTP response header telling caches that the content can "vary" depending on specific HTTP request headers (`Accept` in this case). Caches can then store multiple copies and serve the appropriate one, ensuring each browser gets the format they need.

As I say, what we're looking for is similar to this, but instead of saying something *varies*, we want to say it *doesn't vary*. And instead of that being decided on by HTTP request headers, we want it to be based upon URL params (aka search params). So a new `No-Vary-Search` HTTP response header was created to let you tell HTTP caches what is, and isn't important in query params.

> **Fun fact:** This actually initially came out of Chrome's Speculation Rules work where we wanted to allow speculations to "match" despite seemingly different URLs that were actually effectively the same, as described in the initial explainer. But it was too good an improvement to keep to just that use case, so it was generalised and moved to the IETF for HTTP standardisation.

How does `No-Vary-Search` work then?

Like the `Vary` HTTP response header, servers can respond with a `No-Vary-Search` HTTP response header with various details.

Key-order

You can say the key order doesn't matter:

```
No-Vary-Search: key-order
```

This means `https://www.example.com/?param1=1&param2=5` and `https://www.example.com/?param2=5&param1=1` are treated as having the same content.

Note the values obviously still have to match, and also if other URL params exist that are different then those would also be treated as different URLs.

In my experience it's very, very, very rare (and usually a bad practice) for key order to matter. So you probably wanna include this one directive anyway!

Params

You can say all params don't matter (in which case the `key-order` is basically implied since we're effectively ignoring params):

```
No-Vary-Search: params
```

This means all these URLs are treated as having the same content since, when you remove the params you end up with the same URL:

- `https://www.example.com`
- `https://www.example.com/?utm_source=email`
- `https://www.example.com/?utm_campaign=winter2025`
- `https://www.example.com/?utm_source=email&utm_campaign=winter2025`
- `https://www.example.com/?utm_campaign=winter2025&utm_source=email`
- `https://www.example.com/?barry=pollard`

The rest of the URL needs to match (for example `https://www.example.com/page?utm_source=email` obviously wouldn't match as that's for a different page!)

You can also specify only a set of params to ignore:

```
No-Vary-Search: params=("utm_campaign" "utm_source")
```

Which would have the same effect as all the previous examples, except the last one with `barry=pollard` would now be considered a separate page.

Note this is just an example and you'd almost certainly want to include all five `utm` params, and others and not just these two.

As an HTTP structured header these are a list of quote, space-separated items, so be careful you don't forget the quotes, or accidentally sneak some commas in there if that's your usual list separator in your programming language of choice!

Except params

Alternatively to a potentially large inclusion list of params that aren't important, you can flip it around and say all params except a specific exclusion set can be ignored:

```
No-Vary-Search: params, except=("q" "productid")
```

This means all these URLs are treated as having the same content:

- `https://www.example.com/?productid=1234`
- `https://www.example.com/?productid=1234&utm_source=email`
- `https://www.example.com/?utm_source=email&productid=1234`
- `https://www.example.com/?productid=1234&othertracker=email`
- `https://www.example.com/?utm_source=email&productid=1234&utm_campaign=winter2025`

But anything with a different `productid`, or `q` param would be different.

Where does `No-Vary-Search` work?

Well that's the rub. This is super new and is not supported well yet. I've asked a few friends working at CDNs and they say there is interest in supporting it, but no public statements yet.

However, a big change is that Chrome supports this now (from 141 on desktop, and coming in Chrome 144 on Android). This will also roll out to other Chromium-based browsers.

No word yet from other browsers and position requests are somewhat mixed up with its original speculation rules use case, but fingers crossed. Firefox has a bug to follow along.

How can you use it?

While support may not be that widespread just yet, Chrome is a big one, and there's no downside to including the HTTP header in preparation for this coming along to other browsers or CDNs later.

Adding something like this to your server config should be very safe for most sites to deploy (unless you're one of those sites depending on `key-order` and you're not, are you?):

```
No-Vary-Search: key-order, params=("utm_source"
"utm_medium" "utm_campaign" "utm_term" "utm_content" "gclid" "fbclid" "msclkid" "srsltid")
```

You can expand that to include any other regularly used tracking params or other URL params that don't matter.

This immediately solves scenario 1 above, with scenario 2 and 3 coming along for free if (when!) CDNs support this later.

Alternatively, with a bit more risk or effort you can instead change it to just URL params you know you support. For example if you've a simple blog with no URL params except a search box that takes a `q` param then you can do this:

```
No-Vary-Search: params, except=("q")
```

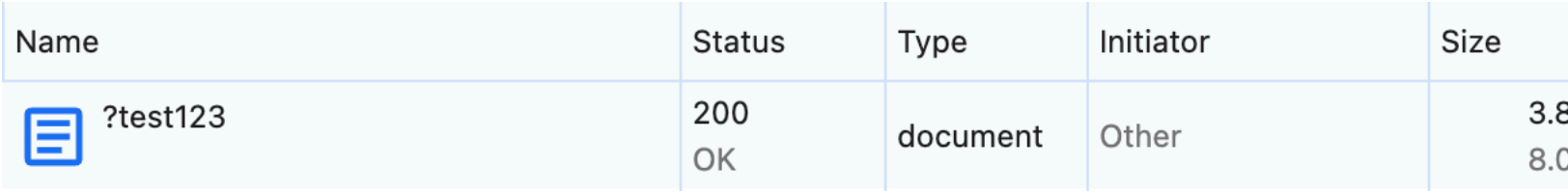
Just make sure you're really, really sure you know all the URL params used on your site if using this one, or you could cause some unintended cache reuse! Maybe check your logs to be doubly sure?

It's also very easily tested:

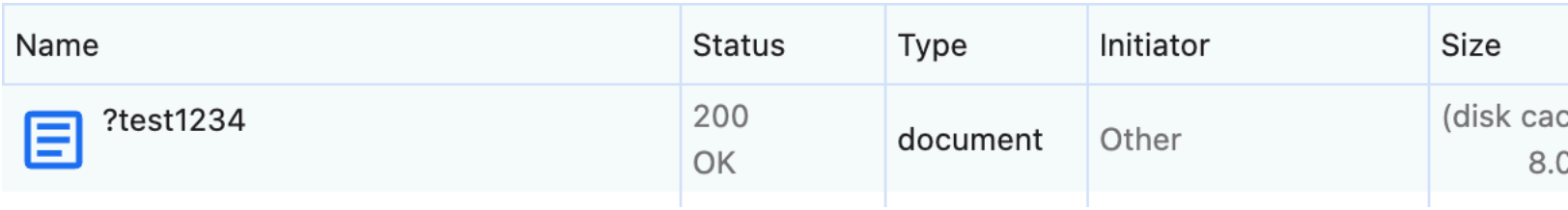
- Set the HTTP Header on your server.
- Make sure "Disable cache" in DevTools is NOT ticked.

- Load your home page with a random URL parameter you're setting to ignore.
- Clear your cache (right click the reload button with DevTools and choose "Empty cache and hard reload").
- Change your random URL param to your home page URL in the address bar and press enter to load the page.
- Confirm in the Network panel that your page is served from the "Disk Cache".

Without `No-Vary-Search`, you get a full page load for each random URL param added:



With `No-Vary-Search`, you get the disk cache being used even when you've never loaded the page with that URL param:



Performance happiness!

Conclusion

I'm very excited, **vary** excited even (boom! boom!) for `No-Vary-Search` and think it has potential to save huge amounts of needless network downloads and origin server work. I really hope it gets broad adoption and hope that platforms learn to set it by default for common URL params like those listed earlier in this article.

CDNs could benefit their customers hugely from standardising this to reduce their CDN work. Alternatively if this header is not being received from origin servers, but CDNs are using their own rules for certain URL params, then setting this header would allow supporting browsers' HTTP cache to benefit from that logic too.

Let me know what you think or if you have any questions (particularly if you work for a CDN or a platform!).

> Shout out to Liviu Tinta, Jeremy Roman, Domenic Denicola, Adam Rice, Lingqi Chi, Hiroki Nakagawa, and Kouhei Ueno to name just a few of those involved in this!