Let’s Talk About HTTP Caching

May 10, 2018

Starting with a question:

What’s the best kind of request you can make? (performance wise)

“The best (fastest) request is one that doesn’t need to be made at all”

Today’s Topics

1. Comparing cache headers ⚽
   a. etag / last-modified
   b. cache-control / Expires

2. Cache-Busting with Webpack 💥
   a. \`hash\` vs. \`chunkhash\`
   b. CommonChunks → \`webpack manifest\`

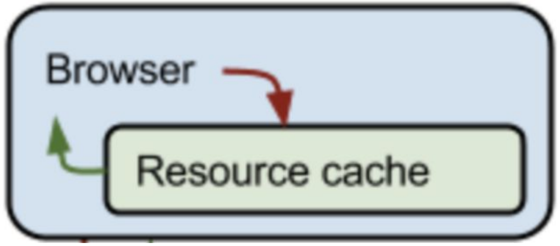
3. Simple Server Setup for Static Resource Caching
   a. HapiJs

What is an HTTP cache?

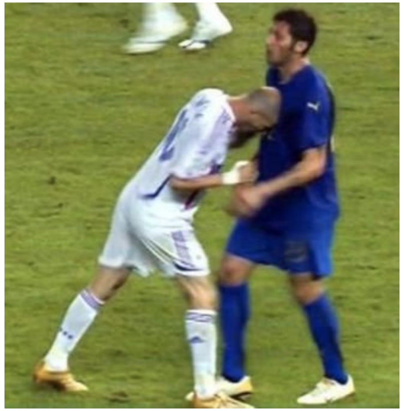
Temporary storage location for resources accessed via http by a client

Contains anything that was ever cached by a site you've visited

- Static files: JS, CSS, HTML, Images
- Responses from data-producing APIs



Check out the chrome http cache by navigating to **chrome://cache**



Comparing Http Cache Headers

Two caching methods

Validation Tokens

Requires a round trip to the server to validate that a resource hasn’t changed.

- 304: not modified
- 200: resource has changed

Headers: etag & last-modified

Specifying Expiration

Can forego round trip to server on subsequent requests!

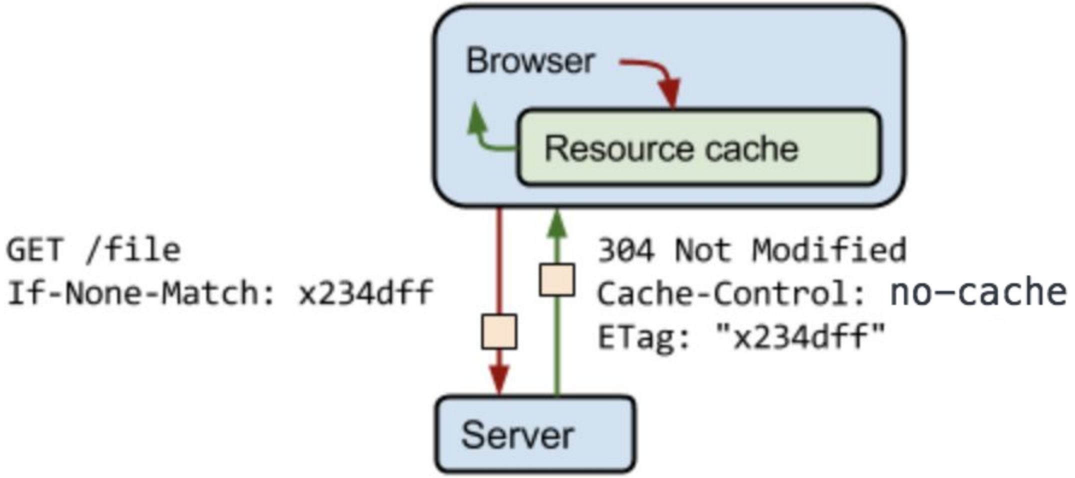
- Cache can become stale...

Headers: cache-control & expires

etag & last-modified

- Used as a **Validation Token** to determine server-side if resource has changed
- If etag or last-modified are present in response headers, data is cached
- Token is communicated via `If-None-Match` or `If-Modified-Since` request headers

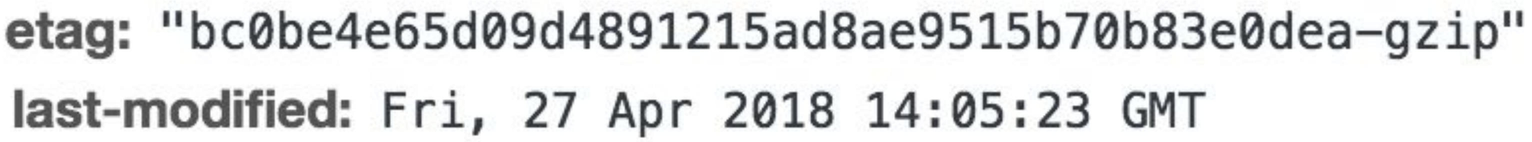
CAUTION: http request necessary on every page load



```
GET /file
If-None-Match: x234dff
```

```
304 Not Modified
Cache-Control: no-cache
ETag: "x234dff"
```

Image: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching



```
etag: "bc0be4e65d09d4891215ad8ae9515b70b83e0dea-gzip"
last-modified: Fri, 27 Apr 2018 14:05:23 GMT
```

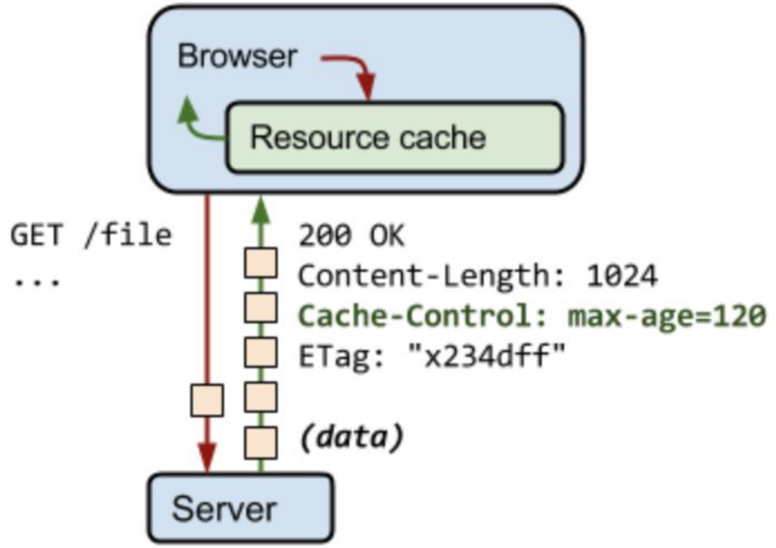
	etag	**last-modified**
Token format?	Unique **hash** created from resource contents	Server's estimate of the **Date** the resource was last updated
expires?	When new **hash** is issued	**Timestamp** updated when file changes
Expected header in request	**If-none-match**: hash_value	**If-modified-since**: date_from_server (provided on initial request of file)

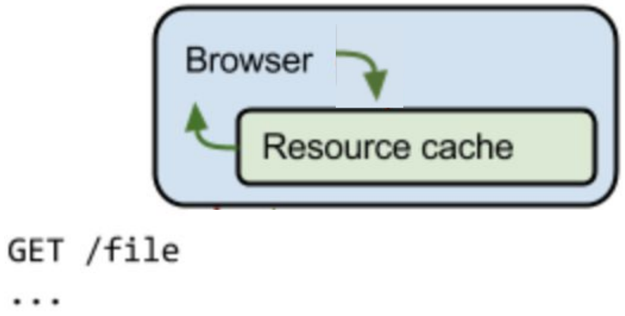
etag is the preferred approach, with last-modified used as fallback

cache-control

Controls **who can cache** the response, under **what conditions**, and for **how long**

Configured correctly, it can **prevent round trips to the server!**





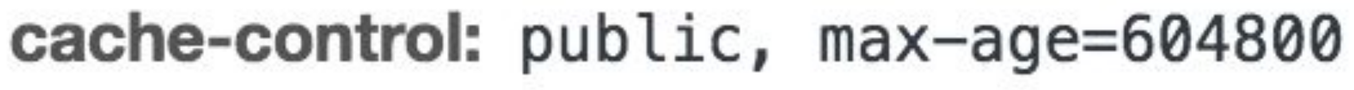
No trip to server required!

Images: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching

cache-control: no-cache

	no-cache	**no-store**
Cacheability	**Yes**, Resource is cached	**No**, Resource is never cached
Round trip to server required?	**Yes**, to confirm via validation token that resource has not changed	**Yes**, to re-fetch resource
re-download required?	***Only*** if validation token indicates that resource has changed	**Yes**, every time

no-cache should always be used with an etag/last-modified caching strategy, to ensure validation tokens are always checked



```
cache-control: public, max-age=604800
```

	public	**private**
Cacheability	**Yes**, Resource is cached	**Yes**, Resource is cached
Where can it be cached?	Can be cached in intermediate cache (**CDN**) as well as client	**Client ONLY**

max-age: duration (in seconds) the resource is cached before it expires and is re-requested

The best request is one that you don't need to make… max-age makes this possible!

Expires

Header looks like this `Expires: Wed, 21 Oct 2015 07:28:00 GMT`

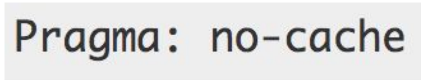
Same behavior as `cache-control: max-age`

If max-age is defined, **Expires is ignored**

Don’t use Expires. Use cache-control: max-age

Expires Info: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expires

Bonus: Pragma

It looks like this 

```
Pragma: no-cache
```

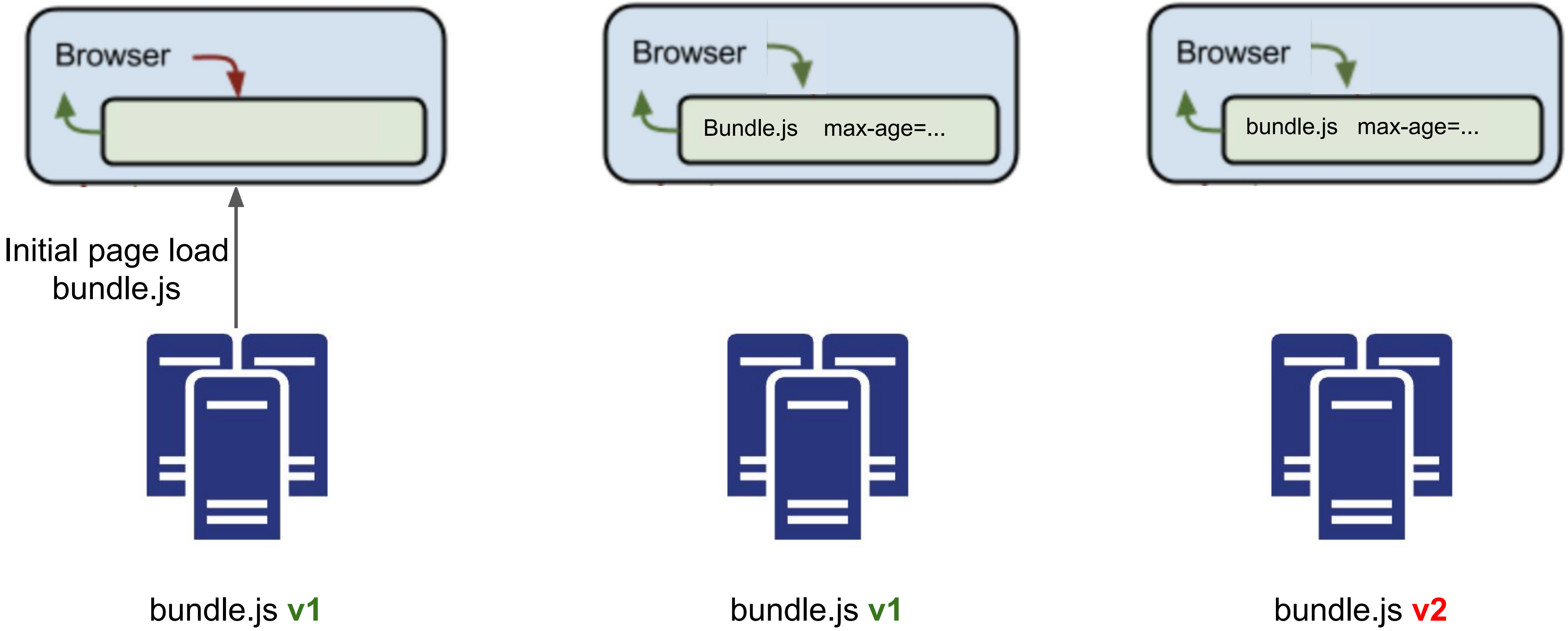
`no-cache` is it’s only valid value

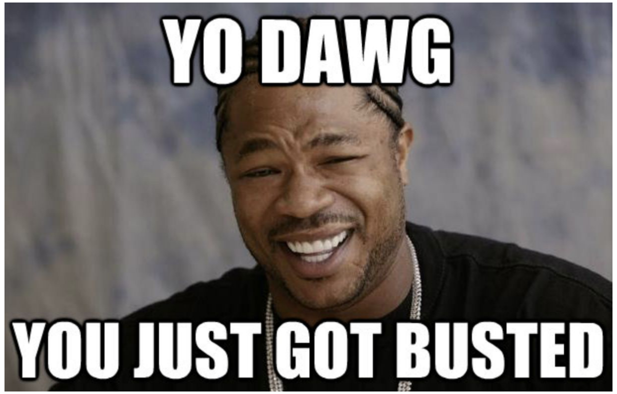
It provides the same behavior as cache-control: no-cache for Http/1.0 clients

Only use Pragma if **backwards compatibility** with Http/1.0 clients is necessary.
Otherwise use **cache-control: no-cache**

Pragma Info: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Pragma

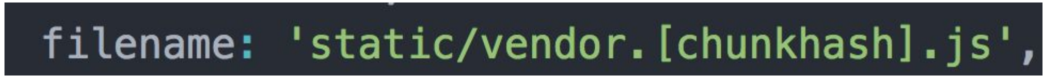
But if no subsequent request is made…
… won't my cached files become *stale*?





Cache Busting with Webpack

What the hash?



```
filename: 'static/vendor.[chunkhash].js',
```

Webpack can be configured to add a **hash to the name of your build outputs**

Two flavors of hashes:

- \`hash\` - assigned **per-build** (changes each re-build)
- \`chunkhash\` - assigned **per-chunk** based on file content + last-modified

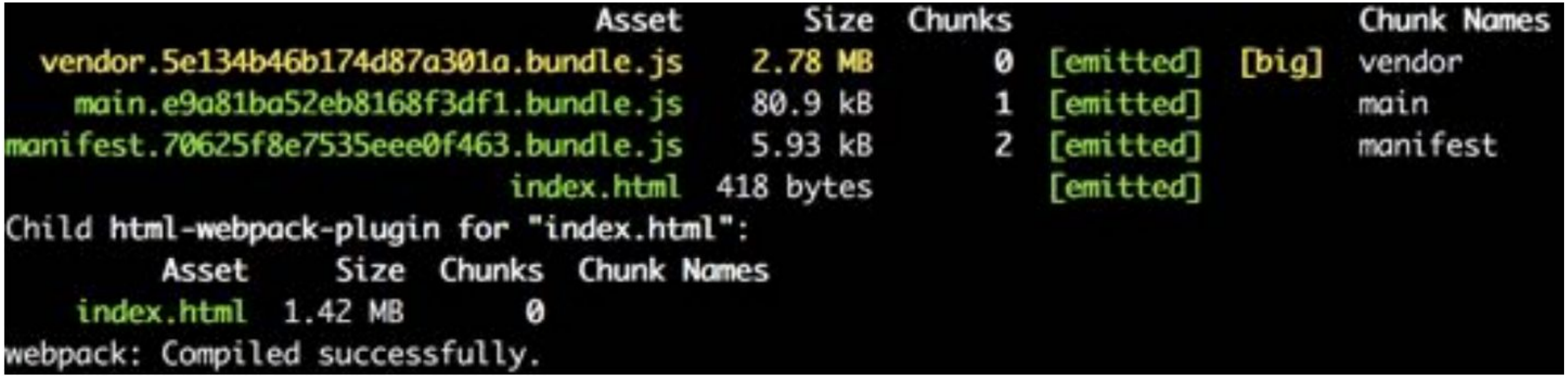
Quiz: which of the above hashing strategies is preferable?

Prefer **chunkhash** - only changes when necessary

Common Hashing issue - Webpack Manifest

`manifest` contains webpack runtime code + module mapping

Use **CommonChunksPlugin** to separate manifest code from main bundles.



```
                              Asset       Size  Chunks                    Chunk Names
  vendor.5e134b46b174d87a301a.bundle.js    2.78 MB       0  [emitted]  [big]  vendor
    main.e9a81ba52eb8168f3df1.bundle.js    80.9 kB       1  [emitted]         main
manifest.70625f8e7535eee0f463.bundle.js    5.93 kB       2  [emitted]         manifest
                             index.html  418 bytes          [emitted]
Child html-webpack-plugin for "index.html":
         Asset     Size  Chunks  Chunk Names
    index.html  1.42 MB       0
webpack: Compiled successfully.
```

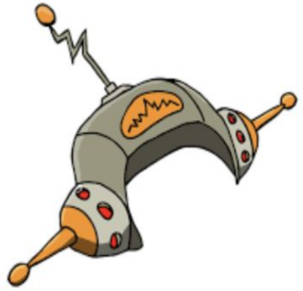
How does hashing affect caching?

- A chunkhash only changes if chunk contents are updated... file **fingerprint**
  - Kind of acts like an etag!

- If you're using webpack chunkhash:
  1. You don't need to send etag headers back in your responses
  2. **You can http cache your webpack outputs FOREVER**

Server Config for Caching Static Assets

HapiJS - inert plugin



Plugin for serving static files from a directory

Gives you the following for free:

- etag / last-modified + cache-control: no-cache headers
- Pre-compressed file lookup (gzip by default lookup by `.gz` extension)

Using Inert gives you **validation token caching** out of the box with no config ✓

Simple configuration
can lead to big performance wins!

Talk is cheap. Show me the code.

— *Linus Torvalds* —

AZ QUOTES

Anything Else To Consider?

Optimize based on <u>data</u> not on <u>hunches</u>

- WebPageTest, NewRelic metrics
- Inspect Waterfalls in the browser

Simple steps can have large impacts

- Configuring http caching correctly + gzipping

Questions?

Additional Resources

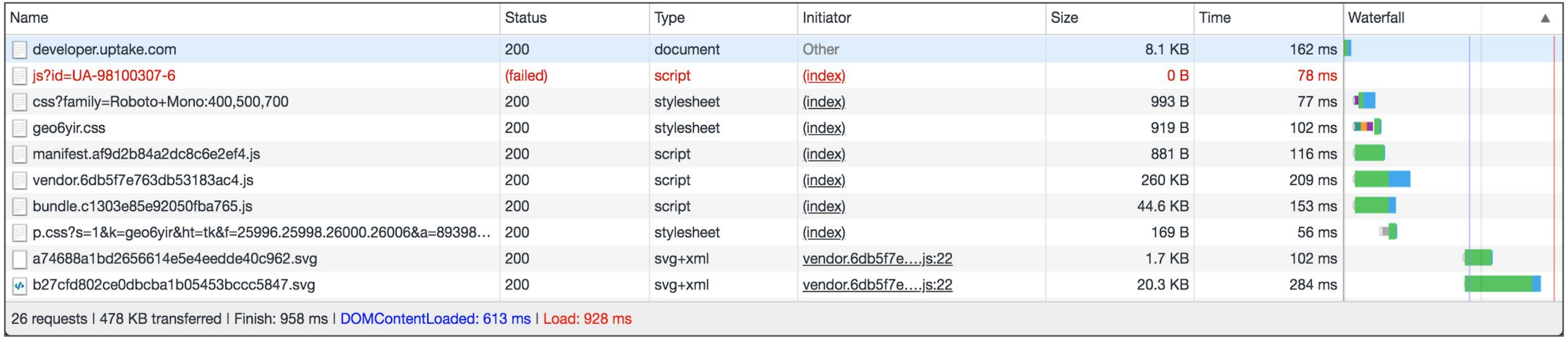
Ilya Grigorik from Google on HTTP Caching
https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching

^This is the best resource that I've read to explain HTTP caching

Appendix

Initial Page Load

Stats: **928ms** page load time ; **478Kb** transferred



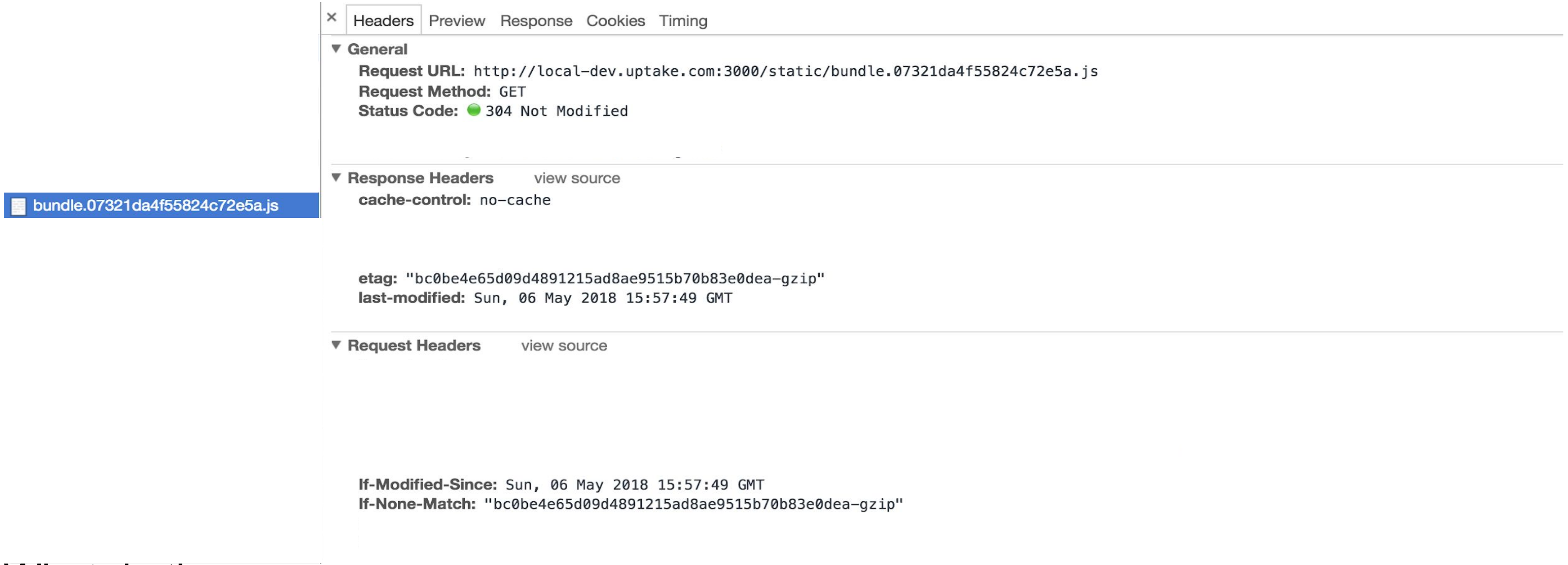
Name	Status	Type	Initiator	Size	Time	Waterfall
developer.uptake.com	200	document	Other	8.1 KB	162 ms	
js?id=UA-98100307-6	(failed)	script	(index)	0 B	78 ms	
css?family=Roboto+Mono:400,500,700	200	stylesheet	(index)	993 B	77 ms	
geo6yir.css	200	stylesheet	(index)	919 B	102 ms	
manifest.af9d2b84a2dc8c6e2ef4.js	200	script	(index)	881 B	116 ms	
vendor.6db5f7e763db53183ac4.js	200	script	(index)	260 KB	209 ms	
bundle.c1303e85e92050fba765.js	200	script	(index)	44.6 KB	153 ms	
p.css?s=1&k=geo6yir&ht=tk&f=25996.25998.26000.26006&a=89398…	200	stylesheet	(index)	169 B	56 ms	
a74688a1bd2656614e5e4eedde40c962.svg	200	svg+xml	vendor.6db5f7e….js:22	1.7 KB	102 ms	
b27cfd802ce0dbcba1b05453bccc5847.svg	200	svg+xml	vendor.6db5f7e….js:22	20.3 KB	284 ms	

26 requests | 478 KB transferred | Finish: 958 ms | DOMContentLoaded: 613 ms | Load: 928 ms

DomContentLoaded === HTML document has been parsed, js and CSS resources downloaded

Load === DomContentLoaded + all images and fonts loaded

Attempt 1: `cache-control: no-cache` + `etag`



bundle.07321da4f55824c72e5a.js

▼ General

Request URL: `http://local-dev.uptake.com:3000/static/bundle.07321da4f55824c72e5a.js`
Request Method: GET
Status Code: ● 304 Not Modified

▼ Response Headers   view source

cache-control: no-cache

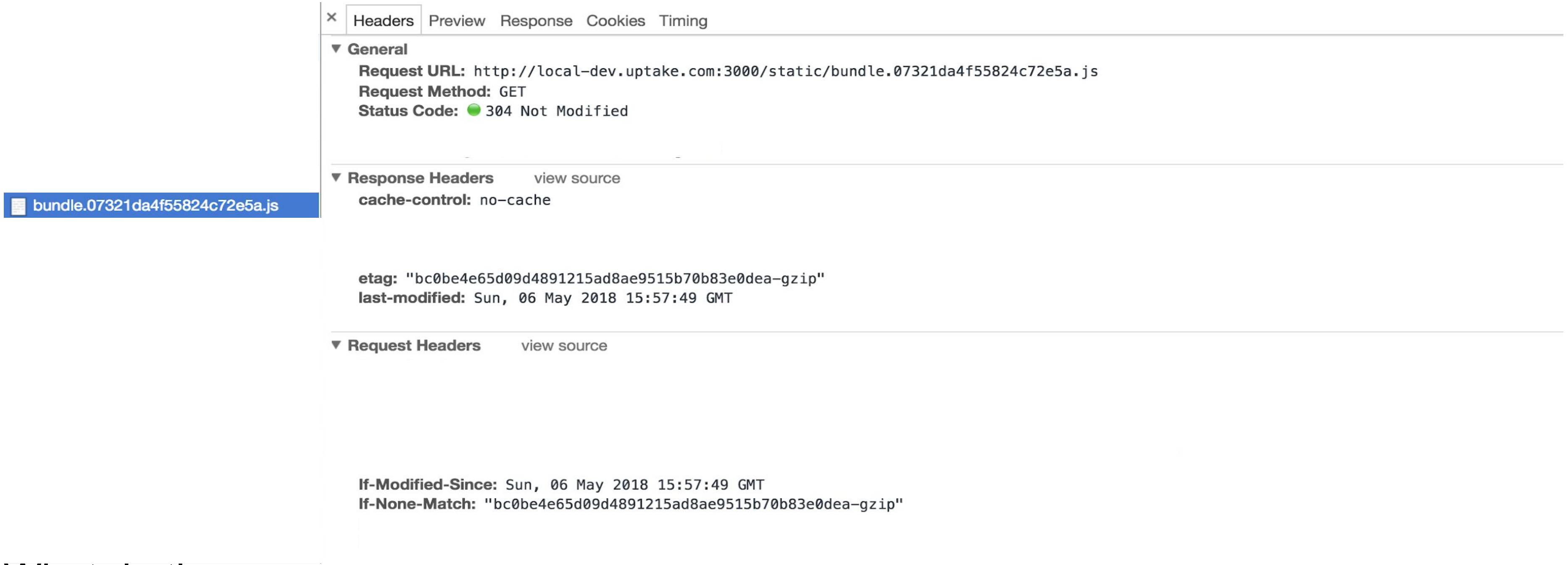
etag: "bc0be4e65d09d4891215ad8ae9515b70b83e0dea-gzip"
last-modified: Sun, 06 May 2018 15:57:49 GMT

▼ Request Headers   view source

If-Modified-Since: Sun, 06 May 2018 15:57:49 GMT
If-None-Match: "bc0be4e65d09d4891215ad8ae9515b70b83e0dea-gzip"

What do the above headers indicate?

Attempt 1: `cache-control: no-cache` + `etag`



```
▼ General
   Request URL: http://local-dev.uptake.com:3000/static/bundle.07321da4f55824c72e5a.js
   Request Method: GET
   Status Code: ● 304 Not Modified

▼ Response Headers     view source
   cache-control: no-cache

   etag: "bc0be4e65d09d4891215ad8ae9515b70b83e0dea-gzip"
   last-modified: Sun, 06 May 2018 15:57:49 GMT

▼ Request Headers     view source

   If-Modified-Since: Sun, 06 May 2018 15:57:49 GMT
   If-None-Match: "bc0be4e65d09d4891215ad8ae9515b70b83e0dea-gzip"
```

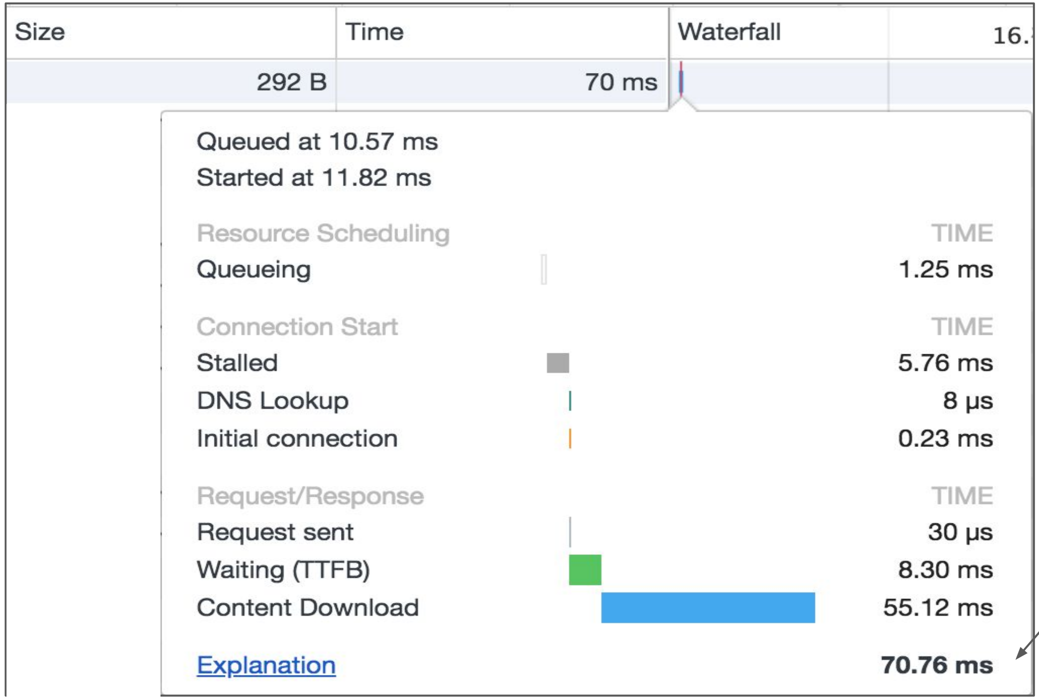
What do the above headers indicate?

cache-control: need to validate every request with issuing server

etag: hash issued by server on initial request, file validated via `If-None-Match`

last-modified: date issued by server on initial request, validated via `If-Modified-Since`

Attempt 1: `cache-control: no-cache` + `etag`



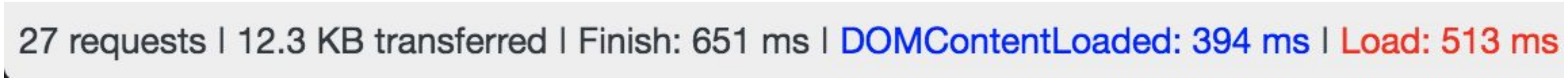
Size	Time	Waterfall	16.
292 B	70 ms		

Queued at 10.57 ms
Started at 11.82 ms

Resource Scheduling	TIME
Queueing	1.25 ms
Connection Start	**TIME**
Stalled	5.76 ms
DNS Lookup	8 µs
Initial connection	0.23 ms
Request/Response	**TIME**
Request sent	30 µs
Waiting (TTFB)	8.30 ms
Content Download	55.12 ms
Explanation	**70.76 ms**

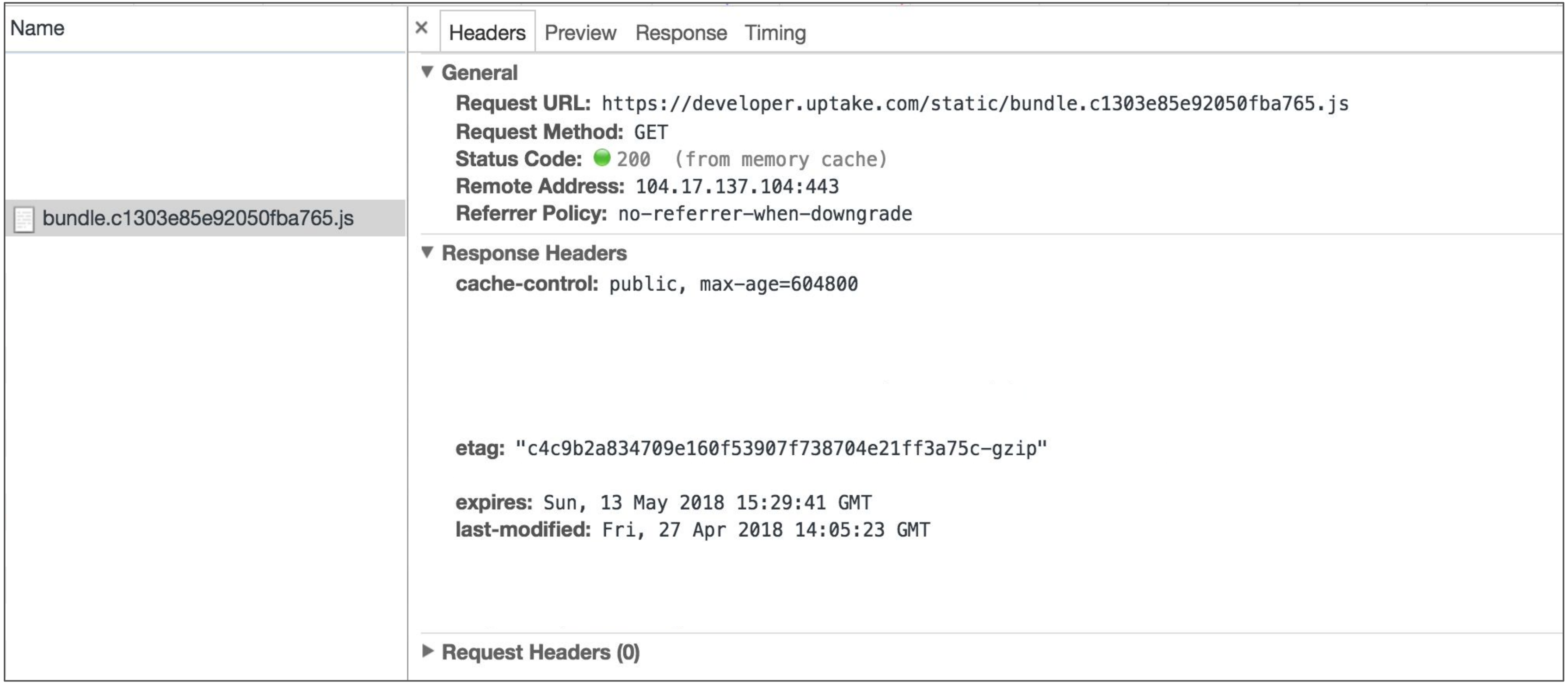
Even though resource was not re-downloaded…

… still took 70.76 ms to revalidate etags!



27 requests | 12.3 KB transferred | Finish: 651 ms | DOMContentLoaded: 394 ms | Load: 513 ms

Attempt 2: `cache-control: public, max-age`



Name

bundle.c1303e85e92050fba765.js

× Headers Preview Response Timing

▼ General

```
Request URL: https://developer.uptake.com/static/bundle.c1303e85e92050fba765.js
Request Method: GET
Status Code: ● 200  (from memory cache)
Remote Address: 104.17.137.104:443
Referrer Policy: no-referrer-when-downgrade
```

▼ Response Headers

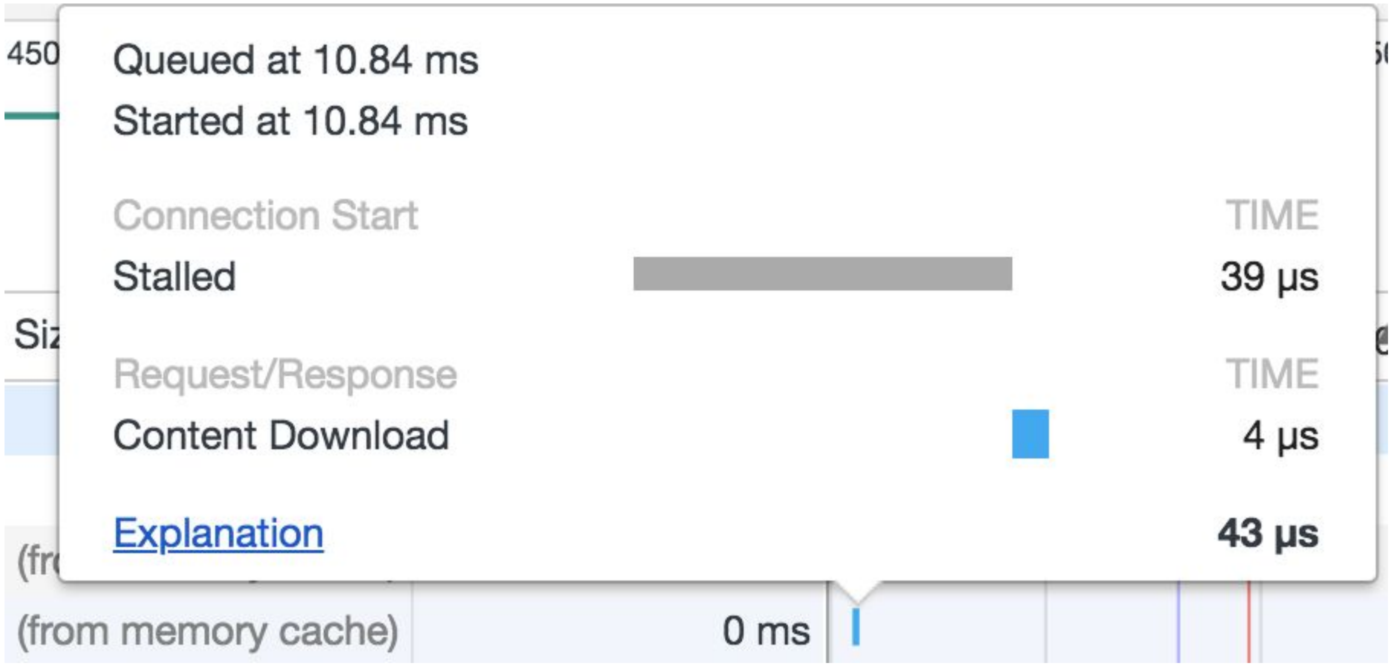
```
cache-control: public, max-age=604800

etag: "c4c9b2a834709e160f53907f738704e21ff3a75c-gzip"

expires: Sun, 13 May 2018 15:29:41 GMT
last-modified: Fri, 27 Apr 2018 14:05:23 GMT
```

▶ Request Headers (0)

Attempt 2: `cache-control: public, max-age`

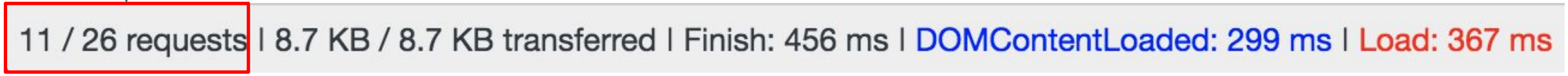


Queued at 10.84 ms
Started at 10.84 ms

Connection Start	TIME
Stalled	39 µs
Request/Response	**TIME**
Content Download	4 µs
Explanation	43 µs

(from memory cache) 0 ms

Only necessary to make 11 of the 26 requests!



11 / 26 requests | 8.7 KB / 8.7 KB transferred | Finish: 456 ms | DOMContentLoaded: 299 ms | Load: 367 ms