A Web Developer’s Guide to Browser Caching

Amir Boroumand
Jul 26, 2017 · 7 min read

Overview

Caching is a useful yet surprisingly complex feature of web browsers.

In this article, we’ll explain the how the browser uses its cache to load pages faster, which factors determine cache duration, and how we can bypass the cache when necessary.

Why is Caching Important?

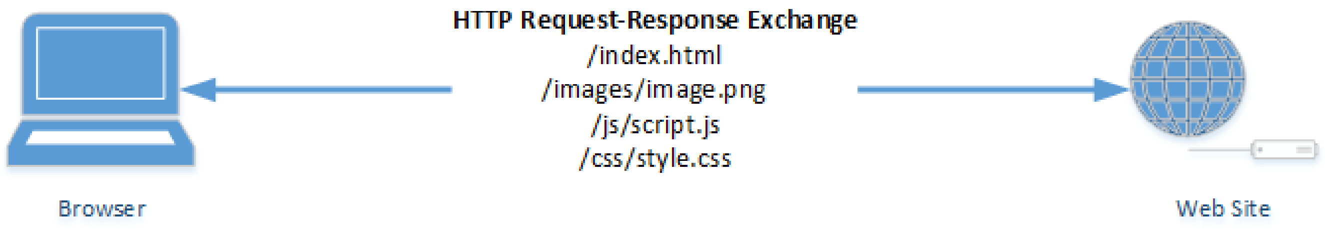
All browsers attempt to keep local copies of static assets in an effort to reduce page load times and minimize network traffic.

Fetching a resource over a network will always be slower than retrieving it from local cache. This is true whether the server is on the same network or it’s located on the far side of the world.

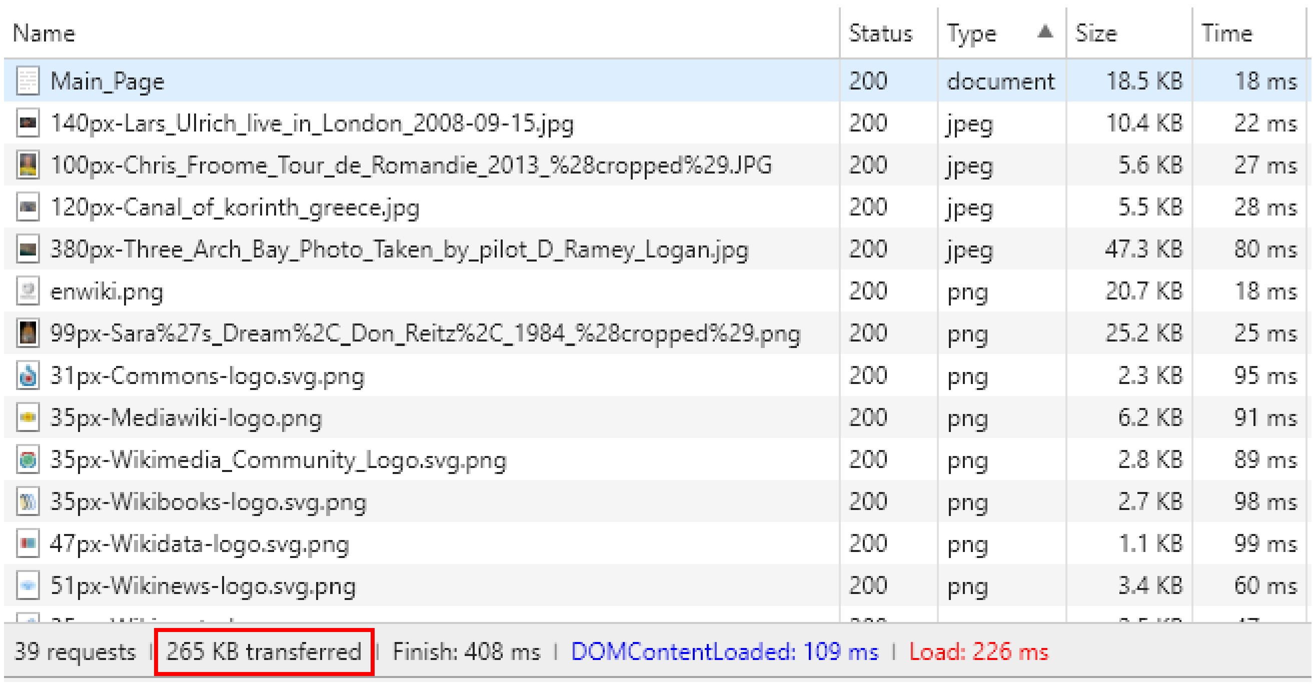
How Browser Caching Works

Case 1: User has not visited the site before

The browser won't have any files cached for the site so it will fetch everything from the server.



Below is a snapshot of the resources downloaded when visiting the Wikipedia home page for the first time. The status bar at the bottom shows that 265KB of data was transferred to the browser.

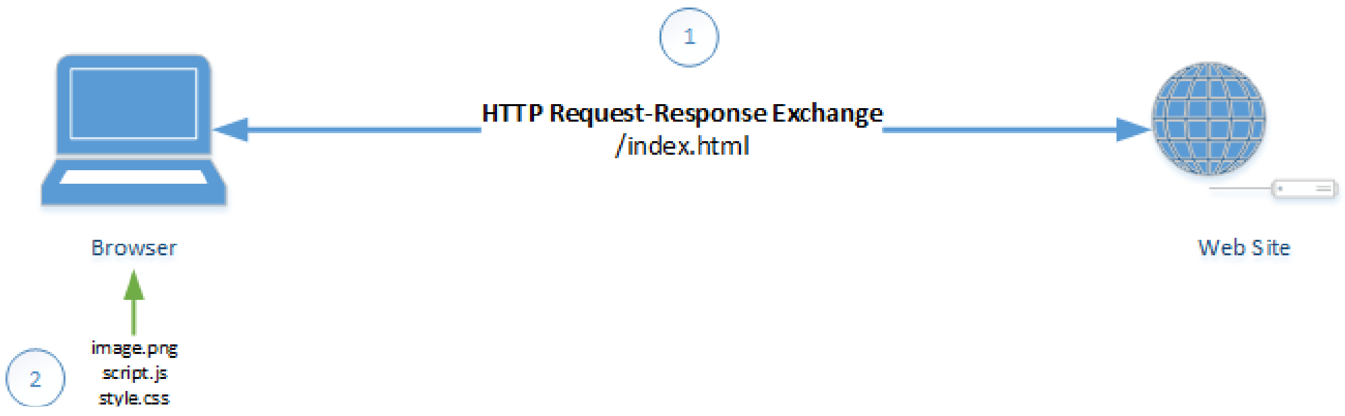


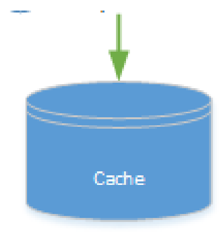
Name	Status	Type	Size	Time
Main_Page	200	document	18.5 KB	18 ms
140px-Lars_Ulrich_live_in_London_2008-09-15.jpg	200	jpeg	10.4 KB	22 ms
100px-Chris_Froome_Tour_de_Romandie_2013_%28cropped%29.JPG	200	jpeg	5.6 KB	27 ms
120px-Canal_of_korinth_greece.jpg	200	jpeg	5.5 KB	28 ms
380px-Three_Arch_Bay_Photo_Taken_by_pilot_D_Ramey_Logan.jpg	200	jpeg	47.3 KB	80 ms
enwiki.png	200	png	20.7 KB	18 ms
99px-Sara%27s_Dream%2C_Don_Reitz%2C_1984_%28cropped%29.png	200	png	25.2 KB	25 ms
31px-Commons-logo.svg.png	200	png	2.3 KB	95 ms
35px-Mediawiki-logo.png	200	png	6.2 KB	91 ms
35px-Wikimedia_Community_Logo.svg.png	200	png	2.8 KB	89 ms
35px-Wikibooks-logo.svg.png	200	png	2.7 KB	98 ms
47px-Wikidata-logo.svg.png	200	png	1.1 KB	99 ms
51px-Wikinews-logo.svg.png	200	png	3.4 KB	60 ms

39 requests | 265 KB transferred | Finish: 408 ms | DOMContentLoaded: 109 ms | Load: 226 ms

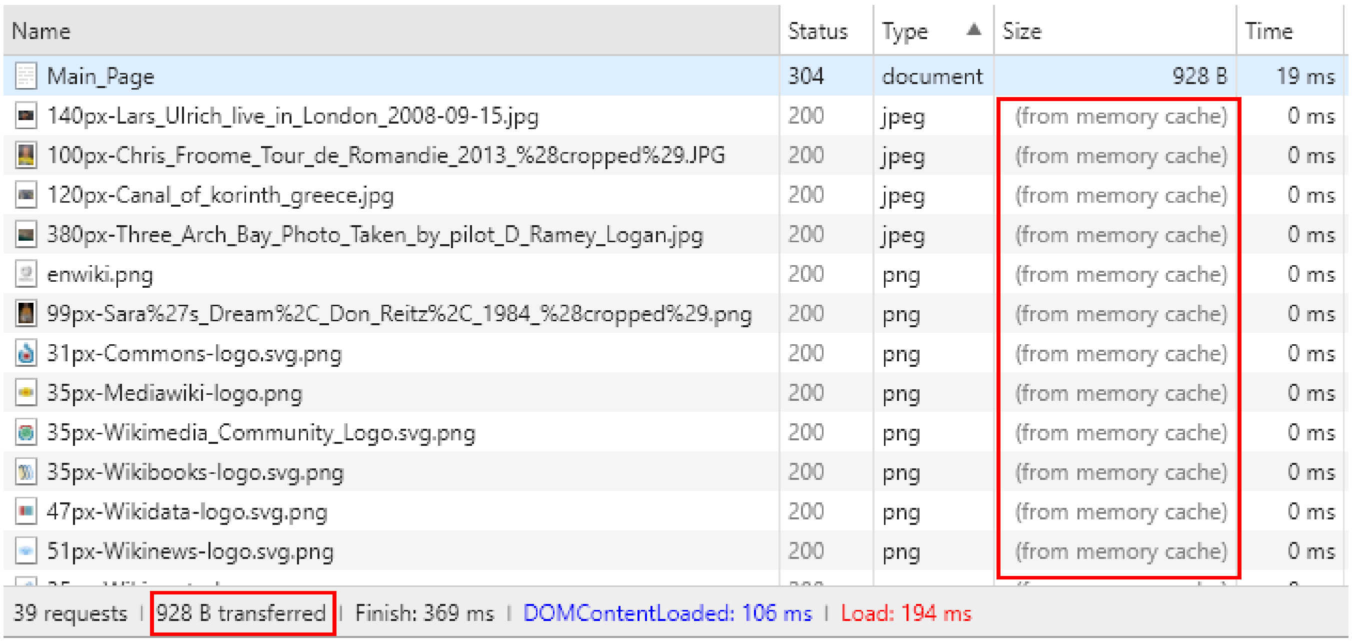
Case 2: User has visited the site before

The browser will retrieve the HTML page from the web server but consult its cache for the static assets (JavaScript, CSS, images).





We can see the difference cache makes when we refresh the Wikipedia page:



Name	Status	Type ▲	Size	Time
Main_Page	304	document	928 B	19 ms
140px-Lars_Ulrich_live_in_London_2008-09-15.jpg	200	jpeg	(from memory cache)	0 ms
100px-Chris_Froome_Tour_de_Romandie_2013_%28cropped%29.JPG	200	jpeg	(from memory cache)	0 ms
120px-Canal_of_korinth_greece.jpg	200	jpeg	(from memory cache)	0 ms
380px-Three_Arch_Bay_Photo_Taken_by_pilot_D_Ramey_Logan.jpg	200	jpeg	(from memory cache)	0 ms
enwiki.png	200	png	(from memory cache)	0 ms
99px-Sara%27s_Dream%2C_Don_Reitz%2C_1984_%28cropped%29.png	200	png	(from memory cache)	0 ms
31px-Commons-logo.svg.png	200	png	(from memory cache)	0 ms
35px-Mediawiki-logo.png	200	png	(from memory cache)	0 ms
35px-Wikimedia_Community_Logo.svg.png	200	png	(from memory cache)	0 ms
35px-Wikibooks-logo.svg.png	200	png	(from memory cache)	0 ms
47px-Wikidata-logo.svg.png	200	png	(from memory cache)	0 ms
51px-Wikinews-logo.svg.png	200	png	(from memory cache)	0 ms

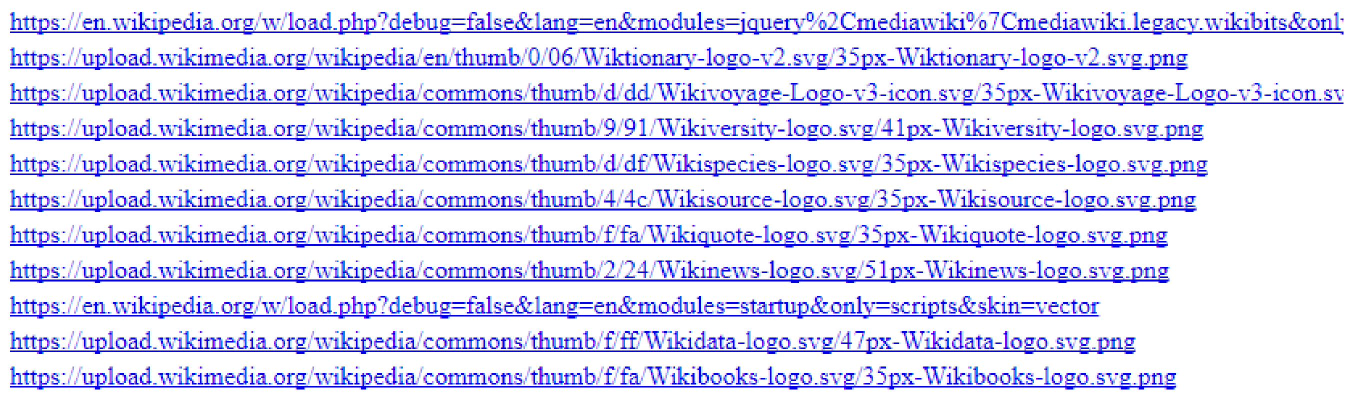
39 requests | 928 B transferred | Finish: 369 ms | DOMContentLoaded: 106 ms | Load: 194 ms

The data transferred went down to 928 bytes — that's 0.3% the size of the initial page load. The Size column shows us that most of the content is pulled from cache.

> Chrome will pull files from either memory cache or disk cache. Since we didn't close our browser between Cases 1 & 2, the data was still in memory cache.

Show the Browser Cache

In Chrome, we can go to `chrome://cache` to view the contents of the cache. This will display a page of links to a detailed view for each cached file.



https://en.wikipedia.org/w/load.php?debug=false&lang=en&modules=jquery%2Cmediawiki%7Cmediawiki.legacy.wikibits&onl
https://upload.wikimedia.org/wikipedia/en/thumb/0/06/Wiktionary-logo-v2.svg/35px-Wiktionary-logo-v2.svg.png
https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Wikivoyage-Logo-v3-icon.svg/35px-Wikivoyage-Logo-v3-icon.sv
https://upload.wikimedia.org/wikipedia/commons/thumb/9/91/Wikiversity-logo.svg/41px-Wikiversity-logo.svg.png
https://upload.wikimedia.org/wikipedia/commons/thumb/d/df/Wikispecies-logo.svg/35px-Wikispecies-logo.svg.png
https://upload.wikimedia.org/wikipedia/commons/thumb/4/4c/Wikisource-logo.svg/35px-Wikisource-logo.svg.png
https://upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Wikiquote-logo.svg/35px-Wikiquote-logo.svg.png
https://upload.wikimedia.org/wikipedia/commons/thumb/2/24/Wikinews-logo.svg/51px-Wikinews-logo.svg.png
https://en.wikipedia.org/w/load.php?debug=false&lang=en&modules=startup&only=scripts&skin=vector
https://upload.wikimedia.org/wikipedia/commons/thumb/f/ff/Wikidata-logo.svg/47px-Wikidata-logo.svg.png
https://upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Wikibooks-logo.svg/35px-Wikibooks-logo.svg.png

How Does the Browser Know What to Cache?

The browser inspects the headers of the HTTP response generated by the web server. There are four headers commonly used for caching:

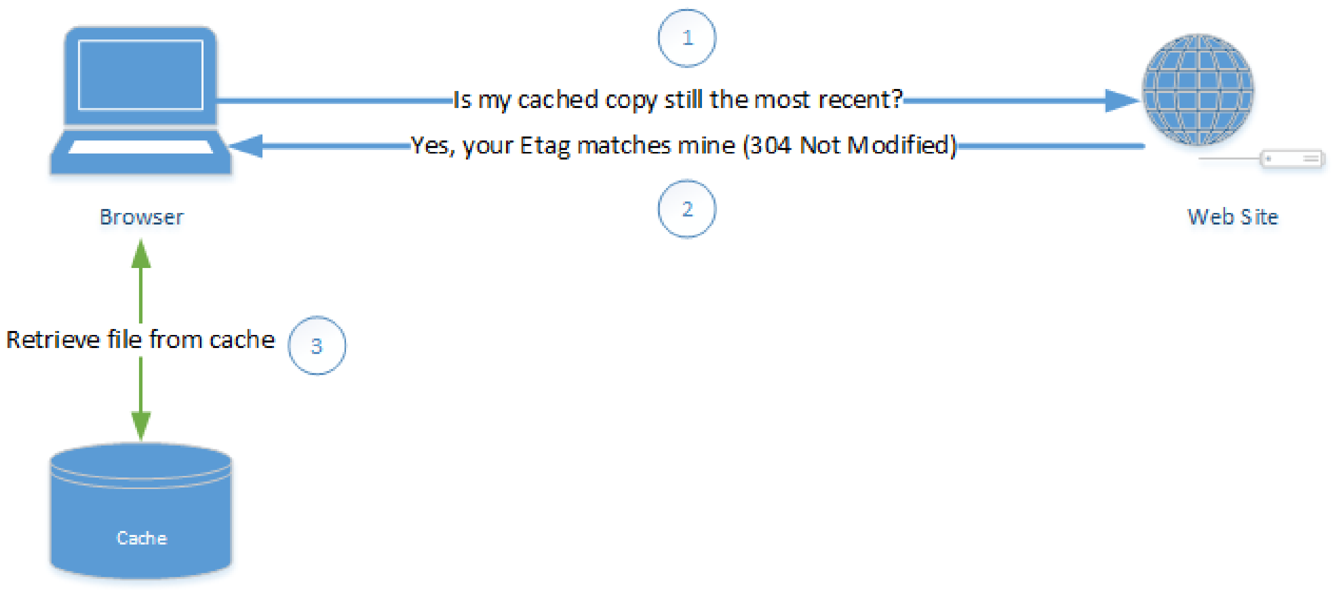
- ETag
- Cache-Control
- Expires
- Last-Modified

ETag

The *ETag* (or Entity Tag) is a string that serves as a cache validation token. This is usually a hash of the file contents.

The server can include an ETag in its response, which the browser can then use this in a future request (after the file has expired) to determine if the cache contains a stale copy.

If the hash is the same, then the resource hasn't changed and the server responds with a 304 response code (Not Modified) with an empty body. This lets the browser know it's still safe to use the cached copy.


1
Is my cached copy still the most recent?
Yes, your Etag matches mine (304 Not Modified)
2
Browser
Web Site
Retrieve file from cache
3
Cache

Note that ETag is only used in requests whenever the file has expired from cache.

Cache-Control

The *Cache-Control* header has a number of directives we can set to control cache behavior, expiration, and validation. These can be combined together as well.

Cache Behavior

```
Cache-Control: public
```

public means that the resource can be cached by any cache (browser, CDN, etc)

```
Cache-Control: private
```

private means that the resource can only be cached by the browser

```
Cache-Control: no-store
```

This tells the browser to always request the resource from the server

```
Cache-Control: no-cache
```

This one is actually a bit misleading. It doesn't mean "do not cache".

This tells the browser to cache the file but not to use it until it checks with the server to validate we have the latest version. This validation is done with the ETag header.

This is commonly used with HTML files since it makes sense for the browser to always check for the latest markup.

Expiration

```
Cache-Control: max-age=60
```

This specifies the length of time in seconds the resource should be cached.So a *max-age=60* means that it should be cached for 1 minute. RFC 2616 recommends that the maximum value for should no longer than 1 year (*max-age=31536000*).

```
Cache-Control: s-max-age=60
```

This is only used by intermediate caches like a CDN.

Validation

```
Cache-Control: must-revalidate
```

This tells the cache it must verify the status of the stale resource before using it and expired ones should not be used.

Expires

The *Expires* header is from the older HTTP 1.0 days but is still used on many sites.

This header field provides an expiration date after which the asset is considered invalid.

```
Expires: Wed, 25 Jul 2018 21:00:00 GMT
```

> The browser will ignore this field if there’s a max-age directive in Cache-Control

Last-Modified

The Last-Modified header is also from the HTTP 1.0 days.

```
Last-Modified: Mon, 12 Dec 2016 14:45:00 GMT
```

This field contains the date and time the resource was last modified.

HTML Meta Tag

Prior to HTML5, using meta tags inside HTML to specify cache-control was a valid approach:

```
<meta http-equiv="Cache-control" content="no-cache">
```

Using a meta tag like this is now discouraged and is not valid HTML5. Why? It's not a good idea because only browsers will be able to parse the meta tag and understand it. Intermediate caches won't.

So always send caching instructions via HTTP headers.

HTTP Response

Let's take a look at an sample HTTP response:

```
Accept-Ranges: bytes
Cache-Control: max-age=3600
Connection: Keep-Alive
Content-Length: 4361
Content-Type: image/png
Date: Tue, 25 Jul 2017 17:26:16 GMT
ETag: "1109-554221c5c8540"
Expires: Tue, 25 Jul 2017 18:26:16 GMT
Keep-Alive: timeout=5, max=93
Last-Modified: Wed, 12 Jul 2017 17:26:05 GMT
Server: Apache
```

- Line 2 tells us that the max-age is 1 hour
- Line 5 tells us that this is a PNG image
- Line 7 shows us the ETag value which will be used for validation after the 1 hour mark to verify that the resource hasn't changed
- Line 8 is the Expires header which will be ignored since max-age is set
- Line 10 is the Last-Modified header which shows when the image was last modified

Caching Pitfalls

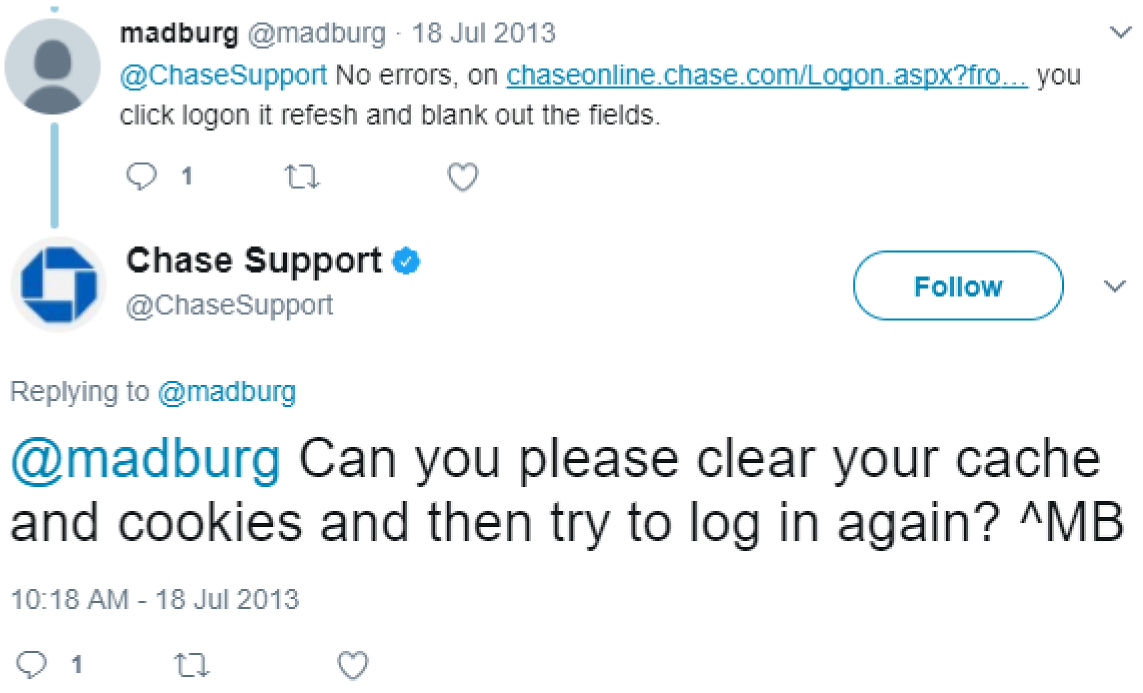
So we've established that browser caching is awesome, and we should take advantage of it.

But we also want users see the latest version of our page when we make updates. We can't expect them to do a hard refresh (Ctrl-F5) every time they visit our site or clear their cache regularly.

These types of caching issues are often a source of frustration for both the developer and end-user. A user may see a broken page or a button that behaves strangely because they have an outdated stylesheet or JavaScript code.

Stale Files

Below is a Twitter exchange between Chase Support and a user having issues with a login form on the banking site.



madburg @madburg · 18 Jul 2013
@ChaseSupport No errors, on chaseonline.chase.com/Logon.aspx?fro… you click logon it refesh and blank out the fields.

1

Chase Support @ChaseSupport

Follow

Replying to @madburg

@madburg Can you please clear your cache and cookies and then try to log in again? ^MB

10:18 AM - 18 Jul 2013

1

The user likely had some old JavaScript cached in their browser which caused the form to reset instead of submit when the Logon button was clicked.

Let’s explore another situation where stale files could bite us.

Suppose we fix a bug in a JavaScript file called *app.min.js* and push the update to our production site.

This is what our HTML looks like:

```
<script src="assets/js/app.min.js">
```

Our web server sets the *max-age* of JavaScript files to 1 week (604,800 seconds).

```
Cache-Control: private, max-age=604800
```

After the update, some users report they are still having issues symptomatic of the bug.

What's going on here?

- Bob visited the site 2 weeks ago and has a cached copy of buggy *app.min.js.* Since his copy is older than max-age, the browser will retrieve the file from the server, and he gets the latest bug-free version.
- Mary visited the site 2 days ago and also has a cached copy of buggy *app.min.js.* Her copy is newer than max-age so her browser is still happily using the cached copy.

In the next section, we'll see how to prevent these issues with a technique called *cache busting*.

Cache Busting

Cache busting is where we invalidate a cached file and force the browser to retrieve the file from the server.

We can instruct the browser to bypass the cache by simply changing the filename. To the browser, this is a completely new resource so it will fetch the resource from the server.

Cache busting also allows us to keep long *max-age* values for resources that may change frequently. Google recommends that max-age be set to 1 year (source).

Versioning

We could add a version number to the filename:

```
assets/js/app-v2.min.js
```

Fingerprinting

We could add a fingerprint based on the file contents:

```
assets/js/app-d41d8cd98f00b204e9800998ecf8427e.min.js
```

Append Query String

We could append a query string to the end of the filename:

```
assets/js/app.min.js?version=2
```

The query string approach has known issues with proxy servers so this method is generally discouraged.

Best Practices

Do

- Use the Cache-Control and ETag headers to control cache behavior for static assets
- Set long max-age values to reap the benefits of browser cache
- Use fingerprinting or versioning for cache busting

Don't

- Use HTML meta tags to specify cache behavior
- Use query strings for cache busting

FAQ

How can I tell if a file was loaded from cache?

Check out the Developer Tools in your browser. In Chrome, this information is shown in the Network tab under the Size column.

How do I prevent caching for a file?

Use the following response header:

```
Cache-Control: no-cache, no-store, must-revalidate
```

Web Development Tech Programming Software Development Technology