Fresher service workers, by default

By Jeff Posnick
Web DevRel @ Google

tl;dr

Starting in Chrome 68, HTTP requests that check for updates to the service worker script will no longer be fulfilled by the HTTP cache by default. This works around a common developer pain point, in which setting an inadvertent `Cache-Control:` header on your service worker script could lead to delayed updates.

If you've already opted-out of HTTP caching for your `/service-worker.js` script by serving it with `Cache-Control: max-age=0`, then you shouldn't see any changes due to the new default behavior.

Background

Every time you navigate to a new page that's under a service worker's scope, explicitly call `registration.update()` from JavaScript, or when a service worker is "woken up" via a `push` or `sync` event, the browser will, in parallel, request the JavaScript resource that was originally passed in to the `navigator.serviceWorker.register()` call, to look for updates to the service worker script.

For the purposes of this article, let's assume its URL is `/service-worker.js` and that it contains a single call to `importScripts()`, which loads additional code that's run inside the service worker:

```
// Inside our /service-worker.js file:
importScripts('path/to/import.js');

// Other top-level code goes here.
```

What's changing?

Prior to Chrome 68, the update request for `/service-worker.js` would be made via the HTTP cache (as most fetches are). This meant if the script was originally sent with `Cache-Control: max-age=600`, updates within the next 600 seconds (10 minutes) would not go to the network, so the user may not receive the most up-to-date version of the service worker. However, if `max-age` was greater than 86400 (24 hours), it would be treated as if it were 86400, to avoid users being stuck with a particular version forever.

Starting in 68, the HTTP cache will be ignored when requesting updates to the service worker script, so existing web applications may see an increase in the frequency of requests for their service worker script. Requests for `importScripts` will still go via the HTTP cache. But this is just the default—a new registration option, `updateViaCache` is available that offers control over this behavior.

updateViaCache

Developers can now pass in a new option when calling `navigator.serviceWorker.register()`: the `updateViaCache` parameter. It takes one of three values: `'imports'`, `'all'`, or `'none'`.

The values determine if and how the browser's standard HTTP cache comes into play when making the HTTP request to check for updated service worker resources.

- When set to `imports`, the HTTP cache will never be consulted when checking for updates to the `/service-worker.js` script, but will be consulted when fetching any imported scripts (`path/to/import.js`, in our example). This is the default, and it matches the behavior starting in Chrome 68.
- When set to `'all'`, the HTTP cache will be consulted when making requests for both the top-level `/service-worker.js` script, as well as any scripts imported inside of the service worker, like `path/to/import.js`. This option corresponds to the previous behavior in Chrome, prior to Chrome 68.
- When set to `'none'`, the HTTP cache will not be consulted when making requests for either the top-level `/service-worker.js` or for any imported scripted, such as the hypothetical `path/to/import.js`.

For example, the following code will register a service worker, and ensure that the HTTP cache is never consulted when checking for updates to either the `/service-worker.js` script, or for any scripts that are referenced via `importScripts()` inside of `/service-worker.js`:

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/service-worker.js', {
    updateViaCache: 'none',
    // Optionally, set 'scope' here, if needed.
  });
}
```

What do developers need to do?

If you've effectively opted-out of HTTP caching for your `/service-worker.js` script by serving it with `Cache-Control: max-age=0` (or a similar value), then you shouldn't see any changes due to the new default behavior.

If you do serve your `/service-worker.js` script with HTTP caching enabled, either intentionally or because it's just the default for your hosting environment, you may start seeing an uptick of additional HTTP requests for `/service-worker.js` made against your server—these are requests that used to be fulfilled by the HTTP cache. If you want to continue allowing the `Cache-Control` header value to influence the freshness of your `/service-worker.js`', you'll need to start explicitly setting `updateViaCache: 'all'` when registering your service worker.

Given that there may be a long-tail of users on older browser versions, it's still a good idea to continue setting the `Cache-Control: max-age=0` HTTP header on service worker scripts, even though newer browsers might ignore them.

Developers can use this opportunity to decide whether they want to explicitly opt their imported scripts out of HTTP caching now, and add in `updateViaCache: 'none'` to their service worker registration if appropriate.

Further reading

"The Service Worker Lifecycle" and "Caching best practices & max-age gotchas", both by Jake Archibald, are recommended reading for all developers who deploy anything to the web.