Offline Google Analytics Made Easy

By Jeff Posnick
Web DevRel @ Google

Warning: The information in this post is out of date. Developers are encouraged to follow these steps to use a solution based on the Workbox libraries.

So you've got a progressive web app, complete with a service worker that allows it to work offline. Great! You've also got existing Google Analytics set up for your web app, and you don't want to miss out on any analytical insights coming from usage that occurs while offline. But if you try to send data to Google Analytics while offline, those requests will fail and the data will be lost.

The solution, it shouldn't surprise you to learn, is service workers! Specifically, it's adding code to your service worker to store Google Analytics requests (using `IndexedDB`) and retry them later when there's hopefully a network available. We shared code to handle this logic as part of the open source Google I/O web app, but realized this was a useful pattern, and copying and pasting code can be fragile.

Today, we're happy to announce that everything you need to handle offline Google Analytics requests within your service worker has been bundled up into an npm package: `npm install --save-dev sw-offline-google-analytics`

Using sw-offline-google-analytics

From within your existing service worker code, add the following:

```
// This code should live inside your service worker JavaScript, ideally
// before any other 'fetch' event handlers are defined:

// First, import the library into the service worker global scope:
importScripts('path/to/offline-google-analytics-import.js');

// Then, call goog.offlineGoogleAnalytics.initialize():
// See https://github.com/GoogleChrome/workbox/tree/master/packages/workbox-googl
goog.offlineGoogleAnalytics.initialize();
```

```
// At this point, implement any other service worker caching strategies
// appropriate for your web app.
```

That's all there is to it!

What's going on under the hood?

`sw-offline-google-analytics` sets up a new `fetch` event handler in your service worker, which responds to requests made to the Google Analytics domain. (The library ignores non-Google Analytics requests, giving your service worker's other `fetch` event handlers a chance to implement appropriate strategies for those resources.) It will first attempt to fulfill the request against the network. If the user is online, that will proceed as normal.

If the network request fails, the library will automatically store information about the request to `IndexedDB`, along with a timestamp indicating when the request was initially made. Each time your service worker starts up, the library will check for queued requests and attempt to resend them, along with some additional Google Analytics parameters:

- A `qt` parameter, set to the amount of time that has passed since the request was initially attempted, to ensure that the original time is properly attributed.
- Any additional parameters and values supplied in the `parameterOverrides` property of the configuration object passed to `goog.offlineGoogleAnalytics.initialize()`. For example, you could include a custom dimension to distinguish requests that were resent from the service worker from those that were sent immediately.

If resending the request succeeds, then great! The request is removed from IndexedDB. If the retry fails, and the initial request was made less than 24 hours ago, it will be kept in `IndexedDB` to be retried the next time the service worker starts. You should note that Google Analytics hits older than four hours are not guaranteed to be processed, but resending somewhat older hits "just in case" shouldn't hurt.

`sw-offline-google-analytics` also implements a "network first, falling back to cache" strategy for the actual `analytics.js` JavaScript code needed to bootstrap Google Analytics.

There's more to come!

`sw-offline-google-analytics` is part of the larger `sw-helpers` project, which is a collection of libraries meant to provide drop-in enhancements to existing service worker implementations.

Also part of that project is `sw-appcache-behavior`, a library that implements caching strategies defined in an existing AppCache manifest inside of a service worker. It's intended to help you migrate from AppCache to service workers while maintaining a consistent caching strategy, at least initially.

If you have other library ideas, we'd love to hear from you. So please file a request in the issue tracker!
