Debugging the Filesystem API

By Eric Bidelman

Engineer @ Google working on web tooling: Headless Chrome, Puppeteer, Lighthouse

The HTML5 Filesystem is a powerful API. With power, comes complexity. With complexity, comes more debugging frustrations. It's an unfortunate fact that the Chrome DevTools currently do not have support for the Filesystem API. It makes debugging it more difficult than it should be. By difficult, I mean being required to write code to list/remove files in the filesystem.

During my endeavors with the Filesystem API, I picked up a couple of tips along the way that I thought I'd pass along. Each tip has its own set of limitations, but using a combination of these will get you 90% of the way. Here are the top 5:

1. **Make sure you aren't running from `file://`**. This is a sneaky one that a lot of people get bit by. Chrome imposes major security restrictions on `file://`. Many of the advanced file APIs (`BlobBuilder`, `FileReader`, Filesystem API,...) throw errors or fail silently if you run the app locally from `file://`. If you don't have a web server handy, Chrome can be started with the `--allow-file-access-from-files` flag to bypass this security restriction. Only use this flag for testing purposes.
2. **The dreaded `SECURITY_ERR` or `QUOTA_EXCEEDED_ERR`**. This usually happens when attempting to write data but you're under the influence of #1. If that's not the case, than it's likely that you don't have quota. There are two types of quota that the filesystem can be opened with, `TEMPORARY` or `PERSISTENT`. If you're using the latter, the user needs to explicitly grant persistent storage to your app. See this post on how to do that.
3. **`filesystem:` URL FTW**. It's handy to open the `filesystem:` URL for the root `DirectoryEntry` of your app's origin. What does that mean? For example, if your app lives on `www.example.com`, open `filesystem:http://www.example.com/temporary/` in a new tab. Chrome will show a read-only list of the files/folders stored your app origin. For more info on `filesystem:` URLs, see http://www.html5rocks.com/en/tutorials/file/filesystem/#toc-filesystemurls.
4. **`chrome://settings/cookies` is your friend**. This page allows you to nuke the data stored for an origin. That includes database storage, AppCache, cookies, localStorage, and stuff in the FileSystem API. Be forewarned though, it's an all or nothing thing. You can't remove just one file or pieces of data.

5. **Don't forget about error callbacks**. The Filesystem API lives in an asynchronous world (unless you're using the sync version in Workers). Always use error callback in your API calls. They're optional, but you'll save yourself a lot of grief later on when things blow up.

Last updated July 2, 2018.