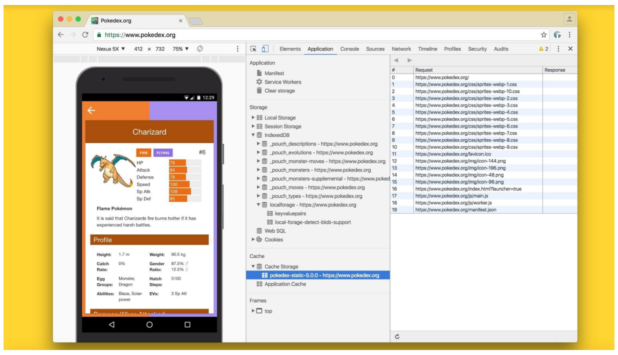
Offline Storage for Progressive Web Apps

By Addy Osmani

Eng Manager, Web Developer Relations

By Marc Cohen

Eng Manager, Web Developer Relations



The Pokedex Progressive Web App uses IndexedDB for application state and the Pokemon data set while the Cache API is used for URL addressable resources.

Internet connections can be flakey or non-existent on the go, which is why offline support and reliable performance are common features in progressive web apps. Even in perfect wireless environments, judicious use of caching and other storage techniques can substantially improve the user experience. In this post, we'll summarize some ideas around offline data storage for PWAs—think JSON payloads, images and general static data required to provide a *meaningful* experience offline.

Recommendation

Let's get right to the point with a general recommendation for storing data offline:

- For the network resources necessary to load your app while offline, use the **Cache API** (part of service workers).
- For all other data, use **IndexedDB** (with a promises wrapper).

Here’s the rationale:

Both APIs are asynchronous (IndexedDB is event based and the Cache API is Promise based). They also work with web workers, window and service workers. IndexedDB is available everywhere. Service Workers (and the Cache API) are now available in Chrome, Firefox, Opera and are in development for Edge. Promise wrappers for IndexedDB hide some of the powerful but also complex machinery (e.g. transactions, schema versioning) that comes with the IndexedDB library. IndexedDB will support observers, which allow easy synchronization between tabs.

Safari 10 has fixed many long-standing IndexedDB bugs in their latest Tech Previews. NOTE: Some folks have run into stability issues with Safari 10’s IndexedDB and PouchDB and have found it to be a little slow. Until more research has been done here, your mileage may vary. Please do test and file browser bugs so the folks @webkit and related OSS library authors can take a look. LocalForage, PouchDB, YDN and Lovefield use WebSQL in Safari by default (due to lack of an efficient way to feature-test for broken IndexedDB). This means these libraries will work in Safari 10 without extra effort (just not using IndexedDB directly).

For PWAs, you can cache static resources, composing your application shell (JS/CSS/HTML files) using the Cache API and fill in the offline page data from IndexedDB. Debugging support for IndexedDB is now available in Chrome (Application tab), Opera, Firefox (Storage Inspector) and Safari (see the Storage tab).

What about other storage mechanisms?

Web Storage (e.g LocalStorage and SessionStorage) is synchronous, has no Web Worker support and is size and type (strings only) limited. Cookies have their uses but are synchronous, lack web worker support and are also size-limited. WebSQL does not have broad browser support and its use is not recommended. The File System API is not supported on any browser besides Chrome. The File API is being improved in the File and Directory Entries API and File API specs but neither is sufficiently mature or standardized to encourage widespread adoption yet.

How much can I store?

Browser	Limit
Chrome	<6% of free space

Browser	Limit
Firefox	<10% of free space
Safari	<50MB
IE10	<250MB
Edge	Dependent on volume size

In Chrome and Opera, your storage is per origin (rather than per API). Both storage mechanisms will store data until the browser quota is reached. Apps can check how much quota they're using with the Quota Management API. In Chrome, apps can use up to 6% of free disk space. In Firefox, apps can use up to 10% of free disk space, but will prompt the user for further storage requests after 50MB data stored. In mobile Safari, apps can use up to 50MB max, whereas desktop Safari allows unlimited storage (and prompts after 5MB). IE10+ maxes out at 250MB and prompts the user at 10MB. PouchDB tracks IDB storage behavior.

How can I tell how much storage space my app is using?

In Chrome, the Quota Management API lets you query for the size of storage space currently used and how much is available to the application. A newer Storage Quota Estimate API tries to make it even easier to discover how much quota an origin is using with support for Promises.

How does cache eviction work?

Browser	Eviction Policy
Chrome	LRU once Chrome runs out of space
Firefox	LRU if the whole disk gets full
Safari	No eviction
Edge	No eviction

An origin is given an amount of space to do with as it pleases. This free space is shared across all forms of origin storage (IndexedDB, Cache API, localStorage etc). The amount given isn’t specified and will vary depending on device and storage conditions.

When web storage is low, a UA will clear storage to make space available. This can harm offline responsiveness so the recently updated Storage spec defines "persistent", and “best effort” strategies, with “best effort” being the default. “Best effort” means the storage can be cleared without interrupting the user, but is less durable for long-term and/or critical data. IndexedDB and the Cache API both fall into the “best effort” category today.

"Persistent" storage is not automatically cleared when storage is low. The user needs to manually clear this storage (via browser settings). Chrome has been experimenting with support for Persistent Storage under an origin trial, and the latest news suggests it will be shipping in Chrome 55.

Current and future offline storage work

If offline storage interests you, the efforts below are worth keeping an eye on.

- Durable Storage: protect storage from the user agent’s clearing policies.
- Indexed Database API 2.0: advanced key-value data management.
- Promisified IndexedDB: native support for a Promise-friendly version of IndexedDB.
- IndexedDB Observers: native IndexedDB observation without needing wrapper around the database.
- Async Cookies API: async JavaScript cookies API for documents and workers.
- Quota Management API: check how much quota an app/origin is using.
- writable-files: allow sites to interact with local files more seamlessly.
- Directory downloads: allow sites to download directories without .zip files.
- File and Directory Entries API: support for file and directory upload by drag-and-drop.
- Support for an Async Cookies API is being sketched out right now with a polyfill in the works.
- Debugging IndexedDB is not currently supported in Edge (however, it is possible to debug the underlying JetDB)—vote here for built in support.
- Although ideas for async LocalStorage have been kicked around in the past, current focus is on getting IndexedDB 2.0 in a good state.
- The writable-files proposal may eventually give us a better standards-track solution for seamless local file interaction.
- For apps requiring more persistent storage, see the on-going work on Durable Storage.

Offline storage isn't quite magical and an understanding of the underlying APIs will go far in helping you make the most out of what we now have available. Whether you prefer to directly use these APIs or work with an abstraction library, take some time to get familiar with your options.

Hopefully this guidance will help you craft an offline experience that makes your PWA shine! ✨

Background reading

- State of Offline Storage APIs by Joshua Bell
- Browser Database Comparison by Nolan Lawson
- IndexedDB, WebSQL, LocalStorage—What Blocks the DOM?
- How to Think about Databases (Pokedex research)
- Which APIs are Supported in Web Workers and Service Workers?

Helpful resources

- sw-toolbox (offline caching for dynamic/runtime requests)
- sw-precache (offline precaching for static assets/application shells)
- Webpack users can directly use the above or offline-plugin

IndexedDB libraries worth checking out

- localForage (~8KB, promises, good legacy browser support)
- IDB-keyval (500 byte alternative to localForage, for modern browsers)
- IDB-promised (~2k, same IndexedDB API, but with promises)
- Dexie (~16KB, promises, complex queries, secondary indices)
- PouchDB (~45KB (supports custom builds), synchronization)
- Lovefield (relational)
- LokiJS (in-memory)
- ydn-db (dexie-like, works with WebSQL)

Thanks to Nolan Lawson, Joshua Bell (whose work on Open Web Storage and BlinkOn talk heavily inspired this article), Jake Archibald, Dru Knox and others for their previous work in the web storage space.

Last updated July 2, 2018.