Browser Storage

Andrew Sutherland
Mozilla
DOM Team

Current Cross-Browser Storage APIs

- Cookies: Synchronous, Strings, Hard Storage Cap
- LocalStorage: Synchronous, Strings, Hard Storage Cap
- IndexedDB: Asynchronous, Structured Serialization w/Blobs/Files, Quota Storage Cap
- Cache API: Asynchronous, Fetch Requests/Responses, Quota Storage Cap

Structured Serialization is not Free

- Structured serialization as used by postMessage() and IndexedDB’s put()/add() is synchronous.
- By spec this is hard to optimize because getters are invoked and the getters are allowed to have side-effects.
- IndexedDB key-paths and indices add additional traversals and serialization overhead.

Blobs are Great

- A handle on a stream. (by-reference, not by-value)
- Structured serializing a Blob is cheap.
  - The data may still need to be moved around later, but that doesn’t need to happen on your thread.
- Can be disk-backed.
- Reference-counted. You can still use a disk-backed Blob from IDB even after deleting the IDB value containing it.

Quota

- navigator.storage.estimate() tells you how much space you can use before the browser will throw exceptions.
- That doesn’t mean the storage is yours forever.
- Quota limits are arbitrary and not standardized.
- navigator.storage.persist() prompts the user. If granted, your storage won’t be evicted without prompting the user.

Buckets: the unit of eviction

- 1 origin = 1 bucket
- All your origin’s data is notionally in that bucket.
- Your cookies will probably survive.
- Eviction is arbitrary and not standardized.
- Less than ideal.

Discussion Interest

- Multiple Buckets
  - How to get sites to use them?
  - How to make sure users can understand them?
    - Download model? Background-fetch?
  - Can this make quota less arbitrary?
- Third-party origins and iframes, impact from tracking protection / other privacy mechanisms.