Storage Buckets API

TPAC 2020 - WHATWG
bit.ly/tpac-2020-storage-buckets
IRC: #storage-buckets

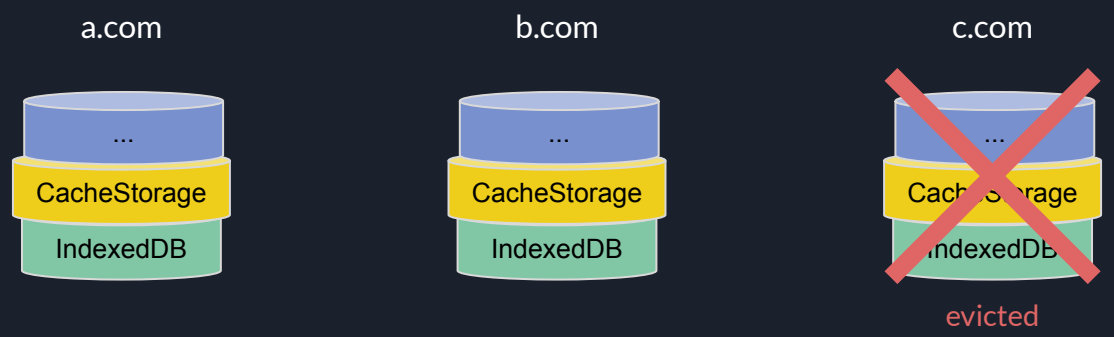
10/29/2020

Chrome Storage Team
Ayu Ishii (she/her) ayui@chromium.org
Victor Costan (he/him) pwnall@chromium.org

Agenda

- Goal
- Use Cases
- Current Explainer Overview
- Questions & Discussion

Problem Statement



- Websites either lose all or none of their data during storage pressure
- No way to express performance or durability trade-offs on partitions of data
- There are no tools / incentives for developers to manage their storage usage

Goal

Give applications more control over how their data is stored and evicted

Storage with Buckets

a.com

b.com

c.com

Storage with Buckets

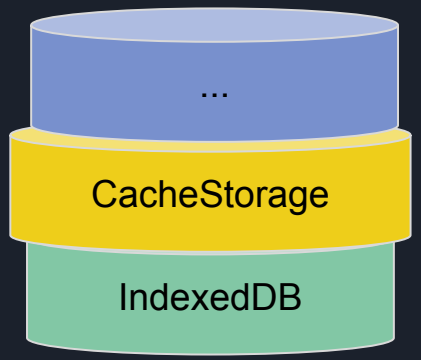
a.com

b.com

c.com

Use Cases

Email Clients Now



Contents:

- Cached inbox messages & attachments
- User drafts not yet stored in the server

Problems:

- Cached items can overload the origin's storage quota
- All data will be lost on storage eviction
- User drafts that are not recoverable will be lost forever

Email Client with Storage Buckets

Inbox Bucket

Contents: messages & attachments cached locally

Eviction: evict on storage pressure, can be reconstructed if lost

Durability: acceptable to lose last few writes on power failure

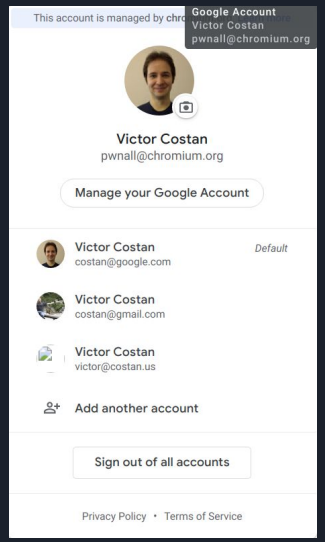
Drafts Bucket

Contents: messages composed offline, which are not yet uploaded to a server

Eviction: evict last, this is irrecoverable user data

Durability: all data should survive power failures, at the cost of more battery consumption

Document Editors with Storage Buckets




Enterprise 1


Enterprise 2


Personal 1


Personal 2

- Each bucket contains cached documents & drafts of each account
- On account log out, the bucket for the account can be deleted

Creating Important Buckets

```
const draftsBucket = await navigator.storageBuckets.openOrCreate(
  "drafts",
  {
    durability: "strict",  // Attempt to minimize data loss.
    persisted: true,       // Request persisted storage.
    title: "Drafts"
  });
```

Bucket creation for data that can not be recreated and should be last to be evicted on storage pressure.

Creating Unimportant Buckets

```
const inboxBucket = await navigator.storageBuckets.openOrCreate(
  "inbox",
  {
    durability: "relaxed",  // Deprioritize on power loss.
    persisted: false,       // Does not need persisted storage.
    title: "Inbox"
  });
```

Bucket creation for data that can be repopulated if lost, and should be evicted first on storage pressure.

Managing Quota

```
const logsBucket = await navigator.storageBuckets.openOrCreate(
  "logs",
  {
    quota: 20 * 1024 * 1024  // 20 MB
    title: "Log data"
  });
```

Per-bucket quota ensures that one application feature won't impact another feature's ability to store data by eating up the entire origin's quota.

Expiration

```
const twoWeeks = 14 * 24 * 60 * 60 * 1000;
const user1111Bucket = await navigator.storageBuckets.openOrCreate(
  "userid1111_documents",
  {
    expires: Date.now() + twoWeeks,  // Expire and remove bucket after 2 weeks.
    title: "bob@email.com Documents"
  });
```

Applications can specify expiration to ensure that the bucket's data isn't accessible to the site after a certain time.

Deleting Buckets

```
await navigator.storageBuckets.delete("inbox");

await navigator.storageBuckets.delete("drafts");
```

Buckets can be deleted upon user log out.

Service Workers & Clear-Site-Data

```
const inboxRegistration = await inboxBucket.serviceWorker.register(
    "/inbox-sw.js", { scope: "/inbox" });

// Clean up via Clear-Site-Data
Clear-Site-Data; "storage:inbox"
```

Each storage bucket can store service worker registrations. Service workers can be part of the same bucket of the stored data it is expected to access. When storage eviction occurs and data because inaccessible, the associated service workers will also be cleaned up along with the data.

Current State: Prototyping in Chromium

- Intent to Prototype (10/8/2020)
- Early TAG Review (10/7/2020)
- Getting feedback
- Identifying developers interested in early testing

Questions for the group

- Other use cases
- General concerns on API
- API shape
- Missing functionality
- Etc.

Thank you!

Links

- Early TAG Review
- Explainer
- 2019 TPAC Presentation
- 2019 TPAC Discussion
- Github repo
- Issues
- Intent to Prototype
- Security & Privacy Self Review

Extra topics

Storage API Access from Buckets

- IndexedDB
- CacheStorage
- File API
- File System Access

Public Signals

Implementers

- Gecko: Positive
- WebKit: No Signal
- Developers: Positive

Interoperability & Compatibility

If no other browsers implement, websites will only be able to use the default bucket that is supported today.

Websites can substitute with namespaces, but will have not get eviction benefits.

Web developers will have to migrate themselves when transitioning to the Storage Buckets API.

Bibliography

Bucket icon created by Vadim Solomakhin from Noun Project