Storage

Living Standard — Last Updated 18 February 2019

Participate:

GitHub whatwg/storage (new issue, open issues)
IRC: #whatwg on Freenode

Commits:

GitHub whatwg/storage/commits
Snapshot as of this commit
@storagestandard

Tests:

web-platform-tests storage/ (ongoing work)

Translations **(non-normative):**

日本語

File an issue about the selected text

Abstract

The Storage Standard defines an API for persistent storage and quota estimates, as well as the platform storage architecture.

Table of Contents

File an issue about the selected text

1. Introduction §

Over the years the web has grown various APIs that can be used for storage, e.g., IndexedDB, `localStorage`, and `showNotification()`. The Storage Standard consolidates these APIs by defining:

- A bucket, the primitive these APIs store their data in
- A way of making that bucket persistent
- A way of getting usage and quota estimates for an origin

Traditionally, as the user runs out of storage space on their device, the data stored with these APIs gets lost without the user being able to intervene. However, persistent buckets cannot be cleared without consent by the user. This thus brings data guarantees users have enjoyed on native platforms to the web.

Example

A simple way to make storage persistent is through invoking the `persist()` method. It simultaneously requests the end user for permission and changes the storage to be persistent once granted:

```
navigator.storage.persist().then(persisted => {
  if(persisted) {
    /* … */
  }
})
```

To not show user-agent-driven dialogs to the end user unannounced slightly more involved code can be written:

```
Promise.all([
  navigator.storage.persisted(),
  navigator.permissions.query({name: "persistent-storage"})
]).then(([persisted, permission]) => {
  if(!persisted && permission.status == "granted") {
    navigator.storage.persist().then( /* … */ )
  } else if(!persistent && permission.status == "prompt") {
    showPersistentStorageExplanation()
  }
})
```

The `estimate()` method can be used to determine whether there is enough space left to store content for an application:

```
function retrieveNextChunk(nextChunkInfo) {
  return navigator.storage.estimate().then(info => {
    if(info.quota - info.usage > nextChunkInfo.size)
      return fetch(nextChunkInfo.url)
    else throw new Error("insufficient space to store next chunk")
  }).then( /* … */ )
}
```

File an issue about the selected text

2. Terminology §

This specification depends on the Infra Standard. [INFRA]

This specification uses terminology from the DOM, HTML, IDL, Permissions API, and URL Standards. [DOM] [HTML] [WEBIDL] [PERMISSIONS] [URL]

A **schemeless origin group** is a group of one of the following:

- Identical opaque origins.
- Tuple origins whose host is identical and not a domain.
- Tuple origins whose host is a domain of which the registrable domain is identical.

Note

This definition will move to a more suitable location eventually.

File an issue about the selected text

3. Infrastructure §

A user agent has various kinds of storage:

Credentials
: End-user credentials, such as username and passwords submitted through HTML forms

Permissions
: Permissions for various features, such as geolocation

Network
: HTTP cache, cookies, authentication entries, TLS client certificates

Site
: Indexed DB, Cache API, service worker registrations, `localStorage`, `history.pushState()`, application caches, notifications, etc.

This specification primarily concerns itself with **site storage**.

Site storage consists of zero or more **site storage units**.

Each origin has an associated site storage unit. A site storage unit contains a single **bucket**. [HTML]

3.1. Buckets §

A bucket has **mode** which is either "`best-effort`" or "`persistent`". A **persistent bucket** is a bucket whose mode is "`persistent`". A **non-persistent bucket** is a bucket whose mode is *not* "`persistent`".

A bucket is considered to be an atomic unit. Whenever a bucket is cleared by the user agent, it must be cleared in its entirety.

File an issue about the selected text

4. Persistence permission §

A bucket can only be turned into a persistent bucket if the user (or user agent on behalf of the user) has granted permission to use the `"persistent-storage"` feature.

Note

When granted to an origin, the persistence permission can be used to protect storage from the user agent's clearing policies. The user agent cannot clear storage marked as persistent without involvement from the origin or user. This makes it particularly useful for resources the user needs to have available while offline or resources the user creates locally.

The **`"persistent-storage"`** powerful feature's permission-related flags, algorithms, and types are defaulted, except for:

permission state

`"persistent-storage"`'s permission state must have the same value for all environment settings objects with a given origin.

permission revocation algorithm

If `"persistent-storage"`'s permission state is not `"granted"`, then set the current origin's site storage unit's bucket's mode to "`best-effort`".

File an issue about the selected text

5. Usage and quota §

The **site storage usage** of an origin *origin* is a rough estimate of the amount of bytes used in *origin*'s site storage unit.

Note

This cannot be an exact amount as user agents might, and are encouraged to, use deduplication, compression, and other techniques that obscure exactly how much bytes an origin uses.

The **site storage quota** of an origin *origin* is a conservative estimate of the amount of bytes available to *origin*'s site storage unit. This amount should be less than the total available storage space on the device to give users some wiggle room.

Note

User agents are strongly encouraged to provide "popular" origins with a larger site storage quota. Factors such as navigation frequency, recency of visits, bookmarking, and permission for `"persistent-storage"` can be used as indications of "popularity".

File an issue about the selected text

6. User Interface Guidelines §

User agents should not distinguish between network storage and site storage in their user interface. Instead user agents should offer users the ability to remove all storage for a given schemeless origin group. This ensures to some extent that network storage cannot be used to revive site storage. This also reduces the amount users need to know about the different ways in which a schemeless origin group can store data.

Credentials storage should be separated as it might contain data the user might not be able to revive, such as an autogenerated password. Since permissions storage is mostly simple booleans it too can be separated to avoid inconveniencing the user. Credentials and permissions are also somewhat easier to understand and differentiate for users from network storage and site storage.

6.1. Storage Pressure §

When the user agent notices it comes under storage pressure and it cannot free up sufficient space by clearing network storage and non-persistent buckets within site storage, then the user agent should alert the user and offer a way to clear persistent buckets.

File an issue about the selected text

7. API §

```
IDL
[SecureContext]
interface mixin NavigatorStorage {
  [SameObject] readonly attribute StorageManager storage;
};
Navigator includes NavigatorStorage;
WorkerNavigator includes NavigatorStorage;
```

Each environment settings object has an associated `StorageManager` object. [HTML]

The **`storage`** attribute's getter must return context object's relevant settings object's `StorageManager` object.

```
IDL
[SecureContext,
 Exposed=(Window,Worker)]
interface StorageManager {
  Promise<boolean> persisted();
  [Exposed=Window] Promise<boolean> persist();

  Promise<StorageEstimate> estimate();
};

dictionary StorageEstimate {
  unsigned long long usage;
  unsigned long long quota;
};
```

The **`persisted()`** method, when invoked, must run these steps:

1. Let *promise* be a new promise.
2. Let *origin* be context object's relevant settings object's origin.
3. If *origin* is an opaque origin, then reject *promise* with a `TypeError`.
4. Otherwise, run these steps in parallel:
    1. Let *persisted* be true if *origin*'s site storage unit's bucket is a persistent bucket, and false otherwise.

       Note

       It will be false when there's an internal error.
    2. Queue a task to resolve *promise* with *persisted*.
5. Return *promise*.

The **`persist()`** method, when invoked, must run these steps:

1. Let *promise* be a new promise.
2. Let *origin* be context object's relevant settings object's origin.
3. If *origin* is an opaque origin, then reject *promise* with a `TypeError`.
4. Otherwise, run these steps in parallel:
    1. Let *permission* be the result of requesting permission to use `"persistent-storage"`.

       Note

       User agents are encouraged to not let the user answer this question twice for the same origin around the same time and this algorithm is not equipped to handle such a scenario.
    2. Let *persisted* be true, if *origin*'s site storage unit's bucket is a persistent bucket, and false otherwise.

File an issue about the selected text

Note

It will be false when there's an internal error.

3. If *persisted* is false and *permission* is `"granted"`, then:
    1. Set *origin*'s site storage unit's bucket's mode to "`persistent`".
    2. If there was no internal error, then set *persisted* to true.
4. Queue a task to resolve *promise* with *persisted*.

5. Return *promise*.

The **`estimate()`** method, when invoked, must run these steps:

1. Let *promise* be a new promise.
2. Let *origin* be context object's relevant settings object's origin.
3. If *origin* is an opaque origin, then reject *promise* with a `TypeError`.
4. Otherwise, run these steps in parallel:
    1. Let *usage* be site storage usage for *origin*.
    2. Let *quota* be site storage quota for *origin*.
    3. Let *dictionary* be a new `StorageEstimate` dictionary whose `usage` member is *usage* and `quota` member is *quota*.
    4. If there was an internal error while obtaining *usage* and *quota*, then queue a task to reject *promise* with a `TypeError`.

    Note

    Internal errors are supposed to be extremely rare and indicate some kind of low-level platform or hardware fault. However, at the scale of the web with the diversity of implementation and platforms, the unexpected does occur.

    5. Otherwise, queue a task to resolve *promise* with *dictionary*.
5. Return *promise*.

File an issue about the selected text

Acknowledgments §

With that, many thanks to Adrian Bateman, Alex Russell, Aislinn Grigas, Ali Alabbas, Ben Kelly, Ben Turner, Dale Harvey, David Grogan, fantasai, Jake Archibald, Jeffrey Yasskin, Jinho Bang, Jonas Sicking, Joshua Bell, Kenji Baheux, Kinuko Yasuda, Luke Wagner, Michael Nordman, Mounir Lamouri, Shachar Zohar, 黃強 (Shawn Huang), and 簡冠庭 (Timothy Guan-tin Chien) for being awesome!

This standard is written by Anne van Kesteren (Mozilla, annevk@annevk.nl).

File an issue about the selected text

Index §

Terms defined by this specification §

Terms defined by reference §

- [DOM] defines the following terms:
  - context object
- [HTML] defines the following terms:
  - Navigator
  - WorkerNavigator
  - environment settings object
  - host
  - in parallel
  - opaque origin
  - origin (for environment settings object)
  - queue a task
  - relevant settings object
  - tuple origin
- [PERMISSIONS] defines the following terms:
  - "granted"
  - permission revocation algorithm
  - permission state
  - powerful feature
  - requesting permission to use
- [URL] defines the following terms:
  - domain
- [WEBIDL] defines the following terms:
  - Exposed
  - SameObject
  - SecureContext
  - TypeError
  - boolean
  - unsigned long long

File an issue about the selected text

References §

Normative References §

[DOM]

Anne van Kesteren. DOM Standard. Living Standard. URL: https://dom.spec.whatwg.org/

[HTML]

Anne van Kesteren; et al. HTML Standard. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[INFRA]

Anne van Kesteren; Domenic Denicola. Infra Standard. Living Standard. URL: https://infra.spec.whatwg.org/

[PERMISSIONS]

Mounir Lamouri; Marcos Caceres; Jeffrey Yasskin. Permissions. URL: https://w3c.github.io/permissions/

[URL]

Anne van Kesteren. URL Standard. Living Standard. URL: https://url.spec.whatwg.org/

[WEBIDL]

Cameron McCormack; Boris Zbarsky; Tobie Langel. Web IDL. URL: https://heycam.github.io/webidl/

File an issue about the selected text

IDL Index §

IDL
```
[SecureContext]
interface mixin NavigatorStorage {
  [SameObject] readonly attribute StorageManager storage;
};
Navigator includes NavigatorStorage;
WorkerNavigator includes NavigatorStorage;

[SecureContext,
 Exposed=(Window,Worker)]
interface StorageManager {
  Promise<boolean> persisted();
  [Exposed=Window] Promise<boolean> persist();

  Promise<StorageEstimate> estimate();
};

dictionary StorageEstimate {
  unsigned long long usage;
  unsigned long long quota;
};
```

File an issue about the selected text