Persistent Storage

By Chris Wilson

Chris is a contributor to Web**Fundamentals**

With Chrome 52, we introduced the ability to make storage persistent. Storage for web applications is a complex topic, and persistence for data on the frequently-ephemeral web doubly so, so I should explain.

Normally, web applications store local data in various ways - in IndexedDB databases, through the Cache API, even (gasp) localStorage. All of this storage for a given domain takes up space on the local machine, of course.

When storage on the local machine is running tight ("under storage pressure"), user agents automatically clear storage to make more available space. Of course, for offline apps, this may be unfortunate, as they may not have synced their data to the server yet, or they may be apps that the user expects to just work offline (like a music player); so the Storage spec defines two different modes for storage for a given domain - "best effort" and "persistent". The default mode, of course, is "best effort". Storage for a domain that is "best effort" (aka "not persistent") can be cleared automatically, without interrupting or asking the user. However, "persistent" data will not be automatically cleared. (If the system is still under storage pressure after clearing all non-persistent data, the user will need to manually clear any remaining persistent storage.)

How do I make my storage persistent?

So how do I make my storage persistent? Well, you have to ask for it explicitly:

```
if (navigator.storage && navigator.storage.persist)
  navigator.storage.persist().then(granted => {
    if (granted)
      alert("Storage will not be cleared except by explicit user action");
    else
      alert("Storage may be cleared by the UA under storage pressure.");
  });
```

This feature was initially still somewhat experimental. So in order to keep from prematurely baking this design in before it was fully specified and agreed upon, we initially implemented

this feature in Chrome 52 as an Origin Trial. To use this API in Chrome 52, you needed to request a token and insert it in your application.

The trial ended in Chrome 55, when we shipped on-by-default support for this feature.

Initially, the permission was automatically granted to any sites that the user has bookmarked, and automatically denied otherwise. Beginning with Chrome 55, Chrome will automatically grant the persistence permission if any of the following are true:

- The site is bookmarked (and the user has 5 or less bookmarks)
- The site has high site engagement
- The site has been added to home screen
- The site has push notifications enabled

The permission is automatically denied in all other cases. The goal is to ensure that users can rely on their favorite web apps and not find they have suddenly been cleared.

You can also use the JavaScript API to tell if persistence has been granted already:

```
if (navigator.storage && navigator.storage.persist)
  navigator.storage.persisted().then(persistent=>{
    if (persistent)
      console.log("Storage will not be cleared except by explicit user action");
    else
      console.log("Storage may be cleared by the UA under storage pressure.");
  });
```

You probably want to request permission, but then use the **`.persisted`** API to decide whether to display offline UI (like enabling a checkbox for "make available offline"), confirming to the user they can be confident it will be available offline (even under storage pressure). This will give a graceful degradation if the user won't get an offline experience.

What about "Clear Data"? Will the user still inadvertently wipe my data?

This is still under development, but in short, the goal is to make users are aware of "persistent" data before clearing it - ideally letting them manually manage any such data. We're still designing the fine-grained options and user flow for how this can best work, but from your app, you can presume that "persistent" means your data won't get cleared without the user being explicitly informed and directly in control of that deletion.

The landscape of persistently storing data in the web platform is still changing, but we're excited to take this strong first step in making web applications more reliable!

Last updated July 2, 2018.