[Comparing Web Storage and WinRT AppData/Settings](http://windows/windows8/DevX/DevXWiki/Blog/Lists/Posts/Post.aspx?ID=7)

HTML5 includes facilities for client-side “web storage” from within JavaScript as a great improvement over cookies. For reference, see <http://en.wikipedia.org/wiki/DOM_storage> for an overview, <http://www.w3.org/TR/webstorage/> for the spec. In Windows Store apps written in HTML/CSS/JavaScript, these features overlap with the application storage APIs in WinRT. In this article I’ll compare and contrast the two.

*Note*: one key difference for any comparison with WinRT APIs is that HTML and native JavaScript APIs are available in the web context whereas WinRT is not. This comparison therefore only applies to pages running in the local context. In the web context you must use the HTML5/JS APIs.

It’s also helpful to note that overlaps such as these exist (a) because of the local and web contexts, and (b) to provide the capability to Windows Store apps written in other languages where an intrinsic API is not available.

In Web Storage, there is **local** and **session** storage that differ in scope and lifetime. Local storage is per-domain and persists after the browser is closed. Session storage is per-page/per-window and is cleaned up when the window is closed. If you have multiple windows open with the same site, the benefit here is that session storage is kept separate.

The storage itself is an associative array of strings at present; there is a proposal spec for structured data.

Two such associative arrays are made available in JavaScript: *localStorage* and *sessionStorage*. You can create any property in them you want, and retrieve them as you want (checking for existence first, of course):

if (sessionStorage.insurance) { ... }

if (!localStorage.pageLoadCount)

localStorage.pageLoadCount = 0;

localStorage.pageLoadCount += 1;

document.getElementById('count').textContent = localStorage.pageLoadCount;

The WinRT APIs are available through the [*Windows.Storage.ApplicationData*](http://msdn.microsoft.com/en-us/library/windows/apps/windows.storage.applicationdata.aspx) class, which gives you access to *StorageFolder* objects for Local, Temp, and Roaming areas of AppData along with *localSettings* and *roamingSettings* (containers for key-value pairs including sub-containers). The [Application](http://msdn.microsoft.com/en-us/library/windows/apps/br229774.aspx) object in the Windows Library for JavaScript (WinJS) also contains a *local*, *temp*, and *roaming* objectsto simplify working with the *ApplicationData* APIs (it also has a *sessionState* object into which you can write values, but these are only restored if an app is restarted after being terminated by Windows when freeing up memory).

Comparing the APIs, now:

* HTML5 *sessionStorage* (per window) in browsers is effectively the same as local variables in an app, because typically you use a single-page navigation model (with DOM replacement) to preserve the script context across pages. From an app’s point of view, then, In other words, *sessionStorage* is just a global array that had an automatic *new* done for it. That’s all.
* WinRT local and HTML5 web local storage both use persistent storage that is available to the app across instances, that is, if the app runs and writes to local storage, is then closed by the user, then restarted, the local storage values are persistent.
* HTML5 web storage has only a simple array model with strings. WinRT is much richer, with type-specific settings, hierarchical containers, composite settings (groups of settings that act as a single unit), and also supports unstructured (binary) data in arbitrary files. You’d have to encode binary data into text for web storage.
* Web storage works on a synchronous model; WinRT is synchronous for settings, asynchronous for files.
* WinRT and HTML5 storage are both kept in the user’s AppData\packages\<package> folder and is automatically cleaned up when the app is uninstalled.
* WinRT supports automatic roaming through the RoamingSettings container and Roaming folder (up to 100K total data); HTML5 web storage is entirely local.
* WinRT supports a temp folder that can be cleaned out with the Disk Cleanup tool. This makes a great place for caching data that can be restored at another time.

For small bits of data, an app might use HTML storage for the sake of simplicity if you’re familiar with the API. However, it means having a different (and UI-thread blocking) programming model from the roaming settings and a lack of unstructured data (file) capability. One also has to watch out for the fact that all the data are a strings and will be handled like strings. So for the code that does the launch count as above (which is pulled from the W3C spec), when you reload the *pageLoadCount* at a later time, the data type is actually a string, and adding 1 doesn’t increment the value, it appends “1” to the string. You have to do a *parseInt* on the value from *localStorage* to get a numerical value again.

The other comparison to make between these methods is to review the storage limits imposed by each mechanism, as summarized in the following table. For completeness, we’ve also included other storage HTML5 mechanisms like IndexedDB and AppCache as these also employ the app’s AppData area on the file system.

|  |  |
| --- | --- |
| Storage Method | Limit (up to file system capacity) |
| Web storage (local) | 10MB |
| Web storage (session) | 10MB |
| WinRT *localSettings* (structured settings) | 8K per individual setting (1)  64K per composite setting (1)  No limit on overall settings |
| WinRT *localFolder* (unstructured files) | None |
| WinRT *roamingSettings* (structured settings) | K per individual setting (1)  64K per composite setting (1)  No limit on overall settings. Exception: sync engine may limit total amount of *roamingSettings* and *roamingFolder* data that will roam (2) |
| WinRT *roamingFolder* (unstructured files) | No storage limit. However, the sync engine limits total the amount of *roamingSettings* and *roamingFolder* data that will actually roam. (2) |
| WinRT *tempFolder* (unstructured files) | None (3) |
| Cookies | 4K per cookie; no limit on total cookies |
| History | None |
| Temporary Internet Files | None (4) |
| IndexedDB | 250MB per app, 12.5% of overall storage or 1TB, whichever is less. |
| HTML5 Appcache | None (4) |

(1)    These are arbitrary limits and may change.

(2)    The default sync engine presently imposes a 100K limit (see the *roamingStorageQuota* property); if the combined contents of the *roamingSettings* and *roamingFolder* exceed that limit, the settings will still only be stored locally and **no** data is roamed. Again, this is a limit of the sync engine: the WinRT APIs themselves don’t impose limits. Different sync engines can behave differently.

(3)    Files created in the *tempFolder* are subject to deletion by Windows’ Disk Cleanup utility.

(4)    Current design with Windows Store apps does not impose a storage limit for AppCache (along with any other cached files from the web); IE will continue to enforce user- and system-defined limits.

In the end, if you’re writing new code, I recommend using the WinRT APIs to have a consistent async programming model across local, roaming, and temp data. If you have existing code that uses HTML5 web storage, it will continue to work subject to the limits outlined above.