# Comparing the HTML File and WinRT File APIs

Kraig Brockschmidt

Access to the file system on the local device is an increasingly important capability for apps that interact with the user and the user’s data on that file system. Until APIs were added to HTML5, client-side JavaScript code in web apps did not have much access to the file system except through plug-ins, which are not necessarily supported in all browsers. For Windows Store apps, access to the file system is of course a very important need, so there are extensive APIs for this purpose as well as protections for that file system. As we’ll see, Windows Store apps really need to use the WinRT APIs to work within those protections; the HTML5 file APIs are of only limited used.

This comparison is different from others posted here in that the HTML APIs and WinRT APIs really address different uses and purposes; they do not overlap as other APIs we’ve compared.

*Note*: a 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 are limited to HTML5/JS APIs.

## HTML File API and File System API

HTML5 has two emerging standards for file-related APIs: the File API and the File System API.

The [File API](http://www.w3.org/TR/2011/WD-FileAPI-20111020/), a working draft at the time of writing, is oriented around handling of filenames that a user provides in a form, typically for upload purposes. It’s entirely oriented around the workflow of obtaining one or more filenames through *<input type="file">* elements, then asynchronously accessing the contents of those files in various forms (text, binary, array, and URL). It is not intended to be a generic means of accessing files, and provides reading capabilities only. Files, in fact, are represented by a *File* object type that only surfaces the name, not the path, of the files—the API, in other words, hides those details that are specific to the underlying operating system.

The [File System API](http://www.w3.org/TR/file-system-api/), also a working draft (and not generally available to apps at this time), provides for more general read/write access to “a sandboxed section of the user’s local file system.” The API provides for folder operations like moving, copying, deleting, and creating files, as well as reading and writing files. The API provides both synchronous and asynchronous forms.

## WinRT File API

The [*Windows.Storage*](http://msdn.microsoft.com/en-us/library/windows/apps/windows.storage.aspx) API is intended as the means of asynchronous file system access for Windows Store apps. The API is fully aware of app container restrictions, brokering, and restrictions placed on the app by the capabilities (or lack thereof) as declared in the manifest. It provides for reading and writing to arbitrary files (if access is allowed) as well as to app-specific data folders (local, roaming, and temp) and user data libraries (music, pictures, videos, and documents, with the latter having additional requirements).

Unlike the working draft HTML5 File System API, the WinRT API is completely available to app Windows Store apps today. The capabilities also go far beyond the HTML5 specifications. WinRT provides a file picker that allows the user to point to and thus give permission to apps to use any non-sensitive area of the file system. Apps can cache this permission for future access through the [*Windows.Storage.AccessCache*](http://msdn.microsoft.com/en-us/library/windows/apps/windows.storage.accesscache.aspx) APIs.

WinRT also provides capabilities to perform deep and detailed searches of the file system through file queries (see [*Windows.Storage.StorageFolder*](http://msdn.microsoft.com/en-us/library/windows/apps/windows.storage.storagefolder.aspx)). Furthermore, WinRT is designed to accommodate the needs of online storage systems (e.g. SkyDrive) and local caching of files obtained from those systems.

## Recommendations

If an app wishes, it can use the HTML5 File API for its intended and specific purpose, that is, for an *<input type="file">* form field and resulting data transfer. It should be noted that using this method, the app will only have the ability to use the methods in the HTML *FileReader* object to read data from the file; if any other capabilities are needed, the app should collect pathnames directly through the WinRT file picker and use the *Windows.Storage* API for reading (and/or writing). This is because the paths are otherwise hidden (and not accessible) in the HTML *File* objects.

The HTML File System APIs, for their part, are not presently available to Windows Store apps at all (they are only available in Google Chrome at present). At such a time that the standard becomes more finalized, it would seem an open question as to how the API would operate. Generally speaking, since the File System API is designed to provide access to an app-specific section of the file system, as is already provided by *Windows.Storage.ApplicationData*, making such an API available to Windows Store apps would seem both redundant and pointless, since WinRT already provides a much richer set of features.