# Comparing the HTML File and WinRT File APIs

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 recently, client-side JavaScript code in web applications did not have much access to the file system except through plug-ins, but this changes with HTML5 APIs for Files and the File System. For Metro style apps in Windows 8, 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, Metro style apps really need to use the WinRT APIs to work within those protections.

## 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/), currently in the status of working draft, 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 API is really intended as the means of asynchronous file system access for Metro style 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).

## Recommendations

If an app wishes, it can use the HTML 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 Metro style apps at all (they are only available in Google Chrome). 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 Metro style apps would seem both redundant and pointless, since WinRT already provides a much richer set of features.