Developing Great HTML5 Apps for PC

HTML5 is an increasingly popular choice for software developers due to its support of rapid UI development and promise of simplified porting to multiple devices. On some devices, native apps have an advantage over HTML5 apps because native apps can fully use device capabilities. When considering HTML5 apps on PCs, most of those advantages can be made available to HTML5 apps with a few simple steps. This article will show you how to make your HTML5 app the best it can be on a PC.

User agents for HTML5 apps on PCs

HTML5 apps run within a user agent; a runtime environment designed to translate the HTML, CSS, and JavaScript* into an interactive experience. On the PC, the most common user agents are web browsers. For many years, browsers were the only options for HTML5 deployment, but now operating systems are providing their own user agents to support HTML5 directly. Microsoft Windows* 8 provides one in the form of Windows Store app APIs. Each of these user agents has different levels of support for PC device features; you should choose the right user agent based on its support for the features your app needs.

HTML5 is versatile and adaptable. Web apps can work on virtually any platform and operating system without porting, with specific features available depending on the browser used. Using Visual Studio*, HTML5 apps can be crafted for Windows 8—both Desktop and Windows Store—allowing access to more and better features than any web browser. Web developers can now use their HTML, JavaScript, and CSS expertise in native apps.

HTML5 in web browsers

The much-desired “write once, run anywhere” mindset for HTML5 finds its home in web browsers. The platform agnostic code uses the browser as an abstraction layer, with straightforward detection of available functionality. Maintenance and feature changes are made to a single code base, without the need for redundant rewriting and inconsistent versions among different devices.

As the name implies, a web app is easily distributed online; no downloading is required for distribution. This makes it ideal for a variety of projects, from social media games to enterprise-level tools. Code changes go live instantly, with no patching or app update necessary. Web apps can be run locally, but using the internet (or at least a network) simplifies distribution.

HTML5 in packaged apps

HTML has always been oriented towards visual composition and user experience, increasingly so with CSS. This lends itself easily to creating user-friendly apps, leveraging the vast array of web developers for packaged apps in Windows 8.

On Windows 8, this works by running the app on the Internet Explorer* (IE) engine. IE has been historically shunned by web developers for being slow and non-standard, but these concerns are growing outdated. IE support of HTML5 and JavaScript has more than doubled from versions 9 to 10, with nearly half that again from 10 to 11 including WebGL*. Windows 8 augments the browser engine, increasing web app performance.

The lower-level features of Windows 8 are afforded by the WinJS framework. This API picks up where the browser stops, offering greater device access.
Windows 8 Desktop UI
There are many approaches to Windows 8 Desktop packaging of HTML5 apps. Windows 8 Desktop presents touch-enabled apps in a familiar environment so users are comfortable with the appearance. Using Win32* libraries is possible here, and Win32 performance analysis tools can aid your development. Many device feature libraries use Win32 functions that are not available to Windows Store apps, restricting their use to the Desktop UI.

Windows Store Apps
Windows Store apps can be built from HTML5 as well. Windows Store app style lends itself to low-overhead touch interfaces focused on content. These apps have intuitive controls that use layout and text features to guide the user experience.

These apps also have access to the charms, a succinct selection of functional mechanics including search and share. By swiping in from the right of the screen (or moving a cursor to either right corner) the charms menu will appear. Windows Store apps can tap into these features, allowing content to be searched and shared among apps.

Windows 8 Live tiles work more like widgets than icons, updated while the app is not running in the foreground. These tiles serve to launch the apps as well as deliver at-a-glance content retrieved via web service. More information and development guidance can be found online.

Key Device Features
The choice of user agent on the PC determines which device features are available to your app. For a given device feature, API access methods vary for each user agent. Key features are described here with some guidance on how they can be accessed by user agents on the PC.

Touch Gestures
Touch gestures in any application on a Windows 8 touch-enabled device correspond to mouse movements, a tap works as a click, dragging works as scrolling, etc., but touch gestures make full use of the interface (Table 1). Many browsers have support for gestures as do Windows apps.

Table 1. The touch-based nature of these apps is strongly supported by the basic touch gestures

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tap</td>
<td>Single touch and release. Used for primary action.</td>
</tr>
<tr>
<td>Press and hold</td>
<td>Single touch and hold. Used to gain more information or access a context menu.</td>
</tr>
<tr>
<td>Slide</td>
<td>One or more fingers touch and move in one direction. Used to pan/scroll through content.</td>
</tr>
<tr>
<td>Swipe</td>
<td>Short slide. Used perpendicular to panning content to select, command, or move an object.</td>
</tr>
<tr>
<td>Turn</td>
<td>Two or more fingers touch the screen and rotate in a clockwise or counterclockwise arc. Used to rotate content.</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Pinch</td>
<td>Two or more fingers touch the screen and move toward each other. Used to zoom out.</td>
</tr>
<tr>
<td>Stretch</td>
<td>Two or more fingers touch the screen and move farther apart. Used to zoom in.</td>
</tr>
</tbody>
</table>

**Sensor Integration**

HTML5 has features to get geolocation data, the real-world geographical location of the device. Maximum fidelity is achieved by combining all information available, including IP geolocation, GPS, WiFi* positioning, and cell tower triangulation. This is widely available in browsers as well as Windows 8 apps.

In addition to geolocation, Windows 8 provides APIs that support a suite of other sensors (Table 2). By combining data from the accelerometer, gyrometer, and magnetometer in an approach called “sensor fusion,” jitter is reduced while minimizing latency. This approach allows for a few more sensors as well. While sensors are available in both Desktop and Windows Store apps, Desktop apps can change the sensitivity to suit specific needs. More information is available on MSDN.

**Table 2. Sensor data allows new control interfaces**

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Accelerometer</td>
<td>Captures acceleration on X, Y, and Z axes.</td>
</tr>
<tr>
<td>3D Gyrometer</td>
<td>Captures changes in angular velocity.</td>
</tr>
<tr>
<td>3D Compass</td>
<td>Captures changes in orientation.</td>
</tr>
<tr>
<td>3D Inclinometer</td>
<td>Captures changes in the inclination angle.</td>
</tr>
<tr>
<td>Light</td>
<td>Senses changes in ambient lighting.</td>
</tr>
<tr>
<td>Device Orientation</td>
<td>Sensor fusion provides fine-grain orientation information.</td>
</tr>
<tr>
<td>Simple Orientation</td>
<td>High-level orientation information such as facing up, facing down, and rotations of 90, 180, or 270 degrees.</td>
</tr>
</tbody>
</table>
3D Rendering using WebGL
WebGL, the web 3D graphics standard, is usable in everything including Windows Store apps as of Windows 8.1. Partial support is available in Firefox*, but IE11 and Chrome* are fully capable of rendering 3D games and apps. As WebGL matures, HTML5 will become an even more viable platform for all varieties of games.

Communicating with Other Devices
Similar to mobile devices, larger form factors such as Ultrabook™ devices can now make use of Near-Field Communication (NFC) to transmit information between devices. This is confirmed in Firefox, and the Windows.Networking.Proximity library for Windows 8.

Responsive Design Using Media Queries
One of the biggest advantages of HTML5 is support of media queries. These allow “responsive design” — dynamic content based on the viewing device information. Web sites previously redirected users to mobile versions of the content, often ruining the experience with formatting glitches or malfunctioning code. Now we can have a single code base that displays the content properly on any screen. Windows 8 goes a step further, providing the view state information so the app can adapt just as easily to being “snapped” to a portion of the screen.

Responsive design is most commonly seen in rearrangement of web site content based on the size of the viewable area (Figure 1). This can be done using CSS, JavaScript, or both.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Comparison of page layouts with varying window sizes}
\end{figure}

The style sheet defining this layout contains conditional segments. Here, the spacing and arrangement are dependent on screen width using media query statements:

\begin{verbatim}
@media screen and (max-width: 980px) { ... }
\end{verbatim}

Any styling between the curly braces will apply when the viewport width is less than 980 pixels. Further segments are defined to continue refining the layout for incrementally smaller displays. This is an elegant solution to multiple screens, but the repeated styling can impede loading times if a large amount is loaded at once. To solve this, the queries can be done using JavaScript to conditionally load different CSS files, or manipulate the content directly.

\begin{verbatim}
if ( screen.width <= 980 ) { ... }
\end{verbatim}

This statement will compare in a similar manner, but varying browser standards make redundant checks necessary. A more efficient control is to use window.matchMedia to store a media query.
var mediaQuery = window.matchMedia("(max-width: 980px) ");

This will store a media query with the given condition, which can be checked at any time using mediaQuery.matches (a Boolean value). Since this check only happens when the code is executed, it needs to be called whenever the window is resized. Fortunately, the variable can be given a listener function which executes upon changes:

```javascript
mediaQuery.addListener(CheckQuery);
function CheckQuery(mq){ … }
```

Within this function, relevant style sheets or JavaScript files can be loaded, controls can be added or removed, and any other modifications necessary to facilitate ease of use given the new size. Note: this function should also be called initially to set the content for the starting size, or it will only occur if the user resizes the window.

The responsive design offered by media queries can serve in Windows apps by using the same code or modifying it for specific APIs in WinJS. With only superficial changes, the web app demo above was ported to a Windows Store app (Figure 2).

![Image](image1.png)

**Figure 2.** Windows* Store app using media queries to modify the layout when snapped

To make an app useful in snapped mode, it will often need modified controls. Detecting the layout change is necessary to maintain productivity and a positive user experience. These changes can also be handled by APIs unique to Windows 8. Windows.UI.ViewManagement.ApplicationView.value can be compared to Windows.UI.ViewManagement.ApplicationViewState.snapped to determine whether the application is in a snapped state, handled in a resize event as before.

Integrating responsive design into web-based and compiled apps has long been a valuable feature for mobilization of web sites, but resizing the screen is no longer the only change users can make to the viewport. In the case of detachable 2-in-1 devices, an app needs to instantly convert its controls between laptop and tablet mode, often without any visible change in the screen size. Handling this smoothly can turn a potentially jarring transition into a slick feature point. More information on media queries and WinJS can be found in the references section below.
Comparing the options
Since many features are specific to individual browsers or app types, this chart compares what’s currently available.

<table>
<thead>
<tr>
<th></th>
<th>Web Apps</th>
<th>Windows* 8 Desktop Apps</th>
<th>Windows Store Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Internet Explorer*</td>
<td>Firefox*</td>
<td>Chrome*</td>
</tr>
<tr>
<td>Intel® WiDi Technology</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Touch Gestures</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Orientation Sensors</td>
<td>IE11</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Light Sensor</td>
<td>No</td>
<td>Bugged</td>
<td>No</td>
</tr>
<tr>
<td>WebGL*</td>
<td>IE11</td>
<td>Partial</td>
<td>Yes</td>
</tr>
<tr>
<td>Intel® Smart Connect</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NFC</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>GPS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Media Queries</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Distribution</td>
<td>Online</td>
<td>Online</td>
<td>Online</td>
</tr>
</tbody>
</table>

ΩRequires an Intel® Wireless Display enabled PC, compatible adapter, and TV. 1080p and Blu-Ray® or other protected content playback only available on 2nd generation Intel® Core™ processor-based PCs with built-in visuals enabled. Consult your PC manufacturer. For more information, see www.intel.com/go/widi

Helpful Tools
In the past, developers have used text editors and visual layout tools to create HTML apps, with suites of complex library dependencies to port to other platforms. Now tools such as the Intel® XDK offer a cross-platform development environment to bring it all together. More information and downloads are available at http://software.intel.com/en-us/html5/tools.
References
Porting Web Apps to Windows 8, Adding Charms - http://html5hacks.com/blog/2013/04/02/make-your-web-app-to-a-windows-8-app/
Geolocation in Browsers - http://diveintohtml5.info/geolocation.html

About the Author
Brad Hill is a Software Engineer at Intel in the Developer Relations Division. Brad investigates new technologies on Intel hardware and shares the best methods with software developers via the Intel Developer Zone and at developer conferences. He also runs Code for Good Student Hackathons at colleges and universities around the country.

Notices
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A
PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel, the Intel logo, AppUp, Core, and Ultrabook are trademarks of Intel Corporation in the U.S. and/or other countries.

Copyright © 2013 Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.