QNX SOFTWARE SYSTEMS

Why HTML5 Is Becoming the HMI Technology of Choice

Andy Gryc, Senior Product Marketing Manager, Automotive
Marc Lapierre, QNX CAR Developer
agryc@qnx.com, mlapierre@qnx.com

Introduction

Since the advent of the internet, the non-initiate who try to understand a new technology are often faced, not with too little information, but with too much. HTML5 is no exception. If anything, information about HTML5 is over-abundant: the W3C documents themselves, business cases, evangelists' musings, and seemingly innumerable and ever-more abundant how-to articles and tutorials chock full of code samples.[1]

This whitepaper attempts to bridge the gap between the musings and the tutorials, and present a brief overview of HTML5 that will be useful to people who are technical enough to understand what we mean by a chromeless browser, but who are not already up to their elbows in HTML5 and looking for, say, a tip on how to optimize a bit of JavaScript for equalizing audio output.

Figure 1. Close up of an automotive infotainment system designed with HTML5.

To this end, we can start by defining what we mean by "HTML5". Strictly, HTML5 is the latest, and as yet incomplete edition of the HTML standard for presenting content. However, in common usage HTML5 usually refers to the HTML5 standard itself, *plus* ancillary standards and technologies: CSS3 (Cascading Style Sheets), the JavaScript scripting language and associated

[1] For example, see HTML5 Rocks for detailed (but usually comprehensible to non-developers) information about HTML5 and how to use it. <http://www.html5rocks.com>

standards, such as AJAX (Asynchronous JavaScript And XML) and JSON (JavaScript Object Notation), Document Object Model (DOM), and other non-proprietary standards, including XML. It is this, broader, definition of HTML5 that interests us here.

Second, we can simplify the discussion by grouping into three areas the bewildering number of new features, capabilities and improvements HTML5 brings over its predecessors:

- Applications: HTML5 now supports enough capability to construct applications either inside or outside of a browser, with all the expected capabilities: databases, threading, input from the device hardware, etc.
- Rendering: using CSS3 animations, the `<canvas>` element, WebGL, and SVG graphics, HTML5 provides control over the HMI rendering that is precise enough for games and flexible enough for applications
- Improved application development model: the influx of application developers (as opposed to web page designers) adopting HTML5 to build cross-platform applications, is re-orienting the HMI development community, which is increasingly following traditional design patterns in its applications, separating the model (HTML/DOM), view (CSS), and controller (JavaScript) in a more maintainable architecture

Finally, we should note that with these changes, HTML5 reaches beyond the traditional domain of HTML. It is no longer just the standard for presenting web content, but a viable technology for HMIs for all sorts of applications—connected and not connected, using traditional browsers or chromeless browsers (rendering engines minus all the browser widgets: navigation, history bookmarking, etc.).

HTML5 applications

HTML5 introduces features that give an HTML page many of the capabilities typically associated with an application. One of the key benefits of these capabilities is that with HTML5, HMI designers don't need to choose between a downloaded app running on the device, and a service hosted in the cloud. They can support both use cases from the same code base.

For example, ideally, an in-vehicle navigation app is always connected, receiving traffic and weather data, updating the vehicle location, and so on. In practice, however, connectivity can't always be guaranteed. Cars travel outside network coverage areas, down urban canyons, and through tunnels. A hybrid navigation app that stores data locally could continue without a blink in any of these situations, synchronizing its locally-stored information with updates when connectivity was re-established.

Local data storage

A key requirement for many off-line apps is local data storage. With previous versions of HTML, local storage was limited, unreliable and restrictive: cookies, (small, hence very limited capacity), plug-ins (annoying, often out of date, blocked by firewalls), or custom browser features (which created browser lock-in). HTML5 offers designers Web Storage and Indexed DB.

Web storage

Web storage supersedes the old cookies local storage model, and is:

- faster—every server request doesn't require a new data transmission
- more secure—stored data is accessible only to the web page that stored it
- more useful—data size is not severely limited as it is with cookies

Web Storage comes in two flavors: Local Storage (data is persistent; it remains on the local device until it is expressly removed) and Session Storage (data is not persistent; it is removed when the browser window with which it is associated is closed). All Web Storage data is stored in key/value pairs.

The catches with Web Storage, are that it does not support:

- indexing—searches on large amounts of data can hurt performance
- locking for database transactions—apps in multiple tabs can overwrite each other's data, so applications must manage their own database transactions to ensure data integrity

Indexed DB

Indexed DB offers faster searches and transaction locking. It is less complex than Web SQL Database, which W3C dropped,[2] perhaps due to neither Microsoft nor Mozilla supporting it.

As its name suggests, Indexed DB supports indexing of specified fields, which speeds searches; and locking of the entire database, tables or individual rows, which ensures data integrity when more than one app accesses the database. Using Indexed DB requires a bit more knowledge than does using Web Storage, but applications can do a lot more with the stored data.

We need to keep in mind, however, that Indexed DB is still in the proposal stage and is not yet part of the HTML5 specification, and is not yet fully supported by all browsers. Though Microsoft has rumored that IE10 will support Indexed DB, few details are available about mobile browser support.

Threading

According to one web site designer, threading support is worth pretty much every other improvement in HTML5. Threading is handled with JavaScript improvements, which provide a pseudo-threading implementation with the newly added Web Workers. A Web Worker is "a single JavaScript file loaded and executed in the background"[3]. Web Workers do not have access to all JavaScript features; for instance, they can't access the DOM or any global variables or JavaScript functions[4]. Despite these limitations, threading means that complex HTML5 applications can be designed much more naturally.

For instance, HTML5 could be used to write an application running a fuel dispensing pump. The application would have to include functionality such as reading the buyer's credit card and managing the connection for the transaction, picking up the fuel rate of flow from native code connected to the pump

---

[2] W3C ceased development of and support for the Web SQL Database specification in November 2010. <http://dev.w3.org/html5/webdatabase/ >

[3] Craig Buckler, "JavaScript Threading With HTML5 Web Workers", *SitePoint*, 10 Dec. 2010. <http://www.sitepoint.com/javascript-threading-html5-web-workers/>

[4] Robert Gravelle, "Introducing HTML 5 Web Workers: Bringing Multi-threading to JavaScript", HTMLGoodies, <http://www.htmlgoodies.com/html5/tutorials/introducing-html-5-web-workers-bringing-multi-threading-to-javascript.html>

hardware, displaying the amount of fuel dispensed, and so on. Without threading this sort of application would be complex and cumbersome—so much so that almost all of the functionality would likely have to be pushed down into underlying C/C++ services. With JavaScript threading support, however, most of the user interface can be implemented within the HMI, providing an appropriate level of decoupled development.

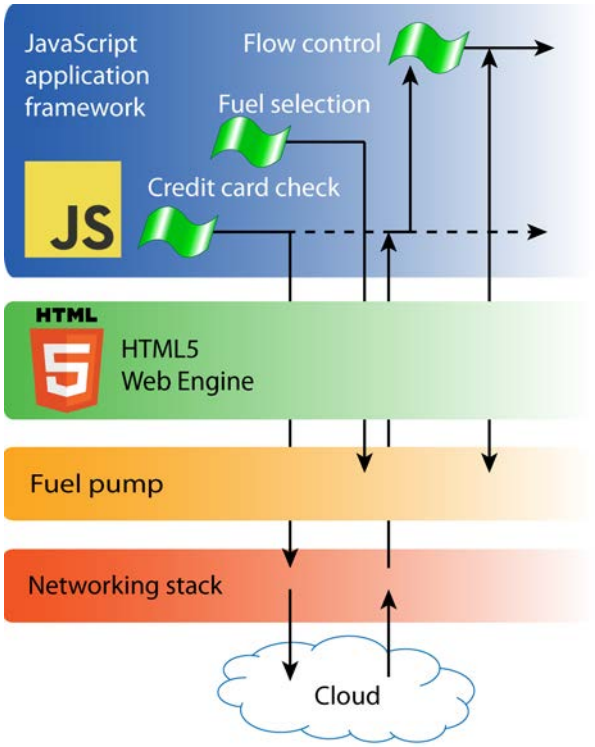

Figure 2. JavaScript Web Workers threading at the start of a transaction in a fuel dispensing pump application.

Multimedia

HTML5 establishes non-proprietary specifications that all content-creation tools and browsers are supposed to respect, thus doing away with most of today's requirements for plug-ins. With new HTML tags such as `<audio>` and `<video>`, developers can drop multimedia content into an HTML page just like an image, and users should no longer be ambushed by required updates for the likes of Flash Player, QuickTime and Silverlight. Anything inside these tags should simply run, providing the system provides the appropriate codec support at the native code level.

Though the expectation is that most audio processing will continue to be handled by lower-level code, such as C or C++, the HTML5 standard allows for JavaScript to perform audio processing and synthesis directly.[5] Giving the top-level developers direct access to audio simplifies HMI development and greatly increases developer efficiency. Performance in JavaScript will never be equal to that in native processing; however, depending on the platform, audio file and complexity of the processing, for some use cases it may very well be a viable alternative.

Unfortunately, HTML5 video specifications are as yet not as far along as audio. The W3C has specified elements and attributes (see its entertaining demo with the Big Buck Bunny movie)[6] but, with various browsers pulling in different directions, no consensus on a standard video codec has yet been reached. The

[5] W3C, "Web Audio API: W3C Editor's Draft", 2012. <https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html>

[6] W3C, "HTML5 Video Events and API". <http://www.w3.org/2010/05/video/mediaevents.html>

strongest contenders appear to be H.264, Ogg and WebM, but the debate is still far from over, and b) everyone seems to have an opinion.

Speech interface

The HTML5 Speech Incubator Group has proposed a standardized interface that allows JavaScript to talk to an underlying speech engine. This interface supports both speech recognition and synthesized speech output, and is designed to allow a "web application author … to add speech to a web application" using familiar methods.[7] Thus, assuming that the underlying system supports a speech engine, a designer building an HMI that includes speech recognition and output only needs to know the HTML5 interface.

Device Interaction

In a nod to the migration of HTML implementations onto mobile platforms—from smartphones to in-vehicle systems, HTML5 offers APIs that provide access to device information such as orientation and geo-location. This information can be used to adjust HMIs based on screen orientation, and for anything from tracking a taxi through its navigation system, to adjusting the time zone on a mobile phone—provided the device has GPS or accelerometer chips, of course.

Rendering control

One of the advantages of HTML pages and browsers is their forgiving nature. Browsers were originally designed to do their best to display what they're served, and ignore what they didn't understand. If a specified font is missing, the browser substitutes another; if an element or attribute is unknown, it is omitted, and so on.

The downside of this tolerance is that no two browsers display content exactly the same way. This is rarely a significant issue with traditional web content: text, images, forms, the occasional embedded video or audio. However, it is a serious impediment to using HTML for displays where precision is required. The location of a ball or brick or whatever being thrown in a video game must be *exactly* controlled. It can't be left up to the browser to do its best and place the object more or less where the game designer intended it to go. Similarly, maps must be precise, as must medical imaging such as, say, a 3D rendering of a mammogram.

HTML5 introduces new features that allow HMI designers to do things such as render images and control displays down to the individual pixel, as well as to safely bring together content from different sources. In addition to offering precision, these features can often take advantage of hardware acceleration.

Canvas

The HTML5 `<canvas>` element sets aside one or more screen areas where JavaScript can be used for precise rendering of shapes and images. With this element, no special code in required to detect the browser, and serve up the appropriate code for that particular browser flavor and edition, so there are no unpleasant surprises. As long as the browser conforms to the HTML5 standard, inside the area specified by the <canvas> `height` and `width` attributes, bitmap images render exactly as specified. Hence, in a game for instance, a

[7] W3C, "HTML Speech Incubator Group Final Report", 6 Dec. 2011. <http://www.w3.org/2005/Incubator/htmlspeech/XGR-htmlspeech-20111206/>

brick will fall precisely where it is supposed to fall. It will not fall on the left foot if it was supposed to fall on the right foot, due to some small difference in browser behaviors.[8]

It is important to note that the `<canvas>` element also imposes design demands. If the application output moves to a different device (for example, from a tablet to an in-vehicle infotainment system) the browser will not automatically adjust a display drawn on a canvas to a new aspect ratio—this means that the application developer must consider it and implement it if required.

WebGL and SVG

SVG (Scalable Vector Graphics) offer relatively light-weight 2D graphics rendering. And designers can now use JavaScript with WebGL to access OpenGL ES 2.0 rendering to display 3D images. HTML5 and JavaScript can thus be used to present any sort of image or animation called for by the application.

Sandbox (nested browsing)

Composite screens—screens made up of frames from different sources—have usually been something to avoid. Allowing the browser to render content from multiple sources left the door ajar for security breaches. For example, if multiple content sources were permitted, a malicious site could lay a hidden page over a legitimate page and use it to gather banking or medical information.

HTML5 lets HMI developers bring content from different sources onto a single HTML page and, critically, manage permissions for this content. In particular, the `<iframe>` element now has a `sandbox` attribute that controls content permissions. Settings allow the frame content a range of permissions, from only display content, to do everything the containing document is allowed to do, including executing scripts and submitting forms. The default is the most restrictive and safest: display only.

An HTML5 HMI can thus bring together onto a single display page, say, a map, navigation instructions, the weather report, points of interest, and even content provided by the point of interest. With the external, unverified content restricted to a sandbox, there is no concern about malicious content.

Better programming model

Despite the large number of new and very useful features it introduces, arguably the most important change with HTML5 is not part of the specification, but the way that the development community is reacting to application development using HTML5. Since HTML5 is becoming a more popular choice for application development outside of traditional web browsers, the tendency to keep application logic and display styles out of the DOM is becoming increasingly popular, bringing over the heavily used MVC— Model, View, Controller —design pattern into the HTML world.

Semantics tags

Like earlier editions of HTML, HTML5 is inspired by SGML and has inherited some of its syntactic features; for example, it maintains the `<!doctype>`

[8] See Mark Pilgrim, “Dive into HTML5” for clear how-to information, with a charming web page design to boot. <http://diveintohtml5.info/canvas.html>

element. However, HTML5 no longer refers to an SGML Document Type Definition (DTD)[9] and it incorporates into its own syntax new elements (`<section>`, `<header>`, `<article>`, `<aside>`, `<time>`, etc.) that, like SGML elements, define the content type rather than its presentation, which is left to the CSS.

Anyone who has tried to mark up documents using `<div>` tags will appreciate the change. HTML5 is not attempting to replace SGML or XML for document content management; rather, by adding a limited set of elements identifying content type, it simplifies page mark-up and content manipulation.

Document Object Model (DOM)

Document Object models are a mechanism HTML5 offers for controlling the HMI by controlling, not the specific page but how it is defined. The best definition of the document object model (DOM) comes from the W3C itself:

> The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page.[10]

JavaScript is used to manipulate the DOM and control pages. Scripts change the DOM based on what the user is doing; these changes are picked up by the web-rendering engine, which displays or hides parts of these pages, as defined by the altered DOM.

The HTML5 improvements allow much easier and consistent manipulation of the DOM. This means that application code becomes simpler to write, and identical across multiple platforms.

Cascading Style Sheets (CSS3)

Cascading Style Sheets have been part of HTML design for a long time. With HTML5 they are confirmed as the primary method for defining how the HTML page is displayed. The HTML determines what is displayed, and the style sheets determine how it is displayed. As long as the devices have appropriate style sheets, applications can be designed once and move easily between devices. The style sheets ensure that applications display correctly on each device, even branding them, or filtering out content such as animations or other distractions when the application moves, say, to an in-vehicle system in which driver distraction is a consideration.

None of this is entirely new to CSS3. However, CSS3 adds, literally, another dimension to cascading style sheets: time. It introduces new, dynamic capabilities, such as 2D and 3D transforms, and transitions, which cause an object to gradually change from one style to another. In keeping with the HTML5 philosophy of keeping the details of how something should be displayed separate from the actual content, these styles permit generalized manipulations, rather than requiring special code for each instance.

---

[9] W3C, "HTML 5 Reference: A Web Developer's Guide to HTML 5, W3C Editor's Draft", 23 March 2009. <http://dev.w3.org/html5/html-author/>

[10] W3C Architecture Domain, "Document Object Model (DOM)", <http://www.w3.org/DOM/#what>.

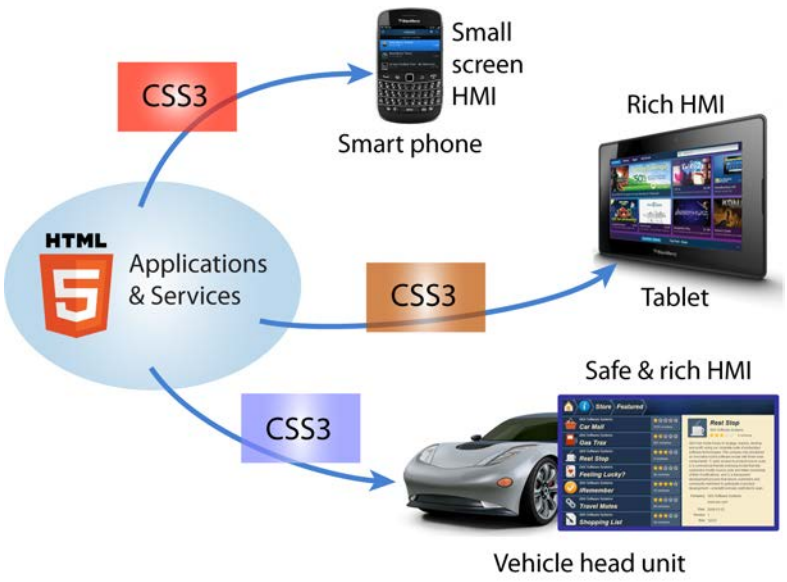

Figure 3. Cascading stylesheets used to facilitate migration of apps between devices

Unfortunately, to date, not all browsers support all the CSS animation styles, and those that do support them require slightly different code in the style; for example, a prefix in front of the `transition` style: `-moz-transition`, `-webkit-transition`, etc.

Conclusion

Taking off in 2011, HTML5 adoption has surpassed expectations. HTML5 is rapidly becoming, not just a standard, but *the* standard, not just for web pages, but for all types of rich user interfaces. Offering simplicity of use in a familiar environment and capabilities previously supported by disparate and incompatible, proprietary technologies, HTML5 appears to be delivering on its promise.

About QNX Software Systems

QNX Software Systems is the leading global provider of innovative embedded technologies, including middleware, development tools, and operating systems. The component-based architectures of the QNX® Neutrino® RTOS, QNX Momentics® Tool Suite, and QNX Aviage® middleware family together provide the industry's most reliable and scalable framework for building high-performance embedded systems. Global leaders such as Cisco, Daimler, General Electric, Lockheed Martin, and Siemens depend on QNX technology for vehicle telematics and infotainment systems, industrial robotics, network routers, medical instruments, security and defense systems, and other mission- or life-critical applications. The company is headquartered in Ottawa, Canada, and distributes products in over 100 countries worldwide.

www.qnx.com