HTML5, CSS3, and related technologies
A rapid-fire guide to new and emerging web standards

Rob Larsen
April 26, 2011

Many technologies are informally lumped under the "HTML5" banner. This article helps clear up any confusion about what's in and what's out of the HTML specification, while shining light on the technologies themselves, such as the many CSS3 modules.

Web standard development and marketing

It's a great time to be a web developer. After a long period of hibernation, the standards bodies and browser vendors have been extremely busy over the past few years, generating a torrent of exciting technology. Developers are greedily seizing on this work, producing demos and full-blown applications at a steady pace. Fed by this activity and further boosted by the growth of their standards-capable mobile browsers, companies like Google and Apple are using these new standards to market their products and services. The wider press is also seizing on this wave and pushing standards hype well beyond the normal circle of web developers and browser vendors.

This volume of discussion has obvious benefits, of course. People getting excited about web standards is a positive development for everyone in the industry. From that perspective, the persistent use of blanket terms, especially HTML5, as a sort of brand shorthand for "emerging web technology" is a useful shortcut. It allows nontechnical people to grasp—in a generalized way—the exciting work being done in the standards space right now.

Interestingly, even the W3C has gotten into the act, using HTML5 and its associated logo to publicize the "web platform."
On the downside, the volume of the specification work being done, coupled with the fast-and-loose labeling has created a bit of confusion, even in the developer community, surrounding the specifications themselves. Unless you're paying close attention to the volume of work, there's a real chance you'll get lost in the acronym soup of standards being generated.

This article helps to clear up any confusion surrounding the ongoing standards work. It outlines the major standards efforts and provides a handy guide to the technologies themselves.

**HTML5, "HTML5," and HTML: the living standard**

The most important of the specifications is, without a doubt, the latest iteration of HTML. For better or worse, "HTML5" has become a catch-all phrase for all emerging technologies. JavaScript developer Peter Paul Koch succinctly described the phenomenon in a blog post from January 2010: "HTML5 is anything you want it to be as long as it's new and cool."

Although that sentence describes the sentiment of a lot of people and captures the essences of the HTML5 "brand" that has developed over the past couple of years, there is a need for people who build websites to move beyond blanket terms and dig down into the actual technical specifications that are driving the hype.

As a note, the Web Hypertext Application Technology Working Group (WHATWG)—the working group driving the current HTML standards effort—has moved away from what it calls the "snapshot standards development process," long practiced by the W3C. Although the group is working closely with the W3C to create the HTML5 snapshot specification, its own standards efforts are focused on simply progressing HTML as a "living standard."

Naming semantics aside, let's step through the technologies actually present in the HTML5 snapshot specification. For a full, deep dive into any of these documents, refer to the Resources section, which contains links to this and all other standards documents referenced in this article. For this specification, be thankful for the web developer-specific version recently released by the WHATWG; it removes several hundred pages of information destined for browser vendors, making for a much more easily digested document.
Technologies in the HTML5 snapshot specification

Before diving into the components, it's worth noting that the WHATWG isn't kidding when it refers to this as a living standard. It's a moving target. Some technologies, like the Canvas 2D API, have started in the core specification and have later been moved into their own documents.

New semantic elements

Of all the new pieces of the specification, the elements being adopted most rapidly are the new semantic elements, like Header, Footer, Section, and Aside. Many were based on common usage patterns found during a web census conducted by editor Ian Hickson. This is clearly seen in Header and Footer, which echo the common id="header" and id="footer" patterns found across the web. Others, like hgroup, aside, and figure, were logical additions and enhancements to the existing elements of HTML.

Although specific browser support for the new elements is limited (browsers don't do anything special with a header, for example), the use of the new semantic elements is becoming quite common. Developers are working through best practices and creating common patterns for utilizing the new elements. For example, although older versions of Windows® Internet Explorer® have some trouble with unknown elements, libraries like Modernizr or the HTML5Shiv snippet smooth the way for cross-browser use of the new elements.

New outlining algorithm

To go along with the new semantic elements, a new outlining algorithm is in place to generate more flexible document outlines. In addition to the standard header elements H1 through H6, HTML5 introduces several new sectional elements, such as section, article, aside, footer, header, and nav. These elements break the outline of the document into a new section. Coupled with the familiar header elements and the ability to add more than one H1 element on a page, there are now much more powerful options for creating semantically rich, structured documents.

Using a tool like the HTML5 outliner (h5o) bookmarklet during development is helpful for getting a handle on the new algorithm. The new options can take some getting used to, and seeing a document broken out in outline form throughout the development cycle can bring clarity to the issue.

Video and audio

Playing audio and video in the browser is such a common event on the web that it's easy to forget that, for most of the history of the web, there was no native method for doing so. Enter the new HTML5 audio and video elements.

From a specification perspective, the inclusion of browser-native APIs for playing audio and video is straightforward and well-designed. Anyone familiar with the way the replaced elements like IMG work now will understand how to embed video and audio. And the associated API is simple, as illustrated in Listing 1:

Listing 1. A simple video example

```html
<-- the HTML -->
```
Unfortunately, getting to the point where the above code works in all major browsers with a single video source is still a long way off. With browser vendors facing off on two sides of the code divide, it's far more complicated than it should be. Until we move away from Apple and Microsoft standing firmly behind the patent-encumbered h.264, versus Google, Opera, and Mozilla backing free, open, and royalty-free video formats like WebM, video on the web will remain more, rather than less, complicated than it was in the Adobe® Flash®-only era.

**Form elements and input types**

HTML5 provides several new form elements to better reflect common input tasks. Formats like Email and URL can now be indicated to the browser in a meaningful way. Completely new form inputs include range (for a slider), date (for a date picker), and color (for a color picker).

Unfortunately, as research by Peter Paul Koch shows, browser vendors other than Opera and Research in Motion have been slow in adopting these technologies. This is a shame.

Although elements like canvas are certainly flashy and "cool," a lot of websites make their money in forms. For them, better forms mean better sites. It would be a real benefit to see some of these enhancements in front of users.

**Offline Web Applications**

Offline Web Applications defined an application cache manifest that allows a site or application developer to specify a set of files needed to run the application without a network connection. This specification is combined with the data-storage capabilities of the web storage specification to allow for some powerful offline capabilities.

**Inline content editing**

HTML5 defined a new API to ease content editing. Although there are novel JavaScript implementations, a native API should help present a more consistent, effective user experience.
Drag and drop

The new drag-and-drop API allows for drag-and-drop file uploads from the desktop to the web. A live example of this functionality can be seen in production with Google Gmail, in both Mozilla Firefox and Google Chrome.

History API

One of the consistent usability issues of the Ajax era is the way Ajax-heavy applications break the standard history stack. Although there are script-based solutions and common schemes available now to handle dynamic state, the HTML5 history API is a welcome improvement over ad hoc solutions. HTML5 adds onto the previous ability to navigate the history by including a method to add entries to the browser history and respond properly when the user clicks the back button.

Microdata

HTML5 defines a standardized method for marking up metadata in the body of an HTML document. Microdata will be familiar to people who have worked with microformats like hCard and hCalendar, with some important differences. Most importantly, microdata moves the pertinent information off of the classes previously hijacked by microformats and onto a new `itemprop` attribute.

In addition, the standard adds other attributes that more precisely define the format of microdata and a method that allows for more straightforward programmatic access to microdata:

- `itemscope` sets the scope of a microdata segment.
- `itemtype` provides a URL that defines the specific microdata format in use.
- `document.getItems()` provides access to top-level microdata items; the method returns a `NodeList` containing the items corresponding to an optional `itemType` argument or all types, if no argument is provided.

Technologies lumped into HTML5 that are in separate specifications and working groups

Several technologies have graduated out of the HTML5 specification into their own documents. Others, always separate from the HTML effort, have been gathered under the "web platform" banner.

SVG

Probably the strangest technology to be roped into the HTML5 catch-all is Scalable Vector Graphics (SVG). SVG is a vector graphics grammar defined in XML. The SVG specification has been under development by the W3C since 1999, so including it as either "new" or part of HTML5 is a stretch.

Still, newfound excitement for SVG is justified, as there's now some real traction for the standard. Some level of support is now available in the latest versions of all the major browsers and an API for older Internet Explorer versions presented by libraries like Raphael.js.
Canvas 2D context

One of the earliest stars of the HTML5 era, the Canvas element and associated API, started life as an Apple extension to HTML. As mentioned earlier, it started the standards process as part of the main HTML5 specification. It now lives at the W3C in a separate document.

The Canvas 2D context provides a scriptable interface for drawing two-dimensional images and bitmaps in the browser. It’s already been leveraged for everything from charting libraries to game engines, a popular system for delivering custom fonts, and even a port of the Processing programming language into JavaScript. Thankfully for those looking to explore Canvas now, there’s a level of support for legacy versions of Internet Explorer provided by the ExplorerCanvas library.

Geolocation

The Geolocation API is a standard interface for retrieving the geographical location of a device. It provides a `window.geolocation` object that in turn provides methods that figure out a device’s location through the use of location information servers. Location information is pulled from a variety of sources, including IP address, device GPS, Wi-Fi and Bluetooth MAC address, radio-frequency ID (RFID), and Wi-Fi connection location.

Web Storage

The Web Storage specification defines an API for persistent data storage of key-value pairs in web browsers. This specification is similar to, but greatly improves upon, the functionality currently offered by cookies.

Storage takes two forms: `sessionStorage` and `localStorage`. Each provides similar methods for managing items (`setItem()`, `removeItem()`, and `getItem()`) and for clearing the entire storage (`clear()`). Session storage is designed to hold information for just the current browsing session. Local storage is meant for longer-term storage of site preferences or other user data. There’s also a storage event that can be listened to, for purposes of monitoring and reacting to storage activity.

For those interested in experiencing this functionality now, the Persist.js library offers a solid, cross-browser solution for using web storage or web storage equivalents in all major browsers.

Two other specifications go hand in hand with Web Storage:

- **IndexedDB** is a related but green specification that offers even greater opportunities for long-term data storage in the browser, including the ability to query the database and, importantly, the ability to store complex objects, not just simple strings.
- **Web Workers** is an incredibly powerful specification that defines an API that allows application developers to spawn background "workers" running scripts in parallel to the JavaScript on their main page. This functionality allows developers to offload lengthy processing tasks to background processes, freeing up the browser for continued interactions in the main context.
File API
This specification provides an API for working with files in web applications. Coupled with several other emerging and established technologies, like XMLHttpRequest, drag and drop, and Web Workers, the File API will allow for much more powerful interactions between the web and the desktop than are available today. Instead of a simple file upload input element passing the file to a web server for processing or a complicated Flash interface, the File API will allow direct access to file data to the browser.

WebSocket
The WebSocket API is designed to allow for bidirectional, simultaneous communications between a web browser and web server over a single TCP socket. WebSocket should actually be far along in the implementation game, with support in the latest versions of Firefox, Opera, Chrome, and Apple Safari, but a security vulnerability discovered means that support is now disabled by default in Firefox and Opera.

Server-sent Events
This specification defines an API for opening an HTTP connection for receiving push notifications from a server in the form of DOM events. This specification reverses the current pattern of polling a server at regular intervals for updates, saving countless unnecessary requests and the associated processor time and bandwidth.

WebGL
The Web-based Graphics Library (WebGL) enhances JavaScript with the ability to create interactive, three-dimensional graphics in the browser. WebGL is a context of the canvas HTML element. The specification went to Version 1.0 on 3 March 2011 and is managed by the nonprofit Khronos Group.

XMLHttpRequest Level 2
The XMLHttpRequest Level 2 specification enhances the core XMLHttpRequest object with new features. The most interesting of these is probably Cross-Origin Resource Sharing, which is a safe way to get around the same-origin security policy that prevents XMLHttpRequest from interacting with a third-party server. Currently, XMLHttpRequest can only communicate with the same server on the same protocol.

With the prevalence of JSON and JSONP, cross-origin sharing may not have the same urgency it once did, but it will still open many possibilities in site and mash-up architecture that were previously blocked by the same-origin policy.

The modular approach of CSS3
For the latest version of CSS, the working group decided to move away from a single, monolithic specification and divided the work into several separate documents called modules. Each module adds a new capability or extends a features defined in CSS, Version 2 (CSS2).
Several modules are mature, having been in development for 5 to 10 (or more) years. These are either proposed recommendations or candidate recommendations in the W3C's completeness hierarchy. That means they're basically done. Let's take a look at some of the more interesting and important modules.

**Selectors**

CSS doesn't work without selectors. Selectors are used to target elements in an HTML document in order to style them.

This module contains the existing selectors from CSS, Version 1 (CSS1) and CSS2, and extends them with new features. The additions focus primarily on a group of structural pseudo-classes like `E:empty`, which matches an empty element, and `E::nth-child(n)`, which matches a specific index $n$ child of an element. Other new selectors include a negation pseudo-class `E:not(s)`, which negates a simple selector, and the UI element states pseudo-classes like `E:enabled` and `E:disabled`, which match elements that are enabled or disabled.

Users of popular JavaScript libraries like jQuery, Dojo, and YUI will be familiar with the new CSS, Version 3 (CSS3) selectors, as they have been included in the libraries' CSS selector engines for some time now.

**Color**

Color specifies the color-related aspects of CSS, including transparency and notations for color values. The new color values include Red Green Blue + an Alpha channel (RGBA) and Hue, Saturation, Lightness + an Alpha channel (HSLA).

**Backgrounds and Borders**

Backgrounds and Borders defines the background properties, such as `background-color` and `background-image`, and defines the style of borders. New functionality in this module includes the ability to stretch a background image, images for borders, box shadows, and, in one of the most requested features in the history of the web, the ability to specify rounded corners on a box.

**Multi-column layout**

Multi-column layout allows developers to flow content into flexibly defined columns.

**Media Queries**

Media Queries is an enhancement of the `@media` rules of CSS and the `media` attribute in HTML, which adds parameters such as display size, color depth, and aspect ratio. This module allows developers to more specifically tailor content to different sizes and capabilities of devices within a class.

Media Queries play a strong role in Ethan Marcotte's Responsive Web Design, the subject of a popular *A List Apart* article and upcoming book.
New and in-development modules

There are exciting, practical, and eminently useful pieces of the mature modules, but much of the most exciting work being done in the CSS space is in a series of (relatively) newly proposed modules. Some of the most noteworthy examples follow.

CSS Fonts Module Level 3

Fonts Level 3 is an extension of the original and familiar font properties available from the dawn of CSS. Level 3 includes the popular \@font-face rule, which allows for much greater control of type on the web by granting the ability to directly embed custom fonts into the document using CSS rules.

CSS 2D Transforms Module Level 3

The 2D Transformations module introduces a new property that allows for the rotation, translation, scaling, and other transformations to a box.

CSS 3D Transforms Module Level 3

Developed in conjunction with SVG, the 3D Transforms module extends the 2D Transforms module with a perspective transformation.

CSS Animations Module Level 3 and CSS Transitions Module Level 3

The Animations module allows the assignment of animations to elements, specifying the properties to animate, the animation timing, and the units to change during the animation. The Transitions module defines a property to animate the transitions between pseudo-classes. A common example of a transition is when an element enters or leaves the \:hover state.

See the future

Figure 2. I've seen the future
That ends our tour of the current state of the web standards world. It really is an exciting time, especially because so many of these technologies are rapidly becoming available to work with today as demos or even production code. With developers busy documenting the quirks of these new technologies and creating libraries to fill in support for legacy browsers, opportunities abound to experience these new technologies first hand. That first-hand experience is unquestionably the best way to experience the future of the web. So, if you're interested in seeing where we're going, now is the time to dive in and see what all the fuss is about.
Related topics

- developerWorks Web content
- HTML5 specification for Web developers
- OpenGL

© Copyright IBM Corporation 2011
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)