InfoWorld

SPECIAL REPORT

NOVEMBER 2010

HTML5 Deep Dive

Deep Dive Articles

HANDS-ON

What HTML5 can do today

The major browsers already support a subset of the draft spec, so use it!

By Dori Smith

THERE'S BEEN LOTS WRITTEN ABOUT THE POLITICS AND PROcess of the emerging HTML5 specification (see "What to expect from HTML5" and "How HTML5 will change the Web" in this special report, as just two examples), but what working Web developers primarily want to know is: What can I do with HTML5, and when can I start using it? The good news is that there's a lot you can do with HTML5. The better news is that there's a lot that you can do with HTML5 today.

But first, one major caveat: You need to know your audience, though, of course, this is true whether or not you want to start using HTML5. If the majority of your site's visitors still use Internet Explorer 6, then you have no reason to rush. On the other hand, if your site is primarily for mobile browsers on iPhones and iPads, what are you waiting for? But if your site falls somewhere in the middle — as most do — here are some handy guidelines to ramping up to HTML5.

While you're rethinking your website to take advantage of HTML5 capabilties, go ahead and make your website mobile-friendly as well for iPhones, Androids, and more. Dori Smith explains how in the InfoWorld.com tutorial "How to make your website mobile today."

WHAT HTML5 FEATURES YOU CAN USE NOW

Although the HTML5 specification is still a draft being worked on by a standard committee, significant portions are already deployed in Apple Safari, Google Chrome, Mozilla Firefox (with more to come in Firefox 4), and Opera — and the forthcoming Microsoft IE9 will adopt much of the draft HTML5 specification as well. The When Can I Use site is a great resource, providing detailed breakdowns of what each major browser supports for HTML5 and related emerging Web standards.

Another site, The HTML5 Test, displays compatibility scores, based on the number of supported HTML5 capabilities (out of 300), for each browser (you need to visit the site in each browser you want scored). For the following versions, the scores are:

- Apple Safari 5.0: **208**
- Google Chrome 5.03: **197**
- Microsoft IE7: **12**
- Microsoft IE8: **27**
- Mozilla Firefox 3.66: **139**
- Opera 10.6: **159**

There's clearly a core of HTML5 features that all the major non-IE browsers do support, which could allow "draft HTML5" websites to be deployed to a large segment of the Web-using population.

Starting from the top. You can use HTML5's `doctype` now; there's no reason not to. You can even do a mass find and replace throughout your site, looking for (for instance):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
  Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
  strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

Which can be turned into:

```
<!DOCTYPE html>
<html>
```

Isn't that considerably clearer and more straightforward? If browsers rendered your pages as standards-compliant before, they will still do so afterward.

Get moving with video. Much of the press about HTML5's `video` tag has been about the current format battles. There are four competitors — Flash, H.264, Ogg, and WebM — all of which hope to be the format of the future, and none of which play in all browsers on all platforms. Sadly, it doesn't appear that browser vendors will agree on a common future format any time soon.

Given that news, it's perfectly reasonable to jump to the conclusion that the `video` tag isn't ready for prime time.

But wait: The bright folks behind HTML5 foresaw this and made `video` format-independent. In fact, because `video` can contain multiple `source` tags, it ends up working out rather well. If your browser doesn't support the first option, it tries again with the second, then again with the third, and so on. It's even valid to fall back to Flash and again to a single image.

The HTML needed to handle this can be found at Video for Everybody, an open source project to support Web-based video using no JavaScript and no browser sniffing.

Semantically speaking. One of the biggest changes coming in HTML5 is the use of semantically meaningful tags. Chances are, your site is full of tags like `<div id="header">` and `<span class="nav">`. HTML5 figures that when a preponderance of sites all have the same elements over and over, we should be using meaningful names like, well, `<header>` and `<nav>`. And of course, we should then use CSS (cascading style sheets) to style those elements.

But wait, you say: No version of IE has ever shipped with support for these elements (though IE9 will), and that's a huge chunk of people! Does that mean we're out of luck? Thankfully, there's a work-around here as well; all you have to do is paste this snippet into the head section of each of your pages:

```
<!–[if lt IE 9]>
<script src=
"http://html5shiv.googlecode.com/svn/trunk/
  html5.js">
</script>
<![endif]–>
```

HTML5 Shiv is an open source project based on a simple discovery: If you create a new DOM element in IE, you can then style any elements with that name. That is, once you create a new DOM element like this: `document.createElement("foo");`. You can then add any number of `<foo>` tags to your page, and IE will style them. HTML5 Shiv contains a list of all the HTML5 tags that IE doesn't (yet) know about and creates them one by one, allowing you to then use and style them to your heart's content. Here are a few of HTML5's new semantic tags to get you started: `article`, `section`, `header`, `footer`, and `nav`.

Smart forms. Another feature of HTML5 is smarter form elements. If you're tired of writing the same old scripts that check (again) to see if visitors entered valid email addresses (or valid telephone numbers, valid URLs, and so on), you aren't alone. It should be reasonable for browsers to handle the most common types of data entry, right? Right.

Here's the syntax:

```
<input type="email">
  <input type="url">
<input type="number">
<input type="tel">
```

What about older/lesser browsers? Here's the cute part: If they see a type attribute with a value they don't understand, they set type to its default value, text – which just happens to be what we'd want as a fallback for each of these.

HTML5-supporting browsers can validate each of these field types to varying degrees, but you'll still want to keep those validation scripts around, at least until IE9 is ubiquitous.

If you're wondering why bother changing if you'll still need those scripts, you've never filled out a Web form on an iPhone. If you have, you've noticed the keyboard change based on the type of input required: Email addresses get an `@` (at sign) up front, telephone numbers get a keypad, and so on. All you have to do to get that functionality is to change the `type` attribute on your input tags.

For even more smarts, there's a new attribute: `placeholder`. Its value is simply the placeholder text that you often see in Web forms, but now the browser handles it for you:

```
<input type="email" placeholder="Your email
  address">
```

The form field automatically clears itself when the cursor enters the field, and then redisplays the placeholder text if the field is empty when the cursor leaves.

WHAT HTML5 FEATURES YOU CAN USE SOON

Not all of HTML5 is ready for the spotlight, for a variety of reasons (and no, they don't all have the initials "IE"). Browser support is coming, though, and here are two elements that you'll be able to use in the not too distant future.

Fancy fonts. At one point or another, every Web designer has wished that they could force every computer visiting their site to install some favorite font. With the CSS3

`@font-face` property, you'll have that power. The holdouts here were Firefox (before version 3.5) and Mobile Safari (before iOS 4). If you have enough visitors with those browsers, you may want to hold off for a while.

However, there's no real reason why you should be trying to make every user agent render your site identically when you can just offer different fonts to different browsers. If you want to offer custom fonts to those that can handle them, with fallbacks for those who can't, you'll want to check Font Squirrel's @font-face Generator.

Shadows and curves. Further designer-pleasing features coming down the road are text shadows, box shadows, and radial corners on boxes. Again, if you can handle your sites not rendering in a pixel-perfect identical fashion on every browser, you could start using these today. Here are some examples, with help from the CSS3 Generator.

Rounded borders (Firefox 3+, Safari 3.1+, Opera 10.5+, Chrome 4+, IE 9+):

```
-webkit-border-radius: 10px;
-moz-border-radius: 10px;
border-radius: 10px;
```

Text shadows (Firefox 3.5+, Safari 1.1+, Opera 9.6+, Chrome 4+):

```
text-shadow: 5px 5px 3px #CCC;
```

Box shadows (Firefox 3.5+, Safari 3+, Opera 10.5+, Chrome 4+):

```
-webkit-box-shadow: 10px 10px 5px #666;
-moz-box-shadow: 10px 10px 5px #666;
box-shadow: 10px 10px 5px #666;
```

WHAT HTML5 FEATURES YOU'LL USE, SOMEDAY

This category consists of elements and features that Web designers and developers have wanted for many years. Unfortunately, it may be a few more years before there's enough real-world support to be able to use them.

Brilliant forms. I referred earlier to smarter forms, which makes these new form fields downright virtuosos — when they're widely supported.

A horizontal slider control allowing a user to move the slider to pick a number:

```
input type="range" min="0" max="100"
  step="2" value="50">
```

A color picker:

```
input type="color">
```

A date field (other valid values for type are `month`, `week`, `time`, `datetime`, and `datetime-local`):

```
<input type="date">
```

If you need something that doesn't match any of these fields, you'll be able to create your own with RegExp (regular expression) patterns. Here's one for credit cards:

```
<input type="text" pattern="[0-9]{13,16}">
```

And finally, a way to tell browsers that a given field must have an entered value:

```
<input type="text" required>
```

None of these tags yet have the cross-browser and cross-platform support they'll need to be usable, but when that time comes, you'll be looking forward to them.

Print-like layouts. Another CSS3 feature, which when fully implemented will add years to the lives of designers, is multiple column layouts. It's currently implemented only as test cases in Firefox and Safari.

```
-moz-column-count: 3;
-moz-column-gap: 20px;
-webkit-column-count: 3;
-webkit-column-gap: 20px;
```

Location detection. With the growing interest in location-based services such as Gowalla and Foursquare, it's useful for a browser to know where its user is. For obvious reasons, this was first implemented in mobile browsers, but Firefox 3.5 and Safari 5 have begun to provide support for geolocation. (Chrome's geolocation support is currently via Gears, which is in the process of being phased out in favor of HTML5.)

Working offline, with local storage. Keeping your data in the cloud is a great idea — until you find yourself without Internet access. Or maybe you have a Web app that works with large amounts of data, more than you want to be continuously shuffling back and forth to a server. Or perhaps you have a limited data plan on your mobile device, so you want to do as much work offline as possible. Whichever the issue, the answer is to use offline Web applications using local storage that sync back to the cloud when you reconnect.

The browsers that currently support HTML5's ability to work offline are Firefox 3.5+, Safari Mobile 3.1+, Safari 4+, and Chrome 4+.

Canvas, animations, and transformations. One of the promises of HTML5 has been the all-illustrated, all-animated Web, using the new canvas element and assorted

TABLE 1: PLANNED SUPPORT FOR SELECT HTML5 CAPABILITIES

Specification	Description	IE version	Firefox version	Safari version	Chrome version
`canvas`	Dynamic graphics	9	3	3.2	3
`canvas` text API	Displaying text on canvas	9	3.5	4	3
Transitions	Animating properties of elements (simple)	?	4	3.2	3
Animation	Animating properties of elements (complex)	?	4	4	3
3D canvas graphics/WebGL	Dynamic 3D graphics with hardware acceleration	?	4	5	5
Transforms	Rotate and scale elements	9	3.5	3.2	3
3D Transforms	Transformations through a third dimension	?	?	5	?

CSS3 properties. Table 1 shows where things stand as of November 5, 2010. Some browser makers have provided Web pages that update their HTML5 support plans regularly, including Apple Safari, Microsoft Internet Explorer, and Mozilla Firefox.

SHORTCUTS TO HTML5'S BLEEDING EDGE

If you're too impatient to wait for IE8 to die of old age, there are a number of ways to skip ahead — again, depending on what your site logs say about your visitors. For instance, Google's Chrome Frame (GCF) plug-in puts an instance of Chrome into IE browsers. If you know your IE-using site visitors are likely to have GCF installed, you can then force its usage with this small addition to the head of your pages:

```
<meta http-equiv="X-UA-Compatible"
  content="chrome=1">
```

That, plus a little JavaScript (provided by Google) to redirect users without GCF installed, is all you may need to work around IE's limitations.

The elements listed in this article are only a few of the ones currently included as part of HTML5 or CSS3. If there's a feature you just have to start using now, chances are good there's an open source project that is figuring out how to make it work for less-capable browsers.

Many of the media reports about HTML5 have focused on the politics, the "not until 2022" sound bite, or on HTML5's prospects as a "Flash killer." The reality of HTML5 is simply that it's the long-needed and long-overdue update to HTML4 — and you can start to implement it today.

Dori Smith is the co-author (with Tom Negrino) of "JavaScript & AJAX for the Web: Visual QuickStart Guide," 7th ed., and "Styling Web Pages with CSS: Visual QuickProject Guide."

NOTABLE HTML5 DEMONSTRATION SITES

Mozilla Bespin	An in-browser programmer's editor written using HTML technologies
YouTube	YouTube's HTML5 player offers experimental HTML5 support
Vimeo	Vimeo movies offer a link at the bottom for switching to an HTML5 player (Chrome, Safari, IE+Chrome Frame)
Merge Design	An HTML5 geolocation demo
Sticky Notes	A demonstration of HTML5 client-side storage
Wolfenstein 3D	Demo using the canvas tag (with how-to) — works in Firefox 3.6
ClouserW Soundboard	An HTML5 sound board showing off multimedia capabilities
Google Wave	Google Wave relies on HTML5 for some of its features
FreeCiv	A game implemented in HTML5

Note: Most of these sites require Chrome, Safari, or IE plus the Chrome Frame plug-in.

—Neil McAllister

STRATEGY

What to expect from HTML5

Support for the next generation of HTML is already appearing. Are you ready?

By Neil McAllister

AMONG WEB DEVELOPERS, ANTICIPATION IS MOUNTING FOR HTML5, the overhaul of the Web markup language currently under way at the Worldwide Web Consortium (W3C). For many, the revamping is long overdue. HTML hasn't had a proper upgrade in more than a decade. In fact, the last markup language to win W3C Recommendation status – the final stage of the Web standards process – was XHTML 1.1 in 2001.

In the intervening years, Web developers have grown increasingly restless. Many claim the HTML and XHTML standards have become outdated, and that their document-centric focus does not adequately address the needs of modern Web applications.

HTML5 aims to change all that. When it is finalized, the new standard will include tags and APIs for improved interactivity, multimedia, and localization. As experimental support for HTML5 features has crept into the current crop of Web browsers, some developers have even begun voicing hope that this new, modernized HTML will free them from reliance on proprietary plug-ins such as Flash, QuickTime, and Silverlight.

But although some prominent Web publishers – including Apple, Google, the Mozilla Foundation, Vimeo, and YouTube – have already begun tinkering with the new standard, W3C insiders say the road ahead for HTML5 remains a rocky one. Some parts of the specification are controversial, while others have yet to be finalized. It may be years before a completed standard emerges and even longer before the bulk of the Web-surfing public moves to HTML5-compatible browsers. In the meantime, developers face a difficult challenge: how to build rich Web applications with today's technologies while paving the way for a smooth transition to HTML5 tomorrow.

MODERNIZING HTML FOR THE RICH WEB

Rich applications and HTML have not always been a natural fit. The father of the Web, Tim Berners-Lee, envisioned HTML as "a simple markup language used to create hypertext documents that are platform independent." With the advent of XHTML, the pure XML formulation of the language, the W3C maintained this focus on Web pages as documents, with the proposed XHTML standards emphasizing such issues as document structure, compatibility with XML tools, and Berners-Lee's vision of the semantic Web.

This frustrated many developers who saw greater potential in the Web as an application platform. In 2004, representatives of Apple, the Mozilla Foundation, and Opera Software founded the Web Hypertext Application Technology Working Group (WHATWG), an independent Web standards consortium. Working outside the W3C, WHATWG began a parallel effort to revamp HTML for a more application-centric view of the Web.

In 2007, with its XHTML 2 work mired in seemingly endless debate, the W3C voted to adopt WHATWG's work as the starting point for a new HTML5 standard. By this time, even Berners-Lee had come around to the notion of an application-centric Web. "Some things are clearer with hindsight of several years," he wrote in 2006. "It is necessary to evolve HTML incrementally. The attempt to get the world to switch to XML … all at once didn't work."

That's not to say the concept of a pure-XML Web markup language is dead. Although HTML has retaken the lead role in the standards effort, an XML formulation of HTML5, to be known as XHTML5, is being developed at the same time. The difference is that while XHTML5 will be available for those who have already made the switch, developers will no longer be required to observe the rigorous syntax of XHTML to take advantage of Web markup's latest features.

HTML5: MARKUP GETS A MAKEOVER

Be that as it may, HTML5 has inherited many additions originally proposed for XHTML 2, including a number of features designed to improve document structure. For example, new HTML tags such as header, footer, dialog, aside, and figure allow content authors to specify common document elements in a consistent way. Previously, developers had to mark such elements using div tags with custom class attributes, an arbitrary method that made HTML documents difficult to parse.

HTML5 also continues the effort to separate Web content from presentation. Developers might be surprised to see the `b` and `i` elements available in the new standard, for example, but these elements are now used to offset portions of text in generic ways, without implying any specific typographic treatment. Where the `i` element once implied italic type, for example, in HTML5 it merely means "a span of text in an alternate voice or mood." Similarly, the `b` element does not imply specifically boldfaced text, but text that is stylistically offset without having any additional importance.

By comparison, the `u` tag, which referred specifically to underlined text, has been dropped from HTML5, along with other presentation-specific elements, including `font`, `center`, and `strike`. Such stylistic attributes are now considered the exclusive domain of CSS.

The new standard introduces additional data types for form input elements, including dates, URLs, and email addresses. Still other elements improve support for non-Latin character sets, including tags for specifying the "ruby text" that appears in some Asian languages. HTML5 also introduces the concept of microdata, a method of annotating HTML content with machine-readable tags, making it easier to process for the semantic Web. Together, these structural enhancements allow content authors to build cleaner, more manageable Web pages that play nicely with search engines, screen readers, and other automated content parsers.

ENABLING A RICHER, STANDARDS-BASED WEB

But the most eagerly anticipated additions to HTML5 are the new elements and APIs that enable content authors to create rich media using nothing more than standards-based HTML. Modern Web pages increasingly incorporate scalable graphics, animation, and multimedia, but so far these capabilities have required proprietary plug-ins such as Flash, RealMedia, and QuickTime. Such plug-ins not only introduce new security risks, but they also narrow the audiences of the resulting pages.

One way HTML5 solves this problem is by aligning itself more closely with related markup languages. Content authors can embed markup written in MathML (for rendering equations) and SVG (for rendering scalable vector graphics) directly into their HTML5 markup. This increased flexibility makes cross-platform HTML more competitive with file formats such as Flash and Silverlight, which were designed with both text and graphics in mind.

But Web developers are clamoring loudest for HTML5's new `audio` and `video` tags, which aim to finally make it easy to embed multimedia content into Web pages. These tags are defined in the HTML5 standard as being codec-neutral, meaning it's up to individual browser vendors to support the codecs needed to play any given content item. Still, the `video` tag in particular is expected to be a godsend, particularly for online video providers who want their content to be available on Apple's iPhone and iPad, neither of which supports Flash.

Taking interactive Web graphics one step further is the `canvas` tag, which can be used to define areas of the browser window as dynamic bitmaps. Web developers can use JavaScript to manipulate the content of `canvas` elements, rendering graphics in real time in response to user actions. In theory, this technique could allow developers to create fully interactive games using nothing more than JavaScript and HTML.

In addition to these onscreen technologies, HTML5 also introduces the concept of browser-based application caches, which allow Web applications to store information on the client device. Like the Google Gears plug-in, these caches can both speed up application performance and allow users to continue to use Web applications even when they do not have access to the Internet — in fact, Google is already planning to phase out support for Gears in favor of the HTML5 technology.

BROWSER PLUG-INS: NOT DEAD YET

But for all of HTML5's new features, users shouldn't expect plug-ins to disappear overnight. The Web has a long history of many competing technologies and media formats, and the inertia of that legacy will be difficult to overcome. It may yet be many years before a pure-HTML5 browser will be able to match the capabilities of today's patchwork clients.

For example, while Vimeo and YouTube are already experimenting with the HTML5 `video` tag, deploying HTML5 multimedia will not be as easy as it sounds. The W3C's decision not to specify media codecs in the HTML5 standard means developers cannot guarantee that any one media format will be playable on every possible client device. Apple, Google, and Microsoft are pushing for H.264 video, for example, but open source browsers such as Firefox and Konqueror are unable (or ideologically unwilling) to license the appropriate patents to support that format. Unless this boondoggle can be resolved, Web content authors who need to reach the widest possible audience may be forced to continue to rely on Flash.

Not every legacy application will be rewritten for HTML5, either. For example, even as Google downplays its own Gears technology in favor of a standards-based approach to local application storage, the Gears API and the HTML5 application cache API are not identical. Google itself admits that "there is not yet a simple, comprehensive way to take your Gears-enabled application and move it (and your entire userbase) over to a standards-based approach." Until there is one, even users of fully HTML-compliant browsers may be forced to install Gears to support some legacy applications.

In the end, browser market share may be the most significant hurdle for developers interested in making the most of HTML5. Internet Explorer 6, for all its rendering quirks and inept handling of Web standards, is seemingly the browser that cannot die. Older versions of Firefox, IE, Opera, Safari, and others all have large user bases, and none support HTML5. Until these legacy browsers are replaced with modern updates, Web developers may be stuck maintaining two versions of their sites: a rich version for HTML5-enabled users, and a version for legacy browsers that falls back on outdated rendering tricks.

In HTML5's favor, Apple's iPhone and iPad will not support Flash, but are expected to gain support for HTML5 features as the standard matures. Similarly, Google's Chrome browser leads the pack in HTML5 support, and devices based on the company's forthcoming Chrome OS are expected to follow suit. Large Web publishers, however, have traditionally been conservative about standards support; even given a large HTML5 installed base, it may be years before the Fortune 500 is willing to risk the upgrade.

HOW TO TRY HTML5 TODAY

Some voices among the Web development community also urge caution. Although Microsoft plans support for HTML5 in Internet Explorer 9, for example, the software giant questions the wisdom of claiming support at this early stage. "Saying you are standards-based but then saying you are the most HTML5-compliant browser does not make sense, because the standard is not [complete] yet," Microsoft's Steven Sinofsky remarked in an interview.

Indeed, no organization is more guarded in its estimates of HTML5 adoption than the W3C itself. The HTML5 working group does not expect the standard to reach Candidate Recommendation status — the feature-complete phase of the W3C standards process — before 2011. Even then, the process of ratifying the standard as a W3C Recommendation is expected to continue until somewhere around 2022. If you're doing the math, that's 21 years from XHTML 1.1 to HTML5.

By any count, HTML5 is likely to remain cutting-edge technology for the next five to 10 years. Early adopters who would like to see it in action today can do so, however, albeit in a limited way. A number of pilot projects and demonstration sites that showcase the various capabilities of the new standard are available online; the key is choosing the right browser. Support for HTML5 features in Firefox is spotty. Browsers based on the WebKit rendering engine, including Chrome and Safari, work best. Ironically, that means Internet Explorer is also an option — but only with the Chrome Frame plug-in installed.

Web developers, likewise, are free to experiment. Whole sites can be built with code that conforms to the current draft of the HTML5 specification, although results with current browsers will be spotty. One of the best online resources for would-be HTML5 developers is Mark Pilgrim's excellent Dive into HTML5, which includes, among other things, a detailed guide to navigating the complex world of the HTML5 `video` element and the various codecs supported by current browsers.

So much work remains to be done on the HTML5 standard, however, that some organizations are liable to dismiss it as yet another overhyped, up-and-coming technology. That would be a mistake. Standards bodies by their very nature move slowly, but work on HTML5 is being driven by large, motivated vendors, including Adobe, Apple, Google, Microsoft, the Mozilla Foundation, Opera Software, and others. These companies recognize the need for an upgrade to the HTML standard, and their work is helping to realize its potential. The resulting opportunities for Web developers are too compelling to ignore.

Contributing Editor Neil McAllister writes InfoWorld's Fatal Exception blog.

STRATEGY

How HTML5 changes the Web

The nine ways the impact of HTML5 will be felt most

By Peter Wayner

MANY FOLKS WHO TUNED INTO THE HTML5 SAGA BECAUSE of the battle between Adobe and Apple are surprised to learn that the push to create a fifth official version of the HTML specification began six years ago. And that's just the first half of the story because the latest implementations, while nice, are far from standards. The HTML5 demos from Apple, for instance, are impressive, but they only run well on Safari.

That's how slowly committees can work. The browser creators and other stakeholders have a big collection of ideas for improving the browser and the Web, and these are gradually coalescing into a fifth generation for the standard. But agreement takes time. Many of the new tags and JavaScript functions exist already as experiments on some of the browsers, but interoperability and standardization are still to come. That's why the Flash groupies joke about HTML5 being a time machine to take you back to 2000.

While the jokes may sting and waiting for more general adoption is tiresome, it would be a mistake to simply ignore HTML5. There are not only powerful companies behind it, but there's also the standard process of technological development. The software — both browsers and tools — tends to absorb all of the orbiting extras, incorporating them into the main standard.

"The battle over Flash may be the most famous skirmish, but the newer expanded powers of HTML5 also threaten other coding silos."

HTML5 will change many aspects of life on the Web. It will not displace Flash or Shockwave: One glance at the games on Miniclip.com, such as Jet Ski Racer, shows how much ground the HTML5 committee must cover. But HTML5 will still remake the Web and enable basic websites to do much more — from tracking our location to storing more of our data in the cloud. HTML5 tags will displace plug-ins for simpler jobs, at least some of the time, and it will open up advanced capabilities to a larger audience. It might even make the Web more secure, more efficient, and more adaptable.

To see where this new standard may take us, I collected the opinions from a number of developers, programmers, and designers. Here is an unordered list of ways that the Web may change as HTML5 is gradually adopted and standardized.

HTML5 WILL REDUCE THE IMPORTANCE OF PLUG-INS

Once upon a time the Web world liked the idea of a browser plug-in or add-on because it encouraged creativity and experimentation. Sounds, moving pictures, and other neat tricks appeared on the Web first through plug-ins built by Sun, Adobe, RealAudio, Microsoft, and many others. The plug-in interface was open to all, and everyone experimented with adding new features to the old, text-based world.

The battle over Flash may be the most famous skirmish, but the newer expanded powers of HTML5 also threaten other coding silos. JavaFX may be wonderful, but who wants to learn another syntax when JavaScript and the new HTML5 `canvas` object will do the job? Who needs the Real ecosystem when the `video` tag will synchronize audio and video? Plug-ins like these are destined to be forgotten.

Will the idea of a plug-in disappear or fall into disfavor? Perhaps, but it depends on what you want to do. If drawing images is your goal, then the `canvas` object may be powerful enough. But if you want to build specialized 3-D worlds like the ones found in the more sophisticated Flash and Shockwave games, you may be pining for the old days when a plug-in could get direct access to the video hardware or run a 3-D game world.

HTML5 WILL ENABLE MORE INTERACTIVE GRAPHICS

The old Web loaded images by downloading a GIF or a JPG file. The new Web can build an image on the fly in a canvas object. A number of good graphing libraries have appeared, and all of them make a website's graphics much more interactive.

Now the JavaScript layer can compute values and draw pictures with the data. Everything can become more alive and much less textual — if the developer has the time and talent to create the solutions. Adobe is just beginning to make it simpler to develop sophisticated graphics for HTML5. The emergence of such tools will unlock additional capabilities, and the sophistication of the graphics will only improve as the tools mature.

There is a legitimate danger that all of this sophistication will overwhelm the poor client-side processors. In the past, some developers deliberately disabled the Flash plug-in to avoid the headaches and overhead of rendering heavy Flash content. That won't be an option in the future. Everyone who's been complaining about Flash may learn that the troubles had little to do with the technology itself — the problems came from the designers battling for our attention.

HTML5 WILL ALLOW APPLICATIONS TO TAP LOCAL FILE STORAGE

Web programmers have always been able to store a surprisingly large amount of information in cookies (300 cookies of up to 4,096 bytes in IE), but to do real work you need more room. The early versions from the Dojo toolkit used the Flash plug-in to commandeer a section of the hard disk, but now the tools can simply use HTML5.

This storage can be used for anything the programmer wants, including undermining the entire cloud paradigm by storing data locally on the hard disk. This makes it possible to deliver and install applications that behave just like classic applications. Applications load their JavaScript code from the HTML5 offline application cache and start right up whether or not the Web connection is working.

The technique does not need to undermine the hard work of cloud proponents, though, because the local databases can act like smart caches. Game programmers might store descriptions and artwork locally, saving the time of downloading the information again and again.

On the downside, these databases are buried deeply in the system folder, so making backups may not be the simplest step. Users who may want to move their local data from machine to machine will pull out their hair. Or perhaps we'll just see a hybrid cloud/local approach appear where the local machine caches the data but the cloud maintains a definitive version that can be accessed from different machines.

HTML5 WILL SIMPLIFY SCRAPING WITH CYBORG DATA

Anyone who's scraped data from Web pages knows that the structure offered by HTML does little except tell the browser where to place the information. There's no insight into the data itself, something that would help a programmer make sense of the information. The so-called microformats in HTML5 provide a mechanism to introduce more sophisticated markup into the HTML that makes it easier to analyze the data.

No one can predict just how much change the microformats will bring to the Web, but it's easy to see how they will empower programmers to whip together solutions. If there's one nice, standard way to represent dates and times, for example, then programmers can knit together the time-related information from websites without bothering to write sophisticated parsers that guess at the format one person chose. Calendars, timelines, and schedules drawn from multiple sources become much simpler to craft.

HTML5 WILL ADD LOCATION TO THE MIX

To the Web server, we were once just IP addresses, relatively anonymous numbers that had only a rough correspondence to the real world. The HTML5 standard now lets JavaScript ask the browser for the latitude and longitude of the user. It typically doesn't work with a desktop system (GPS or Wi-Fi required), but it works quite well with handheld smartphones.

No one knows what clever programmers will create with this location information, but it's bound to integrate cyberspace with "meatspace," in unpredictable and amazing ways.

HTML5 WILL SMOOTH THE WAY TO WEB VIDEO

The HTML5 video tag makes it easier for Web developers to integrate video with the information on the rest of the page, opening up the bag of tricks to jQuery and PHP developers, not just Flash, Silverlight, or JavaFX magicians.

Despite this vision, there's little coherence, as everyone wants to be the ones distributing the codecs for unpacking the moving images and the corresponding sound. The HTML5 standard is codec-neutral, which means that we're replacing the old world where the add-on software was called a plug-in with a new world where the add-on is called a codec.

So there's a standard `video` tag, but the browser may or may not know how to interpret the data.

Erich Ocean, a HTML5 application development lecturer who teaches in Los Angeles, believes the codec wars are already won. "Computer programmers (and Mozilla) are fooling themselves if they think they can dictate video standards to video professionals," he said. "Google's new [WebM] format will see some usage, for example in YouTube, but will never reach anywhere close to the ubiquity of H.264."

Despite the confusion and the lack of complete agreement, the new `video` tag will unlock more of the power of video and make HTML less and less of a textual jungle and more and more of a video playground. It's too soon to stop teaching our kids to read, but maybe the handwriting — er, the Webcam video is projected on the wall.

HTML5 WILL PRODUCE CHATTIER WIDGETS

The widgets that run in iframes have enabled sites to embed information from other sites for years, but they've always been limited by the security boundaries that keep each widget in a separate sandbox.

HTML5 offers a standard mechanism for these widgets to talk with each other. They still won't be able to reach into each other's sandbox, but they'll be able to send messages back and forth, coordinating their work and maybe even gossiping about the person typing at the keyboard.

Advertisers will drool at the chance to coordinate the behavior of disparate rectangles scattered across the page, and developers will surely find other practical uses. For instance, a tennis tournament might synchronize players on the left and the right of the page, an effect that may be so maddening that some will go running back to HTML 1.0.

However, this mechanism for sending messages is just a start. There's still a need to set standards for the information that's passed, so widgets stand a chance of speaking to each other even when they haven't been developed with a specific conversation in mind. In other words, they need more of a standard vocabulary.

HTML5 WILL IMPROVE SECURITY (MAYBE)

Each browser plug-in is a separate program built by a different team of programmers with different standards, different release schedules, and different models for security. Naturally, some plug-ins are more secure than others. And as plug-ins proliferate they increase the complexity of keeping track of the security faults. Was it the plug-in or the browser that had that nasty hole at the end of last year? Was it fixed by updating the browser but not the plug-in or vice versa? Who can remember?

Replacing many plug-ins with features baked into HTML5 removes the dangers that any of these groups will make a mistake, or worse, that someone will use a plug-in API to deliberately install malicious code. If the security team auditing Firefox, Chrome, or IE does the job — granted, that's a big if — then the dangers will be fewer.

This claim of better security, though, is a bit of a wild guess. The devious minds may use their malice aforethought to take advantage of the nice integration, perhaps drawing PayPal logos with the `canvas` object from scratch to impersonate the PayPal site. No one can predict what the dangerous minds will discover in the new capabilities of HTML5.

HTML5 WILL SIMPLIFY WEB DEVELOPMENT

Bill Mill, a developer who works at Lookingglass Cyber Solutions, explains the change succinctly: "I mainly like HTML5 because it allows me to work in one unified environment, the browser plus JavaScript plus DOM, without having to switch back and forth between the Flash world and the HTML5 world. There is one language and one set of tools, not different ones for each plug-in."

He adds, "I think this is noticeable to the user too, where Flash blobs seem to exist in their own world within a Web page."

HTML5 offers one language (JavaScript), one data model (XML and DOM), and one set of layout rules (CSS) to bind text, audio, video, and graphics. The challenge of making something beautiful is still immense, but it's simpler to work with a unified standard.

Now, if only HTML5 came with the nice collection of tools that Adobe makes for Flash.

InfoWorld.com Contributing Editor Peter Wayner covers Web, mobile, and other development toos and methods for InfoWorld.

STRATEGY

The case for Flash over HTML5

7 reasons Web designers will remain loyal to Flash for rich Web content

By Peter Wayner

IN ONE CORNER IS ADOBE'S FLASH, THE ONCE-UNDISPUTED champion in delivering rich content to the glazed eyes of the easily bored public. In the other is HTML5, the once-poor content provider now sporting the number 5 after its name and eager to prove that its new muscle and artful moves will be more than enough to take over the marketplace.

A wide range of pundits and industry heavyweights have been handicapping the fight, heralding HTML5 as the new champ and calling Flash "old," "fragile," "insecure," or worse. The complaints are easy to understand and the new abilities of HTML5 are seductive. But is that enough to bet against Flash?

HTML5 duplicates many of the features that were once the sole province of plug-ins: local disk storage, video display, better rendering, algorithmic drawing, and more. Some of these features are available now in various forms, but the HTML5 spec is still labeled "draft."

Is the sudden interest in HTML5 and support from the likes of Google and Apple enough to win? The fight isn't over by any means. While Steve Jobs might have enough juice to change the outcome, neither the technocrats nor the programmers are the final judges in this bout.

The real battle is in the hearts and eyes of the artists who are paid to create incredibly beautiful objects in the span of just a few hours. The designers will make the final determination. As long as Flash and its cousins Flex and Shockwave remain the simplest tools for producing drop-dead gorgeous websites, they'll keep their place on the Internet.

Here is a list of seven reasons why website creators will stick with Flash for rich Web content, compiled by interviewing a number of artists who work with it every day.

REASON NO. 1: FLASH'S SUB-PIXEL RESOLUTION AND ANTI-ALIASING

Do you want a border around a block of text rendered by HTML and CSS? Well, you can choose between 1 pixel, 2 pixels, or *n* pixels. The spec does allow floating point numbers, but the numbers between the integers tend to be ignored or rounded off in a slightly different way by different browsers.

Flash not only accepts floating point numbers, but uses the same rules to draw lines, boxes, and images on all machines. Using sophisticated anti-aliasing and blending algorithms to render lines, Flash gives the illusion that screens have more precision than they really do. The eye can tell the difference, and that's why Flash sites look better.

This is gradually changing as browsers begin to support the Scalable Vector Graphics (SVG) standard and allow JavaScript programmers to mix it with `canvas` objects. But while SVG is impressive, it's still a long way from efficient, thanks to all of the extra characters the XML standard insists upon. Compression can help, but there's still something funny about the SVG-Whiz suggestion that the artists should "keep an eye on the generated code, and trim it down by hand where possible."

REASON NO. 2: FLASH BEATS CANVAS

A good way to get a feeling for the new `canvas` object in HTML5 is to play a few rounds of FreeCiv, a free, open source version of the classic game Civilization. The developers implemented all of the basic graphics routines using the HTML5 `canvas`, and they all work fairly well. Anyone who assumes that JavaScript's main purpose in life is to check registration forms in Web pages will be very impressed.

But anyone who compares the results to the better games based on Flash, Shockwave, or AIR will just be mildly amused. Many of the `canvas` products are impressive, but sometimes the code just doesn't work or pokes along. Some browsers are fast and some are slow. Some operations are quick on one browser and sluggish on another.

To make matters more complicated, not every browser implements every feature in exactly the same way, a problem that shouldn't be surprising to JavaScript developers. There are good efforts to simplify this with intermediate libraries like Processing.js, but even these can't handle every combination.

Flash isn't immune to the complexity brought to us by the proliferation of operating systems and browsers, but it has been dealing with them for much longer. When the Flash plug-in doesn't crash, the results are slicker, smoother, and more consistent.

But there are signs HTML5 will get there. For example, Smokescreen, which renders native Flash on Canvas objects, doesn't yet run on IE or Opera but is very impressive on other browsers.

REASON NO. 3: FLASH'S GOOD DEVELOPER TOOLS

The world of CSS, HTML, and JavaScript development has come a long way, but it's still hard to find an integrated development environment for producing sophisticated Web applications. There are solutions like JackBe and some decent Web-based tools, but these are mainly aimed at building business applications to be filled with data. They don't help you send flocks of sprites dancing across the screen, seducing the reader — or, should I say, viewer.

Adobe's tools such as Flash Builder and the endless range of design companions in the Creative Suite have been making this relatively easy for years. Sure, it's a mind-numbing array of applications, and many cost a fortune, but true artists manage to make them work.

The biggest competition comes from the AJAX libraries such as jQuery and Dojo that have been wonderfully fertile these last few years, integrating the efforts of a wide variety of contributors. While there are some tools that support these libraries, much of the labor is still done by hand-coding JavaScript.

One designer who asked not to be named because of his company's relationship with Apple, Adobe, and others said that while he felt Flash was "old," he considered Adobe's tools essential. "If Apple wants to kill Flash, it should start by building replacement tools," he said.

Apple may not need to do much. Adobe is hedging its bets and building HTML5 support into Dreamweaver so that you can continue to use Adobe's tools and enjoy the flexibility.

REASON NO. 4: FLASH'S SUPERCOOL FONTS

The world of fonts for the Web is getting better. There are more and more options beyond Verdana, and new frameworks such as the Web Open Font Format and Microsoft's WEFT look seductive. But they are still in their infancy. Flash lets designers embed fonts in their Web pages in a controlled way that makes it possible for font developers to support the Web marketplace.

Adobe has been employing professional font creators and marketing their work for some time. They've been friends of the font foundries, so they pave the way for someone to include a beautiful new font in the presentation without destroying the font creator's livelihood. The licensing may be confusing or complex — for example, ITCFonts lists four options — but the font designers generally reward Adobe with their best work.

Adobe has also been integrating technology from its other products. Algorithms from the fancy text layout engine in InDesign are now available in Flash when outline fonts are embedded into the presentation. (See Adobe's Text Layout Framework.)

REASON NO. 5: FLASH IS 'WRITE ONCE, PLAY EVERYWHERE'

This isn't exactly true. Steve Jobs might not have had a fit if Flash were bug-free on the Mac, but Flash is still a relatively easy way to distribute content to older and newer Macs and Windows PCs, as well as some versions of Linux. Adobe likes to call it "pixel-perfect fidelity across browsers and operating systems."

Jennifer Taylor, director of product management for rich media solutions at Adobe, says that while HTML is nice for content that must flow into different containers, Flash offers visual cross-platform stability.

"The challenge for HTML as a Web content delivery mechanism is providing a consistent display standard across a growing number of different browsers," she wrote. "This has been true from the onset of HTML and is still true with the most recent developments. So the productivity, expressiveness, reach and consistency (cross-operating system/cross-platform, and increasingly cross-device) of the Flash platform remain huge advantages for the Web community as HTML advances."

The Flash playback format (SWF) is open, and it's quite possible for others to create custom SWF files without using any Adobe tools.

This pixel-perfect nature, though, isn't necessary for all mechanisms. HTML users are quick to note that HTML can reflow into smaller screens and differently shaped windows with ease. Designers who specify layouts down to the pixel produce fragile work.

REASON NO. 6: THE FLASH COMMERCIAL ECOSYSTEM

Many users of the Adobe Creative Suite love the third-party plug-ins as much as the Adobe products themselves. Do you want to add slicker effects to your Flash presentation? Check out the numerous third-party commercial options available, such as FlashEFF.

The burst of interest in AJAX development has cut into this dominance. For example, these 30 photo viewers and modal dialogs built on AJAX can rival many created by the Flash community.

The AJAX work, though, is almost all open source — a great advantage for programmers willing to tinker with the code, but not always an asset for JavaScript artists who would like to keep their innovations to themselves. This is largely an accident of the Web architecture; while JavaScript code can be obscured through minification, it's still relatively easy to pirate. Flash effects are compiled into the SWF, making it much harder for anyone to borrow them.

REASON NO. 7: FLASH'S GAME ENGINES

Sure, JavaScript has "libraries," but Flash game developers have "engines." And while the differences between engines and libraries is often academic, what would a red-blooded programmer choose? When Bruce Springsteen wrote the lyrics to "Born to Run," did he ever pause for a moment, scratch his head, and consider asking Wendy to strap her hands around a library?

The platform that once nurtured reworked versions of '80s arcade classics now offers engines that deliver 3-D graphics, real-world physics, and the ubiquitous Facebook integration. Thanks to these ready-made systems, it takes no time at all to whip up a casual game. That may be why so many companies create games with the larger goal of advertising or education.

One website producer who may be indicative of where things are heading said that Flash by itself was a bit boring, and she kept looking beyond to the power of Shockwave and other enhancements.

"I preferred Director/Shockwave because I liked what we could do with it," she says. "The end result is more visually interesting, and the interactions are more intriguing. The animations are less repetitive because Flash animations are just repeating sprites."

She adds: "The developer I've been working with has a film background. He prefers Director/Shockwave, and his work is really creative."

These are the people whom HTML5 must win over.

InfoWorld.com Contributing Editor Peter Wayner covers Web, mobile, and other development toos and methods for InfoWorld.