Understanding HTML5 and its Potential for e-Learning and mLearning (May 10)

By Judy Unrein
May 31, 2010

What does HTML5 mean to us as e-Learning and mLearning designers and developers. What – if anything – does HTML5 add to our capabilities? What are its limitations? What will have to happen for it to be truly useful to our industry? And what don't we know yet?

The Internet is on fire. It's now genuinely difficult to follow technology news or commentary without reading something each day about HTML5 vs. Flash, or at least Apple vs. Adobe. Read one article and it's easy to come away with the impression that HTML5 is a "Flash killer". Read a different one, and you'll be convinced that HTML5 is just a flash in the pan.

There has been considerably less controversy in the e-Learning world – at least in the part of our discourse that gets published on the Internet. But the debate could have a pretty significant impact on our industry, if only because we're very heavily invested in Flash and development software that outputs to Flash formats.

According to the e-Learning Guild's 2009 report, Getting Started in e-Learning: Technologies, Tools, and Media for e-Learning, among the top 15 development tools used by their members, only two of the actual authoring tools *don't* output to Flash formats at all, and those are pretty far down on the list. The rest *primarily* publish to Flash formats, including Flash itself and the software in the #1 spot: Adobe Captivate. Go a little further off the grid (for example, to see Cammy Bean's mind map of which authoring tools people are using) and you'll see even more software, much of which outputs to Flash formats.

And it's easy to see why Flash delivery is popular: Some version of Flash Player, the plugin needed to play Flash content in a Web browser, is on just about every computer in the world. As e-Learning developers, we know it's only smart to meet our audience where it is, with the technology our learners already have. And anyway, Flash makes stuff pretty. It anti-aliases text and makes things move around onscreen. And not least, it gives us interactivity. Want to create branching scenarios, games, even virtual worlds? Flash can do that and much more.

So I think it's time we start taking a good look at this new technology that some say is a threat to an authoring program and delivery format that is pervasive in our industry. But I'm going to try to give you a break from what's happening all over the Internet, and not talk about whether Steve Jobs is an evil overlord, or whether Adobe is lazy about maintaining one of its most important products, or whether open source and Web standards will save us all. I'd like to talk about what HTML5 means to us as e-Learning and mLearning designers and developers. What – if anything – does HTML5 add to our capabilities? What are its limitations? What will have to happen for it to be truly useful to our industry? And what don't we know yet?

What is HTML5?

If you're not familiar with exactly what HTML5 is yet, here are a few definitions before we go further.

HTML5 is technically the latest version of the HTML specification, but it's commonly used to refer to the combination of three technologies together: HTML5, CSS3 (the latest version of Cascading Style Sheets), and JavaScript. Together, they are known as the HTML5 stack, and that's what "HTML5" will refer to in this article. HTML is the markup language, CSS determines how it is rendered, and JavaScript is the programming language; the combination of them provides the capability for a rich user experience similar to what we normally think of as the product of Flash or Silverlight. At the end of this article, I have provided links to examples that I hope will help illustrate what HTML5 can do where dry, technical

explanations fail – but remember, you'll need an HTML5-compatible Internet browser to view them (more on that later).

The features and elements of HTML5

HTML5 provides that aforementioned rich user experience primarily through the use of the new <canvas> and <SVG> elements, which can be placed in the HTML code much like the <image> tag has been used for decades. When a SWF is inserted in HTML, the browser needs the Flash Player plugin to be able to play it – and, for that matter, different plugins are needed to play anything output by Silverlight, Quicktime, or Authorware. These new elements won't require the use of plugins. In addition, HTML5 supports drag and drop – an extremely common interaction type in e-Learning development – within the code itself.

There is also a number of features that allow more data to be stored on the learner's computer, either temporarily or long-term. localStorage and sessionStorage allow for text to be stored locally by a browser either indefinitely (in the case of localStorage) or while that browser window is still open (in the case of sessionStorage). Offline Web applications are pretty much what they sound like: to quote the HTML5 Working Draft, they "enable users to continue interacting with Web applications and documents even when their network connection is unavailable." Like, when our users are on planes? Or submarines? Sounds good to me … as long as it works!

Two other new native elements are <audio> and <video>, which (I'm sure you've guessed by now) allow you place audio and video files directly in the HTML code. The browser then decides which method to use to play them, much like what happens today with images. Though supported video formats have undergone some controversy, the general consensus at this point is that HTML5-compliant browsers will be able to interpret H.264-encoded movies on their own, so movies don't need to be encapsulated as FLVs or MOVs (which, again, would require the learner to have the appropriate plugin).

The browser: if you build it, can they interpret it?

What learners will need to have, instead of a collection of plugins, is an HTML5-compliant browser. So it's important to note that right now, the HTML5 specification is not complete, and it is not expected to reach W3C Recommendation stage (the stage the current version of HTML is in) until at least 2022. Having said that, this oft-quoted statistic can be misleading; the real test of whether HTML5 is viable for you to use is whether your learners' browsers are able to interpret the new code. You may be surprised to find out that much of an HTML5-published page is already supported on whichever browser you're already using, because much of HTML5 encompasses technology that is already in use. So a course or Website that is output to HTML5 will be perfectly readable in older browsers *except* for any code that is not supported in previous versions of HTML. But even then, most of the HTML5 specification is already supported on the most recent versions of Firefox, Chrome, Opera, and Safari. Even Microsoft (late to the party, but very important due to Internet Explorer's market dominance) has announced HTML5 support for IE9.

That's a pretty impressive adoption rate for a major new technology, and it's being mirrored by more sites cropping up every day that now offer an HTML5 option in addition to the plugin-based technology they've been utilizing for years. And that "in addition to" is important: even though the debate is often framed as HTML5 vs. Flash or FLV vs. H.264, browsers have had the ability for years to customize which content you see based on which browser you're using to view the site. That's why you may get an entirely different version of a site when you visit on your mobile device instead of your desktop computer. And it can get far more granular than that: Sites that serve up video and aim to appeal to large audiences will have – or already have – different video formats available depending on who's visiting and what technology they're using. YouTube and Vimeo now offer an HTML5 option for most of their videos... again, only for users with compliant browsers.

Where's the beef?

So what, if anything, does this new technology enable us to do better?

The main reason I started getting interested in HTML5 was Flash's security issues. Okay ... to tell the truth, I don't care all that much about Flash's security issues; I care about the security restrictions that I have to work around in order to do my job, designing and developing e-Learning. I try to integrate my courses with existing resources as much as possible, and these resources are usually on my company's intranet.This practice shortens development and revision timelines because some of the course content is "outsourced" to external content, it makes my learners aware of standalone resources so that they don't have to retake whole modules, and it encourages exploration outside of the strict confines of the course.

Flash is a powerful tool; so much so that it can be used to impact your computer in malicious ways. So things like linking to external content – whether on the local computer or on a network – are limited. The same limitations exist when you integrate Flash content with HTML in a way that the SWF tells the HTML code to do something: for example, embedding a Captivate simulation in a Lectora title, and then putting actions in Captivate to make the navigation buttons in the Lectora title active when the simulation is completed successfully.

There are workarounds to the limitations, of course, but they involve tradeoffs in the final product, or more expertise, or better tools. For example, setting playback security can be done in Flash itself, but not in Captivate or in any other rapid authoring tool I'm aware of. The extra effort is worth it to me, because I try to make things *just work* on the learners' end as much as possible. But if I could get rid of expending that extra effort and still provide a product that is uncompromised in appearance or functionality, I would.

HTML5 doesn't have these security issues to be worked around... at least not yet. In a page displaying only HTML5 elements, all of the interaction the learner has with the browser are just that; there is no need to manage security of elements that are delivered via plugins. Granted, eventually there may be security restrictions for HTML5 as it gets more powerful (even Tim Berners-Lee expresses uncertainty about that), but from what I can see so far, there won't be a need for restrictions on the kinds of interactions used in even a rather robust e-Learning course.

And then there is openness. And I'm not talking about whether something is open-source or license-free, which is what a lot of the debate about HTML5 and related technologies has focused on. I'm talking about what happens when another branch of your company sends you an e-Learning module that your branch could easily adapt and reuse – if you only had the software and/or the native files originally used to develop the course. In an ideal world, the person sending the course to you would know that you need the native files, not the published files, and would have gotten the native files from the developer and archived them, and your two branches would be using the same development software as each other, and if any of these were untrue, your boss would understand and give you plenty of time to redevelop the course. But who among us develops e-Learning in that world? I'll wait... No one? Okay then, let's talk real life.

If what arrives at your door is all inside a SWF, you'll need a decompiler to get at it. What you get as a result of the decompilation is going to depend on which tool was used to create the SWF. Where you go from there depends on your expertise.

On the other hand, if you're given a collection of HTML files, minor edits can be made in Notepad or TextEdit, and they are usually pretty easy to do – even for novices – given some Find and Replace skills. If you can't figure out how to change what you want, you can hand the code to over to a vendor and at least they will have the right raw materials to work with. And that vendor won't have to be a specialist in e-Learning development software and how each tool compiles SWFs; pretty much any Web developer who works with current technology will do. So clearly there are some benefits to working with HTML5 in the development and revision cycle.

The case for HTML5 in mLearning

Once you include mLearning in the conversation, though, the advantages to using HTML5 get more significant. MLearning designers and developers – and mLearning itself – have suffered for years because developing mLearning is like trying to hit a moving target; once you have a product out, the devices it was designed for are obsolete, due to the very short lifespan of most mobile device product cycles.

If those developers were developing desktop-based e-Learning instead, they would have a pretty good solution to this compatibility problem: Flash. But Flash Player's ubiquity on desktops and laptops doesn't extend to mobile devices. Currently, there are no smartphones or mobile devices that run Flash formats, because Flash Mobile – the version of Flash Player that Adobe is working on for mobile devices – has been delayed several times. Recent demonstrations meant to show how close Flash Mobile is to market have been somewhat lacking, though there are other demos that show it working just fine.

On the other hand, as of this writing, almost all mobile devices on the market today have browsers that already support much of HTML5. It's a good choice for mobile devices because rendering HTML5 code is generally not very processor-intensive compared to alternatives we've seen so far, and as mobile device manufacturers push the envelope to accommodate user demand for increased speed and multitasking, it just seems smarter to develop content using technology that demands less bandwidth and processing power. And with HTML5, mobile devices' short product cycles become a good thing instead of a liability, as new devices generally mean new and more updated browsers.

But I think on some level, we all expect that Flash Mobile will be released, probably soon, and already there is no shortage of device manufacturers that want it on their machines. At that point, mLearning designers and developers will have to decide which delivery route to take based on Flash Mobile's performance and which devices their learners are using. And it might come into play that while Apple devices (iPad, iPhone, and iPod Touch) already support HTML5 very well, right now it looks pretty unlikely that they will ever support Flash. Or that might not matter… again, depending on your audience.

The final challenges

That brings me to the final challenges we will face in developing HTML5 content for both e-Learning and mLearning.

Browser adoption

The first challenge is browser adoption, which we've already touched on. This is an area in which you really need to know your learners and their technology; if you're developing for a corporation, you most likely have the advantage of knowing exactly which browser your learners will be using, but you may well also have the disadvantage of developing for an older browser than consumers would have. The older-browser issue may be problematic for mLearning developers who are working for corporations as well, because Blackberries and other RIM devices – generally more popular in corporations than other mobile devices – are behind other smartphones in terms of having a browser that supports HTML5.

Development software

The second challenge is development software. Rapid development tools are proliferating in the e-Learning world, and whether that's due to shrinking training departments or faster development times needed in a down economy or some other cause, it's pretty clear that few e-Learning designers roll their own code. We have software that does it for us, or we have developers who run that software for us, and most of that software outputs to Flash delivery formats. Right now, HTML5 is so new that if you look around Web development forums, you will see developers asking each other if there any programs already out that create HTML5 output. From what I can tell, there aren't many. And that leads me to

believe that it will be a while before we will see any such software that is specialized for creating learning experiences.

That is, except for these two considerations:

First, there *are* already specialized e-Learning/mLearning development tools that output to HTML; Lectora would be the one that currently has the most market share. The introduction of HTML5 (and, specifically, its prevalence on mobile devices) could be a tremendous opportunity for those software publishers who, despite a lot of pressure from their users, never rewrote their software to export everything to SWF.

Second, even if Flash were to go away next year as a delivery format, that doesn't mean for a second that Adobe would go with it. Adobe has Dreamweaver, which is already a premiere HTML development tool, and it has Flash, which is a premiere animation tool, and both of them are already very much in use in learning development. Adobe has already demonstrated HTML5-output tools for both programs, and while those features didn't end up shipping with CS5, the company has now announced an upgrade pack for Dreamweaver that will allow for HTML5 output.

And while neither Dreamweaver nor Flash is a specialized learning development tool, Captivate certainly is, and if Flash can be adapted to output to <canvas>, Captivate can too… if Adobe wants it to. And Adobe could have something else up its sleeve, as well, but whatever it is, it has vowed to make tools that produce HTML5 output, and it's clearly determined to stay in the game… even when the rules of the game change.

Summary

So after all of this, what is the future of HTML5 in e-Learning and mLearning? I'm not going to make any wild predictions, particularly because there have been so many made over the years about where e-Learning and mLearning should be already, and many of them have fallen flat. I think it's pretty obvious that we will see HTML5's prevalence increase on the Internet – both because it's a promising technology and because it's at such an early stage that pretty much the only place it can go is up. And when we see that our learners are ready, and we have the tools to provide the content, we will get there too.

But that's enough talk for now. I've collected the 20 examples below to show a few things that Web developers are already doing with HTML5, and I've tried to keep it relevant to the kinds of output we want and need in developing e-Learning and mLearning. You won't see any learning objectives below, but I think you'll be able to see a few ways HTML5 could be used in your courses and applications.

Remember, you'll need an HTML5-friendly browser to use these. I've tested all of the below in the latest versions of Firefox and Safari, and noted some that don't work as well in one browser or the other. If you'd like to explore other browsers, click here to see which ones support the different HTML5 elements. (Or click here for a cool graphical view.)

Twenty examples of HTML5 in use

1. HTML5 Canvas and Audio Experiment
   Chosen for: Visual impressiveness, interactivity. Click on the bubbles to see tweets about HTML5 and related technologies. Some tweets feature active hyperlinks from inside the <canvas> tag… a feature I'm particularly interested in.
2. Dynamic Content Injection
   Chosen for: Interactivity. Works best in Firefox 3.5+. Click any of the boxes on the right to add images, text, videos, etc. to the video on the left. The video changes dynamically as you change the input.
3. Bonadies Architect
   Chosen for: Visual impressiveness (at least for an HTML-only site). This reminds me of a lot of basic

Flash sites out there today. I think some of the functionality requires more clicks than it should, but it shows that you can make good or bad design choices on any platform.

4. Sketchpad - Online Paint/Drawing application
   Chosen for: Interactivity. This is an impressively full-featured sketch program.
5. Gartic - Canvas Sketch
   Chosen for: Interactivity. This is a much simpler sketch program, but it has a cool feature that allows you to reanimate your drawing process. Try the drawing game, too.
6. A simple drawing program using javascript
   Chosen for: Interactivity. Another sketch program, with a slightly different take.
7. Harmony
   Chosen for: Interactivity and visual impressiveness. This is the last sketch program, I promise... It's so cool I absolutely couldn't leave it out. Be sure to try different options at the top.
8. Function Plotter
   Chosen for: Interactivity. I may not ever teach math in corporate e-Learning courses, but I would definitely use graphs to show the results of different choices. And of course, you might use something just like this in an academic course.
9. Matrix Rain
   Chosen for: Visual impressiveness. Works best in Firefox 3.5+. It's just pretty.
10. HTML5 Canvas Animation
    Chosen for: Similarity to e-Learning staples. Just an animated cartoon, similar to some uses of Flash or Silverlight.
11. spread
    Chosen for: Visual impressiveness and interactivity. However, you can determine how the drawing evolves, so it's interactive, as well.
12. Chrome Canopy Ditto.
13. making waves with HTML5 Ditto again.
14. html 5 canvas test Ditto again.
15. Hypertree - Tree Animation
    Chosen for: Interactivity. I could see using this kind of functionality to let learners explore a map or org chart. Bonus: Here is another one, based on a similar idea.
16. ContentEditable
    Chosen for: Interactivity. This is a good, simple demonstration of editable text and localStorage; edit text in the box, close the window, and then click the link again.
17. Drag and drop
    Chosen for: Similarity to e-Learning staples. Because what would we do without drag and drop?
18. Internet Graffiti Board
    Chosen for: Interactivity. Anyone can come and draw on this, and the graffiti stays around for the public to play with. Not so impressive as a sketch program, but cool for its "bulletin board"-like functionality.
19. jsCanvasBike
    Chosen for: Interactivity. There are tons of HTML5 games out there, many of which are more complex; I just liked that this one was easy to get the hang of immediately.
20. Spark
    Chosen for: Interactivity. Since most of the above examples assume a keyboard or mouse (or both) instead of a touchscreen, here's a cool touch-only example. Be sure to change the mode at the top, and touch using as many fingers as you can!

Want more? Many of the above examples are referenced in the following collections:

http://html5demos.com/
http://www.canvasdemos.com/
http://thejit.org/demos/
http://www.phpguru.org/static/html5-canvas-examples