An Exploration of HTML5, Flash, and Javascript - Building a Presentation Engine

By: Devlin Smith

Supervisor: Prof. Peter Wentworth

15 October 2012

Abstract

HTML5 is an emerging standard that provides new features and capabilities that overlap with those traditionally provided by tools like Flash. Because the standard is still emerging there are no clear guidelines or trade-offs to help choose between using the different technologies. We demonstrate how HTML5 can be used to create a presentation engine, previously only possible in technologies like Flash. The presentations contain various rich and interactive media, including deep zoom viewing, videos, navigation control and sequencing of the presentation slides. These features demonstrate the capabilities of HTML5, combined with Javascript, and the techniques needed to use them.

1. Introduction

The World Wide Web (WWW) and specifically Hypertext Markup Language (HTML) are in a constant state of evolution, and with that come updates of the previous technologies [5]. Tim Berners-Lee's initial creation of the Hypertext Transfer Protocol (HTTP) and HTML, and the creation of masses of devices to browse the WWW, from old computers with the initial browsers like Mosaic and Netscape, to mobile devices and tablets with up-to-date browsers, there has been a mix of ways to interpret HTML [6][26][29].

The World Wide Web Consortium (W3C) co-ordinates groups in an effort to evolve and standardize the technologies of the WWW. Supporters included Tim Berners-Lee and various leading Information Technology companies including; Microsoft, Apple, Google, Adobe, and anyone wanting to join its groups [33]. They are continually updating the current HTML standard to include more applicable concepts and relevant Application Programming Interfaces (API) that are standardised for the different browsers that view the WWW [4]. Along with these HTML updates come new features that were previously only possible to achieve with Add-On technologies like Adobe's Flash [1].

They are also standardising the Cascading Style Sheets (CSS) used to provide styling to HTML pages [34]. These have widely come to be known as CSS3 and HTML5, and are the latest versions of these technologies [17][4]. It is important to note that they are still being standardised and are currently in draft format [17][4]. CSS3 comes with abilities like transitions and animations, and various additions to previous styling and effects. HTML5 comes with many more technologies such as; video, audio, canvas, websockets, geolocation, web workers [35]. Combined with Javascript to form control and many of the abilities of Flash could be substituted with the latest HTML technologies.

2. The Problem

With the release of new standards, and updates, there remains a problem of change, and addition, of syntax with the current version of HTML, as well as how developers should be using it and where its use is appropriate. This research will explore the uses of newer features of HTML and how they affect current technologies on the WWW such as Flash [1]. The exploration intends to expose useful features to a developer along with useful client side browser code, including the Document Object Model (DOM) and scripting to alter it, namely Javascript [36][12].

This is done through the creation of a prototype presentation engine, because a presentation is a media rich concept with vast opportunities for new HTML technology usage. This will only be a prototype presentation as the display of information is the essential concept for an argument towards how HTML will come to compete with technologies like Flash, and development is restricted to Google Chrome for scope purposes.

Seadragon is a library that allows for zoomable compositions of pictures, supported by Microsoft Expression Studio. Seadragon Ajax is a Javascript version that can be distributed over the WWW and is an open source component that is used as a base to create the prototype presentation engine [18]. The API is available from Microsoft Expression's online API with many useful components [18]. Newer HTML elements are used in combination with this to create the feeling similar to previous online Flash presentations [25].

3. Related Work

Other presentations on the WWW were examined to examine the level of richness being produced in Flash and Javascript, with possible new HTML5 features. This included Prezi, a Flash based presenter with many powerful features. For a true display of Flash's visual capabilities a view of this website is recommended, and was used as the base of our requirements specification for the engine [25].

HTML5rocks is another impressive use of CSS3 and HTML5 in combination to produce a smooth visually appealing and promotes the use of CSS, HTML, and Javascript as 'HTML5' deeming it the "Next generation features for modern web development" [8]. HTML5rocks also gives a brief overview of other new HTML5 elements and capabilities.

CreateJS is a Javascript library that uses HTML5 and was also explored for its control of HTML5 elements [10].

Another notable website is Wix, which allows a user to create websites online and use media rich HTML5 templates, as well as Flash ones [32].

4. The Presentation Engine

This section will expose work created and follow useful concepts, and technologies, in the creation of an online presentation. This is done through an analysis of the technologies used throughout the engine.

Any presentation engine requires mechanisms to functions. We identified the following requirements and our technological sources for the presentation in Table 1:

Table 1: List of required abilities and sources

Requirement	Technological source
System should encourage strong overview	Seadragon Ajax
Point to Point Navigation	Seadragon Ajax and custom Javascript
Timeline and Slide/Node control (Object creation and event control)	Custom Javascript
Overlay of custom objects: styling and positioning	Seadragon Ajax and custom Javascript
Video	HTML5: Video
Rich embedded content	HTML: Iframe
Interactivity	Custom Javascript

4.1 Significant HTML5 features for the presentation engine

The new HTML brings new events, elements and possibilities. This chapter covers HTML elements for the prototype presentation engine, as well as how they were previously possible. All HTML elements are controlled with Javascript in the presentation engine.

Video is a rich component, allowing for video to be streamed through HTML over HTTP. Previously this was only possible through a Flash plugin. This is a large area for potential competition, as video could be considered one of the most media rich forms available over the WWW and is utilized in the presentation engine. This integration of video into the semantics of the WWW is created in line with Tim Berners-Lee's hope for an Internet of things that understand one another [7]

The introduction of media requires the introduction of new media style events to control the media-player component, hence HTML5 has also introduced many new event handlers accessible through scripting [4]. These are used to detect behaviour like seeking, pause, play, and

loadstart (when media begins to load), and are controlled with Javascript to integrate the media-player.

Canvas is a content area designed for drawing pixels through scripting, usually through Javascript [15]. It has built-in functions that allow for text, shapes, shading and image drawing [4]. This can be very useful in multimedia experience, and has been used to create online games and various other complex concepts that were previously only possible in Flash [27]. The performance on this pixel by pixel drawing on canvas is constantly changing with new Javascript engines compared to Flash which has been optimising its environment for a much longer time, and is not as efficient across all platforms, as Canvas is immediate-mode graphics that is hardware accelerated on the latest browsers and is not retained as opposed to other technologies (see Section 5 for SVG) [4][13][27], hence require a frame by frame draw for animation. Some simple examples of its use would be seen in graphing, animation and image composition.

Canvas, Video, and Audio tags in HTML5 all have fall back sections, which is displayed when browsers cannot interpret the tag correctly [4]. This fall back section can be used to embed flash content for compatibility essential applications. This allows the designer to create web pages that are compatible on multiple browsers. For example:

```
<video id='vid1'>
        <source src="What is HTML5.mp4" type="video/mp4">
        <source src="What is HTML5.flv" type="video/flv">
        'Your browser does not support the video tag.'
<video />
```

The Iframe has been fully integrated into the new HTML specification, with the removal of the previous fall back section used in Iframes. This fall back section was included to allow designers to release HTML code that could be interpreted differently by browsers that could were not yet able to understand the tag, once again allowing for cross-browsers compatibility [4]. HTML5 now expects all browsers to be able to interpret this tag. They allow us to source content from other pages [4], differently phrased: we can embed another site inside our site. This has been used to source Picture Document Format (PDF) documents and embed them into our presentation, but has very broad application [4].

The above are the most notable HTML5 technologies used in the presentation, but other powerful tools are mentioned under the related work section. They are controlled by events and custom Javascript to control as a part of the presentation engine covered under Section 4.3.

4.2 Seadragon Ajax

Seadragon Ajax was essential to achieving the tool we selected to provide a scene display then navigating to scenes within those scenes. Seadragon Ajax is an open source Javascript library that allows the user to render Deep Zoom Images (DZI) through HTML [18]. The DZI is separated into layers of zoom, composed of 256 x 256 pixel tiles of the image at that level of zoom. Initial load time is reduced by only transferring images from the requested section of the DZI where the user is viewing [18][19]. See Figure 1 for a representation:

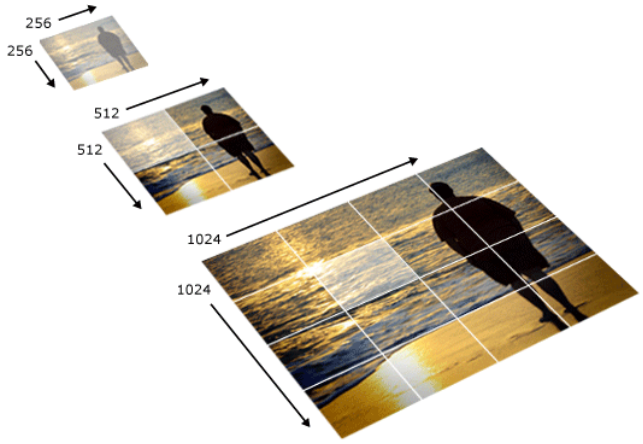

Figure 1: Deep Zoom Image separation of layers

There are many free composers for DZI. The product of choice in this case was Microsoft Deep Zoom Composer, which is also part of Microsoft Expression Studio [22]. They all have the DZI output; the composition is an Extensible Markup Language (XML) or DZI file, describing the folder structure and layout, and a folder structure composed of the 256 x 256 resolution tiles [19].

It is possible to pan and zoom over large or high resolution, images in a DZI with Seadragon Ajax, by using CSS to layer the images as the background to a container, and then load new images on top, which later become the background again [18]. This is all done through Javascript as the user pans and zooms by requesting normalised points, specifically x and y position, depicting coordinates within the container. Zoom is also used to determine which folder to access in the folder structure. These points are used covered again later within the creation of a timeline under Section 4.3.

Overlays allow the designer to post content at a fixed position relative to a point in the composition, but only allow for one size; hence the overlay doesn't appear to be fixed onto the image, and remains fixed, floating above the image, while the image zooms behind it [21]. This approach was passed over in our prototype in favour of the Seadragon rectangle, an advanced

form of overlays, which fixes two corners of the rectangle to (zoomable) points in the underlying image. The content's size is fixed relative to that around it [20]. This creates a better sense that the overlay is part of the underlying presentation. These Seadragon rectangles are vital to embedding custom content, specifically HTML5, into our presentation in order to correctly position and style the content.

4.3 Presentation Engine Prototype

A presentation requires several scenes to provide concepts and ideas; these are points within our DZI with the possibility of Seadragon overlays. These scenes require transition, ordering, timing between transitions, and the coordinates of the point. See Figure 2 for basic class diagram.

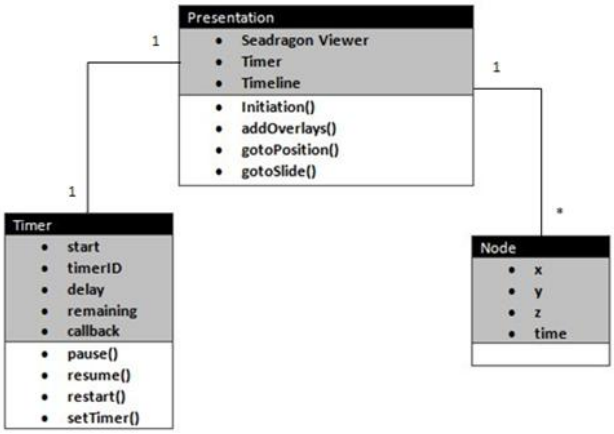

Figure 2: Incomplete Class diagram of prototype presentation

The presentation engine is embedded in an HTML page for execution on the client. It is written entirely in HTML5-compatible Javascript and served with custom Javascript. Javascript has many abilities to control the client side web page and serialize objects, which are utilized in the engine [12].

The Seadragon viewer allows us to control our instance of Seadragon Ajax, and getting coordinates and controlling pan and zoom movements. The presentation is created by saving the coordinates and timing into a waypoint node object, and then the waypoints are held within a Javascript array (the timeline). Consideration was given to whether to support custom events linked to waypoint nodes. For example, that would make it possible to start playing a video when the presentation navigated to that waypoint node. A function uses the timer to recurse through the

timeline, with basic navigation controls from slide to slide. As a slide is navigated to, a timer is activated. When this timer completes, the next navigation event occurs and the next timer initiated. Seadragon Ajax has built in methods for panning and zooming to points with smooth effects all computed through Javascript.

The timer allows the designer to create custom timed events for each slide. The user zooms to the given slide, begins a recursion through the timeline enacting the call-back after a period of time, which proceeds to the next slide and continues the recursion until the timeline is complete. This timer can be controlled in various ways much like a media-player control.

To create more richness in the presentations, Seadragon rectangles are added at points in on the DZI and are used to style customised content in the correct positioning. A designer is required to find the points where an element is to be held, and then proceed to create the element in Javascript, and attach it to the Seadragon viewer. Therefore the designer can embed rich HTML5 video, and canvas, within Seadragon and the browser interprets it with the correct styling and positioning, as done in the prototype presentation engine. Iframes are also used to embed external sources like PDF or Flash websites. This allows a designer to embed all media rich concepts previously available in older HTML. .

Seadragon captures and handles all mouse actions, so the built-in controls that are typical for audio and video do not receive the events directly: they are passed to Seadragon, where it handles to navigation control. Seadragon does provide facilities to create of custom controls, which hover on top of the viewer and can be used to control these elements through Javascript methods. In the prototype presentation we found it necessary to use these elements to control the timeline navigation and so that we could pass the relevant events to the media player. We can also control these elements through Javascript, or disable Seadragon's mouse control allowing events to propagate past it and to the media player.

In order to retain this timeline it can be saved to disk and reloaded. Through Javascript's use of Javascript Object Notation (JSON) objects can be serialized and de-serialized with the JSON library built into the Javascript engine [11]. The JSON output can then be saved into the page being hosted by the creator as the default presentation loaded with the page. It can also be inserted into the URL, which is parsed on load for slide number and timeline details. This also allows other users to create different timelines on the same presentation.

5. Conclusions and Further Work

To conclude, HTML5 brings all the required technologies to compete with Flash's current capabilities without the most of the complexities, but is still in its early stages. When HTML5 becomes a more consistent standard and the browsers have all implemented compatibility then HTML5 is the more likely choice to suit the largest target market. Presently, Flash is much easier to design because of the boilerplates that exist [1]. This future is however not far away as most browsers today are doing their best to keep up with the standard and there are multiple open source Javascript libraries that provide powerful capabilities [10]. See the following for related articles [4][16][24][28].

Other HTML5 technologies were explored in the creation of this presentation, but implementation was deemed out of scope for the project. These included websockets which allow for a Transmission Connect Protocol (TCP) connection to an address and perform the necessary handshakes [14]. Once this connection is made both pairs can both request and respond, which is powerful concept to interpret in an HTTP environment, where the server doesn't make requests or updates unless asked. This opens the potential for browsers to host content such as complex multiplayer games, or faster Rich internet application as a result of taking a layer off the network communication stack [14]. It is not capable of the User Datagram Protocol (UDP) that Flash can also perform, along with TCP [3].

WebGL is another technology that is driven by graphics acceleration in producing 3D artefacts and has not been explored due to scope [30].

CSS3 was also considered in the creation process, but its capabilities were not as relevant to the actual design and function of the presentation, hence out of scope.

SVG elements were also explored and in small detail, allowing for shapes and shading. More complex examples do exists and in combination with Javascript can yield powerful results [23][31]. Good use of these elements is susceptible to much time spent creating shapes, which Adobe has prefabricated in tools like Dreamweaver and Edge [9][2], and other sources [27].

Flash still has a place in the market because of the massive knowledge and information base created around it and its power in certain circumstances, with large boilerplate frame works. But to truly be a part of the semantic web of things, applications should explore the move to a HTML5 design framework for the advantages it may bring and be free from a vendor specific application, as well as future mobile browser compatibility.

6. References

1. Adobe. “Adobe Flash Video File Format Specification Version 10.1“. *Adobe Systems Incorporated.* URL *http:// download. macromedia. com/ f4v/ video_file_format_spec_v10_1. pdf* (2010)
2. Adobe. “Adobe Dreamweaver CS6”. *Adobe*. URL *http:// www. adobe. com/ africa/ products/ dreamweaver. html.* (2011).
3. Adobe Flash. “Flash.net”. *Adobe.* URL *http:// help. adobe. com/ en_US/ FlashPlatform/ beta/ reference/ actionscript/ 3/ flash/ net/ package-detail. html.* (2012).
4. Berjon, Robin, Travis Leithead, Erika Doyle Navara, Edward O'Connor, and Silvia Pfeiffer. “HTML5.” *World Wide Web Consortium (W3C).* URL *http:// dev. w3. org/ html5/ spec/ single-page. html* (2012).
5. Berners-Lee, Tim. "The Original HTTP as defined in 1991." *World Wide Web Consortium (W3C). URL http://www. w3. org/Protocols/HTTP/AsImplemented. html* (1991).
6. Berners-Lee, Tim. “Tim Berners-Lee” *World Wide Web Consortium (W3C). URL http://www. w3. org/Protocols/HTTP/AsImplemented. html* (2012).
7. Berners-Lee, Tim, James Hendler, and Ora Lassila. "The semantic web." *Scientific american* 284.5 (2001).
8. Bidelman, Eric, et. al. “HTML5rocks”. URL *http:// slides. html5rocks. com/ #landing-slide.* (2012).
9. Bright, Peter. “Adobe’s continuing revolution pushes the cutting Edge of HTML5 development”. *arstechniaca*. URL *http:// arstechnica. com/ information-technology/ 2012/ 09/ adobes-continuing-revolution-pushes-the-cutting-edge-of-html5-development/*. (2012).
10. CreateJS. “CreateJS”. *CreateJS.* URL *http://www. createjs. com/*. (2012).
11. Crockford, Douglas. "JSON: The fat-free alternative to XML." *In Proc. of XML, vol. 2006.* (2006).
12. Flanagan, David. “JavaScript: the definitive guide”. *O'Reilly Media*. URL *http:// www. studentcubes. com/ data/ books/ 0/ JavaScript%20The%20Definite%20Guide. pdf* . (2006).
13. Foley, James D., Andries Van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips. “Introduction to computer graphics” *(Vol. 55). Addison-Wesley*. (1994).

14. Hickson, Ian. “The WebSocket API W3C Working Draft 19 April 2011”. *World Wide Web Consortium (W3C).* URL *http:// www. w3. org/ TR/ 2011/ WD-websockets-20110419/.* (2011).
15. Holt, Bob, Rob Larsen, Marc Neuwirth. “CanvasJS”. *github*. URL *https:// github. com/ roblarsen/ CanvasJS.* (2012).
16. Jobs, Steve. “Thoughts on Flash”. *Apple.* URL *http:// www. apple. com/ hotnews/ thoughts-on-flash/.* (2010).
17. Meyer A., Eric, and Bert Boss. “Introduction to CSS3” *World Wide Web Consortium* URL *http:// www. w3. org/ TR/ 2001/ WD-css3-roadmap-20010523/.* (2001).
18. Microsoft Expression. “Seadragon Ajax”. *Microsoft.* URL *http:// gallery. expression. microsoft. com/ SeadragonAjax.* (2012).
19. Microsoft. “About Deep Zoom Composer”. *Microsoft Corporation.* URL *http:// expression. microsoft. com/ en-us/ library/ dd409068.* (2011).
20. Microsoft. “Seadragon Rectangle”. *Microsoft Corporation.* URL http:// expression. microsoft. com/ en-us/ gg413346. (2011).
21. Microsoft. “Overlay Positioning”. *Microsoft Corporation.* URL *http:// expression. microsoft. com/ en-us/ gg413355.* (2011).
22. Microsoft. “Microsoft Expression”. *Microsoft Corporation.* URL *http:// expression. microsoft. com/ en-us/ default.* (2011).
23. Osmani, Addy. “20 SVG uses that will make your jaw drop”. *Netmagazine*. URL *http:// www. netmagazine. com/ features/ 20-svg-uses-will-make-your-jaw-drop.* (2011).
24. Pfeiffer, Silvia. “The Definitive Guide to HTML5 Video”. *Apress* (2010): pg7.
25. Prezi. “Learn Prezi”. *Prezi.* URL *http:// prezi. com/ learn/.* (2012).
26. Raggett, Dave, Arnaud Le Hors, and Ian Jacobs. "HTML 4.01 Specification." *W3C recommendation* 24. (1999).
27. Rousset, David. “The Complete Guide to Building HTML5 Games with Canvas and SVG”. *Sitepoint.* URL *http:// www. sitepoint. com/ the-complete-guide-to-building-html5-games-with-canvas-and-svg/.*
28. Schroeder, Stan. “ Adobe Won’t Support Flash in Andriod 4.1” *Mashable* URL *http:// mashable. com/ 2012/ 06/ 29/ flash-in-android-4-1/.* (2012).
29. Stepp, Marty, Jessica Miller, and Victoria Kirst. "A CS 1.5 introduction to web programming." *ACM SIGCSE Bulletin* 4 Mar. 2009: 122-123.
30. Tavares, Gregg. “WebGL Fundamentals”. *HTML5Rocks*. URL *http:// www. html5rocks. com/ en/ tutorials/ webgl/ webgl_fundamentals/.* (2012).

31. Winter, Andre M., Andreas Neumanns. “carto.net SVG tutorial, example and demonstration site”. *Carto.* URL *http:// www. carto. net/ svg/ samples/*. (2011).
32. Wix. “Wix”. *Wix.com, In*c. URL *http://www.wix.com/*. (2012).
33. World Wide Web Consortium. “Facts”. *World Wide Web Consortium (W3C).* URL *http:// www. w3. org/ Consortium/ facts#people*. (2012).
34. World Wide Web Consortium. “CSS Current Status”. *World Wide Web Consortium (W3C).* URL *http:// www. w3. org/ standards/ techs/ css#w3c_all*. (2012).
35. World Wide Web Consortium. “HTML5 differences from HTML4”. *World Wide Web Consortium (W3C).* URL *http:// www. w3. org/ TR/ html5-diff/*. (2012).
36. World Wide Web Consortium. “Document Object Model (DOM)”. *World Wide Web Consortium (W3C).* URL *http:// www. w3. org/ DOM/*. (2009).