HTML5, Javascript, CSS3 - Rich Online Internet Presentation

Literature Review

Supervisor: Prof. Peter Wentworth

Devlin Smith

28 May 2012
# Table of Contents

1. Introduction ...................................................................................................................... 3

2. Hypertext Markup Language (HTML) .............................................................................. 3

   2.1 Hypertext Transfer Protocol (HTTP) .......................................................................... 5

3. Cascading Style Sheets (CSS) ......................................................................................... 7

4. Javascript .......................................................................................................................... 8

   4.1 Ajax ............................................................................................................................... 9

   4.2 Libraries ...................................................................................................................... 9

   4.3 Seadragon Ajax .......................................................................................................... 9

      4.3.1 Deep Zoom Image in Short (DZI) ..................................................................... 10

      4.3.2 Seadragon API .................................................................................................... 11

      4.3.2.1 The DZI Again ............................................................................................... 11

      4.3.2.2 Mouse Handling ............................................................................................. 11

      4.3.2.3 Co-ordinates .................................................................................................. 12

      4.3.2.4 Viewport ........................................................................................................ 12

      5.2.2.4 Overlays and The Drawer ................................................................................ 12

5. Flash and Silverlight ........................................................................................................ 13

6. General Presenting [ I'd prefer "Presentation"] Concepts ................................................ 14

7. Alternatives ...................................................................................................................... 14

8. Previous Approaches ...................................................................................................... 14

9. This Approach ................................................................................................................ 16

10. Conclusion ..................................................................................................................... 16

11. References ..................................................................................................................... 17
1. Introduction

The following is a review of literature, for an online prototype presentation engine focusing in the following areas in computer science; HTML and HTML5, Javascript, along with AJAX, and more specifically Seadragon (including deep zoom imaging), CSS through CSS3, and more minor topics being Flash and Silverlight. The intent throughout is to explore the evolution of richness (more specifically interactive richness) available on the World Wide Web (WWW) and its relevance in the creation of a presentation engine. Rich Internet Applications (RIA) are a recent development starting around 1998 with the evolution of Flash [2], and allowed the user to download a lightweight application that runs in the browser, through underlying engines [41, 2]. RIAs are normally embedded into a webpage and available to the user by browsing to them, and then interacting with them, allowing for a greater level of richness to be drawn by the user, from the page. The prototype presentation engine will be created solely in Javascript and HTML to prove this method as a viable alternate to Flash creation of RIAs.

2. Hypertext Markup Language (HTML)

The Richness provided by HTML will be evaluated in a chronological order. Excluding The World Wide Web Consortium’s (W3C) most recent proposals for HTML5, HTML has lacked a large amount of multimedia and interactive rich content. HTML was proposed in 1989, formalised and written to paper in May 1990 [5], went through multiple revisions and in December 1999, the W3C’s proposed for HTML 4.01 [44]. These proposals and specifications defined a language that could be used to describe the structure of a webpage in terms of Standard Generalized Markup Language [7].

Richness previously mentioned was supplied through the addition of:

- Form-based file uploads, which gave the ability to upload files from the client side to the server [36]. This gave richness to the end-user by allowing for the creation of file servers, and the storing of data, on the World Wide Web (WWW).
• Tables allowed for structures that the list element did not [13]. One was then able to position and pattern data with columns and rows, with contents of each cell being relative to each other, allowing a web-developer to create a grid style layout.

• Images became available for the client side in 1996 [50]. Browsers were now given the option to interpret image tags as standard picture formats.

With all the tools available at this stage it would be possible to create an image forum that stored and displayed information, through server side scripting. This means that users could now add content to the WWW.

Secure Socket Layer (SSL) technology became available; allowing for secure transmission of information, meaning businesses could act with trust. This began a trend in the internet business community (known as prosumers, a term coined earlier in 1981 for the combination of a consumer and producer, i.e. and end-user customer that produces content for the given operation [52]. Users now had the ability to add richness into the internet, in the form of textual-information and images, i.e. initial forms of media.

To follow the timeline, HTML 3.2 was then published in 1997, which was an attempt to reform HTML to the recommended practices from earlier versions as this is the first revision solely in W3C’s control [45]. This also made revisions to styling, adding some richness, but this will be spoken about in the Cascading Style Sheets (CSS) Section. This was followed by HTML 4.0, and then HTML 4.01 neither made any notable contributions to the richness of the World Wide Web [46, 47].

There was now a large time gap in the development of HTML, as other technologies, such as Javascript (later Flash and Silverlight), and the technologies within these environments, grew greatly. Then in 2008 W3C proposed HTML5 in a working draft, focusing on semantics, in line with Tim Berners-Lee’s ideas on how the WWW should work [26, 6]. Hence the current creation and depreciation of HTML tags through the continual working drafts, and the introduction of some very content rich tags, including Video, Audio, Scalable Vector Graphics (SVG) and Canvas elements. [28]. The advantages of each, in terms of rich media content, are as follows:
- Video: allows for the browser to directly interpret video content, depending on supported formats, including controls such as play/pause, timing, and volume [28].
- Audio: as previously, with video, media content may be directly streamed to the browser. Both use the source tag to determine the content and type to be sent and/or received, and have the same basic controls above [28].
- SVG: Scalable Vector Graphics are vector based graphics, allowing them to scale, while retaining resolution. The SVG tag gives an area in which one can draw vector shapes and was actually being recommended as early as 2001 with version 1.0 [37], but only became part of the W3C specification in 2003, with version 1.1 [28, 17], and is still not supported by all browsers [17, 49]. The SVG area also has performance issues when scaling the number of objects because it is vector graphics not bitmapped graphics, hence uses a retained graphics mode\(^1\).
- Canvas: A bitmapped area, for which several Javascript APIs exists to create text and shapes, along with images [28]. This uses immediate mode graphics\(^2\).

It is important to note that HTML has become a continually updated language that exists mainly in working draft form currently being developed under the W3C group in collaboration with The Web Hypertext Application Technology Working Group (WHATWG) [26].

In the period where these technologies were not available through HTML, HTML provided for "extensibility" by a system of tags and plug-ins. Hence all early videos, music, animation, live context was provided by plugins. HTML evolution followed a common trend: when "add-on" features become prevalent, it becomes advantageous to migrate their functionality into the core with HTML. “Add-on” features are covered under the Flash and Silverlight section.

### 2.1 Hypertext Transfer Protocol (HTTP)

---

\(^1\) Retained mode graphics is a rendering style by which each object in the area is kept track of, updated and individually redrawn [21].

\(^2\) Immediate mode graphics is a rendering style in which the entire area is redrawn every refresh [21].
HTTP was created by Tim Berners-Lee, along with HTML, in order to facilitate HTML transfer [5]. There were three versions in total, with several revisions for each.

The original version was, “...a hypertext medium...”, in Tim Berners-Lee’s original proposal, and then proposed as HTML0.9 in 1991 with only the ability to GET pages [5, 8]. This means one could only retrieve pages, and does not allow for file uploading or other useful actions to be mentioned shortly.

This was then revised and reformulated into HTML 1.0 in 1996 allowing for HEAD and POST commands, as well as Multipurpose Internet Mail Extensions (MIME), which is a way of specifying and describing the format of internet message bodies [9, 33]. HEAD gave us another level of richness by allowing for quick and lightweight requests that only dealt with the message header of the HTTP message, instead of carrying the whole body as well [9]. This is especially useful when all that is needed is metadata about a page, i.e. if the page is there or not, and ignores the “if-modified-since” header attribute [9]. We also gain the ability to POST to a server, sending a body to the server, to quote to RFC1945 [9]:

- “Annotation of existing resources”, allowing for a user to provide information on already existing information.
- “Posting a message to a bulletin board, newsgroup...” this is very similar to the point above but with more robustness.
- “Providing a block of data, such as the result of submitting a form...” as previously mentioned in the above HTML section, the uploading of form data allows for user uploads of many data types [7].
- “Extending a database through an append operation.” allowing for the web-page to become a Graphical User Interface (GUI) to a database, i.e. any browser could become a data modelling engine to the user, on the right website.

All the above points can be summarized, for the purpose of the review, as the gain of the ability to hold an HTML level two-way conversion, between client and server, and messages to be passed either way. This means the client could add their own richness to the environment they are browsing.

HTTP 1.1 was then introduced and updated, and is still the underlying protocol (at the time of this paper), since 1997 for transferring HTML packets across the internet [18,
This has had several additions but none that notably add to the richness argument being proposed [29, 38, 48, 39]. This was done largely in an effort to move to a completely stateless HTML, where state transfer is the means of change [5]. Previously POSTs were used to DELETE and PUT data, hence the addition of these actions, to create a fully develop a “create, read, update and delete” environment, and OPTIONS allowed one to query the available commands.

The client will send some unique identification data with each request, such as the state of the session, in which case the representation of a server side session can be retained with a cookie. A unique session ID could also allow a server to restore a session from data storage.

2.2 Document Object Model (DOM)

DOM is a “platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.” [25]. The DOM stores the structure, content, style, and events that occur on a web page. The limited facilities for detecting user generated events and modifying HTML lead to the creating of ECMAScript (a standard for client-side browser scripting) in 1997 [44, 46] (follow Javascript: see Section 4) [20, 25]. After the ECMAScript standard was released, W3C began work on standardising the DOM for all browsers [44, 46].

3. Cascading Style Sheets (CSS)

CSS is a method, designed for the distribution of styling on “...the World Wide Web.” [10, 12]. Styling can be defined as the attributes, within the style attribute of a tag. To define further, these are the presentation semantics understood by the browser when interpreting HTML and other Markup languages [10]. CSS was being used as early as October 1995, and became part of W3C’s recommendation in March 1998 [10, 12]. It allowed for the separation of content from style and allowed style to be applied generically over content.

Initially CSS included layout, colour, fonts, and simple presentation techniques [10, 11]. This allows for information structuring and highlighting adding basic richness to the
textual level of the website. During the evolution of CSS, through W3C’s working drafts, the ability to present information in a rich fashion, with unique styling, has increased. This in turn allows the web-developer to display information in an appropriate way, showing its nature and/or importance. Some of the rich tools include relative and absolute positioning, with a z index, allowing for the overlapping of elements in CSS2 [css2ref]. CSS3 then goes further into styling by extending features in CSS2, but in a modular style, including animation, transition, and transformation [css3ref]. All of which can be considered useful tools in a media presentation.

4. Javascript

Javascript has a multitude of uses, but the aspects we need to consider are only a part of Javascript, specifically its operations within the browser engine, pertaining to the window API. This API allows the web-developer to run script on the clients browser, uploaded with the web-page, to provide client side functionality, as, to reiterate from above, HTML is stateless therefore requires a full page request to perform a change, but Javascript can alter the HTML of the page while the client is still viewing them [51, 27, 5]. This is a technology that could potentially add a high quality of richness to the presentation application being designed, and will likely be where most of the time is spent, as a consistent page allows for a consistent end-user experience.

Javascript’s abilities within the Document Object Model (DOM) are extensive, including the ability to alter and create styling, tags, and order of tags [25]. This allows the web-developer to upload a script to the client computer, when they request a web-page, to alter that page as they interact with the page [16, 20].

Javascript allows us to interact with the DOM event model, allowing us to control various events with Javascript [20]. This means that we can change the styling and positioning of our elements, and more complex interaction with elements such as the HTML5 canvas, allowing us to draw bitmapped scenes [43]. This ability is further enhanced by the existing free Javascript libraries that provide an API to interfacing with, for example, the canvas element, such as EaselJS, once again enhancing the ease of rich internet application creation [15].
4.1 Ajax

Ajax stands for Asynchronous Javascript and XML, although ajax does not require the use of XML specifically [22]. It allows for the client for formulate and send a request to the server, and asynchronously deal with the result, meaning that the browser does not actively wait for a response and the user can continue browsing, while the Javascript engine deals with the request.

4.2 Libraries

HTML allows Javascript to be written as embedded code, or imported from an external source, as well as being free to develop with many free Integrated Development Environments (IDEs) and debugger built into Chrome, and a debugger is easily obtained in Mozilla Firefox with Firebug [23, 34]. This allows any web-developer to create powerful libraries to perform functions that can be distributed and used by others, to enrich their browsing experience much like the CreateJS suite, but possible by anyone [15].

4.3 Seadragon Ajax

Seadragon Ajax is a Javascript derivative of the software developed by the Seadragon Software Company, formerly known as Sand Codex [30]. The main concept behind the software is to break down an image into a large collection of images, all of the same resolution (256px * 256px), but of varying qualities (covering set amounts of space from the original picture), dependant on a level of zoom, composed into a Deep Zoom Image (DZI) [30]. Then depending on where one attempts to view/zoom/pan it uses AJAX to request the appropriate level image to express the desired quality of the picture, hence allowing the user to request only the desired data of an image, making the illusion of visual clarification [31]. This allows the web-developer to resize larger pictures and embed them inside smaller ones simply with Microsoft’s Deep Zoom Image Composer [31]. Theoretically it could create an almost infinitely large image to view, and the user would continue zooming in to view content, defining it as the navigated, which is a very media rich concept for the WWW.
4.2.1 Deep Zoom Image in Short (DZI)

Deep Zoom Images are a use of the Microsoft Deep Zoom technology, which allows for the viewing of images over a medium with efficient transmission [31]. As mentioned above, it decomposes the larger image into tiles of the same resolution, and “choose only the data that is required for a particular view” [31]. Note the Figure 1 below depicting pixel size and viewed area:

![Figure 1: The DZI tile composition](http://i.msdn.microsoft.com/dynimg/IC141135.png)

This is stored in a directory structure that describes zoom level along with an Extensible Markup Language (XML) file that describes the format, tile size, and size of the whole image at various levels of zoom [31, 14].

A side note on XML: it is another mark-up language, much like HTML, and is designed to carry only data, and has no specific protocol designed for its transfer [14]. It uses the same opening and closing tags as HTML, but allows for the description of any set of data structures, described through tags and attributes of the tags.

---

3 <http://i.msdn.microsoft.com/dynimg/IC141135.png>
4.2.2 Seadragon API

The Seadragon tool gives us a large level of visual richness, which is specifically valuable in a presentation aspect, allowing for the easy creation of massively scalable images on a WWW distribution platform. An important note here: rotation and transformation, on the Seadragon container will not function with the Seadragon point system, as the mouse tracker and point system are only built for uniform display, and this can be seen in a study of the source code [30]. To follow is an analysis of applicable and notable references into the Seadragon API, information that will be need to draw further richness from this tool.

4.2.2.1 The DZI Again

Firstly a developer must be able to load the DZI, which is by either; embedding the viewer and referencing a DZI, with the ‘Seadragon.embed’ function, which does not allow for the maximum level of customisation [30]. The second method is by use of the Seadragon Viewer on the load of the page, which creates a fully controllable Seadragon container, which uses the default navigation methods for Seadragon.

Secondly a developer should understand the composition of the DZI. The output of Microsoft’s Deep Zoom Image Composer leaves a designer with 2 notable items of interest: the image directory, with the image broken down into a folder structure, depicting zoom level and the second is either an XML or DZI file, which describes the size of the image, and composition at each level [31].

4.2.2.2 Mouse Handling

Seadragon has a mouse tracker which passes control of all click and drag functionality to the util class to affect the viewport, which will be explained soon. This implies that Seadragon Ajax has its own methods for dealing with mouse navigation, meaning that mouse events within the Seadragon container will always be stopped from moving through the DOM stack to underlying structures, as can be seen in the Seadragon source code made openly available [30]. i.e. There are going to be issues around direct mouse interaction with any content put inside Seadragon that require their own
interaction such as the video control bar. Note the events behind the controls can still be activated, but the mouse interaction is limited due to Seadragon built in mouse navigation controls. All following information can be found on the Microsoft Live Labs website [30].

4.2.2.3 Co-ordinates

In the Seadragon environment, eg the div tag that contains the Seadragon image within the window is referenced by scale, from one side to the other ranging from zero to one on the greatest length side, with the other side in ratio. For example if a picture is twice as tall as it is long Seadragon coordinates could be graphed in terms of (x,y), as (0.5,1) for the furthest corner from the top left. Complications with this topic will be discussed under overlays. The Seadragon engine can automatically move smoothly (animate) from point to point, allowing the web-developer to easily create a navigation system.

4.2.2.4 Viewport

The viewport is the currently viewable area, defined by the dimension of the Seadragon container, a level of zoom, determining the image set to select, and the point, refining the images that need selection within that set. The viewport also animates from area to area, giving us an event that shows the user has changed position, and we may want to display something else according to our presentation.

5.2.2.4 Overlays and the Drawer

Seadragon allows the web-developer two different styles of adding overlays, one is with fixed positioning inside a Seadragon rectangle, which restrains the shape to the given 4 points that depicted by the corners of the rectangle. The second method is a placement point, in which the overlay size will remain constant, inside the Seadragon viewport, despite the level of zoom. And is fixed to the viewport by a Seadragon point and a placement, being the anchor position on the overlay (BOTTOM, TOP, and so on...).

The overlay concept allows us to add any HTML element, with their richness, into our Seadragon Ajax environment. This implies that it is possible to run video and canvas...
elements, as well as add text, to the Seadragon environment, allowing for a very content rich zoomable area.

At this point in the review the reader should be able to summarise that the review has defined the necessary elements; to a large amount of richness to the online presentation tool, through the combination of HTML styling, more advanced HTML elements, and the Seadragon Ajax library a visually rich information environment with point to point navigation & animation is possible.

5. Flash and Silverlight

Flash and Silverlight are multimedia platforms, used to provide additional richness to web pages and both are not default installations in common browsers, such as Opera, Firefox, and Chrome [40, 24, 35, 1, 32]. They are plugins that a user must download and install to their browser, unlike the underlying Javascript engines, in the above browsers. They both have their own language respective; Action Script (a superset of Javascript) and XAML [1, 32].

If the reader follows the timeline gap between the lack of HTML media content and HTML5’s abundance of; they will note that this content did actually exist but was distributed by Flash through Action Script from around 1998 [2], and later Silverlight came after. Flash has the ability to provide all the above content that has been reviewed, but requires the knowledge of additional languages, and techniques to using the libraries from these languages. It also requires conceptual knowledge of the language, as with all languages, if the designer wants to be properly efficient and secure.

An experienced developer would begin to argue at this point, as many IDEs have been created for Flash that allow for quick and easy development that is secure and efficient, which is exactly what the average person, or more specifically what an artist or expressive designer will need as a tool to develop in these frameworks, without having to know much about the language itself. But at the same time these IDEs detract from the abstraction and this can cause code to be inefficient, which is what a true designer must avoid.
6. General Presentation Concepts

In order to create a functionally appealing tool, concepts of functionality must be framed around good presenting techniques that will be useful as artefacts to interact with the RIA. These include; the use of video and audio, visual presentation of data (through pictures or charts), similar concepts are grouped, point to point navigation, and overall styling and flow to create a visual appeal.

7. Alternatives

Alternative approaches or technologies suitable in the creation of and online presentation engine that were considered included;

Using the CreateJS suite and a Javascript data structure to create a form of presenter, but this option limits the specifications as the design necessary increases the scope immensely in order to create free flowing, zoomable, and content rich applications.

Open Seadragon was a version of Seadragon Ajax that existed for a short period that attempted to expand on Seadragon Ajax’s abilities and fix its bugs [3]. This project is no longer continued, and Microsoft has released newer versions of Seadragon Ajax, since its latest update.

The last option considered was altering the DZI file and or format to include information on a lower level, which allowed for embedding of HTML elements with the Deep Zoom data structure. This was avoided due to the complexities of the technology, and the lack of information on it, and finally the Deep Zoom technology is copyrighted by Microsoft [31].

8. Previous Approaches

Other online RIAs for creating presentations include Impress.js and Prezi. Impress.js is an open source Javascript library that uses CSS3 techniques to create transforms and
translations in a presentation, created by Bartek Szopka in 2011 [4]. Prezi is a Flash
based RIA that even has subscribing members to use its more advanced features. It uses
Action Script to create a fully zoomable and transitional presentation application [42].

There are many other online presentation tools that have notable features. The above
are chosen for their use of impressive transitions and structured information to create an
information rich environment that is not necessarily flow restricted.
9. This Approach

As noted in the above section, the aim is to create a structured presentation, which will have point to point animations and create an information rich visual presentation. Most of these technologies exist, and the tasks becomes about how to integrate the technologies. This will be done through the combination of the Seadragon Ajax library, to provide a navigation and fully zoomable images, with the latest HTML and CSS technologies, in order to provide a similar level of richness one can obtain from applications listed above, with Javascript to provide customization.

10. Conclusion

There has been a large gap in the RIA development technologies, due to the non-existent nature of previous HTML tags, which made it difficult to perform rich tasks such as audio, video, and drawing, leading to the web-developer to becoming dependant on Flash or Silverlight to develop Rich Internet Applications, which lead them to require knowledge of much greater concepts than should be required. This paper will propose a substitute for the requirement of that knowledge, through the building of an RIA that performs similar levels of actions as Flash counterparts. The main consideration is in adapting these technologies to work well together, in a prototype presentation engine.
11. References


[38]: Nottingham, Mark, and Eran Hammer-Lahav. "Defining Well-Known Uniform Resource Identifiers (URIs)." (2010).

