Department of Computer Science Rhodes University

RHODES UNIVERSITY
Where leaders learn

An Exploration of HTML5 and Javascript - Building a Presentation Engine

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science (Honours)of Rhodes University 2012

By

Devlin Robert Smith

November 2012

HTML5 is an emerging standard that provides new features and capabilities that overlap with those 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. This paper demonstrates 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.

ACM Classification

H.3.5 [Information Storage and Retrieval]: Online Information

H.5.4 [Information Storage and Retrieval]: Hypertext/Hypermedia

General Terms: Web-based services, Theory

Acknowledgements

I would like to acknowledge the financial and technical support of Telkom, Tellabs, Stortech, Genband, Easttel, Bright Ideas 39 and THRIP through the Telkom Centre of Excellence in the Department of Computer Science at Rhodes University.
I would also like to acknowledge the intellectual contribution and guidance of my supervisor, Prof. Peter Wentworth.
And a final thank you to all staff at the Rhodes University Computer Science Department for always being willing to lend a helping hand.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

HTML5 introduces new capabilities to HTML. To understand these and put them into context, this thesis explores the creation of a rich media presentation engine. The envisioned presentation will be able to zoom and pan in a large scene. Within the scene there will be several pieces of rich media, created using new HTML5 technologies.

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 [*9*]. 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 lead to the initial browsers like Mosaic and Netscape, then onto to mobile devices, tablets, and personal computers with up-to-date browsers. With these there has been a mix of ways to interpret HTML [*9*,*61*,*69*].

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 [*74*]. 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 [*6*].

They also standardise the Cascading Style Sheets (CSS) used to provide styling to HTML pages [*75*,*39*]. These have widely come to be known as CSS3 and HTML5, and are the latest versions of these technologies [*6*,*42*]. It is important to note that they are still being standardised and are currently in draft format [*6*,*42*]. 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 [*73*]. Along with these HTML updates come new features that were previously only possible to achieve with Add-On technologies like Adobe's Flash [*3*,*40*]. Combined with Javascript to form control and many of the abilities of Flash could be substituted with the latest HTML technologies.

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. The Seadragon Ajax library provides navigation and fully zoomable images, added to the latest HTML technologies, in order to provide a similar level of richness one can obtain from Flash applications, with Javascript to provide customization in order to achieve a presentation engine.

To make the ideas concrete, imagine a public relations presentation for an institution like a university. From a distance one might be able to fly into a more detailed look at facilities. When flying close to the sports field, a video clip showing a game of rugby might start playing. As one zoomed into a lecture theatre the door could become an Iframe which embedded web content from one of the courses. Zooming to the residence dining hall might show an image of a typical menu. Embedded HTML links in the presentation could invite the user to click if they wanted to hear an audio track giving a brief history of the institution.

Chapter 2: Literature Review

2.1 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, and allowed the user to download a lightweight application that runs in the browser, through underlying engines [*40,19*]. RIAs are normally embedded into a webpage and available to the user by browsing to them. Then interacting with them allows for a greater level of richness to be drawn by the user, from the web 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.

2.2 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, went through multiple revisions and in December 1999, the W3C's proposed for HTML 4.01 [*61,9*]. 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 [*10*].

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 [*52*]. 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 [*14*]. 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 [*67*]. 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 another website in the form of images and text through form submissions, expanding the abilities of the end-user of the website.

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 [*72*]. 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 [*60*]. This also made revisions to styling, adding some richness, but this will be spoken about in the Cascading Style Sheets (CSS) Section of this Chapter. 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 [*61*,*62*].

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 [*31*]. 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 [*73*]. 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 [*73*].
- 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 [*73*].
- 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 [*52*], but only became part of the W3C specification in 2003, with version 1.1, and is still not supported by all browsers [*4*,*65*,*73*]. 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 [*73*]. 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) [*31*].

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 (See Section 2.5 of Chapter 2).

[1] Retained mode graphics is a rendering style by which ‘primitive’ in the area is in memory, and when updated the graphics library redraws the ‘primitives’ [*48*].

[2]Immediate mode graphics is a rendering style in which the entire area is procedurally redrawn [*48*].

2.2.1 Hypertext Transfer Protocol (HTTP)

HTTP was created by Tim Berners-Lee, along with HTML, in order to facilitate HTML transfer [*7*]. 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 [*9,7*]. 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 [*11,49*]. 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 [*11*]. 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 [*11*]. We also gain the ability to POST to a server, sending a body to the server, to quote to RFC1945 [*11*]:

- "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 [*10,41*].
- "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 [*21*,*22*]. This has had several additions but none that notably add to the richness argument being proposed [*35*,*54*,*63*,*53*]. This was done largely in an effort to move to a completely stateless HTML, where state transfer is the means of change [*7*]. Previously POSTs were used to DELETE and PUT data, hence the addition of these actions, to create a "create, read, update and delete" environment, and OPTIONS allowed one to query the available commands.

In order to keep a two-way session sides are required to keep state and transfer required state each transaction. This is done through cookies and session variables so that either server or client could keep state [*22*]. Cookies keep client side values in a stored file in the browsers history, whereas session variables are stored on the server while the browsers has a session with the server open [*22*].

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.

2.2.2 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." [*36*]. 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 (follow Javascript: see Section 4 of Chapter 2) [*23*,*36*,*61*,*62*]. After the ECMAScript standard was released, W3C began work on standardising the DOM for all browsers [*61*,*62*].

2.3 Cascading Style Sheets (CSS)

CSS allowed for the separation of content from style and allowed style to be applied generically over content. It is a method, designed for the distribution of styling on "...the World Wide Web." [*12*,*38*]. 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 [*38*]. CSS was being used as early as October 1995, and became part of W3C's recommendation in March 1998 [*38*,*75*].

Initially CSS included layout, colour, fonts, and simple presentation techniques [*38*]. 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 [*12*,*39*]. CSS3 then goes further into styling by extending features in CSS2, but in a modular style, including animation, transition, and transformation [*75*]. All these can be considered useful tools in a media presentation.

2.4 Javascript

The Javascript 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 [*7*,*8*,*32*,*68*]. Javascript has a multitude of uses, but the aspects in consideration are only a part of Javascript, specifically its operations within the browser engine, pertaining to the window API. This is a technology that could potentially add a high quality of richness to the presentation application being designed, as manipulating the end-user's page will provide the dynamic experience.

Javascript's abilities to interact with and transform the Document Object Model (DOM) are extensive, including the ability to alter and create styling, tags, and order of tags [*36*]. This allows the web-developer to upload a script to the client computer when they request a web-page and to alter that page dynamically as the user interacts with the page [*20*,*23*].

Javascript also allows us to interact with the DOM event model, allowing us to control various events with Javascript [*23*]. This means it is possible to 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 [*58*]. This ability is further enhanced by the existing free Javascript libraries that provide specialist APIs. For example, as EaselJS allows for simplified interfacing with the canvas element, once again enhancing the ease of rich internet application creation [*17*].

Any web-developer can create libraries, and/or use them, to perform functions that can be distributed and used by others, to enrich their browsing experience. For example, the CreateJS suite, but possible by anyone, only API knowledge is needed [*17*]. Javascript also developed with many free Integrated Development Environments (IDEs) and the debugger built into Google Chrome, and a debugger is easily obtained in Mozilla Firefox with Firebug [*28*,*50*].

2.4.1 AJAX

AJAX stands for Asynchronous Javascript and XML, although AJAX does not require the use of XML specifically [*26*]. It allows the client to formulate and send a request to the server, and asynchronously deal with the response, 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. AJAX is widely used for trying to make the browser interface behave more like a rich desktop client: it does this by updating individual components on a web page without having to reload the whole page.

2.4.2 Seadragon Ajax

Seadragon is a tool for providing zooming navigation for large high detail images that will provide the overview capabilities of the presentation engine. It transfers only the pixels required for a certain perspective of the scene to be loaded through AJAX at a particular time, and the user can continue zooming in, and panning, to view content, defining it as the navigated (e.g. like those viewable Google Earth), which is a very media rich concept for the WWW.

Seadragon Ajax is a Javascript derivative of the software developed by the Seadragon Software Company, formerly known as Sand Codex [*43*]. 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) [*44*]. 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 [*43*]. This allows the web-developer to resize larger pictures and embed them inside smaller ones simply with Microsoft's Deep Zoom Image Composer [*43*].

2.4.2.1 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 [*43*]. 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" [*43*]. Note the Figure 1 below depicting pixel size and viewed area:

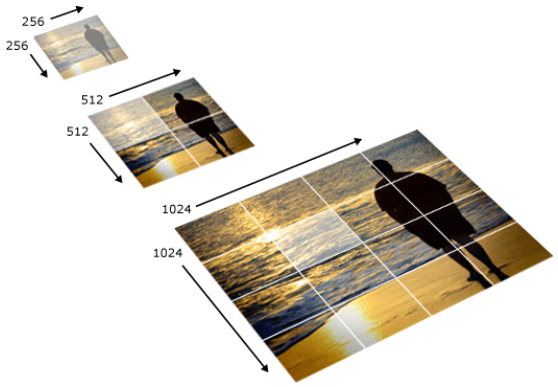


Figure 1: The DZI tile composition at 3 different zoom levels[3]

In the Seadragon implementation, the various resolution tiles are stored in a directory structure that reflects their zoom level. An Extensible Markup Language (XML) file describes the format, tile size, and size of the whole image at various levels of zoom [*43,15*].

A side note on XML: it is another mark-up language, much like HTML. It is designed to carry only data, and has no specific protocol designed for its transfer [*15*]. It uses tags that are similar in style to HTML, but rather than having a fixed set of tags, XML is extensible, allowing new tags to be invented dynamically. This allows for the description of a set of hierarchical data structures, described through tags and attributes of the tags.

2.4.2.2 Seadragon API

The Seadragon tool gives us a large level of visual richness from the tools found in its API. This is specifically valuable in a presentation. It allows for the easy viewing of massively scalable sets of images on a WWW distribution platform. Elements can also be styled so as to be placed on or within the presentation. Point-to-point zooming is also performed through the Seadragon API. Information that will be needed to draw further richness from this tool will be covered later in the project design and experimentation.

[3] <http://i.msdn.microsoft.com/dynimg/IC141135.png>

2.5 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 Google Chrome [*40*,*47*,*55*,*51*,*27*]. 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 and must make themselves compatible for each end device [*40*,*24*,*47*].

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 [*3*], 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.

2.6 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. See Table 1 for an overview of the required presentation concepts.

Table 1: List of required capabilities and sources

Requirement	Technological source
System should encourage strong overview via Zoom. Not just slide by slide summary	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
Custom Drawings	Canvas & SVG

2.7 Alternatives

Alternative approaches or technologies suitable in the creation of an 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 [5]. This project is no longer continued, and Microsoft has released newer versions of Seadragon Ajax.

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 owned by Microsoft and lacks open documentation [*43*].

2.8 Previous Approaches

Other online RIAs for creating presentations include Impress.js and Prezi. Impress.js created by Bartek Szopka in 2011 [*70*] , is an open source Javascript library that uses CSS3 techniques to create transforms and translations in a presentation. Prezi is a Flash based RIA that has subscribing members to use its more advanced features and store presentations online (free basic mode is available). It uses Action Script to create a fully zoomable and transitional presentation application [*59*]. For a true display of Flash's visual capabilities developed by professionals a view of this website is recommended, and was used as the base of our requirements specification for the engine [*59*].

HTML5rocks is another impressive use of CSS3 and HTML5 in combination to produce a smooth visually appealing presentation. It promotes the use of CSS, HTML, and Javascript as 'HTML*5*' deeming it the "Next generation features for modern web development" [*13*]. 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 [*17*].

Another notable use of HTML5 media is a website called Wix, which allows a user to create websites online and use media rich HTML5 templates, as well as Flash ones [*77*].

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. They have been used to provide ideas in the creation of the final presentation engine.

2.9 Literature 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. This lead to the web-developer to becoming dependant on Flash or Silverlight to develop Rich Internet Applications, hence they 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.

Chapter 3: Conceptual Model

3.1 Introduction

The aim is to create a structured prototype presentation engine, which will have point to point navigation animations and zooming for an overview. The Seadragon Ajax library provides navigation and fully zoomable images, added to the latest HTML technologies, in order to provide a similar level of richness one can obtain from Flash applications, with Javascript to provide customization. This creation will provide the reader with a view of the methodologies required to create such a rich media tool.

A prototype presentation engine was chosen, because a presentation is a media rich concept with vast opportunities for new HTML technology use. 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 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 [*44*]. The API is available from Microsoft Expression's online API with many useful components [*44*]. Newer HTML elements are used in combination with this to create the feeling similar to previous online Flash presentations [*59*]. For the prototype presentation engine See Appendix 2.

3.2 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. The project identified the requirements and our technological sources for the presentation in Table 1 of the Literature review (See Chapter 2, 2.6)

3.2.1 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, from events, on to canvas, video, audio and Iframes. 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 [*73*]. 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 [*33*]. It has built-in functions that allow for text, shapes, shading and image drawing [*31*]. 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 [*64*]. 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 of Chapter 3 for SVG) [*6*,*25*,*64*], 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 [*6*]. 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 [*6*]. HTML5 now expects all browsers to be able to interpret this tag. They allow us to source content from other pages [*6*], differently phrased: the project 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 [*6*].

The above are the most notable HTML5 technologies used in our presentation engine, but other powerful tools are mentioned under the related work section. They are controlled by events and custom Javascript as a part of the presentation engine (See Section 4.3 of Chapter 3).

3.2.2 Seadragon Ajax

Seadragon Ajax was the tool 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 [*44*,*43*]. 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 [*44*,*43*]. See Figure 1 in section 2.4.2.1 for a representation:

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 [*43*]. 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 [*43*].

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 [*44*]. 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 covered again later within the creation of a timeline under Section 4.3 of Chapter 3. They are normalised to values and used in the navigation to nodes, and placement of overlays.

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 [*45*]. 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 [*46*]. This creates a better sense that the overlay is part of the underlying scene. These Seadragon rectangles are vital to embedding custom content, specifically HTML5, into our presentation in order to correctly position and style the content.

3.2.1 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 [*44*]. 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 two notable items of interest: the image directory, with the image broken down into a folder structure, depicting zoom level and named for arrangement and the second is either an XML or DZI file, which describes the size of the image, and composition at each level [*43*].

This information is then used in combination with the Seadragon navigation to request tiles from certain folders depending on the given positioning within Seadragon. Seadragon then requests the image and displays it at the given position [*44*].

3.2.2 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 not be delivered through the DOM stack to underlying structures. This problem was confirmed by reading the Seadragon source code made openly available [*44*]. 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 by using Javascript to directly raise events, but the mouse interaction is limited due to Seadragon built in mouse navigation controls. 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 [*44*]. This also demonstrates why our use of CSS on our presentation will be limited.

3.2.3 Co-ordinates

In the Seadragon environment, e.g. 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 (a normalised value). 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 and the drawer (See Section 4.2.5). The Seadragon engine can automatically move smoothly (animate) from point to point by requesting the viewport to pan or zoom, allowing the web-developer to easily create a navigation system [*44*].

3.2.4 Viewport

The viewport is the currently viewable area, defined by the dimension of the Seadragon container, a level of zoom which determines the image set to select, and the point, refining the images that need selection within that folder level. The viewport also animates from area to area, giving us an event that shows the user has changed position. This event is useful in hiding pieces of the presentation that the designer may not want being seen at the overview level.

The viewport, as previously mentioned, has abilities to pan and zoom smoothly with styling. Seadragon refers to this as the ability to spring form point to point. This ability will be used in the point to point navigation, as well as the ability to read off the points in the navigation for persistence [*44*].

3.2.5 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. This is fixed to the viewport by a Seadragon point and an anchor, 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, and allow Seadragon Ajax to style it into the correct

position for the designer. 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 [*45*,*46*].

To conclude, the presentation engine has the necessary elements; to add large amount of richness to the online presentation tool, through the combination of HTML and styling. The Seadragon Ajax library is visually rich information environment with point to point navigation & animation is possible, that will be extended by custom Javascript.

3.2.3 The Overall 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 scene. To represent this information a data structure is designed. See Figure 2 for basic class diagram depicting functionality (for a full class diagram see appendices).

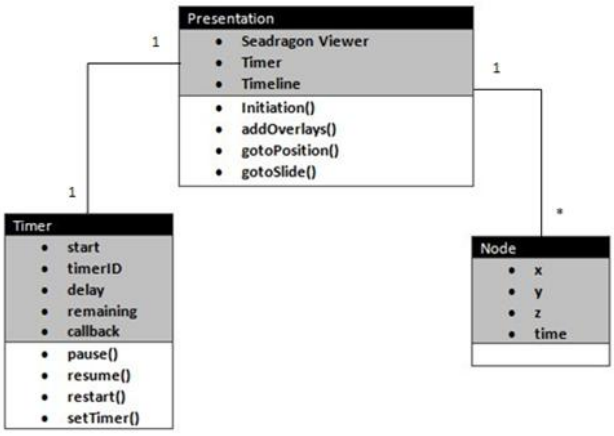


Figure 2: Incomplete Class diagram of prototype presentation (See Appendix 1 for Full Diagram)

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 [*23*].

The Seadragon viewer allows us to control our instance of Seadragon Ajax, 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***).

In order to retain this timeline it can be saved to text 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 [*18*]. 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 if they choose to.

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. This was done by evaluating a string as a function on transition to a location. A function uses the timer to recursively traverse through the timeline, with basic navigation controls from slide to slide. It is important to note Javascript's single threaded nature, and attempts to iterate though the ***timeline*** lead to consuming unnecessary computation time. 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 traversal 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.

Chapter 4: Results and Findings

This project shows that it is possible to create a RIA without Flash, and still retain viability. The level of richness obtained in this presentation engine is very similar to those of presentations made in Prezi, although the rotations that Prezi can do cause incorrect alignment with the Seadragon points when done with CSS. This was avoided due to complexities with the Seadragon mouse handler and the co-ordinate system.

A performance comparison was not performed as creating and obtaining a fair test method was not in scope due to the large differences in the languages and their operations. This project is single threaded which also puts it at a disadvantage to Flash components, but multi-threading is possible through use of web worker, but they must be asynchronously activated. These events must then be caught on finish and don't allow for alterations of the DOM within the web workers [*30*].

HTML5 still has many plans for the future and the Javascript engines behind the browsers are constantly begin updated, allowing for possible performance improvements. From the beginning of this project to the end, several browsers implement many more parts of HTML5 [*37*]. Video and canvas is now supported on all modern browsers with the exception of the consistency of the video codec [*37,1*]. Some support different codecs from one another, but the designer is required to provide each format if they want all browsers to be compatible with their web page's video.

Adobe has also finished its Beta version of Edge [*16*]. Edge is a HTML5 IDE much like Dreamweaver or Adobe Professional CS6 is for Flash. This shows Abode's readiness to become involved in an HTML5 creation and indicates there enthusiasm for the technology.

Chapter 5: Conclusions

HTML5 brings all the required technologies to compete with Flash's current capabilities without the most of the complexities, but it 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 [*2*]. 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 [*17*,*33*]. See the following for related articles to the future of both Flash and HTML [*16*,*34*,*57*,*66*].

Other HTML5 technologies were explored in the creation of this presentation engine, 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. Once this connection is made both pairs can both request and respond. Allowing the server to push changes to the client creates a powerful addition to HTTP, 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 more efficient Rich Internet Applications as a result of decreasing the network packet size [*29*]. Flash has more general networking capabilities: not only can it create TCP sockets, but it also has the capability of using User Datagram Protocol (UDP) as well [*3*].

WebGL is another technology that is driven by graphics acceleration in producing 3D artefacts and has not been explored due to our limited scope [*71*].

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

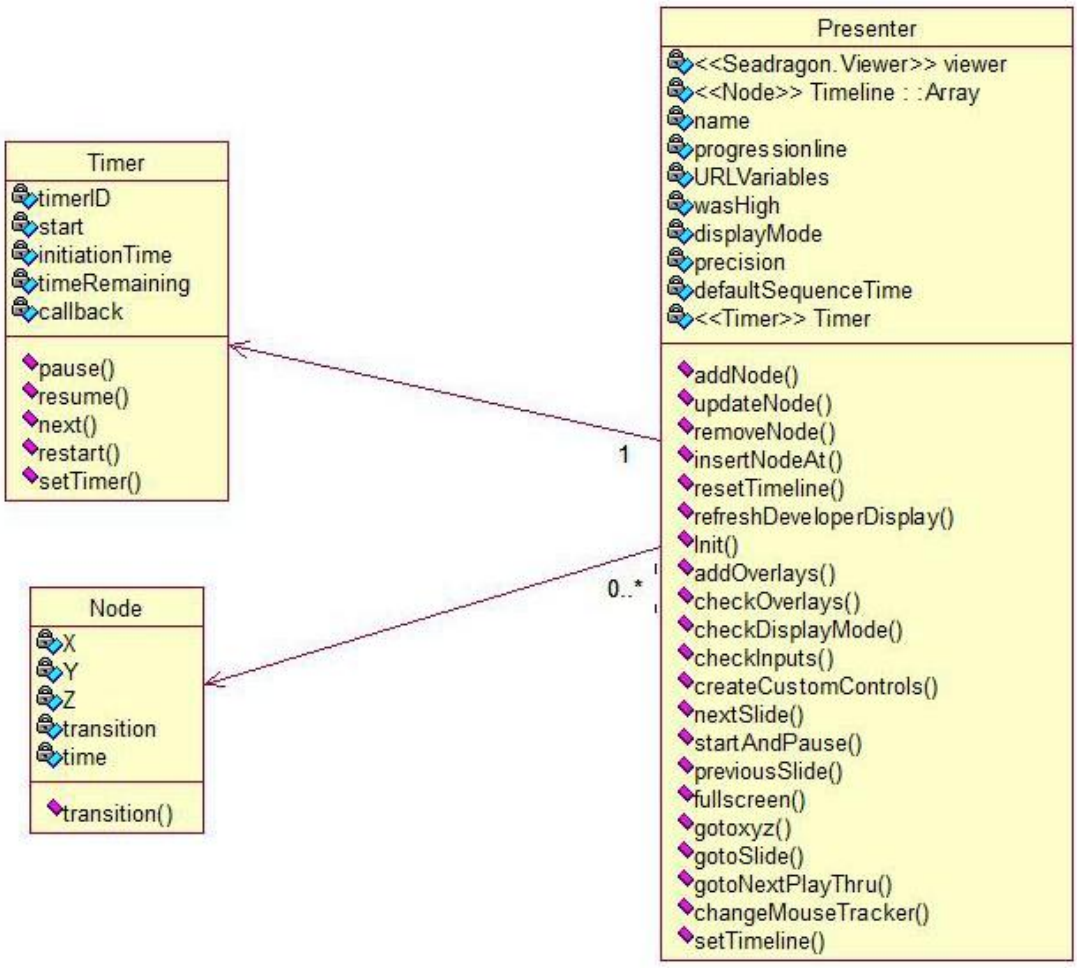
SVG elements were also explored in lesser detail, allowing for shapes and shading. More complex examples do exist and in combination with Javascript can yield powerful results [*56*,*76*]. Good use of these elements demands spending much time spent

creating shapes, or using libraries which Adobe has prefabricated in tools like Dreamweaver and Edge [*2*,*16*], and other sources [*64*].

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. Our exploration of the combination of HTML5, Javascript and Seadragon has demonstrated some of the newer features of HTML5 by showing how it is possible to build a rich presentation engine with these tools.

Appendices

Appendix 1: Class Diagram of Prototype Presentation



Appendix 2: Presentation Engine

This appendix presents the code that implements our presentation engine.

This can be found online at : http://www.cs.ru.ac.za/research/g09S0538/index.html

This code is a single HTML page that could be delivered from a server to a client. This first section embeds some standard CSS styles.

```
<!DOCTYPE html/>
<HTML>
      <style type="text/css">
        body {
            margin: 0px;
            font-family: Verdana;
            line-height: 1.25em;
        }
        #content {
            width: 600px;
            margin-left: 10px;
        }
        h1 {
            font-size: large;
            font-weight: normal;
            display: inline-block;
                  margin: 12px;
            background-color: #f0f0f0;
        }
        h1 strong {
            font-weight: bold;
        }
        h2 {
            font-size: small;
            font-weight: bold;
        }
        table {
            font-size: small;
            background-color: #faf8b6;
            border: 1px solid #f5f270;
            padding: 10px;
        }
            a {
            text-decoration: none;
        }
        p {
            font-size: small;
            background-color: #faf8b6;
            border: 1px solid #f5f270;
            padding: 10px;
        }
```

```
        p a {
            color: #444304;
            background-color: #F5F270;
            font-weight: bold;
        }
        ul {
            font-size: small;
        }
        ul a {
            color: #253649;
            background-color: #F0F0F0;
        }
        #container {
            width: auto;
            height: 500px;
            background-color: black;
            border: 1px solid black;
            color: white;   /* for error messages */
        }
            buttton{
                  margin:auto;
            }
            iframe {
               overflow: hidden;
               opacity:0.8;
            }
    </style>
```

In the Next line of code we import the Seadragon library:

```
<script type="text/javascript" src="/Pres5/js/seadragon-
min.js"></script>
```

To follow is the custom Javascript that will proceed to load the presentation into a container, as with a normal Seadragon instance, and then continue to expand on its media rich abilities:

```
    <script type="text/javascript">
var viewer;
const TIMED = 0;
var precision = 5;
progressionline = new Array();
var sequencerDefaultTime = 5000;
var timer;
timeline = new Array();
urlVars = new Array();
```

Certain Overlays are hidden at the upper zoom levels to avoid clutter in the overview:

```
var wasHigh = false; //hide certain overlays at zoom levels in-order
not to clutter page
```

```
var displayMode = false; //have slide creation tools
var PAN_DISTANCE = 0.05;


//NODES START-------------------------------------------
```

The timeline is a sequence of nodes. To implement our class diagram from Appendix 1 we require a constructor and various methods.

```
//NODES START-------------------------------------------

            function node(x,y,z,transfer,time){
                  this.x = x;
                  this.y = y;
                  this.z = z;
                  this.transition = transfer;
                  this.time = time;
                  //timing and transition?
            }

            function Timer(callback, delay) {
                  var timerId, start, initTime = delay;
                  this.remaining = delay, this.cb = callback;

                  this.pause = function() {
                        window.clearTimeout(timerId);
                        this.remaining -= new Date() - start;
                  };

                  this.resume = function() {
                        start = new Date();
                        timerId = window.setTimeout(this.cb,
this.remaining);
                  };

                  this.next = function(){
                        if(!(timerId == undefined))
                        {
                              window.clearTimeout(timerId);
                              timerId = window.setTimeout(this.cb, 1);
                        }
                  }

                  this.restart = function() {
                        window.clearTimeout(timerId);
                        this.remaining = initTime;
                        this.resume();
                  };
```

```
                this.setTimer = function(callback, delay){
                      this.cb = callback;
                      start = new Date();
                      this.remaining = delay;
                      initTime = delay;
                      timerId = window.setTimeout(this.cb,
this.remaining);
                }

                this.resume();
          }

          function addNode(){
                var xy = viewer.viewport.getCenter();
                var time;
                if(numChk()) time = _$('inputTime').value
                else time = 0;

                //set transition method if not TIMED
                if(_$('transitionselection').selectedIndex == 0){
                      timeline.push(new
node(xy.x,xy.y,viewer.viewport.getZoom(),TIMED,time));
                }
                if(_$('transitionselection').selectedIndex == 1){
                      timeline.push(new
node(xy.x,xy.y,viewer.viewport.getZoom(),_$('output_input').value,time)
);
                }

                refreshPoints();
          }

          function updateNode(node){
                timeline[node] = node;
                refreshPoints()
          }

          function removeNode(index){
                timeline.splice(index,1)
                refreshPoints()
          }

          function insertNodeAt(index,node){
                timeline = timeline.splice(index,0,node)
          }

          function insertNode(){
                var xy = viewer.viewport.getCenter();
                var time;
```

```
                if(numChk()) time = _$('inputTime').value
                else time = 0;
                if(_$('transitionselection').selectedIndex == 0){
                      timeline.splice(_$('inputPosition').value,0,new
node(xy.x,xy.y,viewer.viewport.getZoom(),TIMED,time));
                }
                if(_$('transitionselection').selectedIndex == 1){
                      timeline.splice(_$('inputPosition').value,0,new
node(xy.x,xy.y,viewer.viewport.getZoom(),_$('output_input').value,time)
);
                }

                refreshPoints()
          }

          function resetTimeline(){
                timeline = new Array()
                refreshPoints()
          }
```

The following function refreshes the displayed representation of the timeline and custom controls to zoom to each slide or remove it from the timeline.

```
          function refreshPoints(){
                var stringthing = ''
                for(i=0;i<timeline.length;i++){
                      var btn = document.createElement("Button");
                      btn.id = 'navBtn'+i
                      btn.innerHTML = i
                      btn.onclick =
'gotoxyz('+timeline[i].x+','+timeline[i].y+','+timeline[i].z+')';
                      stringthing += "<span id='node"+i+"'><button
id='navBtn"+i+"' onclick='gotoSlide("+eval(i+1)+")'>"+i+"</button>"+':
('+timeline[i].x.toFixed(precision)
+','+timeline[i].y.toFixed(precision)+') -
'+timeline[i].z.toFixed(precision) +' ~ '+((timeline[i].time=='0') ?
sequencerDefaultTime:timeline[i].time) +' ~ '+"<button
id='removeBtn"+i+"'
onclick='removeNode("+i+")'>Remove</button></span><br/>";
                }
                _$("div1").innerHTML = stringthing;
          }

          //NODES FIN----------------------------------------------
          //NODES FIN----------------------------------------------
          //INIT---------------------------------------------------
```

The following function is run on page load and proceeds to prepare the Seadragon presentation and load the DZI into the container and style it. It also initiates custom variables and event handlers.

```
function init() {
          urlVars = getUrlVars();
    viewer = new Seadragon.Viewer("container");
    viewer.openDzi("GeneratedImages/dzc_output.xml");
```

The string saved in ‘tl’ is the JSON output from our presentation engine. It represents the timeline and can be altered for various timelines within the same presentation.

```
          var tl =
JSON.parse('[{"x":0.5792939476513005,"y":0.13152145343452284,"z":7.5641
28685360022,"transition":"startVideo()","time":"95000"},{"x":0.18389397
274394287,"y":0.3012137888503841,"z":4.564128685360022,"transition":0,"
time":"45000"},{"x":0.8052278857309948,"y":0.2916156197135286,"z":2.782
06434268001,"transition":0,"time":"30000"},{"x":0.1948785678419736,"y":
0.5452667237576055,"z":5.801302948288014,"transition":0,"time":"45000"}
,{"x":0.19045,"y":0.88349,"z":2.7,"transition":"canvasDrawExample2()","
time":"40000"},{"x":0.19045,"y":0.88349,"z":2.7,"transition":"canvasDra
wExample()","time":"40000"},{"x":0.7292935784413999,"y":0.9069781203557
463,"z":2.547182422617788,"transition":0,"time":"0"}]');
          //change this JSON line for defualt presentation
          //any function under transition must be as a string
for the eval function hence "'" must be escapsed -> "\'"
          setTimeline(tl);
```

The following few lines initiate event handlers for our start-up functions. As Javascript is single threaded it requires that certain actions are performed during the page load. Javascript is event based, allowing us to run our methods on events, rather than consume the thread. A choice must be made between checking the overlays on animation, which happens every frame change, or on start and finish animation. Current settings may be more resource consuming, but allow for hiding and un-hiding during the animation.

```
    //viewer.addEventListener("animationfinish", chkOverlays);
    //viewer.addEventListener("animationstart", chkOverlays);
    viewer.addEventListener("open", gotoVarChk);
    viewer.addEventListener("animation", chkOverlays);
          viewer.addEventListener("open", addOverlays);
```

```
                viewer.addEventListener("open",
viewer.clearControls);
                viewer.addEventListener("open", chkDisplayMode);
                viewer.addEventListener("open",
createCustomControls);
                timer = new Timer();

        }
            Seadragon.Utils.addEvent(window, "load", init);

            function chkDisplayMode()
            {//display only? Or editor mode (can change timeline)
                  if (displayMode == true)
                  {
                        _$('output').hidden = true;
                        _$('outputslides').hidden = true;
                        _$('stats').hidden = true;
                        createCustomUserControls();
                  }
            }
            //INIT------------------------------------------------------
            //OVERLAYS--------------------------------------------------
```

The following function updates the styling properties for elements hidden at different zoom levels and updates the position statistics for the editor.

```
            function chkOverlays(){
                  var t = new Date();
                  var t1 = t.getTime();
                  var z = viewer.viewport.getZoom();
                  var xy = viewer.viewport.getCenter();

                  var objects = [_$("canvas1"),_$("img1"),_$("img2")];

                  if((z>2)&&wasHigh){
                  for(obj in objects)
                        objects[obj].hidden = false;
                        wasHigh = false;
                  }
                  else if ((z<2)!=wasHigh)
                  {
                  for(obj in objects)
                        objects[obj].hidden = true;
                        wasHigh = true;
                  }
                  var a = viewer.viewport.getCenter();
                  _$("lbl2").innerHTML = "("+a.x.toFixed(precision)+" :
"+a.y.toFixed(precision)+")";
```

```
                _$("lbl3").innerHTML =
viewer.viewport.getZoom().toFixed(precision);
                var t2 = new Date().getTime() - t1;
                _$("lbl4").innerHTML = t2;
            }
```

The following function demonstrates how to add various overlays at different positions and zooms within the presentation.

```
            function addOverlays(viewer) {
```

First a Video

```
                var vid = document.createElement("video");
                //INSERT VIDEO inside fixed rectangle
                vid.id = 'video1';

                vid.innerHTML = '<source src="What is HTML5.mp4"
type="video/mp4"><source src="What is HTML5.flv"
type="video/flv"><source src="What is HTML5.ogg" type="video/ogg">Your
browser does not support the video tag.';
                //vid.hidden = true;
```

Note the points within the Seadragon rectangle depicting x and y coordinates and then a width and height in Seadragon units relevant to that specific DZI. This is the position of the overlay within the composition.

```
                var rect = new Seadragon.Rect(0.49, 0.10, 0.18,
0.07);
                viewer.drawer.addOverlay(vid, rect);
```

SVG is then embedded. Note SVG must be embedded as it is served with a different document type resulting in a difference of representation and styling.

```
                var svg = document.createElement("embed");
                //INSERT VIDEO inside fixed rectangle
                svg.id = 'svg1';
                svg.src = 'http://openclipart.org/people/Eggib/js-
dom-model.svg';

                var rect = new Seadragon.Rect(0.7, 0.2, 0.25, 0.2);
                viewer.drawer.addOverlay(svg, rect);
```

Next we insert a label:

```
                    //*INSERT LABEL
                    var x = document.createElement("span");
                    x.id = 'lbl1';
                    x.innerHTML = 'Canvas & SVG';
                    var rect = new Seadragon.Rect(0.07, 0.76, 0.2,
0.13);//(x,y,w,h)
                    viewer.drawer.addOverlay(x, rect);//*/
                    //*INSERT LABEL
                    var x = document.createElement("span");
                    x.id = 'ToplblIntro';
                    x.innerHTML = 'Introduction';
                    var rect = new Seadragon.Rect(0.4, 0.05, 0.2,
0.13);//(x,y,w,h)
                    viewer.drawer.addOverlay(x, rect);//*/
```

And then we insert a Canvas:

```
                    //*INSERT Canvas
                    var canvas = document.createElement("canvas");
                    canvas.id = 'canvas1';
                    canvas.hidden = true;

                    var rect = new Seadragon.Rect(0.06, 0.79, 0.2,
0.11);//(x,y,w,h)
                    viewer.drawer.addOverlay(canvas, rect);//*/
```

Proceeded by images:

```
                    //*INSERT Image
                    var img = document.createElement("img");
                    img.id = 'img1';
                    img.src =
'http://i.msdn.microsoft.com/dynimg/IC496481.png'
                    img.hidden = true;

                    var rect = new Seadragon.Rect(0.12, 0.9, 0.2,
0.09);//(x,y,w,h)
                    viewer.drawer.addOverlay(img, rect);//*/

                    //*INSERT Image
                    var img = document.createElement("img");
                    img.id = 'img2';
                    img.src = '/Pres5/img/classdiagram.png'
                    img.hidden = true;

                    var rect = new Seadragon.Rect(0.12, 0.5, 0.14,
0.09);//(x,y,w,h)
                    viewer.drawer.addOverlay(img, rect);//*/
```

And lastly an Iframe:

```
                var iframe = document.createElement("iframe")
                //INSERT Embedded
                iframe.id = 'embeddedIframe'
                iframe.setAttribute("src",
"http://cs.ru.ac.za/research/g09s0538/ResearchPaperRev2(2).pdf");
                iframe.setAttribute("allowtransparency","true");
                //iframe.style.width...
                var rect = new Seadragon.Rect(0.05, 0.1, 0.3,
0.34);//(x,y,w,h)
                rect.className = "overlay";
                viewer.drawer.addOverlay(iframe, rect);
          }
```

Custom functions were built to control the video and draw different representations on the canvas:

```
          function startVideo(){
                var x = _$('video1');
                x.muted = true;
                x.play();
          }
          function canvasDrawExample(){
                var canvas = _$('canvas1');
                //get canvas from the DOM
                if(canvas.getContext){//check for compatability of
Canvas in the browser
                      var ctx = canvas.getContext('2d');//2D drawing
context initiated
                      ctx.clearRect(0,0,300,250);
                      //clear the canvas -- common action for
animations ->clear buffer, print to buffer, print buffer to canvas
                      ctx.strokeStyle = "black";
                      //y axis along the left edge of the canvas
                      ctx.beginPath();//path will draw from point to
point hence draw axis and plot data
                      //origin @ 10,140
                      ctx.moveTo(10,10);
                      ctx.lineTo(10,140);
                      ctx.stroke();
                      ctx.stroke();
                      // x axis along the bottom edge of the canvas
                      ctx.moveTo(10,140);
                      ctx.lineTo(250,140);
                      ctx.lineTo(250,140);
                      ctx.stroke();
                      ctx.stroke();
```

```
                        ctx.closePath();//put pen down and start new
set for color change

                        //Canvas Plot vs screen size
                        ctx.strokeStyle = "purple";
                        ctx.beginPath();
                        ctx.moveTo(10,140);
                        ctx.lineTo(50,130);
                        ctx.lineTo(95,110);
                        ctx.lineTo(200,25);
                        //ctx.arc(10, 120, 50, 0.5, 0.01, true);
                        ctx.stroke();
                        ctx.moveTo(230,90);
                        ctx.lineTo(240,90);
                        ctx.stroke();
                        ctx.closePath();

                        //SVG plot vs screen size
                        ctx.strokeStyle = "red";
                        ctx.beginPath();
                        ctx.moveTo(10,140);
                        ctx.lineTo(50,130);
                        ctx.lineTo(200,90);
                        //ctx.arc(10, 120, 50, 0.5, 0.01, true);
                        ctx.stroke();
                        ctx.moveTo(230,70);
                        ctx.lineTo(240,70);
                        ctx.stroke();
                        ctx.closePath();
                        /*
                        var canvasPlot = [1,3,6,2,20,40,90];
                        var svgPlot = [1,2,3,4,5,6,7,8];
                        //data is estimation to derive graph, values
are not specific
                        //move to origin and start drawing
                        ctx.moveTo(10,10);
                        ctx.strokeStyle = "red";
                        //y axis along the left edge of the canvas
                        ctx.beginPath();
                        j = 0;
                        for (var i in canvasPlot){
                              ctx.lineTo(j*30+10,140-(i+10));
                              ctx.stroke();
                              j++;
                        }
                        ctx.closePath();*/

                        ctx.strokeStyle = "black";
                        //draw labels
                        ctx.fillStyle = '#FFFFFF';
```

```
                        ctx.font = '16px sans-serif';
                        ctx.textBaseline = 'top';
                        ctx.strokeText('Time to Render vs Size of
Screen', 15,0);
                        ctx.fillText('Time to Render vs Size of
Screen', 15,0);
                        ctx.font = '10px sans-serif';
                        ctx.fillText('Canvas', 242,84);
                        ctx.fillText('SVG', 242,64);

                  }
                  else alert('No canvas support Found\nPlease install
the latest Google Chrome');
            }
            function canvasDrawExample2(){
                  var canvas = _$('canvas1');
                  //get canvas from the DOM
                  if(canvas.getContext){//check for compatability of
Canvas in the browser
                        var ctx = canvas.getContext('2d');//2D drawing
context initiated
                        ctx.clearRect(0,0,300,250);
                        //clear the canvas -- common action for
animations ->clear buffer, print to buffer, print buffer to canvas
                        ctx.strokeStyle = "black";
                        //y axis along the left edge of the canvas
                        ctx.beginPath();//path will draw from point to
point hence draw axis and plot data
                        //origin @ 10,140
                        ctx.moveTo(10,10);
                        ctx.lineTo(10,140);
                        ctx.stroke();
                        ctx.stroke();
                        // x axis along the bottom edge of the canvas
                        ctx.moveTo(10,140);
                        ctx.lineTo(250,140);
                        ctx.lineTo(250,140);
                        ctx.stroke();
                        ctx.stroke();
                        ctx.closePath();//put pen down and start new
set for color change

                        ctx.strokeStyle = "purple";
                        ctx.beginPath();
                        ctx.moveTo(10,140);
                        ctx.lineTo(50,130);
                        ctx.lineTo(200,90);
                        ctx.stroke();
                        ctx.moveTo(230,90);
                        ctx.lineTo(240,90);
```

```
                        ctx.stroke();
                        ctx.closePath();

                        ctx.strokeStyle = "red";
                        ctx.beginPath();
                        ctx.moveTo(10,140);
                        ctx.lineTo(50,130);
                        ctx.lineTo(95,110);
                        ctx.lineTo(200,25);
                        ctx.stroke();
                        ctx.moveTo(230,70);
                        ctx.lineTo(240,70);
                        ctx.stroke();
                        ctx.closePath();

                        ctx.strokeStyle = "black";
                        //draw labels
                        ctx.fillStyle = '#FFFFFF';
                        ctx.font = '16px sans-serif';
                        ctx.textBaseline = 'top';
                        ctx.strokeText('Time to Render vs Number of
Objects', 15,0);
                        ctx.fillText('Time to Render vs Number of
Objects', 15,0);
                        ctx.font = '10px sans-serif';
                        ctx.fillText('Canvas', 242,84);
                        ctx.fillText('SVG', 242,64);
                }
                else alert('No canvas support Found\nPlease install
the latest Google Chrome');
            }
```

Here we create custom controls to interface with the timeline and navigation (See Appendix 4 for an image):

```
            function createCustomControls()
            {//media-Player style controls created with Seadragon API

                  var playBtn = new
Seadragon.Button("Play","img/blank_rest.png","img/Play_grouphover.png",
"img/Play_hover.png","img/Play_pressed.png",null,startAndPause,null,nul
l);

                  var pauseBtn = new
Seadragon.Button("Pause","img/blank_rest.png","img/Pause_grouphover.png
","img/Pause_hover.png","img/Pause_pressed.png",timer.pause,null,null,n
ull);
                  var nextBtn = new
Seadragon.Button("Next","img/blank_rest.png","img/Next_grouphover.png",
"img/Next_hover.png","img/Next_pressed.png",nextSlide,null,null,null);
```

```
                var prevBtn = new
Seadragon.Button("Previous","img/blank_rest.png","img/Previous_grouphov
er.png","img/Previous_hover.png","img/Previous_pressed.png",previousSli
de,null,null,null);
                var FullScreenBtn = new Seadragon.Button("Full
Screen","img/blank_rest.png","img/Full_Screen_grouphover.png","img/Full
_Screen_hover.png","img/Full_Screen_pressed.png",goFullScreen,null,null
,null);

                var navBar = new
Seadragon.ButtonGroup([prevBtn,playBtn,pauseBtn,nextBtn,FullScreenBtn])
;

                viewer.addControl(navBar.elmt,
Seadragon.ControlAnchor.BOTTOM_LEFT);
            }
```

Additional controls created intended for use when mouse control is disabled (See Appendix 5)

```
            function createCustomUserControls(){
                var leftBtn = new
Seadragon.Button("Left","img/blank_rest.png","img/panLeft_grouphover.pn
g","img/panLeft_hover.png","img/panLeft_pressed.png",onPanLeft,null,nul
l,null);
                var upBtn = new
Seadragon.Button("Up","img/blank_rest.png","img/panUp_grouphover.png","
img/panUp_hover.png","img/panUp_pressed.png",onPanUp,null,null,null);
                var downBtn = new
Seadragon.Button("Down","img/blank_rest.png","img/panDown_grouphover.pn
g","img/panDown_hover.png","img/panDown_pressed.png",onPanDown,null,nul
l,null);
                var rightBtn = new
Seadragon.Button("Right","img/blank_rest.png","img/panRight_grouphover.
png","img/panRight_hover.png","img/panRight_pressed.png",onPanRight,nul
l,null,null);

                var navBar = new
Seadragon.ButtonGroup([leftBtn,upBtn,downBtn,rightBtn]);

                viewer.addControl(navBar.elmt,
Seadragon.ControlAnchor.BOTTOM_RIGHT);
                changeMouseTracker();
            }

            function goFullScreen()
            {
                viewer.isFullPage() ? viewer.setFullPage(false):
viewer.setFullPage(true);
            }
```

```
            //OVERLAYS FIN-----------------------------------------


            //NAVIGATION-------------------------------------------------
            function onPanLeft(event) {
                  viewer.viewport.panBy(new Seadragon.Point(-
PAN_DISTANCE, 0));
            }

            function onPanRight(event) {
                  viewer.viewport.panBy(new
Seadragon.Point(PAN_DISTANCE, 0));
            }

            function onPanUp(event) {
                  viewer.viewport.panBy(new Seadragon.Point(0, -
PAN_DISTANCE));
            }

            function onPanDown(event) {
                  viewer.viewport.panBy(new Seadragon.Point(0,
PAN_DISTANCE));
            }
```

Seadragon Zoom and pan functions are used to obtain an animation to the given point:

```
            function gotoxyz(x,y,z)
            {
                  viewer.viewport.panTo(new
Seadragon.Point(x,y),false);
                  viewer.viewport.zoomTo(z);
            }
```

This is a simple shortcut to obtain `getElementById` in shorthand

```
            function _$(string)
            {
                  return document.getElementById(string);
            }
```

The following function begins a process of creating a transversal of the timeline by setting the event handler on the timers' timeout to call itself with the next slide.

```
            function gotoNextPlayThru(next){
                  if(next<timeline.length){
                        gotoSlide(next+1);
```

```
      timer.setTimer('gotoNextPlayThru('+(next+1)+')',(timeline[next].t
ime == 0) ? sequencerDefaultTime : timeline[next].time);
                        return true;
                  }
                  else{ return false}
            }
```

The following function shows how to disable and enable the mouse tracking in Seadragon:

```
            function changeMouseTracker(){
                  if(viewer.isMouseNavEnabled())
viewer.setMouseNavEnabled(false)
                  else viewer.setMouseNavEnabled(true)
            }

            //NAVIGATION FIN-----------------------------------------
            //TIMLINE-----------------------------------------------
```

The URL is parsed for additional functionality

```
            function getUrlVars()
            {//JQuery functionality
                        var vars = [], hash;
                        var hashes =
window.location.href.slice(window.location.href.indexOf('?') +
1).split('&');
                        for(var i = 0; i < hashes.length; i++)
                        {
                                    hash = hashes[i].split('=');
                                    vars.push(hash[0]);
                                    vars[hash[0]] = hash[1];
                        }
                        return vars;
            }
```

The following are checks and verifications on user input to ensure the data is as expected:

```
            //checks------------------------
            function gotoVarChk()
            {
                        var chk= false;
                        if (!(typeof urlVars["timeline"] ===
"undefined"))
                        {
```

```
                                    _$('output_input').value =
htmlDecode(urlVars["timeline"]);
                                    chk = loadJSONPresentation();
                        }
                        if (!(typeof urlVars["slide"] ===
"undefined")&&(timeline.length>urlVars["slide"])){gotoSlide(urlVars["sl
ide"])}
                        if (!(typeof urlVars["displayMode"] ===
"undefined")){displayMode =
(htmlDecode(urlVars["displayMode"])==='true')}
            }

            function numChk()
            {
                  var x = _$('inputTime')
                  if(x.value == null || x.value.length ==0){
                        alert("no input")
                        x.focus();
                        x.value = 0;
                        return false
                  }
                  else if ((x.value-0) == x.value && x.value.length>0)
                  {
                        return true;
                  }
                  else {alert('not a number');x.focus();return false;}
            }
            //checks------------------------

            function htmlDecode(input)
            {
                        input = input.replace(/%22/g,'"');
                        return input.length === 0 ? "" : input;
            }
```

The following are functions used to progress through the timeline in a media player style:

```
            //slide navigation----------
            function nextSlide(){timer.next();}
            function startAndPause()
            {
                  if(timer.remaining == undefined){
                        gotoNextPlayThru(0);//start playing thru the
slides
                  }
                  else if (timer.cb ==
"gotoNextPlayThru(undefined)"){//if custom slide progression had led to
no call back
                        gotoNextPlayThru(0);
                  }
```

```
                  else timer.resume()//else resume
            }

            function previousSlide(){
                  if(progressionline.length <=0){
                        gotoSlide(1);
                  }
                  else{
                        progressionline.pop();//pop the current slide
                        timer.pause();
                        var x = progressionline.pop();//pop the
previous slide and keep it
                        if(x == undefined)
                        {
                              gotoSlide(1);
                        }
                        else{

      timer.setTimer('gotoNextPlayThru('+x+')',1);
                        }
                        //goto a previous slide and when resumed,
proceed to the next slide from the most recently set callback in the
timer.
                  }
            }

            function gotoSlide(i){
                        if(i>=0&&i<=timeline.length)
                        {
                                    var current = i-1;

      gotoxyz(timeline[current].x,timeline[current].y,timeline[current]
.z);
      eval(timeline[current].transition);
      //eval is not a cheap function :/
      timer.cb = 'gotoNextPlayThru('+i+')';//set call back for next
button
      timer.remaining = 1;

      progressionline.push(current);
      refreshPoints();
      _$('node'+(current)).style.fontWeight = 'bold';
                        }
            }
            //slide navigation----------

            //TIMELINE SAVE/LOAD functionality
            function setTimeline(obj){
                        timeline = obj;
                        refreshPoints();
```

```
                        return true;
            }
```

The input and output of our timeline in a JSON representation:

```
            function saveTimeline()
            { return JSON.stringify(timeline); }

            function loadJSONPresentation(){
                        var JSONstring = _$('output_input').value;
                        var tl = JSON.parse(JSONstring);
                        //is it really a timeline (*do check)??
security hole here -- but it is all client side activity anyway :)
                        if(setTimeline(tl)==true)return true;
            }
            function saveJSONPresentation(){
                        _$('output_input').value = saveTimeline();
            }
            //TIMLINE-----------------------------------------------
    </script>
</head>
```

This is the end of the custom scripting section, and continues to the initial HTML of the page which is used as a base to start the presentation engine.

```
<body>
      <header><h1>Prototype Presentation Engine</h1></header>
      <br/>
```

To follow is the editor interface aligned to the right as seen in Appendix 3

```
      <!-- Slide GUI -->
      <table id="outputslides" style="clear:
both;width:520px;float:right; margin-left: 10px;margin-right:
10px;margin-top: 10px;" border='1'>
            <tr>
                  <th colspan="3">'Zoom To' Interaction</th>
            </tr> <tr>
                  <td><button id="btn6" onclick="gotoNextPlayThru(0);">
Do a Play-Through </button></td>
                  <td><button id="btn7" onclick="addNode();"> Save
Position </button></td>
                  <td>
                        <button id="btn7" style='float:right'
onclick="resetTimeline();"> Reset </button>
                        <button id="btn8" style='float:right'
onclick="changeMouseTracker();"> Mouse Control </button>
```

```
                </td>
            </tr> <tr>
                <td>
                    <select id="transitionselection"
style="float:right; margin-left: 10px;margin-right: 10px;margin-top:
10px;">
                        <option value='TimedTransition'> Timed
Transition </option>
                        <option value='OnEvent'> Event Driven
</option>
                    </select>
                </td>
                <td><span> Time on Slide (ms)</span></td>
                <td><input id="inputTime" value='0' onblur='numChk()'
style='margin:10px'></input></td>
            </tr> <tr>
                <td><button id="insertNodeBtn"
onclick="insertNode();"> Insert Position </button></td>
                <td><span> Set new Node Position </span></td>
                <td><input id="inputPosition" value='0'
onblur='numChk()' style='margin:10px'></td>
            </tr> <tr>
                <td colspan='3'><div id='div1'></div></td>
            </tr>
    </table>
      <table id="output" style="clear: both;width:520px;float:right;
margin-left: 10px;margin-right: 10px;margin-top: 10px;" border='1'>
                <th>Input & Output</th>
                <tr>
                    <td colspan='2' style="width:100%;margin:auto">
```

The following text area is used to save functions on event transition slides, allowing typed functions to be stored as strings and later evaluated as functions (See Appendix 6). This area is also used as an input and output for a JSON representation of the timeline (See Appendix 3).

```
<textarea id='output_input'>Test</textarea></td>
                </tr>
                <tr>
                    <td><button id='loadbtn'
onclick="loadJSONPresentation()">Load JSON Presentation</button></td>
                    <td><button id='savebtn'
onclick="saveJSONPresentation()">Print JSON Presentation</button></td>
                </tr>
      </table>
      <div id="content">
    <h1><strong>Seadragon</strong> Ajax</h1>
```

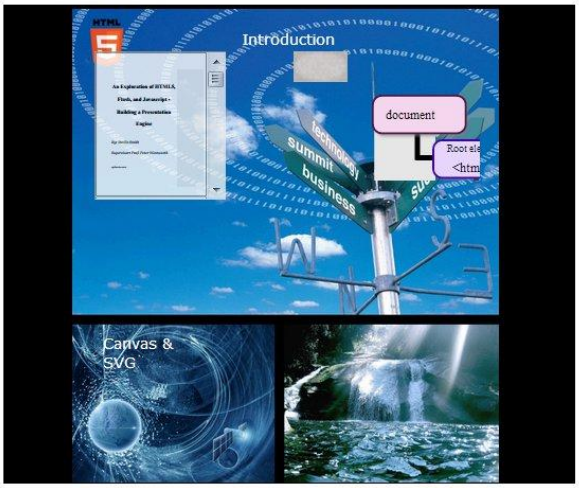
Next; the Seadragon container required for its initial construction and its DZI display and the table that is used to show current location within the presentation.

```
<!-- Seadragon Ajax container -->
    <div id="container"></div>
      <br/>
      <!-- Stats Table-->
      <table id='stats'>
            <tr>
                  <td/><td><h2>Position Statistics</h2></td>
            </tr> <tr>
                  <td>(X:Y)</td><td><span id='lbl2'/></td>
            </tr> <tr>
                  <td>Zoom</td><td><span id='lbl3'/></td>
            </tr> <tr>
                  <td>Time Between Animations</td><td><span
id='lbl4'/></td>
            </tr>
      </table>
    <br/>
</div>
</body>
</html>
```

Appendix 3: Overview of the presentation engine with JSON ouput of the timeline

Prototype Presentation Engine

Seadragon Ajax



Position Statistics	
(X:Y)	(0.50634 : 0.54982)
Zoom	0.75046
Time Between Animations	0

'Zoom To' Interaction

Do a Play-Through | Save Position | Mouse Control | Reset

Timed Transition | Time on Slide (ms) | 0

Insert Position | Set new Node Position | 0

0 : (0.57929,0.13152) - 7.56413 ~ 95000 ~ Remove
1 : (0.18389,0.30121) - 4.56413 ~ 45000 ~ Remove
2 : (0.80523,0.29162) - 2.78206 ~ 30000 ~ Remove
3 : (0.19488,0.54527) - 5.80130 ~ 45000 ~ Remove
4 : (0.19045,0.88349) - 2.70000 ~ 40000 ~ Remove
5 : (0.19045,0.88349) - 2.70000 ~ 40000 ~ Remove
6 : (0.72929,0.90698) - 2.54718 ~ 5000 ~ Remove

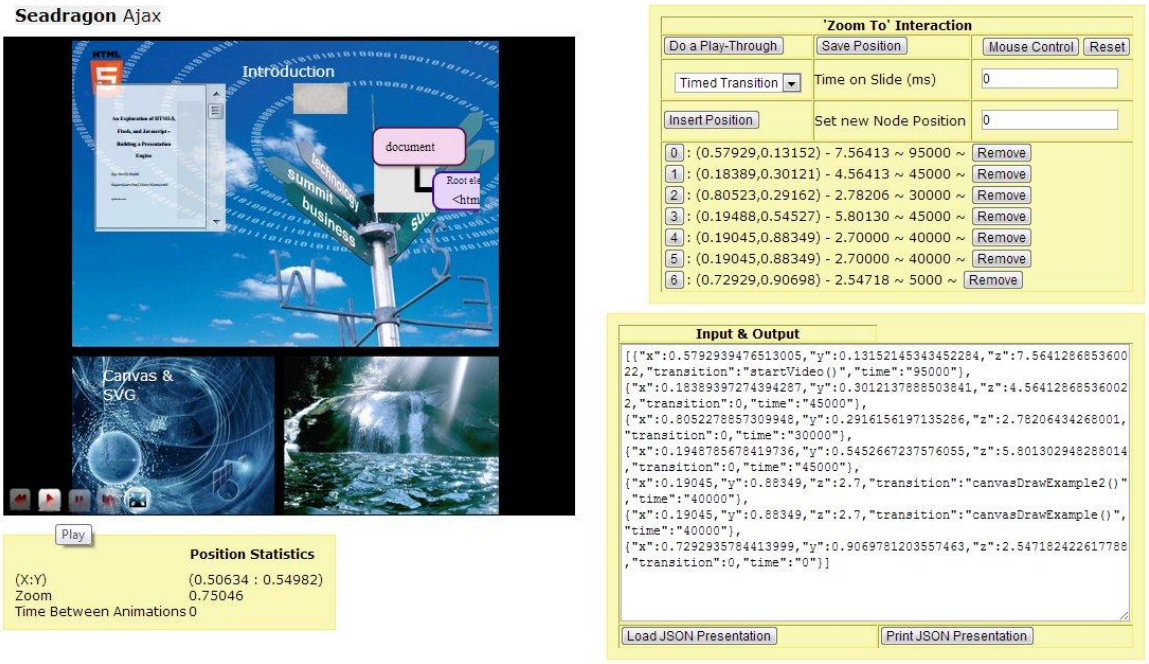
Input & Output


```
[{"x":0.5792939476513005,"y":0.13152145343452284,"z":7.564128685360022,"transition":"startVideo()","time":"95000"},
{"x":0.18389397274394287,"y":0.3012137888503841,"z":4.564128685360022,"transition":0,"time":"45000"},
{"x":0.8052278857309948,"y":0.2916156197135286,"z":2.78206434268001,"transition":0,"time":"30000"},
{"x":0.1948785678419736,"y":0.5452667237576055,"z":5.801302948288014,"transition":0,"time":"45000"},
{"x":0.19045,"y":0.88349,"z":2.7,"transition":"canvasDrawExample2()","time":"40000"},
{"x":0.19045,"y":0.88349,"z":2.7,"transition":"canvasDrawExample()","time":"40000"},
{"x":0.7292935784413999,"y":0.9069781203557463,"z":2.547182422617788,"transition":0,"time":"0"}]
```


Load JSON Presentation | Print JSON Presentation

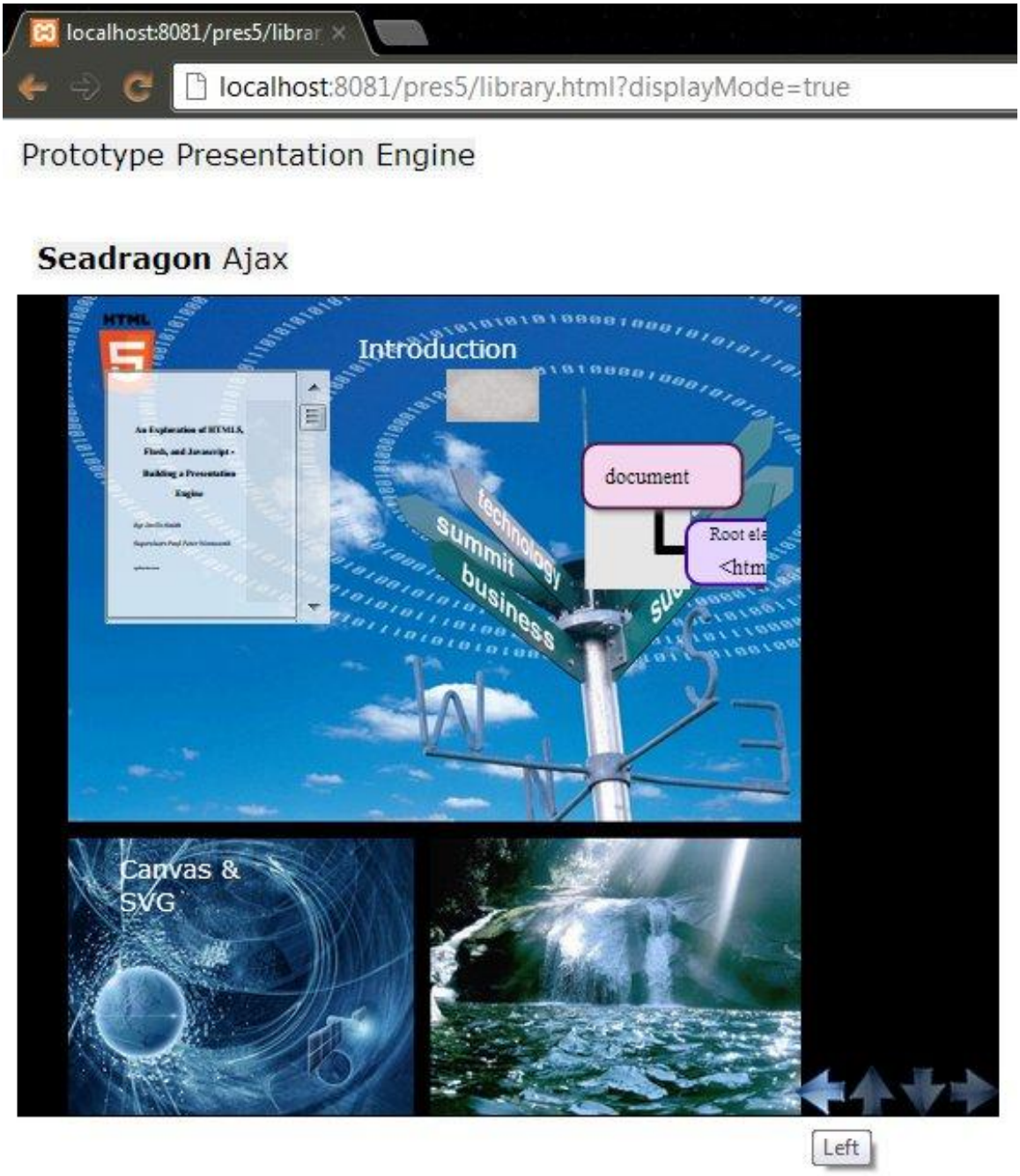
Appendix 4: Showing Custom timeline player controls

Prototype Presentation Engine



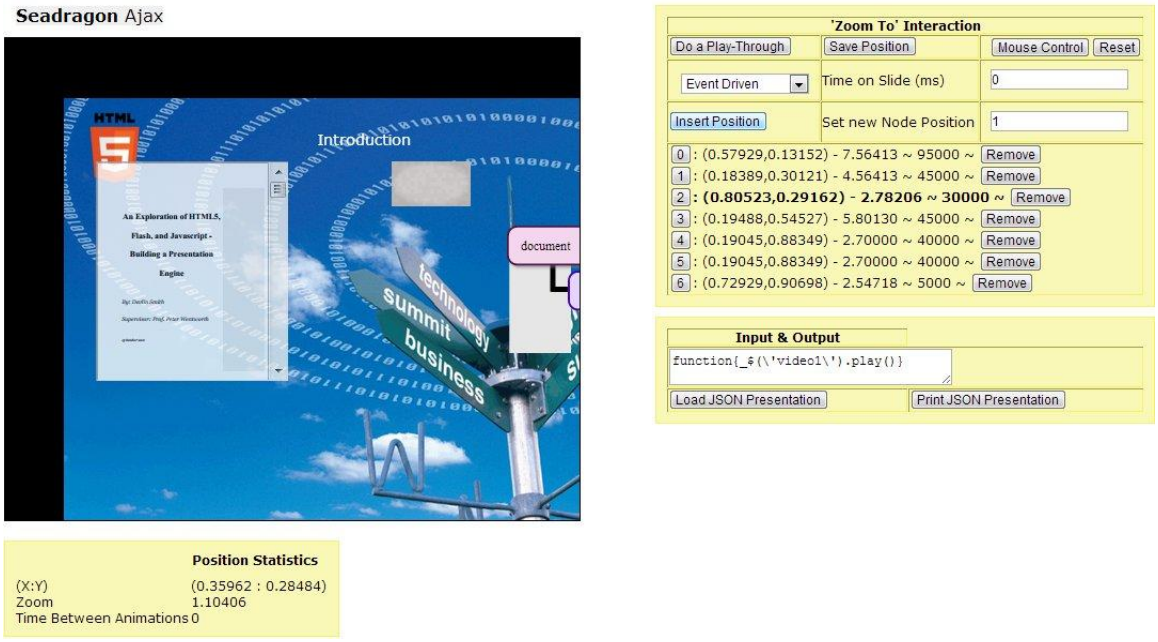
Appendix 5: URL parsing

More custom controls given because mouse control is disabled



Appendix 6: Custom function for event transitions

Prototype Presentation Engine



Bibliography

[1] Adobe. (2010) Adobe. [Online]. http://download.macromedia.com/f4v/video_file_format_spec_v10_1.pdf

[2] Adobe. (2011) Adobe. [Online]. http://www.adobe.com/africa/products/dreamweaver.html

[3] J. Allaire, "Macromedia Flash MX—A next-generation rich client," *Macromedia white paper*, vol. March, pp. 1-2, 2002.

[4] O. Andersson et al., "Scalable Vector Graphics (SVG) 1.1 Specification," *World Wide Web Consortium (W3C), Recommendation*, vol. 14, pp. 1-389, 2003.

[5] Anonymous. (2010, September) Open Seadragon. [Online]. http://openseadragon.codeplex.com

[6] Robin, Travis Leithead Erika Doyle Navara Edward O'Connor Berjon and Silvia Pfeiffer. (2012) W3c. [Online]. http://dev.w3.org/html5/spec/single-page.html

[7] T. Berners-Lee, CERN (Mar. 1989 and May 1990)"Information Management: A Proposal", 2006.

[8] T. Berners-Lee. (1998) Semantic Web Road Map. [Online]. http://student.bus.olemiss.edu/files/conlon/Others/Others/semantic%20web%20papers/roadmap.pdf

[9] T. Berners-Lee, "The Original HTTP as defined in 1991," *World Wide Web Consortium (W3C)*, 1991.

[10] T. Berners-Lee and D. Connolly, "Hypertext markup language-2.0," RFC 1866, November, Tech. rep. 1995.

[11] T. Berners-Lee and R.T. Fielding, "Hypertext transfer protocol--HTTP/1.0," RFC 1945, Tech. rep. 1996.

[12] Boss Bert. (2012) W3C. [Online]. http://www.w3.org/Style/CSS/

[13] Eric Bidelman. (2012) html5rocks. [Online]. http://slides.html5rocks.com

[14] Dennis J. Bouvier, "The state of HTML," *SIGICE Bull.*, vol. 21, no. 2, pp. 8-13, #oct# 1995. [Online]. http://doi.acm.org/10.1145/220230.220236

[15] T. Bray, J. Paoli, and CM Sperberg-McQueen, Extensible Markup Language 1.0 Specification, 2008.

[16] Peter Bright. (2012, October) Arstechnica. [Online]. http://arstechnica.com/information-technology/2012/09/adobes-continuing-revolution-pushes-the-cutting-edge-of-html5-development

[17] CreateJS. (2012) Createjs. [Online]. http://www.createjs.com

[18] D. Crockford, "JSON: The fat-free alternative to XML," in *Proc. of XML*, vol. 2006, 2006.

[19] J., Wright J. Dietrich, "Requirements for rich internet application design methodologies," *Web Information Systems Engineering-WISE 2008*, vol. 5175, pp. 106-119, 2008.

[20] Brendan Eich. (2012, 15 April.) Brendan Eich's Blog. [Online]. http://brendaneich.com/

[21] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext transfer protocol--HTTP/1.1, Jan. 1997," RFC 2068, Tech. rep. 1997.

[22] R. Fielding et al., "Hypertext transfer protocol--HTTP/1.1, June 1999," RFC 2616, Tech. rep. 2002.

[23] D. Flanagan, *JavaScript: the definitive guide*.: O'Reilly Media, Incorporated, 2006.

[24] Adobe Flash, ActionScript 3.0, September 2012.

[25] J.D. Foley, A. Van Dam, S.K. Feiner, J.F. Hughes, and R.L. Phillips, *Introduction to computer graphics*.: Addison-Wesley, 1994, vol. 55.

[26] J.J. Garrett. (2005) Adaptive Path. [Online]. http://adaptivepath.com/ideas/ajax-new-approach-web-applications

[27] Google. (2012) Google. [Online]. https://www.google.com/chrome

[28] Google. (2012) Google Developers. [Online]. https://developers.google.com/chrome-developer-tools/docs/overview

[29] Ian Hickson. (2011, April) W3C. [Online]. http://www.w3.org/TR/2011/WD-websockets-20110419

[30] I. Hickson and D. Hyatt, "HTML5—A vocabulary and associated APIs for HTML and XHTML—W3C Working Draft Jun. 10, 2008, published at http://www. w3. org/TR/2008/WD-htm15-20080122," *Jun*, vol. 10, p. 37, 2008.

[31] B. Hoehrmann, "Scripting Media Types," RFC 4329, Tech. rep. 2006.

[32] Bob, Rob Larsen Marc Neuwirth Holt. (2012) CanvasJS. [Online]. https://github.com/roblarsen/CanvasJS

[33] Steve Jobs. (2010) apple. [Online]. http://www.apple.com/hotnews/thoughts-on-flash

[34] R. Khare and S. Lawrence, "Upgrading to TLS within HTTP/1.1," RFC 2817, May, Tech. rep. 2000.

[35] P. Le H, R. Whitmer, and L. Wood, Document Object Model (DOM). W3C recommendation, 2005.

[36] H. Lie, B. Bos, and C. Lilley, "The text/css media type," RFC 2318, Tech. rep. 1998.

[37] H.W. Lie, H.W. Lie, and B. Bos, *Cascading style sheets: Designing for the web*.: Addison-Wesley Professional, 2005.

[38] Adobe Macromedia. (2012) Adobe. [Online]. http://get.adobe.com/flashplayer

[39] L. Masinter and E. Nebel, "Form-based File Upload in HTML," RFC 1867, November, Tech. rep. 1995.

[40] Eric Meyer A. and Bert Boss. (2001) W3C. [Online]. http://www.w3.org/TR/2001/WD-css3-roadmap-20010523

[41] Microsoft. (2010, July) Microsoft Expression. [Online]. http://expression.microsoft.com/en-us/gg430298

[42] Microsoft. (2011, February) Microsoft Expression. [Online]. http://gallery.expression.microsoft.com/SeadragonAjax

[43] Microsoft. (2011) Microsoft Expression. [Online]. http://expression.microsoft.com/en-us/gg413355

[44] Microsoft. (2011) Microsoft Expression. [Online]. http://expression.microsoft.com/en-us/gg413346

[45] Microsoft. (2012) Microsoft Expression. [Online]. www.microsoft.com/silverlight/

[46] K. Moore, "MIME (Multipurpose Internet Mail Extensions) part two: Message header extensions for non-ascii text," RFC 1522, Tech. rep. 1993.

[47] Mozilla. (2012) Firebug. [Online]. http://getfirebug.com/javascript

[48] Mozilla. (2012) Mozilla. [Online]. www.mozilla.org/en-US/firefox/new

[49] A. Neumann and A.M. Winter. (2001) Carto - Time for SVG. [Online]. http://carto.net/svg/articles/paper_icc_congress_china_2001.pdf

[50] M. Nottingham and R. Fielding, "Additional HTTP Status Codes," RFC 6585, Tech. rep. 2012.

[51] M. Nottingham and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)," RFC 5785, Tech. rep. 2010.

[52] Opera. (2012) Opera. [Online]. www.opera.com

[53] Addy Osmani. (2011) Netmagazine - 20 SVG uses that will make your jaw drop. [Online]. http://www.netmagazine.com/features/20-svg-uses-will-make-your-jaw-drop

[54] S. Pfeiffer, "The Definitive Guide to HTML5 Video," in *The Definitive Guide to HTML5 Video*.: Apress, 2010, p. 7.

[55] M. Pilgrim, *HTML5: up and running*.: O'Reilly Media, 2010.

[56] Prezi. (2012) Prezi. [Online]. http://prezi.com/

[57] D. Raggett, *HTML 3.2 reference specification*.: W3C Recommendation, 1997.

[58] D. Raggett, A. Le Hors, and I. Jacobs, *HTML 4.01 Specification: W3C Recommendation 24 December 1999*.: iUniverse, 1999.

[59] D. Raggett, A. Le Hors, I. Jacobs, and others, *HTML 4.0 Specification*.: W3C Recommendation, 1997.

[60] J. Reschke, "Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)," RFC 2616, Tech. rep. 2011.

[61] David Rousset, The Complete Guide to Building HTML5 Games with Canvas and SVG.

[62] Jeff. URL (2010). Schiller. (2010) codedread. [Online]. http://www.codedread.com/svg-support.php

[63] Stan Schroeder. (2012) Mashable. [Online]. http://mashable.com/2012/06/29/flash-in-android-4-1/

[64] J. Seidman, "A Proposed Extension to HTML: Client-Side Image Maps," RFC 1980, Tech. rep. 1996.

[65] C. Severance, "JavaScript: Designing a Language in 10 Days," *Computer*, vol. 45, no. 2, pp. 7-8, 2012.

[66] M. Stepp, J. Miller, and V. Kirst, "A CS 1.5 introduction to web programming," in *ACM SIGCSE Bulletin*, vol. 41, 2009, pp. 122-124.

[67] Bartek Szopka. (2012) impressjs. [Online]. https://github.com/bartaz/impress.js

[68] Gregg Tavares. (2012) html5rocks. [Online]. http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/

[69] Alvin Toffler, *The Third Wave, chapter The rise of the prosumer*, 3rd ed.: Random House, 1981.

[70] A. van Kesteren. (2008) W3C. [Online]. http://www.w3.org/TR/2008/WD-html5-diff-20080122/

[71] W3C. (2012) W3C. [Online]. http://www.w3.org/Consortium/facts#people

[72] W3C. (2012) W3C. [Online]. http://www.w3.org/standards/techs/css#w3c_all

[73] Andre M., Andreas Neumanns Winter. (2011) Carto. [Online]. http://www.carto.net/svg/samples

[74] Wix. (2012) Wix. [Online]. www.wix.com