University of Waikato

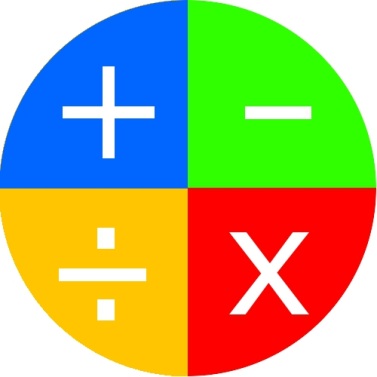
COMP333-14A

Web Application Development

Assignment 4 Report

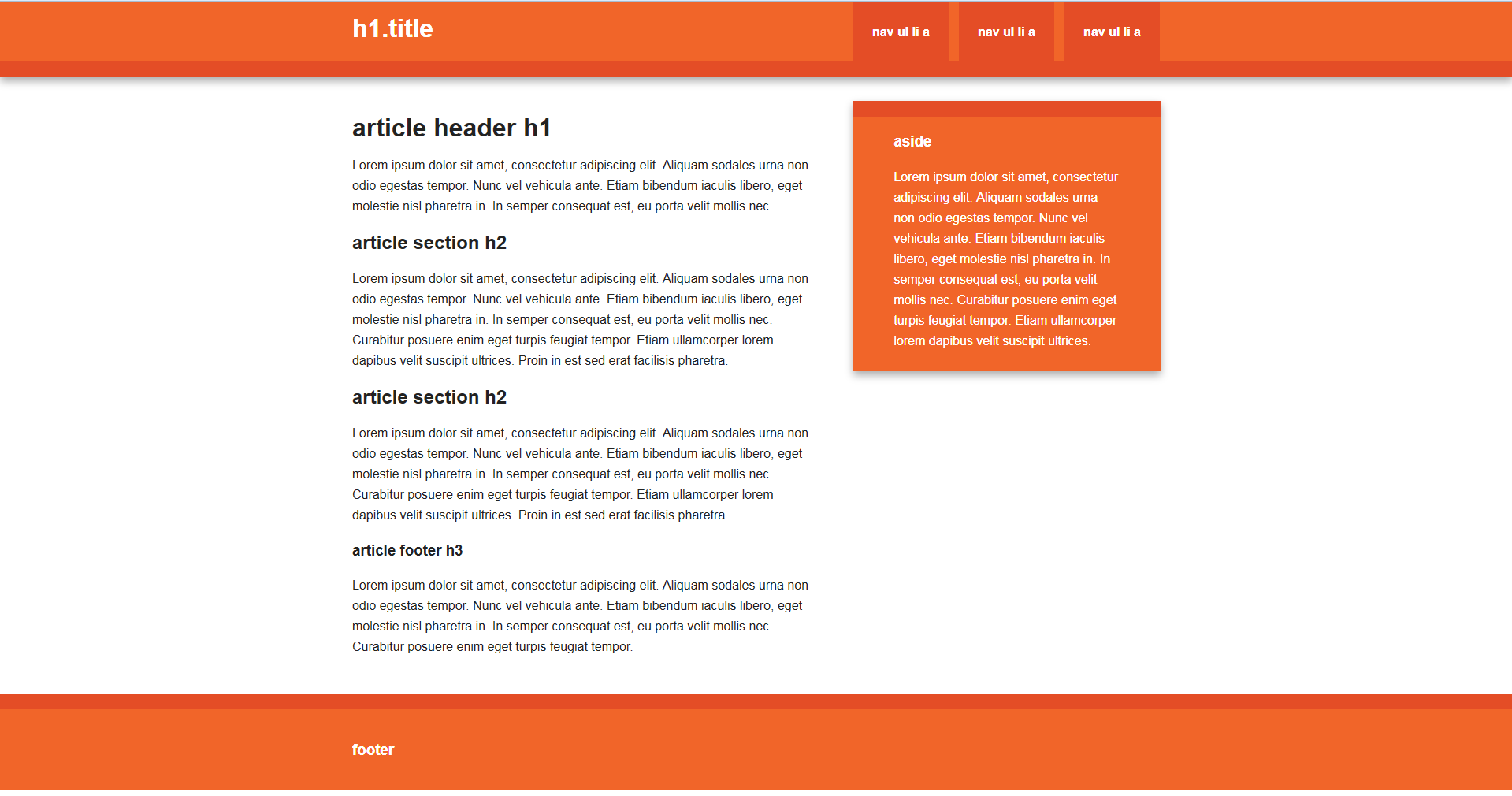
**Red Hats**

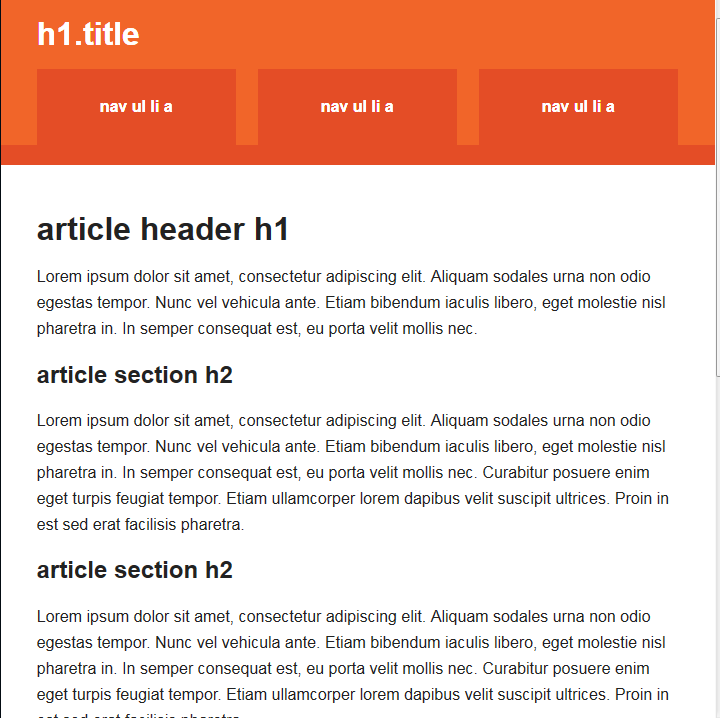
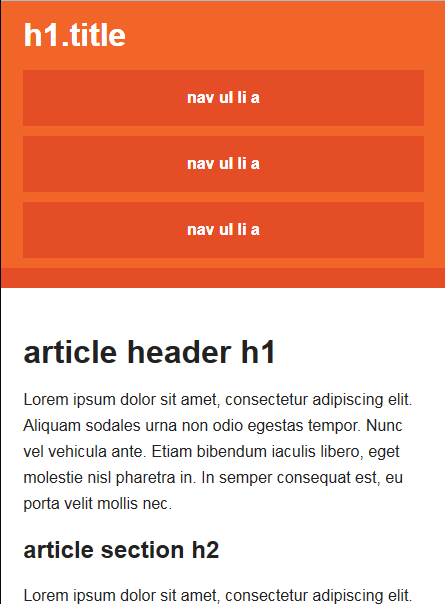
**Math Racer Development and Design Decisions Report**

****

Red Hats: Jarred Green, James Ingerson, Wanida Taylor, Jordan Fisher.

Frameworks:

Math Racer is supported by Initializr, an HTML5 template generator that builds a basic responsive web application layout using HTML5 Boilerplate to kick-start project development.



*Figure 1: Initializrs responsive design template. All three major screen ratios are catered straight out of the box.*

Why Initializr?

Initializr was the template of choice for our application for a number of reasons:

**We got to avoid having to start from nothing.**

By choosing to use an existing template we had a head start on our applications development. Initializr gave us a nice basic design with minimal overhead, allowing us to get on with the more important parts of the project.

**Easy to install and use.**

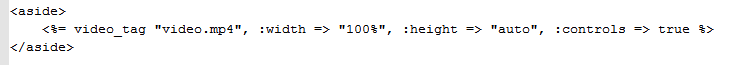
Bringing Initializr over to our existing project was as simple as possible. With just a couple of css files and a few minor changes to our core application html page we’d gone from nothing to a fully responsive, multi-browser supporting, slick looking site.

**It’s responsive straight out of the box.**

As seen in figure 1, Initializr offers support for all three popular device sizes; tablet, mobile and desktop, just by setting it up. Just by choosing to use Initializr we’d already made our web application responsive.

HTML5:

Video Embedding:



*Figure 2: Ruby code to generate an HTML5 video tag.*



*Figure 3: Resulting HTML5 video tag generated by the Ruby code, viewed using page source.*

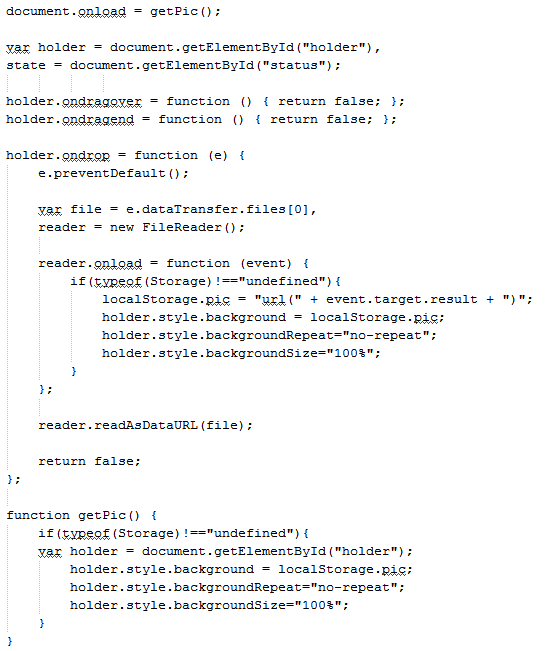


*Figure 4: Resulting HTML5 Video tag generated by the Ruby code when application is deployed on Heroku, viewed using page source.*

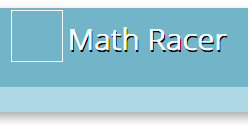
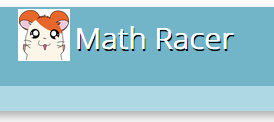
To embed the video in our about page was not as straightforward as we thought. Initially we had just used the HTML5 video tag with a couple of attributes however we ran into some issues with Ruby looking in the wrong place for the video source.

To remedy this we used the Ruby video\_tag helper (figure 2) which takes parameters to build an HTML5 video tag. This produces the HTML5 tag you can see in figure 3 when running the code locally and the tag in figure 4 when running on Heroku.

Web Storage and File Drag-and-Drop:



*Figure 5: Web Storage and File Drag-and-Drop implementation code.*



*Figure 6: Before and after of Web Storage and File Drag-and-Drop behaviour.*



*Figure 7: Result of storing image in web storage.*

In order to implement web storage and file drag and drop we put a space in the header at the top of the page for the users to drag and drop and image of their choosing onto. This image is saved into web storage so that it can be seen on every page the user visits until they decide to change it or delete it from local storage.

Walkthrough of code seen in figure 5:

Make the method that grabs the picture from web storage and display it happen on load.

Grab the div that is used to display the image.

Deal with the drag over and drag end events.

When a drop occurs:

Prevent the default action.

Read the file.

Make a file reader.

Upon loading the reader:

Check that the browser supports local storage, if it does:

Store the image and place it in the div to display it.

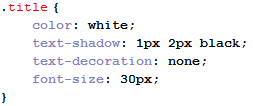
Get image:

Check that the browser supports local storage, if it does:

Store the image and place it in the div to display it.

The result of this is shown in figure 6, if there is no image an empty box is displayed. The user can choose to drag an image onto that space which will be handled by the file API and stored into web storage and it will be shown on the page and all other pages of the site. This is achieved by storing the image in the browser using web storage as seen in figure 7.

CSS3:

Text-Shadow:

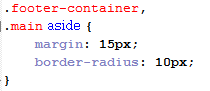
*Figure 8: CSS rule for text shadow.*



*Figure 9: Result of CSS rule.*

Text shadow was applied to the title to make it more distinguished. This was achieved by adding the text-shadow rule to the title class.

Border-Radius:



*Figure 10: CSS rule for border radius.*



*Figure 11: Result of CSS rule.*

Border radius was added to the footer and the aside child of main to give those elements curved edges.