**Criterion E: Product development**

Complex Techniques used to Satisfy Client’s Requirements

* Use of Cascading Style Sheets to customize visual appearance of website
* Use of @media queries
* Use of PHP to auto-generate content
* Use of JavaScript to implement page turn functionality in reader

**Use of Cascading Style Sheets (CSS)**

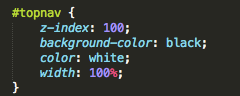
CSS stands for Cascading Style Sheets, and is a markup language used to specify the visual styling one wants to apply to the html of a website.

The CSS used within these pages was written manually into a text editor (Sublime Text). Almost all pages in the site contain these links:



These links each lead to a CSS resource. The rel="stylesheet" part specifies that the files being linked are style sheet files, the type="text/css" states the content of the files are text based and that they are .css files, and the href=”something” parts specify the location of the files being linked.

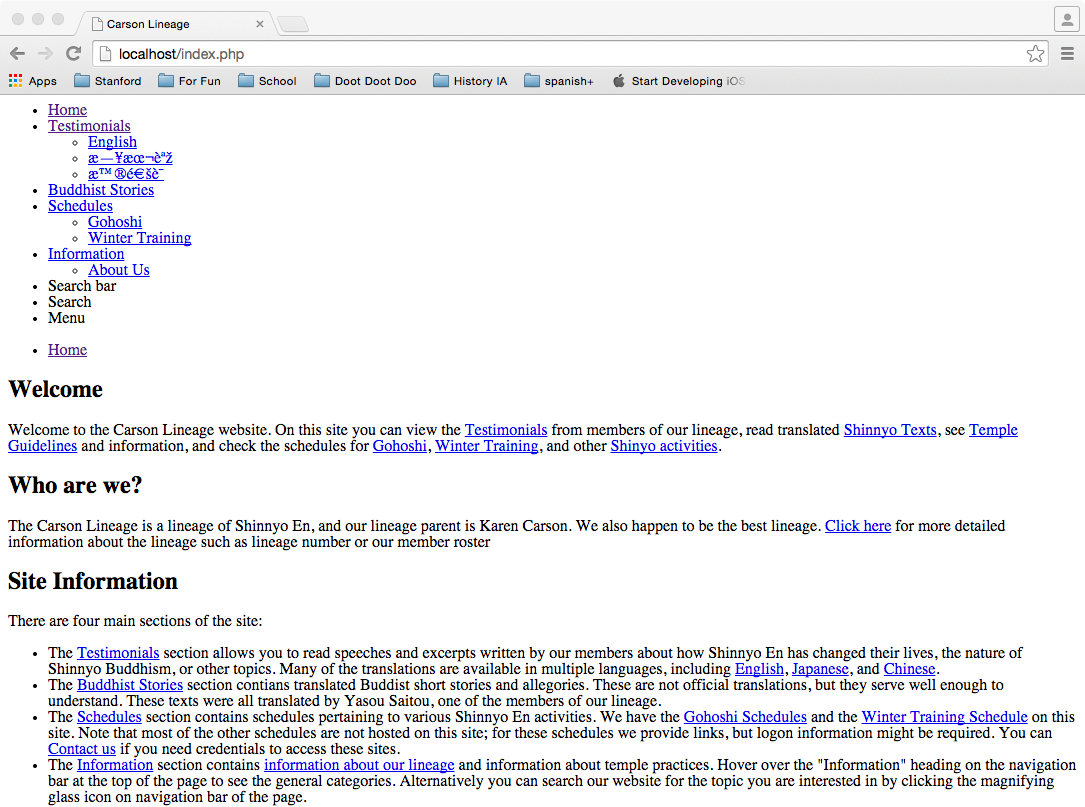
Note that there is a “/” at the beginning of the URLs within the href attribute. The “/” specifies that the path followed should start from the root of the file system (and are relative to the root of the file system, rather than the file this series of <link>s are declared in). Because of this, generally these links will not work if they are run locally, unless the root of the website is placed in the root of the computer’s file system. However, this does work when the site is uploaded, and has the benefit of making my <link> tags consistent throughout the site, reducing the need to refactor each page (this same technique is used with the header).

The actual CSS style sheets look like the image to the left. The block shown selects all elements with the ID “topnav” (which happens to be a <div> that is our navigation bar). This block sets visual attributes of our navigation bar.

Specifically, what we are doing is setting the z-index (the precedence of ‘height’ on the screen) to 100, we set the background color to black, we set the text color to white, and we set the width of the navigation bar to be 100% of the width of the element that is around it (which is the <body> element).

Of course, this is not the only CSS styling that I apply; overall, I apply CSS styling to over 16 elements, creating a visually appealing website. Examples are on the next page.

Before CSS:



Between this shot and the next I changed the styling on quite a few elements.

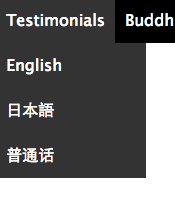
For the list of links at the very top, I changed it so that the second level of bullets would be hidden, and so that the first level of links would align horizontally rather than moving out vertically. This had the effect of making a menu bar that looks like this:



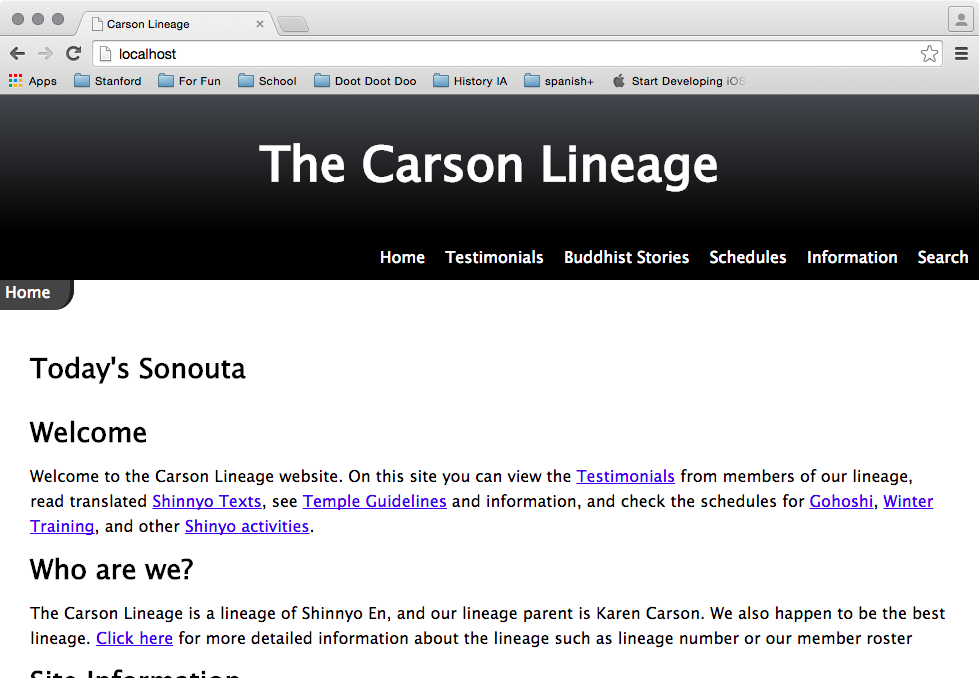
Secondly, I used the CSS pseudo-class :hover to select the sub-bullets of menu buttons that I hovered over and make them visible with this code (notice the li:hover selects the hovered-over menu items):



Which produces an effect like this when some of the menu items are hovered over:

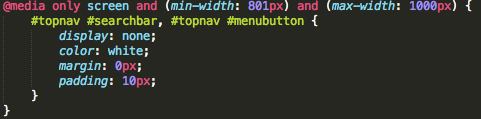


I applied less detailed visual changes to the rest of the site’s elements, and so the page now looks like this after CSS:



**Use of @media Queries**

Alongside the general CSS, I implement CSS that changes with window size using @media queries such as



The media tag ensures that the styling specified only applies when certain things are true about the window the site is being viewed on. This media query ensures that the styling inside only applies when the screen that the website is being viewed on is between 800 and 1000 pixels wide.

This creates adaptive styling, ensuring that the menu bar items do not overlap each other. For example, when the screen is not very wide (less than 800 pixels) then the navigation bar becomes resolved to two buttons, like the picture on the left. This ensures that mobile users can get an adequate website experience.

**Use of PHP to Augment Scalability and Auto-generate Content**

PHP: Hypertext Processor is a scripting language that processes HTML on the server before serving it to someone requesting the webpage.

Because I am using just a text editor instead of Dreamweaver (because I am much more comfortable with Unix-like systems and my free trial ran out already), I needed a way to abstract some parts of the webpage that would be repeated throughout the site. The header and footer are the two main examples.

I used to have my header code local to each individual page of my website; however, this was rather unwieldy because to make any edits I would have to go to all webpages and change or do a multi-file find and replace. Now, instead of the html that forms the header part of the document on each page, I have



Which places the contents of the file at the URL “/header.html” into its place. Note that “$\_SERVER[‘DOCUMENT\_ROOT’]” is gets the URL of the server root from PHP, since PHP does not support using “/” to represent the server root.

By doing this, I only have to edit the content of the html file “header.html” in my servers root folder to propagate the same change to all headers throughout my website.

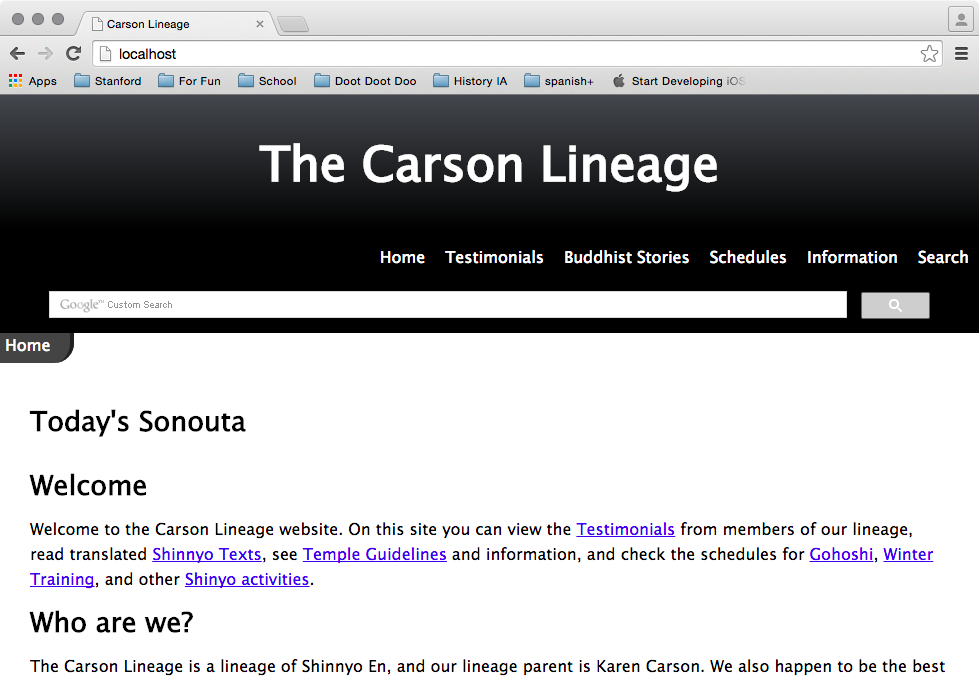
I do a similar thing with the footer on each page using the syntax



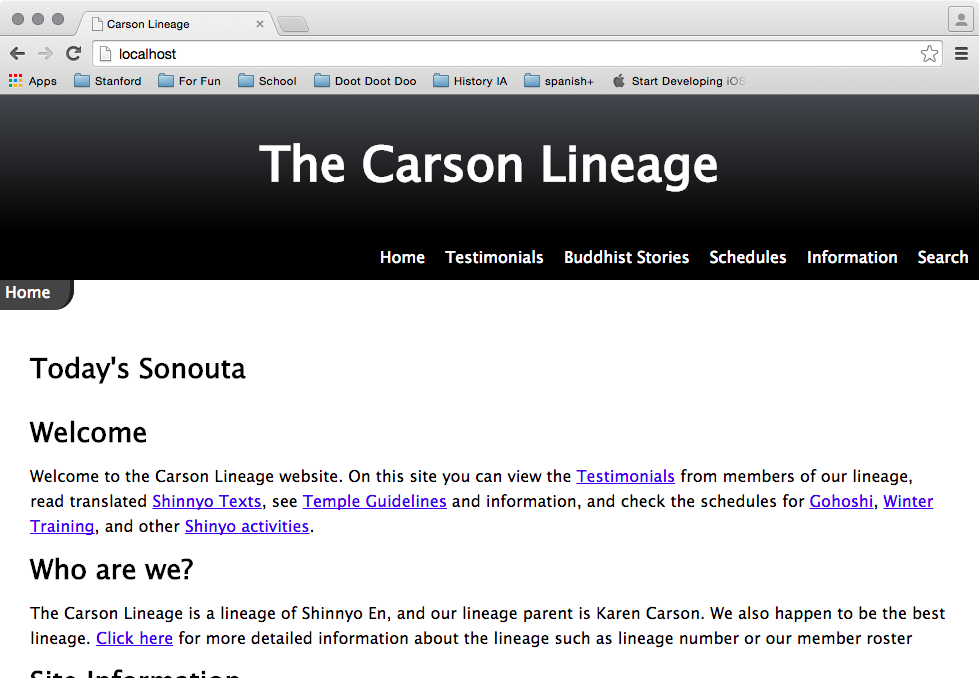
In the future, I may change use MySQL to have the server load the header and footer into memory (since the header and footer documents will be used so often); however, for now, this is a nifty way of propagating changes throughout the site, and will save me a lot of time in terms of updating pages.

**Use of JavaScript**

JavaScript is a programming language common among most major browsers; the language is interpreted and executed on the clients machine, allowing one to update or modify a page quickly on a user’s machine without using up server resources.

In this project I use JavaScript in order to implement the functionality of buttons on certain pages. For example, I would like my site to have a search bar, but it is rather inelegant to have it always be shown on the page like so: 

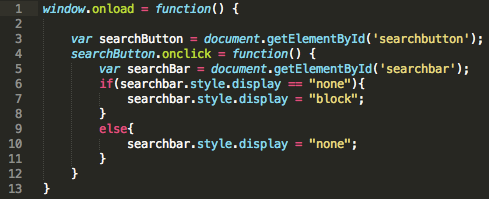
Especially on mobile where vertical horizontal space are already at a premium. Therefore, I have created a search button. By clicking the “Search” text (which is to be replaced by a magnifying glass icon later) you can toggle whether the search bar is shown, resulting in this:



This is done using a JavaScript script loaded within the header like so:



This links to a script which executes whenever the page load (we do this by setting the script to run on window.onload):



In this script the window.onload is set to a wrapper function which holds the code which gives the search button its functionality. On line 3 we find the searchButton which we are adding the functionality to. Note that the JQuery library is normally used for things like this; however, the script and page are so light that it just isn’t worth loading an extra script. On line 4 we again use a wrapper function in order to set the functionality of the searchButton. The rest of the lines use an if-statement in order to toggle the hidden state of the search bar.

I am going to implement something similar with a “menu” button to conditionally show the navigational components of the navigation bar.