**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

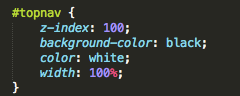
**Use of Cascading Style Sheets (CSS)**

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.

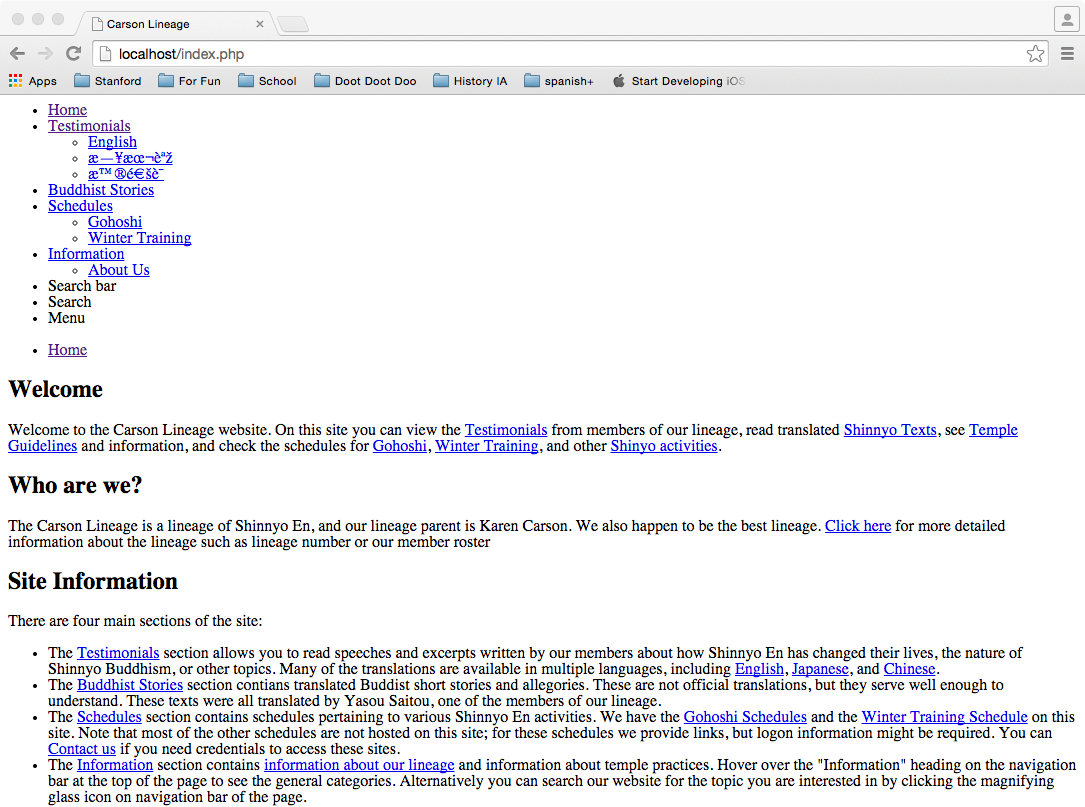
Note that there is a “/” at the beginning of the URLs within the href attribute. The “/” specifies that the path followed should start from the root of the file system (and are relative to the root of the file system, rather than the file this series of <link>s are declared in). Because of this, generally 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.

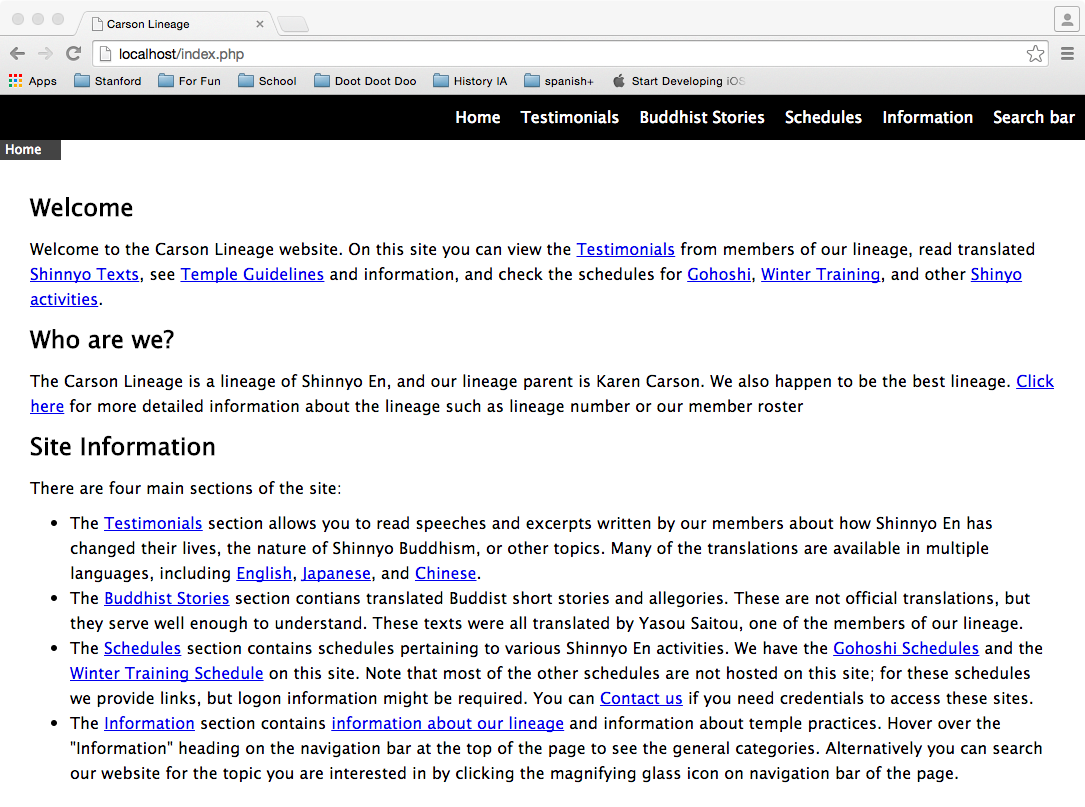
Specifically, what we are doing is setting the z-index (the precedence of ‘height’ on the screen) to 100, we set the background color to black, we set the text color to white, and we set the width of the navigation bar to be 100% of the width of the element that is around it (which is the <body> element).

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

Before CSS:

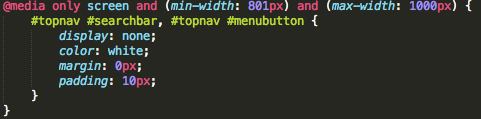


After CSS:

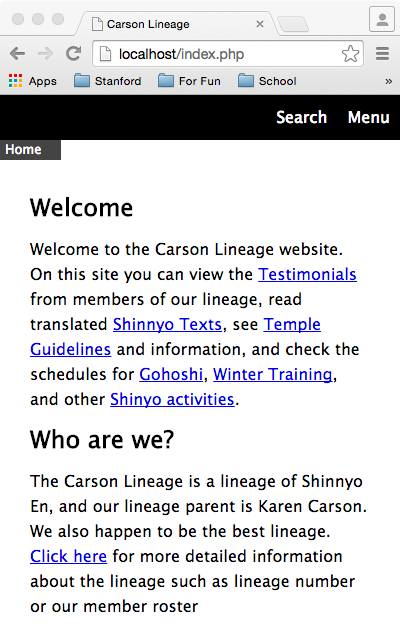


**Use of @media Queries**

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, or PHP: Hypertext Processor, or PHP: Hypertext Processor: Hypertext Processor, or PHP: Hypertext Processor: Hypertext Processor: 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.