**BUA 235 lab #2: add style to an HTML web page with CSS**

The first lab introduced the hypertext markup language, HTML, with which you can turn an ordinary text file into an interactive web page. We left it pretty plain-looking, however, using the browser’s default fonts and colors.

The code use to add style to a web page is called CSS, for Cascading Style Sheets. CSS is the subject of today’s lab.

**Getting Started**

I have created a sample web page today and posted it to GitHub for us to experiment with. If you prefer, you can experiment on your own web page from last week. The sample page is a list of some of my research publications, and is found at:

<https://github.com/joeclark-phd/jwcwww>

The page is unstyled HTML but uses some tags you know, particularly H1, H2, UL, LI, and A.

In order to apply a **style sheet** to this code, we need to create another plain-text file, this one with a name ending in “.css”. I’ll call mine “styles.css” and save it in the same folder as my HTML code file. Then you need to tell the HTML file where to go look for styles, by adding a new line to the “HEAD” section of the code. See this example:

<head>

<title>My Publications</title>

**<link href="styles.css" rel="stylesheet">**

</head>

**Setting Styles**

Styles are specified according to HTML tags, and apply to the content that appears within those tags. For example, you could apply a style to the “P” tag to change the way paragraphs are styled. The basic pattern of code in a CSS stylesheet is:

selector { attribute: style;

...

attribute: style; }

Where the “selector” is usually the name of an HTML tag. There are many attributes that can be styled, and you can learn or look up the options for each.

**1.** **Set a font size, color, and background color for the whole page.** Since the BODY tag surrounds all the content, you can apply a style to BODY that will affect the whole page. Add these lines to “styles.css”:

body { background: lightyellow;

color: darkgray;

font-size: 14pt; }

For a full list of available colors: <http://www.w3schools.com/colors/colors_names.asp>

**2.** **Apply a webfont.** A great selection are available from Google: <https://fonts.google.com/>. Google even gives you the code to add to your HTML and CSS file.

For example, if you like the Oswald font, put this in the HTML before the link to styles.css:

<link href="https://fonts.googleapis.com/css?family=Oswald" rel="stylesheet">

And add this to the CSS:

body { font-family: 'Oswald', sans-serif; }

**3.** **Style the H1 and H2 tags how you want them.** You can size the fonts in “pt” (points) or “px” (pixels), but here I’ll just size them as a relative percentage of the base font size. That way if I change the BODY font size, they’ll stay the same proportions.

h1 { font-size: 250%;

font-weight: bold;

color: navy }

h2 { font-size: 150%;

font-style: italic }

**4. I want the links at the top of the screen to appear side by side instead of on separate lines.** These are <LI> elements. You can set the display attribute to “inline” if you want them all to appear as words in a paragraph, or “block” if you want them on separate lines. Another option is “inline-block” which gives us some of the best of both. That’s my choice.

li { display: inline-block }

Unfortunately this has a side effect: it affects the other <LI> elements in other parts of the page! In this case, I need a more specific CSS “selector” than just “li”. What I’ll do is alter the HTML so that the navigation section is wrapped by <NAV> and </NAV> tags. Then, in CSS, I can specify I want the style to apply to an <LI> inside a <NAV>, like so:

nav li { display: inline-block }

**5. Create borders around elements.** Because the navigation links are “inline blocks” they still have some attributes of “block” elements, like the ability to set borders and margins. Give them borders:

nav li { border: 2px solid black }

You can also give partial borders to certain elements, like an H1:

h2 { border-left: 10px solid orange }

For more on what you can do with borders: <http://www.w3schools.com/css/css_border.asp>

**6. Now see what margins and padding can do.** Basically, every block element has **margins** (the space outside the border that separates an element from other elements) and **padding** (the space inside the border that separates the contents from the edges of the block). If you assign different values to these, you can see what they do to your navigation elements:

nav li { padding: 10px;

margin: 5px;

background: white }

**7. Finish the navigation by styling the links.** Links have different “states”: they can have a different style when you hover over them, when they have already been visited, and so on. In this case, I’ll set one standard color, remove the underlines, and set a different color for when the mouse is hovering:

a { color: darkred;

text-decoration: none }

a:hover { color: red }

**8. Style the publication titles with a CSS ‘class’.** The problem with this is that there’s no way to select the title, because it has no HTML tag around it. The solution is to add a tag. One tag that has no default effect (and is therefore safe) is SPAN. Alter the HTML for article titles like so:

**<span>**Bespoke Analytics at Lie-Nielsen Toolworks.**</span>**

You might use SPAN for other purposes, though, so we don’t want to style all the SPANs the same. Instead, let’s give this one a “class”, indicating what sort of SPAN it is. Change the <span> tag to:

<span **class="title"**>

Now, in the CSS, we’ll specify a style not for SPANs, but for all tags of class “title”, like so:

.title { font-weight: bold;

text-decoration: underline; }

**9. Add an additional class to highlight a special title.** I’m pretty proud of my best paper nomination, so for the first article listed, I’m going to create a class called “special”:

.special { color: darkgreen }

You can apply more than one “class” to an HTML element. They’ll be applied in the order in which they appear in the CSS file, so if two of them conflict, the later one will be applied. Just change the SPAN tag in question to:

<span class="title special">

*CSS is a relatively simple language, but there are dozens (if not hundreds) of attributes and styles you can use. And you can spend plenty of time doing trial and error to make the pages look just like you want them. Don’t be afraid to Google for online references, and be creative!*

My files can be referenced at: https://github.com/joeclark-phd/jwcwww/tree/completed