HTTP 5110: Producing a Web site using HTML and CSS.

Introduction

This is a simplified way to build a small static Web site. The code and methods are identical to more complex Web sites you will build in other classes. There are newer tags - <nav></nav> or <article></article> that you could use here instead if you wish. I would like you to complete this assignment and send the files to me by email by end of class today and no later than Friday September 13th 2024 at 5 pm.

In this handout we will take an existing visual design and implement it in CSS. Then using that CSS design as a template, we will build a small Web site. To finish the Web site all you would have to do is add content. Please do not just copy and paste the code – the more code you type out the easier coding gets!

What do you need?

You are creating two main plain text documents from which you will make several copies. You will need a plain text editor: Notepad for a PC and Adobe Brackets for a Mac. You will need a Web browser: Safari for a Mac and Chrome for a PC. Testing on a PC and a MAC is considered standard among Web developers. While there are different renderings in different browsers we will not worry about too much today.

You will also need your wireframe, or the visual design rendered in Photoshop or some other imaging or prototyping software. Print it out so that you can write on it. These will give you your filenames and the folders where you will be putting your files.

On the desktop create a folder called “website” (no quotes) and within that folder two more folders called: “images” and “styles” (again no quotes). Then open your text editor. How you name your files is very important. Remember that for file names: no spaces between words and all letters are in lowercase.

Let’s begin

1. We will divide our visual design, wireframe, into regions. Giving each region a name: see diagram 1 below.
2. The name of each region will become its style for the CSS. We will use “class” rather than “id” for the styles. But you can use either as the result is the same. (Actually, while they look the same the usage is different. There is an error in the code here – see if you can spot it.)

3. The next thing is to create an html file that will become the main template. The code for this exercise is after the image below.
4. There are a couple of styles that will be added and explained as we go along. For now: type what you see and do not copy and paste. But use your own name rather than mine!

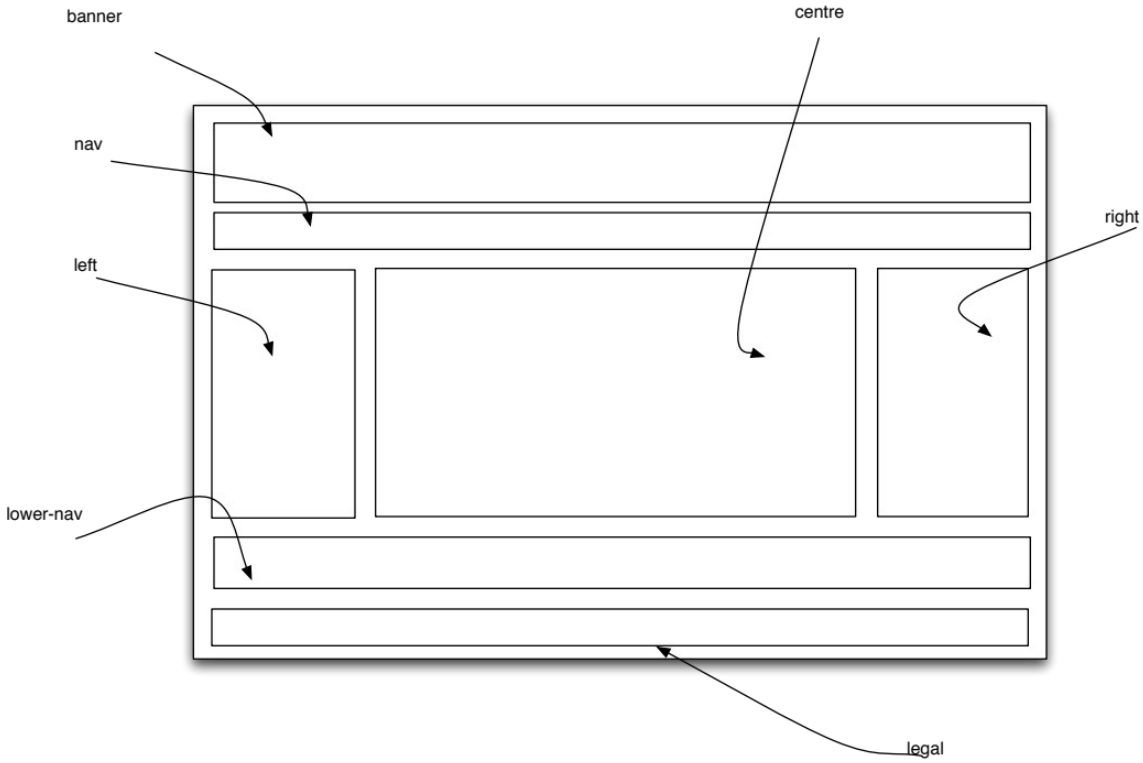

Diagram 1
Wireframe with CSS rules described.

5. Here is the code for the html template:

```
<!DOCTYPE HTML>
<html lang=”en”>
<head>
      <title>Bernie Monette’s Web site</title>
      <link href="styles/styles.css" rel="stylesheet" type="text/css" />

</head>
<body>

      <div class="box"><span class="logo">Bernie Monette</span></div>

      <div class="nav">
      <ul id="navlist">
                  <li><a href="index.html">Home</a></li>
                  <li><a href="portfolio.html">Portfolio</a></li>
                  <li><a href="about.html">About</a></li>
                  <li><a href="contact.html">Contact</a></li>
                  <li><a href="links.html">Links</a></li>
            </ul>
      </div>
```

```
      <div class="left">Left</div>
      <div class="centre">Centre</div>
      <div class="right">Right</div>

      <div class="lower_nav">
      <ul id="navlist">
                  <li><a href="index.html">Home</a></li>
                  <li><a href="portfolio.html">Portfolio</a></li>
                  <li><a href="about.html">About</a></li>
                  <li><a href="contact.html">Contact</a></li>
                  <li><a href="links.html">Links</a></li>
            </ul>
      </div>

<div class="legal"> <ul id="navlist"> <li><a
href="copyright.html">Copyright 2021 Bernie Monette all rights
reserved</a></li><li> <a href="privacy.html">Privacy</a></li> <li><a
href="terms.html">Terms of Use</a></li></ul></div>


</body>
</html>
```

6. In this example we are using “divs” ( ) as a means of creating boxes to put our information in. Each div has many attributes you can adjust but for today we are just using background colour (background-color), border colour (border-color), and border weight (border-weight). Note the use of American spelling in the code itself.

7. You will see that we have used an unordered list (
) for the navigation. We do this as it is easier to manipulate the position of the navigation elements. You may have noticed a new rule called “navlist” and it is an id rather than a class. The purpose for this rule is to use a list for our navigation and have it display along the horizontal rather than the vertical. If you look at this file in the browser, you will see this:

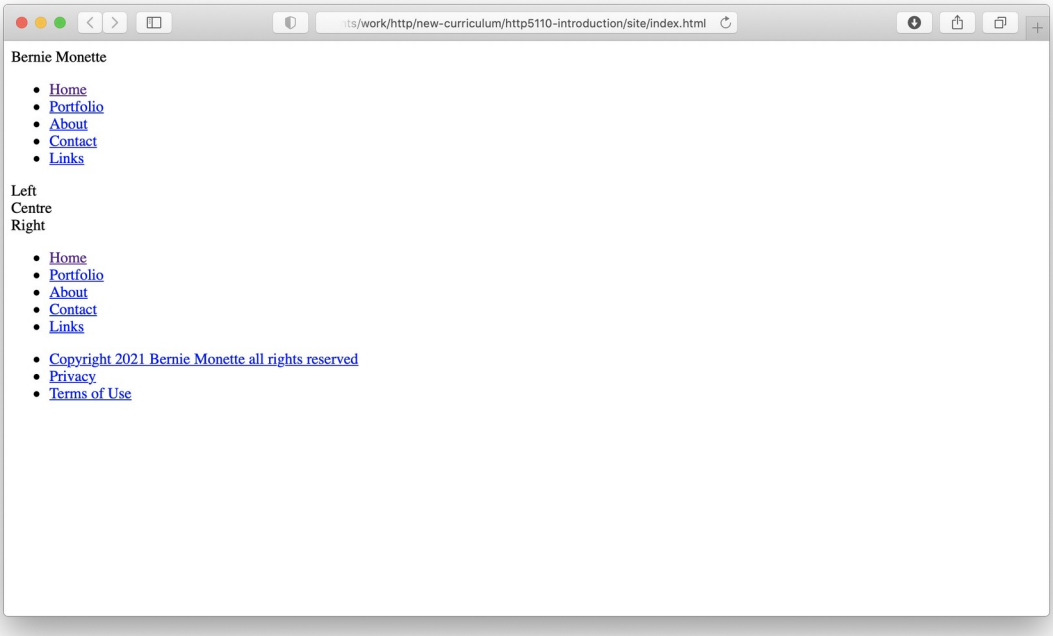

Diagram 2
This is the code without the CSS attached. Dull but functional.

8. Keep in mind that some people cannot see your Web site. Therefore, it is important that your design renders nicely without the CSS. Coding and the sequence of content are very important to insure this. As a rule we want our Web site to be used by more rather fewer people. That includes anyone with a disability. Also, machines, such as the Googlebot search engine, are also blind, so accessibility means your code has to be machine-readable.

9. You will notice that the file names and the links are already written. The proper sequence to writing the code is:

    a. Write the divs. Remember to add the CSS rules using class. E.g. .
    b. Put in the navigation terms
    c. Turn these terms into links: surrounding the link text with the anchor tags (text)
    d. Since the upper and lower navigation are the same: do the top first and then copy them down to the lower navigation area.
    e. In the footer div (or footer sections) enter the legal terms.
    f. Turn the legal terms into links as well.

10. You will now write the stylesheet. Feel free to copy and paste where appropriate here. The code is below:

```
body {
      padding: 0px;
      margin:  0px;
      font-family: Arial, Helvetica, sans-serif;
      background: #fff;
      color: #000;
}

a:link{
      color:#F00
}

a:visited{
      color:#FC0
}

a:hover{
      background-color:#3CF;
      color:#03F
}

a: active{
      color:#F3F
}

.logo{
      font-size:36px
}

#navlist li
      {
      display: inline;
      list-style-type:circle;
      padding-right: 20px;
      border-left: 1px solid #000;
      line-height: 1.1em;
      margin: 0 .5em 0 -.5em;
      padding: 0 .5em 0 .5em;
      }


.box {
      margin:10px;
      padding:10px;
      background-color:#666;
      border-width:1px;
      border-color:black;
      border-style:solid;
      clear:both;

      }
.nav {
      margin:10px;
      padding:10px;
      background-color:#999;
```

```
	border-width:1px;
	border-color:black;
	border-style:solid;

	}
.left {
	width:20%;
	margin:10px;
	padding:10px;
	background-color:#CCC;
	border-width:1px;
	border-color:black;
	border-style:solid;
	float:left;


	}
.centre {
	width:47%;
	margin:10px;
	padding:10px;
	background-color:#CCC;
	border-width:1px;
	border-color:black;
	border-style:solid;
	float:left;


	}
.right {
	width:20%;
	margin:10px;
	padding:10px;
	background-color:#CCC;
	border-width:1px;
	border-color:black;
	border-style:solid;
	float:right;



	}
.lower_nav {
	margin:10px;
	padding:10px;
	background-color:#999;
	border-width:1px;
	border-color:black;
	border-style:solid;
	clear:both;

	}
.legal {
	margin:10px;
	padding:10px;
	background-color:#999;
	border-width:1px;
```

```
border-color:black;
border-style:solid;
float:none;
clear:both;

}
```

11. Notice however in the html code the link to the stylesheet already exists:

    <link href="styles/styles.css" rel="stylesheet" type="text/css" />

12. At this point make sure you save your file. Call it template.html and save it in the root folder. Save the CSS file: call it styles.css and save it to the styles folder.

13. The site now looks like this:

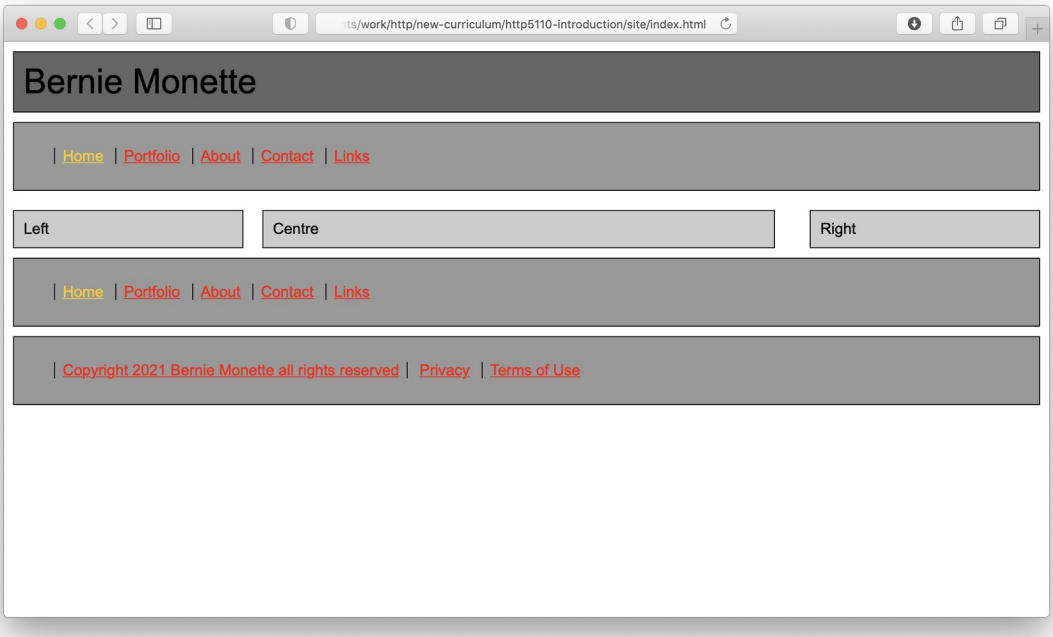

Diagram 3 -
Now the CSS is attached: still dull but more of a design!

14. You will notice that we have put our name in the place of the logo. If you have a logo you want to add: now is the time to add it.

15. The code that is inserted will look like this images is your images folder and logo.gif is the reference to

the logo file. The alt tag is the text used to describe the image for text to voice readers. This is a key accessibility feature.

16. You will now add alternative text and a title to the image. Your code should look like this: . Save your file.

17. The "title" is a very powerful tag that adds a tooltip to your page. It gives you another opportunity to explain what the link is and where the user will go when they click it.

18. We now have a finished the template. However, before we use it: it is important to check it for errors. Look at it in the browser: does it work the way you wish? Look at the links: are the file names and paths to the file correct? If everything is good, we can now create the Web site.

19. With the template file present and foremost, in your editor, go to File -> Save As and click. You will be given a file browser menu: make sure you are saving the files to the correct, in this case the root or website, folder. Give the file a new filename. But make sure it is one of the names you have set out in your site architecture. Start with the home page: index.html. Then do the save as again (as long as there is no content you can just use the foremost file) and save it as portfolio.html and do so for all the links including the legal links at the bottom of the page. We should have 8 pages in total.

20. The 8 files are: index.html, portfolio.html, contact.html, about.html, links.html, copyright.html, terms.html, privacy.html. Template.html should be kept as well.

21. Open the last file in the browser and test it: all the links should work. If they do not then there is an error in the template. Fix the error and resave the files.

22. Once you see your files in the browser and you navigate from page to page you will notice a problem: how do you know what page you are on? In order to add a wayfinding, clue we need to open all 8 of our files.

23. Open all of your files in your editor, but not the template (in fact make sure your template is closed). In each file you want to remove the link that refers to that file and just leave the link name. You will do this for each page. For example:

    Home would become just Home. See below:

```
<ul id="navlist">
```

```
            <li>Home</li>
            <li><a href="portfolio.html">Portfolio</a></li>
            <li><a href="about.html">About</a></li>
            <li><a href="contact.html">Contact</a></li>
            <li><a href="links.html">Links</a></li>
        </ul>
```

24. You will need a new style for the wayfinding information. Open your CSS file and add after the link rules (you can put it after the link rules):

    ```
    .deadnav{
        color:#FFF;
        background-color: #3CF
    }
    ```
    Save your CSS file.

25. Back to your file. Highlight the navigation term you just removed the link from. You will want to surround that term with the code

    ```
    <span class=”deadnav”>link name</span>
    ```

26. Do the same for each file. Remember: save and CLOSE the file once you have finished working on it!

27. You need to do this twice for each regular page (upper and lower links) but only once for the legal pages.

28. Test your result in the browser and you should be able to see how the “deadnav” works.

29. Now you just have to add your content! You may find that you need to add styles to accommodate your content: go ahead! At this point you have enough information to make adjustments and experiment with CSS to see what it and you can do.

30. I would make a copy of this code (since it works) and keep it in a safe place. Then play with the copy. A series of “what happens when I do this…” can help you learn a lot about HTML and CSS coding.

31. You have made a Web site and all the links work: congratulations!