Web Programming Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Activity 1.2 – Café Menu

Points: \_\_\_\_\_\_\_\_\_\_ / 37

CSS tells the browser how to display your webpage. You can use CSS to set the color, font, size, and other aspects of HTML elements. In this activity, you'll learn CSS by designing a menu page for a cafe webpage.

1. Using [MDN](https://developer.mozilla.org/en-US/docs/Web/HTML), look up the following HTML elements and their attributes to see what they do and how they work. Record your research in your engineering notebook.

a. <meta>

i. charset

ii. name

iii. content

b. <link>

i. rel

ii. href

c. <div>

i. class

ii. id

d. <section>

e. <article>

f. <hr>

g. <body>

h. <main>

i. <footer>

2. Using [MDN](https://developer.mozilla.org/en-US/docs/Web/CSS), look up the following CSS properties to see what they change. Record your research in your engineering notebook.

a. background-color, background-image

b. color

c. font-size

d. font-family

e. text-align

f. margin, margin-left, margin-right, margin-top, margin-bottom

g. padding

h. width

i. max-width

3. Download, extract, and rename the linked project from GitHub using the link provided in Google Classroom. When you rename the project folder, make sure you follow proper naming practices.

4. Next, open the file named *index.html*. Add a comment at the top of the document with the project name, your name, today’s date, and the title of course on separate lines. As you learned in the last few steps of the Cat Photo App, there is a basic structure needed to start building your webpage. Add the <!DOCTYPE html> tag at the top of your file. Underneath, add an <html> element with the attribute lang="en".

5. Inside the <html> element, add a <head> element. Inside the <head> element, add a <title> element with the content Café Menu (hint: put the content between the opening and closing tags). The <title> is one of several elements that provide extra information not visible on the webpage itself, but it is useful for search engines or how the page gets displayed.

Inside the <head> element, nest a <meta> element with an attribute charset="uff-8". This will tell the browser how to encode characters for the page. Note that <meta> elements are self-closing.

6. To prepare to create some actual content, add a <body> element below the closing tag of the <head> element. Inside the <body> element, add a <main> element. This <main> element will eventually contain pricing information about coffee and desserts offered by the café.

7. The name of the cafe is CAMPER CAFÉ. Add an <h1> element within your <main> element. Make the content of the tag the name of the café in capitalized letters (CAMPER CAFÉ) to make it stand out. To let the visitors know the café was founded in 2020, add a <p> element below the <h1> element with the text Est. 2020.

8. There will be two sections on the menu, one for coffees and one for desserts. Add a <section> element below the <p> element from the previous step so you have a place to put all the coffees available. Create an <h2> element in the <section> element and give it the text Coffee.

9. Up until now, you have been limited regarding the presentation and appearance of the content you create. To start taking control, add a <style> element within the <head> element under the <title> element.

You can add style to an element by specifying it in the <style> element and setting a property for it like this:

element {

property: value;

}

Inside the <style> element, center your <h1> element by setting its text-align property to the value center like this:

h1 {

text-align: center;

}

10. In the previous step, you used a *type selector* to style the <h1> element. Using the same code, center the <h2> and <p> elements by adding a new type selector for each one to the existing <style> element. These should be underneath the type selector for <h1>.

11. You now have three type selectors with the exact same styling. You can add the same group of styles to many elements by creating a list of selectors. Each selector is separated with commas like this:

selector1, selector2 {

property: value;

}

Delete the three existing type selectors and replace them with one selector list that centers the text for the <h1>, <h2>, and <p> elements. It should look something like this:

h1, h2, p {  
 text-align: center;

}

12. You have styled three elements by writing CSS inside the <style> tags. This works, but since there will be many more styles, it's best to put all the styles in a separate file and link to it.

I have created a separate *styles.css* file for you inside the folder called *styles*. You can open this document by double clicking it. You may also want to adjust your windows so that you can see *index.html*, *styles.css*, and the preview all at once.

Start by copying and pasting the styles you have created into the *styles.css* file. Make sure to exclude the opening and closing <style> tags (i.e., only copy and paste the stuff from inside the tags). Once you have the CSS in the *styles.css* file, go ahead and remove the <style> element and all its content. Once it is removed, the text that was centered will shift back to the left as the styles from the separate file have not been applied yet and you deleted the <style> element within the HTML document.

You will need to link the *styles.css* file so the styles will be applied again. Inside the <head> element under the <title> element, add a <link> element with the attributes rel="stylesheet" and href="styles/styles.css". It should look something like this:

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

14. The text should be centered again, which means the link to the CSS file is working. Add another style to the file that changes the background-color property to brown for the <body> element. That brown background makes it hard to read the text. Change the <body> element's background color to be burlywood so it has some color, but you are still able to read the text.

15. The <div> element is used mainly for design layout purposes unlike the other content elements you have used so far. Add a <div> element inside the <body> element and then move all the other elements inside the new <div>. Inside the opening <div> tag, add the attribute id="menu".

16. The goal now is to make the <div> not take up the entire width of the page. The CSS width property is perfect for this. You can use the *id selector* to target a specific element with an id attribute. An id selector is defined by placing the hash symbol # directly in front of the element's id value. For example, if an element has the id of cat then you would target that element like this:

#cat {

width: 250px;

}

Use the #menu selector to give your element a width of 300px. It should look something like this:

#menu {

width: 300px;

}

17. Comments in CSS look like this:

/\* comment here \*/

In your style sheet, comment out the line containing the background-color property and value, so you can see the effect of only styling the #menu element. This will make the background white again.

Now use the existing #menu selector to set the background color of the <div> element to be burlywood.

In addition, add a comment at the top of the document with the project name, your name, today’s date, and the title of course on separate lines.

18. Now it's easy to see that the text is centered inside the #menu element. Currently, the width of the #menu element is specified in pixels (px). Change the width property's value to be 80%, to make it 80% the width of its parent element (<body>).

19. Next, you want to center the #menu horizontally. You can do this by setting its   
margin-left and margin-right properties to auto. Think of the margin as invisible space around an element. Using these two margin properties, center the #menu element within the <body> element.

20. So far you have been using type and id selectors to style elements. However, it is more common to use a different selector to style your elements. A *class selector* is defined by a name with a dot directly in front of it, like this:

.class-name {

styles

}

Change the existing #menu selector into a class selector by replacing #menu with a class named .menu. To apply the class's styling to the <div> element, remove the id attribute and add a class attribute to the <div> element's opening tag. Make sure to set the class value to menu. It should look like this:

<div class="menu">

21. Since the cafe's main product for sale is coffee, you could use an image of coffee beans for the background of the page. Delete the comment and its contents inside the <body> type selector (i.e., the one that changes the background-color to burlywood). Now add a background-image property and set its value to url("../images/beans.jpg"). The two dots tells the browser to look up a directory to find the images directory.

22. It’s looking good. Time to start adding some menu items. Add an empty <article> element under the Coffee heading. It will contain the flavor and price of each coffee you currently offer. <article> elements commonly contain multiple elements that have related information. In this case, it will contain a coffee flavor and a price for that flavor. Add two <p> elements inside your <article> element. The first one's text should be French Vanilla, and the second's text 3.00.

23. Starting below the existing coffee/price pair, add the following coffee and prices using <article> elements with two nested <p> elements inside each. As before, the first <p> element's text should contain the coffee flavor and the second <p> element's text should contain the price.

Caramel Macchiato 3.75

Pumpkin Spice 3.50

Hazelnut 4.00

Mocha 4.50

24. The flavors and prices are currently stacked on top of each other and centered with their respective <p> elements. It would be nice if the flavor was on the left and the price was on the right. Add the class name flavor to the French Vanilla p element. It should look something like this:

<p class="flavor">French Vanilla</p>

Switch back to your style sheet. Using your new flavor class as a selector, set the text-align property's value to left. It should look something like this:

.flavor {

text-align: left;

}

25. Next, you want to align the price to the right. Add a class named price to your <p> element that has 3.00 as its text. Switch back to your style sheet. Align the text to the right for the elements with the price class. It should look similar to the .flavor class selector you wrote in the previous step.

26. That is kind of what you want, but now it would be nice if the flavor and price were on the same line. <p> elements are *block-level elements*, so they take up the entire width of their parent element.

To get them on the same line, you need to apply some styling to the <p> elements so they behave more like *inline elements*, which will appear next to each other. To do that, start by adding a class attribute with the value item to the first <article> element under the Coffee heading.

27. The <p> elements are nested in an <article> element with the class attribute of item. You can style all the <p> elements nested anywhere in elements with a class named <item> like this:

.item p { }

Switch back to your style sheet. Using the above selector, add a display property with value   
inline-block so the <p> elements behave more like inline elements. It should look something like this:

.item p {

display: inline-block;

}

28. That's closer, but the price didn't stay over on the right. This is because inline-block elements only take up the width of their content. To spread them out, add a width property to the flavor and price class selectors that have a value of 50% each (this should be done in your style sheet).

Well, that **did** **not** **work**. Styling the <p> elements as inline-block and placing them on separate lines in the code creates an extra space to the right of the first <p> element, causing the second one to shift to the next line. One way to fix this is to make each <p> element's width a little less than 50%. Change the width value to 49% for each class to see what happens.

That worked, but there is still a little space on the right of the price. You could keep trying various percentages for the widths. Instead, use the back space key on your keyboard to move the <p> element with the class price next to the <p> element with the class flavor so that they are on the same line in the editor. Make sure there is no space between them. It should look something like this:

<p class="flavor">French Vanilla</p><p class="price">3.00</p>

Now go ahead and change both the flavor and price class' widths to be 50% again.

29. Now that you know it works, you can change the remaining <article> and <p> elements to match the first set. Start by adding the class item to the other <article> elements. Next, position the other <p> elements to be on the same line with no space between them. To complete the styling, add the applicable class names flavor and price to all the remaining <p> elements.

30. If you make the width of the page preview smaller, you will notice at some point, some of the text on the left starts wrapping around to the next line. This is because the width of the <p> elements on the left side can only take up 50% of the space. Since you know the prices on the right have significantly fewer characters, change the flavor class width value to be 75% and the price class width value to be 25%. This should be done in your style sheet.

31. You will come back to styling the menu in a few steps, but for now, go ahead and add a second <section> element below the first for displaying the desserts offered by the cafe. Add an <h2> element in the new section and give it the text Desserts. Add an empty <article> element under the Desserts heading. Give it a class attribute with the value item. Nest two <p> elements inside your <article> element. The first one's text should be Donut, and the second's text 1.50. Put both on the same line making sure there is no space between them. For the two <p> elements you just added, add dessert as the value of the first <p> element's class attribute and the value price as the second <p> element's class attribute.

32. Something does not look right. You added the correct class attribute value to the p element with Donut as its text, but you have not defined a selector for it.

The CSS rule for the flavor class already sets the properties you want. Add the dessert class as an additional selector for this CSS rule.

33. Below the dessert you just added, add the rest of the desserts and prices using three more <article> elements, each with two nested <p> elements. Each element should have the correct dessert and price text, and all of them should have the correct classes.

Cherry Pie 2.75

Cheesecake 3.00

Cinnamon Roll 2.50

34. You can give your menu some space between the content and the sides with various padding properties. In your style sheet, give the menu class a padding-left and a padding-right with the same value 20px.

That looks better. Now try to add the same 20px padding to the top and bottom of the menu. Since all 4 sides of the menu have the same internal spacing, go ahead and delete the four properties and use a single padding property with the value 20px.

35. The current width of the menu will always take up 80% of the <body> element's width. On a very wide screen, the coffee and dessert appear far apart from their prices. Add a max-width property to the menu class with a value of 500px to prevent it from growing too wide.

36. You can change the font-family of text, to make it look different from the default font of your browser. Each browser has some common fonts available to it. In your style sheet, change all the text in your <body>, by adding a font-family property with the value sans-serif. This is a fairly common font that is very readable.

37. It is a bit boring for all the text to have the same font-family. You can still have most of the text sans-serif and make just the <h1> and <h2> elements different using a different selector. Style both the <h1> and the <h2> elements using a single selector so that these elements' text use Impact font.

You can add a fallback value for the font-family by adding another font name separated by a comma. Fallbacks are used in instances where the initial is not found/available. Add the fallback font serif after the Impact font. It should look something like this:

h1, h2 {

font-family: Impact, serif;

}

38. Make the Est. 2020 text italicized by creating an established class selector and giving it the  
font-style property with the value italic inside your style sheet. Now apply the established class to the Est. 2020 text in your HTML document.

39. The typography of heading elements (e.g. <h1>, <h2>) is set by default values of users' browsers. Add two new type selectors (<h1> and <h2>). Use the font-size property for both, but use the value 40px for the <h1> and 30px for the <h2>.

40. Back in the HTML document, add a <footer> element below the <main> element, where you can add some additional information. Inside the <footer>, add a <p> element. Then, nest an <a> element in the <p> that has a src of https://www.freecodecamp.org and has the alt text Visit our website. Add a second <p> element below the one with the link and give it the text 123 Free Code Camp Drive.

41. You can use an <hr> element to display a divider between sections of different content. First, add an <hr> element between the <p> element with the class established and the first <section> element. Note that <hr> elements are self closing.

The default properties of an <hr> element will make it appear as a thin light grey line. You can change the height of the line by specifying a value for the height property. In your style sheet, add a type selector for <hr> and change the height to be 3px. Change the background color of the <hr> element to brown so it matches the color of the coffee beans.

Notice the grey color along the edges of the line. Those edges are known as *borders*. Each side of an element can have a different color or they can all be the same. Make all the edges of the <hr> element the same color as the background of it using the border-color property.

Notice how the thickness of the line looks bigger? The default value of a property named  
border-width is 1px for all edges of <hr> elements. By changing the border to the same color as the background, the total height of the line is 5px (3px plus the top and bottom border width of 1px). Change the height property of the <hr> to be 2px, so the total height of it becomes 4px.

42. Go ahead and add another <hr> element between the <main> element and the <footer> element.

43. To create a little more room around the menu, add 20px of space on the inside of the <body> element by using the padding property (HINT: use the type selector you have already created in your style sheet).

44. Focusing on the menu items and prices, there is a large gap between each line. Use the existing selector that targets all the <p> elements nested in elements with the class named item (HINT: .item p) and set their top and bottom margin to be 5px (HINT: margin-top and margin-bottom). Using the same style selector, make the font size of the items and prices larger by using a value of 18px.

45. Changing the margin-bottom to 5px looks great. However, now the space between the Cinnamon Roll menu item and the second <hr> element does not match the space between the top <hr> element and the Coffee heading. Inside your style sheet, add some more space by creating a class named bottom-line using 25px for the margin-top property. Back in your HTML document, add the bottom-line class to the second <hr> element so the styling is applied.

46. Next you are going to be styling the <footer> element. To keep the CSS organized, add a comment at the end of *styles.css* with the text FOOTER. Moving down to the <footer> element, make all the text have a value of 14px for the font size. This should look something like this:

footer {

font-size: 14px;

}

47. The default color of a link that has not yet been clicked on is typically blue. The default color of a link that has already been visited from a page is typically purple. To make the <footer> links the same color regardless if a link has been visited, use a type selector for the anchor element (<a>) and use the value black for the color property.

You change properties of a link when the link has actually been visited by using a pseudo-selector that looks like this:

a:visited {

propertyName: propertyValue;

}

Change the color of the footer Visit our website link to be grey when a user has visited the link.

You change properties of a link when the mouse hovers over them by using a pseudo-selector that looks like this:

a:hover {

propertyName: propertyValue;

}

Change the color of the footer Visit our website link to be brown when a user hovers over it.

You change properties of a link when the link is actually being clicked by using a pseudo-selector that looks like this:

a:active {

propertyName: propertyValue;

}

Change the color of the footer Visit our website link to be white when clicked on.

To keep with the same color theme you have already been using (black and brown), change the color for when the link is visited to black and use brown for when the link is actually clicked.

48. The menu text CAMPER CAFE has a different space from the top than the address's space at the bottom of the menu. This is due to the browser having some default top margin for the <h1> element. Change the top margin of the <h1> element to 0 to remove all the top margin (HINT: this needs to be done in your stylesheet using margin-top).

49. To remove some of the vertical space between the <h1> element and the text Est. 2020, change the bottom margin of the <h1> to 15px.

50. Now the top spacing looks good. The space below the address at the bottom of the menu is a little bigger than the space at the top of the menu and the <h1> element. To decrease the default margin space below the address <p> element, create a class selector named address and use the value 5px for the margin-bottom property. Put this after your FOOTER comment in the stylesheet.

Now apply the address class to the <p> element containing the street address 123 Free Code Camp Drive by adding class="address".

51. The menu looks good, but other than the coffee beans background image, it is mainly just text. Under the Coffee heading, add an image using the path images/coffee.jpg for the src. Give the image an alt value of coffee icon. The image you added is not centered horizontally like the Coffee heading above it. img elements are "like" inline elements.

To make the image behave like heading elements (which are block-level), create an img type selector and use the value block for the display property and use the applicable margin-left and margin-right values to center it horizontally. It should look something like this:

img {

display: block;

margin-left: auto;

margin-right: auto;

}

52. Add one last image under the Desserts heading using the path images/pie.jpg for the src. Give the image an alt value of pie icon.

53. It would be nice if the vertical space between the <h2> elements and their associated icons was smaller. The <h2> elements have default top and bottom margin space, so you could change the bottom margin of the <h2> elements to say 0 or another number.

There is an easier way, simply add a negative top margin to the img elements to pull them up from their current positions. Negative values are created using a - in front of the value. Use a negative top margin of 25px in the img type selector.

54. Lastly, for the styling of the page to look similar on mobile as it does on a desktop or laptop, you need to add a <meta> element with a special content attribute. Add the following within the <head> element:

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

This is an odd element and is easier to explain by viewing your website on a mobile device. This can be done using the built-in developer tools in Chrome. Have Mr. Pawelski show you how to use these tools to view the website on a mobile device.

At this point, your website is complete. To wrap up this assignment, please do the following…

1. Compress the project folder and submit it to Google Classroom.

2. Print a copy of *index.html* and *styles.css*. Add these to your engineering notebook. Highlight and annotate your code.

3. Take a screenshot of the finished webpage (open it in a browser and use the Snip and Sketch tool). Print a copy of the screenshot and add it to your engineering notebook. Annotate the picture by labeling each component with the element required to insert it.

4. Finally, record the file structure for this project in your engineering notebook.

|  |  |
| --- | --- |
| **Project Criteria** | |
| Criteria | Point(s) |
| <head> element contains all the necessary non-visible and visible elements to provide the necessary meta data for the website. | /5 |
| Website contains a title and subtitle with the correct content and style. | /2 |
| Website contains the coffee menu with the correct content and style. | /5 |
| Website contains the desserts menu with the correct content and style. | /5 |
| Website contains a <footer> with an anchor and address with the correct content and style. | /2 |
| Website has the correct overall style and structure. | /10 |
| **Development Mechanics** | |
| Criteria | Point(s) |
| All elements are structured using best practices. | /2 |
| Project folder/director structure follows industry best practices. | /2 |
| All folders/directories and files use proper naming style (i.e., file-name). | /2 |
| All documents use whitespace and comments to help organize the code. | /2 |
| Project root folder/directory has an appropriate name. | /1 |
| All documents have a comment at the top that includes the program’s title, the student’s name, the date, and the course’s title. | /1 |

/ 37