Assignment 3: Intermediate CSS

In this assignment, we'll explore box model properties in CSS for spacing and alignment, centering elements, wrapping text around elements, and creating layouts that respond to the browser window size using display properties. To do this, we'll create a single-page website with the theme of scenic area.

The starter files contain screenshots of the final project.
You can also check the behaviour of the page here:
https://www.loom.com/share/53b44136bf6943e9bd5a330b84c6b096?sid=a3ea7903-ef77-4229-a2e3-1f3e0238b752

Task 1: Setup

Create a new folder with the name **a3-your-firstname-lastname**. Create files called **index.html** and **style.css** at the following paths:

content/pages/index.html
css/style.css

link these files together so that the CSS in **style.css** is applied to the HTML in **index.html**. The **index.html** should have the basic starter template for an HTML page.

As you work through the assignment, use the browser's **inspect tool** to see what's happening behind the scenes. The inspector tool can help you see why things are laid out in a certain way with respect to box model properties, like margin and padding, and display properties like block, inline-block and inline.

Task 2: HTML

Create the HTML markup code for the page as shown in the screenshots, but with your own choice of the scenic area. Use text content from Wikipedia or some other free source, with attribution. Feel free to take images from any source that allows free use with attribution. (https://www.pexels.com/ and https://unsplash.com/ are often good sources.) Each section should have one image. If the images you're using look quite large, add the following lines of code to your **style.css** file to restrict their size to 250 pixels in width:

```
img {
  width: 250px;
  height: auto;
```

}

Some things to keep in mind as you write the HTML for this page:

- Remember to use HTML semantic tags namely **header**, **nav**, **main**, and **footer** tags for the appropriate parts of the page. The three scenic area should be included within the **<main>** element, and the list of references should be in the **<footer>.**
- As you can see, the main content of the page consists of three sections: one for each scenic location. Mark up these sections using **<section>** tags. The purpose of this tag is to mark up any part of the page that is important enough to have its own heading. (In fact, one is required by W3C standards to have a heading inside a section tag to explain its purpose.)
- Add **id** attributes to each subheading on the page. Use these ids to set up hyperlinks in the navigation menu that cause the browser to scroll to the correct heading when clicked. Here's how you can do that: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a#Linking_to_an_element_on_the_same_page
- Once you've created all the necessary HTML, there is one other thing I'd like you to add: a **<div>** element that contains everything within the body of the document. Add a **class** attribute with the value **"wrapper"** to this <div> element.

  Div elements are used when there is no more specific or appropriate tag for a particular container. In this case, we will use the div element for styling purposes to set the width of the content in the page and centering it. Since a page could have more than one div element, we will often give each div a **class** that we can use to refer to it in our CSS.

  Your code should look something like this:

```
...
<body>
    <div class="wrapper">
        <!-- everything else on the page -->
    </div>
</body>
```

Task 3: Basic CSS

Before we get into the layout, let's implement some basic styles on the page:

- Use the Arial font stack (**Arial, Helvetica, sans-serif**) for the whole page and set the **line height** to something that's not too crowded;
- Set the size of the headings to be a bit larger than the default; use **rem** units instead of pixels. The **rem** unit is equivalent to the width of the letter **m** for text that has been

styled in the **root** of the document. (the **html** element, more specifically.) *Why would we use the text size as a point of reference?*

- Set the background color of the body to **lavender**. Set the color of the hyperlinks to **dodgerblue**. Set the background color of the **main, nav**, and **footer** to **white**.
    - Set a drop shadow on the **nav, main** and **footer** elements by using the **box-shadow** property: https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow
- Remove the underlines on the hyperlinks.
- Set the color for the headings using the CSS **rgb()** function. Choose a color that roughly matches the screenshots, but don't worry about getting it exactly right. The rgb() function is explained here: https://www.w3schools.com/cssref/func_rgb.asp
- Add a thin, solid border to each section with the color **lightcoral**.
- Figure out how to add a border to the **bottom** of your h2 headings; the border should be **dotted** and have the color **lightcoral.**
- The list items within the navigation menu should have the background color **midnightblue** and the hyperlink text within the nav menu should be **white**. Implement this without affecting the list of references in the footer and *without adding any additional classes to the HTML code*.

As you are writing your CSS code, try to keep things clean and organized. Try to avoid having more than one declaration block for a single selector whenever it's possible; try to keep all the styles for each part of the page in one place. Use comments wisely to organize your CSS code. Remember that styles nearer the bottom of the file have higher priority and it may overwrite styles you have defined above. Remember also that certain types of selectors have higher priority than others. For example, the class selector has higher priority than the element selector.

Task 4: CSS Box Model

You may have noticed that now things are a bit crowded on the page. For instance, there is no space between the text and the edge of its container, and containers have no space between them. We can fix this by adding **margin** and **padding**:

- **margin:** the space between boxes;
- **padding:** the space between the inside edge of the box and its content.

  We call these properties **box model** properties. *Where does **border** fit into the box model?*

Use margin and padding for the following:

- For the **main, nav**, and **footer**, add a gap of **1rem** between the content and the container.
- Add a space of **1rem** between the content of each **section** and the edges of the section.

- Find a way to add a vertical space of **1rem** between the **nav, main**, and **footer**. Also add a vertical space of **1rem** between each **section**. Remember that you can use properties such as **margin-bottom** and **margin-top**.

Task 5: Centering Things

As you may recall, centering things is done differently in CSS depending on the display property of the thing you're trying to center. Remember that any element with the display property **inline** or **inline-block** will flow similarly to text.

- Center the text of the headings using the **text-align** property.
- Use the **<div>** that we created earlier to center the content of the page and set its width. Select this div in CSS using its class, and do the following:
    - Use the **max-width** property to prevent the content from getting wider than **900px** while allowing it to shrink when the browser window is resized.
    - Use the **margin** property to center the div within the browser window when it is wider than 900px. *Which value of the margin property allows us to make sure the margin on the left and right is always the same, thereby centering the element?*

Task 6: Floating Images

Next, let's make text wrap around images by using the **float** property.
Unzip task-6.zip and open the folder in VS Code, then follow along the video at https://www.loom.com/share/8f0f48feb5d5492d94bdd6149fd2837d to learn how to use this property.

- The **first** and **third** images should be floated to the **left**, and the **second** image should be floated to the **right.** Add **classes** to your HTML code that you can select the images and float them in a particular direction. (For example, you should be able to use a single CSS selector to select both the first and third image.)
    - Images that are floated to the left should have a bit of space on the right to separate them from the text. Images that are floated to the right should have a bit of space on the left.
- You may see that your images are protruding from the bottom of their containers. This is a common problem when floating elements; their containers no longer take the height of the floated element into consideration when setting their own height. We'll learn how to fix this problem eventually, but for now, use the lazy solution of just adding more text to the section to expand it and prevent the image from leaking out the bottom!
- Note float property is a legacy CSS. You should still know it as you will often need to maintain or work on existing project written with float.

Task 7: Horizontal Menu

- To get the navigation menu items to align horizontally, change their **display** property to a value that allows the following:
    - the elements to flow horizontally in the same way as text, and;
    - the elements to be resized, using width, and have box model properties applied.
- Set the **width** of each menu item to **120px**. Center the text within each box.
- When the browser window is made to be very narrow, the menu items should wrap around to a new line. When this happens, there should be a little bit of space separating the items in the bottom row from the items in the top row.
- Figure out how to remove the bullet points from the list;
- Figure out how to get the menu items to align to the right of the page. (hint: the float property is *not* the best way to do this. If you want to see why, try it!)
- Make sure there is a space of **1rem** between the edges of each menu item box and the text within.

Check Your Work:

- Validate both the HTML and CSS of the site. (remember that these two types of code use different validators!) Take screenshots of your validation results so that I and the marker can easily see that your code is sound.

Checklist

- HTML markup with semantic containers used appropriately, including sections and a wrapper div;
- ids set up, used with hyperlinks in the navigation menu
- Basic fonts, colors, borders;
- Box model properties;
- Centered headings and wrapper;
- Floated images with space separating from text and efficient CSS selectors using classes;
- Horizontal menu using display property.

Hand In

For all the Lab Exercises in this course, you need to do both of the following:
1. Show your work to the instructor before the end of the class to get the mark.
2. Hand in the work on D2L (Learning Hub) following the instructions below before the deadline.

Create a new folder called **screenshots** that contains your validation screenshots and add it to the **a3-firstname-lastname** folder that you created at the beginning. Make sure the work you're submitting does not contain any unnecessary files, such as the screenshots from the starter files. Create a Zip archive from the **a3-firstname-lastname** folder and hand it into the D2L assignment folder. Note that it's important to name the folder *before* zipping it so that we can see the name when we extract it.

Resources:

- Box shadow:
  - https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow
- rgb() function:
  - https://www.w3schools.com/cssref/func_rgb.asp
- Box model and Display Property:
  - https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model#Block_and_inline_boxes (on this page, only this one section is necessary for now)
  - https://css-tricks.com/the-css-box-model/
  - A good introduction to display property: https://www.w3schools.com/css/css_display_visibility.asp
  - Inline-block: https://www.w3schools.com/css/css_inline-block.asp