Assignment 2: Intermediate HTML and Introduction to CSS

The purpose of this assignment is to practice using some more HTML tags to code the meaning of elements within our document. We'll also create a multi-page website and add some CSS code for visual styling.
As usual, you'll see screenshots of what your final project should look like in the starter files, and resources can be found at the end of this document.
Create a new assignment folder called **a2-firstname-lastname** and within that folder, create a new file called **index.html.**

Task 1: HTML for index.html, the Home Page

Don't worry about the colors and styles of the site for now. We're going to start by laying down the HTML code for the homepage in **index.html**. You can see how your site should look after tasks 1 and 2 by checking the screenshots in the **screenshots-no-css** folder.

This site is going to make use of some semantic tags to separate parts of the site that have different functions; namely, **<header>, <nav>, <main>,** and **<footer>**. *What do you think each of these elements is used for?* Add these elements to your page.

As you can see from the screenshots, the subject of the site should be your favorite foods. Choose three categories of food, and three foods within each category that you like. ***If you have privacy concerns about sharing your real info, please feel free to use fake info instead.***

Go ahead and add the appropriate HTML mark up code inside the **<header>, <nav>, <main>** and **<footer>** elements you created. Some hints:

- Remember the document hierarchy when choosing heading tags.
- Use an HTML list for the hyperlinks inside the nav menu. Don't worry about the **href** attributes for the links yet; just leave them blank or fill them in with "#".
- Use an HTML list for the overview of items;
    - This one is a bit tricky, because you're going to need to use *lists within lists.* First, you should create a list with the three categories of food items. (in my example, these are Main Dishes, Desserts, and Hot Drinks.) Then, the list item for each category should contain *another list* that has the foods that belong to this category. The hint here is that a **<ul>** element is only allowed to contain an **<li>** as a direct child. After you've written some code that you think will work, test it by validating it using the W3C validator.

Task 2: Other Pages

As you can see from the screenshots, your site should actually have a separate page for each category of food. Go ahead and create three new HTML files with names that make sense given each category. Each new file should be inside a new folder called **pages.**

Note: when choosing filenames, I recommend using all lowercase letters, with no spaces or special characters. (Using hyphens - should be okay.) This will save us from trouble later when we host our sites on a web server.

Each of these pages is going to have the same basic structure; basically, the main content of each page will contain a heading with the food category and an unordered list of foods in that category. Remember to update the **<title>** tag of each page to include the correct name of the page. The header, nav, and footer with be *almost* the same on each page.

I say "almost" because there will be a few changes:

- The <h1> heading on each page *other than the home page* should be a link back to home page;
- The links in the nav menu on each page should work. That is to say: you should be able to visit each page using the nav menu. This means that the **href** attributes in the nav menu may be different depending on the page.
    - Use this resource to figure out how to get the hyperlinks to work in the above cases: https://www.w3schools.com/html/html_filepaths.asp

One last thing; on the homepage in the overview list, change the name of each category to a link to that category page.

Check the screenshots in the **screenshots-no-css** folder to make sure everything looks okay. Also validate the code, using the W3C validation service, for each of your pages to make sure there are no errors.

Task 3: CSS Styles

You can see how your site should look after tasks 3 and 4 by checking the screenshots in the **screenshots-css** folder.

We're now going to add some CSS code to our site to control its styling and design. Create a new file called **style.css** in a folder called **css.**

In **index.html**, create a new **<link>** tag in the **<head>** of the document, and make sure the **href** attribute points to the stylesheet file you just created. Remember that the **style.css** file is in a folder called **css**, so you'll have to use the correct relative path to the file. (Note that a <**link>** tag is different than an anchor **<a>** (or hyperlink) tag.)

The first bit of styling we're going to add is to change the appearance of the text on the page. To do this, we can select the **root** node of the document, which is the highest level HTML element: the **<html>** element!

```
html {
/* the code we place here will affect the entire document */
}
```

All code between the curly brackets will be applied to the page. So, let's change the font family:

```
html {
        font-family: "Helvetica", sans-serif;
}
```

Basically, this means that the browser will apply the Helvetica font to the page if that font has been installed on the user's system, and will fall back to the generic browser default sans-serif font if not. (which is, in most cases, Arial.)

(If you're not sure what's meant by the term "sans serif", you may optionally check this resource: https://www.w3schools.com/css/css_font.asp)
Now, check the preview of your site. You should see that the text looks noticeably different than it did before! If it's not working, check the syntax of your CSS, and make sure you've linked the **index.html** and **style.css** file properly. Note that you can validate your CSS code using the following W3C service: https://jigsaw.w3.org/css-validator/

Task 4: More Styles:

For the following tasks, these resources may be helpful:

- CSS text properties:
    - https://www.w3schools.com/css/css_text.asp (see all articles in the CSS Text category)
- CSS font properties:
    - https://www.w3schools.com/css/css_font.asp
- CSS borders:
    - https://www.w3schools.com/css/css_border_shorthand.asp
- CSS units:
    - https://www.w3schools.com/cssref/css_units.asp

Complete the following tasks by writing CSS code in **style.css**:

- The lines of text on the page are a bit vertically crowded; increase the line height to **1.5**.
- Select the **body** of the document and change its **background color** to **lightcyan**.
- Select the **h1** heading and do the following:
    - set its color to **mediumvioletred**;
    - Set the font size to **60 pixels**;
    - By default, the browser renders headings in bold text. Find a way to change the **font weight** of the **h1** heading to **normal.**
- Set the color of the **h2** headings to **darkturquoise**;
- Change the color of the hyperlinks on the page to **palevioletred**.
- Figure out how to remove the underline from hyperlinks;
- Change the background color of the header, nav, main, and footer to white. You can select all four of these elements at the same time by separating the selectors by commas, like this:

```
header,nav,main,footer {
        /* your code here. */
}
```

- Add a border around both h1 and h2 headings (using a single selector) that has the following properties:
    - color of **darkturquoise;**
    - style of **solid**;
    - width of **2 pixels**.
- Select *only the list items that are nested one level deep;* that is, only list items that are contained inside of other list items. Change the background color for these list items to **lemonchiffon** and set the font family to **cursive**. (Since this is a default fallback font family, your browser might render this font a bit differently than in my screenshots.)

  To select an item that is inside of another item, check this resource on **descendant selectors/combinators:** https://css-tricks.com/almanac/selectors/d/descendant/

Your homepage should now look very much like the one in the screenshots! Now you just have to get the other pages to use these styles that you've written. To do this, figure out how to edit the <link> tag on each page to link to the **css/style.css** file you created. Note that every page should use the same CSS file. *Why do you think we prefer to use the same file for all four pages?*

Task 5: Color Accessibility

When I take off my glasses, I find it slightly hard to read some of the text on the page because its color is too light with respect to the background text. You may have the same experience. We should fix this by making the text darker, since we don't want anyone to have difficulty reading it.

Accessibility guidelines have been developed for the proper text/background contrast for maximum readability. You can actually test the contrast of the text on your site to see if it's sufficient using this resource: https://webaim.org/resources/contrastchecker/

You can simply enter the color you're using for the text and background behind it, and the tool will tell you if it passes or fails several test for accessibility.

You may notice that this tool is representing colors using a six-character code starting with a # symbol, like this:

#0000FF (this represents blue)

This is different than the creative names we've been using to refer to colors, such as **lemonchiffon**. Use this resource to get the six-character code that refers to each text color: https://www.rapidtables.com/web/css/css-color.html

(note that you can search the page for the specific color you're looking for by typing CTRL + F or Command + F.)

If you see that your color combination fails the contrast test, you can easily darken the text color using the provided "Lightness" slider. Darken each text color until you find that it finally passes the test. (For headings, it's sufficient for the "Large Text" test to be passed, even if the "Normal Text" test still fails.

Update the colors in your CSS code, using the six-character form, so that the contrast will be adequate.

You may be wondering how the six-character codes correspond to an actual color (these are actually called hexadecimal color codes). We'll talk about that more later, but if you're curious now, check out this article: https://www.pluralsight.com/blog/tutorials/understanding-hexadecimal-colors-simple

Check Your Work

Validate your HTML code and CSS code using their respective validation services. (Please be careful to use the correct service.) Take a screenshot of your validation results for each of the four pages and the CSS stylesheet. In Firefox, you can do this by right clicking on the page and choosing "Take a Screenshot" from the menu. (If you're using Chrome, you can consult this article for help with taking screenshots: https://www.howtogeek.com/423558/how-to-take-full-page-screenshots-in-google-chrome-without-using-an-extension/
)

Please note that your screenshot must capture all relevant information about the validation results. Please also note that misrepresenting your validation results in any way, including failure to submit negative results, is considered academic misconduct.

Checklist

- header, nav, main, footer used properly
- nested list
- relative references set up for hyperlinks between pages and links between pages and CSS file
- CSS styles
- colors adjusted for accessibility
- All code validates

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 instruction below before the deadline.

Make sure your project folder is called **a2-firstname-lastname**. Create a new folder called **screenshots** that contains your validation screenshots and add it to this folder. 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 **a2-firstname-lastname** folder and hand it in to D2L Learning Hub.

Resources:

- A great article, with examples, from MDN on semantic mark up:
  - https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure
- Absolute and relative references:
  - https://www.w3schools.com/html/html_filepaths.asp
- CSS color names, if you're wondering where I got all these names from:
  - https://www.rapidtables.com/web/css/css-color.html
- CSS text properties:
  - https://www.w3schools.com/css/css_text.asp (see all articles in the CSS Text category)
- CSS font properties:
  - https://www.w3schools.com/css/css_font.asp
- CSS background color:
  - https://www.w3schools.com/cssref/pr_background-color.asp
  -
- CSS borders:
  - https://www.w3schools.com/css/css_border_shorthand.asp
- CSS units:
  - https://www.w3schools.com/cssref/css_units.asp
- WebAIM color contrast checker
  - https://webaim.org/resources/contrastchecker/
- Even more information on fonts, if you happen to be interested:
  - https://www.freecodecamp.org/news/how-typography-determines-readability-serif-vs-sans-serif-and-how-to-combine-fonts-629a51ad8cce/)
- CSS validation:
  - https://jigsaw.w3.org/css-validator/