**Week 03**

**W03 Learning Activities**

1. [Code Comments](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-commenting.html)
2. [File and Folder Referencing](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-html-references.html)
3. [CSS Selectors: Type, Universal, Class, ID](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-css-class-id.html)
4. [CSS Layout: Float](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-css-float.html)
5. [CSS Layout: Position](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-css-position.html)
6. [CSS Variables](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-css-variables.html)
7. Optional Practice: [Enhancing the Home Page](https://byui-cse.github.io/wdd130-ww-course/week03/ponder-home-enhancement.html) | ✔ [Page Evaluation Tool](https://byui-cse.github.io/wdd130-ww-course/grader/w03-homepage.html)

**Assignments**

1. W03 Assignment: Rafting Site - [Planning Document](https://byui-cse.github.io/wdd130-ww-course/week03/ponder-rafting-introduction.html)
2. W03 Assignment: Rafting Site - [About Us Page](https://byui-cse.github.io/wdd130-ww-course/week03/prove-rafting-aboutus.html) | [Page Evaluation Tool](https://byui-cse.github.io/wdd130-ww-course/grader/w03-project-about.html)

**HTML and CSS: Programming Comments**

**Overview**

Comments written in code by developers provide a way to explain the purpose and function of that code. Since the code will need to be maintained and will change, comments can serve as reminders, 'to do' lists, code snippets, and notes for potential improvement during development. Comments are not processed or displayed to the user in the rendered document in HTML or CSS. Comments can be placed in any available white space.

"A CSS comment is used to add explanatory notes to the code or to prevent the browser from interpreting specific parts of the style sheet. By design, comments have no effect on the layout of a document. Comments can be placed wherever white space is allowed within a style sheet. They can be used on a single line, or traverse multiple lines." - MDN

**Prepare**

* HTML comments are not rendered in the browser.  
  Here is an HTML comment syntax example:

*<!-- This is an HTML comment -->*

Reference: [HTML Comments](https://developer.mozilla.org/en-US/docs/Web/HTML/Comments) - MDN

* CSS comments have no effect on the layout or presentation of a document.  
  Here is a CSS comment syntax example:

*/\* This is a comment \*/*

Reference: [CSS Comments](https://developer.mozilla.org/en-US/docs/Web/CSS/Comments) - MDN

* Use comments in your learning activities and assignments to remind yourself of things you want to reference or document for future use. While these comments are for your personal use, keep in mind that organizations may have specific commenting requirements given many developers might work on a project over its lifetime.
* The concept of comments is also used in your **git** commits which, are especially useful for tracking revision history and providing context to your commits for your development team.

**Activity Instructions**

1. Open any HTML or CSS document in VS Code.
2. Use **ctrl+/** (**⌘+/**) to comment out the line of code where you cursor is currently located or lines of code that you have selected.
3. Pressing **ctrl+/** again will toggle the commented out line(s) of code.

## File and Folder References

### Overview

This activity will summarize the important aspects and principles of linking content within anchor, image, and link element attributes. It will also discuss CSS URL parameters. Understanding these principles is an essential component of web technology and usability.

"A website consists of many files: text content, code, stylesheets, media content, and so on. When you're building a website, you need to assemble these files into a sensible structure on your local computer, make sure they can talk to one another, and get all your content looking right before you eventually upload them to a server. Dealing with files discusses some issues you should be aware of so you can set up a sensible file structure for your website." - MDN

### Prepare

#### Absolute URL References

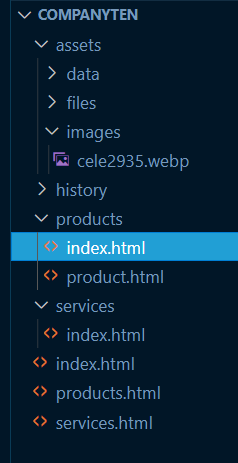
An absolute URL is a complete URL that includes the protocol, domain, and path. For example, the following is an absolute hyperlink reference (**href**) to the BYU-Idaho "About Us" page where **https** is the protocol, **byui.edu** is the domain, and **about** is the path:

<a href="https://www.byui.edu/about/" target="\_blank">About | BYU-Idaho</a>

The **target** attribute allows you to specify that a link should open in a new tab by using the **\_blank** value. Normally this decision should be left up to the user, who can choose to open a link in a new tab, new window, or private/incognito window within their browser.

#### Relative References

A relative URL is a reference that is relative to the current page. You must have knowledge of the site's file and folder structure in order to write references correctly. The following examples show how to write a relative reference from the **index.html** page located inside the **products** folder (as shown in Figure 1):

Example File Structure

##### Image File in the Images Folder Reference

Use **../** syntax to let the browser know that o go **up one directory** and out of the **products** folder. In this case, the **cele2935.webp** image file is located in an **images** folder that is within a folder named **assets**. The **assets** folder is at the same level as the **products** folder where the source file and reference are located.

<img src="../assets/images/cele2935.webp">

##### File in the Same Folder Reference

<a href="product.html">Wiget</a>

##### File in the Parent Folder Reference

<a href="../products.html">New Products</a>

##### File in the Sibling Folder Reference

This reference will automatically load the default file in the **services** directory, which is **index.html**. We do not need to reference index.html files, just their directories.

<a href="../services/">Services</a>

##### Element on the Same Page

<a href="#section-six">Section Six of the Report</a>

#### Optional Resources

* [Working with Files](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/Dealing_with_files) - MDN
* [Anchors](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a) - MDN

## CSS Selectors: Type, Universal, Class and ID

### Overview

CSS selectors allow us to target specific elements on the page to apply a CSS rule which rule is a collection of CSS declarations.

"A CSS selector is the first part of a CSS Rule. It is a pattern of elements and other terms that tell the browser which HTML elements should be selected to have the CSS property values inside the rule applied to them. The element or elements which are selected by the selector are referred to as the subject of the selector." - MDN

### Prepare

* A CSS rule's selector is the first part of the rule that tells the browser which HTML elements should have the CSS property values applied. The elements selected by the selector are referred to as the subject of the selector.

#### CSS Selector Types

|  |  |  |
| --- | --- | --- |
| **Selector Type** | **Example** | **CSS Rule Application** |
| Universal | **\* { ... }** | Applies to all elements |
| Type | **h1 { ... }** | Applies to all **<h1>** elements |
| Class | **.red { ... }** | Applies to all elements with the "red" class value |
| ID | **#red { ... }** | Applies to the one element with the "red" id value |

* A **class** attribute can be applied to many elements on a document. More than one class can be applied to a single element. The class can be selected by using period (.) notation.

#### HTML class attribute and CSS class selector examples

|  |  |
| --- | --- |
| **HTML** | **CSS** |
| **<div class="callout highlight active">** **<span class="highlight">** | **.highlight {declarations}** **.callout {declarations}** **.active {declarations}** |

* A **id** attribute can only be applied to one element per document and each element may only have one id. The id can be selected by [CSS id selector](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/id) using hash (#) notation.

#### HTML id attribute and CSS id selector example

|  |  |
| --- | --- |
| **HTML** | **CSS** |
| **<nav id="sub-menu">** | **#sub-menu {declarations}** |

### Activity Instructions

1. Navigate to this CodePen: [Class and ID Attributes](https://codepen.io/BYU-Idaho/pen/jOgXeyE)
2. In the HTML panel, remove the **note** class from the second paragraph **<p>** leaving only one class for that paragraph named **editorial**. What happens? Use the CSS panel to help answer the question.Check Your Understanding

<p class="editorial">

That second paragraph is no longer styled with the "note" class and therefore loses the italicized font style, bold font weight, and color.

1. In the HTML panel, add an **id** attribute to the fourth (last) paragraph **<p>** and name it **highlight**.
2. In the CSS panel, write a CSS rule for the **highlight** id selector with CSS declarations that
   1. underlines the text
   2. applies a yellowish background color
   3. adds some whitespace around the text (padding)

Check Your Understanding

#### HTML Alterations

<p id="highlight" class="note"> ... </p>

#### CSS Alterations

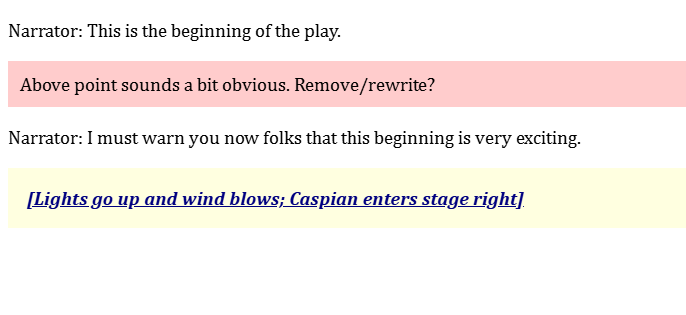
#highlight {

text-decoration: underline;

background-color: lightyellow;

padding: 1rem;

}

Class and ID Attributes with changes

#### Optional Resources

* [CSS Selectors](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors) - MDN
* [Type, Class, and ID Selectors](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Type_Class_and_ID_Selectors) - MDN

## CSS Layouts: Float

### Overview

The CSS float property should be used for specific purposes and not for general layout on a page. For example, floating related images inside containers with paragraph or other article content would be appropriate. The float property has been used for larger schemed layout purposes but should be replaced with modern, more robust CSS layout methods which will be presented later in the course.

"The float CSS property places an element on the left or right side of its container, allowing text and inline elements to wrap around it. The element is removed from the normal flow of the page, though still remaining a part of the flow (in contrast to absolute positioning)." - MDN

### Prepare

float: right;

margin: 20px;

CSS Float is a property that can be applied to elements to remove them from the normal flow of the page and allow other elements to wrap around them. The **float** property can be applied to block-level elements and inline elements. The **float** property can be set to **left** or **right** to move the element to the left or right in the document flow. For example, this code block in this paragraph is floated to the right and contains a shorthand **margin** declaration that applies a margin on all sides of the element.

* By default, the CSS float property is **none**. There is no need to apply the float declaration unless you intend to float the element.
* To avoid unexpected layout issues, using the **clear** declaration to manage elements following a floated element as needed.

### Activity Instructions

1. In this example CodePen, [Float Example](https://codepen.io/BYU-Idaho/pen/oNKJayM), to which HTML element was the float applied?  
   Adjust the rendered output screen to a smaller width to see the effect of the float on the other content on the page.Check Your Understanding

The float is applied to the **img** element that was embedded inside the **div** element with the class of **col1**.

The effect of the float on the other elements and text node on the page is that the **img** element is removed from the normal inline flow of the page and the **p** and **footer** (which is outside of the **div**) content is moved to available space around the inline **img** element.

1. In the same CodePen, what would happen if you applied the **clear: both;** property to the **footer**? Try it in CodePen.Check Your Understanding

The **footer** element content would move back to the normal flow in relation to the floated **img**.

#### Optional Resources

* [CSS Float](https://developer.mozilla.org/en-US/docs/Web/CSS/float) - MDN
* [CSS clear property](https://developer.mozilla.org/en-US/docs/Web/CSS/clear) - MDN

## CSS Layouts: Float

### Overview

The CSS float property should be used for specific purposes and not for general layout on a page. For example, floating related images inside containers with paragraph or other article content would be appropriate. The float property has been used for larger schemed layout purposes but should be replaced with modern, more robust CSS layout methods which will be presented later in the course.

"The float CSS property places an element on the left or right side of its container, allowing text and inline elements to wrap around it. The element is removed from the normal flow of the page, though still remaining a part of the flow (in contrast to absolute positioning)." - MDN

### Prepare

float: right;

margin: 20px;

CSS Float is a property that can be applied to elements to remove them from the normal flow of the page and allow other elements to wrap around them. The **float** property can be applied to block-level elements and inline elements. The **float** property can be set to **left** or **right** to move the element to the left or right in the document flow. For example, this code block in this paragraph is floated to the right and contains a shorthand **margin** declaration that applies a margin on all sides of the element.

* By default, the CSS float property is **none**. There is no need to apply the float declaration unless you intend to float the element.
* To avoid unexpected layout issues, using the **clear** declaration to manage elements following a floated element as needed.

### Activity Instructions

1. In this example CodePen, [Float Example](https://codepen.io/BYU-Idaho/pen/oNKJayM), to which HTML element was the float applied?  
   Adjust the rendered output screen to a smaller width to see the effect of the float on the other content on the page.Check Your Understanding

The float is applied to the **img** element that was embedded inside the **div** element with the class of **col1**.

The effect of the float on the other elements and text node on the page is that the **img** element is removed from the normal inline flow of the page and the **p** and **footer** (which is outside of the **div**) content is moved to available space around the inline **img** element.

1. In the same CodePen, what would happen if you applied the **clear: both;** property to the **footer**? Try it in CodePen.Check Your Understanding

The **footer** element content would move back to the normal flow in relation to the floated **img**.

#### Optional Resources

* [CSS Float](https://developer.mozilla.org/en-US/docs/Web/CSS/float) - MDN
* [CSS clear property](https://developer.mozilla.org/en-US/docs/Web/CSS/clear) - MDN

## CSS Layout: Position

### Overview

Elements are positioned on a document using the **normal flow** by default. Sometimes, the layout calls for placing items by absolute or fixed positions. Because this can lead to unintended outcomes, using absolute or fixed positions should be done sparingly and only for specific areas and not the entire document.

"The position CSS property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements." - MDN

Here is an example of using the **position** property and having text, "New Here?" and "Learn What We Believe" overlaying an image in the same HTML container:

Example of text overlaying an image in the same container

### Prepare

* Read the **absolute** and **fixed** sections under the [Values](https://developer.mozilla.org/en-US/docs/Web/CSS/position#Values) section of the CSS position in the MDN documentation.

#### Images and position/sizing

Working with images can be challenging because they have a set width and height that must be otherwise the image will appear pixelated and/or distorted. Normally, a general rule of thumb is to set only the width or the height, not both in CSS in order to maintain the image's aspect ratio. If the image containers are set, then setting the width to 100% is a common practice to ensure the image fills the container.

It is recommended that you start with the following CSS declaration for all of your work in this class to help control the aspect ratio of images.

img {

width: 100%;

height: auto;

}

The CSS above sets **all** image widths in the document to 100% of the **width** of their containers and the **height** to auto, thus maintaining the image's intrinsic aspect ratio.

The **aspect ratio**, which is the width by height measurement, is often shown as **2:1** where the width is twice the height.

In the example image from the Church's website above, the intrinsic dimensions of the image may not fit the actual container dimensions by just using an auto height. In a case like this, the object-fit property can be used to set the image to fill the container. Here is an example of the CSS rule:

img {

width: 100%;

height: 100%;

object-fit: cover;

}

### Activity Instructions

#### File Setup

1. Create a folder named "**week03**" in your wdd130 directory.
2. Create a folder named "**styles**" within the **week03** folder.
3. Create a file named "**overlay.html**" in the **week03** folder.
4. Add a CSS file named "**overlay.css**"" in the **week03/styles** folder.

#### HTML

1. Give the **overlay.html** document a basic and valid HTML structure.
2. Be sure to **link** the CSS file in your HTML document **head**.
3. Copy the HTML from this CodePen: [CSS position: absolute - overlay](https://codepen.io/BYU-Idaho/pen/poMqQJw) and paste it into the **overlay.html** document's **body**.

#### CSS

1. Copy the CSS from the same CodePen into **overlay.css** file.
2. Move the **section class="stats"** display box from the upper right to the lower left of the image.Check Your Understanding
3. ...
4. bottom: 25px;
5. left: 15px;

...

1. Remove the **position: relative;** property from the container **div** (this div has a class of **herodiv** assigned to it).  
   What happens?Answer

The information **section** overlay moves to the bottom left of the document instead of the bottom left of the **div.herodiv** container. This happens because, with the position property, the absolute positioned item is positioned relative to its ancestors that have a declared, position property. Since none have that property now, it moves to the overall document position instead of the **<div class="herodiv">** container.

1. Also note the footer using a **position: fixed** declaration.

### Submission

1. Test your page in your local browser using Live Server.
2. Commit and push your work to your wdd130 GitHub Pages enabled repository.

https://githubusername.github.io/wdd130/week03/overlay.html

1. Share and discuss your work with your peers on Microsoft Teams.

#### Optional Resources

* [Normal Flow](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Normal_Flow) - MDN
* [CSS Position](https://developer.mozilla.org/en-US/docs/Web/CSS/position) - MDN

## CSS Variables

### Overview

CSS variables are a powerful tool for managing a website's design. They allow for the creation of reusable values that can be used throughout a stylesheet, helping to maintain design congruency. This activity will introduce you to CSS variables and how to use them in your projects.

"Complex websites have very large amounts of CSS, and this often results in a lot of repeated CSS values. For example, it's common to see the same color used in hundreds of different places in stylesheets. Changing a color that's been duplicated in many places requires a search and replace across all rules and CSS files. Custom properties allow a value to be defined in one place, then referenced in multiple other places so that it's easier to work with." - MDN

### Prepare

How are CSS variables used in website projects?

#### The :root Pseudo-Class

* The **:root** pseudo-class is used to select the root element (**html**) of the document.
* It is common to see the **:root** rule at the top of the CSS document, especially since global CSS variables will be declared within that rule.

#### Writing Global CSS Variables

* Global CSS variables are declared within the **:root** selector.
* Global CSS variables are declared using the **--** prefix followed by the variable name, of your choice but follow naming conventions.

Using **Kebab** notation with hyphens (-) is OK when naming CSS variables, e.g., primary-color

* Global CSS variables are used by calling the variable name with the **var()** function.

:root {

--heading-background-color: #f06;

--heading-text-color: #333;

--heading-font: 'Montserrat', Helvetica, sans-serif;

--padding: 1rem;

--border: 1px solid rgb(0 0 0 / 0.1);

}

/\* Some example variable use in declarations. \*/

h1 {

background-color: var(--heading-background-color);

color: var(--heading-text-color);

font-family: var(--heading-font);

padding: var(--padding);

}

aside {

margin: 10px auto;

padding: var(--padding);

box-shadow: 0 0 10px rgb(0 0 0 / 0.1);

}

### Activity Instructions

1. In your **wdd130** root site, open up your existing **styles.css** file.
2. Create a **:root {}** rule at the top of the file.
3. Review your course home page CSS a look for colors, fonts, sizes, and other values that are repeated.
4. Replace those repeated CSS property values with CSS variables.
5. Test your page in your local browser using Live Server.
6. Commit and sync/push your work to your wdd130 GitHub Pages enabled repository.
7. Share your work with others on Microsoft Teams.

#### Optional Resources

* [:root](https://developer.mozilla.org/en-US/docs/Web/CSS/:root) - MDN
* [var()](https://developer.mozilla.org/en-US/docs/Web/CSS/var()) - MDN
* [CSS Variables](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Variables) - MDN
* [Using CSS custom properties (variables)](https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties) - MDN

## W03: Enhancing Your Course Home Page

### Overview

This practice activity has you apply the concepts from the learning activities to enhance your home page with HTML references, using the CSS box model, CSS selectors, and layout.

### Activity Instructions

1. Open your home page, **index.html** for editing.
2. Add an **HTML comment** at the bottom of the document, after the closing **</body>** tag, that contains the date that you have last modified the page.
3. Add an unordered list **<ul>** to your home page.
   * The list should be located in the **main** element at the end (before the closing **</main>** tag).
   * Your list must have at least three (3) list items, listing your favorite temples.
   * The listed temples must be text links to a temple detail page using the [Temple List](https://www.churchofjesuschrist.org/temples/list) at churchofjesuschrist.org or [Newsroom Temples List](https://newsroom.churchofjesuschrist.org/temples-list) or unofficially the [Temples of the Church of Jesus Christ of Latter-day Saints ORG](https://churchofjesuschristtemples.org/temples/)
   * These links must open up a new browser tab when selected.

**Hints**  
[The Unordered List element](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ul) - MDN  
[HTML <a> Attribute](https://www.w3schools.com/tags/att_a_target.asp) - w3schools

Check Your Understanding

<ul>

<li><a href="https://www.churchofjesuschrist.org/temples/details/philadelphia-pennsylvania-temple" target="\_blank">Philadelphia Pennsylvania Temple</a></li>

<li><a href="https://www.churchofjesuschrist.org/temples/details/paris-france-temple" target="\_blank">Paris France Temple</a></li>

<li><a href="https://www.churchofjesuschrist.org/temples/details/lima-peru-los-olivos-temple" target="\_blank">Berlin Germany Temple</a></li>

</ul>

1. Open your **styles.css** page for editing.
2. At the top of the document, create a CSS rule for the universal selector (wildcard) that contains three declarations:
   * Set the **margin** shorthand property to 0.
   * Set the **padding** shorthand property to 0.
   * Set the **box-sizing** property to border-box.

Setting the universal selector to have a margin and padding of zero (0) will ensure that all elements on the page will have no margin or padding, regardless of the browser defaults. This will make it easier for you to control the page layout.

When the **border-sizing** property is set to the non-default value of **border-box**, this tells the browser to account for any border and padding in the values you specify for an element's width and height. - [Box-sizing](https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing) - MDN

Check Your Understanding

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

1. Add a CSS comment at the bottom of your stylesheet that creates a comment header for a section called "Class Selectors". You will be putting your class selector rules below this comment.Check Your Understanding

*/\* Class Selectors \*/*

1. Add a class selector to your CSS after the comment. Name it "**box**".
   * Style the **box** class with basic box model properties for margin, border, padding.
   * Set the value of the margin and padding to equal 1rem which means, match the root font size.
   * Also add decorations of background-color and color of your own design.
   * Remove any duplication of these properties from the **aside** CSS rule.

Check Your Understanding

.box {

margin: 1rem;

border: 1px solid rgb(0 0 0 / .1);

padding: 1rem;

background-color: #ddd;

color: #000;

}

Adjusted **aside** rule example. (Your aside may have other properties. This is just an example.)

aside {

width: 20em;

text-align: center;

}

Did you miss the period (.) in your syntax when defining the **box** class?

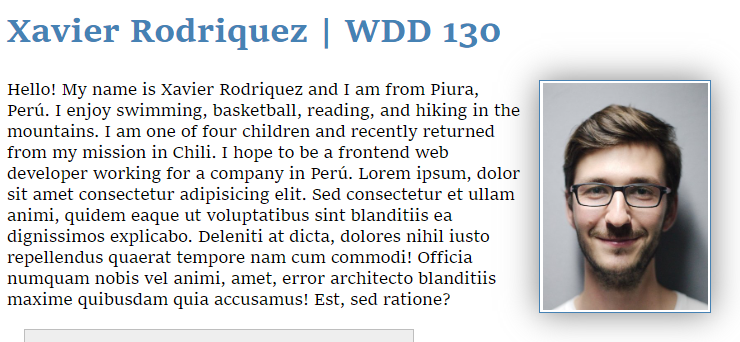
1. In your HTML, give the **aside** element a class of **box**.Check Your Understanding

<aside class="box">

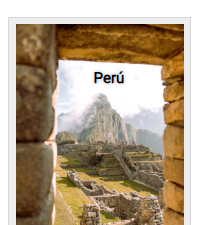
1. In your HTML, given the **ul** element list of temples a class of **box**.Check Your Understanding

<ul class="box">

Are you observing and noting the changes by running Live Server (localhost) while you make changes and build your page?

1. Use CSS to float your profile image to the right. The paragraph should come right after the picture. If your introduction is too short, you may need to add more content to ensure that the layout looks appropriate.Float Profile Image ExampleCheck Your Understanding
2. main img {
3. */\* you most likely will have other declarations here \*/*
4. float: right;

}

1. Heading Overlay ExampleUsing CSS absolute positioning, overlay the the state/country heading element on the image within the **aside** element.

This means that you will need to set the container, **aside**, to have a **position: relative;** declaration.

Check Your UnderstandingThese numbers for top and left may not work for your image. This is just an example.

aside h2 {

position: absolute;

top: 47px;

left: 150px;

}

1. In your HTML **nav** navigation bar, add two (2) more **<a>** links:
   * an external link to your LinkedIn profile if you have one. If you do not have one, link to **https://linkedin.com**.
   * an external link to your Facebook page you have one. If you do not have one or would prefer not to share it, link to **https://facebook.com**.

#### Help

Video Demonstration: ▶️ [Enhancing the Course Home Page](https://video.byui.edu/media/t/1_74i84joo)

#### Testing

1. Be sure to **continuously save and test** your page locally using the **Live Server** extension in VS Code.
2. Review your work using the course's [development standards checklist](https://byui-cse.github.io/wdd130-ww-course/dev-standards.html).
3. **Commit and sync** your page to your remote GitHub Pages enabled wdd130 repository.
4. Enter your GitHub username into this [page evaluation tool](https://byui-cse.github.io/wdd130-ww-course/grader/w03-homepage.html) to self evaluate some of the basic structure and components of your work.

## Introduction to the Whitewater Rafting Website

### Overview

This assignment introduces the course's website project, a whitewater rafting site.

### Prepare

1. Read and reference: [The Whitewater Rafting Site **Project Description**](https://byui-cse.github.io/wdd130-ww-course/week03/rafting-site-description.html)  
   This document provides an overview of the project.
2. Read: [Working with **Wireframes**](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-wireframes.html)  
   This resource introduces you to wireframes, which are used in the planning process.
3. Read: [The Website **Planning Document**](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-site-planning-doc.html)  
   This resource introduces the common components of a website planning document.

### Instructions

#### Review the Site Planing Document

* Download and read the [Whitewater Rafting Website Planning Document](https://byui-cse.github.io/wdd130-ww-course/resources/wwrsiteplan.pdf) (pdf).
* Note the example content in the following sections:
  + **Site Purpose**: The owner of a white rafting business wants to have a relevant and useful online presence. Identify the stated purpose and/or goals of having the website.

If you are unfamiliar with whitewater rafting organizations/companies, review the provided examples in the provided [Whitewater Rafting Site Project Description](https://byui-cse.github.io/wdd130-ww-course/week03/rafting-site-description.html).

* + **Target Audience**: Locate the statement(s) specifically identifying the target audience for a water rafting website. What are the characteristics of the target audience? What are the scenario questions posed about the target audience that will drive the content of the site?

You may need to review the example questions posed about target audiences from the [Website Planning Document](https://byui-cse.github.io/wdd130-ww-course/week03/prepare-site-planning-doc.html).

* + **Logo**: A logo was provided. You are free to choose this logo from the resources provided or use another logo of your own design. Here is a link to the repository of logos: [Whitewater Rafting Logos](https://byui-cse.github.io/wdd130-ww-course/resources/wwr-logos-repo.html). When you build your pages, you will need to match the name of the fictitious company provided in the logo with your headings, as needed.

If you choose a logo from those provided in the linked document, you can right click on the image and **Save image as...** a file to the **images** folder within your **wwr** folder.

* + **Wireframes**: The given wireframe is for the home page of the rafting site in a future week. Additional wireframes will be provided when you are assigned to build that page. You will have the chance to design your own wireframe for the other pages.

#### Customize Your Website Project: Logo and Style Guide

* Choose a **logo** from the provided resources or create your own logo for the whitewater rafting site. The logo should be a simple, clean design that represents the fictitious company. See the **Logo** section above for more information.
* Choose a **color palette**:
  1. Navigate to the free [coolors.co](https://coolors.co/) tool. You do not need to create an account but may do so if desired.
  2. Choose a color scheme by designing a palette of 3–4 colors that you think will work well for a rafting site.

Keep the site purpose and audience in mind. Start by thinking of a color you like or think will work well for a rafting site.

* 1. Once you have built your palette, note that at the bottom of each color column there is a six digit hex code. This is what is known as a hexadecimal color code (base 16 or 0 to F) and is usually preceded by a "#".  
     Copy that code for the colors you would like to use on your rafting site.
  2. Record the color codes and assign one color code for each of the following labeled colors for use in your website's CSS:
     + primary-color
     + secondary-color
     + accent1-color
     + accent2-color

You can change these colors later as changes can be made to the site plan in future weeks.

* 1. Save your color pallette URL from coolors.co using the URL in the address bar of your browser.

The color palette URL is the URL that appears in the address bar of your browser when you are on the coolors.co website. It will look something like this: **https://coolors.co/f2d7ee-d3bcc0-a5668b-69306d-0e103d**.

* 1. Be prepared to submit your color pallette URL and a list of colors.
* Choose your **Typography**:

Using the [Google Fonts API](https://fonts.google.com/), choose two (2) fonts to use for the rafting site. Consider what fonts you will use with headings and with body text, then record the names of the fonts and assign one font to each of the following labels.

Demonstration using Google Fonts: [Google Font Lab](https://video.byui.edu/media/t/1_5mebvivp)

* 1. heading-font
  2. text-font

Check Your Understanding

This is an **example** of embedding code in the **<head>** of your html for two fonts with limited styles (400,700,900) from Google Fonts:

**<link rel="preconnect" href="https://fonts.googleapis.com">**

**<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>**

**<link href="https://fonts.googleapis.com/css2?family=Merriweather:wght@400;700;900&family=Roboto:wght@400;700&display=swap" rel="stylesheet">**

And then, in CSS, use the **font-family** declaration. This is just an **example** application to all headings:

**h1, h2, h3, h4, h5, h6 { font-family: 'Merriweather', serif; }**

#### Submission

In this assignment you will create a portion of the site plan. The items that you will provide include a logo, four colors, two fonts, one for the headings and the other for the general body, and a coolers.co link showing the colors you have selected.

Submit a Word document that includes:

1. The image file you selected for the site logo.
2. Write out the four (4) color labels (listed below) with the color codes you selected:
   * primary-color \_\_\_\_\_\_\_\_\_\_\_\_
   * secondary-color \_\_\_\_\_\_\_\_\_\_\_\_\_
   * accent1-color \_\_\_\_\_\_\_\_\_\_\_\_\_
   * accent2-color \_\_\_\_\_\_\_\_\_\_\_\_\_
3. Write out the two (2) font labels (listed below) and the names of the fonts you selected:
   * heading-font \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   * text-font \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. Provide the URL link to your color palette that you designed in coolors.co  
   Here is an example URL link:

https://coolors.co/f2d7ee-d3cc0-a5668b-6306d-0e103d

Do not copy the example site plan for this assignment, use it for a reference to help you complete the required submission items.

## W03 Rafting Project: About Us Page

### Overview

Build the first page of the rafting website project, "**About Us**". This page provides contact information, history, and other promotions for the company. A wireframe for this page's required layout and content areas is provided.

### Instructions

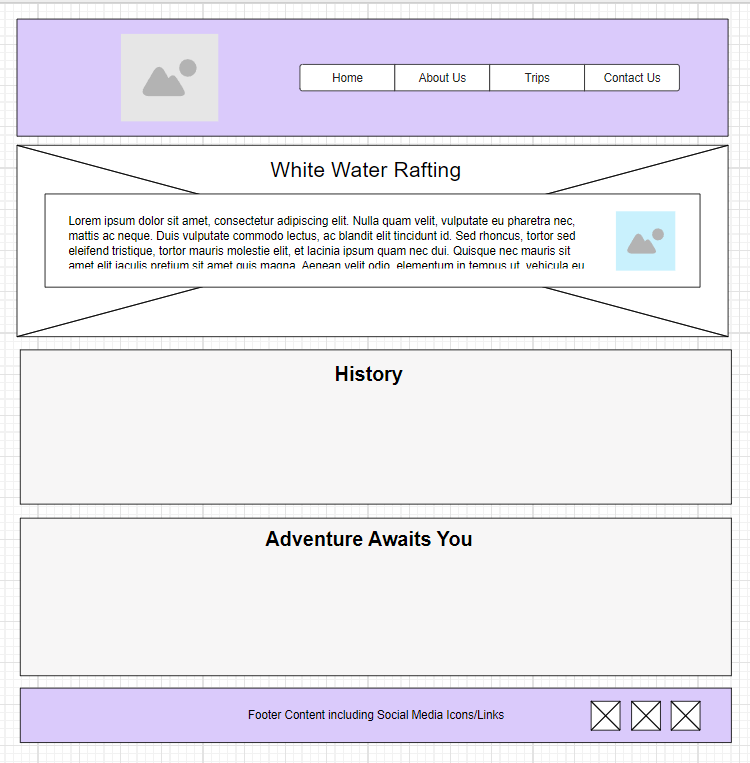
#### Folder and File Setup

1. In VS Code, be sure you have your **wdd130** directory open and **add** a sub-folder named "**wwr**" (wwr stands for white water rafting). This folder will contain the rafting website project and you will add pages and assets to this folder for the rafting project throughout the course.
2. Add two folders within this **wwr** folder per course [naming conventions](https://byui-cse.github.io/wdd130-ww-course/week01/prepare-naming-conventions.html) to contain the images, "**images**", and the stylesheet, "**styles**".

Do not be confused that you already have root level folders named images and styles. The new folders are for your rafting website project and are contained within the **wwr** project folder.

1. Add a file named "**about.html**" in the **wwr** directory.

#### Design: Study the Wireframe

1. Refer to this wireframe sketch as you markup the basic structure of the **about.html** page.Wireframe Design for Rafting Project About Us Page

You do **not** need to complete the header and footer horizontal layouts (i.e., the navigation and social media icons) during this assignment.

#### Develop the Page Structure: HTML

1. In your **about.html** document, begin your HTML page by writing the [standards based](https://byui-cse.github.io/wdd130-ww-course/dev-standards.html) basic HTML structure with the **head** with its standard, required content and an empty **body**
2. Using the typography choices you made in your [rafting site plan](https://byui-cse.github.io/wdd130-ww-course/week03/ponder-rafting-introduction.html), provide the proper **Google Font** **link** references to the fonts and font styles that you plan on using.Check Your Understanding

Here is an example of what should be in the **head**.

**<link rel="preconnect" href="https://fonts.googleapis.com">**

**<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>**

**<link href="https://fonts.googleapis.com/css2?family=Merriweather:wght@400;700;900&family=Roboto:wght@400;700&display=swap" rel="stylesheet">**

1. In the **body**, add the following three main elements: **header**, **main**, and **footer**.
2. The **header** contains two items:
   1. A rafting site **logo** that you have designed and built or that you have selected from the choices provided below. A sample logo was given in the [rafting site plan](https://byui-cse.github.io/wdd130-ww-course/week03/ponder-rafting-introduction.html) link.

Here are additional logos you can choose from if needed: [WWR Sample Logos](https://byui-cse.github.io/wdd130-ww-course/resources/wwr-logos-repo.html)

**Where should this logo and all images used on the site be stored?**Check Your UnderstandingYour logo image and all images used on the site need to be located in a **images** directory within your **wwr** directory. Remember, all images must be optimized (<= 100 kB).

* 1. A **nav** element with four child **<a>** tags and labeled as shown. The **href** attribute values, the page references, can be set now even though the pages do not all exist yet.
     + index.html (the future home page of wwr, not to be confused with the course home page.)
     + about.html
     + trips.html
     + contact.html

Do NOT worry about the horizontal layout shown in the wireframe at this point.

1. The **main** element should contain the following three(3) elements:
   1. A **div** with a class of **hero** that contains these items:
      * A hero image **img** that will be in the entire background of the **div** background.

**Image Resources**  
[Course Rafting Image Repository](https://byui-cse.github.io/wdd130-ww-course/resources/wwr-image-repo.html)  
[Web Frontend Resources](https://byui-cse.github.io/wdd330-ww-course/resources/frontend-resources.html) - see Images and Graphics section.

All images used on your site must be optimized which means not pixelated and less than or equal to 100 kB in size.

* + - A **h1** element that has the title of the rafting company as the text content.
    - An **article** element that contains:
      * An smaller **img** element that portrays a happy client or happy, working employee.
      * A **p** element with the company purpose, mission, creed, motto, etc. It is Ok to use non-sense, placeholder language for now.
  1. A **section** element with the following contents:
     + A **h2** element with a section titled "**History**".
     + A **p** element with a brief history of the company. It is OK to use nonsense language.
  2. A **section** element with the following contents.
     + A **h2** element with a section titled "**Adventure Awaits You!**".
     + A series of five (5) **img** elements representing white water rafting.

Do not worry about the layout of the images. Later you will use CSS to layout the images.

1. Check Your Understanding
2. <main>
3. <div class="hero">
4. <img src="images/..." alt="...">
5. <h1>Rafting Company Name</h1>
6. <article>
7. <img src="images/..." alt="...">
8. <p>Lorem ipsum dolor sit amet consectetur ...</p>
9. </article>
10. </div>
11. <section>
12. <h2>History</h2>
13. <p>Lorem ipsum dolor sit amet consectetur adipisicing ...</p>
14. </section>
15. <section>
16. <h2>Adventure Awaits You!</h2>
17. <img src="images/..." alt="...">
18. <img src="images/..." alt="...">
19. <img src="images/..." alt="...">
20. <img src="images/..." alt="...">
21. <img src="images/..." alt="...">
22. </section>
23. </main>
24. A **footer** element that contains:
    1. A **p** paragraph element containing a copyright symbol, the year, the rafting company name, and your name.
    2. Three (3) **img** elements that are encased by **a** anchor elements pointing to three different social media outlets. The links can be generic.Example
    3. <nav class="sociallinks">
    4. <a href="https://facebook.com">
    5. <img src="images/facebook.svg" alt="Facebook">
    6. </a>
    7. <a href="https://twitter.com">
    8. <img src="images/twitter.svg" alt="Twitter">
    9. </a>
    10. <a href="https://instagram.com">
    11. <img src="images/instagram.svg" alt="Instagram">
    12. </a>

</nav>

**Social Media Icon Resources**  
[iconfinder.com](https://www.iconfinder.com/search?q=facebook&price=free&license=gte__2)  
[Google Icons](https://fonts.google.com/icons)

#### Write the CSS

1. Create a CSS file for your rafting site named "**rafting.css**" and store it in the **styles** folder within the **wwr** folder.
2. Define the CSS variables in the document **:root** pseudo-class selector using the color scheme you selected and documented in this week's site plan/graphic identity assignment.
3. Begin by styling the headings and paragraph elements.
4. Add colors to the fonts.
5. Remove the underlines from the social media image links. Allowing the default underline on a hyperlink that is an image is not a good design and may lead to confusion of function.

To remove the default style of having underlines under **<a>** hyperlinks, use the CSS declaration

text-decoration: none;

However, the CSS rule in which you use this declaration cannot just be all hyperlinks. Your future navigation links and other hyperlinks may still want to use underlines

*/\* The following CSS selector (rule) removes underlines from all hyperlinks. Do not do this.\*/*

a {

text-decoration: none;

}

Use a specific selector that only selects the hyperlinks within the social media series of image hyperlinks.

1. Consider limiting the entire width of the page to make it easier to design the content.

#### Style the Hero Container using Position Absolute

1. The **div** with the class of **hero** is positioned **relative** in order the **h1** and the **article** elements to be positioned on top of the hero image using **absolute** positioning.
2. The hero image fits within the allocated space.

Use a relative width CSS declaration for the hero image and make the image a block to fill the width of the container.

Example

.hero img {

display: block;

width: 100%;

height: auto;

}

We use height auto in order to maintain the aspect ratio of the image which is critical. In addition, your image should be edited to be large enough to fill this space. You cannot force images to be larger than their original size otherwise the images will be pixelated or distorted.

1. The **h1** heading is positioned absolutely.
2. The **article** is positioned absolutely.
3. The **img** inside the article floats to the right of the **p** paragraph.
4. Consider styling the background-color and opacity of the **article** to fit the design and make sure it is readable.

[Click here](https://byui-cse.github.io/wdd130-ww-course/images/w03/about-sample.png) to see a sample of what the page may look like at this point.  
Your page will have different colors, fonts, content, and images than the example page, but at this point it should have a similar layout.

#### Testing

1. Every page in this course will be expected to pass the [development standards checklist](https://byui-cse.github.io/wdd130-ww-course/dev-standards.html).
2. **Validate** and correct any errors with your HTML and CSS using the [Web Developer](https://byui-cse.github.io/wdd130-ww-course/week01/setup-webdev-extension.html) browser extension under Tools and Validate Local HTML and CSS.
3. Test your work continuously as you work through the steps by having your page loaded in your local browser using **Live Server**.

You do **not** need to be connected to the internet while using Live rServer to test your pages locally during development.

1. Commit your changes and sync them to your wdd130 GitHub Pages enabled repository.
2. Audit your page using the [page evaluation](https://byui-cse.github.io/wdd130-ww-course/grader/w03-project-about.html) tool to verify that you have the basic document content.
3. Make corrections as needed and be sure to recommit and sync your updated work.

### Submission

1. Return to I-learn to submit the URL:

https://githubusername.github.io/wdd130/wwr/about.html

[**Home**](https://byui-cse.github.io/wdd130-ww-course/)[**W1**](https://byui-cse.github.io/wdd130-ww-course/week01/)[**W2**](https://byui-cse.github.io/wdd130-ww-course/week02/)[**W3**](https://byui-cse.github.io/wdd130-ww-course/week03/)[**W4**](https://byui-cse.github.io/wdd130-ww-course/week04/)[**W5**](https://byui-cse.github.io/wdd130-ww-course/week05/)[**W6**](https://byui-cse.github.io/wdd130-ww-course/week06/)[**W7**](https://byui-cse.github.io/wdd130-ww-course/week07/)

**Page Evaluation Tool: W03 - Project About Us Page**

Enter your GitHub Username:  Get Report

https://byui-cse.github.io/wdd130-ww-course/grader/w03-project-about.html