**Week 02**

**W02 Learning Activities**

1. [HTML: Common and Semantic Elements](https://byui-cse.github.io/wdd130-ww-course/week02/prepare-html-semantic.html)
2. [HTML: Block and Inline Elements](https://byui-cse.github.io/wdd130-ww-course/week02/prepare-block-vs-inline.html)
3. [CSS: An Introduction to CSS](https://byui-cse.github.io/wdd130-ww-course/week02/prepare-css-introduction.html)
4. [CSS: Values and Units of Measurement](https://byui-cse.github.io/wdd130-ww-course/week02/prepare-values-units.html)
5. [CSS: Box Model](https://byui-cse.github.io/wdd130-ww-course/week02/prepare-css-box-model.html)
6. [Troubleshooting: HTML and CSS Validation](https://byui-cse.github.io/wdd130-ww-course/week02/prepare-validation.html)

**W02 Assignment: Home Page Enhancement**

1. [Home Page Enhancement](https://byui-cse.github.io/wdd130-ww-course/week02/prove-home-enhancement-css.html) | ✔ [Evaluation Tool](https://byui-cse.github.io/wdd130-ww-course/grader/w02-homepage.html)

## HTML: Common and Semantic Elements

### Overview

HTML describes the structure of a web page document. The structure is composed of various categories of elements including the following **content** types: metadata, section, heading, embedded, and interactive elements. Some elements are associated with forms and tabular data. You are **not** expected to memorize all HTML tags. Rather, become familiar with the most commonly used elements and be able to reference others through reliable sources.

"Most HTML elements are a member of one or more content categories — these categories group elements that share common characteristics." - MDN

### Prepare

You have already applied some basic HTML to your home page including required head content. In this activity, some common HTML concepts and elements that you will use in your work are provided.

* Read: [HTML Basics](https://www.w3schools.com/html/html_basic.asp) - w3schools.com

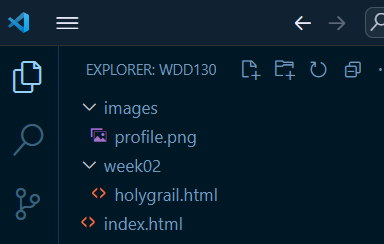
Within this article you will find an organized outline of many types of elements which you should note and reference.

* Read: [HTML Semantic Elements](https://www.w3schools.com/html/html5_semantic_elements.asp) - w3schools.com
* Read the section on [Heading Accessibility Concerns](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements#accessibility_concerns) - "Properly ordered headings that do not skip levels convey the semantic structure of the page, making it easier to navigate and understand when using assistive technologies." - MDN

### Activity Instructions

In this activity you will use HTML to build a common page layout with a **header**, **nav**, **main**, **aside**, and a **footer** element.

#### Folder and File Setup

1. In VS Code, create a "**week02**" folder if you have not already.  
   This new folder should be a subfolder of your **wdd130** main folder.
2. Create a new HTML file name "**holygrail.html**" in the **week02** folder. Make sure this HTML file is located in the **week02** sub-folder and **not** on the **wdd130** root/parent folder.File Structure for Week 02 Holy Grail Assignment

#### HTML Head Content

Start building the **holygrail.html** page by including these, baseline, required HTML elements:

Use the first two sections of the [Development Standards Checklist](https://byui-cse.github.io/wdd130-ww-course/dev-standards.html) reference, which you should bookmark, to help you with the how and why on each of these requirements.

* A valid document type. **<!DOCTYPE html>**
* An **<html>** root container with the proper language attribute.
* A **<head>** container.
* The **meta charset** element.
* The **meta viewport** element.
* A **title** element will relevant content.
* The **meta description** element.  
  The content could be: A common layout example on the web called the 'holy grail' layout.
* The **meta author** element.

Check Your Understanding

**<!DOCTYPE html>**

**<html lang="en-US">**

**<head>**

**<meta charset="utf-8">**

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

**<title>Xavier Rodriquez | Holy Grail Layout Example</title>**

**<meta name="description" content="A common layout on the web called the 'holy grail'">**

**<meta name="author" content="Xavier Rodriquez">**

**</head>**

**<body>**

**</body>**

**</html>**

It is OK to use built-in snippets in VS Code that write common code segments for you. For example, in a blank html document, type **html5** and then click the **tab** key. The basic HTML framework will be written automatically.

#### HTML Body Content

Build the common page layout within the **<body>** using the following semantic elements:

This page will not have any layout or style at this point. It is just the structure. We will add CSS styling and layout in another activity.

1. Start with a **<body>** element that will contain all the page content.
2. Add a **<header>** element.
   * Include a first level heading **<h1>** within the **header**.
   * The content of the heading can be "A Holy Grail Layout Example".
3. After the **header** ends (**</header>**), add a **<nav>** element.
   * The **nav** element contains a single unordered list **<ul>** with two list items.
   * The two (2) list items **<li>** each contain a hyperlink **<a>** to an external site of your choice.
4. Next, include a **<main>** element.
   * The **main** element contains two (2) **<section>** elements.
   * Each **section** element contains:
     + A second level heading **<h2>** with the placeholder language of "Heading", etc.
     + a single paragraph **<p>**

The paragraph content can be **filler/placeholder** text. To automatically fill the paragraph with [Lorem ipsum text](https://en.wikipedia.org/wiki/Lorem_ipsum), in VS Code type **lorem10** and then press the **Tab** key and ten (10) words will be written automatically. You can change the number 10 to the desired size.

1. Next, include a **<aside>** element.
   * In this container element, include an image element **<img>**.
     + Reference (**src**) a temple image from the [Church Media Gallery](https://www.churchofjesuschrist.org/temples/photo-gallery).  
       Copy the image URL of the image you choose by right mouse clicking on the image and selecting **Copy image address** from the popup menu.
     + Set the width attribute (**width**) to a value of "**200**" which means 200 pixels.
2. Next, include a **<footer>** element.  
   The content of the footer element can just be your name.

Check Your Understanding

**<body>**

**<header>**

**<h1>A Holy Grail Layout Example</h1>**

**</header>**

**<nav>**

**<ul>**

**<li><a href="https://byui.edu">BYU-Idaho</a></li>**

**<li><a href="https://churchofjesuschrist.org">The Church</a></li>**

**</ul>**

**</nav>**

**<main>**

**<section>**

**<h2>Heading</h2>**

**<p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Molestiae, tenetur ...</p>**

**</section>**

**<section>**

**<h2>Heading</h2>**

**<p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Molestiae, tenetur ...</p>**

**</section>**

**</main>**

**<aside>**

**<img src="https://content.churchofjesuschrist.org/templesldsorg/bc/Temples/photo-galleries/payson-utah/800x500/payson-utah-temple-celestial-room-1458545.jpg" alt="Payson Utah Temple Celestial Room" width="200">**

**</aside>**

**<footer>**

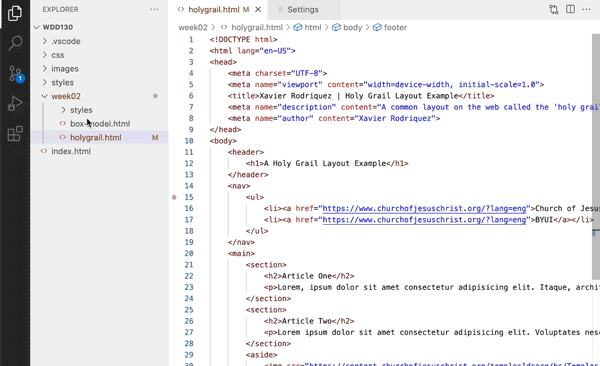
**🌴 Xavier Rodriquez**

**</footer>**

**</body>**

### Submission

1. Save, commit and push/sync your work to your wdd130 GitHub Pages remote site.
   * In VS Code, click on the **Source Control** icon.
   * Type a commit **Message** in the message input field provided. This is for your own reference and is required.
   * Commit the changes by clicking the blue **✔ Commit** button.
   * Sync your committed changes to your local **wdd130**GitHub repository by clicking the blue **Sync** button.

Commit and Push to GitHub from VSCode

1. Test your work by running this [Page Evaluation](https://byui-cse.github.io/wdd130-ww-course/grader/w02-holygrail.html) tool.
2. Share your work with class in Microsoft Teams by posting your URL to this html file:

https://yourgithubusername.github.io/wdd130/week02/holygrail.html

#### Optional Resources

* [HTML Content Categories](https://developer.mozilla.org/en-US/docs/Web/HTML/Content_categories) - MDN

[**Back**](https://byui-cse.github.io/wdd130-ww-course/week02/index.html)[**I-Learn**](https://byui.instructure.com/calendar/)

## HTML: Block versus Inline Elements

### Overview

An HTML element is, by default, either a block-level or inline element. Block elements occupy the full width of their container and start on new lines. Inline elements only occupy the space needed for the content and do not start on a new line.

"A Block-level element occupies the entire horizontal space of its parent element (container), and vertical space equal to the height of its **contents**, thereby creating a "block" ... Inline elements are those which only occupy the space bounded by the tags defining the element, instead of breaking the flow of the content."- MDN

### Preparation Material

Carefully ponder, note, and reference the materials provided and concepts presented in the learning modules.

#### Block-level Elements

Block-level elements start on a new line on a page and take up the full width of their container. This means the element container stretches out to the left and right as far as it can. Some common block-level elements are: **<p>** and **<div>** and all the semantic containers like **<header>**, **<footer>**, **<main>**, **<nav>**, and **<section>**.

#### Inline Elements

Inline elements do not start on a new line and only take up the space bounded by the tags defining the element. Some common inline elements are: **<span>**, **<a>**, **<img>**, **<button>**, and **<input>**.

Video Demonstration: ▶️ [Block versus Inline Elements](https://www.youtube.com/embed/HEnQR3PGMiU) | (4:36 mins, [Transcript](https://byui-wdd.github.io/wdd130/text/block_inline.pdf))

### Activity Instructions

Answer the following questions about HTML elements.

1. Which of the following HTML elements are **inline** elements by default?

<header> <div> <span> <table> <form> <a> <label> <button> <input> <p> <aside> <cite>

Check Your Understanding

<span> <a> <label> <button> <input> <cite>

The other elements that are listed are block-level elements by default.  
  
How are you supposed to figure out which is which? You can try them out on your page. And you should also learn to get information using reliable sources. For example, you can use the [HTML Reference](https://htmlreference.io/) to look up an element to verify if the element is block or inline.

1. Go to this code snippet on [CodePen](https://codepen.io/BYU-Idaho/pen/LYwXOLY) and note that the HTML, CSS, and JavaScript are separated into panels.  
   Focus on the HTML and the block and inline elements that are being used.
   1. In the HTML window, change the second paragraph element to a **<div>** element. What happened?Check Your Understanding

Nothing should have changed. It is OK for a **<div>** block element to contain another **<div>** element.

* 1. Now change the first or outer **<div>** element to a **<p>** element so that a paragraph contains a **div**. What happened?Check Your Understanding

This creates an unwanted result. The paragraph element ends early because, by rule, it cannot contain another block-level element. Paragraph elements can only contain inline elements.

* 1. Take a look at how the inline elements are only as big as their content and display inline if there is room. These inline elements are common. Can you identify them in the CodePen?Check Your Understanding

<span> <img> <a> and <button>

#### Optional Resources

* [Block Elements](https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements) - MDN
* [Inline Elements](https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements) - MDN
* The block-level vs. inline elements distinction was used in HTML specifications up to 4.01. Later, this binary distinction is replaced with a more complex set of [content categories](https://developer.mozilla.org/en-US/docs/Web/HTML/Content_categories).

## An Introduction to CSS

### Overview

CSS (Cascading Style Sheets) is a rule set language that defines the style and presentation of an HTML document. CSS rules select elements from the document and contain style declarations.

"Like HTML, CSS is not a programming language. It's not a markup language either. CSS is a style sheet language. CSS is what you use to selectively style HTML elements ... CSS is used to style and layout web pages — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features." - MDN

### Prepare

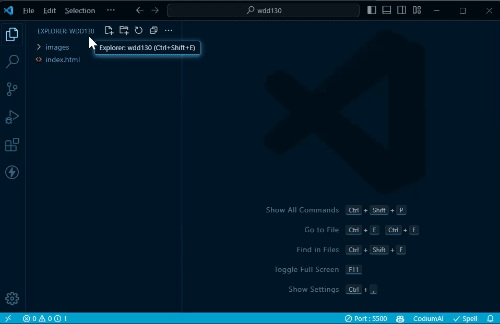
* Video Demonstration: ▶️ [What is CSS?](https://youtu.be/CDQobxknrlE) | (4:07 mins, [Transcript](https://byui-wdd.github.io/wdd130/text/what_is_css.pdf))

### Activity Instructions

In this activity, you will add some style declarations to your course home page.

Currently your home page (index.html) has default styling applied by the browser that renders the page. For example the <h1> element is styled as larger and bolder. In this activity, we will apply custom styling.

#### Create CSS Folder and File

1. In VS Code, with your **wdd130** folder open, create a new folder in the wdd130 root directory named "**styles**".
2. In the new styles folder, add a new file named "**styles.css**"

#### Write some CSS

1. Open the styles.css file and add the following CSS rules.
2. nav {
3. background-color: darkblue;
4. padding: 10px;

}

This CSS rule names the **<nav>** element as the selected element to apply two style declarations.

* 1. The background color will be the color "darkblue", which is a valid, named color.
  2. Padding, or area between the content and border is set to 10 pixels for readability.

nav a {

color: white;

display: block;

padding: 20px;

}

This CSS rule selects all the **<a>** (anchor / link) elements in the HTML that are children of the <nav> element and applies the following declarations:

* 1. The foreground or font color will be white.
  2. The display will change from the default of inline to block.
  3. Each anchor tag will have padding of 20px;

h1 {

color: darkblue;

}

This CSS rule references the **<h1>** (heading 1) element as the selected element and applies the following declaration:

* 1. The foreground, font color is set to darkblue.

img {

width: 150px;

height: auto;

border: 1px solid black;

padding: 2px;

box-shadow: 0 0 30px gray;

}

This CSS rule references all the **<img>** (image) elements as the selector and applies the following declarations:

* 1. a set width with a height that will adjust dynamically to fit the image's aspect ratio.
  2. A border shortcut that names the width, type of border, and color of the border.
  3. Padding between the image and the border.
  4. A box shadow around the image border with zero offset and depth of 30px in gray.

p {

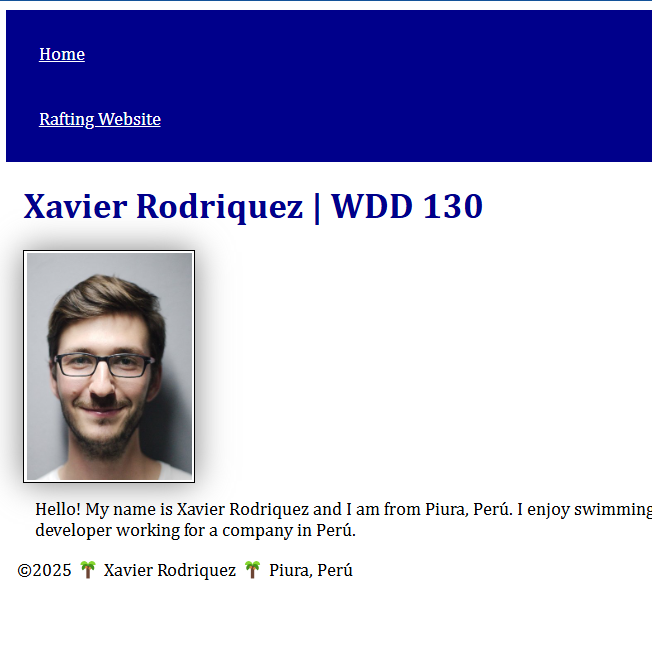
padding: 10px;

}

This CSS rule selects all the **<p>** (paragraph) elements as the selector and applies the following declaration:

* 1. Padding between the paragraph content and the border is set to 10 pixels.

1. Be sure to save your styles.css changes.

Example home page with CSS applied

#### Link the CSS file to the HTML page

1. Open the **index.html** file and add the following line in the **<head>** section of the HTML document.
2. <link rel="stylesheet" href="styles/styles.css">

1. Test the application of the style to your homepage by using **Live Server**. To do this, right mouse click on the file name in the Explorer Panel in VS Code and select **Open Live Server**.
2. Make corrections if needed.

Video Demonstration: ▶️ [Activity Walkthrough](https://video.byui.edu/media/t/1_axrugcro)

### Remote Render and Test

1. In the VS Code **Source Control** area, **publish** the site with these additions and changes by committing with a message and syncing to your wdd130 GitHub Pages remote site.

Publishing activities is good practice and the process will become more routine. Remember that the message you write with the commit is for you to give the commit some context.

1. Open up your GitHub Pages rendered URL to test the page.

#### Optional Resources

* [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) - MDN
* [CSS basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics) - MDN

## CSS: Values and Units of Measurement

### Overview

In CSS, property length values are needed to provide layout and presentation control. There are two main types - absolute and relative. The absolute length is not relative to anything else and is typically the same size on most devices. Relative length is dependent on other items including the parent, sibling, or the size of the viewport.

"Every property used in CSS has a value type defining the set of values that are allowed for that property."  
"The numeric type you will come across most frequently is the length (or unit of measurement). For example, 10px (pixels) or 30em. There are two types of lengths used in CSS — relative and absolute. It's important to know the difference in order to understand how big things will become." - MDN

### Prepare

* Video Demonstration: ▶️ [CSS: Units of Measurement](https://youtu.be/q7a7KX2N19U)| (4:47 mins, [Transcript](https://byui-wdd.github.io/wdd130/text/unit_measurements.pdf))
* Color values in CSS can be coded by:
  + **keyword**: **antiquewhite**, **burlywood**, **yellow**, **lightyellow**, etc.
  + **hexadecimal** RGB (red, green, blue): **#faebd7**  
    where the first two characters are the red value, the next two are the green value, and the last two are the blue value in hexadecimal (base16) format (0 to F).
  + **RGB or RGBA**: **rgb(250, 235, 215)** or **rgb(250 235 215 / 1)**  
    where **r** is a red value ranging from 0 to 255, **g** is a green value ranging from 0 to 255, **b** is a blue value ranging from 0 to 255, and the last value is the alpha value ranging from 0 to 1.
  + **HSLA**: **hsla(34, 78%, 91%, 1)**  
    where **h** is the hue (0 to 360), **s** is the saturation (0% to 100%), **l** is the lightness (0% to 100%), and **a** is the alpha value (0 to 1).

"There are many ways to specify color in CSS, some of which are more recently implemented than others. The same color values can be used everywhere in CSS, whether you are specifying text color, background color, or whatever else." - MDN

### Activity Instructions

In this activity, answer the following questions about CSS value units/lengths.

1. What is the most common **absolute** length unit for screens?

[Help](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units#lengths) - Scroll down to **Absolute length units** to find the answer.

Check Your Understanding

px (pixels)

1. Which CSS **relative** length unit is based upon the font size of the **root** element?Check Your Understanding

rem

1. Given a hexadecimal (base16) color value of #0e421c, what value is the **Green** level?Check Your Understanding

42

There are two hexadecimal values per color. First **red**, then **green**, and then **blue**. That is why there are six total characters with values in hex (0 to F).

#### Optional Resources

* [CSS Values and Units](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units) - MDN
* [Color Section](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units#color) - MDN
* [Named Colors / Keywords](https://developer.mozilla.org/en-US/docs/Web/CSS/named-color) - MDN

## CSS: Box Model

### Overview

The CSS Box Model is a box that wraps around every element. From the inside out, the model defines how the content, padding, border, and margin are rendered.

"When laying out a document, the browser's rendering engine represents each element as a rectangular box according to the standard CSS basic box model. CSS determines the size, position, and properties (color, background, border size, etc.) of these boxes." - MDN

### Prepare

* Video Demonstration: ▶️ [The Box Model](https://www.youtube.com/embed/Np1FuPKLb3E) | (5:19 mins, [Transcript](https://byui-wdd.github.io/wdd130/text/box_model.pdf))
* Read: [Box Sizing: Border Box](https://byui-cse.github.io/wdd130-ww-course/week02/add-box-sizing.html)

#### The CSS Box Model

The box model can be applied to HTML elements that are block-level, table, or inline-block.



* Content: This is where the text and images appear.
* Padding: This is the area around the content and separates the content from the border.
* Border: The border goes around the padding and separates the padding from the margin.
* Margin: This the the area outside of the border and can extend beyond the physical characters of the box model.

#### CSS Shorthand Notation

CSS shorthand notation is generally considered best practice for conciseness, readability, and performance. However, longhand notation has its uses and can be more appropriate in certain situations where specificity is needed.

|  |  |
| --- | --- |
| **Summary: CSS Shorthand Declarations using the margin property** | |
| **Longhand** | **Shorthand** |
| margin-top: 10px;  margin-right: 20px;  margin-bottom: 30px;  margin-left: 40px; | margin: 10px 20px 30px 40px; |
| margin-top: 10px;  margin-right: 20px;  margin-bottom 30px;  margin-left: 20px; | margin: 10px 20px 30px;  The right and left margins are the same at 20 pixels. |
| margin-top: 10px;  margin-right: 20px;  margin-bottom 10px;  margin-left: 20px; | margin: 10px 20px;  The top and bottom are the same (10 pixels) and right and left margins are the same (20 pixels). |
| margin-top: 10px;  margin-right: 10px;  margin-bottom 10px;  margin-left: 10px; | margin: 10px;  All sides of the margin are the same 10 pixels. |

|  |
| --- |
| **Common Border Shorthand** |
| border: 1px solid rgb(0 0 0 / .1);  This applies a 1 pixel border around the element with a solid line and a color that match the background of the containing/parent element. The color is set to 10% opacity. |

### Activity Instructions

#### File Setup

1. In the **week02** folder, create a file named "**box-model.html**".
2. Add a folder named **styles** and put a CSS file named "**box-model.css**" in that folder.

At this point in the course, setting up the folders and files to complete activities and assignments should be a common task where you need little direction.

#### HTML

1. Link to the CSS in your HTML document using the **<link>** tag.  
   Make sure this link element is located within the **<head>** section.Check Your Understanding

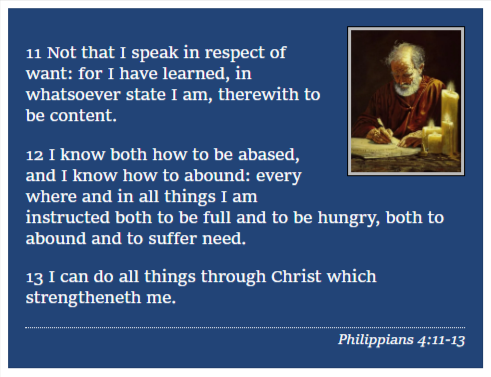
<link rel="stylesheet" href="styles/box-model.css">

1. Copy the HTML from the HTML section of the following CodePen into your **box-model.html** file.  
   You will have to add the minimum HTML outer structure to make this work. [CodePen - Box Model Start](https://codepen.io/BYU-Idaho/pen/poMQdQQ)

#### CSS

1. Now copy the CSS selectors from the CSS section of the given CodePen into **box-model.css** file.
2. Use the CSS selectors to style the HTML elements to match the image below using box model concepts for the .callout class, img, blockquote, and cite elements. The following summary of declarations is your guide:

|  |  |
| --- | --- |
| **Element** | **Required CSS Declarations** |
| callout class | maximum width, margin, border, padding, background color, and text color |
| image | position (float), margin, border, padding, background color, width, height |
| blockquote | margin |
| cite | display, top border, top padding, text alignment, size of font |

1. Screenshot of Example Solution

Check Your Understanding

CodePen - [Box Model Activity - Example Solution](https://codepen.io/BYU-Idaho/pen/XWvyzyL)

Did you notice the semantic HTML used to encase the actual scripture and scripture reference?  
**<blockquote>** was used versus a **<div>** or **<p>** and the reference was encased with a **<cite>** element. There is nothing special about these elements except their semantic names and maybe, default styling by the browser. Otherwise they behave and look like any other element.

Do not worry if you did not use these in your design. Refactoring work as you discover new elements and methods is part of the overall development process.

#### Optional Solution Discussion

* Video Demonstration: ▶️ [Activity Walkthrough](https://video.byui.edu/media/t/1_z86dcjad) | (20:00 mins)

### Testing

1. Test your page in your local browser using Live Server.
2. You can commit and sync up your work to your **wdd130** GitHub Pages enabled repository so that others may review your work, especially if you have questions. Post your link on Microsoft Teams.

#### Optional Resources

* [An Introduction to the CSS Box Model](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Introduction_to_the_CSS_box_model) - MDN

## HTML and CSS Validation

### Overview

Writing HTML markup and CSS can easily lead to errors in the syntax. You can use validation tools to help write standards-based HTML and CSS.

"HTML is not a programming language. Syntax errors in a web page do not commonly cause the web browser to refuse to open the page. This is one of the biggest reasons for the rapid adoption and spread of the web ... allowing people access to the content is more important than complaining about errors to people that won't understand them or be in a position to fix them ... Although web browsers will accept bad (invalid is the official term) web pages and do their best to render the code by making a best guess of the author's intention, it is still possible to check whether the HTML has been written correctly, and indeed it is a good idea to do so, as you'll see below. We call this validating the HTML." - Worldwide Web Consortium w3.org

### Prepare

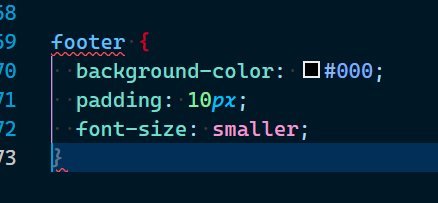
Validation means to check the HTML or CSS code using a validator tool and comparing it against the the rules of the specified doctype and, then reporting any possible errors, warnings, or other information. In this class, you will use the **Web Developer** browser extension tool to conduct the HTML and CSS validation checks, which you were asked to install as part of the course setup.

* Review Setup Activity: ['Web Developer' Browser Extension](https://byui-cse.github.io/wdd130-ww-course/week01/setup-webdev-extension.html) - w3schools.com

An demonstration of how to use the tools was provided in the setup activity.

VS Code has built in help with HTML and CSS. For example, it will provide squiggly underlining of possible issues with your HTML and CSS. Hovering over the squiggly line will provide a description of the issue. You can also use the **Problems** tab to see a list of issues.

In this example screenshot, a closing } was left off the footer selector rule in CSS. VS Code provides a red squiggly underline to indicate a problem.

Example VS Code Helper Notation

The gray } on line #73 in the figure is the system providing a possible fix which you can accept using the tab key when in focus.

### Activity Instructions

#### Validate HTML

1. Open this [demonstration page](https://byui-cse.github.io/wdd130-ww-course/week02/resources/validate-me.html) in your browser. This page contains several HTML mistakes.
2. Validate the HTML by using the **Web Developer** browser extension that you have already added to your browser.
3. Note and identify the errors, warnings, and information provided by the validation report.
4. You should see a report of over 12 errors, warnings, and information.
5. On paper, try to fix the first five messages.Answers
   * **Warning**: The opening html tag should have a language attribute.

<html lang="en-US">

* + **Info**: Remove the trailing slash on any void element like meta tags. This is not required, but is considered good practice.

<meta charset="UTF-8">

* + **Error**: The required title element was missing in the head.

<title>This title should match up with the heading 1</title>

* + **Error**: List elements cannot contain anything other than list items as direct children. Change the **<div>** divider to a **<li>** list item within the **<ul>** list element.
  + **Error**: Do not use units in width and height attributes for **<img>** image elements.

width="200" not width="200px"

The pixel unit is implied in HTML.

1. Post any questions you have about fixing the remaining errors in the **General** channel of Microsoft Teams.

#### Validate CSS

1. With the same page open, validate the CSS using the **Web Developer** browser extension.
2. Note and identify the line numbers of the CSS issues.
3. What are the issues and how might they be fixed?Answers
   1. Line 9 - Value Error: font-size property value should be **1.2pt**
   2. Line 20 - Value Error: padding should be **5px**
   3. Line 70 - Property doesn't exist. A suggestion should be provided.

#### Tips

* The validator will tell you the line number and column of the error. Use this as a starting point when searching for any issues.
* Start with the top of the list and work down when validating. Often fixing one issue will resolve additional listed errors/warnings.
* Do not try to fix everything at once. You may have to validate more than once as you start fixing issues.

### Submission

1. Share anything that you find interesting or may have some questions about with the class in Microsoft Teams.

#### Optional Resources

* [Debugging HTML](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Debugging_HTML) - MDN
* [Debugging CSS](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Debugging_CSS) - MDN

[**Back**](https://byui-cse.github.io/wdd130-ww-course/week02/index.html)[**I-Learn**](https://byui.instructure.com/calendar/)

## W02: Enhancing Your Course Home Page

### Overview

In this assignment you will apply the learning activity concepts to enhance your home page with HTML standards, CSS, and testing with the W3C validators.

### Instructions

#### Learning Activity

If you have not completed the [An Introduction to CSS](https://byui-cse.github.io/wdd130-ww-course/week02/prepare-css-introduction.html) learning activity, do so now or you will have difficulty following the instructions below and passing the page evaluation audit.

#### HTML: Add <head> meta Elements

1. Open your course home page, **index.html** for editing.
2. Add two new meta tags to your **head** section, namely after the **title** and before the CSS external file **link** element. These will be the meta description and the meta author.
3. <meta name="description" content="Describe your home page here using keywords like your name, the course, the purpose of the page.">

<meta name="author" content="Your name">

**Description**  
"Specifying a description that includes keywords relating to the content of your page is useful as it has the potential to make your page appear higher in relevant searches performed in search engines" - MDN

**Author**  
"Specifying an author is beneficial in many ways: it is useful to be able to understand who wrote the page, if you have any questions about the content and you would like to contact them. Some content management systems have facilities to automatically extract page author information and make it available for such purposes." - MDN

#### HTML: <body>

1. In the **body** section, add an **aside** element after the existing **main** element and before the **footer**.
2. In the **aside** element, add the following child elements and element content:
   * **h2** - Your state or country.
   * **img** - A picture that can represent your state or country.

Remember that all images must be [optimized](https://byui-cse.github.io/wdd130-ww-course/week01/prepare-images.html).  
This image cannot be more than 100kB in file size.

* + **p** - A brief paragraph describing your state or country.

Check Your Understanding - Example

...

</main>

<aside>

<h2>Utah</h2>

<img src="images/utah.jpg" alt="Utah landscape">

<p>Utah is a state in the western United States. It is bordered by Colorado to the east, Wyoming to the northeast, Idaho to the north, Arizona to the south, and Nevada to the west. It also touches a corner of New Mexico in the southeast. The state is a center of transportation, education, information technology, and research, and a major tourist destination for outdoor recreation. Salt Lake City, the state capital and largest city, is home to the headquarters of The Church of Jesus Christ of Latter-day Saints (LDS Church).</p>

</aside>

<footer>

...

#### CSS

1. Open the **styles.css** file for editing.
2. Add a CSS declaration to the **main** selector to limit its width for now.Check Your Understanding - Example
3. main {
4. max-width: 840px;

}

1. Add CSS declarations for the following properties for an **aside** element selector.
   * width
   * margin
   * border
   * padding
   * background color
   * color

Be sure to check the contrast ratio between your foreground and background colors. [contrast-ratio.com](https://contrast-ratio.com/)

1. Check Your Understanding - Example
2. aside {
3. width: 20rem;
4. margin: 1rem;
5. border: 1px solid rgb(0 0 0 / 0.2); */\* 1️⃣ \*/*
6. padding: 1rem;
7. background-color: #eee;
8. color: #333;
9. text-align: center;
10. }
11. 1️⃣ The border color uses the preferred rgb color method in order to take advantage of a subtle, transparent border effect. The RGB value consists of four values: red, green, blue, and alpha. The first three values (0, 0, 0) represent black, and the last value (.2) represents the alpha or opacity of the color. An alpha value of .2 means that the color is mostly transparent, allowing the background to show through. By using this border color, the border appears very light and barely noticeable, while still providing a subtle visual cue that helps to separate elements on a webpage. Additionally, it can help to add a layer of visual interest to the design without being overly distracting or detracting from the overall layout.
12. Set the **img** inside the **aside** to have a width of 200 pixels and a height of auto.

This auto setting ensures that we do not change the images aspect ratio.  
Why is that important?

Check Your Understanding

aside img {

width: 200px;

height: auto;

}

Changing the native aspect ratio of an image will distort or pixelate the image.

1. To not allow the existing **img** CSS selector to affect this new image, change the selector for the existing profile **img** to be more exclusive by adding the **main** parent selector.
2. main img {
3. width: 150px;
4. height: auto;
5. ...

}

1. Change the existing paragraph **p** padding property value to 0 pixels.  
   Note that any 0 value does not need to specify the units.

p { padding: 0;}

1. Apply CSS to the **<footer>** element to 1️⃣add a top margin to separate it from the **main** element content, 2️⃣add a top border to design additional distinction from the **main** content, and 3️⃣align any text in the **footer** to the center of its content box.Check Your Understanding
2. footer {
3. margin-top: 15px;
4. border-top: 1px solid #000;
5. text-align: center;

}

**Explanation**  
**margin-top** specifies just the top of the element to have a margin of 15 pixels.  
**border-top** uses a [shorthand](https://www.w3schools.com/csS/css_border_shorthand.asp) property list to set the border-width, border-style, and border-color in one declaration versus three.  
**text-align** sets the text alignment to center.

1. Be sure to test your page locally often using the **Live Server** extension in VS Code. Remember that server tool will render the page in your default browser as a web page.

You do not need to be on the internet to test your page locally when running it with Live Server.

1. **Save** your work.

Video Demonstration: ▶️ [Home Page Enhancement](https://video.byui.edu/media/t/1_5xptpphz) | (10.00 mins)

#### Testing

* 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).  
  Review your rendered page and markup against the checked items.
* Validate the HTML by running the **Web Developer** extension in your browser. Correct any errors.
* Validate the CSS. Correct any errors.
* Commit and sync your changes to your wdd130 GitHub Pages enabled repository.
* Use this [page evaluation](https://byui-cse.github.io/wdd130-ww-course/grader/w02-homepage.html) tool to self check some of the basic requirements.

### Submission

1. Return to I-learn to submit your wdd130 GitHub Pages home page URL:

https://yourgithubusername.github.io/wdd130