**W4 Adding More HTML and CSS**

**Activity Directions**

Estimated time: 60 minutes

This week, we will continue with the webpage we started in the previous activities. Open up the index.html file from last week in your editor to begin.

We got all the content in our page last week, but you might have noticed it doesn't look too great. Remember that HTML is responsible to get content on a page, but to style it we use CSS.

By the end of this activity, it won't look too much better, it will still take a few weeks before it looks like the wireframe.

The first part of this activity will use the following HTML concepts or properties (review the links if you need to):

* [HTML5 Elements](http://www.w3schools.com/html/html5_new_elements.asp)
* [HTML Semantics](http://www.w3schools.com/html/html5_semantic_elements.asp)
* [HTMl Block and Inline](http://www.w3schools.com/html/html_blocks.asp)
* [HTML Links](http://www.w3schools.com/html/html_links.asp)



**Review Wireframe**

First, review the  [wireframe](https://byui-wdd.github.io/wdd130/rafting_images/wireframe_home.png) again to help remember what we are trying to eventually do. Last week we added all of our content to our page, along with some extra HTML to group up related content as necessary.

**Note**

At this point, do not expect your page to *look* like the wireframe. We are laying a foundation, but have not learned how to position elements yet. Don't worry if your page does not look like the wireframe at the end of the activity.



**Internal links**

Your page has <a> tags to link to your other internal web pages that belong to your website. The <nav> element contains these links. You will rename one of the pages later for the child pages that you will be completing.

It is often considered a good practice if you have a link that takes you away from the current site to have it open in a new window or tab. This can be done with the  **target="\_blank"** attribute.

We will do that in the next step.



**External links**

There are some external links in the footer to social media sites. In this case, it was appropriate to make the images the link instead of text.

Make sure the all the social media links open in a new window/tab of the browser with the target="\_blank" attribute added to the opening tag of each <a>. Since these are external sites, it's good practice to open them in a new tab.

More on the target attribute at [w3schools target attribute](https://www.w3schools.com/tags/att_a_target.asp).

Here is an example:

<a href="https://facebook.com" target="\_blank">

<img src="images/facebook.png" alt="fb icon">

</a>



**Adding CSS**

We've been doing a lot with HTML so far. In fact, we have now added almost all the HTML we will need! I think it's about time we got to CSS!

The second half of the activity will use the following HTML/CSS concepts or tags (review the links if they are new to you or if you do not remember them:

* + [External Style Sheet](http://www.w3schools.com/css/css_howto.asp)
  + [CSS Syntax](http://www.w3schools.com/css/css_syntax.asp) (Including class and id selectors)
  + [Background Colors](http://www.w3schools.com/css/css_background.asp)
  + [Text Colors](http://www.w3schools.com/css/css_text.asp)
  + [CSS Width](http://www.w3schools.com/css/css_dimension.asp)
  + [CSS Margin](http://www.w3schools.com/css/css_margin.asp)
  + [CSS Centering](http://www.w3schools.com/css/css_align.asp) (Top half of the page)



**External Stylesheets**

We will use an external stylesheet.

We will add a line in your HTML that looks like this:

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

The link tag tells the html file where to find the CSS it should use to style the page. We will enter all of our CSS in the style.css file.

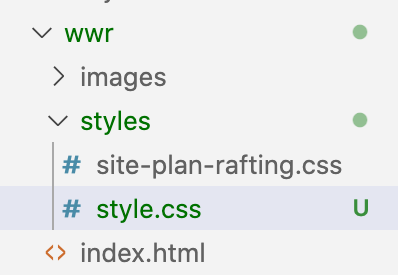
Make sure the link element is in the <head> section.

We now need to actually make this file called style.css. And from the path we see that the css file will be inside the styles folder.



**Creating the CSS File**

We will create a stylesheet inside the styles folder. Make sure you are on the Explorer icon in Visual Studio and that your project is listed there. Select the styles folder and the 'New File' icon. Name the file style.css

Make a CSS file

This CSS file will provide the styles for every page of our rafting website.



**Adding a container for our entire page**

There comes a point on larger monitors that our page will seem overwhelming with our large hero image and other content. We need to set a max-width so when our page gets so big, it will stop growing in size. To do this, go back to the HTML document and we will need a container to target in CSS that we can give a max-width to. Just add the new div element. You already have the other elements.

<body>

<div id="content">

<header>

. . .

</footer>

</div>

</body>

The div will wrap around the entire page, or in other words, the entire contents of the body. The opening <div> tag will be after the opening <body> tag and close before the closing <body> tag.



**Setting a width**

Apply a width to your whole page to keep it from stretching across the whole browser window on very large monitors by making a CSS rule for the <div> we added in the last step with the id of content. Go back to your CSS file. In the new rule we will add, let's set the max-width of the page to 1600px. Doing it on that <div> will make sure that no element on the page can be wider than that.

Why max-width instead of just width you may be asking yourself? Setting a width on an element will make it so that it is **always** that size. We never know what size of a screen someone might be looking at our page with...so forcing something to always be a size is a bit dangerous. The fact is that we don't care what size the content of our page is, as long as it's not *wider* than 1600px. It can certainly be narrower. Using max-width allows for that. Our page can be whatever size the screen...unless the screen is wider than our limit...then it will stop growing!



**Centering the page**

We set the max-width, but the page still wouldn't look right on large monitors because the whole page would be left aligned. It would look better if it were centered. Block elements are centered by setting a width and adjusting the left and right margin. ( [Margin](http://www.w3schools.com/cssref/pr_margin.asp) is the space between elements on a page or between the element and the edge of the browser window. Inline elements are centered by using the [text-align](http://www.w3schools.com/cssref/pr_text_text-align.asp) property.)

Let's center our content. The <div> we set the width on, is a block element, that means we will need to use margin to center it. The idea is to set a width (we did this in the last step), then tell the browser to take whatever margin is available and split it between the left and right of the element. Add margin: 0 auto; to the CSS rule for #content. We gave it two values. The first says to set the top and bottom margins to 0 and the second says to set the left and right margins to auto: meaning to split any margin available between the left and right.

You may not see any difference when you set the max-width and center with margin: 0 auto. This is probably because the monitor you are using to view the page is not wider than 1600px.

This is why it's always important to view your page on many different types of devices. If you want to take a quick look to see if it worked; you can zoom out of the browser page with a Ctrl- (minus sign) until it's small enough to see the margins on either side. Use Ctrl+ (plus sign) to get it back to 100% zoom when you are done.



**More Centering**

While we are centering, notice with the wireframe that many elements are centered within their parent elements. Start a CSS rule for each of the following and apply the text-align: center; style to the following selectors:

nav a  
#hero-msg h1, #hero-msg h4  
.button-box  
main section



**Adding a container for our message background**

We will be using an empty <div> tag to hold a background color for our message. Add a <div> element below the ending </section> tag and above the <img> tag. We will give it an id of background because that is all it will do: serve as a background container. There will be no content.

<section class="rapids-card">

<img class="card-img" src="images/rapids.jpg" alt="rafting boat">

<img class="icon" src="images/oars.png" alt="oars icon">

<h2>Rapids</h2>

</section>

<div id="background"></div>

<img class="mountains" src="images/mountains.jpg" alt="Misty mountains">

<section class="msg">

<h2>More Than Just The Thrill</h2>

<p>Enjoy the breathtaking scenery. From valleys, meadows, canyons, and high peaks; it's way more than just the rapids. It's a great way to get away from it all and relax amongst all the beauty of the great outdoors. </p>

<a class='join' href="rivers.html">Join Us</a>

</section>

Start a new rule in CSS targeting the new div with the id of background and give it a declaration of height: 725px;. Later we will be giving it a background color as well.

Here is an example:

#background {

height: 725px;

}



**Setting background colors**

Our next step is to add background colors to liven up our page a bit. The background-color property will set the background color of an element. Right now, the background colors default to transparent or whatever the background color of the parent element is. We want the following elements to stand out and have their own background colors.

As you apply background colors and text colors make sure you pick colors with a good contrast. Go to your site plan for colors that you had in mind for the site. Use background-color for backgrounds and color for text or font colors.

Add background-color values to the following selectors:

body  
header  
nav a:hover  
.book, .join  
.book:hover, .join:hover  
#background  
.msg  
footer

For example:

body {

background-color: #fcfbfb;

}

header {

background-color:rgba(20,30,13, .7);

}

nav a:hover {

background-color: black;

}

Think about using your own colors from your site plan.

A different shade of white was used for the background in this example.

An rgba was used in the example for the header because eventually we want to see the hero image through the header, but you can choose what you want. You can always change it later.

These colors will need to have good contrast with the colors we set in the next step.



**Foreground colors**

The color you changed the backgrounds to, may make it hard to read the text in the foreground. Use the color property for the text. The color property always refers to the font color.

Again, replace the colors with those from your chosen palette in the site plan. Pick something that looks good with the background color and that is easily readable and has good contrast. You can always use white or black if you need to.

Change the color property for all these selectors. If any selector already has a rule-set, just add the color declaration to it. Don't start a new rule-set.

nav a  
nav a:hover  
.home-title  
h4  
.book, .join  
.book:hover, .join:hover  
.msg h2  
.msg p  
footer  
footer a  
footer a:hover

For example:

nav a:hover {

background-color: black;

color: #d9c2a3;

}



**Add a Few Widths**

Let's add a few widths to some elements.

Add a width of 80px to the logo and each icon image that goes with the cards. I would use the .logo class as the selector for the logo, otherwise all your images might get that width.

Use .icon as the selector for all the card icons.

Let's give a more responsive width to our hero image. Let's give it a width of 100%. Again, use the #hero-img selector so you don't affect any other images by mistake.

For example:

.icon {

width: 80px;

}

We will set more width in later weeks as we work with layout.



**Take off or add the default underline of links**

Underlines aren't always necessary with all links, especially the links we have that will eventually look like buttons or with links that have a hover effect. We will take the underline off with the text-decoration property and set it to a value of none. We will also add the underline value to a hover as well.

Add the text-decoration property with a value of none to the following selectors:

nav a  
.book, .join  
footer a

Here is an example:

nav a {

text-align: center;

color: #fcfbfb;

text-decoration: none;

}

If you already have the selector targeted in your CSS file, just add the new declarations to it. You should **not** have more than one rule applying to the same selector.

Add a larger font size to the footer text.

Here is an example:

footer p {

font-size: 1.2em;

}

Add the text-decoration property with a value of underline to the following selectors:

footer p a:hover

Here is an example:

footer p a:hover {

text-decoration: underline;

}



**Check your work**

Publish your site and check the sharable link to make sure the web page displays correctly. Check all the hover pseudo-classes to make sure they work and look good. Once verified, submit the URL for your page to Canvas.

Make sure to [validate](https://validator.w3.org/) your page as well and fix any errors that show,

and [CSS](http://jigsaw.w3.org/css-validator/#validate_by_input), fix any errors, then submit the URL to your site in ILearn.

Your page still won't be pretty yet. It will look something like this:

