**CSS and this Lesson**

To build the data dashboard at the end of this lesson, you won't need to actually write any CSS. Instead, you'll use libraries that take care of the CSS for you. In this that, that would be the [**Bootstrap library**](https://getbootstrap.com/).

But if you are interested in understanding what Bootstrap is doing under the hood, then you need to understand how to style a website with CSS. This page has a summary of some important aspects of CSS programming.

**What is the Purpose of CSS?**

In most professional websites, css is kept in a separate stylesheet. This makes it easier to separate content (html) from style (css). Code becomes easier to read and maintain.

If you're interested in the history of css and how it came about, here is an interesting link: [**history of css**](https://www.w3.org/Style/CSS20/history.html).

CSS stands for cascading style sheets. The "cascading" refers to how rules trickle down to the various layers of an html tree. For example, you might specify that all paragraphs have the same font type. But then you want to override one of the paragraphs to have a different font type. How does a browser decide which rules apply when there is a conflict? That's based on the cascade over. You can read more about that [**here**](https://www.lifewire.com/what-does-cascade-mean-3466872).

**Different ways to write CSS**

As discussed in the video, there are essentially two ways to write CSS: **inline** or with a **stylesheet**.

Inline means that you specify the CSS directly inside of an html tag like so:

<p style="font-size:20px;">This is a paragraph</p>

Alternatively, you can put the CSS in a stylesheet. The stylesheet can go underneath an html head tag like so:

...

<head>

<style>

p {font-size: 20px;}

</style>

</head>

Or the css can go into its own separate css file (extension .css). Then you can link to the css file within the html head tag like so:

<head>

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

</head>

where style.css is the path to the style.css file. Inside the style.css file would be the style rules such as

p {

color:red;

}

**CSS Rules and Syntax**

CSS is essentially a set of rules that you can use to stylize html. The [**W3 Schools CSS Website**](https://www.w3schools.com/css/default.asp) is a good place to find all the different rules you can use. These including styling text, links, margins, padding, image, icons and background colors among other options.

The general syntax is that you:

1. select the html element, id, and/or class of interest
2. specify what you want to change about the element
3. specify a value, followed by a semi-colon

For example

a {

text-decoration:none;

}

where a is the element of interest, text-decoration is what you want to change, and none is the value. You can write multiple rules within one set of brackets like:

a {

text-decoration:none;

color:blue;

font-weight:bold;

}

You can also select elements by their class or id.

To select by class name, you use a dot like so:

.class\_name {

color: red;

}

To select by id name, you use the pound sign:

**#id\_name** {

color: red;

}

You can make more complex selections as well like "select paragraphs inside the div with id "div\_top" . If your html looks like this,

<div id="div\_top">

<p>This is a paragraph</p>

</div>

then the CSS would be like this:

div**#div\_top** p {

color: red;

}

**Margins and Padding**

The difference between margin and padding is a bit tricky. Margin rules specify a spatial buffer on the outside of an element. Padding specifies an internal spatial buffer.

These examples below show how this works. They use a div element with a border. Here is the div without any margin or padding:

<div style="border:solid red 1px;">

Box

</div>

Box

**Margin**

In this case, the div has a margin of 40 pixels. This creates a spatial buffer on the outside of the div element.

<div style="border:solid red 1px;margin:40px;">

Box

</div>

Box

**Padding**

This next case has a padding of 40px. In the case of padding, the spatial buffer is internal.

<div style="border:solid red 1px;padding:40px;">

Box

</div>

Box

**Margin and Padding**

In this case, the div element has both a margin of 40 pixels and a padding of 40 pixels.

<div style="border:solid red 1px;margin:40px;padding:40px;">

Box

</div>

Box

**Specifying Size: Pixels versus Percent versus EM Units**

In CSS there are various ways to define sizes, widths, and heights. The three main ones are pixels, percentages, and em units.

When you use px, you're defining the exact number of pixels an element should use in terms of size. So

<p style="font-size: 12px;">

means the font-size will be exactly 12 pixels.

The percent and em units have a similar function. They dynamically change sizing based on a browser's default values. For example

<p style="font-size: 100%">

means to use the default browser font size. 150% would be 1.5 times the default font size. 50% would be half. Similarly, 1em unit would be 1 x default\_font. So 2em would be 2 x default font, etc. The advantage of using percents and em is that your web pages become dynamic. The document adapts to the default settings of whatever device someone is using be that a desktop, laptop or mobile phone.

As an aside, percentages and em units are actually calculating sizes relative to parent elements in the html tree. For example, if you specify a font size in a body tag , then the percentages will be relative to the body element:

<body style="font-size: 20px">

<p style="font-size:80%">This is a paragraph</p>

...

</body>

Because different browsers might render html and CSS differently, there isn't necessarily a right or wrong way to specify sizes. This will depend on who will use your website and on what type of devices. You can read more [**here**](https://www.w3schools.com/html/html_responsive.asp). You won't need to worry about all of this because in the web app, you're going to use a CSS framework that takes care of all of this for you.