1.2.6 Add Initial CSS Styles

Okay, so let's take that wonderfully plain `<header>` we've been working on and make it look like a professional navigation bar. We'll start by exploring exactly how we can tell CSS to attach styles to specific HTML elements using `selectors`.

Now that we know the basic ins and outs of how to write CSS, let's actually do it to our page!

HIDE PRO TIP

Typically, it is a good habit to start off your CSS writing with a few styles that apply to the whole page by applying them to the topmost element. By selecting the topmost element, all "child" elements (i.e., `<header>` is the child of `<body>`) will receive the style as well. We do this because it will have an immediate effect on the page and save us the time of having to apply styles to every applicable element.

Let's add this to our CSS (if it's already defined, just overwrite it):

```
body {
  /* more on this crazy alphanumerical value in a minute! */
  color: #39a6b2;
  font-family: Helvetica, Arial, sans-serif;
}
```

By adding this, we are setting the color of the font for the entire page to a light blue / teal color with what's known as a hexadecimal number. **For more about this color type explore the Wikipedia page about the hexadecimal number.** **(https://en.wikipedia.org/wiki/Web_colors#Hex_triplet)**

The `font-family` property sets the font to all the font in the body to Helvetica. The other two values for the `font-family` definition are included just in case the user's computer does not have Helvetica installed. That way, they can fall back to those other font choices.

These are both applied to the `<body>` element on the page because the `<body>` is the parent to all of our other HTML content elements, so we can now control all of them by applying a style to the parent.

DEEP DIVE ▲

DEEP DIVE

Helvetica does not typically come installed on computers running the Windows operating system. This is because most fonts, including Helvetica, belong to companies that own and license out the fonts for a great deal of money.

Think of it like the graphic designer's version of Coke vs. Pepsi, where it's rare to see both companies' products offered at the same place. Microsoft made a deal with Monotype in the early 1990s to license their fonts and include them in Microsoft's software, but Linotype was the owner of Helvetica. Microsoft had Monotype create their own versions of Linotype's fonts (e.g., Arial), but they are all slightly different.

This is also a good time to look at CSS comments as well. We are using one above the `color` declaration:

```
/* I'm a CSS comment! */
```

This syntax is slightly different from HTML's comment syntax, but it behaves the same way. Every programming language has its own flavor of denoting a comment. Some are similar, some can be very different, but they all behave the same way.

DEEP DIVE ▲

DEEP DIVE

We'll get into more detail about web fonts and typography in the upcoming weeks, but in the meantime it might be worthwhile to learn how CSS color values work. To take a closer look start at **the MDN web docs about color values. (https://developer.mozilla.org/en-US/docs/Web/CSS/color_value)** For now, we'll be sticking to using hexadecimal values and maybe a directly named one (e.g., white, black, aquamarine, etc.).

Let's start to add design to our `<header>` with the following CSS:

```
/* apply styles to <header> */
header {
  padding: 20px 35px;
  background-color: #39a6b2;
}
```

We just told the `<header>` element to apply `padding`, which means to add space between where the `<header>` starts and where the content inside it starts. We also applied a `background-color` of a light blue / teal.

HIDE PRO TIP

The `padding` syntax can be done in multiple ways. How you do it is up to you—there is no wrong way. This methodology will also apply to some other style properties such as `margin` and `border`.

Here are a few ways that padding can be applied:

```
/* Applies 20px to every side (top, right, bottom, left) */
header {
  padding: 20px;
}

/* Applies 20px to the top and bottom, then 35px to the lef
header {
  padding: 20px 35px;
}
```

```
/* Applies 10px to the top, 15px to the right, 20px to the l
header {
  padding: 10px 15px 20px 25px;
}

/* Explicitly list the side it should be applied to*/
header {
  padding-top: 10px;
  padding-right: 15px;
  padding-bottom: 20px;
  padding-left: 25px;
}
```

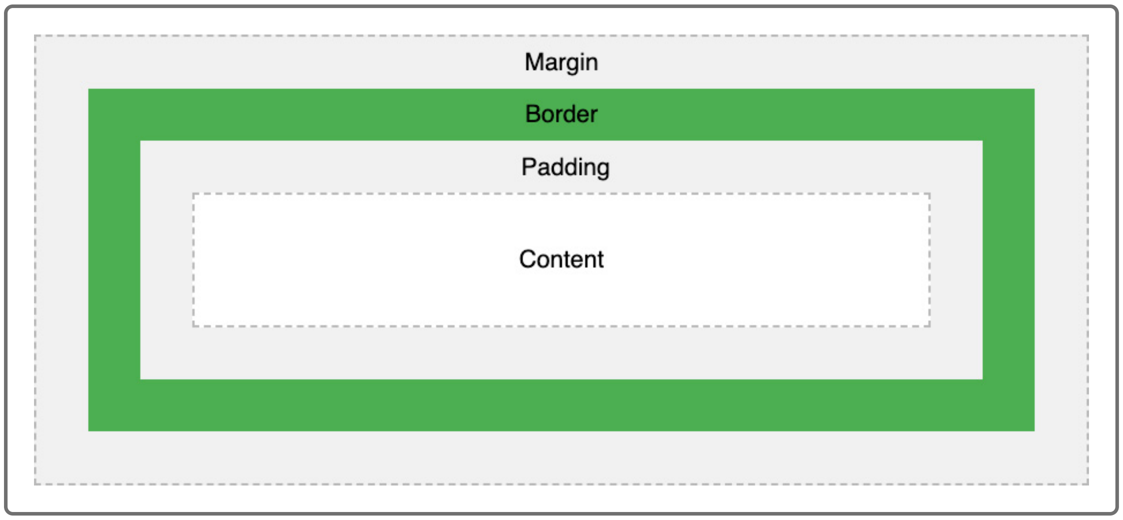
CSS properties that allow listing multiple values at once are known as **shorthand properties**. These can save a ton of time and cut out a lot of repetitive code. To learn more, see **the MDN web docs on shorthand properties (https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties)** .

Before we move on to the rest of these styles, let's take a moment to learn and understand how an HTML element's height and width dimensions are calculated for placement on a page. This is called the **CSS Box Model** and can be a little tricky at first, because we are dealing with some things that aren't visible, but it is an important concept for any web developer to know.

The CSS Box Model

All HTML elements can be represented by a rectangular box, which we can call the **CSS box**. The **CSS Box Model** is a visual display of the properties in the CSS box that includes the content, padding, border, and margins, which are all built around each other like layers in an onion. Some of the styles of each layer—like border thickness, style, and color—can be manipulated using CSS.

Here's an image of how we can visualize the box model:



Let's break down the CSS Box Model:

- **Content** is the innermost box inside our CSS box that will contain text as well as any nested elements. The content box size is determined by the height and width.

- **Padding** refers to the inside margin within the CSS box. Each of the four sides of the padding size can be specified.

- **Border** sits on the outside edge of the padding and the inside edge of the margin. This layer's sides, size, and styles can be specified, similarly to the padding and margin. Such as border-bottom or border-style or even border-top-color. This property also needs a weight of the line, style, and color in order to render.

- **Margin** behaves a lot like `padding` does, except where `padding` creates space *inside* the box, `margin` creates space *outside* the box and pushes any other HTML elements before and after it away. It also behaves like `padding` in the way its values are provided (top, right, bottom, left).

All four of these pieces are included in a browser's calculation of an HTML element's dimensions. This is something that even veteran developers can get tripped up with, as it may be easy to assume that the HTML's height or width should only be accounting for the physical/visible content inside of it. But in reality, the content is only a piece of the puzzle.

If an HTML element needed space between itself and the next HTML element, this would involve adding a margin to it and would actually increase the overall real estate that element took up. A real-world example of this would be the size of a home's property versus the size of the home itself.

If this seems like a tricky thing to nail down, don't worry. Soon, we'll be introduced to some tactics that will make our lives a lot easier. Before that, let's finish styling our header.