CSS

We've built a website to view our blog posts and it does the job, but it could look a lot nicer. We're going to use **C**ascading **S**tyle **S**heets (or CSS for short) to do this. CSS will enable us to modify the appearance of anything from a single element to entire groups of seemingly unrelated elements, even change where they sit on the page.

The first thing we need to do is import the file containing our stylesheet. We do this in the `head` element of the HTML, ensuring that the styling we want to apply has been loaded and is ready to use when the browser creates the elements.

```
<head>
    <!-- There may be other meta-data within this tag -->
    <link rel="stylesheet" href="styles.css">
    <title>My Blog</title>
</head>
```

Styling Elements

The easiest thing for us to do in CSS is to pick one of the element tags and apply a style to it. We don't need to do anything special to the HTML after we have imported the file, it's enough to simply add something to the CSS file. Let's start simple and set the background colour for our page. We want everything to have this colour, so we'll make the change in the `body` element.

```
body{
    background-color: #d0efff;
}
```

Note that we use the US-English spelling, which is common in many aspects of programming. We're using the hex-value for the colour here but we could also use an RGB encoding or, for simply-named colours, just write the name.

Note that the background colour of the entire page has changed! Let's make another change and update the font. Every browser has a selection of fonts built-in, but we can also import one from elsewhere. We're going to use Roboto from Google Fonts in this example, which means we have to load it in the `head` alongside the CSS file.

```
<head>
    <!-- There may be other meta-data within this tag -->
    <link rel="stylesheet" href="styles.css">
    <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=Roboto:wght@300
    <title>My Blog</title>
</head>
```

We can set the `font-family` attribute in the CSS file, but it's a little more complicated than the background colour. Because we are relying on an imported font we need to have a backup in place in case Google's servers are down and we can't load it. We also need to cater for different browsers which may not have font we need available, so we specify some alternatives. If our browser can't load the first in the list it will try the second, and so on.

```
body{
    background-color: #d0efff;
    font-family: Roboto, Arial, Helvetica, sans-serif;
}
```

Note that *all* the text has changed to the new font, even though none of it is directly inside the `body` tag. This is what we mean by a *Cascading* Style Sheet: styling on an element is applied to all the elements inside it too.

Styling the `body` element like this enables us to change the appearance of the whole page, but doesn't fully demonstrate the power of styling elements like this. To do that let's change the colour of our `h3` elements.

```
h3{
    color: #aaaaaa;
}
```

This has changed the colour of the text in *all* our `h3` s. This can be useful if we want to make some sort of site-wide change but has the drawback of being applied to *every* element of that type. If we want to be more selective while still changing multiple elements at once we need to use a class.

Classes

Rather than simply styling every instance of a given element the same way, we can define **classes** in CSS which can be applied to the elements in the HTML file. An element can have

multiple classes and each class can be applied to different types of element at the same time, making them a very powerful tool.

If we consider our blog posts, we see that they're pushed hard up against the left edge of the screen and there isn't much space between each one. Each one is inside a `section` element so we could restyle that, but there are other `sections` in our document which we don't want to style in the same way. We can apply a class to our blog posts which we can then apply our desired styling too.

```
<section>
    <header>
        <h2>Posts</h2>
    </header>
    <main>
        <section class="post">
            <!-- post content -->
        </section>
        <section class="post">
            <!-- post content -->
        </section>
        <section class="post">
            <!-- post content -->
        </section>
    </main>
</section>
```

First we'll push the posts away from the edge of the screen. We can adjust the elements' `margin` property to put some white space between our elements and their neighbours. When we add the class to our CSS file we prefix its name with a dot (`.`) to indicate that we are defining a class. Also note the units we give to our margin adjustment. There are many ways in which we could specify size ranging from an absolute pixel measurement to a percentage of the available screen space. Here we're specifying the margin to be `2em` , or 2 times the size of a capital M in our chosen font.

```
.post{
    margin: 2em;
}
```

We can wrap each post in a border, for which we can specify thickness, style and colour:

```
.post{
    margin: 2em;
    border: 1px solid #000000;
}
```

Finally we can put a bit of space between the content and its border by setting the `padding` value:

```
.post{
    margin: 2em;
    border: 1px solid #000000;
    padding: 1em;
}
```

Sizing content in conjunction with padding, border and margin like this is known as the **box model** and is an important aspect of design and layout.

IDs

Sometimes we want to highlight a specific element for special treatment. This is particularly useful when we add JavaScript to the mix but it also enables us to style that element in its own way with CSS. Let's tidy up our welcome section a bit by giving it an ID which we can refer to in our stylesheet.

```
<main>
    <section id="title-section">
        <h2>Welcome to my blog!</h2>
        <p>Here's a picture of me:</p>
        <img src="https://placebear.com/300/300" alt="placebear">
    </section>

...
```

We'll start by adding a border and some padding as we did with our blog posts. This time we use a hash symbol (`#`) to denote that the styles are being applied to something with a given id.

```
#title-section{
    border: 1px solid #000000;
    padding: 1em;
}
```

This is taking up the whole of the page, which doesn't look great. We can set the `width` property to change that. We'll use a percentage value here, which will set the value to be the given percentage of the element's parent's value. This will still be pushed to the side of the screen, so we set `margin` to `auto` to automatically centre the element.

```
#title-section{
    border: 1px solid #000000;
    padding: 1em;
    width: 50%;
    margin: auto;
}
```

The content is still left-aligned, however. We could set `margin` on each element inside but this could get fiddly, especially since the content inside each is a different length. Instead we will set the `text-align` property. Despite its name this will enable us to align everything inside our element the same way.

```
#title-section{
    border: 1px solid #000000;
    padding: 1em;
    width: 50%;
    margin: auto;
    text-align: center;
}
```

Responsive Design

Since mid-2019 more than half of all web traffic has come from mobile devices. When HTML and CSS were first conceived mobile devices weren't even a thing, never mind the dominant platform, and as such there was a need to develop extra tools to help. Modern CSS supports the use of **Media Queries** which tell the browser when to apply certain styles based on the size of the display. There are also tools such as **Flexbox** and **CSS Grid** which let the user design a layout which will change as the size of the screen changes. For more details on this check the further reading document.

Our website is looking much better now, showing that a little CSS goes a long way. We're still not quite there though, we can't really do much to interact with our site. Next we'll bring JavaScript into play and see how we can add some extra functionality to our site.