Flexbox

Introduction

CSS is used to control the aesthetics of a website. We've seen that CSS can be used to change the appearance of text and other elements on a page. CSS can also be used to control the layout of a page, deciding how elements should be positioned in relation to one another. In this lesson we will learn how to use Flexbox to organise child elements into rows or columns within a parent container element. This could be useful for positioning navigation links within a header, for example, or images within an image gallery.

Properties

Flexbox provides us with two sets of properties that can be used to control the layout of our elements. One that should be applied to the parent element and one that should be applied to the child elements within. CSS Tricks offer a fantastic Flexbox resource, describing and illustrating exactly what effects each of these properties can have on the layout of our elements.

Remember, the most important aspect of Flexbox - if you want to align multiple things horizontally, you need to make its parent element a flex display!

In the context of our example, the parent element will be the `header` , because the 2 element we would like to put next to each other are the follwing:

```
    <header id="main-header">
        <!-- the content of the header goes here -->
        <h1 id="logo">BrightBlog</h1>
        <nav>
            <ul id="nav-ul">
                <li><a href="" class="navbutton">Home</a></li>
                <li><a href="" class="navbutton">About us</a></li>
                <li><a href="" class="navbutton">Contact</a></li>
            </ul>
        </nav>
    </header>
```

In order to start using flexbox, the first thing that we have to do is set the `display` property of our parent element to `flex` .

```css
/* style.css */

body > header {
    background-color: orange;
    display: flex;
    height: 12vh;
}
```

If we save our page, we should see that things have changed. By default, Flexbox will try to squish all of our child elements on to a single row. This usually won't be the only thing we want, but for now, this is alreadz a huge headache removed from our life! We just need to spend a bit of time tweaking some of the properties that are available to us.

We can use the `justify-content` property to space our elements on this row - play around with the options found in the css-tricks website, for us however, `space-around` is what we want.

```css
body > header {
    background-color: orange;
    display: flex;
    height: 12vh;
    justify-content: space-around;
}
```

If you'd like to put the navlinks next to each other, remember what we've learned: we need to change the parent element to a display-flex! In this case, if you want the `li` elements next to each other, we need to target the `ul` element inside the `nav` element like so:

```css
#nav-ul {
    display: flex;
}
```

This will make the whole thing a bit cramped, so let's use what we've learned about the box model and change things around:

```css
#nav-ul > li {
    padding: 8px;
    margin: 16px;
}

body > header > nav > ul {
    list-style: none;
    text-decoration: none;
}
```

To further enhance the look of our navbar, let's add some styling to the buttons. Give the link the `class=navbutton` property in the html file like so:

```
 <ul id="nav-ul">
    <li><a href="" class="navbutton">Home</a></li>
    <li><a href="" class="navbutton">About us</a></li>
    <li><a href="" class="navbutton">Contact</a></li>
</ul>
```

and in the css file, we can target them now as follows:

```
.navbutton {
    color: white;
    text-decoration: none;
    padding: 12px;
    border: 1px solid white;
    border-radius: 10%;
}
```

Conclusion

In this lesson we saw that Flexbox can be used to position elements within a container in a way that responds better to different screen/window sizes. We looked at some of the most common properties that can be used to control the positioning of these elements and found some documentation that should help us to achieve any layout that we might need in the future.