Flexbox.

Objectives:

By the end of this chapter, you should be able to:

- Understand what flexbox is and how it differs from other forms of layout
- Use flexbox to build more complex layouts

Flexbox

We've mentioned flexbox a few times already in this unit. But what, exactly, is flexbox? Let's turn to the MDN documentation for an overview:

> The CSS3 Flexible Box, or flexbox, is a layout mode providing for the arrangement of elements on a page such that the elements behave predictably when the page layout must accommodate different screen sizes and different display devices. For many applications, the flexible box model provides an improvement over the block model in that it does not use floats, nor do the flex container's margins collapse with the margins of its contents.
>
> Many designers will find the flexbox model easier to use. Child elements in a flexbox can be laid out in any direction and can have flexible dimensions to adapt to the display space. Positioning child elements is thus much easier, and complex layouts can be achieved more simply and with cleaner code, as the display order of the elements is independent of their order in the source code. This independence intentionally affects only the visual rendering, leaving speech order and navigation based on the source order.

You can read more about flexbox here, here and here. For now, let's look at a simple example. Here's a fairly standard HTML file, with a few `div`s and some content. If you open this up in the browser (ignoring the stylesheet for now), you shouldn't see any surprises.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="style.css">
  <title>Document</title>
</head>
<body>
  <div class="container">
    <div class="first">
      First
    </div>
    <div class="second">
```

```
      Second
    </div>
    <div class="third">
      Third
    </div>
    <div class="fourth">
      Fourth
    </div>
    <div class="fifth">
      Fifth
    </div>
    <div class="sixth">
      Sixth
    </div>
  </div>
</body>
</html>
```

Now let's style this page using flexbox. Put the following code inside of a file named `style.css`. You'll probably notice quite a few new properties and values in this code. Don't worry, you'll learn about these momentarily!

```
div {
  width: 100px;
  height: 100px;
  color:white;
}

.container {
  display: flex;
  /*FLEX DIRECTION - can reverse etc*/
  flex-direction: column;
  background: #F60009;
  height: 500px;
  width: 500px;
}

.first {
  display: flex;
  background: #53007A;
  /*VERTICAL*/
  align-items: center;
  /*HORIZONTAL*/
  justify-content: space-around;
  /*ORDER  -1 first, highest last */
```

```
  order: 1;
}

.second {
  background: #FC9709;

}

.third {
  background: #A800DF;
  align-self: flex-end;
}

.fourth {
  background: #8BE7E6;
}

.fifth {
  background: #93B112;
}

.sixth {
  background: #E6F0BA;
  order: -1;
}
```

Here's a quick rundown of some of the new things in this code (along with some other common flexbox properties and values):

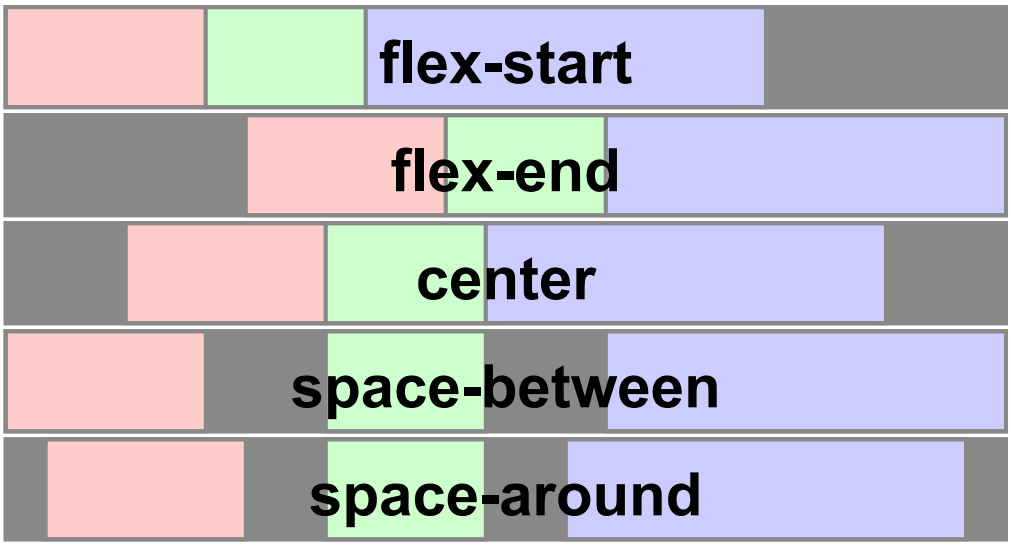
`display` - We've seen this property before. It turns out that this property is of fundamental importance when you want to use flexbox. To start using flexbox, you need to set the `display` property of an element either to `flex` or `inline-flex` to make your elements inline. (The distinction here is similar to the distinction between `block` and `inline-block`; check out this Stack Overflow question for more details.)

`flex-direction` - with the `row` value this will layout items in a row, and with the `column` value it will lay elements out in a column. The default value is `row`. These values can also have a `-reverse` added to them (`row-reverse` / `column-reverse`) to reverse the ordering.

`justify-content` - this affects the space along the `flex-direction`. You can think of this as the horizontal alignment:
- left -> `flex-start`
- center -> `center`
- right -> `flex-end`

- even amount of space between elements -> `space-between`
- even amount of space around elements -> `space-around`

This image, taken from an CSS-Tricks article on `justify-content`, does a good job illustrating the difference between the values listed above:



Note that when the `flex-direction` is `column` or `column-reverse`, `justify-content` refers to the vertical axis, not the horizontal.

`align-items` - this affects the space perpendicular to `flex-direction`. You can think of this as the vertical alignment:
- top -> `flex-start`
- center -> `center`
- bottom -> `flex-end`

(There are a few other possible values, check the docs if you're curious.)

Note that when the `flex-direction` is `column` or `column-reverse`, `align-items` refers to the horizontal axis, not the vertical.

`order` - this property allows you to order items based on their position in the DOM. The order will start at 0, so anything lower will go higher and anything higher will go lower in the order. See the HTML and CSS above for some examples of setting this property, and how it affects the layout of the page.

`align-self` - this property allows you to overwrite the value of `align-items` for particular elements. It accepts the same values as `align-items`.

`flex-basis` - This lets you manually set the size of a flexbox element's content box. At first glance it may seem like it's synonymous with the `width` property, but this isn't always the case. For more details, read this Stack Overflow question.

`flex-grow` - This lets you set proportional sizes of flex items within the same container. For example, an element with a `flex-grow` value of 2 will take up twice as much space as an element with a `flex-grow`

value of 1.

`flex-shrink` - Similar to `flex-grow`, but for shrinking elements when there's not enough space. A useful demo of this property is available at CSS Tricks.

`flex` - This is a shorthand which lets you set `flex-grow`, `flex-shrink`, and `flex-basis` all in one line (similar to the `border` property, which lets you set `border-width`, `border-style`, and `border-color`).

`flex-wrap` - When items in a flex container get too wide, what should happen? This property lets you tell the browser whether it should allow flex elements to wrap onto a new line, or whether they should just shrink in order to fit on the same line. Fore more on this, check out MDN.

There are some great tools available if you're interested in learning more about flexbox. Check out the exercises for more!