Display.

Objectives:

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

- Explain the difference between `block`, `inline-block`, and `inline` elements
- Center elements vertically and horizontally using display

Layout

Since we have a basic understanding of the box model, let's take a stab at creating some layouts using the `display` property.

`display` is CSS's most important property for controlling layout. Every element has a default display value depending on what type of element it is. The default for most elements is usually `block` or `inline`. A block element is often called a block-level element. An inline element is always just called an inline element.

Here are four of the most commonly-used values for `display`:

- `none` - An element with a `display` property of none won't show up on the page.
- `block` - A block-level element starts on a new line and stretches out to the left and right as far as it can; that is, by default it takes up all available horizontal space. Common block-level elements are `div`, `p`, and `form`. New in HTML5 are `header`, `footer`, `section`, and more.
- `inline` - An inline element can wrap some text inside a paragraph without disrupting the flow of that paragraph. `span` is the standard inline element, but there are others too, like `strong`, `b`, and `em`. The `a` element is often the most common inline element, since they are used for links.
- `inline-block` - this value is sort of a hybrid of `block` and `inline`. To see how this works, let's look at an example.

`inline-block` vs. `inline` vs. `block`

There's one more important feature of `block` elements vs. `inline-elements` that we haven't discussed yet. To understand it, take a look at the following example in a web browser:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        /* Let's line up three 200x200 boxes together */
```

```
        /* since divs have a display:block; by default, they will stack on
top of each other like blocks */

        div {
            height: 200px;
            width: 200px;
            margin: 5px;
            background: green;
        }

        /* Let's change the display property for the last two divs; how do
you think this will affect the layout? /*

        #three, #four {
            display: inline;
        }
    </style>
</head>
<body>
    <div id="one">I AM A BOX</div>
    <div id="two">I AM A BOX</div>
    <div id="three">I AM A BOX</div>
    <div id="four">I AM A BOX</div>
</body>
</html>
```

As you can see when you open up this page, the block-level elements both create new lines; were it not for the fixed width, they'd also take up all available horizontal space.

The inline elements don't create new lines: they share the same horizontal space. But you should notice something else as well: even though we're setting the `width` and `height` properties for these elements, they are only as large as the content inside of them requires! This is a general feature of `inline` elements: they don't respect the `width` and `height` property. If you set values for these properties on an inline element, those values will simply be ignored.

Knowing this, let's return to `inline-block`. Update the display on the last two divs in the above example so that their display is `inline-block` instead of `inline`. When you refresh the page, what happens? You should see that the last two divs now respect the `width` and `height` values! In fact, `inline-block` elements behave just like `inline` elements, except for the fact that you can set their `width` and `height`.

Table / Table Cell

While `block`, `inline`, `none`, and `inline-block` are four of the most common values for the `display` property, they aren't the only values. One family of values you'll sometimes see relates to table formatting. Even though tables are supported natively in HTML using `table`, `tr`, `td`, and related elements, sometimes you'll want to position elements as though they were tables without actually using these elements. For this, you can use a number of table-related `display` values (a full list can be found here.

Here's a quick example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        #table {
            display: table;
        }

        .row {
            display: table-row;
        }

        .cell {
            display: table-cell;
            border: 1px solid black;
            padding: 10px;
        }
    </style>
</head>
<body>
    <div id="table">
        <div class="row">
            <div class="cell">Data 1</div>
            <div class="cell">Data 2</div>
            <div class="cell">Data 3</div>
        </div>
        <div class="row">
            <div class="cell">Data 4</div>
            <div class="cell">Data 5</div>
            <div class="cell">Data 6</div>
        </div>
    </div>
</body>
</html>
```

Open up this page in your browser, and you should see a table on the page, even though we didn't use the `table` element!

Vertical Align

The above example might seem a bit contrived: why use less semantic HTML to build a table? And indeed, most of the table values for the `display` property are not used very often.

One possible exception is the `table-cell` value, which can be used to vertically align an element to the middle of its container. In general, vertical alignment can be kind of a pain. But take a look at this example of an element which should be aligned to the middle of the page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        #outer {
            display: table;
            width: 600px;
            height: 600px;
            background-color: blue;
            color: white;
        }

        #inner {
            display: table-cell;
            vertical-align: middle;
            text-align: center;
        }
    </style>
</head>
<body>
    <div id="outer">
        <div id="inner">
            WOAH I'M IN THE MIDDLE
        </div>
    </div>
</body>
</html>
```

The key here is that when an element's display is set to `table-cell`, it respects a new property, called `vertical-align`, which lets you adjust the vertical alignment of the element. Try commenting out the `display` properties in the two divs above, and see how that affects the layout.

For more on table cells and vertical alignment, check out this article. And if all of this feels like a hack, you'll love learning about Flexbox at the end of this unit!