CSS Display and Layout

There are many different techniques for using CSS to layout a page. HTML has a normal/default flow position for every element on the page. You can use CSS to take HTML elements and control where they are positioned relative to their normal flow position. The `display` property is used to specify the type of rendering box for an element.

Below is a list of all the different `display` property values:

```
/* <display-outside> values */
display: block;
display: inline;
display: run-in;

/* <display-inside> values */
display: flow;
display: flow-root;
display: table;
display: flex;
display: grid;
display: ruby;
display: subgrid;

/* <display-outside> plus <display-inside> values */
display: block flow;
display: inline table;
display: flex run-in;

/* <display-listitem> values */
display: list-item;
display: list-item block;
display: list-item inline;
display: list-item flow;
display: list-item flow-root;
display: list-item block flow;
display: list-item block flow-root;
display: flow list-item block;

/* <display-internal> values */
display: table-row-group;
display: table-header-group;
display: table-footer-group;
display: table-row;
display: table-cell;
display: table-column-group;
display: table-column;
display: table-caption;
display: ruby-base;
display: ruby-text;
display: ruby-base-container;
display: ruby-text-container;

/* <display-box> values */
display: contents;
display: none;

/* <display-legacy> values */
display: inline-block;
display: inline-table;
display: inline-flex;
display: inline-grid;

/* Global values */
display: inherit;
display: initial;
display: unset;
```

The most common `display` values are `block`, `inline`, `flex`, and `none`. We will cover them here.

display: block

The `block` value for `display` specifies an element's outer display type, denoting that the element generates a block box. What this means is that the element will start on a new line, and it will take up the entire width of its container. The following HTML elements are set to `display: block` by default, meaning they always start on a new line and take up the full width available.

- `<address>` (Contact information)
- `<article>` (Article content)
- `<aside>` (Aside content)
- `<blockquote>` (Long "block" quotation)
- `<div>` (A document division)
- `<footer>` (Section or page footer)
- `<h1>, <h2>, <h3>, <h4>, <h5>, <h6>` (Heading elements)
- `<header>` (Groups header information)
- `<hr>` (Horizontal rule)
- `<li>` (List item)
- `<main>` (A section containing the central content for a document)
- `<nav>` (A section containing navigation links)
- `<ol>` (Ordered list)
- `<p>` (Paragraph)
- `<pre>` (Preformatted text)
- `<section>` (A section of a page)
- `<ul>` (Unordered list)
- `<video>` (Video player)

There are other "block-level" elements in addition to the ones above, but they are less common. It's important to note that because an elemenet is a "block-level" element doesn't mean you cannot set it's display property to be something else. CSS is designed to change the way HTML is laid out be default. Sometimes it makes sense to change a "block-level" element into an "inline" element.

Example

The CSS code below would select all `button` elements and set them to be `display: block`:

```
button {
    display: block;
}
```

display: inline

The `inline` value for `display` specifies an element's outer display type. An `inline` element generates one or more inline element boxes, does not start on a new line, and only takes up as much width as necessary. This means that inline elements only occupy the space bounded by the tags defining the element, instead of breaking the flow of content. Below is a list of common elements that are inline by default:

- `<a>` (Anchor or link element)
- `<b>` ("Bring attention to" element)
- `<br>` (Line break)
- `<button>` (Button)
- `<code>` (Inline code)
- `<em>` (Emphasis)
- `<i>` (Specifies text that is set off from the normal text, like an icon)
- `<img>` (Image)
- `<input>` (User input)
- `<label>` (A form element label)
- `<select>` (Select box)
- `<span>` (Generic inline text container)
- `<strong>` (Strong importance element)
- `<textarea>` (Larger area for user text input)
- `<time>` (Represents a period in time)

The difference between "inline" and "block-level" elements are the content model and formatting. Inline elements may contain only data and other inline elements. For example, the following would be invalid syntax because a `<div>` is a "block-level" element and a `<span>` is an "inline" element:

```
<span>
    <div>Hello</div>
</span>
```

By default, "inline" elements do not force a new line whereas "block-level" elements do unless otherwise changed using CSS.

Even though there are certain elements classified as "inline," it is possible to use CSS to change their behavior. For example, the following code would turn all `div` elements into `inline` and all `span` elements into `block.`

```
div {
    display: inline;
}

span {
    display: block;
}
```

display: none

The `none` value for `display` lets you hide an element. When you set an element to `display: none`, the element and all its descendant elements will be hidden. Consider the following HTML:

```
<div class="hide">
    <p>Hello World!</p>
</div>

<div>
    <p>Goodbye World!</p>
</div>
```

and the corresponding css

```
.hide {
    display: none;
}
```

The CSS above will hide all elements that have the `hide` class on them. In the example above it will hide the `<div class="hide">` element as well as its descendants. This means that the `<p>Hello World!</p>` will also be hidden. The other `<div>` element and corresponding descendant `<p>Goodbye World!</p>` will not be hidden, because they do not have the `hide` class on them.

display: flex

The `flex` value for `display` is an inner display type, denoting that the element behaves like a block element and lays out its content according to a "flexbox model". You will get more experience with `display: flex` further on in the course.