CSS Basics

HTML defines the structure of a document, not how it looks. CSS is like the skin of a webpage to provide layout and visual styling for HTML elements.

Both HTML and CSS render together to create a static, non-changing website. It is important to note that *static* here doesn't mean totally non-interactive. CSS and HTML can be combined in powerful ways to integrate functions such as animation and minor interactivity to your website.

Making CSS Work with Your Website

You can add CSS to a webpage or element in three ways.

Inline Style Attribute

An inline style attribute accepts CSS as the attribute value on an individual HTML element:

```
style="color: red;"
```

Style Block

A style element or style block is added to the HTML:

```
<style type="text/css"> p { color: red; } </style>
```

External Stylesheet

```
<link media="screen" type="text/css" rel="stylesheet" href="">
```

Working with CSS and Monetate Actions

Many Monetate actions provide an area to include your own CSS as a part of an action. When you use a Monetate CSS field, you should write your CSS directly into this field. You will not be able to embed an external stylesheet using the `link` tag within the Monetate CSS field.

Monetate CSS fields take valid CSS input and include this CSS on the page. Often, the CSS you write in a Monetate action overrides site default CSS. If you find that your Monetate CSS is not overriding, you can increase the specificity of your CSS by adding `!important` to some rules or by increasing the specificity of your selectors.

Core Concepts of CSS

Identifier Attributes: IDs and Classes

Identifier attributes, like a class or an ID, are how you correspond CSS properties defined in a stylesheet or style block with the appropriate HTML element(s) using a selector.

An ID is formatted as `#elementID`.

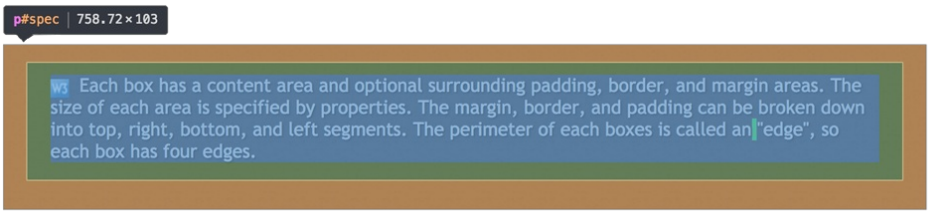
Using an ID unique way to name a single element only in a CSS stylesheet.

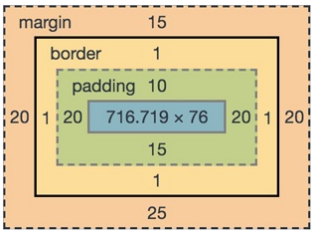
A class is formatted as `.elementClass`.

Using a class is a generic way to name one or more elements in a CSS stylesheet.

The Box Model

As you begin to utilize CSS, you will find it useful to think of every element of your website as a box. You can use CSS to style, position, remove, or resize these boxes and make your content look exactly how you want it to. One of the most basic levels of design necessary is to define how large each area should be.





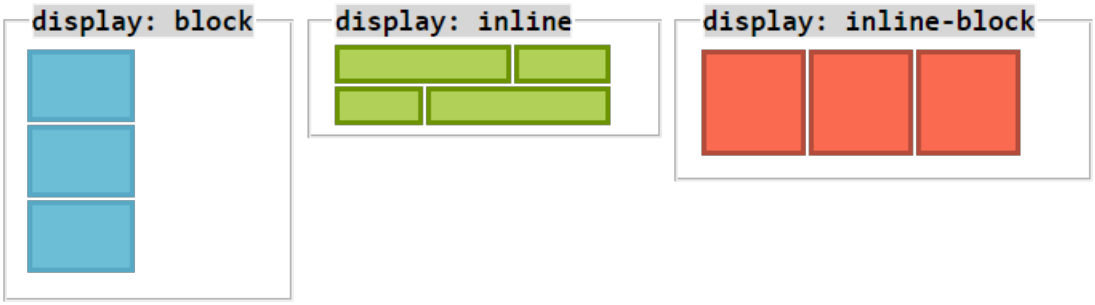
Each box has a content area and optional surrounding padding, border, and margin areas. The size of each area is specified by properties. You can break the margin, border, and padding down into top, right, bottom, and left segments. You may see a content styled in CSS as follows:

```
.calloutblock{
  width: 200px;
  height: 200px;
  border: 3px solid red;
  padding: 10px 20px;
  margin-top: 10px;
  margin-bottom: 10px;
}
```

This styling applies to the box with the class of `calloutblock`, gives the box a height, width, and border, tells how far the content should be from the border (how much padding there is—in this case, 10 px on top and bottom and 20 px on each side), and how far the box should be from other elements on the page (the margin).

You can learn more about the Box Model on CSS Tricks.

Inline vs. Block



Inline-level elements are those elements of the source document that do not form new blocks of content. The content is distributed in lines (for example, emphasized pieces of text within a paragraph, inline images, etc.). Several values of the `display` property make an element inline:

- `inline`
- `inline-table`
- `inline-block`
- `run-in` (part of the time)

Inline-level elements generate inline boxes.

Block-level elements are those elements of the source document that are formatted visually as blocks (for example, paragraphs). They begin on new lines. Several values of the `display` property make an element block-level.

The horizontal position and size of a non-floating block-level element is determined by seven properties:

- `margin-left`
- `border-left`
- `padding-left`
- `width`
- `padding-right`
- `border-right`
- `margin-right`

The sum of these seven properties is always equal to the width of the parent element.

Inline-block elements combine some properties of both inline-level and block-level elements. It allows the element a set width and height as well as margins, padding, and borders, but each consecutive element does not break to a new line automatically and only does so when the available space is filled.

Display

Remember, every element on your page is a box. The `display` property can help you define how that box behaves on your website. The commonly-used values of this property have the following meanings:

`block`

This value causes an element to be styled as a block box.

`inline`

This value causes an element to be styled as an inline box. Inline elements flow inline with other elements on the page. Inline elements ignore any specified `width` or `height` applied to them; inline elements can have `margin` or `padding` applied to them, but this will only impact the horizontal flow.

`inline-block`

This value causes an element to be styled as an inline-block box. This is very similar to `inline`, but you can use `width` and `height` values on inline-block elements.

`list-item`

This value causes an element (For example, `<li>` in HTML) to be styled as a list item. List items are just like block items (they accept `width` and `height` and do not flow together), with one big exception—list items are displayed with a `marker`, the style of which is defined with the property `list-style`.

`flex`

Allows for design with a flexible responsive layout structure that does not require using float or positioning on the elements.

`none`

This value causes an element not to appear in the formatting structure (for example, in visual media the element generates no boxes and has no effect on layout). Descendant elements do not generate any boxes either; the element and its content are removed from the formatting structure entirely. This behavior cannot be overridden by setting the `display` property on the descendants.

A display value of `none` does not create an invisible box. It creates no box at all. CSS includes mechanisms that enable an element to generate boxes in the formatting structure that affect formatting but are not visible themselves.

These values cause an element to behave like table elements.

- `table`
- `inline-table`
- `table-row-group`
- `table-column`
- `table-column-group`
- `table-header-group`
- `table-footer-group`
- `table-row`
- `table-cell`
- `table-caption`

Additional Resources

You may want to review the links below for some additional help with CSS properties:

- :nth-child from CSS-Tricks
- Basic concepts of flexbox from MDN Web Docs
- CSS Flexbox Explained by Road Tripping Across the Country from freeCodeCamp