CSS Setup and Selectors

Inline Styles
Although CSS is a different language than HTML, it's possible to write CSS code directly within HTML code using inline styles.

I'm learning to code!

Inside <style> tag
HTML allows you to write CSS code in its own dedicated section with the <style> element. CSS can be written between opening and closing <style> tags. To use the <style> element, it must be placed inside of the <head> element.

```
<head>
  <style>
    p {
      color: red;
      font-size: 20px;
    }
  </style>
</head>
```

.css file
You can create a CSS file by using the .css file name extension, like so: style.css. With a CSS file, you can write all the CSS code needed to style a page without sacrificing the readability and maintainability of your HTML file.

But if you have a separate .css file, you need to link the html file and the .css file using the **<link>** element.

<link> is a self-closing tag and requires the following three attributes:

href — like the anchor element, the value of this attribute must be the address, or path, to the CSS file.
type — this attribute describes the type of document that you are linking to (in this case, a CSS file). The value of this attribute should be set to text/css.
rel — this attribute describes the relationship between the HTML file and the CSS file. Because you are linking to a stylesheet, the value should be set to stylesheet.

e.g. <link href="https://www.codecademy.com/stylesheets/style.css" type="text/css" rel="stylesheet">

tag name / class name
CSS can select HTML elements by using an element's tag name. It's also possible to select an element by its class attribute.

Here, <p. and <h1> are tag names from html
```
p {
  font-family: Arial;
}

h1 {
  color: maroon;
}
```

In the html file say we have the following code:
<p **class="brand"**>Sole Shoe Company

Then, in the css file we can write something like this:

```
.brand{
   color: teal;
}
```

For multiple classes… say we have html code like the following:

```
<h1 class="title uppercase">Top Vacation Spots</h1>
```

Then, in the css file, we can write something like:

```
.title {
   color: teal;
}

.uppercase {
   text-transform: uppercase;
}
```

id name

HTML File

```
<h1 class="title uppercase" id='article-title'>Top Vacation Spots</h1>
```

CSS file

```
.title {
   color: teal;
}

.uppercase {
   text-transform: uppercase;
}

#article-title {
   font-family: cursive;
   text-transform: capitalize;
}
```

Chaining selectors

It's possible to require an HTML element to have two or more CSS selectors at the same time. This is done by combining multiple selectors, which we will refer to as chaining.

Say we want to write a CSS selector for multiple elements, namely, h2 elements with a class of .destination.

```
h2.destination {
   font-family: cursive;
}
```

nested elements

HTML Code

```
<ul class='main-list'>
   <li> ... </li>
   <li> ... </li>
   <li> ... </li>
</ul>
```

CSS Code

```
.main-list li {
```

…….
}

Important
!important can be applied to specific attributes instead of full rules. It will override any style no matter how specific it is. As a result, it should almost never be used. Once !important is used, it is very hard to override. The !important flag is only useful when an element appears the same way 100% of the time.

e.g.
```
p {
   color: blue !important;
}

.main p {
   color: red;
}
```

➔ Since !important is used on the p selector's color attribute, all p elements will appear blue, even though there is a more specific .main p selector that sets the color attribute to red.

Multiple Selectors
If two or more elements in HTML share the same style attributes, then we can write them all in one code in CSS

```
h1,
.menu {
   font-family: Georgia;
}
```

INSTEAD OF

```
h1 {
   font-family: Georgia;
}

.menu {
   font-family: Georgia;
}
```

CSS Visual Rules

Font Family
Default font is "Times New Roman"
List of fonts: https://www.cssfontstack.com/

```
h1 {
   font-family: "Courier New";
}
```

Font Size
```
p {
   font-size: 18px;
}
```

 # Font Weight
Bold or normal
```
p {
```

```
    font-weight: bold;
}
```

Text Align
Left, center, right

```
h1 {
    text-align: right;
}
```

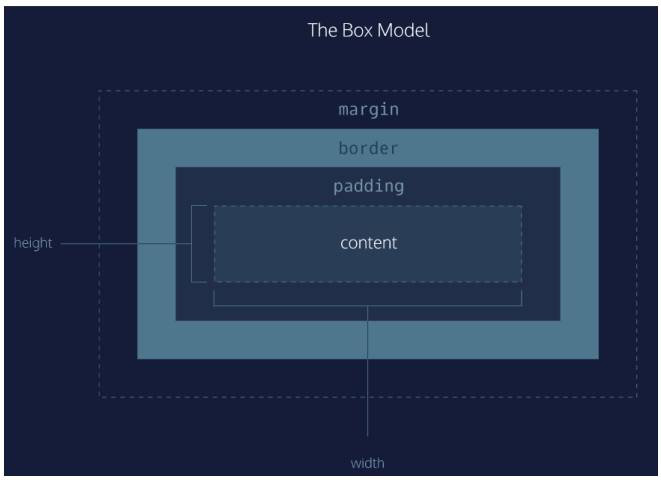
Color
color: this property styles an element's foreground color
background-color: this property styles an element's background color

```
h1 {
    color: red;
    background-color: blue;
}
```

Opacity
```
.overlay {
    opacity: 0.5;
}
```

Image Attributes
```
.image {
    background-image:
    url("https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/soccer.jpeg");
    background-size: cover;
    background-position: center;
    height: 300px;
}
```

Box Model



Height and Width
```
p {
    height: 80px;
    width: 240px;
}
```

Borders

```
p {
   border: 3px solid coral;
}
```

The default border is medium none color, where color is the current color of the element.

width — The thickness of the border. A border's thickness can be set in pixels or with one of the following keywords: thin, medium, or thick.
style — The design of the border. Web browsers can render any of 10 different styles. Some of these styles include: none, dotted, and solid.
color — The color of the border. Web browsers can render colors using a few different formats, including 140 built-in color keywords.

You can modify the corners of an element's border box with the **border-radius** property.

```
div.container {
   border: 3px solid rgb(22, 77, 100);
   border-radius: 5px;
}
```

You can create a border that is a perfect circle by setting the radius equal to the height of the box, or to 100%.

Padding
The space between the contents of a box and the borders of a box is known as padding.

```
p.content-header {
   border: 3px solid coral;
   padding: 10px;
}
```

padding-top / padding-right / padding-bottom / padding-left

```
p.content-header {
   border: 3px solid fuschia;
   padding-bottom: 10px;
}
```

top (6 pixels), right (11 pixels), bottom (4 pixels), and left (9 pixels) sides of the content

```
p.content-header {
   border: 3px solid grey;
   padding: 6px 11px 4px 9px;
}
```

Shortcut if the top and bottom values for padding will equal each other, and the left and right values for padding will also equal each other

```
p.content-header {
   padding: 5px 10px; (first value is top and bottom and second value is for left and right)
}
```

Margins

```
p {
   border: 1px solid aquamarine;
   margin: 20px;
}
```

Margin-top / margin-right / margin-bottom/ margin-left

```
p {
   margin: 6px 10px 5px 12px;
}
```

Shortcut if the top and bottom values for margin will equal each other, and the left and right values for padding will also equal each other

```
p {
   margin: 6px 12px;
}
```

How to center content

The 0 sets the top and bottom margins to 0 pixels. The auto value instructs the browser to adjust the left and right margins until the element is centered within its containing element. In order to center an element, a width must be set for that element.

```
div.headline {
   width: 400px;
   margin: 0 auto;
}
```

min and max width

```
p {
   min-width: 300px;
   max-width: 600px;
}
```

Overflow

The overflow property controls what happens to content that spills, or overflows, outside its box. It can be set to one of the following values:

hidden - when set to this value, any content that overflows will be hidden from view.
scroll - when set to this value, a scrollbar will be added to the element's box so that the rest of the content can be viewed by scrolling.
visible - when set to this value, the overflow content will be displayed outside of the containing element. Note, this is the default value.

```
p {
   overflow: scroll;
}
```

Default settings

```
* {
   margin: 0;
   padding: 0;
}
```

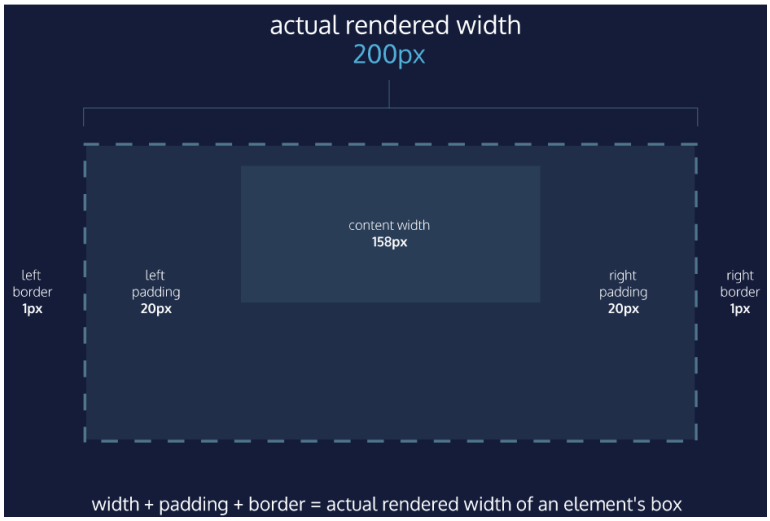
Visibility

```
.future {
   visibility: hidden;
}
```

What's the difference between display: none and visibility: hidden? An element with display: none will be completely removed from the web page. An element with visibility: hidden, however, will not be visible on the web page, but the space reserved for it will.

Changing the Box Model

Box Model: Content Box

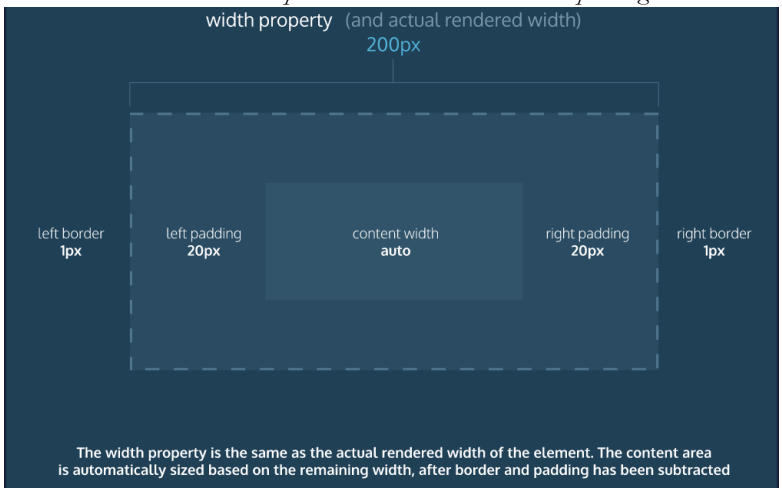


Border Box

```
* {
   box-sizing: border-box;
}

h1 {
   border: 1px solid black;
   height: 200px;
   width: 300px;
   padding: 10px;
}
```

Resets the box model to border-box for all HTML elements. In the example above, the height of the box would remain at 200 pixels and the width would remain at 300 pixels. The border thickness and padding would remain entirely inside of the box.



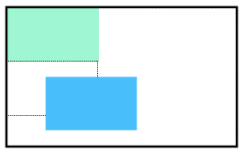
CSS Display and Positioning

five properties for adjusting the position of HTML elements in the browser:
- position
- display
- z-index
- float
- clear

Position - Relative

Static: default
Relative: This value allows you to position an element relative to its default static position on the web page.

```
.box-bottom {
   background-color: DeepSkyBlue;
   position: relative;
   top: 20px;
   left: 50px;
}
```



Position - Absolute
When an element's position is set to absolute, as in the last exercise, the element will scroll with the rest of the document when a user scrolls.

```
.box-bottom {
   background-color: DeepSkyBlue;
   position: absolute;
   top: 20px;
   left: 50px;
}
```

Position - Fixed
We can fix an element to a specific position on the page (regardless of user scrolling) by setting its position to fixed.

```
.box-bottom {
   background-color: DeepSkyBlue;
   position: fixed;
   top: 20px;
   left: 50px;
}
```

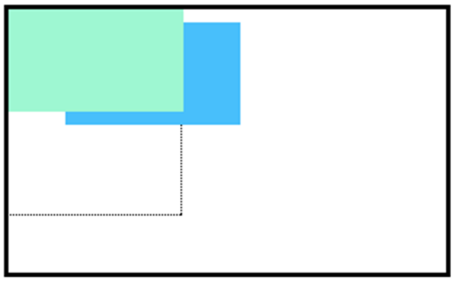
Z-Index
The z-index property controls how far "back" or how far "forward" an element should appear on the web page when elements overlap. The z-index property accepts integer values. Depending on their values, the integers instruct the browser on the order in which elements should be displayed on the web page.

```
.box-top {
   background-color: Aquamarine;
   position: relative;
   z-index: 2;
}

.box-bottom {
   background-color: DeepSkyBlue;
   position: absolute;
```

```
    top: 20px;
    left: 50px;
    z-index: 1;
}
```

➔ In the example above, we set the .box-top position to relative and the z-index to 2. We changed position to relative, because the z-index property does not work on static elements. The z-index of 2 moves the .box-top element forward, because it is greater than the .box-bottom z-index, 1.



Inline Display

The default display for some tags, such as ****, ****, and **<a>**, is called inline. Inline elements have a box that wraps tightly around their content, only taking up the amount of space necessary to display their content and not requiring a new line after each element. The height and width of these elements cannot be specified in the CSS document.

The CSS display property provides the ability to make any element an inline element. This includes elements that are not inline by default such as paragraphs, divs, and headings.

Block Display

Some elements are not displayed in the same line as the content around them. These are called block-level elements. These elements fill the entire width of the page by default, but their width property can also be set. Elements that are block-level by default include all levels of heading elements (**<h1>** through **<h6>**), **<p>**, ** ** and **<footer>**.

```
strong {
    display: block;
}
```

Inline-Block Display

Inline-block elements can appear next to each other and we can specify their dimensions using the width and height properties. Images are the best example of default inline-block elements.

```
<div class="rectangle">
    <p>I'm a rectangle!</p>
</div>
<div class="rectangle">
    <p>So am I!</p>
</div>
<div class="rectangle">
    <p>Me three!</p>
</div>
```

```
.rectangle {
    display: inline-block;
    width: 200px;
    height: 300px;
}
```

➔ The .rectangle s will all appear inline (provided there is enough space from left to right) with a width of 200

pixels and height of 300 pixels, even though the text inside of them may not require 200 pixels by 300 pixels of space.

Float

left - this value will move, or float, elements as far left as possible.
right - this value will move elements as far right as possible.

```
.boxes {
   width: 120px;
   height: 70px;
}

.box-bottom {
   background-color: DeepSkyBlue;
   float: right;
}
```

Clear

The clear property specifies how elements should behave when they bump into each other on the page. It can take on one of the following values

left — the left side of the element will not touch any other element within the same containing element.
right — the right side of the element will not touch any other element within the same containing element.
both — neither side of the element will touch any other element within the same containing element.
none — the element can touch either side.

```
div {
   width: 200px;
   float: left;
}

div.special {
   clear: left;
}
```

➔ The element with class special did not move all the way to the left because a taller blocked its positioning. By setting its clear property to left, the special will be moved all the way to the left side of the page.