

# Introduction to CSS

---

Making things pretty

# CSS or Counterfeit Stainless Steel

---

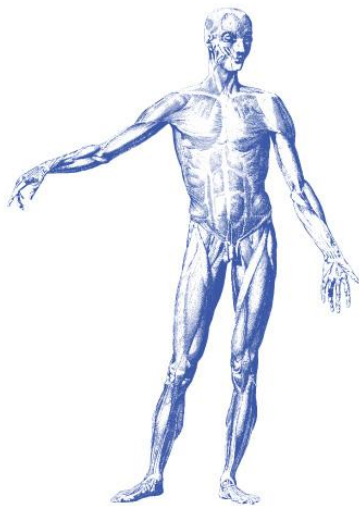
*\*actually it's Cascading StyleSheets*

Structure



HTML

Function



Javascript

Style



CSS

# CSS

---

- We use CSS to change the look of our content.
- CSS is used to change page layout, colors, fonts, text-sizes, image size, etc...
- You can do almost anything *presentational* with CSS.
- You are limited only by your knowledge of CSS.

# CSS - Linking CSS to the HTML

---

## Style directly in the HTML file

```
<!DOCTYPE html>

<html>

  <head>
    <title>A Basic HTML Template</title>
    <style>
      .custom-class {
        color: red;
      }
      /* ... */
    </style>
  </head>
```

## Link CSS in the HTML file

```
<!DOCTYPE html>

<html>

  <head>
    <title>A Basic HTML Template</title>
    <link href="styles.css" rel="stylesheet">
  </head>
```

The `<link>` tag is used to import the CSS file containing rules applying the styles to the DOM elements.

# CSS - Example 1 (using HTML tag)

---

## HTML

```
<h1>My Amazing Blog</h1>
```

## CSS

```
h1 {  
  font-size: 32px;  
}
```

You can target an HTML element directly in CSS by using the tag name.

# CSS - Example 2 (using a class)

---

## HTML

```
<h1 class="blog-title">My Amazing Blog</h1>
```

## CSS

```
.blog-title {  
  font-size: 32px;  
}
```

Assign the element an attribute of “class” and provide it with a value. Use that value to target it in the CSS.

# CSS - Example 3 (using an id)

---

## HTML

```
<h1 id="blog-title">My Amazing Blog</h1>
```

## CSS

```
#blog-title {  
    font-size: 32px;  
}  
  
/* DON'T DO THIS! USE A CLASS INSTEAD */
```

Assign the element an attribute of “id” and provide it with a value. Use that value to target it in the CSS.

# Limitation of ids

---

- A document can only have one copy of an id.
- An element can only have one id.




```
<ul>
  <!-- Oh no! Two elements share the same id! -->
  <li id="thing">thing 1</li>
  <li id="thing">thing 1</li>
  <li id="thing special">thing 3</li>
</ul>

<!-- Oh no! Elements share the same id! -->
<!-- Oh no! An element cannot have two ids! -->
```



## Which selector should I use?

---

Type	Example	Comment
class	.my-class	
tag	h1	
id	#bacon	

**Don't  
use  
ids!**

**Really.**

# Selectors can be combined

---

```
<ul>
  <li>Bingo was his name.</li>
  <li class="secret">I have super-powers!</li>
  <li class="loud">
    I eat <span class="bold">bacon</span>
  </li>
</ul>
```



```
li {
  color: green;
}
.secret {
  color: blue;
}
li.loud {
  text-transform: uppercase;
}
.loud span {
  font-weight: 700;
}
```

# Specificity

---

```
<p class="blue">What color am I?</p>
```

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



# Specificity

---

Different selectors have different *strengths*.

- A class beats a tag.
- A combined class+tag beats a class
- An id beats just about everything.

# Please! 🙏

---

When styling HTML, we should...

- **Always** use classes.
- **Sometimes** use tags.
- **Never** use ids.
- **Never, ever** use ids.



# [1-2]

---

CSS Properties

# CSS Properties

---

**CSS properties are used to indicate which styles to apply.**

```
.mybox {  
  color: #000;  
  border: 1px solid red;  
  background-color: red;  
}
```

- There are *many* CSS properties. Tons.
- Use this CSS properties reference: [CSS Reference \(Mozilla\)](#)
- VS Code can also autocomplete various properties.

# CSS Properties

---

Google is your friend. 😎



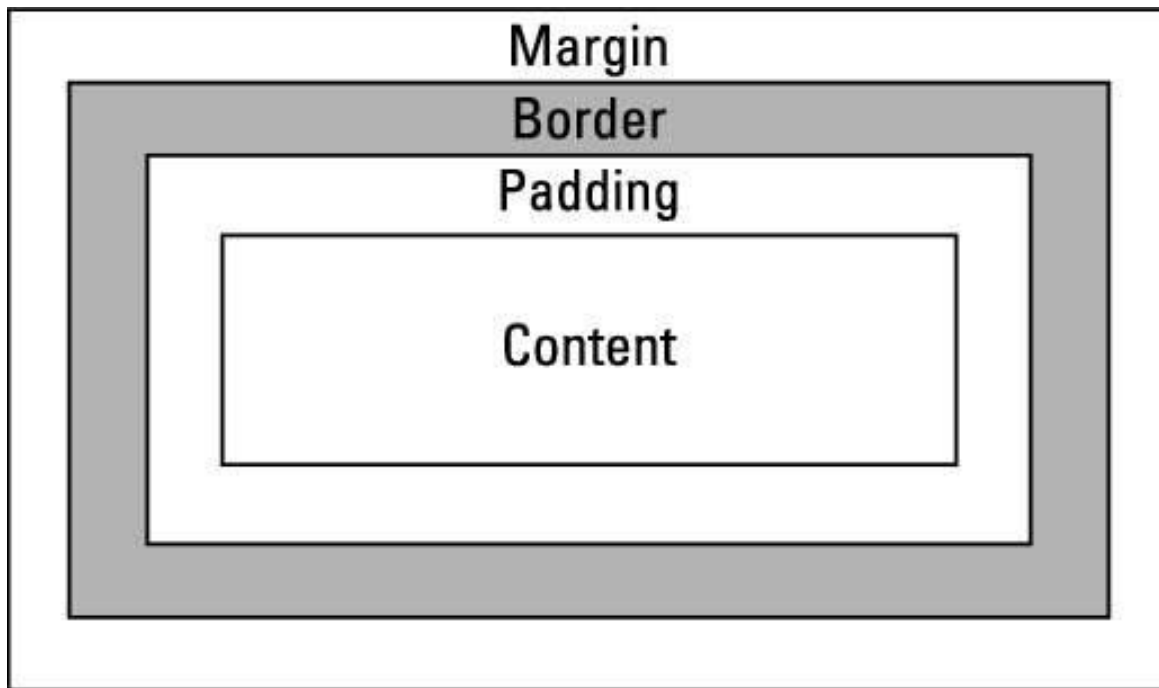
- The human brain cannot hold the full library of CSS properties and values.
- The web has many very good resources on CSS.



# CSS: The box model

---

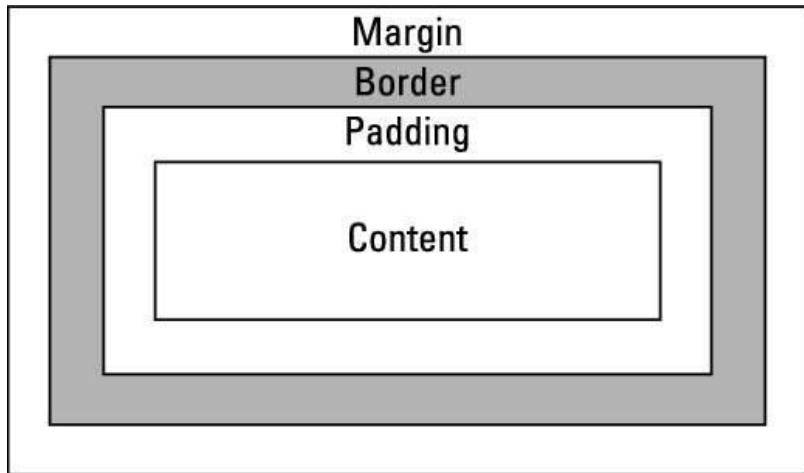
Every HTML element is a box that has four properties.



# CSS: The box model

---

Each side can be set independently or all at once.



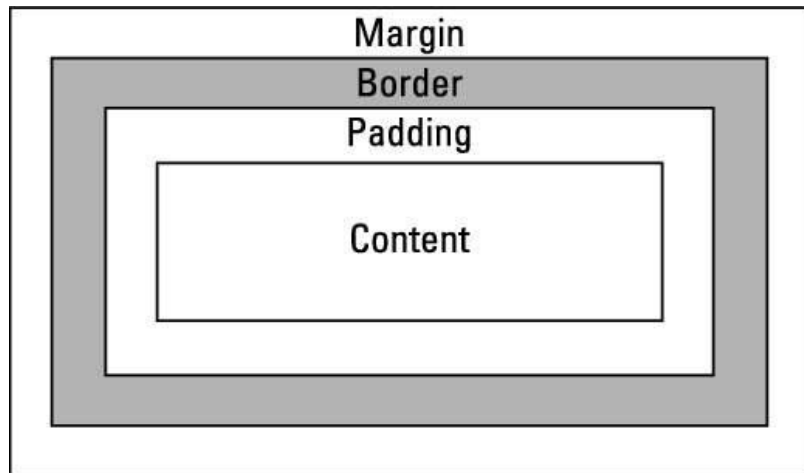
```
<p class="model">Bacon!</p>
```





```
.model {  
  margin-top: 24px;  
  margin-right: 0;  
  margin-bottom: 5px;  
  margin-left: 10px;  
}  
  
/* or */  
.model {  
  margin: 24px 0 5px 10px;  
}  
  
/* TOP RIGHT BOTTOM LEFT */
```

# CSS: The box model

---

The shorthand version is usually the best option.



example				
<code>margin: 24px 0 5px 10px;</code>	24px	0px	5px	10px
<code>margin: 24px 0 5px;</code>	24px	0px	5px	0px
<code>margin: 24px 0;</code>	24px	0px	24px	0px
<code>margin: 24px;</code>	24px	24px	24px	24px

# CSS: The box model - Gotcha!

---

What are the dimensions of the greeting?

```
<p class="greeting">Howdy!</p>
```

We can fix this with the box-sizing property.

```
.greeting {  
  background: gold;  
  border: 10px solid black;  
  height: 100px;  
  margin: 50px;  
  width: 100px;  
}  
  
* {  
  box-sizing: border-box;  
}
```



# CSS - Positioning



Value	Description
static	
fixed	
relative	
absolute	

When using the ``absolute`` property, you will need to set the parent's position as well, anything but ``static``.

# CSS: z-index

---



```
.item {  
  z-index: 1; /* numerical value */  
}
```

- The z-index property specifies the stack order of an element.
- An element with greater stack order is always in front of an element with a lower stack order.
- It only works on *positioned* elements (absolute, relative, fixed, or sticky).

# CSS - Display



```
.item {  
  display: inline-block;  
}
```

Value	Description
block	
inline	
inline-block	
none	
flex	

## CSS - Display flex (basics)

---

Flexbox is a wonderful way of positioning elements on a webpage.

It does have a bit of a learning curve, but well worth it!

Giving the parent container the property “display: flex”, automagically sets all of the items inside the container to be side-by-side.

```
.item {  
  display: flex;  
  flex-direction: row | row-reverse | column | column-reverse;  
  flex-wrap: nowrap | wrap | wrap-reverse;  
  justify-content: flex-start | flex-end | center | space-around | space-between;  
  align-items: stretch | baseline | center | flex-start | flex-end;  
  align-content: stretch | center | flex-start | flex-end | space-around | space-between;  
}
```

<https://flexbox.help/>



# CSS - Display flex (child properties)

---

 [A Complete Guide to Flexbox](#) 

# CSS

---

