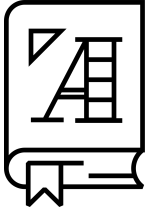# Introduction to CSS Styling
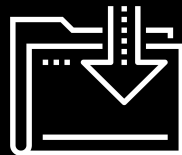
HTML/CSS Definitions

HTML: Hypertext Markup Language (Content)

CSS: Cascading Style Sheets (Appearance)

HTML/CSS are the "languages of the web." Together they define both the content and the aesthetics of a webpage, handling everything from the layouts, colors, fonts, and content placement. (JavaScript is the language that deals with logic, animation, etc.)

HTML/CSS Analogy

HTML Alone	HTML and CSS
Like writing papers in Notepad.	Like writing papers in Microsoft Word.
Used to write unformatted text (i.e, content only).	Used both to write the content *and* format it (color, font, alignment, layout, etc.).

Basic HTML Page

```html
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>My First Website!</title>
6   </head>
7   <body>
8
9       <h1>Awesome Header</h1>
10      <h2>Smaller Awesome Header</h2>
11      <h3>Even Smaller Header</h3>
12
13      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quidem consequatur
        unde aut dolores odio hic, accusamus recusandae ipsam illum enim voluptatibus
        obcaecati totam tempora eum quod sapiente. Corporis, quidem, culpa?</p>
14      <img src="https://pbs.twimg.com/media/BsgYfMQCQAAWVKH.jpg" alt="Awesome" width="
        25%">
15
16      <h3>Menu Links</h3>
17      <ul>
18          <li><a href="http://www.google.com"></a>Google</li>
19          <li><a href="http://www.facebook.com"></a>Facebook</li>
20          <li><a href="http://www.twitter.com"></a>Twitter</li>
21      </ul>
22
23  </body>
24  </html>
```

Basic HTML Page (No CSS)

Awesome Header

Smaller Awesome Header

Even Smaller Awesome Header

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Menu Links
- **Google**
- **Facebook**
- **Twitter**

Basic HTML Page: Result

Awesome Header

Smaller Awesome Header

Even Smaller Awesome Header

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Menu Links

- **Google**
- **Facebook**
- **Twitter**

Boring

Enter CSS

```
26 ▼ <style>
27 ▼     h1 {
28           font-size: 60px;
29           text-align: center;
30           margin-bottom: 15px;
31           text-decoration: underline;
32           background-color: black;
33           color: white;
34       }
35
36 ▼     h2 {
37           font-size: 40px;
38           text-align: center;
39           margin-top: 15px;
40           margin-bottom: 15px;
41       }
42
43 ▼     h3 {
44           font-size: 20px;
45           text-align: center;
46           margin-top: 15px;
47       }
48
49 ▼     img {
50           display: block;
51           margin-left: auto;
52           margin-right: auto;
53       }
54
55 ▼     p {
56           text-align: center;
57           font-size: 20px;
58           font-weight: bold;
59       }
60
61 ▼     ul {
62           text-align: center;
63           font-size: 35px;
64           list-style-position: inside;
65           border-style: solid;
66           border-width: 5px;
67       }
68 </style>
```
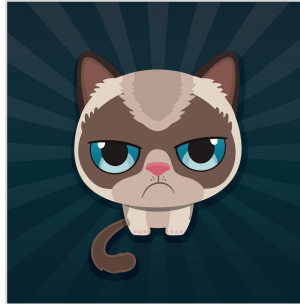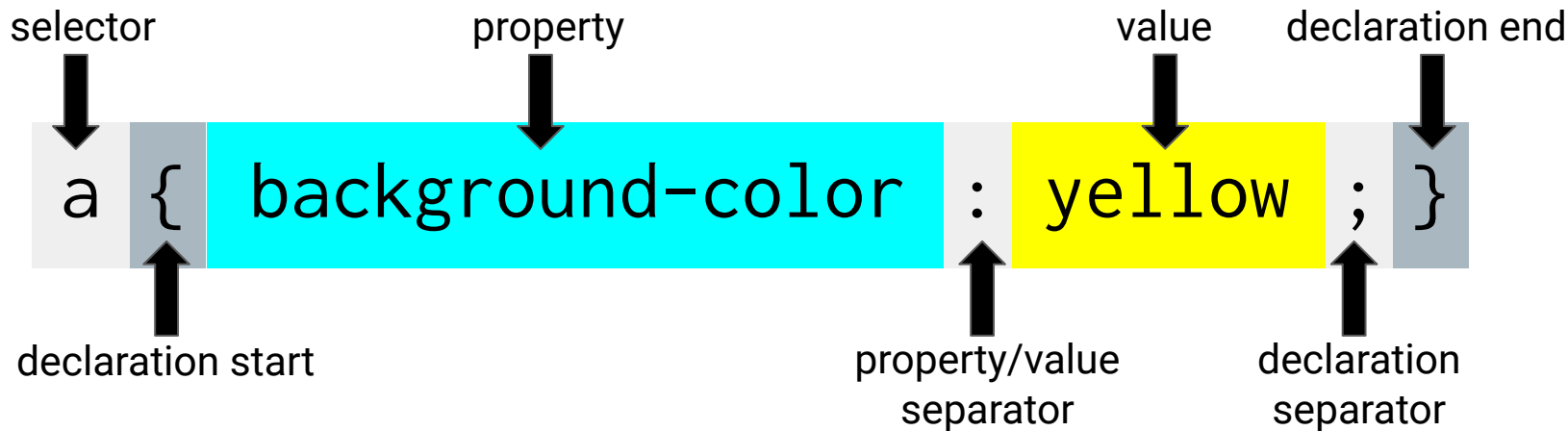
Enter CSS: Result

CSS Syntax

CSS works by hooking onto **selectors** added into HTML using **classes** and **identifiers**.

Once hooked, we apply **styles** to those HTML elements using CSS.

selector   property   value   declaration end

```
a { background-color : yellow ; }
```

declaration start   property/value separator   declaration separator

Key CSS Attributes

Font and Color:	
color:	sets color of text
font-size:	sets size of the font
font-style:	sets italics
font-weight:	sets bold
header	for headers
nav	for navigation bars
footer	for footers
Alignment and Spacing:	
padding (top/right/bottom/left):	adds space between element and its own border
margin (top/right/bottom/left):	adds space between element and surrounding elements
float:	forces elements to the sides, centers, or tops
Background:	
background-color:	sets background color
background-image:	sets background image

Instructor Demonstration
CSS Basics

CSS Example

In the following example, the header would become blue and much larger because of the CSS.

We can incorporate an element's class or ID to apply a CSS style to a particular part of the document. Just remember to include the necessary symbol before the CSS: "." for class, "#" for ID.

Example (HTML)	Example (CSS)

`<p class="bigBlue">Header</p>`	`.bigBlue`
	`{`
	` font-size: 100px;`
	` color: blue;`
	`}`

Box Model

Boxes Upon Boxes

In CSS, every element rests within a series of boxes.

Each box has customizable space properties: margin, border, and padding

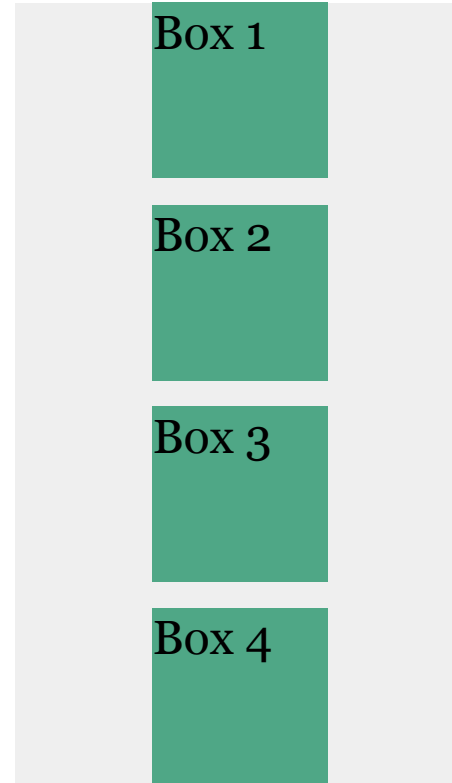**Typical spacing value:** 20px 10px 10px 20px (top, right, bottom, left)

Margin

Border

Padding

Content
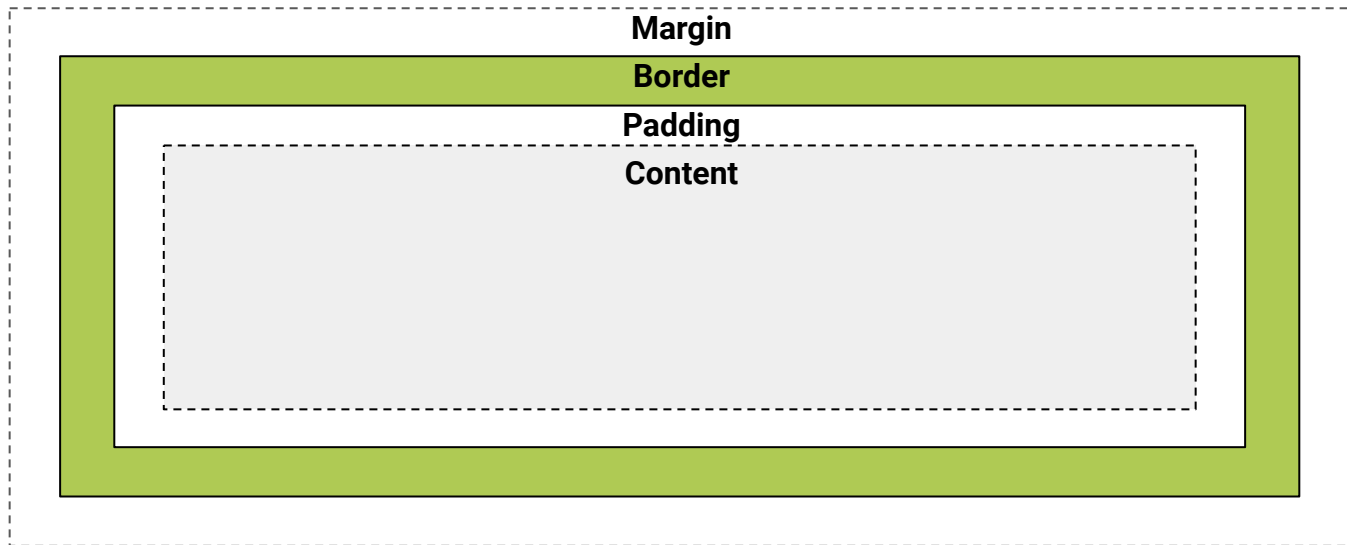
CSS Positioning

Position: Static (Default)

Four boxes placed statically (default):

Box 1

Box 2

Box 3

Box 4

Position: Relative

Switching the boxes to relative will nudge the boxes in relation to their "original" location:

```css
.box {
  background: #2db34a;
  height: 80px;
  position: relative;
  width: 80px;
}
.box-1 {
  top: 20px;
}
.box-2 {
  left: 40px;
}
.box-3 {
  bottom: -10px;
  right: 20px;
}
```
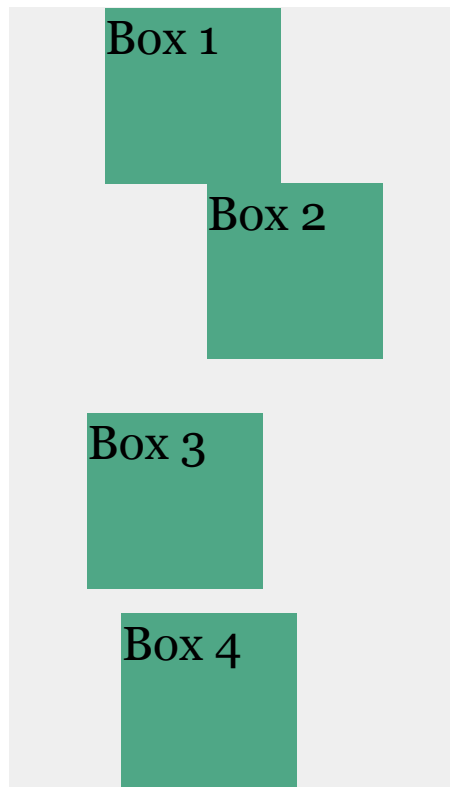
Position: Absolute

Positioned relative to nearest positioned ancestor:

```css
.box-set {
  height: 400px;
  background: darkgray;
  position: relative;
}
.box {
  position: absolute;
  height: 150px;
  width: 150px;
  background: #2db34a;
  border: 2px solid black;
}
.box-1 {
  top: 6%;
  left: 2%;
}
.box-2 {
  top: 0;
  right: -40px;
}
.box-3 {
  bottom: -10px;
  right: 20px;
}
.box-4 {
  bottom: 0;
}
```
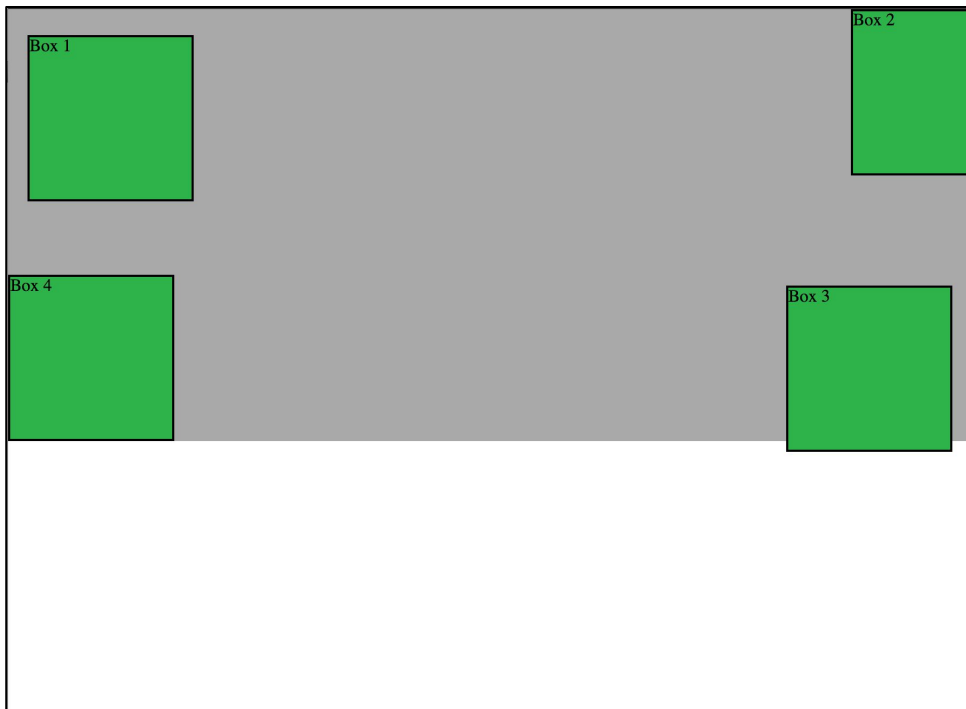
Position: Fixed

Position with exact coordinates in the browser window:

```css
.box-set {
  height: 400px;
  background: darkgray;
}
.box {
  position: fixed;
  height: 150px;
  width: 150px;
  background: #2db34a;
  border: 2px solid black;
}
.box-1 {
  top: 6%;
  left: 2%;
}
.box-2 {
  top: 0;
  right: -40px;
}
.box-3 {
  bottom: -10px;
  right: 20px;
}
.box-4 {
  bottom: 0;
}
```

Box 1

Box 2

Box 4

Box 3
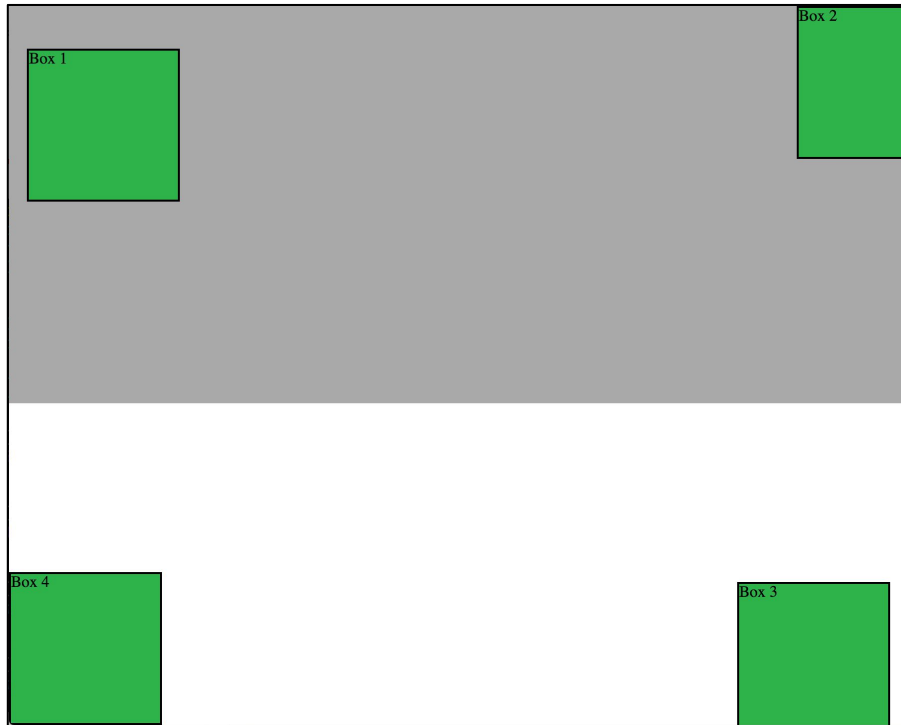
Layering with Z-Index

The z-index property allows you to layer elements on top of each other.
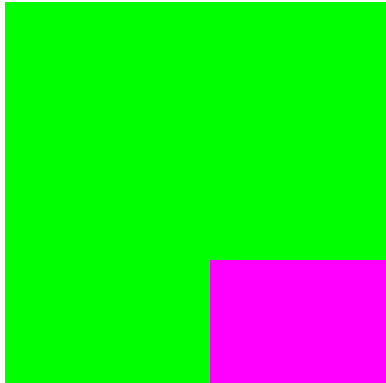
```
position: absolute;
z-index:1;
```

```
position: absolute;
z-index:2;
```
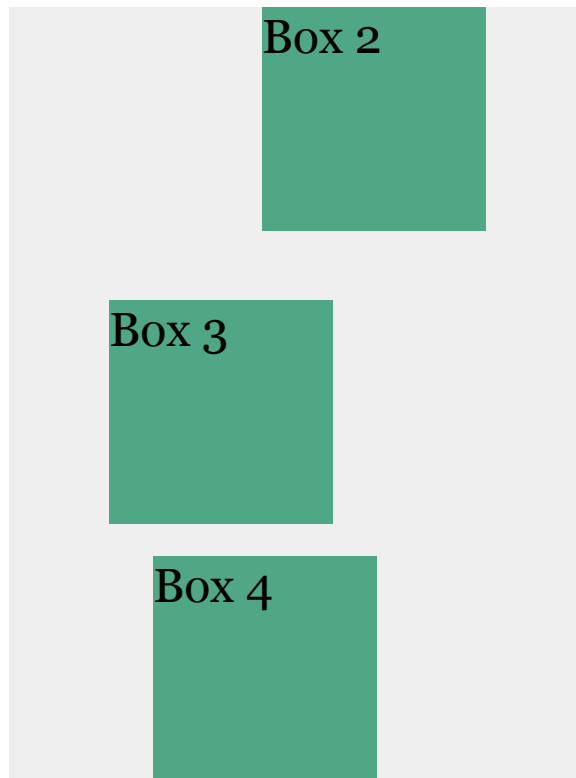
Hiding Things

Display: none allows you to hide elements from view.

This will become useful in later sections, when we'll hide and reveal specific HTML elements of our choosing.

```
.box-1 {
  display: none;
}
```
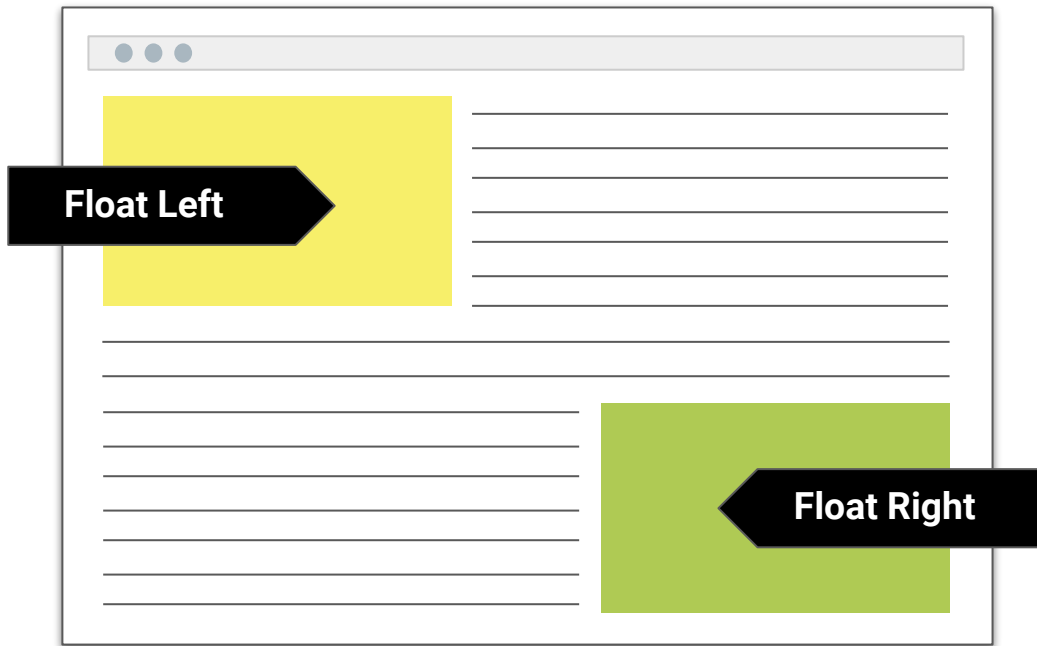
Box 2

Box 3

Box 4

Flow and Float

The Concept of Flow

By default, every HTML element displayed in the browser is governed by a concept called **flow**.

This means that HTML elements force their adjacent elements to **flow around them**.

Flow Analogy to MS Word

This concept of "flow" is very similar to the **wrap-text options** you may be familiar with in Microsoft Word.

Just as in MS Word, you can have images in-line with text, on-top of text, etc.
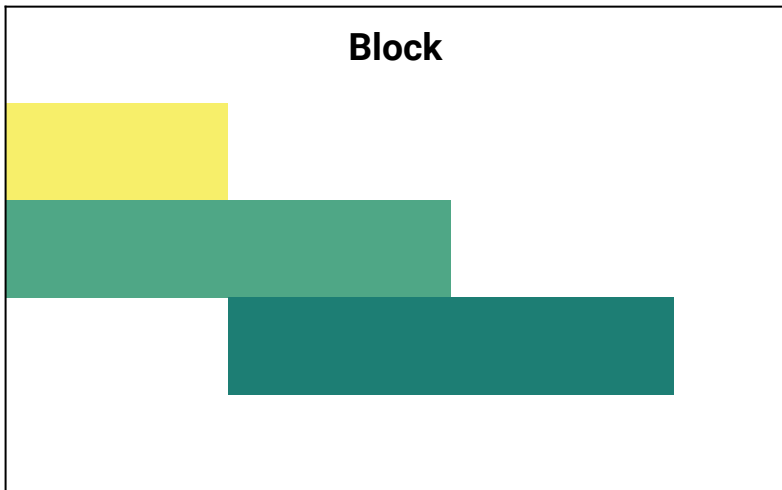
Block Elements

By default, web clients render many HTML elements as **block elements**. Paragraphs, headers, divs, and more receive this treatment.

A block element will take up an entire line of space—unless you intervene with CSS properties.

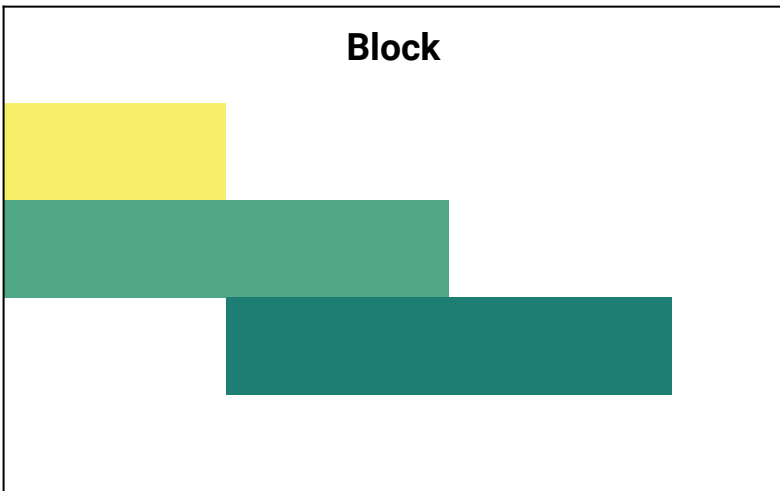**Block**

Block Elements vs. In-Line Elements

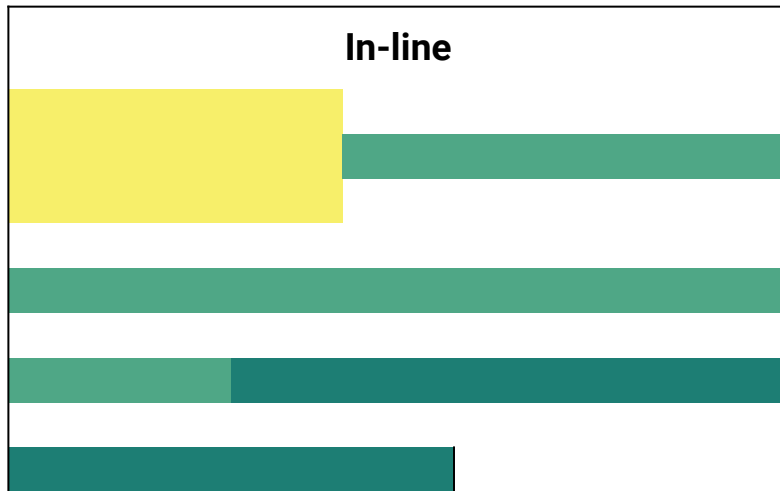Now, contrast block elements with **in-line elements**.

By using **float CSS properties**, we can command our website to display multiple HTML elements adjacently.
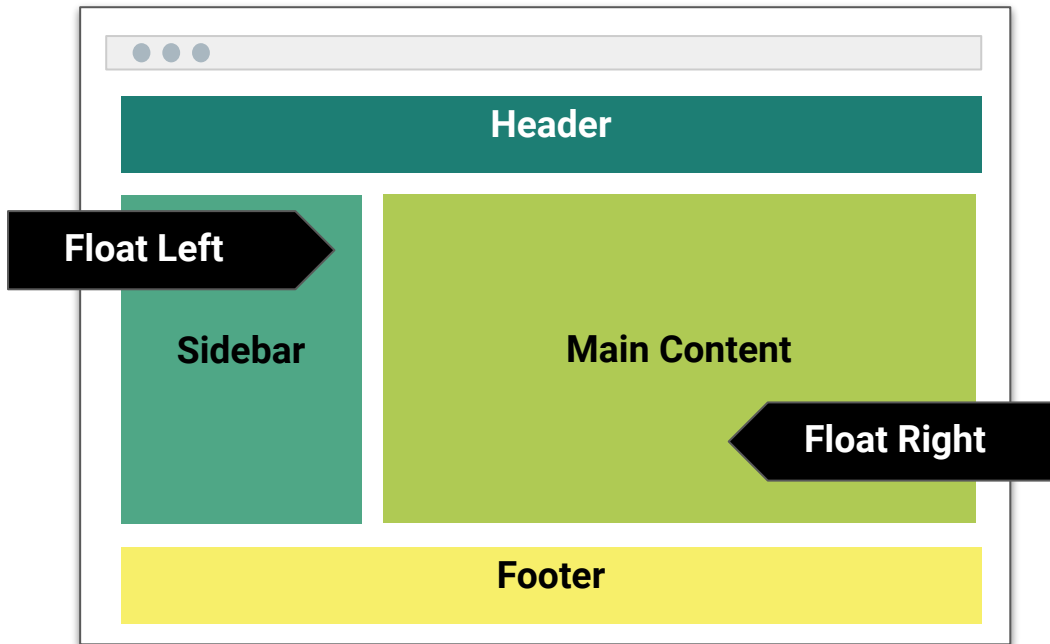
Block

In-line

Floats

To transform these block elements into in-line elements, we use a CSS property called **float**. Floats are necessary when building web layouts.
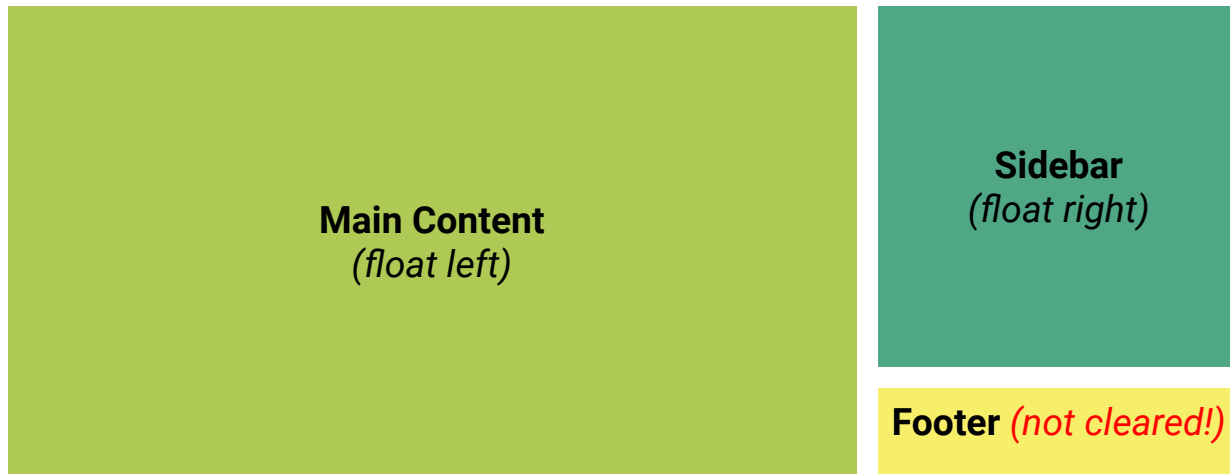

CSS

```
#sidebar {
    float: left;
}
#main-content {
    float: right;
}
```

Clearing the Float

However, floats often get in the way of layouts. Sometimes we don't want to give each element the "in-line" treatment.

Main Content
(float left)

Sidebar
(float right)

Footer *(not cleared!)*

Clearfix Hack

Sometimes when elements don't match up in size, we get situations like this:

`<div>`

Uh-oh! The image is taller than the element containing it, and it's floated, so it's overflowing outside of its container!
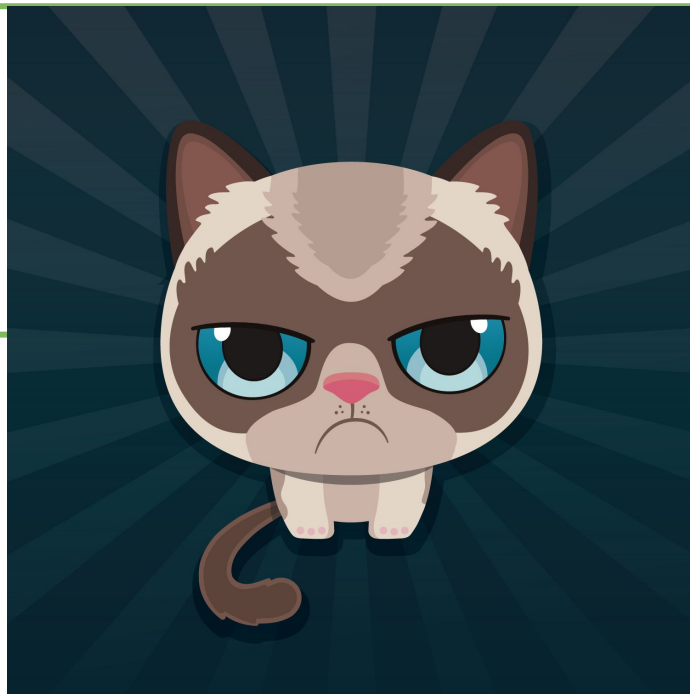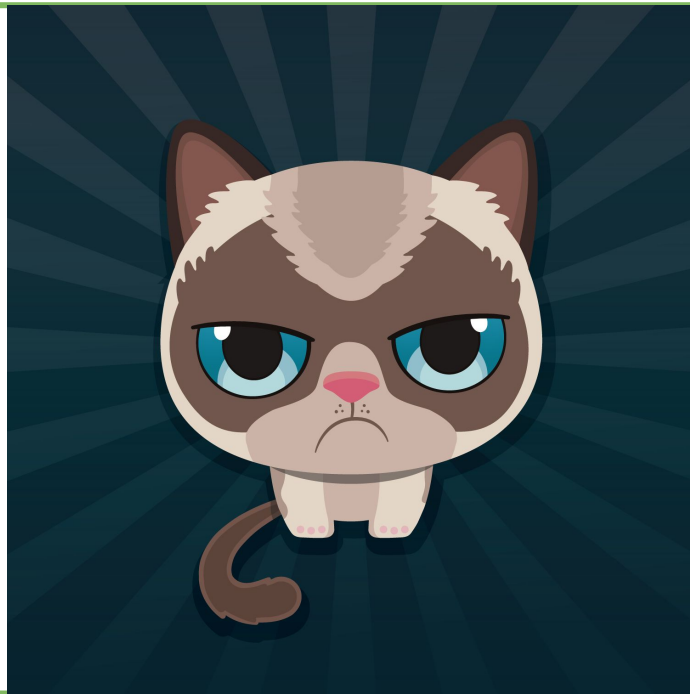
Clearfix Hack

We can get around this by using the **clearfix hack**.

```
<div class="clearfix">
```

Much better!

Clearfix Hack

`::after` is what we call a pseudo-element. We use it to style specific parts of an element.

This will add an HTML element, hidden from view, after the content of the `.clearfix` element. This clears the float.

```css
.clearfix::after {
  content: "";
  display: block;
  clear: both;
}
```