CSE-465

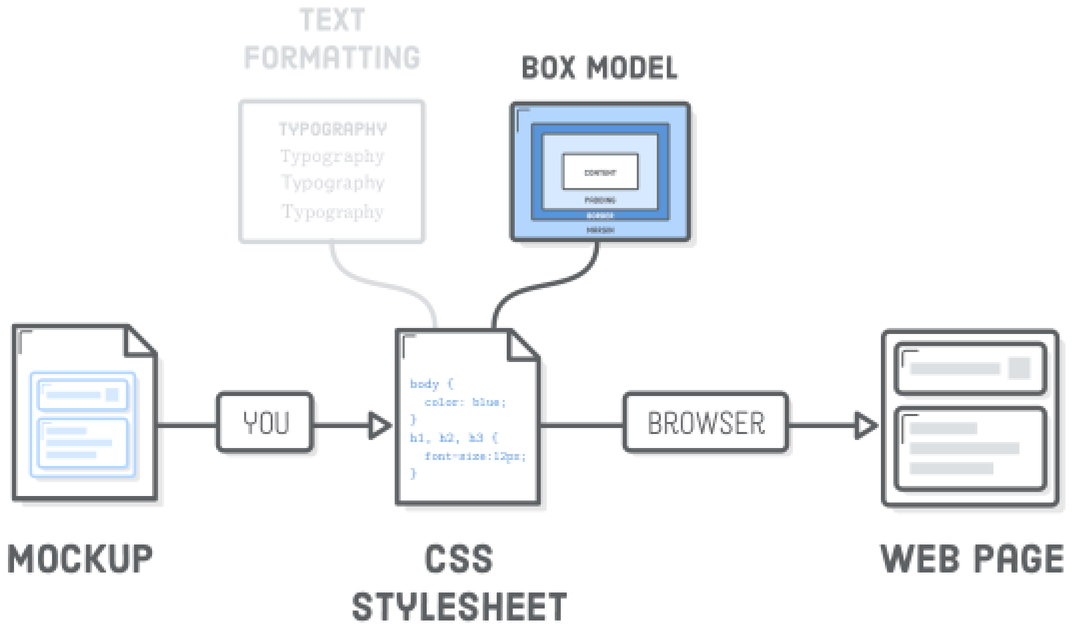
Web Programming

CSS Box Models

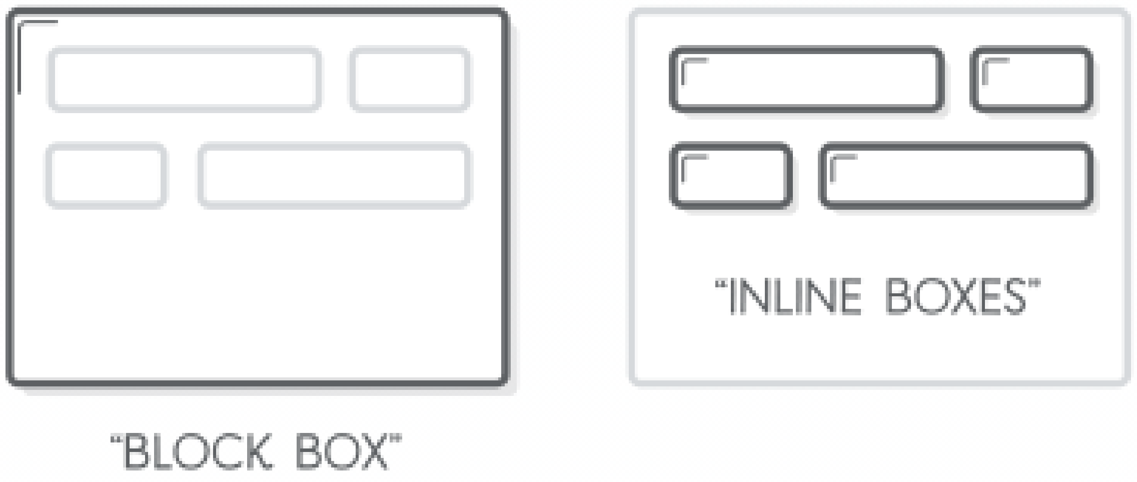
Reference:
https://www.internetingishard.com/

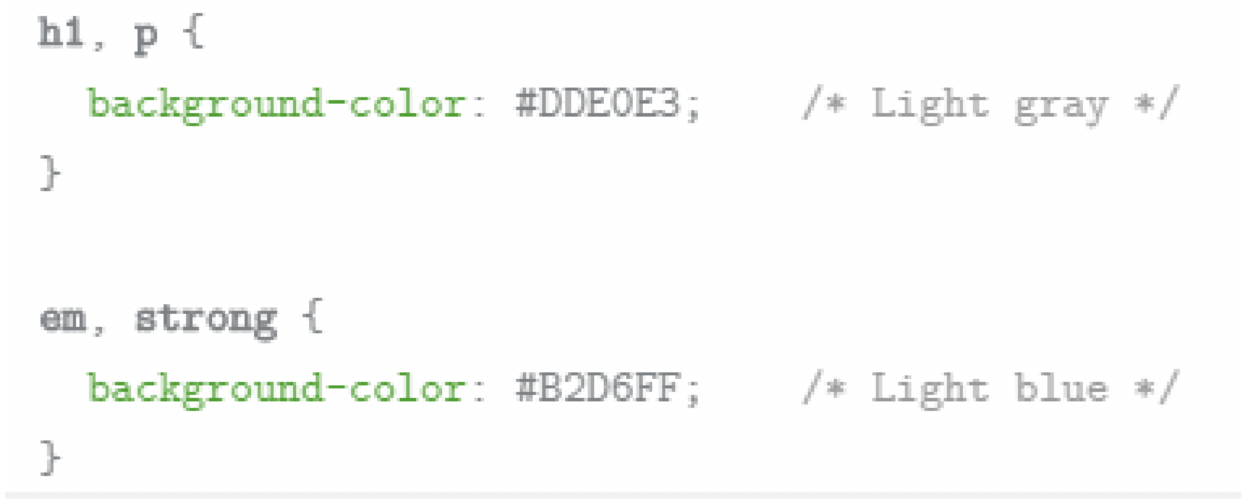
Md. Saidul Hoque Anik
anik@cse.uiu.ac.bd

The Box Model



The Box Elements





```
h1, p {
  background-color: #DDE0E3;     /* Light gray */
}

em, strong {
  background-color: #B2D6FF;     /* Light blue */
}
```

Key Points

1. Block boxes always appear below the previous block element.

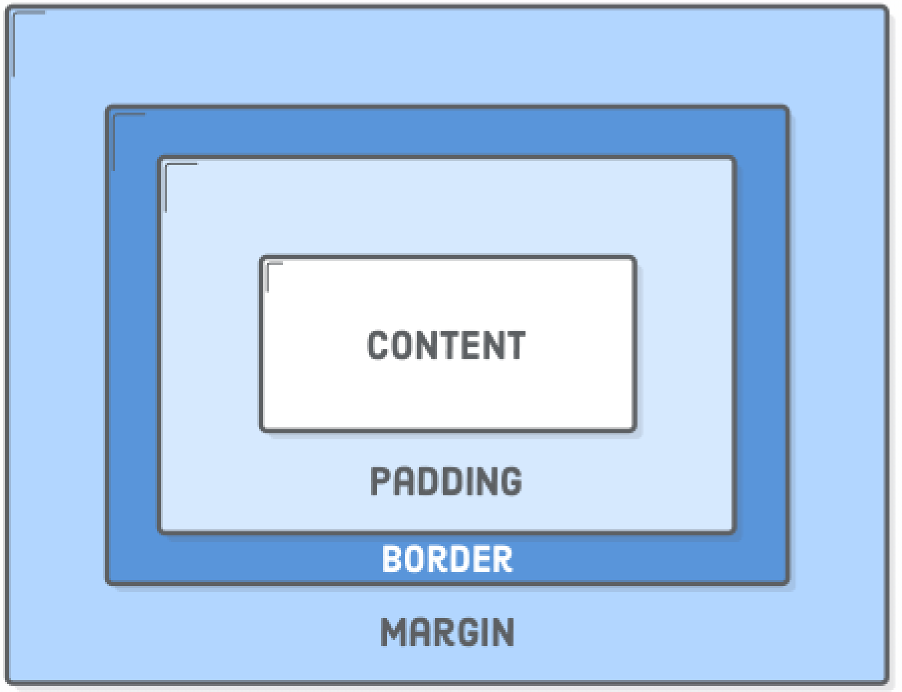
2. The width of block boxes is set automatically based on the width of its parent container. In this case, our blocks are always the width of the browser window.

3. The default height of block boxes is based on the content it contains. When you narrow the browser window, the <h1> gets split over two lines, and its height adjusts accordingly.

4. Inline boxes don't affect vertical spacing. They're not for determining layout—they're for styling stuff inside of a block.

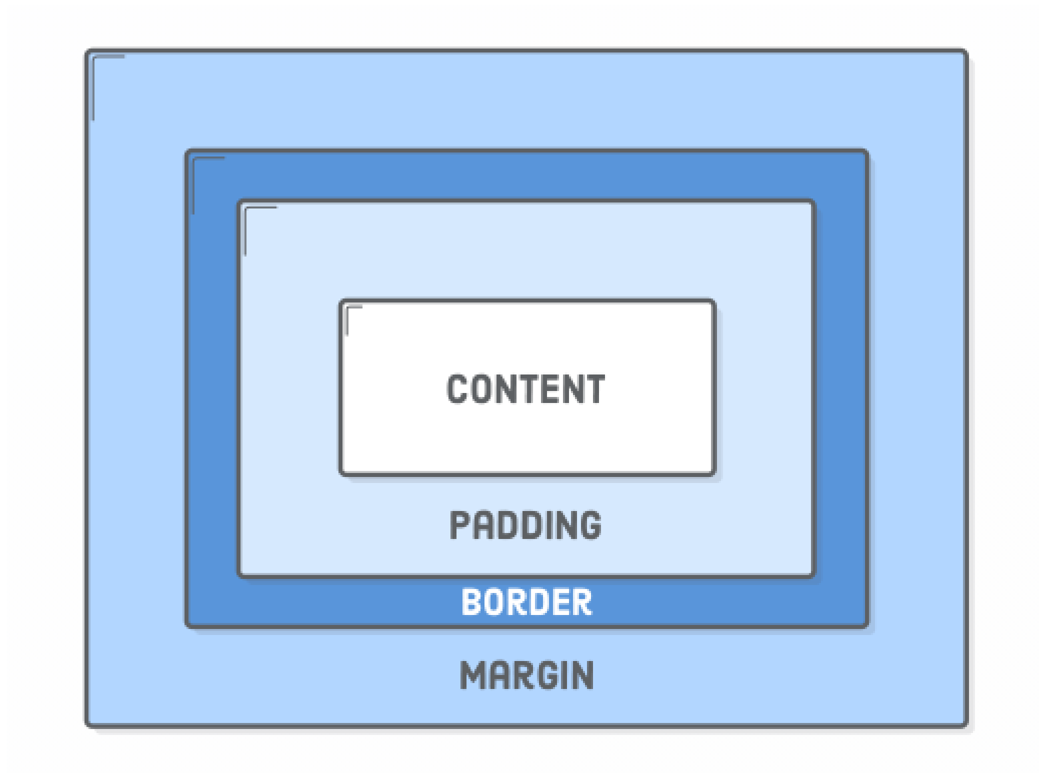
5. The width of inline boxes is based on the content it contains, not the width of the parent element.

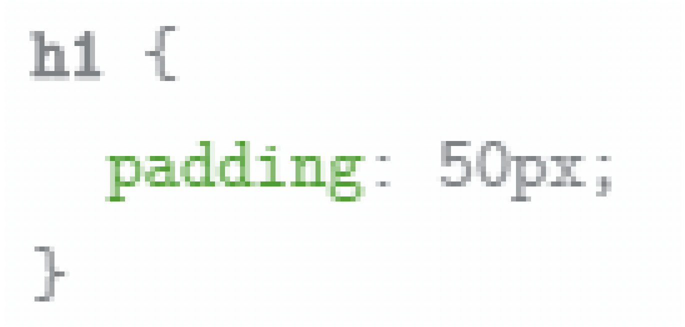
Box Model



- **Content** – The text, image, or other media content in the element.
- **Padding** – The space between the box’s content and its border.
- **Border** – The line between the box’s padding and margin.
- **Margin** – The space between the box and surrounding boxes.

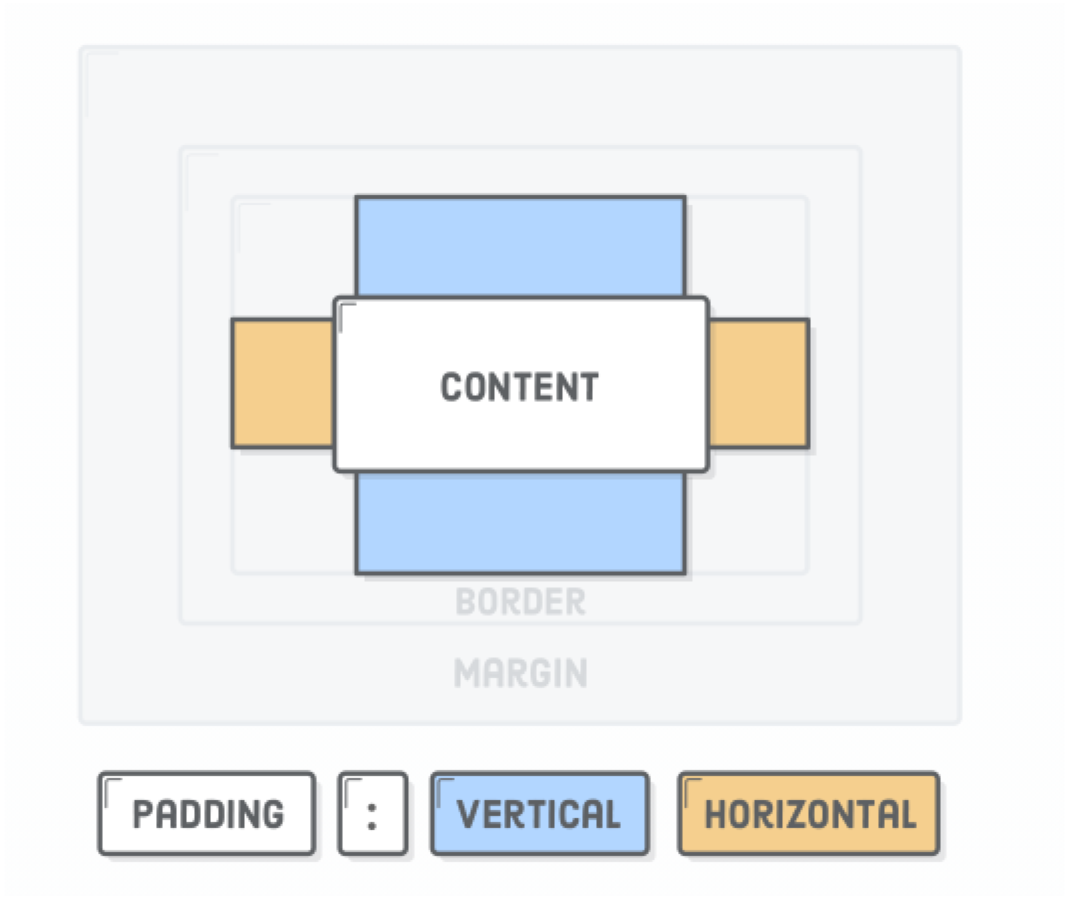
Padding



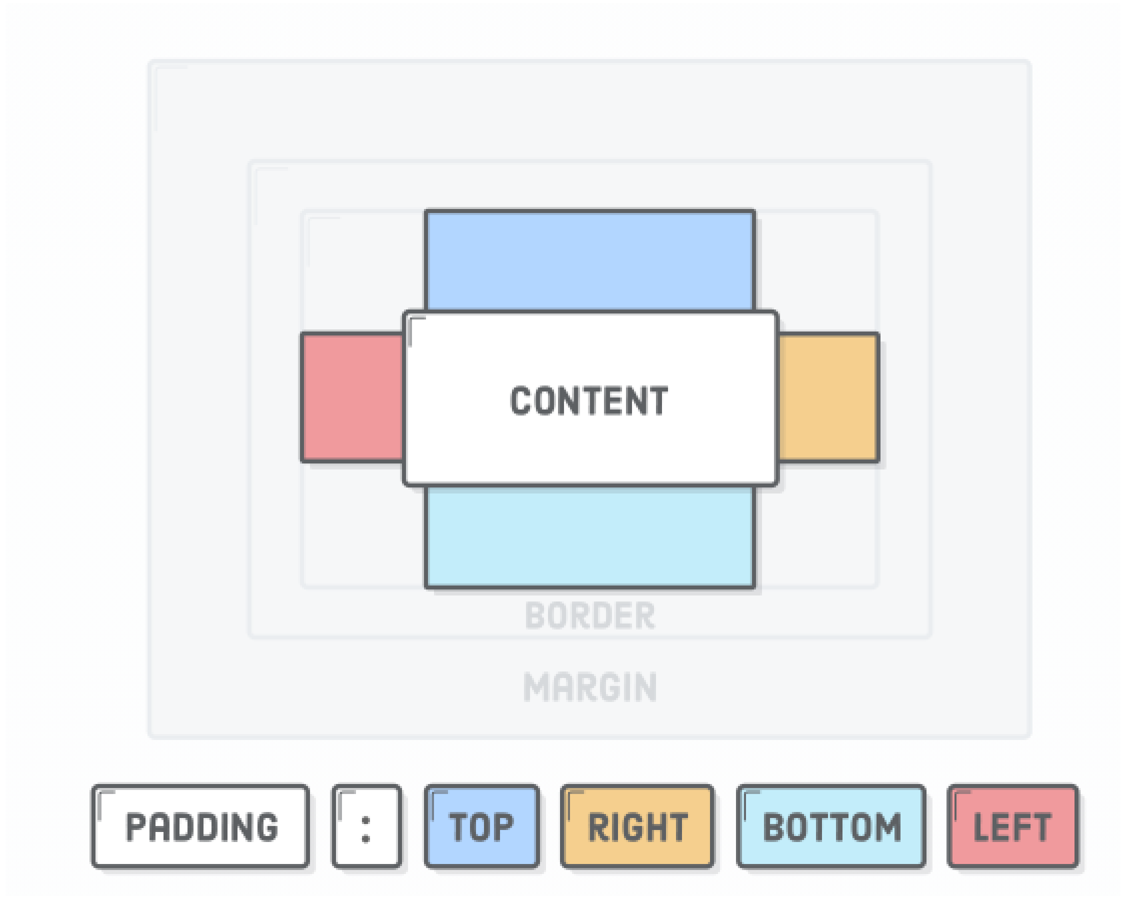


```
h1 {
  padding: 50px;
}
```

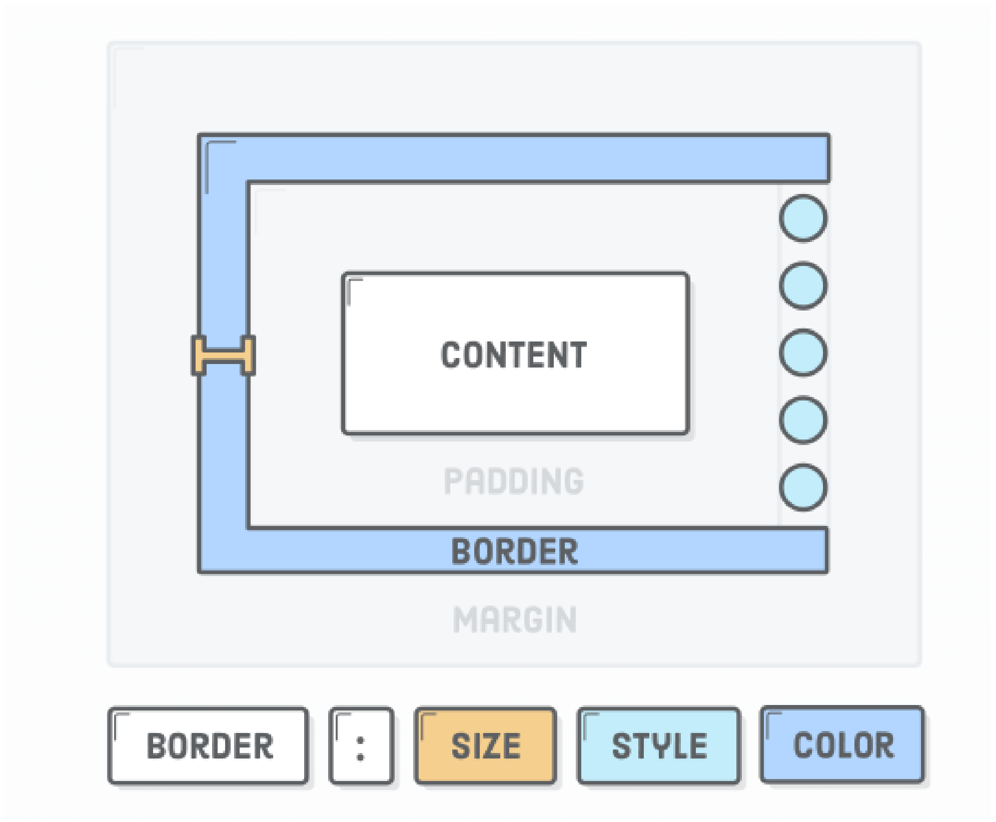
Padding



Padding



Border



```
h1 {
  padding: 50px;
  border: 1px solid #5D6063;
}
```

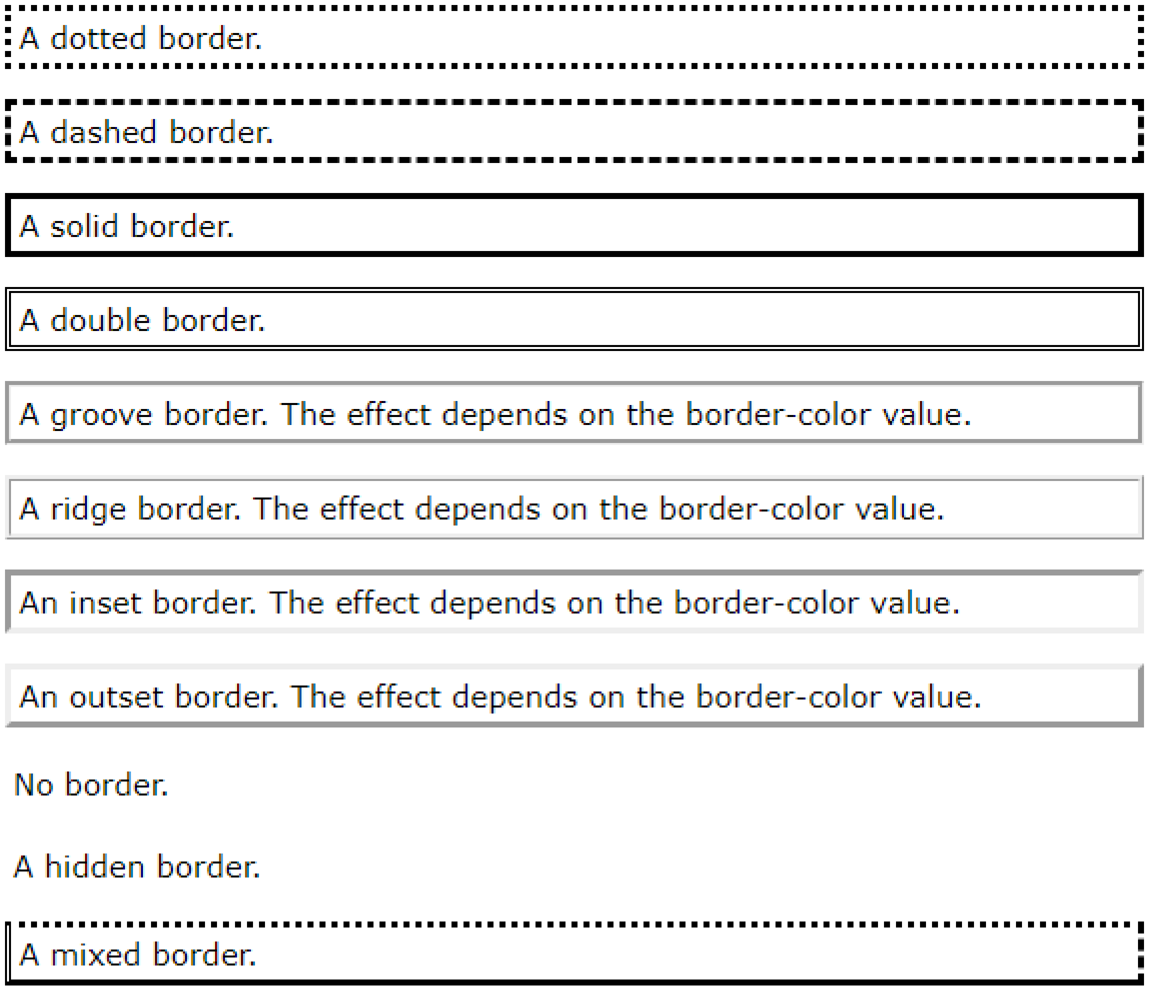
Border-styles

The border-style property specifies what kind of border to display. The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

Source: https://www.w3schools.com/css/css_border.asp

Border-styles Example



A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

Border-?

```
border-bottom: 1px solid #5D6063;
```

For Debugging

```
border: 1px solid red;
```

Padding vs. Margin

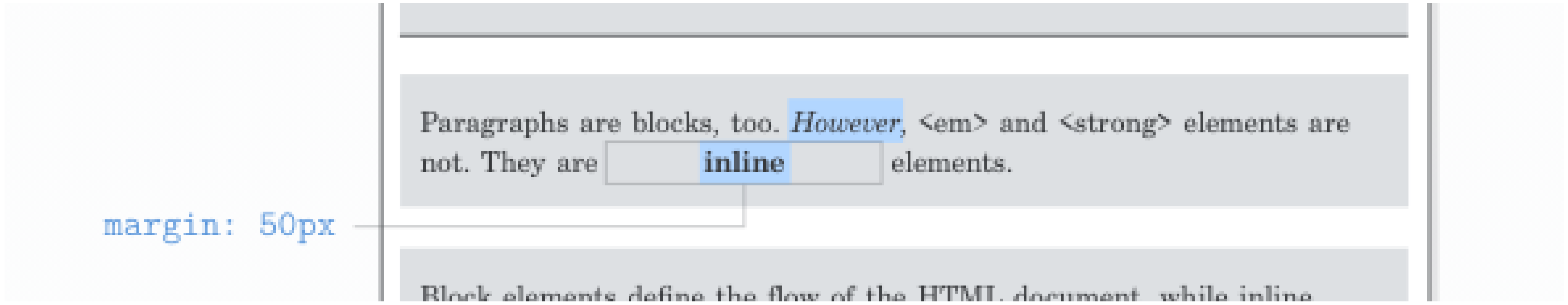
Margins and padding can accomplish the same thing in a lot of situations, making it difficult to determine which one is the “right” choice. The most common reasons why you would pick one over the other are:

- The padding of a box has a background, while margins are always transparent.
- Padding is included in the click area of an element, while margins aren’t.
- Margins collapse vertically, while padding doesn’t (we’ll discuss this more in the next section).

If none of these help you decide whether to use padding over margin, then don’t fret about it—just pick one. In CSS, there’s often more than one way to solve your problem.

Margin on inline elements

Inline boxes *completely ignore* the top and bottom margins of an element



Paragraphs are blocks, too. *However*, <em> and <strong> elements are not. They are **inline** elements.

Block elements define the flow of the HTML document, while inline

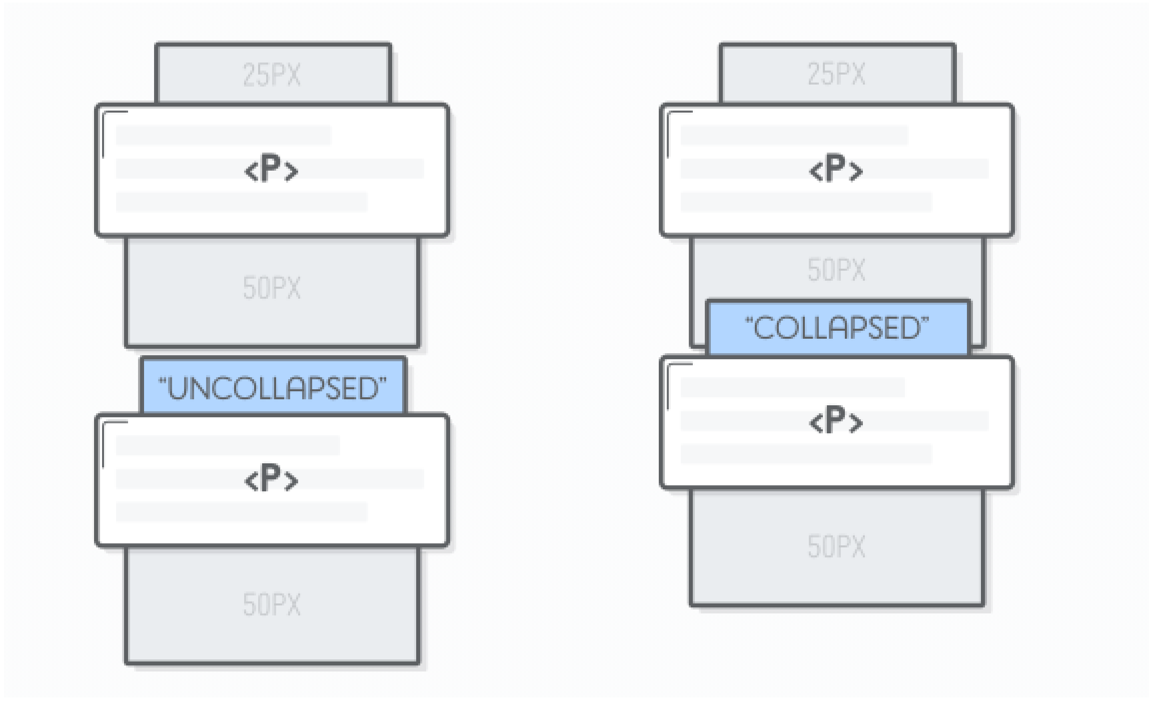
```
strong {
  margin: 50px;
}
```

Vertical Margin Collapse

When you have two boxes with vertical margins sitting right next to each other, they will collapse. Instead of adding the margins together like you might expect, only the biggest one is displayed.

```
p {
  padding: 20px 0 20px 10px;

  margin-top: 25px;
  margin-bottom: 50px;
}
```



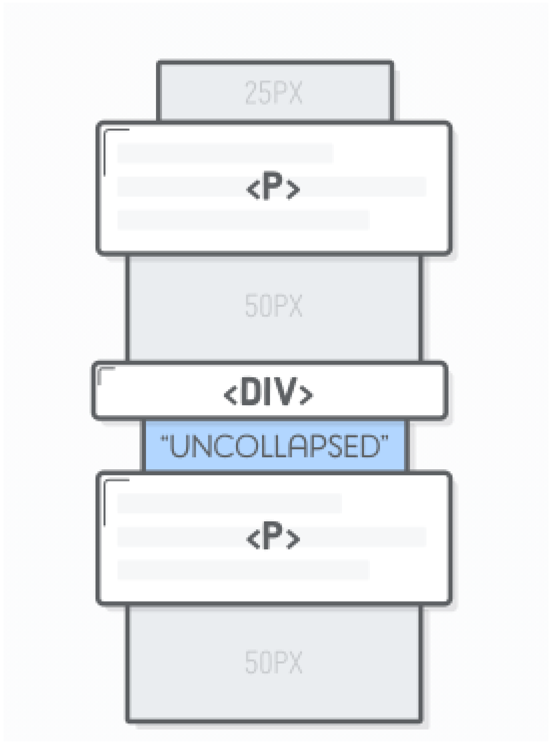
Preventing Vertical Margin Collapse

```
<p>Paragraphs are blocks, too. <em>However</em>, &lt;em&gt; and
&lt;strong&gt;
elements are not. They are <strong>inline</strong> elements.</p>

<div style='padding-top: 1px'></div>  <!-- Add this -->

<p>Block elements define the flow of the HTML document, while inline
elements
do not.</p>
```

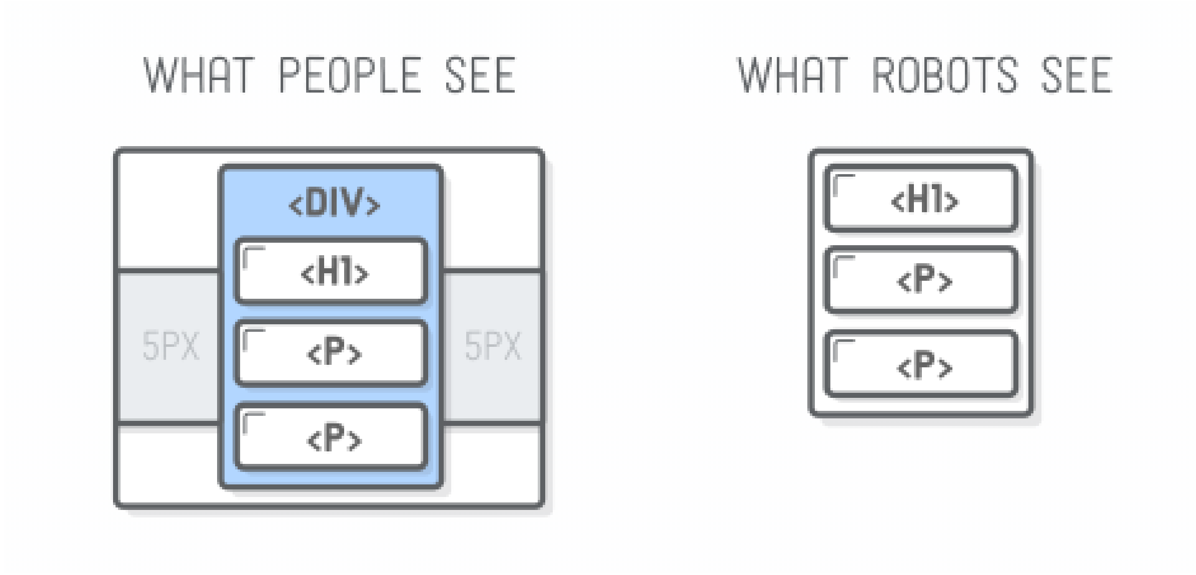
Remember that padding doesn't ever collapse,
so an alternative solution would be to use padding
to space out our paragraphs instead of the margin
property.



Generic Boxes

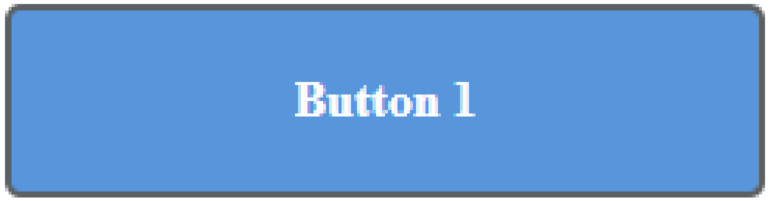
There are many times when we need a generic box purely for the sake of styling a web page. This is what <div> and <span> are for.

Remember that <div> doesn't alter the semantic structure of a page. This makes it a great tool for defining the presentational structure of a web page. By wrapping other HTML elements in <div> tags, we can organize our site into larger layout-oriented chunks without messing up how search engines view our content.



Example

Create the following element using CSS



Solution

```
div {
  color: #FFF;
  background-color: #5995DA;
  font-weight: bold;
  padding: 20px;
  text-align: center;
  border: 2px solid #5D6063;
  border-radius: 5px;
  width: 200px;
}
```

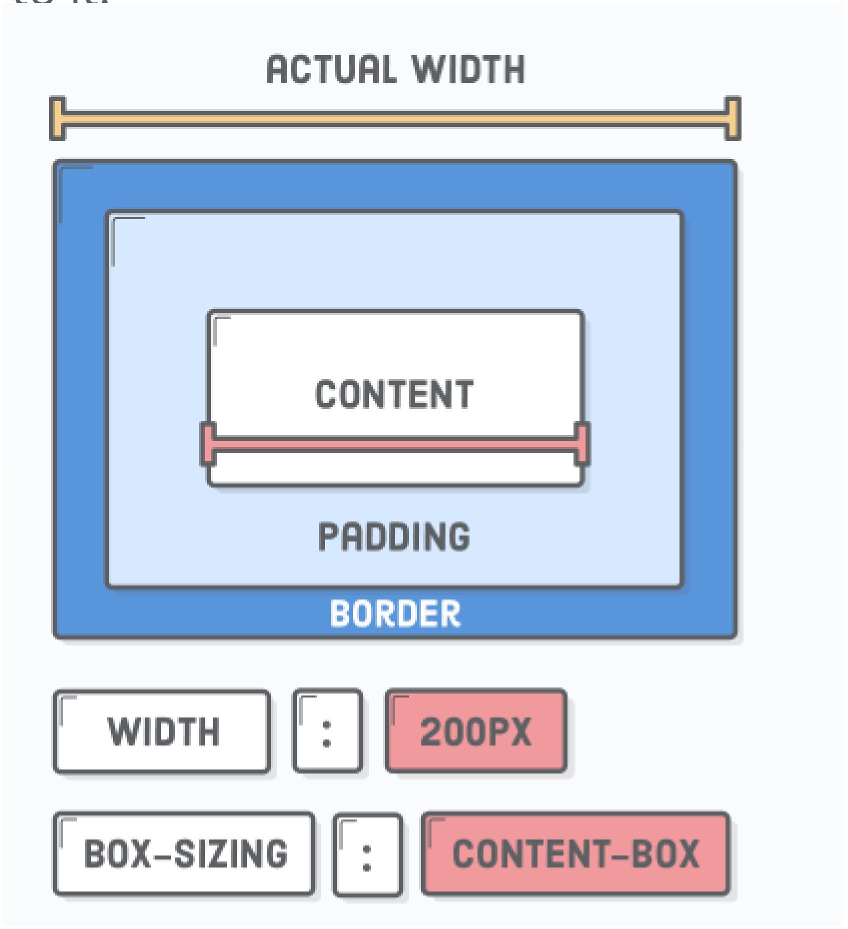
Reusable Solution

```
.btn {
  color: #FFF;
  background-color: #5995DA;
  font-weight: bold;
  padding: 20px;
  text-align: center;
  border: 2px solid #5D6063;
  border-radius: 5px;
  width: 200px;
}
```

Content Box vs. Border Box

The width and height properties only define the size of a box's content. Its padding and border are both added on top of whatever explicit dimensions you set. This explains why you'll get an image that's 244 pixels wide when you take a screenshot of our button, despite the fact that it has a width: 200px declaration attached to it.

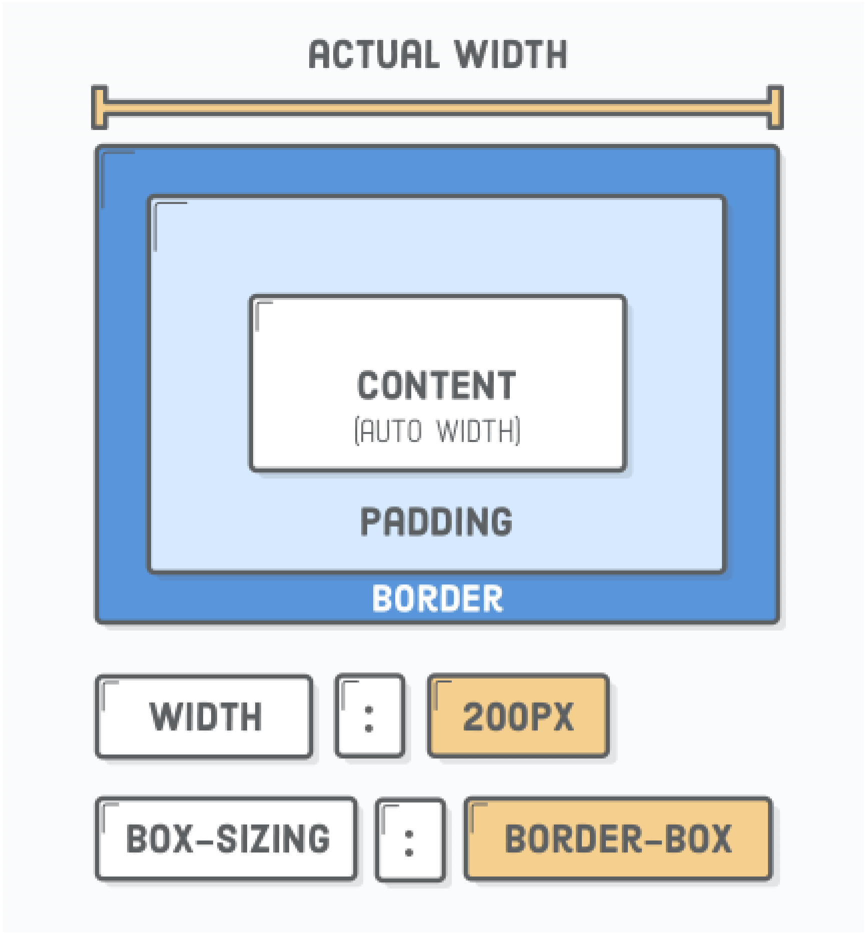
```
div {
  color: #FFF;
  background-color: #5995DA;
  font-weight: bold;
  padding: 20px;
  text-align: center;
  border: 2px solid #5D6063;
  border-radius: 5px;
  width: 200px;
}
```



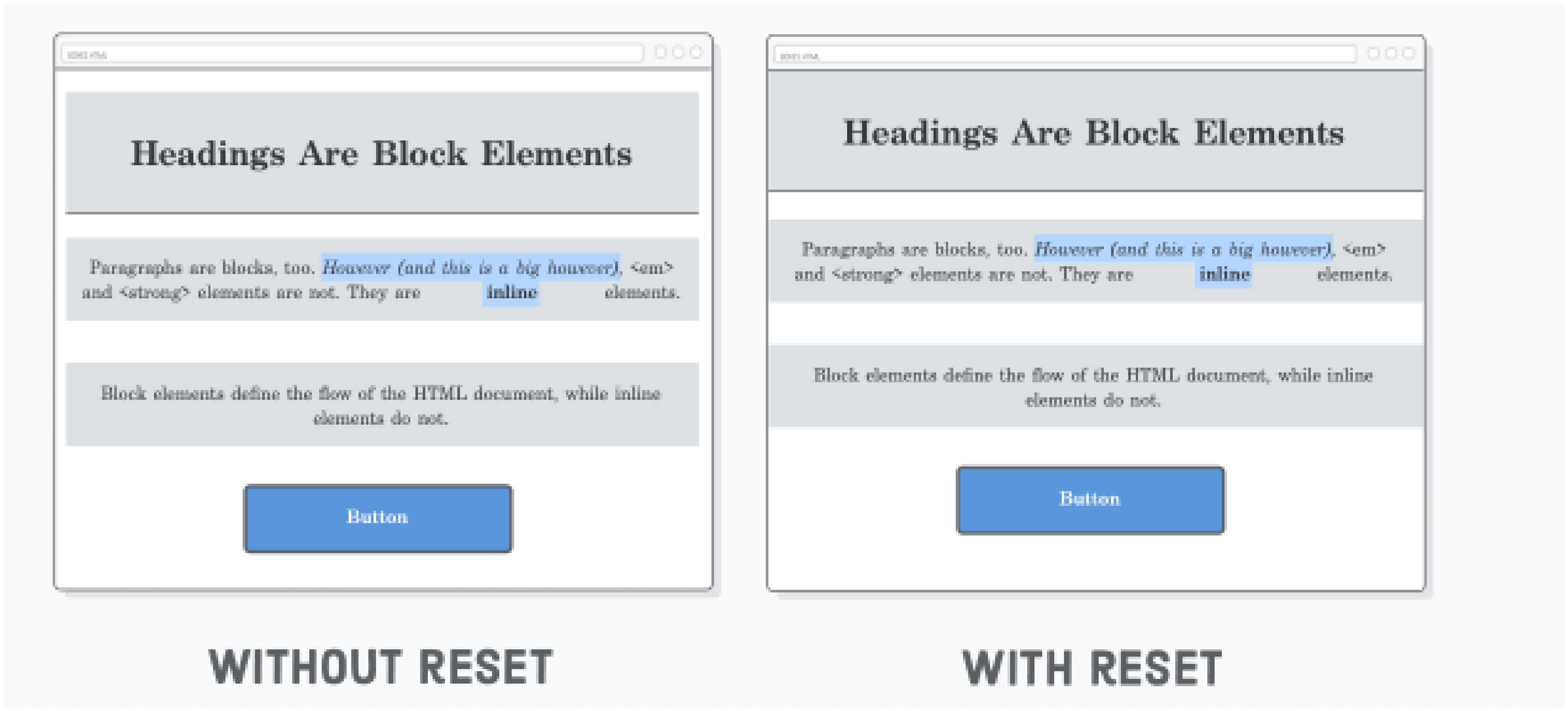
Content Box vs. Border Box

The width and height properties only define the size of a box's content. Its padding and border are both added on top of whatever explicit dimensions you set. This explains why you'll get an image that's 244 pixels wide when you take a screenshot of our button, despite the fact that it has a width: 200px declaration attached to it.

```
div {
  color: #FFF;
  background-color: #5995DA;
  font-weight: bold;
  padding: 20px;
  text-align: center;
  border: 2px solid #5D6063;
  border-radius: 5px;
  width: 200px;
  box-sizing: border-box;
}
```



Resetting Default Style



Headings Are Block Elements

Paragraphs are blocks, too. *However (and this is a big however)*, <em> and <strong> elements are not. They are **inline** elements.

Block elements define the flow of the HTML document, while inline elements do not.

Headings Are Block Elements

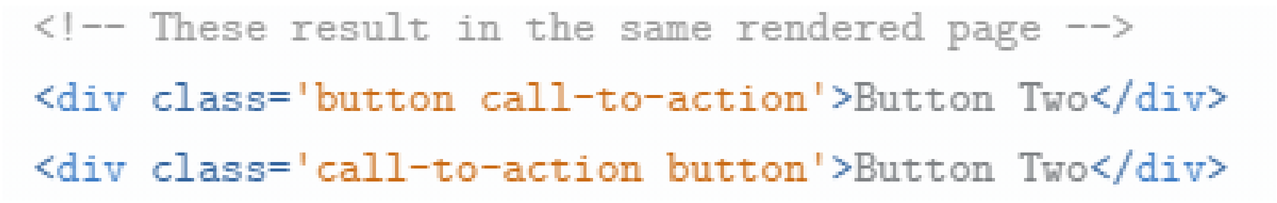
Paragraphs are blocks, too. *However (and this is a big however)*, <em> and <strong> elements are not. They are **inline** elements.

Block elements define the flow of the HTML document, while inline elements do not.

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

Note

1. Order matters **<u>for CSS File. Not HTML Class.</u>**
2. It’s adding a new font-style declaration to the original .button rule.
3. It’s overriding an existing background-color style from .button.
4. Overriding occurs because of the order of .call-to-action and .button in our stylesheet. When there’s two conflicting properties in a CSS file, the last one is always the one that gets applied. So, if you moved .call-to-action to the top of styles.css, .button would have the final word on the value of background-color, and it would remain blue.
5. This means that the order of the class attribute in our HTML element has no effect on override behavior. Multiple classes on a single element are applied “equally” (for lack of a better term), so the precedence is determined solely by the order of the rules in styles.css. In other words, the following elements are effectively equivalent:



```html
<!-- These result in the same rendered page -->
<div class='button call-to-action'>Button Two</div>
<div class='call-to-action button'>Button Two</div>
```

More Things to Learn

1. Flexbox
2. CSS Framework (Semantic UI/Bootstrap)