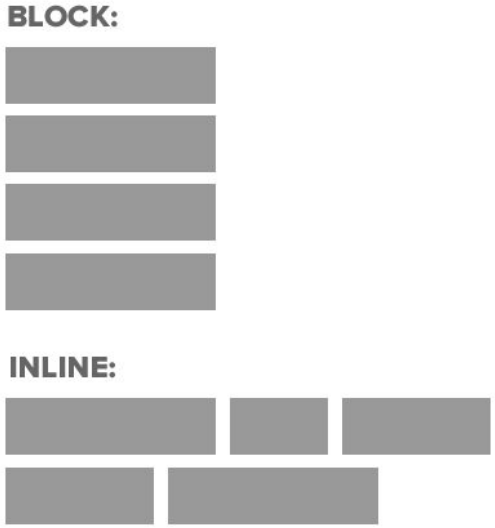
Box Model, Layout & Positioning 1

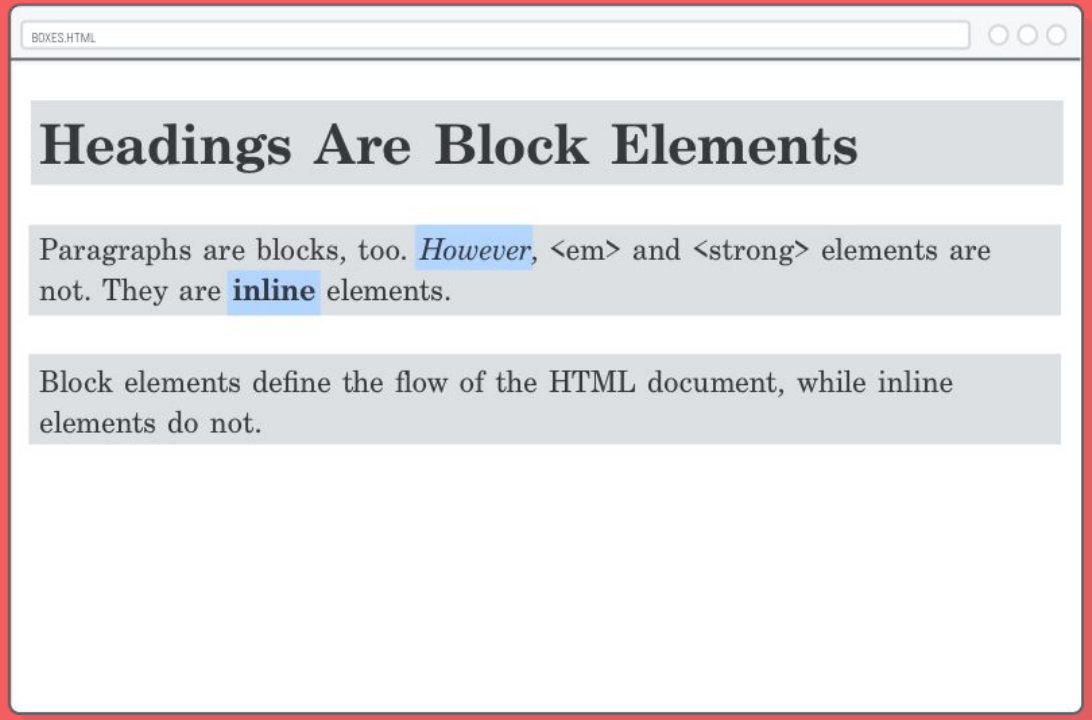
Block Elements vs Inline Elements

Last session, we briefly touched on how CSS uses “boxes” to define the layout of a web page.

Each HTML element rendered on the screen is a box, and they come in two flavors: “block” boxes and “inline“ boxes.

Let's see an example of this




BOXES.HTML
Headings Are Block Elements
Paragraphs are blocks, too. However, and elements are not. They are inline elements.
Block elements define the flow of the HTML document, while inline elements do not.

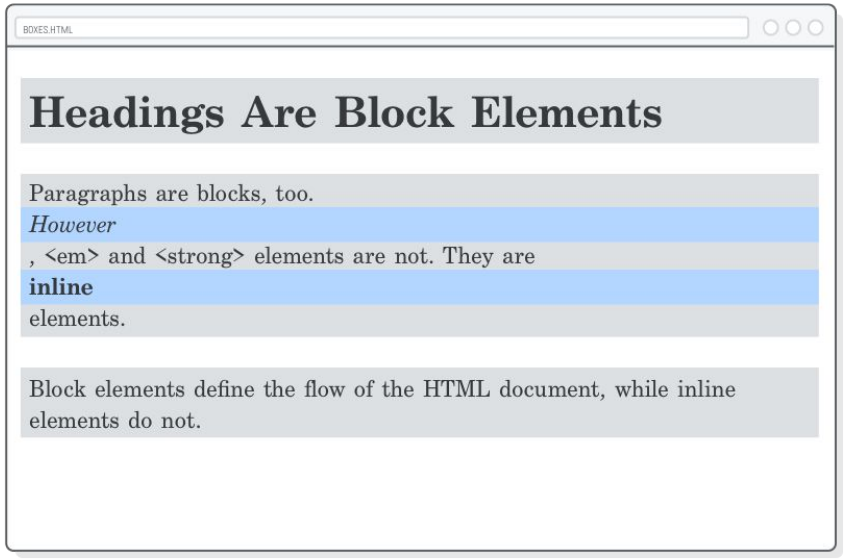
Inline vs Block Boxes

1. Block boxes always appear below the previous block element. This is the “natural” or “static” flow of an HTML document when it gets rendered by a web browser.
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.

Changing Box Behavior

We can override the default box type of HTML elements with the CSS **display** property.

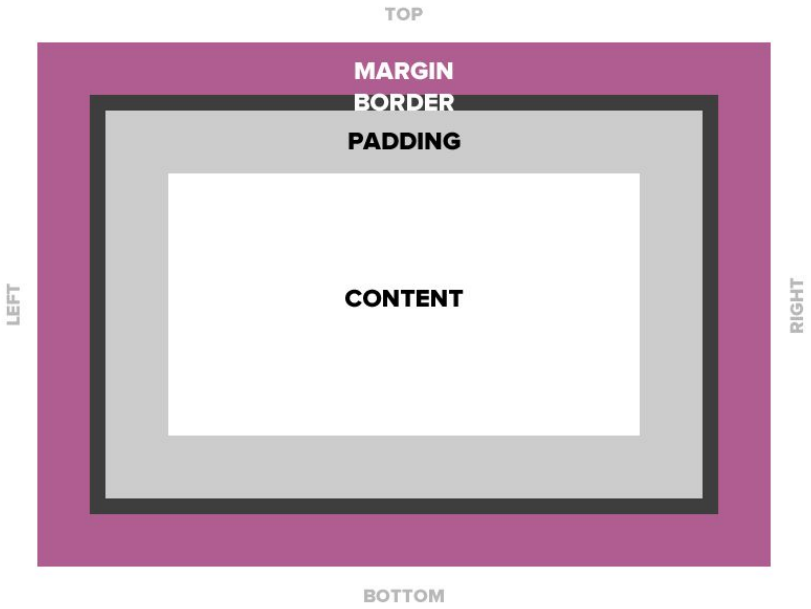
display property can have **inline** or **block** as its value.




```
em, strong {
  background-color: #B2D6FF;
  display: block;
}
```

Box Model

The “CSS box model” is a set of rules that determine the dimensions of every element in a web page.



Box Model

CSS Box Model gives each box (both inline and block) four properties:

- **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.

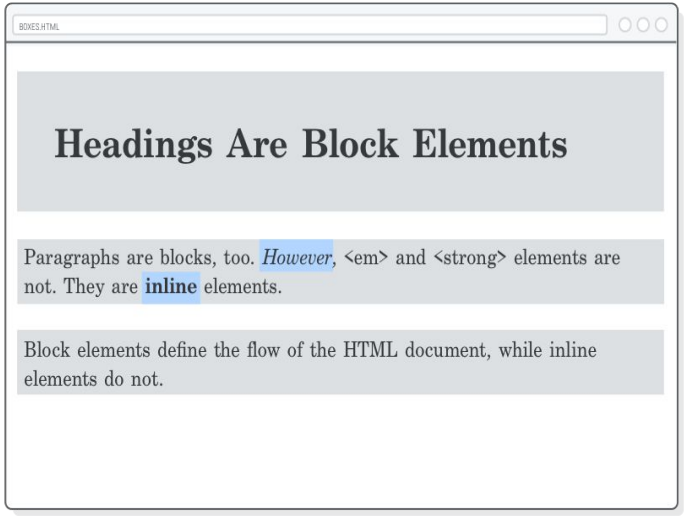
This is everything a browser needs to render an element's box.

The content is what you author in an HTML document, the rest of them are purely presentational, so they're defined by CSS rules.

Padding

The padding property…you guessed it…defines the padding for the selected element:

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



This adds 50 pixels to each side of the <h1> heading.

Notice how the background color expands to fill this space.

Padding

Sometimes you'll only want to style one side of an element. For that, CSS provides the following properties:

```
p {
  padding-top: 20px;
  padding-bottom: 20px;
  padding-left: 10px;
  padding-right: 10px;
}
```

You can use any unit for the padding of an element, not just pixels. Again, em units are particularly useful for making your margins scale with the base font size.

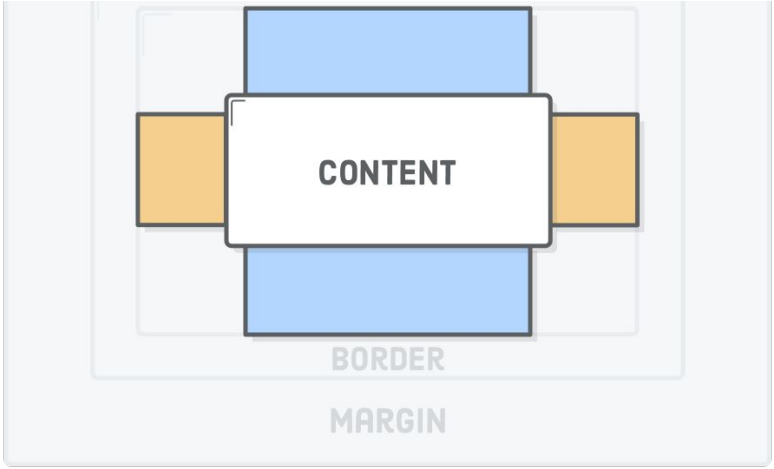
Padding: Shorthand Format

Typing out all of these properties out can be tiresome, so CSS provides an alternative “shorthand” form of the padding property that lets you set the **top/bottom** and **left/right** padding with only one line of CSS.

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

Vertical

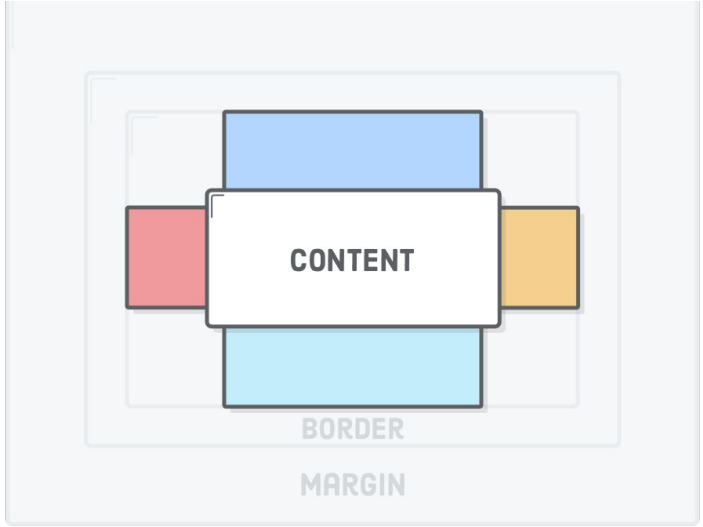
Horizontal



Padding: Shorthand Format

Alternatively, if you provide four values, you can set the padding for each side of an element individually. The values are interpreted clockwise, starting at the top:

```
p {                  Right       Left
  padding: 20px 10px 15px 5px;
}
           To
           p          Bottom
```



Borders

Border is a line drawn around the content and padding of an element.

Borders are specified as "thickness, style, color"

For example this CSS rule creates a solid, thin red border:

```
h1 {
  border: 1px solid red;
}
```

And this one creates a thick, dotted green border:

```
h1 {
  border: 4px dotted green;
}
```

Borders

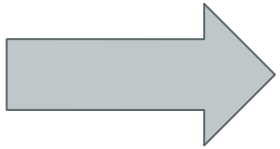
Like padding, there are -top, -bottom, -left, and -right variants for the border property:

```
h1 {
  border-top: 1px solid red;
  border-right: 2px dotted green;
  border-bottom: 3px dashed yellow;
  border-left: 4px double purple;
}
```

Borders

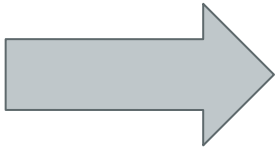
You can specify each property of a border separately, or all three together.

```
h1 {
  border: 4px dotted green;
}
```



```
h1 {
  border-width: 4px;
  border-style: dotted;
  border-color: green;
}
```

```
h1 {
  border-top: 1px solid red;
  border-right: 2px dotted green;
  border-bottom: 3px dashed yellow;
  border-left: 4px double purple;
}
```



```
h1 {
  border-top-width: 1px;
  border-top-style: solid;
  border-top-color: red;
  ...
}
```

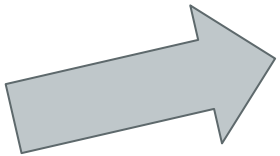
Margins

Margins define the space outside of an element's border. Or, rather, the space between a box and its surrounding boxes.

To define margins, we use the same format for specifying CSS rules as we did for padding:

```
h1 {
  margin: 50px;
}

p {
  margin-top: 20px;
  margin-bottom: 20px;
  margin-left: 10px;
  margin-right: 10px;
}
```



```
p {
  margin: 20px 10px;
}

p {
  margin: 20px 10px 20px 10px;
}
```

Margins vs Padding

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).

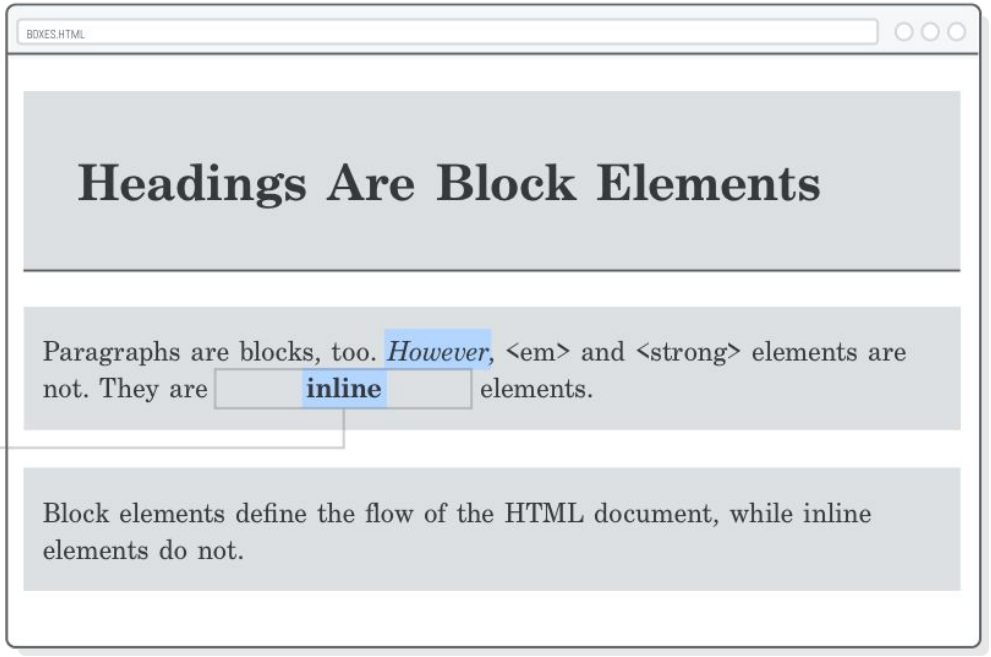
If none of these help you decide whether to use padding over margin, then don’t fret about it—just pick one. In CSS, there’s often more than one way to solve your problem.

Margins on Inline Elements

watch what happens when we add a big margin to our element

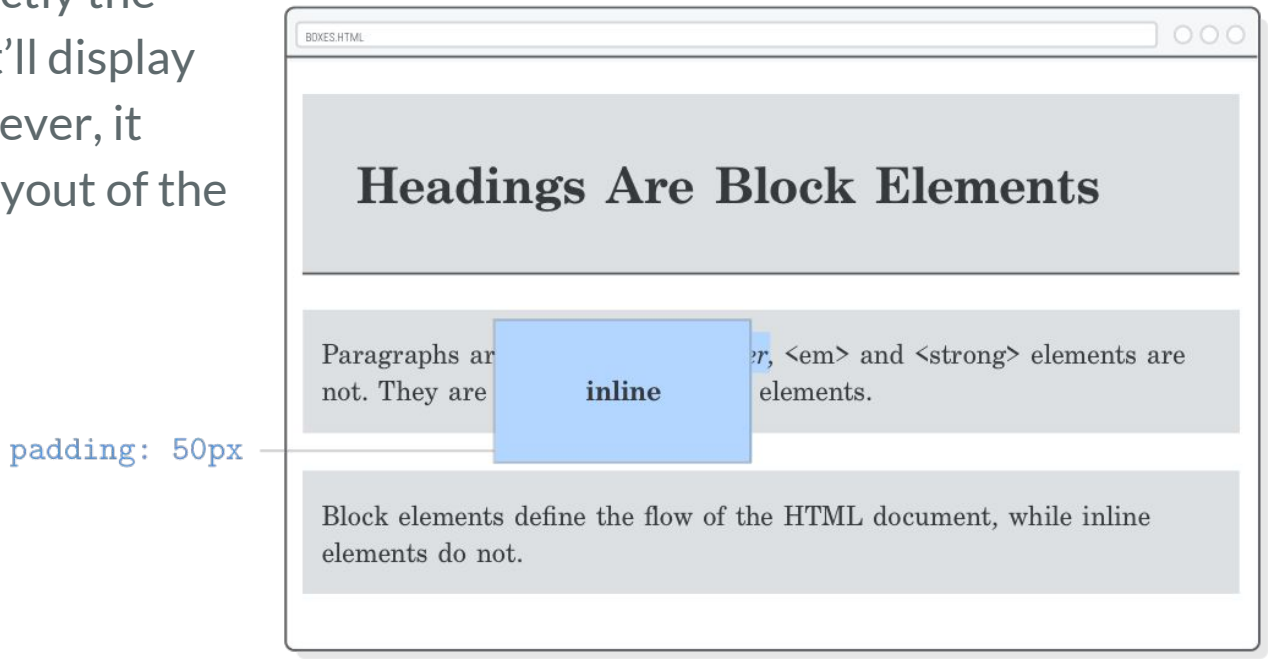
Inline boxes completely **ignore the top and bottom margins** of an element.

The horizontal margins display just like we'd expect



Padding on Inline Elements

If we change margin to padding, we'll discover that this isn't exactly the case for a box's padding. It'll display the blue background; however, it won't affect the vertical layout of the surrounding boxes.



Positioning

Static Positioning

HTML elements are positioned **static by default**.

Static elements are positioned in the **normal flow** of the page

In normal flow, **inline boxes** flow from left to right, **wrapping to next line when needed**.

In normal flow, **block boxes** flow from **top to bottom**, making a **new line after** every box.

Static Positioning for Inline Boxes

```
<img src="img/cookie1.png"/>
<img src="img/cookie2.png"/>
<img src="img/cookie3.png"/>
...
<img src="img/cookie2.png"/>
<img src="img/cookie3.png"/>
```

Static Positioning for Block Boxes

```
<p>Greetings</p>
<p>Hello</p>
<p>Hi there!</p>
```

Greetings

Hello

Hi there!

Relative Positioning

Takes the element **out of the normal flow**, allowing it to be moved to the top, left, right or bottom.

Does not affect the elements surrounding it.

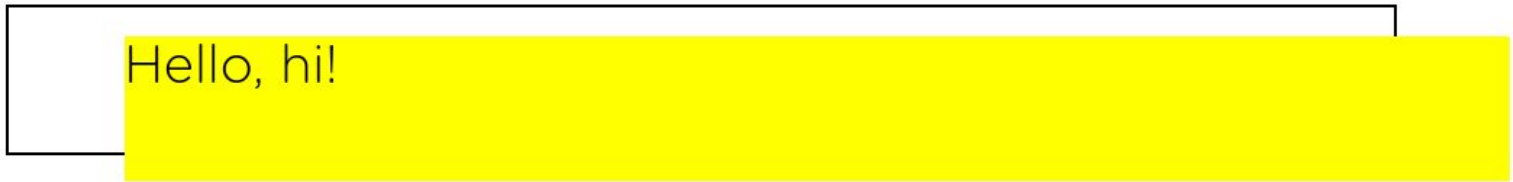
Makes an element a "positioning context" in which to position other elements relative to it.

Relative positioning and absolute positioning are used together.

Relative Positioning

The "relative" value will still put the element in the normal flow, but then offset it according to top/left/right/bottom properties.

```
.relative{
  position: relative;
  left: 80px;
  top: 20px;
  height: 100px;
  background-color: yellow;
}
```



Absolute Positioning

Positions element **outside of the normal** flow.

An absolutely positioned element is **offset from its container block**, positioned relative.

Its container block is the **first element that has a position other than static**.

If no such element is found, the container block is <html>.

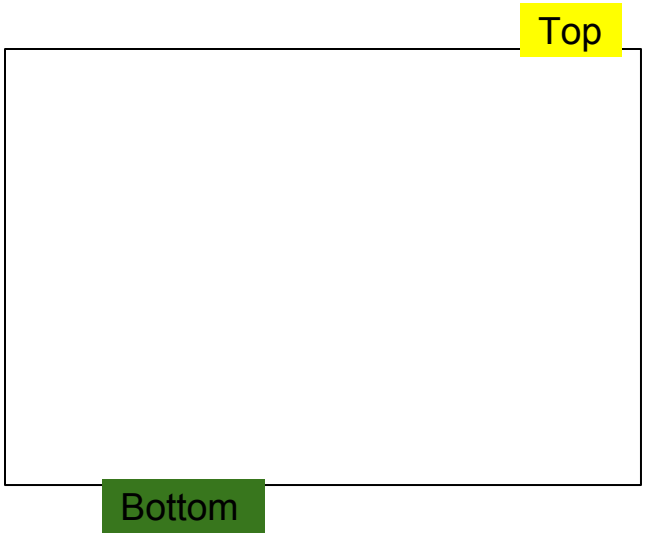
Other elements act as if it's not there.

Determined by its **offset values** in the properties **top, bottom, right and left**.

Absolute Positioning

The "absolute" value will take the element out of the normal flow and position it in relation to the window (or the closest non-static element).

```
.top{
  position: absolute;
  top: -40px;
  right: 10px;
  background-color: yellow
}
.bottom{
  position: absolute;
  bottom: -40px;
  left:60px;
  background-color: green
}
```



Example: Absolute Positioning

Here's an example of an image with a caption absolutely positioned over top of it.

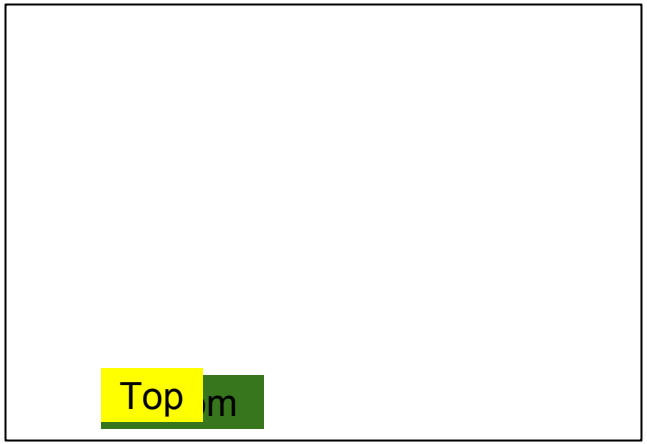
The containing div has a position of relative, and the caption has a position of absolute.

Z-index

Sometimes elements overlap. You can change the order of overlapping with **z-index**.

The element with **highest** z-index goes **on top**.

```
.bottom{
  position: absolute;
  bottom: 10px;
  left:60px;
  background-color: yellow;
}
.top{
  position: absolute;
  bottom: 15px;
  left:60px;
  background-color: green;
  z-index: 2;
}
```



Let's try this!

Let's create a div that contains an image and a caption, and position the caption absolutely overtop the image.