CSS Positioning & Display

Lesson Time: 60 Minutes

USE POSITION FOR FINE DETAIL POSITIONING AND OVERLAP

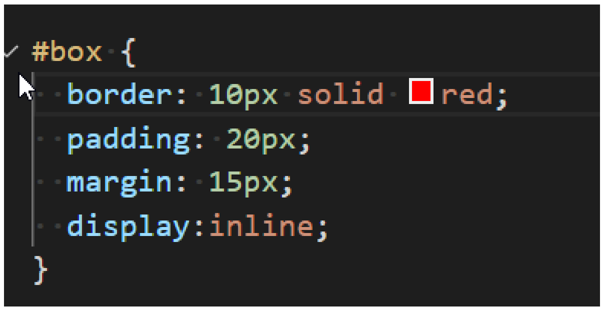
This is probably the hardest lesson on CSS and can be difficult to understand. The youtube video at the end of the lesson knocks it out of the park. Most of the time, we recommend using Flexbox, but there are times where you need absolute control over positioning, such as a case where you want to elements to overlap each other.

First, let’s review BLOCK and INLINE HTML Elements again.

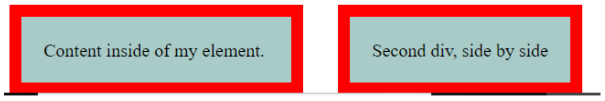
The normal “document flow” is made up of block and inline elements.

BLOCK ELEMENTS	INLINE ELEMENTS
Block elements take up all of the available space that it can. New elements sit on the line below.	Inline elements take up only the space it needs. New elements can sit to the right of inline elements.
<div> <p> <table>
 <h1>	<img> <a> <span>

The CSS Property **display** allows us to modify these default properties. A <div> can be changed to inline and an <img> changed to block.



```css
#box {
  border: 10px solid red;
  padding: 20px;
  margin: 15px;
  display:inline;
}
```



Content inside of my element.

Second div, side by side

In this code, we apply **display:inline** to a group of <div>. They now appear side by side.

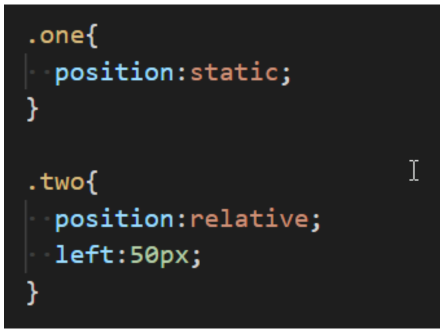
Another powerful feature of display is **display:none**. An element will disappear completely like it was never there when set to **display:none**. This can be used with JS to toggle an element on the page on or off.

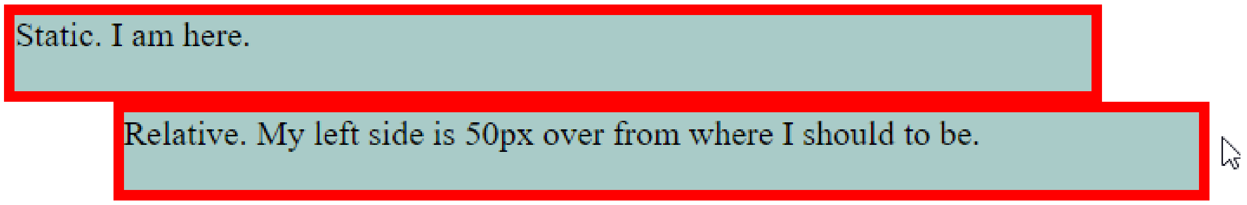
Positioning

Up until now, we've only worked with one kind of positioning, **static** positioning. By default, all html elements have static positioning and follow the rules of block and inline elements.

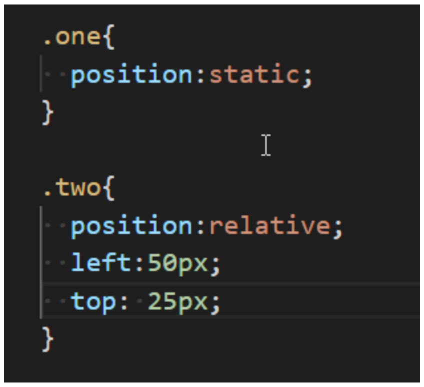
The CSS property position allows us to break away from block and inline rules.

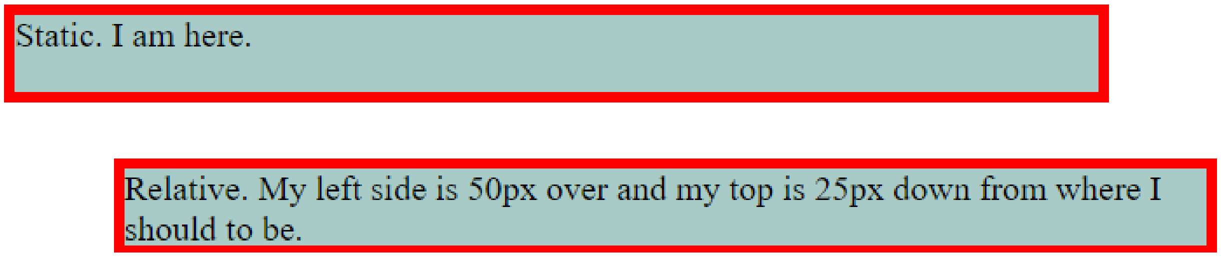
First up is **relative**. Relative gives us access to four new properties, **left**, **right**, **top**, and **bottom**.


```
.one{
  position:static;
}

.two{
  position:relative;
  left:50px;
}
```


Static. I am here.

Relative. My left side is 50px over from where I should to be.


```
.one{
  position:static;
}

.two{
  position:relative;
  left:50px;
  top: 25px;
}
```


Static. I am here.

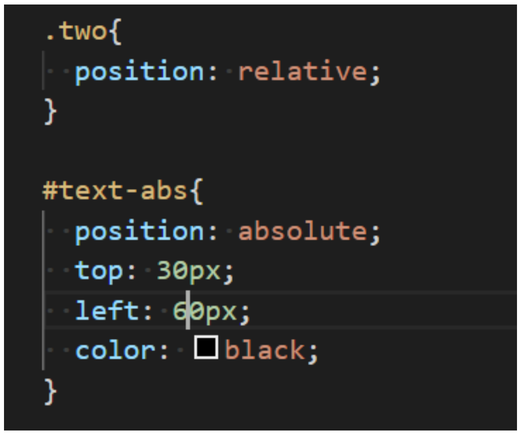
Relative. My left side is 50px over and my top is 25px down from where I should to be.

Some key points on relative.

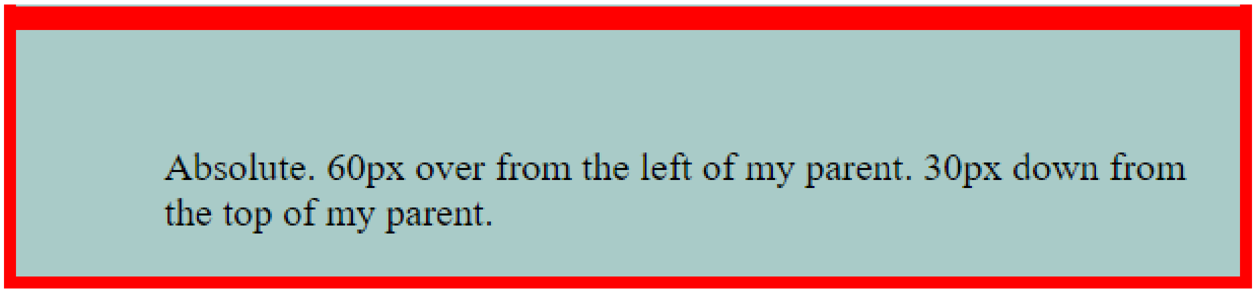
1. The element is moved based on where it would normally be if it were static.
2. Left moves the left edge over by x amount, top moves the top edge DOWN by x, .
3. Right moves the right edge over by x amount , bottom moves the bottom edge UP by x.

Next position is **absolute**.

1. Absolute moves the element based on its parent element, not itself.
2. The parent **must not be static.**



```
.two{
  position: relative;
}

#text-abs{
  position: absolute;
  top: 30px;
  left: 60px;
  color: ■black;
}
```



Absolute. 60px over from the left of my parent. 30px down from the top of my parent.

Last up is **sticky**. Sticky is a mix up relative and absolute. Sticky position starts off just like relative. The difference is that when the element is scrolled off-page, the position becomes absolute to the browser window. The effect is that the element sticks.

So when do we use relative, absolute, and sticky positions?

1. For fine details, complete control over position.
2. For when we want elements on the page to overlap

For **basic** layout, we'll use the much easier FLEXBOX or GRID.

If you are having trouble with this lesson, watch the youtube videos below and look at the W3School tutorials at https://www.w3schools.com/css/css_positioning.asp

Understanding the CSS Display Property
Steve Griffith
Published on Sep 14, 2018

Learn CSS Position In 9 Minutes
Web Dev Simplified
Published on Apr 30, 2019

Key Terms	
Lesson Files	
Additional Resources	https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements
Further Learning	