Lecture 16

CSS Layout - The display Property

The display property is the most important CSS property for controlling layout. The display property specifies if/how an element is displayed. Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline. display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved. visibility:hidden; also hides an element. However, the element will still take up the same space as before.

```
li {
  display: inline;
}
```

```
span {
  display: block;
}
```

```
h1.hidden {
  display: none;
}
```

CSS Layout - The position Property

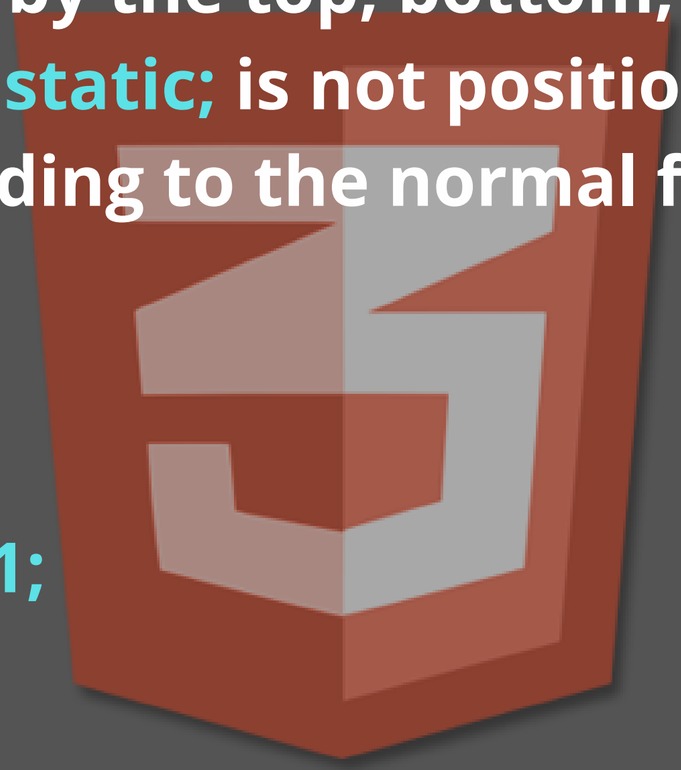
The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky). There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;

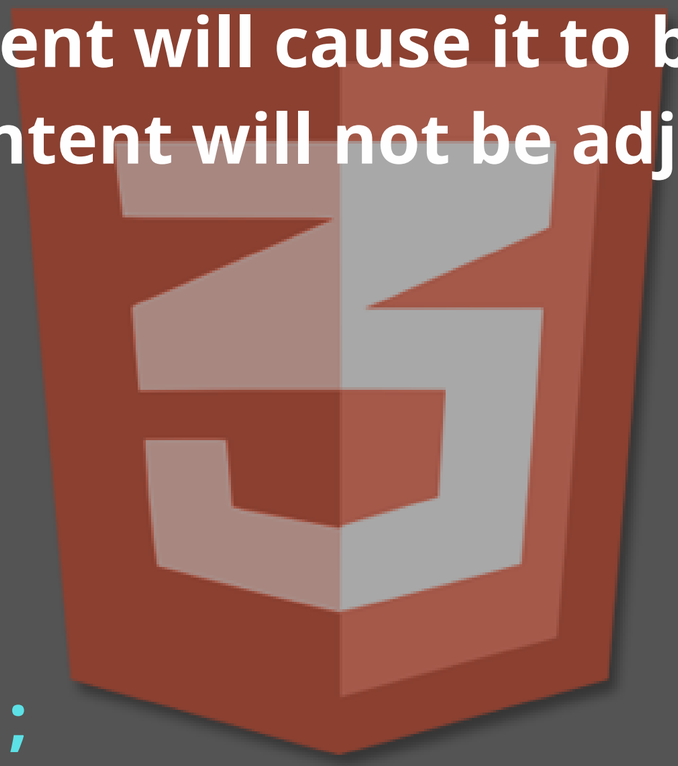
HTML elements are positioned static by default. Static positioned elements are not affected by the top, bottom, left, and right properties. An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:



```
div.static {
  position: static;
  border: 3px solid #73AD21;
}
```

position: relative;

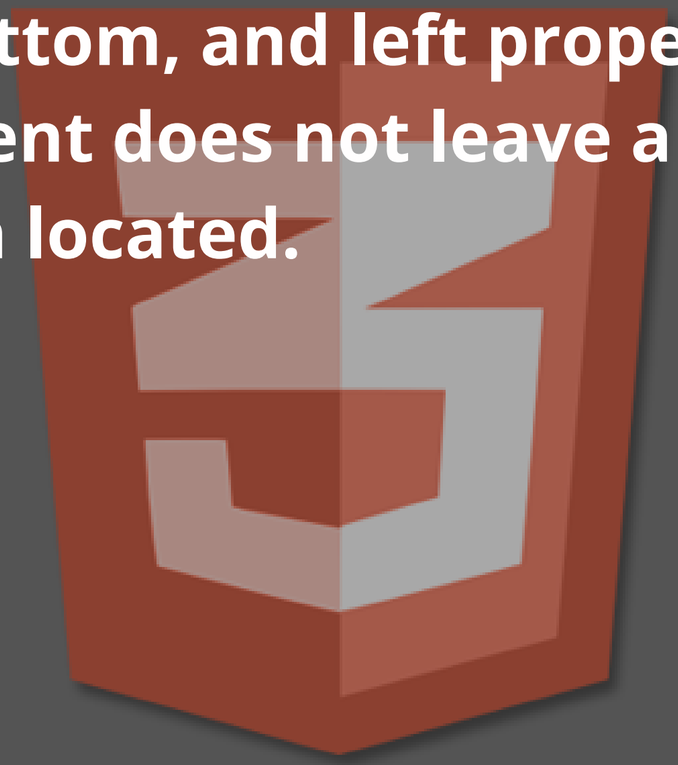
An element with position: relative; is positioned relative to its normal position. Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.



```
div.relative {
 position: relative;
 left: 30px;
 border: 3px solid #73AD21;
}
```

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element. A fixed element does not leave a gap in the page where it would normally have been located.



```
div.fixed {
 position: fixed;
 bottom: 0;
 right: 0;
 width: 300px;
 border: 3px solid #73AD21;
}
```

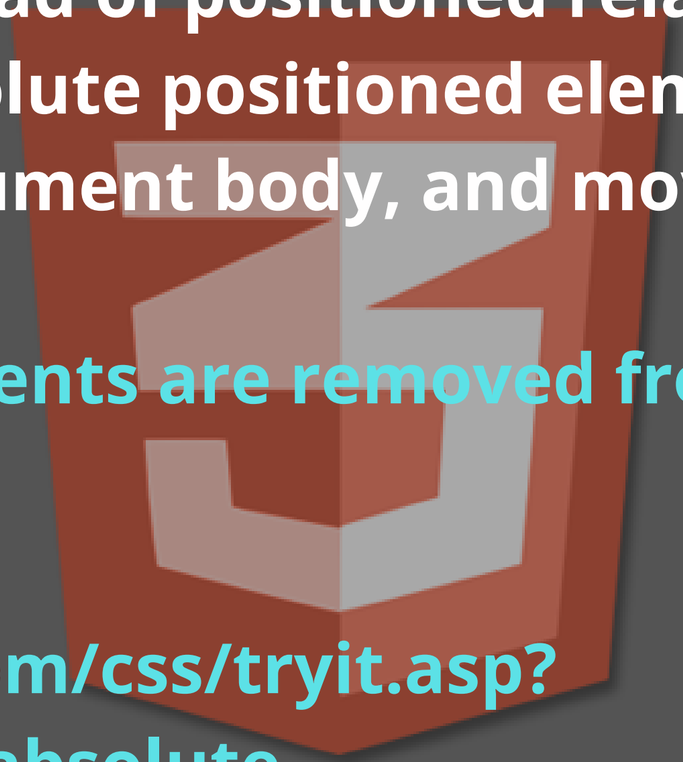
position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed). However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Absolute positioned elements are removed from the normal flow, and can overlap elements.

For example:

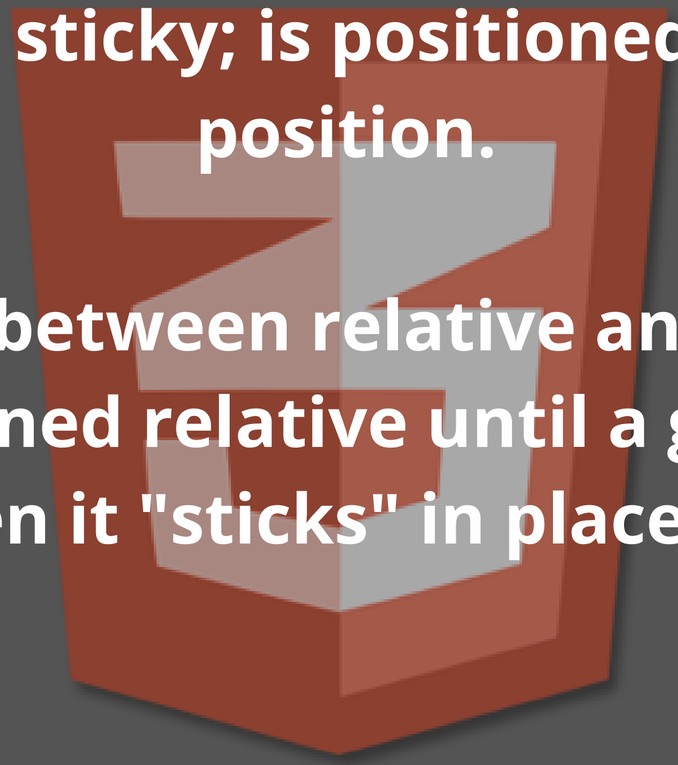
https://www.w3schools.com/css/tryit.asp?filename=trycss_position_absolute



position: sticky;

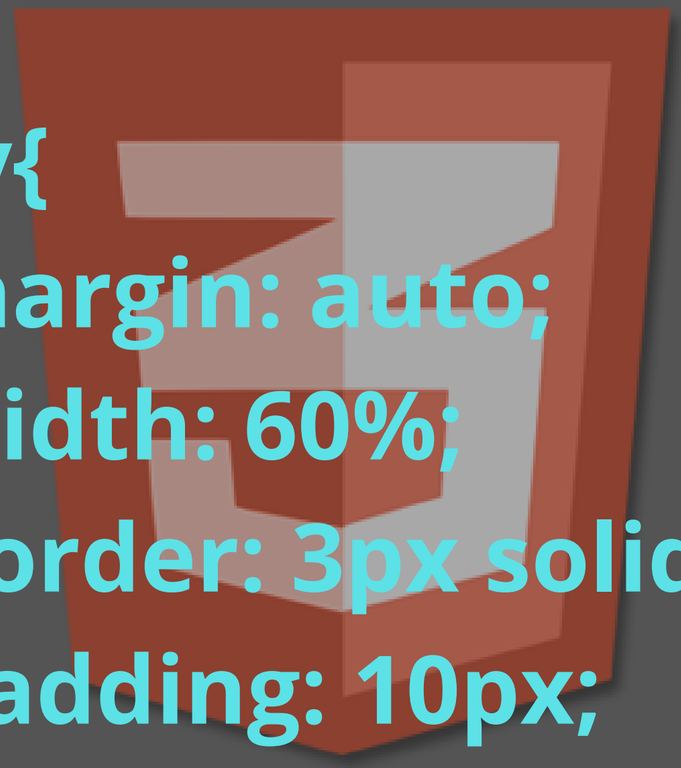
An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).



Center Align block Elements

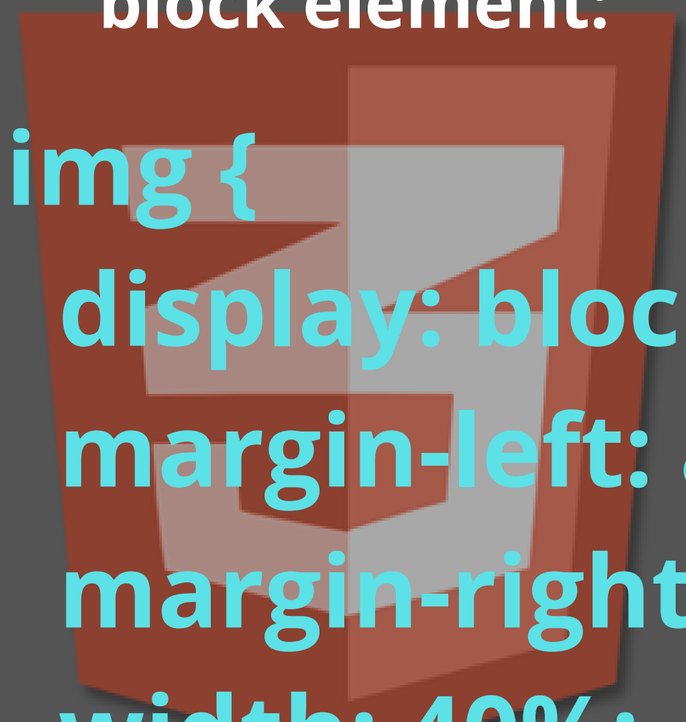
To horizontally center a block element (like), use margin: auto;



```
div{
  margin: auto;
  width: 60%;
  border: 3px solid #73AD21;
  padding: 10px;
}
```

Center an Image

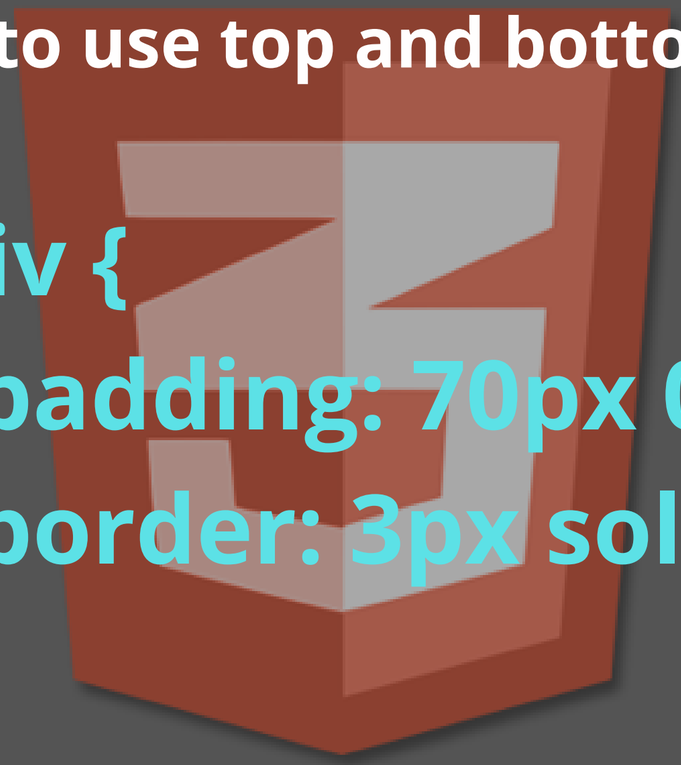
To center an image, set left and right margin to auto and make it into a block element:



```
img {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 40%;
}
```

Center Vertically - Using padding

There are many ways to center an element vertically in CSS. A simple solution is to use top and bottom padding:



```
div {
  padding: 70px 0;
  border: 3px solid green;
}
```