**The position Property**

The position property specifies the type of positioning method used for an element.

There are five different position values:

* static
* relative
* fixed
* absolute
* sticky

## 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:

This <div> element has position: static;

Here is the CSS that is used:

### Example

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.

This <div> element has position: relative;

Here is the CSS that is used:

### Example

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.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

### Example

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

This <div> element has position: fixed;

## 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.

**Note:** A "positioned" element is one whose position is anything except static.

Here is a simple example:

This <div> element has position: relative;

This <div> element has position: absolute;

Here is the CSS that is used:

## Overlapping Elements

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

## CSS Overflow

The CSS overflow property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

The overflow property has the following values:

* visible - Default. The overflow is not clipped. It renders outside the element's box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content
* auto - If overflow is clipped, a scrollbar should be added to see the rest of the content

**Note:** The overflow property only works for block elements with a specified height.

**Note:** In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow: scroll" is set).

## overflow: visible

By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

### Example

div {  
    width: 200px;  
    height: 50px;  
    background-color: #eee;  
    overflow: visible;  
}

## overflow: hidden

With the hidden value, the overflow is clipped, and the rest of the content is hidden:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

### Example

div {  
    overflow: hidden;  
}

## overflow: scroll

Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

### Example

div {  
    overflow: scroll;  
}

## overflow: auto

The auto value is similar to scroll, only it add scrollbars when necessary:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

### Example

div {  
    overflow: auto;  
}

## overflow-x and overflow-y

The overflow-x and overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both):

overflow-x specifies what to do with the left/right edges of the content.  
overflow-y specifies what to do with the top/bottom edges of the content.

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

### Example

div {  
    overflow-x: hidden; /\* Hide horizontal scrollbar \*/  
    overflow-y: scroll; /\* Add vertical scrollbar \*/  
}

## Test Yourself with Exercises!

## All CSS Overflow Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| [overflow](https://www.w3schools.com/cssref/pr_pos_overflow.asp) | Specifies what happens if content overflows an element's box |
| [overflow-x](https://www.w3schools.com/cssref/css3_pr_overflow-x.asp) | Specifies what to do with the left/right edges of the content if it overflows the element's content area |
| [overflow-y](https://www.w3schools.com/cssref/css3_pr_overflow-y.asp) | Specifies what to do with the top/bottom edges of the content if it overflows the element's content area |