**display: block**

Every element on a web page is a rectangular box.The **display**property determines how that rectangular box behaves. A block element is an element that takes up the fullest width available, with line breaks before and after.  
The style rules in the following example display the inline <span> elements as block-level elements:  
  
**The HTML:**

<span>First paragraph.</span>  
<span>Second paragraph.</span>  
<span>Third paragraph.</span>  
<span>Fourth paragraph.</span>  
<span>Fifth paragraph.</span>  
**The CSS:**

span {  
**display: block**;  
}

**display: inline**

An **inline**element only takes up as much width as necessary, and does not force line breaks.  
**The CSS:**

p {  
display: **inline**;  
}

Setting the display property of an element only changes how the element is displayed, not what kind of element it is. So, an inline element with **display:block** is not allowed to have other block elements inside it.

**display:none**

**display:none** hides an element, so it does not take up any space. The element will be hidden, and the page will be displayed as if the element is not there.  
  
**The HTML:**

<h1>This text will not display, as we set the value to none.</h1>  
<p>Headline of this paragraph is not displayed, as we set the value to none.</p>  
**The CSS:**

h1 {  
display: **none**;  
}

There are plenty of other display values, such as **list-item**, **table**, **table-cell**, **table-column**, **grid**, etc. Just play with values to see the difference.

**The visibility Property**

The **visibility**property specifies whether an element is visible or hidden. The most common values are **visible**and **hidden**.   
  
Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:  
**visibility:hidden** hides an element, but it will still take up the same space as before. The element will be hidden, but it will still affect the layout:  
Here is an example:  
  
**The HTML:**

<h1>This is a heading</h1>  
<div class="hidden">This text will not display in browser.</div>  
<p>The space above this paragraph is empty because the visibility of the div element is set to hidden.</p>  
**The CSS:**

div.hidden {  
**visibility: hidden**;  
}

**display:none** hides an element, without holding a place for that element.   
  
Changing **visibility: hidden;**to **display: none;**will produce the following result:

div.hidden {  
**display: none**;  
}

# Positioning Elements

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.  
  
Elements can be 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 positioning method.  
  
**Static Positioning**

HTML elements are positioned **static**by default. A static positioned element is always positioned according to the normal flow of the page.  
**The HTML:**

<p>Paragraph with no position.</p>  
<p>Paragraph with no position.</p>  
<p>Paragraph with no position.</p>  
<p>Paragraph with no position.</p>  
<p>Paragraph with no position.</p>  
<p class="position\_static">This paragraph has a static position.</p>  
**The CSS:**

p.position\_static {  
**position:static;**  
top: 30px;  
right: 5px;  
color: red;  
}

Static positioned elements are not affected by the top, bottom, left, and right properties.

**Fixed Positioning**

An element with a fixed position is positioned relative to the browser window, and will not move even if the window is scrolled.  
The position can be specified using one or more of the properties **top**, **right**, **bottom**, and **left**.   
In the example below, the paragraph is fixed to 30px from the top and 5px from the right.   
  
**The CSS:**

p.position\_fixed {  
**position: fixed;**  
top: 30px;  
right: 5px;  
color: red;  
}

Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist.  
Fixed positioned elements can overlap other elements.

**Relative Positioning**

A relative positioned element is positioned relative to its normal position.   
The properties **top**, **right**, **bottom**, and **left**can be used to specify how the rendered box will be shifted.   
  
**The CSS:**

p {  
width: 350px;  
border: 1px black solid;  
**position: fixed;**  
}  
span {  
background: green;  
color:white;  
**position: relative**;  
**top**: 150px;  
**left**: 50px;  
}

The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.  
This value cannot be used for **table cells**, **columns**, **column groups**, **rows**, **row groups**, or **captions**.

**Absolute Positioning**

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>.  
  
Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.  
Absolutely positioned elements can overlap other elements.

**Floating**

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.  
Float is often used with images, but it is also useful when working with layouts.  
The values for the float property are **left**, **right**, and **none**.   
Left and right float elements in those directions, respectively. **none**(the default) ensures that the element will not float.   
Below is an example of an image that is floated to the right.   
  
**The HTML:**

<p><img src="css\_logo.png" />  
This paragraph has an image that is floated to the <strong>right.</strong>   
It is highly recommended to add a margin to images so that the text does   
not get too close to the image. If you want your text to be easily read, you   
should always add a few pixels between words and borders, images,   
and other content.   
</p>  
**The CSS:**

img {  
**float: right**;  
}

Elements are floated horizontally, meaning that an element can only be floated left or right, not up or down.

**Elements Next to Each Other**

If you place several floating elements one after the other, they will float next to each other if there is enough room.  
As an example, in a print layout, images may be set into the page such that text wraps around them as needed.   
  
**The CSS:**

img {  
float: left;  
**width: 120px;**  
margin-right: 10px;  
}  
p {  
**width: 120px**;   
float: left;  
}

**Clearing the Float**

Elements that come after the floating element will flow around it. To avoid this, use the clear property.  
The **clear**property specifies the sides of an element where other floating elements are not allowed to be.  
  
In the example below, if we set the float property to the div, only the paragraph that comes after the div will be wrapped around the image.  
**The HTML:**

This paragraph is above the div element   
and is not affected by the float right property.   
<br /><br />  
<div class="floating">  
<img src="css\_logo.png" />  
</div>  
This paragraph comes after the div element   
and is affected by the float right property.   
<br /><br />  
This paragraph also comes after the div element  
and is affected by the float right property.  
<br /> <br />  
**The CSS:**

.floating {  
float: clear;  
}

**Using clear**  
  
Use the values **right**, **left**, and **both**to specify the sides of an element where other floating elements are not allowed to be.   
  
The default value is **none**, which allows floating elements on both sides.

**Clearing Floats**

**both**is used to clear floats coming from either direction.   
  
**The HTML:**

This paragraph is above the div element   
and is not affected by the float right property.   
<br/><br/>  
<div class="floating">  
<img src="css\_logo.png" />  
</div>  
This paragraph comes after the div element   
and is affected by the float right property.   
<br/><br class="clearing"/>  
This paragraph is out of the floating group   
and is not affected by the float right property.  
<br/> <br/>  
**The CSS:**

.floating {  
float: right;  
}  
.clearing {   
**clear**: **both**;  
}

**The overflow Property**

As discussed earlier, every element on the page is a **box**. If the height of the box is not set, the height of that box will grow as large as necessary to accommodate the content.   
  
**The HTML:**

<div>  
This text is inside the div element, which has a blue   
background color and is floated to the left. We set a specific   
height and width for the div element, and as you can see,   
the content cannot fit.   
</div>  
**The CSS:**

div {  
**width:** 150px;  
**height:** 150px;  
background-color: LightBlue;  
float: left;  
}

The CSS **overflow**property specifies the behavior that occurs when an element's content overflows the element's box.

**The overflow Property Values**

There are four values for the overflow property: **visible** (the default value), **scroll**, **hidden**, and **auto**.   
  
The value **scroll**results in clipped overflow, but a scrollbar is added, so the rest of the content may be seen.  
  
**The CSS:**

div {  
width: 150px;  
height: 150px;  
background-color: LightBlue;  
float: left;  
**overflow: scroll;**  
}

**auto and hidden**

**auto**- If overflow is clipped, a scroll-bar should be added to make it possible to see the rest of the content.  
**hidden**- The overflow is clipped, and the rest of the content will be invisible.  
  
**The CSS:**

div {  
width: 150px;  
height: 150px;  
background-color: LightBlue;  
float: left;  
**overflow: hidden;**  
}

The default value for the overflow property is **visible**.

**The z-index Property**

When elements are positioned outside the normal flow, 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).  
  
**The CSS:**

.blue {   
background-color: #8EC4D0;  
margin-bottom: 15px;  
width: 120px;  
height: 120px;  
color: #FFF;  
}  
.red {  
background-color: #FF4D4D;  
position: relative;  
width: 120px;  
height: 120px;  
color: #FFF;  
margin-top: -50px;  
margin-left: 50px;  
}

The red box overlaps the blue box, because it was placed **later**in the HTML markup.   
The **z-index** property can change this order.

**Assigning the z-index Property**

Assigning a higher z-index value to the blue div and a lower z-index value to the red div will result in the following:  
  
**The CSS:**

.blue {  
**z-index: 3;**   
position: relative;  
}  
.red {  
**z-index: 2;**  
position: relative;  
}

The z-index works **only**on positioned elements (position:absolute, position:relative, or position:fixed).