### PADDING:

*Padding* property allows you to specify how much space should appear between the content of an element and its border:

The value of this attribute should be either a length, a percentage, or the word *inherit*. If the value is *inherit*, it will have the same padding as its parent element. If a percentage is used, the percentage is of the containing box.

The following CSS properties can be used to control lists. You can also set different values for the padding on each side of the box using the following properties:

* The **padding-bottom** specifies the bottom padding of anelement.
* The **padding-top**specifies the top padding of an element.
* The**padding-left** specifies the left padding of an element.
* The **padding-right** specifies the right padding of anelement.
* The **padding**serves as shorthand for the preceding properties. Now, we will see how to use these properties with examples.

<p style="padding-bottom: 15px; border:1px solid black;"> This is a paragraph with a specified bottom padding

</p>

<p style="padding-bottom: 5%; border:1px solid black;">

This is another paragraph with a specified bottom padding in percent

</p>

**NOTE: Similarly it will be for all the other sides.**

### DIMENSONS:

We have the following properties that allow you to control the dimensions of a box.

* he **height**property is used to set the height of a box.
* The **width** property is used to set the width of abox.
* The **line-height** property is used to set the height of a line of text.
* The **max-height** property is used to set a maximum height that a box can be.
* The **min-height** property is used to set the minimum height that a box can be.
* The **max-width** property is used to set the maximum width that a box can be.
* The **min-width** property is used to set the minimum width that a box can be.

### SCROLLBARS:

CSS provides a property called *overflow*, which tells the browser what to do if the box's contents is larger than the box itself. This property can take one of the following values

<style type="text/css">

.scroll{

display:block; border: 1px solid red; padding:5px;

margin-top:5px; width:300px;

height:50px;

overflow:scroll;

}

.auto{

display:block; border:

1px solid red; padding:5px; margin-top:5px; width:300px; height:50px; overflow:auto;

}

</style>

<p>Example of scroll value:</p>

<div class="scroll">

I am going to keep lot of content here just to show you how scrollbars works if there is an overflow in an element box. This provides your horizontal as well as vertical scrollbars.

</div>

<br />

<p>Example of auto value:</p>

<div class="auto">

I am going to keep lot of content here just to show you how scrollbars works if there is an overflow in an element box. This provides your horizontal as well as vertical scrollbars.

</div>

### CURSORS

The cursor property of CSS allows you to specify the type of cursor that

should be displayed to the user.

One good usage of this property is in using images for submit buttons

on forms. By default, when a cursor hovers over a link, the cursor

changes from a pointer to a hand. However, it does not change form for

a submit button on a form. Therefore, whenever someone hovers over an

image that is a submit button, it provides a visual clue that the image is clickable.

The following table shows the possible values for the cursor property:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Auto | Shape of the cursor depends on the context area it is over. For example, an ‘I’ over text, a ‘hand’ over a link, and so on. |
| crosshair | A crosshair or plus sign. |
| Default | An arrow. |
| Pointer | A pointing hand (in IE 4 this value is hand). |
| Move | The ‘I’ bar. |
| e-resize | The cursor indicates that an edge of a box is to be moved right (east). |
| ne-resize | The cursor indicates that an edge of a box is to be moved up and right (north/east). |
| nw- resize | The cursor indicates that an edge of a box is to be moved up and left (north/west). |
| n-resize | The cursor indicates that an edge of a box is to be moved up (north). |
| se-resize | The cursor indicates that an edge of a box is to be moved down and right (south/east). |
| sw- resize | The cursor indicates that an edge of a box is to be moved down and left (south/west). |
| s-resize | The cursor indicates that an edge of a box is to be moved down (south). |
| w-resize | The cursor indicates that an edge of a box is to be moved left (west). |
| text | The I bar. |
| wait | An hour glass. |
| help | A question mark or balloon, ideal for use over help buttons. |
| <url> | The source of a cursor image file. |

**NOTE:** You should try to use only these values to add helpful information for users, and in places, they would expect to see that cursor. For example, using the crosshair when someone hovers over a link can confuse the visitors.

Here is an example:

<p>Move the mouse over the words to see the cursor change:</p>

<div style="cursor:auto">Auto</div>

<div style="cursor:crosshair">Crosshair</div>

<div style="cursor:default">Default</div>

<div style="cursor:pointer">Pointer</div>

<div style="cursor:move">Move</div>

<div style="cursor:e-resize">e-resize</div>

<div style="cursor:ne-resize">ne-resize</div>

<div style="cursor:nw-resize">nw-resize</div>

<div style="cursor:n-resize">n-resize</div>

<div style="cursor:se-resize">se-resize</div>

<div style="cursor:sw-resize">sw-resize</div>

<div style="cursor:s-resize">s-resize</div>

<div style="cursor:w-resize">w-resize</div>

<div style="cursor:text">text</div>

<div style="cursor:wait">wait</div>

<div style="cursor:help">help</div>

### Relative positioning:

Relative positioning changes the position of the HTML element relative to where it normally appears. So "left:20" adds 20 pixels to the element's LEFT position.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in an HTML document.

* Move Left - Use a negative value for *left*.
* Move Right - Use a positive value for *left*.
* Move Up - Use a negative value for *top*.
* Move Down - Use a positive value for *top*.

**NOTE:** You can use the *bottom* or *right* values as well in the same way as *top* and *left*.

Here is an example:

<div style="position:relative;left:80px;top:2px; background-color:yellow;">

This div has relative positioning.

</div>

### Absolute Positioning:

An element with **position: absolute** is positioned at the specified coordinates relative to your screen top-left corner.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in HTML document.

* Move Left - Use a negative value for *left*.
* Move Right - Use a positive value for *left*.
* Move Up - Use a negative value for *top*.
* Move Down - Use a positive value for *top*.

**NOTE:** You can use *bottom* or *right* values as well in the same way as top and left.

Here is an example:

<div style="position:absolute;left:80px;top:20px; background-color:yellow;">

This div has absolute positioning.

</div>

### Fixed Positioning:

Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling. Specified coordinates will be relative to the browser window.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

* Move Left - Use a negative value for *left*.
* Move Right - Use a positive value for *left*.
* Move Up - Use a negative value for *top*.
* Move Down - Use a positive value for *top*.

**NOTE:** You can use *bottom* or *right* values as well in the same way as *top* and *left*.

<div style="position:fixed;left:80px;top:20px;

background-color:yellow;">

This div has fixed positioning.

</div>

### Positioning the division inside a division

<!DOCTYPE html>

<html>

<head>

<title>Positioning!</title>

<style type="text/css">

div.relative{

width:400px;

position: relative;

height: 200px;

border:3px solid red;

}

div.absolute{

width:150px;

height:50px;

border:1px solid red;

position: absolute;

right:0px;

bottom:0px;

}

</style>

</head>

<body>

<div class="relative"> This is the main divison

<div class="absolute"> This is the sub division

</div>

</div>

</body>

</html>

### Example for all the positions

<!DOCTYPE html>

<html>

<head>

<title></title>

<style type="text/css">

.static{

position: static;

border:1px solid blue;

}

.relative{

position: relative;

left:30px;

border: 1px solid red;

top:20px;

}

.absolute{

position: absolute;

left:30px;

top:20px;

}

.fixed{

position: fixed;

bottom: 0;

right: 0;

width:300px;

border: 3px solid green;

background-color: red;

color: white;

}

.sticky{

position: -webkit-sticky; /\* safari\*/

position: sticky;

top:0px;

background-color: yellow;

color: red;

}

</style>

</head>

<body>

<div class="static">

this div element has position static

</div>

<div class="relative">

this div element is positioned in relative way!

</div>

<div class="absolute">

This is absolute positioning.

</div>

<div class="fixed">

This is fixed positioning.

</div>

<div class="sticky">

this will use sticky positioning!

</div>

</body>

</html>

Pseudo classes and elements

Pseudo-classes are used to add special effects to some selectors. You do not need to use JavaScript or any other script to use those effects. A simple syntax of pseudo-classes is as follows:

selector:pseudo-class {property: value}

CSS classes can also be used with pseudo-classes:

selector.class:pseudo-class {property: value}

The most commonly used pseudo-classes are as follows:

|  |  |
| --- | --- |
| **Value** | **Description** |
| :link | Use this class to add special style to an unvisited link. |
| :visited | Use this class to add special style to a visited link. |
| :hover | Use this class to add special style to an element when you mouse over it. |
| :active | Use this class to add special style to an active element. |
| :focus | Use this class to add special style to an element while the element has focus. |
| :first- child | Use this class to add special style to an element that is the first child of some other element. |
| :last-child | Use this class to add special style to an element that is the last child of some other element. |
| :nth-child | Use this class to add special style to an element that is the nth child of some other element. |
|  |  |

### PSEUDO ELEMENTS:

pseudo-elements are used to add special effects to some selectors. You do not need to use JavaScript or any other script to use those effects. A simple syntax of pseudo-element is as follows:

selector:pseudo-element {property: value}

CSS classes can also be used with the pseudo-elements:

selector.class:pseudo-element {property: value}

The most commonly used pseudo-elements are as follows:

|  |  |
| --- | --- |
| **Value** | **Description** |
| ::first-line | Use this element to add special styles to the first line of the text in a selector. |
| ::first-letter | Use this element to add special style to the first letter of the text in a selector. |
| ::before | Use this element to insert some content before an element. |
| ::after | Use this element to insert some content after an element. |
| ::selection | Use this element to select the data required from the document. |
| ::marker | Used to style the markers of the list. |

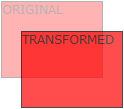
## CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

With the CSS transform property you can use the following 2D transformation methods:

* translate()
* rotate()
* scaleX()
* scaleY()
* scale()
* skewX()
* skewY()
* skew()
* matrix()

## The translate() Method



The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: translate(50px,100px); /\* IE 9 \*/

transform: translate(50px,100px); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h1>The translate() Method</h1>

<p>The translate() method moves an element from its current position:</p>

<div>

This div element is moved 50 pixels to the right, and 100 pixels down from its current position.

</div>

</body>

</html>

## The rotate() Method



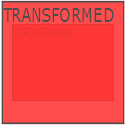
The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the <div> element clockwise with 20 degrees:

div {  
  transform: rotate(20deg);  
}

Using negative values will rotate the element counter-clockwise.

## The scale() Method



The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the <div> element to be two times of its original width, and three times of its original height:

div {  
  transform: scale(2, 3);  
}

The following example decreases the <div> element to be half of its original width and height:

div {  
  transform: scale(0.5, 0.5);  
}

## The scaleX() Method

The scaleX() method increases or decreases the width of an element.

The following example increases the <div> element to be two times of its original width:

div {  
  transform: scaleX(2);  
}

The following example decreases the <div> element to be half of its original width:

div {  
  transform: scaleX(0.5);  
}

## The scaleY() Method

The scaleY() method increases or decreases the height of an element.

The following example increases the <div> element to be three times of its original height:

div {  
  transform: scaleY(3);  
}

The following example decreases the <div> element to be half of its original height:

div {  
  transform: scaleY(0.5);  
}

## The skewX() Method

The skewX() method skews an element along the X-axis by the given angle.

The following example skews the <div> element 20 degrees along the X-axis:

div {  
  transform: skewX(20deg);  
}

## The skewY() Method

The skewY() method skews an element along the Y-axis by the given angle.

The following example skews the <div> element 20 degrees along the Y-axis:

div {  
  transform: skewY(20deg);  
}

## The skew() Method

The skew() method skews an element along the X and Y-axis by the given angles.

The following example skews the <div> element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

div {  
  transform: skew(20deg, 10deg);  
}

If the second parameter is not specified, it has a zero value. So, the following example skews the <div> element 20 degrees along the X-axis:

div {  
  transform: skew(20deg);  
}

## The matrix() Method



The matrix() method combines all the 2D transform methods into one.

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow: matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

div {  
  transform: matrix(1, -0.3, 0, 1, 0, 0);  
}

## CSS Transitions

CSS transitions allows you to change property values smoothly, over a given duration.

you will learn about the following properties:

* transition
* transition-delay
* transition-duration
* transition-property
* transition-timing-function

## How to Use CSS Transitions?

To create a transition effect, you must specify two things:

* the CSS property you want to add an effect to
* the duration of the effect

**Note:** If the duration part is not specified, the transition will have no effect, because the default value is 0.

The following example shows a 100px \* 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds.

The transition effect will start when the specified CSS property (width) changes value.

Now, let us specify a new value for the width property when a user mouses over the <div> element:

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s;

}

div:hover {

width: 300px;

}

</style>

</head>

<body>

<h1>The transition Property</h1>

<p>Hover over the div element below, to see the transition effect:</p>

<div></div>

<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>

</body>

</html>

## Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

div {  
  transition: width 2s, height 4s;  
}

## Specify the Speed Curve of the Transition

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

* ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
* linear - specifies a transition effect with the same speed from start to end
* ease-in - specifies a transition effect with a slow start
* ease-out - specifies a transition effect with a slow end
* ease-in-out - specifies a transition effect with a slow start and end
* cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

The following example shows some of the different speed curves that can be used:

#div1 {transition-timing-function: linear;}  
#div2 {transition-timing-function: ease;}  
#div3 {transition-timing-function: ease-in;}  
#div4 {transition-timing-function: ease-out;}  
#div5 {transition-timing-function: ease-in-out;}

## Delay the Transition Effect

The transition-delay property specifies a delay (in seconds) for the transition effect.

The following example has a 1 second delay before starting:

div {  
  transition-delay: 1s;  
}

## Transition + Transformation

The following example adds a transition effect to the transformation:

div {  
  transition: width 2s, height 2s, transform 2s;  
}

**CSS ANIMATION….**

* @keyframes
* animation-name
* animation-duration
* animation-delay
* animation-iteration-count
* animation-direction
* animation-timing-function
* animation-fill-mode
* animation

What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

@keyframes example {

from {background-color: red;}

to {background-color: yellow;}

}

</style>

</head>

<body>

<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>

<div></div>

<p><b>Note:</b> When an animation is finished, it changes back to its original style.</p>

</body>

</html>

**Note:** The animation-duration property defines how long time an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

@keyframes example {

0% {background-color: red;}

25% {background-color: yellow;}

50% {background-color: blue;}

100% {background-color: green;}

}

</style>

</head>

<body>

<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>

<div></div>

</body>

</html>

The following example will change both the background-color and the position of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>

<div></div>

</body>

</html>

The animation-delay property specifies a delay for the start of an animation.

The following example has a 2 seconds delay before starting the animation:

div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-delay: 2s;  
}

Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for N seconds.

In the following example, the animation will start as if it had already been playing for 2 seconds:

div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-delay: -2s;  
}

## Set How Many Times an Animation Should Run

The animation-iteration-count property specifies the number of times an animation should run.

The following example will run the animation 3 times before it stops:

div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-iteration-count: 3;  
}

The following example uses the value "infinite" to make the animation continue for ever:

div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-iteration-count: infinite;  
}

Run Animation in Reverse Direction or Alternate Cycles

The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles.

The animation-direction property can have the following values:

* normal - The animation is played as normal (forwards). This is default
* reverse - The animation is played in reverse direction (backwards)
* alternate - The animation is played forwards first, then backwards
* alternate-reverse - The animation is played backwards first, then forwards

The following example will run the animation in reverse direction (backwards):

div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-direction: reverse;  
}

Specify the fill-mode For an Animation

CSS animations do not affect an element before the first keyframe is played or after the last keyframe is played. The animation-fill-mode property can override this behavior.

The animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).

The animation-fill-mode property can have the following values:

* none - Default value. Animation will not apply any styles to the element before or after it is executing
* forwards - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
* backwards - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
* both - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

The following example lets the <div> element retain the style values from the last keyframe when the animation ends:

div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  position: relative;  
  animation-name: example;  
  animation-duration: 3s;  
  animation-fill-mode: forwards;  
}

CSS3 Introduced Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

* width and height of the viewport
* width and height of the device
* orientation (is the tablet/phone in landscape or portrait mode?)
* resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: pink;

}

@media screen and (min-width: 480px) {

body {

background-color: lightgreen;

}

}

</style>

</head>

<body>

<h1>Resize the browser window to see the effect!</h1>

<p>The media query will only apply if the media type is screen and the viewport is 480px wide or wider.</p>

</body>

</html>

Example:2

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: tan;

color: black;

}

/\* On screens that are 992px wide or less, the background color is blue \*/

@media screen and (max-width: 992px) {

body {

background-color: blue;

color: white;

}

}

/\* On screens that are 600px wide or less, the background color is olive \*/

@media screen and (max-width: 600px) {

body {

background-color: olive;

color: white;

}

}

</style>

</head>

<body>

<h1>Resize the browser window to see the effect!</h1>

<p>By default, the background color of the document is "tan". If the screen size is 992px or less, the color will change to "blue". If it is 600px or less, it will change to "olive".</p>

</body>

</html>