CSS:

* CSS stands for Cascading Style Sheet.
* CSS is used to design HTML tags.
* CSS is a widely used language on the web.
* HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

CSS Example:

<!DOCTYPE**>**

**<html>**

**<head>**

**<style>**

h1{

color:white;

background-color:red;

padding:5px;

}

p{

color:blue;

}

**</style>**

**</head>**

**<body>**

**<h1>**Write Your First CSS Example**</h1>**

**<p>**This is Paragraph.**</p>**

**</body>**

**</html>**

# **What is CSS**

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

## What does CSS do

* You can add new looks to your old HTML documents.
* You can completely change the look of your website with only a few changes in CSS code.

## Why use CSS

These are the three major benefits of CSS:

## 1) Solves a big problem

Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem. It was a W3C recommendation.

## 2) Saves a lot of time

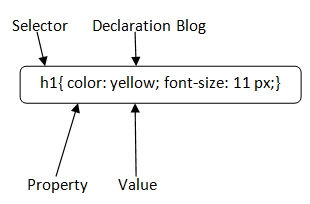
CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

## 3) Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

# **CSS Syntax**

A CSS rule set contains a selector and a declaration block.



**Selector:** Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> etc.

**Declaration Block:** The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

1. color: yellow;
2. font-size: 11 px;

Each declaration contains a property name and value, separated by a colon.

**Property:** A Property is a type of attribute of HTML element. It could be color, border etc.

**Value:** Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

Selector{Property1: value1; Property2: value2; ..........;}

# **CSS Selector**

**CSS selectors** are used to select the content you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1. CSS Element Selector
2. CSS Id Selector
3. CSS Class Selector
4. CSS Universal Selector
5. CSS Group Selector

## 1) CSS Element Selector

The element selector selects the HTML element by name.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

p{

    text-align: center;

    color: blue;

}

**</style>**

**</head>**

**<body>**

**<p>**This style will be applied on every paragraph.**</p>**

**<p** id="para1"**>**Me too!**</p>**

**<p>**And me!**</p>**

**</body>**

**</html>**

## 2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

Let?s take an example with the id "para1".

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

#para1{

    text-align: center;

    color: blue;

}

**</style>**

**</head>**

**<body>**

**<p>**This style will be applied on every paragraph.**</p>**

**<p** id="para1"**>**Me too!**</p>**

**<p>**And me!**</p>**

**</body>**

**</html>**

## 3) CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.

#### **Note: A class name should not be started with a number.**

Let's take an example with a class "center".

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

para1{

    text-align: center;

    color: blue;

}

**</style>**

**</head>**

**<body>**

**<p>**This style will be applied on every paragraph.**</p>**

**<p** class="para1"**>**Me too!**</p>**

**<p>**And me!**</p>**

**</body>**

**</html>**

## CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector.

Let's see an example.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

p.center {

    text-align: center;

    color: blue;

}

**</style>**

**</head>**

**<body>**

**<h1** class="center"**>**This heading is not affected**</h1>**

**<p** class="center"**>**This paragraph is blue and center-aligned.**</p>**

**</body>**

**</html>**

## 4) CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

\* {

   color: green;

   font-size: 20px;

}

**</style>**

**</head>**

**<body>**

**<h2>**This is heading**</h2>**

**<p>**This style will be applied on every paragraph.**</p>**

**<p** id="para1"**>**Me too!**</p>**

**<p>**And me!**</p>**

**</body>**

**</html>**

## 5) CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Let's see the CSS code without group selector.

h1 {

    text-align: center;

    color: blue;

}

h2 {

    text-align: center;

    color: blue;

}

p {

    text-align: center;

    color: blue;

}

As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

h1,h2,p {

    text-align: center;

    color: blue;

}

Let's see the full example of CSS group selector.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

h1, h2, p {

    text-align: center;

    color: blue;

}

**</style>**

**</head>**

**<body>**

**<h1>**Hello iHub.com**</h1>**

**<h2>**Hello iHub.com (In smaller font)**</h2>**

**<p>**This is a paragraph.**</p>**

**</body>**

**</html>**

# **How to add CSS**

CSS is added to HTML pages to format the document according to information in the style sheet. There are three ways to insert CSS in HTML documents.

1. Inline CSS
2. Internal CSS
3. External CSS

## 1) Inline CSS

Inline CSS is used to apply CSS on a single line or element.

For example:

**<p** style="color:blue"**>**Hello CSS**</p>**

## 2) Internal CSS

Internal CSS is used to apply CSS on a single document or page. It can affect all the elements of the page. It is written inside the style tag within head section of html.

For example:

**<style>**

p{color:blue}

**</style>**

## 3) External CSS

External CSS is used to apply CSS on multiple pages or all pages. Here, we write all the CSS code in a css file. Its extension must be .css for example style.css.

For example:

p{color:blue}

You need to link this style.css file to your html pages like this:

**<link** rel="stylesheet" type="text/css" href="style.css"**>**

The link tag must be used inside head section of html.

# **Inline CSS**

We can apply CSS in a single element by inline CSS technique.

The inline CSS is also a method to insert style sheets in HTML document. This method mitigates some advantages of style sheets so it is advised to use this method sparingly.

If you want to use inline CSS, you should use the style attribute to the relevant tag.

Syntax:

**<htmltag** style="cssproperty1:value; cssproperty2:value;"**>** **</htmltag>**

Example:

**<h2** style="color:red;margin-left:40px;"**>**Inline CSS is applied on this heading.**</h2>**

**<p>**This paragraph is not affected.**</p>**

## Disadvantages of Inline CSS

* You cannot use quotations within inline CSS. If you use quotations the browser will interpret this as an end of your style value.
* These styles cannot be reused anywhere else.
* These styles are tough to be edited because they are not stored at a single place.
* It is not possible to style pseudo-codes and pseudo-classes with inline CSS.
* Inline CSS does not provide browser cache advantages.

# **Internal CSS**

The internal style sheet is used to add a unique style for a single document. It is defined in <head> section of the HTML page inside the <style> tag.

Example:

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

body {

    background-color: linen;

}

h1 {

    color: red;

    margin-left: 80px;

}

**</style>**

**</head>**

**<body>**

**<h1>**The internal style sheet is applied on this heading.**</h1>**

**<p>**This paragraph will not be affected.**</p>**

**</body>**

**</html>**

# **External CSS**

The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file.

It uses the <link> tag on every pages and the <link> tag should be put inside the head section.

Example:

**<head>**

**<link** rel="stylesheet" type="text/css" href="mystyle.css"**>**

**</head>**

The external style sheet may be written in any text editor but must be saved with a .css extension. This file should not contain HTML elements.

Let's take an example of a style sheet file named "mystyle.css".

*File: mystyle.css*

body {

    background-color: lightblue;

}

h1 {

    color: navy;

    margin-left: 20px;

}

Note: You should not use a space between the property value and the unit. For example: It should be margin-left:20px not margin-left:20 px.

# **CSS Comments**

CSS comments are generally written to explain your code. It is very helpful for the users who reads your code so that they can easily understand the code.

Comments are ignored by browsers.

Comments are single or multiple lines statement and written within /\*............\*/ .

# CSS Properties

# **CSS Background**

CSS background property is used to define the background effects on element. There are 5 CSS background properties that affects the HTML elements:

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

## 1) CSS background-color

The background-color property is used to specify the background color of the element.

You can set the background color like this:

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

h2,p{

    background-color: #b0d4de;

}

**</style>**

**</head>**

**<body>**

**<h2>**My first CSS page.**</h2>**

**<p>**Hello iHub. This is an example of CSS background-color.**</p>**

**</body>**

**</html>**

## 2) CSS background-image

The background-image property is used to set an image as a background of an element. By default the image covers the entire element. You can set the background image for a page like this

**<html>**

**<head>**

**<style>**

body {

background-image: url("paper1.gif");

margin-left:100px;

}

**</style>**

**</head>**

**<body>**

**<h1>**Hello iHub.com**</h1>**

**</body>**

**</html>**

**Note**: The background image should be chosen according to text color. The bad combination of text and background image may be a cause of poor designed and not readable webpage.

## 3) CSS background-repeat

By default, the background-image property repeats the background image horizontally and vertically. Some images are repeated only horizontally or vertically.

The background looks better if the image repeated horizontally only.

**background-repeat: repeat-x;**

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

body {

    background-image: url("gradient\_bg.png");

    background-repeat: repeat-x;

}

**</style>**

**</head>**

**<body>**

**<h1>**Hello iHub.com**</h1>**

**</body>**

**</html>**

**background-repeat: repeat-y;**

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

body {

    background-image: url("gradient\_bg.png");

    background-repeat: repeat-y;

}

**</style>**

**</head>**

**<body>**

**<h1>**Hello iHub.com**</h1>**

**</body>**

**</html>**

## 4) CSS background-attachment

The background-attachment property is used to specify if the background image is fixed or scroll with the rest of the page in browser window. If you set fixed the background image then the image will not move during scrolling in the browser. Let’s take an example with fixed background image.

<!DOCTYPE html>

<html>

<head>

<style>

body {

background: white url('bbb.gif');

background-repeat: no-repeat;

background-attachment: fixed;

margin-left:200px;

}

</style>

</head>

<body>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>If you do not see any scrollbars, Resize the browser window.</p>

</body>

</html>

## 5) CSS background-position

The background-position property is used to define the initial position of the background image. By default, the background image is placed on the top-left of the webpage.

You can set the following positions:

1. center
2. top
3. bottom
4. left
5. right

<!DOCTYPE html>

<html>

<head>

<style>

body {

background: white url('good-morning.jpg');

background-repeat: no-repeat;

background-attachment: fixed;

background-position: center;

}

</style>

</head>

<body>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>This is a fixed background-image. Scroll down the page.</p>

<p>If you do not see any scrollbars, Resize the browser window.</p>

</body>

</html>

# **CSS Border**

The CSS border is a shorthand property used to set the border on an element.

The [CSS](https://www.javatpoint.com/css-tutorial) border properties are used to specify the style, color and size of the border of an element. The CSS border properties are given below

* border-style
* border-color
* border-width
* border-radius

1) CSS border-style

* The Border style property is used to specify the border type which you want to display on the web page.
* There are some border style values which are used with border-style property to define a border.

|  |  |
| --- | --- |
| **Value** | **Description** |
| none | It doesn't define any border. |
| dotted | It is used to define a dotted border. |
| dashed | It is used to define a dashed border. |
| solid | It is used to define a solid border. |
| double | It defines two borders with the same border-width value. |
| groove | It defines a 3d grooved border. Effect is generated according to border-color value. |
| ridge | It defines a 3d ridged border. Effect is generated according to border-color value. |
| inset | It defines a 3d inset border. Effect is generated according to border-color value. |
| outset | It defines a 3d outset border. Effect is generated according to border-color value. |

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

p.none {border-style: none;}

p.dotted {border-style: dotted;}

p.dashed {border-style: dashed;}

p.solid {border-style: solid;}

p.double {border-style: double;}

p.groove {border-style: groove;}

p.ridge {border-style: ridge;}

p.inset {border-style: inset;}

p.outset {border-style: outset;}

p.hidden {border-style: hidden;}

**</style>**

**</head>**

**<body>**

**<p** class="none"**>**No border.**</p>**

**<p** class="dotted"**>**A dotted border.**</p>**

**<p** class="dashed"**>**A dashed border.**</p>**

**<p** class="solid"**>**A solid border.**</p>**

**<p** class="double"**>**A double border.**</p>**

**<p** class="groove"**>**A groove border.**</p>**

**<p** class="ridge"**>**A ridge border.**</p>**

**<p** class="inset"**>**An inset border.**</p>**

**<p** class="outset"**>**An outset border.**</p>**

**<p** class="hidden"**>**A hidden border.**</p>**

**</body>**

**</html>**

## 2) CSS border-width

The border-width property is used to set the border's width. It is set in pixels. You can also use the one of the three pre-defined values, thin, medium or thick to set the width of the border.

#### **Note: The border-width property is not used alone. It is always used with other border properties like "border-style" property to set the border first otherwise it will not work.**

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

p.one {

    border-style: solid;

    border-width: 5px;

}

p.two {

    border-style: solid;

    border-width: medium;

}

p.three {

    border-style: solid;

    border-width: 1px;

}

**</style>**

**</head>**

**<body>**

**<p** class="one"**>**Write your text here.**</p>**

**<p** class="two"**>**Write your text here.**</p>**

**<p** class="three"**>**Write your text here.**</p>**

**</body>**

**</html>**

## 3) CSS border-color

There are three methods to set the color of the border.

* Name: It specifies the color name. For example: "red".
* RGB: It specifies the RGB value of the color. For example: "rgb(255,0,0)".
* Hex: It specifies the hex value of the color. For example: "#ff0000".

There is also a border color named "transparent". If the border color is not set it is inherited from the color property of the element.

#### **Note: The border-color property is not used alone. It is always used with other border properties like "border-style" property to set the border first otherwise it will not work.**

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

p.one {

    border-style: solid;

    border-color: red;

}

p.two {

    border-style: solid;

    border-color: #98bf21;

}

**</style>**

**</head>**

**<body>**

**<p** class="one"**>**This is a solid red border**</p>**

**<p** class="two"**>**This is a solid green border**</p>**

**</body>**

**</html>**

# **CSS border-radius property**

This CSS property sets the rounded borders and provides the rounded corners around an element, tags, or div. It defines the radius of the corners of an element.

It is shorthand for **border top-left-radius, border-top-right-radius, border-bottom-right-radius** and **border-bottom-left-radius**. It gives the rounded shape to the corners of the border of an element. We can specify the border for all four corners of the box in a single declaration using the border-radius. The values of this property can be defined in percentage or length units.

This CSS property includes the properties that are tabulated as follows:

|  |  |
| --- | --- |
| **Property** | **Description** |
| **border-top-left-radius** | It is used to set the border-radius for the top-left corner |
| **border-top-right-radius** | It is used to set the border-radius for the top-right corner |
| **border-bottom-right-radius** | It is used to set the border-radius for the bottom-right corner |
| **border-bottom-left-radius** | It is used to set the border-radius for the bottom-left corner |

If the bottom-left value is omitted, then it will be same as the top-right. If the value of bottom-right is eliminated, then it will be same as the top-left. Similarly, if top-right is eliminated, then it will be the same as top-left.

Let's see what happens when we provide a single value, two values, three values, and four values to this property.

* If we provide a single value **(**such as **border-radius: 30px;)** to this property, it will set all corners to the same value.
* When we specify two values **(**such as **border-radius: 20% 10% ;)**, then the first value will be used for the top-left and bottom-right corners, and the second value will be used for the top-right and bottom-left corners.
* When we use three values **(**such as **border-radius: 10% 30% 20%;)** then the first value will be used for the top-left corner, the second value will be applied on top-right, and bottom-left corners and the third value will be applied to the bottom-right corner.
* Similarly, when this property has four values **(border-radius: 10% 30% 20% 40%;)** then the first value will be the radius of top-left, the second value will be used for the top-right, the third value will be applied on bottom-right, and the fourth value is used for bottom-left.

### **Syntax**

1. border-radius: 1-4 length | %  / 1-4 length | % | inherit | initial;

### **Property values**

**length:** It defines the shape of the corners. It denotes the size of the radius using length values. Its default value is 0. It does not allow negative values.

**percentage:** It denotes the size of the radius in percentage. It also does not allow negative values.

### **Example**

<!DOCTYPE html**>**

**<html>**

**<head>**

**<title>** CSS border-radius **</title>**

**<style>**

div {

padding: 50px;

margin: 20px;

border: 6px ridge red;

width: 300px;

float: left;

height: 150px;

}

p{

font-size: 25px;

}

#one {

border-radius: 90px;

background: lightgreen;

}

#two {

border-radius: 25% 10%;

background: orange;

}

#three {

border-radius: 35px 10em 10%;

background: cyan;

}

#four {

border-radius: 50px 50% 50cm 50em;

background: lightblue;

}

**</style>**

**</head**

**<body>**

**<div** id = "one"**>**

**<h2>** Welcome to the iHub.com **</h2>**

**<p>** border-radius: 90px; **</p>**

**</div>**

**<div** id = "two"**>**

**<h2>** Welcome to the iHub.com **</h2>**

**<p>** border-radius: 25% 10%; **</p>**

**</div>**

**<div** id = "three"**>**

**<h2>** Welcome to the iHub.com **</h2>**

**<p>** border-radius: 35px 10em 10%; **</p>**

**</div>**

**<div** id = "four"**>**

**<h2>**Welcome to the iHub.com**</h2>**

**<p>**border-radius: 50px 50% 50cm 50em;**</p>**

**</div>**

**</body>**

**</html>**

Now, let's see the **border-radius** for specific corners.

### **Example- border-top-left-radius**

It sets the border radius for the top-left corner.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<title>** CSS border-radius **</title>**

**<style>**

#left {

border-top-left-radius: 250px;

background: lightgreen;

padding: 50px;

border: 6px ridge red;

width: 300px;

height: 200px;

font-size: 25px;

}

**</style>**

**</head>**

**<body>**

**<center>**

**<div** id = "left"**>**

**<h2>**Welcome to the iHub.com**</h2>**

**<p>**border-top-left-radius: 250px;**</p>**

**</div>**

**</center>**

**</body>**

**</html>**

### **Example- border-top-right-radius**

It sets the border-radius for the top-right corner.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

#left {

border-top-right-radius: 50%;

background: lightgreen;

padding: 50px;

border: 6px ridge red;

width: 300px;

height: 200px;

font-size: 25px;

}

**</style>**

**</head>**

**<body>**

**<center>**

**<div** id = "left"**>**

**<h2>**Welcome to the iHub.com**</h2>**

**<p>**border-top-right-radius: 50%;**</p>**

**</div>**

**</center>**

**</body>**

**</html>**

### **Assignment**: Try in the similar way for “**border-bottom-right-radius”** and “**border-bottom-left-radius**”

We can specify separate **horizontal** and **vertical** values by using the slash (/) symbol. The values before the slash (/) is used for the horizontal radius and the values after the slash (/) are for the vertical radius.

There is an example given below using the slash (/) symbol.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

div{

padding: 50px;

border: 6px ridge red;

width: 300px;

margin: 20px;

font-weight: bold;

height: 175px;

float: left;

font-size: 25px;

}

#one {

border-radius: 10% / 50%;

background: lightgreen;

}

#two {

border-radius: 120px / 100px 10px;

background: lightblue;

}

#three {

border-radius: 50% 10em / 10% 20em;

background: lightpink;

}

#four {

border-radius: 100px 10em 120px / 30%;

background: cyan;

}

**</style>**

**</head>**

**<body>**

**<center>**

**<div** id = "one"**>**

**<h2>**Welcome to the iHub.com**</h2>**

**<p>**border-radius: 10% / 50%; **</p>**

**</div>**

**<div** id = "two"**>**

**<h2>**Welcome to the iHub.com**</h2>**

**<p>**border-radius: 120px / 100px 10px; **</p>**

**</div>**

**<div** id = "three"**>**

**<h2>**Welcome to the iHub.com**</h2>**

**<p>**border-radius: 50% 10em / 10% 20em; **</p>**

**</div>**

**<div** id = "four"**>**

**<h2>**Welcome to the iHub.com**</h2>**

**<p>**border-radius: 100px 10em 120px / 30%; **</p>**

**</div>**

**</center>**

**</body>**

**</html>**

# **CSS border-collapse property**

This CSS property is used to set the border of the table cells and specifies whether the table cells share the separate or common border.

This property has two main values that are **separate** and **collapse**. When it is set to the value **separate**, the distance between the cells can be defined using the **border-spacing** property. When the **border-collapse** is set to the value **collapse**, then the **inset** value of **border-style** property behaves like **groove**, and the **outset** value behaves like **ridge**.

### **Syntax**

1. border-collapse: separate | collapse | initial | inherit;

The values of this CSS property are defined as follows.

### **Property Values**

**separate:** It is the default value that separates the border of the table cell. Using this value, each cell will display its own border.

**collapse:** This value is used to collapse the borders into a single border. Using this, two adjacent table cells will share a border. When this value is applied, the **border-spacing** property does not affect.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Now, let's understand this CSS property by using some examples. In the first example, we are using the **separate** value of the **border-collapse** property. In the second example, we are using the **collapse** value of the **border-collapse** property.

### **Example - Using separate value**

With this value, we can use the **border-spacing** property to set the distance between the adjacent table cells.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<title>** border-collapse property **</title>**

**<style>**

table{

border: 2px solid blue;

text-align: center;

font-size: 20px;

width: 80%;

height: 50%;

}

th{

border: 5px solid red;

background-color: yellow;

}

td{

border: 5px solid violet;

background-color: cyan;

}

#t1 {

border-collapse: separate;

}

**</style>**

**</head>**

**<body>**

**<h1>** The border-collapse Property **</h1>**

**<h2>** border-collapse: separate; **</h2>**

**<table** id = "t1"**>**

**<tr>**

**<th>** First\_Name **</th>**

**<th>** Last\_Name **</th>**

**<th>** Subject **</th>**

**<th>** Marks **</th>**

**</tr>**

**<tr>**

**<td>** James **</td>**

**<td>** Gosling **</td>**

**<td>** Maths **</td>**

**<td>** 92 **</td>**

**</tr>**

**<tr>**

**<td>** Alan **</td>**

**<td>** Rickman **</td>**

**<td>** Maths **</td>**

**<td>** 89 **</td>**

**</tr>**

**<tr>**

**<td>** Sam **</td>**

**<td>** Mendes **</td>**

**<td>** Maths **</td>**

**<td>** 82 **</td>**

**</tr>**

**</table>**

**</body>**

**</html>**

# **CSS border-spacing property**

This CSS property is used to set the distance between the borders of the adjacent cells in the table. It applies only when the **border-collapse** property is set to **separate**. There will not be any space between the borders if the [**border-collapse**](https://www.javatpoint.com/css-border-collapse-property) is set to **collapse**.

It can be defined as one or two values for determining the vertical and horizontal spacing.

* When only one value is specified, then it sets both horizontal and vertical spacing.
* When we use the two-value syntax, then the first one is used to set the horizontal spacing (i.e., the space between the adjacent columns), and the second value sets the vertical spacing (i.e., the space between the adjacent rows).

### **Syntax**

1. border-spacing: length | initial | inherit;

### **Property Values**

The values of this CSS property are defined as follows.

**length:** This value sets the distance between the borders of the adjacent table cells in px, cm, pt, etc. Negative values are not allowed.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Let's understand this CSS property by using some examples. In the first example, we are using the single value of the **border-spacing** property, and in the second example, we are using two values of the **border-spacing** property.

### **Example**

Here, we are using the single value of the **border-spacing** property. The **border-collapse** property is set to **separate**, and the value of the **border-spacing** is set to **45px**.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<title>** border-spacing property **</title>**

**<style>**

table{

border: 2px solid blue;

text-align: center;

font-size: 20px;

background-color: lightgreen;

}

th{

border: 5px solid red;

background-color: yellow;

}

td{

border: 5px solid violet;

background-color: cyan;

}

#space{

border-collapse: separate;

border-spacing: 45px;

}

**</style>**

**</head>**

**<body>**

**<h1>** The border-spacing Property **</h1>**

**<h2>** border-spacing: 45px; **</h2>**

**<table** id = "space"**>**

**<tr>**

**<th>** First\_Name **</th>**

**<th>** Last\_Name **</th>**

**<th>** Subject **</th>**

**<th>** Marks **</th>**

**</tr>**

**<tr>**

**<td>** James **</td>**

**<td>** Gosling **</td>**

**<td>** Maths **</td>**

**<td>** 92 **</td>**

**</tr>**

**<tr>**

**<td>** Alan **</td>**

**<td>** Rickman **</td>**

**<td>** Maths **</td>**

**<td>** 89 **</td>**

**</tr>**

**<tr>**

**<td>** Sam **</td>**

**<td>** Mendes **</td>**

**<td>** Maths **</td>**

**<td>** 82 **</td>**

**</tr>**

**</table>**

**</body>**

**</html>**

### **Example**

Here, we are using two values of the **border-spacing** property. The **border-collapse** property is set to **separate**, and the value of the **border-spacing** is set to **20pt 1em**. The first value, i.e., **20pt** sets the horizontal spacing, and the value **1em** set the vertical spacing.

# **CSS Display**

CSS display is the most important property of CSS which is used to control the layout of the element. It specifies how the element is displayed.

Every element has a default display value according to its nature. Every element on the webpage is a rectangular box and the CSS property defines the behavior of that rectangular box.

## CSS Display default properties

|  |  |
| --- | --- |
| default value | inline |
| inherited | no |
| animation supporting | no |
| version | css1 |
| javascript syntax | object.style.display="none" |

## Syntax

1. display:value;

## CSS display values

There are following CSS display values which are commonly used.

1. display: inline;
2. display: inline-block;
3. display: block;
4. display: run-in;
5. display: none;

## 1) CSS display inline

The inline element takes the required width only. It doesn't force the line break so the flow of text doesn't break in inline example.

The inline elements are:

* <span>
* <a>
* <em>
* <b> etc.

Let's see an example of CSS display inline.

<!DOCTYPE html>

<html>

<head>

<style>

p {

display: inline;

}

</style>

</head>

<body>

<p>Hello IHub.com</p>

<p>Java Tutorial.</p>

<p>SQL Tutorial.</p>

<p>HTML Tutorial.</p>

<p>CSS Tutorial.</p>

</body>

</html>

## 2) CSS display inline-block

The CSS display inline-block element is very similar to inline element but the difference is that you are able to set the width and height.

p {

display: inline-block;

}

## 3) CSS display block

The CSS display block element takes as much as horizontal space as they can. Means the block element takes the full available width. They make a line break before and after them.

p {

display: block;

}

## 4) CSS display run-in

This property doesn't work in Mozilla Firefox. These elements don't produce a specific box by themselves.

* If the run-in box contains a bock box, it will be same as block.
* If the block box follows the run-in box, the run-in box becomes the first inline box of the block box.
* If the inline box follows the run-in box, the run-in box becomes a block box.

p {

display: run-in;

}

## 5) CSS display none

The "none" value totally removes the element from the page. It will not take any space.

<style>

h1.hidden {

display: none;

}

</style>

Other CSS display values

|  |  |
| --- | --- |
| **Property-value** | **Description** |
| flex | It is used to display an element as an block-level flex container. It is new in css3. |
| inline-flex | It is used to display an element as an inline-level flex container. It is new in css3. |
| inline-table | It displays an element as an inline-level table. |
| list-Item | It makes the element behave like a <li> element. |
| table | It makes the element behave like a <table> element. |
| table-caption | It makes the element behave like a <caption> element. |
| table-column-group | It makes the element behave like a <colgroup> element. |
| table-header-group | It makes the element behave like a <thead> element. |
| table-footer-group | It makes the element behave like a <tfoot> element. |
| table-row-group | It makes the element behave like a <tbody> element. |
| table-cell | It makes the element behave like a <td> element. |
| table-row | It makes the element behave like a <tr> element. |
| table-column | It makes the element behave like a <col> element. |

# **CSS Cursor**

It is used to define the type of mouse cursor when the mouse pointer is on the element. It allows us to specify the cursor type, which will be displayed to the user. When a user hovers on the link, then by default, the cursor transforms into the hand from a pointer.

Let's understand the property values of the cursor.

|  |  |
| --- | --- |
| **Values** | **Usage** |
| **alias** | It is used to display the indication of the cursor of something that is to be created. |
| **auto** | It is the default property in which the browser sets the cursor. |
| **all-scroll** | It indicates the scrolling. |
| **col-resize** | Using it, the cursor will represent that the column can be horizontally resized. |
| **cell** | The cursor will represent that a cell or the collection of cells is selected. |
| **context-menu** | It indicates the availability of the context menu. |
| **default** | It indicates an arrow, which is the default cursor. |
| **copy** | It is used to indicate that something is copied. |
| **crosshair** | In it, the cursor changes to the crosshair or the plus sign. |
| **e-resize** | It represents the east direction and indicates that the edge of the box is to be shifted towards right. |
| **ew-resize** | It represents the east/west direction and indicates a bidirectional resize cursor. |
| **n-resize** | It represents the north direction that indicates that the edge of the box is to be shifted to up. |
| **ne-resize** | It represents the north/east direction and indicates that the edge of the box is to be shifted towards up and right. |
| **move** | It indicates that something is to be shifted. |
| **help** | It is in the form of a question mark or ballon, which represents that help is available. |
| **None** | It is used to indicate that no cursor is rendered for the element. |
| **No-drop** | It is used to represent that the dragged item cannot be dropped here. |
| **s-resize** | It indicates an edge box is to be moved down. It indicates the south direction. |
| **Row-resize** | It is used to indicate that the row can be vertically resized. |
| **Se-resize** | It represents the south/east direction, which indicates that an edge box is to be moved down and right. |
| **Sw-resize** | It represents south/west direction and indicates that an edge of the box is to be shifted towards down and left. |
| **Wait** | It represents an hourglass. |
| **<url>** | It indicates the source of the cursor image file. |
| **w-resize** | It indicates the west direction and represents that the edge of the box is to be shifted left. |
| **Zoom-in** | It is used to indicate that something can be zoomed in. |
| **Zoom-out** | It is used to indicate that something can be zoomed out. |

The illustration of using the above values of cursor property is given below:

**Example**

<html>

<head></head>

<style>

body{

background-color: lightblue;

color:green;

text-align: center;

font-size: 20px;

}

</style>

<body>

<p>Move your mouse over the below words for the cursor change.</p>

<div style = "cursor:alias">alias Value</div>

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

<div style = "cursor:all-scroll">all-scroll value</div>

<div style = "cursor:col-resize">col-resize value</div>

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

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

<div style = "cursor:copy">copy value</div>

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

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

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

<div style = "cursor:ew-resize">ew-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>

<div style = "cursor:progress">Progress</div>

<div style = "cursor:no-drop">no-drop</div>

<div style = "cursor:not-allowed">not-allowed</div>

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

<div style = "cursor:zoom-in">Zoom-in</div>

<div style = "cursor:zoom-out">Zoom-out</div>

</body></html>

# **CSS Buttons**

In HTML, we use the button tag to create a button, but by using CSS properties, we can style the buttons. Buttons help us to create user interaction and event processing. They are one of the widely used elements of web pages.

During the form submission, to view or to get some information, we generally use buttons.

Let's see the basic styling in buttons.

## Basic styling in Buttons

There are multiple properties available that are used to style the button element. Let's discuss them one by one.

## background-color

As we have discussed earlier, this property is used for setting the background color of the button element.

**Syntax**

element {

    // background-color style

}

**Example**

<!DOCTYPE html>

<html>

<head>

<title> button background Color </title>

<style>

body{

text-align: center;

}

button {

color:lightgoldenrodyellow;

font-size: 30px;

}

.b1 {

background-color: red;

}

.b2 {

background-color: blue;

}

.b3 {

background-color: violet;

}

</style>

</head>

<body>

<h1>The background-color property.</h1>

<button class="b1">Red color button</button>

<button class="b2">Blue color button</button>

<button class="b3">Violet color button</button>

</body>

</html>

## border

It is used to set the border of the button. It is the shorthand property for **border-width, border-color,** and **border-style**.

**Syntax**

element {

    // border style

}

**Example**

<!DOCTYPE html**>**

**<html>**

**<head>**

**<title>**

         button background Color

**</title>**

**<style>**

     body{

        text-align: center;

     }

         button {

            color:lightgoldenrodyellow;

             font-size: 30px;

         }

         .b1 {

             background-color: red;

             border:none;

         }

         .b2 {

             background-color: blue;

             border:5px brown solid;

         }

         .b3 {

            background-color: yellow;

             color:black;

             border:5px red groove;

         }

         .b4{

           background-color:orange;

            border: 5px red dashed;

         }

         .b5{

            background-color: gray;

            border: 5px black dotted;

         }

         .b6{

           background-color: lightblue;

            border:5px blue double;

        }

**</style>**

**</head>**

**<body>**

**<h1>**The border property**</h1>**

**<button** class="b1"**>**none**</button>**

**<button** class="b2"**>**solid**</button>**

**<button** class="b3"**>**groove**</button>**

**<button** class="b4"**>**dashed**</button>**

**<button** class="b5"**>**dotted**</button>**

**<button** class="b6"**>**double**</button>**

**</body>**

**</html>**

## border-radius

It is used to make the rounded corners of the button. It sets the border radius of the button.

**Syntax**

element {

    // border-radius property

}

**Example**

<!DOCTYPE html>

<html>

<head>

<title>

button background Color

</title>

<style>

body{

text-align: center;

}

button {

color:lightgoldenrodyellow;

font-size: 30px;

}

.b1 {

background-color: red;

border:none;

}

.b2 {

background-color: blue;

border:5px brown solid;

border-radius: 7px;

}

.b3 {

background-color: yellow;

color:black;

border:5px red groove;

border-radius: 10px;

}

.b4{

background-color:orange;

border: 5px red dashed;

border-radius: 20px;

}

.b5{

background-color: gray;

border: 5px black dotted;

border-radius: 30px;

}

.b6{

background-color: lightblue;

border:5px blue double;

border-radius: 25px;

}

</style>

</head>

<body>

<h1>The border-radius property</h1>

<h2>Below there is the border name and border-radius</h2>

<button class="b1">none</button>

<button class="b2">solid 7px</button>

<button class="b3">groove 10px</button>

<button class="b4">dashed 20px</button>

<button class="b5">dotted 30px</button>

<button class="b6">double 25px</button>

</body>

</html>

## box-shadow

As its name implies, it is used to create the shadow of the button box. It is used to add the shadow to the button. We can also create a shadow during the hover on the button.

**Syntax**

box-shadow: [horizontal offset] [vertical offset] [blur radius]

            [optional spread radius] [color];

**Example**

<!DOCTYPE html>

<html>

<head>

<title>

button background Color

</title>

<style>

body{

text-align: center;

}

button {

color:lightgoldenrodyellow;

font-size: 30px;

}

.b1{

background-color: lightblue;

border:5px red double;

border-radius: 25px;

color:black;

box-shadow : 0 8px 16px 0 black,

0 6px 20px 0 rgba(0, 0, 0, 0.19);

}

.b2{

background-color: lightblue;

border:5px red dotted;

color:black;

border-radius: 25px;

}

.b2:hover{

box-shadow : 0 8px 16px 0 black,

0 6px 20px 0 rgba(0, 0, 0, 0.19);

}

</style>

</head>

<body>

<button class="b1">Shadow on button</button>

<button class="b2">Box-shadow on hover</button>

</body>

</html>

## padding

It is used to set the button padding.

**Syntax**

element {

    // padding style

}

Let's understand it using an illustration.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>

button background Color

</title>

<style>

body{

text-align: center;

}

button {

color:lightgoldenrodyellow;

font-size: 30px;

}

.b1 {

background-color: red;

border:none;

padding: 16px;

}

.b2 {

background-color: blue;

border:5px brown solid;

padding:15px 30px 25px 40px;

}

.b3 {

background-color: yellow;

color:black;

border:5px red groove;

padding-top:30px;

}

.b4{

background-color:orange;

border: 5px red dashed;

padding-bottom:40px;

}

.b5{

background-color: gray;

border: 5px black dotted;

padding-left: 40px;

}

.b6{

background-color: lightblue;

border:5px blue double;

padding-right: 40px;;

}

</style>

</head>

<body>

<h1>The padding property</h1>

<button class="b1">none</button>

<button class="b2">solid</button>

<button class="b3">groove</button>

<button class="b4">dashed</button>

<button class="b5">dotted</button>

<button class="b6">double</button>

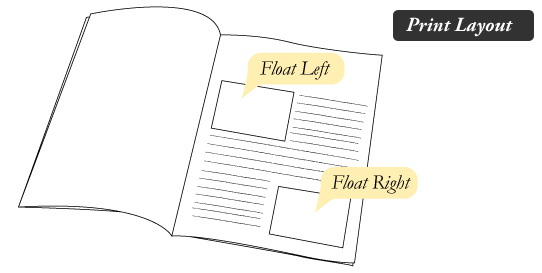
</body>

</html>

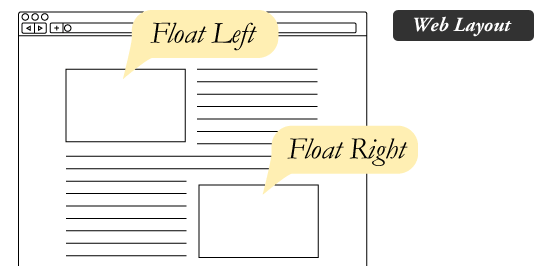
# **CSS Float**

The **CSS float property** is a positioning property. It is used to push an element to the left or right, allowing other element to wrap around it. It is generally used with images and layouts.

To understand its purpose and origin, let's take a look to its print display. In the print display, image is set into the page such that text wraps around it as needed.



Its web layout is also just similar to print layout.



## How it works

Elements are floated only horizontally. So it is possible only to float elements left or right, not up or down.

1. A floated element may be moved as far to the left or the right as possible. Simply, it means that a floated element can display at extreme left or extreme right.
2. The elements after the floating element will flow around it.
3. The elements before the floating element will not be affected.
4. If the image floated to the right, the texts flow around it, to the left and if the image floated to the left, the text flows around it, to the right.

## CSS Float Properties

|  |  |  |
| --- | --- | --- |
| **Property** | **Description** | **Values** |
| clear | The clear property is used to avoid elements after the floating elements which flow around it. | left, right, both, none, inherit |
| float | It specifies whether the box should float or not. | left, right, none, inherit |

## CSS Float Property Values

|  |  |
| --- | --- |
| **Value** | **Description** |
| none | It specifies that the element is not floated, and will be displayed just where it occurs in the text. this is a default value. |
| left | It is used to float the element to the left. |
| right | It is used to float the element to the right. |
| initial | It sets the property to its initial value. |
| inherit | It is used to inherit this property from its parent element. |

## CSS Float Property Example

Let's see a simple example to understand the CSS float property.

<!DOCTYPE html>

<html>

<head>

<style>

img {

float: right;

}

</style>

</head>

<body>

<p>The following paragraph contains an image with style

<b>float:right</b>. The result is that the image will float to the right in the paragraph.</p>

<img src="good-morning.jpg" alt="Good Morning Friends"/>

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

</p>

</body>

</html>

# **CSS Font**

CSS Font property is used to control the look of texts. By the use of CSS font property you can change the text size, color, style and more. You have already studied how to make text bold or underlined. Here, you will also know how to resize your font using percentage.

These are some important font attributes:

1. **CSS Font color**: This property is used to change the color of the text. (standalone attribute)
2. **CSS Font family**: This property is used to change the face of the font.
3. **CSS Font size**: This property is used to increase or decrease the size of the font.
4. **CSS Font style**: This property is used to make the font bold, italic or oblique.
5. **CSS Font variant**: This property creates a small-caps effect.
6. **CSS Font weight**: This property is used to increase or decrease the boldness and lightness of the font.

## 1) CSS Font Color

CSS font color is a standalone attribute in CSS although it seems that it is a part of CSS fonts. It is used to change the color of the text.

There are three different formats to define a color:

* By a color name
* By hexadecimal value
* By RGB

In the above example, we have defined all these formats.

<!DOCTYPE html>

<html>

<head>

<style>

body {

font-size: 100%;

}

h1 { color: red; }

h2 { color: #9000A1; }

p { color:rgb(0, 220, 98); }

}

</style>

</head>

<body>

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<p>This is a paragraph.</p>

</body>

</html>

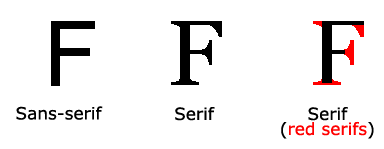
## 2) CSS Font Family

CSS font family can be divided in two types:

* Generic family: It includes Serif, Sans-serif, and Monospace.
* Font family: It specifies the font family name like Arial, New Times Roman etc.

**Serif**: Serif fonts include small lines at the end of characters. Example of serif: Times new roman, Georgia etc.

**Sans-serif**: A sans-serif font doesn't include the small lines at the end of characters. Example of Sans-serif: Arial, Verdana etc.



<!DOCTYPE html>

<html>

<head>

<style>

body {

font-size: 100%;

}

h1 { font-family: sans-serif; }

h2 { font-family: serif; }

p { font-family: monospace; }

}

</style>

</head>

<body>

<h1>This heading is shown in sans-serif.</h1>

<h2>This heading is shown in serif.</h2>

<p>This paragraph is written in monospace.</p>

</body>

</html>

3) CSS Font Size

CSS font size property is used to change the size of the font.

These are the possible values that can be used to set the font size:

|  |  |
| --- | --- |
| **Font Size Value** | **Description** |
| xx-small | used to display the extremely small text size. |
| x-small | used to display the extra small text size. |
| small | used to display small text size. |
| medium | used to display medium text size. |
| large | used to display large text size. |
| x-large | used to display extra large text size. |
| xx-large | used to display extremely large text size. |
| smaller | used to display comparatively smaller text size. |
| larger | used to display comparatively larger text size. |
| size in pixels or % | used to set value in percentage or in pixels. |

<html>

<head>

<title>Practice CSS font-size property</title>

</head>

<body>

<p style="font-size:xx-small;">

This font size is extremely small.</p>

<p style="font-size:x-small;">

This font size is extra small</p>

<p style="font-size:small;">

This font size is small</p>

<p style="font-size:medium;">

This font size is medium. </p>

<p style="font-size:large;">

This font size is large. </p>

<p style="font-size:x-large;">

This font size is extra large. </p>

<p style="font-size:xx-large;">

This font size is extremely large. </p>

<p style="font-size:smaller;">

This font size is smaller. </p>

<p style="font-size:larger;">

This font size is larger. </p>

<p style="font-size:200%;">

This font size is set on 200%. </p>

<p style="font-size:20px;">

This font size is 20 pixels.

</p>

</body>

</html>

## 4) CSS Font Style

CSS Font style property defines what type of font you want to display. It may be italic, oblique, or normal.

<!DOCTYPE html>

<html>

<head>

<style>

body {

font-size: 100%;

}

h2 { font-style: italic; }

h3 { font-style: oblique; }

h4 { font-style: normal; }

}

</style>

</head>

<body>

<h2>This heading is shown in italic font.</h2>

<h3>This heading is shown in oblique font.</h3>

<h4>This heading is shown in normal font.</h4>

</body>

</html>

## 5) CSS Font Variant

CSS font variant property specifies how to set font variant of an element. It may be normal and small-caps.

<!DOCTYPE html>

<html>

<head>

<style>

p { font-variant: small-caps; }

h3 { font-variant: normal; }

</style>

</head>

<body>

<h3>This heading is shown in normal font.</h3>

<p>This paragraph is shown in small font.</p>

</body>

</html>

## 6) CSS Font Weight

CSS font weight property defines the weight of the font and specify that how bold a font is. The possible values of font weight may be normal, bold, bolder, lighter or number (100, 200..... up to 900).

<!DOCTYPE html>

<html>

<body>

<p style="font-weight:bold;">This font is bold.</p>

<p style="font-weight:bolder;">This font is bolder.</p>

<p style="font-weight:lighter;">This font is lighter.</p>

<p style="font-weight:100;">This font is 100 weight.</p>

<p style="font-weight:200;">This font is 200 weight.</p>

<p style="font-weight:300;">This font is 300 weight.</p>

<p style="font-weight:400;">This font is 400 weight.</p>

<p style="font-weight:500;">This font is 500 weight.</p>

<p style="font-weight:600;">This font is 600 weight.</p>

<p style="font-weight:700;">This font is 700 weight.</p>

<p style="font-weight:800;">This font is 800 weight.</p>

<p style="font-weight:900;">This font is 900 weight.</p>

</body>

</html>

# **CSS Font-size**

The font-size property in CSS is used to specify the height and size of the font. It affects the size of the text of an element. Its default value is medium and can be applied to every element. The values of this property include **xx-small**, **small**, **x-small**, etc.

### **Syntax**

1. font-size: medium|large|x-large|xx-large|xx-small|x-small|small|;

The font-size can be relative or absolute.

### **Absolute-size**

It is used to set the text to a definite size. Using absolute-size, it is not possible to change the size of the text in all browsers. It is advantageous when we know the physical size of the output.

### **Relative-size**

It is used to set the size of the text relative to its neighboring elements. With relative-size, it is possible to change the size of the text in browsers.

#### **NOTE: If we do not define a font-size, then for the normal text such as paragraphs, the default size is 16px, which is equal to 1em.**

## Font-size with pixels

When we set the size of text with pixels, then it provides us the full control over the size of the text.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

#first {

font-size: 40px;

}

#second {

font-size: 20px;

}

</style>

</head>

<body>

<p id="first">This is a paragraph having size 40px.</p>

<p id="second">This is another paragraph having size 20px.</p>

</body>

</html>

## Font-size with em

It is used to resize the text. Most of the developers prefer **em** instead of **pixels**. It is recommended by the world wide web consortium (W3C). As stated above, the default text size in browsers is 16px. So, we can say that the default size of **1em** is **16px**.

The formula for calculating the size from **pixels** to **em** is **em = pixels/16**.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

#first {

font-size: 2.5em; /\* 40px/16=2.5em \*/

}

#second {

font-size: 1.875em; /\* 30px/16=1.875em \*/

}

#third {

font-size: 0.875em; /\* 14px/16=0.875em \*/

}

</style>

</head>

<body>

<p id='first'>First paragraph.</p>

<p id='second'>Second paragraph</p>

<p id='third'>Third Paragraph.</p>

</body></html>

## Responsive font size

We can set the size of the text by using a **vw unit**, which stands for the '**viewport width**'. The viewport is the size of the browser window.

**1vw = 1% of viewport width.**

If the width of the viewport is 50cm, then the 1vw is equal to 0.5 cm.

### **Example**

<!DOCTYPE html>

<html>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<body>

<p style="font-size:5vw;">First paragraph having the width of 5vw.</p>

<p style="font-size:10vw;">Second paragraph having the width of 10vw.</p>

</body>

</html>

## Font-size with the length property

It is used to set the size of the font in length. The length can be in cm, px, pt, etc. Negative values of **length** property are not allowed in font-size.

### **Syntax**

font-size: length;

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.length {

color:red;

font-size: 5cm;

}

</style>

</head>

<body>

<h1>font-size property</h1>

<p class = "length">A paragraph having length 5cm.</p>

</body>

</html>

# **CSS font-family**

This CSS property is used to provide a comma-separated list of font families. It sets the font-face for the text content of an element. This property can hold multiple font names as a fallback system, i.e., if one font is unsupported in the browser, then others can be used. The different font-family is used for making attractive web pages.

There are two types of font-family names in [CSS](https://www.javatpoint.com/css-tutorial), which are defined below:

* **family-name:** It is the name of the font-family such as "Courier", "Arial", "Times", etc.
* **generic-family:** It is the name of the generic family that includes five categories, which are "serif", "sans-serif", "cursive", "fantasy", and "monospace". It should be placed at last in the list of the font family names.

Let's define the generic-family categories.

**serif:** It is mainly used when we are writing the text for printing, such as books, magazines, newspapers, etc. It includes the font-family such as Georgia, Garamond, Times New Roman, Minion, and many more.

**sans-serif:** It is a modern, formal, and simple style. It is widely used but most often in the digital form of text. It includes the font-family that are Arial, Calibri, Verdana, Futura, Lato, and many more.

**cursive:** It is mainly used for writing the invitation letter, informal messages, etc. It is like a handwritten text which is written by a pen or a brush. The font-family that it includes is Insolente, Corsiva, Flanella, Belluccia, Zapfino, and many more.

**monospace:** It is for instructions, mailing address, typewritten text, etc. It includes the font-family that is Monaco, SimSun, Courier, Consolas, Inconsolata, and many more.

**fantasy:** It makes the text expressive, decorative, and impactful. It includes the font-family that is Impact, Copperplate, Cracked, Critter, and many more.

**Syntax**

1. font-family: family-name|generic-family|initial|inherit;

**Values**

Let's see the values of the font-family property.

**family-name/generic-family:** It is the list of font-family names and the generic family names.

**initial:** It is used to set the property to its default value.

**inherit:** It is used to inherit the property from its parent element.

Let's understand it by using an illustration.

**Example**

<!DOCTYPE html>

<html>

<head>

<style>

body{

text-align:center;

}

h1.a {

font-family: "Times New Roman", Times, serif;

color:Red;

}

h2.b {

font-family: Arial, Helvetica, sans-serif;

color:blue;

}

</style>

</head>

<body>

<h1>The font-family Property</h1>

<h1 class="a">Hello World :) :)</h1>

<h2 class="b">Welcome to the iHub.com</h2>

</body>

</html>

# **CSS font-weight**

This property is used for setting the thickness and boldness of the font. It is used to define the weight of the text. The available weight depends on the font-family, which is used by the browser.

**Syntax**

1. font-weight: normal | lighter | bolder | bold | number | inherit |initial | unset;

**Property Values**

**normal:** It is the default font-weight whose numeric value is 400.

**lighter:** It considers the existing font-weight of the font-family and makes the font-weight lighter compare to the parent element.

**bolder:** It considers the existing font-weight of the font-family and makes the font-weight heavier compare to the parent element.

**bold:** As its name implies, it is used to define the bold font-weight, and its numeric value is 700.

**number:** It is used to set the font-weight based on the specified number. Its range can be between 1 to 1000.

**initial:** It is used to set the font-weight to its default value.

Let's see an example of this property.

**Example**

<!DOCTYPE html>

<html>

<head>

<title> font-weight property </title>

<style>

body{

font-family: sans-serif;

}

p.one{

font-weight: normal;

}

p.two{

font-weight: lighter;

}

p.three{

font-weight: bolder;

}

p.four{

font-weight: bold;

}

p.five{

font-weight: 1000;

}

p.six{

font-weight: initial;

}

p.seven{

font-weight: inherit;

}

p.eight{

font-weight: unset;

}

</style>

</head>

<body>

<p class="one">

normal property value

</p>

<p class="two">

lighter property value

</p>

<p class="three">

bolder property value

</p>

<p class="four">

bold property value

</p>

<p class="five">

number property value

</p>

<p class="six">

initial property value

</p>

<p class="seven">

inherit property value

</p>

<p class="eight">

unset property value

</p>

</body>

</html>

# **CSS font-stretch property**

The **font-stretch** property in CSS allows us to select a **normal, expanded**, or **condensed** face from the font's family. This property sets the text wider or narrower compare to the default width of the font. It will not work on any font but only works on the font-family that has a width-variant face.

This CSS property only works for the fonts with additional faces like expanded and condensed faces; otherwise, it will be affectless for the fonts that don't have condensed or expanded faces.

The nine keyword values for choosing the width of the font-face are given in the following syntax.

### **Syntax**

1. font-stretch: normal | semi-condensed | condensed | extra-condensed | ultra-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded

### **Property Values**

The property values of this CSS property are tabulated as follows:

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| **normal** | This is the default value, which does not stretch any font. |
| **semi-condensed** | It slightly condensed the text characters of the element. This value makes the text narrower than **normal** but not narrower than **condensed**. |
| **condensed** | This value makes the text narrower than **semi-condensed** but not narrower than **extra-condensed**. |
| **extra-condensed** | This value makes the text narrower than **condensed** but not narrower than **ultra-condensed**. |
| **ultra-condensed** | This value makes the text extremely narrowed. |
| **semi-expanded** | It slightly widened the text characters of the element. This value makes the text wider than **normal** but not wider than **expanded**. |
| **expanded** | This value makes the text wider than **semi-expanded** but not wider than **extra-expanded**. |
| **extra-expanded** | This value makes the text wider than **expanded** but not wider than **ultra-expanded**. |
| **ultra-expanded** | This value makes the text extremely wider. |

Let's understand the above property values by using an example.

### **Example**

<!DOCTYPE html**>**

**<html>**

**<head>**

**<title>**

CSS font-stretch Property

**</title>**

**<style>**

body{

text-align: center;

}

div{

font-family: Arial, Helvetica, sans-serif;

font-size: 20px;

color: blue;

}

.normal {

font-stretch: normal;

}

.semi-condensed {

font-stretch: semi-condensed;

}

.condensed {

font-stretch: condensed;

}

.extra-condensed {

font-stretch: extra-condensed;

}

.ultra-condensed {

font-stretch: ultra-condensed;

}

.semi-expanded {

font-stretch: semi-expanded;

}

.expanded {

font-stretch: expanded;

}

.extra-expanded {

font-stretch: extra-expanded;

}

.ultra-expanded {

font-stretch: ultra-expanded;

}

**</style>**

**</head>**

**<body>**

**<h1>** Example of the font-stretch property **</h1>**

**<div** class = "normal"**>**

normal

**</div>**

**<div** class = "semi-condensed"**>**

semi-condensed

**</div>**

**<div** class = "condensed"**>**

condensed

**</div>**

**<div** class = "extra-condensed"**>**

extra-condensed

**</div>**

**<div** class = "ultra-condensed"**>**

ultra-condensed

**</div>**

**<div** class = "semi-expanded"**>**

semi-expanded

**</div>**

**<div** class = "expanded"**>**

expanded

**</div>**

**<div** class = "extra-expanded"**>**

extra-expanded

**</div>**

**<div** class = "ultra-expanded"**>**

ultra-expanded

**</div>**

**</body>**

**</html>**

# **CSS Colors**

The color property in CSS is used to set the color of HTML elements. Typically, this property is used to set the background color or the font color of an element.

In CSS, we use color values for specifying the color. We can also use this property for the border-color and other decorative effects.

We can define the color of an element by using the following ways:

* RGB format.
* RGBA format.
* Hexadecimal notation.
* HSL.
* HSLA.
* Built-in color.

Let's understand the syntax and description of the above ways in detail.

## RGB Format

RGB format is the short form of '**RED GREEN** and **BLUE**' that is used for defining the color of an HTML element simply by specifying the values of R, G, B that are in the range of 0 to 255.

The color values in this format are specified by using the **rgb()** property. This property allows three values that can either be in percentage or integer (range from 0 to 255).

This property is not supported in all browsers; that's why it is not recommended to use it.

**Syntax**

1. color: rgb(R, G, B);

## RGBA Format

It is almost similar to RGB format except that **RGBA** contains **A (Alpha)** that specifies the element's transparency. The value of alpha is in the range **0.0 to 1.0**, in which **0.0** is for fully transparent, and **1.0** is for not transparent.

**Syntax**

1. color:rgba(R, G, B, A);

## Hexadecimal notation

Hexadecimal can be defined as a six-digit color representation. This notation starts with the **# symbol** followed by six characters ranges from **0 to F**. In hexadecimal notation, the first two digits represent the **red (RR)** color value, the next two digits represent the **green (GG)** color value, and the last two digits represent the **blue (BB)** color value.

The black color notation in hexadecimal is #000000, and the white color notation in hexadecimal is #FFFFFF. Some of the codes in hexadecimal notation are #FF0000, #00FF00, #0000FF, #FFFF00, and many more.

**Syntax**

1. color:#(0-F)(0-F)(0-F)(0-F)(0-F)(0-F);

## Short Hex codes

It is a short form of hexadecimal notation in which every digit is recreated to arrive at an equivalent hexadecimal value.

For example, #7B6 becomes #77BB66 in hexadecimal.

The black color notation in short hex is #000, and the white color notation in short hex is #FFF. Some of the codes in short hex are #F00, #0F0, #0FF, #FF0, and many more.

## HSL

It is a short form of **Hue, Saturation,** and **Lightness**. Let's understand them individually.

**Hue:** It can be defined as the degree on the color wheel from 0 to 360. 0 represents red, 120 represents green, 240 represents blue.

**Saturation:** It takes value in percentage in which 100% represents fully saturated, i.e., no shades of gray, 50% represent 50% gray, but the color is still visible, and 0% represents fully unsaturated, i.e., completely gray, and the color is invisible.

**Lightness:** The lightness of the color can be defined as the light that we want to provide the color in which 0% represents black (there is no light), 50% represents neither dark nor light, and 100% represents white (full lightness).

Let's see the syntax of HSL in color property.

**Syntax**

1. color:hsl(H, S, L);

## HSLA

It is entirely similar to HSL property, except that it contains **A (alpha)** that specifies the element's transparency. The value of alpha is in the range **0.0 to 1.0**, in which **0.0** indicates fully transparent, and **1.0** indicates not transparent.

**Syntax**

1. color:hsla(H, S, L, A);

## Built-in Color

As its name implies, built-in color means the collection of previously defined colors that are used by using a name such as red, blue, green, etc.

**Syntax**

1. color: color-name;

Let's see the list of built-in colors along with their decimal and hexadecimal values.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no.** | **Color name** | **Hexadecimal Value** | **Decimal Value or rgb() value** |
| **1.** | Red | #FF0000 | rgb(255,0,0) |
|  |  |  |  |
| **2.** | Orange | #FFA500 | rgb(255,165,0) |
|  |  |  |  |
| **3.** | Yellow | #FFFF00 | rgb(255,255,0) |
|  |  |  |  |
| **4.** | Pink | #FFC0CB | rgb(255,192,203) |
|  |  |  |  |
| **5.** | Green | #008000 | rgb(0,128,0) |
|  |  |  |  |
| **6.** | Violet | #EE82EE | rgb(238,130,238) |
|  |  |  |  |
| **7.** | Blue | #0000FF | rgb(0,0,255) |
|  |  |  |  |
| **8.** | Aqua | #00FFFF | rgb(0,255,255) |
|  |  |  |  |
| **9.** | Brown | #A52A2A | rgb(165,42,42) |
|  |  |  |  |
| **10.** | White | #FFFFFF | rgb(255,255,255) |
|  |  |  |  |
| **11.** | Gray | #808080 | rgb(128,128,128) |
|  |  |  |  |
| **12.** | Black | #000000 | rgb(0,0,0) |
|  |  |  |  |

The illustration of CSS colors, which includes the above properties, is given below.

### **Example**

<html>

<head>

<title>CSS hsl color property</title>

<style>

h1{

text-align:center;

}

#rgb{

color:rgb(255,0,0);

}

#rgba{

color:rgba(255,0,0,0.5);

}

#hex{

color:#EE82EE;

}

#short{

color: #E8E;

}

#hsl{

color:hsl(0,50%,50%);

}

#hsla{

color:hsla(0,50%,50%,0.5);

}

#built{

color:green;

}

</style>

</head>

<body>

<h1 id="rgb">

Hello World. This is RGB format.

</h1>

<h1 id="rgba">

Hello World. This is RGBA format.

</h1>

<h1 id="hex">

Hello World. This is Hexadecimal format.

</h1>

<h1 id="short">

Hello World. This is Short-hexadecimal format.

</h1>

<h1 id="hsl">

Hello World. This is HSL format.

</h1>

<h1 id="hsla">

Hello World. This is HSLA format.

</h1>

<h1 id="built">

Hello World. This is Built-in color format.

</h1>

</body>

</html>

# **CSS hover**

The :**hover** selector is for selecting the elements when we move the mouse on them. It is not only limited to the links. We can use it on almost every HTML element. To style the link to unvisited pages, we can use the :**link** selector. To style the link for visited pages, we can use the :**visited** selector and to style the active links we can use the :**active** selector.

It is introduced in CSS1. The hover can be used to highlight the web pages as per the preference of users in an effective web-designing program.

The hover feature includes the following effects:

* Change the color of the background and font.
* Modify the opacity of the image.
* Text embedding.
* Create image rollover effects.
* Swapping of images.

#### **NOTE: To make the hover effective, we must have to declare it after the :link and :visited selectors if they are present in the CSS definition.**

Basically, the hover effect modifies the element's property value to enable the animate changes on a stated image/text or the corresponding elements. Embedding of the hover elements in the web pages makes them interactive and functional.

Generally, the hover feature is compatible with all of the main browsers. But, it will be a challenging task to implement it on touch devices. It is observed that an active hover function gets stuck on the non-supportive device.

### **Syntax**

:hover {

  css declarations;

}

Let's understand it by using some illustrations.

**Example 1: Changing the link color on hover by using CSS**

Let's see how the color of the link gets changed when we place the cursor on it. It will create a stylish effect, and its implementation is easy when we are using CSS.

<html>

<head>

<style>

body{

text-align:center;

}

a

{

color: red;

}

a:hover

{

color: green;

}

a:active

{

color: cyan;

}

</style>

</head>

<body>

<h1>Move your mouse on the below link to see the hover effect.</h1>

<a class = "link" href = https://www.iHubTalent.com/css-grid>CSS Grid</a>

</body>

</html>

**Example 2: Apply hover on paragraph, heading and link**

<html>

<head>

<style>

body{

text-align:center;

}

p:hover, h1:hover, a:hover{

background-color: yellow;

}

</style>

</head>

<body>

<h1>Hello World</h1>

<p>Welcome to the iHubTalent.</p>

<a href='https://www. iHubTalent.com/css-grid'>CSS Grid</a>

</body>

</html>

**Example 3- Text overlay on image hover**

This CSS code displays the text on the image during mouse hover. Let's see the image overlay effect during mouse hover.

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body{

text-align:center;

}

\* {box-sizing: border-box;}

.container {

position: relative;

width: 50%;

max-width: 300px;

}

.image {

display: block;

width: 100%;

height: auto;

}

.overlay {

position: absolute;

bottom: 0;

background: rgba(0, 0, 0, 0.2);

width: 100%;

opacity:0;

transition: .9s ease;

font-size: 25px;

padding: 20px;

}

.container:hover .overlay {

opacity: 1.5;

}

</style>

</head>

<body>

<h1>Image Overlay Title Effect</h1>

<h2>Move your mouse over the image to see the effect.</h2>

<center>

<div class="container">

<img src="iHub.png" class="image">

<div class="overlay">Welcome to iHubTalent.com</div>

</div> </center>

</body>

</html>

# **CSS Important**

This property in CSS is used to give more importance compare to normal property. The **!important** means **'this is important'**. This rule provides a way of making the Cascade in CSS.

If we apply this property to the text, then the priority of that text is higher than other priorities. It is to be recommended not to use this CSS property into your program until it is highly required. It is because the more use of this property will cause a lot of unexpected behavior.

If a rule is defined with this attribute, it will reject the normal concern in which the later used rule overrides the previous ones. If we use more than one declaration marked **!important**, then the normal cascade takes it over again. That means the new marked **!important** will replace the previous one.

It increases the priority of the CSS property and ignores the overriding properties.

### **Syntax**

element {

    font-size: 14px !important;

    color: blue  !important;

    ...

}

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

color: white ;

}

H1 {

color:blue !important;

}

body {

background-color:lightblue !important;

text-align:center;

background-color:yellow;

}

</style>

</head>

<body>

<h1>Hello World.</h1>

<h1>Welcome to the iHubTalent.com. This is an example of <i>!important</i> property.</h1>

<p></p>

</body>

</html>

In the above example, we can see that instead of pink, the background color of the body is light blue because, in the body tag, the **!important** is applied after the light blue background color.

Let's take another example of this property to understand it more clearly.

### **Example**

In this example, we are applying the **!important** attribute on the border of the text. The color of the border of **h1** heading will remain **red** despite of other declarations. The color and border-color of heading h2 will remain **green** and **violet** despite of other declarations.

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body{

text-align: center;

}

h1 {

border-color: red !important;

border: 5px green solid;

border-color: black;

}

h2{

color: green !important;

color: red;

border-color:violet !important;

border: 5px green solid;

}

</style>

</head>

<body>

<h1>Hello World :) :)</h1>

<h2>Welcome to the iHubTalent.com</h2>

</body>

</html>

# **CSS Background-color**

This property is used to set the background color of an element. The background of an element covers the total size, including the padding and border and excluding margin. It can be applied to all HTML elements.

### **Syntax**

1. element {
2. background-color: color\_name|transparent|initial|inherit;
3. }

Let's discuss the possible values of this property.

* **color\_name:** It is used for defining the background color value or the color codes. It can be given by using the color name, hexadecimal value, or rgb() value.
* **transparent:** It is the default value of this property, which is used to specify the transparent background-color.
* **initial:** It is not used to set the background color. It sets the default value.
* **Inherit:** It is used to inherit the background-color from its parent.

Let's see an illustration of this property.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>background-color property</title>

<style>

body {

text-align:center;

background-color: lightblue;

}

h1{

color: blue;

}

</style>

</head>

<body>

<h1>Hello World.</h1>

<h1>Welcome to the iHubTalent.com</h1>

</body>

</html>

# **CSS background-attachment property**

The background-attachment property is used to specify that the background image is fixed or scroll with the rest of the page in the browser window.

This property has three values **scroll, fixed,** and **local**. Its default value is **scroll**, which causes the element to not scroll with its content. The **local** value of this property causes the element to scroll with the content. If we set the value to **fixed**, the background image will not move during scrolling in the browser.

This CSS property can support multiple background images. We can specify a different value of the **background-attachment** property for each background-image, separated by commas. Every image will match with the corresponding value of this property.

### **Syntax**

1. background-attachment: scroll | fixed | local | initial | inherit;

The values of this property are defined as follows.

### **Property Values**

**scroll:** It is the default value that prevents the element from scrolling with the contents, but scrolls with the page.

**fixed:** Using this value, the background image doesn't move with the element, even the element has a scrolling mechanism. It causes the image to be locked in one place, even the rest of the document scrolls.

**local:** Using this value, if the element has a scrolling mechanism, the background image scrolls with the content of the element.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Let's understand this property by using some illustrations.

### **Example**

In this example, we are using the **scroll** value of the **background-attachment** property, which is the default behavior. So when the page is scrolled, the background scrolls with it.

<!DOCTYPE html>

<html>

<head>

<title>

background-attachment property

</title>

<style>

#example {

background-image: url("lion.png");

font-size: 35px;

border: 4px solid red;

color: white;

background-position: center;

background-color: green;

background-repeat: no-repeat;

background-attachment: scroll;

}

</style>

</head>

<body>

<h1> background-attachment: scroll;</h1>

<p> If there is no scrollbar on your screen, then try to resize the browser's window to see the effect. </p>

<div id="example">

<p>

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

</div>

</body></html>

# **CSS background-size property**

The **background-size** CSS property is used to set the size of a background image of an element. The background image can be stretched or constrained to fit into the existing space. It allows us to control the scaling of the background image.

This property can be defined using **length, percentage,** or **keyword** values. It has two possible keyword values that are **contain** and **cover**. Its single-value syntax defines the width of the image (in this case, the height sets to auto), whereas the double values define the value of both height and width in which the first value sets the width and second sets the height.

If an element has multiple background images, we can define the comma-separated values to define the different sizes of each one.

The **cover** value of the **background-size** property is used to cover the entire background area of the element. In contrast, the **contain** value of this property scales the image as much as possible without clipping the image.

### **Syntax**

1. background-size: auto | length | cover | contain | initial | inherit;

The values of this property are defined as follows.

### **Property Values**

**auto:** This is the default value, which displays the background image in its original size.

**length:** It is used to set the width and height of the background image. This value stretches the image in the corresponding dimension of the given length. Its single value specifies the width of the image, and the height sets to auto. If two values are given, the first value sets the width, and the second value sets the height. It does not allow negative values.

**percentage:** This value defines the width and height of the background image to the percentage (%) of the background positioning area. Negative values are not allowed.

**cover:** This value is used to resize the background image to cover the entire container. Sometimes, it crops the little bit off one of the edges or stretches the image. It resizes the image to ensure the element is completely covered.

**contain:** Without stretching or cropping, it resizes the background image to ensure the image is completely visible.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Let's understand this CSS property by using some illustrations.

### **Example**

In this example, there are three div elements with a width of 300px and a height of 200px. Every div element has a background-image on which we are applying the **background-size** property.

Here we are using the length and percentage values to set the background-size of div element. The **background-size** of first div element set to **auto**, second div element is set to **150px 150px**, and the **background-size** of third div element is set to **30%**.

<!DOCTYPE html>

<html>

<head>

<title>

background-size property

</title>

<style>

div {

width: 300px;

height: 200px;

border: 2px solid red;

}

#div1{

background-image: url('lion.png');

background-size: auto;

}

#div2{

background-image: url('lion.png');

background-size: 150px 150px;

}

#div3{

background-image: url('lion.png');

background-size: 30%;

}

</style>

</head>

<body>

<h2> background-size: auto; </h2>

<div id = "div1"></div>

<h2> background-size: 150px 150px; </h2>

<div id = "div2"></div>

<h2> background-size: 30%; </h2>

<div id = "div3"></div>

</body>

</html>

Now, in the next example, we are using the **cover, contain,** and **initial** values of the **background-size** property.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>

background-size property

</title>

<style>

div {

width: 300px;

height: 250px;

border: 2px solid red;

background-repeat: no-repeat;

}

#div1{

background-image: url('lion.png');

background-size: contain;

}

#div2{

background-image: url('lion.png');

background-size: cover;

}

#div3{

background-image: url('lion.png');

background-size: initial;

}

</style>

</head>

<body>

<h2> background-size: contain; </h2>

<div id = "div1"></div>

<h2> background-size: cover; </h2>

<div id = "div2"></div>

<h2> background-size: initial; </h2>

<div id = "div3"></div>

</body>

</html>

### **Example - Combining multiple images**

We can also combine the values of this property and can apply them to multiple images. It can be done by comma-separated syntax.

In this example, there are three div elements, each having two background-images. Now, we are applying the **background-size** property on both images.

<!DOCTYPE html>

<html>

<head>

<title>

background-size property

</title>

<style>

div {

width: 250px;

height: 250px;

border: 2px solid red;

background-repeat: no-repeat;

background-position: center;

}

#div1{

background-image: url('lion.png'), url('forest.jpg');

background-size: 300px 150px, cover;

}

#div2{

background-image: url('lion.png'), url('forest.jpg');

background-size: 200px 150px, 300px 200px;

}

#div3{

background-image: url('lion.png'), url('forest.jpg');

background-size: 150px 175px, contain;

}

</style>

</head>

<body>

<h2> background-size: 300px 150px, cover; </h2>

<div id = "div1"></div>

<h2> background-size: 200px 150px, 300px 200px; </h2>

<div id = "div2"></div>

<h2> background-size: 150px 175px, contain; </h2>

<div id = "div3"></div>

</body>

</html>

# **CSS Line Height**

The **CSS line height property** is used *to define the minimal height of line boxes within the element*. It sets the differences between two lines of your content.

It defines the amount of space above and below inline elements. It allows you to set the height of a line of independently from the font size.

CSS line-height values

There are some property values which are used with [CSS](https://www.javatpoint.com/css-tutorial) line-height property.

|  |  |
| --- | --- |
| **value** | **description** |
| normal | This is a default value. it specifies a normal line height. |
| number | It specifies a number that is multiplied with the current font size to set the line height. |
| length | It is used to set the line height in px, pt,cm,etc. |
| % | It specifies the line height in percent of the current font. |
| initial | It sets this property to its default value. |
| inherit | It inherits this property from its parent element. |

CSS line-height example

<!DOCTYPE html>

<html>

<head>

<style>

h3.small {

line-height: 70%;

}

h3.big {

line-height: 200%;

}

</style>

</head>

<body>

<h3>

This is a heading with a standard line-height.<br>

This is a heading with a standard line-height.<br>

The default line height in most browsers is about 110% to 120%.<br>

</h3>

<h3 class="small">

This is a heading with a smaller line-height.<br>

This is a heading with a smaller line-height.<br>

This is a heading with a smaller line-height.<br>

This is a heading with a smaller line-height.<br>

</h3>

<h3 class="big">

This is a heading with a bigger line-height.<br>

This is a heading with a bigger line-height.<br>

This is a heading with a bigger line-height.<br>

This is a heading with a bigger line-height.<br>

</h3>

</body>

</html>

# **CSS Margin**

CSS Margin property is used to define the space around elements. It is completely transparent and doesn't have any background color. It clears an area around the element.

Top, bottom, left and right margin can be changed independently using separate properties. You can also change all properties at once by using shorthand margin property.

There are following CSS margin properties:

## CSS Margin Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| margin | This property is used to set all the properties in one declaration. |
| margin-left | it is used to set left margin of an element. |
| margin-right | It is used to set right margin of an element. |
| margin-top | It is used to set top margin of an element. |
| margin-bottom | It is used to set bottom margin of an element. |

## CSS Margin Values

These are some possible values for margin property.

|  |  |
| --- | --- |
| **Value** | **Description** |
| auto | This is used to let the browser calculate a margin. |
| length | It is used to specify a margin pt, px, cm, etc. its default value is 0px. |
| % | It is used to define a margin in percent of the width of containing element. |
| inherit | It is used to inherit margin from parent element. |

#### **Note: You can also use negative values to overlap content.**

## CSS margin Example

You can define different margin for different sides for an element.

<!DOCTYPE html>

<html>

<head>

<style>

p {

background-color: pink;

}

p.ex {

margin-top: 50px;

margin-bottom: 50px;

margin-right: 100px;

margin-left: 100px;

}

</style>

</head>

<body>

<p>This paragraph is not displayed with specified margin. </p>

<p class="ex">This paragraph is displayed with specified margin.</p>

</body>

</html>

## Margin: Shorthand Property

CSS shorthand property is used to shorten the code. It specifies all the margin properties in one property.

There are four types to specify the margin property. You can use one of them.

1. margin: 50px 100px 150px 200px;
2. margin: 50px 100px 150px;
3. margin: 50px 100px;
4. margin 50px;

## 1) margin: 50px 100px 150px 200px;

It identifies that:

**top** margin value is 50px

**right** margin value is 100px

**bottom** margin value is 150px

**left** margin value is 200px

<!DOCTYPE html>

<html>

<head>

<style>

p {

background-color: pink;

}

p.ex {

margin: 50px 100px 150px 200px;

}

</style>

</head>

<body>

<p>This paragraph is not displayed with specified margin. </p>

<p class="ex">This paragraph is displayed with specified margin.</p>

</body>

</html>

## 2) margin: 50px 100px 150px;

It identifies that:

**top** margin value is 50px

**left and right** margin values are 100px

**bottom** margin value is 150px

<!DOCTYPE html>

<html>

<head>

<style>

p {

background-color: pink;

}

p.ex {

margin: 50px 100px 150px;

}

</style>

</head>

<body>

<p>This paragraph is not displayed with specified margin. </p>

<p class="ex">This paragraph is displayed with specified margin.</p>

</body>

</html>

## 3) margin: 50px 100px;

It identifies that:

**top and bottom** margin values are 50px

**left and right** margin values are 100px

<!DOCTYPE html>

<html>

<head>

<style>

p {

background-color: pink;

}

p.ex {

margin: 50px 100px;

}

</style>

</head>

<body>

<p>This paragraph is not displayed with specified margin. </p>

<p class="ex">This paragraph is displayed with specified margin.</p>

</body>

</html>

## 4) margin: 50px;

It identifies that:

**top right bottom and left** margin values are 50px

<!DOCTYPE html>

<html>

<head>

<style>

p {

background-color: pink;

}

p.ex {

margin: 50px;

}

</style>

</head>

<body>

<p>This paragraph is not displayed with specified margin. </p>

<p class="ex">This paragraph is displayed with specified margin.</p>

</body>

</html>

# **CSS Opacity**

The CSS opacity property is used to specify the transparency of an element. In simple word, you can say that it specifies the clarity of the image.

In technical terms, Opacity is defined as degree in which light is allowed to travel through an object.

## How to apply CSS opacity setting

Opacity setting is applied uniformly across the entire object and the opacity value is defined in term of digital value less than 1. The lesser opacity value displays the greater opacity. Opacity is not inherited.

### **CSS Opacity Example: transparent image**

Let's see a simple CSS opacity example of image transparency.

<!DOCTYPE html>

<html>

<head>

<style>

img.trans {

opacity: 0.4;

filter: alpha(opacity=40); /\* For IE8 and earlier \*/

}

</style>

</head>

<body>

<p>Normal Image</p>

<img src="/csspages/images/rose.jpg" alt="normal rose">

<p>Transparent Image</p>

<img class="trans" src="/csspages/images/rose.jpg" alt="transparent rose">

</body>

</html>

#### **Note 1: Chrome, Firefox, Opera, Safari, and IE9 use the opacity property for transparency. The opacity value ranges from 0.1 to 1.0. Lower value produces the greater opacity.**

#### **Note 2: The older versions of IE use filter: alpha(opacity=x). Here x value varies from 0 to 100. Lower value produces the greater opacity.**

# **CSS filter**

CSS filters are used to set visual effects to text, images, and other aspects of a webpage. The CSS **filter** property allows us to access the effects such as color or blur, shifting on the rendering of an element before the element gets displayed.

The syntax of CSS filter property is given below.

**Syntax**

1. filter: none | invert() | drop-shadow() | brightness() | saturate() | blur() | hue-rotate() | contrast() | opacity() | grayscale() | sepia() | url();

Let's discuss the property values along with an example.

## brightness()

As its name implies, it is used to set the brightness of an element. If the brightness is 0%, then it represents completely black, whereas 100% brightness represents the original one. It can also accept values above 100% that provide brighter results.

We can understand it by using the following illustration.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: brightness(130%);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2>brightness(130%)</h2>

</body>

</html>

## blur()

It is used to apply the blur effect to the element. If the blur value is not specified, then the value 0 is used as a default value. The parameter in blur() property does not accept the percentage values. A larger value of it creates more blur.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: blur(2px);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2>blur(2px)</h2>

</body>

</html>

## invert()

It is used to invert the samples in the input image. Its 100% value represents completely inverted, and 0% values leave the unchanged input. Negative values are not allowed in it.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: invert(60);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2>invert(60)</h2>

</body>

</html>

## saturate()

It sets the saturation of an element. The 0% saturation represents the completely unsaturated element, whereas the 100% saturation represents the original one. The values greater than 100% are allowed that provides super-saturated results. We cannot use negative values with this property.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: saturate(40);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2>saturate(40)</h2>

</body>

</html>

## drop-shadow()

It applies the drop-shadow effect to the input image. The values it accepts are **h-shadow, v-shadow, blur, spread,**and **color.**

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: drop-shadow(10px 20px 30px yellow);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2> drop-shadow(10px 20px 30px yellow);</h2>

</body>

</html>

## contrast()

It adjusts the contrast of the input. Its 0% value will create a completely black image, whereas the 100% values leave the unchanged input, i.e., represents the original one. Values greater than 100% are allowed that provides results with less contrast.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: contrast(50%);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2> contrast(50%)</h2>

</body>

</html>

## opacity()

It is used to apply transparency to the input image. Its 0% value indicates completely transparent, whereas the 100% value represents the original image, i.e., fully opaque.

Let's understand it by an illustration.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: opacity(40%);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2> opacity(40%)</h2>

</body>

</html>

## hue-rotate()

It applies a hue-rotation on the input image. Its perimeter value defines the number of degrees around the color circle; the image will be adjusted. Its default value is 0 degree, which represents the original image. Its maximum value is 360 degrees.

Let's understand it by an illustration.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: hue-rotate(240deg);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2> hue-rotate(240deg)</h2>

</body>

</html>

## grayscale()

It converts the input image into black and white. 0% grayscale represents the original one, whereas 100% represents completely grayscale. It converts the object colors into 256 shades of gray.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: grayscale(80%);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2> grayscale(80%)</h2>

</body>

</html>

## sepia()

It is used to transform the image into a sepia image. 0% value represents the original image, whereas the 100% value indicates the completely sepia.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

body{

text-align:center;

}

#img1 {

filter: sepia(90%);

}

</style>

</head>

<body>

<img src = "tiger.png" > <h2>Original Image </h2>

<img src = "tiger.png" id = "img1"> <h2> sepia(90%)</h2>

</body>

</html>

# **CSS Images**

Images are an important part of any web application. Including a lot of images in a web application is generally not recommended, but it is important to use the images wherever they required. CSS helps us to control the display of images in web applications.

The styling of an image in CSS is similar to the styling of an element by using the borders and margins. There are multiple CSS properties such as **border**property, **height**property, **width**property, etc. that helps us to style an image.

Let's discuss the styling of images in CSS by using some illustrations.

## Thumbnail Image

The border property is used to make a thumbnail image.

**Example**

<!DOCTYPE html>

<html>

<head>

<style>

img{

border: 2px solid red;

border-radius:5px;

padding:10px;

}

h2{

color:red;

}

</style>

</head>

<body>

<h1>Thumbnail Image</h1>

<img src="jtp.png"></img>

<h2>Welcome to iHubTalent</h2>

</body>

</html>

## Transparent image

To make an image transparent, we have to use the **opacity**property. The value of this property lies between 0.0 to 1.0, respectively.

**Example**

<!DOCTYPE html>

<html>

<head>

<style>

img{

border: 2px solid red;

border-radius:5px;

padding:10px;

opacity:0.3;

}

h2{

color:red;

}

</style>

</head>

<body>

<h1>Transparent Image</h1>

<img src="jtp.png"></img>

<h2>Welcome to iHubTalent</h2>

</body>

</html>

## Rounded image

The **border-radius**property sets the radius of the bordered image. It is used to create the rounded images. The possible values for the rounded corners are given as follows:

* **border-radius:** It sets all of the four border-radius property.
* **border-top-right-radius:** It sets the border of the top-right corner.
* **border-top-left-radius:** It sets the border of the top-left corner.
* **border-bottom-right-radius:** It sets the border of the bottom-right corner.
* **border-bottom-left-radius:** It sets the border of the bottom-left corner.

**Example**

<!DOCTYPE html>

<html>

<head>

<style>

#img1{

border: 2px solid green;

border-radius:10px;

padding:5px;

}

#img2{

border: 2px solid green;

border-radius:50%;

padding:5px;

}

h2{

color:red;

}

</style>

</head>

<body>

<h1>Rounded Image</h1>

<img src="jtp.png" id="img1"></img>

<h2>Welcome to iHubTalent</h2>

<h1>Circle Image</h1>

<img src="jtp.png" id="img2"></img>

<h2>Welcome to iHubTalent </h2>

</body> </html>

## Responsive Image

It automatically adjusts to fit on the screen size. It is used to adjust the image to the specified box automatically.

**Example**

<!DOCTYPE html>

<html>

<head>

<style>

#img1{

max-width:100%;

height:auto;

}

h2{

color:red;

}

</style>

</head>

<body>

<h1>Responsive image</h1>

<h2>You can resize the browser to see the effect</h2>

<img src="jtp.png" id="img1" width="1000" height="300"></img>

<h2>Welcome to iHubTalent</h2>

</body>

</html>

## Center an Image

We can center an image by using the **left-margin** and **right-margin** property. We have to set these properties to **auto**in order to make a block element.

**Example**

<!DOCTYPE html>

<html>

<head>

<style>

#img1{

margin-left:auto;

margin-right:auto;

display:block;

}

h1,h2{

text-align:center;

}

</style>

</head>

<body>

<h1>Center image</h1>

<img src="jtp.png" id="img1"></img>

<h2>Welcome to iHubTalent</h2>

</body>

</html>

# **CSS Overflow**

The **CSS overflow property** specifies how to handle the content when it overflows its block level container.

We know that every single element on a page is a rectangular box and the size, positioning and behavior of these boxes are controlled via CSS.

Let's take an example: If you don't set the height of the box, it will grow as large as the content. But if you set a specific height or width of the box and the content inside cannot fit then what will happen. The CSS overflow property is used to overcome this problem. It specifies whether to clip content, render scroll bars, or just display content.

CSS Overflow property values

|  |  |
| --- | --- |
| **Value** | **Description** |
| visible | It specifies that overflow is not clipped. it renders outside the element's box.this is a default value. |
| hidden | It specifies that the overflow is clipped, and rest of the content will be invisible. |
| scroll | It specifies that the overflow is clipped, and a scroll bar is used to see the rest of the content. |
| auto | It specifies that if overflow is clipped, a scroll bar is needed to see the rest of the content. |
| inherit | It inherits the property from its parent element. |
| initial | It is used to set the property to its initial value. |

CSS Overflow Example

Let's see a simple CSS overflow property.

<!DOCTYPE html>

<html>

<head>

<style>

div.scroll {

background-color: #00ffff;

width: 100px;

height: 100px;

overflow: scroll;

}

div.hidden {

background-color: #00FF00;

width: 100px;

height: 170px;

overflow: hidden;

}

</style>

</head>

<body>

<p>The overflow property specifies what to do if the content of an element exceeds the size of the element's box.</p>

<p>overflow:scroll</p>

<div class="scroll">You can use the overflow property when you want to have better control of the layout.

The default value is visible.</div>

<p>overflow:hidden</p>

<div class="hidden">You can use the overflow property when you want to have better control of the layout.

The default value is visible.</div>

</body>

</html>

# **CSS Padding**

**CSS Padding property** is used to define the space between the element content and the element border.

It is different from CSS margin in the way that CSS margin defines the space around elements. CSS padding is affected by the background colors. It clears an area around the content.

Top, bottom, left and right padding can be changed independently using separate properties. You can also change all properties at once by using shorthand padding property.

## CSS Padding Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| padding | It is used to set all the padding properties in one declaration. |
| padding-left | It is used to set left padding of an element. |
| padding-right | It is used to set right padding of an element. |
| padding-top | It is used to set top padding of an element. |
| padding-bottom | It is used to set bottom padding of an element. |

## CSS Padding Values

|  |  |
| --- | --- |
| **Value** | **Description** |
| length | It is used to define fixed padding in pt, px, em etc. |
| % | It defines padding in % of containing element. |

## CSS Padding Example

<!DOCTYPE html>

<html>

<head>

<style>

p {

background-color: pink;

}

p.padding {

padding-top: 50px;

padding-right: 100px;

padding-bottom: 150px;

padding-left: 200px;

}

</style>

</head>

<body>

<p>This is a paragraph with no specified padding.</p>

<p class="padding">This is a paragraph with specified paddings.</p>

</body>

</html>

# **CSS Position**

The **CSS position property** is used to set position for an element. it is also used to place an element behind another and also useful for scripted animation effect.

You can position an element using the top, bottom, left and right properties. These properties can be used only after position property is set first. A position element's computed position property is relative, absolute, fixed or sticky.

Let's have a look at following CSS positioning:

1. CSS Static Positioning
2. CSS Fixed Positioning
3. CSS Relative Positioning
4. CSS Absolute Positioning

## 1) CSS Static Positioning

This is a by default position for HTML elements. It always positions an element according to the normal flow of the page. It is not affected by the top, bottom, left and right properties.

## 2) CSS Fixed Positioning

The fixed positioning property helps to put the text fixed on the browser. This fixed test is positioned relative to the browser window, and doesn't move even you scroll the window.

<!DOCTYPE html>

<html>

<head>

<style>

p.pos\_fixed {

position: fixed;

top: 50px;

right: 5px;

color: blue;

}

</style>

</head>

<body>

<p>Some text...</p><p>Some text...</p><p>Some text...</p><p>........</p><p>.... ...</p

><p>........</p><p>........</p><p>........</p><p>........</p>

<p>........ </p><p>........</p><p>........</p><p>........</p><p>........</p>

<p>........</p><p>........</p><p>Some text...</p><p>Some text...</p><p>Some text...</p>

<p class="pos\_fixed">This is the fix positioned text.</p>

</body>

</html>

## 3) CSS Relative Positioning

The relative positioning property is used to set the element relative to its normal position.

<!DOCTYPE html>

<html>

<head>

<style>

h2.pos\_left {

position: relative;

left: -10px;

}

h2.pos\_right {

position: relative;

left: 30px;

}

</style>

</head>

<body>

<h2>This is a heading with no position</h2>

<h2 class="pos\_left">This heading is positioned left according to its normal position</h2>

<h2 class="pos\_right">This heading is positioned right according to its normal position</h2>

<p>The style "left:-30px" subtracts 30 pixels from the element's original left position.</p>

<p>The style "left:30px" adds 30 pixels to the element's original left position.</p>

</body>

</html>

## 4) CSS Absolute Positioning

The absolute positioning is used to position an element relative to the first parent element that has a position other than static. If no such element is found, the containing block is HTML.

With the absolute positioning, you can place an element anywhere on a page.

<!DOCTYPE html>

<html>

<head>

<style>

h2 {

position: absolute;

left: 150px;

top: 250px;

}

</style>

</head>

<body>

<h2>This heading has an absolute position</h2>

<p> The heading below is placed 150px from the left and 250px from the top of the page.</p>

</body>

</html>

All CSS Position Properties

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **property** | **description** | **values** |
| 1) | bottom | It is used to set the bottom margin edge for a positioned box. | auto, length, %, inherit |
| 2) | clip | It is used to clip an absolutely positioned element. | shape, auto, inherit |
| 3) | cursor | It is used to specify the type of cursors to be displayed. | url, auto, crosshair, default, pointer, move, e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize, text, wait, help |
| 4) | left | It sets a left margin edge for a positioned box. | auto, length, %, inherit |
| 5) | overflow | This property is used to define what happens if content overflow an element's box. | auto, hidden, scroll, visible, inherit |
| 6) | position | It is used to specify the type of positioning for an element. | absolute, fixed, relative, static, inherit |
| 7) | right | It is used to set a right margin edge for a positioned box. | auto, length, %, inherit |
| 8) | top | It is used to set a top margin edge for a positioned box. | auto, length, %, inherit |
| 9) | z-index | It is used to set stack order of an element. | number, auto, inherit |

# **CSS Vertical Align**

The CSS vertical align property is used to define the vertical alignment of an inline or table-cell box. It is the one of the self-explanatory property of CSS. It is not very easy property for beginners.

## What it does

1. It is applied to inline or inline-block elements.
2. It affects the alignment of the element, not its content. (except table cells)
3. When it applied to the table cells, it affect the cell contents, not the cell itself.

## CSS Vertical Align Values

|  |  |
| --- | --- |
| **value** | **description** |
| baseline | It aligns the baseline of element with the baseline of parent element. This is a default value. |
| length | It is used to increase or decrease length of the element by the specified length. negative values are also allowed. |
| % | It is used to increase or decrease the element in a percent of the "line-height" property. negative values are allowed. |
| sub | It aligns the element as if it was subscript. |
| super | It aligns the element as if it was superscript. |
| top | It aligns the top of the element with the top of the tallest element on the line. |
| bottom | It aligns the bottom of the element with the lowest element on the line. |
| text-top | the top of the element is aligned with the top of the parent element's font. |
| middle | the element is placed in the middle of the parent element. |
| text-bottom | the bottom of the element is aligned with the bottom of the parent element's font. |
| initial | It sets this property to Its default value. |
| inherit | inherits this property from Its parent element. |

## CSS Vertical Align Example

<!DOCTYPE html>

<html>

<head>

<style>

img.top {

vertical-align: text-top;

}

img.bottom {

vertical-align: text-bottom;

}

</style>

</head>

<body>

<p><img src="good-morning.jpg" alt="Good Morning Friends"/> This is an image with a default alignment.</p>

<p><img src="good-morning.jpg" class="top" alt="Good Morning Friends"/> This is an image with a text-top alignment.</p>

<p><img src="good-morning.jpg" class="bottom" alt="Good Morning Friends"/> This is an image with a text-bottom alignment.</p>

</body>

</html>

# **CSS White Space**

The **CSS white space property** is used to specify how to display the content within an element. It is used to handle the white spaces inside an element.

## CSS White Space values

There are many white space values that can be used to display the content inside an element.

|  |  |
| --- | --- |
| **Value** | **Description** |
| normal | This is a default value. in this value, text is wrapped when necessary. sequences of white space will collapse into a single whitespace. |
| nowrap | Sequences of white space will collapse into a single whitespace. in this value, text will never wrap to the next line and only break when <br> tag is used. |
| pre | Whitespace is preserved by the browser. it is act like html <pre> tag. text will only wrap on line breaks. |
| pre-line | Sequences of white space will collapse into a single whitespace. texts are wrapped when necessary, and on line break. |
| pre-wrap | Whitespace is preserved by the browser. texts are wrapped when necessary, and on line break. |
| initial | It sets this property to its default value. |
| inherit | It inherits this property from its parent element. |

## CSS White Space Example

<!DOCTYPE html>

<html>

<head>

<style>

p {

white-space: nowrap;

}

</style>

</head>

<body>

<p>

Write some text..Write some text..Write some text..

Write some text..Write some text..Write some text..

Write some text..Write some text..Write some text..

Write some text..Write some text..Write some text..

Write some text..Write some text..Write some text..

</p>

</body>

</html>

# **CSS Width**

The **CSS width property** is used to set the width of the content area of an element.

It does not include padding borders or margins. It sets width of the area inside the padding, border, and margin of the element.

## CSS width values

|  |  |
| --- | --- |
| **Value** | **Description** |
| auto | It is a default value. it is used to calculate the width. |
| length | It is used to define the width in px, cm etc. |
| % | It defines the width of the containing block in %. |
| initial | It is used to set the property to its default value. |
| inherit | It is used to inherit the property from its parent element. |

## CSS Width Example: width in px

<!DOCTYPE html>

<html>

<head>

<style>

img.normal {

width: auto;

}

img.big {

width: 150px;

}

p.ex {

height: 150px;

width: 150px;

}

</style>

</head>

<body>

<img class="normal" src="good-morning.jpg" width="95" height="84"><br>

<img class="big" src="good-morning.jpg" width="95" height="84">

<p class="ex">The height and width of this paragraph is 150px.</p>

<p>This is a paragraph.</p>

</body> </html>

## Practice CSS Width Example: width in %

#### **Note: You can also use the "min-width" and "max-width" property to control the size of image.**

# **CSS Word Wrap**

**CSS word wrap property** is used to break the long words and wrap onto the next line. This property is used to prevent overflow when an unbreakable string is too long to fit in the containing box.

## CSS Word Wrap Values

|  |  |
| --- | --- |
| **Value** | **Description** |
| normal | This property is used to break words only at allowed break points. |
| break-word | It is used to break unbreakable words. |
| initial | It is used to set this property to its default value. |
| inherit | It inherits this property from its parent element. |

### **CSS Word Wrap Example**

<!DOCTYPE html>

<html>

<head>

<style>

p.test {

width: 11em;

background-color: #00ffff;

border: 1px solid #000000;

padding:10px;

word-wrap: break-word;

}

</style>

</head>

<body>

<p class="test"> In this paragraph, there is a very long word: iamsooooooooooooooooooooooooooooooolongggggggggggggggg.The long word will break and wrap to the next line.</p>

</body>

</html>

# **Box-shadow CSS**

It is used to add shadow-like effects around the frame of an element.

### **Syntax**

1. box-shadow: h-offset v-offset blur spread color |inset|inherit|initial|none;

Let's understand property values.

**h-offset:** It horizontally sets the shadow position. Its positive value will set the shadow to the right side of the box. Its negative value is used to set the shadow on the left side of the box.

**v-offset:** Unlike the **h-offset**, it is used to set the shadow position vertically. The positive value in it sets the shadow below the box, and the negative value sets the shadow above of the box.

**blur:** As its name implies, it is used to blur the box-shadow. This attribute is optional.

**spread:** It sets the shadow size. The spread size depends upon the spread value.

**color:** As its name implies, this attribute is used to set the color of the shadow. It is an optional attribute.

**inset:** Normally, the shadow generates outside of the box, but by using inset, the shadow can be created within the box.

**initial:** It is used to set the property of the box-shadow to its default value.

**inherit:** it is inherited from its parent.

**none:** It is the default value that does not include any shadow property.

Let's understand the above attributes by using an illustration.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS box-shadow Property</title>

<style>

div

{

border: 1px solid;

padding: 10px;

}

#hvb

{

/\* box-shadow: h-offset v-offset blur \*/

box-shadow: 5px 10px 10px;

}

#spr

{

/\* box-shadow: h-offset v-offset blur spread \*/

box-shadow: 5px 10px 10px 10px;

}

#col

{

/\* box-shadow: h-offset v-offset blur spread color \*/

box-shadow: 5px 10px 10px 10px orange;

}

#ins

{

/\* box-shadow: h-offset v-offset blur spread color inset \*/

box-shadow: 5px 10px 10px 10px orange inset;

}

#init

{

/\* box-shadow: initial \*/

box-shadow: initial;

}

#non

{

/\* box-shadow: none \*/

box-shadow: none;

}

</style>

</head>

<body>

<div id = "hvb">

<h1>It is a shadow box that has h-offset, v-offset and blur attributes.</h1>

</div>

<br><br>

<div id = "spr">

<h1>It is a box that includes the spread attribute.</h1>

</div>

<br><br>

<div id = "col">

<h1>It is a box that includes the color attribute.</h1>

</div>

<br><br>

<div id = "ins">

<h1>It is a box that includes the inset attribute.</h1>

</div>

<br><br>

<div id = "init">

<h1>It is a box that includes the initial attribute.</h1>

</div>

<br><br>

<div id = "non">

<h1>It is a box that includes the default attribute i.e. none.</h1>

</div>

</body>

</html>

# **CSS Text-shadow**

As its name implies, this CSS property adds shadows to the text. It accepts the comma-separated list of shadows that applied to the text. It's default property is none. It applies one or more than one text-shadow effect on the element's text content.

Let's see the syntax of text-shadow property.

**Syntax**

1. text-shadow: h-shadow v-shadow blur-radius color| none | initial | inherit;

**Values**

**h-shadow:** It is the required value. It specifies the position of the horizontal shadow and allows negative values.

**v-shadow:** It is also the required value that specifies the position of the vertical shadow. It does not allow negative values.

**blur-radius:** It is the blur-radius, which is an optional value. Its default value is 0.

**color:** It is the color of the shadow and also an optional value.

**none:** It is the default value, which means no shadow.

**initial:** It is used to set the property to its default value.

**inherit:** It simply inherits the property from its parent element.

Let's understand it by using some illustrations.

### **Example- Simple shadow**

<!DOCTYPE html>

<html>

<head>

<title> font-weight property </title>

<style>

p.simple{

text-shadow: 3px 3px red;

}

</style>

</head>

<body>

<p class="simple">

Simple Shadow

</p>

</body>

</html>

### **Example- Fuzzy shadow**

<!DOCTYPE html>

<html>

<head>

<title> font-weight property </title>

<style>

p.fuzzy{

text-shadow: 3px 3px 3px violet;

font-size:30px;

text-align:center;

}

</style>

</head>

<body>

<p class="fuzzy">

Fuzzy Shadow

</p>

</body>

</html>

### **Example- Multiple Shadows**

<!DOCTYPE html>

<html>

<head>

<title> font-weight property </title>

<style>

p.multi{

text-shadow: -3px -3px 3px blue, 3px 3px 3px red;

font-size:30px;

text-align:center;

}

</style>

</head>

<body>

<p class="multi">

Multiple Shadows

</p>

</body>

</html>

### **Example- Glow Effect**

<!DOCTYPE html>

<html>

<head>

<title> font-weight property </title>

<style>

.multi{

text-shadow: 0 0 .1em cyan;

background-color: black;

font-size:50px;

text-align:center;

}

</style>

</head>

<body>

<div class="multi">

Glow Effect

</div>

</body>

</html>

# **CSS text-transform**

This CSS property allows us to change the case of the text. It is used to control the text capitalization. This CSS property can be used to make the appearance of text in all-lowercase or all-uppercase or can convert the first character of each word to uppercase.

**Syntax**

1. text-transform: capitalize| uppercase | lowercase | none | initial | inherit;

Let's discuss its property values along with an example.

## capitalize

It transforms the first character of each word to uppercase. It will not capitalize the first letter after the number. It only affects the first letters of the words instead of changing the rest of the letters in the word.

If we **apply the capitalize**property on a word that already has capital letters, then the letters of that word will not switch to lowercase.

The illustration of this property is given below.

**Syntax**

1. text-transform: capitalize;

**Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS text-transform Property

</title>

<style>

body{

text-align:center;

}

h1 {

color: blue;

}

p{

text-transform: capitalize;

}

</style>

</head>

<body>

<center>

<h1>CSS text-transform property</h1>

<h2>text-transform: capitalize</h2>

<p>hello world</p>

<p>WELCOME to the iHubtalent</p>

</body>

</html>

## uppercase

As its name implies, it transforms all characters of the word into uppercase.

**Syntax**

1. text-transform: uppercase;

**Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS text-transform Property

</title>

<style>

body{

text-align:center;

}

h1 {

color: blue;

}

p{

text-transform: uppercase;

}

</style>

</head>

<body>

<center>

<h1>CSS text-transform property</h1>

<h2>text-transform: uppercase</h2>

<p>hello world</p>

<p>WELCOME to the iHubTalent</p>

</body>

</html>

## lowercase

It transforms all characters of the word into lowercase.

**Syntax**

1. text-transform: lowercase;

**Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS text-transform Property

</title>

<style>

body{

text-align:center;

}

h1 {

color: blue;

}

p{

text-transform: lowercase;

}

</style>

</head>

<body>

<center>

<h1>CSS text-transform property</h1>

<h2>text-transform: lowercase</h2>

<p>HELLO WORLD</p>

<p>WELCOME TO THE IHUBTALENT</p>

</body>

</html>

## none

It is the default value that has no capitalization. It renders the text as it is.

**Syntax**

1. text-transform: none;

**Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS text-transform Property

</title>

<style>

body{

text-align:center;

}

h1 {

color: blue;

}

p{

text-transform: none;

}

</style>

</head>

<body>

<center>

<h1>CSS text-transform property</h1>

<h2>text-transform: none</h2>

<p>Hello World</p>

<p>Welcome to the iHubTalent.</p>

</body> </html>

# **CSS Outline**

CSS outline is just like CSS border property. It facilitates you to draw an extra border around an element to get visual attention.

It is as easy as to apply as a border.

**See this example:**

<!DOCTYPE html>

<html>

<style type="text/css">

.box {

background-color: #eee;

outline: 3px solid red;

border: 3px solid lightgreen;

padding: 5px 10px;

}

</style>

<div class="box">Welcome to iHubTalent</div>

</body>

</html>

## Difference between Border and Outline

At first glance, border and outline look similar, but there are some very important differences between them:

* It is not possible to apply a different outline width, style and color for the four sides of an element while in border; you can apply the provided value for all four sides of an element.
* The border is a part of element's dimension while the outline is not the part of element's dimension. Means, it doesn't matter how thick an outline you apply to the element, the dimensions of it won't change.

The outline property is a shorthand property. It can be divided into outline-width, outline-style and outline-color properties. It facilitates you to use any of the property alone, if you need it.

**See this example:**

<!DOCTYPE html>

<html>

<style type="text/css">

.box {

background-color: #eee;

border: 3px solid Lightgreen;

padding: 5px 10px;

outline-width: 3px;

outline-style: solid;

outline-color: red;

}

</style>

<div class="box">Welcome to iHub</div>

</body>

</html>

In the above example, you can see the three outline properties:

**Outline-width:**It is similar to margin and padding. It can be either an absolute value or a relative value or one of the predefined outline width values i.e. thin, medium or thick. It is preferred to use either an absolute value or a relative value because different browsers interpret differently while using predefined outline width values like thin, medium or thick.

**Outline-color:**It specifies the color that you use for the outline. It supports all the colors available in HTML and CSS.

**Outline-style:**In the above example, we have used only solid outline style while there are a lot of outline style i.e. hidden, dotted, dashed, solid, double, groove, ridge, inset and outset.

Let's take an example to demonstrate the usage of different outline-styles.

**See this example:**

<!DOCTYPE html>

<html>

<style type="text/css">

.box {

outline-color: red;

outline-width: 4px;

margin: 10px;

float: left;

border: 2px solid lightgreen;

padding: 5px 10px;

}

</style>

<div class="box" style="outline-style: dashed;">This is dashed outline.</div>

<div class="box" style="outline-style: dotted;">This is dotted outline.</div>

<div class="box" style="outline-style: double;">This is double outline.</div>

<div class="box" style="outline-style: groove;">This is groove outline.</div>

<div class="box" style="outline-style: inset;">This is inset outline.</div>

<div class="box" style="outline-style: outset;">This is outset outline.</div>

<div class="box" style="outline-style: ridge;">This is ridge outline.</div>

<div class="box" style="outline-style: solid;">This is solid outline.</div>

</body>

</html>

## Outline offset

The outline offset is used to create a distance between outline and border.

It takes a CSS length unit and the empty space between the border and the outline will be transparent and then it takes the background color of the parent element. So you can see a visible difference between outline and border.

Let's take an example to see the difference between outline and border.

**See this example:**

<!DOCTYPE html>

<html>

<style type="text/css">

.box {

background-color: #eee;

outline: 3px solid red;

outline-offset: 6px;

border: 3px solid Lightgreen;

padding: 5px 10px;

}

</style>

<div class="box">Welcome to iHub</div>

</body>

</html>

# **CSS Visibility**

The CSS visibility property is used to specify whether an element is visible or not.

**Note:** An invisible element also take up the space on the page. By using display property you can create invisible elements that don't take up space.

**Syntax:**

1. visibility: visible|hidden|collapse|initial|inherit;

## CSS Visibility Parameters

**visible:**It is the by default value. It specifies that the element is visible.

**hidden:**It specifies that the element is invisible (but still takes up space).

**collapse:**It is used only for table elements. It is used to remove a row or column, but it does not affect the table layout.

The space taken up by the row or column will be available for other content.

If collapse is used on other elements, it renders as "hidden"

**initial:**It is used to set this property to its default value.

**inherit:**It is used to inherit this property from its parent element.

## CSS Visibility Example

<!DOCTYPE html>

<html>

<head>

<style>

h1.visible {

visibility: visible

}

h1.hidden {

visibility: hidden

}

</style>

</head>

<body>

<h1 class="visible">I am visible</h1>

<h1 class="hidden">I am invisible</h1>

<p><strong>Note:</strong> An invisible element also take up the space on the page. By using display property you can create invisible elements that don?t take up space.</p>

</body>

</html>

## JavaScript Syntax:

1. object.style.visibility="hidden"

**See the JavaScript example:**

<!DOCTYPE html>

<html>

<head>

<style>

#myDIV {

margin: auto;

width: 400px;

height: 200px;

background-color: lightpink;

border: 1px solid black;

}

</style>

</head>

<body>

<p>Click the "Try it" button to set the visibility property and hide the div element.</p>

<button onclick="myFunction()">Try it</button>

<div id="myDIV">This is my DIV element.</div>

<p><strong>Note:</strong> An invisible element also take up the space on the page. </p>

<script>

function myFunction() {

document.getElementById("myDIV").style.visibility = "hidden";

}

</script>

</body></html>

# **CSS Counters**

CSS counters are similar to variables. These are maintained by CSS and those values can be incremented by CSS rules to track how many times they are used.

CSS counters facilitate simple CSS based incrementing and display of a number for generated content.

## CSS Counter Properties

Following is a list of properties that are used with CSS counter:

* **counter-reset:** It is used to create or reset a counter.
* **counter-increment:** It is used to increment the counter value.
* **content:** It is used to insert generated content.
* **counter() or counters() function:** It is used to add the value of a counter to an element.

#### **Note: Before using a CSS counter, it must be created with counter-reset.**

## CSS Counter Example

Let's take an example to create a counter for a page and increment the counter value for each next element.

**See this example:**

<!DOCTYPE html>

<html>

<head>

<style>

body {

counter-reset: section;

}

h2::before {

counter-increment: section;

content: "Section " counter(section) ": ";

}

</style>

</head>

<body>

<h1>Example of CSS Counters:</h1>

<h2>Java Tutorial</h2>

<h2>HTML Tutorial</h2>

<h2>CSS Tutorial</h2>

<h2>Oracle Tutorial</h2>

<p><strong>Note:</strong> IE8 supports these properties only if a !DOCTYPE is specified.</p>

</body>

</html>

#### **Note: In the above example you can see that a counter is created for the page in the body selector and it increments the counter value for each <h2> element and adds "Section <value of the counter>:" to the beginning of each <h2> element.**

## CSS Nesting Counters

You can also create counters within the counter. It is called nesting of a counter. Let's take an example to demonstrate nesting counter.

**See this example:**

<!DOCTYPE html>

<html>

<head>

<style>

body {

counter-reset: section;

}

h1 {

counter-reset: subsection;

}

h1::before {

counter-increment: section;

content: "Section " counter(section) ". ";

}

h2::before {

counter-increment: subsection;

content: counter(section) "." counter(subsection) " ";

}

</style>

</head>

<body>

<h1>Java tutorials:</h1>

<h2>Core Java tutorial</h2>

<h2>Servlet tutorial</h2>

<h2>JSP tutorial</h2>

<h2>Spring tutorial</h2>

<h2>Hibernate tutorial</h2>

<h1>Web technology tutorials:</h1>

<h2>HTML tutorial</h2>

<h2>CSS tutorial</h2>

<h2>jQuery tutorial</h2>

<h2>Bootstrap tutorial</h2>

<h1>Database tutorials:</h1>

<h2>SQL tutorial</h2>

<h2>MySQL tutorial</h2>

<h2>PL/SQL tutorial</h2>

<h2>Oracle tutorial</h2>

<p><strong>Note:</strong> IE8 supports these properties only if a !DOCTYPE is specified.</p>

</body>

</html>

#### **Note: In the above example you can see that a counter is created for the section and another nesting counter named subsection is created within section.**

# **CSS clearfix**

A clear float (or clearfix) is a way for an element to fix or clear the child elements so that we do not require to add additional markup. It resolves the error which occurs when more than one floated elements are stacked next to each other.

Suppose if we set the position of a sidebar to the left of the main content block, but we get the elements collapse and overlap on each other. We can understand it as if a child element is floated and taller than its parent element; it will overflow outside of its container.

To overcome this, we can use the **overflow: auto;** property to the containing element.

Let us try to understand it by using an example.

### **Example**

In this example, the image is floated and taller than the element containing it, so that it overflows outside of its container.

Now, we are creating a class **jtp** and add the **overflow: auto;** property to the corresponding element.

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 3px solid red;

padding: 5px;

background-color:pink;

font-size:20px;

}

img{

float: right;

border: 3px solid blue;

}

p{

font-size:20px;

clear:right;

}

.jtp{

overflow: auto;

}

</style>

</head>

<body>

<h1>Without Clearfix</h1>

<div>

<img class="img1" src="jtp.png">

Welcome to the iHubTalent.com. Here, you can find the best tutorials that are easy to read and help you to clear your concepts.

</div>

<p>Use the overflow:auto; CSS property</p>

<h1>With Clearfix</h1>

<div class="jtp">

<img class="img2" src="jtp.png">

Welcome to the iHubTalent.com. Here, you can find the best tutorials that are easy to read and help you to clear your concepts.

</div>

</body>

</html>

The above clearfix method works well as long as we manage the paddings and margins. But a modern way to clearfix is safer to use.

# **CSS icons**

Icons can be defined as the images or symbols used in any computer interface refer to an element. It is a graphical representation of a file or program that helps the user to identify about the type of file quickly.

Using the icon library is the easiest way to add icons to our HTML page. It is possible to format the library icons by using CSS. We can customize the icons according to their color, shadow, size, etc.

There are given some of the icon libraries such as **Bootstrap icons, Font Awesome icons,** and **Google icons** that can be used in CSS easily. There is no need to install or download the libraries mentioned above.

Let's discuss the above-mentioned libraries of icons.

## Font Awesome icons

To use this library in our HTML page, we need to add the following link within the **<head></head>** section.

1. **<link** rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"**>**

Let's understand it with an illustration.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Icons</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>

body{

text-align:center;

background-color:lightblue;

}

.fa{

color:red;

font-size:50px;

margin:10px;

}

</style>

</head>

<body style="text-align:center">

<h1>Font Awesome Library</h1>

<i class="fa fa-cloud"></i>

<i class="fa fa-file"></i>

<i class="fa fa-heart"></i>

<i class="fa fa-bars"></i>

<i class="fa fa-car"</i>

</body>

</html>

## Bootstrap icons

As similar to the font awesome library, we can add the bootstrap icons in our HTML page using the link given below in the **<head></head>** section.

1. **<link** rel="stylesheet"
2. href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"**>**

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Icons</title>

<link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<style>

body{

text-align:center;

background-color:lightblue;

}

.glyphicon{

color:red;

font-size:50px;

margin:10px;

}

</style>

</head>

<body style="text-align:center">

<h1>Bootstrap icons</h1>

<i class="glyphicon glyphicon-cloud"></i>

<i class="glyphicon glyphicon-file"></i>

<i class="glyphicon glyphicon-heart"></i>

<i class="glyphicon glyphicon-user"></i>

<i class="glyphicon glyphicon-thumbs-up"></i>

<i class="glyphicon glyphicon-remove"></i>

<i class="glyphicon glyphicon-envelope"></i>

</body> </html>

## Google icons

As similar to the above libraries, we can add it in our HTML page simply by adding the link given below in the **<head></head>** section.

### **Example**

1. **<link** rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons"**>**

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Icons</title>

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

<style>

body{

text-align:center;

background-color:lightblue;

}

.material-icons{

color:red;

font-size:50px;

margin:10px;

}

</style>

</head>

<body style="text-align:center">

<h1>Google icons</h1>

<i class="material-icons">cloud</i>

<i class="material-icons">attachment</i>

<i class="material-icons">computer</i>

<i class="material-icons">favorite</i>

<i class="material-icons">traffic</i>

</body>

</html>

# **CSS justify-content**

This CSS property is used to align the items of the flexible box container when the items do not use all available space on the main-axis (horizontally). It defines how the browser distributes the space around and between the content items.

This CSS property can't be used to describe containers or items along the vertical axis. To align the items vertically, we have to use the **align-items** property.

### **Syntax**

1. justify-content: center | flex-start | flex-end | space-around | space-evenly | space-between | initial | inherit;

The default value of this property is **flex-start.** Let's understand its property values in detail.

### **Property values**

* **center:** As its name implies, it set the position of items at the center of the container.
* **flex-start:** It is the default value that positioned the items at the beginning of the container.
* **flex-end:** It set the position of items at the end of the container.
* **space-around:** It positioned the items with equal spacing from each other. It evenly distributes the items in the line along with the same space around them.
* **space-between:** Items are evenly spaced in which the first item is at the beginning, and the last item is at the end.
* **space-evenly:** It also positioned the items with equal space, but the spacing from the corners differs.

Let's understand the above values by using an illustration.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS filter property</title>

<style>

#flexstart{

display:flex;

justify-content: flex-start;

}

#flexend{

display:flex;

justify-content: flex-end;

}

#cent{

display:flex;

justify-content: center;

}

#evenly{

display:flex;

justify-content: space-evenly;

}

#around{

display:flex;

justify-content: space-around;

}

#between{

display:flex;

justify-content: space-between;

}

.flex-item{

width:50px;

height:50px;

margin:5px;

padding:5px;

color:white;

font-size:2em;

font-weight:bold;

text-align:center;

border: 1px solid black;

background-color:blue;

}

</style>

</head>

<body>

<div id="flexstart">

<h1>flex-start</h1>

<div class="flex-item">1</div>

<div class="flex-item">2</div>

<div class="flex-item">3</div>

<div class="flex-item">4</div>

<div class="flex-item">5</div>

</div>

<div id="flexend">

<h1>flex-end</h1>

<div class="flex-item">1</div>

<div class="flex-item">2</div>

<div class="flex-item">3</div>

<div class="flex-item">4</div>

<div class="flex-item">5</div>

</div>

<div id="cent">

<h1>center</h1>

<div class="flex-item">1</div>

<div class="flex-item">2</div>

<div class="flex-item">3</div>

<div class="flex-item">4</div>

<div class="flex-item">5</div>

</div>

<h1>space-evenly</h1>

<div id="evenly">

<div class="flex-item">1</div>

<div class="flex-item">2</div>

<div class="flex-item">3</div>

<div class="flex-item">4</div>

<div class="flex-item">5</div>

</div>

<h1>space-around</h1>

<div id="around">

<div class="flex-item">1</div>

<div class="flex-item">2</div>

<div class="flex-item">3</div>

<div class="flex-item">4</div>

<div class="flex-item">5</div>

</div>

<h1>space-between</h1>

<div id="between">

<div class="flex-item">1</div>

<div class="flex-item">2</div>

<div class="flex-item">3</div>

<div class="flex-item">4</div>

<div class="flex-item">5</div>

</div>

</body>

</html>

# **CSS text-decoration**

It is a CSS property that decorates the content of the text. It adds lines under, above, and through the text. It sets the appearance of decorative lines on text. This CSS property decorates the text with several kinds of lines. This is shorthand for **text-decoration-line, text-decoration-color,** and **text-decoration-style.**

The syntax of this CSS property is given as follows-

### **Syntax**

1. text-decoration: text-decoration-line text-decoration-color text-decoration-style|initial|inherit;

Let's discuss its property values along with an example.

## text-decoration-line

It sets the kind of text-decoration like **overline, underline,** or **line-through.** It can be used to add a combination of lines to the selected text.

### **Example**

In this example, we are going to use the values **underline, overline, and line-through.** We will also see how to use these values simultaneously.

<!DOCTYPE html>

<html>

<head>

<title>text-decoration</title>

<style>

h1 {

color: red;

}

h2{

color: blue;

}

body {

text-align: center;

}

p{

font-size: 30px;

}

#p1 {

text-decoration: underline;

}

#p2 {

text-decoration: line-through;

}

#p3 {

text-decoration: overline;

}

#p4 {

text-decoration: overline underline line-through;

}

</style>

</head>

<body>

<h1>Welcome to the iHubTalent.com</h1>

<h2>text-decoration: text-decoration-line;</h2>

<div>

<p id="p1">This is underline</p>

<p id="p2">This is line-through</p>

<p id="p3">This is overline</p>

<p id="p4">This is the combination of lines</p>

</div> </body> </html>

## text-decoration-style

This property is used to set the style of the line. Its values are **solid, dotted, wavy, double,** and **dashed.**

The following example explains this property more clearly.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>text-decoration</title>

<style>

h1 {

color: red;

}

h2{

color: blue;

}

body {

text-align: center;

}

p{

font-size: 30px;

}

#p1 {

text-decoration: underline double;

}

#p2 {

text-decoration: line-through dashed;

}

#p3 {

text-decoration: dotted overline;

}

#p4 {

text-decoration: lightblue wavy overline underline line-through;

color:red;

}

</style>

</head>

<body>

<h1>Welcome to the iHub.com</h1>

<h2>text-decoration: text-decoration-line text-decoration-style;</h2>

<div>

<p id="p1">This is double underline</p>

<p id="p2">This is dashed line-through</p>

<p id="p3">This is dotted overline</p>

<p id="p4">This is the wavy combination of lines</p>

</div>

</body>

</html>

## text-decoration-color

This property is used to provide color to the decoration. Its value is any color in a valid format.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>text-decoration</title>

<style>

h1 {

color: red;

}

h2{

color: blue;

}

body {

text-align: center;

}

p{

font-size: 30px;

}

#p1 {

text-decoration: underline double cyan;

}

#p2 {

text-decoration: line-through dashed green;

}

#p3 {

text-decoration: dotted overline blue;

}

#p4 {

text-decoration: lightblue wavy overline underline line-through;

color:red;

}

</style>

</head>

<body>

<h1>Welcome to the iHubTalent.com</h1>

<h2>text-decoration: text-decoration-line text-decoration-style;</h2>

<div>

<p id="p1">This is double underline</p>

<p id="p2">This is dashed line-through</p>

<p id="p3">This is dotted overline</p>

<p id="p4">This is the wavy combination of lines</p>

</div>

</body>

</html>

# **CSS Lists**

There are various CSS properties that can be used to control lists. Lists can be classified as ordered lists and unordered lists. In ordered lists, marking of the list items is with alphabet and numbers, whereas in unordered lists, the list items are marked using bullets.

We can style the lists using CSS. CSS list properties allow us to:

* Set the distance between the text and the marker in the list.
* Specify an image for the marker instead of using the number or bullet point.
* Control the marker appearance and shape.
* Place the marker outside or inside the box that contains the list items.
* Set the background colors to list items and lists.

The CSS properties to style the lists are given as follows:

* **list-style-type:** This property is responsible for controlling the appearance and shape of the marker.
* **list-style-image:** It sets an image for the marker instead of the number or a bullet point.
* **list-style-position:** It specifies the position of the marker.
* **list-style:** It is the shorthand property of the above properties.
* **marker-offset:** It is used to specify the distance between the text and the marker. It is unsupported in IE6 or Netscape 7.

Let's understand the above properties in detail, along with an example of each.

## The list-style-type property

It allows us to change the default list type of marker to any other type such as square, circle, roman numerals, Latin letters, and many more. By default, the ordered list items are numbered with Arabic numerals (1, 2, 3, etc.), and the items in an unordered list are marked with round bullets (•).

If we set its value to **none,** it will remove the markers/bullets.

#### **Note: The list also includes the default padding and margin. To remove this, we need to add padding:0 and margin:0 to <ol> and <ul>.**

The illustration of using this property is given as follows:

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Lists</title>

<style>

.num{

list-style-type:decimal;

}

.alpha{

list-style-type:lower-alpha;

}

.roman{

list-style-type:lower-roman;

}

.circle{

list-style-type:circle;

}

.square{

list-style-type:square;

}

.disc{

list-style-type:disc;

}

</style>

</head>

<body>

<h1>

Welcome to the iHub.com

</h1>

<h2>

Ordered Lists

</h2>

<ol class="num">

<li>one</li>

<li>two</li>

<li>three</li>

</ol>

<ol class="alpha">

<li>one</li>

<li>two</li>

<li>three</li>

</ol>

<ol class="roman">

<li>one</li>

<li>two</li>

<li>three</li>

</ol>

<h2>

Unordered lists

</h2>

<ul class="disc">

<li>one</li>

<li>two</li>

<li>three</li>

</ul>

<ul class="circle">

<li>one</li>

<li>two</li>

<li>three</li>

</ul>

<ul class="square">

<li>one</li>

<li>two</li>

<li>three</li>

</ul>

</body>

</html>

## The list-style-position property

It represents whether the appearing of the marker is inside or outside of the box containing the bullet points. It includes two values.

**inside:** It means that the bullet points will be in the list item. In this, if the text goes on the second line, then the text will be wrap under the marker.

**outside:** It represents that the bullet points will be outside the list item. It is the default value.

The following example explains it more clearly.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Lists</title>

<style>

.num{

list-style-type:decimal;

list-style-position:inside;

}

.roman{

list-style-type:lower-roman;

list-style-position:outside;

}

.circle{

list-style-type:circle;

list-style-position:inside;

}

.square{

list-style-type:square;

}

.disc{

list-style-type:disc;

list-style-position:inside;

}

</style>

</head>

<body>

<h1>

Welcome to the iHub.com

</h1>

<h2>

Ordered Lists

</h2>

<ol class="num">

<li>INSIDE</li>

<li>two</li>

<li>three</li>

</ol>

<ol class="roman">

<li>OUTSIDE</li>

<li>two</li>

<li>three</li>

</ol>

<h2>

Unordered lists

</h2>

<ul class="disc">

<li>INSIDE</li>

<li>two</li>

<li>three</li>

</ul>

<ul class="circle">

<li>INSIDE</li>

<li>two</li>

<li>three</li>

</ul>

<ul class="square">

<li>DEFAULT</li>

<li>two</li>

<li>three</li>

</ul>

</body>

</html>

## The list-style-image property

It specifies an image as the marker. Using this property, we can set the image bullets. Its syntax is similar to the background-image property. If it does not find the corresponding image, the default bullets will be used.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Lists</title>

<style>

.order{

list-style-image: url(img.png);

}

.unorder{

list-style-image: url(img.png);

}

</style>

</head>

<body>

<h1>

Welcome to the iHub.com

</h1>

<h2>

Ordered Lists

</h2>

<ol class="order">

<li>one</li>

<li>two</li>

<li>three</li>

</ol>

<h2>

Unordered lists

</h2>

<ul class="unorder">

<li>one</li>

<li>two</li>

<li>three</li>

</ul>

</body>

</html>

## The list-style property

It is the shorthand property that is used to set all list properties in one expression. The order of the values of this property is type, position, and image. But if any property value is missing, then the default value will be inserted.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Lists</title>

<style>

.order{

list-style: lower-alpha inside url(img.png);

}

.unorder{

list-style: disc outside;

}

</style>

</head>

<body>

<h1>

Welcome to the iHub.com

</h1>

<h2>

Ordered Lists

</h2>

<ol class="order">

<li>one</li>

<li>two</li>

<li>three</li>

</ol>

<h2>

Unordered lists

</h2>

<ul class="unorder">

<li>one</li>

<li>two</li>

<li>three</li>

</ul>

</body>

</html>

## Styling Lists with colors

To make the lists more attractive and interesting, we can style lists with colors. The addition of anything to the <ul> or <ol> tag will affect the entire list, whereas the addition to the individual <li> tag will affect the items of the corresponding list.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Lists</title>

<style>

.order{

list-style: upper-alpha;

background: pink;

padding:20px;

}

.order li{

background: lightblue;

padding:10px;

font-size:20px;

margin:10px;

}

.unorder{

list-style: square inside;

background: cyan;

padding:20px;

}

.unorder li{

background: green;

color:white;

padding:10px;

font-size:20px;

margin:10px;

}

</style>

</head>

<body>

<h1>

Welcome to the iHub.com

</h1>

<h2>

Ordered Lists

</h2>

<ol class="order">

<li>ONE</li>

<li>TWO</li>

<li>THREE</li>

</ol>

<h2>

Unordered lists

</h2>

<ul class="unorder">

<li>ONE</li>

<li>TWO</li>

<li>THREE</li>

</ul>

</body>

</html>

# **CSS :nth-child(n) selector**

This selector is used for matching the elements based on their position regardless of the type of its parent. The **n** can either be a keyword, formula, or a number. It is used to match the elements based on their position within a group of siblings. It matches each element, which is the nth-child.

### **Syntax**

Following are the syntax of this selector:

:nth-child(n) {

    //CSS Property

}

The **"n"** in the parentheses is the argument that represents the pattern for matching elements. It can be a functional notation, even or odd.

Odd values represent the elements whose position is odd in series like 1, 3, 5, etc. Similarly, the even values represent the elements whose position is even in series like 2, 4, 6, etc.

**Functional notation (An+B)**: Functional notation represents those elements whose siblings position match the **An+B** pattern, where **A** is the integer step size, **n** is any positive integer starting from 0, and **B** is the integer offset.

Let' see some illustrations.

### **Example1**

In this example, we are using the functional notation **3n+4** that will represent the elements:

(3×0)+4 = 4, (3×1)+4 =7, and many more.

<!DOCTYPE html>

<html>

<head>

<title>CSS :nth-child Selector</title>

<style>

p:nth-child(3n+4) {

background: yellow;

color: black;

font-size:30px;

}

</style>

</head>

<body style = "text-align:center">

<h1>

Hello World

</h1>

<h2>

Welcome to the iHub

</h2>

<p>It will not affected.</p>

<p>It will be affected.</p>

<p>Not affected.</p>

<p>Not affected.</p>

<p>It will be affected.</p>

</body>

</html>

### **Example2**

In this example, we are going to use odd and even keywords that match those elements whose index is odd or even. It is to be noted that the first child index is 1.

<!DOCTYPE html>

<html>

<head>

<title>CSS :nth-child Selector</title>

<style>

p:nth-child(even) {

background: yellow;

color: black;

font-size:30px;

}

p:nth-child(odd) {

background: blue;

color: white;

font-size:20px;

</style>

</head>

<body style = "text-align:center">

<h1>

Hello World

</h1>

<h2>

Welcome to the iHub

</h2>

<p>Odd</p>

<p>Even</p>

<p>Odd</p>

<p>Even</p>

<p>Odd</p>

</body>

</html>

# **CSS sticky**

The CSS position property is used to set the position for an element. It is also used to place an item behind another element and also useful for the scripted animation effect. The **"position: sticky;"** is used to position the element based on the scroll position of the user.

This CSS property allows the elements to stick when the scroll reaches to a certain point. Depends upon the scroll position, a sticky element toggles in between **fixed** and **relative.** The element will be positioned **relative** until the specified position of offset is met in the viewport. Then, similar to **position: fixed,** the element sticks in one place.

Let us try to understand this CSS property by using an illustration.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

body{

text-align:center;

}

.stick{

position: sticky;

top:50px;

padding: 10px;

font-size:20px;

font-weight:bold;

background-color: lightblue;

border: 1px solid blue;

}

</style>

</head>

<body>

<h1>Scroll and see the effect!</h1>

<div class = "stick">Sticky Element</div>

<div style = "padding-bottom:2000px">

<h2>Hello World</h2>

<h2>Welcome to iHub.com</h2>

</div>

</body>

</html>

# **CSS background-clip**

This CSS property specifies the painting area of the background. It limits the area in which the background color or image appears by applying a clipping box. Anything outside the box will be discarded and invisible.

It sets whether the background of an element extends under the border-box, padding-box, and content-box.

### **Syntax**

1. background-clip: border-box| padding-box| content-box| inherit;

### **Possible values**

Let's understand the property values along with an example of each.

### **border-box**

It is the default value. It means that the background image and color will be drawn inside the border-box. It sets the background color, which spreads over the entire division.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: yellow;

background-clip: border-box;

text-align: center;

border:5px dotted blue;

}

h1,h2{

color: red;

}

</style>

</head>

<body>

<div>

<h1>

Welcome to the iHub.com

</h1>

<h2>

background-clip: border-box;

</h2>

</div>

</body></html>

### **padding-box**

It sets the background within the border, i.e., the background image and color are drawn inside the padding-box.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: yellow;

background-clip: padding-box;

padding: 25px;

text-align: center;

border:5px dashed blue;

}

h1,h2{

color: red;

}

</style>

</head>

<body>

<div>

<h1>

Welcome to the iHub.com

</h1>

<h2>

background-clip: padding-box;

</h2>

</div>

</body>

</html>

### **content-box**

It sets the background-color up to the content only. The background is painted inside the content box, i.e., the background image and color will be drawn in the content box.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: yellow;

background-clip: content-box;

padding: 15px;

text-align: center;

border:5px dashed blue;

}

h1,h2{

color: red;

}

</style>

</head>

<body>

<div>

<h1>

Welcome to the iHub.com

</h1>

<h2>

background-clip: content-box;

</h2>

</div>

</body>

</html>

# **CSS checkbox style**

The checkbox is an HTML element used to take input from the user. It is hard to style the checkbox, but pseudo-elements makes it easier to style a checkbox.

This HTML element is generally used on every website, but without styling them, they look similar on every website. So, styling them will make our site different and attractive. We have to hide the original checkbox in order to style the checkbox. Styling the checkboxes using CSS is an interesting and creative task, which will provide a new and attractive look to the default checkbox.

Styling of the checkbox will be clear by using some illustrations. Let's see some of the illustrations for the same.

### **Example**

In this example, we are using the '**~**' which is the sibling combinator. It selects all elements that are preceded by the previous selector. We have also used the pseudo-class **:hover** to style the checkbox when the user moves the cursor over it.

<!DOCTYPE html>

<html>

<style>

.container {

display: block;

position: relative;

padding-left: 35px;

margin-bottom: 20px;

cursor: pointer;

font-size: 25px;

}

/\* Hide the default checkbox \*/

.container input {

visibility: hidden;

cursor: pointer;

}

/\* Create a custom checkbox \*/

.mark {

position: absolute;

top: 0;

left: 0;

height: 25px;

width: 25px;

background-color: lightgray;

}

.container:hover input ~ .mark {

background-color: gray;

}

.container input:checked ~ .mark {

background-color: blue;

}

/\* Create the mark/indicator (hidden when not checked) \*/

.mark:after {

content: "";

position: absolute;

display: none;}

/\* Show the mark when checked \*/

.container input:checked ~ .mark:after {

display: block;

}

/\* Style the mark/indicator \*/

.container .mark:after {

left: 9px;

top: 5px;

width: 5px;

height: 10px;

border: solid white;

border-width: 0 3px 3px 0;

transform: rotate(45deg);

}

</style>

<body>

<h1>Qualification</h1>

<label class="container">MCA

<input type="checkbox">

<span class="mark"></span>

</label>

<label class="container">BCA

<input type="checkbox">

<span class="mark"></span>

</label>

<label class="container">12th

<input type="checkbox">

<span class="mark"></span>

</label>

<label class="container">10th

<input type="checkbox" checked="check">

<span class="mark"></span>

</label>

</body>

</html>

# **CSS letter-spacing**

This CSS property used to control the space between every letter inside an element or the block of text. It sets the spacing behavior between the characters of the text. Using this property, we can increase or decrease the space between the characters of the text.

It modifies the space between the adjacent characters.

### **Syntax**

1. letter-spacing: normal | length | initial | inherit;

### **Possible values**

**normal:** It is the default value that does not provide any space between the characters. It does not change the default spacing between the letters. It is similar to set the value to 0.

**length:** It is used to specify an additional space between the characters. It allows the negative values that tighten the text appearance instead of loosening it. The greater length implies the maximum space between the letters. This value supports the font-relative values (em, rem), absolute values (px).

### **Example**

In this example, we are going to try different values of **letter-spacing** property in order to see the different results. We will also use the possible length values to see the spacing between the characters.

This example will illustrate the setting of space between the characters.

<!DOCTYPE html>

<html>

<head>

<title>CSS letter-spacing Property</title>

</head>

<body style = "text-align: center;">

<h1 style = "color: blue;">

Welcome to the iHub.com

</h1>

<h2>This is an example of CSS letter-spacing Property</h2>

<p style= "letter-spacing: normal;">

This paragraph has letter-spacing: normal;

</p>

<p style= "letter-spacing: 7px;">

This paragraph has letter-spacing: 7px;

</p>

<p style= "letter-spacing: 0.7em;">

This paragraph has letter-spacing: 0.7em;

</p>

<p style= "letter-spacing: -1px;">

This paragraph has letter-spacing: -1px;

</p>

</body>

</html>

A large negative or positive value of **letter-spacing** will make the word unreadable. If we apply a very large positive value, it will cause the appearance of letters like a series of unconnected and individual letters. A very large negative value will overlap the letter to each other, which makes the word unrecognizable.

# **CSS Navigation bar**

A Navigation bar or navigation system comes under GUI that helps the visitors in accessing information. It is the UI element on a webpage that includes links for the other sections of the website.

A navigation bar is mostly displayed on the top of the page in the form of a horizontal list of links. It can be placed below the logo or the header, but it should always be placed before the main content of the webpage.

It is important for a website to have easy-to-use navigation. It plays an important role in the website as it allows the visitors to visit any section quickly.

Let's discuss the horizontal navigational bar and vertical navigational bar in detail.

## Horizontal Navigation Bar

The horizontal navigation bar is the horizontal list of links, which is generally on the top of the page.

Let's see how to create a horizontal navigation bar by using an example.

### **Example**

In this example, we are adding the **overflow: hidden** property that prevents the **li** elements from going outside of the list, **display: block** property displays the links as the block elements and makes the entire link area clickable.

We are also adding the **float: left** property, which uses float for getting the block elements to slide them next to each other.

If we want the full-width background color then we have to add the **background-color** property to **<ul>** rather than the **<a>** element.

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0px;

overflow: hidden;

background-color: lightgray;

}

li {

float: left;

}

li a {

display: block;

color: blue;

font-size:20px;

text-align: center;

padding: 10px 20px;

text-decoration: none;

}

.active{

background-color: gray;

color: white;

}

li a:hover {

background-color: orange;

color: white;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#">Java</a></li>

<li><a href="#">HTML</a></li>

<li><a href="#">CSS</a></li>

</ul>

</body>

</html>

### **Border dividers**

We can add the border between the links in the navigation bar by using the **border-right** property. The following example explains it more clearly.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0px;

overflow: hidden;

background-color: lightgray;

}

li {

float: left;

border-right: 1px solid blue;

}

li a {

display: block;

color: blue;

font-size:20px;

text-align: center;

padding: 10px 20px;

text-decoration: none;

}

.active{

background-color: gray;

color: white;

}

li a:hover {

background-color: orange;

color: white;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#">Java</a></li>

<li><a href="#">HTML</a></li>

<li><a href="#">CSS</a></li>

</ul>

</body>

</html>

### **Fixed Navigation bars**

When we scrolls the page, fixed navigation bars stay at the bottom or top of the page. See an example of the same.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

position: fixed;

width:100%;

top:0;

margin: 0;

padding: 0px;

overflow: hidden;

background-color: lightgray;

}

li {

float: left;

border-right: 1px solid blue;

}

li a {

display: block;

color: blue;

font-size:20px;

text-align: center;

padding: 10px 20px;

text-decoration: none;

}

.active{

background-color: gray;

color: white;

}

li a:hover {

background-color: orange;

color: white;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#">Java</a></li>

<li><a href="#">HTML</a></li>

<li><a href="#">CSS</a></li>

</ul>

<h1 style="padding-top: 100px; text-align: center;">Hello World</h1>

<h2 style="padding-bottom: 2000px; text-align: center;">Scroll down the page to see the fixed navigation bar</h2>

</body>

</html>

### **Sticky Navbar**

The **position: sticky;** property is used to position the element based on the scroll position of the user.

This CSS property allows the elements to stick when the scroll reaches to a certain point. Depending upon the scroll position, a sticky element toggles in between **fixed** and **relative** property**.**

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

position: sticky;

width:100%;

top:0;

margin: 0;

padding: 0px;

overflow: hidden;

background-color: lightgray;

}

li {

float: left;

border-right: 1px solid blue;

}

li a {

display: block;

color: blue;

font-size:20px;

text-align: center;

padding: 10px 20px;

text-decoration: none;

}

.active{

background-color: gray;

color: white;

}

li a:hover {

background-color: orange;

color: white;

}

</style>

</head>

<body>

<h1> Example of sticky navigation bar</h1>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#">Java</a></li>

<li><a href="#">HTML</a></li>

<li><a href="#">CSS</a></li>

</ul>

<h1 style="padding-top: 100px; text-align: center;">Hello World</h1>

<h2 style="padding-bottom: 2000px; text-align: center;">Scroll down the page to see the sticky navigation bar</h2>

</body>

</html>

### **Dropdown Navbar**

Following example explain how to create a dropdown menu inside a navigation bar.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: lightgray;}

li {

float: left;

}

li a, .dropbtn {

display: inline-block;

color: blue;

font-size:20px;

text-align: center;

padding: 10px 20px;

text-decoration: none;

}

.active{

background-color: gray;

color: white;

}

li a:hover , .dropdown:hover .dropbtn{

background-color: orange;

color: white;

}

.dropdown-content {

display: none;

position: absolute;

background-color: lightblue;

min-width: 160px;

box-shadow: 5px 8px 10px 0px black;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

text-align: left;

}

.dropdown-content a:hover {

background-color: gray;

color:white;

}

.dropdown:hover .dropdown-content {

display: block;

}

h1,h2,h3{

text-align:center;

color: green;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#">Java</a></li>

<li><a href="#">C</a></li>

<li><a href="#">C++</a></li>

<li class="dropdown">

<a href="#" class="dropbtn">Web-designing</a>

<div class="dropdown-content">

<a href="#">HTML</a>

<a href="#">CSS</a>

<a href="#">Bootstrap</a>

</div>

</li>

</ul>

<h1>Welcome to the iHub.com</h1>

<h2>Example of Dropdown Menu inside a Navigation Bar</h2>

<h3>Move your cursor on the "web-designing" to see the dropdown effect.</h3>

</body>

</html>

## Vertical Navigation bar

In this example, we are going to see how to build a vertical navigation bar.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

width: 200px;

background-color: lightblue;

}

li a {

display: block;

color: blue;

font-size:20px;

padding: 8px 16px;

text-decoration: none;

}

.active{

background-color: orange;

color: white;

}

li a:hover {

background-color: orange;

color: white;

}

</style>

</head>

<body>

<h2>Vertical Navigation Bar</h2>

<ul>

<li><a href="#" class = "active">Home</a></li>

<li><a href = "#">Java</a></li>

<li><a href = "#">CSS</a></li>

<li><a href = "#">HTML</a></li>

<li><a href = "#">Bootstrap</a></li>

</ul>

</body>

</html>

# **CSS overlay**

Overlay means to cover the surface of something with a coating. In other words, it is used to set one thing on the top of another. The overlay makes a web-page attractive, and it is easy to design.

Creating an overlay effect means to put two **div** together at the same place, but both will appear when required. To make the second div appear, we can hover or click on one div. In these two divs, one div is the overlay div that contains what will show up when the user hovers over the image, and the second div is a container that will hold both the image and its overlay.

## Fade overlay effect

In this overlay effect, when we move the cursor on the image, then the overlay will appear on the top of the image. Let's see how it works.

### **Example**

<!DOCTYPE HTML>

<html>

<head>

<title>Image Overlay</title>

<style>

.container img {

width: 300px;

height: 300px;

}

.container {

position: relative;

width: 25%;

height: auto;

}

.overlay{

position: absolute;

transition: 0.5s ease;

height: 300px;

width: 300px;

top: 0;

left: 20px;

background-color: lightblue;

opacity: 0;

}

.container:hover .overlay {

opacity: 0.9;

}

</style>

</head>

<body>

<center>

<h1>Fade in Overlay</h1>

<h2>Move the cursor over the image to see the effect.</h2>

<div class="container">

<img src= "jtp.png">

<div class="overlay"></div>

</div>

</center>

</body>

</html>

## Image overlay slide

We can create a sliding overlay effect by four different types that are **top, bottom, left, and right.** We are going to discuss it one by one using an example of each.

### **Slide in Overlay from top to bottom**

First, we see how to create the slide in an overlay from the top using an example.

### **Example**

<!DOCTYPE HTML>

<html>

<head>

<style>

.container img {

width: 300px;

height: 300px;

}

.container {

position: relative;

width: 25%;

height: auto;

}

.container:hover .overlay {

opacity: 1;

height: 300px;

}

.overlay{

position: absolute;

transition: 0.7s ease;

opacity: 0;

width: 300px;

height: 0;

top: 0;

right: 20px;

background-color: lightblue;;

}

</style>

</head>

<body>

<center>

<h1>Slide in Overlay from top to bottom</h1>

<h2>Move the cursor over the image to see the effect.</h2>

<div class="container">

<img src= "jtp.png">

<div class="overlay"></div>

</div>

</center>

</body>

</html>

### **Slide in Overlay from bottom to top**

In this overlay effect, when we move the cursor over the image, there will be sliding from bottom to top. It will be clear in the following illustration.

### **Example**

<!DOCTYPE HTML>

<html>

<head>

<style>

.container img {

width: 300px;

height: 300px;

}

.container {

position: relative;

width: 25%;

height: auto;

}

.container:hover .overlay {

opacity: 1;

height: 300px;

}

.overlay{

position: absolute;

transition: 0.7s ease;

opacity: 0;

width: 300px;

height: 0px;

bottom: 0;

right: 20px;

background-color: lightblue;;

}

</style>

</head>

<body>

<center>

<h1>Slide in Overlay from bottom to top</h1>

<h2>Move the cursor over the image to see the effect.</h2>

<div class="container">

<img src= "jtp.png">

<div class="overlay"></div>

</div>

</center>

</body>

</html>

### **Slide in Overlay from left to right**

As its name implies, there is sliding from left to right. See the following example to understand it in detail.

### **Example**

<!DOCTYPE HTML>

<html>

<head>

<style>

.container img {

width: 300px;

height: 300px;

}

.container {

position: relative;

width: 25%;

height: auto;

}

.container:hover .overlay {

opacity: 1;

width: 300px;

}

.overlay{

position: absolute;

transition: 0.7s ease;

opacity: 0;

width: 0;

height: 100%;

bottom: 0;

left: 20px;

background-color: lightblue;;

}

</style>

</head>

<body>

<center>

<h1>Slide in Overlay from left to right</h1>

<h2>Move the cursor over the image to see the effect.</h2>

<div class="container">

<img src= "jtp.png">

<div class="overlay"></div>

</div>

</center>

</body>

</html>

### **Slide in Overlay from right to left**

It is similar to the above sliding effects except that the sliding in it is from right to left.

### **Example**

<!DOCTYPE HTML>

<html>

<head>

<style>

.container img {

width: 300px;

height: 300px;

}

.container {

position: relative;

width: 25%;

height: auto;

}

.container:hover .overlay {

opacity: 1;

width: 300px;

}

.overlay{

position: absolute;

transition: 0.7s ease;

opacity: 0;

width: 0;

height: 100%;

bottom: 0;

right: 20px;

background-color: lightblue;;

}

</style>

</head>

<body>

<center>

<h1>Slide in Overlay from right to left</h1>

<h2>Move the cursor over the image to see the effect.</h2>

<div class="container">

<img src= "jtp.png">

<div class="overlay"></div>

</div>

</center>

</body>

</html>

## Image Overlay title

In the image overlay effect, when we move the cursor over an image, we will see the title on the image. See the below illustration for the same.

### **Example**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body{

text-align: center;

}

\* {box-sizing: border-box;}

.container {

position: relative;

width: 50%;

max-width: 300px;

}

img {

display: block;

width: 100%;

height: auto;

}

.overlay {

position: absolute;

bottom: 0;

background: rgba(0, 0, 0, 0.2);

width: 100%;

opacity:0;

transition: .9s ease;

font-size: 25px;

padding: 20px;

}

.container:hover .overlay {

opacity: 1.5;

}

</style>

</head>

<body>

<h1>Image Overlay Title Effect</h1>

<h2>Move your mouse over the image to see the effect.</h2>

<center>

<div class="container">

<img src="jtp.png">

<div class="overlay">Welcome to iHub.com</div>

</div> </center>

</body>

</html>

## Image Overlay icon

In this overlay effect, on mouse hovering, there will be an icon that covers the entire image. We can set the link on that icon to switch between the pages. It can be clear from the following example.

### **Example**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>

.container {

position: relative;

width: 100%;

max-width: 400px;

}

.image {

display: block;

width: 100%;

height: auto;

}

.overlay {

position: absolute;

top: 0;

height: 100%;

width: 100%;

opacity: 0;

transition: 1s ease;

background-color: lightblue;

}

.container:hover .overlay {

opacity: 1;

}

.icon {

color: blue;

font-size: 100px;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<center>

<h1>Image Overlay icon Effect</h1>

<h2>Move your mouse over the image to see the effect.</h2>

<div class="container">

<img src="jtp.png" class="image">

<div class="overlay">

<a href="#" class="icon">

<i class="fa fa-bars"></i>

</a>

</div>

</div>

</center>

</body>

</html>

# **CSS :root selector**

This pseudo-class in CSS matches the root element of the document. It selects the highest-level parent in the document tree or DOM.

In HTML, the **<html>** element is always the root element. Although the **:root** is equivalent to **html** selector because both of them target the same element, but the **:root** selector has a higher specificity.

### **Syntax**

:root {

    // CSS property

}

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>:root selector</title>

<style>

:root {

background: lightblue;

color: blue;

text-align: center;

}

</style>

</head>

<body>

<h1>Welcome to the iHub.com</h1>

<h2>This is an example of :root selector</h2>

</body>

</html>

Now, let's try **html** selector and **:root** selector simultaneously. Although they both work similar but in the matter of specificity, the **:root** selector wins.

### **Example**

In this example, we are going to define the same properties in **html** selector as well as in **:root** selector. The properties in **:root** selector will work because of higher specificity. But those properties that are not in **:root** selector but mentioned in **html** selector, then the properties of **html** selector will work.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>:root selector</title>

<style>

:root {

background-color: lightblue;

color: blue;

text-align: center;

}

html{

background-color: red;

color: white;

font-size: 30px;

}

</style>

</head>

<body>

<h1>Welcome to the iHub.com</h1>

<h2>This is an example of :root selector and html selector</h2>

</body>

</html>

In the above example, we can see that the **background-color** and **color** properties are both mentioned in **html** as well as in **:root** selector, but in the output, the properties mentioned in **:root** selector will work. This is because of the higher specificity of the **:root** selector.

# **CSS Specificity**

When more than one conflicting rules of CSS indicates the same element, then the browser will follow some rules for determining the particular one. Specificity is the way which helps the browsers to decide which property value is most relevant for the element. It determines which style declaration is applied to an element.

Before going in deep about specificity, let's discuss some points about it:

* The CSS specificity is important only when various selectors are affecting the same element. In this case, the browser needs a way to identify the style to be applied to the matching element, and CSS specificity is the way of doing it.
* When two or more selectors have equal specificity value, then the latest one considers.
* Universal selectors (\*) and the inherited values have lower specificity, i.e., 0 specificity.
* The **style** property has a greater specificity value compare to the selectors (except the **!important** in the stylesheet selector).
* The **!important** alter the selector specificity. When two selectors have equal specificity, then the selector having **!important**

## Specificity hierarchy

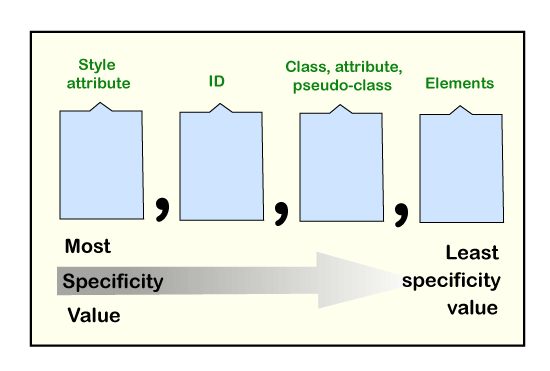
Each selector has a place in the hierarchy. There are mainly four categories that define the selector's specificity level:

**Inline styles:** It is directly attached to the element which is to be styled. For example: <p style="color: red;">. It has the highest priority.

**IDs:** It is a unique identifier for the elements of a page that has the second-highest priority. For example: #para.

**Classes, attributes, and pseudo-classes:** It includes classes, attributes, and pseudo-classes (like :focus, :hover, etc.).

**Elements and pseudo-elements:** It includes the name of elements (div, h1) and pseudo-elements (like :after and :before). They have the lowest priority.



## Specificity rules

Specificity is the weight, which is applied to the CSS declaration. It is determined by the number of every selector type in the matching selector. Let's see the calculation of specificity.

The specificity rules are discussed below, along with an example.

### **The specificity of ID selectors is higher than attribute selectors**

Let us try to understand it with an example.

### **Example**

In this example, we are going to use the id selector with the background-color property.

<!DOCTYPE html>

<html>

<head>

<style>

body{

text-align: center;

font-size: 30px;

color: blue;

background-color: red;

}

#div1 {

background-color: red;

}

div#div1 /\*Higher specificity\*/

{

background-color: yellow;

}

div[id=div1] {

background-color: blue;

}

</style>

</head>

<body>

<div id="div1"> Welcome to the iHub.com </div>

</body></html>

### **In equal specificity, the latest rule will count**

In the external stylesheet, if the same rule is applied twice, then the latest (or last) rule will be applied.

### **Example**

In this example, the specificity of the name of elements is same. In this case, the latest specified element name will be considered.

<!DOCTYPE html>

<html>

<head>

<style>

body{

font-size: 30px;

text-align: center;

}

div

{

background-color: yellow;

color: red;

}

div

{

background-color: red;

color: yellow;

}

</style>

</head>

<body>

<h2> Example of equal specificity </h2>

<div> Welcome to the iHub.com </div>

</body>

</html>

### **The specificity of class selector is greater than the element selectors**

A class selector (.nav, .high, etc.) is highly specific than element selectors (like div, h1, p, etc.)

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.intro {

background-color: blue;

text-align: center;

color: yellow;

font-size :40px;

}

div {

background-color: red;

text-align: right;

color: blue;

}

</style>

</head>

<body>

<h1>Hello World</h1>

<div class="intro">Welcome to the iHub.com</div>

</body>

</html>

# **CSS text-indent**

This CSS property sets the indentation of the first line in a block of text. It specifies the amount of horizontal space that puts before the lines of text.

It allows the negative values, and if any negative value is defined, then the indentation of the first line will be towards left.

### **Syntax**

1. text-indent: length | inherit | initial;

This property has the value **length,** but here, we will discuss its experimental values.

### **Values**

**length:** This value sets the fix indentation with the units cm, pt, em, px, and others. Its default value is 0. It allows negative values. The indentation of the first line is on the left when its value is negative.

**percentage:** It specifies the amount in space in the percentage of the width of the containing block.

**initial:** It sets the property to its default value.

This CSS property has two experimental values, which are discussed as follows. These two following values are not supported in browsers.

**hanging:** It is unofficial and experimental value. It inverts the indented lines. It indents each line except the first. Usually, bibliographies are written with the hanging indents, in which the first line is with the left-hand margin, and the rest of the content is indented.

**each-line:** It is also an experimental value. It affects every line, including the first line after a forced line break (using **<br>**).

### **Example**

In this example, we are using the text-indent property with the length values in **px, em,** and **cm.** We are also applying the **text-align: justify;** property in order to see the better results.

<!DOCTYPE html>

<html>

<head>

<title>

CSS text-indent Property

</title>

<style>

div{

font-size: 20px;

width: 500px;

height:200px;

text-align: justify;

}

.jtppx {

text-indent: 100px;

}

.jtpem {

text-indent: -5em;

}

.jtpcm {

text-indent: 7cm;

}

</style>

</head>

<body>

<center>

<h1>Example of text-indent Property</h1>

<h2>text-indent: 70px;</h2>

<div class = "jtppx">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</div>

<h2>text-indent: -5em;</h2>

<div class = "jtpem">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</div>

<h2>text-indent: 7cm;</h2>

<div class = "jtpcm">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</div>

</center>

</body>

</html>

Let's see another demonstration using the percentage values.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS text-indent Property

</title>

<style>

div{

font-size: 20px;

width: 500px;

height:200px;

text-align: justify;

}

.jtpper {

text-indent: 65%;

}

</style>

</head>

<body>

<center>

<h1>Example of text-indent Property</h1>

<h2>text-indent: 65%;</h2>

<div class = "jtpper">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</div>

</center>

</body>

</html>

# **CSS text-stroke**

This CSS property adds a stroke to the text and also provides decoration options for them. It defines the color and width of strokes for text characters.

This CSS property is the shorthand of the following two properties:

**text-stroke-width:** It describes the thickness of the stroke effect and takes the unit value.

**text-stroke-color:** It takes the value of a color.

The text-stroke can only be used with the **-webkit-** prefix.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS text-stroke property

</title>

<style>

body{

text-align: center;

}

.jtp {

color: white;

font-size: 50px;

-webkit-text-stroke-width: 2px;

-webkit-text-stroke-color: red;

}

</style>

</head>

<body>

<h1 class= "jtp">Welcome to the iHubTalent.com</h1>

<h2 class= "jtp" style= "-webkit-text-stroke-color: blue;">This is an example of CSS text-stroke property</h2>

</body>

</html>

# **CSS Zoom**

This CSS property scales the content. It manages the magnification level of the content. Instead of using this property, we can also use the **transform: scale();** property.

The use of this CSS property is not recommended because it is not supported in some browsers. It is a non-standard feature, which is recommended not to use on production sites. It was initially implemented in the IE browser.

### **Syntax**

1. zoom: normal | number | percentage | reset ;

This property supports the values mentioned above are discussed as follows.

**normal:** It shows the element to its normal size. It defines the normal size of the element. Tt holds the normal content that does not zoom-out or zoom-in. It has the value **zoom: 1;**

**number:** It is a positive float value that indicates a zoom factor. Its value smaller than 1.0 represents zoom out (or size reduction), and larger than 1.0 represents zoom in (increase the size). Suppose we specify **zoom: 1.5;** then, the content will be 1.5 times larger than the original content.

**percentage:** It defines a value in percentage. Its **100%** value is equivalent to normal. It scales the content using the percentage value. Its value greater than 100% represents zoom in, and value less than 100% represents zoom out. Suppose if we define **zoom: 175%;** then it means that the content will be 175% larger than the original content.

We can understand this property by using some illustrations, which are given below.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS zoom

</title>

<style>

h1 {

color: red;

}

h2{

color: blue;

}

.magnify1{

zoom: 0.75;

}

.magnify{

zoom: 1.5;

}

</style>

</head>

<body>

<center>

<h1>CSS zoom property</h1>

<div>

<h2>original image</h2>

<img class="original" src= "jtp.png">

<h2>Magnified image zoom: 0.75;</h2>

<img class="magnify1" src= "jtp.png">

<h2>Magnified image zoom: 1.5;</h2>

<img class="magnify" src= "jtp.png">

</div>

</center>

<body>

</html>

# **CSS order**

This CSS property specifies the order of the flex item in the grid container or flex. It is basically used for ordering the flex items. It is to note that, if the element isn't flexible, then this property will not work.

The elements will get displayed in the increasing order of their order values. If two elements use the same order value, then they will display based on their occurrence defined in the source code.

The **order** property modifies the visual order of the flex items. The item with the lowest order value comes first, and a higher value will be placed afterward. It only affects the visual order of elements rather than the tab or logical order. It must not be used on non-visual media such as speech.

It is allowed to define the negative values of **order**. Negative values are useful when we want to display one item first and leave the order of other items unchanged. When there is no value is specified, then the value 0 will be used, which is its default value. So, when we want to display an item first, we can provide it a negative value such as **-1.** As this negative value is smaller than 0, then the corresponding item will always be displayed first.

### **Syntax**

1. order: integer | initial | inherit;

### **Values**

The order property uses the integer values for defining the order of the items.

**integer:** It is used to specify the order of the flexible item. Its default value is 0.

**initial:** It sets the property value to its default value.

**inherit:** As its name implies, the element uses the computed value of its parent element.

Let's understand the **order** property using some illustrations.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

body{

text-align: center;

}

.container {

display: flex;

background-color: blue;

height: 150px;

width: auto;

flex-wrap: wrap;

}

div {

background-color: cyan;

line-height: 40px;

color: black;

padding: 10px;

text-align: center;

font-size: 35px;

width: 100px;

margin: 20px;

}

</style>

</head>

<body>

<h1> CSS order Property </h1>

<div class = "container">

<div style = "order: 3"> 1 </div>

<div style = "order: 0"> 2 </div>

<div style = "order: 0"> 3 </div>

<div style = "order: 1"> 4 </div>

<div style = "order: -1"> 5 </div>

<div style = "order: 2"> 6 </div>

<div style = "order: 1"> 7 </div>

<div style = "order: -2"> 8 </div>

</div>

</body>

</html>

In the above example, we have used the negative values as well as the same order values of some elements. The elements having small value will display first and the same order values will display on the basis of their occurrence in code.

As in the above example one div element has the order value -2 then, it will display first and after that the element with order value -1 displayed and so on.

# **CSS Descendant Selector**

The CSS descendant selector is used to match the descendant elements of a particular element. The word Descendant indicates nested anywhere in the DOM tree. It can be a direct child or deeper than five levels, but it will still be referred to as a descendant.

The **Descendant combinator** is represented using a single space. It combines two selectors in which the first selector represents an ancestor (parent, parent's parent, etc.), and the second selector represents descendants. The elements matched by the second selector are selected if they have an ancestor element that matches the first selector. Descendant selectors use the descendant combinators.

### **Syntax**

selector1 selector2 {

  /\* property declarations \*/

}

We can understand CSS descendant selector using the following example. Let us see the implementation of CSS descendant selector.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div p {

background-color: lightblue;

font-weight: bold;

}

</style>

</head>

<body>

<div>

<p> This is 1st paragraph in the div. </p>

<p> This is 2nd paragraph in the div. </p>

<div>

This is second div in the first div

<p> This is the paragraph in second div. It will also be affected. </p>

</div></div>

<p> Paragraph 4. It will not be affected because it is not in the div. </p>

</body>

</html>

# **CSS calc()**

It is an inbuilt CSS function which allows us to perform calculations. It can be used to calculate length, percentage, time, number, integer frequency, or angle. It uses the four simple arithmetic operators add (+), multiply (\*), subtract (-), and divide (/).

It is a powerful CSS concept because it allows us to mix any units, such as percentage and pixel.

### **Syntax**

calc( Expression );

### **Values**

This CSS function takes a single parameter expression, and the result of this mathematical expression will be used as a value. It can be any simple expression using the four arithmetic operators (+, -, \*, /). The expression is mandatory to define.

### **Important points**

* The arithmetic operators add (+) and subtract (-) should always be surrounded by whitespace. Otherwise, the expression will be treated as an invalid expression. As an example, **calc(60%-4px)** will be invalid because it is parsed as a percentage, followed by a negative length. On the other hand, the expression **calc(60% - 4px)** will be parsed as a subtraction operator and a length.
* Although the operators **\*** and **/** does not requires whitespace, but it is recommended to add it for consistency.
* Nesting of **calc()** function can be possible.

### **Simple Example**

In this example, we are using the **calc()** function to define the **width** and **height** of the **div** element. There is the subtraction in the expression of **calc()** function with same units.

The result of the expression will be treated as the value of the property, so, the value of width will be 75% and the value of height will be 275px.

<!DOCTYPE html>

<html>

<head>

<title> calc() function </title>

<style>

.jtp {

width: calc(150% - 75%);

height: calc(350px - 75px);

background-color: lightblue;

padding-top: 50px;

}

.jtp1 {

font-size: 30px;

font-weight: bold;

color: blue;

}

h1 {

color: blue;

}

h2{

color: green;

}

</style>

</head>

<body>

<center>

<div class = "jtp">

<div class = "jtp1"> Welcome to the iHubTalent.com </div>

<h1> This is an example of calc() Function </h1>

<h2> width: calc(150% - 75%); </h2>

<h2> height: calc(350px - 75px); </h2>

</div>

</center>

</body>

</html>

In the above example, we can directly mention the values of height and width. Although the expression in the above example has the same units, what happens when the units are different, then it will be hard to write the values directly.

# **CSS Clip**

This CSS property specifies the visible area of an element. It applies to absolutely positioned elements **(position: absolute;)**. It is generally used when the image is larger than its containing element.

It allows us to define a rectangle, which is specified as four coordinates for clipping an absolutely positioned element.

### **Syntax**

1. clip: auto | shape | initial | inherit;

### **Possible values**

**auto:** It is the default value that shows the element as it is. There will be no clipping.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.auto {

position: absolute;

width: 400px;

height: 400px;

clip: auto;

}

</style>

</head>

<body>

<h2>clip: auto; property</h2>

<img src= "jtp.png" class="auto">

</body>

</html>

**shape:** It is used to clip an element. It clips the defined area of the element. Its valid value is **rect(top, right, bottom, left).**

### **Example**

<!DOCTYPE html>

<html>

<head>

<style type = "text/css">

div {

background: url(jtp.png);

clip: rect(0px, 150px, 250px, 0px);

border:3px solid red;

height:200px;

width: 250px;

position: absolute;

}

</style>

</head>

<body>

<div></div>

</body>

</html>

# **CSS clip-path**

This CSS property is used to create a clipping region and specifies the element's area that should be visible. The area inside the region will be visible, while the outside area is hidden. Anything outside the clipping will be clipped by browsers, including borders, text-shadows, and many more.

It allows us to define a particular region of the element to display, instead of displaying the entire area. It makes it easier to clip the basic shapes by using ellipse, circle, polygon, or inset keywords.

### **Syntax**

1. clip-path: **<clip-source>** | [ **<basic-shape>** || **<geometry-box>** || none

The CSS clip-path property has four values:

* clip-source
* basic-shape
* geometry-box
* none

Let's discuss the above property values.

**clip-source:** It is a url that reference to an SVG **<clippath>** element.

**basic-shape:** It clips the element to a basic shape. It has four basic shapes: circle, ellipse, polygon and inset.

It is a shape in which the value of **<geometry-box>** defines position and size. If there is no geometry-box is defined, then the border-box will be used as a reference box.

**geometry-box:** The **<geometry-box>** defines the reference box for the basic shape. If it is defined with the combination of the **<basic-shape>**, then it will act as the reference box for the clipping done by the **<basic-shape>**.

It can have the below values:

**margin-box:** It can be used as the reference box. It can be defined as the shape specified by the outside margin edge and includes the corner radius of the shape.

**border-box:** It can be used as the reference box. It is a value that is defined by the outside border edge.

**padding-box:** It can be used as the reference box.. It specifies the shape which is enclosed by the outside padding edge.

**content-box:** It can be used as the reference box.

**fill-box:** The object bounding box can be used as the reference box.

**stroke-box:** The stroke bounding box can be used as the reference box.

**view-box:** It uses the closest SVG viewport as the reference box.

## Defining basic shapes with clip-path

As discussed above, there are four basic shapes. Let's discuss them with an example of each.

## polygon

It is one of the shapes of the available basic shapes. It allows us to define any amount of points. The given points are in the pair of **X** and **Y** coordinates of any unit (such as percent based or pixel-based).

We can understand this basic shape by using the below example. In the following example, we have defined two polygon shapes: diamond shape polygon and star shape polygon.

### **Example**

<!DOCTYPE html>

<html>

<head>

</head>

<style>

.example {

clip-path: polygon(50% 0%, 100% 50%, 50% 100%, 0% 50%);

}

.example1{

clip-path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%, 50% 70%, 21% 91%, 32% 57%, 2% 35%, 39% 35%);

}

</style>

<body>

<center>

<h3> Clip-path property example </h3>

<img src="jtp.png" class="example">

<h4>Diamond shape polygon</h4>

<p>clip-path: polygon(50% 0%, 100% 50%, 50% 100%, 0% 50%);</p>

<img src="jtp.png" class="example1">

<h4>Star shape polygon</h4>

<p>clip-path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%, 50% 70%, 21% 91%, 32% 57%, 2% 35%, 39% 35%);</p>

</center> </body></html>

## circle

The default syntax of defining the circle is the **circle(radius at posX posY)**. The position is optional, and its default value is **50% 50%**.

## ellipse

The syntax to define ellipse is: **ellipse(radiusX radiusY at posX posY)**. Like the circle, the position in it is optional and default to 50% 50%.

## inset

Using inset, we can define an inner rectangle, and anything outside will be discarded. It makes the cropping of an image or an element easier.

### **Example**

<!DOCTYPE html>

<html>

<head>

</head>

<style>

.example

{

clip-path: inset(20% 25% 20% 10%);

}

.example1

{

clip-path: inset(45% 0% 33% 10% round 10px);

}

</style>

<body>

<center>

<h2> Clip-path property example </h2>

<img src="jtp.png" class="example">

<h3>clip-path: inset(20% 25% 20% 10%);</h3>

<img src="jtp.png" class="example1">

<h3>clip-path: inset(45% 0% 33% 10% round 10px);</h3>

</center>

</body>

</html>

## Adding animation

We can also apply animation to this property. Animation and transitions will create interesting effects with this CSS property.

Let's see the illustration for the same.

### **Example**

<!DOCTYPE html>

<html>

<head>

</head>

<style>

img.example {

animation: anime 4s infinite;

border: 5px dashed red;

}

@keyframes anime {

0% {

clip-path: polygon(0 0, 100% 0, 100% 80%, 0% 70%);

}

20% {

clip-path: polygon(40% 0, 50% 0, 100% 100%, 0% 80%);

}

40% {

clip-path: polygon(0 0, 60% 72%, 100% 100%, 0 35%);

}

60% {

clip-path: polygon(70% 0, 20% 0, 100% 100%, 0% 80%);

}

80% {

clip-path: polygon(0 70%, 60% 0, 100% 32%, 0 40%);

}

100% {

clip-path: polygon(0 0, 60% 0, 100% 100%, 0% 30%);

}

}

</style>

<body>

<center>

<h2> Animation in Clip-path property </h2>

<img src="jtp.png" class="example">

</center>

</body>

</html>

# **CSS background-blend-mode**

This CSS property sets the blend mode for each background layer (image/color) of an element. It defines how the background image of an element blends with the background color of the element. We can blend the background images together or can blend them with background-color.

This property is not supported in Edge/Internet Explorer.

### **Syntax**

1. background-blend-mode: normal | multiply | screen | color-dodge | difference | darken | lighten | saturation | luminosity | overlay | hard-light | soft-light | exclusion | hue | color-burn | color;

This property has numerous property values. Let's discuss the above blend modes with an example of each.

## normal

It is the default value which sets the property mode to normal.

## multiply

It multiplies the background-color and background-images and leads to a darker image than before. It is used to set the blending mode to multiply.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

#div1 img{

width: 300px;

height: 300px;

}

#example{

width: 400px;

height: 400px;

background-repeat: no-repeat;

background-image: url("lion.png"), url("forest.jpg");

background-blend-mode: multiply;

}

</style>

</head>

<body>

<center>

<div id = "div1">

<h2> Original Images </h2>

<img src = "lion.png">

<img src = "forest.jpg">

</div>

<h2> background-blend-mode: multiply; </h2>

<div id="example"></div>

</center>

</body>

</html>

# **CSS Radio Button**

The radio button is an HTML element that helps to take input from the user. Although it is hard to style the radio button, pseudo-elements makes it easier to style the radio button. Radio buttons are applied when there is the requirement of a single selection from a group of items.

This HTML element is generally used on every website, but without styling them, they look similar on every website. So, styling them will make our site different and attractive. Designing the radio button using [CSS](https://www.javatpoint.com/css-tutorial) is an interesting and creative task, which will provide a new look to the default radio button.

To create the custom radio buttons, we require to write an HTML markup, and to style, we have to write the CSS.

Styling of the radio button will be clear by using some illustrations. Let's see some of the illustrations for the same.

### **Example**

In this example, we are using the '**~**' which is the sibling combinator. It selects all elements that are preceded by the previous selector. We have also used the pseudo-class **:hover** to style the radio button when the user moves the cursor over it.

<!DOCTYPE html>

<html>

<style>

.container {

display: block;

position: relative;

padding-left: 40px;

margin-bottom: 20px;

cursor: pointer;

font-size: 25px;

}

/\* Hide the default radio button \*/

.container input {

position: absolute;

opacity: 0;

cursor: pointer;

}

/\* custom radio button \*/

.check {

position: absolute;

top: 0;

left: 0;

height: 30px;

width: 30px;

background-color: lightgray;

border-radius: 50%;

}

.container:hover input ~ .check {

background-color: gray;

}

.container input:checked ~ .check {

background-color: blue;

}

.check:after {

content: "";

position: absolute;

display: none;

}

.container input:checked ~ .check:after {

display: block;

}

.container .check:after {

top: 8px;

left: 8px;

width: 15px;

height: 15px;

border-radius: 50%;

background: white;

}

</style>

<body>

<h1> Example of Custom Radio Buttons</h1>

<h2> Select your category </h2>

<label class="container">GEN

<input type="radio" name="radio" checked>

<span class="check"></span>

</label>

<label class="container">OBC

<input type="radio" name="radio">

<span class="check"></span>

</label>

<label class="container">SC

<input type="radio" name="radio">

<span class="check"></span>

</label>

<label class="container">ST

<input type="radio" name="radio">

<span class="check"></span>

</label>

</body>

</html>

# **CSS Superscript and Subscript**

In HTML, there is the use of **<sub>** and **<sup>** tags to define the subscript and superscript text. The superscript text appears in a smaller font and the half character above the normal line. It is generally used to write mathematical equations (like **x2 + y2 = r2**), footnotes, and many more.

Unlike superscript, the subscript text appears in a smaller font and the half character below the normal line. It is generally used to write chemical equations or chemical formulas such as **H2O, H2SO4,** etc.

In CSS, the **vertical-align** property is used to achieve the same thing. We can specify the superscript and subscript of text using [CSS](https://www.javatpoint.com/css-tutorial) as well. This CSS property specifies the vertical alignment of the text.

Now, let's see how to achieve the superscript and subscript using the **vertical-align** property.

### **Syntax**

1. vertical-align: baseline | super | sub ;

### **Property values**

**baseline:** It is the default value that aligns the text to the baseline of the parent element.

**super:** It is the superscript that raises the text.

**sub:** It is the subscript that lowers the text.

When the values **super** and **sub** of this property are applied, then the text becomes superscript or subscript.

### **Example- Superscript**

<!DOCTYPE html>

<html>

<head>

<style>

#super{

vertical-align:super;

font-size: medium;

}

p{

font-weight: bold;

font-size: 20px;

}

</style>

</head>

<body>

<h1> Using vertical-align: super; </h1>

<p> Exponen ts (powers of a number), mathematical equations or formulae are the common uses of superscripted text. </p>

<h3>x<span id="super">2</span>+ y<span id="super">2</span> = r<span id="super">2</span></h3>

<h3> (a + b)<span id="super">2</span> = a<span id="super">2</span> + b<span id="super">2</span> + 2ab </h3>

<h3>5<span id="super">th</span></h3>

</body>

</html>

### **Example- Subscript**

<!DOCTYPE html>

<html>

<head>

<style>

#sub{

vertical-align: sub;

font-size: medium;

}

p{

font-size: 20px;

}

</style>

</head>

<body>

<h1> Using vertical-align: sub; </h1>

<p> Its common examples are chemical equations. </p>

<h3> Chemical formula of Water is: H<span id="sub">2</span>O</h3>

<h3> Ch emical formula of Sulphuric acid is: H<span id="sub">2</span>SO<span id="sub">4</span></h3>

</body>

</html>

# **CSS Text Effects**

We can apply different effects on the text used within an HTML document. Some properties can be used for adding the effects on text.

Using CSS, we can style the web documents and affects the text. The properties of the text effect help us to make the text attractive and clear. There are some text effect properties in CSS that are listed below:

* word-break
* text-overflow
* word-wrap
* writing-mode

Let's discuss the above CSS properties along with illustrations.

## word-break

It specifies how words should break at the end of the line. It defines the line break rules.

### **Syntax**

1. word-break: normal |keep-all |  break-all | inherit ;

The default value of this property is normal. So, this value is automatically used when we don't specify any value.

### **Values**

**keep-all:** It breaks the word in the default style.

**break-all:** It inserts the word break between the characters in order to prevent the word overflow.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>word-break: break-all</title>

<style>

.jtp{

width: 150px;

border: 2px solid black;

word-break: break-all;

text-align: left;

font-size: 25px;

color: blue;

}

.jtp1{

width: 156px;

border: 2px solid black;

word-break: keep-all;

text-align: left;

font-size: 25px;

color: blue;

}

</style>

</head>

<center>

<body>

<h2>word-break: break-all;</h2>

<p class="jtp">

Welcome to the iHubTalent.com

</p>

<h2>word-break: keep-all;</h2>

<p class="jtp1">

Welcome to the iHubTalent.com

</p>

</center>

</body>

</html>

## word-wrap

CSS word-wrap property is used to break the long words and wrap onto the next line. This property is used to prevent overflow when an unbreakable string is too long to fit in the containing box.

### **Syntax**

1. word-wrap: normal| break-word| inherit ;

### **Values**

**normal:** This property is used to break words only at allowed break points.

**break-word:** It is used to break unbreakable words.

**initial:** It is used to set this property to its default value.

**inherit:** It inherits this property from its parent element.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.test {

width: 200px;

background-color: lightblue;

border: 2px solid black;

padding:10px;

font-size: 20px;}

.test1 {

width: 11em;

background-color: lightblue;

border: 2px solid black;

padding:10px;

word-wrap: break-word;

font-size: 20px;

}

</style>

</head>

<body>

<center>

<h1> Without Using word-wrap </h1>

<p class="test"> In this paragraph, there is a very long word:

iamsooooooooooooooooooooooooooooooolongggggggggggggggg. </p>

<h1> Using word-wrap: break-word; </h1>

<p class="test1"> In this paragraph, there is a very long word:

iamsooooooooooooooooooooooooooooooolongggggggggggggggg. The long word will break and wrap to the next line. </p>

</center>

</body> </html>

## text-overflow

It specifies the representation of overflowed text, which is not visible to the user. It signals the user about the content that is not visible. This property helps us to decide whether the text should be clipped or show some dots (ellipsis).

This property does not work on its own. We have to use **white-space: nowrap;** and **overflow: hidden;** with this property.

### **Syntax**

1. text-overflow: clip | ellipsis;

### **Property Values**

**clip:** It is the default value that clips the overflowed text.

**ellipsis:** This value displays an ellipsis (…) or three dots to show the clipped text. It is displayed within the area, decreasing the amount of text.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.jtp{

white-space: nowrap;

height: 30px;

width: 250px;

overflow: hidden;

border: 2px solid black;

font-size: 25px;

text-overflow: clip;

}

.jtp1 {

white-space: nowrap;

height: 30px;

width: 250px;

overflow: hidden;

border: 2px solid black;

font-size: 25px;

text-overflow: ellipsis;

}

h2{

color: blue;

}

div:hover {

overflow: visible;

}

p{

font-size: 25px;

font-weight: bold;

color: red;

}

</style>

</head>

<center>

<body>

<p> Hover over the bordered text to see the full content. </p>

<h2>

text-overflow: clip;

</h2>

<div class="jtp">

Welcome to the iHubTalent.com

</div>

<h2>

text-overflow: ellipsis;

</h2>

<div class="jtp1">

Welcome to the iHubTalent.com

</div>

</center>

</body>

</html>

## writing-mode

It specifies whether the text will be written in the horizontal or vertical direction. If the writing direction is vertical, then it can be from **left to right (vertical-lr)** or from **right to left (vertical-rl).**

### **Syntax**

1. writing-mode: horizontal-tb | vertical-lr | vertical-rl | inherit ;

### **Property values**

**horizontal-tb:** It is the default value of this property in which the text is in the horizontal direction and read from left to right and top to bottom.

**vertical-rl:** It displays the text in the vertical direction, and the text is read from right to left and top to bottom.

**vertical-lr:** It is similar to vertical-rl, but the text is read from left to right.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

h2 {

border: 2px solid black;

width: 300px;

height: 100px;

}

#tb {

writing-mode: horizontal-tb; }

#lr {

writing-mode: vertical-lr;

}

#rl {

writing-mode: vertical-rl;

}

</style>

</head>

<center>

<body>

<h1> Example of CSS writing-mode property </h1>

<h2 id="tb"> writing-mode: horizontal-tb; </h2>

<h2 id="lr" style= "height: 300px;"> writing-mode: vertical-lr; </h2><br>

<h2 id="rl" style= "height: 300px;"> writing-mode: vertical-rl; </h2>

</center>

</body>

</html>

# **CSS text-align**

This CSS property is used to set the horizontal alignment of a table-cell box or the block element. It is similar to the **vertical-align** property but in the horizontal direction.

### **Syntax**

1. text-align: justify | center | right | left | initial | inherit;

### **Possible values**

**justify:** It is generally used in newspapers and magazines. It stretches the element's content in order to display the equal width of every line.

**center:** It centers the inline text.

**right:** It is used to align the text to the right.

**left:** It is used to align the text to the left.

Let's see an example that will demonstrate the **text-align** property.

### **Example**

<html>

<head>

</head>

<style>

h2{

color: blue;

}

</style>

<body>

<h1>Example of text-align proeprty</h1>

<h2 style = "text-align: center;">

text-align: center;

</h2>

<h2 style = "text-align: right;">

text-align: right;

</h2>

<h2 style = "text-align: left;">

text-align: left;

</h2>

<h2 style = "text-align: justify;">

text-align: justify; To see its effect, it should be applied on large paragraph.

</h2>

</body>

</html>

# **CSS Variables**

The CSS variables are used to add the values of custom property to our web page. The **custom properties** are sometimes referred to as **cascading variables** or **CSS variables**. The authors define these entities that contain specific values and can be reused throughout the document. These entities are set using the custom property notation and can be accessed by using the **var()** function.

The variables store the values and have a scope in which they can be used.

CSS variables are advantageous because they allow us to reuse the same value at multiple places. The name of the variable is easy to understand and use, as compared to the color values.

element {

  --main-color: brown;

}

A variable in CSS is defined by using the two dashes (--) at the beginning, followed by the name, which is case-sensitive.

In the above syntax, the **element** indicates the selector that specifies the scope of the custom property. If we define the custom properties on the **:root** pseudo-class, then it will be globally applied to our HTML document. The names of the custom properties are case-sensitive, i.e., **--main-color** and **--Main-color** will be treated as the separate custom properties.

## The var() function

The **var()** function in CSS is used to insert the custom property value. The name of the variable can be passed as the argument to the **var()** function.

**Syntax**

1. var( --custom-name, value )

**Parameters**

The **var()** function only allows two arguments that are defined as follows:

**--custom-name:** This parameter accepts the name of the custom property. It must begin with the two dashes **(--).** It is the required parameter.

**value:** It is an optional parameter and accepts the fallback value. It is used as the substitution when the custom property is invalid.

Fallback values are not used to fix the compatibility of the browser. The fallback value will not be useful when any browser does not support the custom properties. The fallback value works as the substitution for the browser that supports the [CSS](https://www.javatpoint.com/css-tutorial) custom properties for choosing a different value if the variable has an invalid value or if the variable is not defined.

There are some of the valid and invalid ways to define the fallback value that are given as follows:

element {

  color: var(--main-color, orange); /\* orange if --main-color is not defined \*/

}

element {

  background-color: var(--main-color, var(--main-background, blue)); /\* blue if --main-color and --main-background are not defined \*/

}

element {

  background-color: var(--main-color, --main-background, gray); /\* Invalid\*/

}

Now, let's understand the CSS variables by using some illustrations.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Variables</title>

<style>

:root {

--bg-color: lightgreen;

--text-color: red;

}

/\* var() function used here \*/

body {

background-color: var(--bg-color);

text-align: center; }

h1 {

color: var(--text-color);

}

div {

color: var(--text-color);

font-size: 30px;

}

</style>

</head> <body>

<h1>Welcome to the iHubTalent.com</h1>

<div>

This is an example of CSS variables

<h3>--bg-color: lightgreen;</h3>

<h3>--text-color: red;</h3>

</div> </body> </html>

## Use of calc() with the var()

We can use the **calc()** on the variable values. Let's see an example in which we are using the calc() function with the var() function.

In this example, we are using the **calc()** function with **var()** function to adjust the padding and font-size of the elements.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>CSS Variables</title>

<style>

:root {

--bg-color: lightgreen;

--extra-padding: 1.2em;

--txt-size: 90px;

}

body {

background-color: var(--bg-color);

text-align: center;

}

h1 {

color: var(--text-color, blue); /\* --text-color is not set, so the fallback value 'blue' will be used \*/

font-size: calc(var(--txt-size) - 20px);

}

div {

color: var(--text-color, blue);

font-size: 30px;

border: 8px ridge red;

padding: calc(var(--extra-padding) + 20px);

}

</style>

</head>

<body>

<h1>Welcome to the iHubTalent.com</h1>

<div>

This is an example of using the calc() function with the var() function

</div>

</body>

</html>

# **CSS page-break-before property**

As the name implies, this CSS property is used to add the page break before the element, when printing the document. It inserts a page break before the specified element during printing the document. We cannot use this CSS property on absolutely positioned elements or an empty **<div>** element that does not generate a box.

This CSS property represents whether or not the page break is allowed before the element's box. Including **page-break-before,** the CSS properties **page-break-after** and **page-break-inside** help us to define the behavior of the document when printed.

### **Syntax**

1. page-break-before:  auto | always | left | right | avoid | initial | inherit;

### **Possible values**

The brief description of the values of this CSS property is tabulated as follows.

|  |  |
| --- | --- |
| **Values** | **Description** |
| auto | It is the default value that inserts a page break before the element, if necessary. |
| always | This value always forces a page break before the specified element. |
| avoid | It is used to avoid a page break before the element whenever possible. |
| left | This value forces either one or two page breaks before the element so that the next page will be depicted as the left-hand page. |
| right | The **right** value forces either one or two page breaks before the element so that the next page will be depicted as the right-hand page. |
| initial | It sets the property to its default value. |
| inherit | If this value is specified, the corresponding element uses the computed value of its parent element **page-break-before** property. |

Let's understand the above values using an example of each.

### **Example - Using the auto value**

The value **auto** is the default value that inserts a page break automatically when required. In this example, we are using the two <div> elements and a button. The button is responsible to print the page. After clicking on the button, we will see the effect of the value.

<html>

<head>

<style type = "text/css">

div{

font-size: 20px;

page-break-before: auto;

}

</style>

</head>

<body>

<div>

<h2>Hello World!!</h2>

<h2>Welcome to the iHub.com.</h2>

</div>

<div>

This site is developed so that students may learn computer science related technologies easily. The iHub.com is committed to providing easy and in-depth tutorials on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</div>

<br>

<button onclick = "func()">Print this page</button>

<script>

function func() {

window.print();

}

</script>

</body></html>

# **CSS page-break-inside property**

As the name implies, this CSS property is used to specify the page break inside the element, when printing the document. This CSS property cannot be used on absolutely positioned elements or an empty **<div>** element that does not generate a box. It inserts or avoids the page break inside the specified element during printing the document.

If we want to avoid the page break inside the images, list of items, code snippets, tables, then we can use the **page-break-inside** property.

This CSS property represents whether or not the page break is allowed inside the element's box. Including **page-break-inside,** the CSS properties **page-break-before** and **page-break-after** help us to define the behavior of the document when printed.

### **Syntax**

1. page-break-inside:  auto | avoid | initial | inherit;

### **Possible values**

The brief explanation of the values of this [CSS](https://www.javatpoint.com/css-tutorial) property is tabulated as follows.

|  |  |
| --- | --- |
| **Values** | **Description** |
| auto | It is the default value that inserts a page break inside the element, if necessary. |
| avoid | It is used to avoid a page break inside the element whenever possible. |
| initial | It sets the property to its default value. |
| inherit | If this value is specified, the corresponding element uses the computed value of its parent element **page-break-inside** property. |

Let's understand the above values using an example of each.

### **Example - Using the auto value**

The value **auto** is the default value that inserts a page break automatically when required. This value neither prevents nor forces the page break inside the element's box.

In this example, we are using the two <div> elements and a button. The button is responsible to print the page. After clicking on the button, we will see the effect of the value.

<html>

<head>

<style type = "text/css">

div{

font-size: 20px;

page-break-inside: auto;

}

</style>

</head>

<body>

<div>

<h2>Hello World!!</h2>

<h2>Welcome to the iHubTalent.com.</h2>

</div>

<div>

This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is committed to providing easy and in-depth tutorials on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</div>

<br>

<button onclick = "func()">Print this page</button>

<script>

function func() {

window.print();

}

</script>

</body>

</html>

# **CSS page-break-after property**

This CSS property is used to adjust the page break after the element when printing the document. It inserts a page break after the specified element during printing. We cannot use this property on absolutely positioned elements **(position: absolute;)** or an empty **<div>** element that does not generate a box.

This CSS property represents whether or not the page break is allowed after the element's box. Including **page-break-after,** the CSS properties **page-break-before** and **page-break-inside** help us to define the behavior of the document when printed.

### **Syntax**

1. page-break-after:  auto | always | left | right | avoid | initial | inherit;

### **Possible values**

The brief description of the values of this CSS property is tabulated as follows.

|  |  |
| --- | --- |
| **Values** | **Description** |
| auto | It is the default value that inserts a page break after the element, if necessary. |
| Always | It always forces a page break after the specified element. |
| avoid | It is used to avoid a page break after the element whenever possible. |
| left | It forces either one or two page breaks after the specified element so that the next page will be depicted as the left-hand page. |
| right | It forces either one or two page breaks after the specified element so that the next page will be depicted as the right-hand page. |
| Initial | It sets the property to its default value. |
| Inherit | If this value is specified, the corresponding element uses the computed value of its parent element. |

Let's understand the above values using an example of each.

### **Example - Using the auto value**

The value **auto** is the default value that inserts a page break automatically when required. In this example, we are using the two <div> elements and a button. The button is responsible to print the page. After clicking on the button, we will see the effect of the value.

<html>

<head>

<style type = "text/css">

div{

font-size: 20px;

page-break-after: auto;

}

</style>

</head>

<body>

<div>

<h2>Hello World!!</h2>

<h2>Welcome to the iHub.com.</h2>

</div>

<div>

This site is developed so that students may learn computer science related technologies easily. The iHub.com is committed to providing easy and in-depth tutorials on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</div>

<br>

<button onclick = "func()">Print this page</button>

<script>

function func() {

window.print();

}

</script>

</body>

</html>

# **CSS content property**

This CSS property generates dynamic content. It can be used with the pseudo-elements **::before** and **::after**. This CSS property is fully supported in all browsers and used to insert the generated content on a web page.

It replaces the element with the generated value.

### **Syntax**

1. content: normal | none | counter | string | attr | open-quote | close-quote | no-close-quote | no-open-quote | url | initial | inherit;

### **Property Values**

This CSS property has numerous values that are defined in the following table:

|  |  |
| --- | --- |
| **Values** | **Description** |
| **normal** | It is used to set the default value |
| **none** | This value does not set the content. |
| **counter** | It sets the content as the counter. It is generally a number. It is displayed by using the **counter()** or **counters()** function. |
| **string** | It is used to set any string value. It should always be enclosed within quotation marks. It generates any string after or before the HTML element. |
| **attr** | It inserts the value of the specified attribute after or before the element. If the selector does not have a particular attribute, then an empty string will be inserted. |
| **open-quote** | It is used to insert the opening quotation mark, or it sets the content to be an opening quote. |
| **close-quote** | It is used to insert the closing quotation mark, or it sets the content to be a closing quote. |
| **no-close-quote** | If the closing quotes are specified, then it is used to remove the closing quote from the content. |
| **no-open-quote** | If the opening quotes are specified, then it is used to remove the opening quote from the content. |
| **url** | It is used to set the content to some media, which could be an image, video, audio, and many more. |
| **Initial** | It is used to set the property to its default value. |
| **Inherit** | It inherits the property from its parent element. |

Let's see some of the illustrations of this CSS property.

**Example -** Using **normal** and **none** value

In this example, we are using **::before** pseudo-element for inserting the text **"Welcome "** before the paragraph elements. The text will not be added to those paragraph elements on which we applied the values **normal** and **none.**

<!DOCTYPE html>

<html>

<head>

<title>

CSS content Property

</title>

<style>

body{

text-align: center;

}

p{

font-size: 25px;

}

p::before {

content: "Welcome ";

}

#para::before {

content: normal;

}

#para1::before {

content: none;

}

</style>

</head>

<body>

<h1> CSS content property </h1>

<h2> Use of content: normal; and content: none; </h2>

<p> to the iHub.com </p>

<p id = "para"> This is a paragraph using <b>normal</b> value. </p>

<p id = "para1"> This is another paragraph using <b>none</b> value. </p>

</body>

</html>

# **CSS word-spacing**

This CSS property is used to control the space between the words. Using this property, we can increase or decrease the space between the words.

It modifies the space placed between the words. It is similar to the **letter-spacing** property, but instead of specifying the space between the individual characters, this CSS property defines the space between the words in the piece of text.

A large negative or positive value of **word-spacing** will make the word unreadable. If we apply a very large positive value, then it will cause the appearance of words like a series of unconnected and individual words. A very large negative value will overlap the word to each other, which makes the word unrecognizable.

### **Syntax**

1. word-spacing: normal | length | initial | inherit;

### **Property Values**

**normal:** It is the default value that defines the normal space (0.25em) between the words. It is used to specify the space which is defined by the browser.

**length:** It specifies an extra space between the words in terms of length (in **pt, px, em, cm,** etc.). It allows negative values.

**initial:** It is used to set the property to its default value.

**inherit:** It inherits the value from its parent element.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS word-spacing Property

</title>

<style>

body{

text-align: center;

}

#space{

word-spacing: 40px;

}

p{

color: red;

font-weight: bold;

font-size: 20px;

}

</style>

</head>

<body>

<h1>The word-spacing Property</h1>

<div>

<h2>word-spacing: normal;</h2>

<p>

Welcome to the iHub.com

</p>

</div>

<div>

<h2>word-spacing: 40px;</h2>

<p id= "space">

Welcome to the iHub.com

</p>

</div>

</body>

</html>

# **CSS object-fit property**

This CSS property specifies how a video or an image is resized to fit its content box. It defines how an element fits into the container with an established width and height.

It is generally applied to images or videos. This property specifies how an element reacts to the width and height of its container.

### **Syntax**

1. object-fit:  fill | contain | cover | none | scale-down | initial | inherit;

### **Values**

Mainly five values of this property are defined as follows:

**fill:** It is the default value. Using this value, the entire object fills in the container. The replaced content is sized to fill the content box. If the aspect ratio of the object doesn't match the aspect ratio of the box, the object will be squished or stretched to fit in the box. The image will fill the area; even its aspect ratio is not matched.

**contain:** It resizes the content of an element to stay contained inside the container. It fits the image in the width and height of the box while maintaining the aspect ratio of the image. The replaced content is scaled for maintaining the aspect ratio while fitting within the content box of the element.

**cover:** It resizes the content of an element to cover its container. It fills the entire box with the image. If the aspect ratio of the object is not matched with the aspect ratio of the box, it clips the object to fit.

**none:** It does not resize the replaced content. The image retains its original size and ignores the height and width of the parent.

**scale-down:** It sized the content as **none** or as **contain**. It results in the smallest object size. It finds the smallest concrete object size by comparing the **none** and **contain**.

**initial:** It sets the property to its default value, i.e., the image is stretched to fit in the container because the default value is **fill**.

**inherit:** It inherits the value from its parent element.

Now, let's understand the above property values by using an example of each.

### **Example: Using fill value**

<html>

<head>

<style>

body{

text-align: center;

}

#img1{

width: 300px;

height: 300px;

border: 7px solid red;

}

#obj {

width: 500px;

height: 500px;

object-fit: fill;

border: 7px solid red;

}

#left{

float: left;

text-align: center;

padding-left: 200px;

}

#center{

float: center;

text-align: center;

}

</style>

</head>

<body>

<h1> Example of Object-fit property </h1>

<div id = "left">

<h2> Original image </h2>

<img id = "img1" src = "forest.jpg">

</div>

<div id= "center">

<h2> object-fit: fill; </h2>

<img id = "obj" src="forest.jpg" width="300" height="300"</div>

</body>

</html>

# **CSS object-position property**

This CSS property is used to specify the alignment of the content within the container. It is used with the **object-fit** property to describe how an element like <video> or <img> is positioned with x/y coordinates within its container.

When using the object-fit property, the default value for **object-position** is **50% 50%**, so, by default, all images are positioned in the center of their container. We can change the default alignment by using the **object-position** property.

### **Syntax**

1. object-position: **<position>** | inherit | initial;

#### **Values**

**position:** It defines the position of the video or the image within the container. It takes two numerical values (such as **0 10px**) in which the first value manages the x-axis, whereas the second value controls the y-axis. It can be a string (left, right or center) or can be number (in % or px). It allows negative values. Its default value is **50% 50%**. We can use string values like the **right top, left bottom**, etc.

**initial:** It sets the property to the default value.

**inherit:** It inherits the property from its parent element.

Now, lets' see some examples that will illustrate the **object-position** property.

### **Example**

<!DOCTYPE html>

<head>

<title>CSS object-position property</title>

<style>

body{

text-align: center;

}

img {

width: 400px;

height: 400px;

border: 5px solid red;

background-color: lightblue;

object-fit: none;

object-position: 90px 200px;

}

</style>

</head>

<body>

<h2> object-position: 90px 200px </h2>

<img src= "train1.png"/>

</body>

</html>

# **CSS columns**

The **columns** property in CSS sets the number and width of columns in a single declaration. It is a shorthand property that can take several values at a time.

It is used to set both **column-count** and **column-width** properties at the same time. Both of these properties are used to control how many columns will appear. The **column-count** property is used to set the number of columns, whereas the **column-width** property specifies the width of the columns.

Together using **column-count** and **column-width** properties creates a multi-column layout that breaks automatically into a single column at narrow browser widths without using the media queries. It is not always helpful to use both of them because it can restrict the responsiveness and flexibility of the layout.

If the column-count and width don't fit in the element's width, then the browser reduces the column count automatically to fit the specified column widths.

### **Syntax**

1. columns: auto | column-width column-count| initial | inherit;

### **Values**

The property values are tabulated as follows with their description.

|  |  |
| --- | --- |
| **Value** | **Description** |
| **auto** | It is the default value which sets the values of **column-count** and **column-width** to the default browser values. |
| **column-width** | It is used to set the minimum width for columns. However, the actual width of the column may be narrower or wider based on the available space. |
| **column-count** | It specifies the maximum number of columns. |
| **Initial** | It is used to set the property to its default value. |
| **Inherit** | It inherits the property from its parent element. |

If any of the value is omitted, then by default, the browser assumes the corresponding value to **auto**.

### **Example**

In this example, we are defining two **<div>** elements, including text. For the first div element, the minimum width is 100px, and the maximum no. of columns can be four. Whereas for the second div element, the minimum width is 100px, and the maximum no. of columns can be six.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

.div1 {

  columns: 100px 4;

  border: solid 2px black;

  font-size: 20px;

}

.div2 {

  columns: 100px 6;

  border: solid 2px black;

  font-size: 20px;

  }

**</style>**

**</head>**

**<body>**

**<h1>** The columns Property **</h1>**

**<h4>** The columns property is a shorthand property for column-width and column count **</h4>**

**<h3>** Set the column-width to 100px, and column-count 4 **</h3>**

**<div** class="div1"**>**

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is committed to providing easy and in-depth tutorials on various technologies. Here, you will find lots of tutorials that are easy to understand and learn.

No one is perfect in this world, and nothing is eternally best. But we can try to be better.

**</div>**

**<h3>** Set the column-width to 100px, and column-count to 6 **</h3>**

**<div** class="div2"**>**

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is committed to providing easy and in-depth tutorials on various technologies. Here, you will find lots of tutorials that are easy to understand and learn.

No one is perfect in this world, and nothing is eternally best. But we can try to be better

**</div></body>** **</html>**

# **CSS pointer-events property**

This CSS property specifies whether or not an element shows some action when the pointer event is triggered upon it. The pointer-events are triggered by touch, stylus, mouse click, and many more.

The **pointer-events** property controls how the HTML elements respond to events such as the CSS active/hover states, mouse/touch events, JavaScript click/tap events, and also controls whether or not the cursor is visible.

The **none** value of this property is used for deactivating the click target and allows the elements to target whatever is underneath that element.

### **Syntax**

1. pointer-events: none | auto |initial | inherit;

Although this property includes eleven possible values, but the values given in the above syntax are the valid values for the HTML elements, other values are reserved for use with SVG.

### **Property Values**

**none:** This value indicates that an element doesn't react to pointer events. It avoids all state, click, and the cursor options on the specified HTML element.

**auto:** It is the default value. It indicates that an element must react to pointer events such as the **click** and **:hover**.

Let's understand these values by using some examples.

### **Example - Using none value**

In this example, we are using the none value which does not target the pointer-events.

<!DOCTYPE html>

<html>

<head>

<title>

pointer-events Property

</title>

<style>

body {

text-align:center;

}

p{

font-size: 20px;

pointer-events: none;

}

</style>

</head>

<body>

<CENTER>

<h1> Welcome to the iHub.com </h1>

<h2> pointer-events:auto; </h2>

<p>

<a href="https://www.iHub.com/"> iHubTalent.com </a>

</p>

</body>

</html>

# **CSS hyphens property**

This CSS property is used to control the hyphenation of the text in the block-level elements. It defines how the word is hyphenated if it is too long or when the text wraps across multiple lines.

This property allows us to split the word into two lines to improve the text layout.

### **Syntax**

1. hyphens: none | manual | auto | initial | inherit;

The values of this CSS property are defined as follows.

### **Property Values**

**none:** This value does not hyphenate the words. It never hyphenates the words at line breaks or even if the word is too long.

**manual:** It is the default value that hyphenates the word only when the characters in the word suggest hyphenation opportunities. The two Unicode characters are defined below, which can be used manually to specify the possible line breakpoints in the text.

**U+2010 (HYPHEN) -** It is the 'hard' hyphen character that specifies the visible line-break opportunity. The hyphen is rendered, even if the line is not broken at that point.

**U+00AD (SHY) -** It is an invisible 'soft' hyphen. It is not visibly rendered; instead, it spots the place where the word should be required to break. In [html](https://www.javatpoint.com/html-tutorial), for a soft hyphen, we can use **&shy;**.

**auto:** In this value, the algorithm decides where the words are hyphenated.

**initial:** It sets the property to its default value.

**inherit:** It inherits the value from its parent element.

Let's understand this CSS property by using an example.

### **Example**

<!DOCTYPE html>

<html>

<head>

<title>

CSS hyphens Property

</title>

<style>

div {

width: 50px;

border: 3px solid blue;

background-color: lightblue;

}

.none{

hyphens: none;}

.manual{

hyphens: manual;

}

.auto{

hyphens: auto;

}

</style>

</head>

<body>

<h2> Example of the hyphens property </h2>

<h3> hyphens: none; </h3>

<div class="none">

It is very­very looooo­ooooo­oong word.

</div>

<h3>hyphens: manual</h3>

<div class="manual">

It is very­very looooo­ooooo­ooong word.

</div>

<h3>hyphens: auto</h3>

<div class="auto">

It is very-very looooo-ooooo-oong word.

</div>

</body>

</html>

# **CSS font-variant property**

CSS font-variant property specifies how to set a font variant of an element. Its values may be **normal** and **small-caps**. By using the **small-caps** value, the lowercase letters converted into uppercase, but in a smaller size compared to the original uppercase letters. This property specifies that the text should be appeared in the **small-caps** font or not.

The output generated by the **font-variant** property can be affected by the value of the **text-transform** property. If the value of **font-variant** is **small-caps** and the value of **text-transform** is set to **lowercase**, then the characters will appear in the lowercase. If the value of **font-variant** is **small-caps** and the value of **text-transform** is set to **uppercase**, then the characters will appear in the uppercase.

The **small-caps** value of this CSS property will not work if the letters in the source code are written in the uppercase. For example, suppose we have a paragraph in which letters are written in uppercase, and we are applying the **font-variant** property with the value **small-caps** to the corresponding paragraph, the font will be displayed as the regular uppercase instead of small-caps.

### **Syntax**

1. font-variant: normal | small-caps | initial | inherit;

### **Property values**

The values of this CSS property are defined as follows:

**normal:** It is the default value, which specifies the normal font-face.

**small-caps:** It is used to specify the small-caps font face, in which the lowercase letters are displayed as uppercase letters but in a smaller size.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

### **Example**

In this example, we are applying the **font-variant: small-caps;** on the paragraph elements. In the first paragraph, the text is already in uppercase letters, so the **small-caps**value will not override it. But the text in the second paragraph is affected by the **small-caps** value because it is not in uppercase.

<!DOCTYPE html>

<html>

<head>

<style>

p {

font-variant: small-caps;

font-size: 25px;

}

h2 {

font-variant: normal;

}

</style>

</head>

<body>

<h2> This heading is shown in the normal font. </h2>

<p> HELLO WORLD </p>

<p> hello world. This text is affected by the <b> small-caps </b> value. </p>

</body>

</html>

# **CSS left property**

This CSS property specifies the left offset for the horizontal positioned elements and does not affect the non-positioned elements. It is one of the four offset properties that are **right, top,** and **bottom**.

When both **left** and **right** properties are defined, the right value has a preference if the container is right-to-left, and the left value has preference if the container is left-to-right.

The effect of this property depends on how the corresponding element is positioned, i.e., the value of the **position** property. The **left** property does not affect when the **position** property is set to the value **static**.

The effects of this property on positioned elements other than the value **static** are listed as follows:

* When the element is absolutely or fixed positioned (i.e., **position: absolute;** and **position: fixed;**), the left property specifies the distance between the element's left edge and the left edge of its containing block (ancestor to which the element is relatively positioned).
* If the element is relatively positioned (i.e., **position: relative;**), the left property sets the element's left edge to the left/right from its normal position.
* If the **position** is set to **sticky,** e., **position: sticky;** then the positioning context is the viewport. When the element is inside the viewport, the left property behaves like its position is relative. When the element is outside, the left property behaves like its position is fixed.

### **Syntax**

1. left: auto | length | percentage | initial | inherit;

### **Property Values**

The values of this property are defined as follows:

**auto:** This is the default value. It allows the browser to calculate the left edge position.

**length:** This value defines the position of left property in px, cm, pt, etc. It allows negative values.

**percentage:** This value defines the position of left property in percentage (%). It is calculated to the width of the element's containing block. It also allows negative values.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

### **Example**

In this example, there are four absolutely positioned (i.e., **position: absolute;**) div elements. We are applying the **left** property to them. The div elements with **left: initial;** and **left: auto;** will overlap because of having similar dimensions and default values.

In the output, we can see the div element with the yellow border is with the **left: auto;** and the div element with the light blue border is with the **left: initial;**.

<!DOCTYPE html>

<html>

<head>

<title>

CSS left Property

</title>

<style>

div{

position: absolute;

width: 200px;

height: 200px;

font-size: 30px;

}

#len {

left: 250px;

border: 5px solid lightgreen;

}

#per {

left: 65%;

border: 5px solid blue;

}

#auto {

left: auto;

border: 8px solid yellow;

font-size: 40px;

}

#init {

left: initial;

border: 5px solid lightblue;

}

</style>

</head>

<body>

<h1> Example of the left Property </h1>

<div id = "len"> left: 250px; </div>

<div id = "per"> left: 65%; </div>

<div id = "auto"> left: auto; </div>

<div id = "init"> left: initial; </div>

</body>

</html>

# **CSS right property**

This CSS property specifies the right offset for the horizontal positioned elements and does not affect the non-positioned elements. It is one of the four offset properties that are **left, top,** and **bottom**.

When both **left** and **right** properties are defined, the right value has a preference if the container is right-to-left, and the left value has preference if the container is left-to-right.

The effect of this property depends on how the corresponding element is positioned, i.e., the value of the **position** property. The **right** property does not affect when the **position** property is set to the value **static**.

The effects of this property on positioned elements other than the value **static** are listed as follows:

* When the element is absolutely or fixed positioned (i.e., **position: absolute;** and **position: fixed;**), the **right** property specifies the distance between the element's right edge and the right edge of its containing block (ancestor to which the element is relatively positioned).
* If the element is relatively positioned (i.e., **position: relative;**), the **right** property sets the element's right edge to the left/right from its normal position.
* If the **position** is set to **sticky,** e., **position: sticky;** then, the positioning context is the viewport. When the element is inside the viewport, the **right** property behaves like its position is relative. When the element is outside, then the **right** property behaves like its position is fixed.

### **Syntax**

1. left: auto | length | percentage | initial | inherit;

### **Property Values**

The values of this property are defined as follows:

**auto:** This is the default value. It allows the browser to calculate the **right** edge position.

**length:** This value defines the position of the **right** property in px, cm, pt, etc. It allows negative values.

**percentage:** This value defines the position of the **right** property in percentage (%). It is calculated to the width of the element's containing block. It also allows negative values.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

### **Example**

In this example, there are four absolutely positioned (i.e., **position: absolute;**) div elements. We are applying the **right** property to them. The div elements with **right: initial;** and **right: auto;** will overlap because of having default values and similar dimensions.

<!DOCTYPE html>

<html>

<head>

<title>

CSS right Property

</title>

<style>

div{

position: absolute;

width: 200px;

height: 100px;

font-size: 40px;

}

#len {

right: 200px;

border: 5px solid lightgreen;

}

#per {

right: 50%;

border: 5px solid blue;

}

#auto {

right: auto;

border: 10px solid red;

}

#init {

right: initial;

border: 5px solid yellow;

}

</style>

</head>

<body>

<h1> Example of the right Property </h1>

<div id = "len"> right: 200px; </div>

<div id = "per"> right: 50%; </div>

<div id = "auto"> right: auto; </div>

<div id = "init"> right: initial; </div>

</body>

</html>

# **CSS bottom property**

The **bottom** property in CSS is used to specify the bottom position for the vertical positioned elements. It does not affect the non-positioned elements. It is one of the four offset properties that are **left, right,** and **top**.

The effect of this property depends on the position of the corresponding element, i.e., the value of the **position** property. The **bottom** property does not affect when the **position** property is set to the value **static**.

### **Syntax**

1. bottom: auto | length | percentage | initial | inherit;

### **Property Values**

The values of this property are defined as follows:

**auto:** This is the default value. It allows the browser to calculate the bottom edge position.

**length:** This value defines the position of **bottom** property in px, cm, pt, etc. It allows negative values.

**percentage:** This value defines the position of **bottom** property in percentage (%). It is calculated to the height of the element's containing block. It also allows negative values.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

The effects of this property on positioned elements other than the value **static** are listed as follows:

* When the element is fixed or absolutely positioned (i.e., **position: fixed;** and **position: absolute;**), the **bottom** property specifies the distance between the element's bottom edge and the bottom edge of its containing block (ancestor to which the element is relatively positioned).
* If the element is relatively positioned (i.e., **position: relative;**), the bottom property moves the element's top edge to above/below from its normal position.
* If the **position** is set to **sticky**, i.e., **position: sticky;** then, the positioning context is the viewport. When the element is inside the viewport, the **bottom** property behaves like its position is relative. When the element is outside, the **bottom** property behaves like its position is fixed.

Now, let's understand this property by using some examples.

### **Example**

In this example, there are four absolutely positioned div elements. We are applying the **bottom** property to them. The div elements with **bottom: initial;** and **bottom: auto;** will overlap because of having similar dimensions and default values.

Here, the length of the **bottom** property is defined in **px** and **em**.

<!DOCTYPE html>

<html>

<head>

<title>

CSS bottom Property

</title>

<style>

div{

position: absolute;

width: 150px;

height: 150px;

font-size: 30px;

}

#len {

bottom: 200px;

border: 5px solid green;

}

#em {

bottom: 1em;

border: 5px solid blue;

}

#auto {

bottom: auto;

border: 5px solid red;

}

#init {

bottom: initial;

border: 5px solid darkviolet;

}

h1{

text-align: center;

}

</style>

</head>

<body>

<h1> Example of the bottom Property </h1>

<div id = "len"> bottom: 200px; </div>

<div id = "em"> bottom: 1em; </div>

<div id = "auto"> bottom: auto; </div>

<div id = "init"> bottom: initial; </div>

</body>

</html>

# **CSS top property**

This **top** property in CSS is used to specify the top position for the vertical positioned elements. It does not affect the non-positioned elements. It is one of the four offset properties that are **left, right**, and **bottom**.

The effect of this property depends on how the corresponding element is positioned i.e., the value of the **position** property. The **top** property has no effect when the [**position**](https://www.javatpoint.com/css-position) property is set to the value static.

The effects of this property on positioned elements other than the value **static** are listed as follows:

* When the element is absolutely or fixed positioned (i.e., **position: absolute;** and **position: fixed**;), the **top** property specifies the distance between the element's top edge and the top edge of its containing block (ancestor to which the element is relatively positioned).
* If the element is relatively positioned (i.e., **position: relative**;), the top property moves the element's top edge to above/below from its normal position.
* If the **position** is set to **sticky** (i.e., **position: sticky**;) then, the positioning context is the viewport. When the element is inside the viewport, the **top** property behaves like its position is relative. When the element is outside, then the **top** property behaves like its position is fixed.

### **Syntax**

1. top: auto | length | percentage | initial | inherit;

### **Property Values**

The values of this property are defined as follows:

**auto:** This is the default value. It allows the browser to calculate the top edge position.

**length:** This value defines the position of **top** property in px, cm, pt, etc. It allows negative values.

**percentage:** This value defines the position of **top** property in percentage (%). It is calculated with respect to the height of the element's containing block. It also allows negative values.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Now, let's understand this property by using some examples.

### **Example - Using negative values**

In this example, there are four relatively positioned div elements. We are applying the **top** property to them with negative values. Here, the negative length values of the **top** property are defined in **px** and **em**.

<!DOCTYPE html>

<html>

<head>

<title>

CSS top Property

</title>

<style>

div{

position: relative;

width: 150px;

height: 150px;

font-size: 30px;

}

#len {

top: -75px;

border: 5px solid green;

}

#em {

top: -2em;

border: 5px solid blue;

}

#auto {

top: auto;

border: 5px solid red;

}

#init {

top: initial;

border: 5px solid darkviolet;

}

h1{

text-align: center;

}

</style>

</head>

<body>

<h1> Example of the top Property </h1>

<div id = "len"> top: -75px; </div>

<div id = "em"> top: -2em; </div>

<div id = "auto"> top: auto; </div>

<div id = "init"> top: initial; </div>

</body>

</html>

# **CSS word-break property**

This CSS property specifies how words should break at the end of the line. It defines the line break rules. Using this property, the lines that don't fit in the content box will break at a certain point.

### **Syntax**

1. word-break: normal |keep-all |  break-all | inherit ;

The default value of this property is **normal**. So, it is automatically used when we don't specify any value.

### **Values**

**keep-all:** It breaks the word in the default style. It shouldn't be used for Japanese/Chinese/Korean (CJK) text.

**break-all:** It inserts the word-break between the characters in order to prevent the word overflow. When this value is applied, the browser will break the lines at any point. It can break the word from the middle if it is too long to fit and comes at the end of the line. It does not apply hyphens.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

### **Example**

In this example, there are three containers. We are applying the **normal** value to the first container's content, **break-all** value on the second container's content, and **keep-all** value on the third container's content.

<!DOCTYPE html>

<html>

<head>

<style>

p{

width: 350px;

border: 2px solid black;

text-align: left;

font-size: 20px;

color: blue;

}

.jtp{

word-break: normal;

}

.jtp1{

word-break: break-all;

}

.jtp2{

word-break: keep-all;

}

</style>

</head>

<center>

<body>

<h2>word-break: normal;</h2>

<p class="jtp">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2>word-break: break-all;</h2>

<p class="jtp1">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2>word-break: keep-all;</h2>

<p class="jtp2">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

</center>

</body>

</html>

# **CSS max-height property**

It sets the maximum height of the element's content box. It means that the height of the content box can be smaller than the max-height value, but cannot be greater. It sets the upper bound on the element's height.

When the content is larger than the maximum height, it will overflow. If the content is smaller than the **max-height**, this property does not affect. This property ensures that the value of height property cannot be greater than the value of the **max-height** property. It does not allow negative values.

Sometimes it is useful to limit the element's height to a certain range.

### **Syntax**

1. max-height: none | length | initial | inherit;

The values of this CSS property are defined as follows.

**none:** It is the default value that does not limit the size of the content box.

**length:** This value defines the max-height in px, cm, pt, etc.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Now, let's see an example of this CSS property.

### **Example**

In this example, there are four paragraph elements with the content. We are defining the maximum-height of these paragraphs using the length value of the **max-height** property. The maximum height of the first paragraph is **60px**, the second paragraph is **6em**, the third paragraph is **130pt**, and the fourth paragraph is **5cm**.

The content of the first paragraph is larger than the value of **max-height** property, so in the output, we can see that the content of the first paragraph overflows the content box.

<!DOCTYPE html>

<html>

<head>

<title>

max-height property

</title>

<style>

p{

border: 4px solid blue;

background-color: lightblue;

font-size: 20px;

}

#px {

max-height: 60px;

}

#em {

max-height: 6em;

}

#pt {

max-height: 130pt;

}

#cm {

max-height: 5cm;

}

</style>

</head>

<body>

<h2> max-height: 60px; </h2>

<p id = "px">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<br>

<h2> max-height: 6em; </h2>

<p id = "em">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2> max-height: 130pt; </h2>

<p id = "pt">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2> max-height: 5cm; </h2>

<p id = "cm">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

</body>

</html>

# **CSS max-width property**

Sometimes it is useful to limit the element's width to a certain range. There are two properties **max-width** and **min-width** used to set the maximum and minimum width of the element.

The **max-width** property in CSS is used to set the maximum width of the element's content box. It means that the width of the content box can be smaller than the **max-width** value, but cannot be greater. It sets the upper bound on the element's width.

When the content is larger than the maximum width, then it will automatically change the height of the element. If the content is smaller than the **max-width**, this property has no effect. This property ensures that the value of width property cannot be greater than the value of **max-width** property. It does not allow negative values.

### **Syntax**

1. max-width: none | length | initial | inherit;

The values of this CSS property are defined as follows.

**none:** It is the default value that does not limit the width of the content box.

**length:** This value defines the length of max-width in px, cm, pt, etc.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Now, let's see an example of this CSS property.

### **Example**

In this example, there are four paragraph elements with the content. We are defining the maximum-width of these paragraphs using the length value of **max-width** property. The maximum width of the first paragraph is **175px**, the second paragraph is **20em**, the third paragraph is **350pt**, and the fourth paragraph is **10cm**.

The content of the first paragraph is larger than the value of **max-width** property, so in the output, we can see that the height of the first paragraph changed automatically.

<!DOCTYPE html>

<html>

<head>

<title>

max-width property

</title>

<style>

p{

border: 4px solid blue;

background-color: lightblue;

font-size: 20px;

}

#px {

max-width: 175px;

}

#em {

max-width: 20em;

}

#pt {

max-width: 350pt;

}

#cm {

max-width: 10cm;

}

</style>

</head>

<body>

<h2> max-width: 175px; </h2>

<p id = "px">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2> max-width: 20em; </h2>

<p id = "em">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2> max-width: 350pt; </h2>

<p id = "pt">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2> max-width: 10cm; </h2>

<p id = "cm">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

</body>

</html>

# **CSS min-height property**

It set the minimum-height of the element's content box. It means that the height of the content box can be greater than the **min-height** value, but cannot be shorter. It sets the lower bound on the element's height.

It will be applied when the content is smaller than the minimum height; otherwise, if the content is larger, this property has no effect. This property ensures that the value of height property cannot be less than the value of the **min-height** property. It does not allow negative values.

### **Syntax**

1. min-height: none | length | initial | inherit;

The values of this CSS property are defined as follows:

**none:** It is the default value that does not limit the size of the content box.

**length:** This value defines the min-height in px, cm, pt, etc. Its default value is 0.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Now, let's see an example of using this CSS property.

### **Example**

In this example, there are four paragraph elements with the content. We are defining the minimum-height of these paragraphs using the length value of **min-height** property. The minimum height of the first paragraph is **50px**, the second paragraph is **6em**, the third paragraph is **130pt**, and the fourth paragraph is **5cm**.

<!DOCTYPE html>

<html>

<head>

<title>

min-height property

</title>

<style>

p{

border: 4px solid blue;

background-color: lightblue;

font-size: 20px;

}

#px {

min-height: 50px;

}

#em {

min-height: 6em;

}

#pt {

min-height: 130pt;

}

#cm {

min-height: 5cm;

}

</style>

</head>

<body>

<h3> min-height: 50px; </h3>

<p id = "px">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h3> min-height: 6em; </h3>

<p id = "em">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h3> min-height: 130pt; </h3>

<p id = "pt">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h3> min-height: 5cm; </h3>

<p id = "cm">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

</body>

</html>

# **CSS min-width property**

It is used to set the minimum width of the element's content box. It means that the width of the content box can be greater than the **min-width** value, but cannot be shorter. It sets the lower bound on the element's width.

It will be applied when the content is smaller than the minimum width; otherwise, if the content is larger, this property has no effect. This property ensures that the value of CSS **width** property cannot be less than the value of **min-width** property. It does not allow negative values.

### **Syntax**

1. min-width: none | length | initial | inherit;

The values of this CSS property are defined as follows:

**none:** It is the default value that does not limit the width of the content box.

**length:** This value defines the length of min-width in px, cm, pt, etc.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

Now, let's see an example of using this CSS property.

### **Example**

In this example, there are four paragraph elements with the content. We are defining the minimum-width of these paragraphs using the length value of **min-width** property. The minimum width of the first paragraph is **425px**, the second paragraph is **22em**, the third paragraph is **220pt**, and the minimum width of the fourth paragraph is set to **initial**.

<!DOCTYPE html>

<html>

<head>

<title>

min-width property

</title>

<style>

p{

border: 4px solid blue;

background-color: lightblue;

display: inline-block;

}

#px {

min-width: 425px;

}

#em {

min-width: 22em;

}

#pt {

min-width: 220pt;

}

#cm {

min-width: initial;

}

</style>

</head>

<body>

<h3> min-width: 425px; </h3>

<p id = "px">

Hi, Welcome to the iHubTalent.com.

</p>

<h3> min-width: 22em; </h3>

<p id = "em">

Hi, Welcome to the iHubTalent.com.

</p>

<h3> min-width: 220pt; </h3>

<p id = "pt">

Hi, Welcome to the iHubTalent.com.

</p>

<h3> min-width: initial; </h3>

<p id = "cm">

Hi, Welcome to the iHubTalent.com.

</p>

</body>

</html>

# **CSS border-image property**

This CSS property defines an image to be used as the element's border. It draws an image outside the element and replaces the element's border with the corresponding image. It is an interesting task to replace the border of an element with the image.

The **border-image** property can be applied to all elements except the elements of the internal table (such as tr, th, td) when [**border-collapse**](https://www.javatpoint.com/css-border-collapse-property) is set to **collapse**.

It is the shorthand property for **border-image-source, border-image-slice, border-image-width, border-image-outset**, and **border-image-repeat**. We can set all these properties at once using the **border-image** property. If any of the values are not specified, then they set to their default values. The default value of this property is:

1. border-image: none 100% 1 0 stretch

### **Syntax**

1. border-image: source slice width outset repeat | initial | inherit;

The values of this property are tabulated as follows.

|  |  |
| --- | --- |
| **Values** | **Description** |
| **border-image-source:** | It specifies the source of the border-image. It sets the path of the image, or we can say that it specifies the location of the image to be used as the border. |
| **border-image-slice:** | It is used to divide or slice the image, which is specified by the **border-image-source** property. The values of this property specify how to slice the image for creating the pieces of the border. This property divides the image into nine sections that are:   * Four corners * Four sides, and * a center region   It can accept four unitless positive values. Its default value is **100%**. |
| **border-image-width:** | It sets the width of the border-image. It can accept a unitless positive value, a percentage value, or the keyword **auto**. Its default value is **1**. We can specify up to four values for providing the width of individual sides. |
| **border-image-outset:** | It sets the amount of space by which the border image is set out from its border box. |
| **border-image-repeat:** | It controls the repetition of the image to fill the area of the border. We can specify up to two values for this property. If we specify one value, then it is applied on both vertical and horizontal sides. But if we specify two values, then the first value is applied on horizontal sides, and the second value is applied on vertical sides. The values of this property are listed below.   * stretch * repeat * round * space   The default value of this property is **stretch**. |
| **Initial** | It sets the property to its default value (**border-image:** none 100% 1 0 stretch ). |
| **inherit** | It inherits the property from its parent element. |

Now, let's see some of the examples to understand how to set the border-image using the **border-image** property.

### **Example**

In this example, we are replacing the border of the paragraph elements with the image. In the first paragraph, we are specifying the single value (i.e., **round**) of the **border-image-repeat** property, whereas in the second paragraph, we are specifying two values (**round, stretch**) of it, the first value for the horizontal sides and second value for the vertical sides.

<!DOCTYPE html>

<html>

<head>

<title>

CSS border-image Property

</title>

<style>

p{

border: 10px solid transparent;

padding: 15px;

text-align:center;

font-size: 25px;

color: darkviolet;

}

#border {

border-image: url('border.png') 60 / 20px 20px round;

}

#border1 {

border-image: url('diamond.png') 43 / 10px 15px round stretch;

}

</style>

</head>

<body>

<h2>border-image property</h2>

<p id = "border">

Welcome to the iHubTalent.com

</p>

<p id = "border1">

Welcome to the iHubTalent.com

</p>

</body>

</html>

# **CSS cubic-bezier() function**

It is an inbuilt function in CSS that defines a Cubic Bezier curve. The Bezier curve is the mathematically defined curve used in 2D graphical applications such as (inkspace, adobe illustrator, etc.). This CSS function is the transition timing function, which is used for the smooth and custom transitions.

It is defined by the four points (that are P0, P1, P2, and P3). The points P0 and P3 (called as anchors) are the starting and the ending points, and the points P1 and P2 (called as handles) are the middle points.

For CSS Bezier curves, the points P0 and P3 are always in the same spot. P0 is at (0,0) that represents the initial state and initial time, and P3 is at (1,1), which represents the final state and the final time. This means that the parameters passed to the cubic-bezier() function can only be between 0 and 1.

### **Syntax**

1. cubic-bezier( x1, y1, x2, y2 )

This CSS function accepts four mandatory numeric values. The value of x1 and x2 must be lies between 0 and 1, or the value will be invalid. If we pass the parameters that are not in this interval, the function will set to its default i.e., the linear transition that has the value **cubic-bezier(0, 0, 1, 1)**.

We can understand this CSS function by using the following illustration.

### **Example**

This function can be used with **animation-timing-function** and **transition-timing-function** property.

Here, we are using the **transition-timing-function** property.

<!DOCTYPE html>

<html>

<head>

<title> cubic-bezier() function </title>

<style>

.jtp {

width: 200px;

height: 50px;

background: blue;

transition: width 3s;

animation-timing-function: cubic-bezier(.59,1.01,0,.01)

}

div:hover {

width:300px;

}

p{

font-size: 20px;

color: darkviolet;

}

</style>

</head>

<body>

<h1> The cubic-bezier() Function </h1>

<p> Move your mouse over the below div element, to see the transition effect. </p>

<div class = "jtp">

</div>

</body>

</html>

# **CSS quotes**

The **quotes** property in CSS specifies the type of quotation mark for the quotation used in the sentence. It defines which quotation mark to be used when the quotation is inserted by using the **open-quote** and **close-quote** values of the **content** property.

### **Syntax**

1. quotes: none | string | initial;

### **Values**

**none:** It is the default value that specifies that the **open-quote** and **close-quote** values of the **content** property do not generate any quotation marks.

**string:** This value specifies the type of quotation mark to be used in sentence. The first pair represents the outer-level quotation; the second pair indicates the first nested level, the third pair indicates the third level, and so on.

**initial:** It sets the property to its default value.

There are some of the quotation mark characters tabulated as follows.

|  |  |  |
| --- | --- | --- |
|  | **Description** | **Entity number** |
| **"** | double quote | \0022 |
| **'** | single quote | \0027 |
| **„** | double quote (double low-9) | \201E |
| **«** | double, left angle quote | \00AB |
| **»** | double, right angle quote | \00BB |
| **‹** | single, left angle quote | \2039 |
| **›** | single, right angle quote | \203A |
| **'** | left quote (single high-6) | \2018 |
| **'** | right quote (single high-9) | \2019 |
| **“** | left quote (double high-6) | \201C |
| **”** | right quote (double high-9) | \201D |

We can understand the **quotes** property more clearly by using some examples.

### **Example - Using none value**

In this example, we are using the none **value** of the **quotation** property and the **open-quote** and **close-quote** values of the **content** property. The **none** value prevents the values of the **content** property to generate the quotation marks.

<!DOCTYPE html>

<html>

<head>

<title>

CSS quotes Property

</title>

<style>

p {

quotes: none;

font-size: 20px;

}

p:before{

content: open-quote;

}

p:after{

content: close-quote;

}

</style>

</head>

<body>

<center>

<h1> Example of quotes: none; </h1>

<p> Welcome to the iHubTalent.com </p>

</center>

</body>

</html>

# **CSS transform-origin property**

This CSS property is used to change the position of transformed elements. It is a point around which the transformation is applied. It establishes the origin of the transformation of an element. It can be applied to both 2D and 3D rotations.

The **transform-origin** property must be used with the **transform** property. The 2d transformation can change the x-axis and y-axis of the element, whereas the 3D transformation can change the z-axis along with the x-axis and y-axis.

This property can be specified by using one, two, or three values. The first value affects the horizontal position, the second value affects the vertical position, and the third value indicates the position of the z-axis. The third value can also work on 3D transformations and cannot be defined using a percentage.

* If we specify only one value, the value must be a length or percentage, or one of the keyword values **left, center, right, top,** and **bottom**.
* If we specify two values, the first value must be a length or percentage or the keyword values **left, right,** and **center**. The second value must be a length or percentage or one of the keyword values **center, top**, and **bottom**.
* When we specify three values, then the first two values are same as the two-value syntax, but the third value represents the z-offset, so it must be a length.

The default value of the **transform-origin** property is **50% 50%**, which represents the center of the element.

### **Syntax**

1. transform-origin: x-axis y-axis z-axis | initial | inherit;

The values of this property are tabulated as follows.

|  |  |
| --- | --- |
| **Values** | **Description** |
| **x-axis** | It represents the horizontal position. This value specifies the position of the view at x-axis. Its possible values are length, percentage, **left, right**, and **center**. |
| **y-axis** | It represents the vertical position. This value specifies the position of the view at y-axis. Its possible values are **length, percentage, top, bottom,** and **center**. |
| **z-axis** | This value is used with the 3D transformations. This value specifies the position of the view at z-axis. It can be defined using the **length**values. It does not allow the percentage values. |
| **initial** | It sets the property to its default value. |
| **inherit** | It inherits the property from its parent element. |

Let's understand this property by using some illustrations.

### **Example**

In this example, we are applying the **transform-origin** property with the length and percentage values. There is a rotation of 45deg in both of the elements. Here, there is the 2D transformation of elements.

<!DOCTYPE html>

<html>

<head>

<style>

div{

height: 100px;

width: 400px;

border: 5px dotted violet;

font-size: 20px;

}

.outer {

margin: 100px;

background-color: cyan;

}

.box {

background: url( "diamond.png");

transform: rotate(35deg);

transform-origin: 5% 2%;

}

.outer1{

margin-left: 500px;

background-color: cyan;

}

.box1 {

background: url( "diamond.png");

transform: rotate(45deg);

transform-origin: 5% 2%;

}

</style>

</head>

<body>

<h1> Example of the CSS transform-origin Property </h1>

<div class="outer"> transform-origin: 5% 2%;

<div class="box"></div>

</div>

<div class="outer1"> transform-origin: 100px 200px;

<div class="box1"></div>

</div>

</body>

</html>

# **CSS resize property**

This CSS property allows the user to control the resizing of an element just by clicking or dragging the bottom right corner of the element

This CSS property is used to define how an element is resizable by the user. It doesn't apply on the block or inline elements where **overflow** is set to **visible**. So, to control the resizing of an element, we have to set the **overflow** other than **visible** like **(overflow: hidden** or **overflow: scroll)**.

It is possible to **resize** the elements either in a horizontal or vertical direction or in both directions. After applying the resize property to an element, we can see a small triangular knob at the bottom right corner of the element. The user can drag the knob to enlarge the textarea in either vertical, horizontal, or in both directions.

Sometimes resizing the element may affect the entire layout in an undesirable way. So, depending on the layout, it is sometimes preferable to not allow the element from being resized or restrict the resizability to only one direction.

### **Syntax**

1. resize: none | horizontal | vertical | both | initial | inherit;

### **Property values**

The property values of this CSS property are defined as follows:

**none:** It is the default value of this property, which does not allow resizing the element.

**horizontal:** This value allows the user to resize the element's width. It resizes the element in a horizontal direction. There is a unidirectional horizontal mechanism for controlling the width of an element.

**vertical:** It allows the user to resize the height of an element. It resizes the element in a vertical direction. There is a unidirectional vertical mechanism for controlling the height of an element.

**both:** It allows the user to resize the width and height of an element. It resizes the element in both horizontal and vertical directions.

**initial:** It sets the property to default value.

**inherit:** It inherits the property from its parent element.

Let's understand this CSS by using some examples.

### **Example: Using horizontal value**

This value has a unidirectional resizing that allows the user to adjust the element's width.

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 2px solid red;

padding: 20px;

font-size: 25px;

width: 300px;

resize: horizontal;

overflow: auto;

background-color: lightgreen;

color: blue;

}

</style>

</head>

<body>

<center>

<h1>Example of the resize: horizontal; </h1>

<div>

<p> This is the div element. </p>

<p> To see the resizing effect, click and drag the bottom right corner of this div element. </p>

</div>

</center>

</body>

</html>

# **CSS text-overflow property**

This property specifies the representation of overflowed text, which is not visible to the user. It signals the user about the content that is not visible. This property helps us to decide whether the text should be clipped, show some dots (ellipsis), or display a custom string.

This property does not work on its own. We have to use **white-space: nowrap; and overflow: hidden;** with this property

### **Syntax**

1. text-overflow: clip | ellipsis | string | initial | inherit;

### **Property Values**

**clip:** It is the default value that clips the overflowed text. It truncates the text at the limit of the content area, so that it can truncate the text in the middle of the character.

**ellipsis:** This value displays an ellipsis (?) or three dots to show the clipped text. It is displayed within the area, decreasing the amount of text.

**string:** It is used to represent the clipped text to the user using the string of the programmer's choice. It works only in the Firefox browser.

**initial:** It sets the property to its default value.

**inherit:** It inherits the property from its parent element.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div{

white-space: nowrap;

height: 30px;

width: 250px;

overflow: hidden;

border: 2px solid black;

font-size: 25px;

}

.jtp{

text-overflow: clip;

}

.jtp1 {

text-overflow: ellipsis;

}

h2{

color: blue;

}

div:hover {

overflow: visible;

}

p{

font-size: 25px;

font-weight: bold;

color: red;

}

</style>

</head>

<center>

<body>

<p> Hover over the bordered text to see the full content. </p>

<h2>

text-overflow: clip;

</h2>

<div class="jtp">

Welcome to the iHubTalent.com

</div>

<h2>

text-overflow: ellipsis;

</h2>

<div class="jtp1">

Welcome to the iHubTalent.com

</div>

</center>

</body>

</html>

# **CSS writing-mode property**

The **writing-mode** CSS property specifies whether the text is written in the vertical or horizontal direction. If the direction is vertical, it can be from **right to left (vertical-rl)** or from **left to right (vertical-lr)**. This property sets whether the lines of text are laid out vertically or horizontally. It specifies the direction in which the content flows on the page. It specifies the **block flow direction**, the direction in which the block-level boxes (or containers) are stacked, and the direction in which they flow within the container.

### **Syntax**

1. writing-mode: horizontal-tb | vertical-lr | vertical-rl;

### **Property values**

The values of this property are defined as follows.

**horizontal-tb:** It is the default value of this property. On using this value, the text will be in the horizontal direction and read from left to right and top to bottom.

**vertical-rl:** This value displays the text in the vertical direction, and the text is read from top to bottom and right to left.

**vertical-lr:** This value works similar to the vertical-rl, but the text is read from left to right.

### **Example1**

In this example, we are using all the main values of the [CSS](https://www.javatpoint.com/css-tutorial) **writing-mode** property. Here, there are three paragraph elements having some lines of text. We are applying the **writing-mode: horizontal-tb;** to the first paragraph element, **writing-mode: vertical-lr;** to the second paragraph and **writing-mode: vertical-rl;** to the third paragraph.

In the output, we can see that the content of the second paragraph lays vertically in left to right direction, and the content of the third paragraph also lays out vertically but in right to left direction.

<!DOCTYPE html>

<html>

<head>

<style>

p {

border: 2px solid black;

width: 300px;

height: 300px;

font-size: 23px;

}

#tb {

writing-mode: horizontal-tb;

}

#lr {

writing-mode: vertical-lr;

}

#rl {

writing-mode: vertical-rl;

}

</style>

</head>

<center>

<body>

<h1> Example of CSS writing-mode property </h1>

<h2> writing-mode: horizontal-tb; </h2>

<p id="tb">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2> writing-mode: vertical-lr; </h2>

<p id="lr">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h2> writing-mode: vertical-rl; </h2>

<p id="rl">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

</center>

</body>

</html>

# **CSS background-origin property**

This CSS property helps us to adjust the background image of the webpage. It specifies the background position area, i.e., the origin of a background image. This CSS property will not work when the value of the **background-attachment** is set to be **fixed**.

The **background-origin** property is similar to the **background-clip** property, but it resizes the background instead of clipping it. By default, the origin of an element is the top-left corner of the screen.

If the element has more than one background image, then we can specify a different value of the **background-origin** property for each background-image, separated by commas. Every image will match with the corresponding value of the **background-origin** property.

### **Syntax**

1. background-origin: padding-box | border-box | content-box | initial | inherit;

The values of this property are tabulated as follows.

|  |  |
| --- | --- |
| **Values** | **Description** |
| **padding-box** | It is the default value that positions the background relative to the padding-box. The background starts from the top left corner of the padding edge. |
| **border-box** | It positions the background relative to the border-box. The background starts from the top left corner of the border. |
| **content-box** | It positions the background relative to the content-box. The background starts from the top left corner of the content. |
| **initial** | It sets the property to its default value. |
| **inherit** | It inherits the property from its parent element. |

Let's understand this property by using some illustrations.

### **Example1**

In this example, there are three div elements with a background image. Here, we are using the [**padding-box**](https://www.javatpoint.com/css-padding)**,**[**border-box**](https://www.javatpoint.com/css-border)**, and**[**content-box**](https://www.javatpoint.com/css-content-property) values of the **background-origin** property.

<!DOCTYPE html>

<html>

<head>

<title> background-origin property </title>

<style>

div{

padding: 20px;

width: 350px;

height: 175px;

background-image: url('jtp.png');

background-repeat: no-repeat;

border: 8px dashed blue;

color: red;

font-size: 25px;

text-align: center;

}

#border{

background-origin: border-box;

}

#padding{

background-origin: padding-box;

}

#content{

background-origin: content-box;

}

h2{

color: red;

}

</style>

</head>

<body>

<h2> background-origin: border-box; </h2>

<div id = "border">

<p>

Welcome to the iHubTalent.com

</p>

</div>

<h2> background-origin: padding-box; </h2>

<div id = "padding">

<p>

Welcome to the iHubTalent.com

</p>

</div>

<h2> background-origin: content-box; </h2>

<div id = "content">

<p>

Welcome to the iHubTalent.com

</p></div></body></html>

# **CSS text-orientation property**

This CSS property specifies the orientation of characters in the line of content. It only applies to the vertical mode of content. This property does not affect elements with horizontal writing mode.

It helps us to control the display of languages that use a vertical script. This property has five values: **mixed, sideways, upright, sideways-right,** and **use-glyph-orientation**. Its default value is **mixed.** All of the values of this property work only in vertical mode.

This property depends upon the **writing-mode** property. It works only when the **writing-mode** is not set to **horizontal-tb**.

### **Syntax**

1. text-orientation: mixed | upright | sideways | sideways-right | use-glyph-orientation | initial | inherit;

The values of this property are tabulated as follows.

### **Property values**

|  |  |
| --- | --- |
| **Values** | **Description** |
| **mixed** | It is the default value that rotates the characters 90o degree clockwise. It set the characters of vertical script naturally. |
| **upright** | This value sets the characters of horizontal scripts naturally (upright), as well as the glyphs for the vertical scripts. It makes all characters to be considered as left-to-right. |
| **sideways** | It rotates the line to 90o clockwise. This value causes the characters to be laid out as horizontally. This value does not work in Google Chrome and other major browsers except Firefox, i.e., it only works in Firefox. |
| **sideways-right** | It is kept for compatibility purposes. It is an alias to the value sideways. |
| **use-glyph-orientation** | This value is not used anymore. |
| **initial** | It sets the property to its default value. |
| **inherit** | It inherits the property from its parent element. |

Let's understand this property by using some examples.

### **Example1**

In this example, there are two paragraph elements with the CSS properties **writing-mode: vertical-rl;** and **writing-mode: vertical-lr;** Here, we are applying the **mixed** and **upright** values of the **text-orientation** property.

In the output, we can see the effect of the **upright** value of this CSS property.

<!DOCTYPE html>

<html>

<head>

<style>

#lr, #rl {

border: 2px solid black;

width: 300px;

height: 300px;

}

#lr {

writing-mode: vertical-lr;

text-orientation: mixed;

}

#rl {

writing-mode: vertical-rl;

text-orientation: upright;

}

</style>

</head>

<center>

<body>

<h1> Example of CSS text-orientation property </h1>

<h3> writing-mode: vertical-lr; and text-orientation: mixed; </h3>

<p id = "lr">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

<h3> writing-mode: vertical-rl; and text-orientation: upright; </h3>

<p id = "rl">

Hi, Welcome to the iHubTalent.com. This site is developed so that students may learn computer science related technologies easily. The iHubTalent.com is always providing an easy and in-depth tutorial on various technologies. No one is perfect in this world, and nothing is eternally best. But we can try to be better.

</p>

</center>

</body>

</html>

# **CSS transition-delay property**

This CSS property specifies the duration to start the transition effect. The value of this property is defined as either **seconds (s)** or **milliseconds (ms)**. The CSS transitions are effects that are added to change the element gradually from one style to another, without using flash or JavaScript.

Without using the **transition-delay,** the animation will start with the hover on the element, but using this CSS property, we can delay the animation by an amount of time.

### **Syntax**

1. transition-delay: time | initial | inherit;

The default value of the **transition-delay** property is **0,** which means that the transition will start to occur immediately without any delay.

### **Property values**

**time:** It specifies the amount of time (in seconds or milliseconds) to wait before the transition starts.

**initial:** It sets this property to its default value.

**inherit:** It inherits this property from its parent element.

The delay can be negative, positive, or zero.

The negative value of the **transition-delay** property will immediately start the transition effect i.e., the effect will be animated as though it had already begun. The positive value of this property causes the transition effect to start for the given time.

We can specify multiple delays that are helpful when transitioning several properties. Each delay will be applied to the related property, as defined by the **transition-property** property. For example, suppose we provide two **transition-delay** values. The first value affects the first property given by the **transition-property** property. The second **transition-delay** affects the second property from the list of property names given by the **transition** property.

Let's see some examples of the **transition-delay** property.

### **Example**

In this example, we are using the **transition-property, transition-duration,** and **transition-delay** properties. There is a delay of **0.5s** to start the transition effect, i.e., the background color of the div element will be changed after the given amount of time.

<!DOCTYPE html>

<html>

<head>

<title>

CSS transition-delay Property

</title>

<style>

div{

width: 100px;

height: 100px;

background: lightblue;

transition-property: background-color;

transition-duration: 1s;

transition-delay: 0.5s;

/\* For Safari browser \*/

-webkit-transition-property: background-color;

-webkit-transition-duration: 1s;

-webkit-transition-delay: 0.5s;

}

div:hover {

background-color: brown;

}

</style>

</head>

<body>

<div></div>

<p> Move the cursor over the div element above, to see the transition effect. </p>

</body>

</html>

# **CSS Grid**

A grid can be defined as an intersecting set of horizontal lines and vertical lines. CSS Grid layout divides a page into major regions. It defines the relationship between the parts of a control built from HTML primitives in terms of layer, position, and size. Grid property offers a grid-based layout system having rows and columns. It makes the designing of web pages easy without positioning and floating.

Similar to the table, it enables a user to align the elements into rows and columns. But compare to tables, it is easy to design layout with the CSS grid. We can define columns and rows on the grid by using **grid-template-rows** and **grid-template-columns** properties.

The CSS **grid property** is supported in browsers such as **Google Chrome, Internet Explorer, Firefox, Safari,** and **Opera**.

## Grid Container

We can define the grid container by setting the **display** property to **grid** or **inline-grid** on an element.

Grid container contains grid items that are placed inside rows and columns.

Let's see a simple example of a grid in CSS.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.main {

display: grid;

grid: auto auto / auto auto auto auto;

grid-gap: 10px;

background-color: black;

padding: 10px;

}

.num {

background-color: grey;

text-align: center;

color: white;

padding: 10px 10px;

font-size: 30px;

}

</style>

</head>

<body>

<div class="main">

<div class="num">One</div>

<div class="num">Two</div>

<div class="num">Three</div>

<div class="num">Four</div>

<div class="num">Five</div>

<div class="num">Six</div>

<div class="num">Seven</div>

<div class="num">Eight</div>

</div>

</body>

</html>

# **CSS Layout**

**CSS layout** is easy to design. We can use CSS layout to design our web page such as home page, contact us, about us etc.

There are 3 ways to design layout of a web page:

1. **HTML Div with CSS**: fast and widely used now.
2. **HTML Table**: slow and less preferred.
3. **HTML Frameset**: deprecated now.

A CSS layout can have header, footer, left pane, right pane and body part. Let's see a simple example of CSS layout.

## CSS layout example

<!DOCTYPE html>

<html>

<head>

<style>

.header{margin:-8px -8px 0px;background-image:linear-gradient(145deg,#7379ff,#b524ef);color:white;text-align:center;padding:10px;}

.container{width:100%}

.left{width:15%;float:left;}

.body{width:65%;float:left;background-color:pink;padding:5px;}

.right{width:15%;float:left;}

.footer{margin:-8px;clear:both;background-image:linear-gradient(145deg,#7379ff,#b524ef);color:white;text-align:center;padding:10px;}

</style>

</head>

<body>

<div class="header"><h2>IHubTalent</h2></div>

<div class="container">

<div class="left">

<p>Left Page</p>

</div>

<div class="body">

<h1>Body Page</h1>

<p>Page Content goes here</p><p>Page Content goes here</p><p>Page Content goes here</p>

<p>Page Content goes here</p><p>Page Content goes here</p><p>Page Content goes here</p>

<p>Page Content goes here</p><p>Page Content goes here</p><p>Page Content goes here</p>

<p>Page Content goes here</p><p>Page Content goes here</p><p>Page Content goes here</p>

<p>Page Content goes here</p>

</div>

<div class="right">

<p>Right Page</p>

</div>

</div>

<div class="footer">

<p>Footer</p>

</div>

</body>

</html>

# **CSS Table**

We can apply style on HTML tables for better look and feel. There are some CSS properties that are widely used in designing table using CSS:

* border
* border-collapse
* padding
* width
* height
* text-align
* color
* background-color

## CSS Table Border

We can set border for the table, th and td tags using the CSS border property.

<!DOCTYPE>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

}

</style>

</head>

<body>

<table>

<tr><th>First\_Name</th><th>Last\_Name</th><th>Marks</th></tr>

<tr><td>Sonoo</td><td>Jaiswal</td><td>60</td></tr>

<tr><td>James</td><td>William</td><td>80</td></tr>

<tr><td>Swati</td><td>Sironi</td><td>82</td></tr>

<tr><td>Chetna</td><td>Singh</td><td>72</td></tr>

</table>

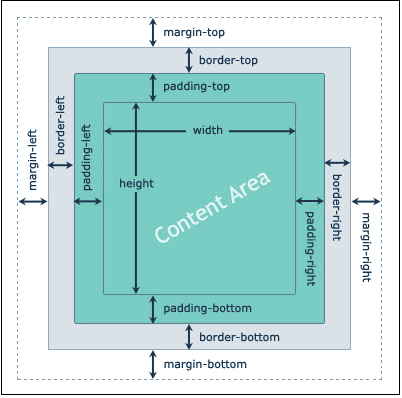
</body>

</html>

# **CSS Box Model**

The components that can be depicted on the web page consist of one or more than one rectangular box. A CSS box model is a compartment that includes numerous assets, such as edge, border, padding and material. It is used to develop the design and structure of a web page. It can be used as a set of tools to personalize the layout of different components. According to the CSS box model, the web browser supplies each element as a square prism.

The following diagram illustrates how the CSS properties of width, height, padding, border and margin dictate that how much space an attribute will occupy on a web page.



The CSS box model contains the different properties in CSS. These are listed below.

* **Border**
* **Margin**
* **Padding**
* **Content**

Now, we are going to determine the properties one by one in detail.

**Border Field**

It is a region between the padding-box and the margin. Its proportions are determined by the width and height of the boundary.

**Margin Field**

This segment consists of the area between the boundary and the edge of the border.

The proportion of the margin region is equal to the margin-box width and height. It is better to separate the product from its neighbor nodes.

**Padding Field**

This field requires the padding of the component. In essence, this area is the space around the subject area and inside the border-box. The height and the width of the padding box decide its proportions.

**Content Field**

Material such as text, photographs, or other digital media is included in this area.

It is constrained by the information edge, and its proportions are dictated by the width and height of the content enclosure.

## Elements of the width and height

Typically, when you assign the width and height of an attribute using the CSS width and height assets, it means you just positioned the height and width of the subject areas of that component. The additional height and width of the unit box is based on a range of influences.

The specific area that an element box may occupy on a web page is measured as follows-

|  |  |
| --- | --- |
| **Size of the box** | **Properties of CSS** |
| Height | height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom |
| Width | width + padding-left + padding-right + border-left + border-right + margin-left + margin-right |

### **Example**

Here, to explain the CSS box model, we have an instance.

<!DOCTYPE html>

<head>

<title>CSS Box Model</title>

<style>

.main

{

font-size:30px;

font-weight:bold;

Text-align:center;

}

.gfg

{

margin-left:50px;

border:50px solid Purple;

width:300px;

height:200px;

text-align:center;

padding:50px;

}

.gfg1

{

font-size:40px;

font-weight:bold;

color:black;

margin-top:60px;

background-color:purple;

}

.gfg2

{

font-size:20px;

font-weight:bold;

background-color:white;

}

</style>

</head>

<body>

<div class = "main">CSS Box-Model Property</div>

<div class = "gfg">

<div class = "gfg1">IHubTalent</div>

<div class = "gfg2">A best portal for learn Technologies</div>

</div>

</body>

</html>