

Cascading Style Sheet (CSS)

CSS is a simple mechanism of describing common presentation semantics for every page in a website. It can control the layout of multiple web pages all at once.

Version of CSS:

CSS 1

CSS 2

CSS 3

CSS Syntax

Selector { Property1: property1-value; Property2: property2-value; PropertyN: propertyN-value; }

```
Selector {  
    Property1: property1-value;  
    Property2: property2-value;  
    PropertyN: propertyN-value;  
}
```

Selector - The selector points to the HTML element or HTML Tag you want to style. It is used to find or select HTML elements based on their element name, id, class, attribute, and more.

Property – It indicates that these properties are defined for that element or selector.

Property-Value – These are values assigned to Properties.

Syntax: -

Selector { Property1: property1-value; Property2: property2-value; PropertyN: propertyN-value; }

Ex: -

p { color: red; font-size: 22px; }

```
p {  
    color: red;  
    font-size: 22px;  
}
```

Element Selectors

The element selector selects elements based on the element name.

Syntax: -

Selector { Property1: property1-value; Property2: property2-value; PropertyN: propertyN-value; }

Ex: -

p { color: red; font-size: 22px; }

h1 { color: blue; font-size: 22em; }

Way of inserting CSS

- External style sheet
- Internal style sheet/ Embedded Style Sheet
- Inline style

External Style Sheet

An external style sheet is a separate document that contains only CSS rules. An external style sheet helps to change the look of an entire website by changing just one css file. It should not contain any HTML Tags. It has .css extension.

Example

```
p { color: red; font-size: 24px;}
```

```
h1 { color: blue; font-size: 24em;}
```

1. Open notepad++ or any other Editor or IDE
2. Write CSS Code
3. Save with .css extension for example geekyshows.css

How to link Web Page to an External Style Sheet

The href attribute with <link> element inside the <head> tag is used to link web page to an external style sheet.

```
<html>
    <head>
        <title>Welcome to Geekyshows</title>
        <link rel="stylesheet" href="geekyshows.css">
    </head>
    <body>
        <h1>I am Heading</h1>
        <p>I am first Paragraph.</p>
        <p>I am second Paragraph</p>
    </body>
</html>
```

Internal Style Sheet

Internal Style sheet is a set of style that is created as a part of HTML document. An internal style sheet may be used if one single page has a unique style.

Internal Style sheets are created using `<style>` element, which is added inside the `<head>` element of the HTML document.

<html>

<head>

<title>Hello CSS</title>

<style type="text/css">

p { color: red; font-size: 24px;}

h1 { color: blue; font-size: 24em;}

</style>

</head>

<body>

<h1>I am Heading</h1>

<p>I am first Paragraph.</p>

<p>I am second Paragraph</p>

</body>

</html>

Inline Styles

Inline style is useful when we need to define specific style for individual elements present on a web page. The *style* attribute in a specific Tag or element, is used to create inline style. The style attribute can contain any CSS property between double quotes.

Ex:-

For paragraph:-

```
<p style="color: red; font-size: 22px;"> I am first paragraph</p>  
<p> I am second paragraph</p>
```

For Heading: -

```
<h1 style="color: red; font-size: 22px;"> I am Heading</h1>
```

```
h1 { color: orange; font-size: 25em; }
```

```
p { color: green; font-size: 20px; }
```

Save as : geekyshows.css

Multiple Style Sheet

<html>

<head>

<title>Hello CSS</title>

<link rel="stylesheet" href="geekyshows.css">

<style type="text/css">

h1 {color: blue; font-size: 35em;}

p {color: red; font-size: 30px;}

</style>

</head>

<body>

<h1 style="color: cyan font-size: 45em;" >I am Heading</h1>

<p style="color: yellow font-size: 40px;" >I am 1st Para</p>

</body>

</html>

External Style Sheet

Internal Style Sheet

Inline Styles

Priority of Style Sheets

- Inline Styles
- External or Internal Style Sheets
- Browser default

Priority - External or Internal

If the internal style is defined after the link to the external style sheet then Internal Style has highest priority.

Ex: -

```
<html>
```

```
<head>
```

```
<title>Hello CSS</title>
```

```
<link rel="stylesheet" href="geekyshows.css">
```

```
<style type="text/css">
```

```
  p { color: red; font-size: 30px; }
```

```
  h1 { color: blue; font-size: 35em; }
```

```
</style>
```

```
</head>
```

External Style Sheet



Internal Style Sheet

Priority - External or Internal

If the internal style is defined before the link to the external style sheet then External Style has highest priority.

Ex: -

```
<html>
```

```
  <head>
```

```
    <title>Hello CSS</title>
```

```
    <style type="text/css">
```

```
      p { color: red; font-size: 30px;}
```

```
      h1 { color: blue; font-size: 35em;}
```

```
    </style>
```

```
    <link rel="stylesheet" href="geekyshows.css">
```

```
  </head>
```

Internal Style Sheet

External Style Sheet

Priority of Style Sheets

- **Inline Styles**
- **External or Internal Style Sheets**
- **Browser default**

Comment

A CSS comment starts with `/*` and ends with `*/`.

```
<html>
  <head>
    <style>
      #geekyshows {
        /* I am comment */
        color: red; font-size: 60px;
      }
    </style>
  </head>
  <body>
    <p id="geekyshows">Hello World!</p>
    <p>This paragraph is not affected by the style.</p>
  </body>
</html>
```

id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the id selector is used to select one unique element.

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Rules

- Must begin with a letter A-Z or a-z
- Id name cannot start with a number
- Must not contain any space characters
- Can be followed by: letters (A-Za-z), digits (0-9), hyphens ("-"), and underscores ("_")
- In HTML, all values are case-insensitive

```
<html>
  <head>
    <style>
      #geekyshows {
        color: red; font-size: 60px;
      }
    </style>
  </head>
  <body>
    <p id="geekyshows">Hello World!</p>
    <p>This paragraph is not affected by the style.</p>
  </body>
</html>
```

Class Selector/ Style Class

Class selectors are used where you want to apply a style either on some of tag or across the several tags without repeating the style rule in an HTML document. Using this method of creating styles, you can create styles in the form of style classes in an external style sheet or as an internal style sheet. The class attribute is used to apply style class in HTML Tag.

Type of Style Class

- Universal style classes
- Element specific style classes

Rules

- Must begin with a letter A-Z or a-z
- A class name cannot start with a number
- Must not contain any space characters
- Can be followed by: letters (A-Za-z), digits (0-9), hyphens ("-"), and underscores ("_")
- In HTML, all values are case-insensitive

Universal style classes

Universal style class starts with a dot operator followed by the class name.

Syntax: -

<style>

.class_name { class definition }

</style>

Ex:-

<style>

.red {color: #FF0000; font-size: 60px;}

</style>

How to use

<html>

<head>

<title>Geekyshows</title>

<style>

.red {color: #FF0000; font-size: 60px;}

</style>

</head>

<body>

<h2 class="red">I am Heading</h2>

<p class="red">I am first paragraph</p>

<p>I am second paragraph</p>

</body>

</html>

Element Specific Style Classes

An element specific style class starts with the element name, followed by a dot operator, which is followed by the class name.

Syntax: -

<style>

element_name.class_name{class definition}

</style>

Ex:-

<style>

p.red {color: #FF0000; font-size: 60px;}

</style>

How to use

<html>

<head>

<title>Geekyshows</title>

<style>

p.red {color: #FF0000; font-size: 60px;}

</style>

</head>

<body>

<h2 class="red">I am Heading</h2>

<p class="red">I am first paragraph</p>

<p>I am second paragraph</p>

</body>

</html>

Use Two or more class

```
<html>

  <head>

    <title>Geekyshows</title>
    <style>

      p.red {color: #FF0000;}
      p.look{font-size: 60px;}
      .algn{text-align: center;}

    </style>

  </head>
  <body>
    <h2 class="algn">I am Heading</h2>
    <p class="red look">I am first paragraph</p>
    <p class="red algn">I am second paragraph</p>
  </body>

</html>
```

Grouping Selectors

To group selectors, separate each selector with a comma.

```
h1 {  
  color: red; font-size: 30px;  
}
```

```
p {  
  color: red; font-size: 30px;  
}
```

```
h2 {  
  color: red; font-size: 30px;  
}
```

```
h3 {  
  color: red; font-size: 30px;  
}
```

```
h1, p, h2, h3 {  
  color: red; font-size: 30px;  
}
```

Background

We can use background properties to enhance the background styles of the elements present in the HTML web page.

background-color
background-image
background-repeat
background-position
background-attachment

background-clip
background-size
background-origin
background

background-color

The background-color property allows to set the background color of an HTML element. We can set this property to a predefined color name, color value or transparent.

Color name – *red, green, blue* etc

Color value

Hex value - *#F0E68C*

RGB value – *rgb(240,230,140)*

- `body { background-color: khaki; }`
- `body { background-color: #F0E68C; }`
- `body { background-color: rgb(240,230,140); }`

background-image

It is used to set background image of an HTML element. By default, the background-image property repeats an image both horizontally and vertically so it covers the entire element. While setting the background image for an HTML element, we should also specify a background color that will be used if the image is not available. We can set this property to *none* or url of an image.

- `body{background-image: url(image/geek.png);}`
- `body{background-image: url(geek.png);}`

background-repeat

It specifies whether the background image is repeated or not. We can set this property to *repeat*, *repeat-x*, *repeat-y* or *no-repeat*. By default, the background-image property repeats an image both horizontally and vertically so it covers the entire element.

Horizontal – repeat-x

Vertical – repeat-y

No repeat – no-repeat

```
body{  
    background-image: url(image/geek.jpg);  
    background-repeat: repeat-x;  
}
```

background-position

It specifies the initial position of a background image. We can set this property to *Left top*, *Center top*, *right top*, *center left*, *center center*, *center right*, *left bottom*, *center bottom*, *right bottom*, *x% y%* , *x-position y-position*.

```
body{  
    background-image: url(image/geek.jpg);  
    background-repeat: no-repeat;  
    background-position: right top;  
}
```

background-attachment

It specifies whether a background image is fixed or scrolls when the user scrolls the rest of the page. We can set this property to either scroll or *fixed*.

```
body{  
    background-image: url(image/geek.jpg);  
    background-repeat: no-repeat;  
    background-position: right top;  
    background-attachment: fixed;  
}
```

background

It works as a shorthand property for all background properties such as *background-color*, *background-image*, *background-repeat*, *background-position* and *background-attachment* etc. It sets all these background properties in one declaration.

```
body {background: khaki url(image/geek.jpg) no-repeat}
```

background-size

The background-size property allows to specify the size of background image. The size can be specified in lengths, percentages, or by using one of the two keywords: *contain* or *cover*.

- contain - scales the background image to be as large as possible
- cover - scales the background image so that the content area is completely covered by the background image

```
body {  
    background-image: url(geek.jpg);  
    background-size: 80px;  
    background-repeat: no-repeat;  
}
```

```
body {  
    background-image: url(geek.jpg);  
    background-size: cover;  
    background-repeat: no-repeat;  
}
```

background-origin

The background-origin property specifies where the background image is positioned. We can set this property to *border-box*, *padding-box*, *content-box*.

- border-box - The background image starts from the upper left corner of the border.
- padding-box - The background image starts from the upper left corner of the padding edge. Its default value.
- content-box - The background image starts from the upper left corner of the content.

background-origin

```
p {  
    border: 5px solid black;  
    padding: 40px;  
    background-image: url(geek1.jpg);  
    background-size: 80px;  
    background-repeat: no-repeat;  
    background-origin: border-box;  
}
```

background-clip

The background-clip property specifies the painting area of the background.

We can set this property to *border-box*, *padding-box*, *content-box*.

- border-box - The background is painted to the outside edge of the border. Its default value
- padding-box - The background is painted to the outside edge of the padding.
- content-box - The background is painted within the content box.

```
body {  
    border: 5px dotted black;  
    padding: 40px;  
    background-color: khaki;  
    background-clip: content-box;  
}
```

Multiple Background Image

```
body {
```

```
    background-image: url(geek.jpg), url(geeky.jpg);
```

```
    background-position: left top, right top;
```

```
    background-size: 80px, 60px;
```

```
    background-repeat: no-repeat, no-repeat;
```

```
}
```

```
body { background: url(geek.png) left top no-repeat, url(geeky.png) right top  
no-repeat;
```

```
}
```

Full Background Image

```
html
```

```
{
```

```
    background: url(bimage.jpg) no-repeat center fixed;
```

```
    background-size: cover;
```

```
}
```

Text

Text Properties provides various text formatting options.

- color
- direction
- text-align
- letter-spacing
- line-height
- text-decoration
- text-indent
- text-shadow
- text-transform
- vertical-align
- white-space
- word-spacing
- text-align-last
- text-overflow
- word-break
- word-wrap

color

Color property set the color of text. We can set this property to either a color name or color value.

Color name – *red, green, blue* etc

Color value

Hex value - *#F0E68C*

RGB value – *rgb(240,230,140)*

```
h1{color: red;}
```

```
p{color: #F0E68C;}
```

direction

It specifies the text direction. We can set this property to either *ltr* (left to right) or *rtl* (right to left).

```
h1 {direction: rtl;}
```

text-align

It specifies alignment of text. We can set this property to *left*, *right*, *center*, or *justify*.

```
h1 {text-align: right;}
```

```
p {text-align: center;}
```


letter-spacing

It specifies the space between text characters. We can set this property to either normal or specify it in the form of the distance between two consecutive letters using identifier, such as px (pixels), in (inches), pt(points) and cm (centimeters).

Length = px, in, pt, cm, em

```
h1 {letter-spacing: 50px;}
```

```
h1 {letter-spacing: 3px}
```

line-height

It specifies the distance between two lines. We can set this property to *normal* or specify it in the form of length, number or percentage.

```
p{line-height: 60%;}
```

```
h1{line-height: 150px;}
```

text-decoration

It specifies the decoration added to the text. We can set this property to *none*, *underline*, *overline*, *line-through*.

```
h1 {text-decoration: underline;}
```

text-indent

It specifies the indentation of the first line of text in a block. we can specify this property in the form of either length or percentage.

Negative values are allowed. The first line will be indented to the left if the value is negative.

```
p{text-indent: 30px;}
```

text-shadow

It specifies the shadow effect added to text. We can add more than one shadow to the text, separated by comma.

Syntax: - text-shadow: h-shadow v-shadow blur-radius color

h-shadow - The position of the horizontal shadow. Negative values are allowed.

v-shadow - The position of the vertical shadow. Negative values are allowed.

blur-radius - The blur radius. Default value is 0

color – The color of the shadow

none - No shadow

```
h1{text-shadow: 2px 2px 5px red;}
```

```
h2{text-shadow: 2px 2px 5px red, 5px 5px 8px blue;}
```

text-transform

text-transform is used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word. We can set this property to *none*, *uppercase*, *lowercase* or *capitalize*.

```
h1 {text-transform: uppercase};
```

vertical-align

It specifies the vertical positioning of text in the element. We can set this property to *baseline*, *sub*, *super*, *top*, *text-top*, *middle*, *bottom*, *text-bottom* or specify in the form length or percentage value.

```
.sp{vertical-align: super};
```

white-space

It specifies how white space inside an HTML element is handled. We can set this property to *normal*, *pre*, *nowrap*, *pre-line* or *pre-wrap*.

normal - Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary.

pre - Whitespace is preserved by the browser. Text will only wrap on line breaks. It is just like the `<pre>` tag in HTML

nowrap - Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a `
` tag is encountered

pre-line - Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary, and on line breaks

pre-wrap - Whitespace is preserved by the browser. Text will wrap when necessary, and on line breaks

white-space

```
p{ white-space: nowrap; }
```

word-spacing

The word-spacing property increases or decreases the white space between words. We can set this property to *normal* or specify in the form of length value.

```
p{word-spacing: 20px};
```

text-align-last

The text-align-last property specifies how to align the last line of a text. The text-align-last property will only work for elements with the text-align property set to "justify". We can set this property to *auto*, *left*, *right*, *center*, *justify*, *start*, or *end*.

```
p { text-align: justify; text-align-last: right; }
```

word-wrap

word-wrap property allows long words to be able to be broken and wrap onto the next line. We can set this property to either *normal* or *break-word*.

```
p { word-wrap: break-word; }
```

word-break

The word-break property specifies line breaking rules for non-CJK scripts. CJK scripts are Chinese, Japanese and Korean ("CJK") scripts. We can set this property to *normal*, *break-all* or *keep-all*.

normal - Default value. Break words according to their usual rules

break-all - Lines may break between any two letters

keep-all - Breaks are prohibited between pairs of letters

```
p{word-break: break-all;}
```

text-overflow

The text-overflow property specifies how overflowed content that is not displayed should be signaled to the user. We can set this property to *clip*, *ellipsis*, or *string*.

clip - Clips the text.

ellipsis - Render an ellipsis ("...") to represent clipped text

string - Render the given string to represent clipped text

```
p { text-overflow: ellipsis;}
```

Font

The Font properties is used to set font related styles for the text present on an HTML.

- font-family
- font-size
- font-stretch
- font-style
- font-variant
- font-weight
- font

font-family

We can specify a list of font names for the text contained inside an HTML element, using font-family property.

Ex:-

```
p { font-family: Arial, serif, "Times New Roman"; }
```

If the name of a font family is more than one word, it must be in quotation marks.

Ex:-

“Times New Roman”

Times

font-size

We can specify a font size for the text. We can specify size in the form of percentage, px, em.

Ex:-

```
p { font-size: 50%; }
```

```
p { font-size: 16px }
```

```
p { font-size: 2em; }
```

```
p { font-size: xx-small; }
```

- xx-small
- x-small
- small
- medium
- large
- x-large
- xx-large
- smaller
- larger

The default text size in browsers is 16px or 1em.

font-stretch

It specifies a normal, condensed or extended font face. We can set this property to *normal*, *wider*, *narrower*, *ultra-condensed*, *extra-condensed*, *condensed*, *semi-condensed*, *semi-expanded*, *expanded*, *extra-expanded*, *ultra-expanded*.

Ex:-

```
p { font-stretch: condensed; }
```

font-style

It specifies the style of the font. We can set this property to *normal*, *italic*, *oblique*.

Ex:-

```
p { font-style: italic; }
```

By default text is normal

font-variant

It specifies whether or not a font is a small-caps font. We can set this property to either *normal* or *small-caps*.

Ex: -

```
p { font-variant: small-caps; }
```

font-weight

The font weight property allows to set the font weight of the text present on an HTML Page. We can set this property to *normal*, *bold*, *bolder*, *lighter*, *number* (*100, 200, 300 upto 900*)

Ex: -

```
p { font-weight: bold; }
```

```
p { font-weight: 500; }
```

font

It defines a shorthand property for font-style, font-variant, font-weight, font-size, line-height, and font-family properties.

Ex:-

```
p { font: bold 20px serif;}
```

The properties must be set in order: "font-style font-variant font-weight font-size/line-height font-family"

The font-size and font-family values are required.

List

List property is used to create list. There are two type of lists:

- Unordered Lists – An unordered list uses bullets as the list item markers.
 - Ordered Lists – An ordered list uses numbers or letters as the list item markers.
- list-style-type
 - list-style-image
 - list-style-position
 - list-style

list-style-image

In this an image is used as the list item marker. We can set this property to none or web address of an image.

Ex: -

```
ul { list-style-image: url('book.png'); }
```


list-style-type

It specifies the appearance of the list item marker if the **list-style-image** property has the value *none* or if the image specified through this property cannot be displayed. We can set this property to *none*, *disc*, *circle*, *square*, *decimal*, *decimal-leading-zero*, *lower-roman*, *upper-roman*, *lower-alpha*, *upper-alpha*, *lower-greek*, *lower-latin*, *upper-latin*, *hebrew*, *armenian*, *georgian*, *cjk-ideographic*, *hiragana*, *katakana*, *hiragana-iroha*, or *katakana-iroha*.

Ex: -

```
ul { list-style-type: circle; }
```

list-style-position

It specifies the position of the list item maker. You can set this property to either *inside* or *outside*.

Ex: -

```
ul { list-style-position: inside; }
```

list-style

It is a shorthand property for *list-style-type*, *list-style-image* and *list-style-position*.

Ex:-

```
ul { list-style: square inside; }
```

Order: list-style-type, list-style-position, list-style-image.

border-style

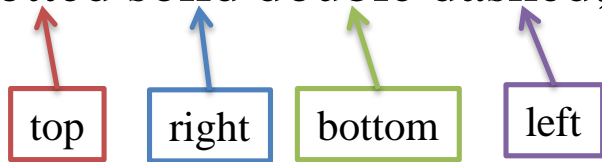
This property sets border style of an element. We can set this property to *none*, *hidden*, *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset*, *outset*.

Ex:-

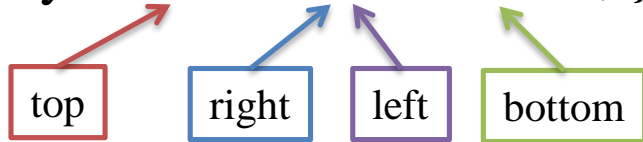
```
p { border-style: dashed; }
```

border-style

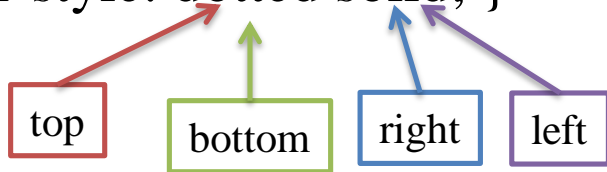
p { border-style: dotted solid double dashed; }



p { border-style: dotted solid dashed; }



p { border-style: dotted solid; }



p { border-style: dotted; }



border-width

It sets width of element's border. We can set this property to thin, medium, thick, length in the form of px.

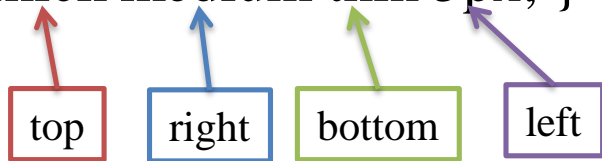
Ex:-

```
p { border-style: solid; border-width: thick; }
```

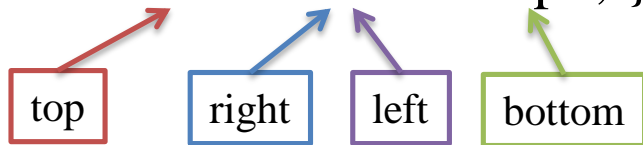
```
p { border-style: solid; border-width: 5px; }
```

border-width

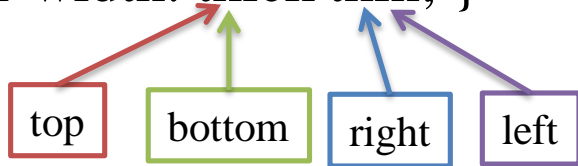
p { border-width: thick medium thin 5px; }



p { border-width: thick medium 5px; }



p { border-width: thick thin; }



p { border-width: 5px; }



border-color

It sets color of border. We can set this property to transparent and color value.

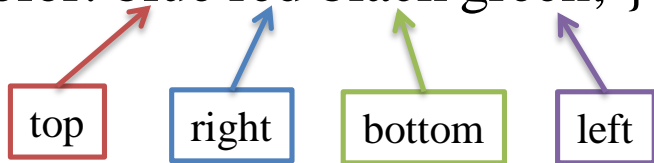
Ex:-

```
p { border-style: solid; border-color: blue;}
```

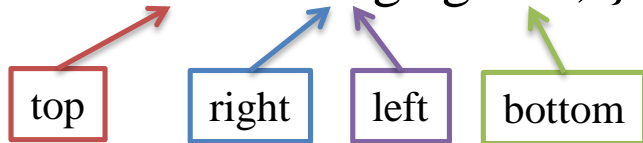
```
p { border-style: solid; border-color: #5A4218;}
```


border-color

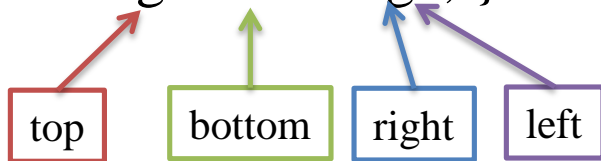
p { border-color: blue red black green; }



p { border-color: blue orange green; }



p { border-color: green orange; }



p { border-color: blue; }



border

It is shorthand of all border properties. The order should be : border-width, border-style, and border-color.

Ex: -

```
p { border: 10px solid blue; }
```

```
p { border: solid blue; }
```

Table

- border
- border-collapse
- border-spacing
- caption-side
- empty-cells
- table-layout

border

For specifying table we use border property.

Ex: -

```
table, th, td { border: 2px solid blue; }
```

border-collapse

This property sets whether the table borders are collapsed into a single border or detached. We can set this property to *separate* (default) and *collapse*.

Ex:-

```
table { border-collapse: collapse; }
```

border-spacing

This property is used to set the distance between the borders of adjacent cells. This will work only when border-collapse is separated. We can set this property to horizontal spacing and vertical spacing in the form of cm, px.

Ex:-

```
table { border-collapse: separated; border-spacing: 5px 10px; }
```

caption-side

This property is used to place a caption for table. We can set this property to *top* (default) or *bottom*.

Ex:-

```
caption { caption-side: bottom; }
```

empty-cells

This property is used to set whether or not to display borders and background on empty cells in a table. It will work only if the border-collapse is set to separate. We can set this property to *show* (default) or *hide*.

Ex:-

```
table { empty-cells: hide; }
```


table-layout

This property is used to set table layout. We can set this property either *auto* (default) or *fixed* .

Ex: -

```
table { table-layout: fixed; }
```

How to set Table width and Height

We use *width* and *height* property to set table width and height.

We can also set table heading (th) width and height.

```
table { width: 100% ;}
```

```
th { height: 10px; }
```

How to set Table Padding

This property is used to control the space between the border and the content in a table, this is used on <td> and <th> elements.

```
th, td { padding: 10px; }
```

How to set Table color

```
table {background-color: pink;}
```

Display

This property is used to define how an element should display. Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

- inline
- block
- flex
- inline-block
- inline-flex
- inline-table
- list-item
- run-in
- table
- table-row-group
- table-caption
- table-column-group
- table-header-group
- table-footer-group
- table-cell
- table-column
- table-row
- none

Display

Block-level Elements - A Block-level element always starts on a new line and takes up the full width available.

Ex: -

<div>

<h1>

<p>

Inline Elements - An inline element does not start on a new line and only takes up as much width as necessary.

Ex: -

<a>

Display

- inline
- block
- flex
- inline-block
- inline-flex
- inline-table
- list-item
- run-in
- table
- table-row-group
- table-caption
- table-column-group
- table-header-group
- table-footer-group
- table-cell
- table-column
- table-row
- none

```
p { display: inline; }
```

Display

- inline – When we set this value, the element does not start on a new line and only takes up as much width as necessary (we can't set width/height it won't work)
- block – When we set this value, element always starts on a new line and takes up the full width available (we can set width/height)
- inline-block – It is combination of inline and block value. It doesn't start on new line but we can set width and height.
- none - The element will not be displayed at all (has no effect on layout)
- flex - Displays an element as a block-level flex container.
- inline-flex - Displays an element as an inline-level flex container.
- inline-table - The element is displayed as an inline-level table
- run-in - Displays an element as either block or inline, depending on context

Display

- table – It works like a <table> element
- table-caption - It works like a <caption> element
- table-column-group - It works like a <colgroup> element
- table-header-group - It works like a <thead> element
- table-footer-group - It works like a <tfoot> element
- table-row-group - It works like a <tbody> element
- table-cell - It works like a <td> element
- table-column - It works like a <col> element
- table-row - It works like a <tr> element
- list-item - It works like a element

Visibility

This property is used to specify whether or not an element is visible. We can set this property to visible (default), hidden or collapse.

- hidden - The element is invisible not removed.
- collapse – This value is only used for table elements. collapse removes 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 will be treated as “hidden”.

Ex:-

```
p { visibility: hidden;}
```

Link

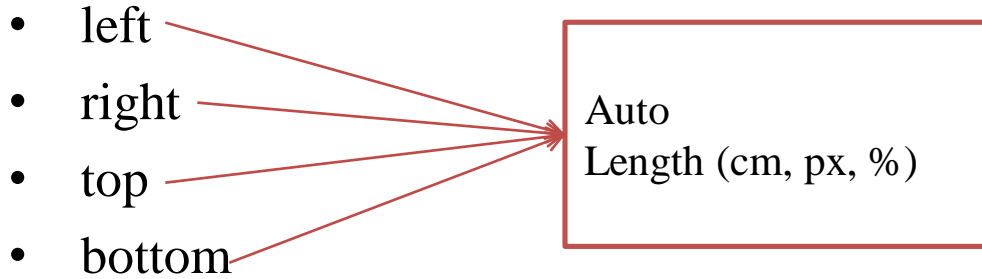
- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouse over it
- a:active - a link the moment it is clicked

Ex:-

```
a:link {  
    color: red;  
}
```

a:hover MUST come after a:link and a:visited

a:active MUST come after a:hover



left: auto

right: auto

top: auto

bottom: auto

left: 55px

right: 23px

top: 20%

bottom: 23cm

left: 50%

right: 50%

top: 40px

bottom: 40%

These properties are used to position an element.

We can not use any of these properties without setting up position property.

Position

- static
- fixed
- relative
- absolute
- left
- right
- top
- bottom

static

When we set static position for an element then it is not positioned in any special way, it is always positioned according to the normal flow of the page.

left, right, top and bottom property won't work with static position.

Ex: -

```
h1 {  
    position: static;  
    border: 5px solid red;  
}
```

```
h1 {  
    position: static;  
    border: 5px solid red;  
    left: 50px;  
}
```

fixed

When we set fixed position for an element it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

```
img {  
    position: fixed;  
    right: 0;  
    top: 50%;  
    border: 3px solid red;  
}
```

relative

When we set relative position for an element, elements positioned to its normal flow.

```
h1 {  
  position: relative;  
  left: 50px;  
  border: 3px solid red;  
}
```


absolute

When we set absolute position for an element, element is positioned to the nearest positioned ancestor. If there is no positioned ancestor then it follow the normal position according to browser.

```
div {  
  position: relative;  
  width: 400px;  
  height: 200px;  
  border: 3px solid red;  
}
```

```
div {  
  position: absolute;  
  top: 80px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid red;  
}
```

How to set element Layer (z-index)

z-index property is used to set the layer of elements. We can set this property to *auto* (default) and *number*. It works only with position property value: fixed, relative, absolute.

The element which has higher number appear at the top.

Ex: -

```
img {  
    position: fixed;  
    z-index: 565;  
}
```

Overflow

This property is used to define what happens if content overflows an element's box. It means it will define whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area. This property only works for block elements with a specified height. We can set this property to *visible* (default), *scroll*, *hidden*, *auto*.

Ex:-

```
p { overflow: hidden; }
```

Overflow-x

This property is used to define whether to clip content, render a scroll bar, or display overflow content of a block-level element, when it overflows at the left and right edges. We can set this property to *visible* (default), *hidden*, *scroll*, *auto*.

Ex: -

```
p { overflow-x: hidden; }
```

Overflow-y

This property is used to define whether to clip content, render a scroll bar, or display overflow content of a block-level element, when it overflows at the top and bottom edges. We can set this property to *visible* (default), *hidden*, *scroll*, *auto*.

Ex: -

```
p { overflow-y: hidden; }
```

float

This property is used to specify that an element should be placed along the left or right side of its container, where text and inline elements will wrap around it. We can set this property to *left*, *right*, *none* (default).

Ex: -

```
img { float: left;}
```

clear

This property is used to specify whether an element can be next to floating elements that precede it or must be moved down (cleared) below them. We can set this property to *left*, *right*, *both*, *none* (default).

Ex:-

```
p { clear: left;}
```

float

This property is used to specify that an element should be placed along the left or right side of its container, where text and inline elements will wrap around it. We can set this property to *left*, *right*, *none* (default).

Ex: -

```
img { float: left;}
```


clear

This property is used to specify whether an element can be next to floating elements that precede it or must be moved down (cleared) below them. We can set this property to *left*, *right*, *both*, *none* (default).

Ex:-

```
p { clear: left;}
```

<html>

<head>

<style>

p{

color: red; font-size: 60px;

}

</style>

</head>

<body>

<p >Hello World!</p>

<p>This paragraph is not affected by the style.</p>

</body>

</html>

Id Selector

<html>

<head>

<style>

```
#geekyshows {  
    color: red; font-size: 60px;  
}
```

</style>

</head>

<body>

<p id="geekyshows">Hello World!</p>

<p>This paragraph is not affected by the style.</p>

</body>

</html>

Class Selector

- Universal

<html>

<head>

<style>

.red {color: #FF0000; font-size: 60px;}

</style>

</head>

<body>

<h2 class="red">I am Heading</h2>

<p class = "red">I am first paragraph</p>

<p>I am second paragraph</p>

</body>

</html>

Class Selector

- Element Specific

<html>

<head>

<style>

p.red {color: #FF0000; font-size: 60px;}

</style>

</head>

<body>

<h2 class="red">I am Heading</h2>

<p class = "red">I am first paragraph</p>

<p>I am second paragraph</p>

</body>

</html>

Use Two or more class

<html>

<head>

<style>

```
p.red {color: #FF0000;}  
p.look{font-size: 60px;}  
.algn{text-align: center;}
```

</style>

</head>

<body>

```
<h2 class="algn">I am Heading</h2>  
<p class="red look">I am first paragraph</p>  
<p class="red algn">I am second paragraph</p>
```

</body>

</html>

div and span

`<div> </div>`

The div tag is used to group various other HTML elements. It is a block level element. We can say it also create a block.

Ex:-

- `<div> This is a Div Tag </div>`
- `<div>`
 `<h1> Heading </h1>`
 `<p> Para </p>`
`</div>`

div and span

` `

The span tag is used to group inline elements. It is an inline element.

Ex:-

- `<p> I am example of span tag </p>`
- `<p> I am example of span tag </p>`

- ID
- Class
- Element

Child Selector

<html>

<head> </head>

<body>

<div>

<p>Hello World!</p>

<p>This paragraph is not affected by the style.</p>

<div>

<p>Geekyshows </p>

</div>

<div>

</body>

</html>

Child Selector

<html>

<head> </head>

<body>

<div>

<p>Hello World!</p>

<p>This paragraph is not affected by the style.</p>

<div>

<p>Geekyshows </p>

</div>

<div>

</body>

</html>

Pseudo-classes

A pseudo-class is used to define a special state of an element.

Syntax:-

```
selector:pseudo-class { property:value; }
```

Ex: -

```
div: hover { background-color: red; }
```

Pseudo-classes

- :link
- :visited
- :hover
- :active
- :focus
- :checked
- :first-child
- :first-of-type
- :nth-child(n)

first-child

This will set CSS rule to every first element of selector which is someone's child.

last-child

This will set CSS rule to every last element of selector which is someone's child.

first-of-type

This will set CSS rule to every first selector type of element.

last-of-type

This will set CSS rule to every last selector type of element.

Pseudo Element

This is used to style specified parts of an element.

Syntax: -

```
selector::pseudo-element { property:value; }
```

Ex: -

```
p::first-letter { color: red; }
```

- ::first-letter
- ::first-line
- ::selection
- ::before
- ::after

::first-letter

This is used to add a style to the first letter of the specified selector (each).
This can only be used with block-level elements.

- background properties
- font properties
- padding properties
- color properties
- margin properties
- border properties
- line-height
- text-transform
- text-decoration
- vertical-align (only if float is 'none')
- float
- clear

Ex: -

```
p::first-letter { color: red; }
```

::first-line

This is used to add a style to the first line of the specified selector (each). This can only be used with block-level elements.

- background properties
- font properties
- text-decoration
- color properties
- word-spacing
- letter-spacing
- text-transform
- vertical-align
- line-height
- clear

Ex: -

```
p::first-line { color: red;}
```

::selection

This is used to matches the portion of an element that is selected by a user then apply the CSS rule in the selected content.

- background properties
- color properties
- cursor properties
- outline properties

Ex: -

```
p::selection { background-color: red; }
```

::before

This is used to insert something before the content of each specified selector (each). We use content property to specify the content to insert.

Ex: -

```
p::before { content: “ Geekyshows Says”; }
```

::after

This is used to insert something after the content of each specified selector (each). We use content property to specify the content to insert.

Ex: -

```
p::after { content: “ By Geekyshows”; }
```


Content

It is used to generated content at run time dynamically with the ::before and ::after pseudo-elements, to insert generated content.

Ex:-

```
p::before { content: "Geekyshows Says "};
```

- normal
- none
- counter
- attr(attribute)
- string
- open-quote
- close-quote
- no-open-quote
- no-close-quote
- url(url)

Margin

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

Margin Property

This property is used to set the margin of specified element.

- margin-left
- margin-right
- margin-top
- margin-bottom
- margin

margin-left

This property is used to set left margin of element. We can set this property to auto or length in the form of %, cm, px etc. It is by default set 0px.

When we set auto value browser decide the margin.

Ex: -

```
img { margin-left: 20%; }
```

```
h1 { margin-left: 40px; }
```

```
p { margin-left: auto; }
```

margin-right

This property is used to set right margin of element. We can set this property to auto or length in the form of %, cm, px etc. It is by default set 0px.

When we set auto value browser decide the margin.

Ex: -

```
img { margin-right: 20%; }
```

```
h1 { margin-right: 40px; }
```

```
p { margin-right: auto; }
```

margin-top

This property is used to set top margin of element. We can set this property to auto or length in the form of %, cm, px etc. It is by default set 0px.

When we set auto value browser decide the margin.

Ex: -

```
img { margin-top: 20%; }
```

```
h1 { margin-top: 40px; }
```

```
p { margin-top: auto; }
```

margin-bottom

This property is used to set bottom margin of element. We can set this property to auto or length in the form of %, cm, px etc. It is by default set 0px.

When we set auto value browser decide the margin.

Ex: -

```
img { margin-bottom: 20%; }
```

```
h1 { margin-bottom: 40px; }
```

```
p { margin-bottom: auto; }
```


margin

Its shorthand margin property It declares all margin properties in one single line. We can set this property to auto or length in the form of %, cm, px etc. It is by default set 0px.

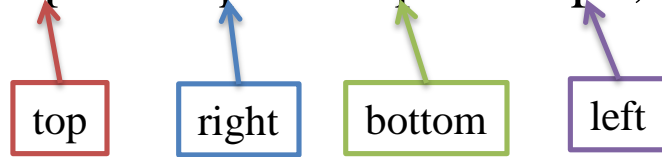
When we set auto value browser decide the margin.

Ex: -

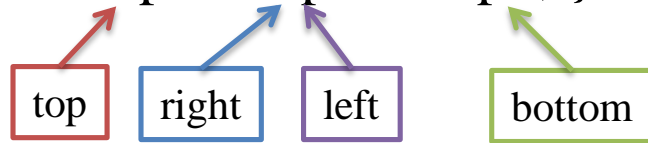
```
img { margin: 10px 20px 30px 40px;}
```

margin

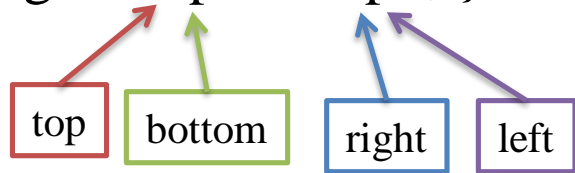
p { margin: 10px 20px 30px 40px; }



p { margin: 10px 20px 30px; }



p { margin: 10px 20px; }



p { margin: 20px; }



Padding

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

Padding Property

This property is used to set the padding of specified element.

- padding-left
- padding-right
- padding-top
- padding-bottom
- padding

padding-left

This property is used to set left padding of element. We can set this property to length in the form of %, cm, px etc.

Ex: -

```
img { padding-left: 20%; }
```

```
h1 { padding-left: 40px; }
```

padding-right

This property is used to set right padding of element. We can set this property to length in the form of %, cm, px etc.

Ex: -

```
img { padding-right: 20%; }
```

```
h1 { padding-right: 40px; }
```

padding-top

This property is used to set top padding of element. We can set this property to length in the form of %, cm, px etc.

Ex: -

```
img { padding-top: 20%; }
```

```
h1 { padding-top: 40px; }
```

padding-bottom

This property is used to set bottom padding of element. We can set this property to length in the form of %, cm, px etc.

Ex: -

```
img { padding-bottom: 20%; }
```

```
h1 { padding-bottom: 40px; }
```


padding

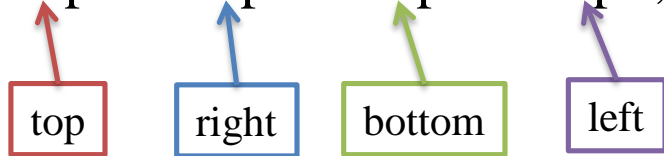
Its shorthand padding property It declares all padding properties in one single line. We can set this property to length in the form of %, cm, px etc.

Ex: -

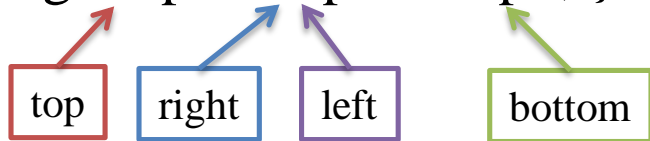
```
img { padding: 10px 20px 30px 40px;}
```

padding

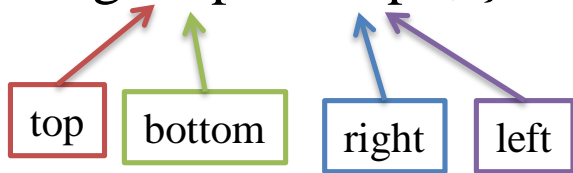
p { padding: 10px 20px 30px 40px; }



p { padding: 10px 20px 30px; }



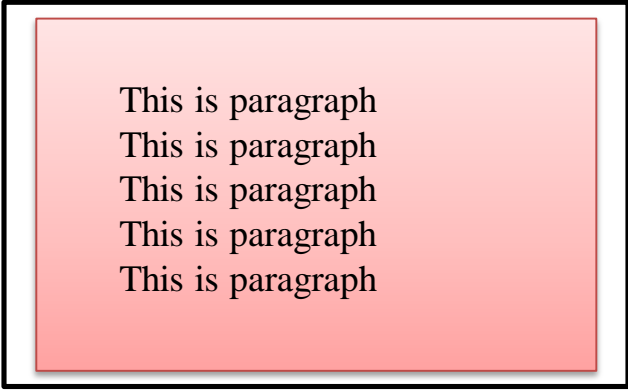
p { padding: 10px 20px; }



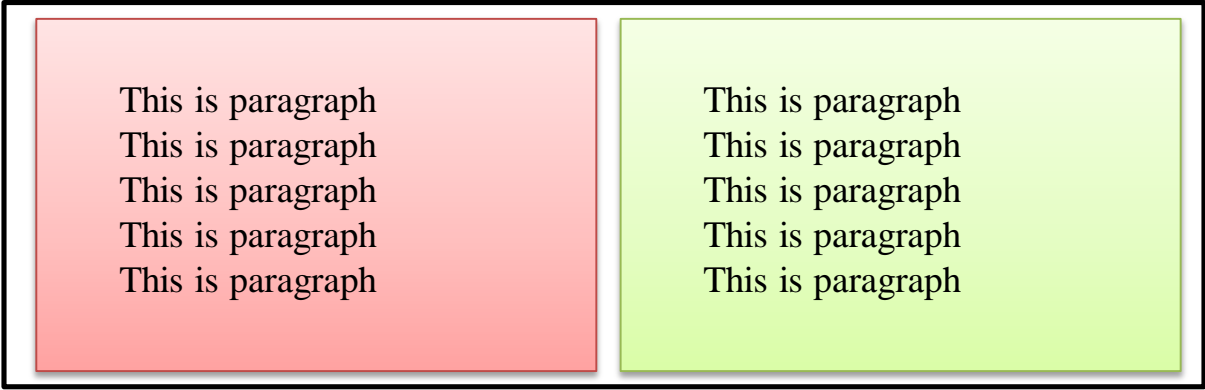
p { padding: 20px; }



Margin Vs Padding

A diagram illustrating padding. It consists of a large black rectangular border. Inside this border is a smaller, light red rectangular area. The text "This is paragraph" is repeated five times, stacked vertically, within the red area. The text is indented from the left and right edges of the red area, demonstrating padding.

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

A diagram illustrating margin. It consists of a large black rectangular border. Inside this border are two side-by-side rectangular areas: a light red one on the left and a light green one on the right. Each area contains the text "This is paragraph" repeated five times, stacked vertically. The text is aligned to the left of each colored area, and there is a clear gap between the text and the black border, demonstrating margin.

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

This is paragraph
This is paragraph
This is paragraph
This is paragraph
This is paragraph

- border-left-style
- border-left-width
- border-left-color
- border-left

border-left-style

- none - This is default which specifies no border.
- hidden – This is very similar to none value, except in border conflict resolution for table elements
- solid – It specifies a solid border
- dotted – It specifies a dotted border
- double – It specifies a double border
- dashed – It specifies a dashed border
- groove – It specifies a 3D grooved border.
- ridge – It specifies a 3D ridged border.
- inset – It specifies a 3D inset border.
- outset – It specifies a 3D outset border.

border-left-width

- medium – This is default value which specifies a medium left border.
- thin – It specifies a thin left border
- thick – It specifies a thick left border
- length – We can define left border length in the form of % px etc.

border-left-color

- color value
- transparent

border-left

Syntax:-

```
selector{ border-left: border-left-width border-left-style border-left-color;}
```

Ex:-

```
p{ border-left: 10px solid red;}
```


- border-right-style
- border-right-width
- border-right-color
- border-right

border-right-style

- none - This is default which specifies no border.
- hidden – This is very similar to none value, except in border conflict resolution for table elements
- solid – It specifies a solid border
- dotted – It specifies a dotted border
- double – It specifies a double border
- dashed – It specifies a dashed border
- groove – It specifies a 3D grooved border.
- ridge – It specifies a 3D ridged border.
- inset – It specifies a 3D inset border.
- outset – It specifies a 3D outset border.

border-right-width

- medium – This is default value which specifies a medium right border.
- thin – It specifies a thin right border
- thick – It specifies a thick right border
- length – We can define right border length in the form of % px etc.

border-right-color

- color value
- transparent

border-right

Syntax:-

```
selector{ border-right: border-right-width border-right-style  
border-right-color;}
```

Ex:-

```
p{ border-right: 10px solid red;}
```

- border-top-style
- border-top-width
- border-top-color
- border-top

border-top-style

- none - This is default which specifies no border.
- hidden – This is very similar to none value, except in border conflict resolution for table elements
- solid – It specifies a solid border
- dotted – It specifies a dotted border
- double – It specifies a double border
- dashed – It specifies a dashed border
- groove – It specifies a 3D grooved border.
- ridge – It specifies a 3D ridged border.
- inset – It specifies a 3D inset border.
- outset – It specifies a 3D outset border.

border-top-width

- medium – This is default value which specifies a medium top border.
- thin – It specifies a thin top border
- thick – It specifies a thick top border
- length – We can define top border length in the form of % px etc.

border-top-color

- color value
- transparent

border-top

Syntax:-

```
selector{ border-top: border-top-width border-top-style border-top-color;}
```

Ex:-

```
p{ border-top: 10px solid red;}
```

- border-bottom-style
- border-bottom-width
- border-bottom-color
- border-bottom

border-bottom-style

- none - This is default which specifies no border.
- hidden – This is very similar to none value, except in border conflict resolution for table elements
- solid – It specifies a solid border
- dotted – It specifies a dotted border
- double – It specifies a double border
- dashed – It specifies a dashed border
- groove – It specifies a 3D grooved border.
- ridge – It specifies a 3D ridged border.
- inset – It specifies a 3D inset border.
- outset – It specifies a 3D outset border.

border-bottom-width

- medium – This is default value which specifies a medium bottom border.
- thin – It specifies a thin bottom border
- thick – It specifies a thick bottom border
- length – We can define bottom border length in the form of % px etc.

border-bottom-color

- color value
- transparent

border-bottom

Syntax:-

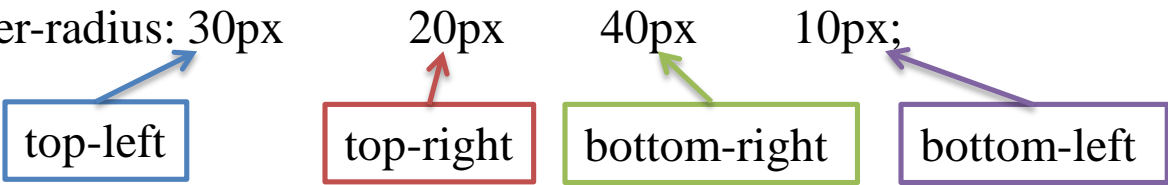
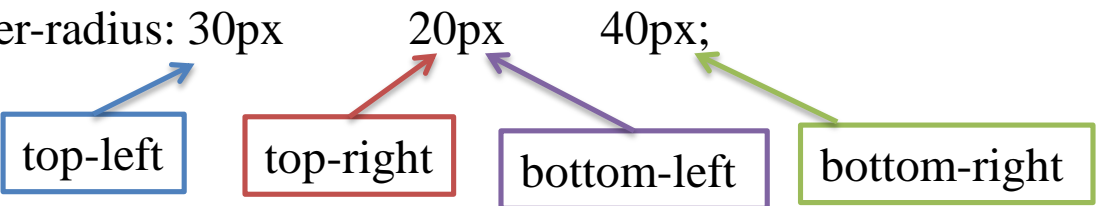
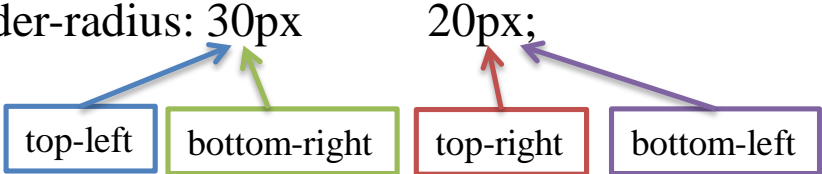
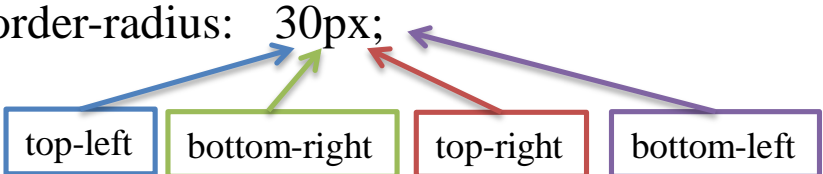
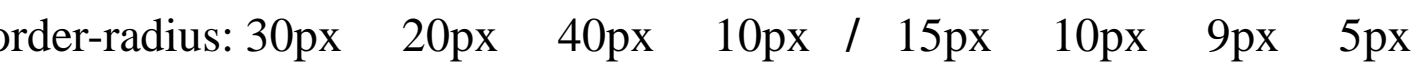
```
selector{ border-bottom: border-bottom-width border-bottom-style border-bottom-color;}
```

Ex:-

```
p{ border-bottom: 10px solid red;}
```

Border- Rounded Corners

- border-top-left-radius
- border-top-right-radius
- border-bottom-left-radius
- border-bottom-right-radius
- border-radius

- border-radius: 30px 20px 40px 10px;

- border-radius: 30px 20px 40px;

- border-radius: 30px 20px;

- border-radius: 30px;

- border-radius: 30px 20px 40px 10px / 15px 10px 9px 5px


Border Image

- border-image-source
- border-image-slice
- border-image-width
- border-image-outset
- border-image-repeat
- border-image

border-image-source

This property is used to specify source of image which is used as border. We can set this property to none or image.

Ex:-

```
#borderis { border-image-source: none; }
```

```
#borderis { border-image-source: url(border.png); }
```

If the value is "none" or if the specified image can not be found, the border styles will be used.

border-image-slice

This property is used to specify how to slice the image specified by border-image-source property. The image is always sliced into nine sections: four corners, four edges and the middle. The "middle" part is treated as fully transparent, unless the fill keyword is set. We can set this property to number, % and fill.

Ex:-

```
#borderis { border-image-slice: 10; }
```

```
#borderis { border-image-slice: fill); }
```

border-image-outset

This property is used to specify the amount by which the border image area extends beyond the border box. We can set this property to number and length.

Ex:-

```
#borderis { border-image-outset: 10; }
```

```
#borderis { border-image-outset: 10px; }
```

border-image-repeat

This property is used to specify whether the border image should be repeated, rounded or stretched. We can set this property to *stretch*, *round*, *repeat* and *space*.

Ex:-

```
#borderis { border-image-repeat: repeat; }
```

border-image

Its shorthand of all other border-image property.

Syntax: -

Selector {border-image: border-image-source border-image-slice border-image-width border-image-outset border-image-repeat; }

Ex:-

```
#borderis {border-image: url(border.png) 10 round;}
```

box-shadow

This property is used to attach one or more shadows to an element.

Syntax: - box-shadow: **h-shadow** **v-shadow** **blur** **spread** **color** **inset**

h-shadow - The position of the horizontal shadow. Negative values are allowed.

v-shadow - The position of the vertical shadow. Negative values are allowed.

blur - The blur distance.

Spread – Size of shadow. Negative values are allowed

color – The color of the shadow

none - No shadow. Default

inset - Changes the shadow from an outer shadow (outset) to an inner shadow

```
div {box-shadow: 2px 2px 5px red;}
```

```
div {box-shadow: 2px 2px 5px red, 5px 5px 8px blue;}
```


Cursor

- default - The default cursor
- auto - Default. The browser sets a cursor
- all-scroll - The cursor indicates that something can be scrolled in any direction
- none - No cursor is rendered for the element
- not-allowed - The cursor indicates that the requested action will not be executed
- text - The cursor indicates text that may be selected
- URL - A comma separated list of URLs to custom cursors. Always specify a generic cursor at the end of the list, in case none of the URL-defined cursors can be used
- row-resize - The cursor indicates that the row can be resized vertically
- s-resize - The cursor indicates that an edge of a box is to be moved down (south)
- se-resize - The cursor indicates that an edge of a box is to be moved down and right (south/east)
- sw-resize - The cursor indicates that an edge of a box is to be moved down and left (south/west)

- n-resize - The cursor indicates that an edge of a box is to be moved up (north)
- ne-resize - The cursor indicates that an edge of a box is to be moved up and right (north/east)
- nesw-resize - Indicates a bidirectional resize cursor
- ns-resize - Indicates a bidirectional resize cursor
- nw-resize - The cursor indicates that an edge of a box is to be moved up and left (north/west)
- nwse-resize - Indicates a bidirectional resize cursor
- copy - The cursor indicates something is to be copied
- alias - The cursor indicates an alias of something is to be created
- e-resize - The cursor indicates that an edge of a box is to be moved right (east)
- ew-resize - Indicates a bidirectional resize cursor
- pointer - The cursor is a pointer and indicates a link
- progress - The cursor indicates that the program is busy (in progress)
- zoom-out - The cursor indicates that something can be zoomed out
- zoom-in - The cursor indicates that something can be zoomed in

- vertical-text - The cursor indicates vertical-text that may be selected
- w-resize - The cursor indicates that an edge of a box is to be moved left (west)
- wait - The cursor indicates that the program is busy
- context-menu - The cursor indicates that a context-menu is available
- col-resize - The cursor indicates that the column can be resized horizontally
- cell - The cursor indicates that a cell (or set of cells) may be selected
- no-drop - The cursor indicates that the dragged item cannot be dropped here
- crosshair - The cursor render as a crosshair
- grab - The cursor indicates that something can be grabbed
- grabbing - The cursor indicates that something can be grabbed
- help - The cursor indicates that help is available
- move - The cursor indicates something is to be moved

filter

This property is used to define visual effects like blur and brightness to an element.

Ex:-

```
img { filter: blur(10px);}
```

- none - Default value. Specifies no effects
- blur(px) – This is used to apply blur effect to the image. A larger value will create more blur.
- brightness(%) – This is used to adjust the brightness of the image. 0% will make the image completely black. 100% is default and represents the original image. Values over 100% will provide brighter results.
- contrast(%) – This is used to adjust the contrast of the image. 0% will make the image completely black. 100% is default and represents the original image. Values over 100% will provide results with less contrast.

- `drop-shadow(h-shadow v-shadow blur spread color)` – This is used to apply a drop shadow effect to the image. This is very similar to `box-shadow` property.

h-shadow – This is used to specify a pixel value for the horizontal shadow. Negative values place the shadow to the left of the image.

v-shadow – This is used to specify a pixel value for the vertical shadow. Negative values place the shadow above the image.

blur(px) – This is used to add blur effect to the shadow. A larger value will create more blur. Negative values are not allowed. If no value is specified, 0 is used.

spread(px) – This is used to specify size of shadow. Negative values will cause the shadow to shrink. If not specified, it will be 0.

color – This is used to add a color to the shadow. If not specified, the color depends on the browser (often black).

- `grayscale(%)` – This is used to convert the image to grayscale. 0% is default and represents the original image. 100% will make the image completely gray. Negative values are not allowed.
- `hue-rotate(deg)` - This is used to specify a hue rotation on the image. 0deg is default, and represents the original image. Maximum value is 360deg.
- `invert(%)` - This is used to invert the samples in the image. 0% is default and represents the original image. 100% will make the image completely inverted. Negative values are not allowed.

- `opacity(%)` - This is used to set the opacity level for the image. This is similar to the `opacity` property. 0% is completely transparent. 100% is default and represents the original image. Negative values are not allowed.
- `saturate(%)` - This is used to saturate the image. 0% will make the image completely un-saturated. 100% is default and represents the original image. Values over 100% provides higher-saturated results. Negative values are not allowed.
- `sepia(%)` - This is used to convert the image to sepia. 0% is default and represents the original image. 100% will make the image completely sepia. Negative values are not allowed.
- `url()` - The `url()` function takes the location of an XML file that specifies an SVG filter, and may include an anchor to a specific filter element.
filter: `url(svg-url#element-id)`

Text-decoration

This property is used to specify the decoration added to text.

- text-decoration-line
- text-decoration-style
- text-decoration-color
- text-decoration

text-decoration-line

This property is used to specify what type of line the decoration will have. We can set this property to *none*, *underline*, *overline*, *line-through*

We can combine more than one value, like underline and overline to display lines both under and over the text.

Ex:-

```
h1 { text-decoration-line: underline;}
```

```
h1 { text-decoration-line: underline overline;}
```

text-decoration-style

This property is used to specify how the line will display.

We can set this property to *solid*, *double*, *dotted*, *dashed*, *wavy*.

Ex:-

```
h1 {      text-decoration-line: underline;
          text-decoration-style: dotted;      }
```

text-decoration-color

This property is used to specify the color of the text-decoration-line.

Ex:-

```
h1 {      text-decoration-line: underline;
          text-decoration-color: red;      }
```

text-decoration

This is shorthand of all three text-decoration properties.

Syntax: -

Selector {text-decoration: text-decoration-line text-decoration-style text-decoration-color}

Ex:-

```
h1 { text-decoration: underline dotted red ; }
```

Transition

Transitions is used to change property values from one value to another, over a given duration. The transition effect will start when the specified CSS property changes value.

How to Use Transitions?

- CSS property you want to add an effect to
- Duration of the effect

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

Transition

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

transition-property

This property is used to specify the name of the CSS property the transition effect is for (the transition effect will start when the specified CSS property changes). Always specify the transition-duration property, otherwise the duration is 0, and the transition will have no effect.

We can set this property to *none*, *all* and *property*.

Ex: -

```
div { transition-property: all; }
```

none - No property will get a transition effect

all - All properties will get a transition effect. This is Default value.

property - Defines a comma separated list of CSS property names the transition effect is for

transition-duration

This property is used to specify how many seconds or milliseconds a transition effect takes to complete. We can set this property to time in the form of *seconds* or *milliseconds*. The default value of this property is 0s which means there will be no effect.

Ex:-

```
div { transition-duration: 2s; }
```

```
div { transition-duration: 1000ms; }
```

transition-timing-function

This property is used to specify the speed curve of the transition effect. This property allows a transition effect to change speed over its duration. We can set this property to *ease*, *linear*, *ease-in*, *ease-out*, *ease-in-out*, *step-start*, *step-end*, *steps(int, start or end)*, *cubic-bezier(n, n, n, n)*.

Ex:-

```
div { transition-timing-function: ease-in; }
```

- ease – It Specifies a transition effect with a slow start, then fast, then end slowly (equivalent to cubic-bezier(0.25, 0.1, 0.25, 1)). This is Default value.
- linear – It specifies a transition effect with the same speed from start to end (equivalent to cubic-bezier(0, 0, 1, 1))
- ease-in – It specifies a transition effect with a slow start (equivalent to cubic-bezier(0.42, 0, 1, 1))
- ease-out – It specifies a transition effect with a slow end (equivalent to cubic-bezier(0, 0, 0.58, 1))
- ease-in-out – It specifies a transition effect with a slow start and end (equivalent to cubic-bezier(0.42, 0, 0.58, 1))

- step-start - Equivalent to steps(1, start)
- step-end - Equivalent to steps(1, end)
- steps(int, start or end) - Specifies a stepping function, with two parameters. The first parameter specifies the number of intervals in the function. It must be a positive integer (greater than 0). The second parameter, which is optional, is either the value "start" or "end", and specifies the point at which the change of values occur within the interval. If the second parameter is omitted, it is given the value "end"
- cubic-bezier(n,n,n,n) - Define your own values in the cubic-bezier function. Possible values are numeric values from 0 to 1

transition-delay

This property is used to specify when the transition effect will start. We can set this property to time in the form of *seconds* or *milliseconds*.

Ex:-

```
div { transition-delay: 2s; }
```

```
div { transition-delay: 1000ms; }
```

transition

The transition property is a shorthand property for the four transition properties.

Syntax: -

Selector { transition: transition-property, transition-duration, transition-timing-function, and transition-delay; }

Ex:-

```
div { transition: width 2000ms ease; }
```

Note: Always specify the transition-duration property, otherwise the duration is 0s, and the transition will have no effect.

Width

This property is used to set the width of an element. We can set this property to auto and length in the form of px, cm, % etc.

Ex:-

```
div { width: 100px; }
```

Height

This property is used to set the height of an element. We can set this property to auto and length in the form of px, cm, % etc.

Ex: -

```
div { height: 100px; }
```


Outline

An outline is a line that is drawn around elements, outside the borders. Outlines never take up space, as they are drawn outside of an element's content.

- `outline-color`
- `outline-style`
- `outline-width`
- `outline`

outline-style

This property is used to specify the style of an outline. We can set this property to *none* (default), *hidden*, *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset*, *outset*.

Ex: -

```
div { outline-style: solid;}
```

outline-color

This property is used to specify the color of an outline. We can set this property to invert (default) or color.

Ex: -

```
div { outline-style: solid; outline-color: red; }
```

outline-width

This property is used to specify width of an outline. We can set this property to *medium* (default), *thin*, *thick*, *length*.

Ex: -

```
div { outline-style: solid; outline-width: thin; }
```

```
div { outline-style: solid; outline-width: 5px; }
```

outline

This is shorthand of other outline properties.

Syntax: -

Selector {outline: outline-color outline-style outline-width;}

Ex: -

```
div { outline: red solid 3px;}
```

outline-offset

This property is used to add space between an outline and the edge or border of an element.

Ex: -

```
div { outline-offset: 5px;}
```

Columns

This property is used to define columns.

- column-count
- column-width
- columns

column-count

This property is used to specify the number of columns an element should be divided into. We can set this property to *number* or *auto* (default).

Ex:-

```
div { column-count: 2;}
```


column-width

This property is used to specify width for the columns. We can set this property to *length* or *auto* (default).

Ex:-

```
div { column-count: 2;  
        column-width: 100px;}
```

```
div { column-width: 200px;}
```

columns

This is shorthand which is used to set column-width and column-count.

Syntax: -

Selector { columns: column-width column-count; }

Ex:-

```
div { columns: 200px 2; }
```

column-rule

- column-rule-style
- column-rule-width
- column-rule-color
- column-rule

column-rule-style

This property is used to specify the style of the rule between columns. We can set this property to none (default), hidden, dotted, dashed, solid, double, groove, ridge, inset and outset.

Ex: -

```
div { column-rule-style: double;}
```

column-rule-width

This property is used to specify width of the rule between columns. We can set this property to medium (default), thin, thick and length.

Ex: -

```
div { column-rule-width: 5px;}
```

column-rule-color

This property is used to specify color of the rule between columns. We can set this property to color value.

Ex: -

```
div { column-rule-color: red; }
```

column-rule

This is shorthand of other column-rule-* properties.

Syntax: -

Selector {column-rule: column-rule-width column-rule-style column-rule-color }

Ex: -

```
div { column-rule: 5px solid red;}
```

column-gap

This property is used to specify the gap between the columns. We can set this property to normal (default) and length.

Ex: -

```
div { column-gap: 30px;}
```


column-span

This property is used to specify how many columns an element should span across. We can set this property to none (default) and all.

Ex: -

```
div { column-span: all;}
```

column-fill

This property is used to specify how to fill columns, balanced or not. We can set this property to balance (default) and auto.

Ex: -

```
div { column-fill: auto;}
```

Attribute Selector

We can set style using attribute or its value.

Set Style using attribute : -

Syntax: -

Selector [attribute] { CSS ;}

Ex: -

div [id] { color: red; }

Attribute Selector

Set style using attribute and its value: -

Syntax: -

Selector [attribute = “value”] { CSS }

Ex: -

div[id=“data”] { color: red; }

Attribute Selector

Set Style using an attribute value containing a specified word.

Syntax: -

[attribute ~= "value"] { CSS }

Ex:-

[class ~= "game"] { color: red; }

<div class="game">game</div>

<div class="supergame">supergame</div>

<div class="game-super">game-super</div>

<div class="gamesuper">gamesuper</div>

<div class="super-game">super-game</div>

<div class="game super">game super</div>

<div class="super game">super game</div>

Attribute Selector

Set Style using with the specified attribute starting with the specified value.

Syntax: -

[attribute |= “value”] { CSS }

Ex:-

[class |= “game”] { color: red; }

The value must be a whole word.

Ex: - “game” or “game-super”

```
<div class="game">game</div>
```

```
<div class="supergame">supergame</div>
```

```
<div class="game-super">game-super</div>
```

```
<div class="gamesuper">gamesuper</div>
```

```
<div class="super-game">super-game</div>
```

```
<div class="game super">game super</div>
```

```
<div class="super game">super game</div>
```

Attribute Selector

Set Style whose attribute value begins with a specified value.

Syntax: -

[attribute ^= “value”] { CSS }

Ex:-

[class ^= “game”] {color: red;}

```
<div class="game">game</div>
```

```
<div class="supergame">supergame</div>
```

```
<div class="game-super">game-super</div>
```

```
<div class="gamesuper">gamesuper</div>
```

```
<div class="super-game">super-game</div>
```

```
<div class="game super">game super</div>
```

```
<div class="super game">super game</div>
```

It is not necessary that the value should be a whole word.

Attribute Selector

Set Style whose attribute value ends with a specified value.

Syntax: -

[attribute \$= "value"] { CSS }

Ex:-

[class \$= "game"] {color: red;}

```
<div class="game">game</div>
```

```
<div class="supergame">supergame</div>
```

```
<div class="game-super">game-super</div>
```

```
<div class="gamesuper">gamesuper</div>
```

```
<div class="super-game">super-game</div>
```

```
<div class="game super">game super</div>
```

```
<div class="super game">super game</div>
```

It is not necessary that the value should be a whole word.

Attribute Selector

Set Style whose attribute value contains a specified value.

Syntax: -

[attribute *= "value"] { CSS }

Ex:-

[class *= "ga"] {color: red;}

<div class="game">Game</div>

<div class="supergame">superGame</div>

<div class="game-super">game-super</div>

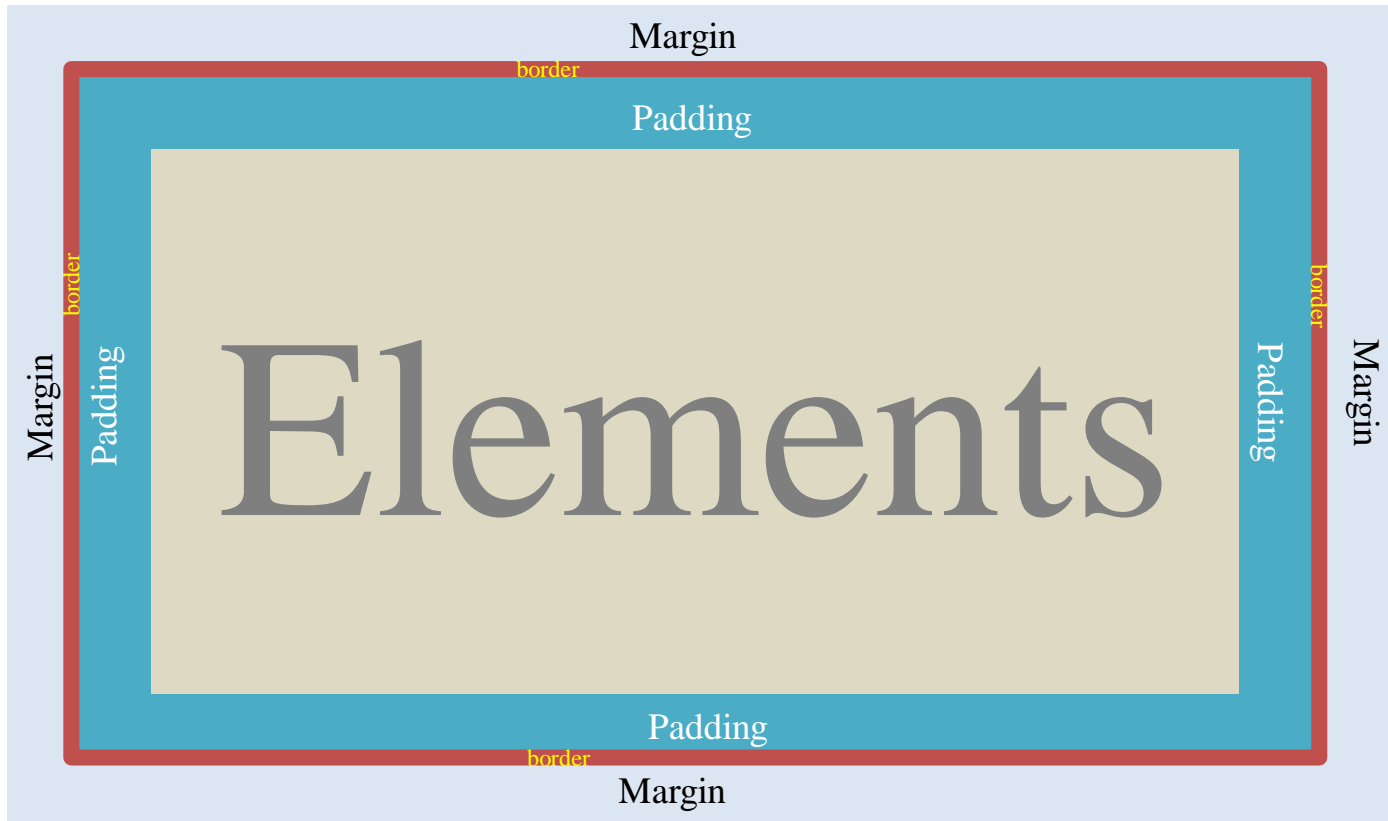
<div class="gamesuper">gamesuper</div>

<div class="super-game">super-Game</div>

<div class="game super">Game super</div>

<div class="super game">super Game</div>

<div class="super">super Game</div>



Width = left margin + left border + left padding + width + right padding + right border + right margin

Height = top margin + top border + top padding + height + bottom padding + bottom border + bottom margin

box-sizing

The box-sizing property tells the browser what the sizing properties should include. We can set this property to *content-box* (default) or *border-box*.

The box-sizing property is used to include the padding and border in an element's total width and height.

Ex:- `div { box-sizing: border-box; }`

content-box - The width and height properties (and min/max properties) includes only the content. Border, padding, or margin are not included.

border-box - The width and height properties (and min/max properties) includes content, padding and border, but not the margin.

box-sizing

The box-sizing property tells the browser what the sizing properties should include. We can set this property to *content-box* (default) or *border-box*.

The box-sizing property is used to include the padding and border in an element's total width and height.

Ex:- `div { box-sizing: border-box; }`

content-box - The width and height properties (and min/max properties) includes only the content. Border, padding, or margin are not included.

border-box - The width and height properties (and min/max properties) includes content, padding and border, but not the margin.

FlexBox or Flexible Box

- What is it ?
 - Its Layout mode a CSS 3 concept
- Why we need it ?
 - It helps to create responsive web pages

Flexbox Properties

- display
- flex-direction
- justify-content
- align-items
- flex-wrap
- align-content
- flex-flow
- order
- align-self
- flex

Display

This property is used to define how an element should display. Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

- inline
- block
- flex
- inline-block
- inline-flex
- inline-table
- list-item
- run-in
- table
- table-row-group
- table-caption
- table-column-group
- table-header-group
- table-footer-group
- table-cell
- table-column
- table-row
- none

Display

- inline – When we set this value, the element does not start on a new line and only takes up as much width as necessary (we can't set width/height it won't work)
- block – When we set this value, element always starts on a new line and takes up the full width available (we can set width/height)
- inline-block – It is combination of inline and block value. It doesn't start on new line but we can set width and height.
- none - The element will not be displayed at all (has no effect on layout)
- flex - Displays an element as a block-level flex container.
- inline-flex - Displays an element as an inline-level flex container.
- inline-table - The element is displayed as an inline-level table
- run-in - Displays an element as either block or inline, depending on context

Display

- table – It works like a <table> element
- table-caption - It works like a <caption> element
- table-column-group - It works like a <colgroup> element
- table-header-group - It works like a <thead> element
- table-footer-group - It works like a <tfoot> element
- table-row-group - It works like a <tbody> element
- table-cell - It works like a <td> element
- table-column - It works like a <col> element
- table-row - It works like a <tr> element
- list-item - It works like a element

flex-direction

This property is used to set the direction of flexible items. If the element is not a flexible item, the flex-direction property has no effect. We can set this property to *row* (default), *row-reverse*, *column*, *column-reverse*.

Ex: -

```
div { flex-direction: column-reverse; }
```

row - This is used to display the flexible items horizontally as row

row-reverse - This is used to display the flexible items same as row, but in reverse order

column - This is used to display the flexible items vertically as column

column-reverse - This is used to display the flexible items same as column, but in reverse order

justify-content

This property is used to align the flexible container's items when the items do not use all available space on the main-axis (horizontally). We can set this property to *flex-start* (default), *flex-end*, *center*, *space-between*, *space-around*.

Ex: -

```
div { justify-content: center; }
```

flex-start – This is used to position the item at the beginning of the container

flex-end – This is used to position the item at the end of the container

center - This is used to position the item at the center of the container

space-between - This is used to position the item with space between the lines

space-around - This is used to position the item with space before, between, and after the lines

align-items

This property is used to specify the default alignment for items inside the container. This property can be override using align-self property for each item. We can set this property to stretch (default), center, flex-start, flex-end, baseline.

Ex: -

```
div { align-items: center; }
```

stretch – This is used to stretch the item to fit the container

center – This is used to position the item at the center of the container

flex-start - This is used to position the item at the beginning of the container

flex-end - This is used to position the item at the end of the container

baseline - This is used to position the item at the baseline of the container

flex-wrap

This property is used to specify whether the flexible items should wrap or not. If the elements are not flexible items, the flex-wrap property has no effect. We can set this property to *nowrap* (default), *wrap* and *wrap-reverse*.

Ex: -

```
div { flex-wrap: wrap; }
```

nowrap – It specifies that the flexible items will not wrap

wrap – It specifies that the flexible items will wrap if necessary

wrap-reverse – It specifies that the flexible items will wrap, if necessary, in reverse order

align-content

This property is used to modify the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines. There must be multiple lines of items for this property to have any effect. We can set this property to *stretch* (default), *center*, *flex-start*, *flex-end*, *space-between*, *space-around*.

Ex: -

```
div { align-content: center; }
```

stretch - Lines stretch to take up the remaining space

center - Lines are packed toward the center of the flex container

flex-start - Lines are packed toward the start of the flex container

flex-end - Lines are packed toward the end of the flex container

space-between - Lines are evenly distributed in the flex container

space-around - Lines are evenly distributed in the flex container, with half-size spaces on either end

flex-flow

It is a shorthand property for the flex-direction and the flex-wrap properties. If the elements are not flexible items, the flex-flow property has no effect.

Syntax: -

Selector { flex-flow: flex-direction flex-wrap; }

Ex: -

div { flex-flow: row-reverse wrap; }

Flex-direction values:

row (default)

row-reverse

Column

column-reverse.

Flex-wrap values:

nowrap (default)

wrap

wrap-reverse

order

This property is used to specify the order of a flexible item relative to the rest of the flexible items inside the same container. If the element is not a flexible item, the order property has no effect. We can set this property to *number* (default 0).

Ex: -

```
div {order: 1}
```


align-self

This property is used to specify the alignment for the selected item inside the flexible container. The align-self property overrides the flexible container's align-items property. We can set this property to *auto* (default), *stretch*, *center*, *flex-start*, *flex-end*, *baseline*.

Ex: -

```
div {align-self: center;}
```

auto - The element inherits its parent container's align-items property, or "stretch" if it has no parent container

stretch – This is used to position the element to fit the container

center - This is used to position the element at the center of the container

flex-start - This is used to position the element at the beginning of the container

flex-end - This is used to position the element at the end of the container

baseline - This is used to position the element at the baseline of the container

flex-grow

This property is used to specify how much the item will grow relative to the rest of the flexible items inside the same container. If the element is not a flexible item, the flex-grow property has no effect. We can set this property to number (default 0).

Ex: -

```
div {flex-grow: 1;}
```

flex-shrink

This property is used to specify how the item will shrink relative to the rest of the flexible items inside the same container. If the element is not a flexible item, the flex-shrink property has no effect. We can set this property to number (default 1).

Ex: -

```
div {flex-shrink: 2;}
```

flex-basis

This property is used to specify the initial length of a flexible item. If the element is not a flexible item, the flex-basis property has no effect. We can set this property to *number* and *auto* (default).

Ex:-

```
div {flex-basis: 3;}
```

number - A length unit, or percentage, specifying the initial length of the flexible item(s)

auto - The length is equal to the length of the flexible item. If the item has no length specified, the length will be according to its content

flex

This property is used to specify the length of the item, relative to the rest of the flexible items inside the same container. If the element is not a flexible item, the flex property has no effect.

The flex property is a shorthand for the **flex-grow**, **flex-shrink**, and the **flex-basis** properties.

Syntax: -

Selector { flex: flex-grow flex-shrink flex-basis }

Ex: -

```
div { flex: 2 3; }
```

2D Transform

- transform
- transform-origin

transform

This property is used to apply a 2D or 3D transformation to an element. We can set below values: -

- `translate()`
- `rotate()`
- `scale()`
- `skew()`
- `matrix()`

Ex:-

```
div { transform: rotate(20deg); }
```

translate()

This method is used to move an element from its current position.

Syntax: -

transform: translateX(px)

transform: translateY(px)

transform: translate(Xpx, Ypx)

Ex: -

transform: translateX(10px)

transform: translateY(5px)

transform: translate(10px, 5px)

rotate()

This method is used to rotate an element clockwise or anti-clockwise according to a given degree.

Using negative values will rotate the element anti-clockwise.

Syntax: -

```
transform: rotate(ndeg)
```

Ex: -

```
transform: rotate(50deg)
```

```
transform: rotate(-40deg)
```

scale()

This method is used to increase or decrease the size of an element.

Syntax: -

transform: scaleX(width)

transform: scaleY(height)

transform: scale(width, height)

Ex: -

transform: scaleX(3)

transform: scaleY(4)

transform: scale(3, 4)

skew()

This method is used to skew an element along the X and Y-axis by the given angles.

Syntax: -

transform: skewX(deg)

transform: skewY(deg)

transform: skew(Xdeg, Ydeg)

Ex: -

transform: skewX(30deg)

transform: skewY(40deg)

transform: skew(30deg, 40deg)

transform: skew(30deg)

matrix()

This method is used to combines all the 2D transform methods into one.

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

The parameters are as follow:

`matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())`

Syntax: -

transform:

transform:

transform:

Ex: -

transform:

transform:

transform:

none

No transform

Ex: -

transform: none;

3D Transform

- transform
- transform-origin
- transform-style
- perspective
- perspective-origin
- backface-visibility

transform

This property is used to apply a 2D or 3D transformation to an element. We can set below values: -

- `translate()`
- `rotate()`
- `scale()`
- `skew()`
- `matrix()`
- `translate3d()`
- `rotate3d()`
- `scale3d()`
- `matrix3d()`
- `perspective()`

translate3d()

Syntax: -

transform: translateX(px)

transform: translateY(px)

transform: translateZ(px)

transform: translate3d(Xpx, Ypx, Zpx)

Ex: -

transform: translateX(10px)

transform: translateY(5px)

transform: translate3d(10px, 5px, 20px)

rotate3d()

Syntax: -

transform: rotateX(deg)

transform: rotateY(deg)

transform: rotateZ(deg)

transform: rotate3d(Xdeg, Ydeg, Zdeg, Angledeg)

Ex: -

transform: rotateX(50deg)

transform: rotateY(40deg)

transform: rotateZ(20deg)

transform: rotate3d(50deg, 50deg, 50deg, 60deg)

scale3d()

Syntax: -

transform: scaleX(x)

transform: scaleY(y)

transform: scaleZ(z)

transform: scale3d(x, y, z)

Ex: -

transform: scaleX(3)

transform: scaleY(4)

transform: scaleZ(5)

transform: scale3d(3, 4, 5)

matrix()

Syntax: -

transform: matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)

Ex: -

transform:

perspective()

Syntax: -

transform: perspective(n)

Ex: -

transform: perspective()

none

No transform

Ex: -

transform: none;

3D Transform Properties

- transform
- transform-origin
- transform-style
- perspective
- perspective-origin
- backface-visibility

transform-origin

This property is used to change the position of transformed elements. This property must be used together with the transform property. We can set this property to x, y and z axis.

2D transformations can change the x and y axis of an element.

3D transformations can change the x, y and z axis of an element.

Syntax: -

Selector { transform-origin: x-axis y-axis z-axis; }

Ex: -

```
div { transform-origin: 10% 20% 30% ; }
```

transform-origin

x-axis – This is used to define where the view is placed at the x-axis.

- left
- center
- right
- length (px, cm, em, %)

y-axis – This is used to define where the view is placed at the y-axis.

- top
- center
- bottom
- length (px, cm, em, %)

z-axis – This is used to define where the view is placed at the z-axis.

- length (px, cm, em, %)

transform-style

This property is used to specify how nested elements are rendered in 3D space. This property must be used together with the transform property. We can set this property to flat (default) and preserve-3d.

Ex: -

```
div { transform-style: preserve-3d ; }
```

Note - child elements will preserve its 3D position

perspective

This property is used to define how many pixels a 3D element is placed from the view. This property allows you to change the perspective on how 3D elements are viewed. This will only effect on 3D transformed elements.

Whenever we define the perspective property for an element, it is the child elements that get the perspective view, not the element itself.

We can set this property to *none* (default) and *length*

Ex: -

```
div { perspective: 50px ; }
```

Perspective-origin

This property is used to define where a 3D element is based in the x- and the y-axis. This property allows you to change the bottom position of 3D elements. This will only effect on 3D transformed elements. We can set this property to x (default 50%) and y axis (default 50%).

Whenever we define the perspective-origin property for an element, it is the child elements that are positioned, not the element itself.

Ex: -

```
div { perspective-origin: 20% 20% ; }
```

backface-visibility

This property is used to define whether or not an element should be visible when not facing the screen. This property is useful when an element is rotated, and you do not want to see its backside. We can set this property to *visible* (default) and *hidden*.

Ex: -

```
div { backface-visibility: hidden; }
```

Media Query

A media query consists of an optional media type and zero or more expressions that limit the style sheets' scope by using media features, such as width, height, and color.

Syntax: -

@media not|only **mediatype** and (expressions/ media features) { CSS; }

Ex: -

```
@media screen and (min-width: 480px) {  
  body {  
    background-color: lightgreen;  }  
}
```

```
@media (min-width: 480px) {  
  body {  
    background-color: lightgreen;  }  
}
```

all - Used for all media type devices

print - Used for printers

screen - Used for computer screens, tablets, smart-phones etc.

speech - Used for screenreaders that "reads" the page out loud

If you use the not or only operators, you must specify an explicit media type.

- And
 - @media (min-width: 800px) { CSS }
 - @media (min-width: 800px) and (orientation: landscape) { CSS }
 - @media screen and (min-width: 800px) and (orientation: landscape) { CSS }
- Or (Comma ,)
 - @media (min-width: 800px), (orientation: landscape) { CSS }
- Not
 - @media not all and (min-width: 800px) { CSS }
 - @media not (all and (min-width: 800px)) { CSS }
 - @media (not all) and (min-width: 800px) { CSS }
 - @media not screen and (color), print and (color) { CSS }
 - @media (not (screen and (color))), print and (color) { CSS }

Media Features

- any-hover - Does any available input mechanism allow the user to hover over elements? (added in Media Queries Level 4)
- any-pointer - Is any available input mechanism a pointing device, and if so, how accurate is it? (added in Media Queries Level 4)
- aspect-ratio - The ratio between the width and the height of the viewport
- color - The number of bits per color component for the output device
- color-index - The number of colors the device can display
- grid - Whether the device is a grid or bitmap
- height - The viewport height
- hover - Does the primary input mechanism allow the user to hover over elements? (added in Media Queries Level 4)
- inverted-colors - Is the browser or underlying OS inverting colors? (added in Media Queries Level 4)
- light-level - Current ambient light level (added in Media Queries Level 4)

Media Features

- max-aspect-ratio - The maximum ratio between the width and the height of the display area
- max-color - The maximum number of bits per color component for the output device
- max-color-index - The maximum number of colors the device can display
- max-device-aspect-ratio - The maximum ratio between the width and the height of the device
- max-device-height - The maximum height of the device, such as a computer screen
- max-device-width - The maximum width of the device, such as a computer screen
- max-height - The
- max-monochrome - The maximum number of bits per "color" on a monochrome (greyscale) device
- max-resolution - The maximum resolution of the device, using dpi or dpcmmaximum height of the display area, such as a browser window
- scan - The scanning process of the output device
- scripting - Is scripting (e.g. JavaScript) available? (added in Media Queries Level 4)

Media Features

- max-width - The maximum width of the display area, such as a browser window
- min-aspect-ratio - The minimum ratio between the width and the height of the display area
- min-color - The minimum number of bits per color component for the output device
- min-color-index - The minimum number of colors the device can display
- min-device-aspect-ratio - The minimum ratio between the width and the height of the device
- min-device-width - The minimum width of the device, such as a computer screen
- min-device-height - The minimum height of the device, such as a computer screen
- min-height - The minimum height of the display area, such as a browser window
- min-monochrome - The minimum number of bits per "color" on a monochrome (greyscale) device
- min-resolution - The minimum resolution of the device, using dpi or dpcm
- min-width - The minimum width of the display area, such as a browser window
- monochrome - The number of bits per "color" on a monochrome (greyscale) device

Media Features

- update-frequency - How quickly can the output device modify the appearance of the content (added in Media Queries Level 4)
- width - The viewport width
- orientation - The orientation of the viewport (landscape or portrait mode)
- overflow-block - How does the output device handle content that overflows the viewport along the block axis (added in Media Queries Level 4)
- overflow-inline - Can content that overflows the viewport along the inline axis be scrolled (added in Media Queries Level 4)
- pointer - Is the primary input mechanism a pointing device, and if so, how accurate is it? (added in Media Queries Level 4)
- resolution - The resolution of the output device, using dpi or dpcm