**Contents**

1. Css Introduction & Advantages

2. CSS Syntax

3. Style Sheet Types

4. Different Types of Selectors

5. Box Model

6. Box-Dimension

7. Back Ground Properties

8. Image Splicing

9. overflow

10.Float

11.Clear

12.Display

13.Visibility

14.Position

15. Z-Index

16. Opacity

17. List-Style

18. Length Units

19. Variables

20. Calc function ------>pending

21. import

1. CSS Introduction & Advantages: Css is design language intended to simplify process of making web pages presentable. Css handles look & feel part of webpages. using css, you can control text color, style of font, back ground images etc.

Advantages:-

* Css saves time:- You can write cascade style sheet once, then reuse same sheet for multiple HTML pages.
* Pages faster load:- If you are using css, you don’t need to write beautification attributes to all same caterogy tags. Just write one style of tag and apply it to all occurrence of same tags. so less code means faster down load time.
* Superior style to HTML:- Css has much wider array of attributes than HTML, so you can give far better look to your HTML page in comparision of HTML Elements.

**2.CSS Syntax**:-

The cascading style sheet syntax or rule consists of selector and declaration block.

Selector

{

Declaration-1;

Declaration-2;

Declaration-n;

}

2.1)Selector:- The selector points to html element in html code. The selector allows us to select html element & apply styles on that.

There are eight types of selectors.

1. Element Selector.

2. Id Selector.

3. Attribute Selector.

4.Class Selector.

5. Grouping Selector.

6. Combinator.

7.Pseudo classes.

8.Pseudo elements.

2.2)Declaration Block:- The block contains 0 or more declarations . The declarations are separated by (;).The declaration contains property name and value. The property name and value are separated by (:) .

The declaration syntax is as follows:

propertyName:value;

All declarations should be written inside open brace and closed brace.

**3.Style Sheet types**:- There are 3 types of style sheets.

1.1. External style sheet.

1.2. Internal style sheet.

1.3. Inline style sheet.

3.1.External Style sheet:- It can be written in any text editor . The file have only css rules but not html tags. The file must be saved with .css extension.

Each page must include link to external style sheet with <link> tag. The link tag must be inside <head> section.

Reusability can be increased by using External style sheet.

Example:

File name:One.css

P{background-color:red;}

Html filename:one.html

<head>

<link type=’css/text’ href=’one.css’ rel=’stylesheet’>

</head>

3.2.Internal Style sheet:- The internal style sheet should be used when single document has unique style. You define internal sheet in the head section of html page, inside<style>tag. The html code & styles are in same file. So that it is called internal style sheet.

Drawback:- The internal styles can’t be used for another html code.

Example:

<head>

<style>

H1{color:red;}

P{margin-left:20px;}

</style>

</head>

3.3.Inline-style:-To use in-line styles, Add style attribute to relevant HTML element. The style attribute can contain only declaration part of style sheet.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Table</title>

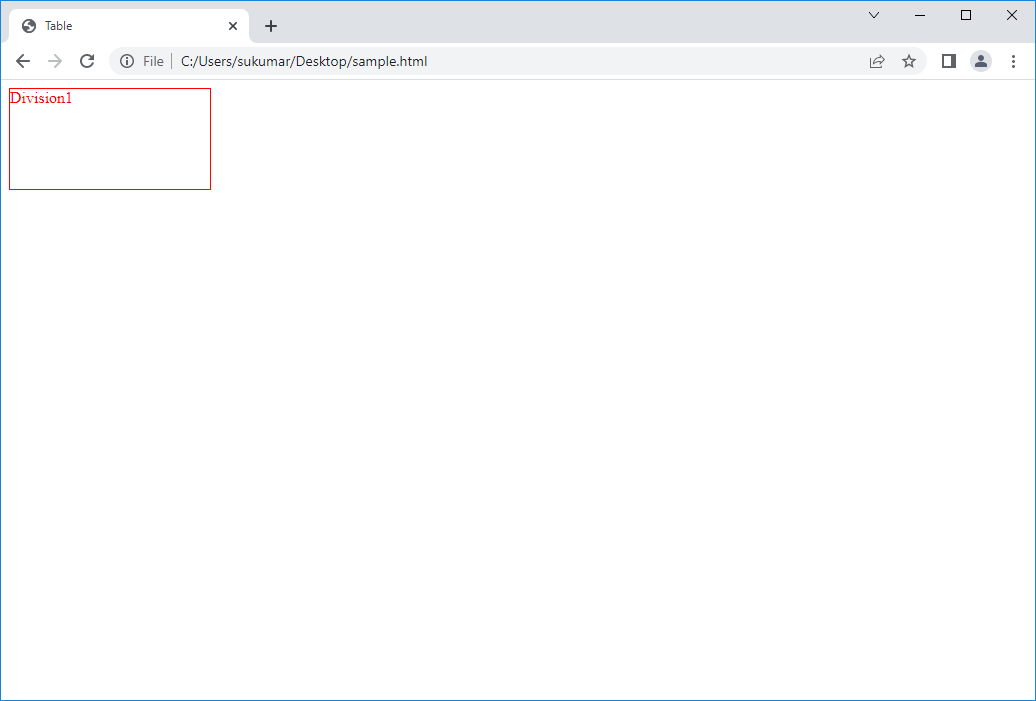
</head>

<body>

    <div style='height:100px;width:200px;border:1px solid'>Division1</div>

</body>

</html>



Drawbacks of Inline styles:-

1. Styles can’t be reused.
2. It is not used in real time.
3. Html code becomes cumbersome(confusing).

Note :- Prioriteis of style steets are

1. Inline style sheet. (highest priority)

2. Internal style sheet.

3. External Style sheet.

4. Browser default.

**4.Different Types of Selectors**:-

**4.1.Id Selector**:- It uses id attribute of tag to find specific element.

* To find element with specific id ,write hash character(#), followed by id of element.
* HTML code have 5 same kind Elements. We want to apply style to one of 5 elements when we should use id selector.

Example:

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Id Selector</title>

    <style>

        #one

        {

           color:red;

           border:1px solid green;

           height:100px;

           width:100px;

        }

        #two

        {

           color:blue;

           border:1px solid red;

           height:100px;

           width:100px;

        }

    </style>

</head>

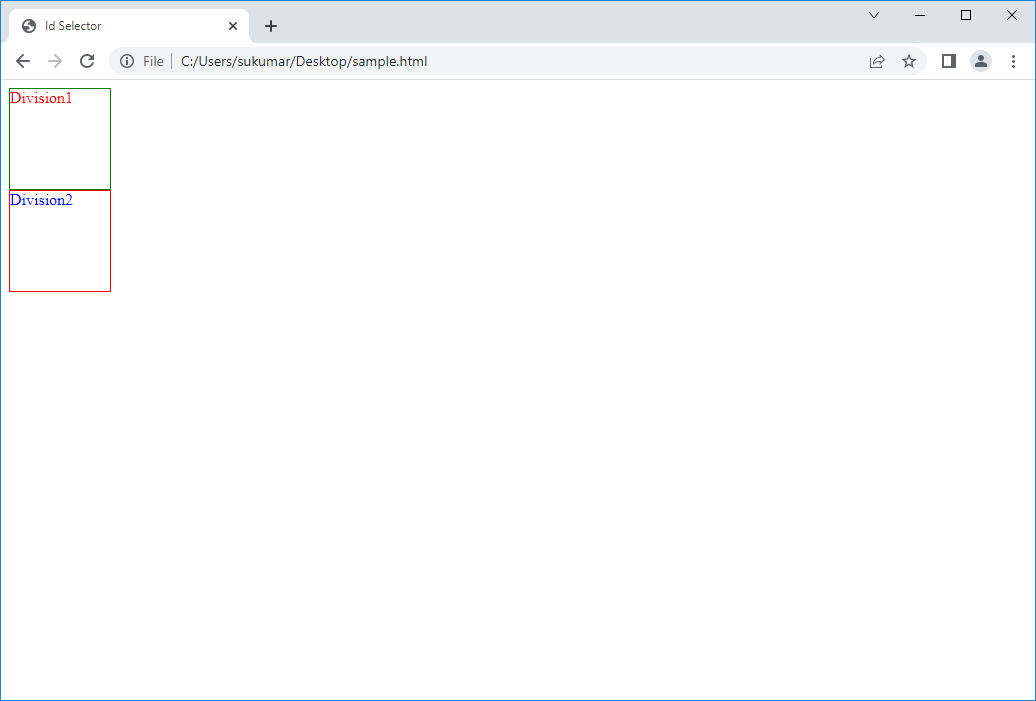
<body>

    <div id='one'>Division1</div>

    <div id='two'>Division2</div>

 </body>

</html>



4.2.Class Selector:-The class selector finds html elements with specific class.

* + To find element with specific class , write period character(.) followed by class name.
  + If HTML code has 10 related tags & we want to apply common style to few of 10 or to all, when we should use class selector.

Example:

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>class Selector</title>

    <style>

        .one

        {

           color:red;

           border:1px solid green;

           height:100px;

           width:100px;

        }

    </style>

</head>

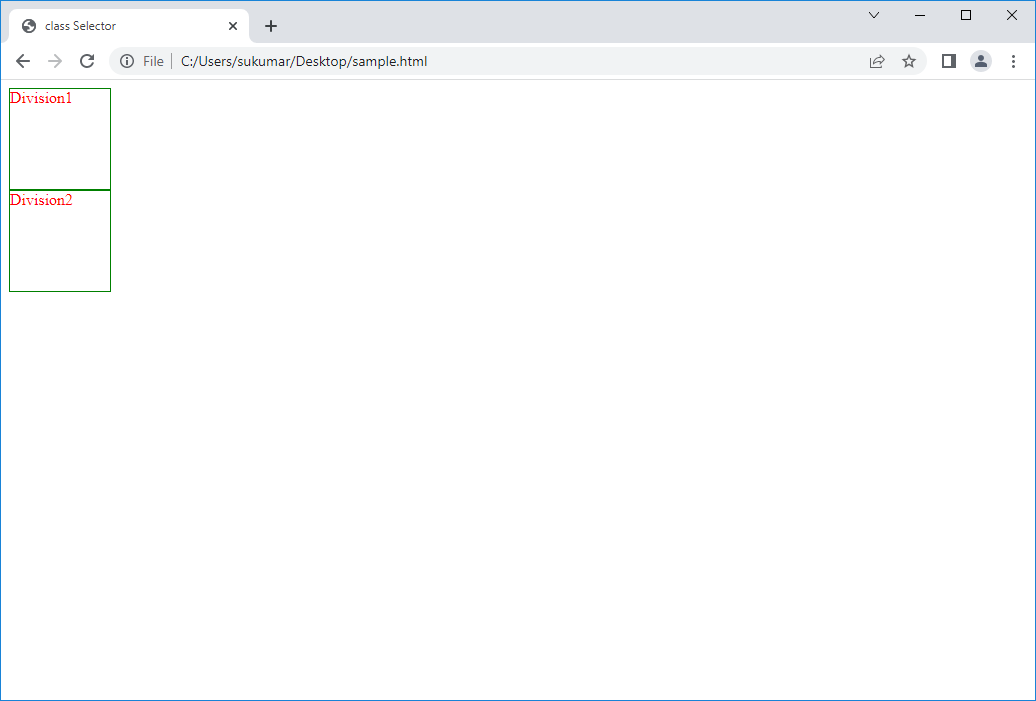
<body>

    <div class='one'>Division1</div>

    <div class='one'>Division2</div>

 </body>

</html>



4.3)Tag Selector/Element Selector*:-* It selects element based on element name.If html code have 5 same kind of tags. We want to apply same styles to all of them when we use tag selector.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Tag Selector</title>

    <style>

        div

        {

           color:red;

           border:1px solid green;

           height:100px;

           width:100px;

        }

     </style>

</head>

<body>

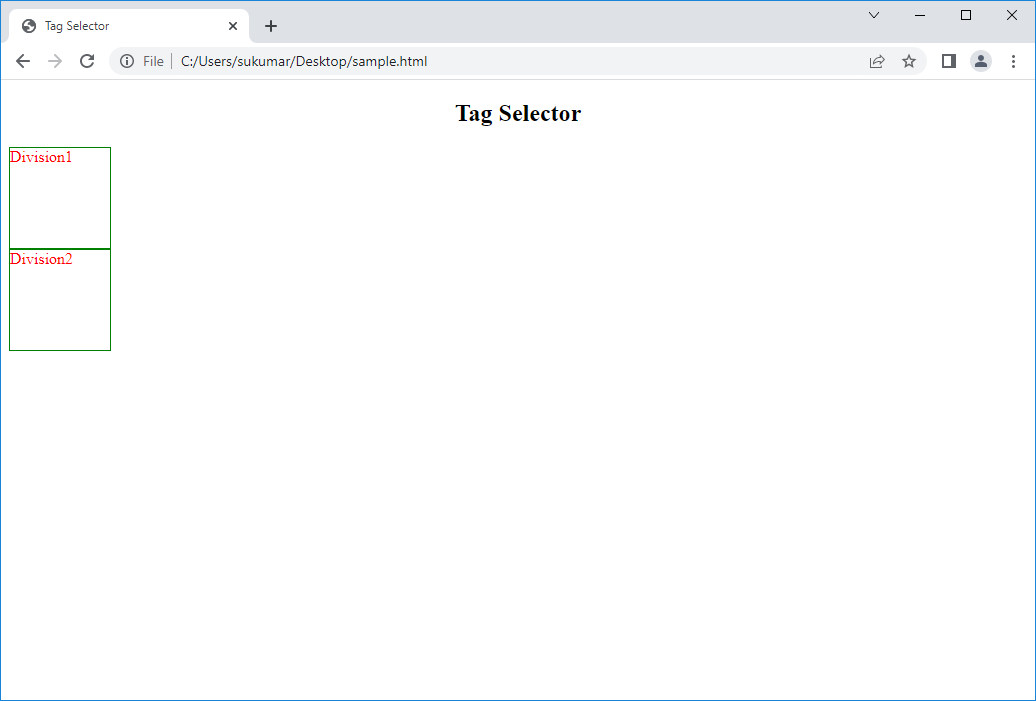
    <h2 align='center'>Tag Selector</h2>

    <div class='one'>Division1</div>

    <div class='one'>Division2</div>

 </body>

</html>



4.4. Priority chain of CSS: - As we have the four different ways to apply the CSS Properties for the HTML Element, if the same CSS Property been applied to a single element using all the four different ways with different values, by default Browser follows the below priority order,

• Among all the four different ways CSS been applied inline (through STYLE attribute) will always takes the higher priority in order.

• CSS been applied through ID, which takes the second priority order.

• CSS been applied through Class, which takes third priority in order.

• CSS been applied through TAG, which takes least priority in order.

Example:

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Selectors Priority</title>

    <style>

        div

        {

           color:red;

           border:1px solid green;

           height:100px;

           width:100px;

        }

        .one

        {

           color:orange;

           border:1px solid red;

           height:200px;

           width:200px;

        }

        #two

        {

           color:yellow;

           border:1px solid blue;

           height:300px;

           width:300px;

        }

  </style>

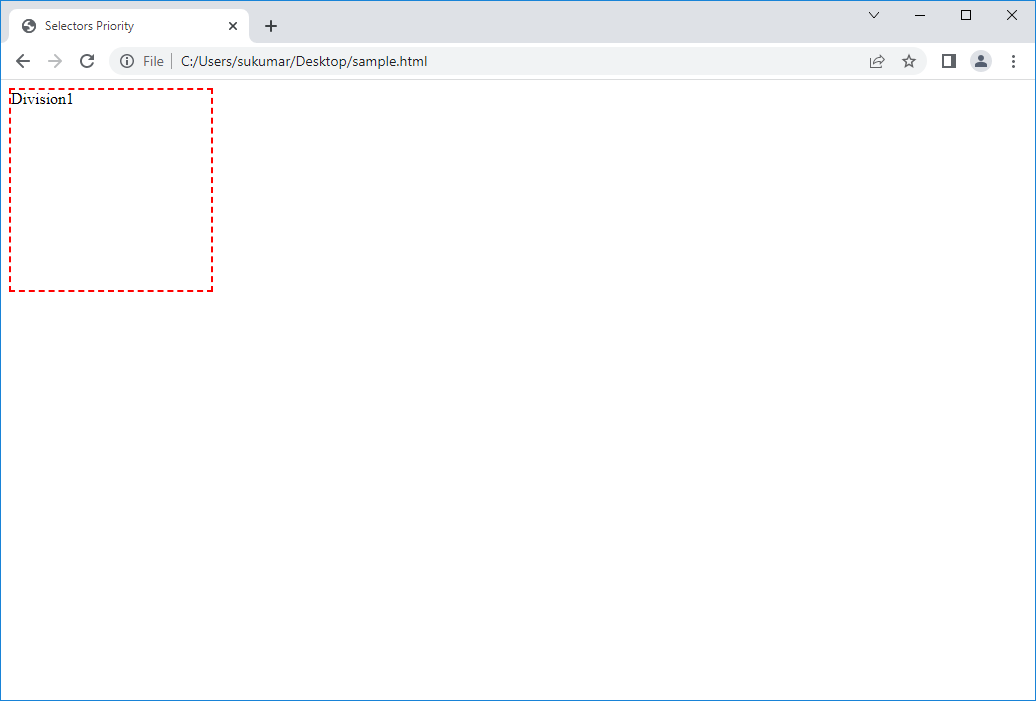
</head>

<body>

    <div class='one' id='two' style='color:black;border:2px dashed red;height:200px;width:200px'>Division1</div>

 </body>

</html>



Note:-Irrespective of the above priority order any CSS property being added with ‘! Important’ will always takes the Higher Priority.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Selectors Priority</title>

    <style>

        div

        {

           color:red !important;

           border:1px solid green !important;

           height:100px !important;

           width:100px !important;

        }

        .one

        {

           color:orange;

           border:1px solid red;

           height:200px;

           width:200px;

        }

        #two

        {

           color:yellow;

           border:1px solid blue;

           height:300px;

           width:300px;

        }

    </style>

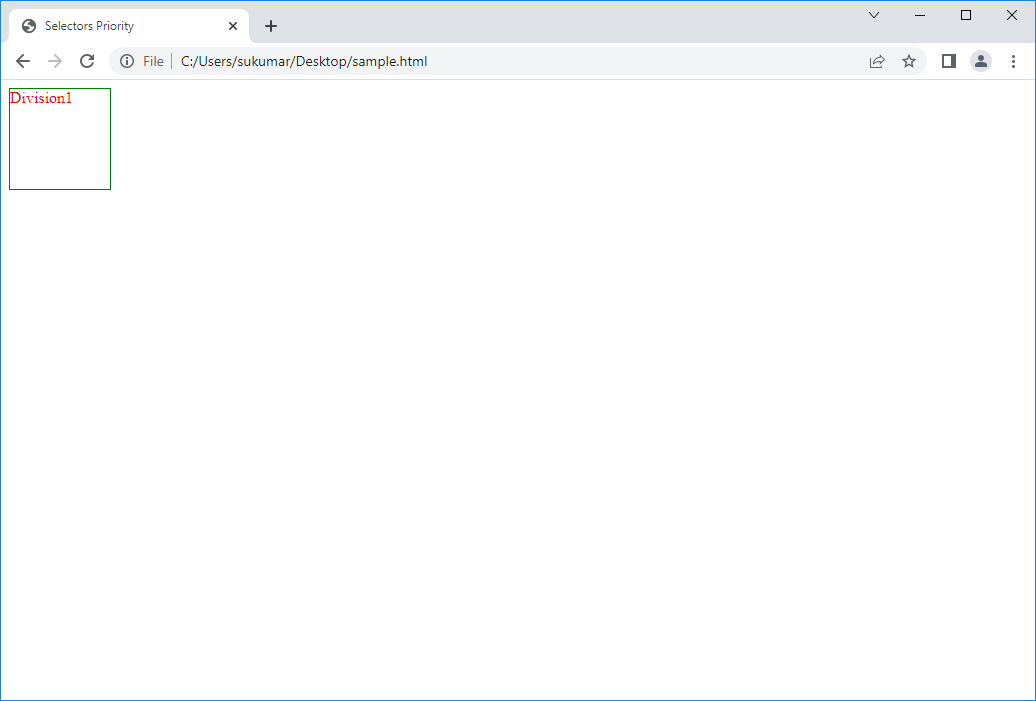
</head>

<body>

    <div class='one' id='two' style='color:black;border:2px dashed red;height:200px;width:200px'>Division1</div>

 </body>

</html>



4.5.Grouping Selector:-To avoid duplicated style sheets, we group the selectors. In group,separate each selector with comma.

Syntax:- selector1,selector2,…selector-n

{

Declatations;

}

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Grouping Selectors</title>

    <style>

        p,span

        {

            border:1px solid blue;

            color:tomato;

        }

    </style>

</head>

<body>

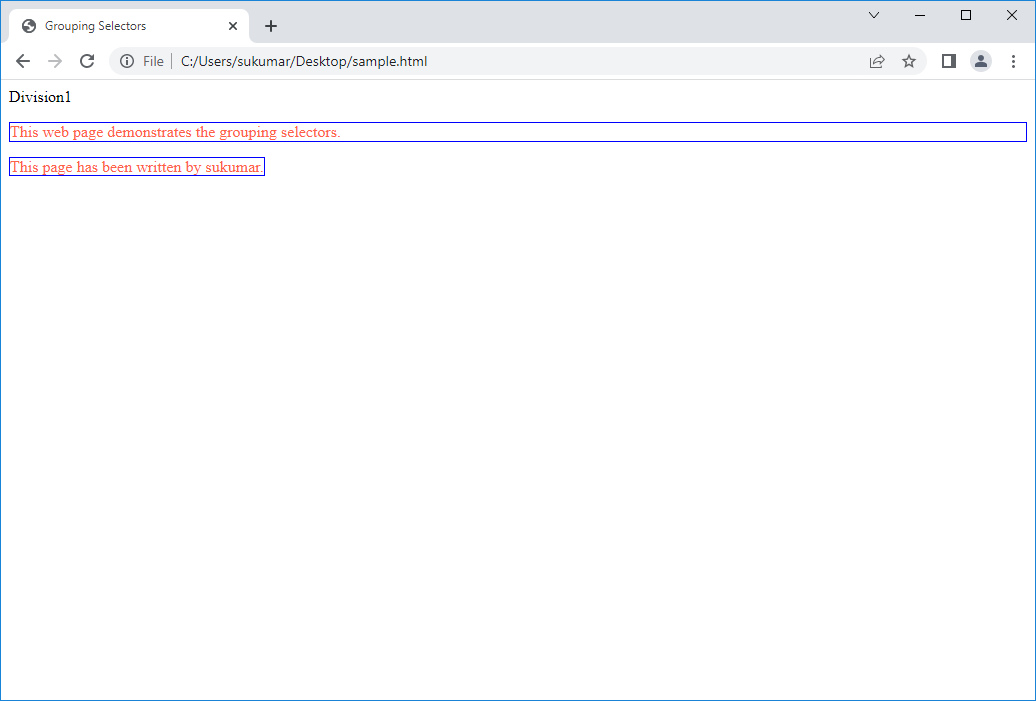
    <div>Division1</div>

    <p>This web page demonstrates the grouping selectors.</p>

    <span>This page has been written by sukumar.</span>

 </body>

</html>



4.6. Combinators:-The combinatory is special character that explains relationship between two selectors. We put the combinator between two selectors.

There are four combinators.

1. Descendent Selector(space):- The descendent selector matches all element that are descendents of specified element.

Syntax: - element-1 element-2

{

Declaration-1;

Declaration-2;

}

**b.**Child Selector(>):- The child selector selects all elements that are immediate children of specified element.

Syntax:- element1 > element2

{

Declarations;

}

**c.**Adjacent Sibling Selector:-

Sibling:- The sibling elements must have same parents and have brother or sister relationship.

Syntax:- element1 + element2

{

declarations;

}

It selects element2 that is **immediate adjacent sibling** of specified element1.

**d**. General sibling selector:- The selector selects all elements that are siblings of specified elements

that are siblings of specified element.

Syntax:-

Element1 ~ element2 {

css declarations;

}

4.7.Attribute Selector:-

4.7.1. [attributename] selector:- It is used to select elements which has specific attribute.

Syntax1: [attrname]{declarations}

Syntax2: tagname[attrname]{declarations}

4.7.2.[attributename=value]:- It is used to select elements with specified attribute name and value.

Syntax1: [attributename=value]{declatations}

Syntax2: tagname[attributename=value]{declarations}

4.7.3. [attribute ~=value] selector:- It is used to select elements with attribute value containing specified word.

Syntax1: [attrname ~=value]{declarations}

Syntax2: tagname[attrname~value]{declarations}

4.7.4.[attribute|=value] selector:- It is used to select element with specified attribute starting with specified value.

\*\* The value has to be whole word, either alone, like class=’top’ or followed by (-) like class=’top-text’.

Syntax: [attrname|=value]{declarations}

Syntax: tagname[attrname|=value]{declarations}

4.7.5.[attribute$=value] selector:- It is used to select elements whose attribute value ends with specified value.

\*\* The value has to be whole word, either alone, like class=’top’ or followed by (-) like class=’text-top’.

Syntax1:[attrname $= value] { declarations}

Syntax1: tagname[attrname $=value]{declarations}

4.8. pseudo-classes:- It is used to define the special state of element. New state means , element have new properties and element have new values to existing properties.

Example:- when u put cursor on element, the element leave old state and it enters the new state. When u take back cursor from the element, the element again comes into old state.

Syntax:

Selector:pseudo-class{

Property-1;

Property-2;

….etc.

}

4.8.1. Link Related Pseudo-classes:-

1. a:Link – It select the all unvisited links.
2. a:Visited- It selected all visited links.
3. a:active- It select all active links.

4.8.2.Form Related Pseudo-classes:-

1. Input:required:- select <input>element which have required attribute.
2. Input:optional:- select<input> elements which does not have required attribute.
3. Input:read-only: select<input>elements which have the read-only attribute.
4. Input:read-write:select<input>elements which does not have the read-only attribute.
5. Input:disabled: select<input> elements which is in disabled state.
6. Input:enabled: select<input> elements which is in enabled state.
7. Input:checked: select all checked <input> elements.
8. Input:valid: select all <input> elements with all valid values.
9. Input:invalid: it select all <input> elements with all invalid values.
10. Input:focus: it select <input> elements with focus.

4.8.3.Other Pseudo-classes:-

i.selctor:hover

ii.selector:first-child:

EX: p:first-child

It select every <p> that is first-child of every its parent.

iii. selector:last-child:

Ex: p:last-child

It select every <p> that is last-child of every its parent.

iv. selector:nth-child(n):

Ex: p:nth-child(2)

It select every <p> that is second-child of every its parent.

v. selector:only-child:

Ex: p:nth-child

It select every <p> that is only child of every its parent.

vi.selector:first-of-type:

Ex: p:first-of-type

It select every <p> that is first <p> element of its parent.

vii. slector:last-of-type;

Ex: p:last-of-type

It select every <p> that is last<p> element of its parent.

4.9.psedudo-elements:

Syntax:

selector :: selection{

Properties;

}

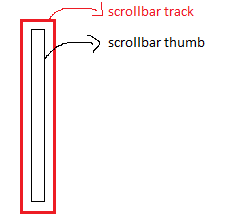
1.::selection:selector matches the portion of an element that is selected by a user.

2.

::-webkit-scrollbar

::-webkit-scrollbar-track

::-webkit-scrollbar-thumb



The above 3 pseudo-elements allows us to modify the look of the browser scrollbar.

3.::first-line: It is used to add special styles to first-line of text.

4. ::last-line: it is used to add special styles to last-line of text.

5. ::first-letter: It is used to add special styles to first-letter of text.

6. ::before: This element add a text before the content of element.

Syntax:

selector ::before{

content:’ ‘;

}

7. ::after: this element add a text after the content of element.

Syntax:

Selector:after{

Content:’ ‘;

}

The difference between psedudo-elements and pseudo-classes is that

The styles will be applied to complete element by pseudo-classes. We can apply style to part of complete element by pseudo-element.

**5.BOX MODEL:-**The The web browser renders every element as rectangular box.The Box model has following properties:

1.padding

2.border

3.margin

4.content.



1.content:- The content area consists of text,image or others. Its dimension are given by content box height and content box width.

2.Padding area:- The padding area is space between content & border line of box. There are four types of css properties which is related to paddings.

Those are

1.padding.

2.padding-top.

3.padding-bottom .

4.padding-left .

5.padding-right.

2.1.padding:- The four paddings can be specified in one property so it is called short-hand property.

Syntax:-

Padding:{ {length|percentage}|inheritance}

A)This property can have from 1 to 4 values.

Syntax:- padding: v1 v2 v3 v4.

our values sets top ,bottom,left & right padding respectively.

B)This property can have from 1 to 2 values.

Syntax:- padding: v1 v2

V1 refers to top & bottom paddings.

V2 refers to left & right paddings.

C)This property can have from 1 to 3 values.

Syntax:- padding: v1 v2 v3.

V1 refers to top padding.

V2 refers left & right padding.

V3 refers to bottom padding.

2.2.padding-top:- It sets padding on top side of element using value specified.

Syntax:

padding-top:value;

2.3)padding-bottom:- it sets padding on bottom side of element using value specified.

Syntax:

Padding-bottom:value;

2.4)padding-left:- It sets padding on left side of element using value specified.

Syntax:

Padding-left:value;

2.5)padding-right:- it sets padding on right side of element using value specified.

Syntax:

Padding-right:value;

3.Margin:-The margin is gap/space from border line of element. It is used to separate the neighbour elements.There are four types of css properties interms of margin.

Those are

1.margin

2.margin-left

3.margin-right

4.margin-bottom

5. margin-top.

3.1)The four margins can be specified in one property so it is called short-hand property.

Syntax:-

margin:{ {length|percentage}|inheritance|auto}

A) This property can have from 1 to 4 values.

Syntax:- margins: v1 v2 v3 v4.

Four values sets top ,bottom,left & right margins respectively.

B) This property can have from 1 to 2 values.

Syntax:- margins: v1 v2

V1 refers to top & bottom margins.

V2 refers to left & right margins.

C) This property can have from 1 to 3 values.

Syntax:- margins: v1 v2 v3.

V1 refers to top margins.

V2 refers left & right margins.

V3 refers to bottom margins.

3.2) margin -top:- It sets margins on top side of element using value specified.

Syntax:

Margin-top:value;

3.3) margin -bottom:- it sets margins bottom side of element using value specified.

Syntax:

Margin-bottom:value;

3.4) margin -left:- It sets margins on left side of element using value specified.

Syntax:

Margin-left:value;

3.5) margin -right:- it sets margins on right side of element using value specified.

Syntax:

Margin-right:value;

Note:- You can set the margin property to auto to horizontally center the element within its container. The element will then take up the specified width, and the remaining space will be split equally between the left and right margins.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title>

    <style>

        div

        {

            border:1px solid black;

            height:100px;

            width:500px;

        }

        p

        {

            margin:auto;

            border:2px solid red;

            width:200px;

        }

    </style>

</head>

<body>

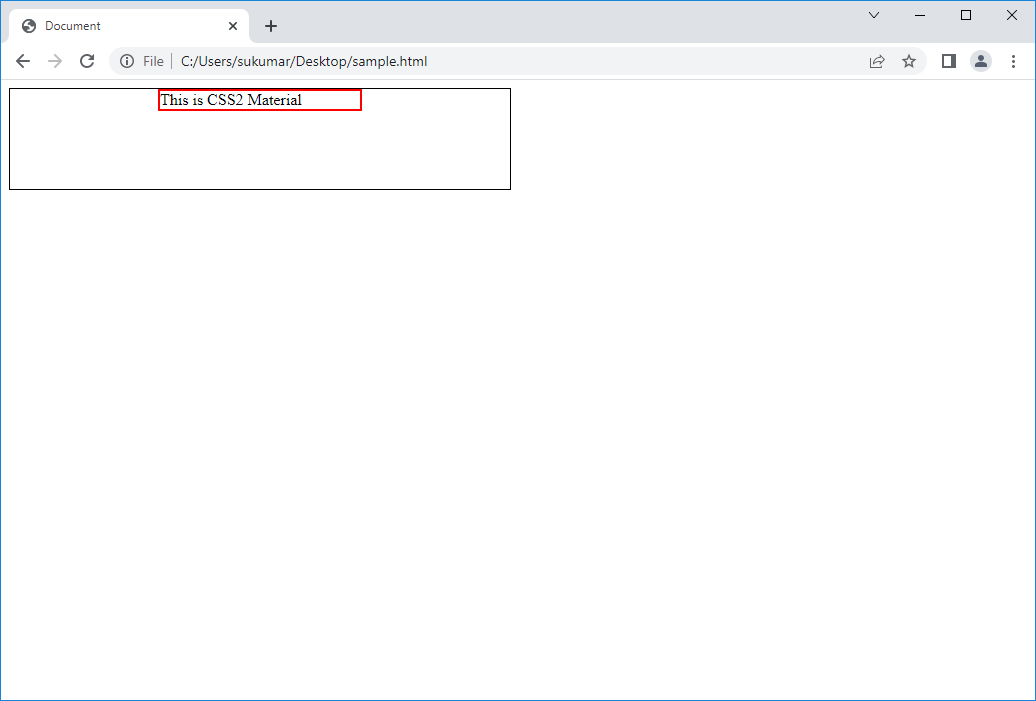
      <div>

          <p>This is CSS2 Material</p>

        </div>

</body>

</html>



4.Border:- The border is drawn around the element. There are several properties which is related to border of element.

4.1)Border:- Using this property, we can specify width, style and color of four border lines. So it is called short hand property.

Syntax:

border: v1 v2 v3;

V1 is width of border line.

V2 is style of border line.These styles have been divided into two types

2D styles which is dotted,dashed,solid,double.

3D styles which is inset,outset ,ridge,groove.

V3 is color of border line.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title>

    <style>

       .rock

       {

         height:40px;

         width:200px;

         border:1px solid green;

         margin:auto;

         margin-top:10px;

       }

       #one { border:1px solid green;}

       #two { border:1px dashed blue;}

       #three {border:1px dotted yellow;}

       #four { border:2px double green;}

       #five {border:1px groove orange;}

       #six {border:5px inset red;}

       #seven {border:5px outset blue;}

       #eight{ border:5px ridge cyan;}

    </style>

</head>

<body>

    <p class='rock' id='one'>solid Border</p>

    <p class='rock' id='two'>dashed Border</p>

    <p class='rock' id='three'>dotted Border</p>

    <p class='rock' id='four'>Double border</p>

    <p class='rock' id='five'>Groove Border</p>

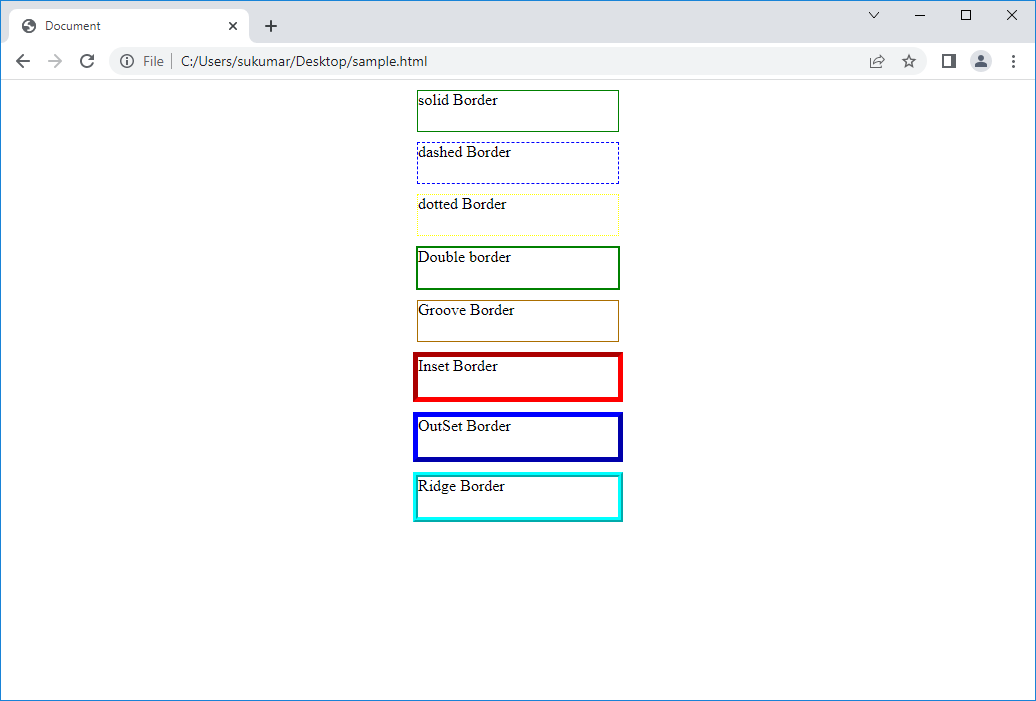
    <p class='rock' id='six'>Inset Border</p>

    <p class='rock' id='seven'>OutSet Border</p>

    <p class='rock' id='eight'>Ridge Border</p>

</body>

</html>



4.2)border-top: we use this to set top border of element.

Syntax:

border-top: width style color.

4.3)border-bottom:- we use this to set bottom border of element.

Syntax:

border-bottom: width style color.

4.4)border-left:- using this, we set left border of element.

4.5)border-right:- using this, we set right border of element.

4.6)border-color:- it is used to set color of border.

4.6.1) border color shortand property:-

Syntax1:- border-color:v1;

Syntax2:- border-color: v1 v2;

Syntax3:- border-color: v1 v2 v3;

4.6.2)We use following syntaxes to specify different colors for different sides.

a) border-top-color:value.

b)border-bottom-color :value.

c) border-left-color :value.

d) border-right-color:value.

4.7)border-width:-It is used to set width of border.

4.7.1) border width shorthand property:-

Syntax1:- border-width:v1;

Syntax2:- border-width:v1 v2;

Syntax3:- border-width:v1 v2 v3;

4.7.2) we use following syntaxes to specify different widths for different sides.

a) border-top-width

b)border-bottom-width

c) border-left-width

d) border-right-width.

4.8) border-style:- it is used to set style of border.

4.8.1) border style shorthand property:-

Syntax1:- border-style:v1;

Syntax2:- border-style:v1 v2;

Syntax3:- border-style:v1 v2 v3;

4.8.2) we use following syntaxes to specify different styles for different sides.

a) border-top-style

b)border-bottom-style

c) border-left-style

d)border-right-style.

5.outline:- An outline is line that is drawn around the border-line. The css out-line properties are

1. Outline 2)outline-color 3) outline-style 4)outline-width.

5.1)outline:- it is shorthand property.

Syntax:- outline:width style color.

5.2)outline-color:- it is used to set color of outline.

Syntax:- outline-color:colorvalue;

5.3)outline-style:- it is used to set style of outline.

Syntax:- outline-style:value;

5.4)outline-width:- it is used to set width of outline.

Syntax:- outline-width:value.

6)border-radius:-

Syntax:- border-radius: v1|v1 v2|v1 v2 v3|v1 v2 v3 v4;

If **one** value is set, this radius applies to**all 4 corners**.  
If**two**values are set, the**first**applies to top-left and bottom-right corner, the **second** appliesto top-right and bottom-left corner.  
**Four** values apply to the top-left, top-right, bottom-right, bottom-left corner inthatorder.  
**Three** values: The second value applies to top-right and also bottom-left.

**6.Box-Dimension:**-Usually, The box-height does not include the top&bottom-padding,top&bottom-border width and top&bottom-margin as well as The box-width does not include left&right-padding ,left&right-border width and left&right-margin.

Therefore height of box is equal to height of content box.

Width of box is equal to width of contnt box.

To change above rule , we can use **box-sizing property.**

1**. Box-size:-**

Syntax:- box-sizing:content-box|border-box|auto;

Content-box:- Height & width property does not include padding,borderwidth & margin.It is default value.

Border-box:- Height & width property includes padding,borderwidth & margin.

**2.height**:- It sets the height of content block.

Syntax: height:{percentage|length|auto}.

If the height of the containing block isn’t specified explicitly, and this element isn’t absolutely positioned, the percentage value is treated as auto.

NOTE:- If the content block requires more vertical space than height you assigned,**then that behaviour is controlled by overflow propery.**

**3.Width**:- It sets the width of content block.

Syntax: width:{percentage|length|auto}.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title>

    <style>

        .dv

        {

            background-color: red;

            border:10px solid green;

            margin:auto;

            margin-top:10px;

            height:50px;

            width:100px;

            padding:5px;

        }

        .dv1

        {

            box-sizing: content-box;

        }

        .dv2

        {

            box-sizing: border-box;

        }

    </style>

</head>

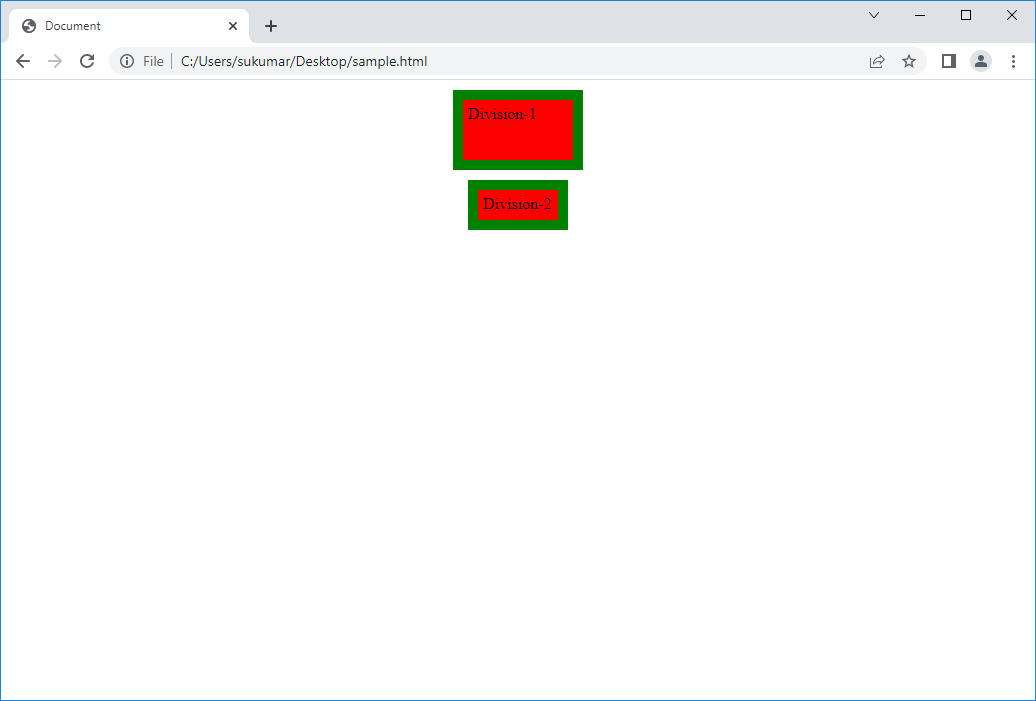
<body>

    <div class='dv dv1'>Division-1</div>

    <div class='dv dv2'>Division-2</div>

</body>

</html>



**7.BackGround Properties:-**The background is area of box which include contenxt box height & width,padding &borders.It does not includes margin.

1. Background-color:- This property sets background color of box.

Syntax:

background-color:{color|transparent|inherit}

The transparent is default value.

1. Background-image:- This property sets one or more images for element.

Syntax:

background-image:{url(‘’)|linear-gradient|radial-gradient|repeating-linear-gradient|repeating-radial-gradient|inherient|none}

The image start at top left corner of box. The image is repeated horizontally and vertically.

If back-ground image diamensions are smaller than its container dimension, then Back ground image is repeated in x-direction ,y-direction and in both.

1. Background-repeat:-

Syntax:

background-repeat:{no-repeat|repeat-x|repeat-y|repeat|inherit}

If background-position is not specified , then image is started at top left corner of the box.

1. Background-position:- it sets the starting position of background-image.

Syntax:- background-position:value;

|  |  |  |
| --- | --- | --- |
| **Value** | **Description** | **Play it** |
| left top left center left bottom right top right center right bottom center top center center center bottom | If you only specify one keyword, the other value will be "center" |  |
| *x% y%* | The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you only specify one value, the other value will be 50%. . Default value is: 0% 0% |  |
| *xpos ypos* | The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) or any other [CSS units](https://www.w3schools.com/cssref/css_units.asp). If you only specify one value, the other value will be 50%. You can mix % and positions |  |

**5.background-attachment:-**

It sets whether background-image is fixed or scrolls with rest of page.

Syntax:- background-attachment:value

1.scroll:- The background scrolls along with element. This is default.

2.fixed:- The background is fixed. It does not scroll along with element.

6.background-size:-This property sets size of background-image.

Syntax:-background-size:{auto|length|percentage|cover|contain|inherit}

6.1)auto:- it is default value. The background image contain its actual height width and height.

6.2)Length:-

Syntax: v1 v2

V1 specifies width of background-image.

V2 specifies height of background-image.

6.3)Percentage:-

Syntax:- w% h%;

W% specifies width of background-image in percent of parent width.

H% specifies height of background-image in percent of parent height.

6.4)Cover:scale background image to be as large as possible.Background area is completely covered by background image. Some parts of background image may not be in view with in background position area.

6.5)Contain:- It scales image as large as possible with out cropping.

7. background-origin:-This property sets origin position of background-image.

Syntax:- background-origin:padding box|border-box|content box|initial;

8.background-clip:- The background-clip property defines how far the background (color or image) should extend within an element.

Syntax:-background-clip:padding box|border-box|content box|initial;

Example**:**

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Document</title>

    <style>

        .dv

         {

            height:300px;

            width:200px;

            margin:auto;

            margin-bottom:5px;

            border:2px solid black;

            float:left;

         }

         #one

         {

            background-color: red;

            border:5px inset blue;

            margin:auto;

         }

        #two

        {

            background-image:url('file:///E:/Photos/DSC00162.JPG');

        }

        #three

        {

            background-image:url('file:///E:/Photos/MLEC%20SPEECH-1.jpg') ;

            background-size:50px 50px;

        }

        #four

        {

            background-image:url('file:///E:/Photos/MLEC%20SPEECH-1.jpg') ;

            background-size:50px 50px;

            background-repeat:repeat-x;

            clear:both;

            float:left;

        }

        #five

        {

            background-image:url('file:///E:/Photos/DSC00162.JPG');

            background-position: 250px 1970px;

        }

        #six

        {

            background-image:url('file:///E:/Photos/DSC00162.JPG');

            background-clip:border-box;

        }

    </style>

</head>

<body>

    <div id='one' class='dv'>

    </div>

    <div id='two' class='dv'>

    </div>

    <div id='three' class='dv'>

    </div>

    <div id='four' class='dv'>

    </div>

    <div id='five' class='dv'>

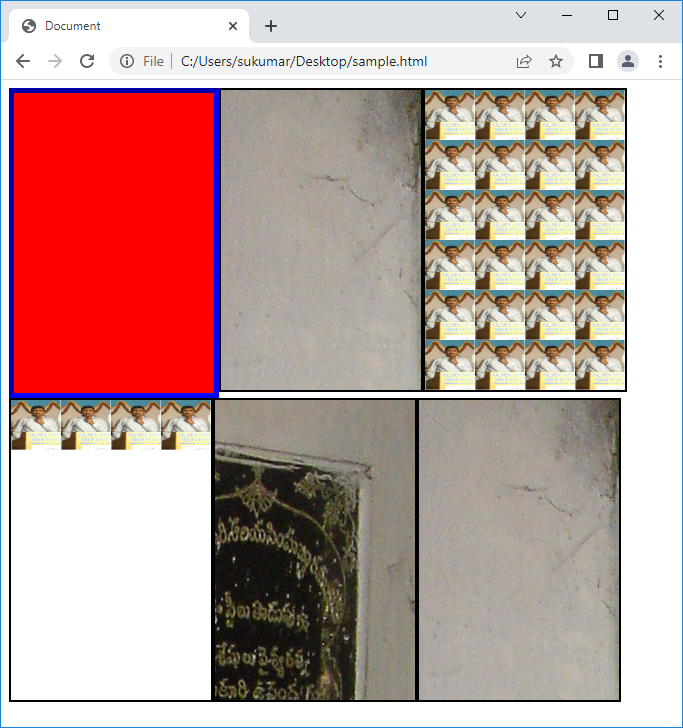
    </div>

    <div id='six' class='dv'>

    </div>

</body>

</html>

****

**8.Image Spriting:** The concept of merging all the static images to form a single image, through background position property, we show only required image is called as image spriting. Through image spriting we increase the performance of the page by loading all the static images in a single column.

**➢ Note: Image spriting is only recommended for static web pages but not for dynamic web pages.**

Example:

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Image Spriting</title>

    <style>

    .bi

    {

      height:200px;

      width:90%;

      border:1px solid red;

      margin-bottom:5px;

    }

    .b

    {

      height:100%;

      width:40%;

      box-sizing:border-box;

      border:1px solid green;

      float:left;

      margin-right:5px;

    }

    .b1

    {

      background-image:url('file:///C:/Users/sukumar/Desktop/image\_spriting.jpg');

      background-position:-6px -33px;

      background-repeat: no-repeat;

    }

    .b2

    {

      background-image:url('file:///C:/Users/sukumar/Desktop/image\_spriting.jpg');

      background-position:-4px -344px;

      background-repeat: no-repeat;

      background-origin:center center;

    }

    .pd

    {

    }

  </style>

</head>

<body>

   <div class='bi'>

         <div class='b b1'>

         </div>

         <div class='pd' >

            <p>Name:A.sukumar</p>

            <p>Age:41</p>

         </div>

   </div>

   <div class='bi'>

        <div class='b b2'>

        </div>

        <div class='pd'>

          <p>Name:A.veena</p>

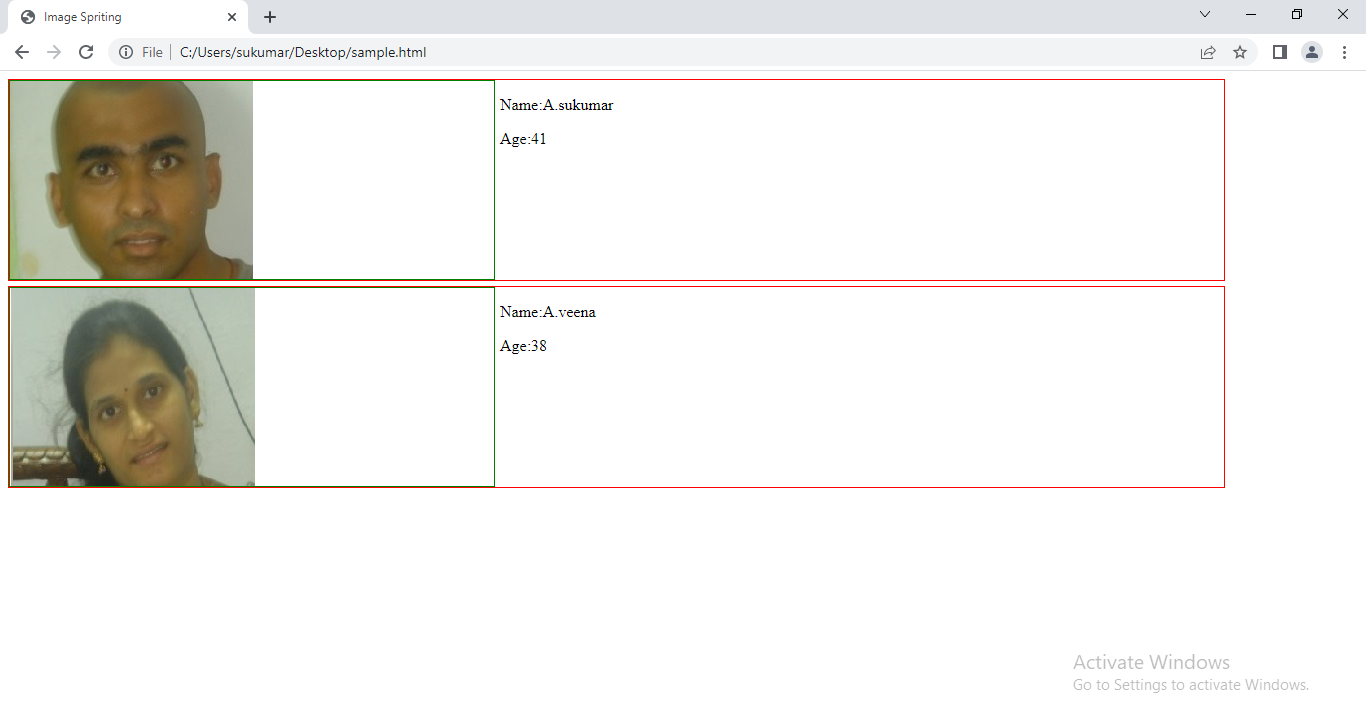
          <p>Age:38</p>

        </div>

   </div>

</body>

</html>



**9.OverFlow:-** The dimension of child elements are greater than dimension of parent element then child element is not completely inside the parent element and child element is not completely outside the parent element. To control it, we should use of overflow property.

Syntax:-

Overflow:visible|hidden|scroll|auto;

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

Note:-

1.If x-axis dimension of child element is only greater than parent element x-axis dimension then horizontal scroll bar is in active state and vertical scroll is in-active state.

2. If y-axis dimension of child element is only greater than parent element y-axis dimension then horizontal scroll bar is in in-active state and vertical scroll is in active state.

3. even dimensions(x&y) of child element is lesser than dimensions of parent element, two scroll bars will be in in-active state.

* auto - Similar to scroll, but it adds scrollbars only when necessary

Example:

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Grouping Selectors</title>

    <style>

         div

         {

             height:100px;

             width:100px;

             border:1px solid green;

             background-color: red;

             margin:50px;

          }

         #one

         {

             overflow: visible;

         }

         #two

         {

             overflow:hidden;

         }

         #three

         {

             overflow:scroll;

         }

         #four

         {

             overflow:auto;

         }

    </style>

</head>

<body>

    <div>

          <p id='one'>This Code demonstrates the overflow:visible</p>

    </div>

    <div>

          <p id='two'>This code demostrates the overflow:hidden</p>

    </div>

    <div>

          <p id='three'>This code demostrates the overflow:scroll</p>

    </div>

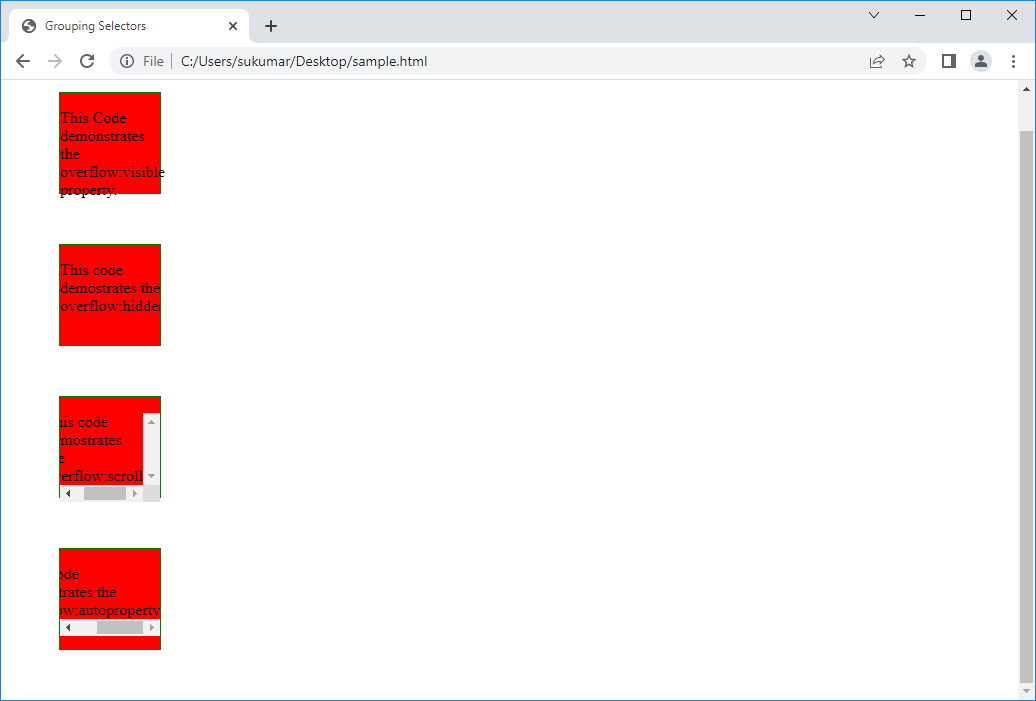
    <div>

          <p id='four'>This code demostrates the overflow:autoproperty</p>

    </div>

  </body>

</html>



**10.Float:-** Generally, The block level elements are rendered from top to bottom and Inline elements are rendered from left to right. To change this default rendering , we should use float property.

The browser change default rendering direction of multiple continuous DOM elements(next same siblings) to either left or right using float property Element. Those continuous elements will be rendering in same line irrelevant of block/inline element.

Note:- The Element which is next to float element is flown around to floating element.

Syntax: - float:left|right|none;

Left- The element must float on the left side of its containing block.

Right- The element must float on the right side of its containing block.

None-The element must not float.

Example:1

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Float property</title>

    <style>

       div

       {

           height:100px;

           width:200px;

           border:1px solid green;

           margin-bottom:5px;

           text-align:center;

       }

       #one

       {

           background-color: red;

           float:left;

       }

       #two

       {

           background-color:blue;

           float:left;

       }

       #three

       {

           background-color:tomato;

           float:left;

       }

    </style>

</head>

<body>

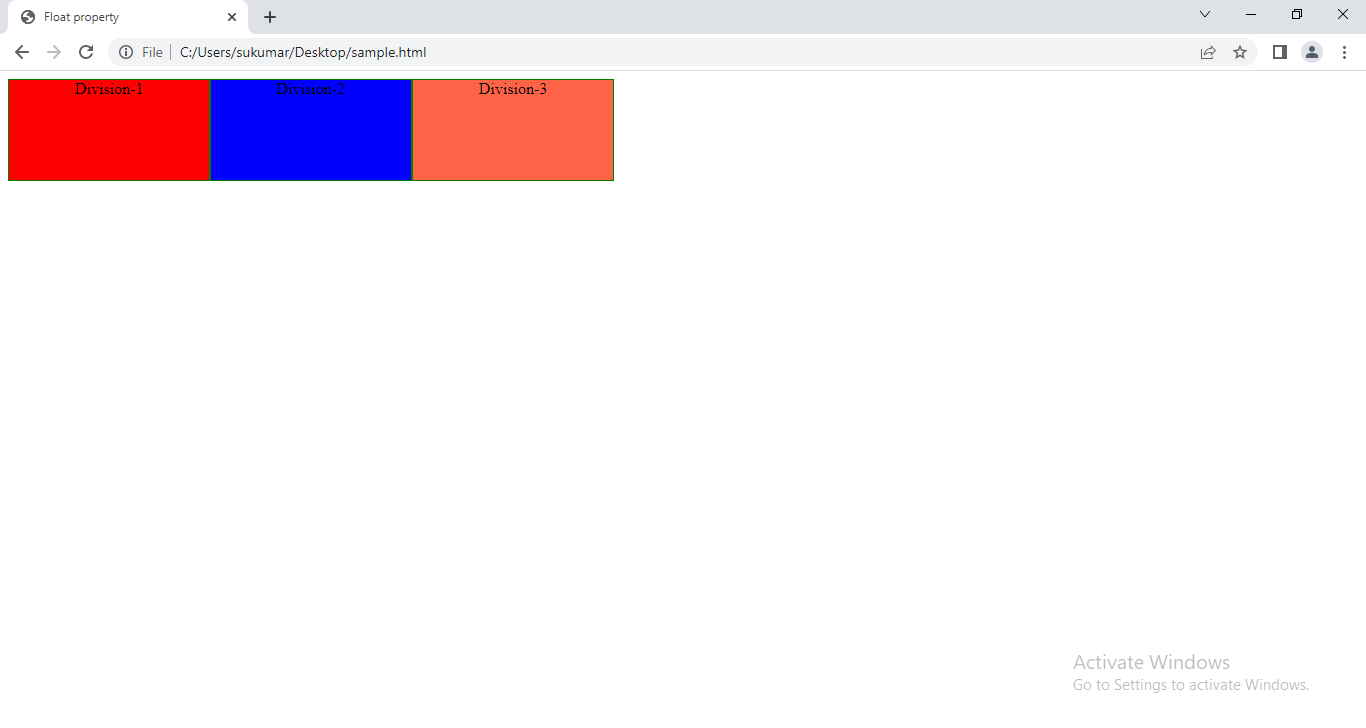
    <div id='one'> Division-1</div>

    <div id='two'> Division-2</div>

    <div id='three'> Division-3</div>

</body>

</html>



Example:2

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Float property</title>

    <style>

       div

       {

           height:100px;

           width:200px;

           border:1px solid green;

           margin-bottom:5px;

           text-align:center;

       }

       #one

       {

           background-color: red;

           float:right;

       }

       #two

       {

           background-color:blue;

           float:right;

        }

       #three

       {

           background-color:tomato;

           float:right;

        }

    </style>

</head>

<body>

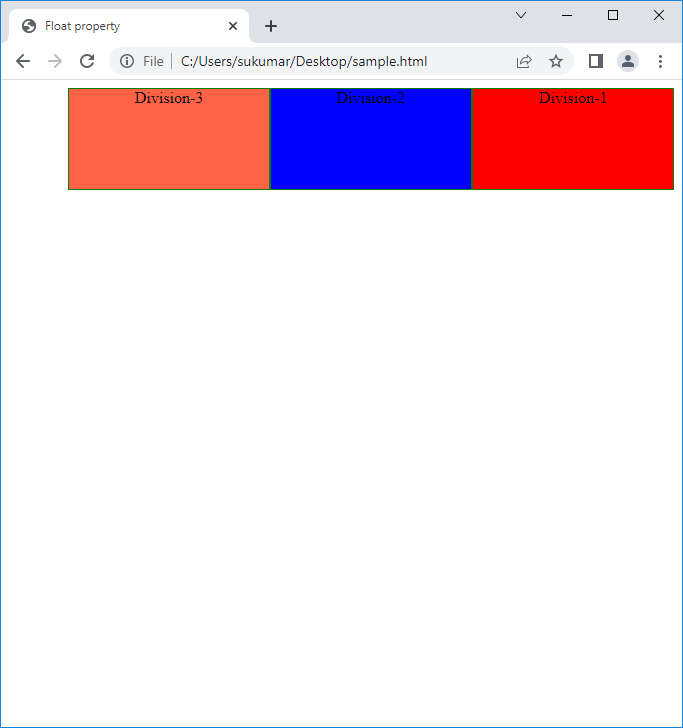
    <div id='one'> Division-1</div>

    <div id='two'> Division-2</div>

    <div id='three'> Division-3</div>

</body>

</html>



Example:3

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Float property</title>

    <style>

       div

       {

           height:100px;

           width:200px;

           border:1px solid green;

           margin-bottom:5px;

           text-align:center;

       }

       #one

       {

           background-color: red;

           float:left;

       }

       #two

       {

           background-color:blue;

           float:right;

        }

       #three

       {

           background-color:tomato;

           float:right;

        }

    </style>

</head>

<body>

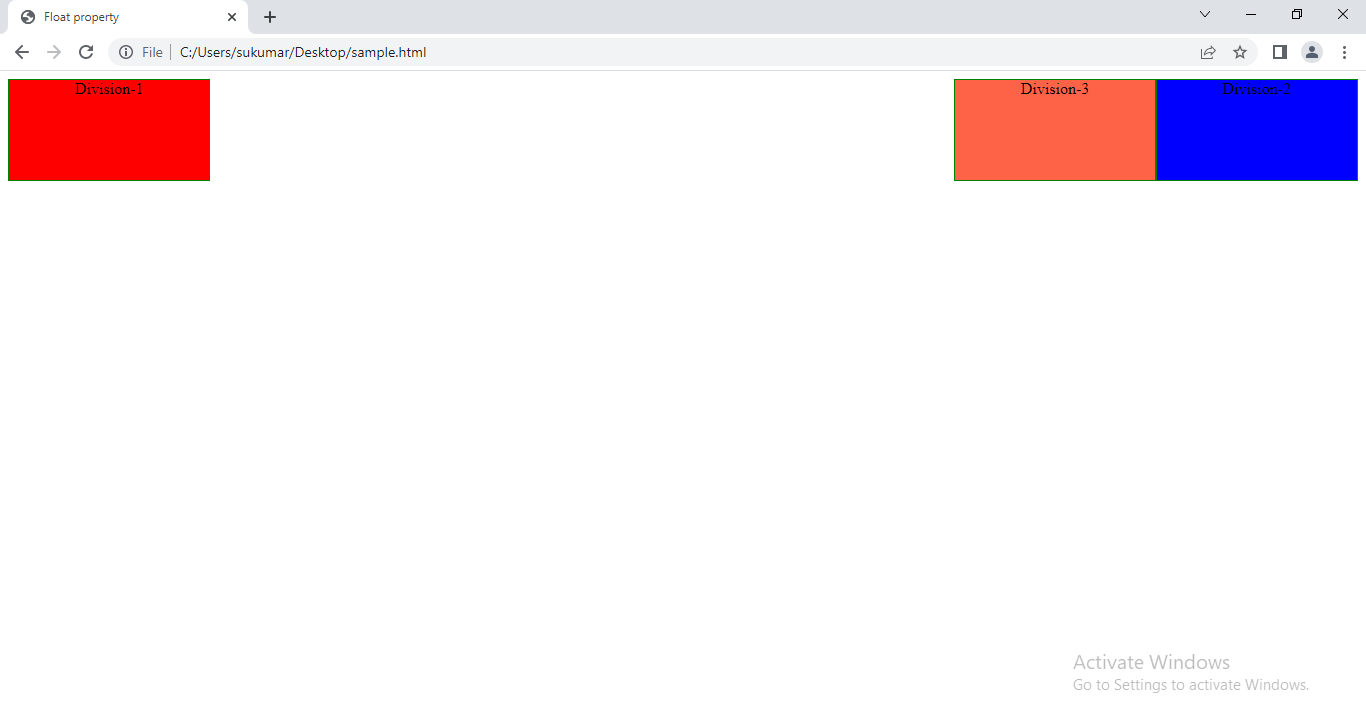
    <div id='one'> Division-1</div>

    <div id='two'> Division-2</div>

    <div id='three'> Division-3</div>

</body>

</html>



Example:4

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Float property</title>

    <style>

       div

       {

           height:100px;

           width:200px;

           border:1px solid green;

           margin-bottom:5px;

           text-align:center;

       }

       #one

       {

           background-color: red;

           float:left;

       }

       #two

       {

           background-color:blue;

           float:right;

        }

       #three

       {

           background-color:tomato;

           float:left;

        }

    </style>

</head>

<body>

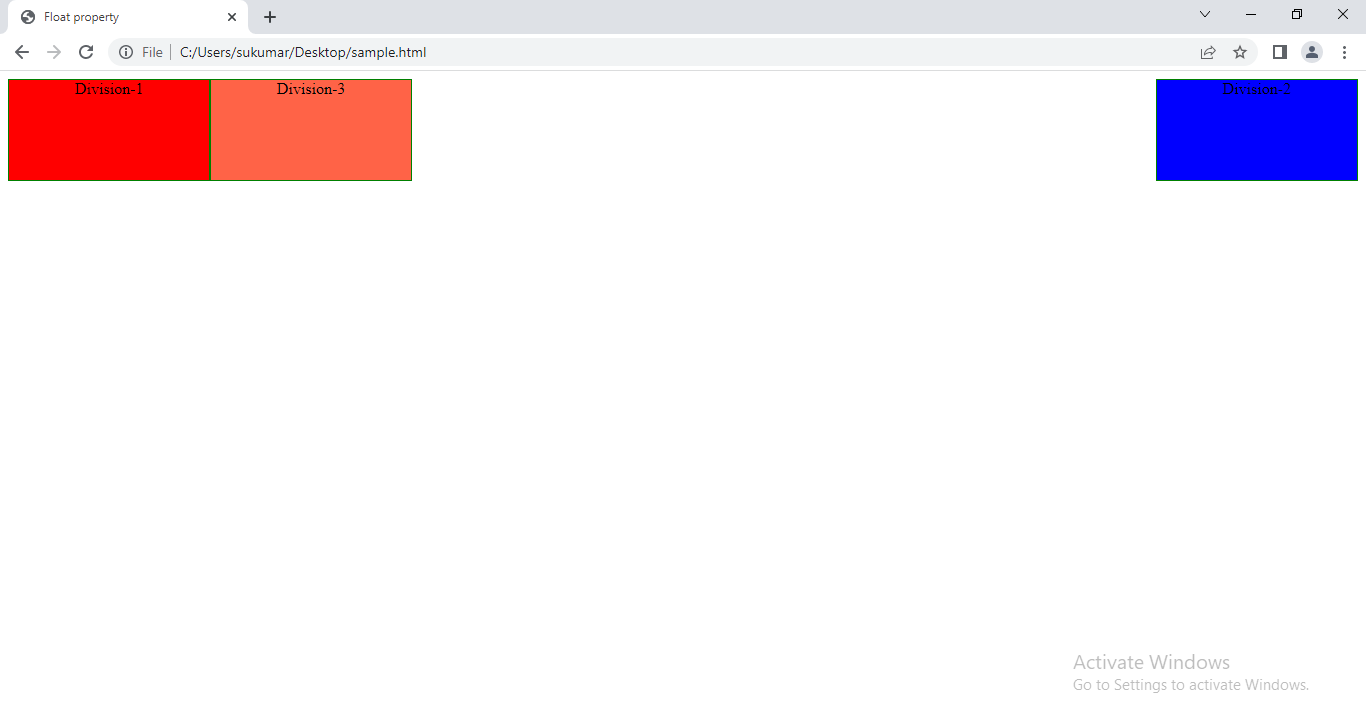
    <div id='one'> Division-1</div>

    <div id='two'> Division-2</div>

    <div id='three'> Division-3</div>

</body>

</html>



**11.Clear:-**The element which is next to floating element flows arounds the floating elemeing. By rendering like this, floating element over laps the next element . So next element does not appear completely or partially.

Example:

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Float property</title>

    <style>

       div

       {

           height:100px;

           width:200px;

           border:1px solid green;

           margin-bottom:5px;

           text-align:center;

       }

       #one

       {

           background-color: red;

           float:left;

       }

       #two

       {

           background-color:blue;

       }

       #three

       {

           background-color:tomato;

       }

    </style>

</head>

<body>

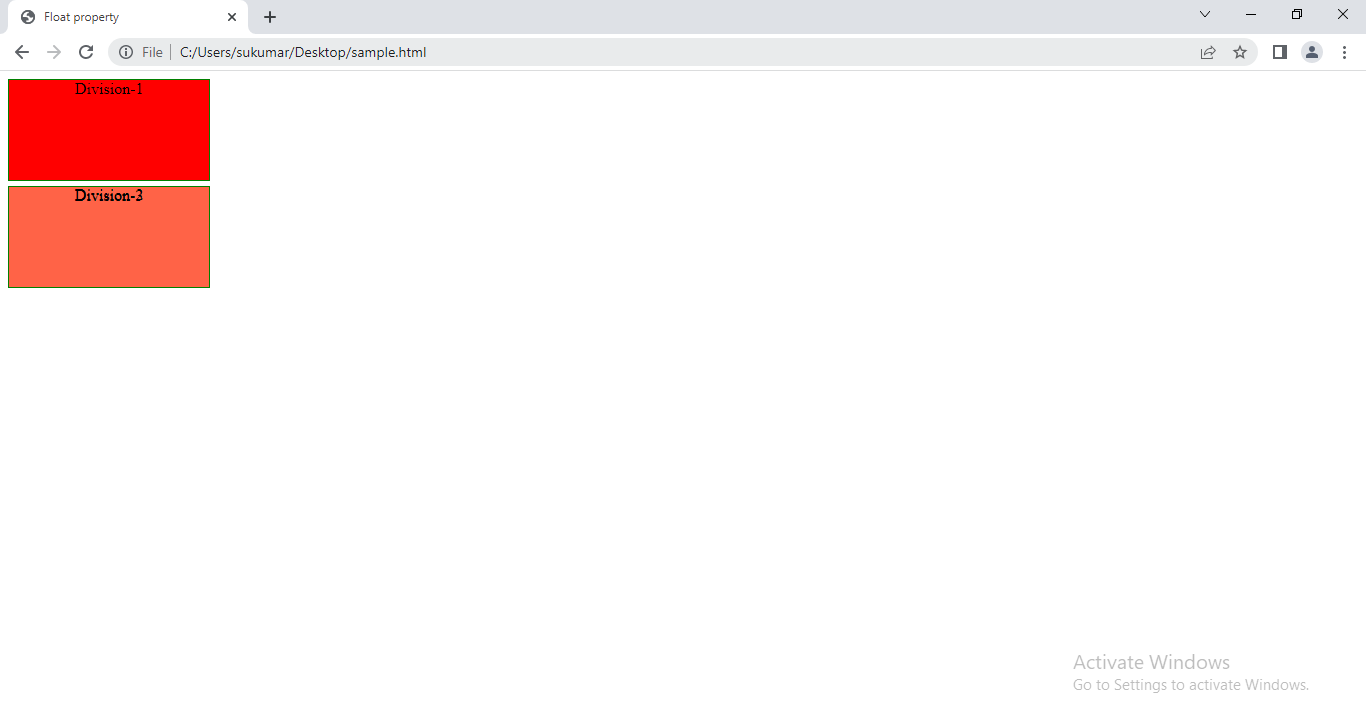
    <div id='one'> Division-1</div>

    <div id='two'> Division-2</div>

    <div id='three'> Division-3</div>

</body>

</html>



To solve this problem, Clear property is useful.

Syntax: clear:left|right|both;

Left-

Right-

Both-

Solution:

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Float property</title>

    <style>

       div

       {

           height:100px;

           width:200px;

           border:1px solid green;

           margin-bottom:5px;

           text-align:center;

       }

       #one

       {

           background-color: red;

           float:left;

       }

       #two

       {

           background-color:blue;

           clear:both;

       }

       #three

       {

           background-color:tomato;

       }

    </style>

</head>

<body>

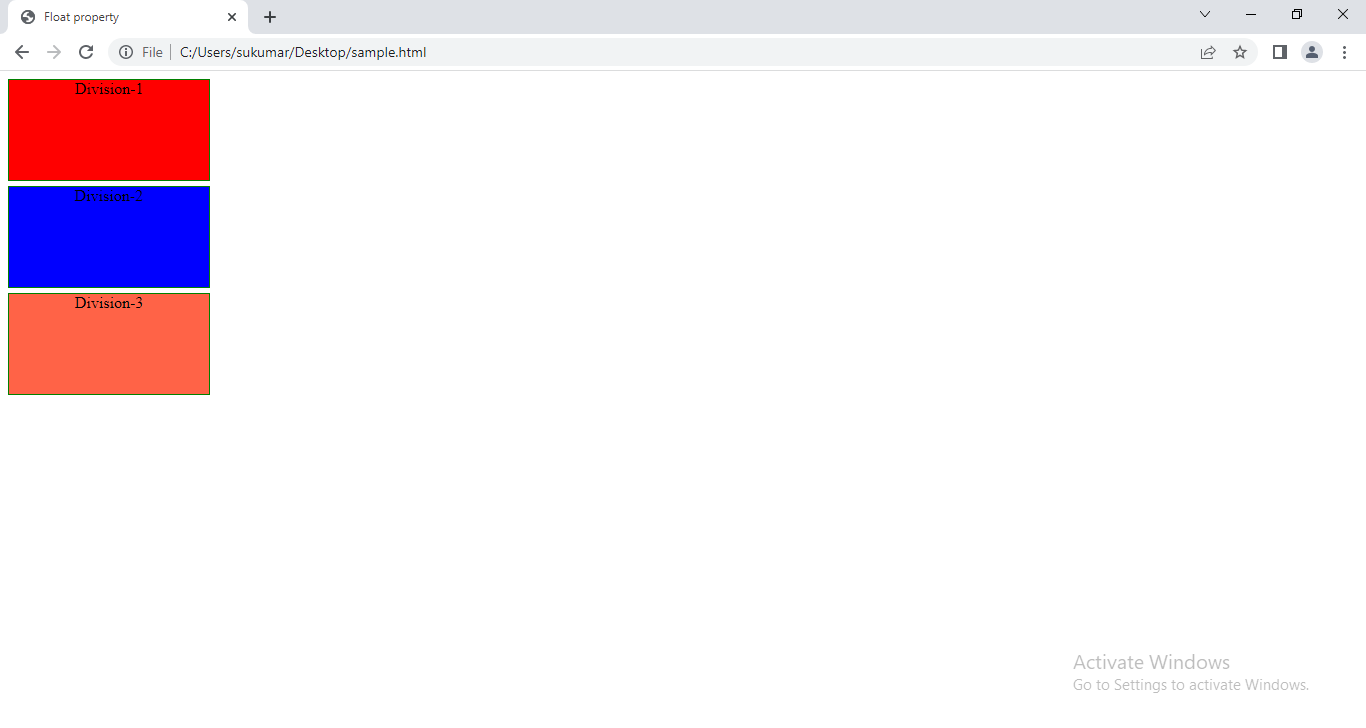
    <div id='one'> Division-1</div>

    <div id='two'> Division-2</div>

    <div id='three'> Division-3</div>

</body>

</html>



**12.Display:-**‘Display’ is a CSS Property through which we could able to change the default rendering type of any DOM Element.

Syntax:

Display:block|inline|inline-Block|none.

Following that the possible values it takes,

* Block:- Makes the DOM Element to render like a block level element.
* inline:- Makes any DOM Element to render like an Inline Element.
* inline-block:- Makes any DOM Element to render like a Block Level Element, but occupies in the same line as like in Inline Element.

Ex:- <img> is inline-block.

* None:- Makes element to not to be shown on the page. It still exist in DOM Structure).

Example:

<!DOCTYPE html>

<head>

    <meta charset="UTF-8">

    <title>Image Spriting</title>

    <style>

    #one,#two

    {

      display:inline;

      border:1px solid green;

    }

    #three,#four

    {

      display: block;

      height:200px;

      border:1px solid red;

    }

    p

    {

      display:none;

    }

    </style>

</head>

<body>

   <div id='one'>

     Division-1

   </div>

   <div id='two'>

     Division-2

   </div>

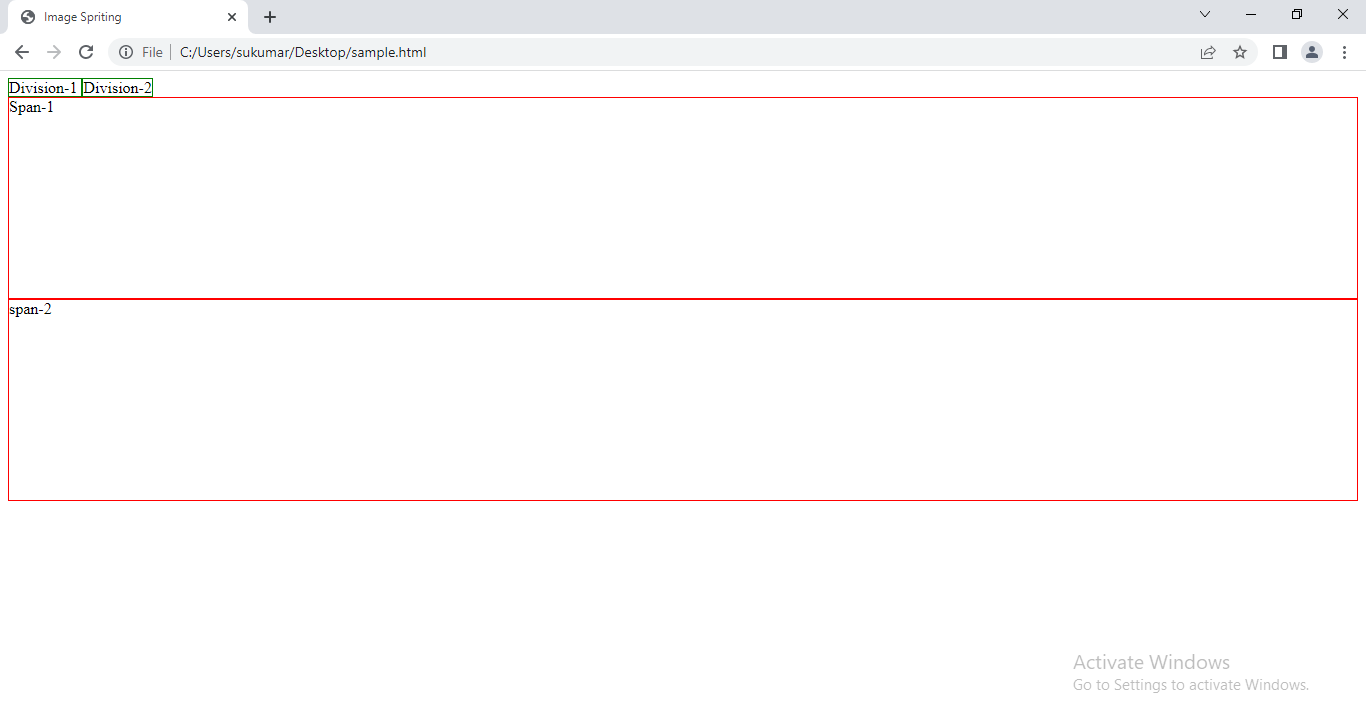
   <p>This program demonstrates the display property.</p>

   <span id='three'>Span-1</span>

   <span id='four'>span-2</span>

</body>

</html>



**13.Visibility:-**The visibility property specifies whether (or) not an element is visible.

Syntax:- visibility: hidden|visible|collapse|initial|inherit.

A.hidden:- The element is hidden But the element take default(or) specified space on webpage. The Decendant boxes can be visible.

B.visible:- It is default value. The element is visible.

C.collapse:-The value collapse is only meaningful for certain internal table objects: rows, row groups, columns, and column groups. It causes the object to be removed from the display; the space it occupied will be filled by subsequent siblings.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Grouping Selectors</title>

    <style>

       table

       {

           float:left;

           margin:10px;

       }

       #one

       {

           visibility:hidden;

       }

       #two

       {

           visibility:collapse;

       }

    </style>

</head>

<body>

    <table border='1' >

        <tr>

            <th>sno</th>

            <th>sname</th>

            <th>age</th>

        </tr>

        <tr>

            <td>1</td>

            <td>suku</td>

            <td>41</td>

        </tr>

        <tr><td>2</td>

            <td>veena</td>

            <td>39</td>

        </tr>

        <tr>

            <td>3</td>

            <td>sulakshmi</td>

            <td>12</td>

        </tr>

    </table>

    <table border='1' >

        <tr >

            <th>sno</th>

            <th>sname</th>

            <th>age</th>

        </tr>

        <tr id='one'>

            <td>1</td>

            <td>suku</td>

            <td>41</td>

        </tr>

        <tr><td>2</td>

            <td>veena</td>

            <td>39</td>

        </tr>

        <tr>

            <td>3</td>

            <td>sulakshmi</td>

            <td>12</td>

        </tr>

    </table>

    <table border='1' >

        <tr >

            <th>sno</th>

            <th>sname</th>

            <th>age</th>

        </tr>

        <tr id='two'>

            <td>1</td>

            <td>suku</td>

            <td>41</td>

        </tr>

        <tr><td>2</td>

            <td>veena</td>

            <td>39</td>

        </tr>

        <tr>

            <td>3</td>

            <td>sulakshmi</td>

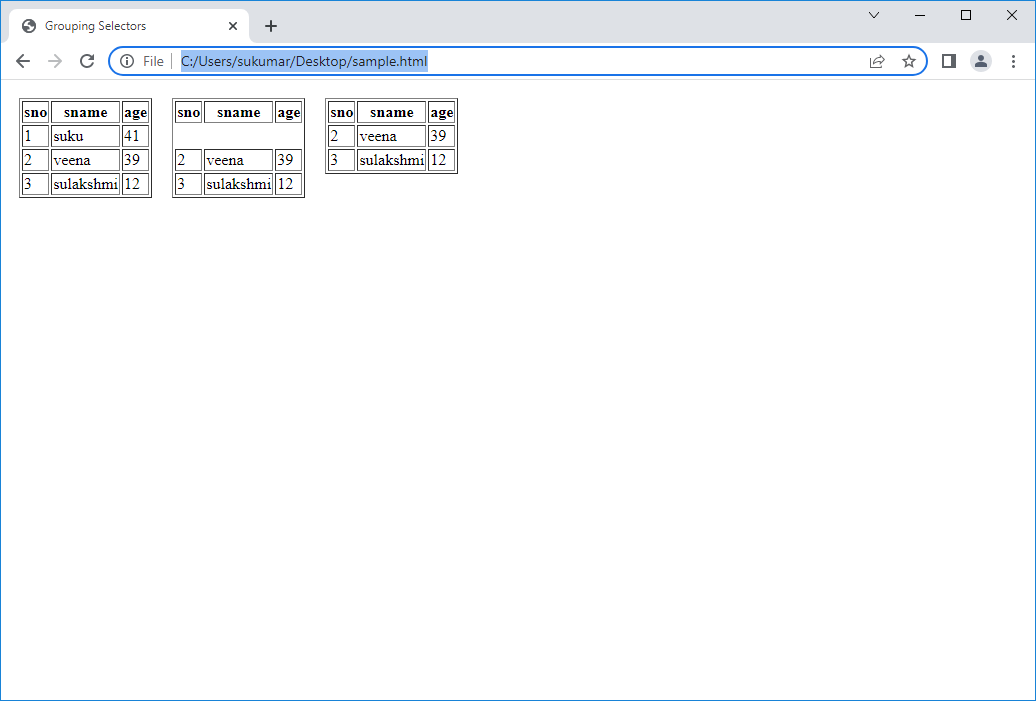
            <td>12</td>

        </tr>

    </table>

  </body>

</html>

****

**14.Position:-**Till now in order to move the DOM Elements to a specified position we are using Padding and Margin properties. While using these Padding or Margin properties to move the DOM Element, it is not actually moving the DOM Element but it is increasing the dimensions.

In order to actually move the DOM Elements to a required position without increasing its dimensions we use the following CSS Properties (Top, Left, Right and Bottom).

Not every DOM Element is capable of considering the above four CSS Properties but the DOM Element which are position can only consider above properties. CSS Position is the property through which we could able to control the position of any DOM Element.

Following the possible value a position attribute takes,

1. Static

2. Relative

3. Absolute

4. Fixed

5. Sticky

**14.1.Element with position Static:** Every DOM Element by default holds the static position which indicates the DOM Element cannot be moved to any position from its default position. (It will not consider the Top, Left, Right and Bottom properties).

Syntax: position : static;

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Grouping Selectors</title>

    <style>

       div

       {

       position:static;

       top:400px;

       left:200px;

       border:1px solid green;

       }

    </style>

</head>

<body>

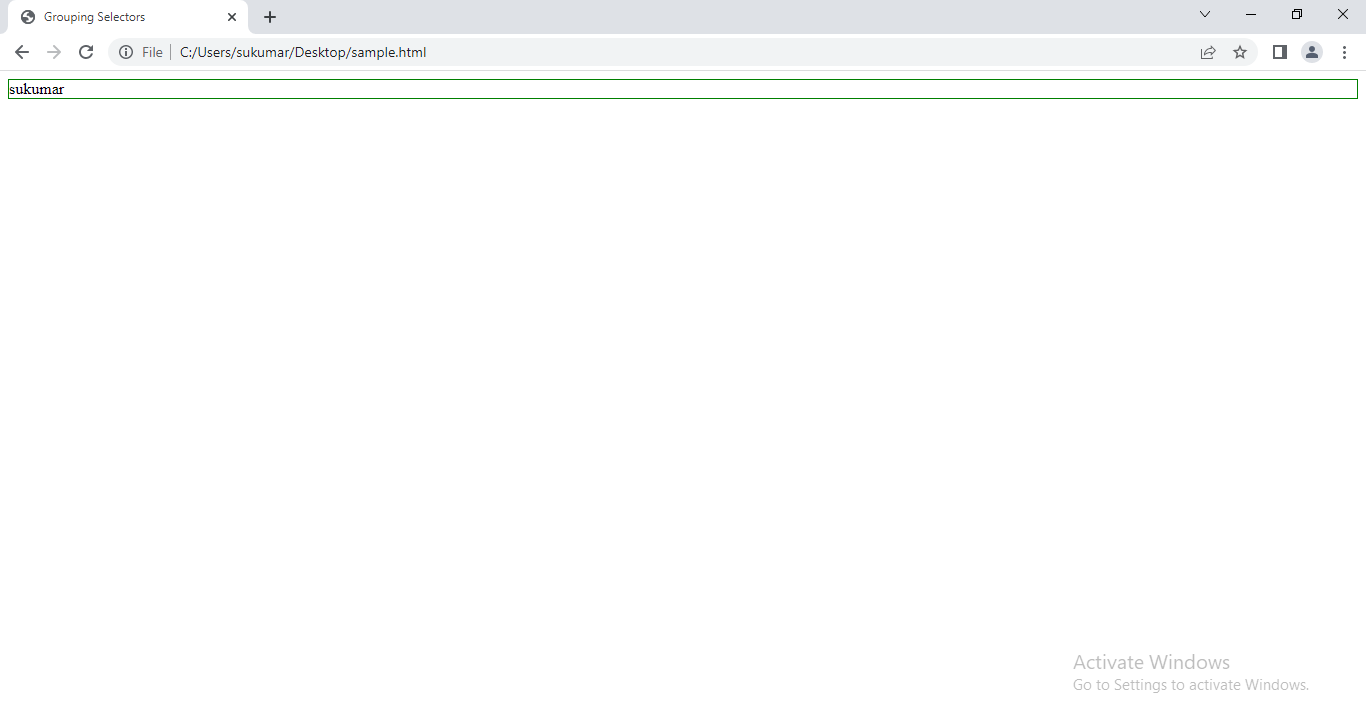
   <div>

     sukumar

   </div>

</body>

</html>

****

**14.2.Element with Position Relative:-**Any DOM Element with position relative holds the following properties,

• It is capable of moving to any required position within the page. (It considers the Top, Right, Left and Bottom properties).

• While moving to a new position it never loses space been occupied on load of the page.

• While moving to a new position it always moves relevant to its default position.

Syntax:

position : relative;

**Example**:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

        .block {

                    border: 1px solid;

                    text-decoration: underline;

                    text-align: center;

                    width: 130px;

                    height: 130px;

                }

        .block1 {

                    background: green;

                }

        .block2 {

                    background: blue;

                    left: 500px;

                    top: -120px;

                    position: relative;

                }

        .block3 {

                    background: orange;

                }

        </style>

  <title>Document</title>

</head>

<body>

  <div class="block block1">    Block 1

  </div>

  <div class="block block2">    Block 2

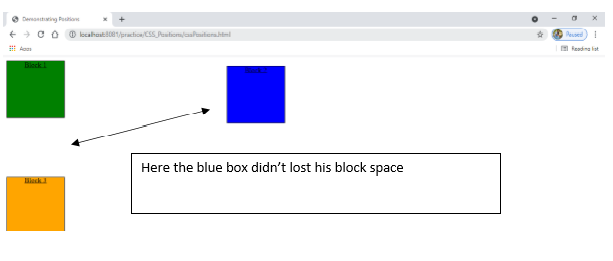
  </div>

  <div class="block block3">    Block 3

  </div>

  </body>

</html>



**14.3.Element with Position absolute:-**Any DOM Element with position absolute holds the following properties,

* It is capable of moving to any required position.
* It considers Top, Left, Right and Bottom properties.
* Element with position absolute will automatically loses its default space it occupies in the page.
* While moving to a new position it always moves relevant to its parent position.
* While depending on the parent position it only depends on the parent whose position value is non-static.
* If its intermediate parent doesn’t hold position on static, it travers to its grandparents until it finds an element or a parent with position non-static.

Note: Elements with position absolute automatically jumps from default X-Y Axis to Z-Axis.

Syntax: position : absolute;

Example:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

      .d1

      {

        border:1px solid red;

        height:100px;

        width:100px;

      }

      #one

      {

        background-color: blue;

      }

      #two

      {

        background-color: orange;

        position: absolute;

        left:400px;

      }

      #three

      {

        background-color: green;

      }

  </style>

  <title>Document</title>

</head>

<body>

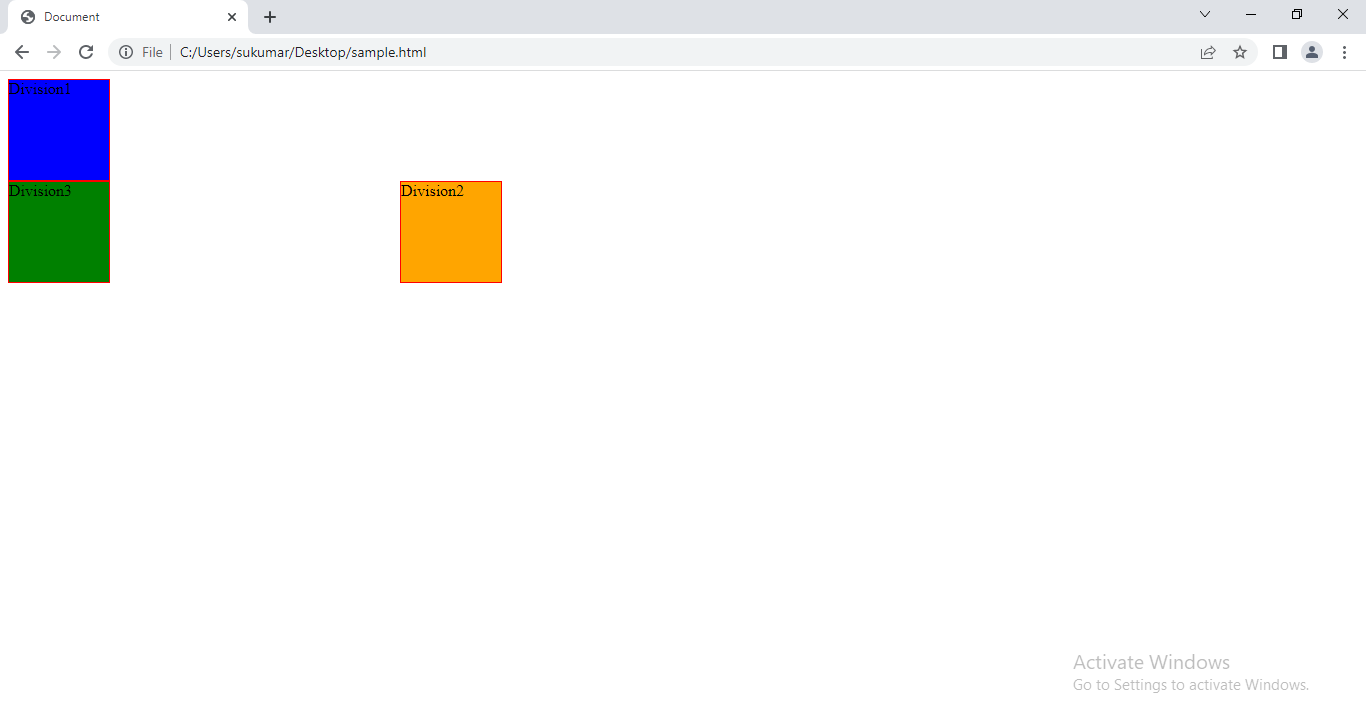
   <div class='d1' id='one'>Division1</div>

   <div class='d1' id='two'>Division2</div>

   <div class='d1' id='three'>Division3</div>

</body>

</html>



Example:2

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

      .d1

      {

        border:1px solid red;

        height:100px;

        width:100px;

      }

      #one

      {

        background-color: blue;

      }

      #two

      {

        background-color: orange;

      }

      #three

      {

        background-color: green;

      }

      #two-1

      {

        height:50px;

        width:50px;

        border:1px solid orange;

      }

      p

      {

        position:absolute;

        top:10px;

        left:500px;

        background-color: blue;

      }

      </style>

  <title>Document</title>

</head>

<body>

   <div class='d1' id='one'>Division1</div>

   <div class='d1' id='two'>Division2

     <div id='two-1'>

       <p>sukumar</p>

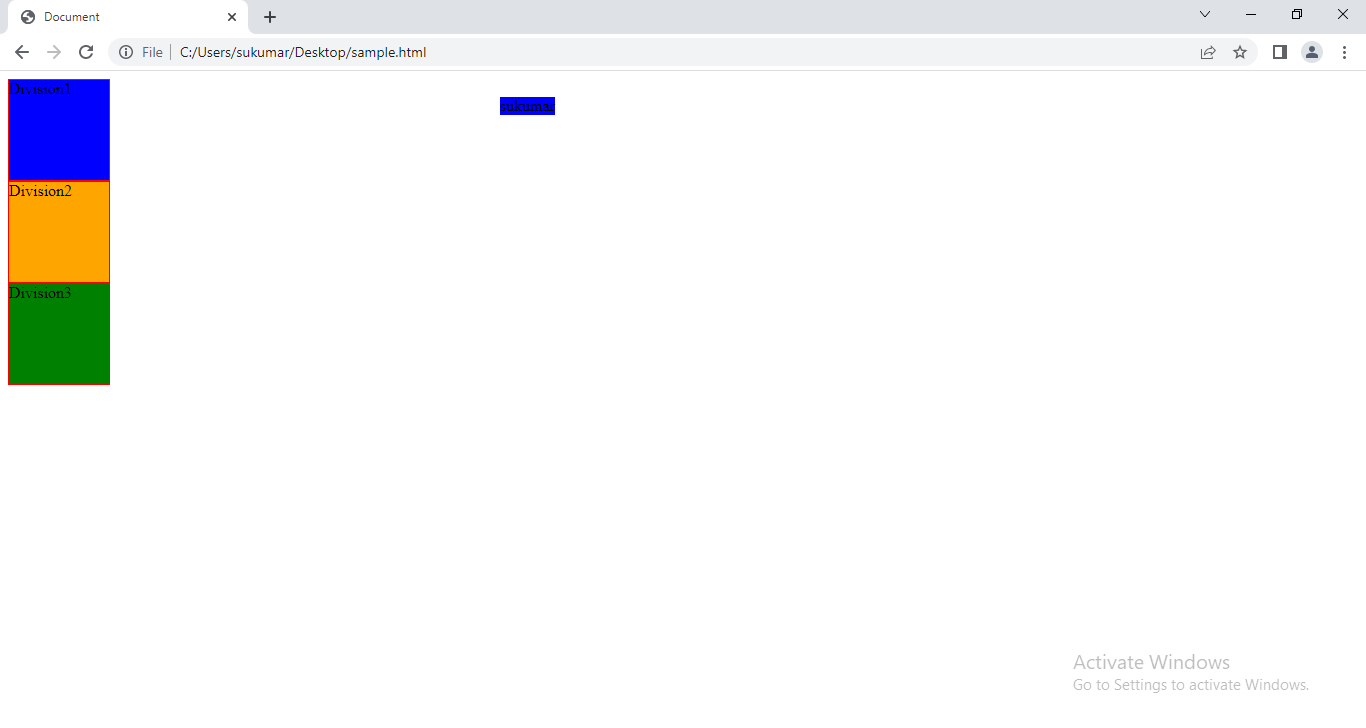
     </div>

   </div>

   <div class='d1' id='three'>Division3</div>

</body>

</html>



**14.4.Element with Position fixed**:- Element is positioned relative to browser window. Where the only difference is once the element gets its fixed position it doesn’t move from its original position even when we scroll. It’s place is occupied by immediate next element.

**Syntax:**

**Position:fixed;**

Example:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

       #m

       {

         height:200px;

         width:400px;

         margin:auto;

         overflow:auto;

         border:5px inset black;

       }

       .d1

       {

         height:100px;

         width:100px;

         border:1px solid red;

       }

       #two

       {

          position:fixed;

          background-color: green;

       }

    </style>

  <title>Document</title>

</head>

<body>

  <div id='m'>

   <div class='d1' id='one'>Division1</div>

   <div class='d1' id='two'>Division2</div>

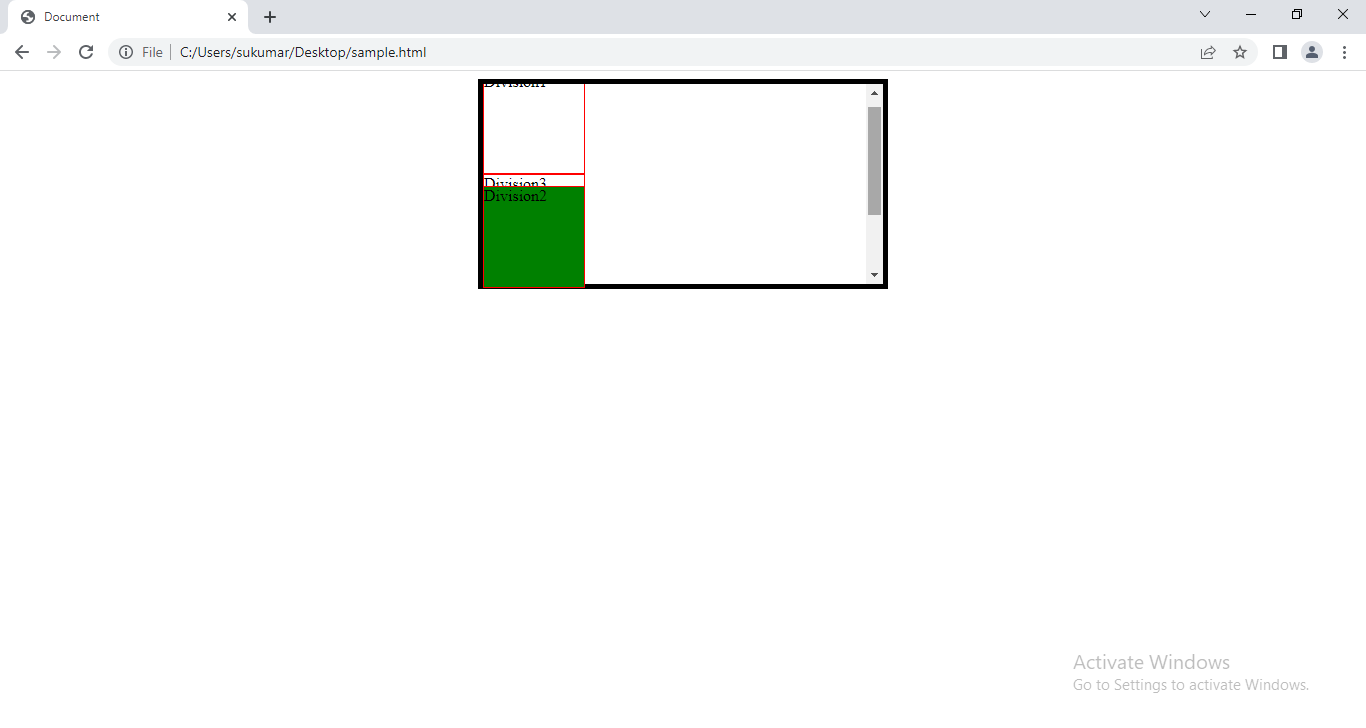
   <div class='d1' id='three'>Division3</div>

   <div class='d1' id='four'>Division4</div>

  </div>

</body>

</html>



**Note:-** even you scroll , division will not move.

**14.5.Element with position sticky**:-Any DOM Element with position sticky is almost like an element with position relative, where the only difference is once its position values been given (Top, Left, Right and bottom properties), if we try to scroll the element out of its view port, it automatically turns to fixed position and doesn’t get scrolled.

Syntax:- position:sticky;

Example:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

       #m

       {

         height:200px;

         width:400px;

         margin:auto;

         overflow:auto;

         border:5px inset black;

       }

       .d1

       {

         height:100px;

         width:100px;

         border:1px solid red;

       }

       #two

       {

          top:50px;

          position:sticky;

          background-color: green;

       }

    </style>

  <title>Document</title>

</head>

<body>

  <div id='m'>

   <div class='d1' id='one'>Division1</div>

   <div class='d1' id='two'>Division2</div>

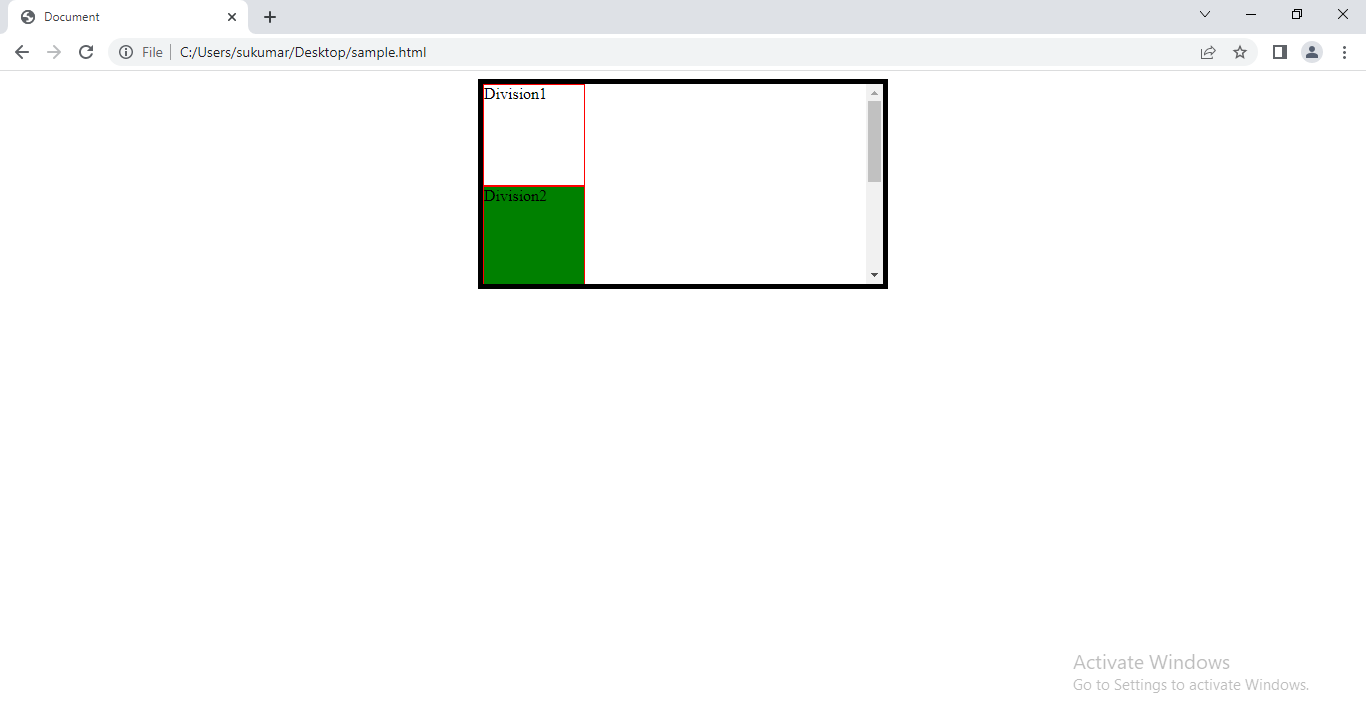
   <div class='d1' id='three'>Division3</div>

   <div class='d1' id='four'>Division4</div>

  </div>

</body>

</html>



**15.Z-Index Property**:-The elements which are falling under Z-Axis there is a chance of multiple elements override each other while rendering on the page. While the Elements are overriding we can control the rendering order through CSS Z-Index Property.

Z-Index is a CSS Property takes a positive number as a value through which we can specify the priority order, the with the higher priority order always renders on the top.

Syntax: z-index: integer\_value;

• Note: Z-Index property can only be applied to elements which fall under Z-Axis (Elements with position Non-Static).

Example:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

      .d1

      {

        height:200px;

        width:200px;

        margin:5px;

      }

      #one

      {

        background-color: red;

      }

      #two

      {

        background-color:blue;

        position:absolute;

        top:5px;

        left:5px;

      }

      #three

      {

        background-color:orangered;

      }

    </style>

  <title>Document</title>

</head>

<body>

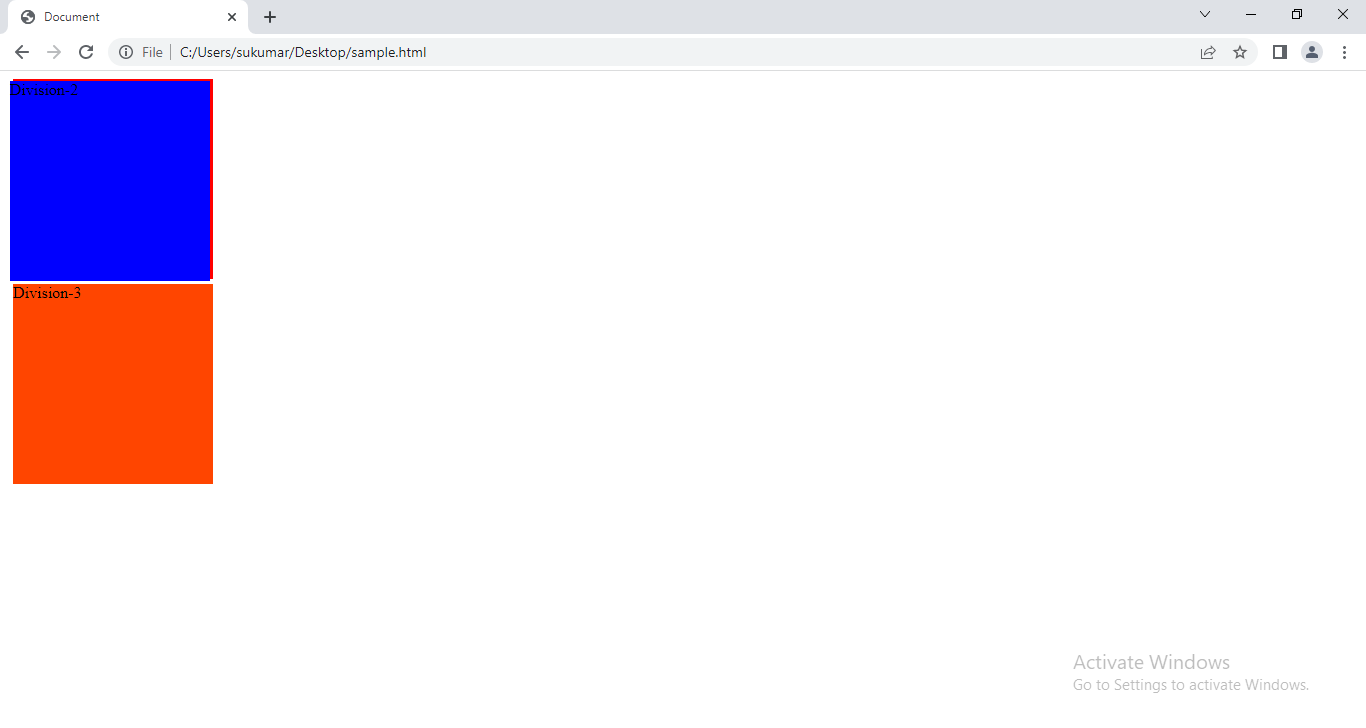
  <div id='one' class='d1' >Division-1</div>

  <div id='two' class='d1'>Division-2</div>

  <div id='three' class='d1'>Division-3</div>

</body>

</html>



Example:2

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

      .d1

      {

        height:200px;

        width:200px;

        margin:5px;

      }

      #one

      {

        background-color: red;

        position:relative;

        z-index:1

      }

      #two

      {

        background-color:blue;

        position:absolute;

        top:5px;

        left:5px;

      }

      #three

      {

        background-color:orangered;

      }

    </style>

  <title>Document</title>

</head>

<body>

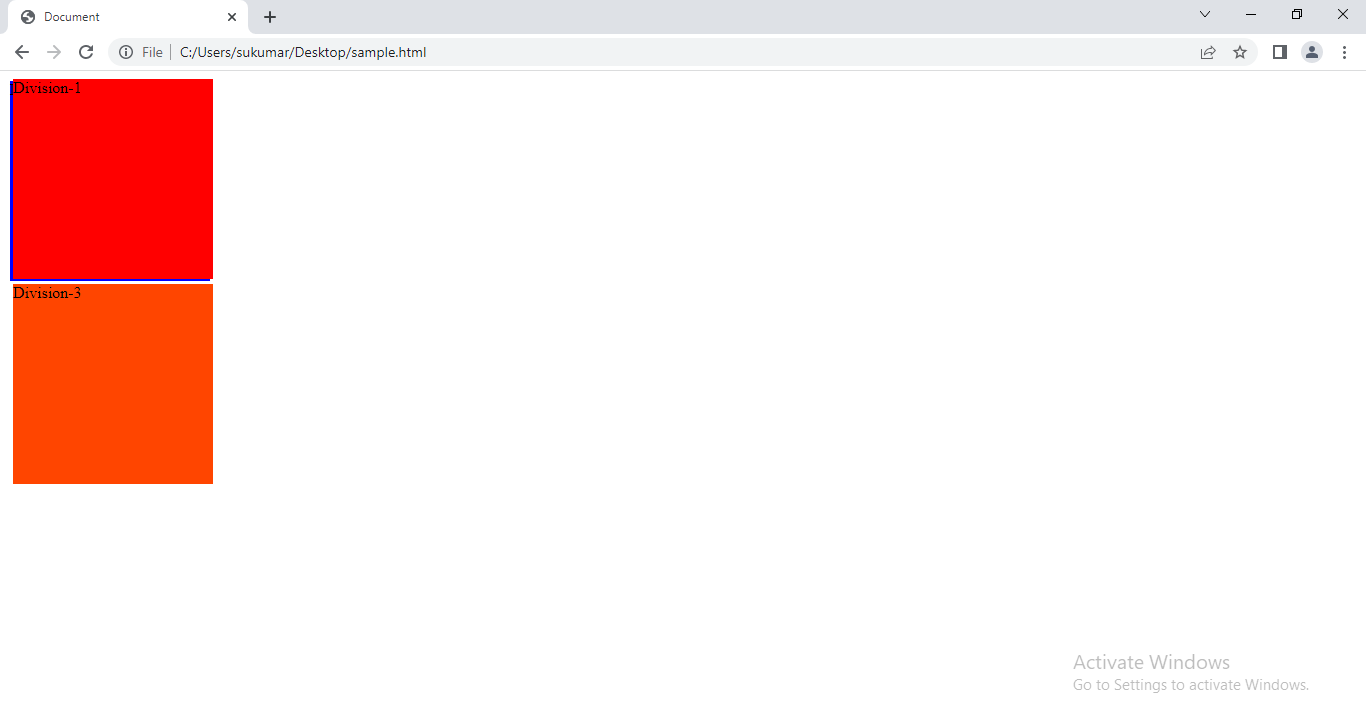
  <div id='one' class='d1' >Division-1</div>

  <div id='two' class='d1'>Division-2</div>

  <div id='three' class='d1'>Division-3</div>

</body>

</html>



**16.Opacity:-**While elements get override each other we could able to control the transparency level of the elements through CSS Opacity property.

It takes a value between 0 to 1

0 means element has no transparency.

1 means element has 100% transparency.

**Syntax: opacity: value;**

Example:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

      .d1

      {

        height:200px;

        width:200px;

        margin:5px;

      }

      #one

      {

        background-color: red;

        position:relative;

        z-index:1;

        opacity:0.3;

      }

      #two

      {

        background-color:blue;

        position:absolute;

        top:5px;

        left:5px;

      }

      #three

      {

        background-color:orangered;

        opacity:0.9;

      }

    </style>

  <title>Document</title>

</head>

<body>

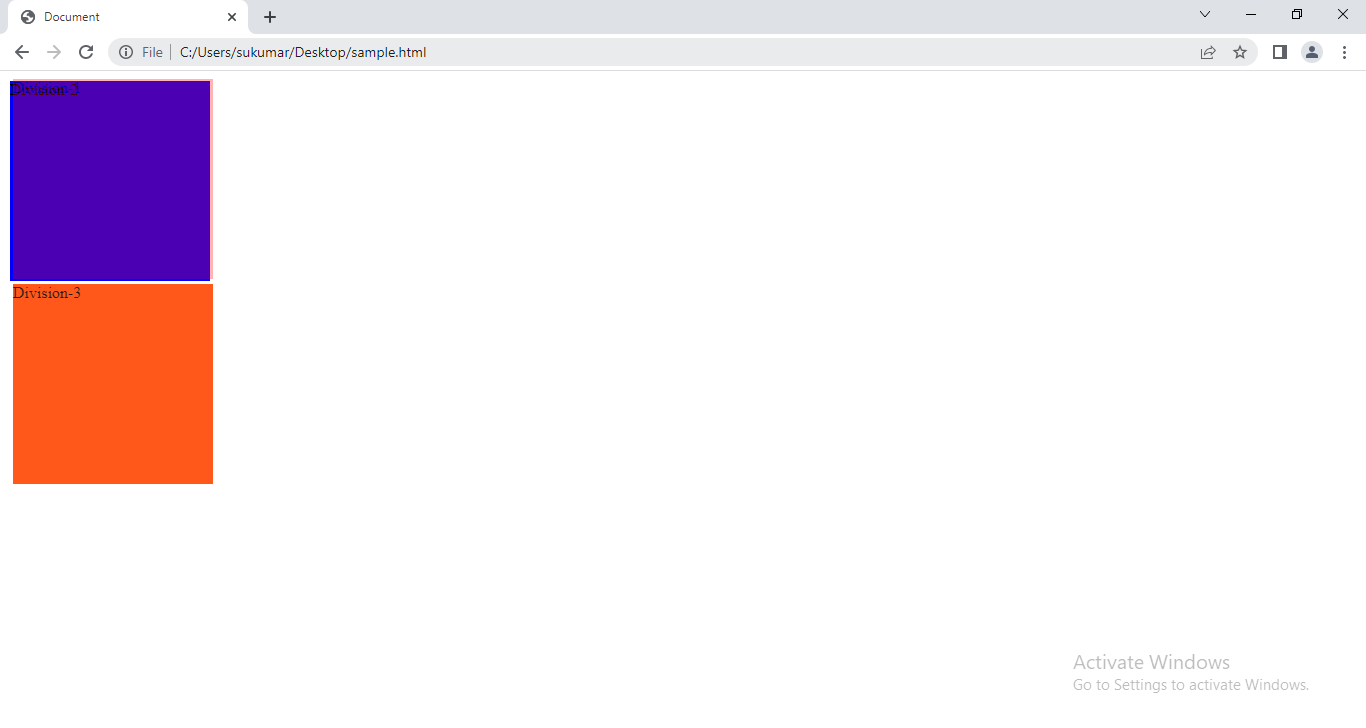
  <div id='one' class='d1' >Division-1</div>

  <div id='two' class='d1'>Division-2</div>

  <div id='three' class='d1'>Division-3</div>

</body>

</html>



**17.List Style Properties:-**The css list properties allow us to

1. Set list item marker for unorder list.
2. Set list item marker for order list
3. Set image as list item marker.

The css properties are

1. List-style-type:- It specifies type of marker for list-item in list.‘list-style’ is the CSS Property through which we can control the type of List style getting rendered for both ‘UL’ and ‘OL’ tags.

Syntax:- list-style-type:value;

Value is Disc,circle,square,lower-alpha,upper-alpha,lower-roman,upper-roman,lower-greek,upper-greek and none.

1. List-style-position:- it specifies “ list-item marker should appear inside or outside the block box.”.

Syntax:- list-style-position: inside|outside|initial.

3. List-style-image:- This property specifies the image to use as a list marker for an item in a list (an element whose display property has the value list-item). If the specified image is available, it will replace any marker specified by the list-style-type property.

Syntax: list-style-image:url(‘ ‘);

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=<style type="text/css">

    <style>

       ol

       {

         list-style:decimal;

       }

       li

       {

         border: 1px solid red;

         margin-bottom: 2px;

         width:150px;

       }

       .one

       {

         list-style-position: inside;

         border:1px groove blue;

       }

    </style>

  <title>Document</title>

</head>

<body>

   <h2 align='center'>Fruits-List </h2>

   <ol>

     <li>Mango</li>

     <li>Banana</li>

     <li>Apple</li>

     <li>Cherry</li>

   </ol>

   <h2 align='center'>Cricket Player List </h2>

   <ol>

     <li class='one'>Robinsingh</li>

     <li class='one'>Allan Donald</li>

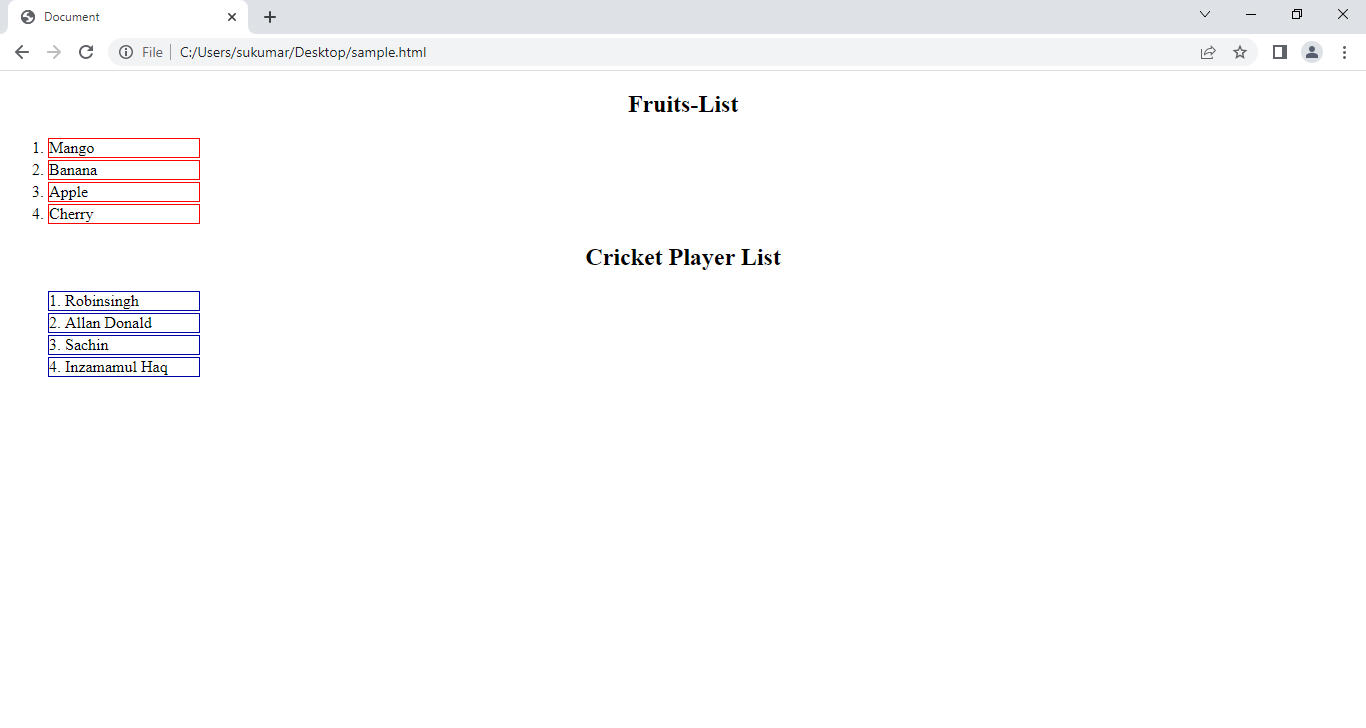
     <li class='one'>Sachin</li>

     <li class='one'>Inzamamul Haq</li>

   </ol>

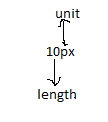
</body>

</html>

****

**18. Length Units:-** css has several length units .

Example:

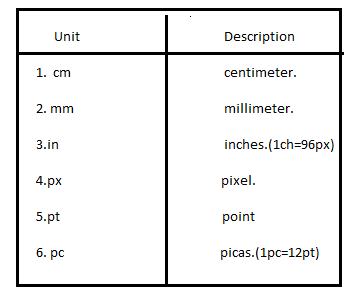


They have been divided into two groups.

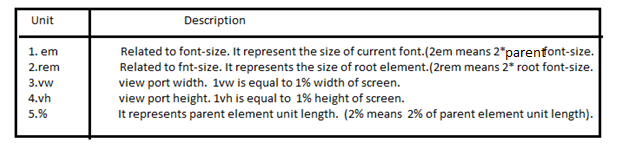
18.1. Absolute Length units.

18.2. Relative Length units.

18.1. Absolute Length Units:- The absolute length unit is fixed. It is not recommended for use on screen because screen sizes vary. However, they can be used if the output medium and their height and width is known, such as for print layout.



18.2. Relative length Units:-Relative length units specify a length relative to another length property.  They can be used , if web page is opened in different mediums.



**19. Variables:**

**Syntax to variable defintion:**

--varname: value;

The variable has either local scope or global scope.

If u declare the variable inside **:root** selector, then variable has global scope.

If u declare the vairable inside other selector then it has local scope.

**Syntax to Access the variable value:**

**Var(--varname, value-1);**

Where var is function . It returns value of variable , if it found the variable. Otherwise , it returns the second argument.

Example:

<!DOCTYPE html>

<html lang="en" >

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title>

    <style>

      :root{

        --bgColor:red;

      }

      div{

        background-color: var(--bgColor);

      }

      p{

        background-color: var(--bColor,blue);

      }

    </style>

</head>

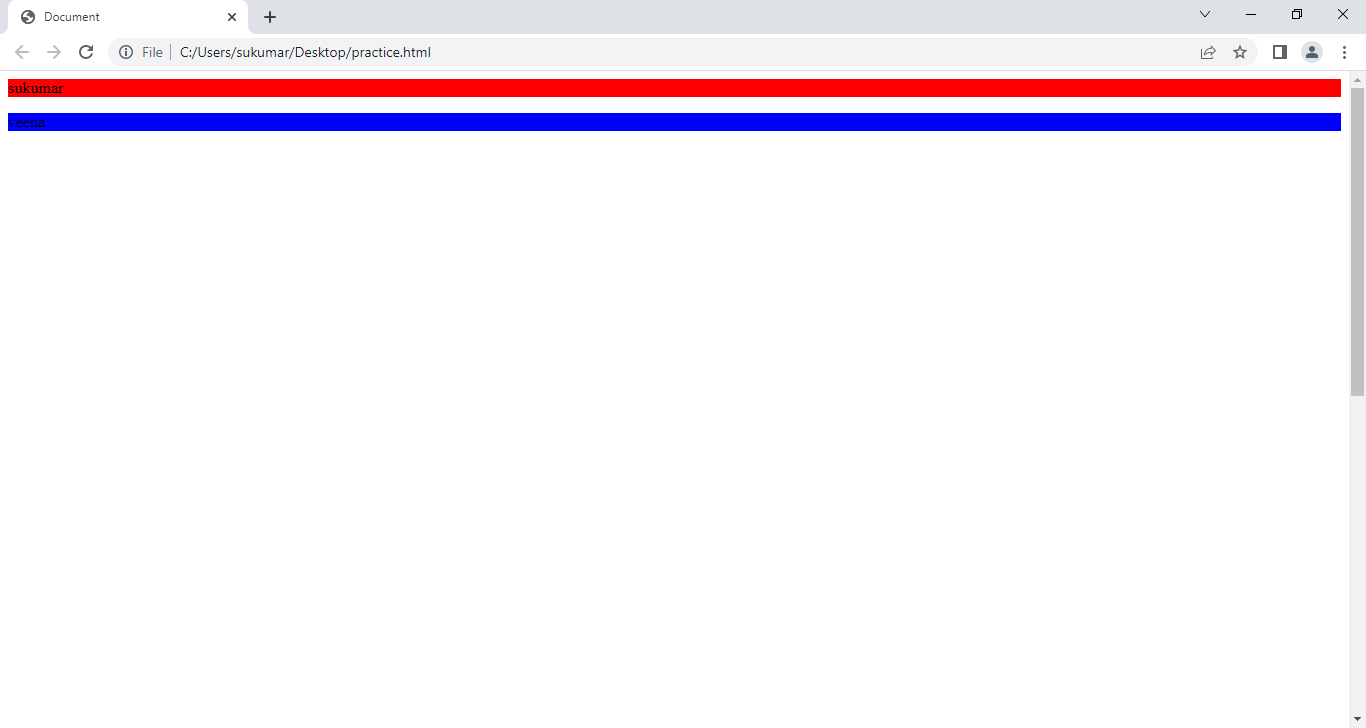
<body style="height:200vh">

    <div> sukumar </div>

    <p>veena</p>

</body>

</html>



21.@import:- The @import rule allows you to import a style sheet into another style sheet. The @ import rule must be at top of document. The @ import rule also supports media queries, so you can allow import to be media-dependent.

Syntax:-

@import url|string list-of-mediaqueiries;

* url or string representing the location of resource to import. The url may be absolute or relative.
* A comma-separated list of media queries conditioning the application of css rules defined in the linked URL.

Example1:-

@import “abc.css” print;

Import “abc.css” style sheet only if the media is print.

Example2:-

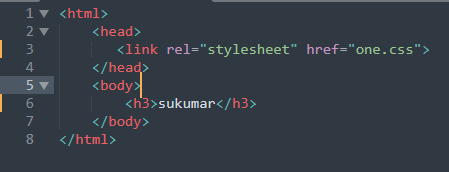
@import “abc.css” screen and (max-width:600px);

Import the “abc.css” style sheet only if the media is screen and viewport is maximum 768px.

Example Program:-

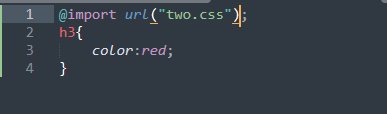
Sample.html

===========



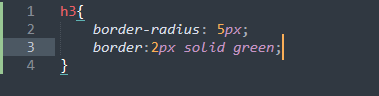
One.css

=====



Two.css

======



Output:

======

