**Contents**

1.Clip-Path

2.shadow

3.User Interface properties

4.Multiple Columns

5.TrypoGraphical Properties

6.Transform

7.Transition

8.Animation

9.3D Transformation

10.Flex

11.Grid

12.Media

13.Gradients

**1.clip-path:**The clip-path property lets you clip an element to a basic shape or to an SVG source.

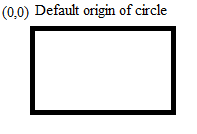
Syntax:

Clip-path:basic-shape|margin-box|padding-box|content-box|stroke-box|view-box|fill-box|none|inset.

The basic shapes are circle,ellipse or polygon.

**i.**circle(radius at x% y%)

x and y are center point position. By default, x and y are element top left most corner.



ii.inset(top%,bottom%,left%,right%)

The % of element is cut from specified side.

iii. ellipse(x% y% at a% b%)

a and b specifies the eclipse center point position. X and y are x-axiz and y-axis length of eclipse.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

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

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

    <title>Document</title>

    <style>

        div{

            height:20vh;

            width:50vw;

            margin:20px;

            border:2pxsolidgreen;

            border-radius: 5px;

            background-color: red;

            outline:5pxsolidblue;

        }

        div:first-child{

            clip-path: circle(10% at 50%50%);

        }

        div:nth-child(2)

        {

            clip-path:ellipse(20%30% at 50%50%);

        }

        div:nth-child(3)

        {

            clip-path: inset(10%20%40%20%);

        }

    </style>

</head>

<body>

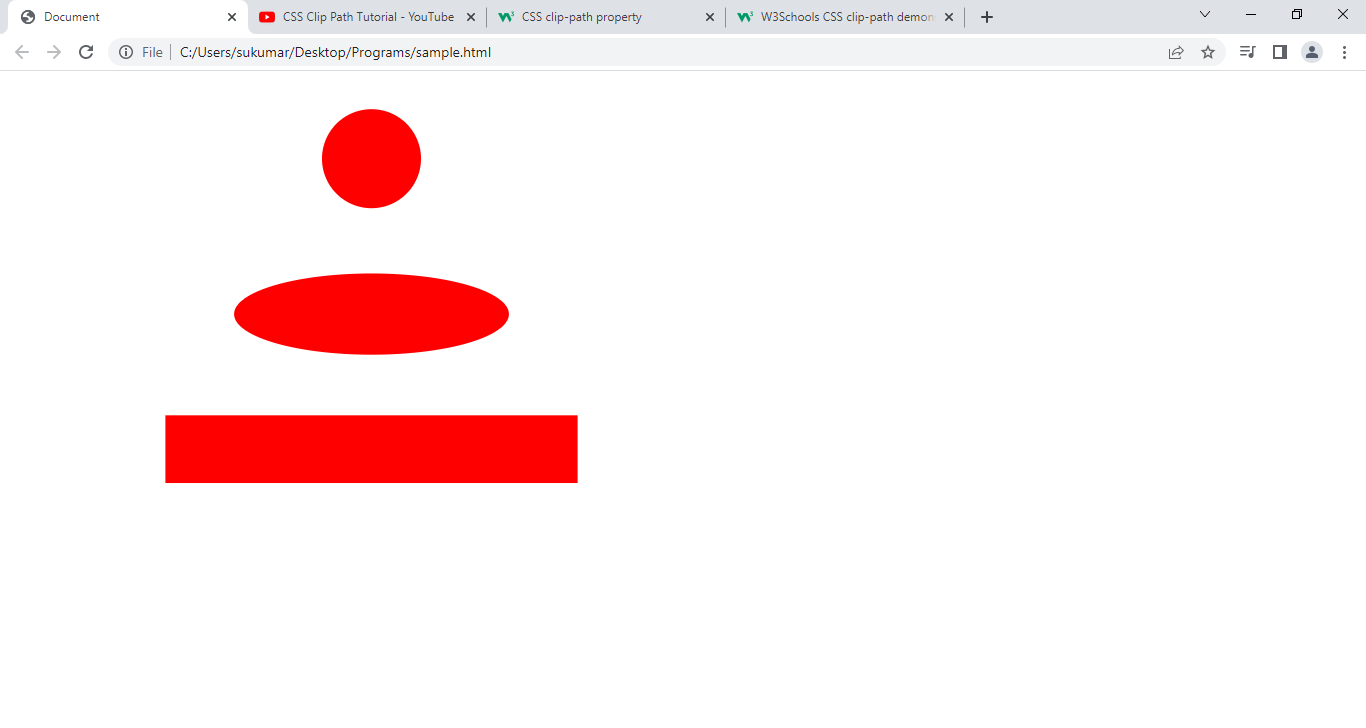
    <div>sukumar</div>

    <div>veena</div>

    <div>sumasulaha</div>

</body>

</html>



**2.shadow:**The shadow property adds one or more shadows to box.

Syntax:-

Shadow: none|horizontal vertical blur spread color inset initial.

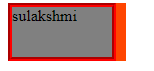
* + - horizontal:- if value is positive, shadow will be right side. If value is negative, shadow will be leftside.
    - vertical:- if value is positive, shadow will be below the box. If value is negative, shadow will be above the box.
    - blur:- It is optional. The value represents blur of shadow.
    - spread:- it is optional. Positive value increases box-shadow. Negative value decreases box-shadow.
    - inset:- It is optional.It changes from outer shadow to innershadow.

Example:

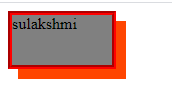
box-shadow: 0px 10px orangered;



box-shadow: 10px 0px orangered;



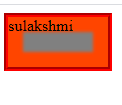
box-shadow: 10px 10pxorangered;



box-shadow: 0px 0px 15px inset orangered;



box-shadow: 0px 0px 3px 15px inset orangered;



**3.UserInterface Properties**:-

3.1.cursor:This property sets the type of cursor to be displayed for a pointing device.

cursor: { [ uri ], { auto | crosshair | default | e-resize | help | move | n-resize | ne-resize | nw-resize | pointer | progress | s-resize | se-resize | sw-resize | text | w-resize | wait } | inherit } ;

3.2.Resize:-It specifies wheather or not element is resizable by the user.

**Note:** The resize property does not apply to inline elements or to block elements where overflow="visible". So, make sure that overflow is set to "scroll", "auto", or "hidden".

Syntax:- resize: none |both |horizontal |vertical |initial |inherit;

**4.MultipleColumns**:We can take chunk of content and flow it into multiple columns as in news paper. We do this using multiple-columns.

1. column-colunt: It specifies no.of columns an element content should be divided into.

Syntax:

column-count:number|auto|initial |inherit.

Number: the optional no.of columns into which content of element will be flowed.

Auto:default value. The no.of columns depends on other property like column-width.

2.Column-gap: It specifies gap between columns.

Syntax:

column-gap: number|normal|initial|inherit;

Length: A specified length that will set gap between columns.

Normal: It is default value. Normal gap between columns is 1em.

3.Column-rule:- it is a short-hand property. It set rule style,color of rule between columns.

Syntax:

column-rule: width style color|initial|inherit;

4. Column-rule-width: It specifies width of rule between columns.

Syntax:

column-rule-width: thick|thin|medium|length|initial|inherit.

5.Column-rule-style: It specifies style of rule between columns.

Syntax:

column-rule-style: one|solid|dashed|dotted|double|groove|inset|outset|initial|inherit.

6.Column-rule-color: It specifies color of column-rule between columns.

Syntax:

column-rule-color:color|initial|inherit;

7.Column-width: It specifies width of columns.

Syntax:

column-width: auto|number|initial|inherit;

Auto: It is default value, which is determined by browser.

8.Column-span: The column-span property specifies how many columns an element should span across.

Syntax:

column-span: all|1|initial|inherit;

**5.TrypoGraphical properties:**

5.1.font-Related Properties:- The font properties define font family,size ,style and boldness of the text. The font properties are

1. font-family
2. font-size
3. font-style
4. font-weight
5. font-variant.

1.font-family:- This property specifies font for element. There are two types of font family names.

1.family name:- specific font-familites like Times new Roman, Arieal.

2.Generic-family:-

|  |  |  |
| --- | --- | --- |
| Generic Family | Family Names | Used For |
| Serief | Georigia,Times,Times new roman. |  |
| Sans serief | Verdana,Arial ,Helvetica. |  |
| Cursive | Monotype ,cursive | HandWriting |
| Monospace | Courier,couriernew | Coding |
| Fantasy | Impact | Headings |

Syntax:- font-family: familyname1,familyname2,…genericfamily;

* + - The value is list family names , generic families or both.The names are separated by comma.If browser does not support first font, it tries the next font.
    - if you specify both then we should first specify family names next we should specify generic families.
    - Font family names are quoted or unquoted strings, while generic family names are keywords and shouldn’t be quoted.

2.font-size:- This property specifies the font-size to content of element.

Syntax:

font-size: { absolute-size | relative-size | length | percentage | inherit } ;

2.1) Absolute-size/Tshirt-size:-The values are xx-small , x-small, small, medium ,large , x-large , xx-large.

2.2) Relative-size:- smaller,larger.

Those keywords will make the font size smaller or larger than the inherited value, by some factor that isn’t exactly defined.

3.font-weight:- It specifies thickness of font.

Sytax:

font-weight: { 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | bold | bolder | light|lighter |medium| normal | inherit } ;

The following keywords can also be used for this property:

Bold: is a synonym for 700 .

Bolder: selects a font weight that’s darker than that inherited from the parent element. lighter: selects a font weight that’s lighter than that inherited from the parent element. normal: is a synonym for 400.

4.font-style:- It specifies font style for text.

Syntax:

font-style:value

Values are normal,italic,oblique and initial.

Q) What is diff between italic & oblique?

A) In the purest (type designer) sense, an oblique is a roman font that has been skewed a certain number of degrees (8-12 degrees, usually). An italic is created by the type designer with specific characters (notably lowercase a) drawn differently to create a more calligraphic, as well as slanted version.

5.2.TextRelated Properties:-

1.textcolor:- The text color property is used to set color of text.

Syntax:

color:colorname|rgbvalue|initial.

2.TextDirection: This property specifies text writing direction.

Syntax:

direction:ltr|rtl|initial.

Default value is left to right.

3.Letter-spacing:-This property increases space between characters in line.

Syntax:

lette-spaccing:normal|length|initial.

Normal is default value.

4.text-align: This property specifies horizontal alignment of text. If text-direction is ltr, default align is ltr& if text direction is rtl , then default text align is rt.

Note:- Arrangement o ftext-relative to margin is text-alignment.

Syntax:

Text-align:left|right|center|justified|initial.

5.text-decoration: This property specifies decoration added to text.

Syntax:

text-decoration:underline|none|overline|line-through|initial.

6.text-indent: This property specifies indentation of first line in text-block.

Syntax:

text-indent: length|%|initial.

7.text-shadow:- This property applies a shadow to text.

Syntax:

text-shadwo:h-shadow|v-shadow|blue|color|initial|none.

8.text-transform:- This property controls captialization of text.

Syntax:

text-transform:none|capitialize|uppercase|lowercase|initial.

9.Word-spacing:- It incresse(or) decrease space between words in text.

Syntax:

word-spacing:normal|length|initial.

10. White-space:- This property specifies how white-space inside element is handled.

Syntax:

white-space:normal|nowrap|pre|pre-line|initial.

10.1)Normal:- sequence of white space Is collapsed into single space.

Ex:-

<div >This is sukumar .Hai

</div>

Output:-



10.2)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 <br> tag is encountered.

Example:-

<div >This is sukumar.I like programming skills & I have very interest in web technologies ,dbms,html,other programming language.This is sukumar.I like programming skills & I have very interest in web technologies ,dbms,html,other programming language.This is sukumar.

</div>

10.3) pre:- White space is preserved by browser. Text will wrap only on line breaks</br>.

10.4) pre-wrap: white space is preserved and text will wrap only on line breaks or when necessar.

10.4)pre-line:- sequence of white space is collapsed into single space.Text will wrap when necessary and on line breaks.

11.User Select:Thisproperty specifies whether the text of an element can be selected. In web browsers, if you double-click on some text it will be selected/highlighted. This property can be used to prevent this.

Syntax:user-select:auto|text|none|all.

All:- Text selection is made with one click instead of a double-click.

Text:- The text is only selected.

Auto: text will be selected, if browser allows.

None:- text will not be selected.

**6.Transform:** The transform is a property. That allows you to move,rotate,scale and skew the given element.

Syntax:

transform:none|[transform-function1transform-function2 ..etc]

1.None:- It specifies that no transform should be applied.

2.The transformation-functions are

a.Translate

b. rotate

c. scale

d.skew

e.matrix

**2.1. Translate**:- This function moves an element in the horizontal and /or vertical direction.

Syntax:

Transform:translate(tx,ty).

Where tx is distance in horizontal direction.

Ty is distance in vertical direction.

Tx,ty values can be negative.

2.2.TranslateX(tx):- It transforms element in horizontal direction.

Syntax:

Transform:translateX(tx,ty).

2.3.TranslateY(ty):- It transforms element in vertical direction.

Syntax:

Transform:translateY(ty).

Note:-

1. By default, the origin is top left corner of box. From origin , the element is moved.
2. After translate element, the space of element in normal flow as it is on webpage.
3. In-line element can’t take the transform property.

**2.2.Rotate:**It rotates an element around a fixed point on 2D plane. The fixed point is also called as transform origin. This default to center of element.

Syntax:-

Transform:rotate(angle);

If angle is positive, element is rotated in clock-wise.

If angle is negative, element is rotated in counter clock wise.

**2.3.Scale:**This transformation resizes the element in horizontal and /or vertical direction on 2D plane.

Syntax:

Transform: scale(sx,sy).

Where sx,sy are scaling factors.

If both are same,uniform scaling will be done.Ohterwise non-uniform scaling will be performed.

Tx and ty can be either positive or negative value.

If they are positive, element size is increased.

If they are negative, element size is decreased.

If tx& ty are 1 , no scaling will be done.

scaleX(sx):- It resize the element in only horizontal direction on 2Dplane.

scaleY(sy): It resize the element in only vertical direction on 2d plane.

Note: while performing scaling, the fixed point is center of element.

**2.4skew/shear:**This transformation that slants or turn shape of object is called shear transformation.

Syntax:- transform:skew(sx,sy)

Transform:skewX(sx);

Transform:skewY(sy);

Where

Sx and sy can be positive or negative angle.

The methods slants shape of object in given angle, depending on parameters given for x-axis and y-axis lines.

2.5**. transform-origin**: It sets origin (or) fixed point co-ordinate for 2D transformations.

Syntax:

transform-origin: *x-axis y-axis z-axis*|initial|inherit;

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

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

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

    <title>Document</title>

    <style>

        div{

            border:2pxsolidgreen;

            height:30px;

            width:100px;

            margin:10vh;

            text-align: center;

            }

        div:first-child{

            transform:translate(200px,200px);

        }

        div:nth-child(2)

        {

            transform: rotate(90deg);

        }

        div:nth-child(3)

        {

            transform: scale(1.5);

        }

        div:nth-child(4)

        {

            transform: skew(30deg);

        }

        div:nth-child(5)

        {

            transform-origin: centercenter;

            transform: rotate(90deg);

        }

   </style>

</head>

<body>

    <div>One</div>

    <div>Two</div>

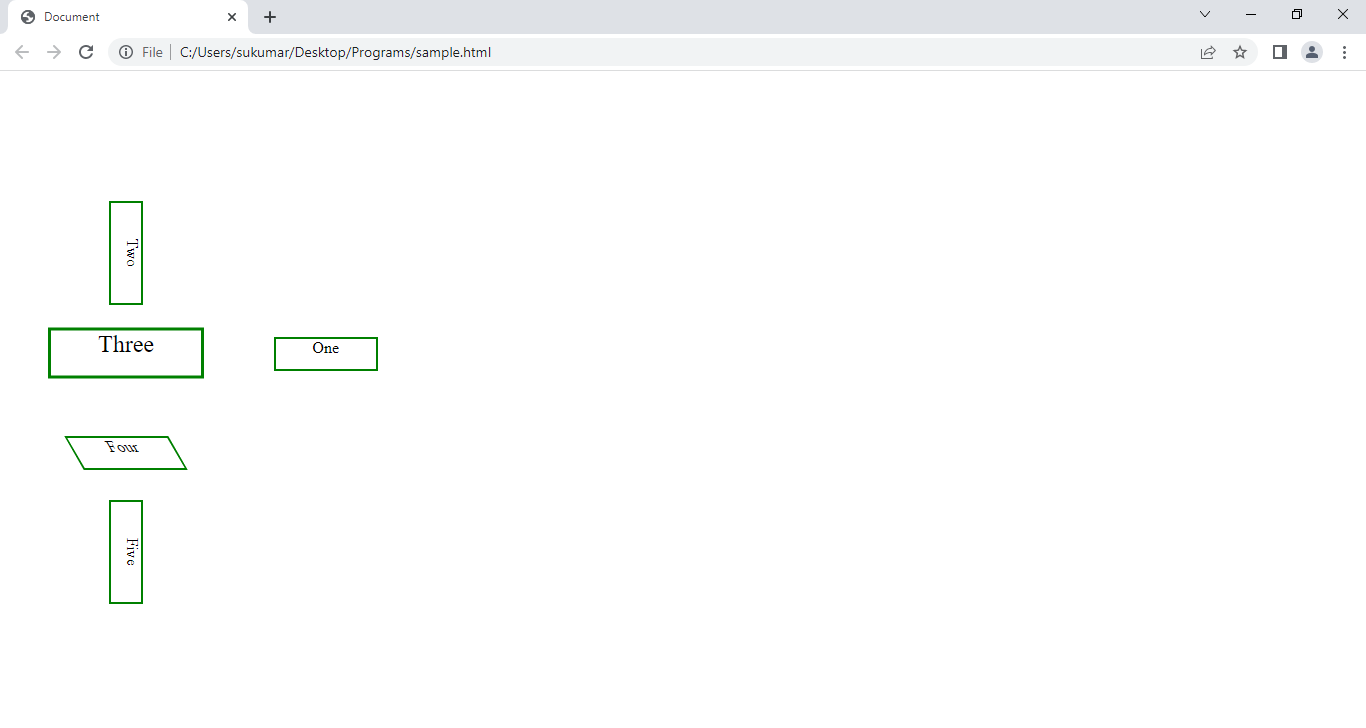
    <div>Three</div>

    <div>Four</div>

    <div>Five</div>

</body>

</html>

****

**7.Transition:**When you change the css property of element, the result is immediately updated(i.e) element immediately changes from old property to new property. We define the transition with css properties using pseudo classes like hover , active,…etc . The element slowely changes from old property to new property over the specified time.

For Example: If you place the cursor over the transition element, element slowely changes from old property to new property. When you remove the cursor from the element, the element immediately changes from new properties to old properties.

Syntax:

Transition:properties duration timing-function delay|initial|inherit.

1. Transition-property: It specifies the css properties to which a transition effect should be applied.

Syntax:

transition-property: none|all|properties| initial|inherit

None: no-property will get transition effect.

All: it is default value. All properties will get transition effect.

Property:Definescommalist of property names and their time duration .

Initial: It set to its default value.

1. Transition-duration: This property specifies how many seconds (or) milliseconds a transition-effect takes to complete.

Syntax:

Transition-duration: time|initial|inherit;

Ex:-Transition-duration:10s;

1. Transition-timing-function:- This property specifies speed of transition-effect.

Syntax:

Transition-timing-function:linear|ease|ease-in|ease-out|ease-in-out|step-start|step-end|steps.

Linear: Tranistion effect with same speed from start to end.

Ease: It is default value. Transition effect with slow start,then fast, and end slowely.

Ease-in:Transition effect with slow start and then fast.

Ease-out: Transition effect with slow end.

Ease-in-out: Transiton effect with slow start and slowend.

Steps(int,start/end): The first parameter specifies the no.of intervals. The duration is divided into no.of intervals.

1. Transition-delay:- This property specifies when transition effect will start.

Syntax: transition-delay: time|initial|Inherit;

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

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

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

    <title>Document</title>

    <style>

        div{

            border:2pxsolidgreen;

            height:30px;

            width:100px;

            margin:10vh;

            text-align: center;

transition: all 13s;

            transition-timing-function: ease-in-out;

            transition-delay: 2s;

            }

            div:hover{

                height:100px;

                width:200px;

                background-color: red;

                transform:rotate(90deg);

           }

   </style>

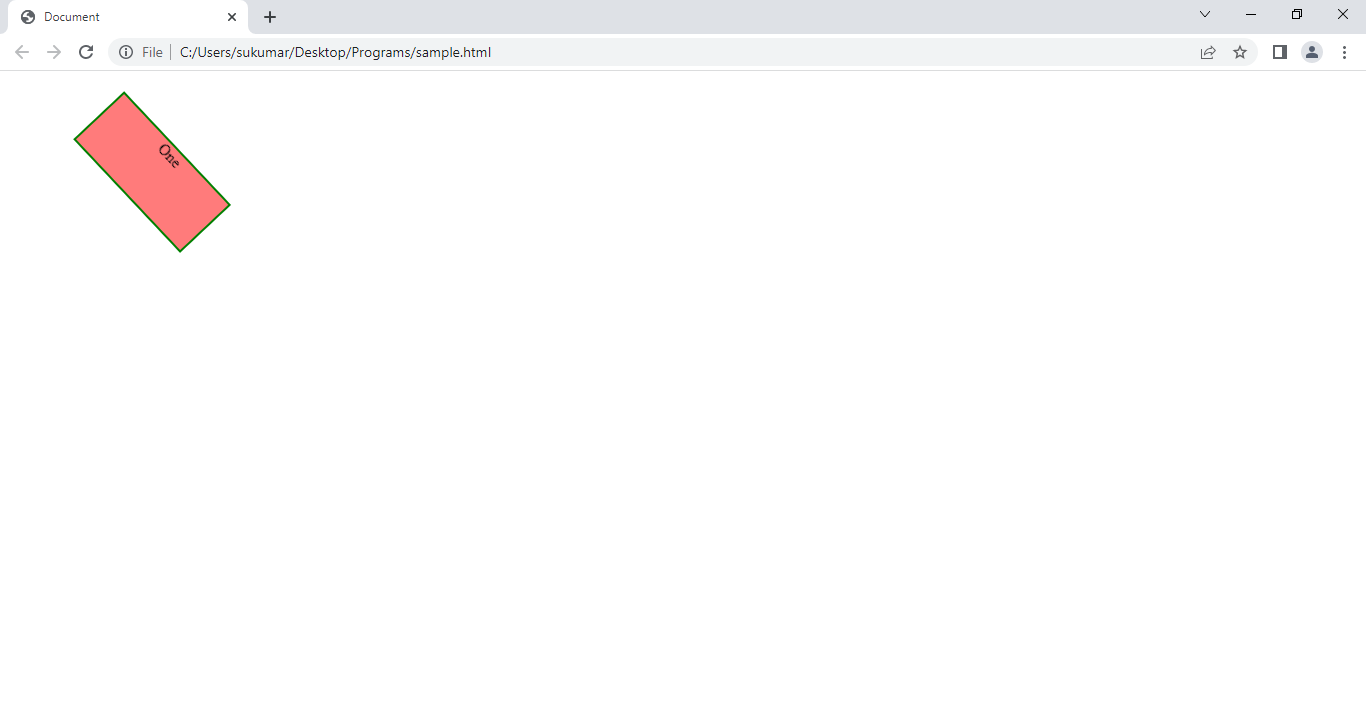
</head>

<body>

    <div>One</div>

</body>

</html>



**8.Animation:**The animation property allows animate the element with out using the js. To use CSS animation, you must first specify some keyframes Rule for the animation. The key frames rule holds css styles.When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

Syntax:

@keyframes animation-name{

from{styles}

To{ styles}

(or)

%{styles}

%{styles}..etc

}

The animation has to bind to element. This binding is done by writing animation properties in element css rule.

1.Animation: It is short-hand property for six animation properties.

Syntax:

animation: name duration timing-function delay iteration-count

Direction fill-mode play-state.

2.Animation-delay: This property makes delay the animation.

Syntax:

Animation-delay:time.

3.animation-iteration-count: It specifies the that how many times animation should be played.

Syntax:

Animation-iteration-count:number|infinite.

4.animation-name: It specifies name of animation. Here we should write the keyframes Rule name.

Syntax:

Animation-name: name.

5.animation-duration: It defines the that in how much time the animation has tobe completed.

Syntax:

Animation-duration: time in seconds|milliseconds.

Default fault value is zero. That means animation should not happens.

6.animation-timing-function: It specifies the speed of animation.

Syntax:

Animation-timing-function: linear|ease|ease-in|ease-out|ease-in-out.

7.animation-direction: It specifies play direction of animation.

Syntax:

Animation-direction: normal|reverse|alternative|alternative-reverse.

Alternate:- plays as normal in even times(2,4,6..etc) and plays reversely in odd times(1,3,5..etc).

Alternate-reverse: plays as normal in odd times(1,3,5.etc) and plays reversely in even times(2,4,6..).

Example: car-Animation Game.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

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

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

    <title>Document</title>

    <style>

        #one{

            height:50vh;

            width:50vw;

            position: relative;

            top:25vh;

            left:25vw;

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

            background-repeat: no-repeat;

            background-size: cover;

            box-sizing: border-box;

            overflow: hidden;

            background-repeat: repeat-x;

        }

        #two{

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

            background-repeat: repeat-x;

            height: 30vh;

            width:500%;

            z-index: 1;

            position: absolute;

            top:24vh;

            left:0px;

            right:0px;

            clip-path: inset(26%0%15%0%);

            box-sizing: border-box;

            animation-name: highway;

            animation-duration: 5s;

            animation-timing-function: linear;

            animation-iteration-count: infinite;

        }

        @keyframeshighway

        {

        100%{transform:translateX(-1000px);}

        }

        #three{

           position: relative;

           top:24vh;

        }

        img{

            height: 10vh;

            width:20vw;

            position: relative;

            top:10vh;

            z-index: 2;

            animation: car 3slinearalternateinfinite;

        }

        @keyframescar{

            50%{ transform:translateY(-20px);}

            100%{transform: translateX(25vw);}

        }

    </style>

</head>

<body>

    <divid="one">

           <divid="two"></div>

            <divid="three">

                <imgsrc="car.png"alt="">

            </div>

    </div>

</body>

</html>

**9. 3D Transformation:**This property allows us to move,rotate,scale&skew the element in 3D space.

1.perspective:- It specifies distance from viewer to object. To activate the 3D space, the element requires perspective. There can be two ways to apply:

1. The first technique is transform property with perspective technique.

Transform:perspective(distance in pixel);

1. The second technique is perspective property.

Perspective: value;

Q)What is diff between first & second techniques?

1. In the first technique , 3D space is active for current&single element.

In the second technique, if u apply on parent element then child elements also shares 3D space.

2.perspective-Origin:

By default the vanish point is at center of element. We can change vanish point position using perspective –origing property.

Syntax:-perspective-origin:x-position y-position z-position|length|percentage;

x-position is left,center or right.

y-position is top,center or bottom.

3.3D transformation functions:- The 3d functions are

3.1) rotateX(deg):- This property rotate element in x-direction in 3D space.

3.2) rotateY(deg):- This property rotate element in y-direction in 3D space.

3.3) rotateZ(deg):- This property rotate element in z-direction in 3D space.

The deg can be positive or negative value.

3.4) translateZ(val):- The element is moved along z-axis. If value is positive, then object comes towards the viewer and object appear as bigger than normal size. If value is negative, then object moves away from the viewer and object appear as smaller than normal size.

3.5) scaleZ(sz):- This property scales element in z-direction.

4.Transform-style: This property specifies how nested elements are rendered in 3D space.

Syntax:- transform-style: flat|preserve-3d|initial|inherit;

Flat:- specified child elements will not preserve its 3D position.

Preserve-3d:- specified child elements will preserve its 3D position.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

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

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

    <title>Document</title>

    <style>

       body{

        background-color: red;

       }

        #container{

            height:200px;

            width:200px;

            border:1pxsolidgreen;

            position:relative;

            top:30vh;

            left:40vw;

            transform-style: preserve-3d;

            transition: all20s;

        }

        .ab{

            animation-name: abcd;

            animation-duration: 20s;

            animation-iteration-count: infinite

        }

        @keyframesabcd{

            100%{transform: rotateX(360deg) rotateY(360deg) rotateZ(360deg);}

        }

        .same{

            height:100%;

            width:100%;

            position: absolute;

        }

        #front{

            z-index: 6;

            background-image: url('front.jpg');

            background-position: 0px0px;

            background-size: 100%100%;

            background-repeat: no-repeat;

            transform:translateZ(100px);

            }

        #back{

            z-index: 1;

            background-image: url('back.jpg');

            background-position: 0px0px;

            background-size:100%100%;

            background-repeat: no-repeat;

            transform: translateZ(-100px);

        }

        #top{

            z-index:2;

            background-image: url('top.jpg');

            background-position: 0px0px;

            background-size: 100%100%;

            background-repeat: no-repeat;

            transform: translateY(100px) rotateX(-90deg);

            }

        #bottom{

            z-index: 3;

            background-image: url('bottom.jpg');

            background-position: 0px0px;

            background-size: 100%100%;

            background-repeat: no-repeat;

            transform: translateY(-100px) rotateX(90deg);

        }

        #left{

            z-index: 4;

            background-image: url('left.jpg');

           background-position: 0px0px;

            background-size: 100%100%;

            background-repeat: no-repeat;

            transform: translateX(-100px) rotateY(90deg);

        }

        #right{

            z-index: 5;

            background-image: url('right.jpg');

            background-position: 0px0px;

            background-size: 100%100%;

            background-repeat: no-repeat;

            transform: translateX(100px) rotateY(-90deg);

        }

    </style>

</head>

<body>

    <divid="container"class="ab">

        <divclass="same"id="front"></div>

        <divclass="same"id="back"></div>

        <divclass="same"id="left"></div>

        <divclass="same"id="right"></div>

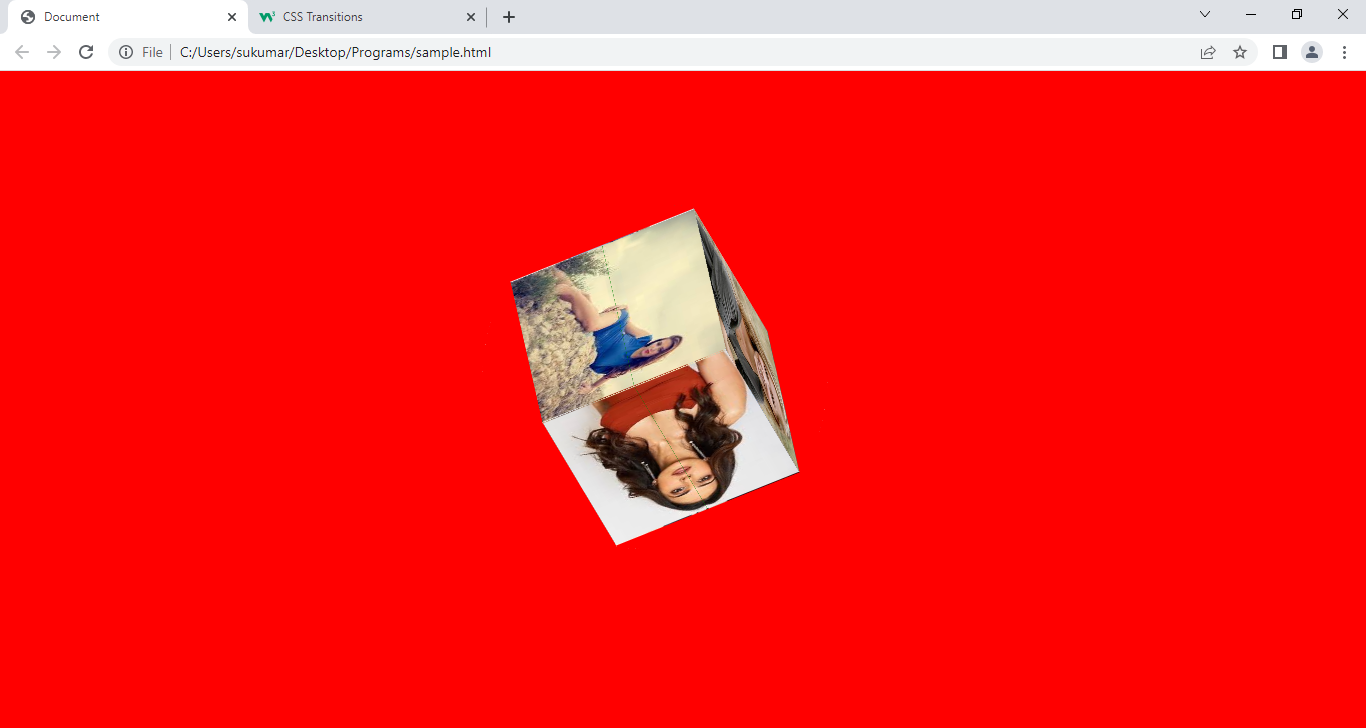
        <divclass="same"id="top"></div>

        <divclass="same"id="bottom"></div>

    </div>

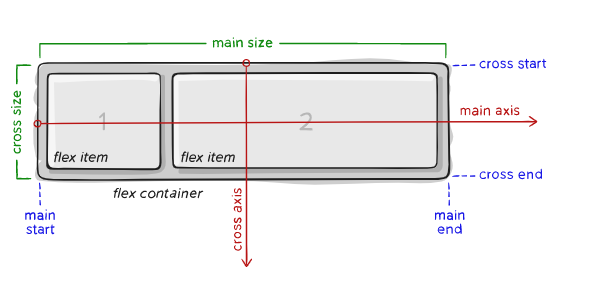
</body>

</html>



**10.Flex:** It is Layout module. It is used to design flexible responsive layout with out using float and position properties. 1D layout can only be created using Flex box module.

10.1) Basic Terminology:-



The flex container contains the items which are called flex items.There are two types of flex container.

1)Inline flex containter.

2)flex container/block flex container.

The flex items will be placed along with main axis or cross axis.The flex items are placed within the container starting from main-start and going to main-end (or) The flex items are placed within the container starting from cross-start and going to cross-end.

10.2) Properties for Flex container:-

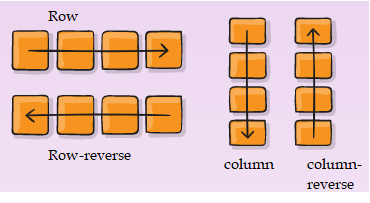
1. Display: It defines flex container; inline or block depending on given value.

Syntax: display:flex|inline-flex;

1. Flex-direction:- It specifies that in which direction , the flex-items will be placed in flex-container.

Syntax:- flex-direction:row|row-reverse|column|column-reverse;

Default value is row.



Row|row-reverse:- The flex-items will be placed in main-axis. The flex-item automatically becomes the inline-block element. It take width as it need and it takes total height of its parent element.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

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

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

  <title></title>

  <style>

   #one{

    height:200px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: row;

    background-color: black;

      }

   #two{

    height:200px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: row-reverse;

    background-color: orange;

      }

    .x{

      background-color: red;

      margin:5px;

      }

  </style>

</head>

<body>

  <h1style="text-align:center">flex-direction:row</h1>

  <hr>

 <divid="one">

  <divclass="x">one</div>

  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

  <h1style="text-align:center">flex-direction:row-reverse</h1>

 <divid="two">

  <divclass="x">one</div>

  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

</body>

</html>



Column|column-reverse:- The flex-items will be place in cross-axis. Each item will become block-level element and occupy full width of flex-container.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

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

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

  <title></title>

  <style>

   #one{

    height:200px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: column;

    background-color: black;

      }

   #two{

    height:200px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: column-reverse;

    background-color: orange;

      }

    .x{

      background-color: red;

      margin:5px;

      }

  </style>

</head>

<body>

  <h1style="text-align:center">flex-direction:column</h1>

  <hr>

 <divid="one">

  <divclass="x">one</div>

  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

  <h1style="text-align:center">flex-direction:column-reverse</h1>

 <divid="two">

  <divclass="x">one</div>

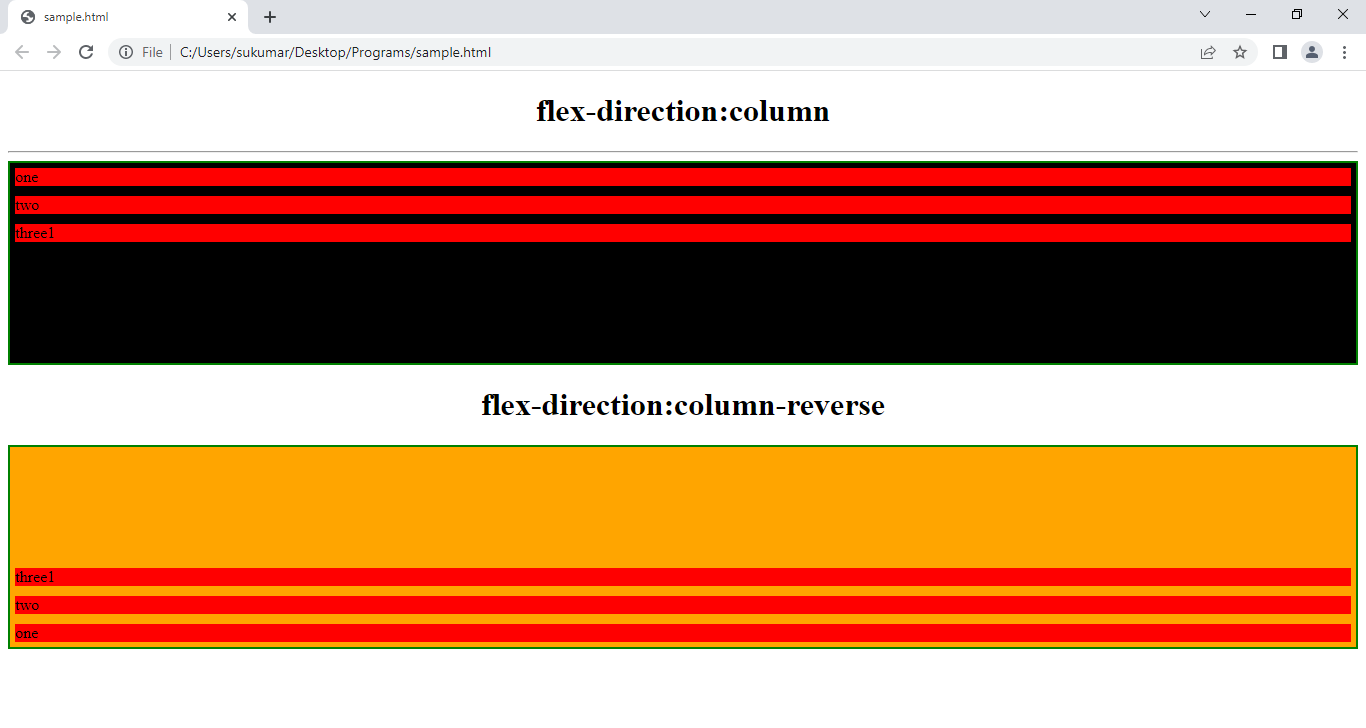
  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

</body>

</html>



c)flex-flow:-

syntax: flex-flow:no-wrap|wrap|wrap-reverse;

nowrap (default): all flex items will be on one line

wrap: flex items will wrap onto multiple lines, from top to bottom.

wrap-reverse: flex items will wrap onto multiple lines from bottom to top.

Example:-

-----------

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: row;

flex-wrap: wrap;

}

#one > div{ height: 30px;width: 150px;background-color: orange;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;}

</style>

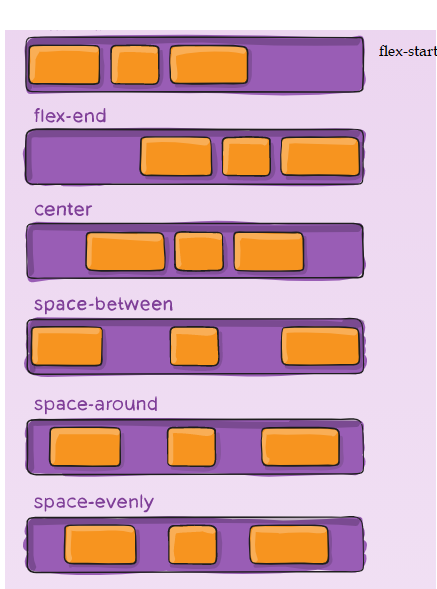
Output:-

----------



d)justify-content:- Through which we can align flex items in flex container.

Syntax:- justify-content: flex-start|flex-end|space-between|space-evenly|space-around;



* flex-start (default): items are packed toward the start of the flex-direction.
* flex-end: items are packed toward the end of the flex-direction.
* start: items are packed toward the start of the writing-mode direction.
* end: items are packed toward the end of the writing-mode direction.
* left: items are packed toward left edge of the container, unless that doesn’t make sense with the flex-direction, then it behaves like start.
* right: items are packed toward right edge of the container, unless that doesn’t make sense with the flex-direction, then it behaves like end.
* center: items are centered along the line
* space-between: items are evenly distributed in the line; first item is on the start line, last item on the end line
* space-around: items are evenly distributed in the line with equal space around them. Note that visually the spaces aren’t equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.
* space-evenly: items are distributed so that the spacing between any two items (and the space to the edges) is equal.

Example:1

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

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

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

  <title></title>

  <style>

   #one{

    height:100px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: row;

    background-color: black;

    justify-content: flex-start;

      }

   #two{

    height:100px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: row;

    background-color: orange;

    justify-content: center;

      }

      #three{

    height:100px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: row;

    background-color: orange;

    justify-content: flex-end;

      }

    .x{

      background-color: red;

      margin:5px;

      }

  </style>

</head>

<body>

  <h1style="text-align:center">flex-direction:row  AND justify-content:flex-start</h1>

  <hr>

 <divid="one">

  <divclass="x">one</div>

  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

  <h1style="text-align:center">flex-direction:row  AND justify-content:center</h1>

 <divid="two">

  <divclass="x">one</div>

  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

  <h1style="text-align:center">flex-direction:row  AND justify-content:flex-end</h1>

 <divid="three">

  <divclass="x">one</div>

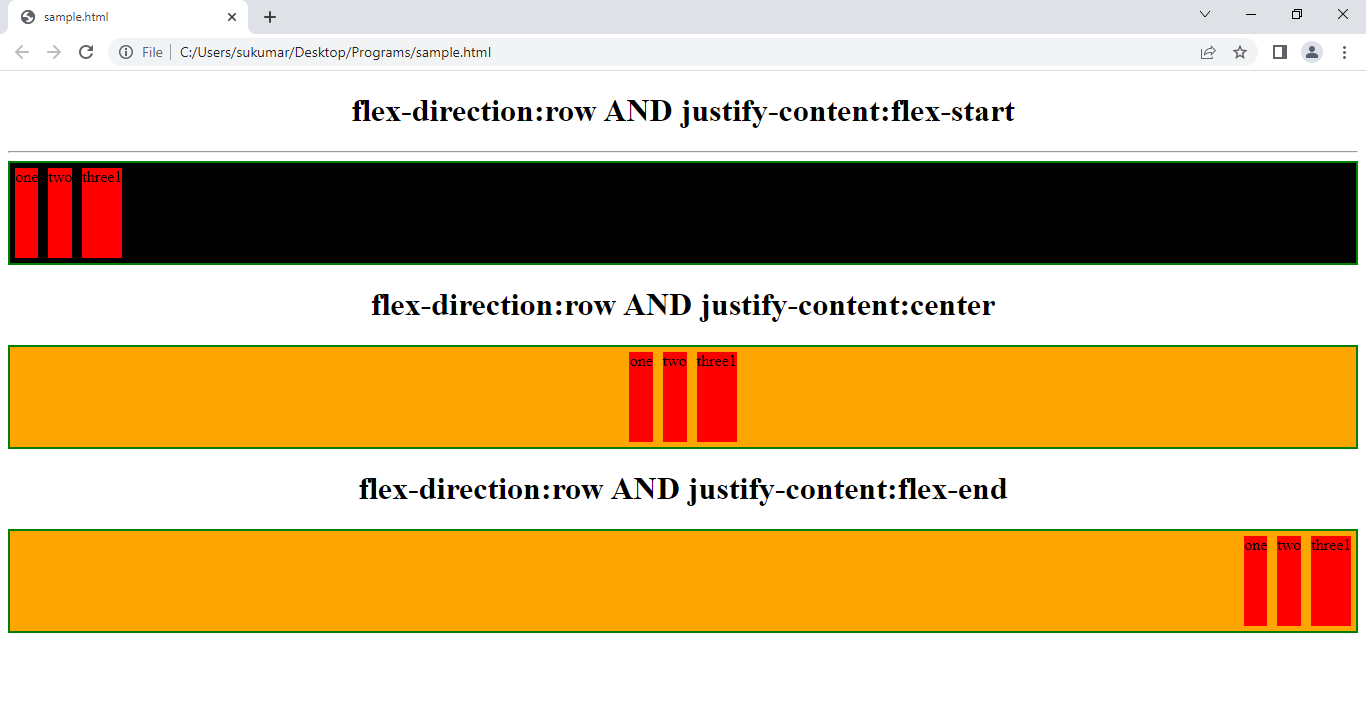
  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

</body>

</html>



Example:2

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

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

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

  <title></title>

  <style>

   #one{

    height:150px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: column;

    background-color: black;

    justify-content: flex-start;

      }

   #two{

    height:100px;5

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: column;

    background-color: orange;

    justify-content: center;

      }

      #three{

    height:150px;

    border: 2pxsolidgreen;

    display: flex;

    flex-direction: column;

    background-color: orange;

    justify-content: flex-end;

      }

    .x{

      background-color: red;

      margin:5px;

      }

  </style>

</head>

<body>

  <h1style="text-align:center">flex-direction:column  AND justify-content:flex-start</h1>

  <hr>

 <divid="one">

  <divclass="x">one</div>

  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

  <h1style="text-align:center">flex-direction:column  AND justify-content:center</h1>

 <divid="two">

  <divclass="x">one</div>

  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

  <h1style="text-align:center">flex-direction:column  AND justify-content:flex-end</h1>

 <divid="three">

  <divclass="x">one</div>

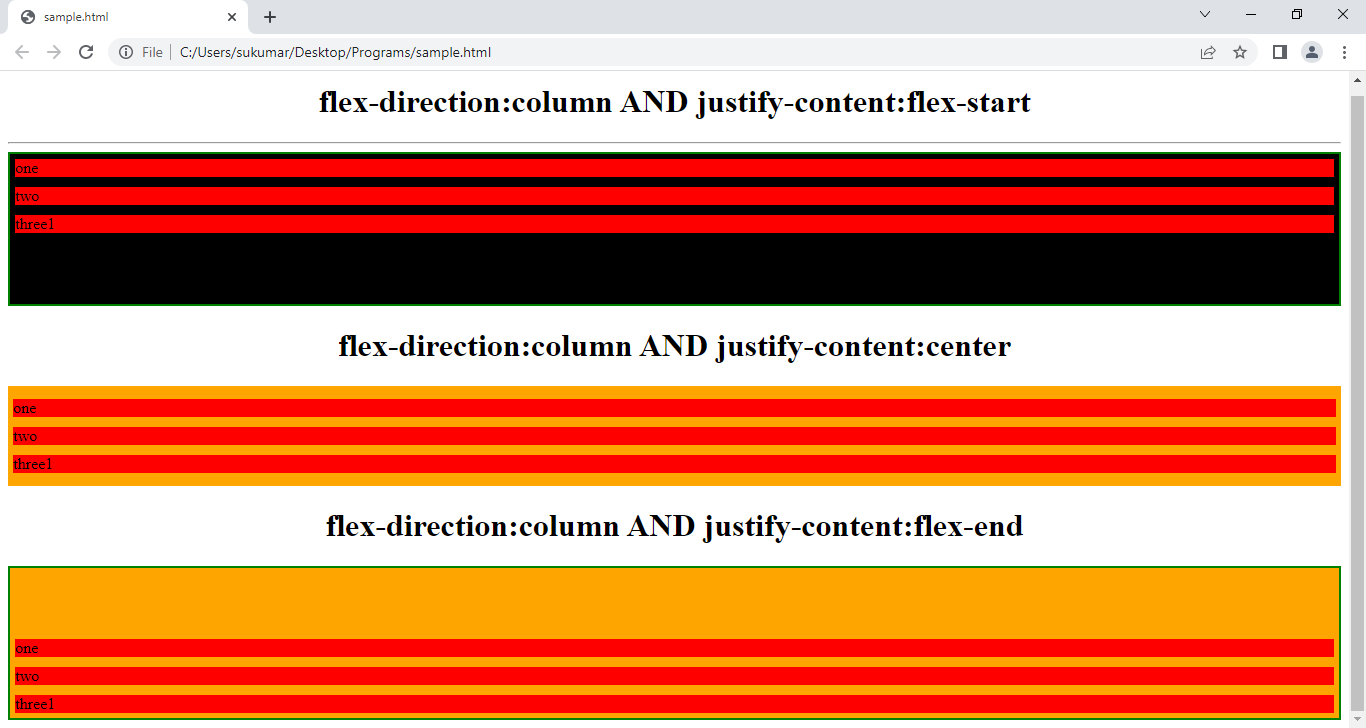
  <divclass="x">two</div>

  <spanclass="x">three1</span>

</div>

</body>

</html>



f)align-items:-

Syntax:- align-items: flex-start|center|flex-end|stretch;

->if flex-direction is row(or)row-reverse , then align-items property makes flex-items as in-line elements and align flex items in cross-axis.

Example1:-

---------------

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: row;

height: 500px;

align-items: flex-start;

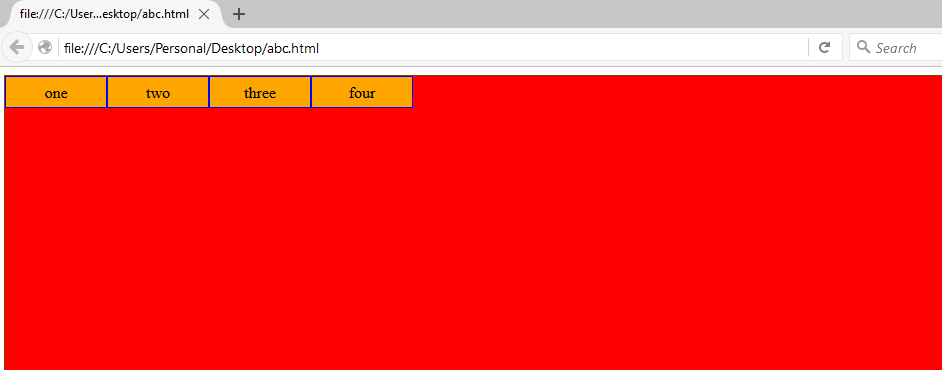
}

#one > div{ background-color: orange;width:100px;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;ali}

</style>

Output:-

-----------



Example:2

---------------

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: row;

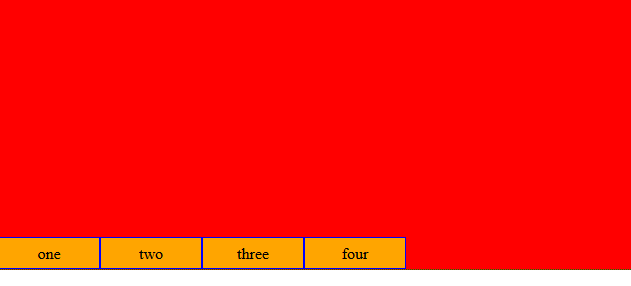
height: 500px;

align-items: flex-end;

}

#one > div{ background-color: orange;width:100px;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;ali}

</style>



Example:3

--------------

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: row;

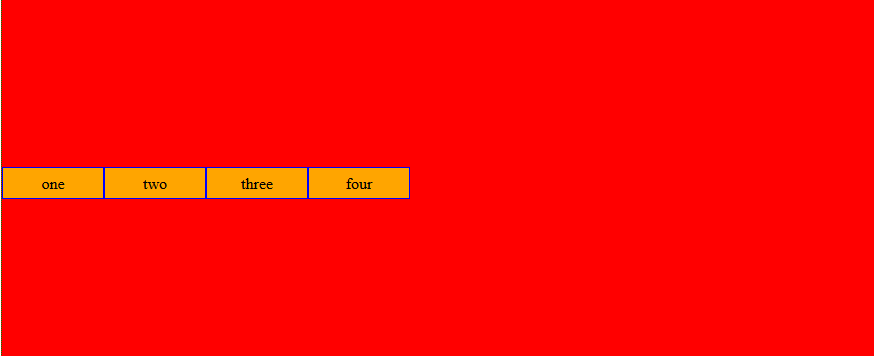
height: 500px;

align-items: center;

}

#one > div{ background-color: orange;width:100px;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;ali}

</style>



->if flex-direction is column(or)column-reverse, then align-items property makes flex-items as inline elements and aligns flex items in main-axis.

Example:1

---------------

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: column;

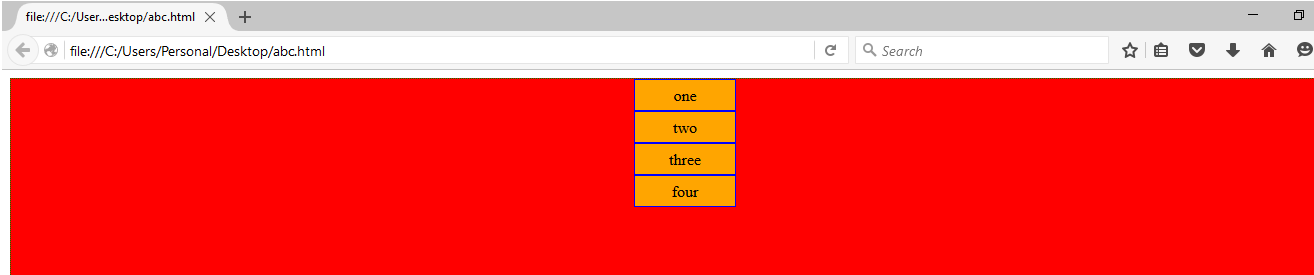
height: 500px;

align-items: center;

}

#one > div{ background-color: orange;width:100px;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;ali}

</style>



Example:2

-------------

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: column;

height: 500px;

align-items:flex-start;

}

#one > div{ background-color: orange;width:100px;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;ali}

</style>



Example:3

-------------

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: column;

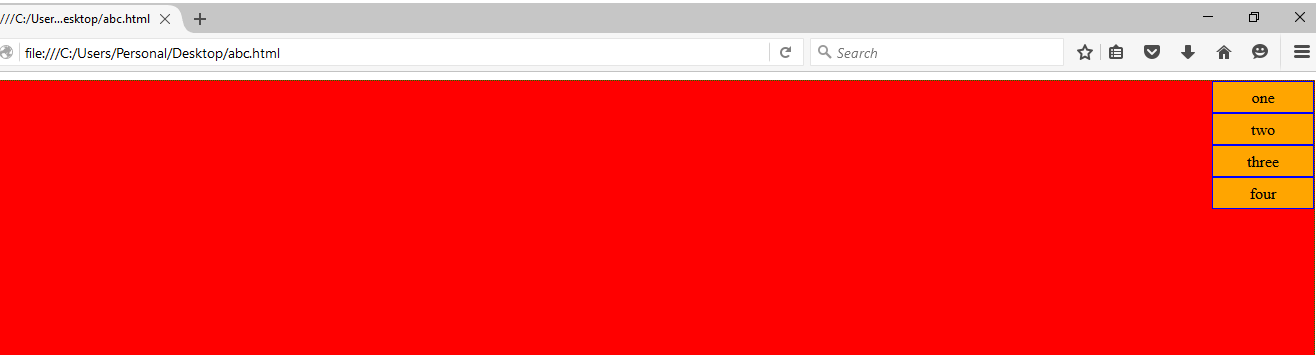
height: 500px;

align-items:flex-end;

}

#one > div{ background-color: orange;width:100px;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;ali}

</style>



g) row-gap:- It defines gap between rows.

h)column-gap:- It defines gap between columns.

i)align-Content: It align content inside the flex container.

10.3)properties for flex-items:-

a)order:- the order property controls the order in which they appear in the flex container.

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: row;

gap: 2px;

}

.two{ width:100px;b

order: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;ali}

</style>

</head>

<body>

<div id='one' >

<div class='two' style="order:4;background-color: wheat;">

one

</div>

<div class='two' style="order:1;background-color: orange;">

two

</div>

<div class='two' style="order:3;background-color:green ;">three</div>

<div class='two' style="order:2;background-color: yellow;">four</div>

</div>

</body>

</html>



b)flex-grow:-If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all children. If one of the children has a value of 2, the remaining space would take up twice as much space as the others (or it will try to, at least).

Syntax:flex-grow: +v/-v|0;

If value is 0; the flex-items will not be grown.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: row;

}

.two{ width:100px;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;flex-grow: 1;}

</style>

</head>

<body>

<div id='one' >

<div class='two' style="order:4;background-color: wheat;">

one

</div>

<div class='two' style="order:1;background-color: orange;">

two

</div>

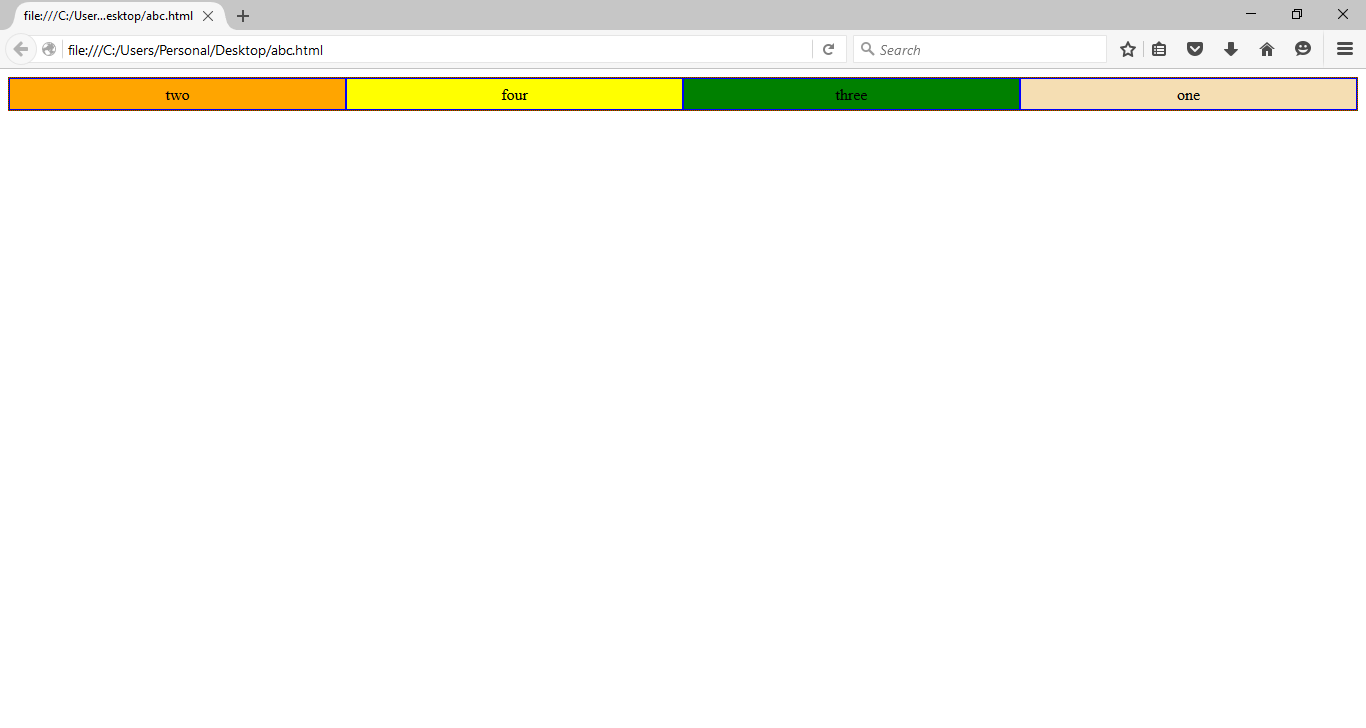
<div class='two' style="order:3;background-color:green ;">three</div>

<div class='two' style="order:2;background-color: yellow;">four</div>

</div>

</body>

</html>



Ex:-2

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<style type="text/css">

#one{

background-color: red;

border: 1px dotted green;

display: flex;

flex-direction: row;

gap: 2px;

}

.two{ width:100px;border: 1px solid blue; text-align: center;font-size: 1em;line-height: 30px;}

</style>

</head>

<body>

<div id='one' >

<div class='two' style="order:4;background-color: wheat;flex-grow:1">

one

</div>

<div class='two' style="order:1;background-color: orange;flex-grow: 2;">

two

</div>

<div class='two' style="order:3;background-color:green ;">three</div>

<div class='two' style="order:2;background-color: yellow;">four</div>

</div>

</body>

</html>

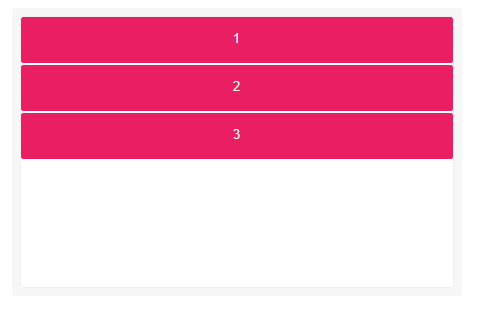
**11.Grid:**It is a powerful tool that allows for two-dimensional layouts to be created on the web.

i) grid:Create a grid container by setting the display property with a value of grid or inline-grid.

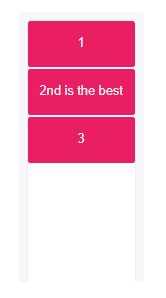
**Syntax:**

Display: grid|inline-grid;

1. display:grid :- Grid items are placed in rows by default and span the full width of grid container.



1. Display:inline-grid :-



ii.grid-template-columns: This property can also be used to specify the width of columns.

Syntax:

Grid-template-columns: v1[ v2 v3 …etc.]

If u specify the n values, then row will have n columns and each value becomes width of appropriate column.

iii. grid-template-rows: It is used to specify the height of row[s].

syntax: grid-template-rows: v1 [v2 v3 …etc.]

iv. justify-content:- it is used to align the whole grid along the x-axes inside the container.

Syntax:

Justify-content:space-between|start|end|center|space-around-space-even.

v. align-items: It is used to align the whole grid along the y-axes inside the container.

Syntax:

Align-items:space-between|start|end|center|space-around-space-even.

The following are grid-item properties:

i.grid-column: It defines on which column flex-item to be started and on which column flex-item to be ended.

Syntax:1

Grid-column:start-number/end-number; // column starts on start-number and end before the end-number.

Syntax:2

Grid-column:start-number /span n; // column starts on start number and span n columns.

ii.grid-row: It defines on which row flex-item to be started and on which row flex-item to be ended.

Syntax:1

Grid-row:start-number/end-number; // row starts on start-number and end before the end-number.

Syntax:2

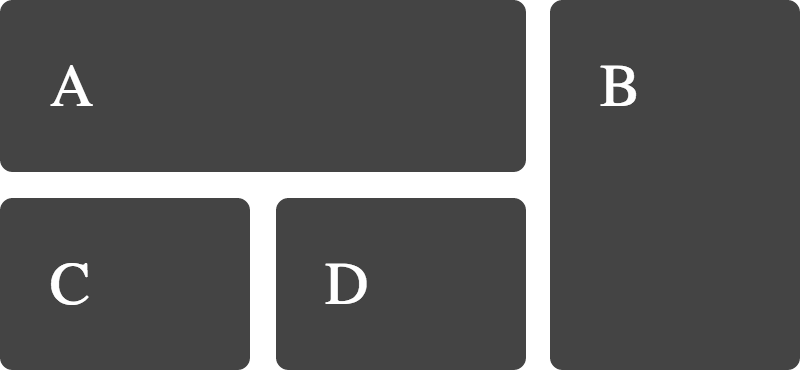
Grid-row:start-number /span n; // row starts on start number and span n columns.

iii.grid-area: It is used to name to grid-item.

Syntax:

Grid-area:name;

Example:1



<!DOCTYPEhtml>

<html>

<head>

<style>

.grid-container {

  display: grid;

  grid-template-columns: autoauto;

  background-color: #2196F3;

  padding: 10px;

}

.grid-item {

  background-color: rgba(255, 255, 255, 0.8);

  border: 1pxsolidrgba(0, 0, 0, 0.8);

  padding: 20px;

  font-size: 30px;

  text-align: center;

}

#one{

   grid-column-start: 1;

   grid-column-end:3;

}

#two{

     grid-row-start:1;

     grid-row-end:3;

     grid-column-start:3;

     grid-column-end:4;

}

</style>

</head>

<body>

<h1>Grid Elements</h1>

<p>A Grid Layout must have a parent element with the <em>display</em> property set to <em>grid</em> or <em>inline-grid</em>.</p>

<p>Direct child element(s) of the grid container automatically becomes grid items.</p>

<divclass="grid-container">

  <divclass="grid-item"id="one">1</div>

  <divclass="grid-item"id="two">2</div>

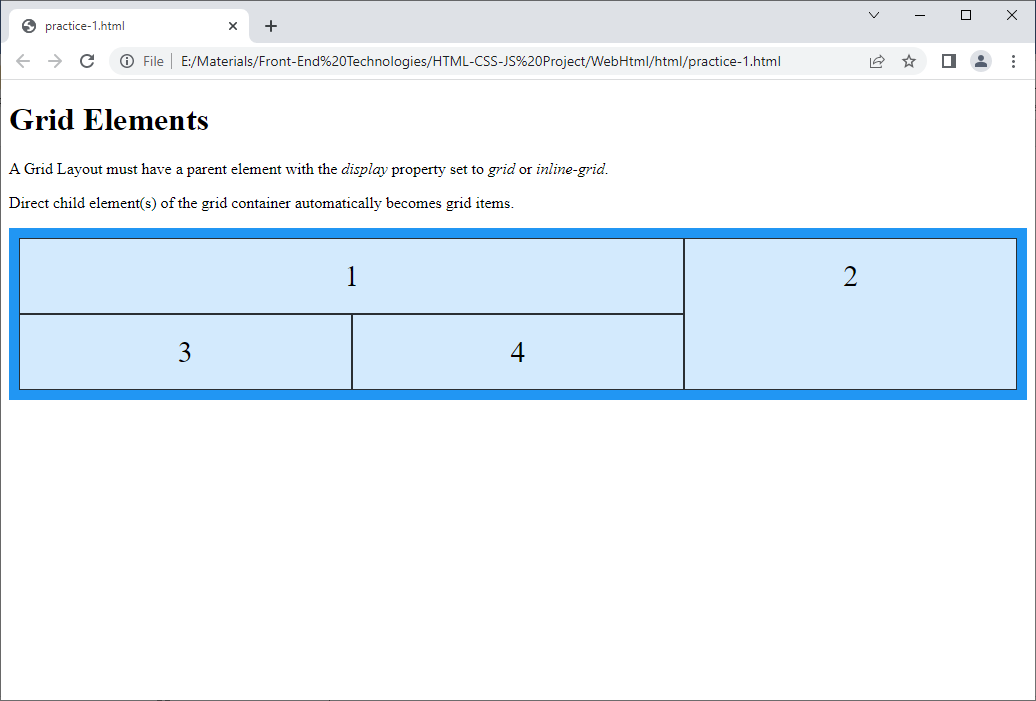
  <divclass="grid-item">3</div>

  <divclass="grid-item">4</div>

</div>

</body>

</html>



**12.Media:**Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website/webpage, to make it look good on all devices (desktops, tablets, and phones).

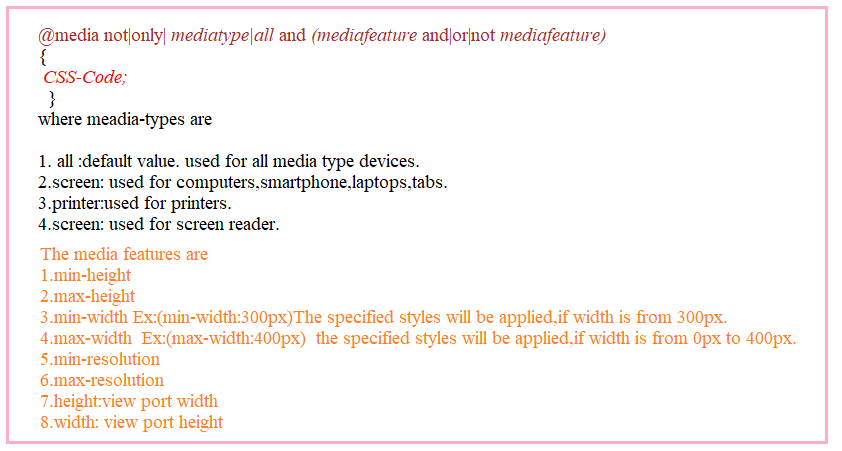
To create responsive web page, add following tag to all web pages in website.



This will set the viewport of your page, which will give the browser instructions on how to control the page's dimensions and scaling.

The @media rule is used in media queries to apply different styles for different media types/devices.

Syntax:-



**Note:-** There is a element to which we should decorate in all types of devices similarly. The style sheet which have decoration code is written outside the all @media queries in .css file.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

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

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

    <title>Document</title>

</head>

<style>

    @mediascreen and (max-width:300px) and (max-height:300px)

    {

       body{

        background-color: red;

       }

    }

    @mediascreen and (min-width:301px) and (min-height:301px)and (max-width:600px) and (max-height:600px) {

        body{

        background-color: blue;

       }

    }

</style>

<body>

</body>

</html>

**13.Gradients:** The gradient is an image which is made up of smooth transition between two or more colors. There are 3 types of gradiants.

i.Linear Gradient.

ii.Radial Gradient.

iii.Conic Gradient.

13.1.Linear Gradient: It is a function.

Syntax:

Background-image: linear-gradient(direction,col1,col2, [col…3])

Direction: The directions are to bottom, to right, to left, to top.

13.1.1. Linear Gradient-repeat: It is a function.

Syntax:

Background-image:linear-gradient-repeat( col1[%],col2 [%], […etc]);

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

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

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

  <title>Document</title>

  <style>

    div:first-child{

      height:200px;

      width:400px;

      border: 2pxsolidblue;

      background-image: linear-gradient(to top,white,black);

    }

    div:nth-child(3){

      height:200px;

      width:400px;

      border: 2pxsolidblue;

      background-image: repeating-linear-gradient(to right,blue,orange,green10%);

    }

  </style>

</head>

<body>

  <div>

    This is sukumar.

</div>

<p> linear Gradient Repetation</p>

<div></div>

</body>

</html>

