CSS (CASCADING STYLE SHEET)

- `STYLE` means changing apperance bt using various properties like color , border etc.. `CASCADING` means suppose you created a paragraph and you have assign different bg-color properties one with blue color another with yellow and another with pink now which color to implement this is reffered to a cascading.
- `Ways to implement CSS` :- INLINE , INTERNAL AND EXTERNAL.

inline>internal (internal and external having priority according to ordering) you can mark `!important` to change the priorities.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <!--EXTERNAL(Most preffered)-->
    <link rel="stylesheet" href="styles.css">
    <!--Internal-->
    <style>
           h1{
                background-color: aqua;
            }
            p{
                background-color: pink !important;
            }
    </style>
</head>
<body>
     <h1>Welcome my friend!</h1>
    <!--INLINE CSS-->
    <p style="background-color: beige;">Lorem ipsum, dolor sit amet consectetur adipisicing elit.
        Excepturi fugiat delectus ipsum et voluptates nulla quas natus, sapiente,
         cupiditate aliquid laudantium distinctio sint ex molestias officiis eius optio
    </p>

</body>
</html>
```

- **SELECTORS:-** it is a way by which we can easily target one or multiple html elements and apply css on them. `(Element , id, class and group selectors)`

```
/* Styles.css(EXTERNAL CSS) */
/* element selector */
h1{
    font-size: small;
    text-align: center;
}

/* id selector */
#para1{
    background-color: blueviolet
}

/* class selector */
.parag{
    background-color: aquamarine;
}

/* group selector */

p#para2 {
```

```
    color: greenyellow;
}
```

- **BASIC FONT PROPERTIES:-**

```
p{
    font-style: italic;
    font-size: 15px;
    font-family: 'Times New Roman', Times, serif; /*Fallback system */
    font-weight: 600;
    line-height: 30px;
}
/*External Fonts*/
https://fonts.google.com/specimen/Oswald
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Oswald&display=swap" rel="stylesheet">

p{
  font-family: 'Oswald', sans-serif;
}
```

- **COLORS IN CSS:-** **(named , rgb , hex , hsl)**

```
#para1{
    background-color: yellow;
}
#para2{
    background-color: rgb(100, 109, 345,50); /*Fourth value is optional which denotes transparency.*/
}
#para3{
    background-color: #ffab23;
}
#para4{
   background-color:  hsl(0,100%,50%);
}
```

- **BORDERS IN CSS:-(ADDS BORDER AROUND YOUR CONTENT)**

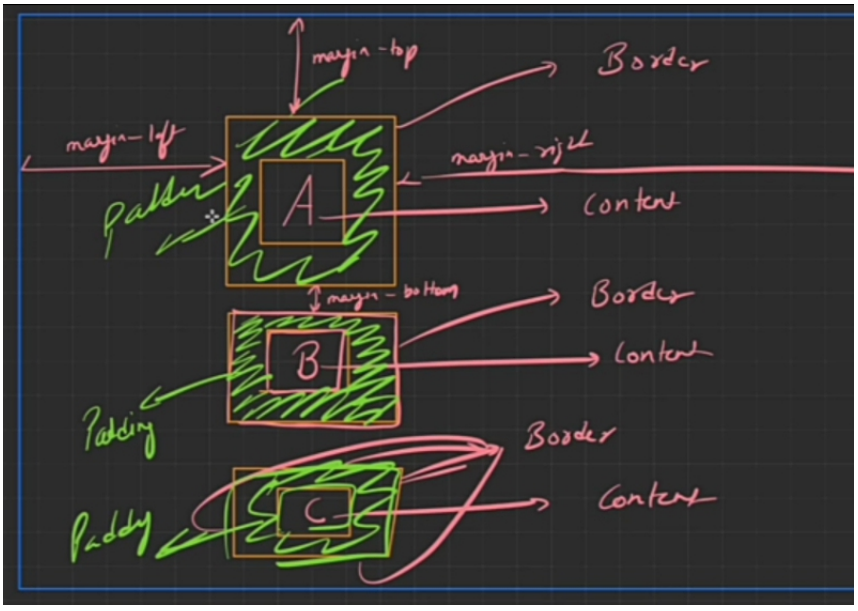
```
h1{
    background-color: aquamarine;
    /* border-style: solid;
    border-width: 2px; */
    border-color:solid brown 2px;
    /* border-radius: 4px; */
    /* border-top-left-radius: 3px; */
    border: solid black 3px;
    /* height: 200px; */
}

.firstPara{
    background-color: aqua;
    border-bottom: solid black 1px;
}
.secondpara{
    background-color: bisque;
}
.thirdpara{
```

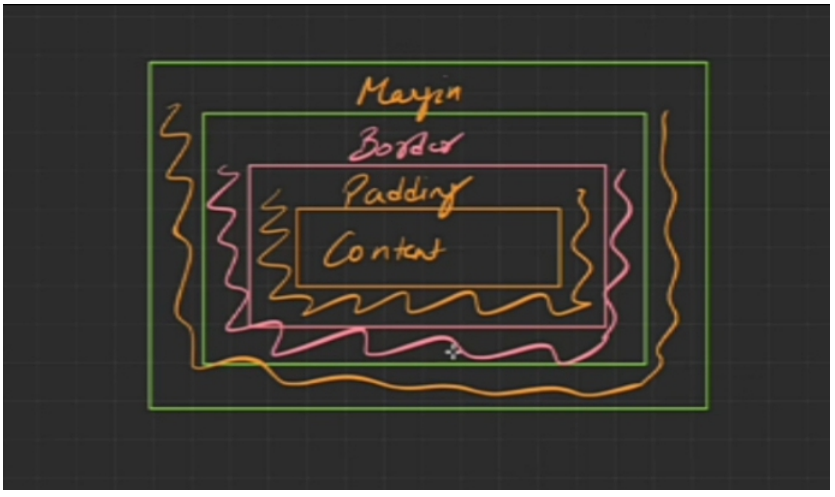
```
    background-color: blueviolet;
}
```

- **BOX MODEL:-** All of your content in html is rendered inside a rectangular box. A box model is essentially a box that wraps around html element, It consists of : `margins,border,padding and actual content.`

BEST PRACTICES: (BY DEFAULT INSIDE UNIVERSAL SELECTOR MARGINS AND PADDINGS ARE MADE "0".)

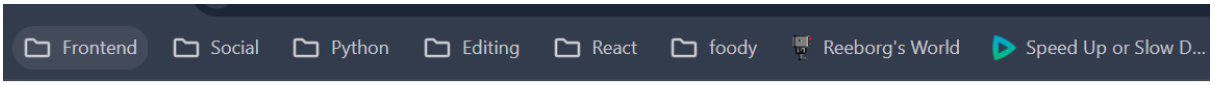
```
*{
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}
/* if border-box property is applied and then margin,padding etc.. are added then these
margins and padding gets included with the elements height and width i.e no extra space is
captured */
/* Suppose you created a box of size 100px is border-box property is not applied than after
adding padding of(20px) the size of full box becomes 120px */
```






```
.firstPara{
    background-color: aqua;
    border-bottom: solid black 1px;
```

```
    border-radius: 5px;
    width: 350px;
    /* padding: 30px 40px 55px 44px; */
    padding: 40px;
    /* margin: 30px,20px,23px,23px; */
    margin: 50px;
}
```



Lorem ipsum dolor sit amet consectetur adipisicing elit. Iste nulla aliquam nisi molestias, fugit at fuga quo iusto repellat, magni amet nam! Perspiciatis culpa ipsam, error amet esse quasi iure expedita perferendis aperiam, pariatur ut excepturi voluptas voluptatem quis accusantium eos, quae tenetur officiis quidem sequi! Quasi velit aliquid, eos deleniti ducimus soluta error repellendus.

- **DISPLAY PROPERTY:-** Defines how to display or render a particular element on the viewport.

`(inline,block,inline-block)`

Block-level elements:(comes in new line , custom-width height , margin and padding can be applied)

1. `<div>` - Defines a division or a section in an HTML document.
2. `<p>` - Represents a paragraph.
3. `<h1>` to `<h6>` - Define header tags, where `<h1>` is the highest level and `<h6>` is the lowest.
4. `<ul>` - Defines an unordered list.
5. `<ol>` - Defines an ordered list.
6. `<li>` - Represents a list item within `<ul>` or `<ol>`.
7. `<table>` - Defines a table.
8. `<tr>` - Represents a table row.
9. `<td>` - Represents a table cell.
10. `<form>` - Defines an HTML form for user input.

Inline elements:(tries to come in same line if space is available , no custom width and height ,no custom margins but custom paddings can be applied.)

1. `<span>` - Defines a small, inline container that can be used to apply styles or scripting.
2. `<a>` - Creates a hyperlink.
3. `<strong>` - Represents strong importance, typically displayed as bold text.
4. `<em>` - Represents emphasized text, typically displayed as italicized text.
5. `<img>` - Embeds an image in the HTML document.
6. `<br>` - Represents a line break.
7. `<i>` - Represents italic text.
8. `<u>` - Represents underlined text.
9. `<code>` - Represents a fragment of computer code.

10. `<sup>` - Represents superscript text.

INLINE-BLOCK BEHAVES LIKE INLINE BUT WE CAN SET CUSTOM HEIGHT AND WIDTH PROPERTY.

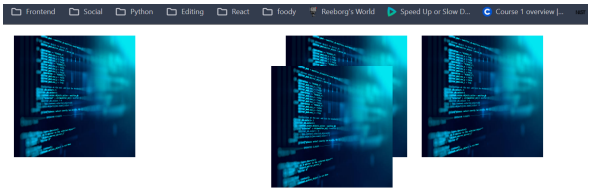
```
span{
    display: block;
/* display: inline-block;
    display: inline; */
}
```

- **POSITIONING IN CSS:- Defines where and how to place different Elements i.e how much far it should be from top or how much far it should be from left etc.... `(static,relative,fixed,absolute,sticky)` By default it is static i.e we cant move the element left ,right,top or bottom.**

1. **RELATIVE:-** Position element w.r.t the cuurent window by using left,right,top and bottom.

"**GAP**" is still maintained even after moving the element from its position.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        img{
            height: 200px;
            width: 200px;
            margin: 10px;
        }

        .img2{
            position: relative;
            top: 50px;
            left: 200px;
        }
    </style>
</head>
<body>
    <div>
        <img src="istockphoto-537331500-612x612.jpg" alt="" class="img1">
        <img src="istockphoto-537331500-612x612.jpg" alt="" class="img2">
        <img src="istockphoto-537331500-612x612.jpg" alt="" class="img3">
        <img src="istockphoto-537331500-612x612.jpg" alt="" class="img4">
    </div>
</body>
</html>
```

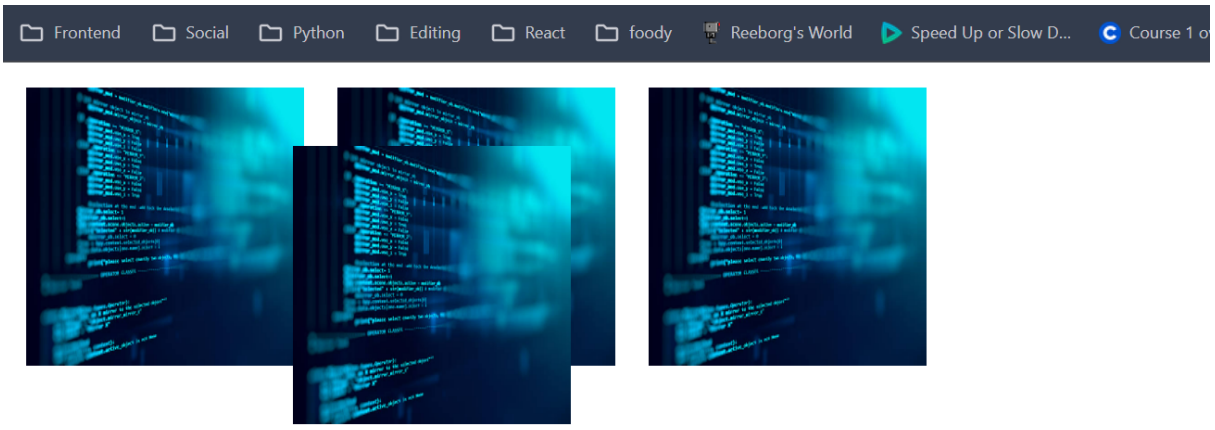


2. **ABSOLUTE:-** Generally used or preffered when we want overlapping. Elements are positioned relative to closest position ancestors. for eg... in above code img2 will be positioned w.r.t div.

"Gap is not maintained."

```
.img2{
            position: absolute;
```

```
            top: 50px;
            left: 200px;
        }
```



- **FIXED:-** The element stays where it is even after scrolling the window. It basically fixes the element at its position.
- **STICKY :-** THE element remains fixed on the particular position until the element is in its parent container. (in the below image the image will be fixed only inside div1)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        img{
            height: 200px;
            width: 200px;
            margin: 10px;
        }
        .div1{
            background-color: aquamarine;
            height: 700px;
        }
        .img2{
            position: sticky;
            top: 50px;
            left: 200px;
        }
        .div2{
            background-color: bisque;
            height: 600px;
        }
    </style>
</head>
<body>
    <div class="div1">
        <img src="istockphoto-537331500-612x612.jpg" alt="" class="img1">
        <img src="istockphoto-537331500-612x612.jpg" alt="" class="img2">
        <img src="istockphoto-537331500-612x612.jpg" alt="" class="img3">
```

```
        <img src="istockphoto-537331500-612x612.jpg" alt="" class="img4">
    </div>
    <div class="div2">
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.
            Asperiores esse fuga provident in nam distinctio at, repudiandae voluptas
            rem doloremque? Nesciunt amet in temporibus fugit, unde nemo esse a reprehenderit
             voluptate blanditiis eligendi incidunt cum minima iste doloribus alias voluptas explicabo
             ullam sed! Veniam, eligendi! Ipsa culpa cum omnis perspiciatis?
        </p>
    </div>
</body>
</html>
```

- **DIFFERENT CSS UNITS:-** The units are categorized into two types :-

1. Absolute units(having fixed values):- `px,pt,pc`

1px = 1/96 of inch =0.26mm , The more the value of ppi(pixels per inch)the more will be resolution of the image.

Logical pixels are CSS units that are resolution-independent, while physical pixels correspond to the actual device's screen pixels.

2. **RELATIVE UNITS:-** whole value is relative to other elements. `(%,vh,vw,rem,em)`

“50% means 50 percent of the parent”

⭐**em:-** if font-size is by default 16px . Then 1 em = 16px.

⭐rem:- It is defined according to <html> element.

in the below code fontsize of paragraph will be equal to 2*4(parent)=8px(1 rem = 4px)

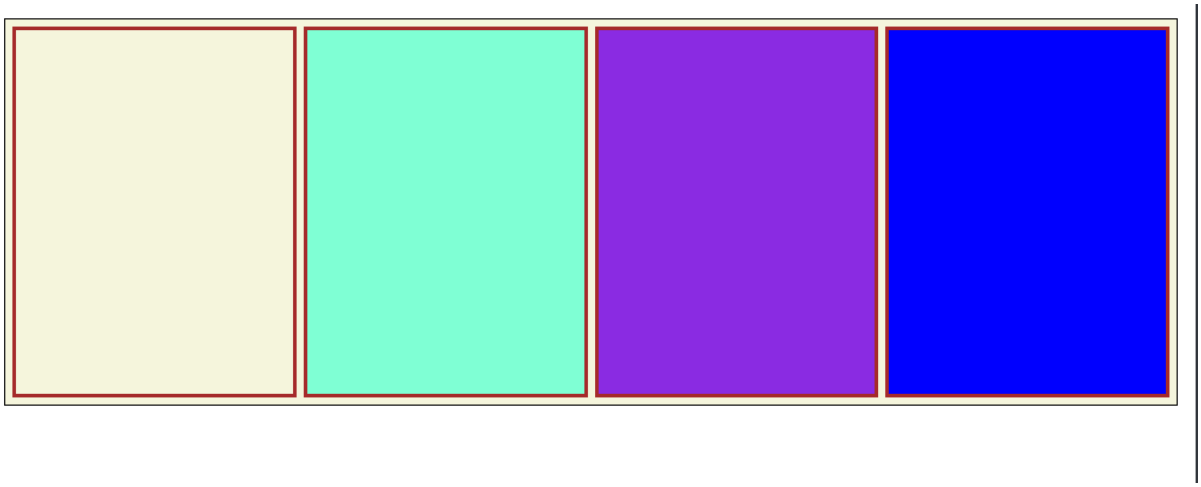
```
.div5{
        font-size:4px;
    }

    p{
        font-size: 2rem;
    }
    <div class="div3">
        <p>
            Lorem ipsum dolor sit amet.
        </p>
    </div>
```

- **FLEX-BOX:-** 1-D Layout Method. Different ways of positioning elements is reffered to as Layout. One of the powerful way of creating a layout is using flexbox.

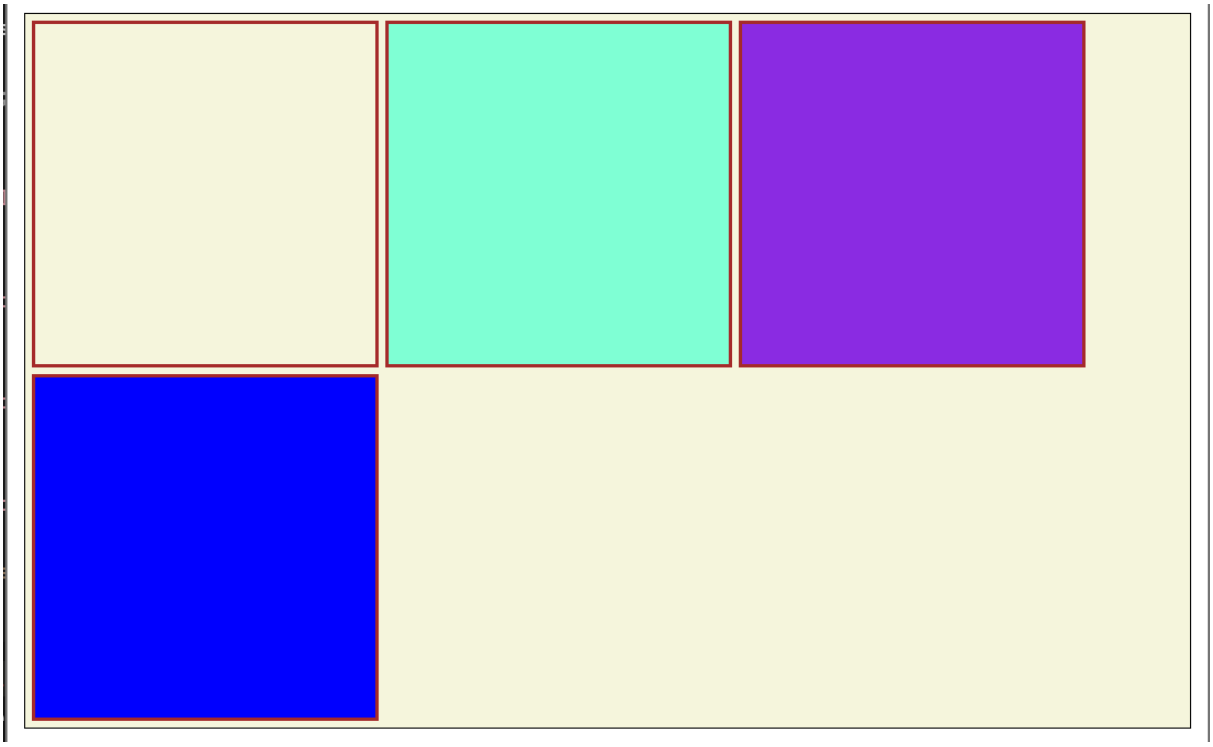
DISPLAY PROPERTY AND FLEX-DIRECTION:-➖

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
        <div class="container">
            <div class="box" id="box1"></div>
            <div class="box" id="box2"></div>
            <div class="box" id="box3"></div>
            <div class="box" id="box4"></div>
        </div>

</body>
</html>
```

```css
.container{
    background-color: beige;
    border: 1px solid black;
    margin: 2px;
    padding: 2px;
    display: flex;
}
.box{
    height: 200px;
    width: 200px;
    border: 2px solid brown;
    margin: 2px;
    padding: 2px;
}j
#box1{
   background-color: aqua;
}
#box2{
    background-color: aquamarine;
}
#box3{
    background-color: blueviolet;
}
#box4{
    background-color: blue;
}
/* by default flex direction is row */
```

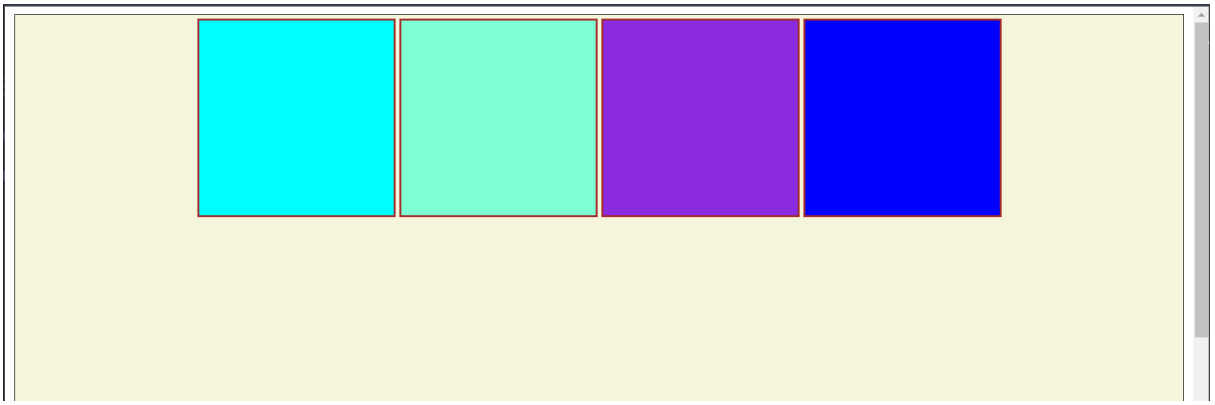


FLEX-WRAP:- `flex-wrap: wrap;` `by default it is no-wrap other properties include` `wrap-reverse` **Even if you reduce the size of screen, the size of div (height 200px) will remain same and the div's will start moving into the next line and the div's will start squeezing or reducing in size when their is no space is available .**



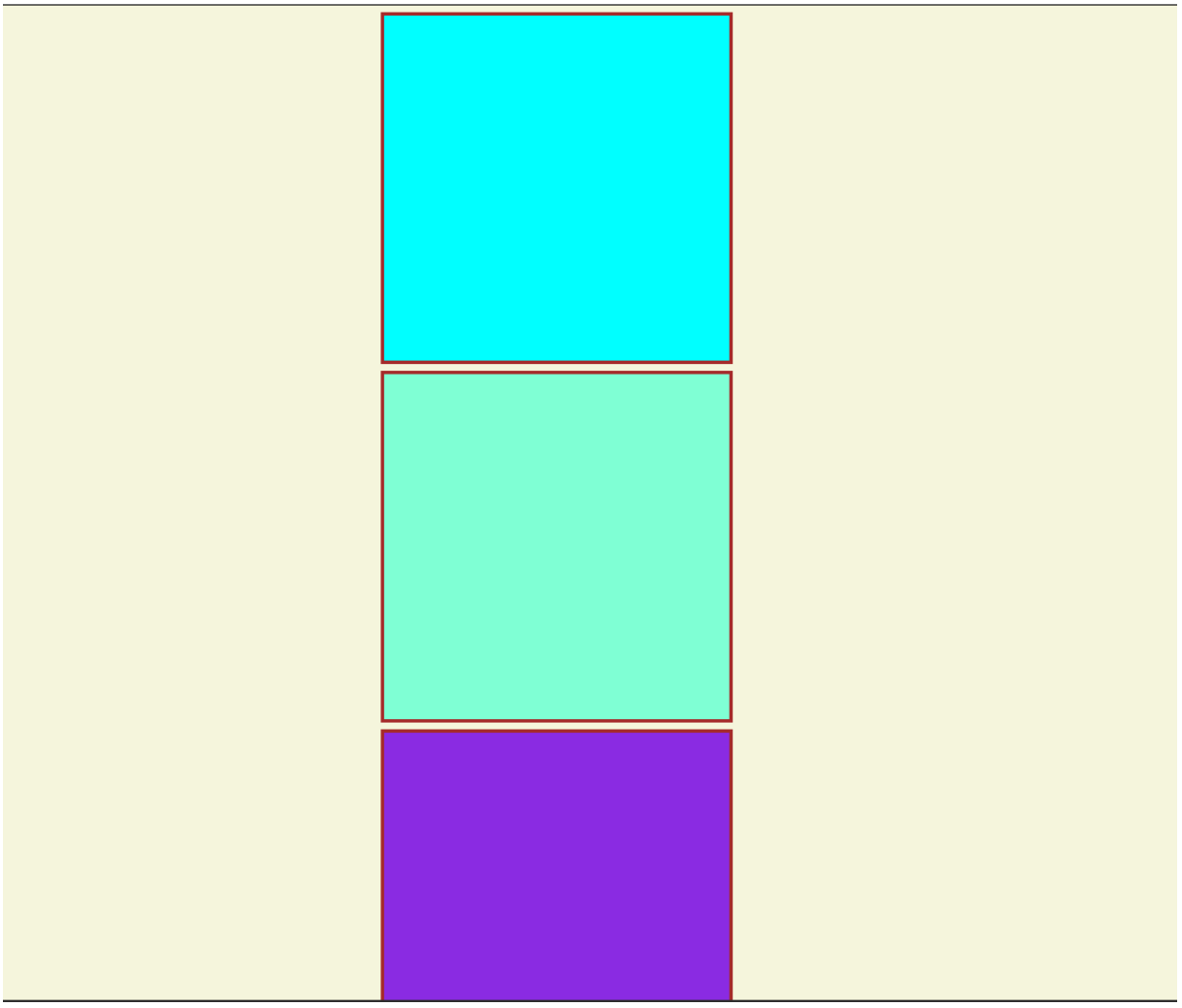
JUSTIFY-CONTENT `(start,end,center,space-around,space-evenly,space-between)` **:-** property that Places content according to main axis i.e in case flex-direction is row the the main axis is horizontal and vice-versa.

```
    display: flex;
    flex-direction: row;
    justify-content: center;
```



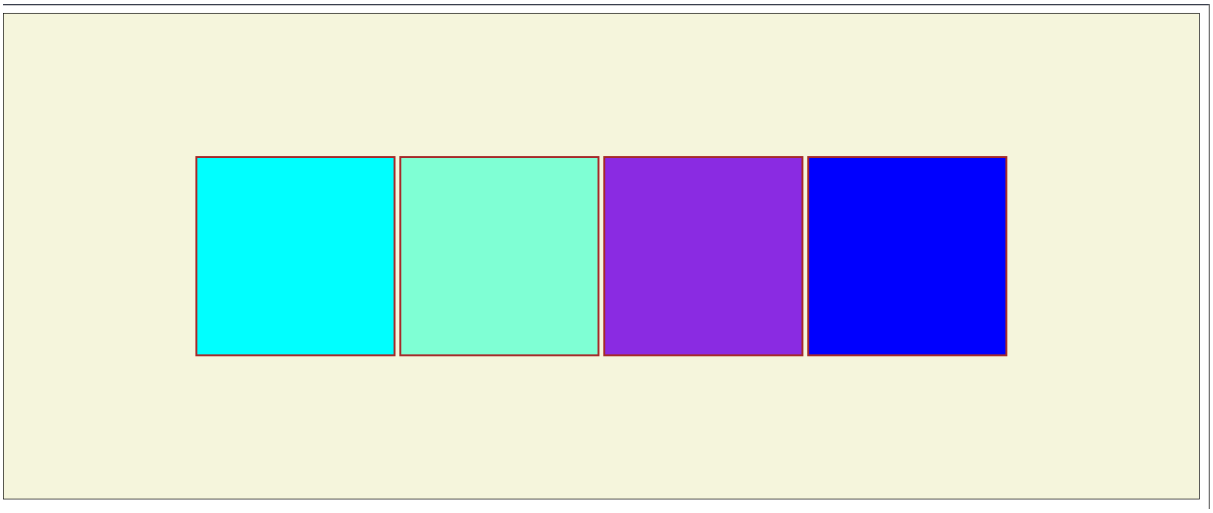
ALIGN-ITEMS `(start,end,center,space-around,space-evenly,space-between,stretch) by default it is stretch` **:- This property is similar to justify-content property the only difference is by default it works corresponding to vertical axis or cross axis.**

```
    display: flex;
    flex-direction: column;
    align-items: center;
```

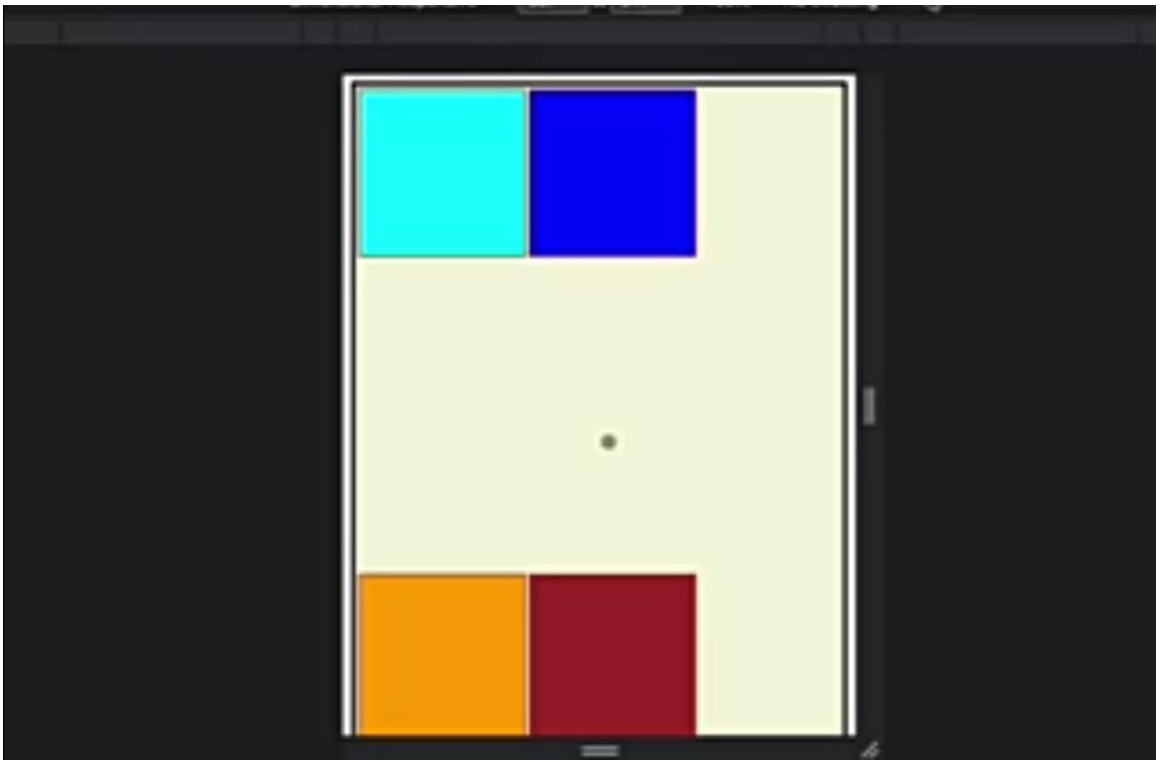


```
    display: flex;
    flex-direction: row;
```

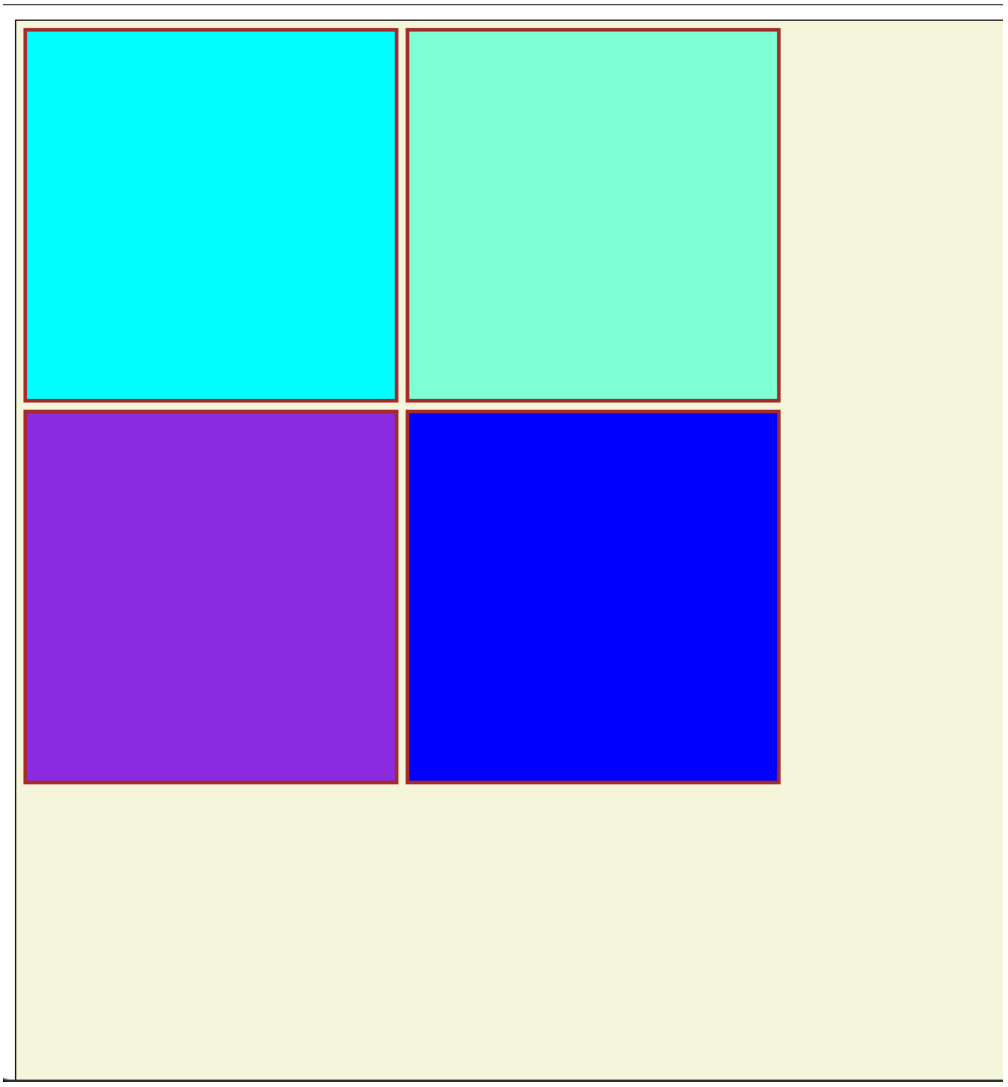
```
    align-items: center;
    justify-content: center;
```



ALIGN-CONTENT:- SUPPOSE SOME SPACING REMAINS BETWEEN SOME ELEMENTS SUCH AS BELOW EXAMPLE..... So to handle space between multiple line of content we use Align-items.



```
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    align-items: start;
    justify-content: start;
    align-content: start;
```



PROPERTIES FOR FLEX-ITEMS:-

1. ORDER:- Use to assgin order to different flex-items.

```
#box1{
   background-color: aqua;
   order: 500;
}
```

2. FLEX-SHRINK & FLEX-GROW:- Used to define at what ease or at what speed a particular flex item will grow or shrink in comparision to other flex items.

```
#box1{
   background-color: aqua;
   order: 500;
   flex-shrink: 4;
}

/*As you reduce the screen size the box1 will shrink 4times more than other boxes*/
```

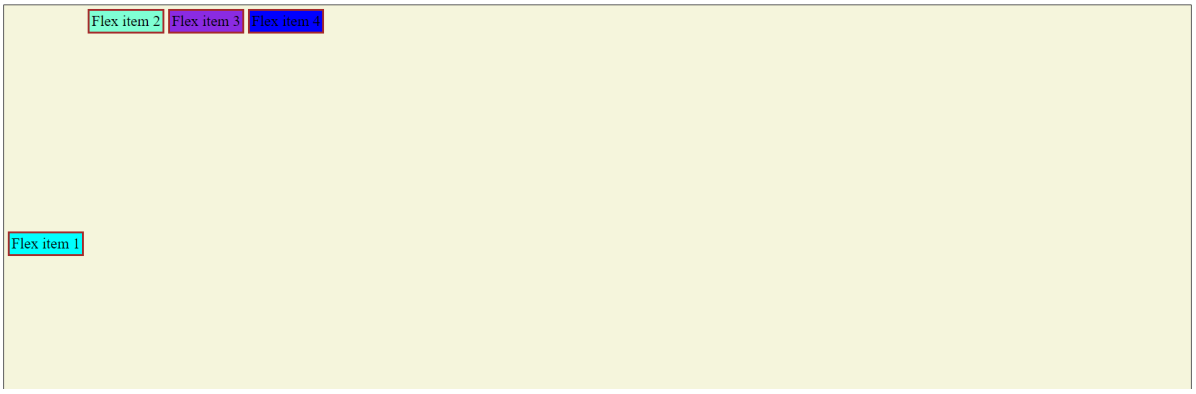
3. **FLEX-BASIS:-** Used to set dimensions of flex-items.

```
#box4{
    background-color: blue;
    flex-basis: 400px;
}

/*will set the dimensions of box4 to 400px*/
```

`flex: 4 3 400;` :- Short-hand Notation `(flex-grow,flex-shrink,flex-basis)`

4. **ALIGN-SELF** `(stretch,center,start,end)` **:-** Defines how an item aligns itself(by default alignment is accross x-axis).

```
#box1{
   background-color: aqua;
   align-self: center;
}
```



MEDIA-QUERIES:- **Tool Responsible** for making a responsive design suitable on all media devices or on different display sizes i.e rendering of a website in different ways on different devices.

```
.box{
    height: 400px;
    width: 400px;
    background-color: aqua;
    border: 2px solid black;
}
@media (min-width:700px) {
    .box{
        background-color: blueviolet;
    }
}

/* On increasing the width more than 700px the color of div will chsnge from aqua to blueviolet*/
```

```css
@media (max-width:500px) {
    .box{
        background-color: blueviolet;
    }
}
/* upto 500px the color will remain blueviolet */
```

```css
@media (min-width:300px)and(max-width:500px) {
    .box{
        background-color: blueviolet;
    }
}
/* Between 300px and 500px the color remains blue violet */
```

SHADOWS:-

1. **BOX_SHADOW:- (by default color of shadow is black.)**

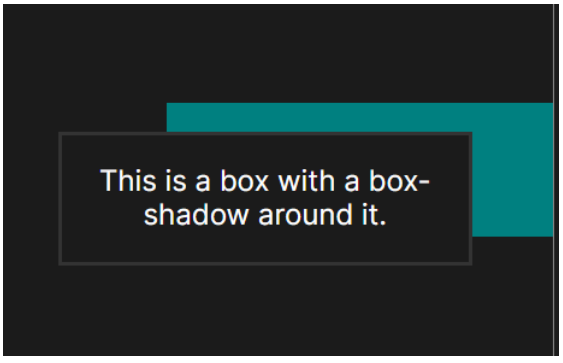
`box-shadow: 10px 5px 5px black;` (x-offset , y-offset , blur radius , color).

`box-shadow: 10px -5px;` (negative offset in y direct means shadow will be formed downwards in y axis)

```css
/* Four length values and a color */
/* <length> | <length> | <length> | <length> | <color> */
box-shadow: 2px 2px 2px 1px rgb(0 0 0 / 20%);

/* inset, length values, and a color */
/* <inset> | <length> | <length> | <color> */
box-shadow: inset 5em 1em gold;
/*Adding multiple shadows*/
box_shadow: 10px 15px red , 12px 10px blue;
```

example:-

```css
box-shadow: 60px -16px teal;
```



EXAMPLE 2:-

```css
*{
    margin: 0px;
    padding: 0px;
    box-sizing: border-box;
}
:root{
    --custom-width: 330px;
}
.container{
    background-color: beige;
    border: 2px solid black;
```

```
    height: 450px;
    display: flex;
    justify-content: center;
    align-items: center;
    margin: 4px;
    padding: 4px;
}
.box{
    --primary_color: aqua;
    background-color: var(--primary_color);
    height: 200px;
    width: 400px;
    border: 2px solid black;
    box-shadow: 10px 15px 5px 50px grey
}
```

2. **TEXT-SHADOW:-** works exactly similar to bax-shadow.

```
/* offset-x | offset-y | blur-radius | color */
text-shadow: 1px 1px 2px black;

/* color | offset-x | offset-y | blur-radius */
text-shadow: #fc0 1px 0 10px;

/* offset-x | offset-y | color */
text-shadow: 5px 5px #558abb;

/* color | offset-x | offset-y */
text-shadow: white 2px 5px;

/* offset-x | offset-y
/* Use defaults for color and blur-radius */
text-shadow: 5px 10px;

/* Global values */
text-shadow: inherit;
text-shadow: initial;
text-shadow: revert;
text-shadow: revert-layer;
text-shadow: unset;
```

ANIMATIONS IN CSS:- Shifting or changing of styles or states of html elements.

```
/* @keyframes duration | easing-function | delay |
iteration-count | direction | fill-mode | play-state | name */
```

```
animation: 3s ease-in 1s 2 reverse both paused slidein;

/* @keyframes duration | easing-function | delay | name */
animation: 3s linear 1s slidein;

/* two animations */
animation:
  3s linear slidein,
  3s ease-out 5s slideout;
```

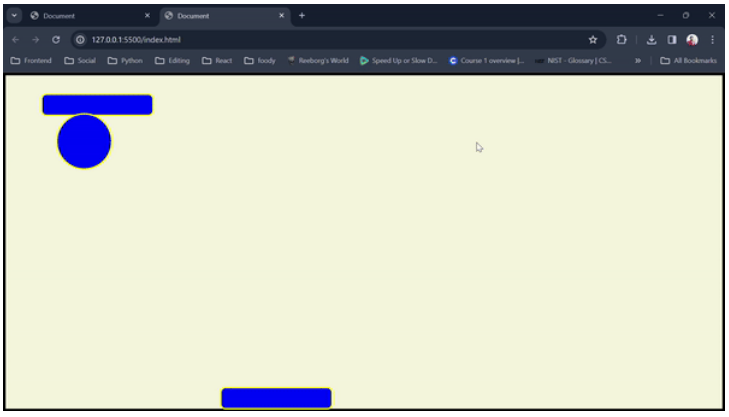
EXAMPLE :-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <div class="box">

        </div>

        <div class="circle">

        </div>
        <div class="box2">

        </div>
    </div>

</body>
</html>
```

```
*{
    margin: 0px;
    padding: 0px;
    box-sizing: border-box;
}
.container{
    width: 100vw;
    height: 100vh;
    background-color: beige;
    border: 5px solid black;
}
.box{
    width: 200px;
    height: 40px;
    background-color: blue;
    border-radius: 10px;
    margin: 10px;
    border: 3px solid yellow;
    position: relative;
    animation-name: rightMovement;
    animation-duration: 4s ;  /*time taken by an animation to complete one cycle */
    animation-iteration-count: infinite; /*number of time an animation cycle must play*/
    animation-delay: 1s; /* after how much time animation must start*/
    /* animation-timing-function: ease-in; start slow, fast ending */
    /* animation-timing-function: cubic-bezier(0.1,0.7,1.0,0.1); */
    animation-timing-function: linear;
    animation-direction: alternate;
```

```
    /* animation-fill-mode: forwards;animation will stop after reaching at end */
}
.box2{
    width: 200px;
    height: 40px;
    background-color: blue;
    border-radius: 10px;
    margin: 10px;
    border: 3px solid yellow;
    position: relative;
    animation-name: movement2;
    animation-duration: 3s ;
    animation-iteration-count: infinite;
    animation-delay: 1s;
    animation-timing-function: linear;
    animation-direction: alternate;
}

.circle{
        width: 100px;
        height: 100px;
        background-color: blue;
        border: 3px solid yellow;
        border-radius: 50px;
        margin: 10px;
        position: relative;
        top: 30%;
        left: 40%;
        animation: 9s ease-in-out 0s infinite alternate-reverse none running circlemovement;
}

@keyframes rightMovement {
    from{
        top: 22px;
        left: 0;
    }
    to{
        top:22px;
        left: 900px;
    }

}


@keyframes movement2 {
    from{
        top: 380px;
        left: 900px;
    }
    to{
        top:380px;
        left: 0px;
    }

}

@keyframes circlemovement {
    0%{
        top:1%;
        left: 5%;
    }
    30%{
        top: 9%;
        left: 90%;
```

```
    }
    60%{
        top: 67%;
        left: 10%;
    }
    100%{
        top: 20%;
        left: 90%;
    }

}
```



TRANSITION IN CSS(`used along with psedo classes such as hover` **):-** Transition means changing from one state to final state or destination state.

Difference between transition and animation is Animation if formed by series of intermediate state or actions (series of complex movements.) while transition is defined as change of intial state to final state.

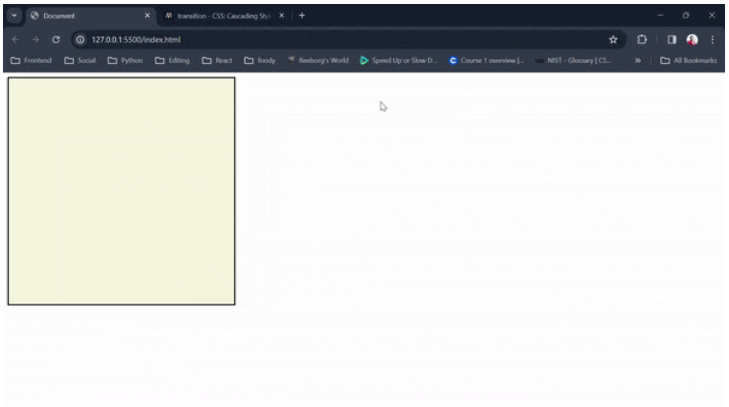
```
/* Apply to 1 property */
/* property name | duration */
transition: margin-right 4s;

/* property name | duration | delay */
transition: margin-right 4s 1s;

/* property name | duration | easing function */
transition: margin-right 4s ease-in-out;

/* property name | duration | easing function | delay */
transition: margin-right 4s ease-in-out 1s;

/* property name | duration | behavior */
transition: display 4s allow-discrete;

/* Apply to 2 properties */
transition:
  margin-right 4s,
  color 1s;

/* Apply to all changed properties */
transition: all 0.5s ease-out allow-discrete;
transition: 200ms linear 50ms;

/* Global values */
transition: inherit;
transition: initial;
transition: revert;
```

```
transition: revert-layer;
transition: unset;
```

Example:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="box"></div>

</body>
</html>
```

```
.box{
    height: 400px;
    width: 400px;
    background-color: beige;
    border: 2px solid black;

    transition-property: all;
    transition-duration: 4s;
    transition-delay: 1s;
    transition-timing-function: ease-in; /*first slow then fast*/
    /* Shorthand notation:- transition: all 2s ease-in 5s; */
    /* transition: width 2s, height 5s; */
}
.box:hover{
    width: 1200px;
    background-color: aquamarine;
}
```



TRANSFORMS:- **Transforming different objects such as rotating,Moving,chaging shapes etc..**

```
/* Keyword values */
transform: none;

/* Function values */
transform: matrix(1, 2, 3, 4, 5, 6);
transform: matrix3d(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
transform: perspective(17px);
```

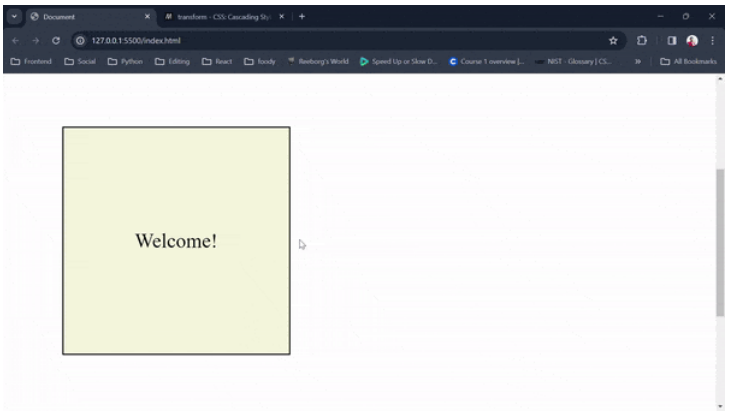
```css
transform: rotate(0.5turn);
transform: rotate3d(1, 2, 3, 10deg);
transform: rotateX(10deg);
transform: rotateY(10deg);
transform: rotateZ(10deg);
transform: translate(12px, 50%);
transform: translate3d(12px, 50%, 3em);
transform: translateX(2em);
transform: translateY(3in);
transform: translateZ(2px);
transform: scale(2, 0.5);
transform: scale3d(2.5, 1.2, 0.3);
transform: scaleX(2);
transform: scaleY(0.5);
transform: scaleZ(0.3);
transform: skew(30deg, 20deg);
transform: skewX(30deg);
transform: skewY(1.07rad);

/* Multiple function values */
transform: translateX(10px) rotate(10deg) translateY(5px);
transform: perspective(500px) translate(10px, 0, 20px) rotateY(3deg);

/* Global values */
transform: inherit;
transform: initial;
transform: revert;
transform: revert-layer;
transform: unset;
```

EXAMPLE:-

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <div class="box">
            <p>Welcome!</p>
        </div>
    </div>


</body>
</html>
```

```css
.box{
    width: 400px;
    height: 400px;
    background-color: beige;
    border: 2px solid black;
    display: flex;
    justify-content: center;
    align-items: center;
    margin: auto;
    transition: all 2s linear 1s;
}
.container{
    height: 100vw;
```

```
    width: 100vh;
    display:flex;
    justify-content: center;
    align-items: center;
}
.box:hover{
    transform: rotate(45deg); /* negative values can also be used to rotate opposite side/*
    /* transform: scale(2);  increase size */
    /* transform: skew(20deg); will tilt the box */
    /* transform: translate(500px 500px);  the box will move 500px in x direction and 500px in y direct

}
p{
    font-size: 36px;
}
```



GRID:- A method which is also used for creating layouts. The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

- The `gap` CSS shorthand property sets the gaps (gutters) between rows and columns.

  `gap: 10px 20px;`

- The `grid-template` CSS property is a shorthand property for defining grid columns, grid rows, and grid areas. `grid-template-rows: repeat(3, 200px);` "3 rows each of 200px"

EXAMPLE-1:-

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body{
    height: 100vh;
    width: 100vw;
}
.container{
    background-color: rgb(59, 57, 57);
    height: 100%;
    width: 100%;
    display: grid;
    padding: 4rem;
    row-gap: 1rem;
    column-gap: 1rem;
    /* grid-template-rows: 1fr 1fr; */
    grid-template-rows: repeat(2,1fr);
    /* grid-template-columns: 1fr  1fr 1fr;     */
    grid-template-columns: repeat(3,1fr);
```

```
}
.item{
    background-color: rgb(94, 94, 194);
    border: 5px solid black;

}
```

EXAMPLE-2:-

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body{
    height: 100vh;
    width: 100vw;
}
.container{
    background-color: rgb(59, 57, 57);
    height: 100%;
    width: 100%;
    display: grid;
    padding: 4rem;
    row-gap: 1rem;
    column-gap: 1rem;
    grid-template-columns: repeat(1,1fr);
    grid-template-rows: repeat(6,1fr);
    /* grid-template-rows: 1fr 1fr; */
    /* grid-template-columns: 1fr  1fr 1fr;     */
}
.item{
    background-color: rgb(94, 94, 194);
    border: 5px solid black;
    color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 1.5rem;

}
@media (min-width:768px){
    .container{
        grid-template-rows: repeat(4,1fr);
```

```
        grid-template-columns: repeat(3,1fr);
    }
    .header,.footer{
        grid-column-start: 1;
        grid-column-end: 4;
    }
    .sidebar{
        grid-row-start: 2;
        grid-row-end: 4;
    }
    .content1{
        grid-column-start: 2;
        grid-column-end: 4;
    }
}
```