10/02/25
Monday

CSS
Cascading Style Sheet

CSS:
* CSS stands for cascading style sheet
* It is used to create attractive web pages.
* It is used to style the or design the web page.

History of CSS
Hakon Wium Lii introduce the idea to style the sheet to the W3C and it accept that idea.

- In 1996, the first version of CSS introduced i.e CSS1.
- In 1998, the second version of CSS introduced i.e CSS2.
- Later on in 2011, the third version of CSS came i.e CSS3.

Ways To Give CSS -
There are three ways to give CSS
1. Inline CSS
2. Internal or Embedded CSS
3. External CSS.

1. Inline CSS
- Inline CSS is used to provide the style inline for a single element.
- Inline CSS are defined within the "style attribute" of the relevant element.

• Example

```
<h1 style = "background-color: red; color: blue;"> Hello,
this is Heading Tag </h1>
<h1 style = "background-color: blueviolet; color: yellow">
Heading 2 </h1>
```

(2) Internal CSS / Embedded CSS-
The internal CSS is defined inside the <style> element, inside the <head> of an HTML page.

• Example

```
<head>
<style>
h1 {
    background-color: yellow;
    color: red
  }
</style>
p {
    background-color: beige;
    color: pink
  }
</head>

In <body>
<!-- Internal CSS -->
<h1> Virat Kohli </h1>
<p> Lorem ipsum dolor sit amet </p>
```

(3) External CSS

- External CSS are defined within the <link> element, inside the <head> section of an HTML page.
- We have to create separate css file.
- We have to connect both the files
- Inside the <head> we have to use link:css
- There are two attributes:

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

rel → relation

./index.css → path

index.css

```
h1 {
    background-color : blanchedalmond;
    color : pink
}
p {
    background-color : brown;
    color : aqua
}
```

```
<body>
<!-External CSS- -->
<h1> Virat Kohli </h1>
<p> Lorem ipsum dolar sit amet </p>
```

Properties of CSS-

- The first priority always goes to Inline CSS
- Next priority depends on the sequence which one is closer.

Syntax of CSS:

Selector, property, value
p { color : value ; }
blue.

SELECTORS-

Selectors are used to target the HTML elements in CSS.
There are 5 types of CSS selector.

1. Simple selector
2. Combinator Selector
3. Attribute Selector
4. Pseudo class Selector
5. Pseudo Element Selector

1. Simple Selector-

A selector which is used to target single HTML element that selector is known as Simple Selector.

There are 5 types of Simple Selector:

1. id selector (#)
2. class selector (.)
3. Universal Selector (*)
4. Element Selector (tag)
5. Grouping Selector (,)

1) id selector (#) - Id selector is used to target any HTML element by its id name.
- Id selector is represented by #.

- we use id attribute to target the specific HTML element uniquely.
- we are not suppose to same id value for multiple it should be unique.

Example

```
<h1 id = "heading1" > Heading 1 </h1>
<h1 id = "heading2" > Heading 2 </h1>
```

In CSS

```
# heading1 {
    background-color : orchid;
    color : black
}
# heading2 {
    background-color : black;
    color : white
}
```

2) Class Selector (.) - Class selector is used to target the multiple element at the same time.
- class selector is represend by .

Example

```
<h1 class = "xyz" > Heading 1 </h1>
<h1> Heading 2 </h1>
<h1 class = "xyz" > Heading 3 </h1>
```

In CSS

```
.xyz {
    background-color : burlywood;
    color : blue
}
```

3.) Element Selector / Tag Selector – Element Selector is used to target any HTML element by its element Name or Tag Name

Example

```
<h1> Virat Kohli </h1>
<p> Lorem ipsum dolar sit amet. </p>
<h1> M.S. Dhoni </h1>
<p> Lorem ipsum dolar sit amet. </p>
```

In CSS

```
h1 {
    background-color : pink;
    color : blue
}
p {
    background-color : beige ;
    color : brown
}
```

4.) Grouping selector – Grouping selector is used to target the multiple elements or group of elements.

- In grouping selector each element is separated by coma (,).

Example

```
<h1> Heading 1 </h1>
<h2> Heading 2 </h2>
<h3> Heading 3 </h3>
<p> Lorem ipsum dolar sit amet. </p>
<h1> Heading 1 </h1>
```

<h2> Heading 2 </h2>
<h3> Heading 3 </h3>
 Lorem ipsum dolar sit amet.

In CSS

```
h1, h2, h3 {
    background-color : pink ;
    color : green
}
```

5) Universal selector - It is used to target all the elements in HTML file.
- It is represented by *.

Example

In CSS

```
* {
    background-color : yellow ;
}
```

=) Priority of Simple Selector -
1. First priority always gives to id selector.
2. Second priority gives to class selector.
3. Third priority gives to Element or grouping selector
4. Last priority gives to Universal Selector.

11/01/25

(20) Combinator Selector -
- Combinator selector is used to target the element based on its relation.
- There are four types -
1. Descendent Selector
2. Child Selector

3. Adjacent sibling selector
4. General Sibling Selector

1) Descendent Selector - Descendent selector is used to target the element based on its ancestors

```
<section>
    <main>
        <div> </div>
    </main>
</section>
```

```
<article>
    <main>
        <div> </div>
    </main>
</article>
```

- The descendent selector is represented by space.

syntax: section ☐ main div { }

Space

Note* → We should avoid to give less number of id and class name.
→ It will affect in JavaScript.

2) Child selector - Child selector is used to target the element based on its nearest parent.

```
<article>
    <section>
        <main>
            <div> </div>
        </main>
    </section>
</article>
```

parent is different

```
<article>
    <aside>
        <div> </div>
    </aside>
</article>
```

- It is represented by > (greater than.

Syntax:
main > div { }

3.) Adjacent Sibling selector-
Adjacent sibling selector is used to target the one HTML element based on its adjacent sibling.
elder

younger brother will take reference of elder brother

```
<section>
  <div>        </div>
  <p>          </p>
  <p>          </p>
  <span>       </span>
  <p>          </p>
</section>
```

- It is represented by +

Syntax:
→ section > div + p { }
Adjacent Sibling

→ section > p + p { }

→ section > span + p

4.) General Sibling selector - general sibling selector is used to target multiple HTML element based on its adjacent elder sibling.

Rena
Meena
Tina

- It is represented by ~ (tilde).

- It is used to target all the elements that are next siblings of a specified element.

Syntax:

Section > div ~ p

General sibling (tilde) Selector

12/02/2

(4)

(3) Attribute Selector -
Attribute selector is used to target the element by its attribute.

Syntax:
input[type = "text"] {

}

label [for = "userName"] {

}

Syntax:
tagname [attribute='value'] {

}

In html

```
<form action = " " >
    <div class = " form-grp ">
        <label for = "userName" > : </label>
        <input type = "text" name = " " id = " " >
    </div>
    <div class = " form-grp " >
        <label for = " password " > Password : </label>
```

```
<input type = "password" name = " " id = " " >
</div>
</form>
```

12/09/25

(4) Pseudo-Class Selector-

A CSS pseudo-class is a keyword added to a selector that specify a special state of the selected element.

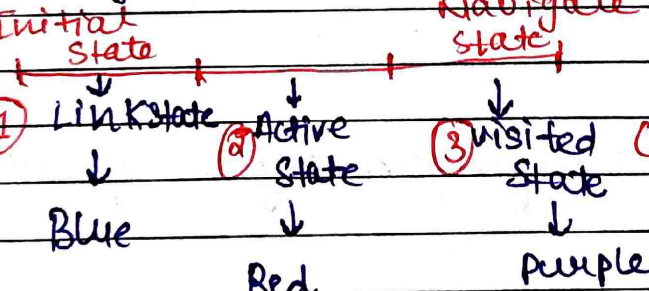
Syntax:

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

Eg

```
<a href = "https://amazon.com" > amazon </a>
```

for hyperlink there are 3-states-



(Before clicking on that)

Active State - The state between link state and navigate state

Pseudo classes are used to style the states of the element.

* we have to change the color using pseudo class
* Represented by :

Types of Pseudo classes

> Dynamic Pseudo classes
Link
Visited
Active
Focus → only work for input
Hover → any

> Structural pseudo classes
first child
last child
Nth-child()
first-of-type
last-of-type

> UI element pseudo-classes
Enabled
Disabled
checked

(1.) Dynamic Pseudo classes -

(a.) Link -
It is used to change the style of the link state.

```
a:link{
    color: green
}
```

(b.) Active -
It is used to change the style of active state.

```
a:active{
    color: purple
}
```

(c.) visited -
It is used to change the style of visited state

```
a : visited{
    color: red
}
```

* Once we visited any website it goes permanently to the visited state. Again it want to comeback then do some changes.

(d) focus -
It is used to change the style for inputs.
Eg
<input type = "text" name = "" id = "" >

```
input : focus {
    background-color : blueviolet ;
}
```

(e) hover -
It is used to change the style when we hover on any elements.
Eg
<h1 > HELLO </h1>

```
h1 : hover {
    color : aquamarine ;
    background-color : red ;
}
```

(2) Structural pseudo class.
(1) first child
It is used to target first child.

```
<ul>
  <li> Delhi </li>
  <li> Gurugram </li>
  <li> Noida </li>
  <li> Pune </li>
```

Syntax

```
ul li : first-child {
    color : red
}
```

b) last-child
It is used to target last child.

```
ul li:last-child{
    color: pink
}
```

c) nth-child
It is used to target any child.

```
ul li: nth-child(2){
    color: purple
}
```

d) first-of-type
It is used to target the first ~~child~~ type of specific element.

```
ul p: first-of-type{
    color: brown
}
```

e) last-of-type
It is used to target the last type of specific ~~child~~ element.

```
ul p: last-of-type{
    color: orange
}
```

Notes :-

```
<ul>
    <p> Hello </p>
```

whenever we click on input, it will go on focused state
→ outline is something inside the border.
field in browser

```
<li> Delhi </li>
<li> Gurugram </li>
<p> Heelo </p>
<li> Noida </li>
<li> Pune </li>
</ul>
```

CSS

```
ul li : first-of-type {
    color : red;
}
```

13/02/25

③ UI element pseudo-classes. → <form>
UI pseudo-class is a proposal that was intended to match elements that are part of the user-interface, but it is not widely supported in modern browsers.

a) Enabled - Matches elements that are enabled.
It is used to target the enabled input.

```
input : enabled {
    background-color : lightblue;
}
```

b) disabled - Matches elements that are disabled.
It is used to target the disabled input.

```
input : disabled {
    background-color : lightgray;
}
```

c) Checked - Matches checkboxes or radio buttons that are selected.

```
input : checked {
    cursor : pointer;
    outline : 5px solid red;
}
```

input [type="password"] {
outline-color : brown;
outline-offset : 10px;
}

⇒ outline-color - to set color of the input field on clicking.

⇒ outline-offset - it will give the border outside the input field according to offset value.

(5.) Pseudo-Element Selector -

- Pseudo-elements are used to style the specified parts of the element.

syntax:

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

- Pseudo-Elements are represented by :: (double colon)

Types of Pseudo-Elements

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

Emoji → window + (.) dot

1. ::first-line -
It is used to target the first line of the element.

Eg
 Lorem - - - -

CSS
p::first-line {
color : brown
}

2. ::first-letter -
It is used to target the first letter of the element.

Eg Lorem - - -

CSS
p::first-letter {
font-size : 200px;
color : blue.
}

3. ::before -
It is used to target before the element add the content

CSS
p::before {
content : "☺";
}

(4) :: after -
It is used to add the content after the element.

```
p :: after {
    content : "♥"
}
```

(5) :: selection -
It is used to add the style and element after selecting.

CSS

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

(6) :: marker -
It is used to style only for the typesn of the list in the element.

```
ol li : nth-child(odd) :: marker {
    content : " 😊 "
    font-size : larger ;
    color : red ;
}

ol li : nth-child (even) :: marker {
    content : " ♡ "
    font-size : larger ;
    color : red ;
}
```

```
<ol>
    <li> Apple </li>
    <li> Mango </li>
    <li> Banana </li>
    <li> Litchi </li>
```

18/07/25

CSS Color

TEXT PROPERTY IN CSS

In text property there are two types
1. Text formatting
2. Text Effect

(1) Text formatting -
- color
- Text - align
- Text - transform
- Text - shadow
- Text - decoration
- Letter - spacing
- Word - spacing
- Text - indention

(2) Text Effect -
- Text - overflow : clip | ellipses
- word - wrap : break word
- word - break : Keep all | break all
- writing - mode : horizontal | vertical - rl

Text formatting -

1. Color -
- In CSS there are six ways to give color

1. predefined color

2. RGB
3. RGBA
4. HSL
5. HSLA
6. Hexadecimal number

→ Color → It is used to give predefined colours

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

→ RGB → Red Green Blue.

```
h1 {
  color: rgb(red, green, blue);
}
```

~~minimum value~~ = 0

In this RGB starting from 0 and end point is 255.
maximum value = 255

```
h1 {
  Color: rgb(0, 0, 0)        → Black
}

h1 {
  color: rgb(255, 255, 255)  → white
}
```

→ RGBA → Red green Blue alpha.

↓

It means transparency.
min value = 0
max value = 1

```
h1 {
    color : rgba ( 0 , 0 , 0 , 0.6 )
}
```

→ when alpha = 0 then transparency is 100%.
alpha = 1 then transparency is 0%.

→ HSL - Hue Saturation & Lightness
→ for Hue the value should be in degree
→ for saturation value should be in percentage
→ for Lightness value should be in percentage

```
h1 {
    color : hsl ( 120 deg , 70% , 50% )
}
```

→ HSLA - Hue Saturation Lightness Alpha.
↓
min = 0
max = 1

Hue → in degree (120 deg)
Saturation → percentage (50%)
Lightness → percentage (60%)
Alpha → (max = 1 , min = 0)

→ Hexadecimal number -
In CSS all the hexadecimal numbers we have to prefix with hash (#).
number.

```
h1 {
    color : #1aa34a ;
}
```

2. Text - align -
To align our text, we use text-align property.

DATE
PAGE No.

- In text-align, four values are there.
  - center
  - start / left
  - end / Rights
  - Justify

3. Text - transform -

In text-transform, there are 3 properties-
1. capitalize
2. uppercase
3. lowercase

4. Text - shadow -

Syntax : text-shadow: (x-value, y-value, Distance, color)

(x-value ↓ x-axis; y-value ↑ y-axis; Distance ↑ z, ↓ spreadness)

text-shadow : (x-axis, y-axis z color).

Only in x-axis & y-axis we can pass negative values

5. Text-decoration -

It is used to decorate text,
In text-decoration, 4 subproperties are there.
1. text-decoration line
2. text-decoration style
3. text-decoration color → To change the color of line.
4. text-decoration thickness

(1) text - decoration line
1) underline
2) line - through

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

(2) text-decoration color- It is used to change the color of line.

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

(3) text-decoration thickness- It is used to change the thickness of color.

```
h1{
    text-decoration-thickness: 10px;
}
```

(4) text-decoration style - It is used to style the line.
There are 5 types-

1. dashed
2. wavy
3. double
4. dotted
5. solid

For all four properties - there is one shortened property i.e text-decoration.

Syntax
text-decoration-line, style color thickness

=> Text-decoration - It is the short end property which consist 4 parameters. We don't need to use 4 properties.

=> If we use all 4 property and also text-decoration so the result in the form of text-decoration because it is old property overridden by new ones.

6. Letter spacing - It is used to give the space inbetween each letter.

```
h1{
    letter-spacing : 20px ;
}
```

7. word spacing - It is used to give the space inbetween each word.

```
h1{
    word-spacing : 60 px ;
}
```

8. text-indention - To give space in the starting we use text-indent.

```
p{
  text-indent : 100 px ;
}
```

9. line-height - It is used to give the space between each line in a paragraph.

```
p{
    line-height : 30 px ;
}
```

The entire paragraph align to center.

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

10. text-align-last - It is used to align the last line of the paragraph.

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

(09) Text effect

→ In paragraph we are having space inbetween so it will adjust to the browser.

(1) word-break

→ If we are having no space inbetween, to break the line we use word break property.

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

(07) Overflow - It is used to fill the data inside the container we got one scrollbar.

```
Section {
    height : 300 px ;
    width : 200 px ;
    border : 1 px solid ;
    overflow-y : auto ;
}
```

- It should be scrollable but hidden.

It is used to modify the scrollbar.

* Pseudo-element

::-webkit-scrollbar {
display: none;
}

→ It is used to hide the scrollbar but it should be scrollable.
If we are not giving any element it will apply globally.

Section ::-webkit-scrollbar {
display: none;
}

19/02/23

BACKGROUND PROPERTY -

1. Background ~~-color~~
2. Background-color
3. Background-image: url("image.jpg")
4. Background-size: cover / 100%
5. Background-repeat: no-repeat / repeat-x/y
6. Background-position: right / left / center

(1) background - It is used to provide background color but it provides multiple background colors.

h1 {
background: red;
}

(2) Background-color -
It can provide the single background colo

whereas background property can provide multiple background colors.

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

Background

In background property we have linear-gradient function in which we can provide multiple colours.

Linear-gradient() function.

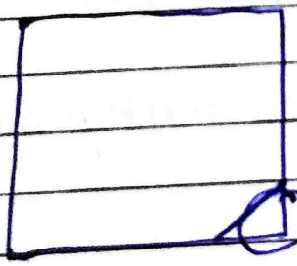
```
h1{
    background : linear-gradient(red, green, yellow, blue);
}
```

Direction of colours are Top to bottom

⇒ Create a container.

<section> </section>

```
Section{
    height : 300px;
    width : 300px;
    border : 1px solid;
    background : linear-gradient(to right, red, green, yellow)
}
```



Change the direction of colour :

1. From left to Right color

```
h1{
    background : linear-gradient(to right, red, green, yellow);
}
```

(2) change color to diagonally

```
h1 {
    background : linear-gradient(to right bottom, red, green,
                                 yellow, blue);
}
```

* linear-gradient() → It is used to pass multiple linear color gradients.

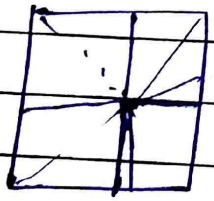
* radial-gradient()

```
section {
    background : radial-gradient (red, blue, yellow,
                                  green);
}
```

- Radial-gradient() is used to provide multiple background colors from the center of the element.

* conic-gradient() → There is no suggestions provided in VS-code.

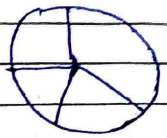
It will provide the multiple background colors in angles (degrees).



```
section {
    background : conic-gradient ( red 0deg 135deg ,
    purple 135deg 200deg , green 200deg 270deg ,
    blue 270deg 360deg );
}
```

* border-radius → To give the curves.

```
section{
    border-radius: 50%;
}
```



How to provide background image?

<aside> </aside>

```
aside{
    height : 300px;
    width : 300px;
    border : 1px solid;
    background-image : url("./photo.jpg");
    background-repeat : no repeat;
    background-position : center;
    background-size : cover;
}
```

background-image → It is used to provide background image.

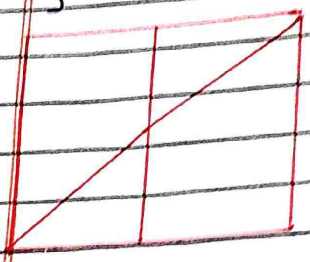
background-repeat → It will specifies the repetition of the image.

background-position → It will specify the position of bg image.

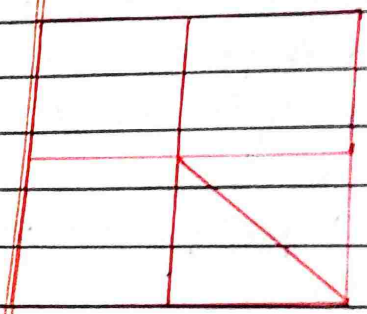
background-size → There are two values i.e content & cover.

Whenever we are dealing with bg image, there are 4 properties are mandatory to use.

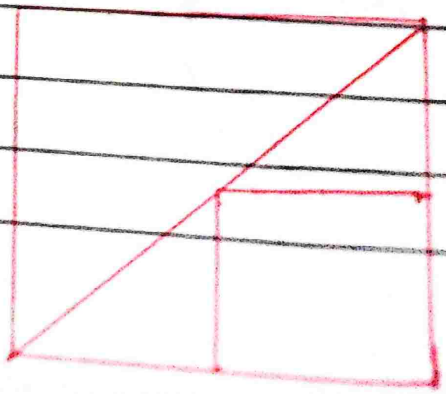
(1) section {
height : 300 px ;
width : 300 px ;
border : 1 px solid ;
background : conic-gradient (blue 0 deg 45 deg , purple 45 deg 180 deg , green 180 deg 225 deg , red 225 deg 360 deg) ;
}



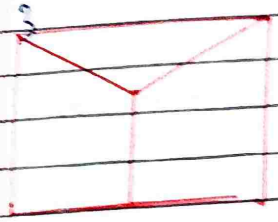
(2) article {
background : conic-gradient (red 0 deg 90 deg blue 90 deg 135 deg , green 135 deg 180 deg , yellow 180 deg 270 deg , purple 270 deg 360 deg) ;
}



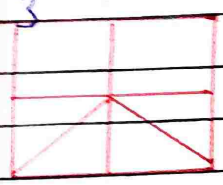
(3) #a {
background : conic-gradient (blue 0 deg 45 deg , yellow 45 deg 90 deg , green 90 deg 180 deg , red 180 deg 225 deg , blue 225 deg 0 deg)
}



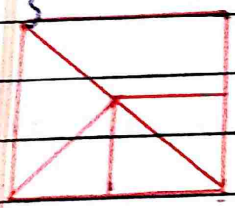
5. #b{
background : conic-gradient(red 0deg 45 deg , pink 45deg 180 deg , blue 180 deg 315 deg , red 315 deg 360 deg)



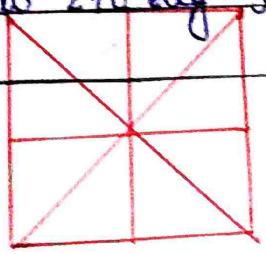
6. #c{
background : conic-gradient(orange 0deg 90 deg , pink 90 deg 135 deg yellow 135 deg 180 deg , purple 180 deg 225 deg , red 225 deg 270 deg green 270 deg 360 deg)



7. #d{
background : conic-gradient(green 0deg 90 deg , red 90 deg 135 deg , yellow 135 deg 180 deg , blue 180 deg 225 deg , pink 225 deg 315 deg , green 315 deg 360 deg)



8. #e{
background : conic-gradient(blue 0deg 45 deg , red 45 deg 90 deg , yellow 90 deg 135 deg , pink 135 deg 180 deg , green 180 deg 225 deg , purple 225 deg 270 deg , yellow 270 deg 315 deg , pink 315 deg 360 deg
}



Google font → Different - Different font families.

20/2/25

FONT PROPERTY

It is used to provide how to change font-style.

- font-size : large / small / medium
- font-weight : bold / bolder / lighter / normal
- font-style : italic
- font-family : font styles.
- font-varient (It is depricated, we can't use).

① font-size : It is used to change the size of font.

 loremm - - - - - - - - - -

also in pixel.

```
#para{
    color : blue;
    font-size : 1orgem;
}
```

② font-family - It is used to change the style of font.

There are limit font-family.

If you want more font-style you can go to google fonts

If you want to give external font style we use ② font-face.

```
@font-face{
    font-family : qspider;
    src : url("./fonts/fonts name - - - .ttf")
}
```

```
#para {
    color : blue ;
    font-size : x-large ;
    font-family : "qspider" ;
}
```

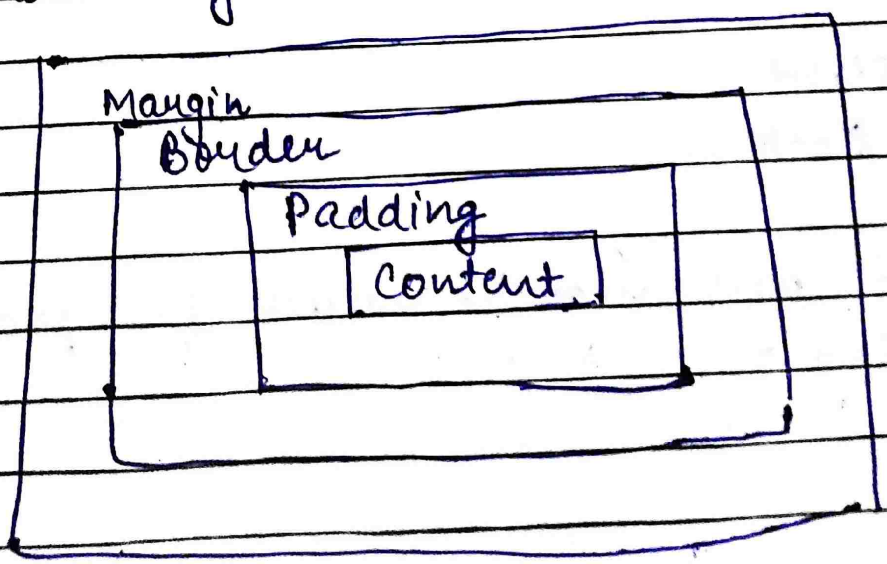
(3) font-style - It is used to give italic font style. or in html we use <i> or

(4) font-weight - It is used to provide bold, bolder, lighter, normal ~~formal~~ font. It should not be in pixel.

```
#para {
    color : blue ;
    font-size : x-large ;
    font-family : "qspider" ;
    font-weight : 900 ;
}
```

BOX MODEL

Box Model is a mechanism which wraps around every HTML element.



(1) Content Box: A box which is created by giving some specific height and width that box is known as content Box.

(2) Padding Box: It is the internal space inbetween content and border.
for all the sides there is shortend property i.e. padding.

padding: Top Right Bottom Left.
T R B L

```
.box {
    height : 300px;
    width : 300px;
    border : 2px solid;
    padding-left : 10px;
    padding-right : 20px;
    padding-top : 30px;
    padding-bottom : 40px;
}
```

```
.box {
    padding: 10px 30px 20px 10px;
}
```

Left = Right
Top = Bottom

X We can't pass negative values for padding.
padding : -10px;

(3) Border Box: To provide border there are some properties like Border-width, Border-style, Border-color.

Border-style : 1. dashed
2. dotted
3. double
4. groove
5. solid
6. hidden

There is shortend i.e Border.

Syntax

Border : Borderwidth style color

```
.box {
    height : 300px ;
    width : 300px ;
    border : 2px solid ;
    border : 2px solid red ;
}
```

The same property will get overridden by new property.

```
.box {
  height : 300px ;
  width : 300px ;
  boudertop : 2px solid red ;
  border bottom : 10px dotted pink ;
  border-left : 20 px groove purple ;
  border-right : 20 px double blue ;
}
```

(4) Margin Box:

- The space between two elements or outside space of every element is known as margin.
- Margin can be in negative.

for margin there is shortend property.

Syntax

margin : Top Right Bottom left

21/02/25

Padding and Border affecting the size of the other Block level element,
To overcome this issue we have property Box-size property.

```
* {
    box-sizing : border-box/content-box;
}
```

Box-Sizing

- This property will specify the sizing of the element.
- It has two values i.e
  - Border Box
  - Content Box
- The default value for this property is content Box.

To remove the default space in the website

this space →

```
div{ height: 300px;
     width : 300px;
     border : 10px solid;
     background-color : red;
     padding : 200px;
}

section{
    height : 300px;
    width : 300px;
    border : 1px solid;
    background-color:
                blue;
}
```

```
* {
    padding : 0;
    margin : 0;
    box-sizing : Border box;
}
```

* This is default CSS we have to write this in every CSS file.

Positions in CSS → It is used to change the position of the element.

There are five positions in CSS -

- Static
- Relative
- Fixed
- Absolute
- Sticky

① **Static** - Position static is fixed on the web page.

- we can't change that position.

```
position : static ;
```

② **Relative** - Position relative will take the reference of its current position.

```
position : relative ;
left : 50px ;
top : 50px ;
bottom : 20px ;
left : 20px ;
}
```

DATE
PAGE No.

(3) Absolute:

```
<section>
  <article>
    <div></div>
  </article>
</section>
```

```
*{
  padding : 0;
  margin : 0;
  box-sizing : border-box;
}

section{
  height : 500px;
  width : 500px;
  border : 1px solid;
}
```

Similarly create for article & div.

If you want to align the box horizontally at center.

```
margin : 0px auto;
```

(Top-bottom → left-right)

This auto will work only for horizontal (x-axis) because width is fix for any device whereas height is not fix.

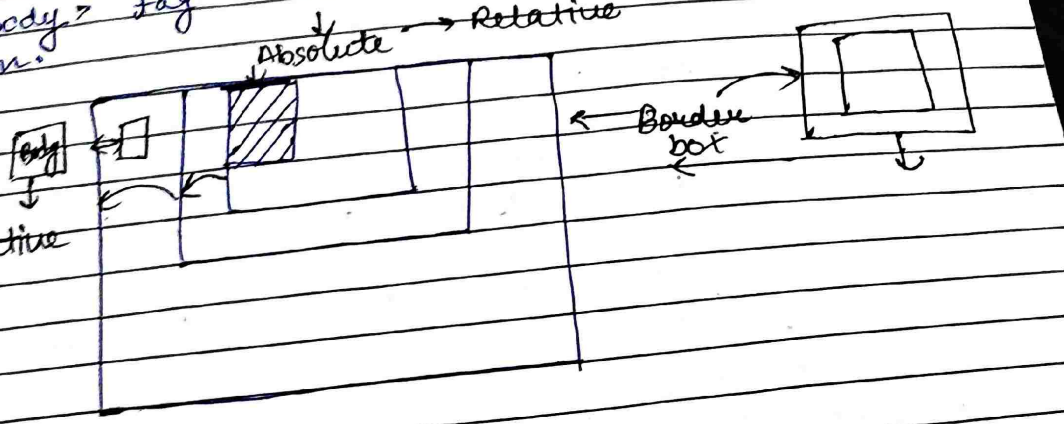
You can give margin auto for article section i.e. for nested container also.

→ Position Absolute will take the reference of its nearest parent but the nearest parent should have position relative. If the nearest parent position is not relative then it will go to next parent.

→ If none of the parent is having position relative then it will take reference of

Inside <P> tag, we should not use tag, we can do vice versa bit.

the <body> tag which have by default relative position.



(4.) Sticky

Position sticky is scrollable upto the certain point once it reached that point it will become fix.

```
nav {
    height : 150px ;
    border : 1px solid ;
    background-color : pink ;
    position : sticky ;
    top : 0px ;
}
```

→ we have to give same position where we want to fix the content.

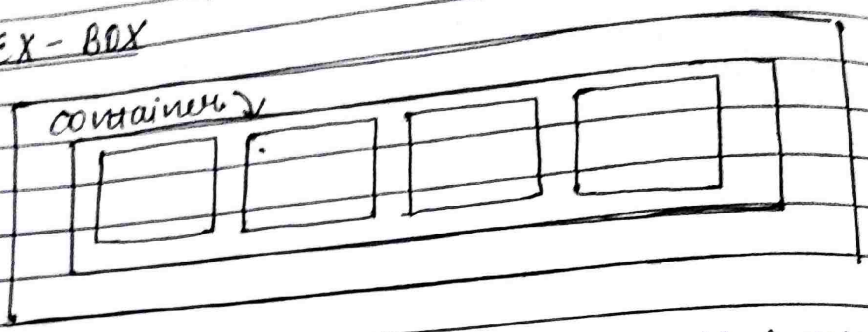
(5.) Fixed -

It is used to fix the position of element, whether the page is scrollable or not.

```
div {
    height : 150 px ;
    width : 100 px ;
    border : 2 px solid ;
    border-radius : 50% ;
    position : fixed ;
    right : 20 px ;
    position : 20 px ;
```

- This property is used to fix the element or UI (User Interface).

FLEX-BOX



Flex-property

- Flex is one dimensional layout, it is used to arrange the elements in one direction i.e either in x-axis or y-axis.
- The default direction of flex is row (x-axis).
- To change the direction there is a property called flex direction.

Flex-direction -

- Flex-direction property we can use to change the direction of flex-items.
- In this, we have 4 values-
  - column
  - column-reverse
  - row
  - row-reverse

```
* {
  padding : 0px;
  margin : 0x auto;
  box-sizing: border-box;
}
```

flex-wrap →
Flex box is totally responsive it will not let his child go outside.

```
• flexContainer {
    height : 500px ;
    width : 500px ;
    border : 1 px solid ;
    background-color : yellow ;
    margin : 0x auto ;
    display : flex ;
    flex : wrap ;
    flex-direction : row ;
}
```

```
• flexitems {
    height : 100 px ;
    width : 100px ;
    border : 1px solid ;
    background-color : red ;
    font-size : xx-large ;
    color : white ;
}
```

flex-wrap -
- It will define the wrapping style of the
- flex container.

It has ~~three~~ values:
1. wrap
2. wrap reverse.
3. nowrap.

```
• flexContainer {
    flex-wrap : wrap ;
}
```

Here main size is 500 x 500
is each box/child size is 100 x 100.

Total 8 child are there, which means 800 X 800 but flex box being responsive it will not allow his child for using child-orginal dimension we use flex-wrap property (by making them in new line.)

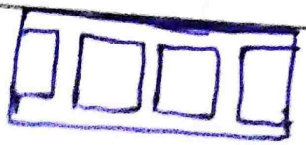
suppose there is space remains at the top.
- To utilize the space in flex container.
- There are two properties.
  1. Justify - content
  2. Align - items
     align - content

Both the properties are opposite to each other.
If J.C is work on x-axis then it will work on y-axis & vice-versa.

In parent, we have to use
.flexcontainer {
  height: 100px;
  width: 100px;
  border: 1px solid;
  Background-color: yellow;
  Justify-content: start;
}

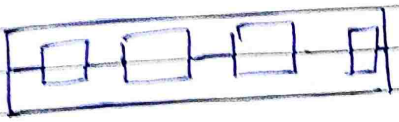
Justify-content - It is used to manage space on main axis.
There are 6 values:
1. Start / flex-start / left
2. end / flex-end / Right
3. center
4. space-around
5. space-between
6. space-evenly

space-between: space between the boxes

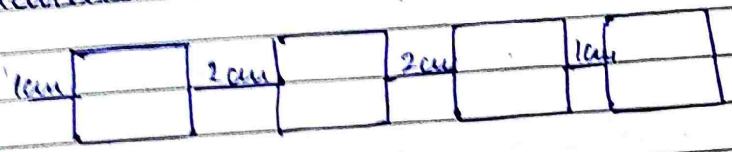


flex-start | flex-start | left
new browser old browser में work करता है।

space-evenly : will distribute space evenly / every space same.

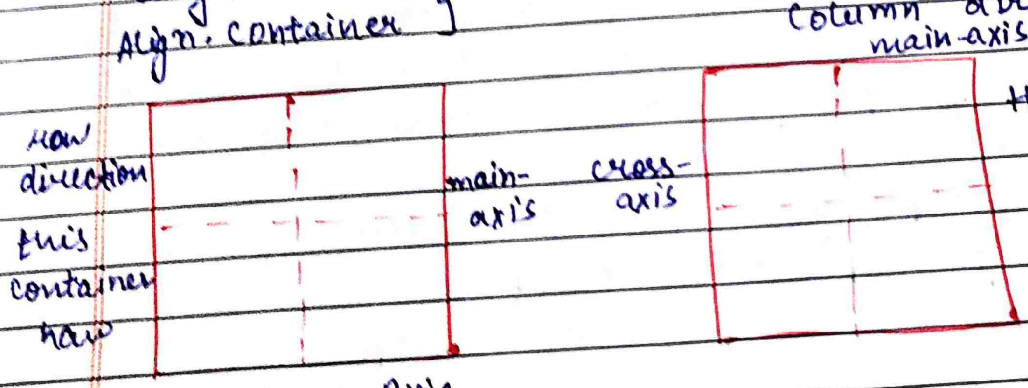


space-around : It will give same space around the element.



Justify Content
Now we use to justify-content in x-axis so here we will use align-items for y-axis.

Justify-content → main axis ⇒ work.
align-items } Cross-axis
Align-container }



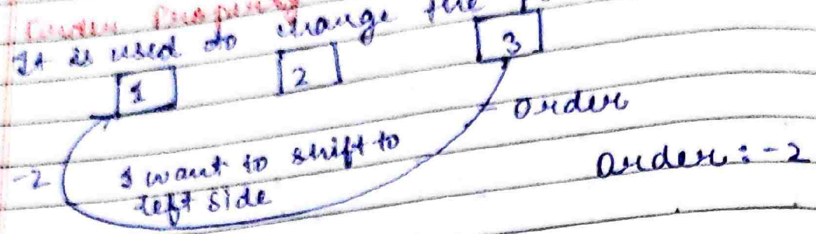
Direction is row, so main-axis will be row, means main-axis will be horizontal.
Here main-axis J.C will work and
AI | AC ⇒ cross-axis

Direction is column, main-axis will be horizontal ⇒ J.C property will work
in cross-axis
AI | AC property will work.

flex- container (parent)
&
flex. item (child)

In browser, search for flex box froggy, we have to put some property and put froggy in circle.

Order Property is for flex items (for child)
It is used to change the position.



Order : -2

For left pass -ve values
For right pass +ve values

Align self → same like align items, but it is for flex-items (child)

Flex-Flow -
The flex-direction and flex-wrap are used so often together that the short hand property was created to combine them.
This property accepts the value of two property separated by space.
Eg : flex-flow- row-wrap

* Difference between align content and align items -
Align-contents - Determine the spacing between lines.
Align-items- Determines how items are whole are align within the container.
when there is only one line, align-content has not effect.

25/02/25
\# CREATE Na

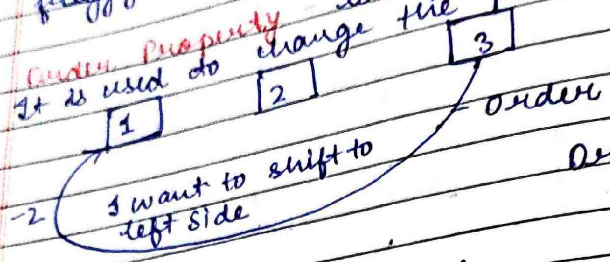
Logo

navigation.
<!DOCTYPE
<html lang
<head>
<meta c
<meta n
initial
<title>
<link re
<link
libs
integr
Inx
=
</head>
<body>
<nav

flex-container (parent)
↓
flex-item (child)

In browser, search for flexbox froggy, we have to put some property and put froggy in circle.

Order Property - is for flex items (for child)
It is used to change the position.



Order : -2

For left pass -ve values
For right pass +ve values

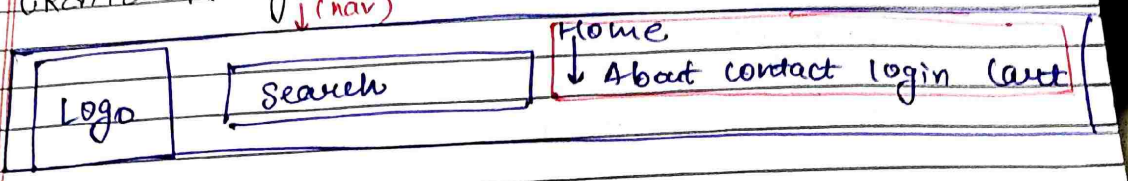
align self → same like align items, but it is for flex-items (child)

Flex-Flow -
The flex-direction and flex-wrap are used so often together that the short hand property was created to combine them.
This property accepts the value of two property separated by space.
Eg : flex-flow - row-wrap

* Difference between align content and align items -
Align-contents - Determine the spacing between lines.
Align-items - Determines how items are whole are align within the container.
When there is only one line, align-content has not effect.

25/02/25

CREATE Navigation Bar-



navigation.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="./navigation.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/
    libs/font-awesome/6.7.2/css/all.min.css"
    integrity="sha512-Evv84MzCkqVQRNSgIGL/F/aIDqQb7xQ2vcrd
    IwxfjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg=
    =" crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>

<body>
  <nav class="navBar">
    <div class="logo"></div>
    <input type="text" name="" id="" class="search"
      placeholder="Search the products">
    <ul>
      <li> Home </li>
      <li> About </li>
      <li> Contact </li>
      <li><i class="fa-solid fa-cart-shopping fa-2xl" style="color:
        purple;"></i></li>
    </ul>
  </nav>
</body>
```

navigation.css -

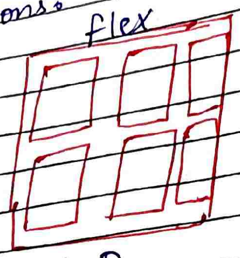
```
* {
    padding : 0;
    margin : 0;
    box-sizing : border-box;
}

.navBar {
    height : 12vh;
    width : 100%;
    background-color : black;
    color : white;
    display : flex;
    justify-content : space-between;
    align-items : center;
    padding : 0px 20px;
}

.navBar .logo {
    height = 80%;
    width = 10%
    background-color : white;
}

.navBar ul {
    height : 100%;
    width : 40%;
    display : flex;
    list-style-type : none;
    justify-content : space-between;
    font-size : larger;
    align-items : center;
}
```

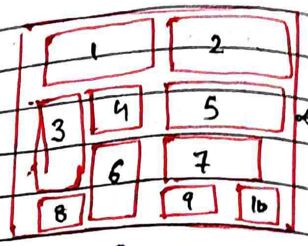
```
.navBar .search {
    height: 50%;
    width: 30%;
    outline: none;
    font-size: Large;
    border: 1px solid black;
    padding: 0px 5px;
    border-radius: 5px;
}

.navBar ul li {
    padding: 7px 13px;
    border-radius: 5px;
}

.navBar ul li:hover {
    background-color: gray;
}

.navBar ul li:last-child:hover {
    background-color: transparent;
}
```

10/03/25

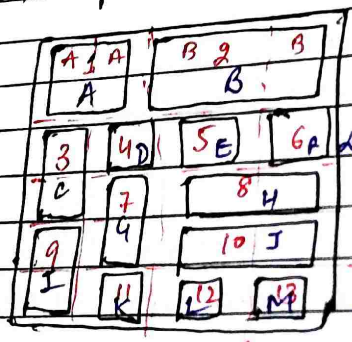
Grid

Grid is a 2-dimensional layout which is used to arrange the elements in both the directions.





grid-area → To provide area name.



<section class = "container">

div*13.box$ ← count like box1, box2

(div → tagname, 13 → 13boxes, .box → classname)

</section>

A, B, C, D like these different areas are there which is divided in the box.
To pass or give this area, we have grid area property.

grid-area:

In parent, area for grid we have to use one property parent container

```
.container { display: grid;
             grid-template areas: " A A B B
                                    B B B
```

Columns wither, we added ⇒ columns as A size is less than B.

```
.container { grid-template areas: "A A B B B B"
                                  "C D E E F F"
                                  "C G H H H H"
                                  "I G J J J J"
                                  "K K L L M M" }
```

If I want provide gap, there is one property gap property.

```
.container { display: grid;
             grid-template areas:
             gap: 10px;
           }
```

grid.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="./grid.css">
</head>
```

```
<body>
  <section class="container">
    <div class="box1"></div>
    <div class="box2"></div>
    <div class="box3"></div>
    <div class="box4"></div>
    <div class="box5"></div>
    <div class="box6"></div>
    <div class="box7"></div>
    <div class="box8"></div>
    <div class="box9"></div>
    <div class="box10"></div>
    <div class="box11"></div>
    <div class="box12"></div>
    <div class="box13"></div>
  </section>

</body>
</html>
```

grid.css

```
*{
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

.container{
  height: 600px;
  width: 600px;
  border: 1px solid;
  margin: 0px auto;
```

```
display : grid ;
grid-template-areas :
    "A A B B B B"
    "C D E E F F"
    "C G H H H H"
    "I G K K K K"
    "I J L L M M" ;
gap : 10px ;
background-color : black ;
}

div {
    border : 1px solid ;
}

.box1 {
    grid-area : A ;
    background-color : pink ;
}

.box2 {
    grid-area : B ;
    background-color : brown ;
}

.box3 {
    grid-area : C ;
    background-color : aqua ;
}

.box4 {
    grid-area : D ;
    background-color : blanchedalmond ;
}
```

```
.box5{
    grid-area : E;
    background-color : violet;
}

.box6{
    grid-area : F;
    background-color : red;
}

.box7{
    grid-area : G;
    background-color : yellow;
}

.box8{
    grid-area : H;
    background-color : green;
}

.box9{
    grid-area : I;
    background-color : gray;
}

.box10{
    grid-area : J;
    background-color : aqua;
}

.box11{
    grid-area : K;
    background-color : pink;
}
```

```
.box12{
    grid-area : L ;
    background-color : salmon ;
}

.box13{
    grid-area : M ;
    background-color : bisque ;
}
```

11/03/25

=> Transition -
Changing the state over the time i.e transition.

=> Transformation -
Changing the state immediately over the time i.e transformation.

=> To change the state immediately we use transform property.

```
rotate()
rotateX('angle')
rotateY('angle')
rotateZ('angle')
scale()
scaleX()
scaleY()
scaleZ()
Skew()
SkewX()
SkewY()
```

```
translate()
translateX()
translateY()
translateZ()
```

DATE
PAGE No.

12/03/25

transform: rotate(30deg);
transform: rotateX(40deg);
transform: rotateY(50deg);

→ shorthand for X and Y

* Inside scale() we have to pass numbers
transform: scale(1); → Same
transform: scale(2); → Increase
Below(1) → Reduce

we can't pass negative values, it should be in positive value.

.container: hover{
transform: scale(1.8);
}

transform: scale(2); → shorthand for X and Y
transform: scaleX(1.8);
transform: scaleY(1.8);

transform: skew(40deg); → parallelogram
transform: skew(90deg);
transform: skew(60deg);

transform: translateX(200px); → to shift
transform: translateY(200px);
transform: translate(100px, 300px); → shorthand property for X&Y
↓ X ↓ Y

12/03/25

ANIMATION :

- Animation is continuous transition.
- Transition is unidirectional and animation is multi-directional.
- There are two ways to create animation in CSS.
  1. from and to ---> for unidirection animations.
  2. Percentage ---> for multidirection animations.

animation-name → for animation we have to take one animation-name.

@keyframes identifier { ← name of animation
It is used to designed the animation.

@keyframes car {
  from { ← It is initial position
  }
  to { ← It is destination position.
  }
}

animation-duration : 5s ; → It is used to start the animation

animation-iteration-count : infinite ; → for continuous again and again

animation-timing-function : ease ;

values:
- ease → starting fast then slow
- ease-in → starting slow then fast
- ease-out → starting fast then slow
- ease-in-out → constant
- steps() → starting fast in end slow travelling distance in steps. → 10

animation.html

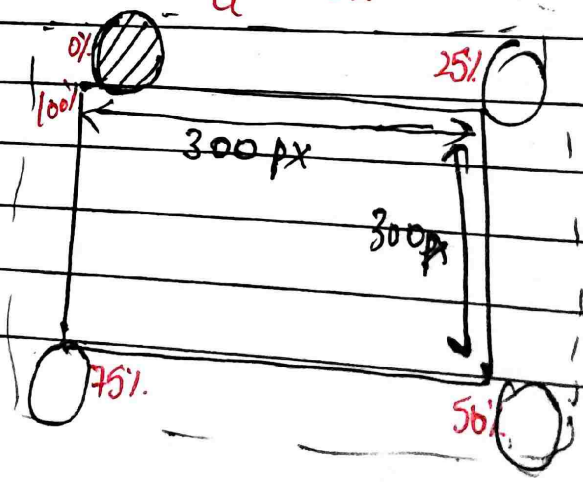
```
<div class="car"></div>
```

animation.css

```
.car {
    height: 100px;
    width: 200px;
    background-color: red;
    animation-name: Bike;
    animation-duration: 5s;
    animation-iteration-count: infinite;
    animation-timing-function: steps(10);
}

@keyframes Bike {
    from {
    transform: translate(0px);
    width: 200px;
    }
    to {
    transform: translate(1400px);
    width: 1400px;
    background-color: purple;
    }
}
```

$\frac{100}{4} = 25\%$

Task



animation-delay : 2s ;
animation-direction : alternate

animations.html
<section class = "container">

</section>

animation.css
.container{
height : 400px ;
width : 400px ;
margin : 0 auto ;
}

.ball{
height : 45px ;
width : 45px ;
border : 1 px solid ;
border-radius : 50% ;
background-color : purple ;
animation-name : ball ;
animation-duration : 5s ;
animation-iteration-count : infinite ;
animation-timing-function : ease-in-out ;
animation-delay : 2s ;
animation-direction : alternate ;
}

.box{
height : 300 px ;

17/03/25

```
    width: 300px;
    border: 2px solid;
    background-color: orchid;
}

@keyframes ball {
    0% {
        transform: translate(0px);
    }
    25% {
        transform: translate(305px, 0px);
        background-color: green;
    }
    50% {
        transform: translate(305px, 345px);
        background-color: red;
    }
    75% {
        transform: translate(-50px, 350px);
        background-color: pink;
    }
    100% {
        transform: translate(-50px, 0px);
        background-color: burlywood;
    }
}
```

17/03/23

In order to make design responsive we use media queries.
For mobile view of something we use @ media queries

To make 100% responsive we use media query. we have to write different-different css or make change in css for different screens.
When our width is going below than this css will set applied.

NavBar

classtask.html

```
<nav class="navBar">
  <div class="Logo"></div>

  <ul>
    <li> Home </li>
    <li> Contact </li>
    <li> About </li>
    <button class="button"> Login </button>
    <button class="button"> Login </button>
  </ul>
</nav>
```

classtask.css

```
*{
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

.navBar{
```

```
    height : 12vh;
    width : 100%;
    background-color : black;
    color : white;
    display : flex;
    justify-content : space-between;
    align-items : center;
    padding : 0px 20px;
}

.navBar .logo{
    height : 90%;
    width : 10%;
    background-color : white;
    background-image : url(spider.jpeg);
}

.navBar ul{
    height : 100%;
    width : 50%;
    display : flex;
    life-style-type : none;
    justify-content : space-between;
    font-size : x-large;
    align-items : center;
}

.navBar ul li{
    padding : 7px 13px;
    border-radius : 5px;
}
```

```
.navBar ul li:hover {
    background-color : gray;
}

.button {
    height : 50px;
    width : 100px;
    border : none;
    background-color : rgb(118, 118, 232);
    font-size : x-large;
    border-radius : 5px;
}

@media screen and (max-width: 768px) {
    .navBar {
        height : 100vh;
        width : 40%;
        flex-direction : column;
    }
    .navBar .logo {
        display : none;
    }
    .navBar ul {
        flex-direction : column;
    }
}
```

In mobile view logo will not yet displayed

* When we are designing we should not use margins and positions because these things are not good for responsiveness.

Now, suppose I want to add a box shadow for Emoji ☺

with the help of box shadow property it is not possible as it comes like 😊

To solve this we have another property which is filter.

filter : drop-shadow (0px 0px 10px red ;)
↓ ↓ ↓ ↓ ↓
It is a function of filter property — x-axis, y-axis, spreadness, colour.

After Applying my Emoji will be: 

classtask.html

```
<div class = "emoji "> <i class = "fa-solid fa-face-
smile fa-8x" style = "color : #0c0d0d; " ></i></ div
```

classtask.css

```
•emoji {
  margin : 100px ;
  /* box-shadow : 0px 0px 10px red ; */
  filter : drop-shadow (0px 0px 10px red)
}
```

I want to create a section containing image in a box -

class task.html -

```
<section class = "box" >
    <img src = "image 4.jpg" alt = " " >
</section>
```

classtask.css -

```
.box{
    margin : 50px;
    height : 400px;
    width : 400px;
    border : 1 px solid;
    background-color : grey;
    display : flex;
    justify-content : center;
}

.box img{
    mix-blend-mode : multiply;
}
```

* To modify background of image → means ke box ke background and image ke background color ko apas mein blend karne ke liye. we use this property.