CSS

-->css stands for cascading style sheets
-->its a language to make our web-page presentable
-->designed to make style sheets for webpage

-->acronmy:
cascading :- falling of style
styles :- adding designs/styling our html tags
sheets :- writing our style in different documents

HISTORY OF CSS

--->1994 first proposed by HAKON WIUM LIE on 10TH October
--->1996 css was published on 17th November with influencer BERT BOS.
--->later he became co-author of css.
--->1996 css became official with css was published in December

---

TYPES OF CSS OR DIFFERENT WAYS OF ADDING CSS

*INLINE STYLE
*INTERNAL STYLE
*EXTERNAL STYLE

1)INLINE STYLE:-
-->inline css is used to apply css on single line or element
-->before css this way the only way to apply styles
-->not an efficient way to write as it has lot of redundancy

eg:- hello

2)INTERNAL STYLE:-
-->internal css is used to apply css on a single document or page.
it can effect all the element of the page.
-->it is written inside the style tag within head section of html.

```
eg:- <head>
    <style>
     p{
        color : blue;
      }
   </style>
   </head>
```

3)EXTERNAL STYLE:

-->External CSS contains separate CSS file which contains only style property with the help of tag attributes.
(For example class, id, heading, … etc).

-->CSS property written in a separate file with .
css extension and should be linked to the HTML document
using link tag.

```
eg:-  <!DOCTYPE html>
<html>
   <head>
      <link rel="stylesheet" href="style1.css"/>
   </head>

   <body>
     <h1 id="demo" > HELLO </h1>
   </body>
</html>
```

Example: The file given below contains CSS property.
This file save with .css extension. For Ex: style1.css

```
   #demo{
        color : "yellow";
        }
```

PRIORITY ORDER:
INLINE->INTERNAL->EXTERNAL

CSS SELECTORS:

--> SELECTOR ARE USED TO TARGET ELEMENT AND APPLY CSS TO IT.

SYNTAX:-

```
   selector_name{
         property:value;
          }
```

1)SIMPLE SELECTOR
-->ID(#)
-->CLASS(.)

-->ELEMENT SELECTOR(TAGS NAME/ELEMENT NAME)
-->GROUPING SELECTOR(,)
-->UNIVERSAL SELECTOR(*)
2)COMBINATOR SELECTOR
3)PSEUDO CLASS
4)PSEUDO ELEMENT

---

1)SIMPLE SELECTOR:-

---

a)ID SELECTOR(#):-
ID attribute is used to select html element used to target specific or unique element.

```
eg: #p1{
        color:red;
    }

<p id="p1">hiiii</p>
```

---

b)CLASS SELECTOR(.):-
class selector selects HTML elements with a specific class attribute. (.(dot) symbol) followed by the class name.

```
eg: .p1{
        color:red;
    }

<p class="p1">hiiii</p>
```

---

c)ELEMENT SELECTOR(TAG/ELEMENT NAME):-
The element selector selects the HTML element by tag name.

```
EG:- p{
        color:red;
    }

<p>hiiii</p>
```

D)Group Selector(,):-
The grouping selector is used to select all the elements with the same style.
-->Grouping selector is used to minimize the code. Comma(,) is used to separate each selector in grouping.

eg:-

```
h1,h2,p {
    text-align: center;
    color: blue;
}
```

---

e) Universal Selector(*):-

The universal selector is used as a wildcard character.
It selects all the elements on the pages.
-->when u want to use similar css to all tags

eg:-

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

PRIORITY ORDER ID->CLASS->ELEMENT

---

2)Combinators selector:-

A combinator is something that explains the relationship/combination between the selectors.

There are four different combinators in CSS:

descendant selector (space)
child selector (>)
adjacent sibling selector (+)
general sibling selector (~)

descendant selector (space)
-->it will take all the direct child and in-direct child

child selector (>)
-->it will take only direct child

adjacent sibling selector (+)
-->it will select adjacent sibling or immediate sibling
right after the first sibling.

general sibling selector (~)
-->it will select all the siblings right after it.

\\PSEUDO ELEMENT

pseudo element selector (::)
A CSS pseudo-element is a keyword added to a selector that lets you style a specific part of the selected elements.

Style the first letter or line of an element
Insert content before or after the content of element

Syntax:-

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

::first-line pseudo element :-
first-line is used to style only first-line of the content

```
<p>
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae
    libero maxime voluptatibus saepe tempore, fugiat soluta ad sit fugit,
    veniam culpa voluptatem nobis amet enim labore. Recusandae, labore rem.
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae
    libero maxime voluptatibus saepe tempore, fugiat soluta ad sit fugit,
    veniam culpa voluptatem nobis amet enim labore. Recusandae, labore rem.
  </p>
```

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

::first-letter pseudo element:-
it is used to style only first-letter/characther in a content

```
<p>
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae
    libero maxime voluptatibus saepe tempore, fugiat soluta ad sit fugit,
    veniam culpa voluptatem nobis amet enim labore. Recusandae, labore rem.
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae
    libero maxime voluptatibus saepe tempore, fugiat soluta ad sit fugit,
    veniam culpa voluptatem nobis amet enim labore. Recusandae, labore rem.
  </p>
```

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

::before pseudo element
The ::before pseudo-element can be used to insert some content/img before the content of an element.

Syntax :

```
selector::before {
    property:value;
}
```

```
<p>
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae
```

libero maxime voluptatibus saepe tempore, fugiat soluta ad sit fugit,

veniam culpa voluptatem nobis amet enim labore. Recusandae, labore rem.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae

libero maxime voluptatibus saepe tempore, fugiat soluta ad sit fugit,

veniam culpa voluptatem nobis amet enim labore. Recusandae, labore rem.

</p>

eg:-

```
p::before {
  content: url("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTtZFJN1r
        G7233vqSCo-DF0xwsCo6MZaHohkg&usqp=CAU");

}
```

::after pseudo element :

The ::after pseudo-element can be used to insert some content after the content of an element.

Syntax :

```
selector::before {

    property:value;

}
```

Marker :-

---->it is used to style only items in a list.

```
<ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JS</li>
  </ul>

li::marker {
  color: blueviolet;
}
```

```
<ul>
     <li>HTML</li>
     <li>CSS</li>
     <li>JS</li>
   </ul>
```

```
li::marker {
  content: "🧿";
}
```

::Selection pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user.

The part will selecting will reflect changes

Syntax:

```
     selector::selection{
         property:value;
    }
```

```
  <p>

     Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae
     libero maxime voluptatibus saepe tempore, fugiat soluta ad sit fugit,
     veniam culpa voluptatem nobis amet enim labore. Recusandae, labore rem.
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae
     libero maxime voluptatibus saepe tempore, fugiat soluta ad sit fugit,
     veniam culpa voluptatem nobis amet enim labore. Recusandae, labore rem.

   </p>
   <ul>
     <li>HTML</li>
     <li>CSS</li>
     <li>JS</li>
   </ul>
```

```
  p ::selection {
 color: blueviolet;
 background-color: aquamarine;

}
```

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;
}

Anchor pseudo class -
The anchor pseudo-classes represent the state of links as visited, unvisited,active/currently selected

Visited - purple
Active / currently selected
Unvisited
Focus

Anchor pseudo-classes also enables you to activate the HTML elements or apply a specified style to an element when the mouse pointer is kept over it

The anchor pseudo-classes include the following :

:link (Applies styles to non visited links)
:visited (Applies styles to visited links)
:hover (Applies styles to an element over which the mouse-pointer moves)
:active (Applies styles to an active element)

Syntax
I) Link : once we visit the color will change to mentioned one.
a:link {
color: darkgoldenrod;
}

Ii) Visited

```
<body>
   <a href="https://www.google.co.in/">click here</a>
</body>

/* visited */
a:visited {
  color: red;
}
```

---

3) Hover : when u place cursor color will change

```
a:hover{
    color: brown;
}
```

Iv) Active :
when user clicks on the link

```
a:active{
    color: black;
}
```

V) Focus : when its in active state we can add

```
input:focus {
  color: gray;
}
```

v1)checked : when user checks on radio or check-box input field.

```
input:checked{
    height : 100px;
    width :100px;
    accent-color : blue;

 }
```

Pseudo child

First-child

```
 <div>
    <p>First child1</p>
</div>
<div>
    <p>First child2</p>
    <p>First child3</p>
</div>
    <p>First child4</p>

p:first-child {
  color: rgb(11, 245, 11);
}
```

Last-child

```
<p>First child1</p>
<div>
   <p>First child2</p>
   <p>First child3</p>
</div>
   <p>First child4</p>
```

```
p:last-child {
  color: rgb(11, 245, 11);
}
```

Nth-child
Odd

```
  <p>First child1</p>
   <p>First child2</p>
   <p>First child3</p>
   <p>First child4</p>
```

```
p:nth-child(odd) {
  color: rgb(11, 245, 11);
}
```

Even

```
  <p>First child1</p>
   <p>First child2</p>
   <p>First child3</p>
   <p>First child4</p>
```

```
p:nth-child(even) {
  color: rgb(11, 245, 11);
}
```

---

first of type
Specify a background color for the first <p> element of its parent:
eg:

```
<h1>Welcome to My gallerypage</h1>

<p>This paragraph is the first child of its parent (body).</p>


 <h1>Welcome to My Homepage</h1>
<p>This paragraph is not the first child of its parent.</p>
```

```
<div>
  <p>This paragraph is the first child of its parent (div).</p>
  <p>This paragraph is not the first child of its parent.</p>
</div>
```

```
p:first-of-type {
   background-color: #ff0000;
}
```

Text Property

color : color_name
Text-align : right , left , center
Text-transform : capatilize , uppercase , lowercase
Text-shadow : x-axis , y-axis , blur colorname
Text-decoration : underline , linethrough , overline
Letter-spacing : provides space between each letters
Word-spacing : provides space between each word
Text-indention : provides space at the starting of phara

Font Property

Font-size : large/small/medium
Font-weight : bold/bolder/lighter/normal
Font-style : italic
Font-family : font styles
Font-varient : normal/small-caps

Background Property

Background-image : url (“image.jpg”)
Background-repeat : no-repeat/repeat-x/Y
Background-size : cover/100%
Background-Position : right/left/center/top/bottom
Background-attachment : scroll/fixed

color/background-color

Color : (color name)
Color : #efefef;
Color : rgb(255,255,0)
Color : rgba(red, green, blue, alpha(0 - 1.0))
Color : hsl(hue(deg), saturation(%), Lightening(%))
Color : hsla(hue(deg), saturation(%), lightening(%), alpha)

Background-color : (use any color property)

Hue:
Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, 240 is blue.

Saturation:
It takes value in percentage; 0% means a shade of gray and 100% is the full color.

Lightness:
Lightness is also a percentage; 0% is black, 100% is white.

Position Property

---->The position property specifies the type of positioning method used for an element.

Note: There are five different position values Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

Property Values:

- static - Default value. Elements render in order, as they appear in the document flow
- absolute - The element is positioned relative to its first positioned (not static) ancestor element
- fixed - The element is positioned relative to the browser window
- relative - The element is positioned relative to its normal position.
- sticky - The element is positioned based on the user's scroll position.

Note: A sticky element toggles between relative and fixed, depending on the scroll position.

CSS Box Model

- The CSS box model is a container that contains multiple properties including borders, margin, padding, and the content itself.

- content: This property is used to displays the text, images, etc, that can be sized using the width & height property.
- padding: This property is used to create space around the element, inside any defined border. The padding is transparent
- border: This property is used to cover the content & any padding, & also allows to set the style, color, and width of the border. A border that goes around the padding and content
- margin: This property is used to create space around the element ie., around the border area.

Transform

2D Transforms
CSS transforms allow you to move, rotate, scale and skew an elements

2D transformation methods:

Tanslate(): allows you to move elements
: we can use translateX and translateY also

```
<h1>2D transform</h1>
```

```
h1 {
  border: 2px solid black;
  width: 300px;
  margin-left: auto;
  margin-right: auto;
  margin-top: 40px;
  background-color: aqua;
  transform: translate(100px, 150px);
}
```

Css translate moves an element up and down or side-to-side:
By indicating one value, you move the element to the right side. Negative values move elements to the left.
The second value moves the element down. Negative values moves element up.

2) **Rotate(deg):** Rotates the element clockwise from its current position.

We have to use one value only

Rotate()
eg:
```
.rotated {
  transform: rotate(45deg); /* Equal to rotateZ(45deg) */
  background-color: pink;
}
```

Scale(): Affects the size of the element. This also applies to the font-size, padding, ,height, and width of an element, we cannot use px ,we can use decimal value, normally we give just number.
scaleX()

```
          h1{
 border: 2px solid black;
  width: 300px;
  margin-left: auto;
  margin-right: auto;
 margin-top: 40px;
  background-color: aqua;
  transform: scaleX(5);
}
```

---

scaleY()
```
h1{
 border: 2px solid black;
  width: 300px;
  margin-left: auto;
  margin-right: auto;
  margin-top: 40px;
  background-color: aqua;
  transform: scaleY(5);
}
```
Scale(x,y)

Skew(): a transformation that skews an element on the 2D plane
skewX()

```
h1 {
  border: 2px solid black;
  width: 300px;
  margin-left: auto;
  margin-right: auto;
  margin-top: 40px;
  background-color: aqua;
  transform: skewX(110deg);
}
```

---

skewY()

```
h1 {
  border: 2px solid black;
  width: 300px;
 margin-left: auto;
  margin-right: auto;
  margin-top: 40px;
  background-color: aqua;
  transform: skewY(120deg);
}
```

Skew(x,y)

```
h1 {
  border: 2px solid black;
 width: 300px;
  margin-left: auto;
  margin-right: auto;
  margin-top: 40px;
  background-color: aqua;
  transform: skew(110deg, 210deg);
}
```

---

matrix():-
The matrix() function is specified with six values.
syntax:-
matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())

eg:-

```
#div1 {
  transform: matrix(1, 2, -1, 1, 80, 80);
  background-color: pink;
}
```

Transitions

CSS transitions allows you to change property values smoothly over a given duration

List of transition properties
Property
transition

A shorthand property for setting the four transition properties into a single property

---

transition-delay

Specifies a delay (in seconds) for the transition effect

---

transition-duration

Specifies how many seconds or milliseconds a transition effect takes to complete

---

transition-property

Specifies the name of the CSS property the transition effect is for particular element

---

transition-timing-function

Specifies the speed curve of the transition effect

Transition
Syntax :

```
 transition:  width  2s  ease;
   transition: all 0.3s ease-in;
```

Transition-delay : refers to how long you want to wait before starting the duration. When the transition effect will start. This value is written in seconds/milliseconds. 3s

transition-delay: 5s;
transition-delay: 15ms;

Transition-duration : refers to the duration of the transition
transition-duration: 3s;

Transition-property: refers to the CSS property you wish to transition.
transition-property: rotate;
:backgroundcolor,height,width

Transition-timing-function: refers to how to transitions occurs.
All transitions have a value of linear by default, which means the property changes evenly until the end of the transition.
And it describes about the speed curve of the transition effect.
It helps to change the speed of an transition over the duration.

Syntax transition-timing-function: linear|ease,ease-in,ease-out,ease-in-out;

Transition timing
function
Description

Linear
Specifies a transition effect with the same speed from start
To end
---
ease
Default value. Specifies a transition effect with a slow start, then
Fast, then end slowly.

Ease-in

Specifies a transition effect with a slow start.

Ease-out
Specifies a transition effect with a slow end.

Ease-in-out
Specifies a transition effect with a slow start and end.

```
.card {
  width: 100px;
  border: 10px solid teal;
  padding: 10px;
  margin-top: 50px;
  background: corol;
  margin-left: auto;
  margin-right: auto;
  height: 50px;
  transition: background-color 2s ease-in-out;
  /* ease-in-out slow start slow end */
}
.card:hover {
  cursor: pointer;
  background-color: aquamarine;
  border: 10px solid crimson;
}
```

Flexbox

Flexlayout is based on flex flow directions
-->Block , for sections in a webpage
-->Inline for text
-->The main idea behind the flex layout is to give the container the ability to alter its items width/height (and order) to best fill the available space (mostly to accommodate kind of display devices and screen size).
Properties

Display : flex | inline-flex ;
Flex-direction : row |column | row-reverse | col-reverse
Flex-wrap : wrap | nowrap |wrap-reverse
Justify-content : flex-start | flex-end |center| space-around | space-between | space-evenly
Align-items : flex-start | flex-end | center
Align content : flex-start | flex-end | center

Gradients

The Gradient in CSS is a special type of image that is made up of progressive & smooth transition between two or more colors.

There are 2 types of gradient

Linear-gradient (goes down/up/left/right/diagonally)

Background-image: linear-gradient(direction, color-stop1, color-stop2,…);
Background - image: linear-gradient(yellow, green)
It includes the smooth color transitions to going up, down, left, right, and diagonally.
The minimum two-color required to create a linear gradient.
More that two color elements can be possible in linear gradients.

The staring point and the direction are needed for the gradient effect.

Radial Gradient (defined by their center):

Background - image: radial- gradient(shape size at position, start-color,…., last-color);
The radial-gradient () function is an inbuilt function in css which is to set a radial gradient as the background image,
It starts at a single point and emanates outward.
By default, the first color starts at the center position of the element and then fade to the end color towards the edge of
The element.
Fade happens at an equal rate until specified.
By default, shape is ellipse, size is farthest-corner, and position is center.
Background-image: radial-gradient(circle, orange, yellow, green)

Uses of different size keywords in gradient
The size parameter defines the size of the gradient. It can take four values:

Closest-side
Farthest-side
Closest-corner
Farthest-corner

Position

It sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements.
There are five values the position property can take. They are:
Static-(by default)it will not take any properity like top, left, bottom, right
Relative - fixed- we can fix top right or bottom(it will take whitespace)
Absolute- for element , we can change
Fixed - position will be fixed cannot move to other position
Sticky - its a combination of fixed and relative

Note:
If the position property is set to relative, absolute or fixed
you can set the top, right, bottom and left properties
If the position property is set to static top, right bottom and
left properties.

Description

static

Normal position for the element(where top ,right, bottom, and left
have no effect)

Div { position: static;}
---
relative

Position the element absolutely relative to its conatiner

Div { position: absolute; top:10px;left:15px; }
---
absolute

Position the element absolutely relative to its container.
---
fixed

Position the element relative to the screen's viewport and stay fixed on
screen when scrolling

Div { position : fixed; top:10px;left:15px
