Dayeeu(:

Cascading Style Sheets .....

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 a lot of redundancy.

eg :- `<p style = "color : blue ;">hello </p>`

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 seperate css file which contains only style property with the help of tag attributes.

(For example class, id, heading,... etc).

- CSS property written in a seperate 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
   a. ID (#)
   b. Class (.)
   c. Element Selector (Tags name / Element name)
   d. Grouping Selector (,)
   e. 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;
     }
```

```
<P1 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 seperate 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.

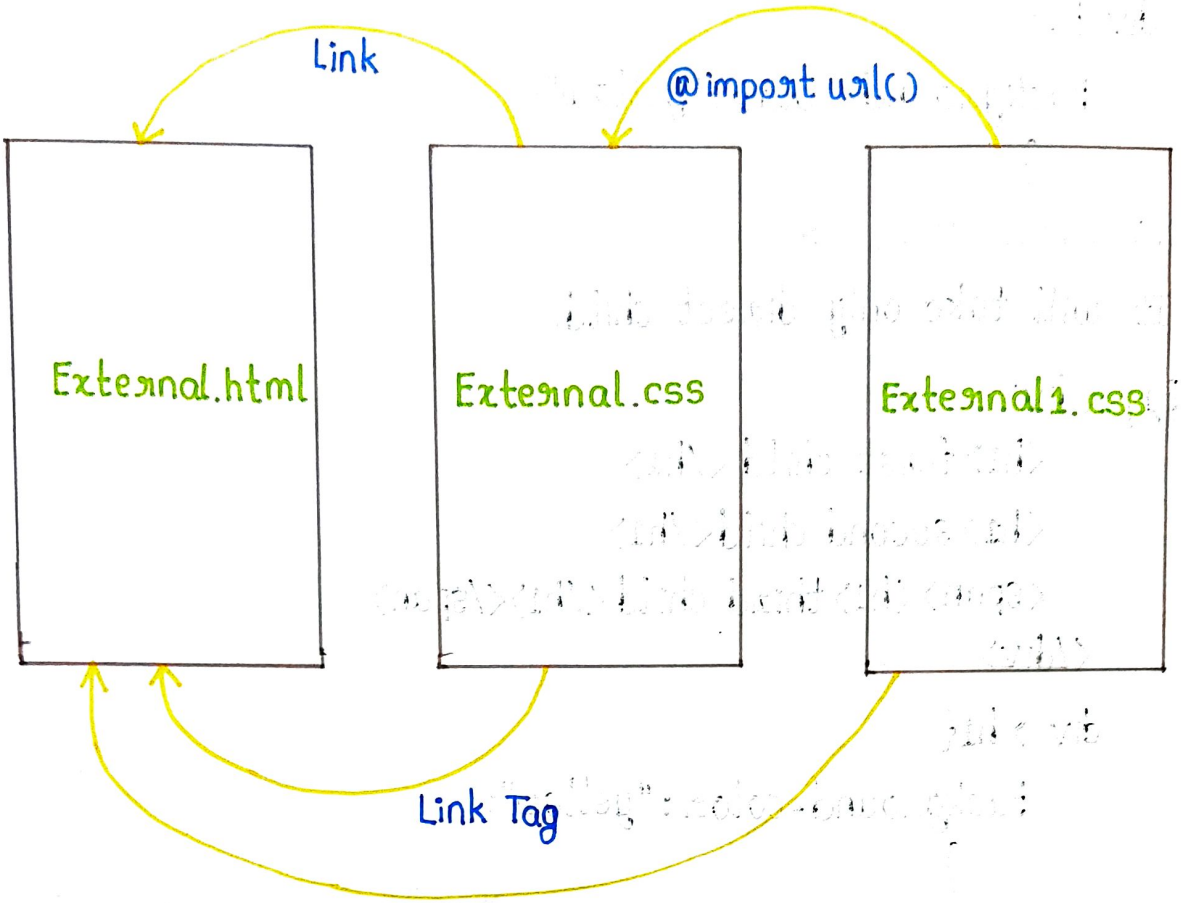
It selects all the elements on the pages.

--> When u want to use similar css to all tags.

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

/* Priority order : */

ID → Class → Element



---

2. Combination selector :-

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

There are four different combinators in CSS :

a) Descendant selector (space)

b) Child selector (>)

c) Adjacent sibling selector (+)

d) General sibling selector (~)

a) Descendant selector (space) :-

→ It will take all the direct child and in-direct child.

eg :-
```
<div>
    <h1> first child </h1>
    <h1> second child </h1>
    <span><h1> third child </h1></span>
</div>
```

```
div h1{
    background-color : "yellow";
      }
```

b) child selector (>) :-

⟶ It will take only direct child.

Eg:
```
<div>
   <h1> first child </h1>
   <h1> second child</h1>
   <span><h1> third child </h1></span>
</div>

div > h1{
    background-color : "yellow";
      }
```

c) Adjacent sibling selector (+) :-

⟶ It will select adjacent sibling or immediate sibling right after the first sibling.

Eg:
```
<div>
   <h1> first child </h1>
   <h1> second child </h1>
     <span><h1> third child </h1></span>
</div>

<div>
<h1> first sibling </h1>
<h1> second sibling </h1>
</div>

div + h1{
    background-color : "yellow";
      }
```

d) General sibling selector (~):-

⟶ It will select all the siblings right after it.

Eg:
<h1> first child </h1>
<h1> second child </h1>
<h1> third child </h1>

<h1> first sibling </h1>
<h1> second sibling </h1>

```
div ~ div h1{
    background-color : "yellow";
        }
```

---

3. 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 consectutetr adipisicing eit. Velit
  recusandae libero maxime voluptatibus saepe tempore, fugiat
  soluta ad sit fugit,
</p>

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

:: first-letter pseudo element :-

⟶ It is used to style only first-letter/character in a content.

```
<p>
  Lorem ipsum dolor sit amet consectetur adipisicing elit.
  Velit recusandee
</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;
  }
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit recusandae libero maxime voluptatibus sapae tempore, fugiat

Eg:-

```
p::before{
  content: url("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTtZFJN1rG7233vqSCo-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::after{
    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
</P>

P::selection{
color: blueviolet;
background-color: aquamarine;
}
```

4. Pseudo-class Selector:-

→ A css pseudo-class is a keyboard 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 ;
}
```

III) 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 style property once user clicks on any input fields.

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

VI) 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 child 4 </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;
}
```

---

last-of-type :-

→ It will select only particular tag from the bottom of the body.

```
Eg: <body>
    <div>
    <p> first child </p>
    <p> second child </p>
    <p> third child </p>
    </div>
    <h1> h1 from body </h1>
    <p> p from body </p>
    </body>
  p: last-of-type{
      color: red;
  }
```

Text Property :-

color : color_name

Text-align : right, left, center

Text-transform : capitalize, upper case, lower case

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-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.

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 and 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 i.e., around the border area.

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 (non-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.

Transform :-

2D Transforms :

CSS transforms allow you to move, rotate, scale and skew an elements.

2D transformation methods :-

1. Translate() : 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. Negative values move elements to the left.

The second value moves the element down. Negative colors values moves element up.

2. Rotate (deg) :-

eg:

```
.rotated {
  transform : rotate (45 deg); /* Equal to rotateZ(45deg)*/
```

```
    background-color : pink ;
  }
```

3. 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.

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

```
}
```

scale(X,Y)

4. 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;
transform : skew(110deg, 210deg);
}
```

5. Matrix() :

The matrix() function is specified within 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 allow you to change property values smoothly over a given duration.

List of transition properties :-

- 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.

```
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;
: background-color, height, width
```

Transition-timing-function : refers to how to transition 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 ~~occurs~~ curve of the transition effect.

It helps to change the speed of the transition over the duration.

syntax :-

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

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: 100 px;
  border: 10 px solid teal;
  padding: 10 px;
  margin-top: 50 px;
  background-color: coral;
  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: 10 px solid indigo;
}
```

- 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

◉ 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 model wheel from 0 to 360. 0 is red, 120 is green and 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.

◉ 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 and 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 margin property is used to create space around the element i.e., around the border area.

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 starting 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, red);

Uses of different size keywords in gradients

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 property like top, bottom, left, right.

Relative - fixed - we can fix top right or bottom (it will take white space).

Absolute - for element, we can change

Fixed - position will be fixed cannot move to another 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 container.

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: 10 px; left: 15px;}

◉ Flexbox

Flex layout 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 accomodate 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