Cascading Style Sheets (CSS)

- Introduction
- History
- ways of adding the CSS
- comments
- priorities of CSS
- selectors
- Border Properties
- Box model
- colours
- Background Properties
- Gradients
- Text and font Properties
- Filter Property
- Display Property
- Units
- Position Property
- Float Property
- Flex
- Grid
- Transition
- Transform
- Animation
- Media Queries

Introduction :-

→ CSS stands for cascading style sheets

→ cascading refers to pouring, here the styles are poured upon Html elements.

→ CSS is mainly used to provide stylings and alignment to our webpage

→ In general CSS will make our web pages look attractive

History :-

→ CSS was developed by Hakon Wium Lie in 1994

CSS 1.0 → 1996

CSS 2.0 → 1998

CSS 3.0 → 2000

NOTE :- currently we are using CSS 3 version.

Ways of Adding CSS

There are 3 ways of adding the css to our html elements

1) Inline CSS
2) Internal CSS
3) External CSS

1) Inline Cascading style sheets :-

It refers to adding the style in the same line of html tag using style attribute

syntax :- style = "property: value;"

```
<h1 style = "color: □ white; background-color: □ red;">
    EE SALA CUP NAMDU </h1>
```

2) Internal CSS :-

It refers to adding styles in the same html file inside the head tag by specifying style tag

```
<style>
    selector {
        property: value;
        }
</style>
```

Ex:

```
<!Doctype Html>
<Html lang="en">
<head>
    <title> ways of adding css </title>
    <style>
      input{
          color: □ white;
          background-color: □ Black;
      }
      P{
          color: □ white;
          background-color : □ purple
      }
    </style>
</head>
<body>
    <input type="text">
    <p> Lorem 20 </p>
</body>
</Html>
```

3) External CSS :-

It refers to adding styles by maintaining a seperate file saved with .css extension.

syntax :- selector {
property : value;
}

-> To establish connection between Html and css file specify link tag inside the head tag

```
<head>
  <link rel="stylesheet" href="path of css file">
</head>
```

Ways of Adding CSS · Html

```
<Head>
  <title> Ways of Adding
          CSS </title>
  <link rel="stylesheet"
   href="./External·css">
</Head>
<body>
  <b> Iam bold </b>
  <h6> Heading-6 </h6>
</body>
```

External · CSS

```
b {
  color: □ white;
  background-color:
      □ green;
}
h6 {
  color: □ white;
  background-color:
      □ blue;
}
```

Priorities of CSS :-

→ Amoung the 3 ways of adding the CSS, first highest priority is given to Inline CSS

→ Amoung internal and external CSS priorities are decided based on the latest value inside the head Tag.

NOTE :- To override the priorities we use a value called !Important

```
h1 {
    color: blue !important;
}
```

Comments :-

comments are the piece of code that is not executed on the browser

comments are mainly used to improve the readability of the code

syntax :- /* comment */

Selectors :-

They help to target the Html Elements upon which stylings needs to be added.

Types of selectors :-

There are 5 types of selectors in CSS

1) Simple selectors
2) Combinator selectors
3) Pseudo class selectors
4) Pseudo Element selectors
5) Attribute selectors

1) Simple selectors :- These selectors are used to target the elements based on id (#), class, tagname, grouping and universal selectors.

(i) id (#) :- It is used to target one unique element within our webpage

id cannot have multiple values

• Html	• CSS
<h1 id = "special"> Beula </h1>	#special {
Kayadu Lohar	color: white;
Bagyasree	background-color: blue;
	}

(**) class (.) :- It is used to target multiple elements and apply common stylings.

class can have multiple values

• Html

```
<h1 class="common">HTML</h1>
<p> Lorem10 </p>
<h2> class="common something"> CSS </h2>
<p> Lorem10 </p>
<h3 class="common"> JS </h3>
<p> Lorem10 </p>
```

• CSS

```
.common {
    color: white;
    background-color: green;
}
.something {
    text-align: center;
}
```

(iii) Tag Name :- It is used to target the elements based on the name of a tag

```html
<input type="email">
<input type="password">
```

```css
input {
    color: white;
    background-color: blue;
}
```

(iv) Grouping (,) :- It allows to apply the same styles to multiple selectors like id, class, Tagname etc by grouping them together in a single rule.

```html
<h1> HTML2 </h1>
<p> Lorem 10 </p>
<h2>class = "common" > CSS</h2>
<p> Lorem 10 </p>
<h3 id = "common" > JS </h3>
<p> Lorem 10 </p>
```

```css
h1, .common,
#common {
    color: white;
    background-color:
        green;
}
```

(v) Universal (*) :- It is used to target all the elements in a Html document.

```css
* {
    color: white;
    background-color: hotpink;
}
```

Priorities of simple selectors :- Among the type of simple selectors the priorities are as follows :-

1) id 2) class 3) Tagname 4) Universal

2. Combinator Selector :-

These are used to target the elements based upon the relationship with other elements.

Relationship can be parent child and sibling.

Types :-

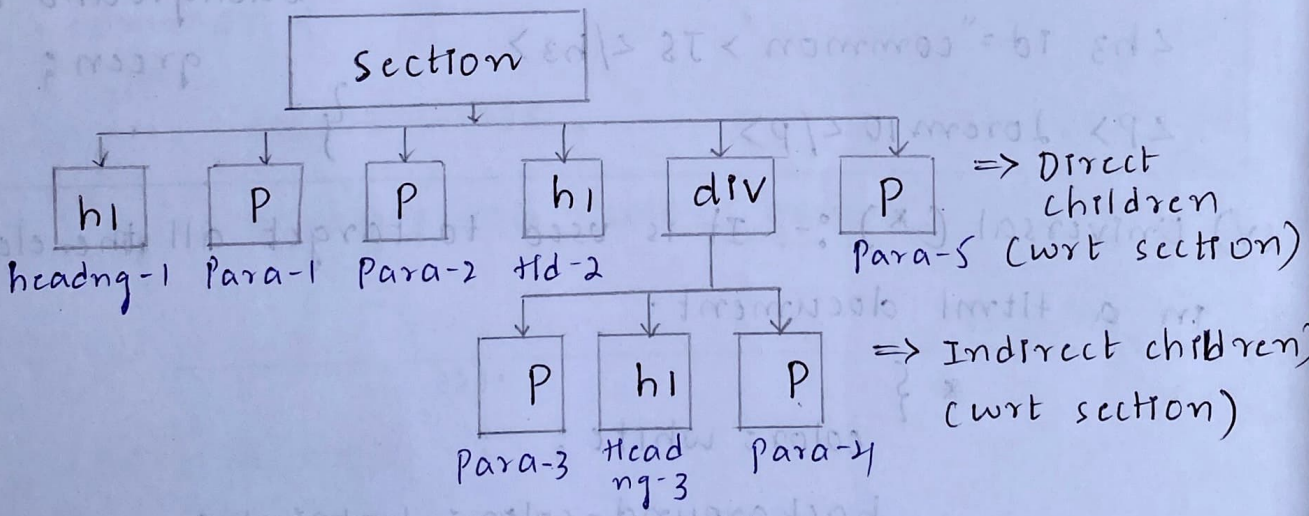
There are 4 types of combinator selectors in CSS

(I) Descendent selector (space)

(II) child selector (>)

(III) Adjacent sibling selector (+)

(IV) General sibling selector (~) Tilde



(I) Descendent selector :- This selector is used to target both direct and indirect childrens of given parent element

Syntax :-

```
parent element  child element
{
    property : value ;
}
```

Examples:-

```
section h1 {
    color: white;
    background-color:
        blue;
}
```

```
div p {
    color: white;
    background-color:
        green;
}
```

(iii) child selector :- This selector is used to target only the direct children of the given parent element.

syntax:-

```
parent element > child element
{
    property: value;
}
```

Example:-

```
section > p {
    color: white;
    background-color:
        blue;
}
```

```
div > p {
    color: white;
    background-color:
        green;
}
```

(iv) Adjacent sibling selector :- This selector targets immediate next element for the given sibling element

syntax:-

```
sibling element + Target element
{
    property: value;
}
```

Example:-

```
h1+P {
    color: white;
    background-color:
        green;
}
```

```
p+p {
    color: white;
    background-color:
        yellow;
}
```

(iv) General sibling selector : This selector targets all of the upcoming elements or siblings for the given sibling element.

Syntax:- sibling element ~ target element

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

Example:-

```
p~p {
    color: white;
    background-color:
        green;
}
```

```
p~h1 {
    color: white;
    background-color:
        brown;
}
```

3) Pseudo-class selectors :- They are used to target the elements whenever they are at special state

-> special state can be moving the curser upon a element, clicking on input field.

-> They are denoted by the symbol (:)

Types :-

(I) :hover :- It targets the element whenever the mouse curser is moved upon it.

```
button:hover {
    color: white;
    background-color: green;
}
```

(II) :focus :- It targets the input fields when it is clicked upon it.

```
input:focus {
    color: white;
    background-color: green;
}
```

(iii) : active :- It targets the anchor tags whenever they are in active states

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

(iv) : visited :- It targets the anchor tags whenever they are in visited state

```
a: visited {
    color: green;
}
```

(v) : first-child :- It is used to target the first child element of the given parent Element.

```
div > p : first-child {
    color: white;
    background color: green;
}
```

(vi) : last-child :- It is used to target the last child element of given parent Element.

```
div > p : last-child {
    color: white;
    background-color: green;
}
```

(VII) : nth-child :- It is used to target specific child element of given parent element based on value specified

Examples :-

```
div>p: nth-child (5) {
        color: white;
        background-color: Yellow;
        }

div>p : nth-child (3n) {
        color: white;
        background-color: brown;
        }

div>p: nth-child (even) {
        color: white;
        background-color: green;
        }

div>p: nth-child (odd) {
        color; white;
        background-color: green;
      }
```

*: shortcut for writing 10 para tags

```
div>p * 10 { ... }
```

(VIII) : checked :- It is used to target radio buttons and checkboxes during checked state

• Html

```
<form>
  <label> Gender: </label>
  <input type="radio" name="gender" id="male">
  <label for="male"> male </label>
  <input type="radio" name="gender" id="female">
  <label for="female"> female: </label>
</form>
```

• CSS

```
input: checked + label {
        color: red;
      }
```

(IX) : NOT() :- It allows to exclude specific elements from being styled

• Html

```
<ul>
  <li> neha </li>
  <li> sneha </li>
  <li class="exclude"> indu </li>
  <li> Pooja </li>
</ul>
```

• CSS

```
li: not(.exclude) {
  color: white;
  background-color:
          green;
}
```

4) Pseudo-Element Selectors :- It is used to target the specific portion of the content

It is denoted by the symbol :: (double colon)

Types

(i) ::first-letter :- It is used to target the first-letter from given content.

```
p::first-letter{
    color: red;
    font-size: XX-large;
}
```

(ii) ::first-line :- It is used to target the first-line from given content

```
p::first-line {
    color: white;
    background-color: maroon;
}
```

(iii) ::before :- It is used to place the content before the targeted html element.

```
p::before{
    content: "How are you";
}
```

(iv) :: after :- It is used to place the content after the targetted html element

```
p:: after {
    content: "emoji";
}
```

(v) :: selection :- It is used to style the content when the cursor is dragged upon it

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

(vi) :: marker :- It is used to target the 1st type

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

(vii) :: Placeholder :- It is used to target the placeholder of an input field.

```
input:: placeholder {
    color: green;
    fontsize: xx-small;
}
```

5) Attribute selectors :- They are used to target the element based upon attribute name or along with its value

They are denoted by the symbol [ ] (square braces)

Types of Attributes :-

(I) [attr] :- It is used to target the element based on the name of attribute

```
[type] {
    color: white ;
    background-color: golden ;
}
```

(II) [attr = value] :- It is used to target the elements based on name of the attribute along with its value

```
[alt = "Anushka"] {
    border-radius : 50% ;
}
```

Border Properties :-

Border :- It is a line that surrounds the content

→ To add a border for Html element we have three basic properties

(I) Border-width

(II) Border-style

(III) Border-color

(I) Border-width :- It is used to specify the thickness for an elements border

Example :- border-width : 10px ;

To specify border width for specific sides we have the properties like

```
border-top-width : 10px;
border-top-width : 20px;
border-right-width : 30px;
border-left-width : 40px;
```

(II) Border-style :- It is used to specify the style for an

It can take the values like

```
border-style : none;
border-style : solid;
border-style : dashed;
border-style : dotted;
border-style : double;
```

```
border-style : groove;
border-style : ridge;
```

-> To set border-style for the specific side we have the properties like

```
border-top-style : solid;
border-right-style : dashed;
border-bottom-style : dotted;
border-left-style : double;
```

(III) Border-color :- It is used to specify the color for an elements border

Example :- `border-color : white;`

-> To set border-color for a specific sides we have the properties like

```
border-top-color : red;
border-right-color : green;
border-bottom-color : Yellow;
border-left-color : maroon;
```

-> Border :- It is the shorthand property for border-width, border-style and border-color.

Examples :- `border : 7px double green;`

`border : maroon 10px dashed;`

→ Border-radius :- It is used to add the curved edges for an elements border

example :- border-radius : 90px;

→ To set the border-radius for a specific edges we have properties like

```
border-top-left-radius : px10;
border-top-right-radius : 20px;
border-bottom-left-radius : 30px;
border-bottom-right-radius : 40px;
```

NOTE :- To convert square into the circle set the border-radius property to 50%.

Box Model :-

Box model :- It is a concept that describes the structure (area) of an element on our webpage.

-> It is divided into four categories :-

(I) margin

(II) Border

(III) Padding

(IV) Content

(I) Margin :- It is the space around the elements border

```
margin: 30px;
```

-> To set margin for a specific side we have the properties like

```
margin-top: 10px;
margin-Right: 20px;
margin-bottom: 30px;
margin-left: 40px;
```

-> Margin can take negative values which will make them move in the opposite direction.

```
margin-top: -10px;
```

-> Margin can totally take four values each effecting on top, right, bottom and left side

margin: 10px 20px 30px 40px ;
T R B L

→ If a value is missing it will take or consider from the opposite side

margin: 10px 20px 30px ;
T R/L B

margin: 10px 20px ;
T/B R/L

margin: 10px ;
All sides

(II) Border :- It is a line that passes between the elements margin and padding.

Border can be customised in terms of

Border-width, Border-style, Border-color.

Example :- border: 5px solid black ;

(III) Padding :- It is the space between elements border and content

Example :- padding: 10px ;

→ padding cannot take negative values

→ To set Padding for specific side we have properties like

padding-top: 10px ;
Padding-right: 20px ;
Padding-bottom: 30px ;
Padding-left: 40px ;

→ padding can take totally four values each effecting on top, right, left, bottom

```
padding : 10px 20px 30px 40px ;
          T    R    B    L
padding : 10px 20px 30px ;
          T    R/L  B
padding : 10px 20px ;
          T/B  R/L
padding : 10px
          All sides
```

(iv) Content :- It is the innermost part of an element which includes the actual data displayed such as text, images etc

→ contents area is defined by the properties like height and width.

```
Example :- width : 200px ;
           height : 300px ;
```

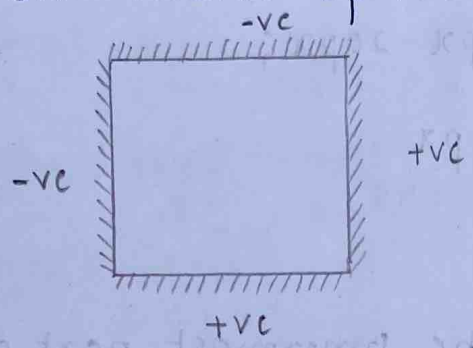
→ Box-sizing :- It defines how the width and height should be calculated for an element

→ Box-sizing property takes in 2 different values i.e content-box (default) and border-box

- box-sizing : content-box ; - height & width will be taken only for content
- box-sizing : border-box ; - height & width will be taken till the border (content + padding + border)

→ Box-shadow :- It adds shadow effects around an elements frame

box-shadow: horizontal-offset vertical-offset blur-radius spread-radious color inset



Colors :-

→ In Css colors can be specified in 4 different ways

1) Named colors
2) RGB values
3) RGBA values
4) Hexcode / Hexadecimal values

1) Named Colors :- Css provides a set of limited predefined colors to choose from like red, yellow, etc

Ex :- background-color: black;

2) RGB values :- RGB stands for Red, Green, Blue where each color can have the values between the range 0-255 where

0 → no color (lightest)
255 → full color (darkest)

Ex :- background-color : rgb (0,255,0);

background-color: rgb (17, 224, 235)

3) RGBA values :- RGBA standards for Red, Green, Blue and Alpha

→ Alpha value varies between 0-1 which decides the opacity or transparency of color.

0 → Transparent
1 → opaque (full color)

Ex:- background-color : □ rgba (0,0,0,1);

background-color : □ rgba (12,0,0,0.629);

4) Hexcode / Hexadecimal values :

→ Hexcode is a six-digit code that represents a color using the hexa-decimal (base 16) number system.

→ It is preceeded by # symbol

Example:- # RRGGBB ;

→ They can have the values between the range 0-9 and where, 0 → no color

F → full color

Ex:- background-color : □ # ffffff ;

background-color : □ # 00f ;

Background Properties

In CSS to manipulate the background of an element we have several background properties.

1) background-color;
2) background-image;
3) background-repeat;
4) background-position;
5) background-attachment;
6) background-size;
7) background;

1) Background-color :- It is used to set background-color for an element.

-> It can take values in terms named colors, rgb and rgba values and Hexcodevalues.

syntax: background-color : # ff13d8;

2) Background-image :- It is used to set the image for an elements background

syntax :- background-image : url("path of img");

3) Background-repeat :- This property is used to manipulate the repeating behavior of a background-image

-> It can take values like

background-repeat : repeat; (default)

background-repeat : repeat-x;

background-repeat : repeat-y;

background-repeat : no repeat;

4) Background-position :- It is used to set the position of a background-image within its containing element.

syntax :- background-position : x-axis y-axis ;

-> X-axis and Y-axis values can be specified interms of pixel units or predefined values like left, right, bottom, top and center.

-> If there is only one value specified by default the second value is set to center.

```
background-position : top left;
background-position : center;
background-position : 200px 100px;
background-position : 200px;
```

5) Background-Attachment :- This property defines how the background-image should be attached when the page is scrolled

```
background-attachment : scroll; (default)
background-attachment : fixed;
```

6) Background-size :- It is used to set the dimensions of a background-image

```
background-size : width height;
```

-> It can take the values in terms of pixel or predefined values like cover and contain

-> If only one value is specified, the second value is considered as auto.

```
background-size : 300px 500px
background-size : 300px;
background-size : cover;
background-size : contain;
```

7) BackGround : It is the shorthand property for most of background properties like background-color, background-image, background-repeat, background-position and background-attachment.

```
background : rgba(0,0,0.21) url("path of img")
no repeat center fixed;
background-size : 400px 400px
```

Gradients :-

-> These are used to add smooth transition between two or more colors.

-> They are commonly used for properties like

background : or background-image

Types :- There are three types of gradients

1) linear gradient

2) Radial gradient

3) conic gradient

1) linear gradient :- It creates a transition between the colors along a straight line

syntax :- background : lineargradient (direction, col1 col2 ....)

Examples :-

background : linear-gradient (pink, green, brown);

background : linear-gradient (to bottom left, midnight, salmon, yellow);

background-image : linear-gradient (to bottom, orange 0% 20%, white 20% 50%, green 50% 60%)

background-image : repeating-linear-gradient (to top left, orange 0% 1%, green 2% 3%);

2) Radial-gradient :- It creates a transition between the colors radially from a center point.

syntax :- background : radial-gradient (shape, col1, col2.....)

Examples :-

background : radial-gradient (voilet, red, green);

background : radial-gradient (ellipse, voilet, red ...);

background : radial-gradient (circle, voilet 0% 2% indigo 2% 4%, blue 4% 6% .....)

background : repeating-radial-gradient (circle, voilet 0% 2%, indigo 2% 4%, green 4% 6% ....);

3) Conic-gradient :- It creates a transition between the color along the conical shape

-> In real time conic gradients are used to create pie charts.

syntax :- background : conic-gradient (col1, col2.....)

Examples :- background : conic-gradient (tomato, lime, silver);

background : conic-gradient (tomato 0% 20%, lime 20% 30%, silver 30% 60%)

background : repeating-conic-gradient (red 0% 2%, black 2% 4%);

Text Properties :-

1) Text-decoration-line :-

-> It is used to add a decorative line for the specified content.

-> It can take the values like

Example :- text-decoration-line : overline/ underline / line-through /none ;

2) text-decoration-style :-

-> It is used to set the style for a specified decorative line

-> It can take values like

Example :- text-decoration-style : solid / dashed / double / dotted / wavy ;

3) text-decoration-color :-

-> It is used to set the color for the specified decoration line

-> It can take the values like

Example :- text-decoration-color : orange / rgb(255, 30, 10) / #ffffff ;

4) text-decoration-thickness :-

→ It is used to set the thickness for the specified decorative line

→ Example : text-decoration-thickness : 2px ;

5) text-decoration :-

→ It is shorthand property for text decoration line, text-decoration-color, text-decoration-style, text-decoration-thickness

Example :- text-decoration : 5px line-through wavy orange ;

→ Text-decoration-line is given more priority compare to other text-properties.

→ To remove the underline for hyperlink in anchor Tags use Text-decoration-line : none ;

6) text-indent :-

It is used to add the spacing before the starting of a content

Example :- text-indent : 300px ;

7) Text-transform :-

→ It is used to convert the content into uppercase or lowercase or captalise form (starting letter of each word is in uppercase)

Example :- text-transform : uppercase / lowercase / captilise

8) word-spacing :-

→ It is used to add the space between the word by default. It takes the values are normal.

Example: word-spacing : normal / 10px ;

9) letter-spacing :-

→ It is used to add space between the letters. It takes the value as normal.

Example :- letter-spacing : normal / 30px ;

10) line-height :-

→ This property is used to specify the space between the lines

→ By default it takes value as normal

Example :- line-height : normal / 200px ;

11) text-align :-

→ It is used to set the horizontal alignment for text

→ It can take the values like

Example :- text-align : left / center / right / justify ;
whole para

12) text-align-last :-

→ It is used to set the horizontal alignment for the last line of a text

→ It can take the values like

Example :- text-align-last : left / center / right / justify ;
for last line of paragraph

13) text-shadow :-

→ It is used to add the shadow effect for text content

→ This property takes 4 values

Example :- text-shadow : x-axis y-axis blur color ;

∴ text-shadow : 10px 20px 10px blue ;

Font-Properties

1) Font-style :-

-> It is used to specify the style for given content

-> By default, it takes value as normal

-> It takes values like

Example :- font-style : normal / oblique / italic ;

2) font-size :-

-> It is used to define the size for given content

Example :- font-size : xxsmall / larger / 80px ;

3) Font-weight :-

-> It is used to add the thickness for given content

-> It take values ranges between 100 to 900 or predefined values like bold, bolder....

Example :- font-weight : 900 ;

4) font-family :-

-> It is used to specify the type of the font (hand writing)

-> The purpose of using multiple font family is to act like fallback font

site :- google fonts

Example :- font-family : cursive ;
font-family : monospace ;
font-family : 'Gill sans', 'Gill sans MT', 'calibri', sans serif ;

Filter Properties

Filter: Filter properties are used to apply graphical effects like blur, adding brightness, grayscale etc

1) filter: blur(): It is used to add the blur effect for an element (text or image)
   -> larger the value more the blur
   -> specify the value for blur in pixel units

   Example: filter: blur(10px);

2) filter: brightness():
   -> It is used to adjust the brightness for element
   -> we can give the values in percentage (0-100) or in range between 0-1

   Example:- filter: brightness(30%);

3) filter: grayscale():
   -> It is used to add grey color to a particular element
   -> we can give the value in percentage (0-100) or in range between 0-1

   Example:- filter: grayscale(0.9);
   filter: grayscale(40%);

4) filter: opacity():-

→ It is used to adjust the transparancy of an element

→ usually the value varies from 0-1

Example: filter: opacity(1);

5) filter: drop-shadow():-

→ using drop-shadow allows the element to add a shadow effect to edges of the image

Example:- filter: drop-shadow(10px 20px 30px green);

Overflow :-

→ In CSS the overflow property is used to control, what happens to the content of an element exceeds the space allocated to it.

→ There are several properties values that can be specified overflow property.

1) overflow: visible :- It is the default value for the overflow property in which the content will exceed the box or possibly overlapping with other elements.

2) overflow: hidden :- This value hides the content that exceeds the size of a container

3) Overflow: scroll :- It adds a scroll bar to the container allowing users to scroll and see the hidden content.

4) overflow: auto :- It will add the scroll bar to container only on required axis.

NOTE :- To add overflow property on specific axis we have properties like overflow-x, overflow-y

Display Properties

Display :- Display property is used to specify how the elements should behave, whether inline or block or inline-block

1) display: inline;

specifying this property for an element makes it to behave like an inline level element.

```
Example:- b{
            display: inline;
          }
```

2) display: block;

specifying this property for an element makes it to behave like a block level element

```
Example:- b {
             display: block;
           }
```

Difference between block level and inline level Elements

Block-Level Elements :-

-> Takes entire width of the webpage

-> Height and width are applicable

-> Takes margin and padding on all sides

Ex:- Heading Tags, Paragraph Tags, Semantic Tags.

Inline-Level Elements :-

→ Takes only the content width

→ Height and width are not applicable

→ Takes margin on left and right side but padding on all sides

Example:- formatting Tags, span, anchor tag, label tags.

3) display : inline-block :-

→ These are the elements that behave like inline but take block-level properties like height and width

Example :- input, img

```
div {
    height : 200px;
    width : 200px;
    border : 5px solid black;
    display : inline-block;
}
```

4) display: none :- when the property is set to display none, the element will be removed from the document flow and the space is not reserved

visibility : hidden :- when the property is set to visibility hidden, the element is hidden in the same place (docsnot appear) but the space is reserved.

Units :-

units :- units are used to specify measurements for various properties like width, height, font-size etc

Types :- In general there are 2 type of units in CSS

(i) Absolute units

(ii) Relative units

1) Absolute units :- These are fixed units and will not change their value based upon other elements

(i) Pixel (px) :

→ It is often used to set fixed sizes

→ Approximately 1px represents a single dot on the screen

→ 1 inch = 96 px

→ 2.5 cm = 96 px

2) Relative units :-

→ These are the units that changes its value with respect to other elements

(i) percentage (%) :- It changes its value wrt its parent element

Example:- If a child element has a width of 50% it means that the element is taking half width respect of its parent element

```
# parent {
        width: 500px;
      }

# child {
        width: 50%;  (i.e it takes 250px)
      }
```

(ii) vh :-

→ vh stands for viewport height

→ It changes its value with respect to viewport (webpage) height.

```
Example:- body {
            height: 100vh;
          }
```

(iii) vw :-

→ vw stands for viewport width

→ It changes its value with respect to viewport (webpage) width

```
Example: body { width: 50vh;
              }
```

(iv) em :-

-> em stands for element

-> It is mainly used to deal with the font-size which changes with respect to parent element.

-> By default 1em is equal to 16px

Example:- If the parent element is having a font size of 20px then 1em is equal to 20px.

(v) rem :-

-> rem stands for root element

-> It is mainly used to deal with the font size which changes with respect to the html tag (root Element)

-> By default the font size of html tag is set to 16px i.e 1rem = 16px

Example:- If html tag font size is set to 10px then 1rem = 10px.

Position Properties :-

position :- position property is used to align the elements on our webpage

→ There are several values that can be specified for position property

position : static, relative, absolute, fixed, sticky ;

→ To specify the exact position of an element css provides helper properties like

top, right, left, bottom

1) Position : static ;

→ By default all the elements are positioned with static value

→ It doesnot have the effect of properties like top, left, bottom, right and z-index

Example :-
```
div{
    position: static;
    top : 20px; (not effected)
}
```

2) position : relative ;

→ Elements will move wrt its current position

→ It takes negative values which moves in opp directn

top : -200px ; (goes upwards)

Example:-
```
div {
    position: relative;
    top: 200px;
    left: 550 px;
}
```

3) Position: absolute :-

→ Elements will move with respect to the nearest relatively positioned ancestor (parent)

→ with respect to the web page (viewport) when ever the parent element is not specified with position relatv

→ Appears as if they are floating on webpage

Example:-
```
# parent {
    height: 700px;
    width: 800px;
    background-color: yellow;
    margin: 40px;
    position: relative;
}
# child {
    height: 330px;
    width: 300px;
    background-color: blue;
    position: absolute;
    top: 20px;
}
```

4) position: fixed;

-> elements are positioned with respect to the webpage which means element stays fixed at specified position even if the page is scrolled

-> Elements appear as if they are floating on webpage

Example:- div {
height: 100px;
width: 100px;
background-color: green;
position: fixed;
top: 50px;
left: 50px;
}

5) position: sticky;

-> Elements are positioned with respect to the users scroll position

-> these elements behave like fixed after reaching a specific point

Example:- div {
height: 100px;
width: 100px;
background: blue;
position: sticky
top: 50px;
}

Z-index:-

-> This property decides the stack order of an element

-> This property only works with position property except static value

-> Elements with higher z-index value are placed at the top level

Example:-
```
#img1{
    position: absolute;
    z-index: 100;
}
#img2{
    position: absolute;
    z-index: 200;
```

NOTE:- specifying navigation value will make elements move in opposite direction

-> Amoung top and bottom, top is considered as highest priority in case of left and right, left is more prior

FLOAT:-

-> In css the float property specifies whether the element should be floated (placed) to right or left of its containing element (parent element)

-> It allows the content or next element to occupy its spce when specified with float property

-> Float property can take values: float: none/left/right;

Flex :-

-> It is also known as flex-box, which is used to create layouts

-> It is used to align the elements in one dimension i.e either row or column

-> To align the elements using flex, they need to have a common parent for which the 'display: flex' property needs to be applied.

Example :

<section> ——> parent Element —> display : flex —> also known as "flex container"

} —> child elements —> also known as flex items

</section>

Flex container properties :-

1) Display: flex; (display:)

-> To create flex box (1D layout) set the display property to flex for parent element i.e display: flex;

(ii) flex-direction :-

→ This property is used to change the direction of flex-items

→ It take values like :

flex-direction: row/row-reverse/column/column-reverse;

(iii) flex-wrap :-

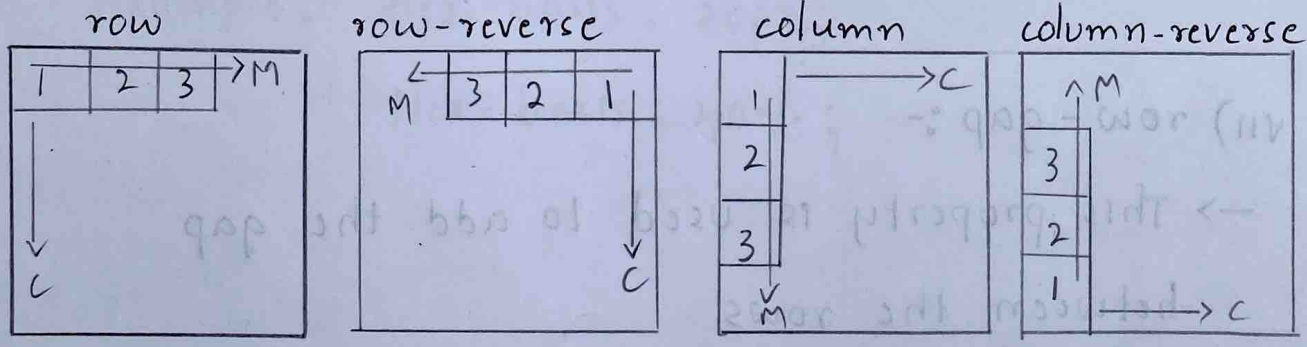
→ This property is mainly used to make the flex-items to take their original dimensions and try to adjust as much as possible inside the flex-container.

→ It take values like :

flex-wrap: nowrap/wrap/wrap-reverse;

Main axis and cross axis :-

→ Main Axis is the direction in which the elements are flowing where as cross Axis is the direction that is perpendicular (⊥) to the main axis.



(iv) justify-content :

→ This property is used to align the flex items along the main axis

→ It can take values like

justify-content: flex-start / center / flex-end / space between / space-arround / space evenly;

v) align-items :

→ This property is used to align flex items with respect to cross axis

→ It can take values like

align-items : flex-start / center / flex-end ;

vi) column-gap :-

→ This property is used to add the gap between the columns.

Example : column-gap : 20px ;

vii) row-gap :-

→ This property is used to add the gap between the rows

Example : row-gap : 20px ;

viii) gap :

—> It is the short hand property for row gap and column gap.

syntax: gap: row-gap column_gap;

Example: gap: 20px 30px;

gap: (20px) 30px; —> takes for both column gap and row gap.

Flex-items-property :

—> It is used to change the dimensions of flex items depending on the flex direction

—> If the flex direction is column, flex-basis acts like height

—> If the flex direction is row then the flex-basis acts like width

Example:- flex-basis : 20px;

flex-basis : 40%;

Grid :-

-> It is mainly used to create layouts

-> It is used to align the elements in 2D ie both rows and columns

-> To align the elements using grid they need to have a common parent to which display grid property should be displayed.

<section> ------> parent Element —> display: grid; —> grid container

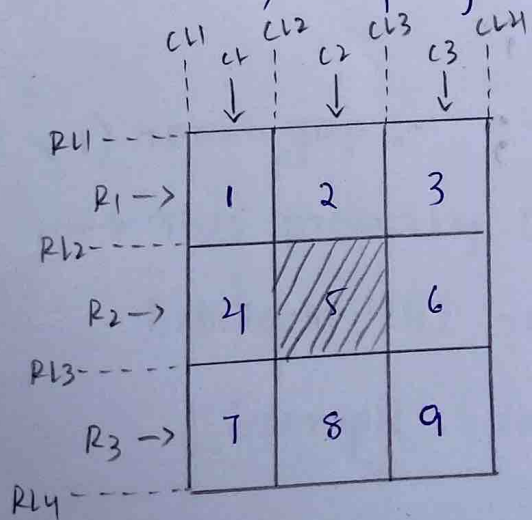
} child Elements } Grid-items

</section>

Grid container properties :-

1) Display :- To create a 2D layout, set the display property to grid for parent element.

-> display: grid;

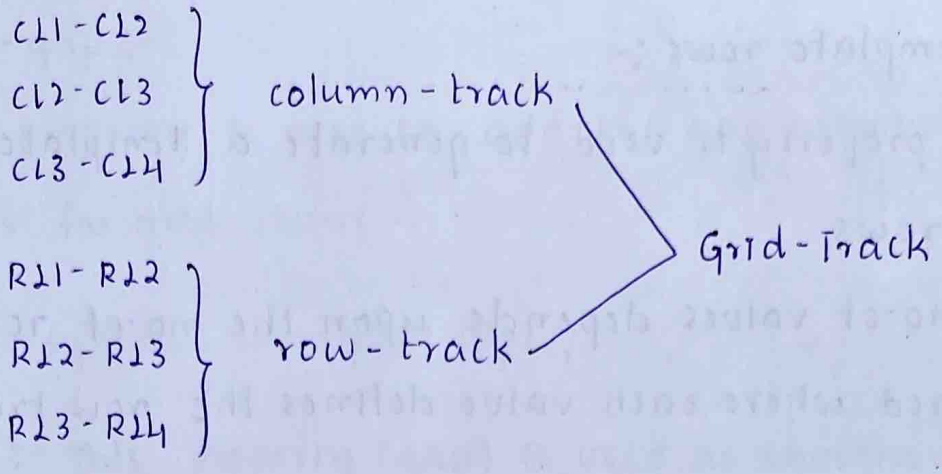


-> grid-cell

CL1, CL2, CL3, CL4 -> column lines

R21, R22, R23, R24 -> row lines

} Grid-lines.



2) Grid Template columns :-

-> This property is used to generate a template (layout) wrt columns

-> The no·of values depends upon the no·of columns required where each value defines the column(gap) track

Example :- To generate a template of 2 columns we need to specify 2 values where each value defines track size

```
grid-template-column: 250px 250px;
```

In the above scenario when the values are repeating we can create template using repeat()

```
grid-template-columns: repeat(2, 250px);
```

To create a grid template with 2 columns when the first track size is fixed and second track size should be taken with the remaining space left out within the parent Element we specify

```
grid-template-columns: 250px auto;
```

3) Grid Template rows :-

-> This property is used to generate a template (layout) wrt rows.

-> The no·of values depends upon the no·of rows required where each value defines the row track

-> Example :- To generate a template of 2 rows we need to specify 2 values where each value defines tracksize

```
grid-template-row: 250px 250px;
```

-> In the above scenario where the values are repeating we can create template using repeat()

```
grid-template-row: repeat(2,250px);
```

-> To create a grid template with 2 rows when the first tracksize is fixed and second tracksize should be taken with the remaining space left out within the parent Element we specify

```
grid-template-row: 250px auto;
```

4) column-gap :-

-> This property is used to add the gap between the columns for grid-items

```
column gap: 20px;
```

5) row-gap :-

-> This property is used to add the gap between the rows for grid-items

syntax: row-gap: 10px;

6) gap :- This property (gap) is used as shorthand property for rowgap and columngap

gap: 10px 20px;

gap: 10px; (takes for both row/column gap)

Grid-item Properties :-

1) grid-column-start :- This property is used to define grid item from which column line it should start with

Example :- grid-column-start: 1;

2) grid-column-end :- This property is used to define grid item from which column line is should end.

Example :- grid-column-end: 2;

3) grid-column :- It is shorthand property for grid-column-start and grid-column-end.

Example :- grid-column: 1/3;

4) grid-row-start :- This property is used to define grid item from which row line should start

Example :- grid-row-start : 1 ;

5) grid-row-end :- This property is used to define grid item from which row line should end

Example :- grid-row-end : 3 ;

6) grid-area :- It is shorthand property for grs, gcs, gre, gce

Example :- grid-area : grs/ gcs/ gre/ gce ;

grid-area : 1/1/3/3 ;

Transition :-

css transition helps to add smooth changes from one state to another state over a duration of time.

some of the transition properties are :-

1) Transition - property :- It refers to name of the css property to which we want to apply the smooth changes

In general we specify the value as all

Example :- Transition - property : background - color
border radius

Transition - property : all

2) Transition - duration :- It specifies how many seconds or milliseconds should be taken for transition.

It can take values in terms of seconds or milliseconds

Transition - duration : 5s

Transition - duration : 500ms

3) Transition-delay :- It specifies amount of time to wait before starting transition effect.

It takes values interms of seconds or milliseconds.

```
Transition-delay: 2s;

Transition-delay: 200ms;
```

4) Transition-timing

This transition decides speed of transition effect

It can take values like

ease : default, slowstart, speedsup, slow-end

ease-in : slow start, speedsup

ease-out : speedsup, slowend

ease-in-out : slowstart, constant speed in middle, slow end

linear : constant speed from start to end.

5) Transition :- It is shorthand property for Transition property, Transition duration, Transition delay, Transition time function.

```
Transition: all 1s 200ms ease;
```

Transform :-

-> Transform properties are used to change the appearance of the element on our webpage.

-> The property used for transformation is transform

-> some of the values that can be specified for transform property are :-

1) rotate :- This will make the element to rotate based upon specified degree value

-> If the degree value is positive it rotates in clock-wise direction and if degree value is negative it rotates in anti-clockwise direction

Example :- transform : rotate (55 deg);
transform : rotate (-55 deg);

-> To make the element to rotate only with respect to x-axis or y-axis we have properties like

rotate x :- transform : rotate x (125 deg);

rotate y :- transform : rotate y (90 deg);

2) Translate :-

-> It moves an element along x and y axis from its current position

Example :- transform : translate (500px, 150px);

transform : translate (500px);

-> To move an element only w.r.t x or y axis we have properties like

translate x :- transform : translate x (-300px);

translate y :- transform : translate y (200px);

3) Scale :-

—> It is used to change the dimensions of an element i.e (height and width) by scaling along x and y axis

Example : transform : scale (2, 0.2);

transform : scale (2);

-> To make element to change its dimensions only wrt x or y axis we have properties

x-axis :- translate : scalex(2);

y-axis :- translate : scaley (4);

4) skew :-

-> It skews an element i.e either tilt or slant by the specified angle, along x and y axis

Example :- transform : skew (30deg, 45deg);

transform: skew (30deg);

-> To tilt the element only with respect to x or y axis we have the properties like

X - axis : transform : skew x (-45deg);

y - axis : transform : skew y (45deg);

NOTE :- To make the element to move, change its dimension and to rotate we can make use of the following syntaxes

transform : translate (500px, 200px) scale (0.4)

rotate (300deg) skew (40deg);

(transx)

Animation :-

-> css provides a set of animation properties that allows to create smooth and visually appealing animation on our webpage

-> some of the animation properties are :-

1) animation-name :-

-> specifies the name of the @keyframes rule that defines the animation behavior

-> Establishes the link between animation and keyframes

syntax :- animation-name: identifier;

The identifier must watch the name used in @keyframes

Example :- animation-name: slider;

```
@keyframes slider {
    /* animation steps here */
}
```

2) animation-duration :

-> specifies how long one cycle of the animation takes

-> values can be in seconds (s) or millisecond (ms)

Example: animation-duration : 5s;

3) animation-delay:

-> sets the delay time before the animation starts

-> values can be in seconds (s) or milliseconds (ms)

Example: animation-delay: 2s;

4) animation-timing-function:-

-> controls the pacing/speed curve of animation.

syntax: animation-timing-function: timing-functn;

common values:- ease, ease-in, ease-out, ease-in-out, linear.

Example:- animation-timing-function: ease;

5) animation-direction:-

-> specifies the direction of the animation playback.

common values:- normal, reverse, alternate, alternate-reverse.

Example: animation-direction: alternate;

6) animation-iteration-count:-

-> Defines how many times the animation should repeat.

-> can be a number or infinite

syntax: animation-iteration-count: count;

7) animation :-

→ It is a shorthand property that combines multiple animation properties into a single line like animation-name, animation-duration, animation-delay, animation-timing-function, animation-direction, animation-iteration-count.

Example :- animation : slider 1s ease 0s alternate infinite ;

@keyframes :-

→ It is used to define the animation by specifying a sequence of styles at different points during an animation

→ It must be paired with animation-name to create smooth animations

syntax :- @keyframes identifier {
from {
} to
{ }
}

(or) @keyframes identifier {
0% {
} 50% {
→
} 100% {
} }

Media Queries :-

→ It allows to override and create new styles when the size of screen changes

→ It is mainly used to achieve responsive webdesign

→ To make use of media queries in css we use the @media rule

syntax:- @media (min-width: value;) and (max-width: value;)

Example:- @media (min-width: 400px) and (max-width: 500px)

```
div {
    height: 300px;
    width: 300px;
    border-radius: 50%;
    background-color: maroon;
}
```

NOTE:-

1) min-width:- It is for the styles to add when the width of the screen is equal to value or above it

2) max-width:- It is for the styles to add when the width of the screen is equal to the value or below it.