

7/1/23

CSS Basics - 1

* What is CSS?

Cascading Style Sheets [CSS] → appearance
→ styling
→ formatting
of HTML Doc.

Selectors in CSS :-

1. Simple Selector
→ element Selector.
→ Class Selector.
→ ID Selector.
2. Pseudo-class Selector
3. Multiple Selector

Definition :- It is a way that select an element(s)

ex:- table {
border: 1px solid black;
}
 ↓ ↓
property value

Element Selector / Type Selector / TAG Selector

→ CSS can select HTML elements by using an element's tag name.

```
<style>
  p {
    color: red;
  }
```

Tag/Element

```
</style>
```

Class Selector

To select an HTML element by its class using CSS, a period (.) must be prepended to the class's name.

```
ex:- .class {
      CSS declaration;
}
```

```
ex:- .green {
      color: green;
}
```

```
.blue {
  color: blue;
}
```

ID Selector

The ID selector uses the id attribute of an HTML element to select a specific element.

The ID of an element is unique within a page. Used only one time.

To select an element with a specific id, write a (#) character, followed by the id of the element.

Example:-

```
<p id="intro"> Hello Everyone </p>
```

```
<style>
  #intro {
    CSS property declaration;
  }
```

* Pseudo-Classes Selector

→ A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected elements.

For example:- :hover can be used to change a button's color when the user's pointer hovers over it.

- * Style visited and unvisited links differently.
- * Style an element when it gets focus.

Syntax:-

Selector : pseudo-class {
 property : value;
}

ex:-

```
button : hover {  
    color : yellow;  
}
```

<button> Submit </button>

ex:-

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

Note:- We don't need to remember all pseudo class selectors. We can search it on MDN web docs.

* Multiple Selector / Grouping Selector

We can select multiple elements in the same CSS rule by separating them with commas.

ex:-

```
table, th, td {  
    border : 1px solid black;  
}
```

↓
Grouping

Home Work :-

- Universal Selector.
- Nested Selector.
- Attribute Selector.

(a) Universal Selector (*)

- * The * Selector selects all elements. ✓
- * It can also select all elements inside another element. ✓

Syntax:-

```
* {  
    CSS declaration;  
}
```

```
div * {
```

```
    background-color : green;  
}
```

It will select all elements inside div elements.

(b) Nested Selector

When we use an element/tag for applying CSS property inside another parent element/tag, called Nested Selector. Separated by " " (space).

eg:-

```
div p {  
    color : green;  
}
```

Applying CSS to p tag which is inside <div> tag.

+ Adjacent Sibling Selector

ex:- `div + p { color: red; }`

div aur p tag ke beech mein
p tag hi property milani

just 3 tag ke beech mein
CLASSMATE Page No.
Date

Note:- Combinators

Example 2:-

• box p {
color: red;
}

<p> tag inside
a class selector

• outbox img {
width: 50%;
height: 50%;
}

Here, Child Selector (>)

ex:- `article > p` (it denotes the something)

(C) Attribute Selectors

The CSS attribute selector matches elements based on the presence or value of a given attribute.

Syntax:-

(i) `[attr]` → attribute name

ex:- `a[title] {
color: purple;
}`

General Sibling Selector

ex:- `div ~ p {
color: red;
}`

div aur p tag ke beech mein
p tag hi property milani

CLASSMATE Page No.
Date

(ii) `[attr = value]` → Represents attribute having exact value.

ex:- `a[href = "https://example.org"]`

(iii) `[attr ~ = "value"]` → used to select elements with an attribute value containing a specified word.

ex:- `[title ~ = flower] {
border: 5px solid yellow;
}`

(iv) `[attr ^ = "value"]` → whose value starts with the specified value.

ex:- `[class ^ = "top"] {
background: yellow;
}`

(v) `[attr $ = "value"]` → whose value ends with a specified value.

(vi) `[attr * = "value"]` → whose attribute value contains a specified value of characters.

ex:- `[class * = "te"] {
background: yellow;
}`

* Styling in HTML

- * Inline ✓ Apply in opening tag
- * Internal ✓ Apply in `<style>...</style>`
- * External ✓ External CSS sheets.

→ `<link rel="stylesheet" href="style.css">`

~~Specificity~~

Specificity (Precedence order of styling)

If there are two or more CSS rules that point to the same element, the selector with the highest value will "win" and its style declaration will be applied to that HTML element.

There are four categories which define the specificity level of a selector.

- | | | |
|-------------------|---|--|
| Higher value
↑ | 1.) Inline styles | <code><h1 style="color: pink;"></code> |
| | 2.) IDs | <code>#navbar</code> |
| | 3.) Classes, pseudo-classes, attribute selectors. | Ex: <code>.test</code> , <code>:hover</code> , <code>[href]</code> |
| | 4.) Elements and pseudo-elements | Ex: <code>h1</code> , <code>:before</code> |
| Low value
↓ | | |

Note:- Inline style gets a specificity value of 1000 and always given the highest priority!

Note 2:- There is one exception to this rule:

If you use the "!important" rule, it will even override inline

* The !important rule in CSS is used to add more importance to a property / value than normal.

* It will override All previous styling rules.

Ex:- `p { color: blue; !important; }` ✓

`.para-1 { color: orange; }`

`#black-para { color: green; }`

→ This will apply. ✓

`<p class="para-1" id="black-para" style="color: aqua">`

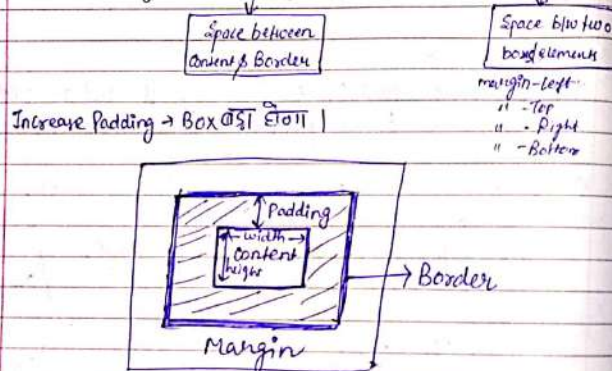
Lorem - - - - </p>

↓ Not this
→ 2

Box Model in CSS

The box model is the basic building block of CSS.

According to the box model concept, every element on a page is a rectangular box and may have width, height, padding, borders and margins.



Increase Padding → Box \uparrow Size \uparrow Margin

→ **Box-sizing** :- allows us to include the padding & border in an element's total width and height.

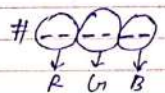
box-sizing: border-box; On an element, padding & border are included in the width and height.
(Content size is fixed, Box size same as width)

Colors in CSS

- Hexadecimal Colors
- RGB Colors
- predefined / Cross-Browser color names
- RGBA colors
- HSL colors
- HSLA colors

* Hexadecimal Colors

Specified with #RRGGBB
↓ ↓ ↓
Red Green Blue



All values must be b/w 00 and FF.

Blue → #0000FF
Green → #00FF00
Red → #FF0000

Black → #000000 → Black
White → #FFFFFF → White

* RGB Colors

An RGB Colors specified with rgb() function.

Syntax:- rgb(red, green, blue).

* Value can be from [0 to 255]

Blue → rgb(0, 0, 255), red → rgb(255, 0, 0) etc.

rgb(0,0,0) → black
rgb(255,255,255) → white

* RGBA

Specified with rgba (red, green, blue, alpha)
alpha :- specifies the opacity for a color.
↳ value b/w 0.0 (fully transparent) to 1.0 (Not transparent at all)

* Predefined Colors

140 color names are predefined in HTML and CSS.

ex:- Blue, Green, Magenta, Aqua etc.

* HSL Colors

Specified with hsl (hue, saturation, lightness).

Hue is a degree from 0-360°. (0 → red, 120 → green, 240 → blue)

Saturation is a % value from 0-100%. (0% → gray shade, 100% → full color)

Lightness is also % from 0-100%. (0% → black, 50% → neither light nor dark, 100% → white)

* HSLA

hsla (hue, saturation, lightness, alpha)
opacity / transparent

Font in CSS

- font-family (styling of font) different names
- font-weight (boldness of font)
- font-style (italic, oblique, normal face)

→ Emphasis & Importance

↓
emphasis marks to text

↓
to give importance to a property.

ex:- (Happy)

Units in CSS

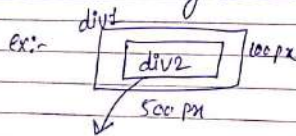
1. Absolute Unit
2. Percentage Unit
3. Relative Unit
 - (i) Relative to font-size
 - (ii) Relative to Document

* Absolute Units (fixed Units)

→ mm → cm → in → px ($\frac{1}{96}$ of inch)
 $1\text{cm} = 37.3\text{px}$ $1\text{in} = 96\text{px}$ $= 2.54\text{cm}$ → Smallest display unit

* Percentage Unit

Shows relevancy with the parent element(s)



div { width: 10px }

→ 10% of Parent Width
 10% of 500px.

⇒ 50px

(parent)

#div1 {

width: 500px;
 height: 100px;

}

#div2 {

width: 10%;
 height: 10%;

}

* Relative to font-size :-

* em → relative to parent element font size.

div → 18px

Ex:-

div2 → 1em

If No parent, then
 em relative to
 <body> font size
 = 16px (default)

1 em = 1 × 18 px = 18px

2 em = 2 × 18 px = 36px

* rem → Relative to Root Element (<html>)

So, 1 rem = 1 × 16 = 16px. default font size 16px.

* Relative to Viewport :-

* vw = 1% of Viewport's width or $\frac{1}{100}$ of Viewport's width

→ vh = $\frac{1}{100}$ of height of Viewport.

CSS - Basics - 2

CSS Gradients

CSS Gradients let you display smooth transitions between two or more specified colors.

Three types of gradients:-

- * Linear Gradients (goes down/up/left/right/diagonally)
- * Radial Gradients (defined by their centers)
- * Conic Gradients (rotated around a center point)

* Linear Gradient (at least 2 colors with directions).

By default, Direction of Linear ~~gradient~~ gradient.

is:- Top-down



background-image: linear-gradient(Red, Blue);

→ Left to Right: linear-gradient(Red, Blue);

← Right to Left: linear-gradient(to Top Left, Red, Blue);

Note:- If there are two or more gradients, then latest gradient or last gradient value will be applied.

Bottom - Top → linear-gradient(to top, Red, Blue);

for diagonally,



linear-gradient(to bottom right, Red, Blue);



linear-gradient(to bottom left, Red, Blue);



to top left



to top right



Syntax:-

background-image: linear-gradient(direction, color1, color2, ...)

Directions:-

Default, Specific, Using Angles,
Using transparency.

Using angle like → linear-gradient(45 deg, Red, Blue);

Any deg
angle

using transparency :-

background-image: linear-gradient(to bottom,
 rgba(255,0,0,0), rgba(0,0,255,1));



Red color
Zero Transparency

Blue color
Full Transparency

#* Radial Gradients :-

A radial gradient is defined by its center.

Syntax:-

background-image: radial-gradient(shape size at position,
 start color, -----,
 last color);

Shape :- * ellipse (default)
* circle

ex:- background-image: radial-gradient(circle, red, blue, orange).

Not:- we can use % for colors

radial-gradient(yellow 5%, orange 10%, red 50%).

#* Conic Gradient :-

A conic gradient is a gradient with color transitions rotated around a center point.

Syntax:-

background-image: conic-gradient(from angle at position,
 color deg, color deg, -----);

ex:- background-image: conic-gradient(red, yellow, green);

with degree
background-image: conic-gradient(red 45deg, yellow 90deg,
 green 210deg);

Pie-charts of colors :-

background-image: conic-gradient(red, yellow, green, blue,
 magenta);


border-radius: 50%;

from angle at position :-

background-image: conic-gradient(red 0deg, red 90deg,
 yellow 90deg, yellow 180deg,
 green 180deg, green 270deg,
 blue 270deg);

Conic Gradient With Specified from Angle :-

From Angle specifies an angle that the center conic gradient is rotated by.



background-image: conic-gradient (from 90 deg, red, yellow, green);

With Specified Center position :-



background-image: conic-gradient(at 60% 45%,
red, yellow, green).

Repeating Conic Gradient :-

Repeating - conic - gradient () function is used to repeat conic gradients.

background-image: repeating-conic-gradient(2rad 10%, yellow 20%);
border-radius: 50%;

border-radius : 50% ;



All CSS Gradient functions

Conic-gradient ()
Linear-gradient ()
Radial-gradient ()

Repeating - conic - gradient ()
Repeating - linear - gradient ()
Repeating - radial - gradient ()

CSS Shadow Effects :-

→ Text-shadow:

→ Box-Shadow:

Text-Shadow:-

Apply Shadow to text.

→ Horizontal / Vertical shadows.

→ Color can be added.

→ Blue can be added

- Multiple Shadows on 1 text can be added.

- How can we add Border using shadows?

Ex:- (i) $h \perp \{$

text-shadow: 3px 3px 3px red;

Horizontal
Blur effect
Vertical
Color

② Multiple Shadow

```
h1 {
  text-shadow: 3px 3px 3px red, 5px 5px 5px green;
}
```

Horizontal Blue
Vertical

Box-Shadow :- Default color of shadow is text color.
Apply shadow to a box.

example - ①

```
box {
  background-color: aqua;
  width: 200px;
  height: 200px;
  box-shadow: 10px 10px 15px green;
}
```

Horizontal shadow
Vertical shadow Blue

② Multiple Shadow :-

```
box-shadow: 10px 10px 15px green, 15px 15px 15px red;
```

NOTE :- By using -ve value, for left side shadow.

```
-10px -10px 15px green;
```

Ques How can we add border using Box-Shadow?

Ans By using :-

```
box-shadow: 0px 0px 0px 3px red;
```

Vertical shadow
Horizontal shadow
Blue
Spread for spreading color area

Inset for inside shadow.

```
box-shadow: inset 2px 2px 2px 2px blue;
```

CSS Dimension Properties :-

- * Width
- * Height
- * Min-height
- * Max-height
- * Min-width
- * Max-width

* To Control Overflow

Ex-①

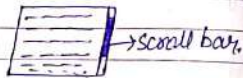
```

    .box {
        max-height: 100px; → upto 100px limit
        border: 1px solid black;
        overflow: scroll;
    }
  
```

यहाँ तक height की limit है और Content box में रहेगा, नहीं तो overflow हो जायेगा।



इसलिए हम overflow को scroll को use करेंगे।

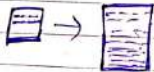


Ex-②

```

    .box {
        min-height: 100px;
        border: 1px solid black;
    }
  
```

कम से कम height 100px होगी, ज्यादा से ज्यादा कितनी भी Content के according हो जायेगी।



CSS Overflow Property :-

- ① overflow: visible; (default) Content visible in overflow.
- ② overflow: hidden; Content visible till box height. i.e. overflow content will hide.
- ③ overflow: scroll; Scroll is added.
- ④ overflow: auto; automatically set visible or scroll.

CSS position Property :-



- static (by default)
- Relative
- Fixed
- Absolute
- Sticky

- ① static position by default as we add Content in html doc, that will show. No change in Order.
ex:- position: static;

(ii) relative (top, left, right, bottom).

Element is positioned relative to its normal position.


ex:-

position: relative;	Before	
left: 20px;	After	
top: 10px;		

(iii) fixed

The element is positioned fixed relative to the viewport or browser window, and element will be removed, occupy by other element. It means, element stays in the same place even if the page is scrolled.

Top, Right, Bottom & Left properties are used to position the element.

ex:- position: fixed; 

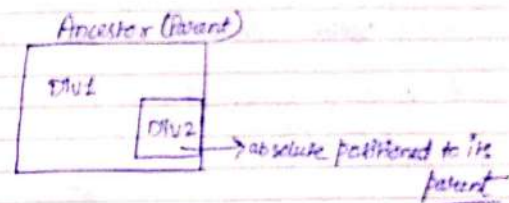
(iv) absolute

The element is positioned absolute relative to the nearest positioned ancestor. (instead of positioned relative to the viewport, like fixed.)

If an absolute positioned element has no positional ancestor, it uses the document body and moves along with page scrolling.

Notes:- Absolute positioned elements are removed from the normal flow, and can overlap elements.

Example:-



Ancestor

```
.Div1 {  
  position: relative;  
  width: 100px;  
  height: 200px;  
  border: 3px solid blue;  
}
```

```
.Div2 {  
  position: absolute;  
  top: 80px;  
  right: 0px;  
  width: 50px;  
  height: 50px;  
  border: 3px solid blue;  
}
```

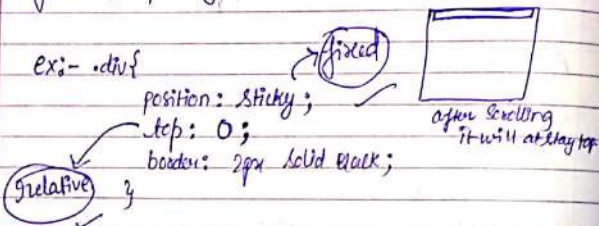

Sticky

An element with position: sticky; is positioned based on the user's scroll position.

It is positioned relative until a given offset position is met in the viewport then it "sticks" in place (like position: fixed).

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

Ex:-



CSS - 2D Transforms :-

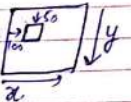
Transforms allow you to move, rotate, scale and skew elements.

following methods are :-

- translate() → for Movement :- transform: translate(100px, 50px);
- rotate() → for Rotation :- transform: rotate(45deg)
- scaleX() → for Scaling horizontally - axis
- scaleY() → for Scaling vertically - axis
- scale() → for Scaling both X and Y axis
- skewX()
- skewY()
- skew()
- matrix()

(i) translate()
 Movement in X or Y direction

transform: translate(100px, 50px);



(ii) rotate()
 Rotation in +ve (clockwise) or -ve anti-clockwise

transform: rotate(45deg) or (-45deg)



(iii) Scale() → for Zoom content / Stretch

transform: Scale(2, 3)

horizontal zoom

vertical zoom

decimal value is accepted.

(iv) ScaleX() → for scaling X-axis.

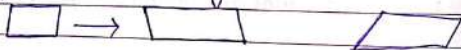
transform: ScaleX(2);

(v) ScaleY() → for scaling Y-axis

transform: ScaleY(4);

(vi) Skew() → for Tilt Contents

transform: Skew(20deg); Skew(-20deg)



Similarly for SkewX(), SkewY() ✓

we can both like

transform: Skew(20deg, 10deg);

to X-axis

to Y-axis



(vii) matrix()

takes 6 parameters which allow you to rotate, scale, move (translate) and skew elements

Syntax:-

$\text{matrix}(\overset{a}{\text{ScaleX()}}, \overset{b}{\text{SkewY()}}, \overset{c}{\text{SkewX()}}, \overset{d}{\text{ScaleY()}}, \text{translateX()}, \text{translateY()})$

$\text{matrix}(a, b, c, d, e, f)$

$a, d \rightarrow \text{Scale}(a, d)$

$b, c \rightarrow \text{Skew}(b, c)$

$e, f \rightarrow \text{Translate}(e, f)$

CSS - 3D transforms :-

Working on Z-axis -

ex: • translated {

transform: perspective(10px) translateZ(-10px);

• scaled {

transform: perspective(15px) scaleZ(2);

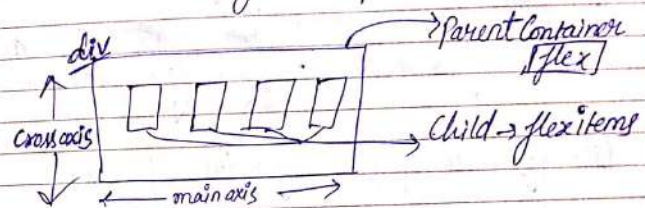


CSS Basics - 3

CSS Flex Box

Flexbox is a great way to get more flexibility in your layout and to simplify responsive layout design.

Layout Model → space distribution
→ alignment capabilities.



To make container flexible

use first, `display: flex;`

* Flex-Container Properties :-

`display: flex; /`

- Flex-direction
- Flex-wrap
- Flex-flow
- Justify-content
- align-items
- align-content

(i) flex-direction property :-

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

(ii) flex-wrap property :- (width preserve)

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

(iii) flex-flow property :-

Shorthand Notation of all both above.

flex-flow : row wrap;
 " : column wrap;
 " : row nowrap;
 " : column-reverse wrap;
 etc., ; ; ;

Imp Note :- `gap: 10px;` for gapping b/w div box
`row-gap` `column-gap`

(iv) justify-content property :- (Work on Main axis / Horizontal)

justify-content : flex-start;
 justify-content : flex-end;
 " : center;
 " : space-around;
 " : space-between;
 " : space-around;

(v) align-items property :- (Work on Cross axis / Vertical align)

align-items : flex-start;
 align-items : flex-end;
 align-items : center;
 align-items : stretch;
 align-items : baseline;

(vi) align-content property :-

align-content : flex-start;
 align-content : flex-end;
 align-content : center;
 align-content : space-between;
 align-content : space-around;
 align-content : space-between;

* Flex-items properties :-

- Order Order: 1; Set Order
- flex-grow (default 0) flex-grow: 1; Grow
- flex-shrink (default 1) flex-shrink: 4; Shrink Speed
- flex-basis flex-basis: 100px; Width up to (Content)
- flex
- align-self

* flex property combines above four properties :-

flex: 3 1 2 120px;
 ↓ ↓ ↓
 order grow shrink flex basis (width upto content).

* align-self

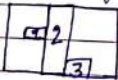
align only 1 item, vertically,

ex:- #box2 {

align-self: stretch;

#box3 {

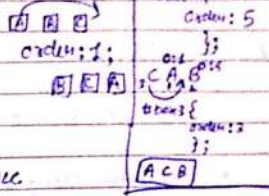
align-self: flex-end;



Box3

→ Order To set the order of items.

Same order first set in order.



→ flex-grow (default value 0) occupy available space. To grow box in responsive width

ex:- flex-grow: 1;



→ flex-shrink (default value 1)

To shrink box in responsive.

ex:- flex-shrink: 4;

→ flex-basis:-

flexible width

To responsive width upto content

ex:- flex-basis: 100px / 100%;

But width: 100px; content hidden

Important

Ques:- Difference b/w `` and background-image property

Ans:-

:-

1. When it is not just a design element but also a part of the content, such as diagram, logo or a person.
2. When you expect that people will print your page and want the image to be included by default.
3. When you want to be indexed by the search engine. Google doesn't index background images automatically. But `` with alt & title attributes will help to be recognized by screen readers.
4. When you want to improve animation performance over a background.

* CSS background-image: url(" "); property

1. If the image is only used for design and not part of the content.
2. If you expect that people can print your page, and don't want the image to be included by default.
3. If you want to improve the download time.
4. If there is a need to repeat images.
5. If you want to make stretch the background image to fill its window.

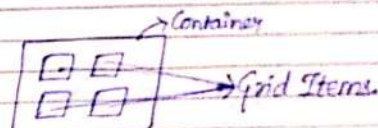
Normal \rightarrow display: flex \rightarrow display: grid

CSS - Basics - 4

Note:- Flex Box is related to positioning of content.
Grid is related to layout creation.

CSS GRID

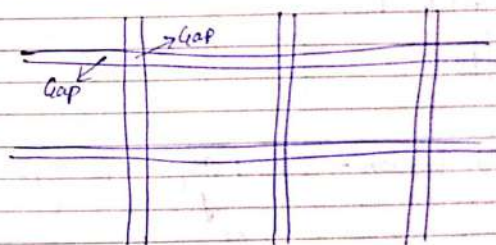
display: grid ;



* A Grid layout is a two-dimensional layout system for the web.

- * It lets you lay content out in rows and columns.
- * It has many features that make building complex layout straightforward.

* Gaps between rows and columns are called Gutter.



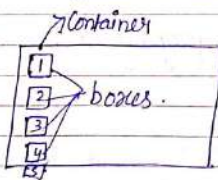
Note: Flexbox is 1-D but Grid is 2-D.

↓
We work in only 1 direction
Main axis or Cross axis

↓
We work in 2 directions,
Rows and Columns.

* To initialize Grid.

display: grid;



* To add Columns & Rows in Grid.

Columns → grid-template-columns: 200px 200px 200px;
↓ ↓ ↓
1st 2nd 3rd
Column Col. Col.



Rows → grid-template-rows: 90px 90px 90px 90px;
↓ ↓ ↓ ↓
R1 R2 R3 R4

	200px C1	200px C2	200px C3
90px R1	1	2	3
90px R2	4	5	
90px R3			
90px R4			

→ Grid of 4 Rows and 3 Columns

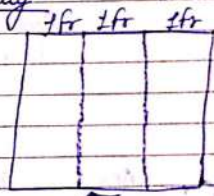
* For Creating Equals size of Columns and Rows.
we use "fr" unit that refers to the
fraction of grid sizes.

* grid-template-columns: 1fr 1fr 1fr;

divide columns equally

Also called
Flexible
GRID

OR Alternate Method
Using Repeat() :-



* grid-template-columns: repeat(3, 1fr);

We can add flexible width to any column.

example:-

grid-template-columns: repeat(3, 1fr);

grid-template-columns: 100px repeat(2, 1fr) 200px;

1st Col. 2nd & 3rd Col. 4th Column

Similarly for all Rows procedures are same.

ex:-

grid-template-rows: repeat(4, 1fr);

Equally sizes of Rows \rightarrow 1fr 1fr 1fr 1fr

Note:- we can do any fraction (not only limited to 1fr.)

ex:-

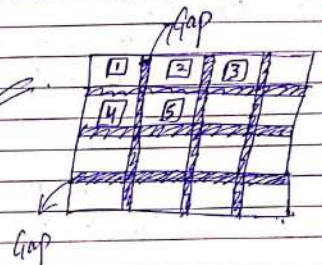
grid-template-columns: repeat(3, 2fr);

grid-template-rows: repeat(4, 3fr);

Gap

We can add gaps b/w rows and columns.

gap: 15px;



1	2	3
4	5	6
7	8	9
10		

Grid of 4 Rows & 3 Cols.

```

• container {
  background-color: brown;
  height: 500px;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(4, 1fr);
  gap: 15px;
}

```

```

• box {
  background-color: aqua;
  width: 200px;
  height: 80px;
}

```

(Remove this width for more)

Note:- Grid overflow में Boxes Add करने पर Automatic height adjust कर देता। Responsive Height

CLASSTEAM Page No. _____
Date: / /

To increase Columns and Rows Flexible Grid

→ #box1 {
grid-column-start: 1;
grid-column-end: 3; (2 कॉ) एक कॉ
}

box 1 और box 2 की width के साथ increase कर आगे।

R1	1	2
R2	3	4
R3	6	7
R4	9	10

(Set width की बात है)

→ #box3 {
grid-row-start: 2;
grid-row-end: 4; (2 कॉ कॉ) (Row 3 कॉ)
}

1	2
3	4
8	9

(Set height की बात है)

Create:-

100px	1fr	1fr
Header	Box1	
Box2	Box3	
Box4		

CLASSTEAM Page No. _____
Date: / /

#box1, #box4 {
grid-column-start: 1;
grid-column-end: 4; → (2 कॉ कॉ)
}

#box3 {
grid-column-start: 2;
grid-column-end: 4; (एक कॉ कॉ)
}

• Container {
background-color: brown;
height: 500px;
border: 3px solid black;
display: grid;

grid-template-columns: 100px 1fr 1fr;
grid-template-rows: 30px 1fr 30px;

gap: 5px;

• box {
background-color: aqua;
border: 3px solid black;
}

Important

for Rows & Columns Creation

grid-template-rows: R1 R2 --;
grid-template-columns: C1 C2 --;

for flexible Grid.

grid-column [grid-column-start: -;
grid-column-end: -;
grid-row [grid-row-start: -;
grid-row-end: -;

Short hand Notation

→ grid-row: 1/3; ^(200px)

1	2	3
---	---	---

→ grid-row: 1/span 3;

1	2	3
---	---	---

↓
Span create in 1st row
Row No. 1 and span 3

→ grid-row: 2/-1; ^(200px)

2	3
1	

→ grid-row: auto/auto; (default span of 1 ex: 1/1;).
Similarly for Column.

→ grid-column: 1/3; ^(200px)

1	2
3	

→ grid-column: 2/-1; ^(200px)

1	
2	3

→ grid-column: 1/span 2; ^(100px)

1	2
3	

→ grid-column: auto/auto; (default span of 1).

Grid area (Short hand Notation)

grid-area: a / b / c / d ;

