CSS Validation

# CSS Syntax, How to add CSS, CSS Selector

## CSS

→ cascading style sheet

*** CSS Syntax :-**

Selector — select html elements to style → p {

Property Name

```
p {
    colon : blue ;
}
```

→ value

```
p{
    colon : blue
    font-size : 12 px ;
}
```

*** How to add css :-**

Three ways to Insert CSS :-
- External css
- Internal css
- Inline Css

• **External css** :- Make new file Style1.css
↓
any Name

then link in head tag.

```
<link rel="stylesheet" href="style1.css">
```

• **Internal css** :-

```
<head>
    <style>
        p{
            colon: □ red;
        }
    </style>
</head>
```

• **Inline css** :- (style inside html elements)

```
<body>
    <p style="colon: □ red"></p>
</body>
```

→ If three are written together then inline will get more priority. After that jisko last meh likha uska display hoga.

*** CSS selector :-**
- element ~~selector~~
- Id
- class

→ ~~Id can be~~ Id name can be used only once in web page / Id name is specific

∴ Id name should be unique / or need to make different id name every ~~this~~ time you used.

class name can be used multiple type.
∴ It is used for group styling.

Eg:-

Html:-

```
<body>
    <h1 id="H1"> Heading </h1>
    <P> Paragraph 1 </P>
    <P class="green"> Paragraph 2 </P>
    <P> Paragraph 3 </P>
    <P class="green"> Paragraph 4 </P>
    <P> Paragraph 5 </P>
</body>
```

CSS :-

```
P {
    colon : red;
}
# H1 {
    colon : blue;
}
.green {
    colon : green;
}
```

You can give both id & class also in elements.

Eg :-

```
<h1 id="H1" class="green"> Heading </h1>
```

→ Id > class > Element (Id will get more priority)

Eg:-

Html :-

```
<body>
    <P class="green" id="P2"> Paragraph 2
    </P>
    <P class="green" id="P3"> Paragraph 3
    </P>
</body>
```

CSS :-

```
P {
    colon : red;
}
.green {
    colon : green;
}
#P2 {
    colon : yellow;
}
#P3 {
    colon : pink;
}
```

# Comment in CSS, Colors in CSS

result will be the color of id will display.
If we remove id then class color will display.
Then element.

→ we write class by . ---- in style.
↓ name

→ " " id by # ----- in style.
↓ name

* comment in CSS :-

/* ......... */
↳ write

* colors in CSS :-

Predefined color names
RGB (Red, green, blue)
RGBA
HEX
HSL
HSLA

→ RGB :- • rgb (red, green, blue)
- color value b/w 0 & 255
- Black : rgb(0, 0, 0)
- white : rgb (255, 255, 255)
- red : rgb (255, 0, 0)
- Green : rgb (0, 255, 0)
- Blue : rgb (0, 0, 255)

→ RGBA :- • rgba (red, green, blue, alpha)
- alpha parameter is a no. b/w 0.0 (fully transparent) & 1.0 (not transparent at all)

Eg:- rgba (255, 90, 71, 0.5)

→ HEX :- • #rrggbb
- rr(red), gg(green) & bb (blue)
- hexadecimal values b/w 00 & ff (Same as decimal 0-255)
- Red : #ff0000
- Black : #000000
- white : #ffffff

→ HSL :- • hsl (hue, saturation, lightness)
- Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green & 240 is blue.
- saturation is a Percentage value, 0% means a shade of gray & 100% is the full color.
- Lightness is also a Percentage, 0% is black, 50% is ~~neight~~ neither light nor dark, 100% is white.

Eg:- RED : hsl (0, 100%, 50%

# Background

~~* HSL~~

→ HSLA :- hsla (hue, saturation, lightness, alpha)

Eg:- hsla (0, 100%, 50%, 0.5)

→ Search online for the codes of different color.

## * Backgrounds :-

background-color : ... ;

colour name

→ We can change body colour also

```
body {
    background-color : ... ;
}
```

→ We can attach image as background also.

background-image : url(' ');

location of image

→ background-repeat : no-repeat;

to not repeat an image

→ background-repeat : repeat-x ;

x-axis meh repeat hogeh bar jayega

→ By default it is background-repeat : repeat;

→ background-repeat : repeat-y :

y axis meh repeat hoga

→ background-position : ....... ;

To set the position of background.

X-axis — { right top ; right bottom ; right center ; center center ; } → y-axis

.... Px ... Px → in Pixel

.... % .... %

and more....

→ background-size : ______ ;

Size of background

height & width

→ background-attachment : ______ ;

by default it is scroll.

# Border

→ background-attachment: fixed;
↓
background nhi hilega
scroll karne seh

→ background-size: cover;
↓
background ban dega bina
repeat kareh / but image
~~kat~~ khat tha hain.

→ background-size: contain;
↓
resize kartha hain but
image ka quality karab nhi kartha.
Pura background image dikayega
kat tha nhi. but image
chotha hoga,
ratio same hona chahiye tab hi
page bar jayega.

→ Short~~cut~~hand for background.

background: background-color background-image background-repeat background-position;

Eg:-

background: #ffffff url('') no-repeat fixed 5% 10%;

→ we can use two or more background images?

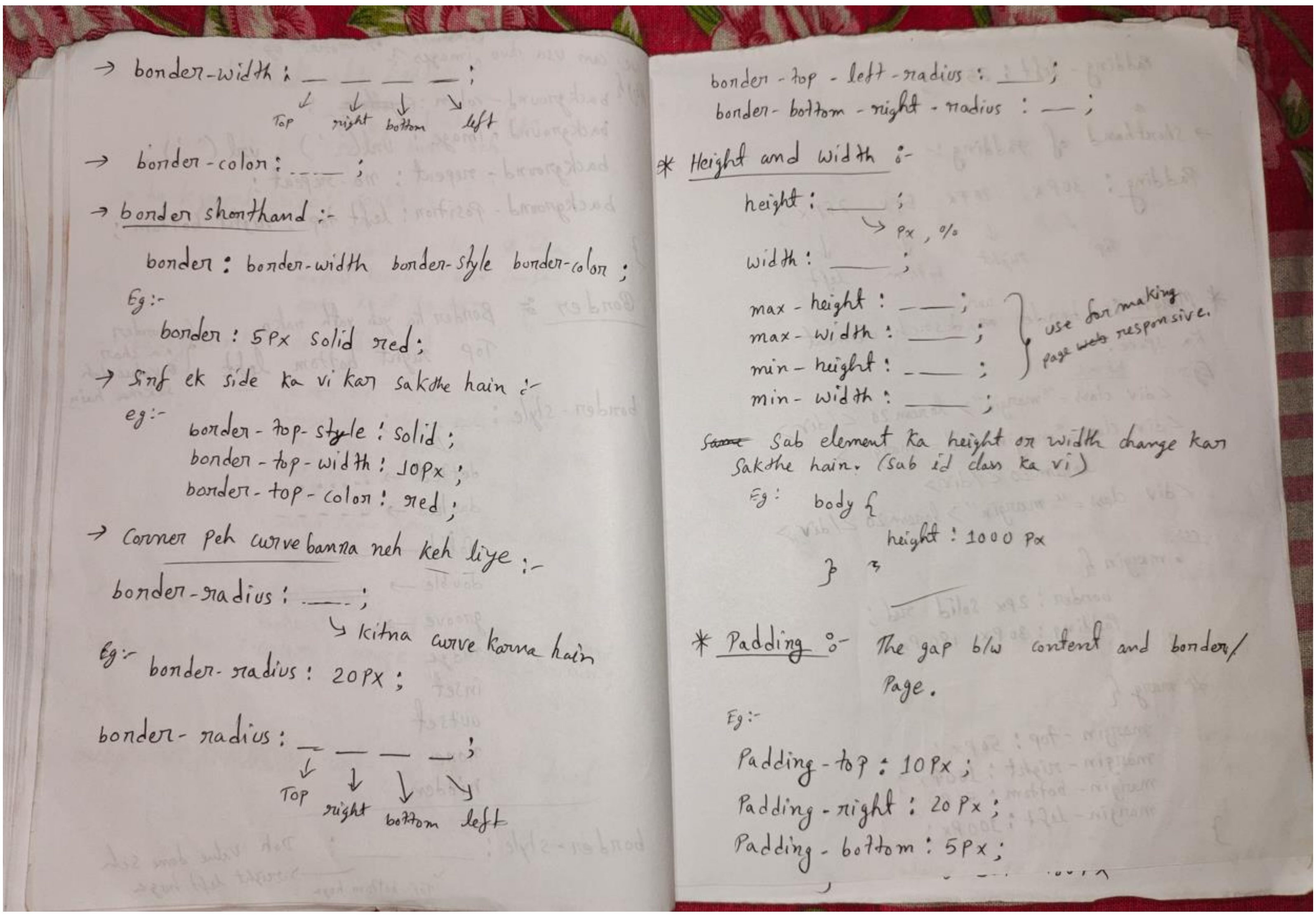
Eg:- #bg{
background-color: ~~yellow~~ ___;
background-image: url(''), url('');
background-repeat: no-repeat;
background-position: left top, right bottom;
}

* Border: Border ka yeh yath rako
Top right bottom left
{ border ka char value deh saktha hain

border-style: ______;
↓
dotted → . . . . . .
dashed → - - - - - -
solid → ______
double → ══════
groove →
ridge
inset
outset
none
hidden

border-style: ___ ___; Doh value dene seh
→ Top bottom hoga
→ right left hoga

## Height and Width, Padding



→ border-width : ___ ___ ___ ___ ;
↓ Top ↓ right ↓ bottom ↘ left

→ border-color : ____ ;

→ border shorthand :-

border : border-width border-style border-color ;

Eg :-

border : 5px solid red;

→ Sirf ek side ka vi kar sakthe hain :-

eg :-

border - top - style : solid ;
border - top - width : 10px ;
border - top - color : red ;

→ Corner peh curve banana neh keh liye :-

border-radius : ____ ;
↳ kitna curve karna hain

Eg :- border - radius : 20px ;

border - radius : ___ ___ ___ ___ ;
↓ Top ↓ right ↓ bottom ↘ left

border - top - left - radius : ____ ;
border - bottom - right - radius : ____ ;

* Height and width :-

height : ____ ;
↳ px , %

width : ____ ;

max - height : ____ ;
max - width : ____ ;
min - height : ____ ;
min - width : ____ ;
} use for making page web responsive.

~~Same~~ Sab element ka height or width change kar sakthe hain. (Sab id class ka vi)

Eg : body {
height : 1000 px
}

* Padding :- The gap b/w content and border/ Page.

Eg :-

Padding - top : 10px ;
Padding - right : 20 px ;
Padding - bottom : 5px ;

# Margin

Padding - left : 25Px ;

→ Shorthand of Padding :-

Padding : 30Px 20Px 5Px 25Px ;

Top right bottom left

* Margin :- border aur dusreh element ka space.

Eg:- html

<div class = "margin"> lorem20 </div>

<div class = "margin" id = "marg">
lorem20 </div>

<div class = "margin"> lorem20 </div>

css

```
.margin {
    border : 2Px solid red ;
    Padding : 30Px 180Px ;
}

# marg {
    margin - top : 50Px ;
    margin - right : 100Px ;
    margin - bottom : 50Px ;
    margin - left : 100Px ;
}
```

→ Shorthand of margin :-

margin : __ __ __ __ ;

Top right bottom left

margin : __ __ ;

Top & bottom right & left

→ we can give margin negative value also.

Eg:- margin : -50Px 20Px ;

→ we can't give negative value in Padding.

→ If ek element ka margin-Top or dusre element ka margin-bottom dono diya hua hain toh Jiska Jada value hoga usko lega naki add hoga.

Eg:- html

<div class = "margin" id = "marg"> lorem20 </div>

<div class = "margin" id = "marg1"> lorem20 </div>

css

```
.margin {
    border : 2Px solid red;
    Padding : 30Px 180Px
}
```

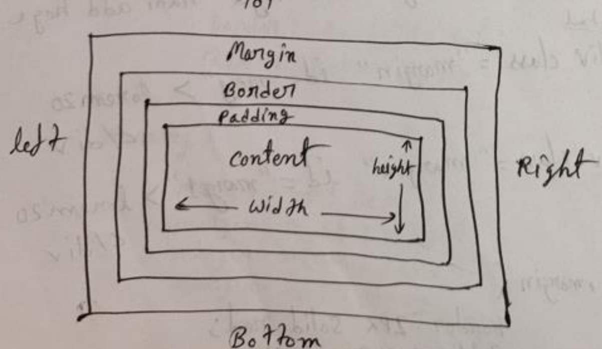
## Box Model, Text

```
# marg {      ↗ Top    ↗ right    ↗ bottom    ↗ left
   margin: 50px 100px 70px 100px;
}

# marg1 {
   margin: 40px 100px 50px 100px;
}              ↓      ↓       ↓       ↓
              Top   right   bottom   left
```

marg ka bottom ~~or~~ aur marg1 ka ~~b~~ Top add nahi hoga.
marg ka value lega (70px) keuki marg1 ka (40px) hain.

→ For left and right margin value will be added. → margin: auto; → Center peh lane ke liye

### * Box Model :-




### * Text :-

color: __ ; → Text ko colour karne ke liye

text-align: ______ ;
- left → by default
- right → text from right to left
- center → in center
- justify → ~~a~~ to occupy all space

direction: rtl; → Text ko right to left likneh ke liye

→ html

```html
<div id="text"> Lorem10 <img src=" " alt=" "
width=" " height=" "> lorem 30 </div>
```

If image is given in b/w text then use vertical-align to align image :-

css:

```css
#text {
    border: 2px solid red;
    width: 500px;
    color: blue;
    text-align: justify;
}
img {
    border: 1px solid red;
    vertical-align: top;
}
```

↳ here top / bottom / middle / ⋮

~~Then~~ Then ~~a~~ text will show from top
if ~~a~~ vertical-align: top; is given.

→ text-decoration: ______ ;

underline → underline all texts
overline → upper line
line-through → text ko Katna

→ text-transform: ______ ;

uppercase → sab capital letter
lowercase → sab small letter
capitalize → first ka letter capital in each words

and many-more

→ line-height: __ Px ;

lines ka gap kitna hoga

→ text-Indent: __ Px ;

First letter ko kitna width seh start karna hain.

→ letter-spacing: __ Px ;

~~Spacin~~ gap b/w letters

→ word-spacing: __ Px ;

har ek word ke beatch ka gap

white-space: pre; → Jessa html meh likha hain wesa dikayega.

white-space: nowrap; → ek hi line meh rakhega no extra line.
(ek hi line meh sab words)

→ agar container seh bara word hain toh usko adha ~~likw~~ adha likwane ke liye.

word-wrap: break-word;

→ text pe shadow bananeh ke liye :-

Eg :- html
<div id="shadow"> lorem 5 </div>

css
```
# shadow {
    font-size : 70 px;
    text-shadow: 5px 10px 5px red;
}
```

horizontal kitna hain — (-) deke left leju saktha hain
vertical upper niche karneh ke liye — (-) deke uppar deja saktha hain
kitna blur
colour

# Links, Fonts

→ writing-mode: vertical-lr; → vertical likne/dikane ke liye
or
vertical-rl;

* **Links** :- link ko design karneh ke liye.

html :-

<a href=" " target="_blank"></a>

css :-

a{
text-decoration: none;
}
↳ under line hata neh ke liye

a:link {
colour: ___ ; → colour of link
}

a: visited {
colour: ___ ; → Jab already us web pe visit kar chuka
}

a: hover {
colour: ___ ; → hover karne seh (curser us pe lane seh)
}

a: active {
colour: ___ ; → click karne seh
}

→ iss tags ko serial istamal karo.
:link
:visited
:hover
:active

* **Fonts** :-

font-size: ___ ;
font-weight: ___ ;
font-style: ___ ;
font-varient: ___ ;
↳ small-caps → capital letter banayega but
Eg:- LOREM

→ ~~Download~~ ~~go to~~ Search google font and copy link from embed option.

font-family: ___ ;

## CSS Units

→ download free web fonts :

~~and write~~

~~css~~ To attach write in css :

@font-face {

font-family : abc; → any name

Src: url('') → location

and

# Para {

font-family : abc; ← Same name as above to apply

}

→ font shorthand :-

font : font-style font-varient font-weight font-size font-family ;

Eg:-

font : italic small-caps bold 12px georgia, serif;

* CSS Units :-

we use length in width, height, margin, font-size, Padding, etc,

→ Two types of units in css :-

absolute → independent

Relative → dependent

→ Absolute units in css :-
- cm
- mm
- In
- Px (1 inch = 96 px)
- Pt (1 inch = 72 pt)
- Pc (1 pc = 12 pt)

→ Relative units in css :-

Em - em is relative to the size of its direct Parent.

Rem - rem is only relative to root (html tag) size.

Vh - 1% of view Port height

Vw - 1% of view Port width

% - relative to Parent.

→ em ka ~~eg~~ example :-

Html :-

<div id = "Parent">

<p> hi i am parent </p>

<div id = "child">

<p> hi i am child </p>

</div>

# CSS Cursor, CSS Important

</div>

CSS :-

```
# Parent {
    border: 2px solid red;
    font-size: 30px;
# child {
    border: 2px solid blue;
    font-size: 2em;
}
```

2 em dene seh Parents ka double hoga. (child ka multiple hota hain & jitna em dya.)

em ka child ka multiple hota hain with respect to parent.

→ rem ka example :-

HTML same as em ka example.

CSS :-

```
html {
    font-size: 30px;
}
# Parent {
    border: 2px solid red;
    font-size: 2rem;
}
# child {
    border: 2px solid blue;
    font-size: 1rem;
}
```

Parent aur child ka font size html ke respect meh barega.

useful bcz by changing html the Parent & child changes automatically.

→ Eg :-

~~weight :~~

height : 50 vh; → View Port ka 50%

width : 80 vw; → ViewPort ka 80%

---

* CSS cursor :- cursor : ____ ;   Eg :- cursor: wait;

cursor ush div/element meh jaaneh seh.

* CSS Important :-

Eg :-

```
div {
    border: 1px solid red !important;
}
div {
    border: 1px solid blue;
}
```

ye apply hoga

→ important dene seh usi ko apply karega, order follow nhi karega.

# Box Shadow, CSS Opacity

## * Box Shadow :-

box-shadow : ___ ___ ___ ___ ___ ___ ;

- x-axis
- y-axis
- Kitna blur
- Spread radius (Kitna pelega)
- color
- inset dene seh shadow box ke andar banega.

→ minus videh sakthe hain.

Eg:-

box-shadow : 10px 10px 15px 3px grey inset;

inset → isko dene seh shadow andar banega.

→ multiple shadow vi bana sakthe hain.

Eg:-

box-shadow : 10px 10px 15px 3px red, 10px 10px 15px 3px blue;

∴ Just comma deh dusra likneh seh pehleh

## * CSS opacity :-

kissi vi element ko transparent set kar sakthe hain.

Value 0 seh 1 tak

1 → non-transparent
0 → fully-transparent or invisible.

Eg:-

html :-

```html
<body>
    <div id="box1"></div>
    <div id="box2"></div>
</body>
```

css :-

```css
#box1 {
    background-color: chartreuse;
    width: 100px;
    height: 100px;
    margin-top: 100px;
    margin-left: 400px;
    border: 1px solid red;
}

#box2 {
    background-color: crimson;
    width: 100px;
    height: 100px;
    border: 1px solid greenyellow;
    margin-top: ~~100px~~ -100px;
    margin-left: 400px;
}
#box2:hover {
    opacity: 0;
}
```

# CSS Filter, Image Sprites, CSS gradients

* **CSS Filter :-** image peh add kar sakthe hain

filter: blur (5px);
filter: brightness (50%);
filter: contrast (70%);
filter: drop-shadow (20px 10px 3px red)

(20px → horizontal, 10px → vertical, 3px → blur, red → color)

filter: grayscale (80%); → black & white
filter: hue-rotate (150deg);
filter: invert (30%);
filter: opacity (50%);
filter: saturate (30%);
filter: sepia (30%);

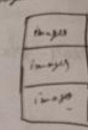
→ filter shorthand : Just add jitna add karna hain

Eg:-
filter: brightness(80%) contrast (50%) blur (3px)
drop-shadow (25px 10px 3px red);

* **Image sprites :-** agar multiple kam size ka image ho toh you can use it. ~~for speed up~~
Don't use for big size images.
First combine multiple small images & make it one.
remember x- & y axis of small images inside it.

Eg:- html
<body>
<div id = "sprite"></div>
</body

Eg: image



css:-
```
#sprite {
    height : 128px;
    background-image: url ('images/sprite.png');
    background-repeat : no-repeat;
    background-position: 0px  -288px;
}
```
(0px → x-axis, -288px → y axis)

→ ye sirf background image meh kam karega.

* **CSS gradients :-** color mix kar sakthe hain.
sirf background-image ke liye.

background-image : ______ ;
↓
linear-gradient ();
radial-gradient ();

linear-gradient (red, blue, green); → can add multiple color.
linear-gradient (to right, red, blue, green); → can set position
- to left
- to top
- to bottom right

CSS overflow, CSS Resizeable, CSS Lists

to bottom left
to top right
to top left

linear-gradient (to top right, red 20%, blue 30%; green);
↓
can add cappacity to occupy

radial-gradient (red, blue, green); → can add multiple color.
radial-gradient (circle, red, blue, green);
radial-gradient (~~circle~~ red 30%, blue 40%);
add cappacity to occupy.

* CSS overflow :- agar content /Text Jada ho aur space nhi hain toh overflow kam atta hain.

Eg:-

```html
<div id= "flow">
    <p> lorem1000 </p>
</div>
```
} html

css :-

```css
#flow {
    height : 500 px ;
    width : 500 px ;
    background - color : aqua ;
    border : 2px solid red ;
    Padding : 0px  10px  10px  10px ;
    margin : 50px 330px ;
    overflow : auto ;
}
```

overflow : auto ; → automatic kam karega agar content Jada ho toh kud se va kud scroll dika yega.

overflow : scroll ; → scroll dikane ke liye.
overflow : hidden ;
overflow - x : hidden ;
overflow - y : scroll ;

* CSS Resizable :-
resize : ____ ; → work for overflow scroll, hidden, auto except visible.
↓
both
horizontal
vertical

* CSS lists :-
Eg:- html

```html
<ul>
    <li> item1 <li>
    <li> item2 <li>
<ul>
<hr>
<ol>
    <li> item 1 <li>
    <li> item 2 <li>
<ol>
```

# CSS Tables, CSS Function

CSS :-

```
ul {
    list-style-type: ___ ;
    list-style-image: ___ ;      <- image dalna hai toh
    list-style-position: inside;
}
ol {
    list-style-type: ___ ;
    list-style-position: inside;   <- border ke under
}
li {
    border: 1px solid red;
}
```

## * CSS Tables :-

Table meh ~~border~~ ye sab deh sakthe hain.

→ border: ___ ;
border-collapse: collapse; → Table ko same border dene ke liye
border-collapse: Separate; → Table ko Separate border dene ke liye
margin: ___ ;
empty-cells: hide;
border-spacing: ___px ___px ___px ___px;
Top, right, bottom, left

table-layout: auto;
table-layout: fixed;

→ td meh → border, height, width, Padding, text-align, Vertical allign: ___ ; } Same for th
→ top / bottom / center

→ caption meh → border, Padding, height, width, caption-side: ___ ; → bottom / top / etc.

## * CSS Function :-

CSS meh vi function hota hain.
Jis tag ke piche ( ) rahega woh function hain
eg :- url ( ), etc.

Some useful function :- calc ( )

create variable with this func → :root {
    - variable : ___ ;   ← variable name
}

Eg :- html :-

```
<body>
    <div id="func"></div>
    <div id="func1"></div>
</body>
```

# Box Sizing, Initial vs Inherit, Object-fit

css :-

~~# fun~~

```
:root {
    --func-bg : aqua;
    --func1-bg : rgb(231, 81, 17);
}

#func {
    border: 1px solid black;
    background-color: var(--func-bg);
    height: 200px;
    width: calc(100% - 200px);
}

#func1 {
    border: 1px solid black;
    background-color: var(--func1-bg);
    height: 200px;
    width: calc(100% - 200px);
}
```

* **Box Sizing :-**

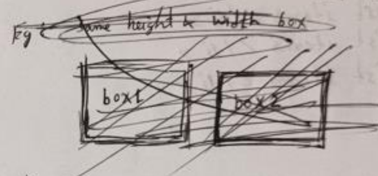
box-sizing : content-box; → by default

Isko laganeh seh box ka size bahar seh barega. content ka Jaga wesah hi rahega.

box-sizing : border-box;

→ Isko laganeh seh box ka size nhi barega Aur content ka Jaga kam hoga, Padding / border lagega ~~content ka under~~ but box ka size nhi barega.

Eg ~~same height & width box~~




* **Initial ~~and~~ vs Inherit :-**

Initial → default css value

Inherit → Parent ka value

* **object-fit :-**

```
img {
    border: 1px solid red;
    width: 500px;
    height: 400px;
    object-fit: ___;
    object-position: ___;
}
```

- contain → aspect ratio bigar jayega
- fit → aspect ratio nhi bigarega
- cover → Kutch Part ni dikayega

# Pseudo-Classes, Pseudo-Element

* **Pseudo-classes :-**

Selector : Pseudo-classes { } → Syntax

Eg:- html:-

```html
<ul>
    <li> list items </li>
    <li> list items </li>
    <li> list items </li>
    <li> list items </li>
    <li> list items </li>
</ul>
```

css:-

```css
li:first-child {
        color: red;
}
```

or

```css
li:last-child {
        color: red;
}
```

or

```css
li:nth-child(3) {
        color: red;
}
```

or

```css
li:nth-child(even) {
        color: red;
}
```

or

```css
li:nth-child(odd) {
        color: red;
}
```

or

```css
li:nth-child(2n) {
        color: red;
}
```

or

```css
li:nth-last-child(2) {
        color: red;
}
```

```css
input:focus {
        border: 2px solid red;
}
```

* **Pseudo-element :-**

selector:: Pseudo-element { } → Syntax

Eg:- html:-

```html
<body>
    <h1> hello </h1>
    <p> Lorem 100 </p>
</body>
```

css:-

```css
p::first-letter {
        font-size: 50px;
        color: blue;
}
```

# Display

```css
P:: first-line {
    color: aqua;
}

h1 :: after {
    content: url(''):
}

h1:: before {
    content: "added by css";
}
```

any text

→ To change Selected Text :-

```css
:: selection {
    background-color: black;
    color: white;
}
```

*** Display :-** display: ____;

↳ block, inline, & inline-block, none

→ **block** :-
- Start ~~with~~ in new line
- Take full width of screen
- can change height and width

→ **inline** :-
- Do not start in new line
- Take required width
- can't change height & width

→ **inline-block** :-
- Do not start in new line
- Take required width
- can change height & width

→ **none** :- Display: none; ⇒ makes content invisible (does n't take space.)

Visibility: hidden; → makes content invisible but takes space.

**Eg:-** html:-

```html
<div class="displayed" id="one"> I am div 1 </div>
<div class="displayed"> I am div 2 </div>
```

css :-

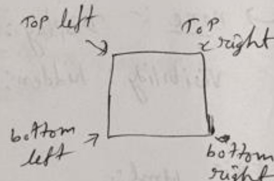
```css
.displayed {
    border: 2px solid red;
    height: 100px;
    width: 500px;
    display: inline; or
    display: inline-block;
}
```

# Position

```
# one {
    display: none;
        or
    visibility: hidden;
}
```

* Position :- ① static → (default)
position : __ ;
② relative
③ ~~Absolu~~ absolute
④ fixed
4 types

→ If relative ~~&~~, absolute, fixed is applied we can use :- Top : __ ;
bottom : __ ;
left : __ ;
right : __ ;




② relative :- → Khud ke position seh change karega position (top, ~~to~~ right, bottom, left).

③ absolute :- → body ~~seh~~ ke corners seh ~~top,~~ top, right, bottom, left change hoga agar parent ka position relative, absolute, fixed nhi diya toh.
Agar parent ka position ya ~~ash~~ ach pas ka parent ka position relative, absolute, fixed ho to uske respect meh ~~pa~~ top, ~~t~~ right, bottom, left change karega.

~~④ fixed :-~~ ~~aur~~
Aur absolute meh woh khud ka jaga chor dega.

④ fixed :- ~~body~~ woh khud ke jaga meh fixed jaga kahi vi rakhe.
Viewing screen meh ~~fixed~~ fixed rahega.
~~Scrool~~ ~~scroll~~ scroll karneh seh vi fixed rahega.

Ex. of Position :-

html :-

```
<div id="parent">
    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>
</div>
```

css :-

```
body {
    height: 1000px;
}
```

# Layers and Z-Index

```
#Parent {
    border: 2px solid black;
    width: 500px;
    position: relative;
    margin: 100px;
}
#one {
    border: 2px solid black;
    background-color: red;
    width: 100px;
    height: 100px;
    display: inline-block;
    position: fixed;
    bottom: 20px;
    right: 20px;
}
#two {
    border: 2px solid black;
    background-color: blue;
    width: 100px;
    height: 100px;
    display: inline-block;
    position: absolute;
    top: 107px;
    left: 0px;
}
#three {
    border: 2px solid ~~red~~ black;
    background-color: green;
    width: 100px;
    height: 100px;
    display: inline-block;
    position: relative;
    top: 107px;
    left: 398px;
}
```

*** Layers and Z-Index :-**

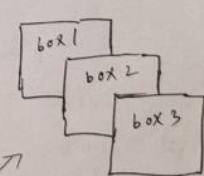
z-Index: ___ ; → Any value

→ Jiska z-Index sabse Jada hoga uska layer Upar dikega.

→ z-Index lagane ke liye position → ~~relative~~ relative / absolute / fixed hona zaruri hain.

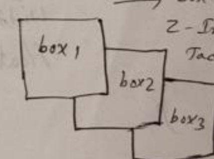
Eg :-

Suppose box1 ka z-Index : 3;
box2 " " : 2;
box3 " " : 1;




yaha box3 ka z-Index Jada hain

Toh ye




→ box1 ka z-Index Jada hain.

## Float and Clear

### * Float and clear :-

float : __;
↳ left
right } change Position

clear : __;
↳ left
right
both } leave space

Example:-

Html:-

```
<div id = "Paragraph">
    <P> lorem 50 <img Src = " " alt = " "
        id = "img1" > lorem 50 </P>
</div>
```

css:-

```
#Paragraph {
    color : blue;
}

#img1 {
    height : 150 Px ;
    Width : auto;
    Float : right;
}
```

Example:-

html:-

```
<div id = "bar">
    <ul>
        <li> Home </li>
        <li> About </li>
        <li> contact us </li>
        <li> Share </li>
        <li> Courses </li>
    </ul>
</div>
```

css:-

```
# bar {
    background-color : black;
    color : white;
    overflow : auto;
    Padding : 0px 0px 16px 185px;
    margin : 0px -8px;
}

li{
    list-style-type : none;
    float : left;
    Padding : 10px 30px;
}
```

Example :-

html :-

```
<div id= "blog">
    <P> <img src=" " alt=" " id= " ">
        <img src=" " alt=" " id= " ">
        lorem 100 </P>
</div>
```

css :-

```
#blog {
        color : green;
}

#image1 {
        Width : 100Px ;
        height : auto ;
        float : right ;
}

# image2 {
        width : 100Px ;
        height : auto ;
        float : right ;
        clear : both ;
}
```

Example :- html

```
<div id = "content">
        <P> content </P>
</div>
<div id = "Sidebar">
        <P> Sidebar </P>
</div>
<dive id= "footbar">
        <P> footbar </P>
</div>
```

css :-

```
# content {
        Width : 70% ;
        background - color : blue ;
        height : 200Px ;
        float : left ;
}
# Sidebar {
        width : 30% ;
        background - color : red ;
        height : 150 Px ;
        float : left ;
}
# footer {
        background - color : green ;
        Clear : both ;
}
```

# CSS 2D Transforms

* CSS 2D Transforms :- transform : __ ;

→ translate () → move an element from its current Position
↓
Parameter in x-axis & y-axis.

~~rotate()~~ → eg:- transform : translate (50px, 100px);

→ rotate () :- rotates an element clockwise or counter-clockwise according to a given degree.
eg:- transform: rotate (20 deg);

→ scale () :- increases or decrease the size of an element
↓
Parameter in width & height.
eg:- transform: scale (2,3);

→ scale x() :- increase / decrease the width of element
eg:- transform: scale x (2);

→ scale Y() :- increase/decrease the height of element
eg:- transform: scale Y(3);

→ skew x () :- Skew an element along the x-axis by the given angle.
eg:- transform: skew x(20 deg);

→ skew Y() :- skew an element along the y-axis by the given angle.
eg:- transform: skew Y(30 deg);

→ skew() :- skew element along x & y-axis by given angle
Eg:- transform : skew (20 deg, 10deg);

→ matrix() :- have 6 Parameter.
~~Eg mat~~
matrix ( scale x(), skew Y(), skew x(), scale Y(), translate x(), translate Y())
Eg:- transform: matrix (1, -0.3, 0, 1, 0, 0);

Example :-

html :-

```
<div id="wrapper">
    <a href=" "> click here </a>
</div>
```

css :-

```
#wrapper {
    Padding : 10px 10px;
    margin : 150px 300px;
    background-color : aqua;
    transform: matrix(1,-0.3,0,1,0,0);
}

a{
    text-decoration : none;
    background-color : black;
```

# CSS 3D Transforms

color : alice blue;
Padding : 20Px;
display : inline-block; → required to use transform.
transform : skew(-30deg, -20deg);
on
transform : skew X(45deg);
on
transform : skew Y(33deg);
on
transform : rotate(60deg);
transform-origin : 0% 100%; → To change the origin of element
}

a : hover {
display : inline-block; → require to use transform
transform : translate(30Px, -30Px);
on
transform : rotate(-180deg);
on
transform : scale(2,3);
on
transform : scale(0.8, 0.7);
on
transform : scale X(3);
on
transform : scale Y(4);
}

* CSS 3D Transforms :-

→ Perspective :- Specifies the Perspective on how 3D elements are viewed.
Eg:- Perspective (300 Px);

→ Perspective-origin :- Specifies the bottom position of 3D elements.

→ rotate X( ) , eg:- transform: rotate X(150deg);

→ rotate Y( ) , eg:- transform: rotate Y(150 deg)

→ rotate Z( ) , eg:- transform: rotate Z(150 deg)

~~Exa~~

→ rotate 3d ( __ , __ , __ , __ )
x y z angle

Example :-

transform: Perspective(300Px) rotate(-2, 54, -17, 40deg);

# CSS Transition, CSS Animation

## * CSS Transition :-

→ Short hand of transition :-

transition: property, duration, timing-function, delay;

Example of css transition :-

html:-
```
<body>
    <div id= "change"> </div>
</body>
```

css:-
```
#change {
    height : 10px;
    width: 10px;
    margin: 100px 0px;
    background-color: red;
    transition-property : width;
    transition-duration: 2s;
    transition-delay: 500ms;
    transition-timing-function : steps(5000);
    or
    transition: width 3s cubic-bezier(0.19, 1, 0.22, 1) 400ms, background-color 2s;
}
#change :hover {
        width : 100%;
        background-color : black;
}
```

s → second
~~ms → second~~
ms < s

## * CSS Animation :-

Example :-

html :-
```
<div id = "animate" > </div>
<div id = "animate1" > </div>
```

css :-
```
@keyframes moving {
    0% { width : 100px; }
    10% { width : 20%; background-color : red; }
    20% { width : 40%; background-color: blue; }
    30% { width: 60%; background-color : green; }
    40% { width: 80%; background-color : yellow; }
    50% { width: 100%; background-color : orange; }
    60% { width: 80%; background-color: yellow; }
    70% { width: 60%; background-color : green; }
    80% { width : 40%; background-color : blue; }
    90% { width : 20%; background-color : red; }
    100% { width : 100px; }
}
#animate {
    background-color : black;
    height : 50 px;
    width : 100 px;
```

## Print Style in CSS, Responsive Web Design, Viewport meta tag, Media queries

```
animation-name : moving;
animation-duration : 5s;
animation-iteration-count : infinite;
animation-direction : alternate;
animation-timing-function : ease-in-out;
animation-delay : 50 ms;
}

@Keyframes movingSquare {
    from {
        left : 50px;
    }
    to {
        left : 900px;
    }
}

#animate1 {
    background-color : gold;
    width : 100px;
    height : 100px;
    margin : 50px 0px;
    position : relative;
    left : 50px;
    animation-name : movingSquare;
    animation-duration : 2s;
    animation-iteration-count : infinite;
    border-radius : 60px;
}
```

→ we can use from to or 0% to 100% $\underbrace{\phantom{xx}}_{\text{in between.}}$

*** Print style in CSS :-**

css meh

- @media screen { } → Screen ke liye
- @media print { } → Jab koi aabke web ko print karega toh kesa dikhega usko style kar sakte hain.

*** Responsive web design :-** It is very important.

*** Viewport Meta tag :-**

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

*** Media Queries :-**

→ check how big is my browser

Example of media Queries :-

html :-

```
<div id="div1">
    <h1>div 1</h1>
</div>
<div id="div2">
    <h1>div2</h1>
</div>
```

# Advanced selectors

CSS:-

```
#div1{
    background-color: rgb(215, 252, 6);
    width: 70%;
    height: 200px;
    float: left;
}
#div2{
    background-color: rgb(10, 245, 135);
    width: 30%;
    height: 200px;
    float: left;
}
h1{
    font-size: 80px;
}
@media screen and (max-width: 700px) {
    #div1{
        width: 70%;
    }
    #div2{
        width: 30%; }
    h1{
        font-size: 40px;
}
```

```
@media screen and (min-height: 700px) and (max-height:
                                              1200px) {
    #div1{
        height: 400px;
    }
    #div2{
        height: 400px;
    }
}
```

→ CSS framework :- bootstrap ↙ Learn

→ used for making web responsive more easily

## * Advanced selectors :-

1. Type selector :- Ex :-

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

2. ID selector :- Ex :-

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

3. class selector :- Ex :-

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

4. Pseudo class and Pseudo Element :-

Ex :-

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

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

5. Attribute Selector :-

Ex :-

```
a [href = "https://example.com"] {
        color: red;
}
```

6. universal Selector :-

Ex :-

```
* {
    color: red;
    border: 1px solid black;
}
```

7. combinator selector :-

→ Descendant combinatory (child at any stage)

element element {}

Ex :-

```
.red-p span {
        color: green;
}
```

→ child combinatory (Direct child)

element > element {}

Ex :-

```
.red-p > span {
        color: green;
}
```

→ Adjacent combinatory (right after sibling)

Ex :-

```
.red-p + h1 {
        color: green;
}
```

→ General sibling combinatory (all sibling)

Ex :-

```
.red-p ~ h1 {
        color: green;
}
```

→ Add selector general sibling combinatory (right after sibling)

Ex :-

```
.red-p, h1 {
        color: green;
}
```

Examples :- html

```
<div id="all">
    <h1> selector </h1>
    <P> lorem 100 </P>
    <P> lorem 50 </P>
    <a href="www.google.com">Google </a>
    <br>
    <a href="www.firefox.com">firefox </a>
    <ul>
        <li> list 1 </li>
        <li> list 2 </li>
        <li id="list 3"> list 3 </li>
        <li> list 4 </li>
        <li> list 5 </li>
        <li class="list 67">list 6 </li>
        <li class="list 67"> list 7 </li>
        <li> list 8 </li>
    </ul>
    Name: <input type="text">
</div>
<P> lorem 50 </P>
<a href="www.amazon.in">amazon </a>
<P> lorem 50 </P>
```

css :-

```
/* attribute selector */
a[href="www.firefox.com"]{
        colon: red;
        font-size: 30px;
}
input [type="text"]{
        colon: red;
}
/* universal selector */
*{
    border: 1px solid red;
}
/* Descendant combinatory (element element) */
divP{
    colon: chartreuse;
}
```

~~/* Descendan~~ /* or */

```
#all p{
    colon: yellow;
}
```

# Specificity

```
/* child combinatory (element > element) */
ul > li {
        color : blue ;
}
/* adjacent combinatory (element + element) */
div + p {
        color : pink ;
}
/* general sibling (element 1 ~ element 2) */
div ~ p {
        color : brown;
}
/* Add selector */
#list3 , .list67, h1 {
            color: rgb (4, 104, 24);
}
```

* Specificity :-

- If one element is styled by more than one rule, which style will be applicable.

<u>general rule</u> :-
- !Importand will override everything
- Inline can override everything except !importand.
- External will work according to specificity.

<u>Specificity Rule</u> :- (#) Id selector has highest value
(*) universal selector has lowest value.

<u>Specificity Rule Trick (Point System)</u> :-

- 10,000 Point to !importand
- 1000 to inline
- 100 to id
- 10 to class, attribute or pseudo-class
- 1 for element selector & pseudo-elements
- 0 to universal selector

```
p {
   color : green;        ⟶   1 point
}
.red-p {                 ⟶   10 point
    color : red ;
}
   div.red-p {           ⟶   11 point
       color: blue ;
   }
```

# Multiple columns, Flexbox

→ If same rule written two/more times in external sheet then last rule will be applicable.

* Multiple columns :-

Ex :-

html :-

<h1> lorem20 </h1>

<P> lorem 500 </P>

css :-

```
p{
  column-count : 3 ;
  column - gap : 50px ;
  Column- rule - Style : solid ;
  Column - rule - width : 2px ;
  column - rule - color : red;
        Or Shorthand
  column-rule : 2px Solid red ;
  column - width : 300px ;
}

h1 {                    ← so that h1 doesn't
                          divide in column.
  column - Span : all ;
}
```

* Flex box :-

~~Flexcontai~~

flex container Properties are: • flex-direction
• flex - wrap
• flex - flow
• Justify - content
• align - items
• align - content

Examples :

~~html~~

→ Example of flex- direction / wrap ~~&~~ / flow.

html :-

```
<div id= "Parent_container">
   <div class = "child_container" >1 </div>
     ''    ''    ''    ''   2 ''
     ''    ''    ''    ''   3 ''
                 ''         4 ''
                 ''         5 ''
                 ''         6 ''
                 ''         7 ''
                 ''         8 ''
   <div class = "child-container" > 9 ''
</div >                     10 '' </div>
```

css :-

```
#Parent_ container {
   background-color : brown;
   display: flex ;
   flex - direction : row ;
                    : row - reverse ;
                    : Column ;
                    : Column - reverse ;

   flex - wrap : wrap;
                 no wrap;
```

:wrap-reverse;

flex-flow : row-wrap;

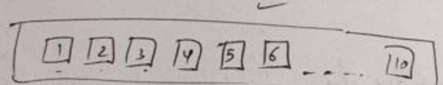
}

.child_container {

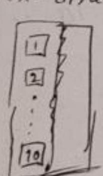
background-color : aqua;
margin : 10px;
padding : 10px;
width : 100px;
text-align : center;

}

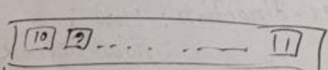
→ Here:- flex-direction: row;



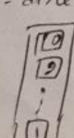

→ flex-direction : column;



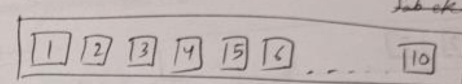

→ flex-direction : row-reverse;




→ flex-direction : column-reverse;

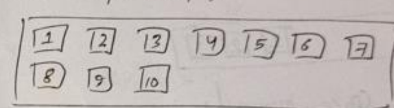



→ flex-wrap : ~~no~~ nowrap; ~~Iska matlab~~ ~~Sabko ek~~ ~~Sab ek hi Width~~



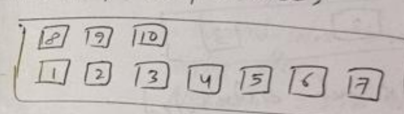

→ yeh row nowrap hain

→ flex-wrap: wrap;




→ yeh row wrap hain.

→ flex-wrap: wrap-reverse;




→ yeh row wrap-reverse hain.

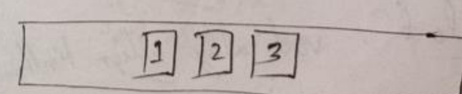
⇒ flex-flow: ____ ____; → short hand

yaha flex-direction:

yaha flex-wrap:

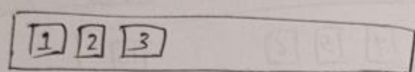
~~Justify-con~~

⇒ justify-content: center;

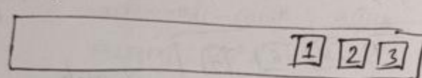



→ center peh

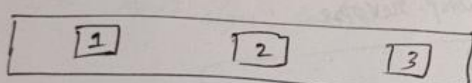
→ justify-content : flex-start ;




→ justify-content : flex-end ;

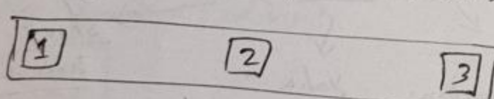



→ justify-content : space-around ;




same width meh distributed

→ justify-content : space-between ;




space in between.

→ justify-content : ~~vertica~~ horizontally allign Kartha hain

→ align-content :- vertically allign Kartha hain

→ Example of ~~justify~~ align-items :-

~~→ align - items~~

html :-

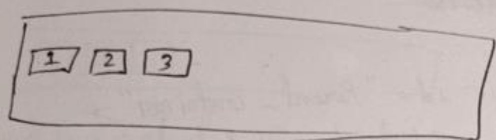
```html
<div id = "Parent_container">
    <div class="child-container" > 1 <div>
      "          "                    2  "
      "          "                    3  "
</div>
```

css :-

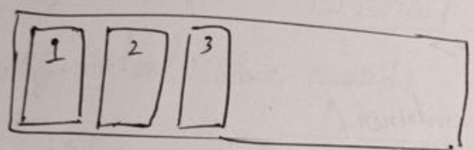
```css
#Parent_container{
    background-color : brown;
    display : flex;
    flex-flow : row nowrap;
    align-items : stretch;
                  flex-start ;
                  flex-end ;
                  center ;
}

.child-container{
    background-color : aqua ;
    margin : 10px ;
    Padding : 10 px ;
    width : 100 px ;
    text-align : center ;
}
```

here, If

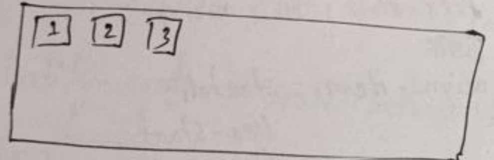
→ ~~#~~ align-items : center;



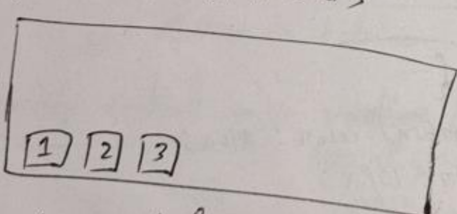
→ align-items : stretch;



→ align-items : flex-start;



→ align-items : flex-end;



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

→ Ex of align-content :-

html :-

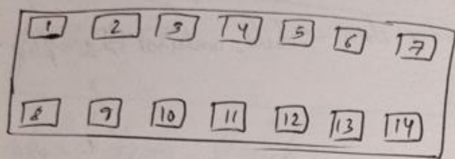
```
<div id = "Parent_container">
        <div class = "child_container"> 1 </div>
                                        2
                                        3
                                        4
                                        5
                                        6
                                        7
                                        8
                                        9
                                        10
                                        11
                                        12
                                        13
                                        14
</div>
```

css :-

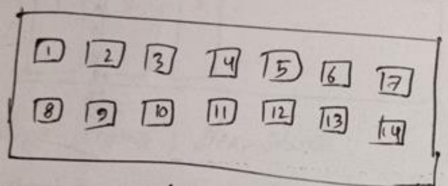
```
#Parent_container {
        background_color : brown;
        display : flex;
        flex-flow : row wrap;
        height : 300px;
        align-content : space-around;
                        space-between;
                        center;
                        flex-start;
                        flex-end;
}
.child_container {
        background-color : aqua;
        margin : 10px;
        padding : 10px;
        width : 100px;
        text-align : center;
}
```

→ here ~~set~~ if

→ → align-content : space-between;



→ align-content : space-around;



→ align-content : center;
→ align-content : flex-start;
→ " " : flex-end;

} same as before (try it)

→ ek hi child-container ko center ~~meh~~ meh laneh ke liye :- justify-content : center;
align-items : center;

→ child meh Vi Properties laga sakthe hain

The flex item ~~Proter~~ Properties are :—

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

→ ~~order~~ example ~~3~~ of order & flex-grow :-

html :-

```
<div id="Parent_container">
    <div class="child-container" id="one"> 1 </div>
          "                      id="two"> 2 </div>
          "                      id="three"> 3 </div>
</div>
```

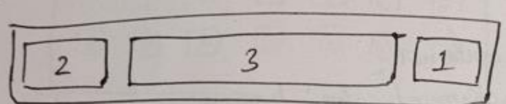
css :-

```
#Parent_container{
        background-color : brown;
        display : flex;
        flex-flow : row wrap;
}
.child_container{
        background-color : aqua;
        margin : 10px;
        Padding : 10px;
        width : 100px;
        text-align : center;
        text-align : center;
}
#one{
        order : 3;
        flex-grow : 1;
}
```

```
#two {
    order : 1;
    flex-grow : 1;
}
#three {
    order : 2;
    flex-grow : 8;
}
```

in ratio

output :-




Here, flex-grow is in ratio 1 : 1 : 8

→ Eg. of flex-shrink & flex-basis :-

html :-

```
<div id = "Parent_container">
    <div class= "child_container" id= "one">1</div>
        ,,                                  2
        ,,                                  3
        ,,            id= "four"> 4 </div>
        ,,                                  5
        ,,                                  6
        ,,                                  7
        ,,                                  8
</div>
```

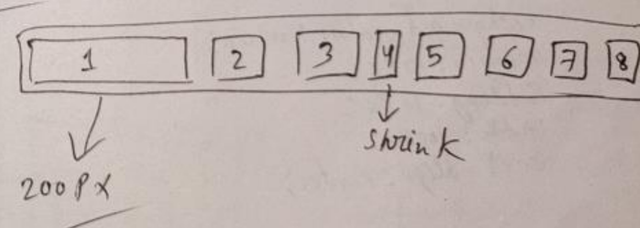
css :-

```
#Parent_container {
    background-color : brown;
    display : flex;
    flex-flow : row nowrap;
}
.child_container {
    background-color : aqua;
    margin : 10px;
    Padding : 10px;
    width : 100px;
    text-align : center;
}
#one {
    flex-shrink : 1;
    flex-basis : 200px;
}
#four {
    flex-shrink : 100;
}
```

flex-shrink : 1 → 1 is default value

flex-basis : 200px → increase width size.

output :-



# CSS Grid

→ flex : flex-grow, flex-shrink, flex-basis;
↓
Shorthand

Eg:- flex: 0 0 200px ;

→ Ex of align-self :-

html:-

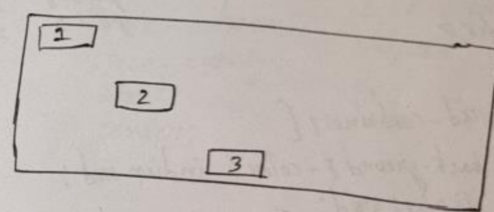
```
<div id="Parent_container">
    <div class="child_container" id="one">1</div>
              "                  id="two">2
              "                  id="three">3
</div>
```

css:-

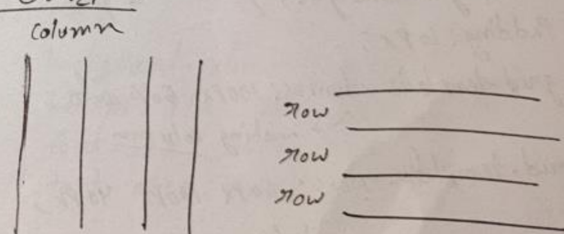
```
#Parent_container{
        background-color: brown;
        display: flex;
        flex-flow: row nowrap;
        height: 300px;
}

.child_container{
        background-color: aqua;
        margin: 10px;
        Padding: 10px;
        width: 100px;
        text-align: center;
}
```

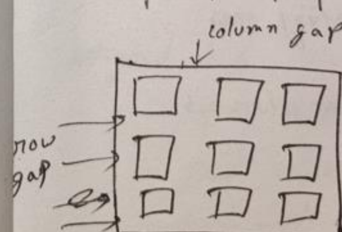
```
#one{
        align-self: flex-start;
}
#two{
        align-self: center;
}
#three{
        align-self: flex-end;
}
```

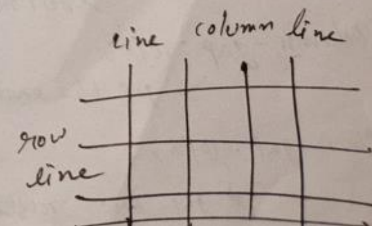
output:-




* CSS Grid :-



→ Examples of grid :-

html:
```
<div id="grid-container">
    <div class="grid-item" id="grid-1">1 </div>
      ,,                        grid-2   2
      ,,                        grid-3   3
      ,,                        grid-4   4
      ,,                        grid-5   5
      ,,                        grid-6   6
      ,,                        grid-7   7
      ,,                        grid-8   8
                                grid-9   9
</div>
```

css:
```
#grid-container {
    back-ground-color: indian red;
    display: grid;
    /*display : inline grid;*/  → for inline & grid
    Padding: 10Px;
    grid-template-columns: 100Px 50% auto;
                    → making column
  grid-template-rows: 50Px 100Px 40Px;
                    → for making row
  column-gap: 10Px;
          → gap b/w ~~rows~~ columns
  row-gap: 10Px;
          → gap b/w rows

  justify-content: start;
                   end;           → if width is
                   space-around;    left then
                   space-between;   to adjust
                   space-evenly;
                   center;

  align-content: end;           → if height is
                 space-around;    left then to
                 space-between;   adjust.
                 space-evenly;
                 start;
                 center;
}

.grid-item {
    background-color: aqua;
    border: 1Px solid black;
    font-size: 30Px;
    text-align: center;
}

#grid-1 {
    /* To combine column */
    grid-column-start: 1;   → count
    grid-column-end: 4;       grid-lines
                              and use.
```

or
/* shorthand */

grid-column : 1/3 ; → want to combine 2 column starting by ~~1~~ column line 1

grid-column : 1/ span 2;

start end

/* To combine rows */

grid-row - start: 1 ;
grid-row - end : 3 ;

/* shorthand ~~of~~ to combine rows */

grid-row : 1/5 ; → count by row-lines

start end

grid-row : 1/span 3 ; → span is for how many rows.

/* To combine both row & column */

/* $\underbrace{\text{row line/ colum line}}_{\text{start}}$ / $\underbrace{\text{row line/columnline}}_{\text{end}}$ */

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

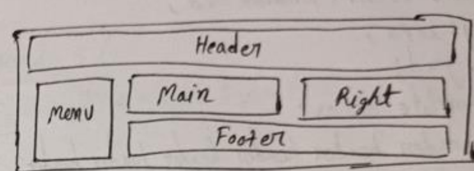
or

grid-area : 1/1 /span 2/span 2;

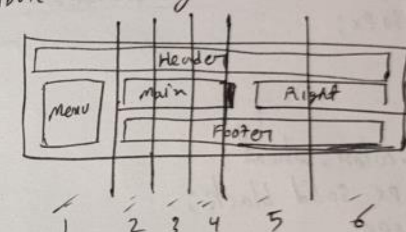
}

Ex of grid :-

To make output like :-




tip: ~~first~~ distribute it by lines first




Source code :-

html:-
```
<div id="grid-container">
    <div class="grid-item" id="grid-1"> Header
                                          </div
         "                   grid-2 " Menu "
         "                   grid-3 " Main "
         "                   grid-4 " Right "
         "                   grid-5 " Footer "
</div>
```

CSS Validation

css :-

```
#grid-container{
    background-color: indian red;
    Padding: 20px;
    display: grid;
    grid-template-areas:
        'header header header header header header'
        'menu main main main right right'
        'menu footer footer footer footer footer';
    grid-gap: 10px;
}
.grid-item {
    background-color: wheat;
    border: 1px solid black;
    Padding: 20px;
    font-size: 40px;
    text-align: center;
}
#grid-1 {
    grid-area: header;      → we can write name in grid-area also.
}
#grid-2 {
    grid-area: menu;
}
#grid-3 {
    grid-area: main;
}
```

```
#grid-4 {
    grid-area: right;
}
#grid-5 {
    grid-area: footer;
}
```

* CSS validation :- go to browser and check your CSS ~~code~~ source code.