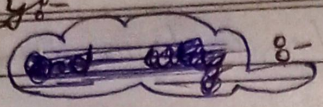
61

(10). # Flex-Box

① Flex theory & Component
② container & child property.

Date .....

(1). Theory :-

what is :- 

- make it easier to design flexible responsive layout structure without using float or positioning.

- flex-box deals with layout in one dimension at a time - either as a row or column.
But grid is two-dimensional model that controls columns and rows together.

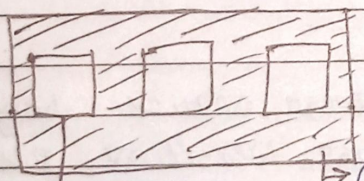
Important components :-

(1). Flex-container :- (Parent container jisme display flex ho)

- Flex-container create karne ke liye, parent container me likho `display: flex;`

↳ Is se direct childrens iss container ke flex items bn jayenge.

eg:-



Notes :- (i). Hum control kr sakte hai ki jo flex-container hai wo kaisa ho, wo inline container ho ya block container.

(i) for inline level flex container,

```
display: inline flex;
   or
inline-flex;
```

(ii). for block level flex container,

```
display: block flex;
   or
flex;
```

Spiral

Note:2⇒ flex-container content/child/flex-items default behaviour :-

(i) flex-direction : row (items displayed in a row).

(ii) Item start from start edge of main-axis.

(iii). Items do not stretch on main dimension but can shrink.

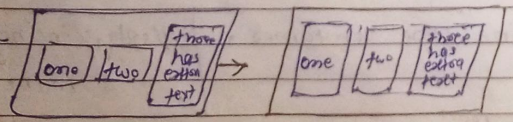
(iv). Item stretch hoga cross-axis me size ko fill krne ke liye.

(v). flex-basis property set to auto.

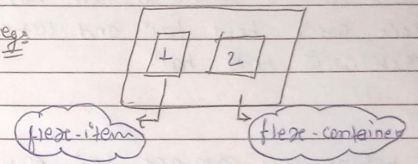
(vi). flex-wrap : no wrap;

Note:3⇒ by default upr wali saari cheeze applied hongi; and jiski wagan se saare flex-items ek row me honge and agr flex-item jyada hoge to wrap nahi hoga balki overflow hoga container me se.

And agr kuchh item ko be honge to iski wagan baaki chhote items v stretch ho jayega utna hi.

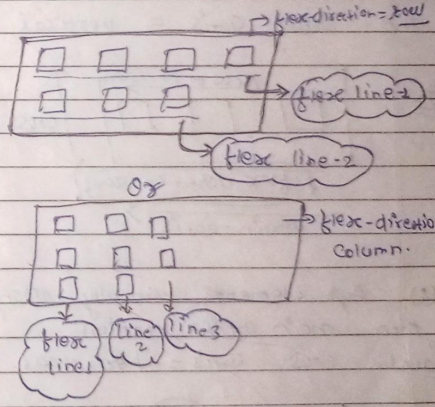
eg:- 

2. flex-item :-
Iis parent me display : flex likha ho usse parent container ko flex container kehte hai and uske childs ko flex-item kehte hai.

eg:- 


```
<div class="container">
  <div class="child">
  <div class="child">
</div>
```

3. flex-line :-
(align-content deal with this).



→ flex-line horizontal ya vertical line hoti hai jisper flex-items hote hai.

flex-line ek se jyada ho sakta hai flex container ka. due to flex-wrap.

flex-line horizontal ya vertical kisi taraf ho sakta hai depends on flex-direction.

Note:2 ⇒ flex-container content/child/flex-items default behaviour :-

(i) flex-direction : row (items displayed in a row)

(ii) item start from start edge of main-axis.

(iii) Items do not stretch on main dimension but can shrink.

(iv) Item stretch hoga cross-axis ke size ko fill krne ke liye.

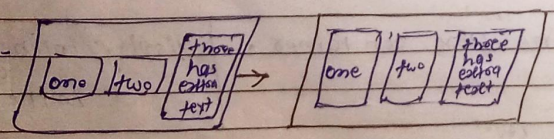
(v) flex-basis property set to auto.

(vi) flex-wrap : no wrap;

Note:3 ⇒ by default upar wali saari cheeze applied hongi, and jiski wajah se saare flex-items ek row me honge and agr flex-item jyada hoga to wrap nahi hoga balki overflow hoga container me se.

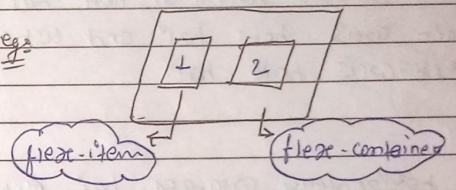
And agr kuchh item 1 ko shrink honge to iski wajah baaki chhote items v stretch ho jayega utna hi.

eg:-



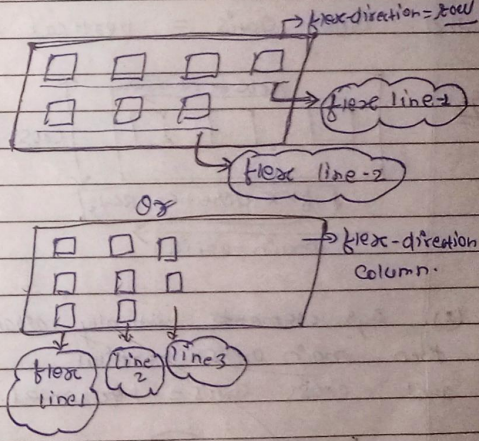
2. flex-item :-
jis parent me display : flex likha ho usi parent container ko flex container kehte hai and uske childs ko flex-item kehte hai.

eg:




```
<div class="container">
  <div class="child1"></div>
  <div class="child2"></div>
</div>
```

3. flex-line :-
(align-content deal with this).



→ flex-line horizontal ya vertical line hoti hai jispar flex-items hote hai.

flex-line ek se jyada ho sakta hai flex container ka. due to flex-wrap.

flex-line horizontal ya vertical kisi taraf ho sakta hai depends on flex-direction.

Date 6/1

3. Axis :-

→ there are two types of axis's.

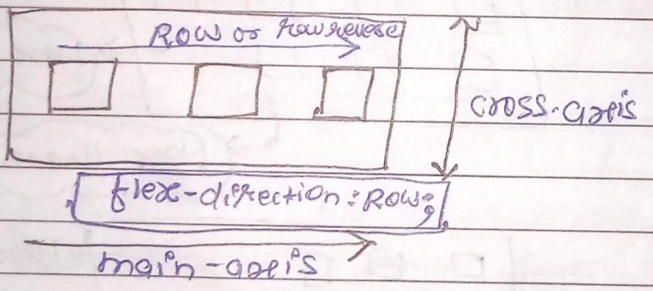
→ Flex-box me items ya to horizontally hote hai ya vertically depends on flex-direction is row or column.

To jo main direction hamne di hoti hai (row or column) wo main axis hota hai and uske perpendicular cross-axis hota hai.

Situations:-

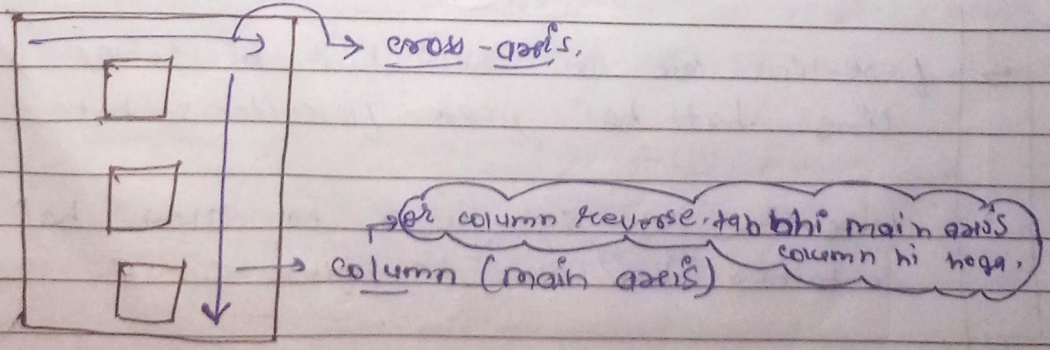
(1). Agr elements horizontally arranged hai row me, then main axis = horizontal } by default
and cross axis = vertical.

eg:-



(2). Agr elements vertically arranged hai column me, then main axis = vertical
and cross axis = horizontal.

eg:-



Note: ① alignment main-axis pr = justify content.
② " cross-axis pr = align element. Spiral

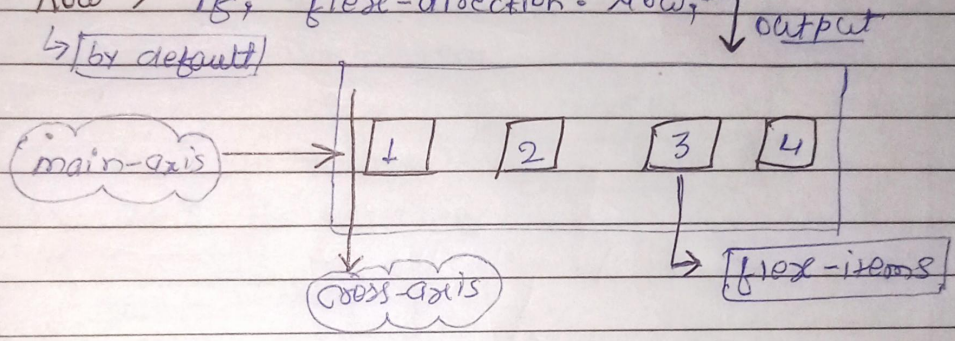
Date .....

(2): # Flex-Container properties :-

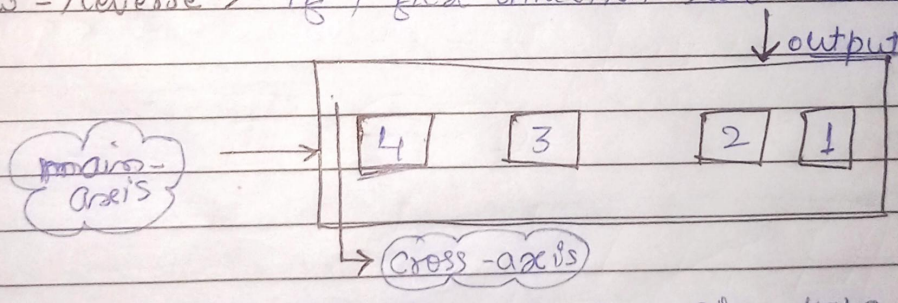
(i). Flex-direction :-
ye bata ta hai ki, main or cross axis kon sa hoga (horizontal or vertical) and kaise flex item arrange hoga, row-wise ya column wise.

→ values : → by default

(a). Row → if, flex-direction : row;

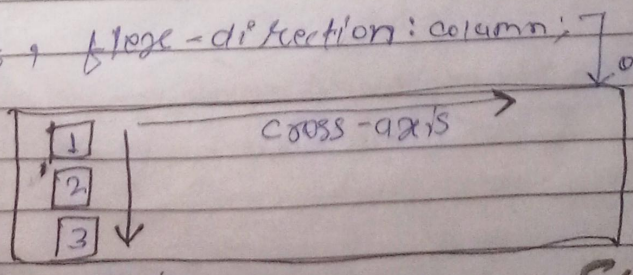


(b). Row-reverse → if, flex-direction : row-reverse;



Note: ya to doosre side se 1, 2, 3, 4 aise kahe samajh lo ya 4, 3, 2, 1 peche wale jagah se aise read kahe samajh lo row-reverse.

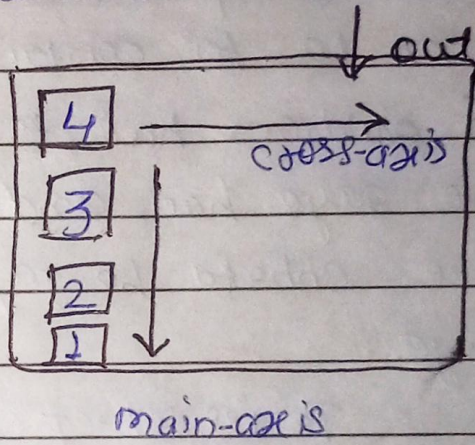
(c). column → if, flex-direction : column;



Spiral

Date .....

(d). Column-reverse → it, flex-direction: column-reverse;



Note:- column-reverse me ya to niche se 1, 2, 3, 4 likhe samajho ya fir pehle wale jagah se hi likha yani 4, 3, 2, 1 samjho.

X X

Date .....

(ii). Flex-wrap :-

Maan lo ki container ka size 300px hai and usme 5 children hai and sabhi 100px ke hai to 500px ho gaye hai and ye wo overflow karenge ya unka size chhota ho jayega taaki container me fit ho paye.

To isi object ko solve karne ke liye flex-wrap aaya hai, ye simply flex-items ko multiple flex-line me convert kr deta hai with original width of flex items.

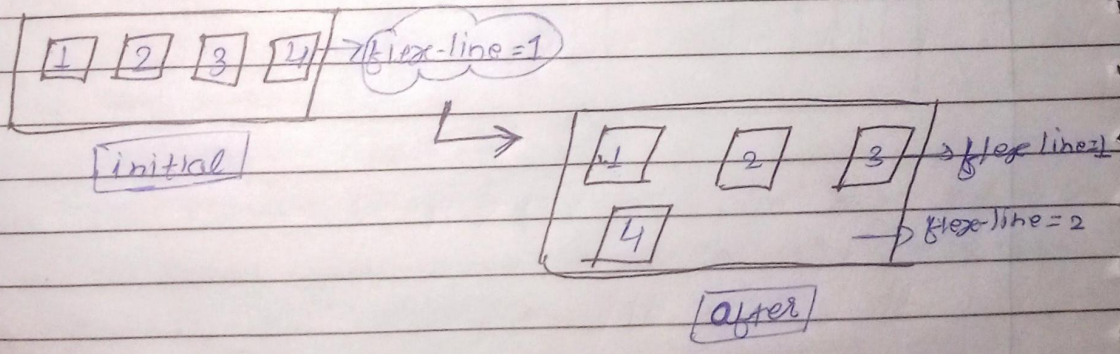
values :

→ by default

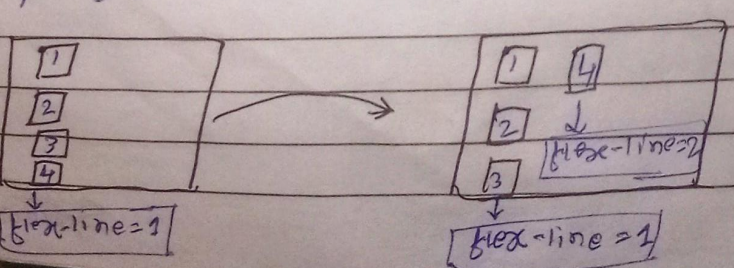
(a). nowrap → if, flex-wrap : nowrap ;

↳ all things are same.

(b). wrap → if, flex-wrap : wrap ; ↓ output



Note :- Agar flex-item ka main axis column ho to,



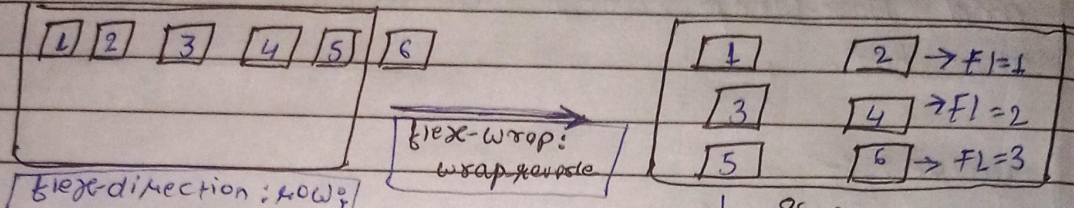
Spiral

FL = flex-line

Date .....

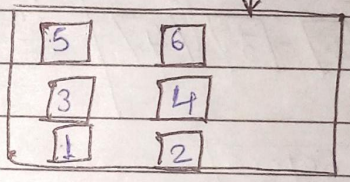
© wrap-reverse :-

case 1 : if main axis is horizontal ,



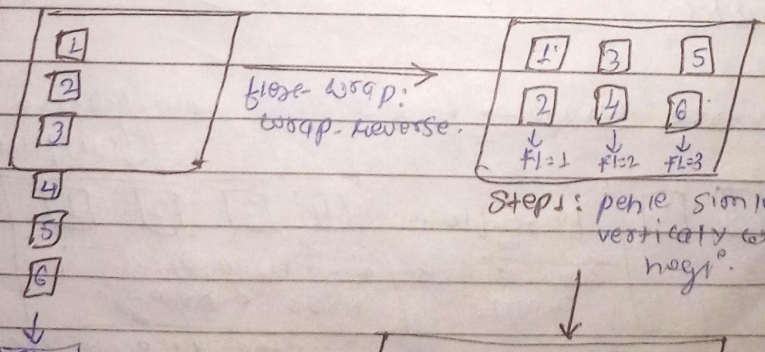
Step-1 = pehle simple wrap karo.

Yahan row-wise wrap hoga kyuki main-axis row hai.

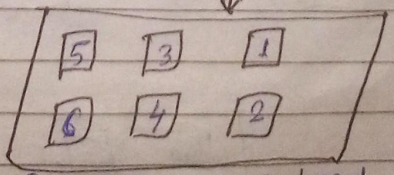


Step-2 = ya to upar wali-line or row ko niche se interchange. ya fir niche se row-wise 1, 2 likhna start karo.

case 2 : if main-axis is vertical ,



Step1 : pehle simple vertically wrapping hogi.



Step2 : wrap ke baad, FL uthao and last me le jao and wahan se padna shuru karo. ya sidha last se hi start karo

Spiral

Date .....

(iii). **flex-flow property :-**

flex-flow property se hum flex-direction and flex-wrap dono ko ek sath apply kr sakte hai :

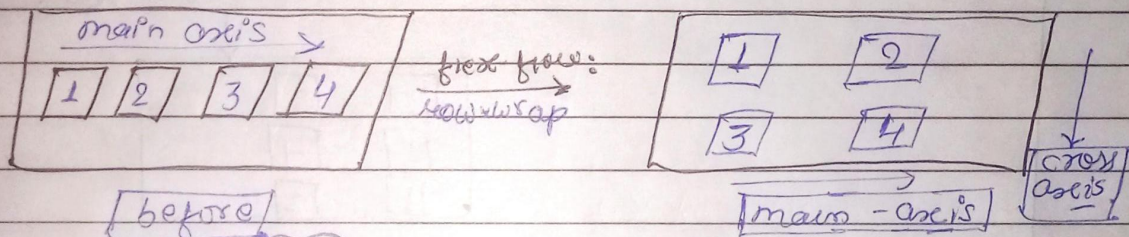
(c).

values :-

case 1: with row,

(a). row wrap ⇒ if, flex-flow : row wrap ;

↓ output.



before
by default

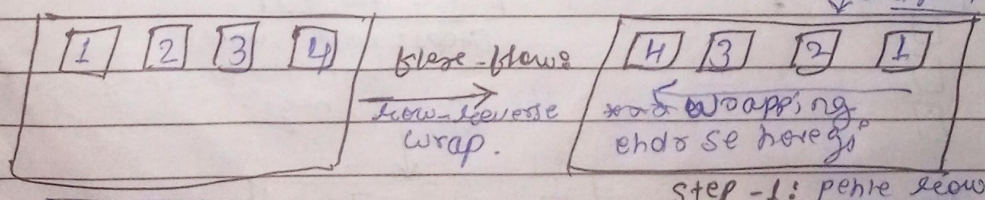
Step-1 : row me by default hai hi bs wrap krdo.

Notes- row wrap means direction ko row set krdo, fir. usko wrap kardo, main axis ko horizontal consider karke.

(d)

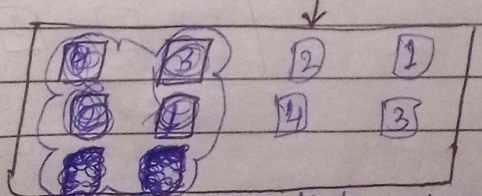
(b). row-reverse wrap → flex-flow : row-reverse wrap ;

↓ output.



before

Step-1 : pehle row ko reverse krdo.

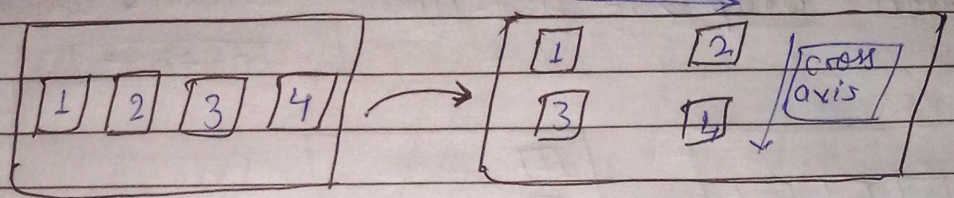


step 2 : fir, wrap krdo. main axis ko horizontal consider karke.

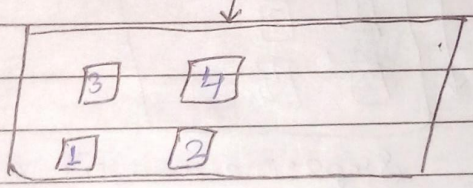
Spiral

Date 7th

(c). row wrap-reverse → if, flex-row: row wrap-reverse;

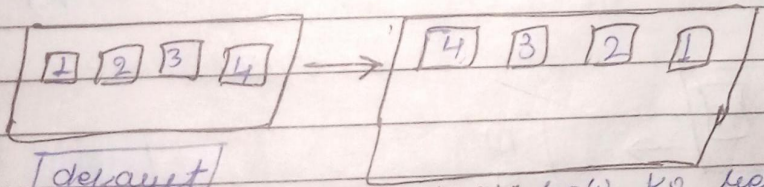


Step1: direction ko row set krdo jo by default row me hi hai, fir wrap krdo. main-axis ko horizontal or row ko consider kr ke.

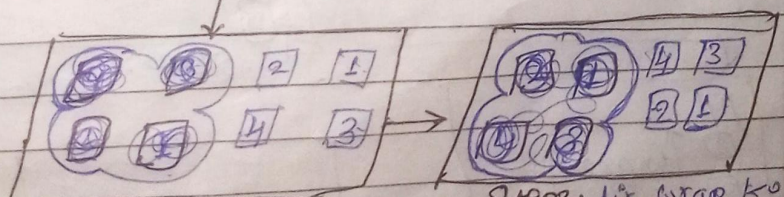


Step2: uske baad, wrap ko reverse krdo.

(d). row-reverse wrap-reverse →



step1: row ko reverse krdo. (flex-direction: row-reverse)



Step2: fir, wrap krke main axis ko horizontal consider kr ke.

Step3: fir, wrap ko reverse krdo.

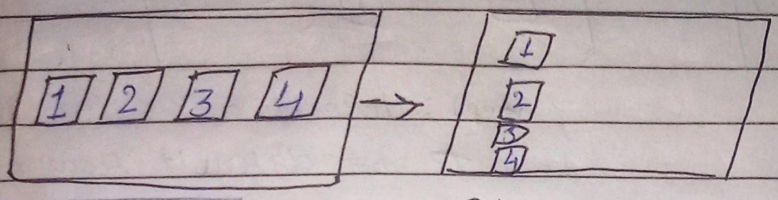
Spiral

Date .....

case 2 : with column ;

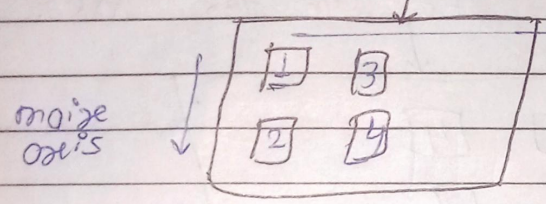
(a). column wrap → if flex-flow : column * wrap ;

↓ output



default

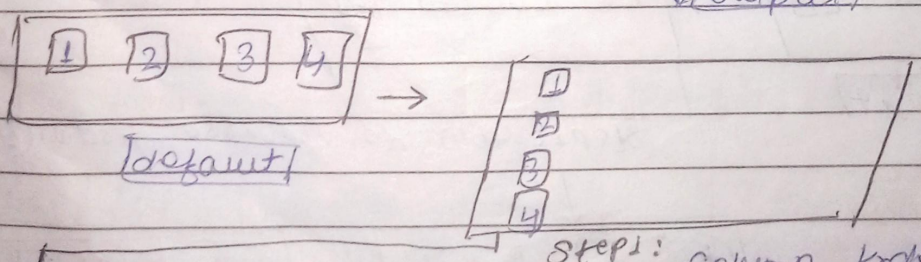
Step1 : flex direction ko column krdega.



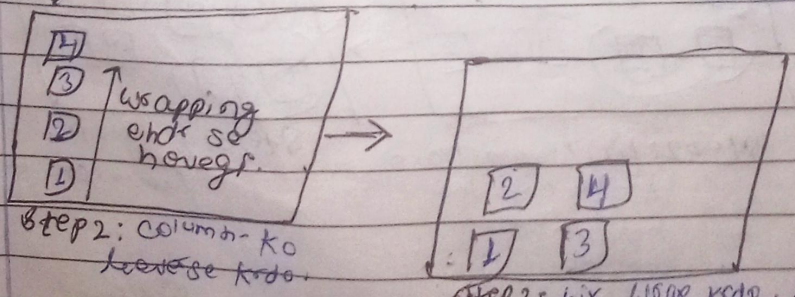
Step2 : main-axis ko column mein kr wrap krdega.

(b). column-reverse wrap → if flex-flow : column-reverse wrap ;

↓ output



Step1 : column krdo direction.



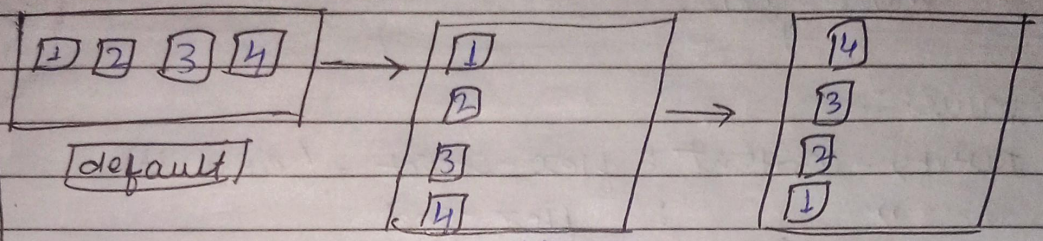
Step2 : column-ko ~~reverse krdo~~.

Step3 : fir wrap krdo, main axis vertical consider krke.

Spiral

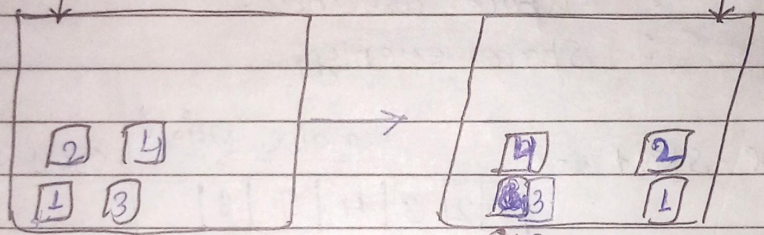
Date .....

(a) column-reverse wrap-reverse →



step1: direction ko column karda.

step2: fir, column ko reverse karda.



step3: reverse seed column ko wrap karda. vertical ko main axis maan ke.

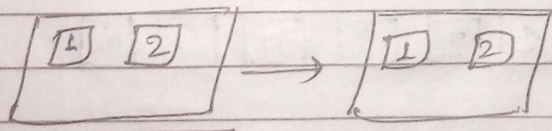
step4: fir 1 ko doosre side le aaye isse wrap-reverse ho gaye.

case 2: general,

→ by default of wrap property

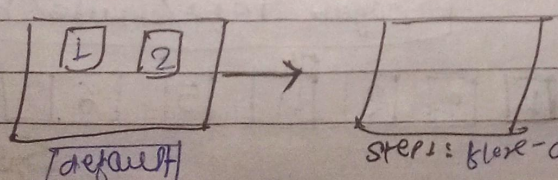
(a) row nowrap → if flex-flow: row-nowrap;

↓ output



step1: flex-direction ko row karke pr wrap mt karo.

(b) column nowrap →



step1: flex-direction ko column set karke pr wrap mt karo.

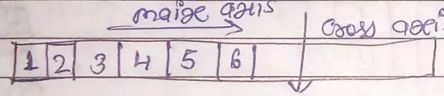
Spiral

(iv). Justify-content :-
ye flex items ko Align krta hai along/ex main axis.

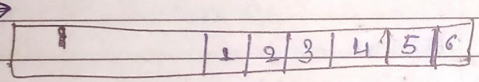
values :-

(a). Justify-content : flex-start ; (default).
(b). " : flex-end ;
(c). " : center ;
(d). " : space between ;
(e). " : space around ;
(f). " : space evenly ;

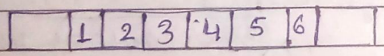
Ex:- (a) flex-start → main axis, cross axis.



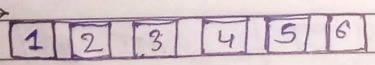
(b). flex-end →



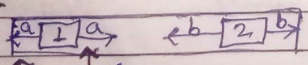
(c). center →



(d). space-between →



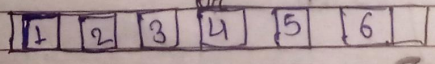
(e). space-around →



or

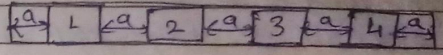
left se jitni jagah chhuti hogi utna hi box ke aage v jagah left/chhuti hogi.

or



Spiral

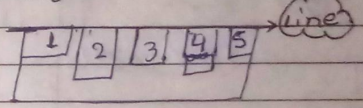
(f). space-evenly →



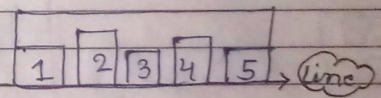
(v). Align-item :-
Is se hum cross-axis ki direction me flex-items ki alignment ko set kar sakte hai.

values :-

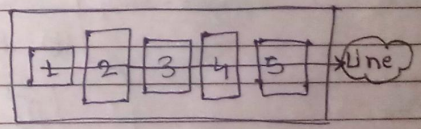
(a). flex-start → (by default).



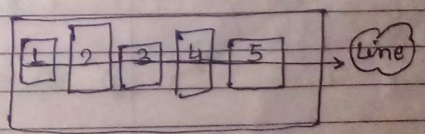
(b). flex-end →



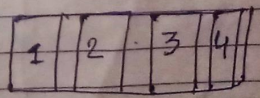
(c). center →



(d). baseline →



(e). stretch →



Spiral

flex-line decide hoti hai flex-direction se

Date

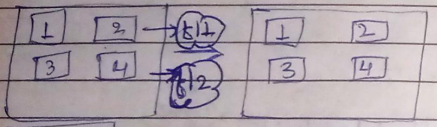
or cross-axis pe [illegible] kaise hai use align-items

(vi). align-content :-

ye flex-line pe kam karti hai ; flex line ko align karne ke liye align-content use karte hai. ye wrap karne baad ek se jyada flex-line ban jaati hai tb hum isko use karte hai.

values :-

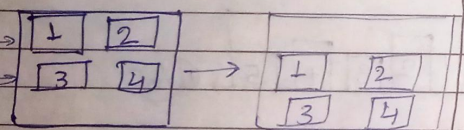
(a). flex-start →



default
flex-wrap

flex-start

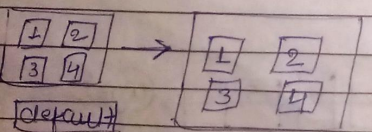
(b). flex-end →



default

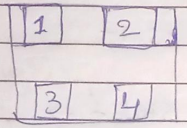
flex-end

(c). center →

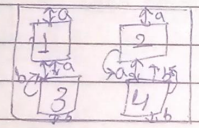


default

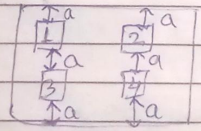
(d). space-between →



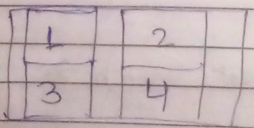
(e). space-around →



(f). space-evenly →



(g). stretch (by default) →



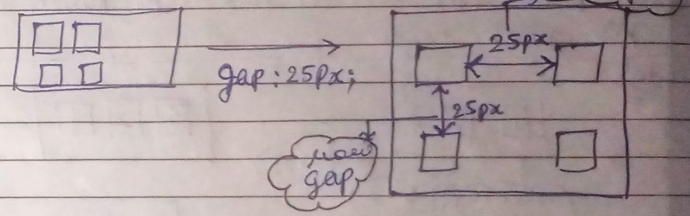
Date

written in margin also

(vii). (a) gap : 50px ; (row-gap / column gap) both applied.

(b) gap : 2px 5px ; (row gap, column gap)

eg:



77

direct child of flex-container automatically becomes flexible.

Date .....

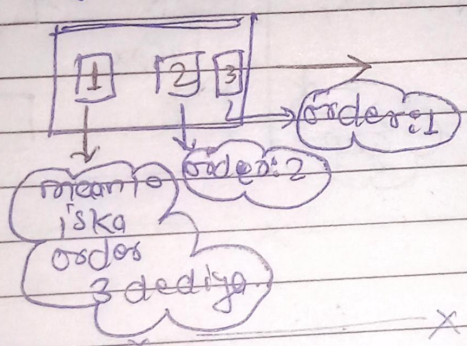
3. Flex- Child property :-

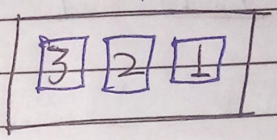
- flex child property se ham individual childs ko align ya usper ke V properties laga sakte hai. but unke parent me display: flex likha hona chahiye.

(i). order :- (default value = 0).

ye batata hai kon sa flex-item kahan hoga means kis jagah hoga.

eg:





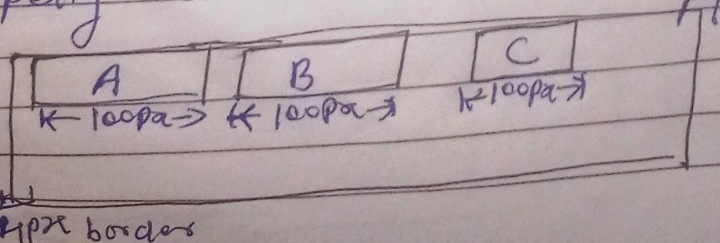
any number can we write till ∞.

(ii). flex-grow :- (default value = 0).

- ye batata hai ki, ek flex item kitna grow hoga/ bada hoga relative to rest of flex item.
  - ↳ baaki bache hue flex-items ke mutable.

eg:-

- maan lo ek container hai jiski size 1300px hai and unke to bacche hai unki 100 px hai sabki ke. to bacha hua 1000px kis kis bachhe me kitna batega wo ye grow property batata hai.



Spiral

Date .....

(border)

1300 - 8 px = 1292 px

- 300 px (all child width)

992 px → itni jagah khaali hai.

4

248 px → khaali jagah ko chaar jagah baat diye. to har ek hissa 248 aaya.

⇒ A + B + C

248 × 2.5 + 248 × 1.5 + 100 × 1 =

↑ chaar hisso me se 2.5 hissa isko milega.

↑ 1.5 hissa isko

⇒ 620 + 372 + 100

⇒ 620 + 100 (original size) + 372 + 100 + 100

⇒ 720 + 472 + 100

⇒ 1292

CSS:

```
A {
    grow : 2.5 ;
}
B {
    grow : 1.5 ;
}
C {
    grow : 1 ;
}
```

Date .....

(iii). flex-shrink → (default value = 1).

- How much a flex item shrink relative to rest of flex-items.

eg:

- Jab container sabhi flex-items ko shrink kr rahe ho container me fit karne ke liye.

And ham chahe ki koi particular item shrink na ho to isko flex-shrink: 0; kar sakte hai use.

eg:- [1 2 3 4 5 6] → [1 2 3 4 5 6]

default

↳ we set its flex-shrink as: 0.

↳ main axis me length ko pehle se de sakte hai.

(iv). Flex-basic →

Is se ham kisi flex item ki length pehle hi set kar sakte hai. yo kahin override kr dega agr pehle se sabke saath hi width diya hai to.

eg:- [1] [2] [3] [4] → [1] [2] [3] [4]

yahan sabhi ki length 100px set hai.

↳ Yahan hamne flex-basis: 200px set kiya hai.

div {
width: 100px
}

Date .....

(v). flex →

Is se flex-grow and flex-shrink and flex-basis ek saath use kar sakte hai.

eg:- flex: ___ ___ ___;

flex-grow | flex-shrink | flex-basis.

↳ auto (parent ka jo align-item rahega usi ko manega).

(vi). Align-self: (like align-item).

Is se kisi individual flex-item ko cross axis pr align kar sakte hai.

values:

(a) flex-start →

[1] [2] [3] → [1] [2] [3]

align-item: center lage hai container pr.

align-self: flex-start; lage hai [2] flex-item pr.

(b) flex-end.

(c) centre.

(d) baseline.

(e) stretch.

① Media Query :-

- Is se hum website ko responsive bana sakte hai.
- viewport ka width and height ko check kar sakte hai.
- Is se hum phone, tablets and desktop ke liye style kar sakte hai.

---

② # media types :-

	value	description
(i).	all	used for all media type devices
(ii).	print	used for print preview mode.
(iii).	screen	used for Computer screens, tablets, smart-phones etc.

media features :-

(i) orientation → orientation of the viewport. (landscape or portrait)

(ii). max-height → maximum height of viewport.

(iii). min-height → minimum height of viewport.

(iv). max-width → max. width of viewport

(v). min-width → min. " " ".

(vi). height → Height of viewport. (including scrollbar).

(vii). width → width of viewport (including scrollbar).

Date .....

Syntax :-

A media query consist media type and ek ya ek se jyada media feature ko contain kr sakta hai.

@media not/only mediatype and (media feature) and (media feature).

{

// css-code;

}

Notes (1). mediatype is optional, Agr nahi lagaye to all set krdega (means sabhi pr).

(2). Agr not/only use kr rahe hai to mediatype dena jaruri hai.

↳ not → ye poore media query ke meaning ko ulta kr deta hai, like not(!) operator who makes true as false and vice versa.

↳ only → ye poorane browser ko rokta hai media query ke andr likhe style ko chalane se, ye un browser ko rokta hai jo mediaquery ko support nahi karte.

↳ and → ye mediatype and ek ya ek se jyada media feature ko combine karta hai

Date .....

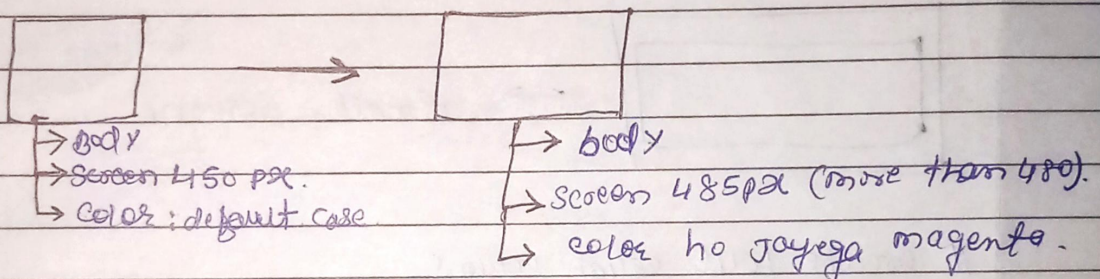
(3).

Implementations or examples :-

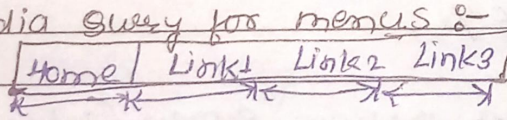
(1). @media screen and (min-width : 480px)

```
{
    body {
        bg color : magenta ;
    }
}
```

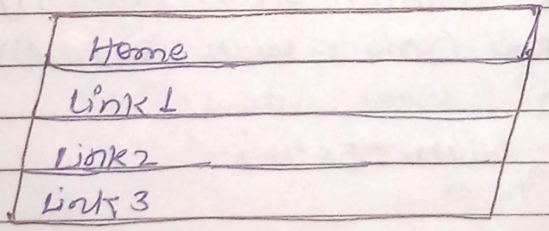
output :



(2). Media query for menus :-



→ Large screens



→ Small screens.

// container

```
.topnav {
    overflow : hidden ;
    bg color : #333 ;
}
```

// media query.

```
@media screen and (max-width : 600px) {
    .topnav a {
        float : none ;
        width : 100% ;
    }
}
```

// all links

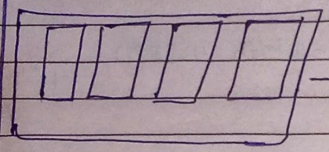
// Links (anchor).

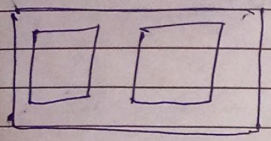
```
.topnav a {
    float : left ;
    display : block ;
    color : White ;
    text-align : center ;
    padding : 14px 16px ;
    text-decoration : none ;
}
```

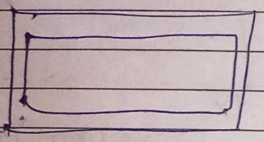
Spiral

Date .....

3. media Queries for columns :-


→ large Screens


→ medium Screens


→ Small Screens.

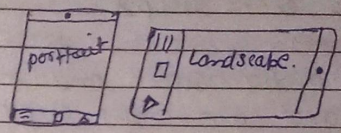
```
// create four equal columns.
.column {
    float: left;
    width : 25%;
}

// media query for medium screen (tablets).
@media screen and (max-width : 992px) {
    .column {
        width : 50%;
    }
}

// media query for small screen.
@media screen and (max-width : 600px) {
    .column {
        width : 100%;
    }
}
```

Spiral

Date .....

(4). # Orientation → Isk a do values possible hai:

(i). portrait
(ii) Landscape.

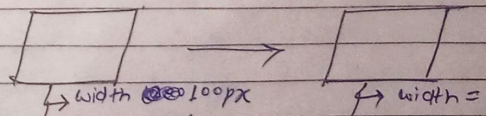



```
// media query for orientation.

@media only screen and (orientation : landscape)
{
    body {
        bg color : lightblue;
    }
}
```

(5). min-width and max-width both can apply at a time :-

```
// media query

@media screen and (max-width : 900px) and
(min-width : 600px ) {
    div {
        font-size: 50px;
    }
}
```

output



Spiral

Date .....

(6). // when width is b/w 600px and 900px or above 1100px.

```
@media screen and (max-width: 900px)
and (min-width: 600px), (min-width: 1100px) {
    div {
        fs : 50px
    }
}
```

* *

Date .....

(10) # Responsive web-design :-

(i) • RWD uses only html and CSS.
• RWD is not a program or a JavaScript.

(ii) # viewport :-

• viewport is the user's visible area of a web page.

• It varies with device and will be smaller on a mobile phone than on a Computer.

NOTE :- pehle web pages sirf desktop ke liye banaye jaate tha and static design hota tha. and fixed size hota tha.

Fir jab tablet and mobile se internet surf karne lage to Fixed size kaafi large tha phone ke viewport se. ① (automatically by browsers) :- Isko thik karne ke liye browsers ko scaled down kr dete the web pages ko taaki screen me fit ho sake.

ye perfect tarika nahi tha pr quick tha.

② using html
Second method to make web pages look good at different viewport (devices) :-

• Html5 introduced a method jis se web-designer control kar paaye viewport through, <meta> tag.

• Sabhi pages me <meta> tag viewport ko set karne ke liye use karna hoga.

↳ Syntax :-

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

↳ ye browser ko batayega ki kaise control kare page ke dimension ko.

- width=device-width → ye part, page ka width ko set karte hai°, screen ke width ko follow karte hue. (and screen ki° size change hoti rahegi according to device).

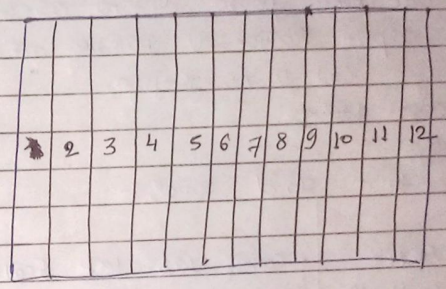
  initial-scale=1.0 → ye part, starting me set karta hai° ki° kitna zoom rahega screen. minimum (0.1 to 10 value).

Note (i). Large fixed width elements ko use mt karo. eg. agr image ki° width viewport se jyada hui° to horizontally scroll karna padega and is se poor experience hoga.

(ii). Media query ko use karo alag alag screen pr style karne ke liye.

(iii) Grid view :-

- Grid view me poora viewport 12 columns me divided hota hai° and total width 100% hota hai°. and ye expand or shrink hoga jaise jaise hum resize karenge browser window.



(iv) RWD images :-

width : 100% hai and height : auto hai tab image responsive ho jayega pr kabhi kabhi image ki° size 100% se jyada scale-up ho jayega. Isliye max-width use karna acha hota hai°, is se image scale down to hoga pr apne original size se jyada scale-up nahi hoga.

↳ ① Background images →

(i) Background-size : contain. ye scale karega and try karega content area me fit apne ko. pr image apna aspect ratio rakhega.

↳ aspect ration means width ke mukable height kitna hona chahiye. 2:3

↳ height is 1.5 times width. to ye scale hoga to is tarah hoga. ki° width 3 hua to height 4.5 time 3 hoga.

Date .....

(2). background-size: 100% 100%;
Is se bg img stretch hoga and pora div or content area cover karega.

(3). background-size: cover;
Isse bg img scale hoke pora content area fill karega and aspect ration v maintain kis se thod bahut image cut v ho jayega.

diff b/w contain and cover:-

contain	cover
• contain aspect ratio ko utna hi badhata hai jitne me height ya width apne corner ko touch na karde. Isiliye contain kehte hai.	• cover tab tak scale karta hai jab tak width and height dono na touch ho jaye. eg:- agar 16:9 ratio 2:3 ka hai, tab jab scale hoga to height touch ho jayega par width nahi par fir v width ko badhayega jisk wajah se height ko v badhana hoga and kuch height image ko cut jayega kyuki touch to pehle hi ho gaya tha.

(2) ↳ min-width ki jagah min-device-width use kar sakte hai; isse sidha device (phone, tablet) ki width check karega na ki browser ke viewport ki width.

Date .....

fir jab ham browser ko resize karenge tab image change nahi hoga. kyuki ye sirf device ki width par hi hoga.

↳ (3). <picture> ko use kar sakte hai responsive ke liye. ye <audio> and <video> ki tarah hota hai.

eg:- <picture>
<source srcset="img_smallflower.jpg" media="(max-width: 400px)">
(media → optional; srcset → required)
<source srcset="img_flowers.jpg">
<img src="img_flowers.jpg" alt="Flowers">
↳ ye iss liye agr kisi browser me <picture> na support kare.
</picture>

(5) RWD video :-
video {
width: 100%; → isiki jagah max-width use karo. Same reason as image.
height: auto;
}