CSS SELECTORS

css selectors are used to select element so that we can style them.

- `.example` // selects all elements with example class
- `#id` // selects the element with id = "id"
- `h1` // selects all h1 element
- `P.class` // selects all p elements with class = "class"
- `div, p` // selects all div and p elements
- `div > h2` // selects all h2 element whose parent is div
- `p ~ ul` // selects all ul that are preceded by P.

- `[target]` = selects every element with target attribute
- `[target=_parent]` // selects every element with att target = "_parent"
- `[title~=Pratham]` // selects every element with title att. containing word "Pratham"
- `[href^="https"]` // selects every element whose href starting with https
- `[href$=".png"]` // ends with .png

- `:not(h1)` // selects every element that is not h1
- `:root` // Selects the document root's elements

- `P:nth-child(2)` // second child of its parent
- `P:nth-of-typ(2)` // selects every p element i.e, second p element of its parent
- `P:only-child` // selects p thats only child

CSS PSEUDO CLASSES.

Selector	Example	Description.
:active	a:active	selects the active link
:checked	input:checked	selects every checked input element.
:enabled	input:enabled	selects every enabled input element
:empty	p:empty	selects every p elements that has no children.
:first-child	p:first-child	selects every p elements that is the first child of its parent.
:first-of-type	p:first-of-type	selects every p element that is the first p element of its parent.
:focus	input:focus	selects the input element that has focus.
:hover	a:hover	selects a on mouse over.
:in-range	input:in-range	selects input elements with a value within a specified range.
:not(selector)	:not(p)	selects all element ~~ex~~ except p.
:nth-child	p:nth-child(2)	selects every p elements that is second child of ils parent.
:only-of-type	p:only-of-type	selects every p elements that is the only p element of its parent.
:optional	input:optional	selects input element with no required attribute.

CSS Units

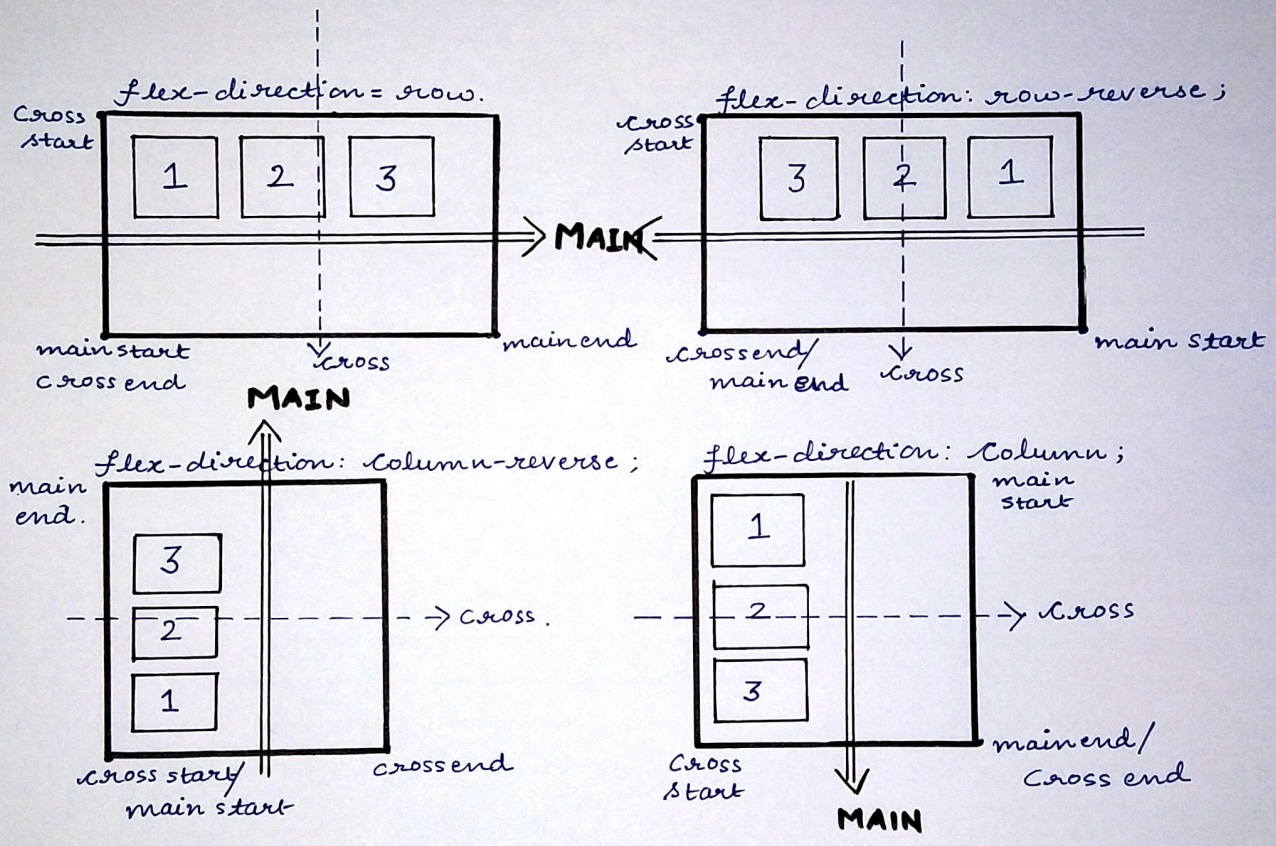
Unit	Name	Equivalent
px	pixel	1 px = 1/96th of 1 in
cm	Centimeters	1 cm = 38 px
mm	millimeters	1 mm = $\frac{1}{10}$ cm
Q	Quarter-millimeter	1 Q = $\frac{1}{40}$ cm
in	Inches	1 in = 2.54 cm
pc	Picas	1 pc = $\frac{1}{6}$ in
pt	Points	1 pt = $\frac{1}{72}$ in.

Unit	Description
em	Font size of parent, in the case of typographical prop like font-size, and font size of the element itself, in the case of other properties.
ex	X-height of the element's font.
ch	The advance measure (width) of the glyph "0" of element's font.
rem	Font size of the root element.
lh	line height of the element.
vw	viewport's width
vh	viewport's height
vmin	1% of the viewport's smaller dimensions.
vmax	1% of the viewport's larger dimension.

TWO AXES OF FLEXBOX.

Main axis: Defined by flex direction.

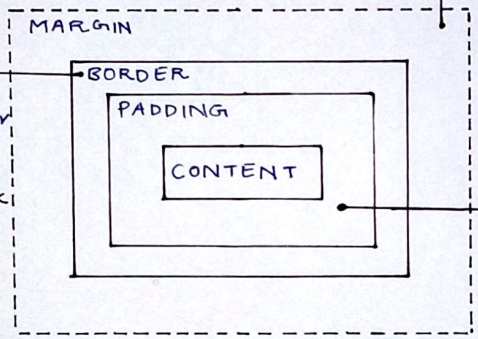
Cross axis: Runs perpendicular to main axis.



CSS BOX-MODEL

- Everything in CSS is a box or rather everything in HTML is a box-model which is surrounded by 4 different box virtually.

1. Content (original content inside element)
2. Padding (create space b/w content and element's border)
3. Border (create border around element)
4. Margin (space between element)



You can create border around element by specifying the width, color and style.

for ex: border: 1px solid black;

Solid:

dotted :

dashed :

double :

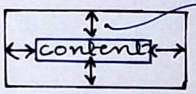
Margin defined the space between element.
for ex. margin: 10px;
It will create 10px empty space around element in all direction.

margin: 25px 5px 6px 10px;
↓ top ↓ right ↓ bottom ↓ left

margin: 25px 10px 25px
// Top, right and left, bottom

margin: 25px, 10px; and
// Top, bottom; right, left

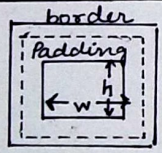
Padding allows you to create space between content and element's boundary for ex.



padding: 10px;

// similarly as margin, you can pass four, three, two or one value in padding as well.

you can write padding: 10%;
%- specify a padding or margin in % of the containing element.



If you add width as 100px, padding as 10px and border as 2px then the entire width becomes 112px (100+10+2).

Box-sizing: border-box;

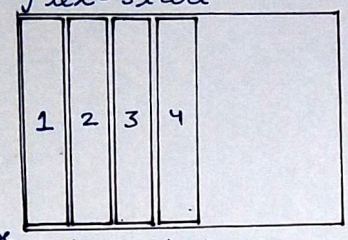
The box-sizing property defines how the width and height of an element are calculated:
if we apply box-sizing: border-box then the padding and border will be adjusted in the width and height of an element.

justify content

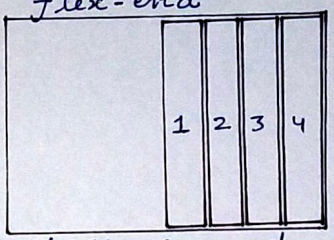
justify content property align the flex items horizontally within the flex container.

// In space-around items are evenly distributed in the line with half size spaces on either end.

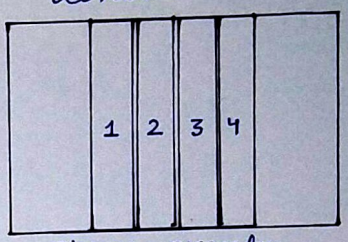
flex-start



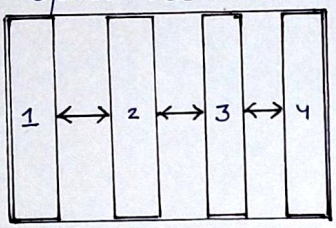
flex-end



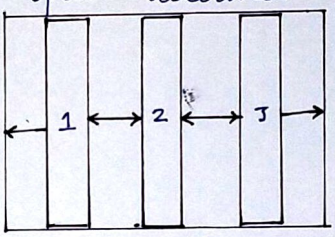
center



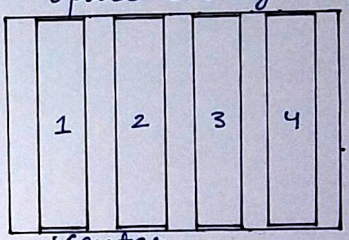
space-between



space-around



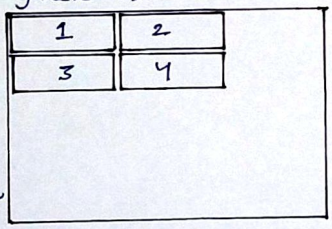
space-evenly



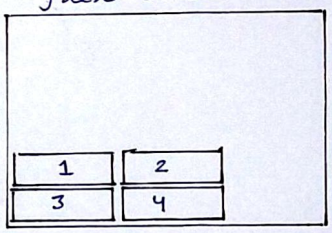
align-content

align-content property align the flex items vertically within the flex container.

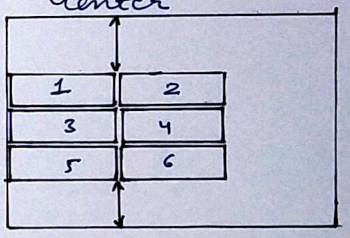
flex-start



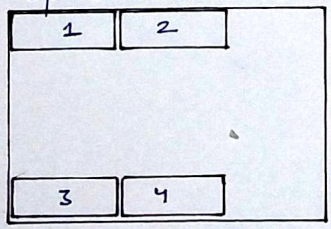
flex-end



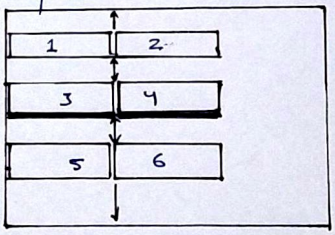
center



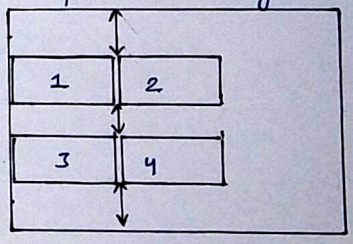
space-between



Space-around



space-evenly.



Grid Overview

```
.one {
  grid-column: 1/4;
}
```

```
grid-template-column:
  1fr 2fr 1fr
```

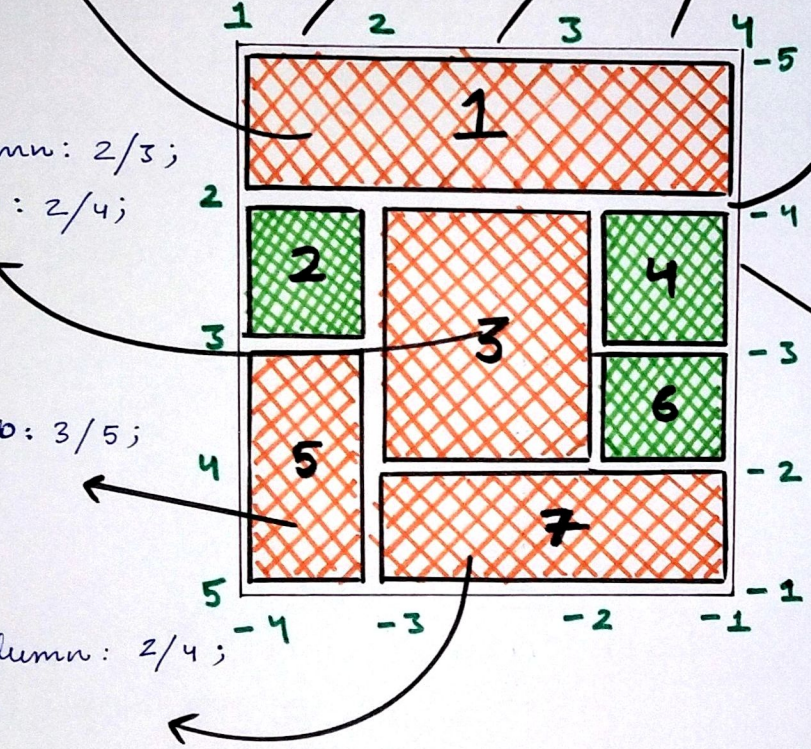
```
.three {
  grid-column: 2/3;
  grid-row: 2/4;
}
```

```
grid-gap: 10px;
```

```
.five {
  grid-row: 3/5;
}
```

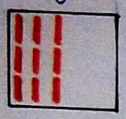
```
.seven {
  grid-column: 2/4;
}
```

```
.parent {
  justify-content:
    center;
  align-content:
    center
}
```



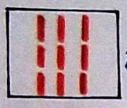
Justify-items

align content in a grid item along row

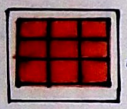


justify-items: start;

justify-items: end;



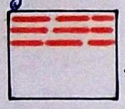
justify-items: center;



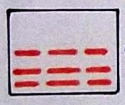
justify-items: stretch;

Align-items

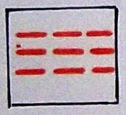
align-content in a grid item along column axis



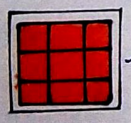
align-items: start;



align-items: end;

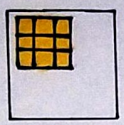


align-items: center;

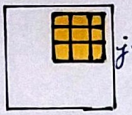


align-items: stretch;

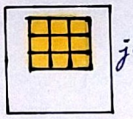
Justify-content



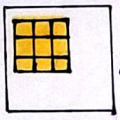
justify-content: start;



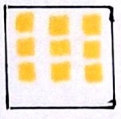
justify-content: end;



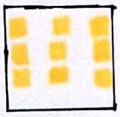
justify-content: center;



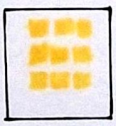
justify-content: stretch;



justify-content: space-around;



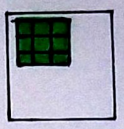
justify-content: space-between;



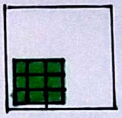
justify-content: space-evenly;

justifies all grid content on row axis when total grid size is smaller than container

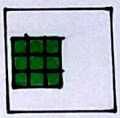
Align-content



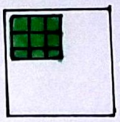
align-content: start;



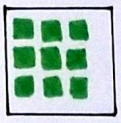
align-content: end;



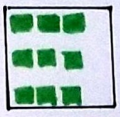
align-content: center;



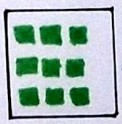
align-content: stretch;



align-content: space-around;



align-content: space-between;



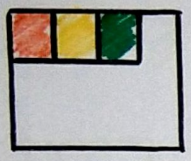
align-content: space-evenly;

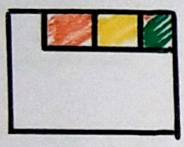
Justifies all grid content on column when total grid size is smaller than container.

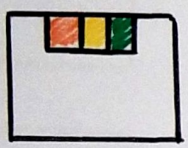
Pratham♥

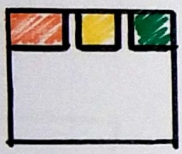
Alignment in CSS

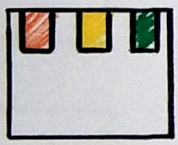
justify-content

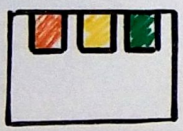

flex-start


flex-end


Center

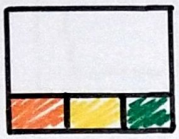

space-between


space-around

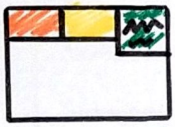

space-evenly.

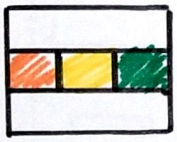
align-items


flex-start

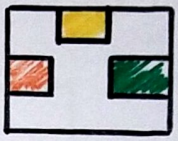

flex-end

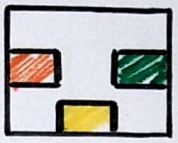

Stretch

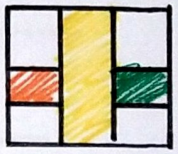

baseline

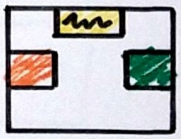

center

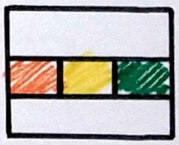
align-self


flex-start


flex-end


Stretch

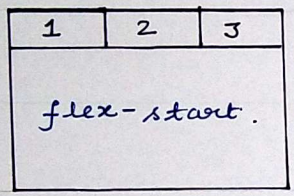

baseline

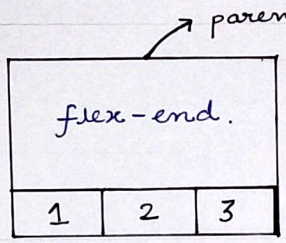

center

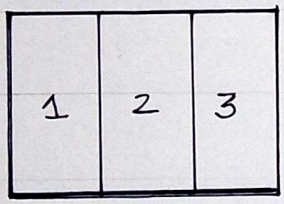
* align-self is applid on yellow item.

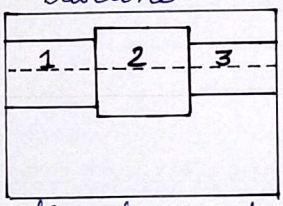
Align-items.

Specifies the default alignment for items inside the flexible container.

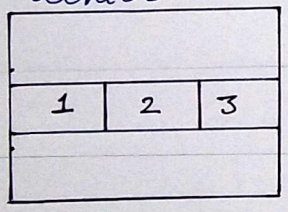


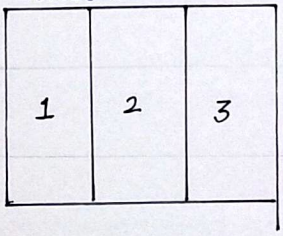






align along content's baseline

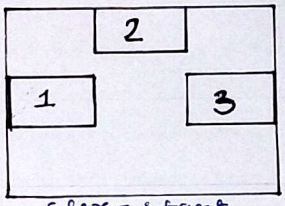




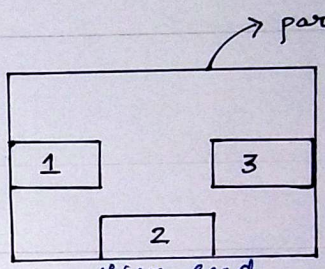
Align-self.

Specifies the alignment of selected items inside flexible container

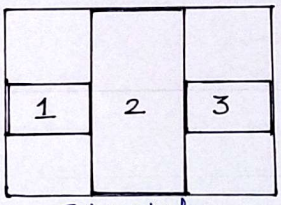
```
.two {
    align-self: ;
{
```



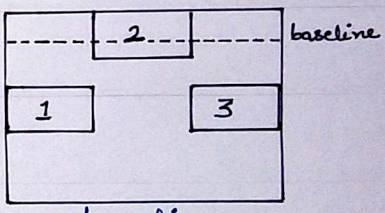
flex-start



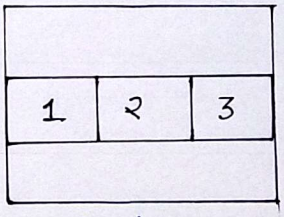
flex-end



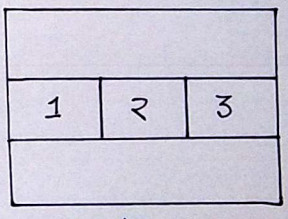
Stretch



baseline.



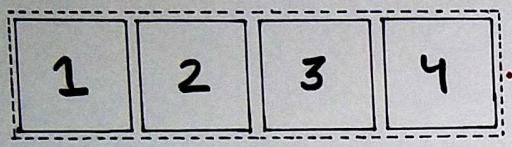
center.



auto
inherit align-items.

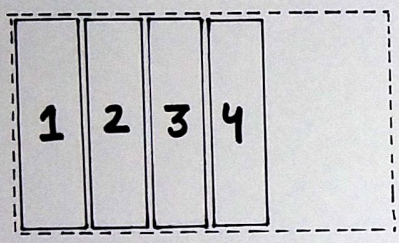
FLEX CHEAT SHEET

- CSS flexible box layout.
- Commonly known as ~~flex~~ flexbox.



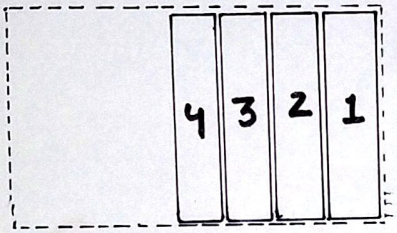

```
.container {
    display: flex;
}
```

Flex-direction. This property specifies how flex-item are placed in the flex-container.



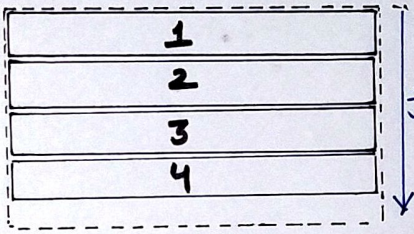
flex-direction: row;

in a row from left hand side



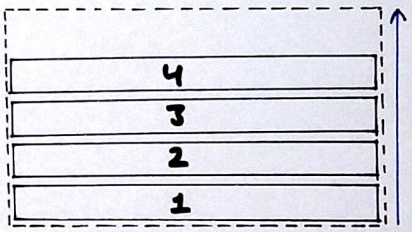
flex-direction: row-reverse

in a row but from right hand side



flex-direction: Column;

in column from top.



in a column from bottom.

flex-direction: Column-reverse

* row is a default value for flex-direction

justify content

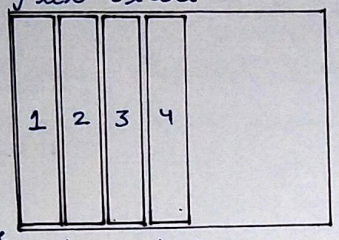
justify content property align the flex items horizontally within the flex container.

// In space-around items are evenly distributed in the line with half size spaces on either end.

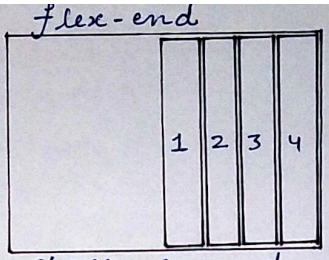
align-content

align-content property align the flex items vertically within the flex container.

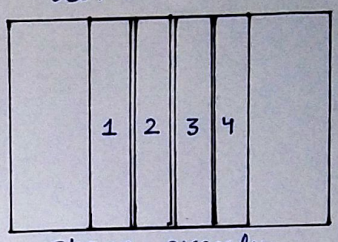
flex-start



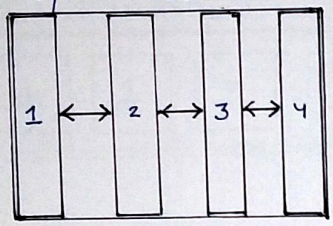
flex-end



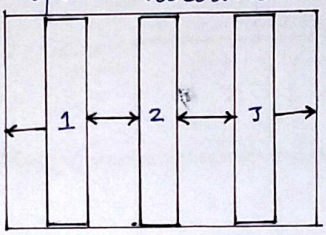
center



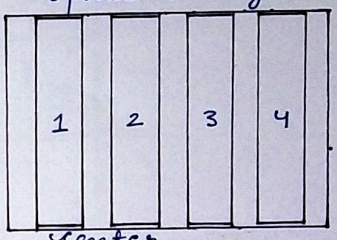
space-between



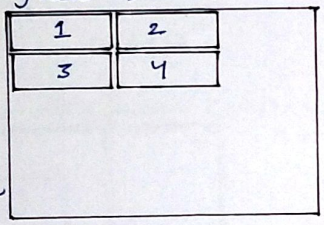
space-around



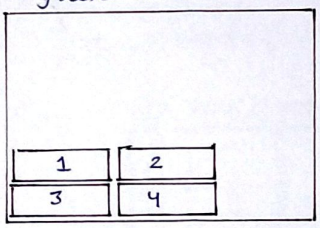
space-evenly



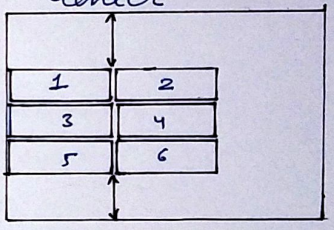
flex-start



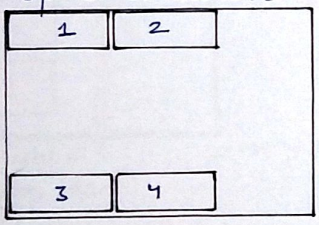
flex-end



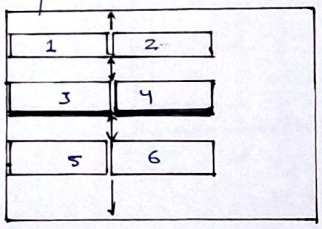
center



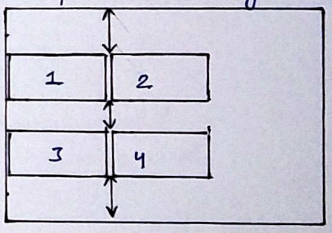
space-between



Space-around

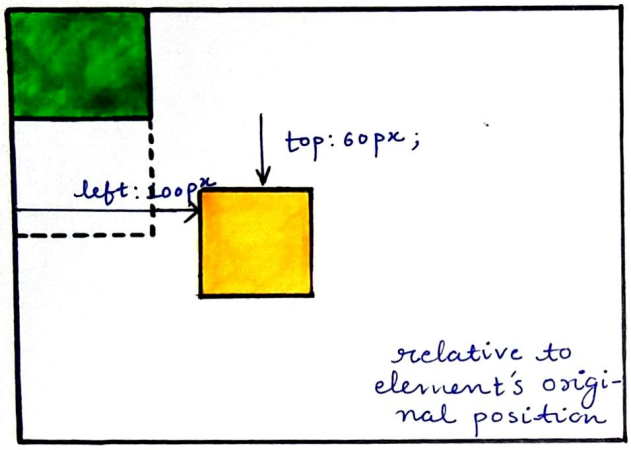


space-evenly.

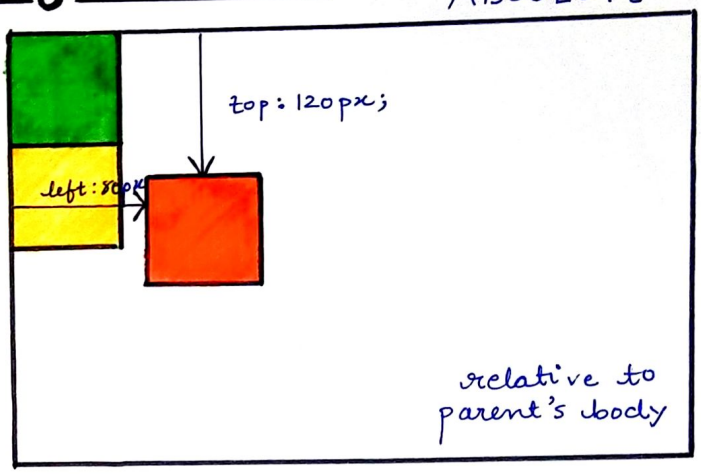


Positioning in Css

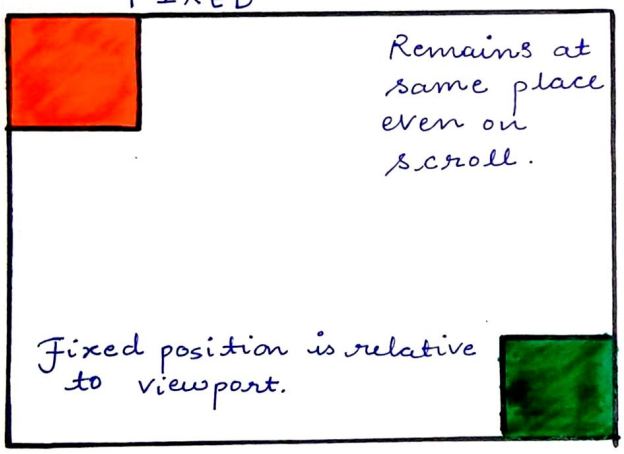
RELATIVE



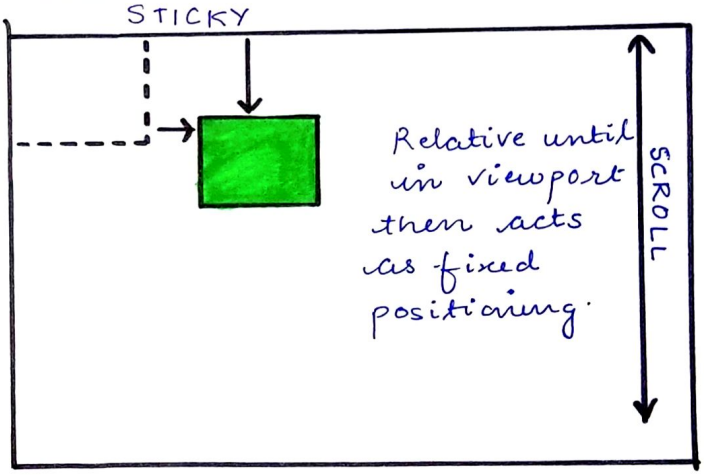
ABSOLUTE



FIXED

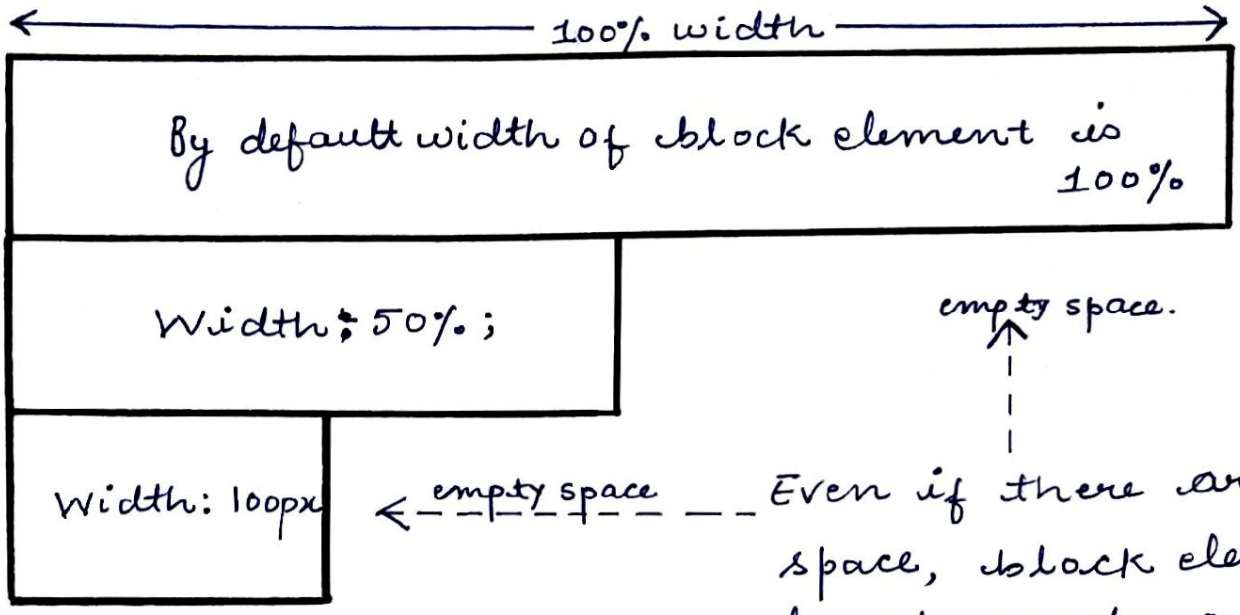


STICKY



Display : Block, inline and inline-block.

Block Elements.



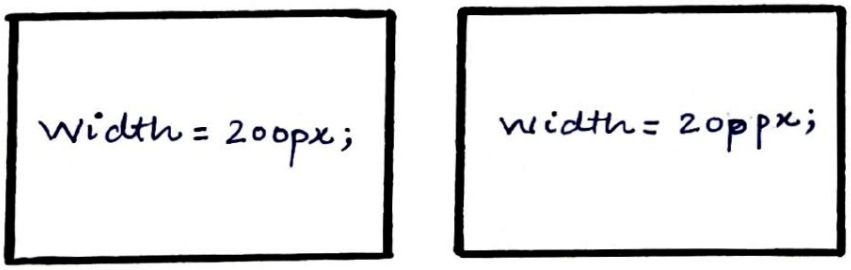
Even if there are space, block element do not render on a same line. In general they do not sit next to each other.

Inline

You can't control width and height of inline element. for ex. span.

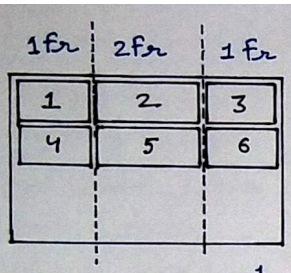
Inline-block.

(As the term suggest, they are combination of block and inline)

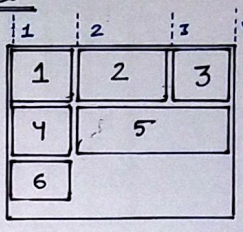


The element is formatted as inline element but you can apply height and width as well.

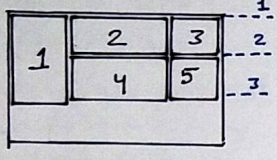
Grid cheat sheet.



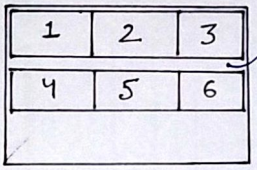

```
grid-gap : 10px;
grid- template-column : 1fr 2fr 1fr;
grid-gap : 10px;
display : grid ;
```



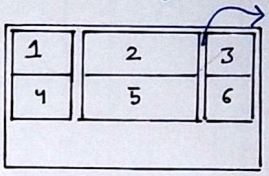

```
. five {
    grid-column : 2/4;
}
// fifth element start-
from 2nd column and
ends at 4th colum
```



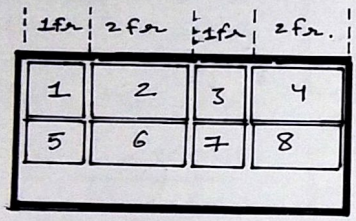

```
. one {
    grid-row: 1/3;
}. // row gap
```



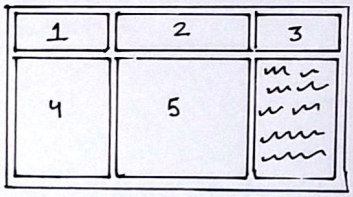

```
grid-row-gap : 10px
```



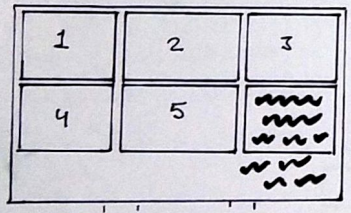

```
grid-column-gap : 10px;
// column gaps.
// we can use
grid-gap. It will
add equal space
b/w rows and
column.
```



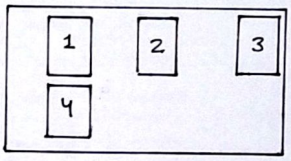

```
grid-template-column :
    repeat(2, 1fr 2fr);
// It will repeat
column 2 times as
1 fraction an two fraction
```



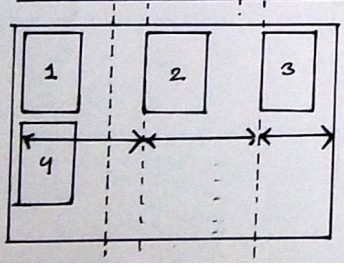

```
grid-auto-column :
    minmax(100px, auto);
// min height 100px and
max will ube auto.
```



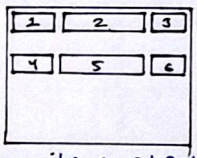

```
. parent {
    grid-auto-row : 100px;
}
// every item has 100px
height. Due to which
content overflows. In
order to prevent it.
```




```
. parent { justify-items : end; }
// it will take center value
as well. which will align
the items center horizontally.
```

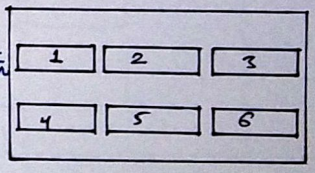



```
. parent {
    justify-items: start;
}
// it will align
horizontally.
```

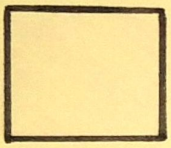



```
align-items: start
// It will align vertically.
```

```
{ align-items : end}
// align-items : center
will align all grid
items at center.
at vertically.
```

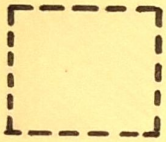


border-style



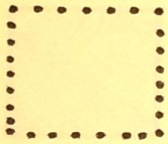
Solid

Displays a single straight solid line



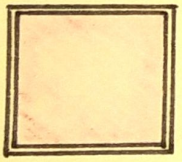
Dashed

Displays the small square of same length.



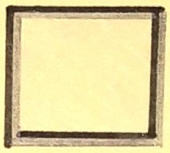
Dotted

Displays a series of rounded dots.



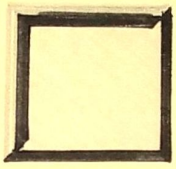
Double

Displays two straight line



Groove

Displays the border with carved appearance.

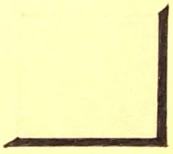


Ridge

Displays the border with an extruded appearance

Inset

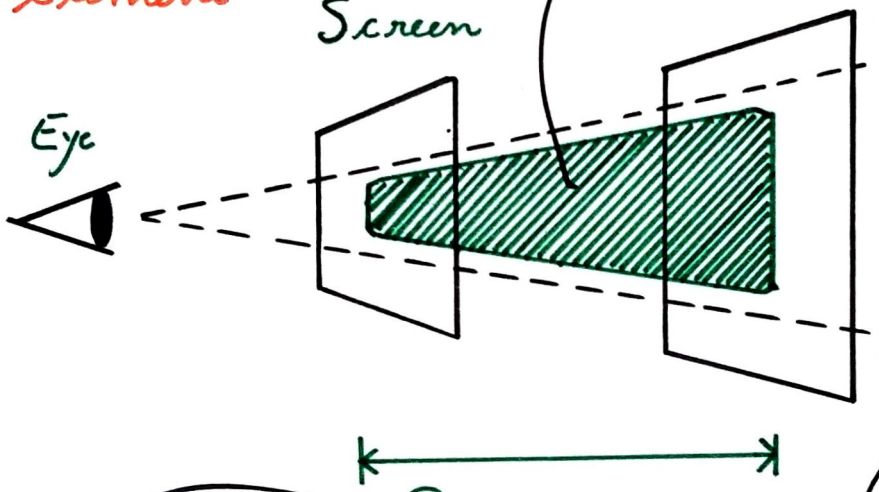
makes the element appear embedded.



Outset

Displays a border that makes element appear embossed.

The "perspective" is used to give 3D look to the element



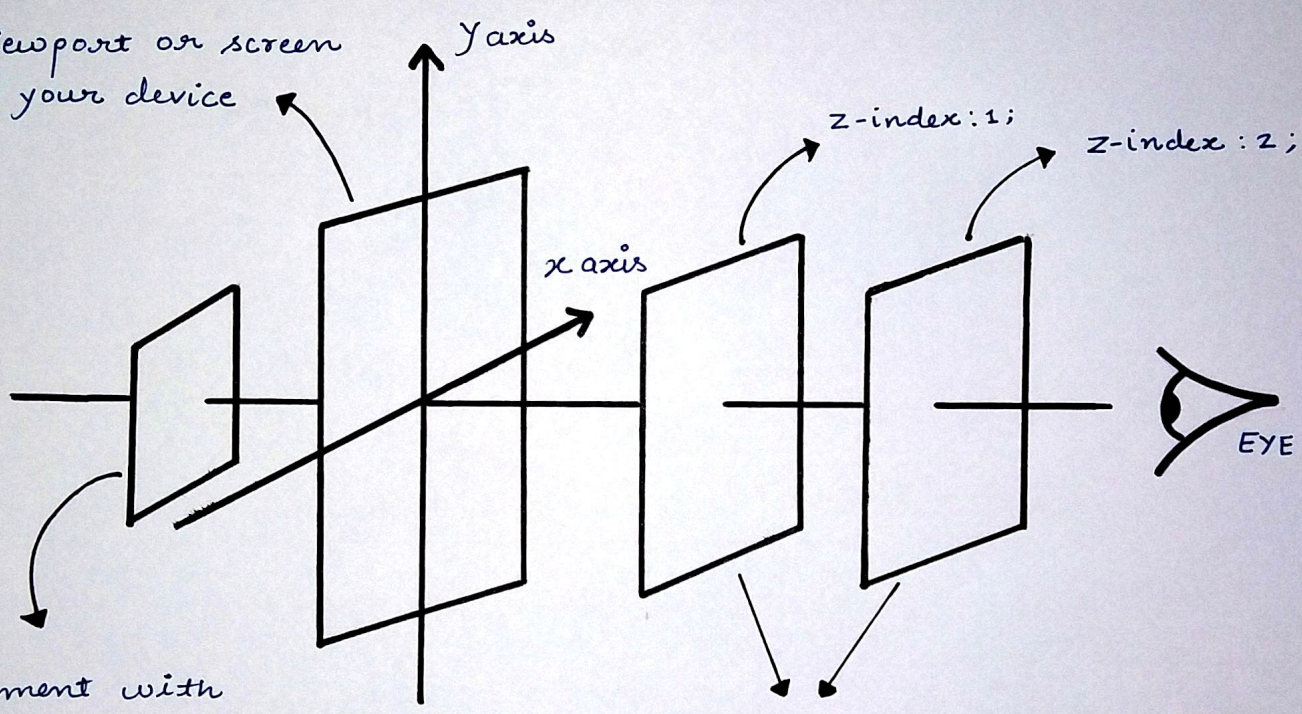

```
.element {
  transform:
  rotateY(40deg);
}
```

Perspective: 500px;

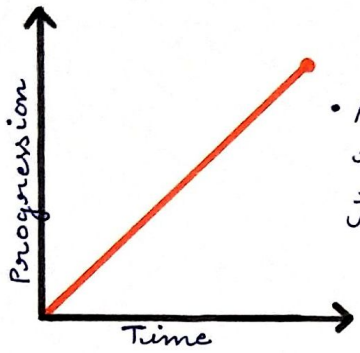
Lower value will create more intensive 3D look

The higher the value the, lower the 3D look

Z-INDEX

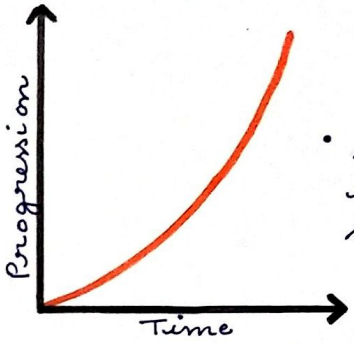

Viewport or screen of your device
Y axis
z-index: 1;
z-index: 2;
x axis
EYE
Element with negative z-index
Let's say
z-index: -1;
Elements with positive z-index.

Linear



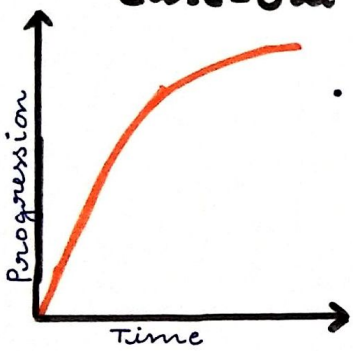
- Animation has the same speed from start to end.

ease-in



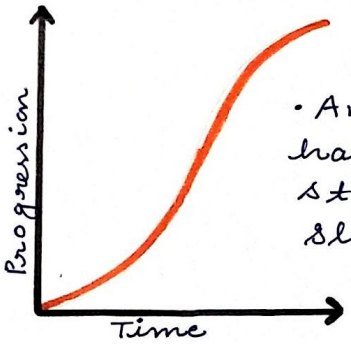
- Animation has the slow start

ease-out



- The animation has a slow end.

ease-in-out



- Animation has slow start and a slow end.