

## 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 elements

p: nth-child(2) // second child of its parent

p: nth-of-type(2) // selects every p element i.e, second p element of its parent

p: only-child // selects p that's only child

## CSS PSEUDO CLASSES.

<u>Selector</u>	<u>Example</u>	<u>Description</u>
: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 <del>except</del> except p.
:nth-child	p: nth-child(2)	Selects every p elements that is second child of its 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.

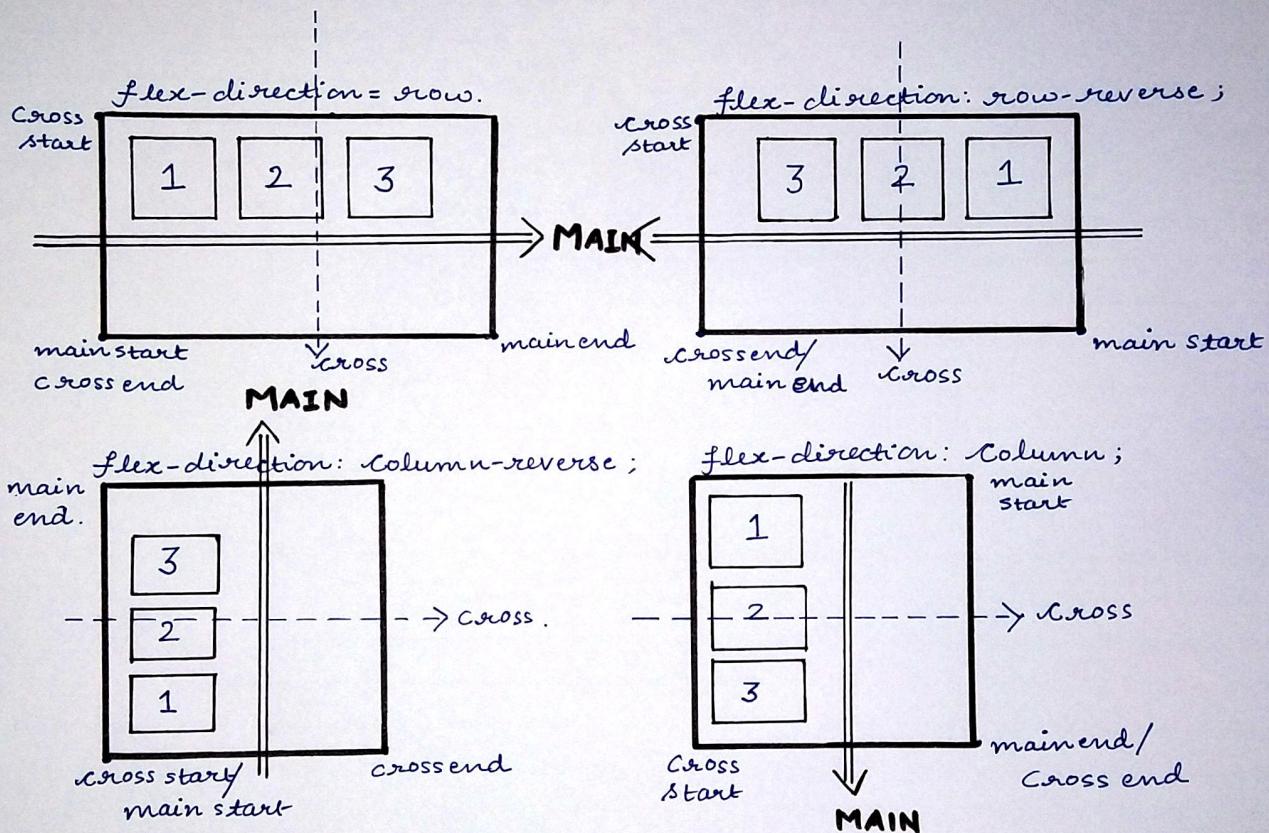
## CSS Units

unit	Name	Equivalent	unit	Description
px	pixel	$1\text{px} = \frac{1}{96}\text{th of 1 in}$	em	Font size of parent, in the case of typographical properties like font-size, and font size of the element itself, in the case of other properties.
cm	Centimeters	$1\text{cm} = 38\text{ px}$	ex	x-height of the element's font.
mm	millimeters	$1\text{mm} = \frac{1}{10}\text{ cm}$	ch	The advance measure (width) of the glyph "o" of element's font.
Q	Quarter-millimeter	$1\text{Q} = \frac{1}{40}\text{ cm}$	rem	Font size of the root element.
in	Inches	$1\text{in} = 2.54\text{ cm}$	lh	line height of the element.
pc	Picas	$1\text{pc} = \frac{1}{6}\text{ in}$	vw	viewport's width
pt	Points	$1\text{pt} = \frac{1}{72}\text{ in.}$	vh	viewport's height
			vmin	$1\%$ of the viewport's smaller dimensions.
			vmax	$1\%$ of the viewport's larger dimension.

## TWO AXES OF FLEX BOX.

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 <sup>content</sup> and element's border)
  3. Border (create border around element)
  4. Margin (space between element)

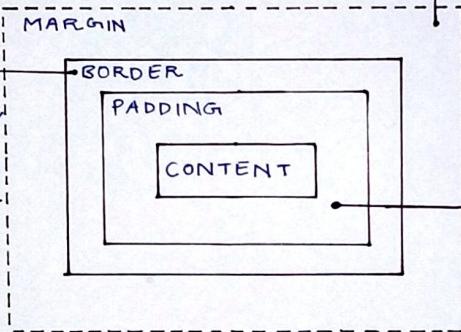
You can create border around element by specifying the width, color and style.

for ex: border: 1px solid black

dotted:

dashed:

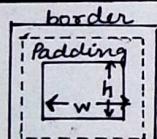
double:



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.

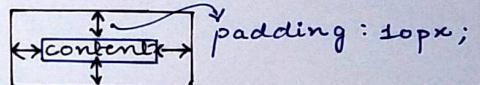


→ Margin defined the space between element.  
for ex. margin: 10px;  
It will create 10px empty space around element in all direction.

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

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

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



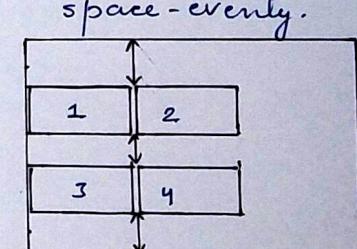
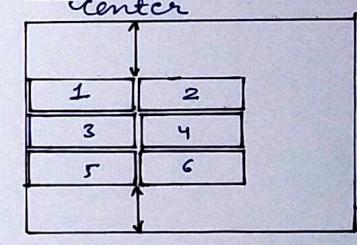
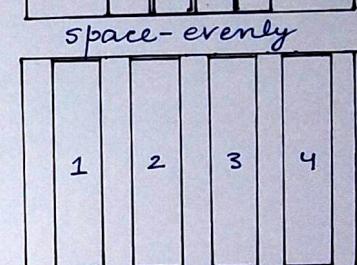
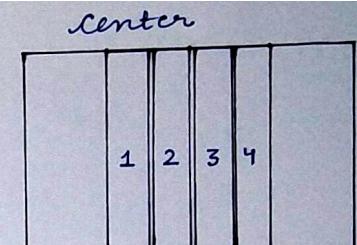
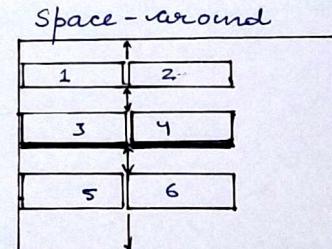
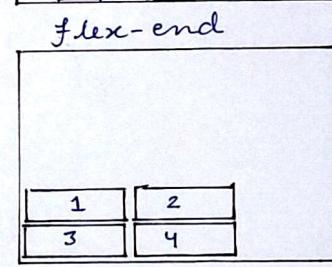
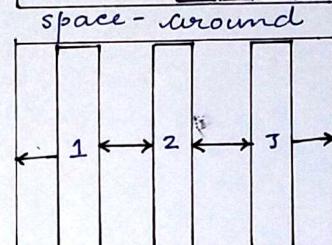
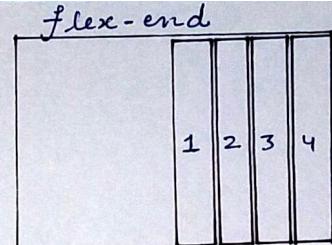
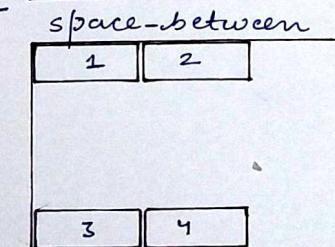
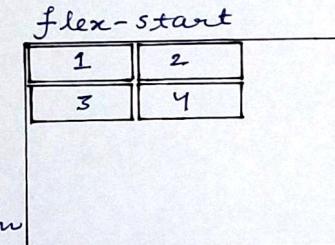
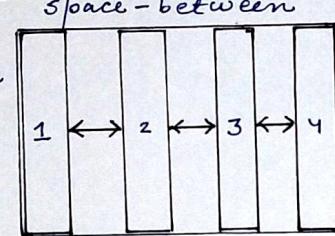
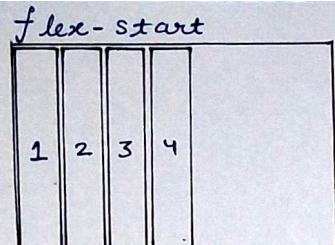
// similarly as margin, you can pass four, three, two or one value in padding as well.

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

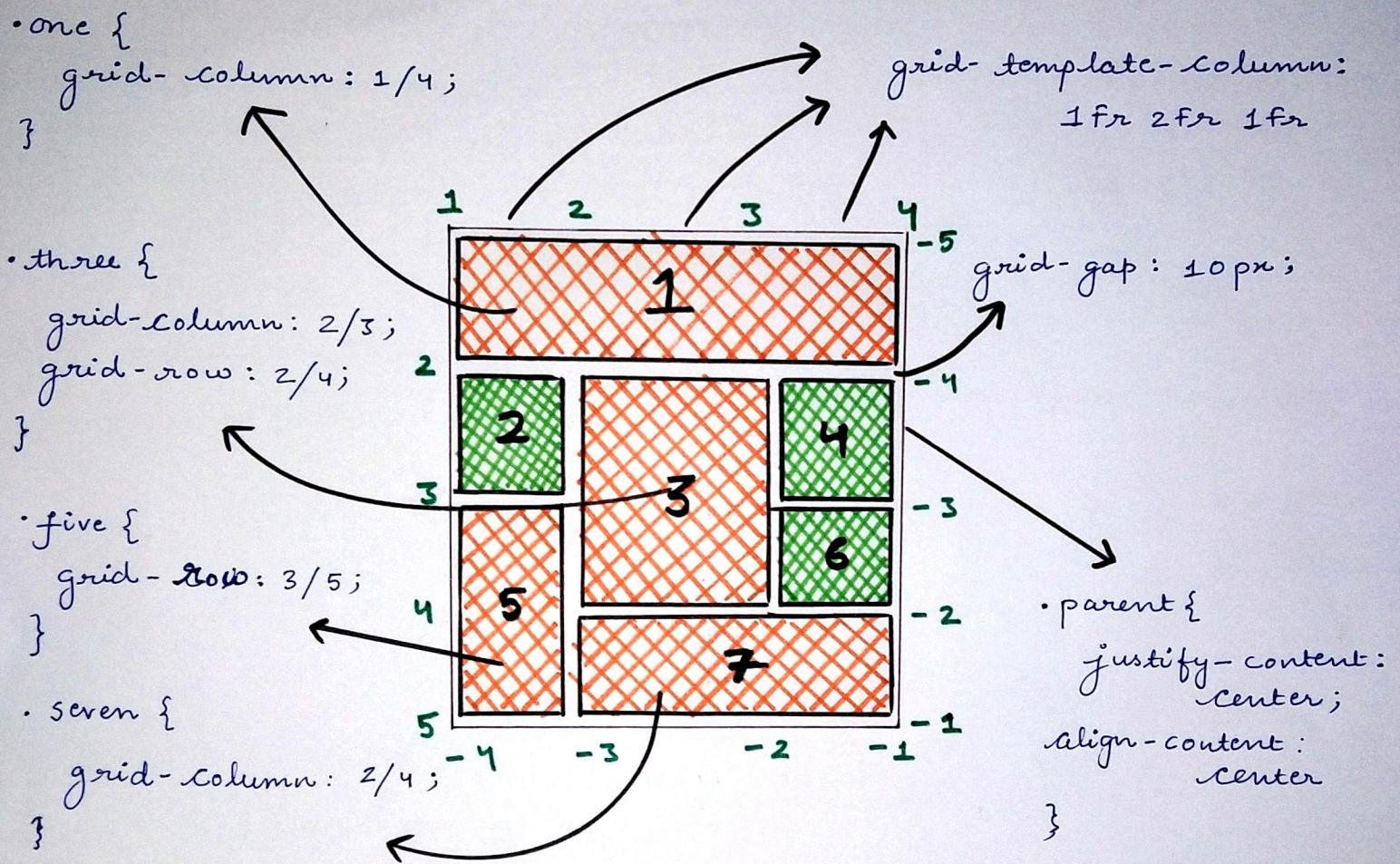
justify-content  
justify-content property align the flex items horizontally within the flex container.

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

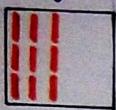
align-content  
align-content property align the flex items vertically within the flex container.



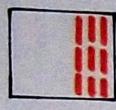
# Grid Overview



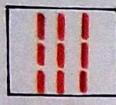
### Justify-items



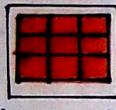
justify-items: start;



justify-items: end;

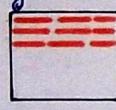


justify-items: center;

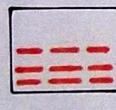


justify-items: stretch;

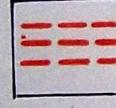
### Align-items



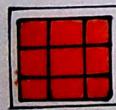
align-items: start;



align-items: end;

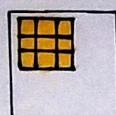


align-items: center;

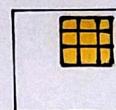


align-items: stretch;

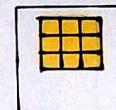
### Justify-Content



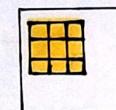
justify-content: start;



justify-content: end;



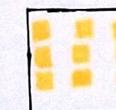
justify-content: center;



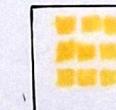
justify-content: stretch;



justify-content: space-around;

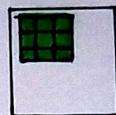


justify-content: space-between;

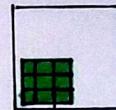


justify-content: space-evenly;

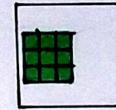
justifies all grid content on row axis when total grid size is smaller than container



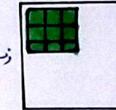
align-content: start;



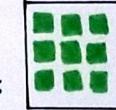
align-content: end;



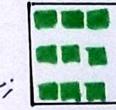
align-content: center;



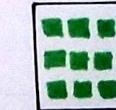
align-content: stretch;



align-content: space-around;



align-content: space-between;

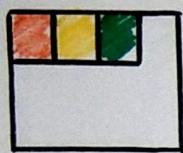


align-content: space-evenly;

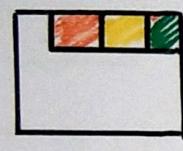
Justifies all grid content on column when total grid size is smaller than container.

## Alignment in CSS

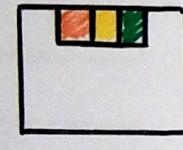
**justify-content**



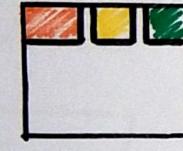
flex-start



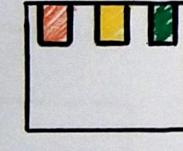
flex-end



center



space-between

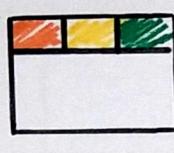


Space-around

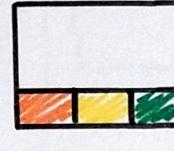


space-evenly

**align-items**



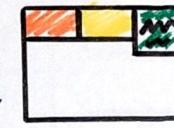
flex-start



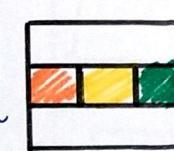
flex-end



stretch

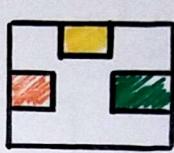


baseline

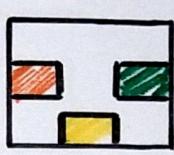


center

**align-self**



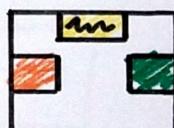
flex-start



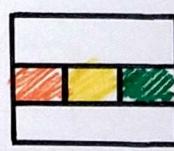
flex-end



stretch



baseline

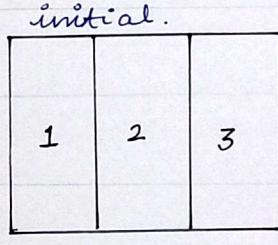
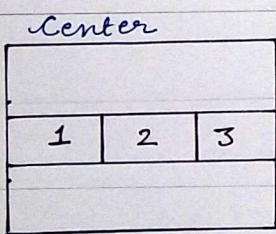
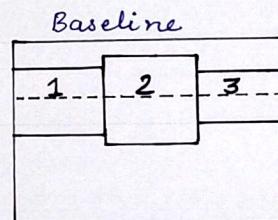
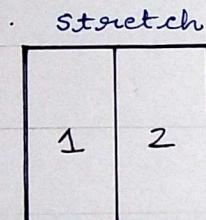
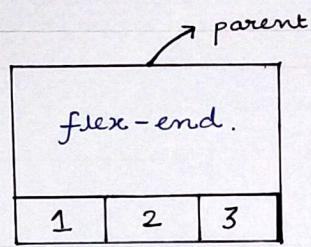
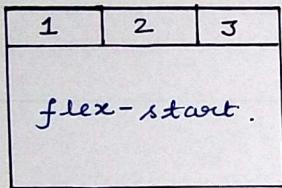


center

\* align-self is applied on yellow item.

### Align-items.

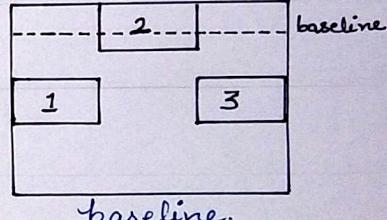
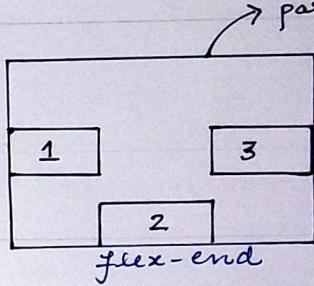
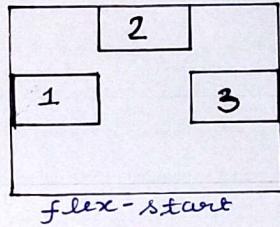
specifies the default alignment for items inside the flexible container.



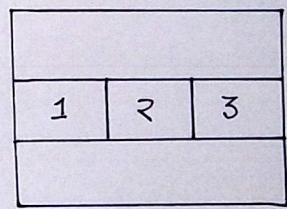
### Align-self.

specifies the alignment of selected items inside flexible container

.two {  
  align-self: ;  
}

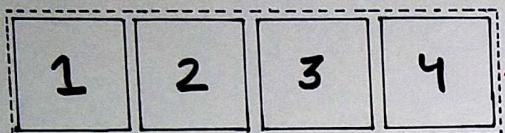


center.



auto  
inherit align-items.

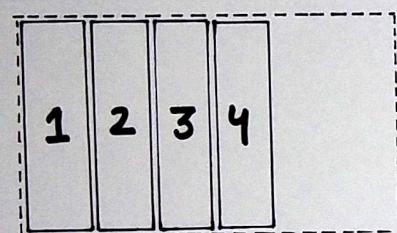
## FLEX CHEAT SHEET



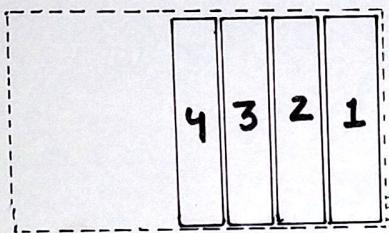
```
.Container {  
    display: flex;  
}
```

• CSS flexible box layout.  
• Commonly known as **flexbox**.

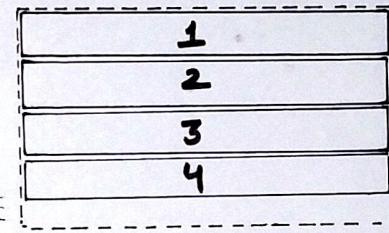
Flex-direction: This property specifies how flex-items are placed in the flex-container.



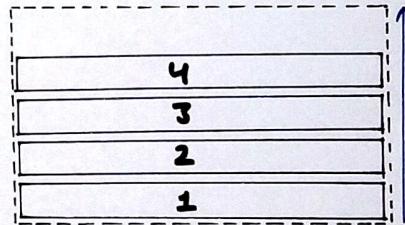
flex-direction: row;  
in a row from  
left hand side



flex-direction: row-reverse  
in a row but from  
right hand side



flex-direction:  
Column;  
in column  
from top.



in a column  
from bottom.

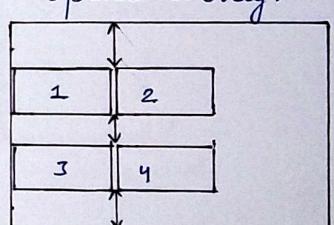
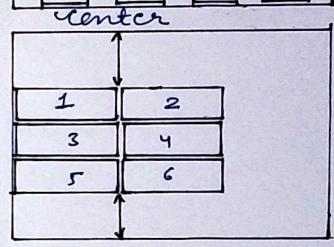
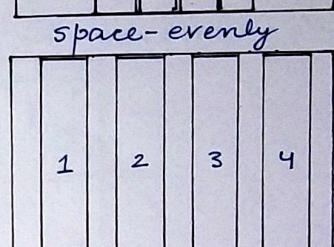
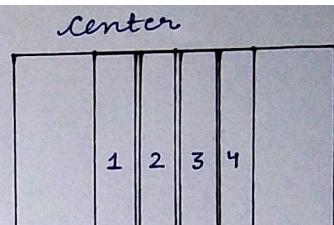
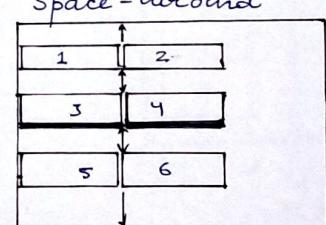
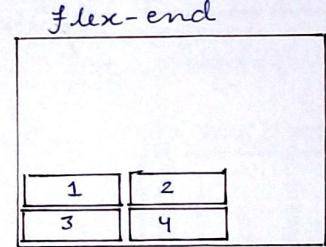
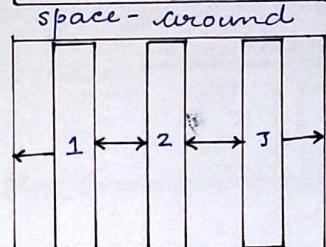
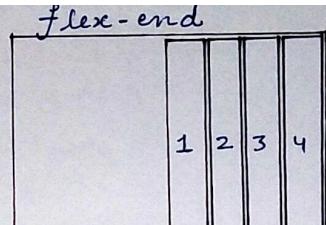
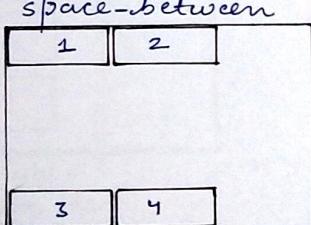
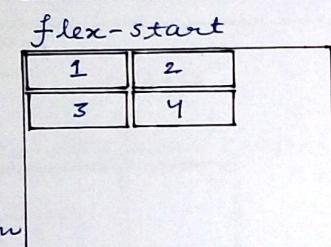
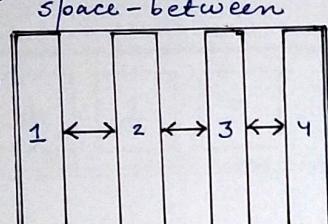
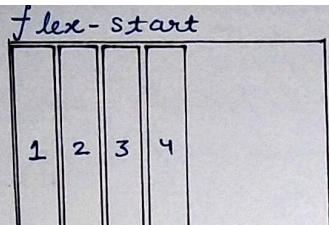
flex-direction: Column-reverse

\* row is a default value for flex-direction

justify-content  
justify-content  
 property aligns the flex items horizontally within the flex container.

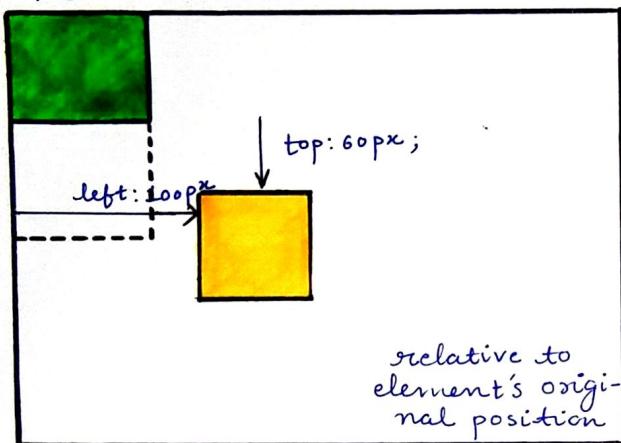
// In space-around items are evenly distributed in the line with half size spaces on either end.

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

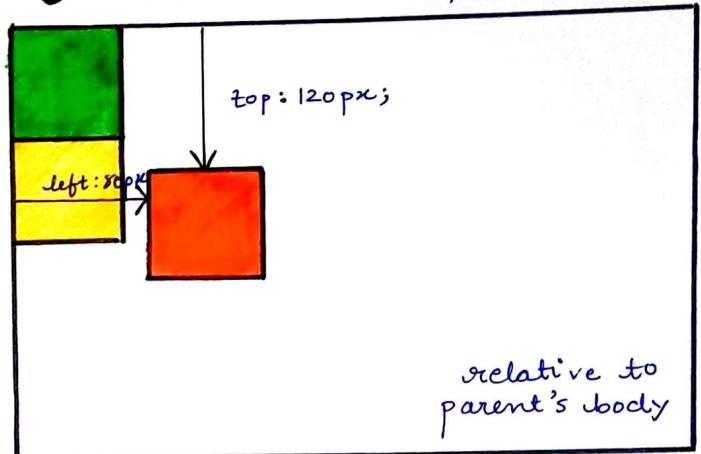


# Positioning in CSS

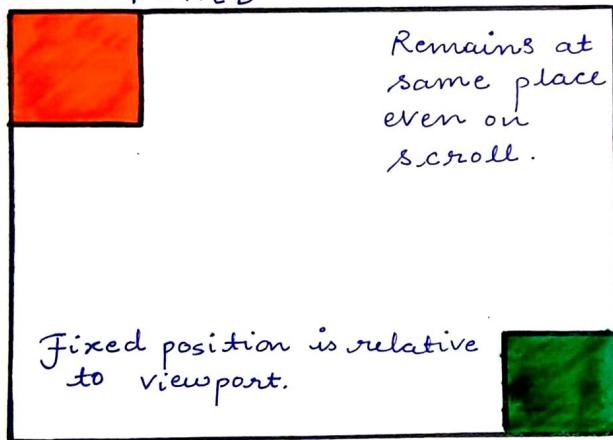
RELATIVE



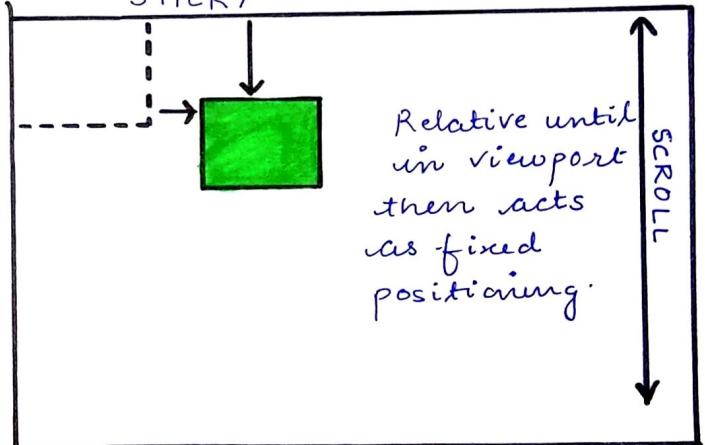
ABSOLUTE



FIXED

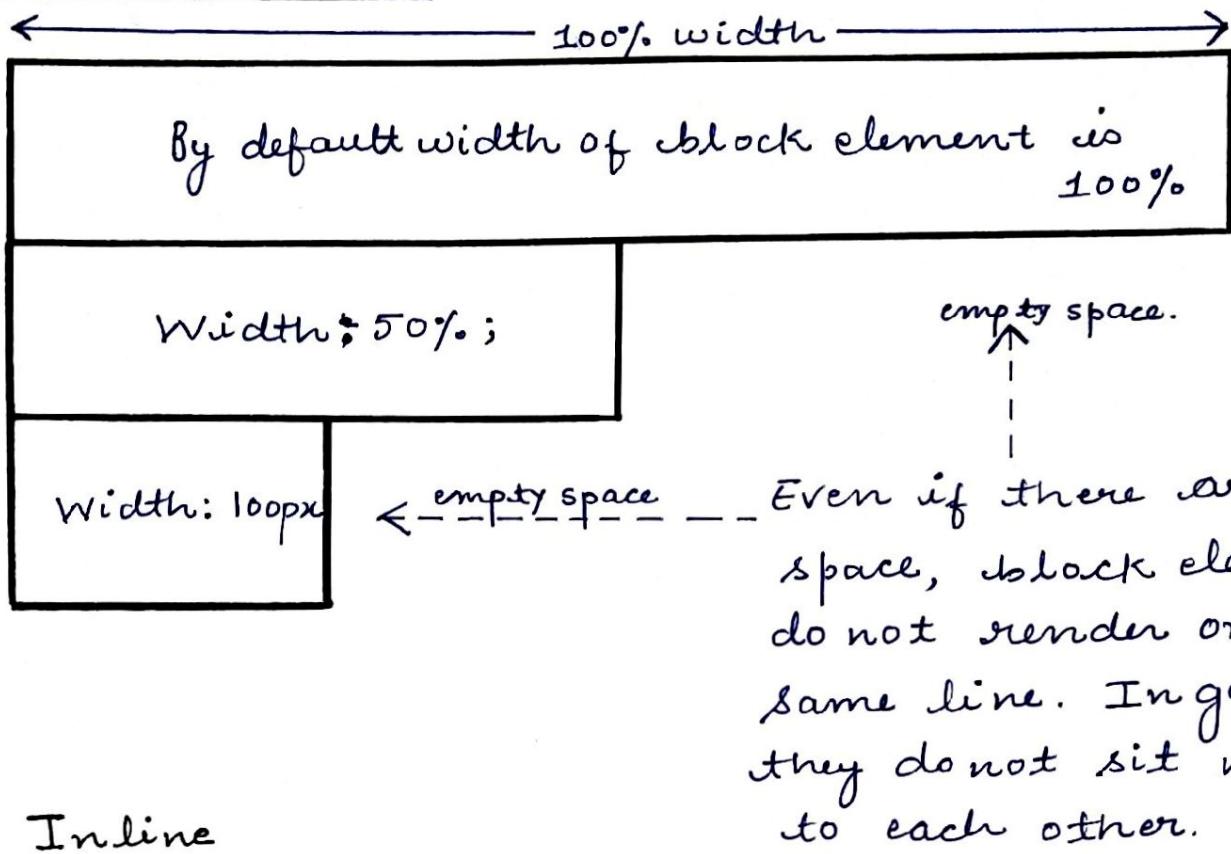


STICKY



## Display : Block, inline and inline-block.

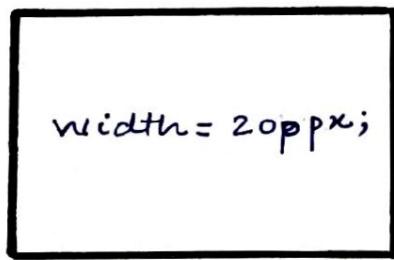
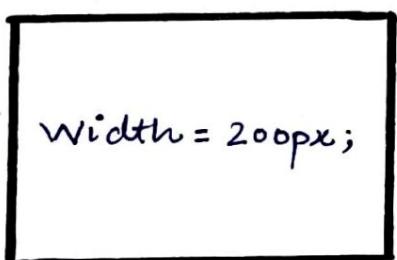
### Block Elements.



### 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

1fr	2fr	1fr
1	2	3
4	5	6

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

1	2	3
1	4	5
2	3	6

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

1	2	3
4	5	6

1	2	3	4
1	4	5	6

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

1fr	2fr	1fr	2fr.
1	2	3	4
5	6	7	8

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

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

grid-auto-columns:  
 minmax(100px, auto);  
 // min height 100px and max will be auto.

1	2	3
4	5	6
7	8	9

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

1	2	3
4	5	6

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

1	2	3
4		

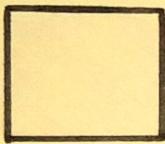
parent {  
 justify-items: start;  
 }  
 // it will align horizontally.

1	2	3
4	5	6

{ align-items: end; }  
 // align-items: center will align all grid items at center vertically.  
 align-items: start  
 // It will align vertically.

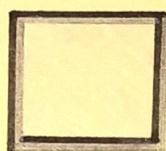
1	2	3
4	5	6

## border-style



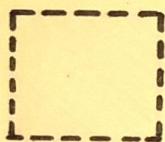
Solid

Displays a single straight solid line



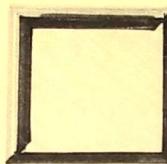
Groove

Displays the border with carved appearance.



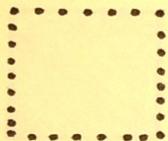
Dashed

Displays the small square of same length.



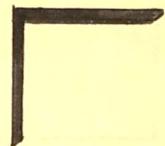
Ridge

Displays the border with an extruded appearance



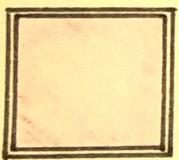
Dotted

Displays a series of rounded dots.



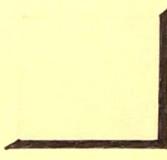
Inset

makes the element appear embedded.



Double

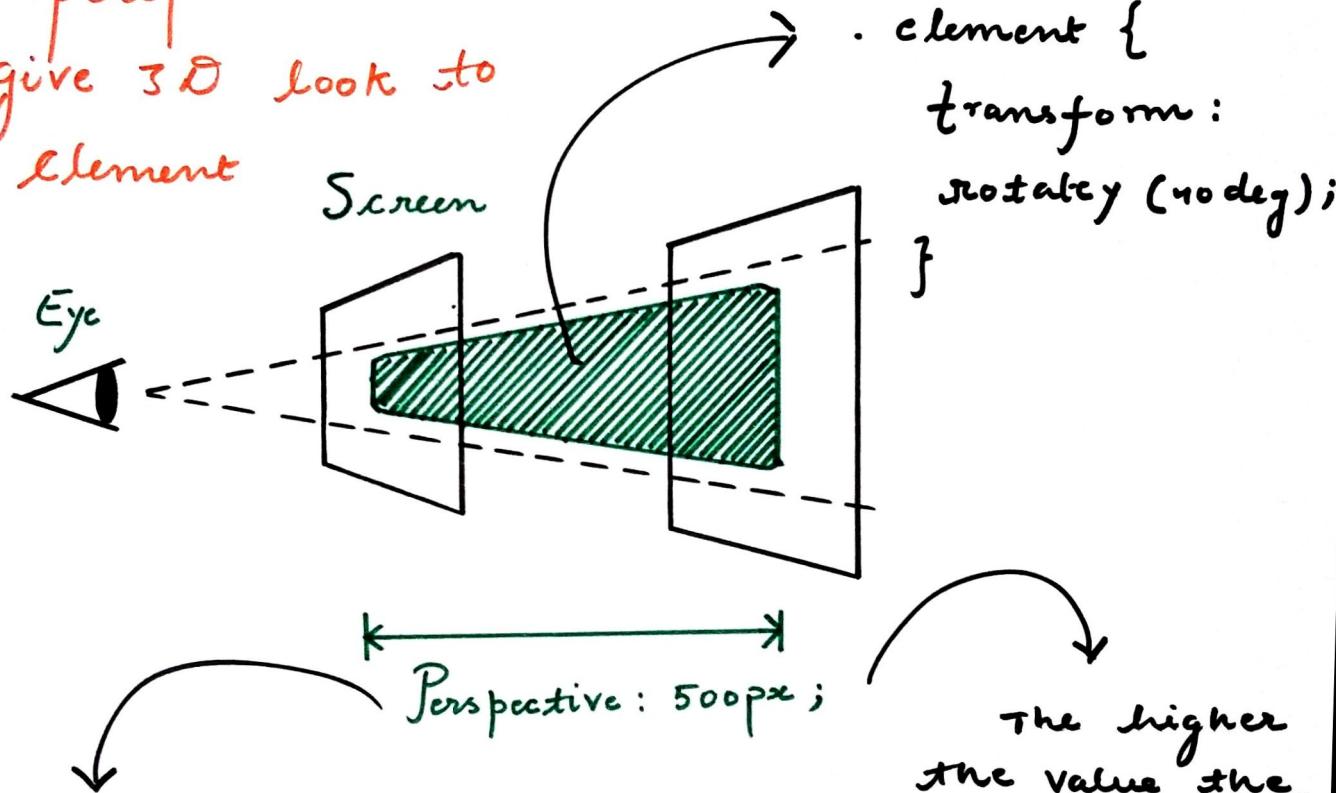
Displays two straight lines



Outset

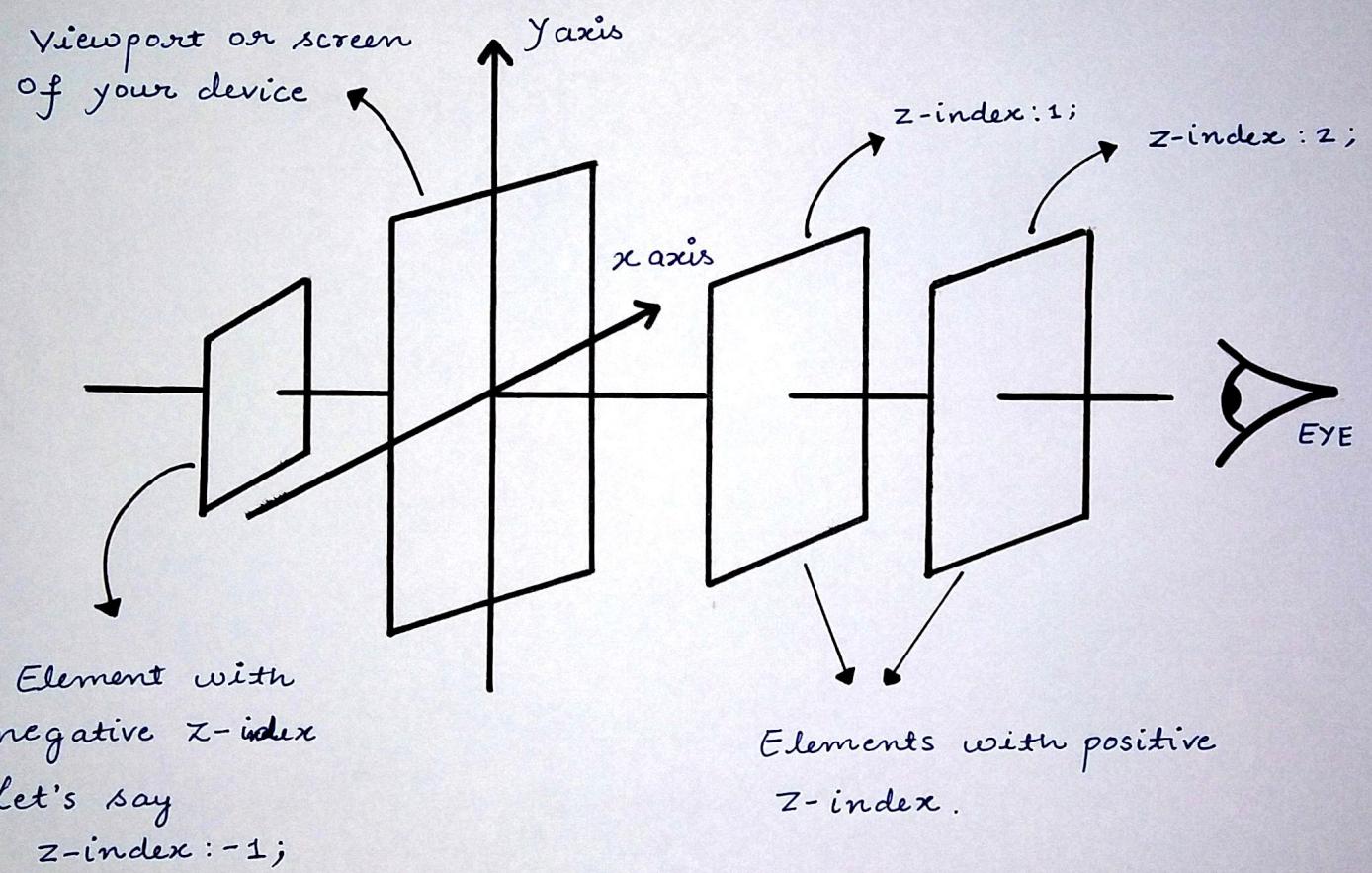
Displays a border that makes element appear embossed.

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

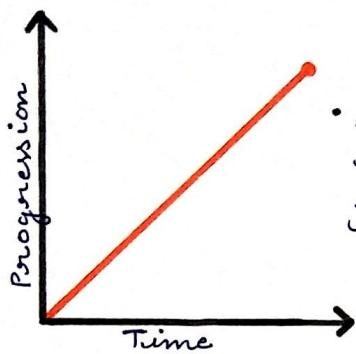


Lower value will create more intensive 3D look

## Z- INDEX

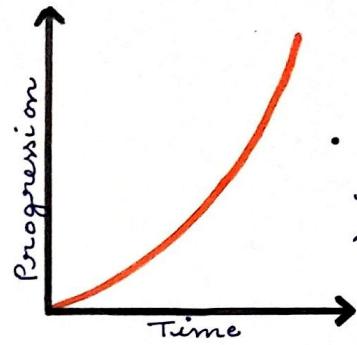


### Linear



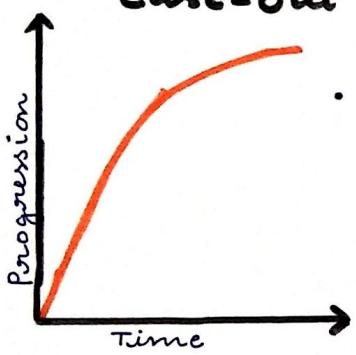
- Animation has the same speed from start to end.

### ease-in



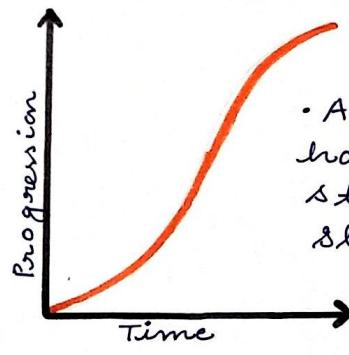
- Animation has the slow start

### ease-out



- The animation has a slow end.

### ease-in-out



- Animation has slow start and a slow end.

Prathmesh