

## Twitter Thread by [Pratham ■■■■](#)



[Pratham ■■■■](#)

[@Prathkum](#)

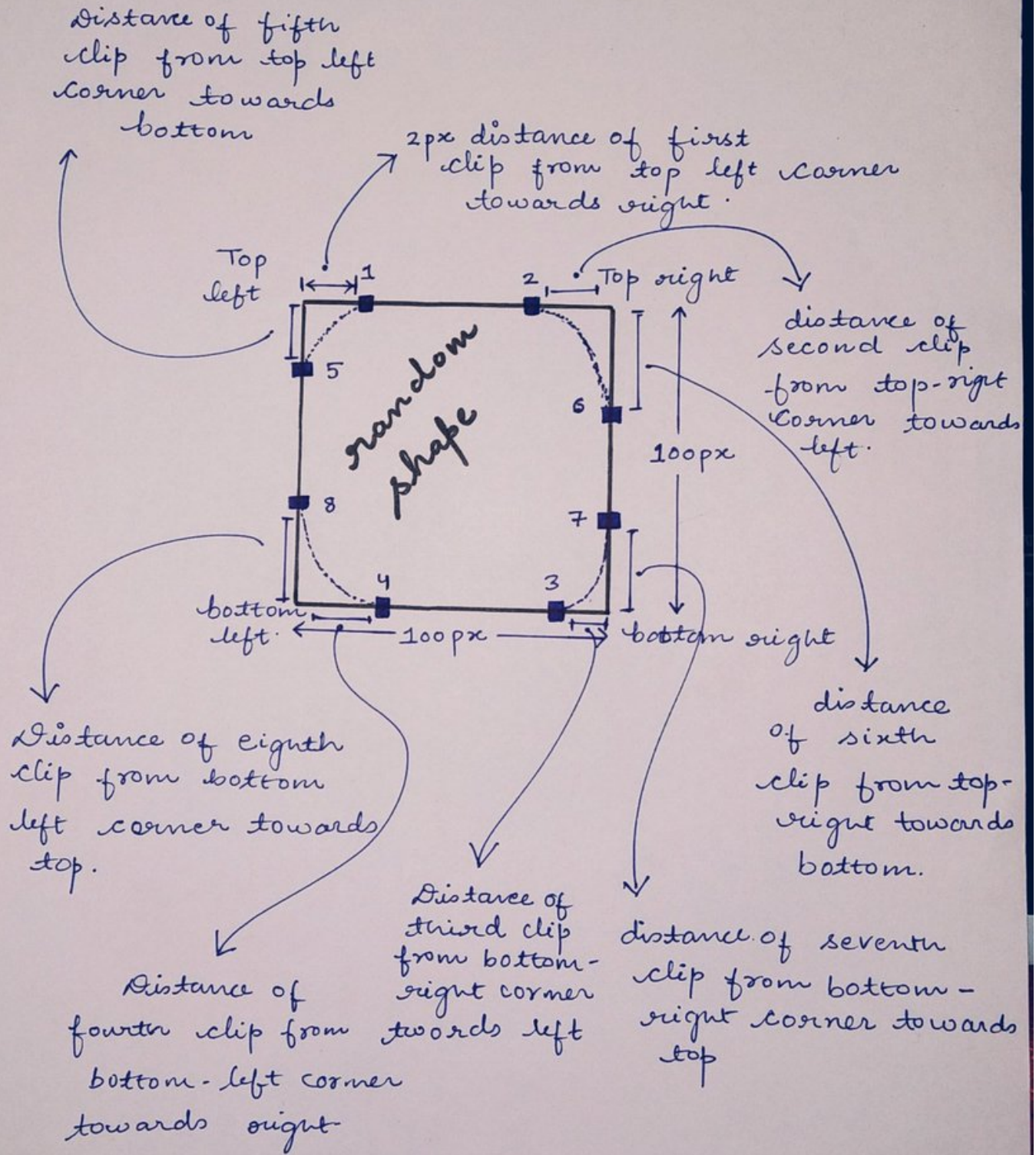


**A compiled tweet of the CSS cheats sheet I created in the last few days**



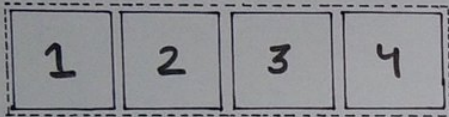
Border-radius

border-radius: 1 2 3 4 / 5 6 7 8 clips  
position



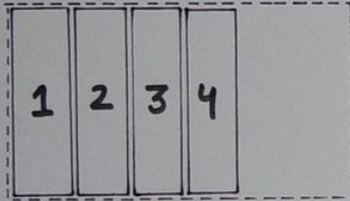
# FLEX CHEAT SHEET

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



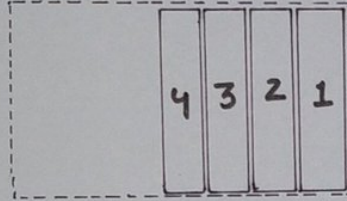
```
.container {  
  display: flex;  
}
```

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



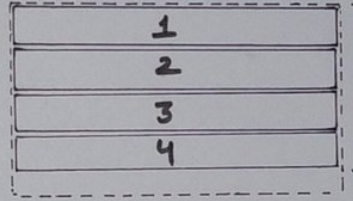
flex-direction: row;

→  
in a row from left hand side



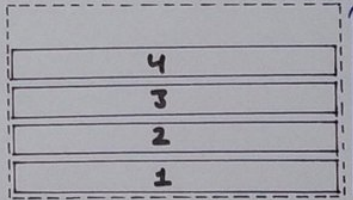
flex-direction: row-reverse

←  
in a row but from right hand side



flex-direction: column;

↓  
in column from top.



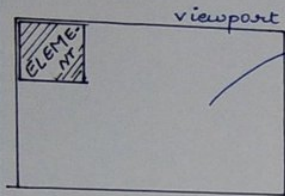
flex-direction: column-reverse

↑  
in a column from bottom.

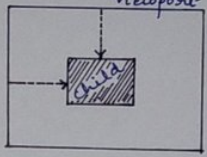
\* row is a default value for flex-direction

Relative and absolute positioning

# Relative and Absolute positioning.

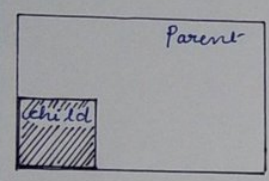


Original position of element. without applying any position property.



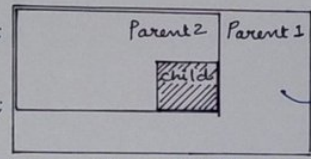
\* Relative position is relative to element's original position.  
 .child {  
 position: relative;  
 left: 100px;  
 top: 100px;  
 }

As you can see child element is shifted 100px from left and top from its original position.



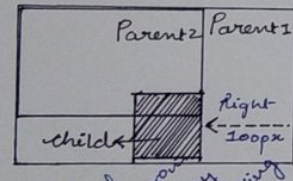
.parent {  
 position: relative;  
 }  
 .child {  
 position: absolute;  
 bottom: 0;  
 }

Absolute position is relative to its closest ancestor who is having position property applied.



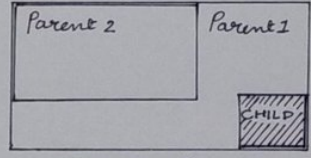
// In this child's closest ancestor is parent 2 having position property set. Hence when we apply absolute positioning to child, it will be relative to parent 2.

.parent1 {  
 position: relative;  
 }  
 .parent2 {  
 position: relative;  
 }  
 .child {  
 position: absolute;  
 bottom: 0;  
 right: 0;  
 }



.parent1 {  
 position: relative;  
 }  
 .parent2 {  
 // No position property applied  
 }  
 .child {  
 position: absolute;  
 right: 100px;  
 bottom: 0;  
 }

child is shifted 100px right from parent 1 boundary as parent 1 is having position property whereas parent 2 don't have any position.



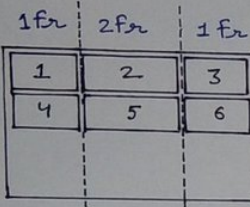
.parent1 {  
 position: relative;  
 }  
 .parent2 {  
 /\* No position \*/  
 }

```
<div class="parent1">  
  <div class="parent2">  
    <div class="child"></div>  
  </div>  
</div>
```

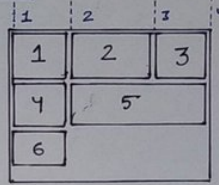
.child {  
 position: absolute;  
 bottom: 0;  
 right: 0;  
 }  
 child's closest ancestor having position applied is parent 1. Hence its relative to parent 1.

## Grid Layout

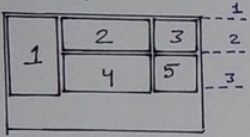
# Grid cheat sheet



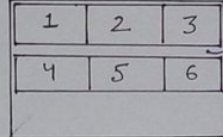
grid-gap: 10px;  
 grid-template-columns: 1fr 2fr 1fr;  
 grid-gap: 10px;  
 display: grid;



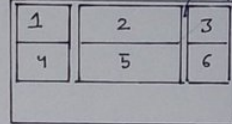
. five {  
 grid-column: 2/4;  
 }  
 // fifth element start from 2<sup>nd</sup> column and ends at 4<sup>th</sup> column



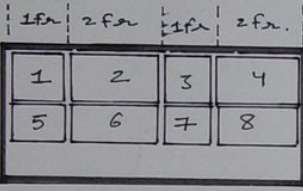
. one {  
 grid-row: 1/3;  
 }  
 // row gap



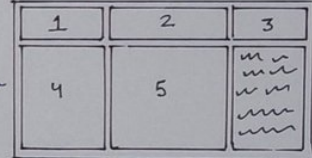
grid-row-gap: 10px



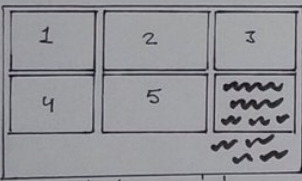
grid-column-gap: 10px;  
 // column gaps.



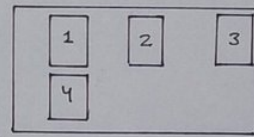
grid-template-columns:  
 repeat(2, 1fr 2fr);  
 // It will repeat column 2 times as 1 fraction and two fraction



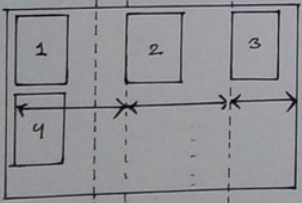
grid-auto-columns:  
 minmax(100px, auto);  
 // min height 100px and max will use auto.



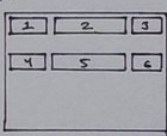
. parent {  
 grid-auto-row: 100px;  
 }  
 // every item has 100px height. Due to which content overflows. In order to prevent it.



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

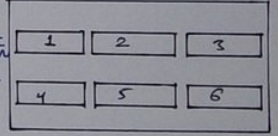


. parent {  
 justify-items: start;  
 }  
 // it will align horizontally.



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

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



CSS box model





## CSS BOX-MODEL

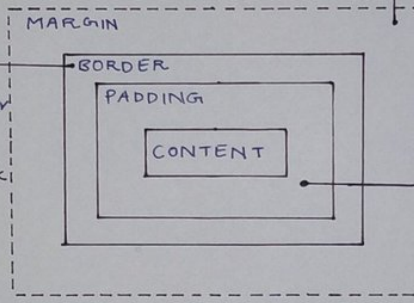
• Everything in CSS is a box or rather everything in HTML is a box-model which is surrounded by 4 different box virtually.

1. Content (original content inside element)
2. Padding (create space b/w <sup>content</sup> and element's)
3. Border (create border around element) border)
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:       Solid:   
 dashed:   
 double: 

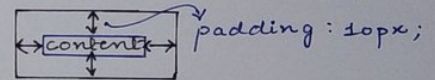


→ Margin defined the space between element.  
for ex. `margin: 10px;`  
It will create 10px empty space around element in all direction.

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

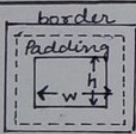
`margin: 25px 10px 25px`  
 // Top, right and left, bottom  
`margin: 25px, 10px;` and  
 // Top, bottom, right, left

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



// 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.

PRATHAM 😊

## Alignment in flexbox

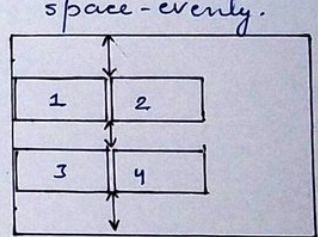
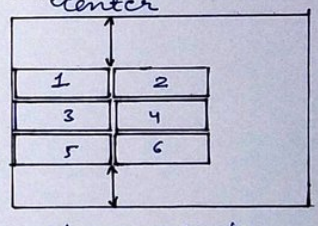
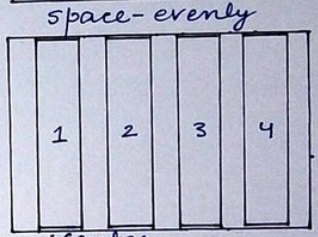
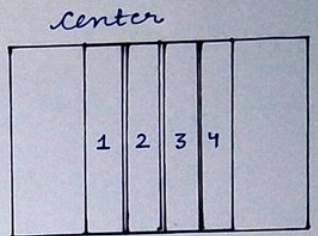
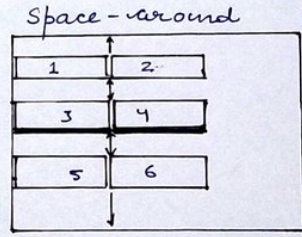
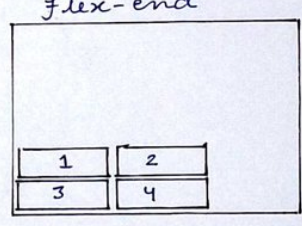
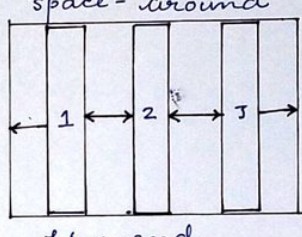
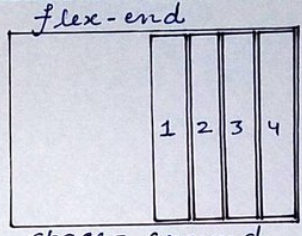
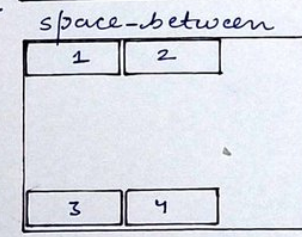
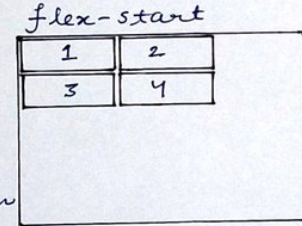
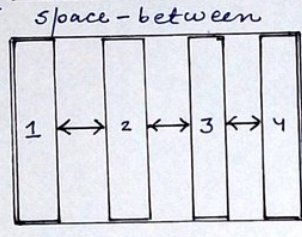
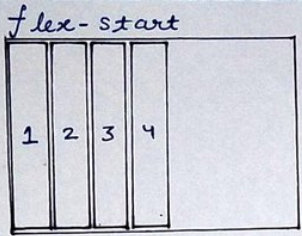
justify content

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.



More amazing content is coming  
stay around 