Twitter Thread by Pratham ■■■■
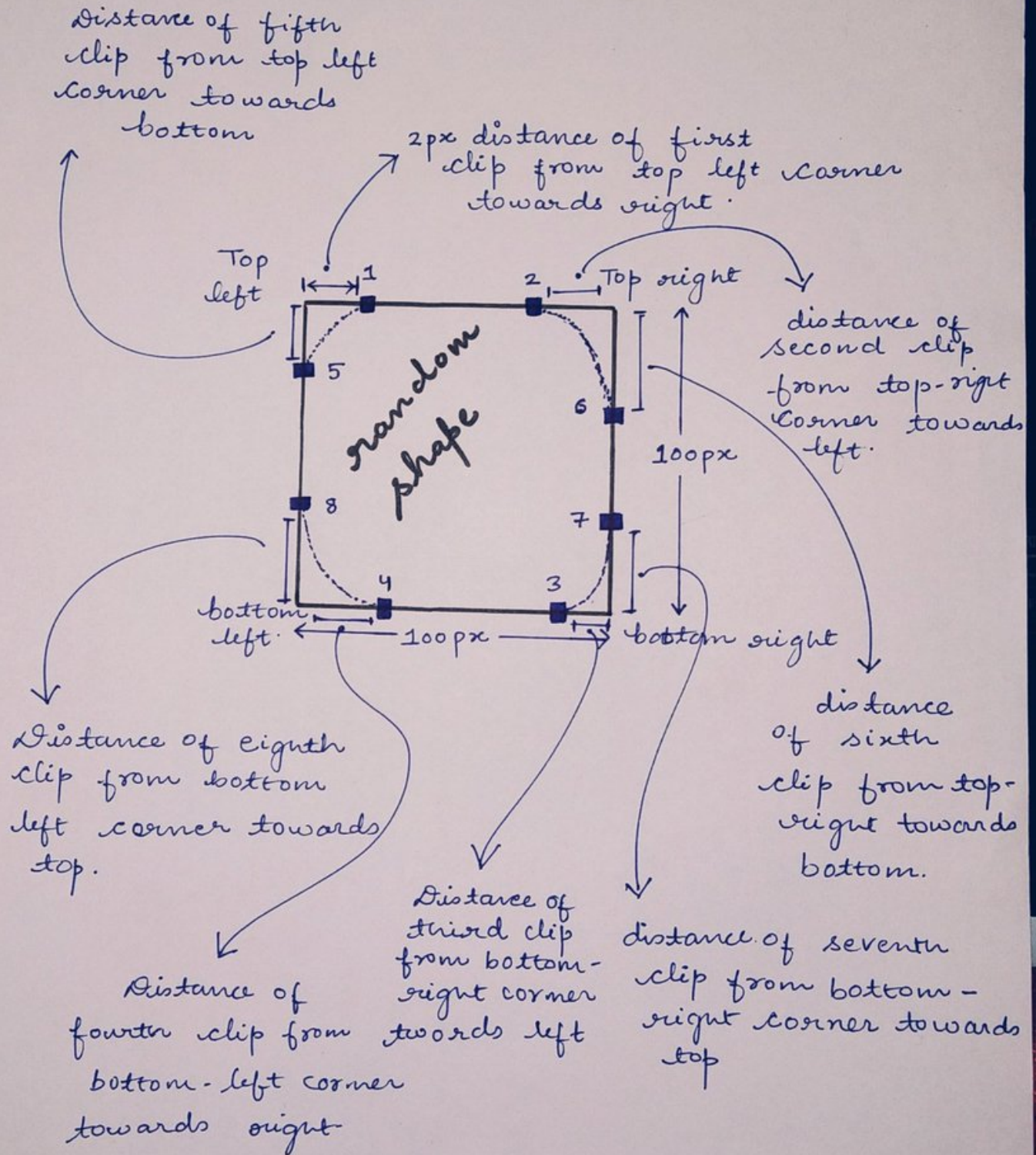
Pratham ■■■■
@Prathkum

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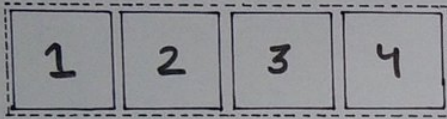
■■

Border-radius

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

Distance of fifth clip from top left corner towards bottom

2px distance of first clip from top left corner towards right.

Top left

1

2

Top right

distance of second clip from top-right corner towards left.

random shape

5

6

100px

8

7

bottom left

4

3

100px

bottom right

Distance of eighth clip from bottom left corner towards top.

distance of sixth clip from top-right towards bottom.

Distance of fourth clip from bottom-left corner towards right

Distance of third clip from bottom-right corner twords left

distance of seventh clip from bottom-right corner towards top

Flexbox

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.
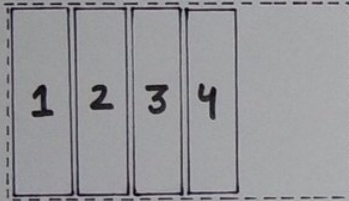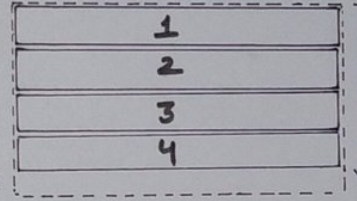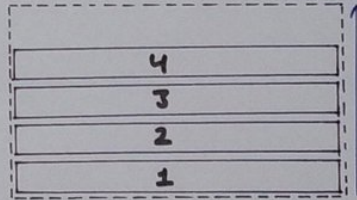

flex-direction: row;
⟶
in a row from
left hand side
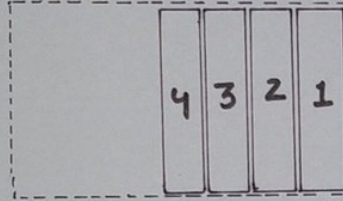
flex-direction: row-reverse
⟵
in a row but from
right hand side

flex-direction:
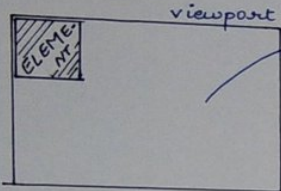Column;
↓ in column
from top.
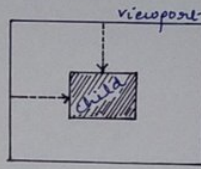
in a column
from bottom.

flex-direction: Column-reverse

* row is a default value for flex-direction

Relative and absolute positioning
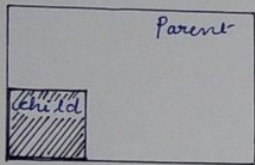
Relative and Absolute positioning.

viewport → Original position of element. Without applying any position property.

viewport →

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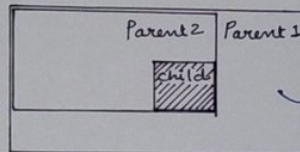
* Relative position is relative to element's original position.

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 it's closest ancestor who is having position property applied.
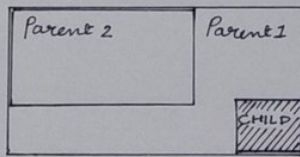
Parent2 Parent1

// 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.

```
· parent 1 {
    position: relative;
}
· parent 2 {
    position: relative;
}
· child {
    position: absolute;
    bottom: 0;
    right: 0;
}
```

```
<div class="parent1">
    <div class="parent 2">
        <div class="child"></div>
    </div>
</div>
```

Parent2 Parent1
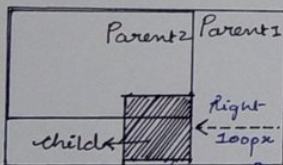
Right ←----- 100px

child ←

```
· parent 1 {
    position: relative;
}
· parent 2 {
    // No position
    property applied
}
· child {
    position: absolute;
    right: 100px;
    bottom: 0;
}
```

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

Parent 2   Parent1

CHILD

```
· parent 1 {
    position: relative;
}
· parent 2 {
    /* No position */
}
```

```
· child {
    position: absolute;
    bottom: 0;
    right: 0;
}
```

child's closest ancestor having position applied is parent1 hence its relative to parent1.

Grid Layout

Grid cheat sheet.

1fr | 2fr | 1fr

```
1   2   3
4   5   6
```

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

```
1   2   3
4   5
6
```

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

```
1
  2   3
  4   5
```

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

```
1   2   3
4   5   6
```
→ grid-row-gap : 10px

```
1   2   3
4   5   6
```
→ grid-column-gap : 10px;
// column gaps.

// we can use
grid-gap. It will
add equal space
b/w rows and
column.

1fr | 2fr | 1fr | 2fr.

```
1   2   3   4
5   6   7   8
```

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

```
1   2   3
4   5
```

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

```
1   2   3
4   5
```

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

```
1   2   3
4
```

. 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.
at vertically.

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

```
1   2   3
4   5   6
```

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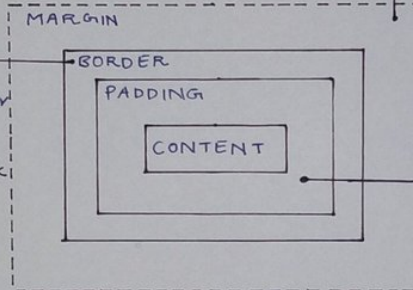3. Border (create border around element)
4. Margin (space between element)

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

for ex: border: 1px solid black

dotted : ⸬⸬⸬⸬

dashed : ▯▯▯▯

double : ▯

Solid: ▯

```
MARGIN
  BORDER
    PADDING
      CONTENT
```

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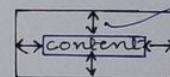
margin : 25px 10px 25px
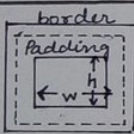// Top, right and left, bottom

margin: 25px, 10px ; 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.

PRATHAM :)

Alignment in flexbox
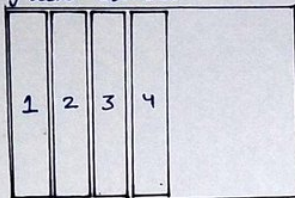
justify content

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.
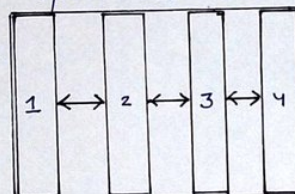
align-content

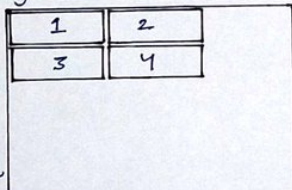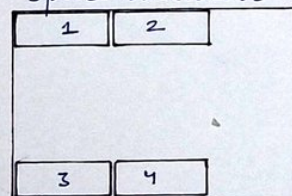align-content property align the flex items vertically within the flex container.

flex-start
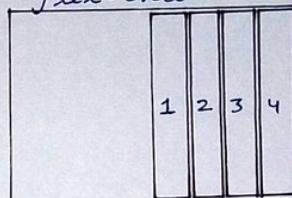
| 1 | 2 | 3 | 4 |

flex-end

| 1 | 2 | 3 | 4 |

center

| 1 | 2 | 3 | 4 |

space-between

| 1 | ←→ | 2 | ←→ | 3 | ←→ | 4 |

space-around

← 1 ←→ 2 ←→ 3 →

space-evenly

| 1 | 2 | 3 | 4 |

flex-start

| 1 | 2 |
| 3 | 4 |

flex-end

| 1 | 2 |
| 3 | 4 |

center

1	2
3	4
5	6

space-between

| 1 | 2 |
| 3 | 4 |

Space-around

1	2
3	4
5	6

space-evenly

| 1 | 2 |
| 3 | 4 |
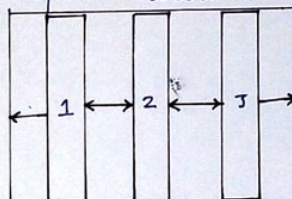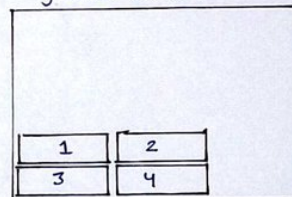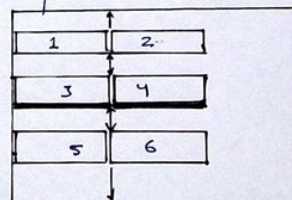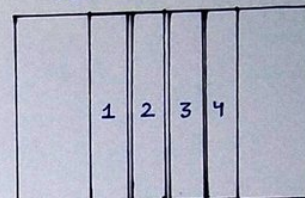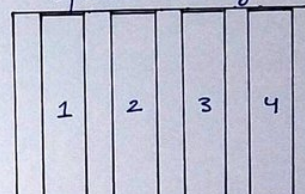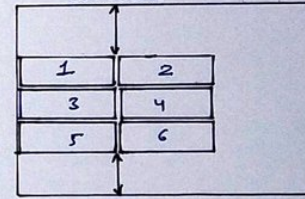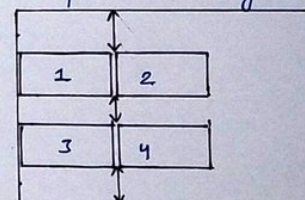
More amazing content is coming
stay around■❤■