* Grid - Desplay...

→ By listening to grid what firstly comes to our mind is a 2D structure. A 2D structure like a Matrix.

We use property → display : grid;
display : inline-grid;

Ex →
```
.grid-card {
    display : grid;
    grid-template-colums : 40px auto
                           50px auto;
}
```

So, grid-template-columns gives the space to the elements in that particular 2D structure or as of now we have table so in the table.

→ When we do,

(display: grid;)

→ The elements will occupy the entire width on the screen.

Whereas, when we do -

(display: inline-grid)

→ The elements will only occupy the width which they need acc. to their actual element width not the complete width present on the screen.

* Properties →

1. grid-template-columns: 50px 50px auto auto;

Apart from 'px' and 'auto' we can also use 'fr' which stands for 'fraction'.

(Ex → grid-template-columns: 1fr 1fr 1fr 1fr;)

② grid-template-rows: 50px 50px 50px;

③ grid-row-gap: 25px;
"Provides a margin or gap of 25px between each row."

④ grid-column-gap: 40px;
"Provides a gap of 40px b/w each column".

* We have a shorthand property for "grid-row-gap" and "grid-column-gap"

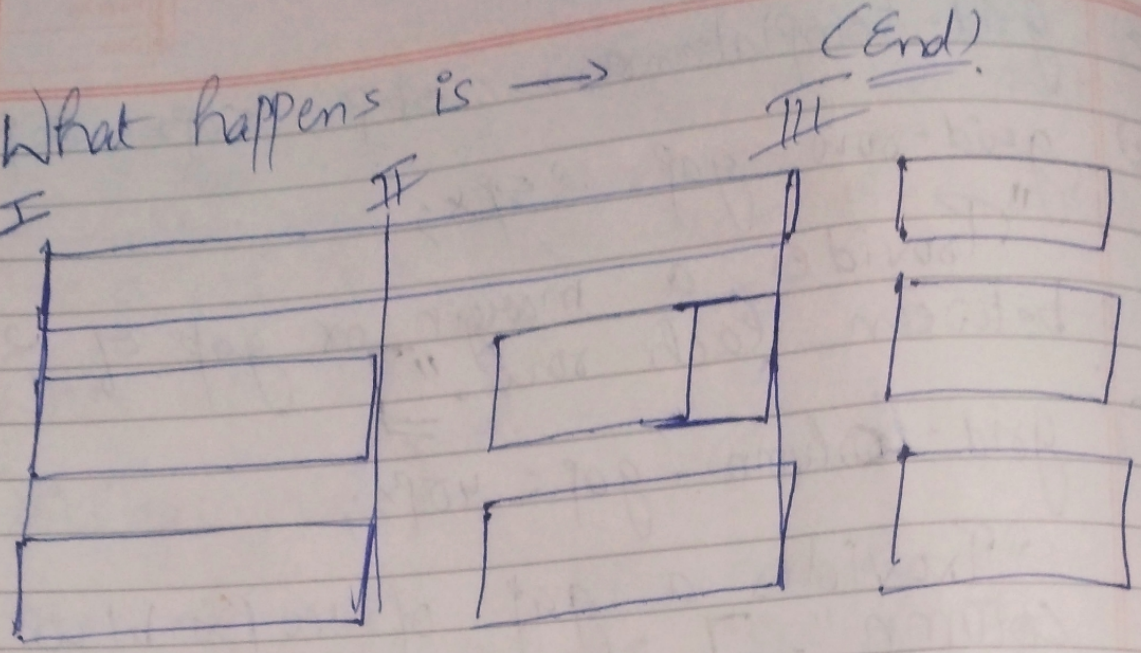
→ grid-gap: 25px 40px;
↓ row ↓ column.

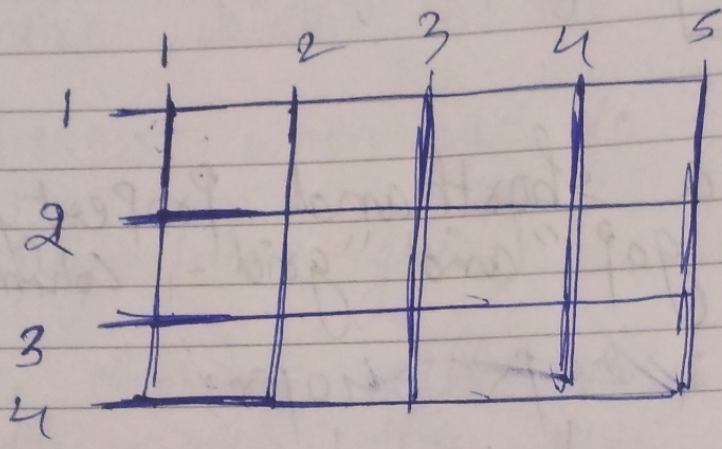
* Just in case we want to style a particular element of a column or a column.

→ We do →

```
{
grid-column-start: 1;
grid-column-end: 3;
background-color: yellow;
}
```

What happens is →

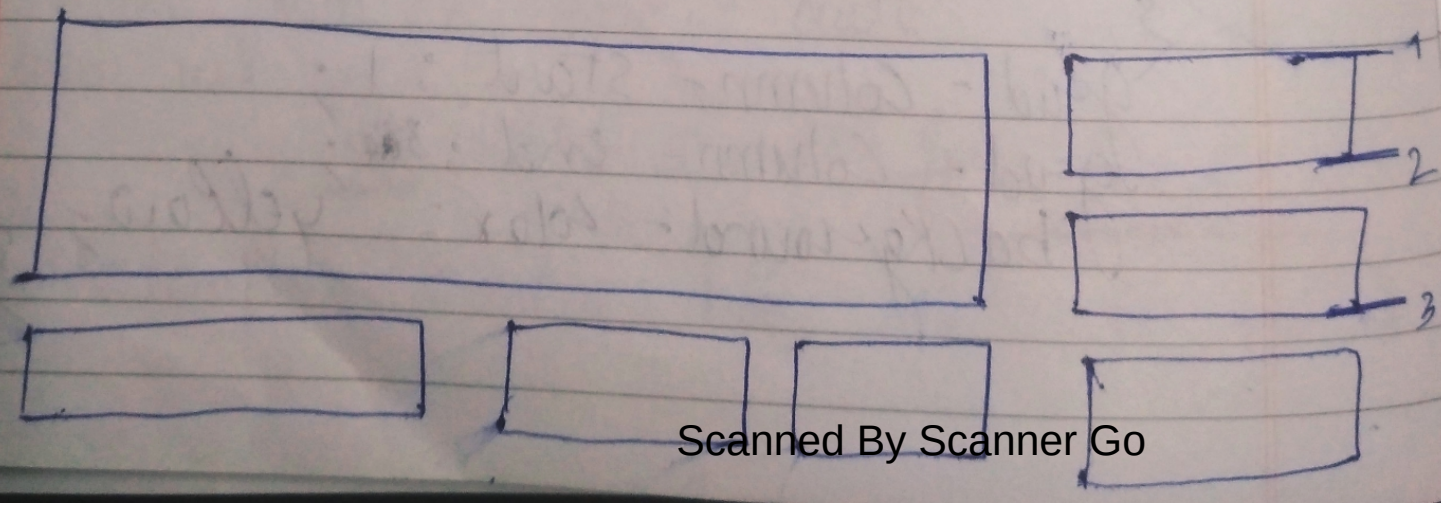




→ Similarly, in the case of rows →

We use,

```
{
    grid-row-start : 1;
    grid-row-end : 3;
}
```



* If we want to have same width for all columns so instead of writing it again and again what can be done is →

→ grid-template-columns: repeat(4, 150px);
(4 → no. of columns; 150px → width..)

* Shorthand-Property for →
[grid-column-start and grid-column-end]

→ grid-column: 1/4; or 1/span 3;

& for row →
grid-row: 1/3; or 1/span 2;

* Grid-Alignment →

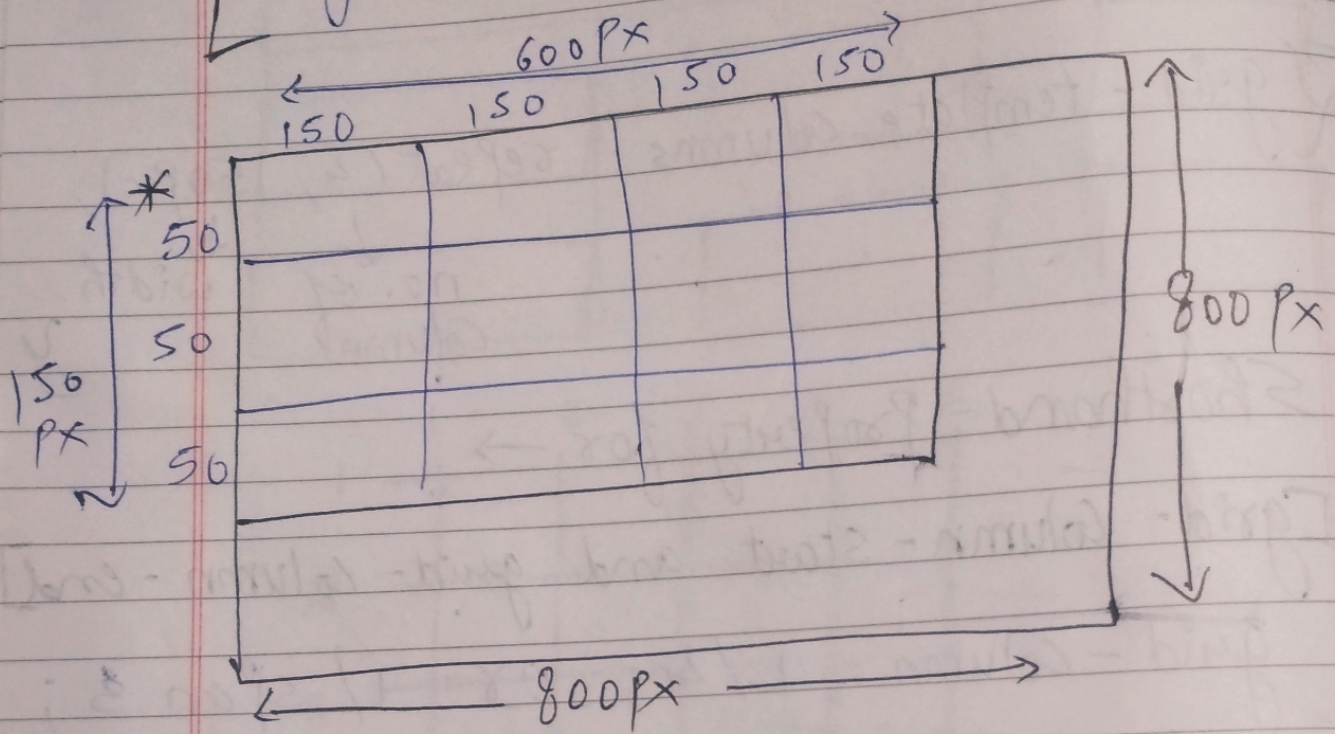
→ To align the items inside grid in a particular cell.
→ for aligning ~~them~~ the content horizontally we do is →

justify-items: center/stretch/start/end;

the content in the grid

→ To Align vertically we do is →

[align-items: baseline / center / stretch / flex-start / flex-end;]



→ So, as in the above fig. the actual container is of 800 px width and is of height 800px.

→ Now, when we align content of grid inside the main-container with the width which we require then it will not cover up the whole space in the container.

→ So, in-order to fix this we use-

"justify-content" and "align-content".

[Horizontally] [Vertically]

{ justify-content: start / end / center / space-between / space-around / space-evenly ; }

(gives space blw the columns)

{ align-content: space-between / space-around / space-evenly / start; }

(gives space blw the rows)

* In order to align a particular cell we use →

{ align-self: center; }

→ It works vertically...

{ justify-self: center; }

→ It works horizontally...

* Responsive Design →

→ It means that how you are making your website so that it gets accommodated easily in every screen size, be it mobile view, web-view etc...

① → 320 - 480px → Mobile Devices
② → 481 - 769 px → iPad, tablets
③ → 769 - 1024 px → small screens, laptops.
④ → 1025 - 1200 px → Desktops, large screens
⑤ → 1201 and more → Tv, xl screens.

We do it with →

[Complete height/width..]
[The screen on which we see everything on mobile screen.]

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

[The width on which rendering is happening should be equal to the device-width.

[We want things to be shown as it is no zoom-in/out.]

So, when we go to mobile view and the width is 360px so everything will be rendered in 360px only.]

We use →

```
@media screen and (min-width: 320px)
        and (max-width: 480px) .
  {
    body {
        background-color: yellow;
    }
  }
```