

6. CSS RESPONSIVE LAYOUT & GRID

Order:

A flexbox property that allows you to control the visual order of flex items in a flex container.

By default the flex items have order of 1.

`order: -1;`

Items will be displayed in ascending order.

Items with lowest order will come first, items with highest order

will be displayed at the end.

`flex-shrink : 3; // Default value is 1.`



To shrink one flex item more than the others.

This will cause the flex item to shrink 3 times more than the other items.

`flex-grow : 10;`



To grow one flex item more than the others.

The flex item will grow 10 times more than the other items.

Flexbox Froggy → website

Responsiveness: Ability of website or web application to adopt & provide an optimal UX across various devices & screen sizes.

Responsive Design: It ensures that the content & layout of a website adjust dynamically based on device characteristics such as screen width, height & orientation.

Is flexbox enough for responsive design?

Flexbox + Grid → 99% code
responsive

Media Queries:

You are a chef preparing a dish.

Some like mild, medium, spicy.

Chef doesn't prepare different dishes for these scenarios.

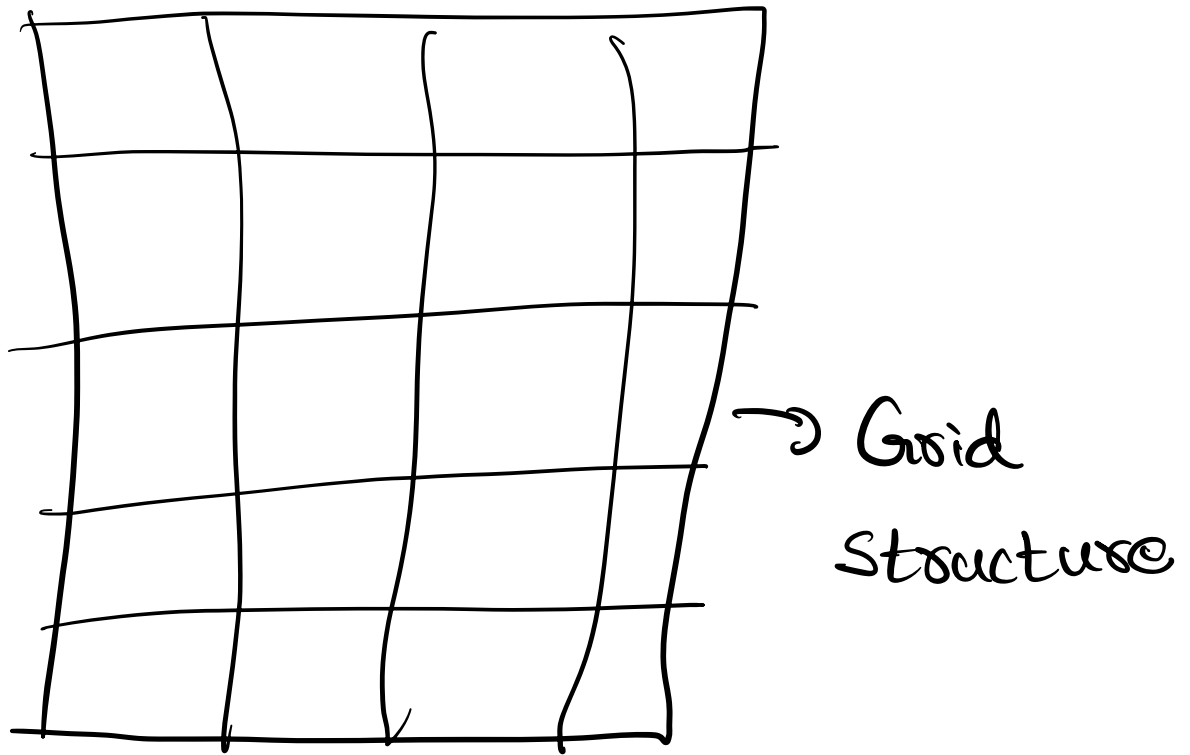
Media Queries → extra salt, pepper, chili that is added for customization.

↓
Do not Use (Use as less as possible)

Use only to make minor changes.

Media Queries come into picture to style based on screen size.

Grids:



Flexbox → Arranges elements in
main or cross axis.
(1D Layout)

Grid → Arranges element in 2D
layouts.

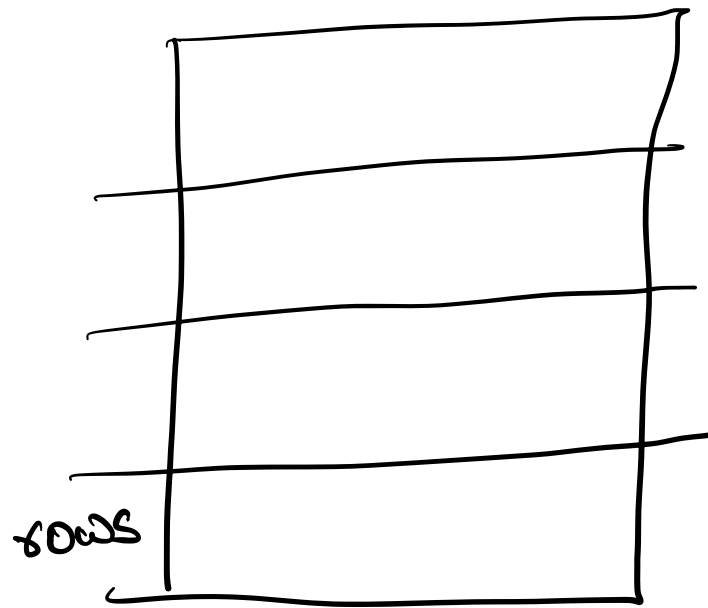
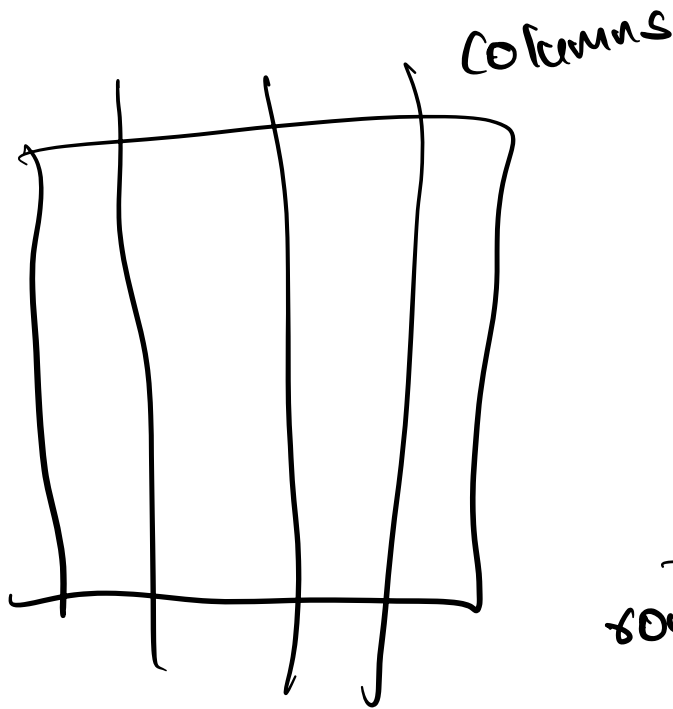
display: grid;



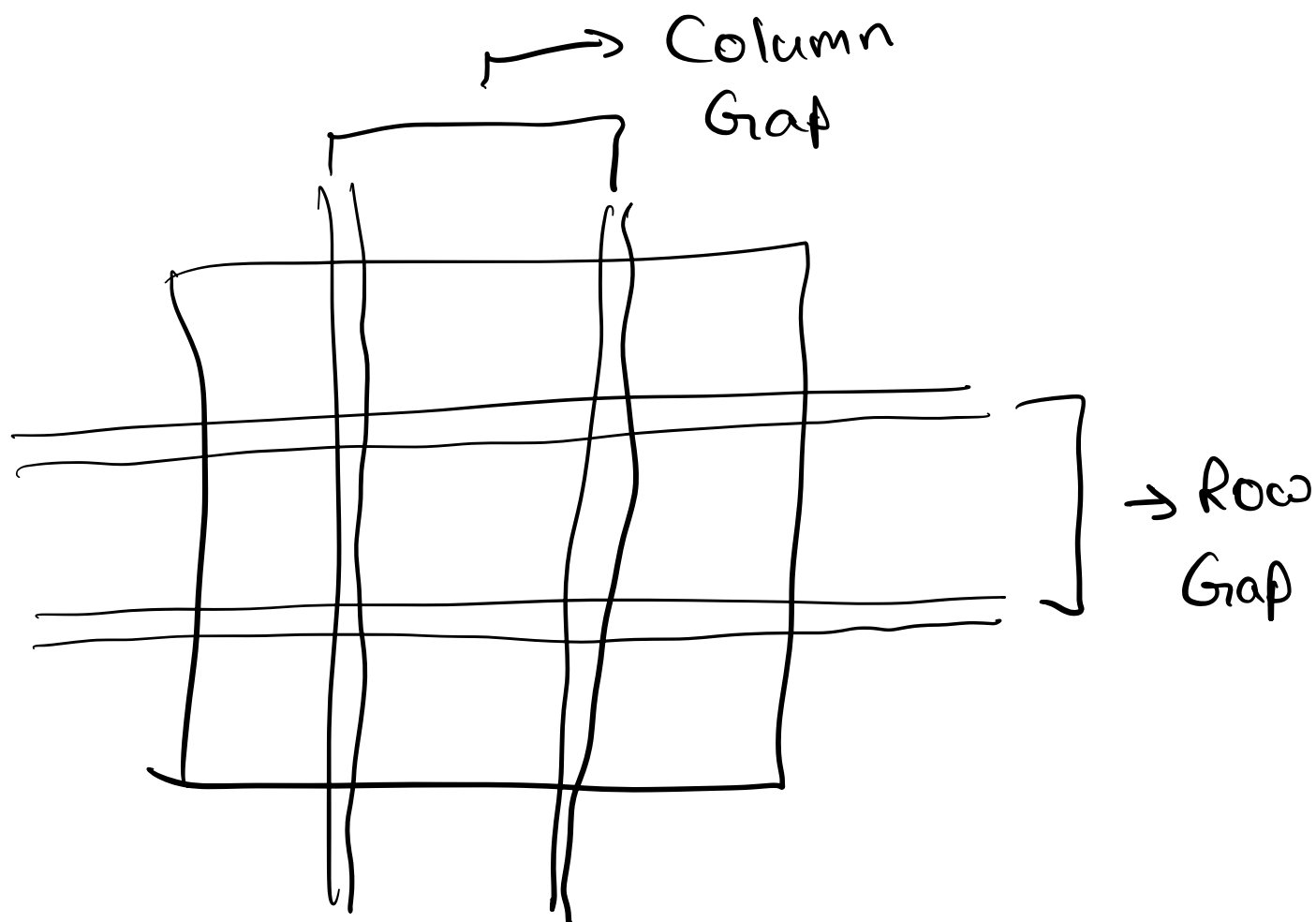
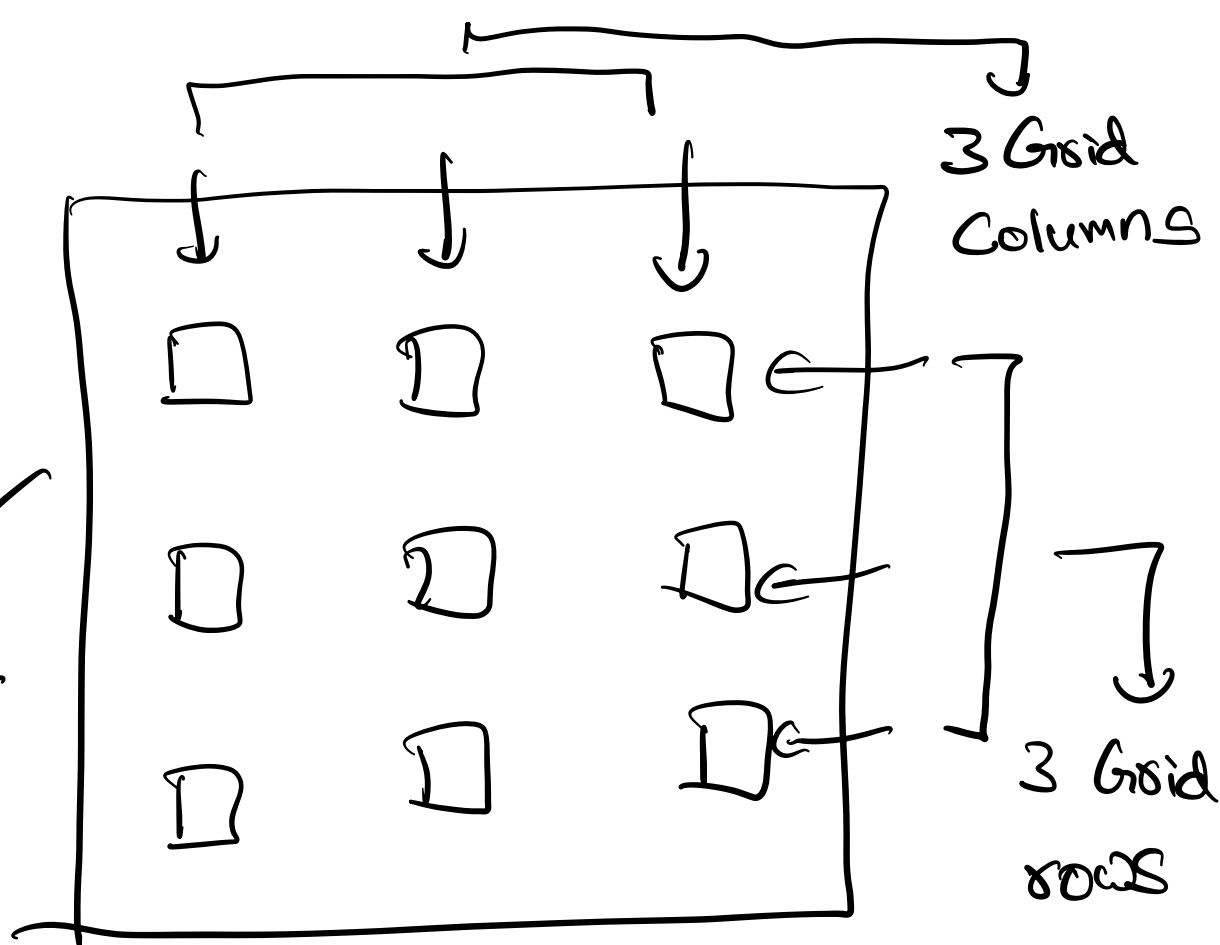
Given to a container.

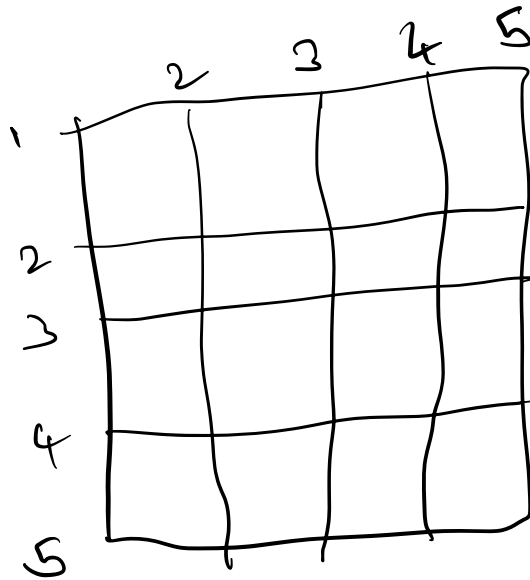
It can contain grid items.

Grid Container is visualized in terms of
Grid rows & Grid columns.



Grid
Containers





rows = 4

columns = 4

row lines = 5

column lines = 5

row lines = number of rows + 1

column lines = number of columns + 1

grid-template-columns: 100px 100px 100px;



Gives 3 columns with 100px width.

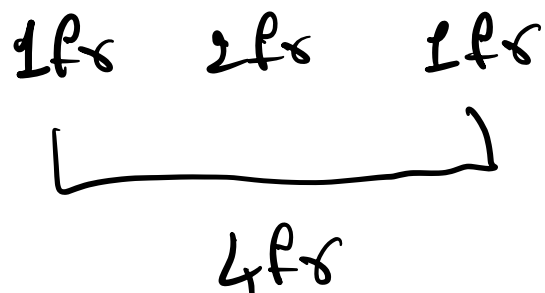
Using pixel units is not a good thing.

Better thing would be to do it based on container width.

grid-template-columns: 1fr 1fr 1fr;

fr \rightarrow frame

The container width in this case 1200px is divided into 3 columns & each column will be 1fr.



Behind the scenes it's broken down into 4 frames. 2nd column is assigned 2 frames.

Let's say container width is 1200 px
grid-template-columns: 1fr 2fr 1fr;

$$\text{Total frames} = 1 + 2 + 1 = 4$$

$$\text{Size of each frame} = 1200 / 4 = 300 \text{ px}$$

$$1 \text{ fr} = 300 \text{ px}$$

$$\text{first column} = 1 \text{ fr}$$

$$= 1 \times 300 \text{ px}$$

$$= 300 \text{ px}$$

$$\text{Second column} = 2 \text{ fr}$$

$$= 2 \times 300 \text{ px}$$

$$= 600 \text{ px}$$

third column = 1fr

= 1 x 300px

= 300px

column-gap: 20px;



To give space between columns.

row-gap: 30px;



To give space between rows.

gap: 10px;



To give space between to both rows & columns equally.

grid-template-rows: 100px 100px 100px;



Similar to grid-template-columns

grid-template-columns: repeat(3, 1fr);



Give me 3 1fr.

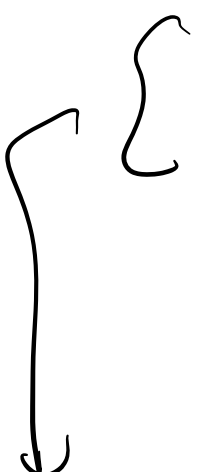
To apply style to a grid item without using classes like inline styling,

.item:nth-of-type (2) {

background-color: pink

}

Select second element with item class.

align-self : center; → arranges horizontally.
justify-self : center;
↓
arranges vertically

arranges in the particular cell.

These are given to grid items.

grid-row: 1/3;

↓

Grid item should start at row line number 1 & end at row line number 3.

grid-column → similar to
grid-row