CSS Flexbox

Short hand for flexible box layout ~~modble~~ module

Flexbox makes it easier to design a flexible responsive layout structure, without using float or positioning

Before flexbox, there were 4 layout models:

i) Block, for sections in a webpage

ii) Inline, for text

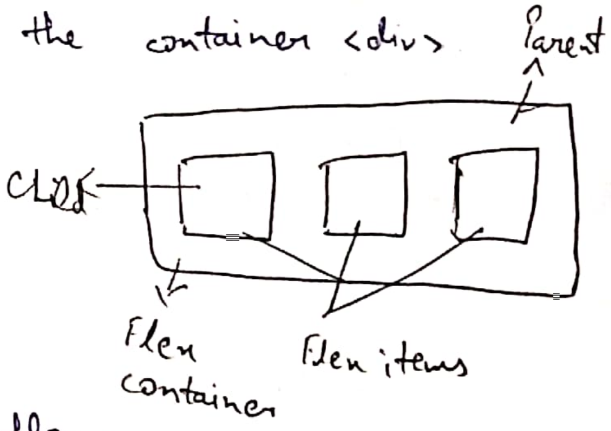
iii) Table, for 2D Table data

iv) Positioned, for explicit position of an element

A flexbox always consists of:

-) a Flex Container : the parent (container) <div> element

-) Flex items : the items inside the container <div>



Flex Container Properties:

Flex container becomes flexible by setting the display property to flex.

Properties..

i) flex-direction

ii) flex-wrap

iii) flex-flow

iv) justify-content

v) align-items

vi) align-content

Flex - direction

The flex-direction property defines in which direction the container wants to stack the flex items

Ex
```
.flex-~~direction~~container {
    display : flex;
    flex-direction: row | column | row-reverse | column-reverse;
                     ↓
                   default
}
```

Flex-wrap

The flex-wrap property speifies whether the flex items should wrap or not

Ex
```
.flex-container {
      display: flex;
      flex-wrap: wrap | no-wrap | wrap-reverse ;
                          ↓
                        default
}
```

Flex - flow

Short hand for setting both direction & wrap property

Ex
```
.flex-container {
      display: flex;
      flow - flow: row wrap;
}
```

Justify - content

Justify - content property is used to align the flex-items.

__Imp__ It is responsible for the alignment in horizontal axis (main-axis)

- justify-content can have one of the following values:

* center
* flex-start (default)
* flex-end
* space-around
* space-between
* space-evenly

__center__

The center value aligns the flex items at the center of the container.

__flex-start__

Aligns the flex items at the beginning of the container

__flex-end__

Aligns the flex items at the end of the container

__space-around__

displays the flex items with space around them.

__space-between__

displays the flex-items with space between them. There is no space ~~before or after~~ at the start / end of boxes

__space-evenly__

The space distribution at the start, between and end will be equal.

Ex

```
.flex-container {
    display: flex;
  justify-content: center | flex-start | flex-end | space-around /
                      space-between | space-evenly ;
  }
```

Align-items

The align-items property is used to align the flex items. Used for single row only.

Imp It is responsible for the alignment of the flex items in the vertical axis. (cross-axis)

The align-items property can have one of the following values:

- center
- flex-start
- flex-end
- stretch (default)
- Baseline
- normal

center

The center value aligns the flex items in the middle of the container.

flex-start:

aligns the flex items at the top of the container.

flex-end

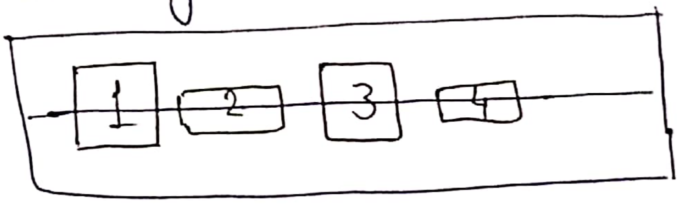
The flex-end value align the flex items at the bottom of the container

stretch

The stretch value stretches the flex items to fill the container

Baseline

The baseline value aligns the flex items as their baselines aligns



Align-content

It is same as align-items but it is used when our content is of more than one rows.

Properties

- space-between
- stretch
- flex-start
- space-around
- center
- flex-end

Q How to perfectly center a flex item / div?

A- Set both the justify-content & align-items properties to center, and the flex item / div will be perfectly center.

CSS Flex Items

A direct child elements of a flex container automatically becomes flexible (flex) items.

Properties of flex-items are:

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

order

specifies the order of the flex items inside the flex container

Default value is 0.

flex-grow

The flex-grow property specifies how much a flex item will grow relative to the rest of the flex items

- The value must be a number, default value is 0.

Applied to individual flex-items

flex-shrink

The flex-shrink property speifies how much a flex item will shrink relative to the rest of the flex items

Default value is 1

flex-basis

speifies the initial length of the flex items.

Difference between width and flex-basis

width	flex-basis
i) Visually you won't notice any difference when content is less	(Same no change as in width)
ii) when the content is more than the width of the flex, then the additional content will be trimmed out	If the content is more than the flex width, then the width will be increased.
iii) Overflow will be hidden	Content will not be trimmed out
iv) That flex will not be responsive	Flex will be responsive

flex

The flex property is a shorthand property for the flex-grow, flex-shrink & flex-basis properties.

Syntax

```
flex :  0     0     200px;
        ↓     ↓      ↓
      grow  shrink  basis
```

Align-self

speifies the alignment for selected items inside the flex container

The align-self property overrides the default alignment set by the container's align-items property.

Syntax

```
align-self: center | flex-start | flex-end;
```

CSS Grid

The grid layout module offers a grid-based layout system, with rows & columns, making it easier to design webpages without having to use floats & positioning.

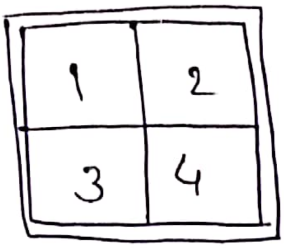
Grid vs Flexbox

CSS Grid layout should be used for 2-D layout, with rows AND columns.

CSS Flexbox layout should be used for 1-D layout with rows OR columns

Grid Elements

A grid layout consists of a parent element with one or more child elements.



1	2
3	4

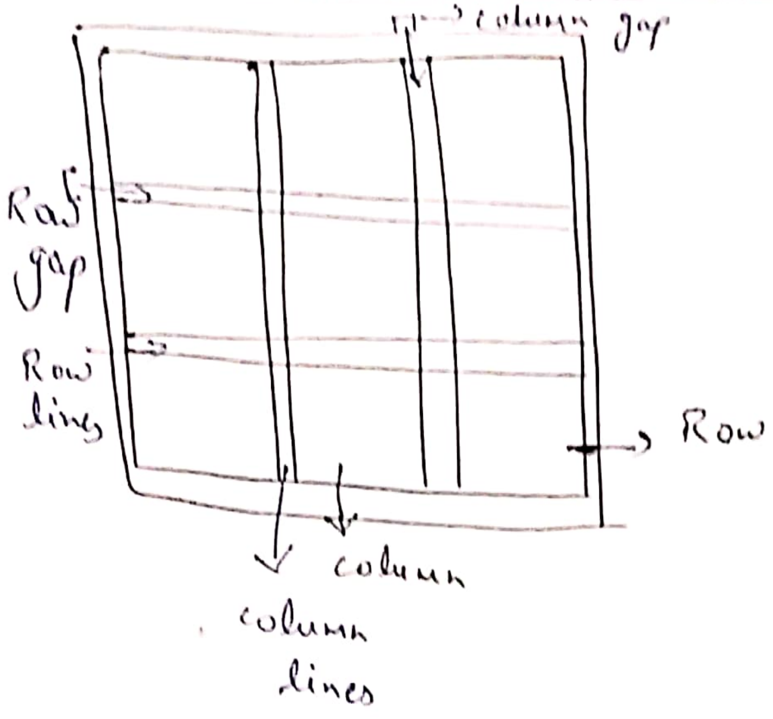
Display Property

An HTML element becomes a grid container when its display property is set to grid or inline-grid

Ex

```
.grid-container {
        display : grid | inline-grid;
}
```

You can adjust the gap size by using one of the following properties:
i) column-gap
ii) row-gap
iii) gap



Grid Container

Grid container is a parent container that consists of grid items (child container) in row and column format

grid - template - column Property

Defines no. of columns in your grid layout & it can define the width of each column

Ex display : grid;
grid - template - column : auto auto auto auto;

The rows & columns of a grid is defined with the grid - template - rows & the grid - template - columns properties (or the shorthand grid or grid - template)

Justify - content property

The justify - content property is used to align the whole grid inside the container

* space - evenly
* space - around
* space - between
* center
* start
* end

align - content property

The align - content property is used to vertically align the whole grid inside the container.

(Same as justify - content properties)

Grid Items

A grid container contains grid items

The grid - column Property

grid - column property defines on which ~~column~~ column to place an item

You ~~will~~ define where the item will start , and where the item will end

shorthand property for grid - column - start and grid - column - end

Ex .item {
grid - column : 1/5;   [column 1 start & before column 5 end]

Cyd

```
.item2 {
    grid-column: 2/ span 3;
}
```

[start on col. 2 and end ~~before~~ sol.
span 3 columns]

The grid-row property

Defines on which row to place an item
Shorthand property for grid-row-start &
grid-row-end

The grid-area Property

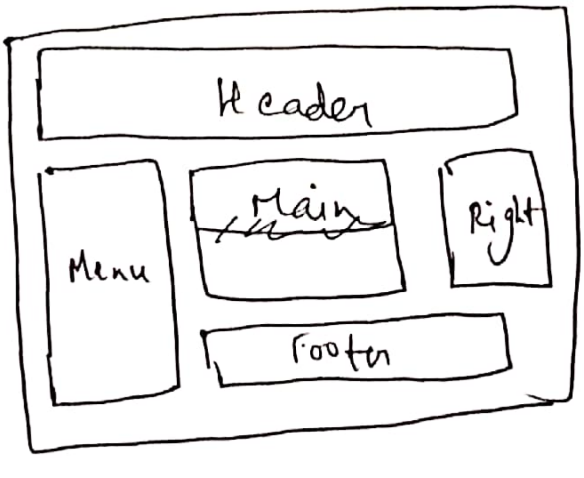
The grid-area property can be used as a shorthand property for grid-row-start, grid-column-start, grid-row-end, grid-column-end.

Ex
```
.item8 {
    grid-area: 1/2/5/ 6; -> col lin 6 end.
}
```
-> rowline 5 end

↓ row start 1

↓ col start 2

Naming grid items.



Naming grid items can be referred to by the grid-template-areas property of the grid container.

En grid-template-areas: 'myArea myArea myArea myArea';

Each row is defined by apostrophes (' ').

Advanced grid concepts:

1) fr unit

fractional unit.

Allows you to define grid tracks as fractions of the available space

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

2) repeat()

The CSS repeat() function is used to repeat a set of columns or rows in a grid

Syntax: repeat (repeat-count, tracks) specifies the set of cols or rows that will be repeated.

specifies the no. of times the cols or rows should be repeated

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

tracks: can be :→a length value (px, cm, %, fr, ch)

→min-content

→max-content

→auto

→minmax( )

→fit-content()

→named lines