HTML

* It is a skeleton of the web.
* Tags: < this is a tag > the thing with in the < > these two are known as tags
* Elements: <h1> </h1> collection of starting and ending of the tag is known as Element
* head: this contains the meta data of the web page or SEO of the page or title or style.css etc.

Types of tags

* semantic / meaning: - <h1, h2..h6> <a> <p> <img>
* structure: html head body section
* formatting: <b> is b strong <em> these tags do not have ending tag.

Types of Lists

* <li>: list items
* <ol>:ordered list
* <ul>:unordered list

DIV tag : wrapping a group of text under a parent so that we can modify later on we use div tag which means a division

Block vs Inline Tags :

* Block tags are those tags which can takes all the row and next line me ata Phir se likhe toh eg : <p>
* rowspan and Col span :it expands the row or col
* eg: rowspan = "2" :- it expands the rows till 2 cells

Forms in HTML:

we use label and Input to create Forms

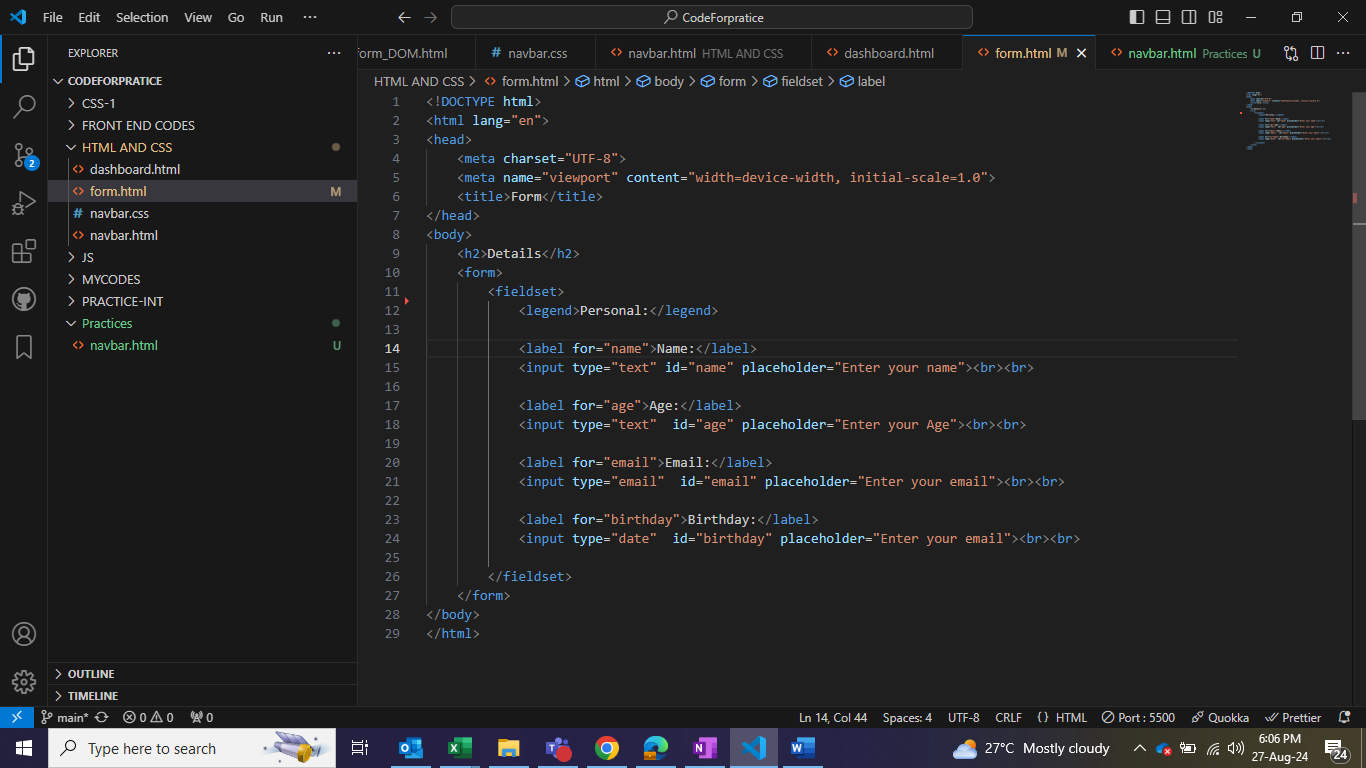
label 'for' and input 'id' should be same

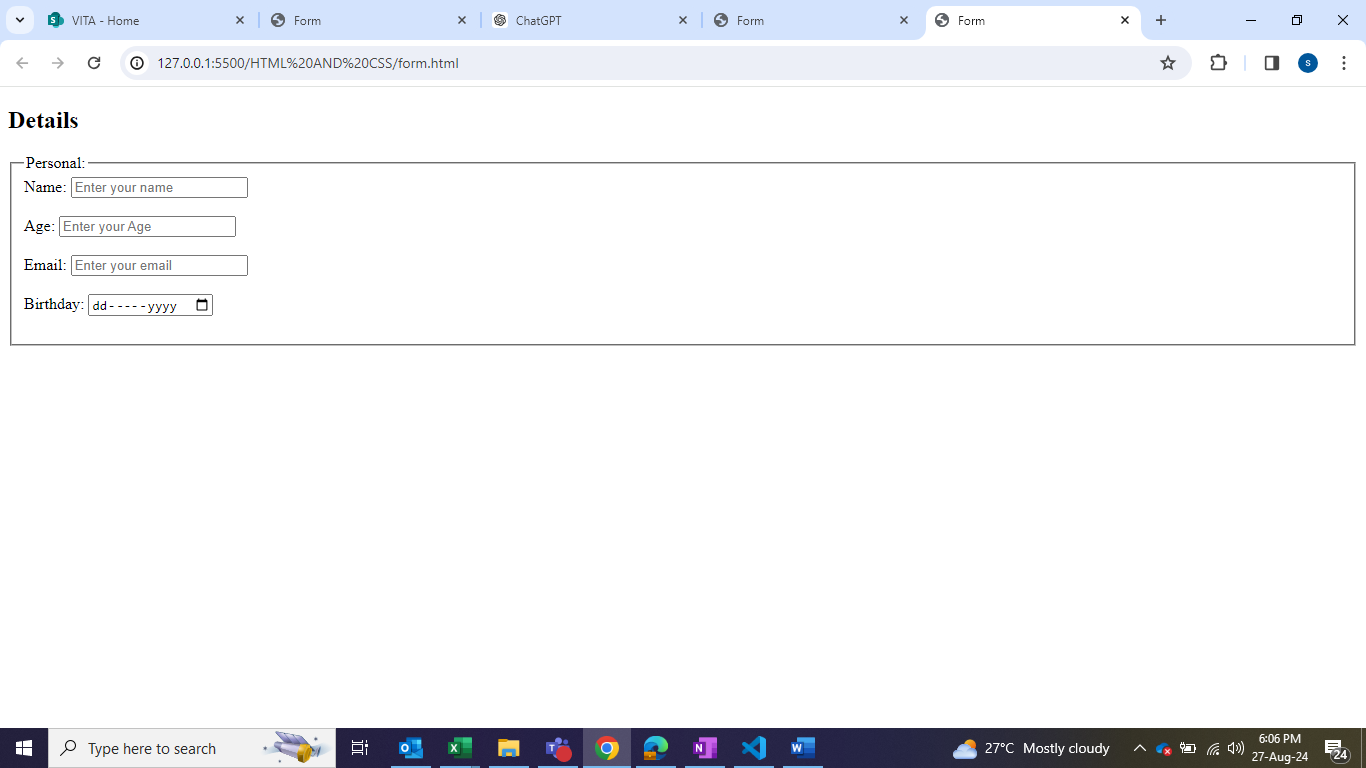
Place holder : faded text which will be there in the text box for example.com

Note : we use placeholder in input section

legend : we use it for upper heading

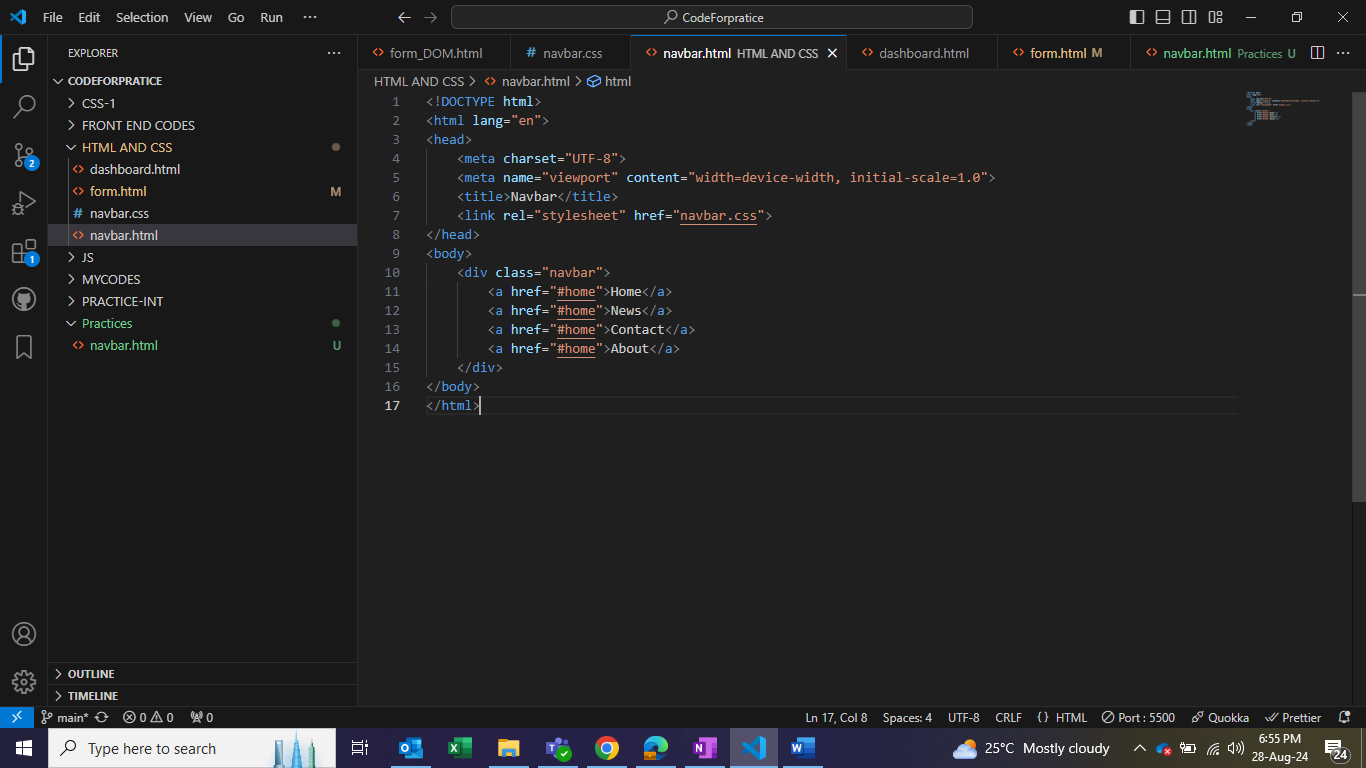
Field-set : It's a box around the content

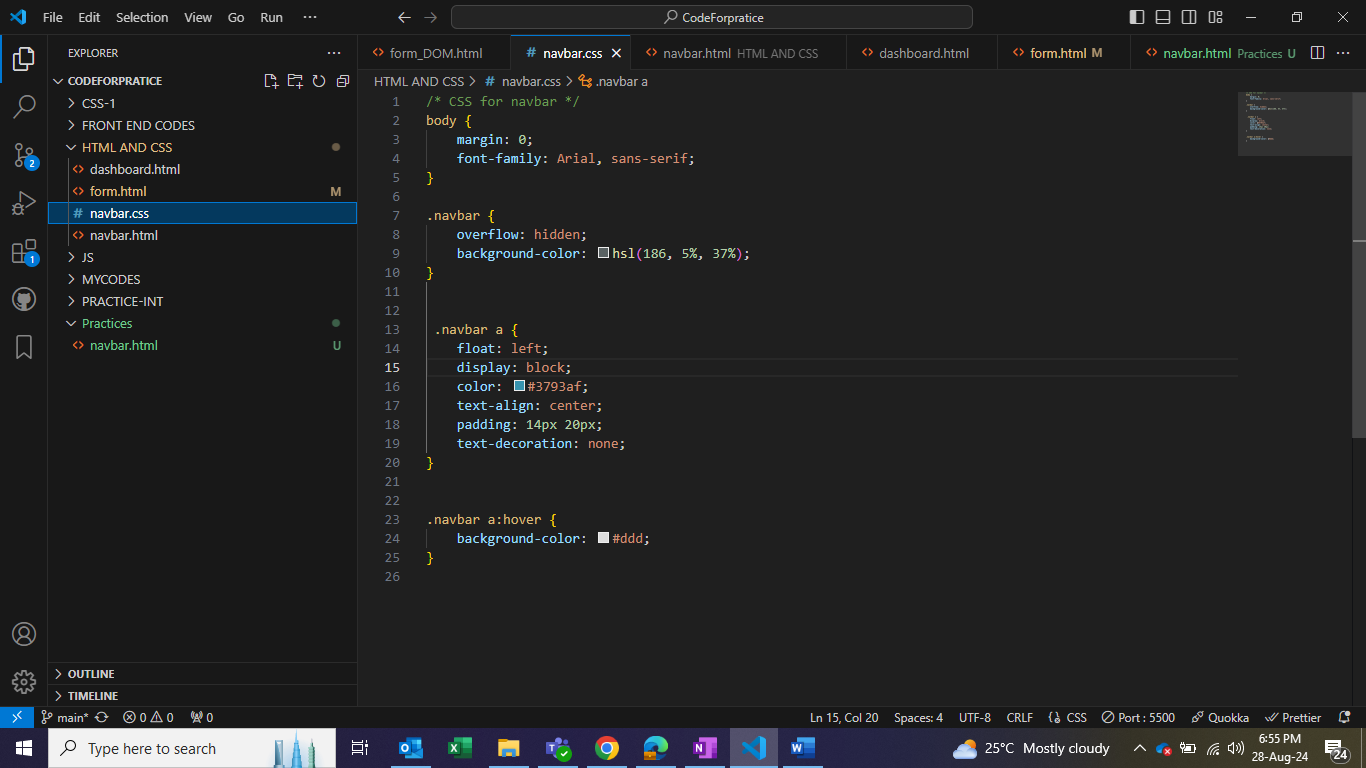




CSS

Creating Navbar





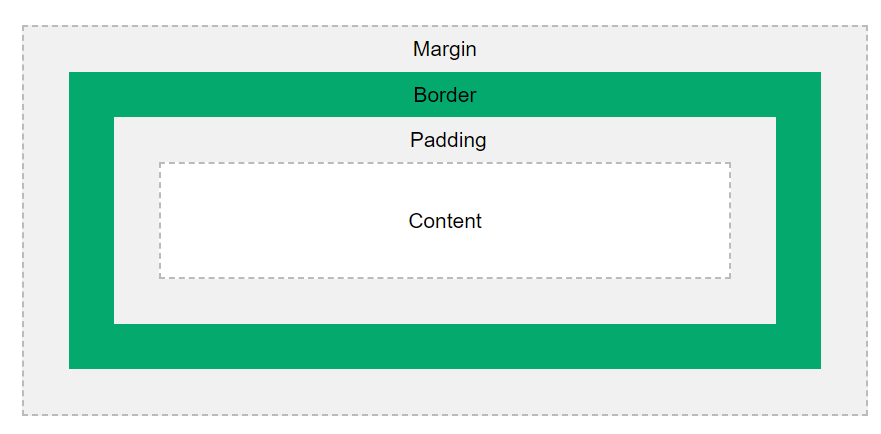
CSS Box Model : whatever we write it will take in the form of box and even images will be considered as box only

1. Margin : outer layer

2. Border : layer between Padding and Margin

3. Padding : layer between border and content

4. Content : a data which is in form of a box



Note: By default margin and padding should be 0 by using universal selector(\*)

Display Properties:

* Block Elements : these are the elements which takes the entire line but we can customize with css(margin , padding, width, height).

Eg: div , h1, p;

* Inline Elements : It take the space size of the element

Eg: span

Note: we can convert inline to block and vise versa by using (display : block or inline)

**Inline-Block** : these are the properties which are inline but we can customize.

**CSS Position Property:**

Static Position : When we define the elements it will be static.

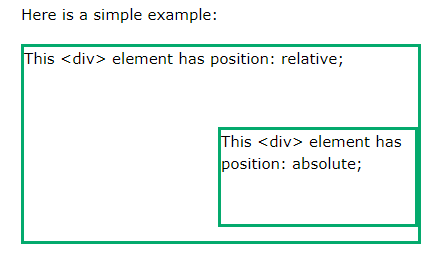
Fixed Position : It will be fixed even if you scroll down it will be on same position till the end

Sticky Position : It will be fixed even if you scroll down until its div is there after that it will disappear if we scroll

z-index: The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

Relative Position [Parent] : It will be applied on the current window(static) we use 4 properties for relative property [top, bottom ,left, right] and the space will be blank

Absolute Position [Child] : When we want to overlap we use absolute property and we mark the closest ancestor (div) as relative and the space will be overlapped



**Float and Clear** :

Float property is used where the items should flow, for example I need an img to be at right side I will use float:right the img will come on right side (just like in wikipedia)

Clear : It is used to clear the float img.

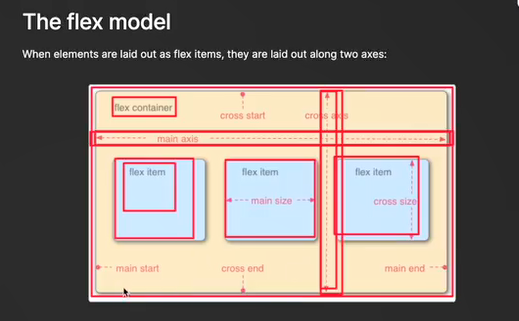
Eg: let's say there are two images img1 and img2 , will use clear : right the img 1 will come below the img2 because we tried doing clear the float on right side.

Overflow: this is nothing but when the data is exceeding the range of the parent tag

Eg : let's say there is a container along with border and the data is more than the container in that case we should use,

Overflow properties to manage the data

**Flex Box**: It’s a 1d layout method , layout means arrangements in the web page

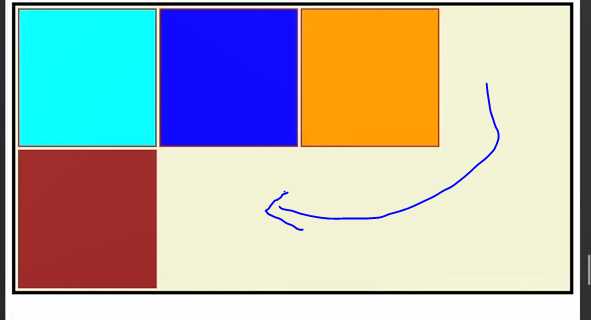


By default flex direction is **row** if we want in column use flex-direction we also have reverse.

Eg: flex-direction : row-reverse

flex-wrap: when we reduce the size of the screen the content size should not be get reduced so we use wrap.

flex-wrap: wrap



* Flex-wrap: nowrap : there will be no wrap
* Flex-wrap:wrap-reverse
* Note: It will not change the size of the content as long as space is available on the page. Last tak squeeze kardalo dubla hojata
* Flexflow : it is shorcut for flex-direction and flex-wrap
* Justify-content : it will modify the content based upon the flex-direction if the direction is row the main axis will be horizontal if its column then vertical.
* Justify-content - center : the content will come on the center based upon the axis.

There are other properties like :

: start :end :space-around :

:space-evenly this will give equal space

* Align-items: it will modify the content based upon the cross-axis , it has properties like center, stretch, start, end etc
* Eg: flex-direction is row then cross-axis will be vertical (opposite ino ) if the flex direction is column then cross axis will be horizontal.

Note : to bring the content on center of the window use align-item : center and justify-content : center (both are opposite to each other axis wise)

* Flex-direction: row (justify-content ka center abb horizontal and align-items ka vertical)
* Align-content : it will modify the space between the contents, it has properties like center start, end, space-between, space-around
* Flex-shrink: when we reduce the size it will shrink which means reduce it fast compare to other items(eg:boxes)
* Flex-shrink:2
* Flex-grow: It will increase the size of the item
* Flex-grow:4
* Flex-basis: it will increase the dimensions or width
* Flex-basis:300px
* Flex-shorthand ?
* Align-self : This property specifies the alignment in the block direction for the selected item inside a flexbox or grid container based on cross axis
* Align-self: center

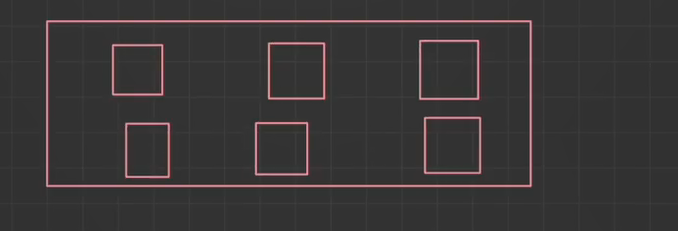
Media Queries :



**Grid** : It’s a 2 dimensional layout

Row-gap & column-gap : this property gives gaps in rows and columns

For 2d structures like this

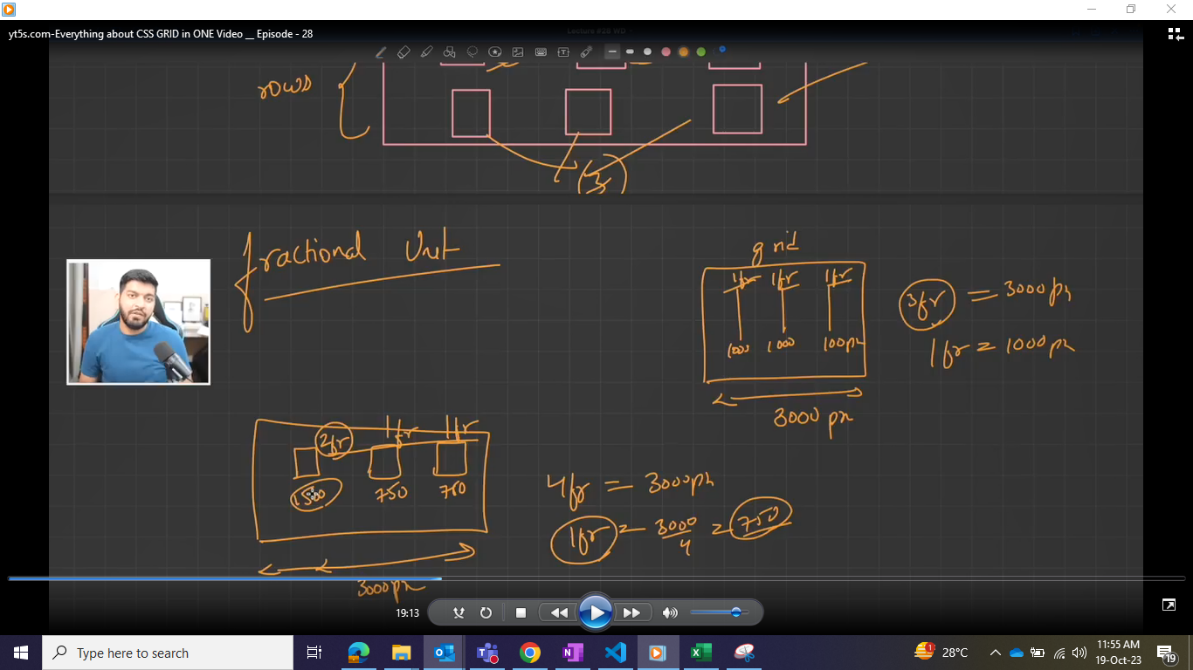


We need to define the rows and columns for this

Grid-template-rows : 1fr 1fr // columns or rows me kitte space hona

grid-template-column : 1fr 1fr 1fr

Note: we can also use px but fr is fractional unit we use this to make our website responsive



Note 2: if there are so many boxes we cannot use 1fr 1fr….. It takes time so we have repeat

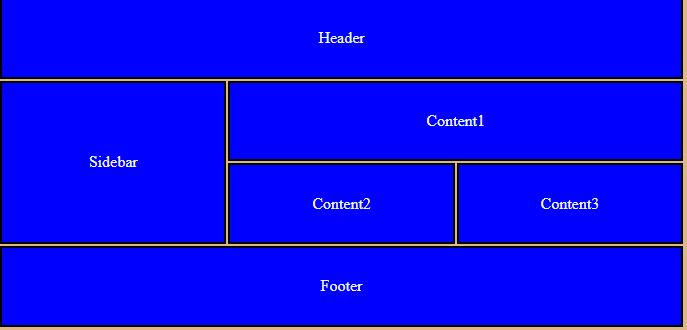
Grid-template-row: repeat(2,1fr) this is repeat 2 times 1fr same with column

Note: when we use grid it will be block grid

Difference between block grid and inline-grid

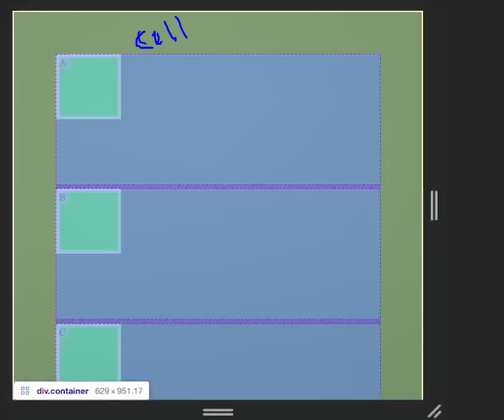
Grid-row-start & grid-row-end || grid-column-start & grid-column-end : this is used to expand the size of the boxes in grid in row wise same with column as well .

The following is created by using above properties eg : header grid-column-start: 1 & grid-column-end: 4

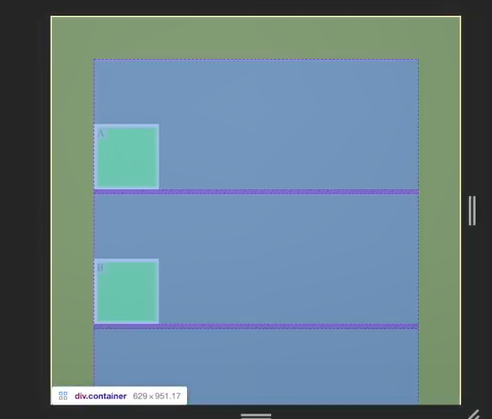


Justify-self or justify-items : it is used to move the items within the cell in horizontal direction

Eg : start, end , stretch

n

Align-self or align-items: it is also used to move the items with in the cell but in vertical direction



Place-self or place-items: it’s a combination of both align and justify self both vertically and horizontally.

Tailwind Css

Go to PostCSS method 🡪

* npm install -D tailwindcss postcss autoprefixer
* npm i vite
* npx tailwindcss init -p
* Go to Tailwindconfig.js 🡪 content[“\*”]
* Go to package.json 🡪

 "scripts": {

    "start": "vite"

  },

Create a .css file and add the below files init

@tailwind base;

@tailwind components;

@tailwind utilities;

Add .css file in html file

<link rel="stylesheet" href="main.css">

Properties of tailwind :

To change the color :

bg-green-200 text-red-600

* p-2 : padding 2
* mt-2 : margin top 2

Note : suppose if we want to write our own side eg : px then we have to choose brackets eg : p-[25px]

Height and width

* h-22 or h-1/2
* w-12

Margin

* (Margin-left) ml - 1 (means 0.25 rem from left)
* mr
* mt
* Mb
* M -4 (margin from all 4 directions)
* Mx (left to right)
* My(top to bottom)

Padding

Same as margin

* pr, pt, pb, pl, p, px, py

Borders

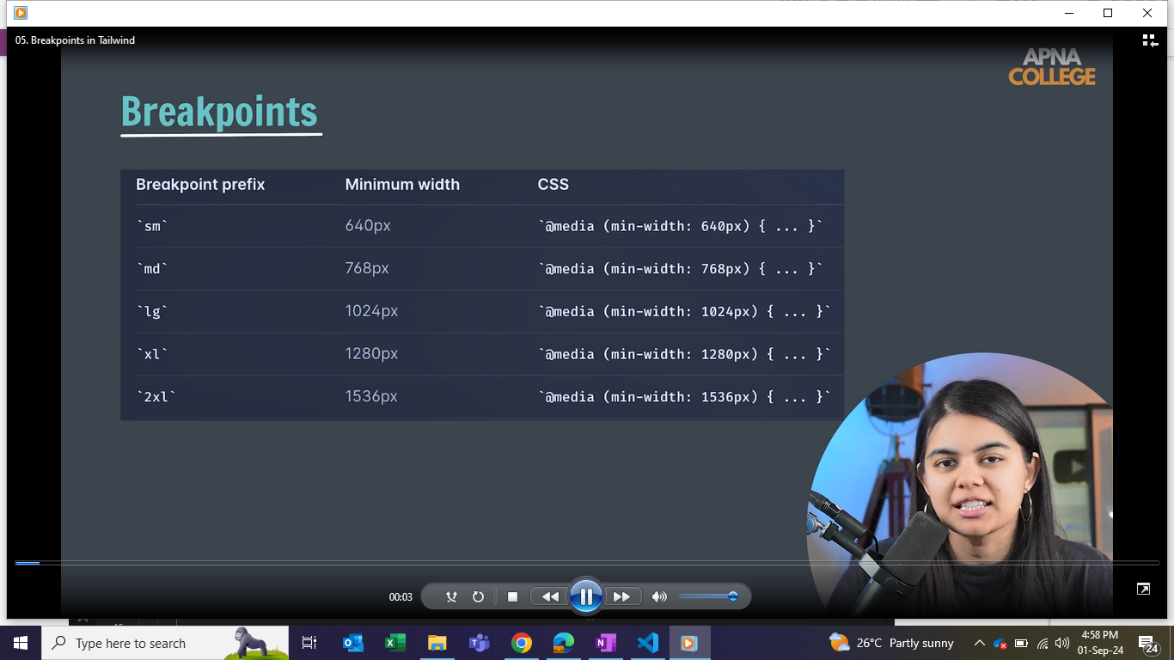
* Border-0 (means 0px border)
* Border-10(10px border)
* Border-black(black-border)

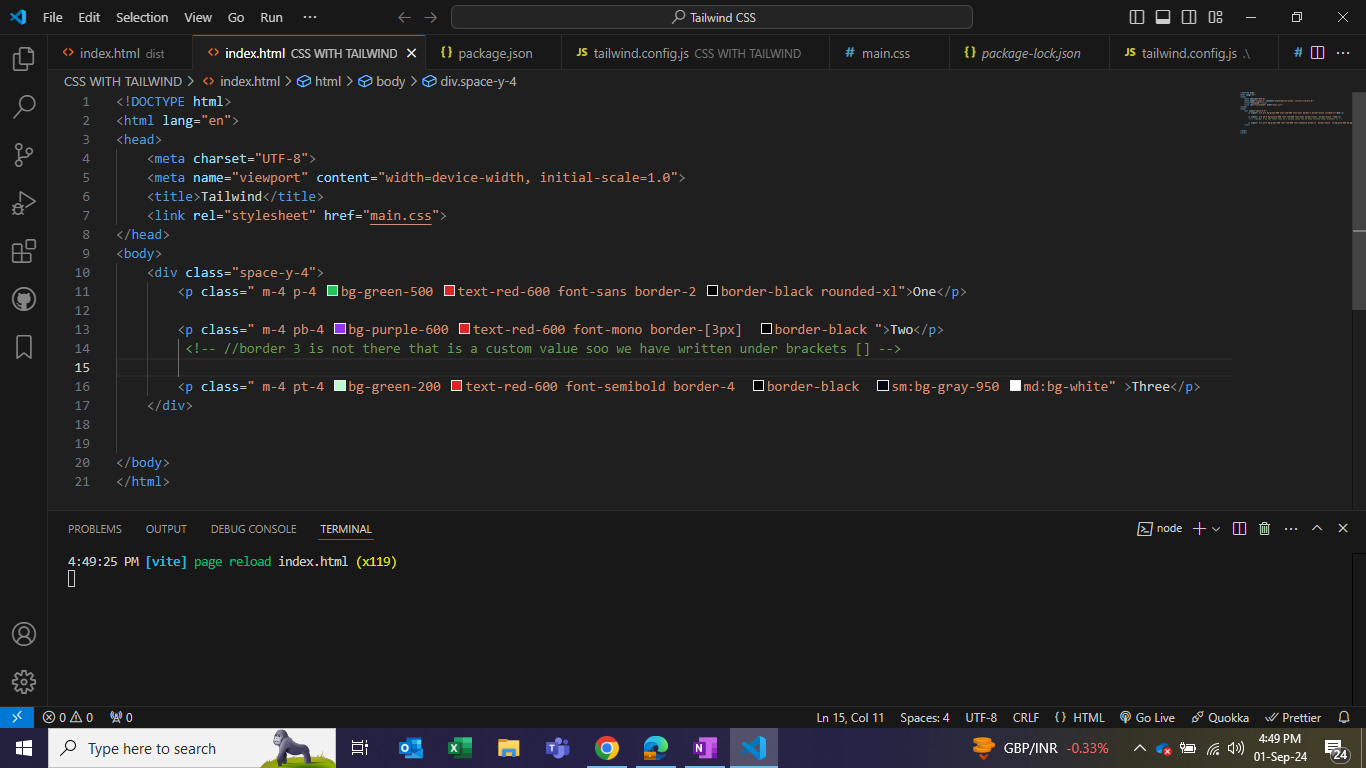
Note : Always remember when we are giving the space for vertical elements to come in horizontal direction it will be reflected from 2nd element not from first element, because the first element will be on top if we give margin top from 2nd element it will be reflected space-x-4 (1 rem from left to right)

Custom values [ ]

* It's means in tailwind all the values with proper units are not available we have to write by our own by using the brackets
* Eg : [h-25px] [h-12rem] we write brackets with units It will create its own custom class

Breakpoint / media queries

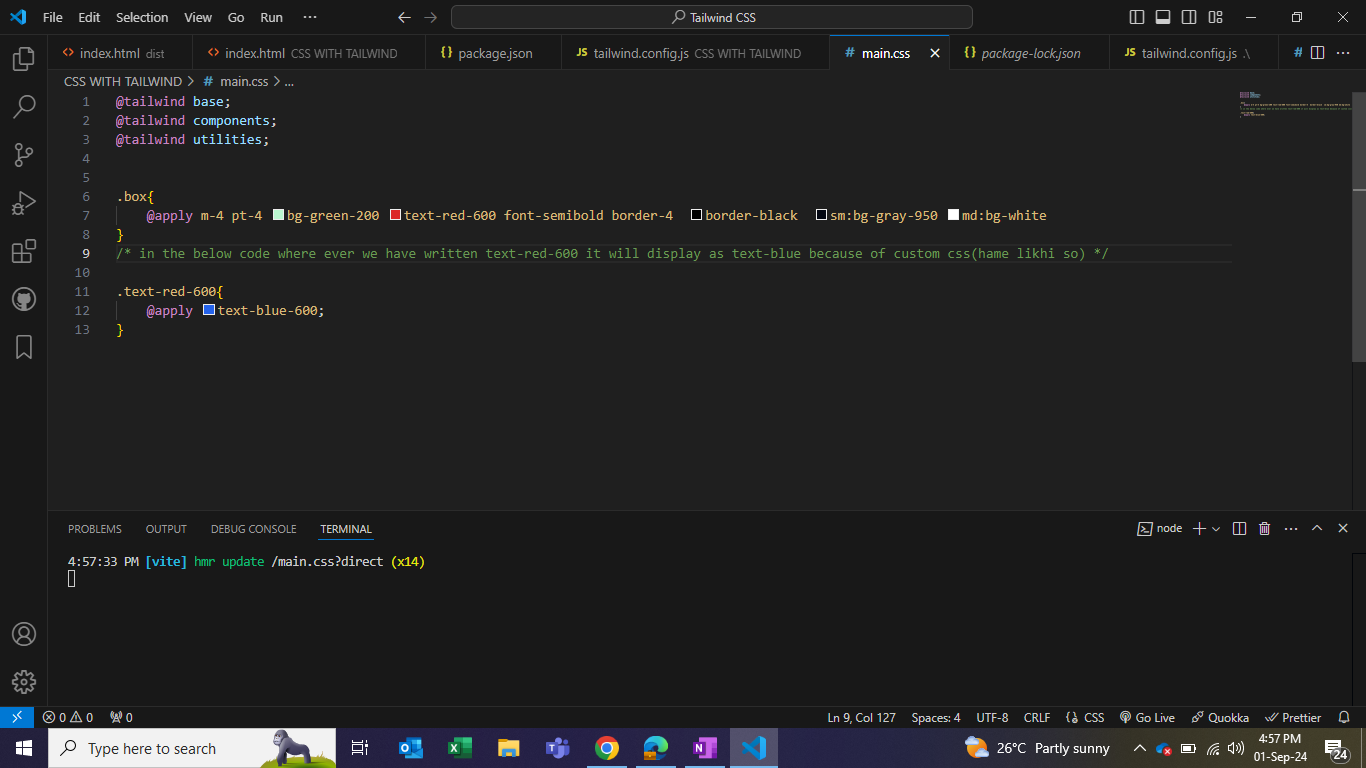




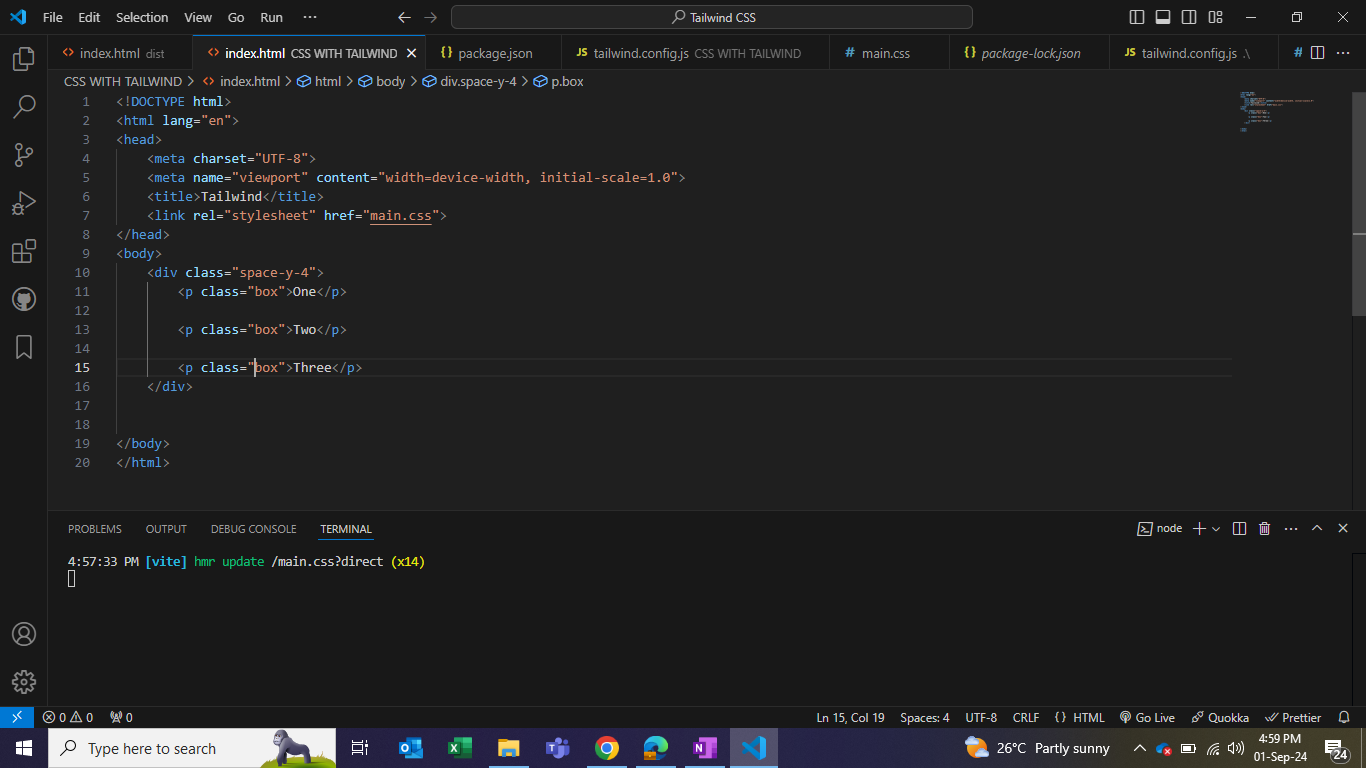
Apply directive

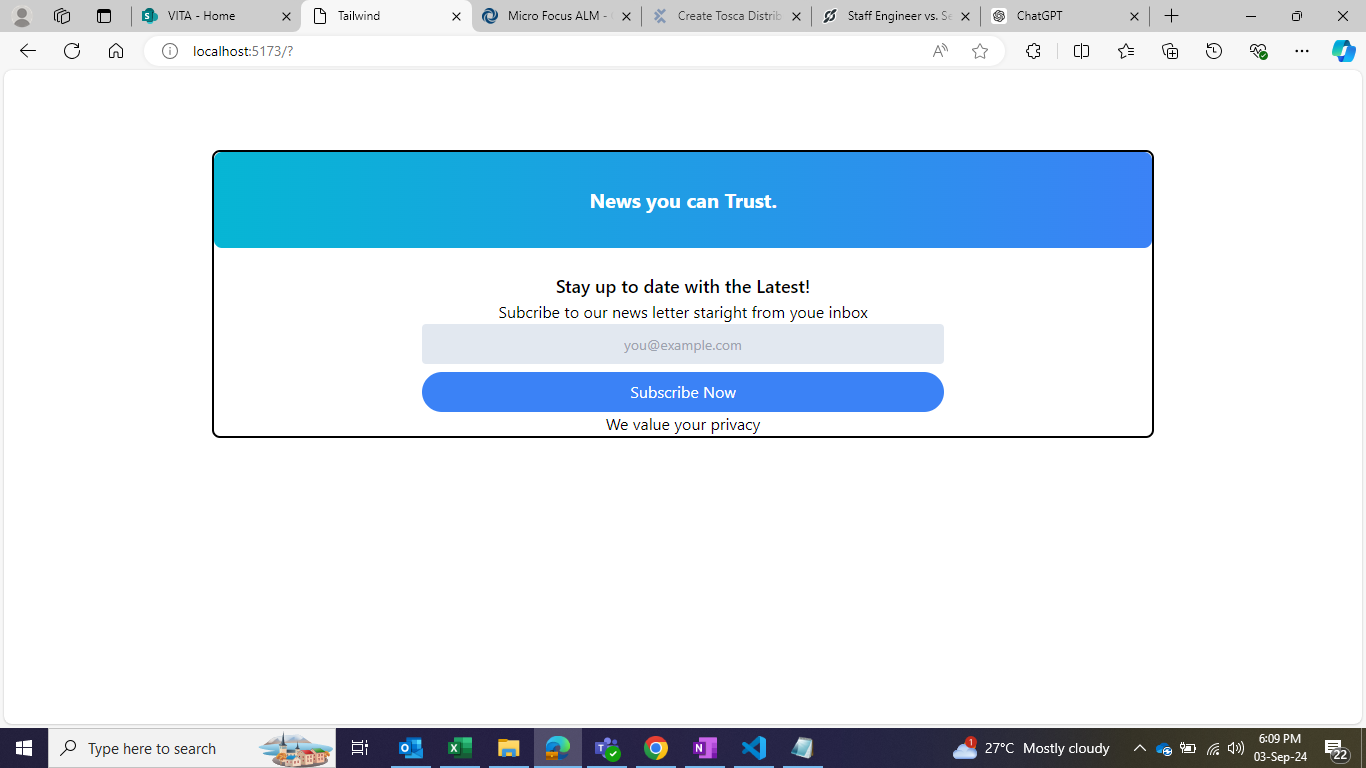
* When there is a common css (above all the 3 lines has common css soo we can write in the separate css file by using @apply

In CSS we can



Since we have written Tailwind CSS in separate css file here will write only html with class



Project using Tailwind

