Tailwind css basics

Width – w

w-full 2= 0.5rem

Height – h

They both accept digits

My – is margin on vertical axis (top & bottom)

Mx – is margin on horizontal axis (right&left)

P - padding

Rounded-md = border radius

Instead of using predefined utilities to use custom we use [eg 50px] : text-[50px]

LAYOUT

THISIS THE ARRANGEMENT AND POSITIONING OF ITEMS IN A WEBSITE

Most of the time youl be using dispay potion and overflow

POSITION – Determines how a html element is positioned within its containing element or overall website

- Relative – shifts a div from its normal spot but everything else behaves like its still there
- Absolute- make the element move independently like a puzzle piece based on a nearbly parent
- Fixed- element remains in one position even after scrolling
- Sticky – element acts normal but element stay at a point at a point

DISPLAY- Determines how the element behaves in terms of layout and visibility within the document

Controls how elements are displayed

- Block
- Inline
- Inline block
- Non
- Flex
- Grid

MEDIA QUERIES

- Max-sm – min width of 640px
- Max-md – min with of 768px
- Max-lg – min width of 1024px
- Max-xl – min width of 1280px
- Max-2xl – min width of 1536px

When it comes to hover just add a hover in the class and add the action eg hover:bg-blue-700

Same as to focus as hover

Config especially extend is used to predefine utils adding on to existing one

INSTALLATION

1. Npm install tailwindcss @tailwindcss/vite
2. Configure vite plugin

GLASSMORPHISM

Effect	Tailwind Class
Semi-transparent	`bg-white/10` or `bg-gray-200/20`
Blur effect	`backdrop-blur`, `backdrop-blur-sm` / `md` / `lg`
Border	`border`, `border-white/20`
Rounded corners	`rounded-xl`
Shadow	`shadow-lg`, `shadow-xl`

NB: TAILWIND IS MOBILE FIRST

This means that in every thing start up with mobile design moving upwards

Flex-1 makes things of equal size

::before and :: after pseudo elements

They let you insert content before and after the actual content without adding extra html elements

Its uses:

- Add decorative elements like lines, icons and badges
- Custom dividers mostly vertical lines
- Animations and tooltips

Properties used

Property	Purpose
`content`	Required! Adds content (text or empty)
`position`	Often set to `absolute` for control
`width`, `height`	Defines size (for lines, shapes, etc.)
`background`	Adds color or gradients
`top`, `left`	Positioning relative to parent
`display`	Often `block` or `inline-block`
`z-index`	Controls layering

GRID

Works just like a chess board where by you can place the items wherever you like

To create grid:

```
.container {
  display: grid;
}
```

To define Columns and rows:

```
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
  grid-template-rows: 100px 100px;
}
```

This create 3 columns with each being 200px wide and 2 rows each being 100px tall

Instead of px we can use fr which is a fraction of the container- it divides into 3 equal parts

Repeat() – It is used to avoid repetition in code

grid-template-columns: repeat(3, 1fr); means the same us **grid-template-columns: 1fr 1fr 1fr;**

It takes in 2 queries- repeat(number of cols/rows, size)

Positioning Items(colums)

Across

.box1 {

grid-column: 1 / 3;

}

What this does is place the container from column 1 to where column 2 ends and 3 begins

Vertically(rows)

.box2 {

grid-row: 1 / 3

}

Spans the container from row 1 to where row to ends and three begins

@apply

It is used to define a large style that can be reused in dif components

```
.services{
  @apply w-100 h-60 bg-white/10 backdrop-blur rounded-xl shadow-sm
}
```

@theme

Used define your own utilities