**TAILWIND CSS**

**Tailwind CSS** is a utility-first CSS framework. That providing a set of predefined utility classes and also we can customized the predefined utility classes according to your requirement. But unlike other CSS frameworks like [Bootstrap](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)" \o "Bootstrap (front-end framework)), it does not provide a series of predefined classes for [elements](https://en.wikipedia.org/wiki/HTML_element" \o "HTML element) such as buttons or tables.

**Note**

Benefit of using tailwind CSS that we easily style or customize our HTML element without leaving html file with the help of the classes. Not like Bootstrap (Bootstrap add a lot of extra code and file to your website which can effect it’s performance and loading speed suppose we create a div container and use Bootstrap classes it automatically add display-flex property that is not required for container but tailwind CSS Dose not add extra things)

**CUSTOMIZING YOUR CONFIGURATION FILE**

If you want to change things like your color palette, spacing scale or breakpoints, add your customization to the them section of your tailwind.config.js file

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

  theme: {

screens: {

// we add our classes (over ride the previous utilities classes )

      sm: '480px',

      md: '768px',

      lg: '976px',

      xl: '1440px',

    },

    colors: {

      'blue': '#1fb6ff',

      'pink': '#ff49db',

      'orange': '#ff7849',

      'green': '#13ce66',

      'gray-dark': '#273444',

      'gray': '#8492a6',

      'gray-light': '#d3dce6',

    },

    fontFamily: {

      sans: ['Graphik', 'sans-serif'],

      serif: ['Merriweather', 'serif'],

    },

extend: {

// we jus add some classes in previous utilities classes

      spacing: {

        '128': '32rem',

        '144': '36rem',

      },

      borderRadius: {

        '4xl': '2rem',

      }

    }

  }

}

**Applying Conditionally**

CSS frameworks allow you can use utility classes to quickly apply to your elements conditionally. like

* Hover
* Focus
* Breakpoints.

**Breakpoints**

Every utility class in Tailwind can be applied conditionally at different breakpoints, which makes it a piece of cake to build complex responsive interfaces without ever leaving your HTML.

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

  content: ["./src/\*\*/\*.{html,js}"],

  theme: {

      // Breakpoints for every screen size

        screens: {

         // These are the default breakpoints and The default breakpoints are inspired by

common device resolutions………………………………………………………………………………………………………………………………………

          'sm': '640px',

          'md': '768px',

          'lg': '1024px',

          'xl': '1280px',

          '2xl': '1536px',

         // To completely replace or override the default breakpoints…………………………………………………

              exs:'450px',

              sm: '680px',

              md: '960px',

              lg: '1024px',

              xl: '1240px',

          },

         // To override a single breakpoints (like lg) or we can add extra breakpoints

        extend: {

         // override a single breakpoints………………………………………………………………………………………………………

            lg:"1110px",

         // Add extra breakpoints……………………………………………………………………………………………………………………………

            maxScreen:'1730px'

    },

  },

  plugins: [],

}

**Way to define breakpoints for different screen size**

* Alternate way

If you define md:text-red breakpoints in that case after the md text follow md

properties till the full width of view port

<div class=" exs:text-green-600 sm:text-red-500 md:text-yellow-300 lg:text-blue-700 xl:text-pink-600">

* Between

If you define esx:max-md:text-red breakpoints in that case after the esx and befour

the md breakpoint text follow esx properties till the full width of view port

<div class="exs:max-sm:bg-pink-500 sm:max-md:bg-red-500 md:max-xl:bg-black md:max-xl:text-white" >

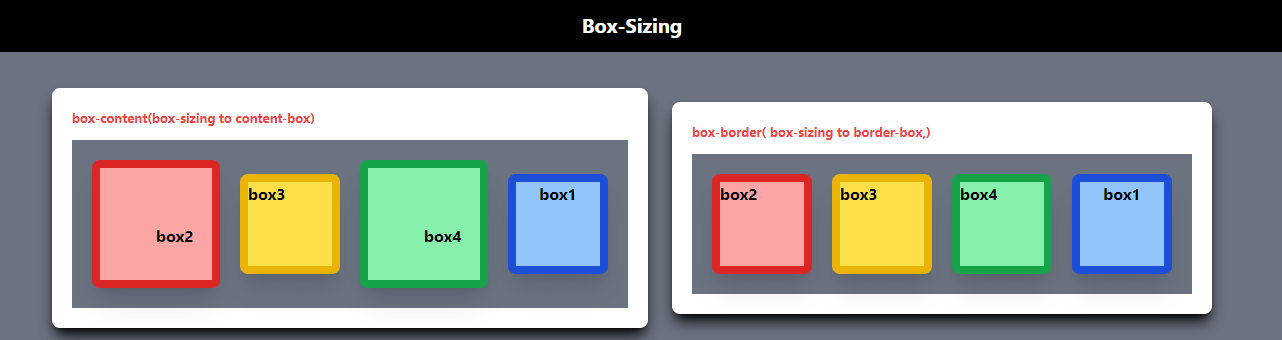
**BOX-SIZING**

* **box-content(box-sizing  to content-box)**

When you give a div a specified width and height and then add padding or border, the total size of the div increases because the padding is added to the specified width and height. This is the default behavior in CSS, known as content-box box-sizing.

**box-border(box-sizing to border-box)**

When using box-sizing: border-box, the specified width and height of an element include the padding and border. This means that the total size of the element will not exceed the width and height values you set, even when padding and border are added.



**Display Properties**

**What is inline elements in HTML?**

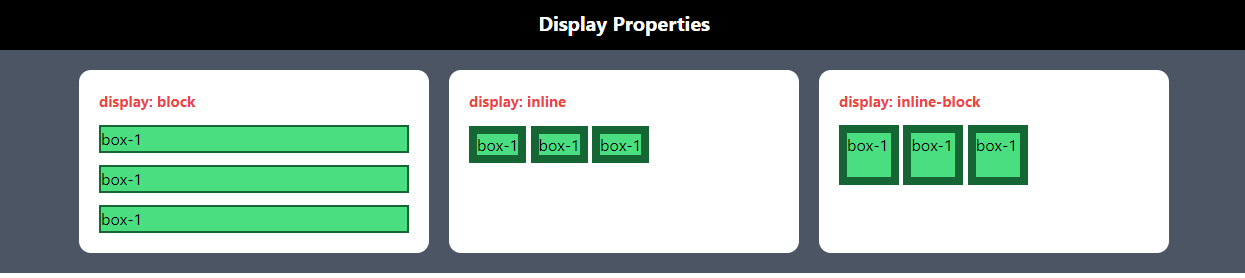
* Inline elements are those elements who take only as much width as necessary of view port that do not start on a new line .
* Examples of inline elements include <span>, <a>, <strong>, <em>, <img>, and <br>
* Inline elements do not typically accept width, height, margin, or padding properties.er, you can apply some styles like font size and text decoration.

**What is block elements in HTML?**

* Block elements are those elements who take the full width of view port that start on a new line .
* Examples of block-level elements include <div>, <p>, <h1> to <h6>, <ul>, <ol>, <li>, <table>, and <form>.
* Block-level elements can accept width, height, margin, padding, and other positioning properties.

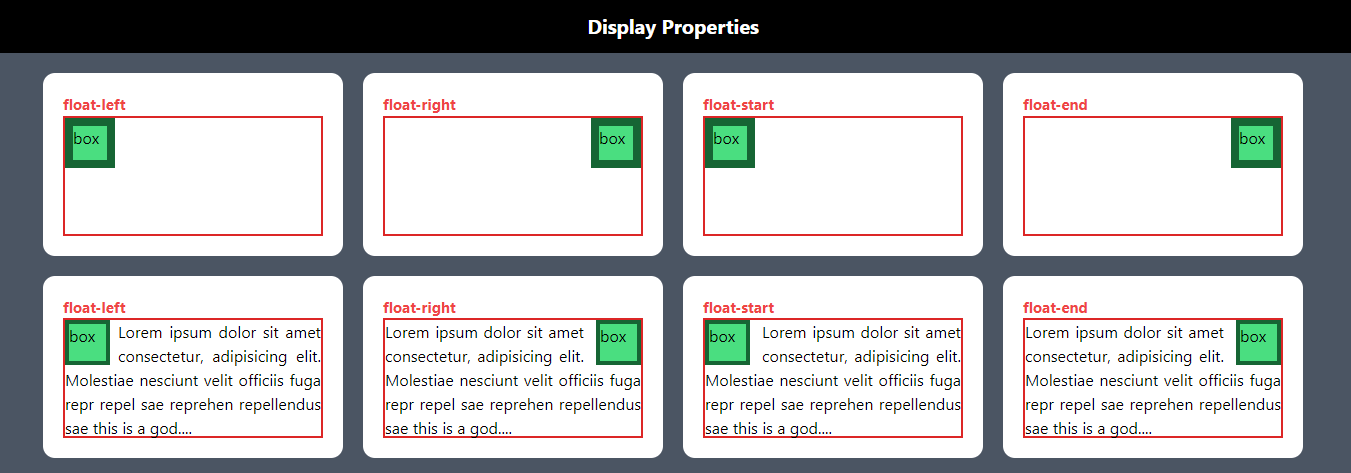
**What is inline-block elements in HTML?**

* **Display**: The display: inline-block; property makes the div elements behave like inline elements, so they sit next to each other in a row.
* **Width and Height**: You can set the width and height of inline-block elements, which is not possible with purely inline elements.
* **Margins and Padding**: You can also set margins and padding on inline-block elements, giving you more control over spacing.



**Float Properties**

Utilities for controlling the wrapping of content(paragraph) around an element(image.)

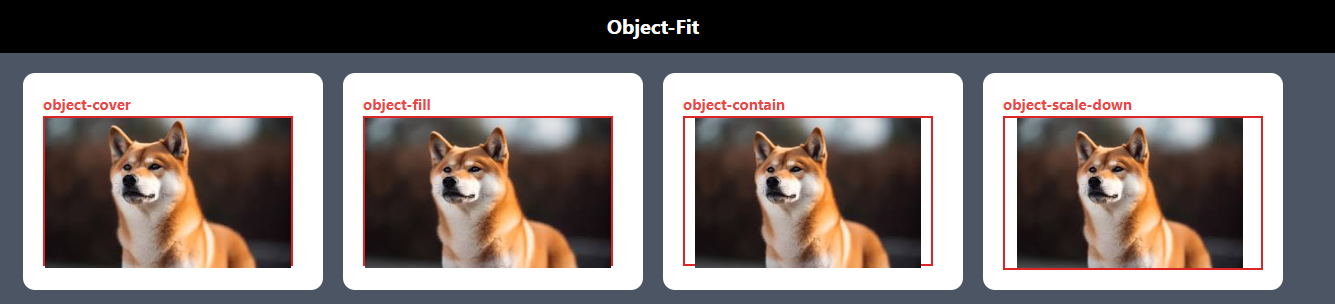


**Object-FIt**

Suppose you create a container and give it some width and height and put image or video inside the container in that case this image not take container height and width it take own height and width whatever you provide of image tag.

We use some CSS properties for maintaining the image expect ratio as well as cover height and width of that image whatever we provide. If we not use this CSS properties also cover the width and height but it not maintain the except ratio

* **object-contain**
* **object-cover**
* **object-fill**
* **object-none**
* **object-scale-down**



**Position**

# The position Property

The CSS position values decide how an element is positioned in a document. And the CSS position properties determine the final location of positioned elements.

There are five different position values and four different position properties ( Top, Right, Bottom and Left) **value**

* static
* Relative
* absolute
* fixed
* Sticky

1. **position: static** **:-** HTML elements are positioned static by default .Static positioned elements are not affected by the top, bottom, left, right and Z-Index properties.
2. **position: relative** **:-**We can positioned any element corresponding to the current window using top, right, bottom and left properties. And these properties determine the final location of positioned elements.

Note

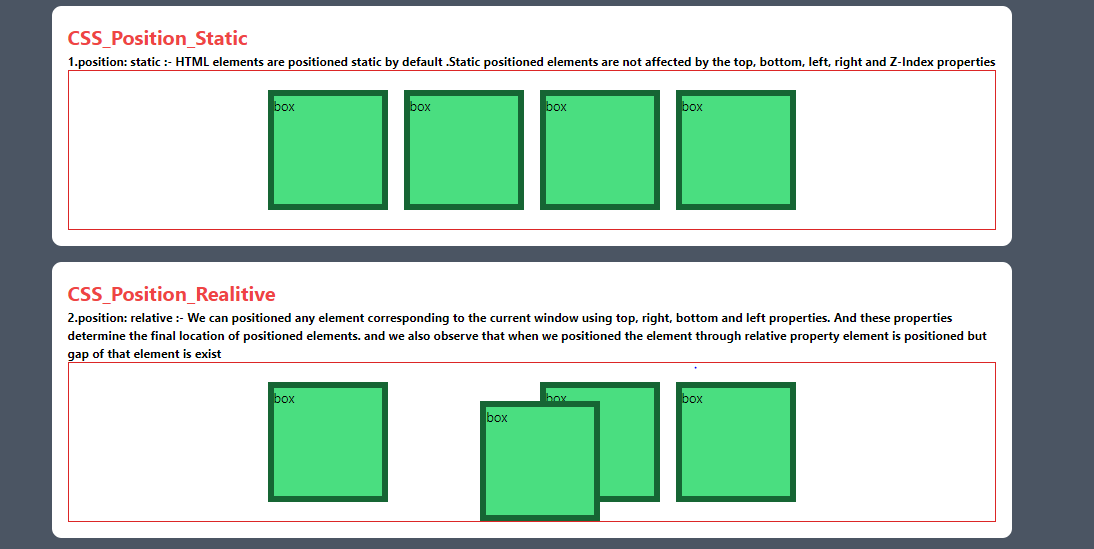
When we positioned any element with the help of relative value old positions are maintained.

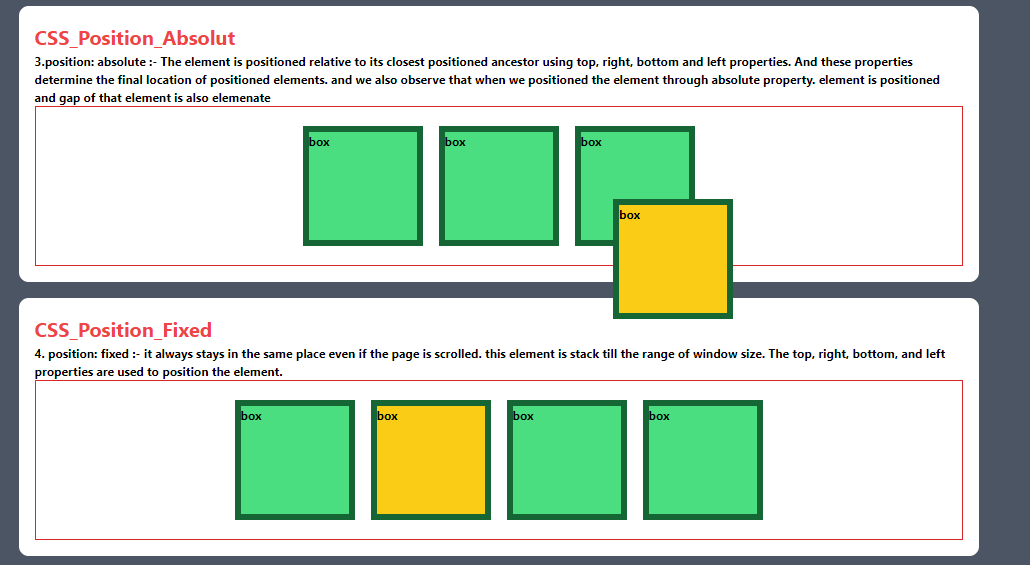
1. **position: absolute** **:-** The element is positioned relative to its closest positioned ancestor(inside the parents ) using top, right, bottom and left properties. And these properties determine the final location of positioned elements.

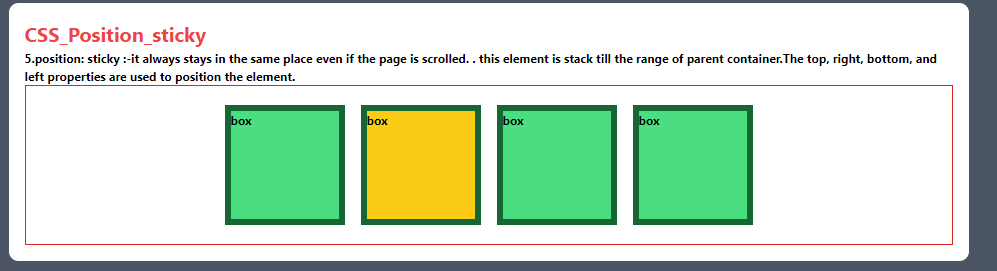
Note

When we positioned any element with the help of absolute value old positions are not maintained.

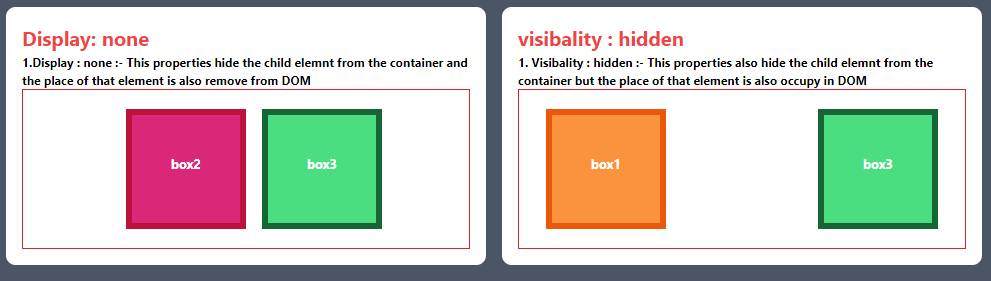
1. **position: fixed** **:-** this element is stack till the range of window size. The top, right, bottom, and left properties are used to position the element.
2. **position: sticky** **:-** It is used to stack any element . this element is stack till the range of parent container





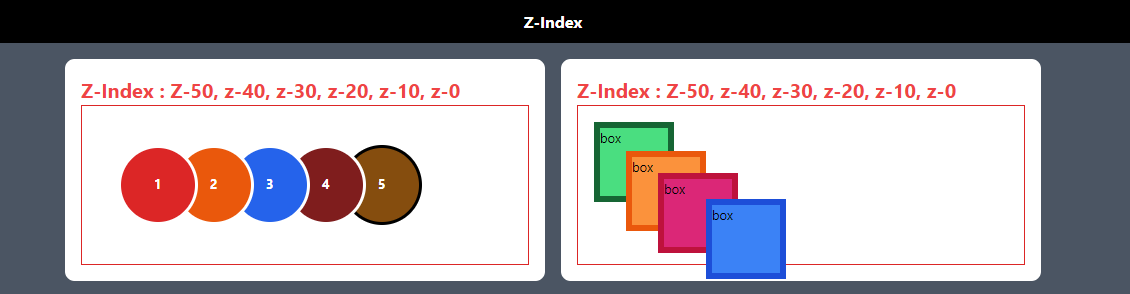


**Difference between Display none and Visibility hidden properties**

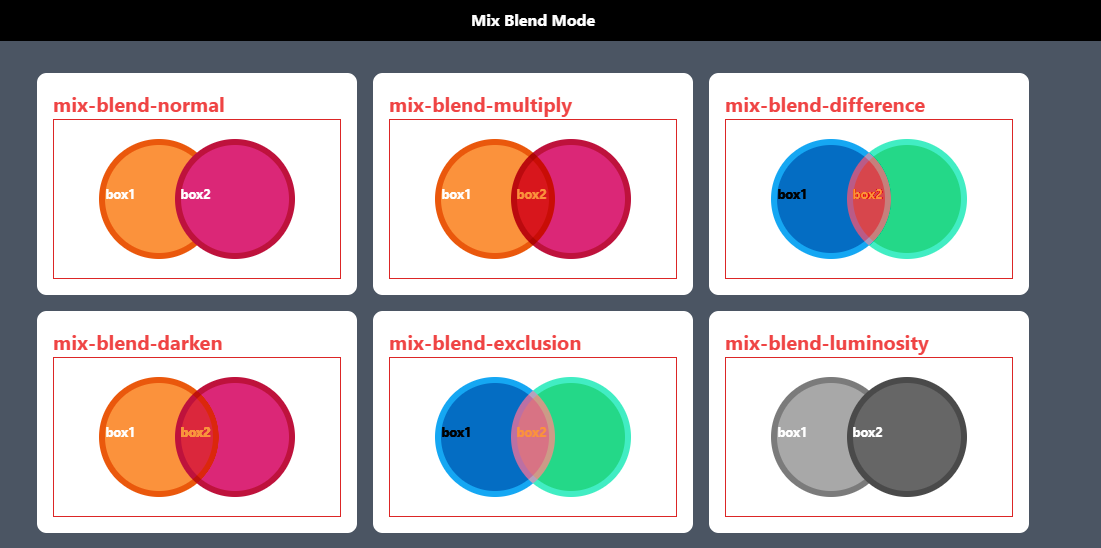


1. **Index**

Utilities for controlling the stack order of an element.



**Mix Blend Mode**



**TYPOGRAPHY**

**Height.**

**h-screen : it takes full hight of viewport**

**Width**

**Font Family.**

* font-sans
* font-serif
* font-mono
* **Applying conditionally**

CSS frameworks allow you can use utility classes to quickly apply specific font families to your elements conditionally. like Hover,focus, and Breakpoints.

 <p class="font-sans hover:font-serif"> Use Hover effect  </p>

 <p class="font-sans md:font-serif"> Use Breakpoints for changing the font families according to screen size</p>

* **We can customizing your font families**

**Font Size with Line-height**

* text-base: set the text size to base
* text-xs : set the text size to extra-small
* text-sm : set the text size to small
* text-lg : set the text size to large
* text-xl : set the text size to extra-large,
* text-2xl : set the text size to 2 times extra-large
* text-3xl : set the text size to 3 times extra-large
* Applying conditionally
* Set an element’s line-height at the same time you set the font size by adding a line-height modifier to any font size utility

<p class="text-base/6 max-w-[600px] border">// here 6 define line height

<p class="text-sm/[17px] max-w-[600px] border">// we set custom line height in px.

**Font Smoothing**

* **Antialiased**
* **subpixel-antialiased**

<p class="subpixel-antialiased max-w-[700px] border m-8">

* Applying conditionally

**Letter Spacing**

* Tracking-tighter
* Tracking-tight
* Tracking-normal
* Tracking-wide
* Tracking-wider
* Tracking-widest
* Applying conditionally

**Line Clamp**

=> Suppose you write a paragraph inside a div and this paragraph contain 6 line and you want to show only 2 line when size of div is 400px . we can use line-clamp-2 . when you expend div container expend your para automatically.

**Line Height**

**Relative line-height**

leading-none

* leading-normal
* leading-tight
* leading-relaxed
* leading-loose

**[Fixed line-heights](https://tailwindcss.com/docs/line-height" \l "fixed-line-heights)**

<p class="leading-6 ...">So I started to walk into the water...</p>

<p class="leading-7 ...">So I started to walk into the water...</p>

<p class="leading-8 ...">So I started to walk into the water...</p>

* Applying conditionally

**List Style Image**

   <ul class="list-image-[url(checkmark.png)] ...">

        <li>5 cups chopped Porcini mushrooms</li>

        <!-- ... -->

    </u>

**Text Align**

* Text-left
* Text-center
* Text-right
* Text-justify

**Text Transform**

* Uppercase
* Lowercase
* Capitalize : it convert the first latter of each word in capital.
* normal-case

**Text overflow**

* Truncate
* text-ellipsis
* text-clip

**Text Wrap**

* text-wrap
* text-nowrap
* text-balance
* text-pretty

**Text indent**

Use the indent-n utilities to set the amount of empty space (indentation) that’s shown before text in a block.

* indent-1
* indent-2
* indent-46

**SPACING**

Padding

Use the pt-\*, pr-\*, pb-\*, and pl-\* utilities to control the padding on one side of an element.

Margin

Use the mt-\*, mr-\*, mb-\*, and ml-\*utilities to control the margin on one side of an element.

Space Between

Use the space-x-\* utilities to control the horizontal space between elements.

Use the space-y-\* utilities to control the vertical space between elements.

**TAILWIND CSS**

1. **GRID IN TAILWIND**

Grid Template Columns:

* Utilities for specifying the columns in a grid layout. Means we can create the columns according to screen size .
* Use the grid-cols-n utilities to create grids with n equally sized columns.

Like

* Grid-cols-1 : This class is used to create a single column in row.
* Grid-cols-2 : This class is used to create a two column In row.
* This grid-cols-n class is also used with some condition CSS properties or screen size.

Like

* md:grid-cols-6 : This is define the 6 columns when your screen size is medium
* Hover:grid-cols-6
* How to merge columns using grid class properties

Like

* **Col-span-full : it merge all the boxes and makes a single box depend upon how many box are available in a column .**
* **Col-span-3**

* **Col-start-\* and col-end-\* utilities define where you want to start and where you want to end columns. We can also use col-span-\*.**

Grid Template Rows:

Utilities for specifying the rows in a grid layout.

* grid-rows-4: Define 4 rows in one columns.
* grid-flow-col: define the flow of rows.
* This grid-cols-n class is also used with some condition CSS properties or screen size.

Like

* md:grid-row-4 : This is define the 6 rows when your screen size is medium
* Hover:grid-row-6
* How to merge row using grid class properties

Like

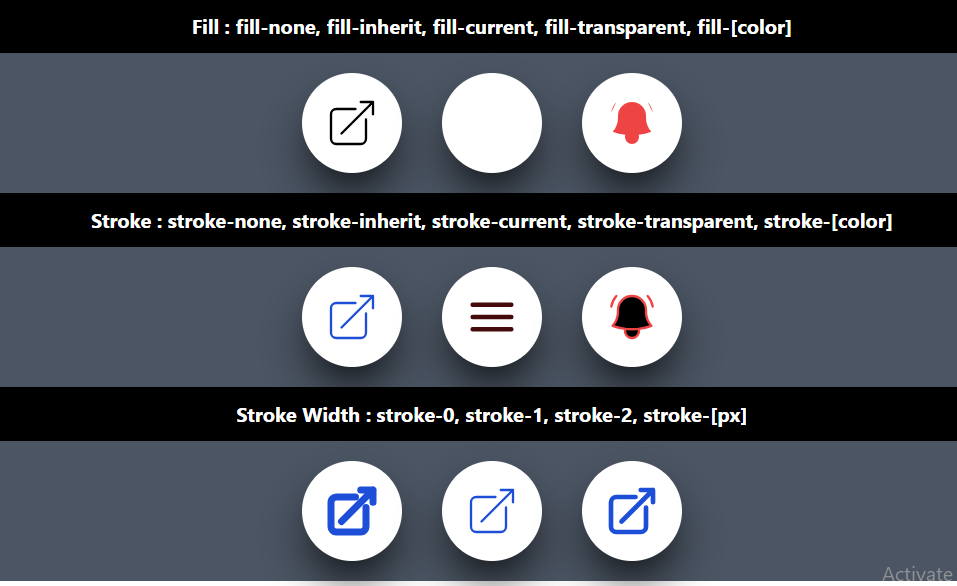
* **row-span-full : it merge all the boxes and makes a single box depend upon how many box are available in a column .**
* **row-span-3**

* **row-start-\* and row-end-\* utilities define where you want to start and where you want to end columns. We can also use col-span-\*.**

1. **GRID IN TAILWIND**

Place-content-center : it is use to horizontally and vertically center the item

**SVG Customization**



<div class="bg-white border-solid border-2 border-white size-[100px] shadow-2xl shadow-black rounded-full flex justify-center items-center">

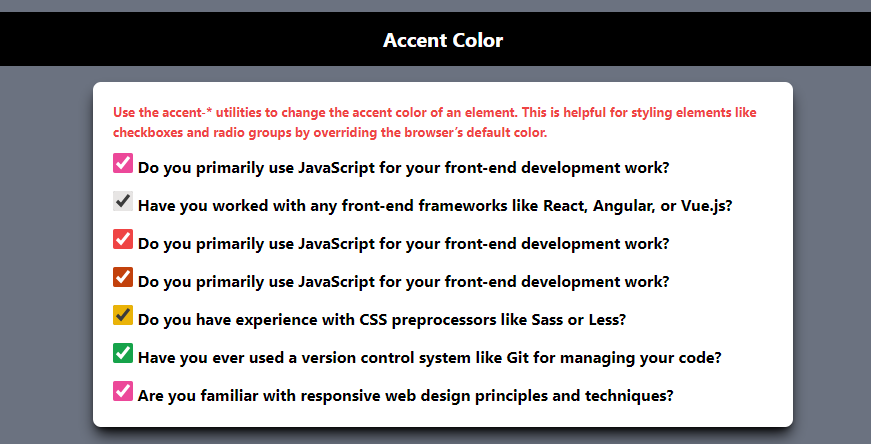
             <svg class="stroke-current text-blue-700 stroke-2 size-14 " xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">

              <path stroke-linecap="round" stroke-linejoin="round" d="M13.5 6H5.25A2.25 2.25 0 0 0 3 8.25v10.5A2.25 2.25 0 0 0 5.25 21h10.5A2.25 2.25 0 0 0 18 18.75V10.5m-10.5 6L21 3m0 0h-5.25M21 3v5.25" />

              </svg>

</div>

**Accent Color**

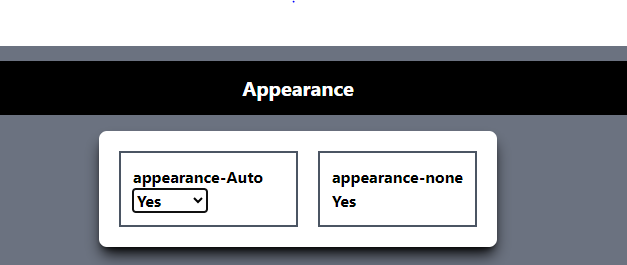


 <label>

        <input type="checkbox" class="accent-pink-500 size-5 " checked> Do you primarily use JavaScript for your front-end development work?

 </label>

**Appearance**



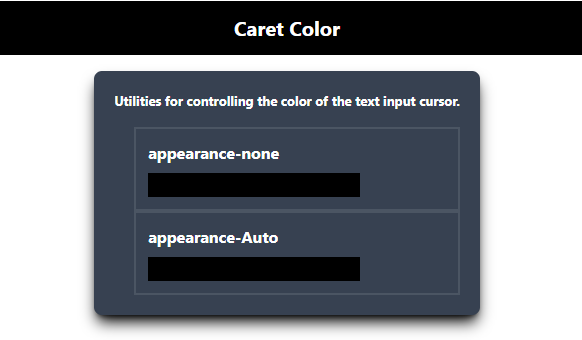
**Cursor**

Utilities for controlling the cursor style when hovering over an element.

* **cursor-auto**
* **cursor-pointer**
* **Cursor-wait**

**Caret Color**

Utilities for controlling the color of the text input cursor.



**Overflow**

Utilities for controlling how an element handles content that is too large for the container.

* overflow-auto
* overflow-x-auto
* overflow-y-auto
* overflow-hidden
* overflow-visible

Difference between overflow-auto and over flow-scroll

· overflow: auto;:

* This value adds scrollbars only when necessary (i.e., when the content overflows the container).
* If the content fits within the container, no scrollbars will be displayed.

· overflow: scroll;:

* This value always adds scrollbars, regardless of whether the content fits within the container or not.
* Even if the content does not overflow, scrollbars will be displayed, which might not be aesthetically pleasing.

**SCROLL**

* **Scroll Behavior**
* scroll-auto
* scroll-smooth
* **Scroll Margin**
* scroll-m-0
* scroll-mx-0
* scroll-my-0
* **Scroll Padding**
* **Scroll Snap Aling**
* snap-start
* snap-end
* snap-center

|  |
| --- |
|  |
|  |
|  |
|  |