TAILWIND

SHEERAZ MEHBOOB

Contents

TailWind CSS

- Uses an approach to build Web Apps called Utility First approach.
- You can create a unique Web app without using CSS
- Unlike Bootstrap Tailwind has no classes, it only has a responsive grid and flex box classes

➔ You can either download GitHub repository or setup the environment by creating a new folder tailwind that contains “index.html” file.

Requirements:

- NPM
- NPX (use to rum modules without actually install them but install temporary copy of any package)
- Git
- Install Tailwind
    - ❖ Tailwind JIT CLI
        - Method -1:
            1. Command: npx tailwindcss -o ./css/styles.css --jit --purge "./**/*.html" –watch
            2. --purge (automatically find all classes in html files)
            3. JIT (Just in Time Compiler)
            4. --watch (lookup for changes in files)
        - Method -2:
            1. npm init -y
            2. npm install -D @tailwindcss/jit tailwindcss postcss
            3. npx tailwindcss init –p
            4. customize tailwind.config.css and package.json accordingly
            5. run npm start
    - ❖ CDN (not preferred when working on project as it is not fully customizable)
        - No Customization
        - No Plugins
        - No Tree Shaking (No automatic detection whether a part of code is removed or added)
        - No JIT

Styles:

Custom Builds

By default, styles are defined like break points but we can customize them or add new ones. We can do this by extend the portion of “config” file by adding additional elements to existing object.

- New:

```
  theme: {
    extend: {
      screens: {
        medium: '1200px'
      }
    },
}
```

- Customize:

```
  theme: {
    extend: {
    },
screens: {
        sm: '640px',
        md: '768px',
        lg: '1024px',
        xl: '1280px',
        ‘2xl’: '1536px'
      }
}
```

Adding Styles

- Tailwind CSS cause crazy amount of classes but we can create our own custom classes.
- Add a new file ‘styles.css’ into ‘src’ folder. The styles.css is input file that is preprocessing for tailwind i.e. Tailwind will look into this input file to override the existing tailwind classes and output the styles based on that.

➔ First of all we will incorporate following

```
@tailwind base: (Series of styles Tailwind need to reset how things looks on browser).
 @tailwind components: This directive in styles.css tells Tailwind CSS to include all of the styles for its built-in components in your compiled CSS.
 @tailwind utilities: Has the styles that Tailwind CSS needs to output certain utilities.
```

In order to do all this we need two commands @**layer** and @**apply**. @**layer** allows us to choose which layer to target like headline etc and to add components just simply put those in components. @apply is used when you want to apply a specific Tailwind CSS style.

➔ Modify your packge.json file as:

```
  "scripts": {
    "start": "tailwindcss -i ./src/styles.css -o ./build/css/styles.css --watch"
},
```

➔ Stop the server using “Ctrl + C” command and restart it by “npm start”.
➔ Now just simply modify your input file and see the changes.

- We can even write simple css to create our own custom classes or customize the elements.

"Now on we will discuss classes so we might not have much of the documentation ahead."

Fonts

➔ Your font may not have all sort of styling like all weights etc.

Font Family:

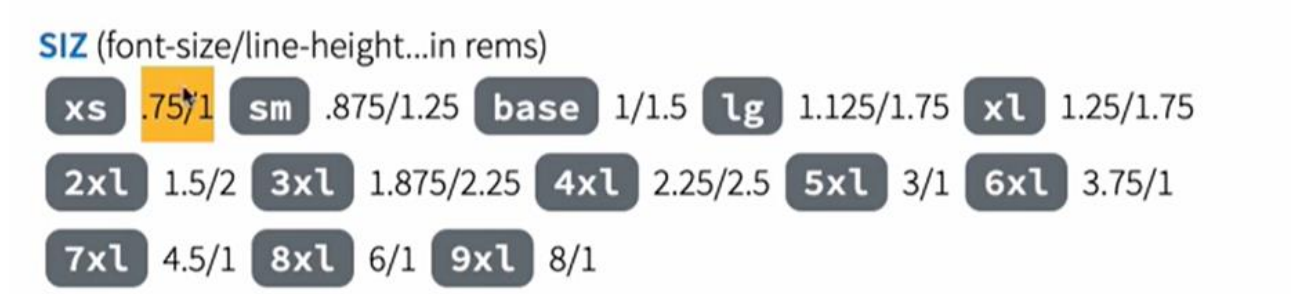
- We can modify or add any font.
- Add fontFamily in config.js file just after theme tag as:

```
theme: {
  fontFamily: {
    'sans': ['Helvetica', 'Arial', 'sans-serif'],
    'display': ['Custom Font', 'Other Font'],
  }
}
```

You can replace the array with any of your desired fonts.

Font Size:

text-size – size will be replaced by any of the tag like 'sm', 'lg', 'xs' etc.



Font Weight:

font-weight – weight will be replaced by any of the weights like font-100, font-200, font-300 or font-normal etc.

Font Style:

This contains only **italic** or **not-italic**.

Text

Color:

"text-color-shade" however shade is optional like color-red-500 or color-transparent, black or white etc.

Text Opacity:
"text-opacity-amount" to control the transparency of the text like text-opacity-40.
The opacity amount ranges from 0 to 100.

Text Alignment:
"align-position" replace the position with left, right, center etc.

Text Decoration:
"underline", "line-through", "no-underline" are the few classes to decorate the text.

Text Transform:
"uppercase", "lowercase", "capitalize", "normal-case" are the few classes to transform the text.

Text Overflow:
"truncate", "overflow-ellipsis", "overflow-clip" are few classes. Overflow is used to specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

List Styling

In tailwind list are shown without bullet points or numbers and any indentation. So, to style list we will be discussing few classes.

List Style Type:
"list-style" replace style by none, disc, decimal are the classes for styling bullets in list.

List Style Position:
"list-position" Adds indentation to the list like inside, outside.

Spacing Typography:

The style to control spacing between lines, characters etc.

Line Height:
In tailwind it is called leading. "leading-size" size will be replaced by 3,4,5 to 10.

We can also use relative line height to make it relative to the root or current container like “leading-none”, “leading-snug”, “leading-relaxed”, “leading-normal”, “leading-loose” etc.

Letter Spacing:
Space between the letters also called tracking. “tracking-size” and size is replaced by tighter, tight, normal, wide, wider, widest.

Borders

Border Width:
“Border-side-amount” like border-r-2
Sides: l, r, b, t
Amounts: 0, 2, 4, 8
“border-amount” will add border on all size of specified weight. Default weight is 1px.

Border Opacity:
“border-opacity-amount” where amount ranging from 0 to 100 by the difference of 5 like border-opacity-50.

Border Color:
“border-color-strength” color are the default like white, blue, red etc. and strength of the color ranges from 100,200 to 900.

Border Radius:
“rounded-side-amount” where sides are t, l, b, r, tl, tr, bl, br etc. and amount can be none, full, sm, md, lg, xl, 2xl etc.

Border Style:
“border-style” where styles are none, solid, dashed, dotted, double etc.

Rings

Rings are similar to borders but they have shadows and some other options.

Ring Width:
“ring-amount” amount ranges from 0 to 8 and an “inset”. And by default if amount not specified it will be 3.

Ring Opacity:
“ring-opacity-amount” amount ranges from 0 to 100.

Ring Color:
“ring-color-strength” color is already defined colors like red, blue etc. and strength is color strength ranges from 100 to 900.

Ring Offset Width:
This add a small border around the rings. “ring-offset-amount” amount ranges from 0 to 8.

Ring Offset Color:
In addition to offset you can control the color of that offset as well. “ring-offset-color-thickness” thickness is strength of color ranges from 100 to 900.

Divide

Allows to setup borders between elements. “divide-direction-amount” direction can be x or y axis and amount ranges from 0 to 8 and reverse. “reverse” is used with flex reverse-column or row property.

Divide Opacity:
“divide-opacity-amount” amount ranges from 0 to 100.

Divide Color:
“divide-color-strength” color can be red, blue, green etc. while strength ranges from 100 to 900.

Divide Styles:
“divide-style” styles can be none, solid, dashed, doted or double.

Backgrounds

These classes use to control images, gradient or color background properties.

Background Color:
“bg-color-strength”

Background Opacity:
“bg-opacity-amount” amount ranges from 0 to 100.

Background Image:
“bg-none” or “bg-gradient-to-direction”. bg-none will just have no background. “**bg-gradient-to-l” “from-red-600” “to-blue-500”** this will create a background gradient image.
Directions are t, tr, r, br, b, bl, t, tl. “to” specifies the end position of color to grow.

Gradient Color Stops:
“(from or via or to)-color-amount”
From: “from-red-500” this will gradient the color form red to transparent
from and to: “from-red-500” “to-blue-500” will gradient the color starting from red and ending to blue.
from, via and to: “from-red-500” “to-blue-500” “via-yellow-300” will gradient the color starting from red and ending to blue but have yellow patch in the middle.

Background Size:
These are related to background image, image URL should be inserted manually. “bg-size” size can be auto, cover, contain.

Background Repeat:
“bg-type” type can be repeat, no-repeat, repeat-x, repeat-y, repeat-round, repeat-space.

Background Positions:
“bg-position” position can be the bottom, center, left, left-bottom, left-top, right, right-bottom, right-top

Background Origin:
Help us to choose the image positioned in relationship to the borders. “bg-origin”. Origin can be padding, border or content.

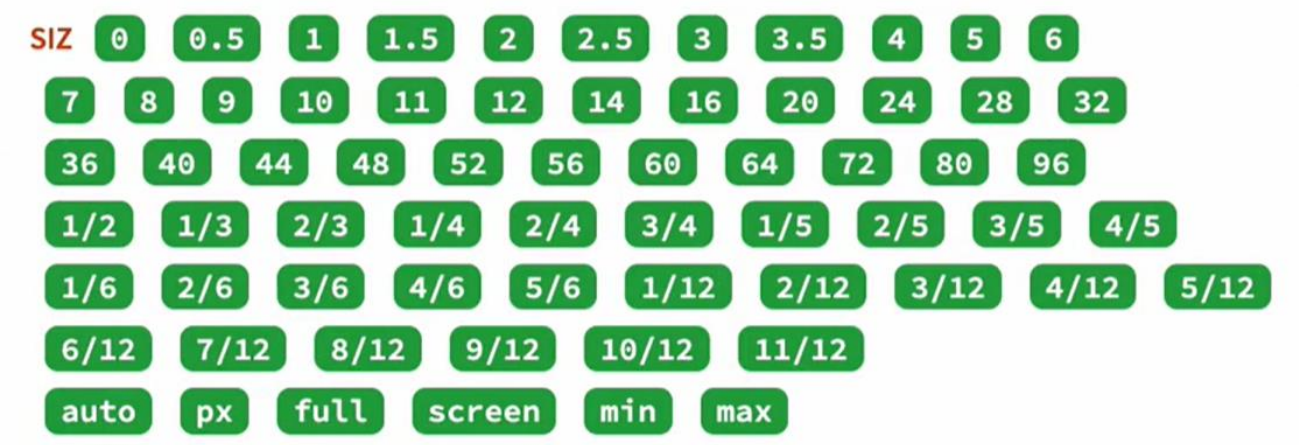
Box Decoration Break:
“decoration-slice” adds the padding to the start and end of the paragraph while “decoration-clone” adds the padding the start and end of each line of paragraph.

Box Shadow

“shadow-amount” where amount can be none, inner, sm, md, lg, xl, 2xl.

Width & Height

Width:
“w-size” where size ranges as:



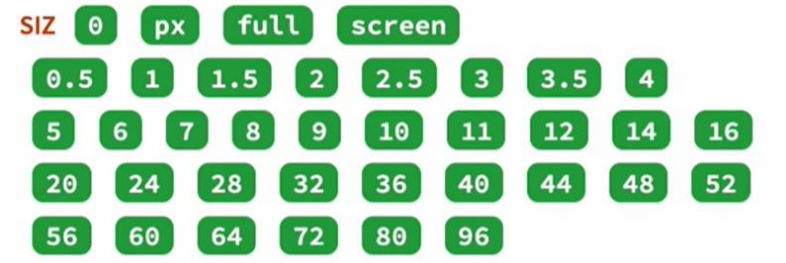
Min Width:
“min-w-size” where size can be 0, full, min, max.

Max Width:
“max-w-size” where size can be:

Height:
"h-size" where size is same as above.

Min Height:
"min-h-size" where size can be 0, full, screen.

Max Height:
"max-h-size" where size can be:



Padding & Margin

Padding:
Controls the size/space around the content and inside the borders. "p-direction-size" directions are px, py, l, r, t, b. size can be from 0 to 96 as above and px which represents 1px.

Margin:
Controls the spacing outside the border. "(-)m-direction-size" first – is optional which shows negative margin. Directions and size are same as that for padding.

Box Sizing:
"box-border" let say you specified the size of the content to 200px and if you add padding of 10px that will be adjusted within the same size and overall size will be same 200px. While "box-content" add the padding and other space with the content and for the above scenario it will be 210px.

Display & Position

This controls how an element displayed and positioned within the browser.

Display:
Elements can have any of the property instead of default. Following are some display properties: block, inline-block, inline, flex, inline-flex, table, inline-table, table-caption, table-cell, table-column, table-column-group, table-footer-group, table-header-group, table-row-group, table-row, flow-root, grid, inline-grid, contents, list-item, hidden.

Position:
Controls how elements should be positioned. By default, it's static means will be displayed as per the html structure but we can change them like static, fixed, sticky, relative, absolute. e.g. "static", "fixed" etc.
There are some other position properties as well, "(-)Side-direction-size" where Side can be inset, top, bottom, left, right & direction can be x or y and size ranges from 0 to 96, auto, px or full. e.g. "top-0", "bottom-x-44" etc.

Object Floating & Containment

This let us control how object can be float around the image or stick into the container.

Float:
"float-side" side can be none, left, right.
There is a property to undo the floating called **clear**, "clear-side" where side can be none, left, right, both.

Object Fit:
This is use to control the images size that basically works like background image. "object-side" side can be contain, cover, full, none, scale-down.

Object Position:
This is to control how we position the image w.r.t corners. "object-position" position can be left, right, top, center, bottom, left-top, right-top, left-bottom, right-bottom.

Z-index

Content in HTML is layered using a property called z-index. It is just like the stack of card stacked on one another. An individual card is stacked by the "z-amount" and amount can be any number. However, higher z-number decide to present the card at the top of stack so that will have the visibility at HTML page.

Visibility

The element can be visible or invisible in the browser using "visible" or "invisible" property but the space of the content remains even if it is invisible.

Overscroll

This controls what will happen when an element set to scroll is done scrolling. "overscroll(-direction)(-type)" where direction can be x or y and type can be auto, contain or none. The direction **auto** allows you to scroll the page once you finishing scrolling the content.

Layouts

Flexbox

Flex:
Controls how an item grow or shrink. “flex-property” where property can be 1, auto, initial, none. Flex-1 let the item grow or shrink as much as it can.

Flex Direction:
To organize the layouts, we can use flexbox. “flex(-type)(-direction)” type can be row or column while direction can be “reverse”.

Flex Wrap:
Instead of resizing when screen size gets small it just move content to next space. “flex-type” type can be wrap, wrap-reverse or nowrap. “nowrap” will re-size the content in the specifies space.

Flex Order:
It allows to reframe how items are ordered. “order-ord” where ord can be from 1 to 12 or first last or none. Item with lower order number will appear first.

Grid

A modern way how we can lay things in HTML. To control a grid create a Grid container that will have 12 columns in a row.

Grid Template:
“grid-cols(-amount)” amount ranges from 1 to 12 or none.

Gap:
Gap allows you to control room in x or y direction between the items. “grid(-direction)-amount” direction can be x or y and size can range from 1 to 96.

Grid Template Columns
When working with column we can control how many columns each element will take. “col(-type)(amount)” where type can be auto, span, start, end. Amount can be 1 to 12 or full.

Grid Template Rows:
This will let you control the number of items in a row. “grid-rows(-amount)” amount can be 1 to 6 or none. e.g. grid-row-3 will place 4 items per row.

Grid Flow:
This property will let you control how items will be places. “grid-flow-type” type can be col or row.

Grid Auto Flow:
This will let you control how items are placed on the grid. “grid-flow(-type)” type can be row, col, row-dense or col-dense. Dense will just put the items that can be fit in the empty space.

Box Alignment

This will let you control how items can fixed with the layouts.

Space Between:

Add space in between the elements. “space(-direction)(-amount)” direction will be x or y while amount can be from 1 to 96 or px or reverse.

Justify Content:

“justify-direction” direction can be start, end, between, center, around or evenly. Allows to position the elements or place the space between items.

Justify Items:

Allows you to how items are placed within the asex. “justify-direction” where direction can be auto, start, end, center, stretch. Allows items to position in the row.

Justify Self:

It is similar to Justify items but it’s only for the single items instead of all the items. “justify-self-direction ” where directions can be auto, start, end, center, stretch.

Align Content:

Align items over the cross axis in gird or a container. “content-direction” where direction can be center, around, end, start, between or evenly.

Place items:

It justifies and align at all the items at the same time. “place-items-direction” direction can be auto, start, end, center or stretch.

Place self:

It justifies and align an individual item. “self-direction” direction can be auto, start, end, center or stretch.