**Introduction to Tailwind CSS**

Tailwind CSS is a **utility-first CSS framework** that provides an extensive set of utility classes. These classes can be directly applied to HTML elements to style them, allowing for rapid, customized designs without the need to write extensive custom CSS. It promotes flexibility and consistency by offering a modular approach to styling, where each utility class is responsible for a single style rule (e.g., margin, padding, color, etc.).

Another advantage of a utility framework is its superior performance with caching due to not busting the cache for small CSS changes. Tailwind has been designed with incredible sensible defaults to make your designs look and feel "polished" and "designed" through the careful use of color palettes, color shades, sizing consistency and modern web best-practices. It's also worth noting, that Tailwind is mobile-first and has an easy naming convention with responsive designs built-in.

Tailwind's robust customization system will allow you to customize/tweak/modify everything in the framework through the use of the tailwind.config.js file. The customizations of Tailwind are beyond the scope of this course but rest assured that the default Tailwind classes are almost always exactly what you will need.

**Advantages of Tailwind CSS**

1. No need for huge CSS les with custom CSS.
2. Easy to make design changes right from the view files.
3. Better developer experience.
4. Mobile-first designs from the start.
5. Designed with defaults for a polished and designed look and feel.
6. Easily customizable with the config files.

**How to use Tailwind CSS?**

1. **Using CDN Link:**

Using Tailwind CSS via CDN link allows you to quickly include Tailwind CSS in your HTML file without the need for any installation or setup. Simply add the Tailwind CSS CDN link to the <head> section of your HTML file, and you can start using Tailwind CSS utility classes immediately.

But this way is recommended for industry use because of large file size of tailwind and poor performance and no customization.

CDN link:

<script src="https://cdn.tailwindcss.com"></script>

1. **Using Package Manager:**

First install node.js. To use Tailwind CSS in your HTML project via a package manager like npm, you need to install Tailwind CSS as a dependency. After installation, you can include Tailwind CSS in your HTML file using a <link> tag. This approach allows for customization of Tailwind CSS configuration and tree shaking to optimize the final CSS bundle size.

**Step 1:** Initialize a new HTML project or navigate to your existing project directory.

mkdir my-tailwind-project  
cd my-tailwind-project

**Step 2:** Install tailwindcss via npm, and create your tailwind.config.js file:

npm install -D tailwindcss

npx tailwindcss init

**Step 3:**Add the paths to all of your template files in your tailwind.config.js file:

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

module.exports = {

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

theme: {

extend: {}, },

plugins: [],

}

**Step 4:**Add the @tailwind directives for each of Tailwind’s layers to your main CSS file:

@tailwind base;

@tailwind components;

@tailwind utilities;

**Step 5:** Run the CLI tool to scan your template files for classes and build your CSS:

npx tailwindcss -i ./input.css -o ./output.css --watch

**Step 6:**Add your compiled CSS file to the <head> and start using Tailwind’s utility classes to style your content:

<!doctype html>

<html>

<head> <link href="./output.css" rel="stylesheet">

</head>

<body>

<h1 class="text-3xl font-bold underline"> Hello world!</h1>

</body>

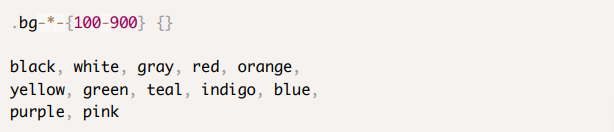
</html>

**Extension in VS Code:** Tailwind css intellisense, Tailwind fold

**1. Background classes & shades**

This set of classes change the background color of an element using a scale of 100-900 for shades and a palette of over 90 shades.

**Example:**

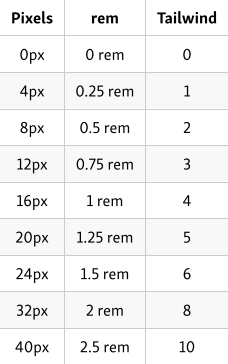


**2. Element sizing is achieved using the w-, h- and size classes.**

All of the numbers in Tailwind are based around the **rem** unit of measurement. 1 rem is equal to the size of the base font of the document. As an example, if the base font size is 16px then 1 rem is equal to 16px and we can deduce that 1.25 rem is equal to 20px . To help with these fractional numbers, Tailwind's numbered classes are multiplied by 4 to avoid having numbers with decimal places.

**Tailwind's Numbering System**

Assuming the base fontsize of the document is 16px



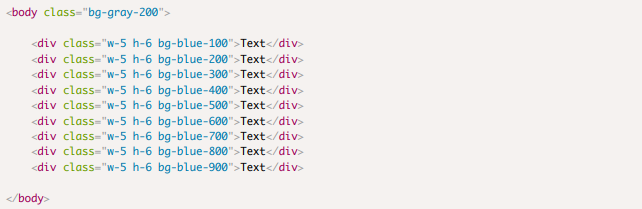
Continues to all of the available default sizes 12, 16, 20, 24, 32, 40, 48, 56, 64,

**Size:** Utilities for setting the width and height of an element at the same time. Example: size-1 = 0.25rem of width and height.

**Sizing with w-\* and h-\* classes**

****

**Example: Write all shades of blue using Tailwind**

****

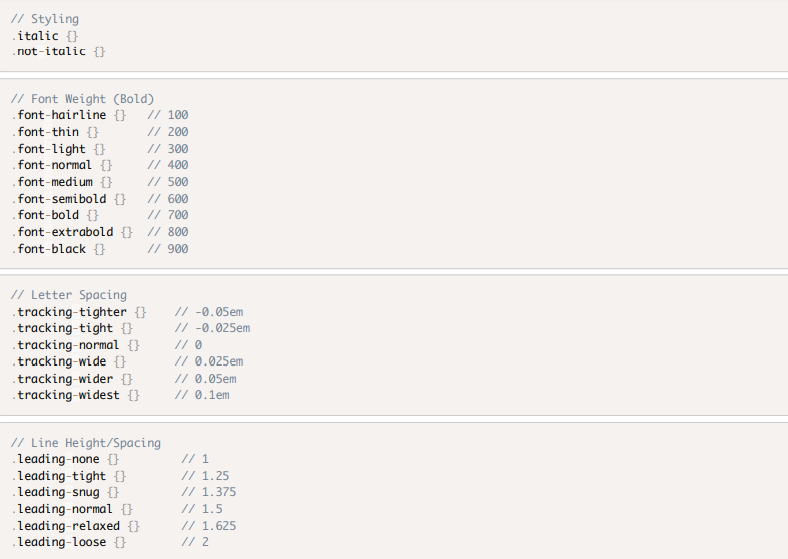
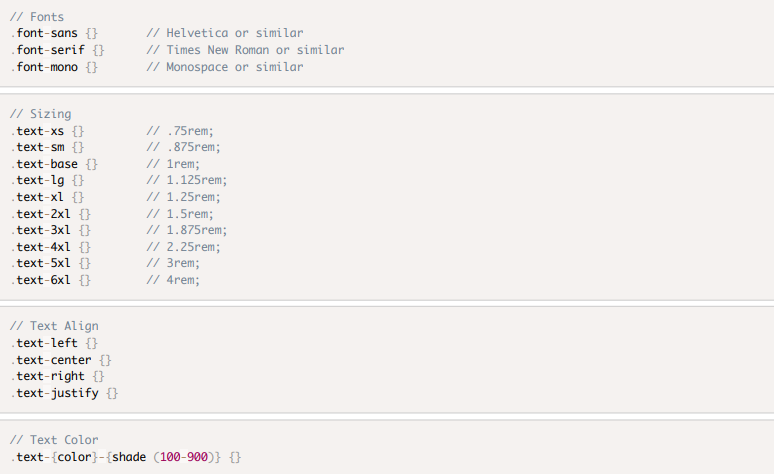
**3. Padding and margins with p-\* , m-\* , py-\* and my-\***

These classes add padding and margin to an element using the Tailwind numbering system.

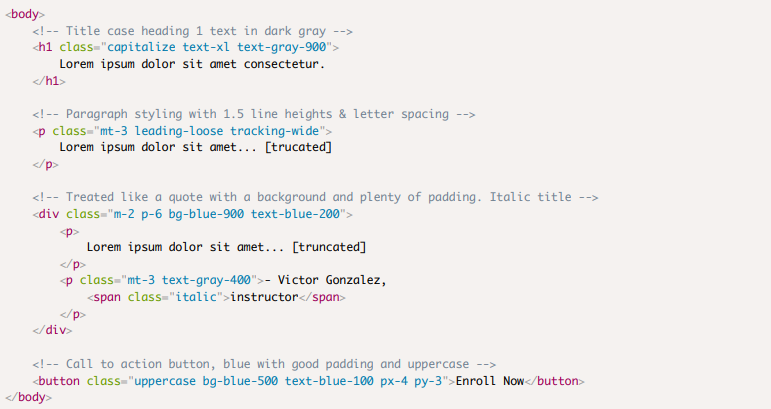
****

**4. Styling Text with utility classes**

Font styling is a huge part of any design and Tailwind has plenty of classes that we can use to style the text on our apps. It even includes utility classes for transformations like uppercase.

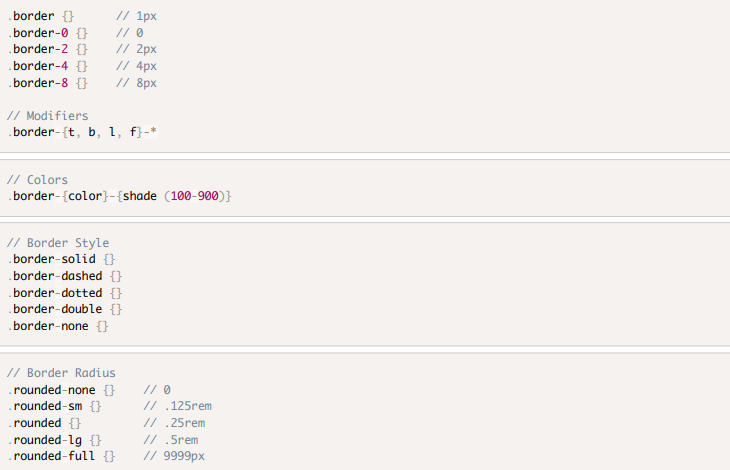


**Example:**



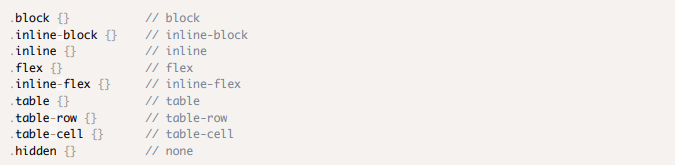
**5. Borders**

These classes will color, stylize and add radius to any border or corners.



**6. Display Modes**

Most items are display *block* by default but that can easily be changed with the available display classes

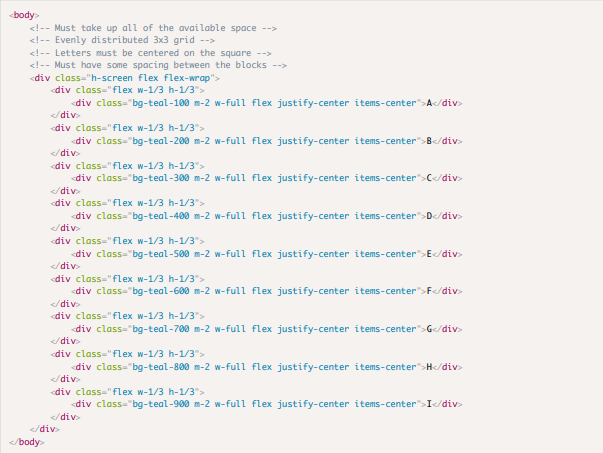
****

**7. Flexbox**

Tailwind uses Flexbox for the layout of items on the document. Flexbox is a css display property that denes a ex container. Once a container has been assigned as a ex container, we can use all of the alignment utility classes to achieve the desired look.



**Example:**

****

**8. Responsive Design**

All modern applications should be able to responsive to the screen size. Tailwind is a mobile-first framework, meaning that all of the classes that we have talked about thus far, are for mobile and trickle up to desktop. But we can change this with a couple of modifiers.



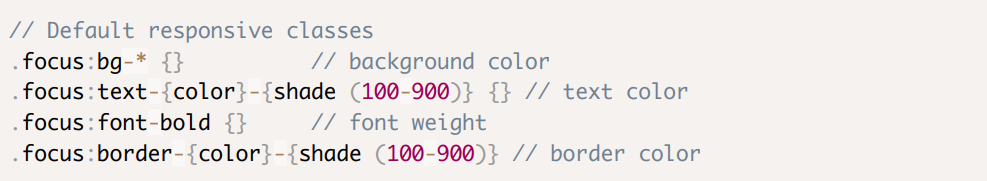
**9. Hover Modifier**

We can tap into the built-in hover state in CSS using the hover: modifier along with any of the default classes to achieve interactive designs.

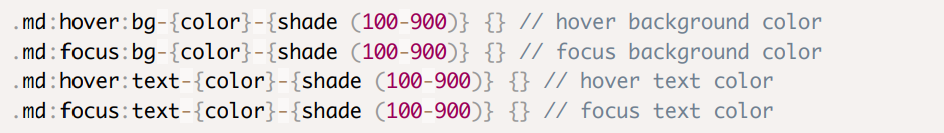


**10. Focus Modifier**

Adding a focus state is simple using the focus: modifier along with any of the default classes provided by Tailwind CSS.

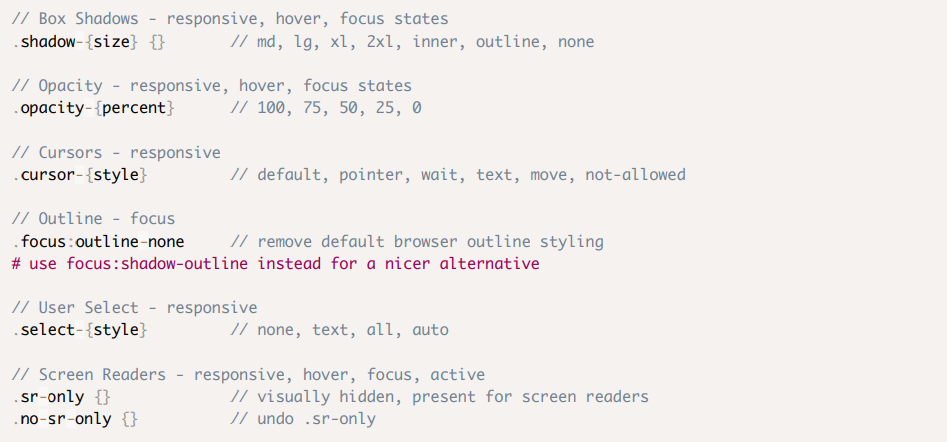


**11. Combination Modifiers**

Sometimes, the design may require 2 modifiers at the same time. **Responsive prefixes** are used to combine multiple breakpoints. These prefixes (sm:, md:, lg:, xl:, etc.) allow you to apply different styles at different screen sizes by adding them before the utility classes. For example, you may need to change the hover state for a background color but only in md: size. 

**12. Other Utilities**

There are some other useful utilities that you should know about.



**12. Shadow :**

The shadow utility in Tailwind CSS is used to apply **box-shadow effects** to an element. For example, shadow-md applies a medium shadow, and shadow-lg applies a larger shadow. Tailwind provides a range of preconfigured shadow utilities for different intensities and styles.

**Further Reading and Resources**

* [Official Tailwind CSS Documentation](https://tailwindcss.com/docs)
* [Tailwind CSS Play CDN](https://play.tailwindcss.com/)
* [Tailwind UI Components](https://tailwindui.com/)