

# Installing Tailwind

In this lecture, we're going to install **Tailwind**. It's necessary to install **Tailwind** because the static template uses many of the classes defined by **Tailwind**.

If we want the same stylings to appear in the application, we'll need to install it.

**There are two solutions for loading tailwind into an app.**

- We can directly link these style sheets in a **src/index.html** file.

Linking the style sheets can be a great way to learn a new CSS framework.

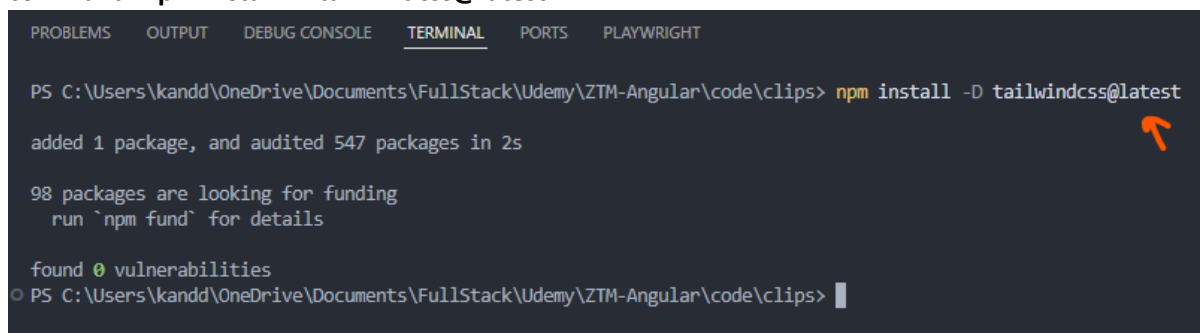
It's a decent solution, but it's not the solution will be implementing for this project.

- We want our CSS code to be processed by the Angular CLI. By having our code processed, it'll be optimized for production. We want the smallest bundle possible when shipping an app.

This project is going to contain a lot of code, which will mean a large file size. We should take every opportunity to optimize our code, including CSS. If we want Angular to process our code, we'll need to do things differently. Luckily, Angular officially supports **Tailwind**. If it detects **Tailwind**, it'll automatically handle most of the work for integrating it into our project.

There are three steps for installing **Tailwind**.

1. The first step is to install the **Tailwind** package in the command line. If you have left the server running, turn it off. Typically, we would open another command line without interrupting the server. However, problems could arise with this installation. It's better to turn off the server during installation. After turning off the server, run the following command: **npm install -D tailwindcss@latest**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  PLAYWRIGHT

PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\clips> npm install -D tailwindcss@latest

added 1 package, and audited 547 packages in 2s

98 packages are looking for funding
  run `npm fund` for details

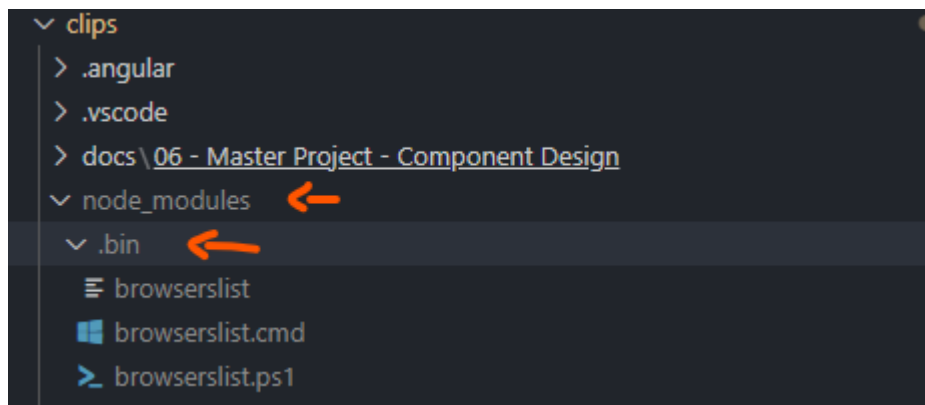
found 0 vulnerabilities
PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\clips>
```

We're installing the tail wind package.

This is a no brainer. if we want to use **Tailwind** in our project, we need to install this package.

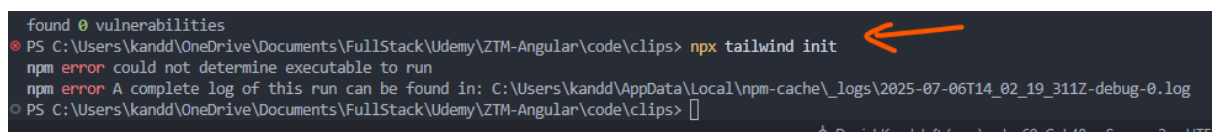
The CLI will automatically check if this package exists within our project. It'll integrate Tailwind into our projects. We've taken care of the first step of installing tailwind.

2. The second step is to create a configuration file. It's super simple. In the **node\_modules** directory, there is a folder called **bin**. Some packages will create an executable for assisting us with development. Not all packages, but many do. If a package comes with an executable, it'll be placed inside this directory:



**Tailwind** has an executable for helping us generate a configuration file. Inside the command

Line, we will run the following command: **npx tailwind init**



The issue is with your command syntax. You typed:

```
bash
CopyEdit
npx tailwind init
```

But the correct command is:

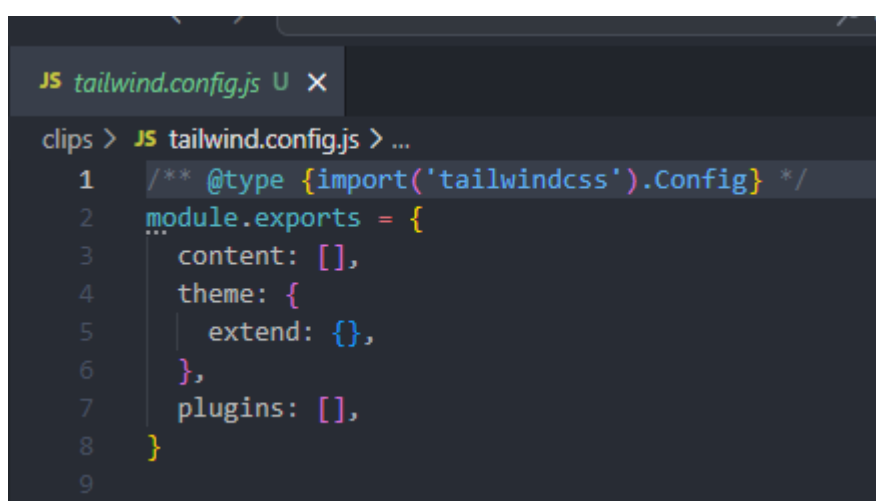
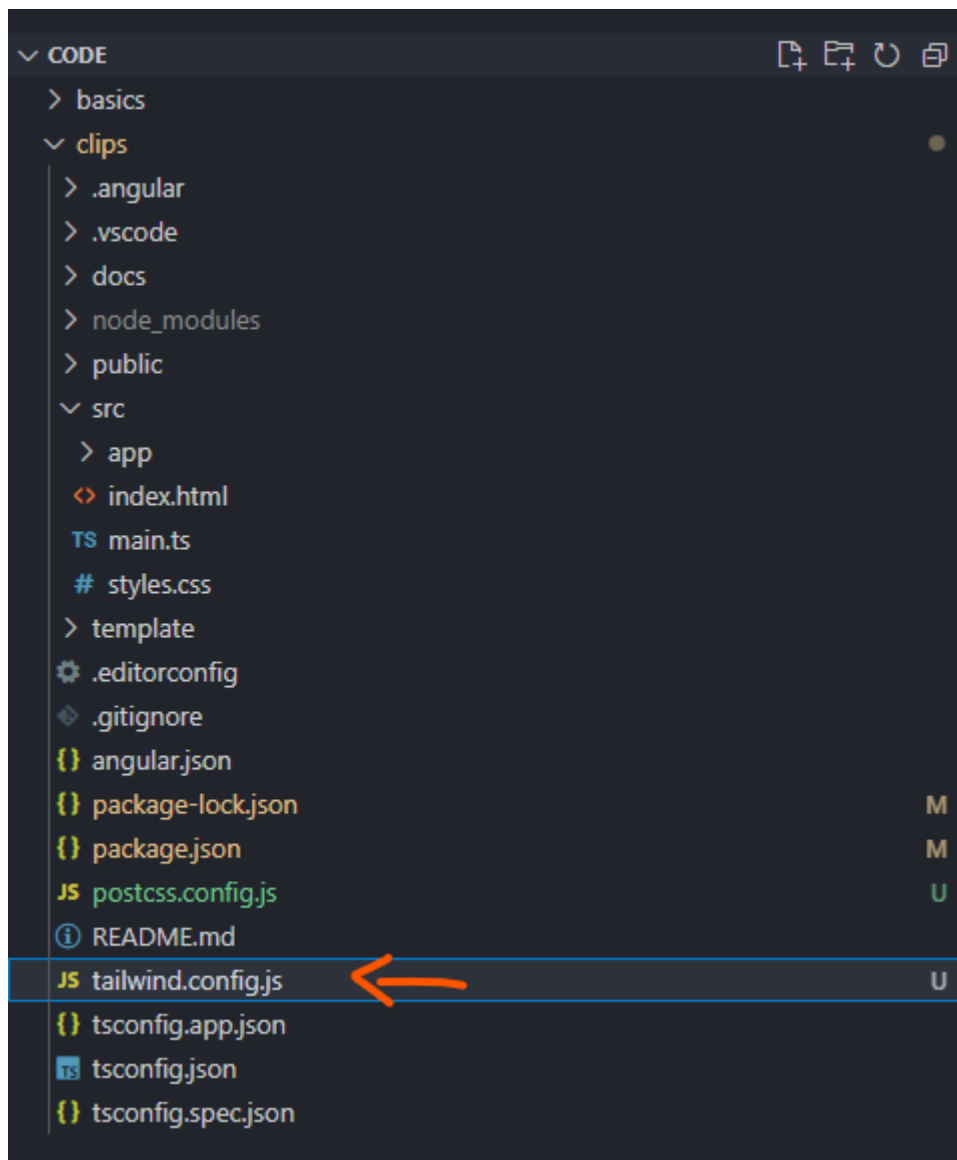
```
bash
CopyEdit
npx tailwindcss init
```

There is a problem with version 4, therefore, I have installed latest version 3:

```
npm install -D tailwindcss@^3.0 postcss autoprefixer
```

```
npx tailwindcss init -p
```

The npx command is how we can instruct node.js to run an executable inside the bin folder. After running this command, a new file will be created in the root directory of our project, called **tailwind.config.js**.



If we want to change how a class behaves, we need to configure the class through JavaScript.

Sounds crazy, I know, but that's the beauty of **Tailwind**. Instead of using a different programming language for processing CSS, **Tailwind** will process CSS with JavaScript. During this process, we can manipulate our styles through JavaScript.

Every class in **Tailwind** is customizable. It's recommended we customize the classes through JavaScript instead of through CSS.

This is the official file for configuring Tailwind. As I mentioned before, we can modify the classes generated by **Tailwind**. We can modify colors, breakpoints, font families, etc.

The first option is called **contents**.

Behind the scenes **Tailwind** removes unused CSS from our app. It's capable of scanning the **contents** of our files for classes.

We can configure the purge settings through this option. At the moment, nothing will be purged since it doesn't know where to find our CSS. We should update this option to help purge CSS to find our access code. We can tell Purge to search for our files in multiple locations.

However, everything in our app will be written in the source directory.

The **content** option is an array of files. We will pass in the following: `'./src/**/*.html,ts'`.



```
JS tailwind.config.js U X
clips > JS tailwind.config.js > ...
1  /** @type {import('tailwindcss').Config} */
2  module.exports = {
3    content: ['./src/**/*.html,ts'],
4    theme: {
5      extend: {},
6    },
7    plugins: [],
8  };
9
```

The value we passed in will tell the **Purger** to search through the source directory for HTML and TypeScript files. It'll search for **Tailwind** classes. If there are **Tailwind** classes.

It'll make sure they're not removed from the final bundle. By scanning the **contents** of our files, **Tailwind** will always know which classes are needed.

There's one last step we need to take. We need to load **Tailwind** into our project.

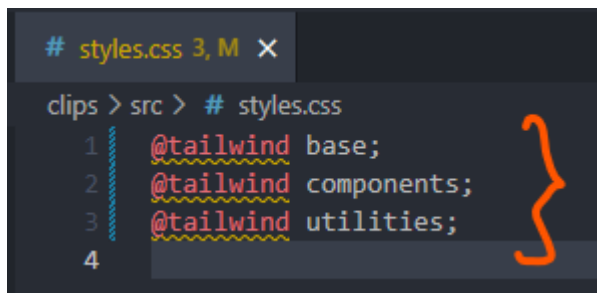
At the moment we've installed the dependencies.

**Tailwind** comes with various features. We don't have to load every feature in **Tailwind**.

We have the option of picking and choosing what we want.

We can load **Tailwind** by updating these styles [src/styles.css file].

Inside this file, we will add the following: at Tailwind Base at tailwind components, at Tailwind Utilities.



```
# styles.css 3, M X
clips > src > # styles.css
1 @tailwind base;
2 @tailwind components;
3 @tailwind utilities;
4
```

The **@tailwind keyword** is called the **directive**. It's **not** to be confused with **Angular directives**.

**Tailwind** created these directives. They'll inject **Tailwinds** classes into our project.

**Tailwind classes** aren't automatically injected into our project.

We need to tell it where to load its styles. These **directives** will perform that action.

This syntax is **not** a **CSS feature**, but as I mentioned before, our code will be processed into JavaScript. From there, **Tailwind** will be able to replace these **directives** with actual CSS.

We're adding these **directives** to these **styles.css** file.

Normally, we should add styles to a components respective style sheet.

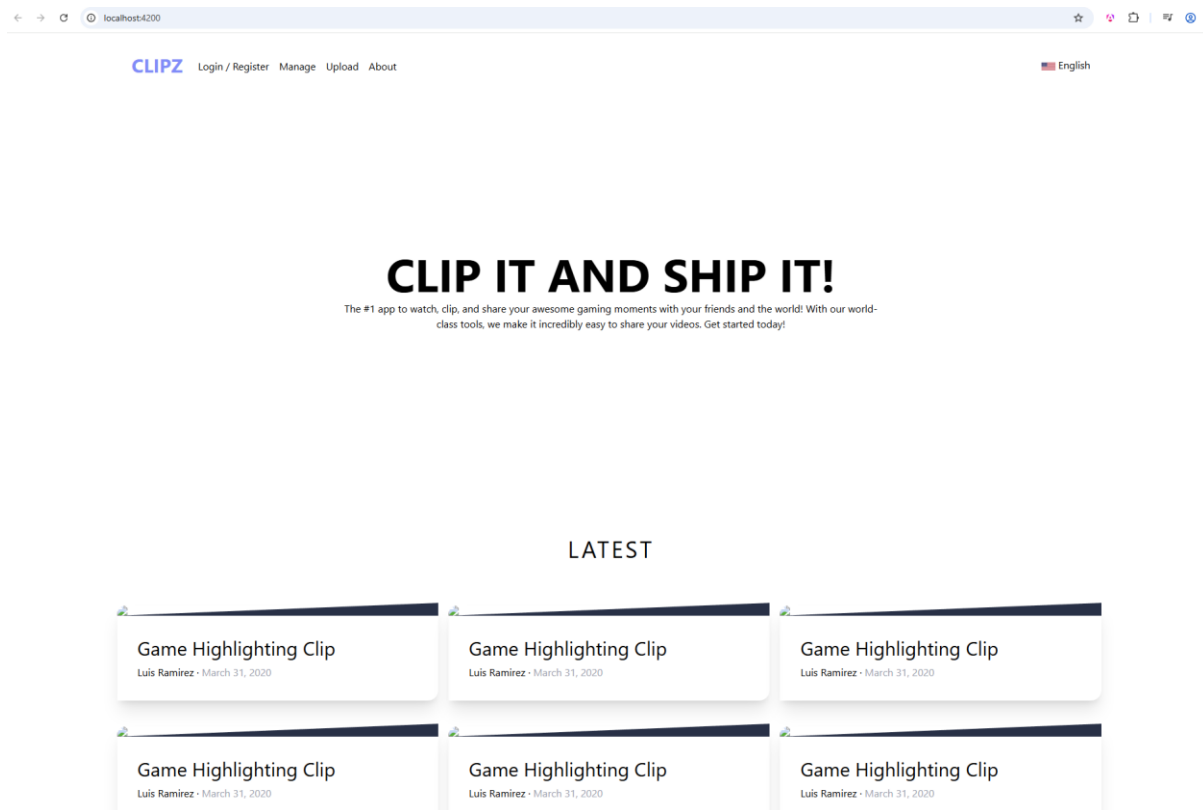
However, there are hundreds of classes, we would constantly be switching back and forth between files to add CSS.

Luckily, thanks to **PurgeCSS**, we don't have to worry about excessive CSS or overlapping styles.

**Tailwind** will be smart enough to understand what we need.

Let's run the server with the **ng serve** Command.

Next, let's view the browser:



Our app is starting to look like the original.

There are some modifications we need to make.

In the next lecture, we will be exploring configuring Tailwinds to get the exact look we had in the static version.