**Tailwind CSS**

**Website:** [*https://tailwindcss.com/*](https://tailwindcss.com/)

**Documentation:** *https://tailwindcss.com/docs/installation*

**Installing Tailwind CSS without PostCSS:**

1. Create package.json file using **npm init –y** command.  
   -y means to create package.json with default options.
2. Install tailwind css using **npm install tailwindcss**
3. Install autoprefixer file using **npm install autoprefixer.**This file consist all the css browser dependent code for Chrome, Mozilla, Safari etc. So that further tailwind will render CSS properly in different browsers with different browser dependent CSS code.
4. Setup **Folder Structure**.  
   ROOT FOLDER 🡪 {Folder(public ) 🡪index.html, Folder (src) 🡪**tailwind.css**(  
   ADD,  
   @tailwind base;  
   @tailwind components;  
   @tailwind utilities;  
     
   Here, we can write our custom css ).
5. Add a build script in package.json,   
   “scripts”: {  
   “build”: “tailwindcss build **./src/tailwind.css –o ./public/tailwind.css”**  
   }  
   This means that, process/build tailwindcss from tailwind.css file present under src folder and gives output/generate another tailwind.css file under public folder or if it already existed inside public folder then inbuilt all tailwind css code will overwrite inside tailwind.css file of public folder.
6. Run **npm run build** to build tailwind.css file under public folder
7. Now, we’ve to do external linking of ./public tailwind.css file like we link an normal external css file inside index.html file.  
   Ex: <link rel = ‘stylesheet’ href = “tailwind.css”>

**Installing Tailwind CSS with PostCSS (For real-World Projects):**

1. **Step 1 to Step 4** would be same as of without PostCss.
2. Add a new dependencies by running this command, **npm install postcss-cli**
3. Now we need configuration file of tailwind &postCSS. So, by executing this command **npx tailwindcss init –p** to create configuration file of tailwind & postcss (*tailwind.config.js* & *postcss.config.js* respectively)  
   These files consists lots of important stuff, like purge, theme, plugins etc.  
   Inside purge, we’ve to maintain the location of only those files that we want to render in user’s front-end.  
   Ex: purge: [‘./public/\*\*/\*.html’]  
   Move to public folder, recursively, move to all the files with extension .html and return back all the unused classes from the tailwind.css and keep only used classes of .html files in tailwind.css.  
   We can also purge here our JS file.
4. Add a build script in package.json,   
   “scripts”: {  
   “build”: “postcss **./src/tailwind.css –o ./public/tailwind.css”**  
   }  
   This means that, process/build postcss from tailwind.css file present under src folder and process to gives output/generate another tailwind.css file under public folderor if it already existed inside public folder then inbuilt all tailwind css code will overwrite inside tailwind.css file of public folder.
5. Run **npm run build** to build tailwind.css file under public folder
6. Finally, we’ve to do external linking of ./public tailwind.css file like we link an normal external css file inside index.html file.  
   Ex: <link rel = ‘stylesheet’ href = “tailwind.css”>

**Deploy Tailwind CSSproject (For both with & without PostCss):**

Basically, which means to avoid unused CSS from tailwind.css file.

1. Install a package **npm install win-node-env**
2. Write a prod (production) script inside package.json file,  
   **Without PostCSS:**  
   “scripts”: {  
   “prod”: “NODE\_ENV=production npx tailwindcss build ./src/tailwind.css –o ./public/tailwind.css”  
   }  
   **With PostCSS:**  
   “scripts”: {  
   “prod”: “NODE\_ENV=production postcss ./src/tailwind.css –o ./public/tailwind.css”  
   }
4. Now to configure tailwind. Run this command **npx tailwindcss init** to create a configuration file of tailwind.

This file consists lots of important stuff, like purge, theme, plugins etc.  
Inside purge, we’ve to maintain the location of only those files that we want to render in user’s front-end.  
Ex: purge: [‘./public/\*\*/\*.html’]  
Move to public folder, recursively(means to look any file), look to all the files with extension .html and return back all the unused classes from the tailwind.css and keep only used classes of .html files in tailwind.css.  
We can also purge here our JS file using comman (**,**).

1. Now we’ve to execute the prod script using **npm run prod**

**Note:**It is highly recommended to run prod script at last (after full completion of the projcect) for production. Because after production, we’ll not able to directly add new tailwind classes in our html file or any file that were mentioned in purge of tailwind.config.js file.

Because after production prod script would removed all the previously unused css classes for the files mentioned in purge of tailwind.config.js file.

So, after production in order to use new tailwind classes, we’ve again overwrite all in-built tailwind css code in our tailwind.css file of public folder by again executing build script.

**Install Tailwind CSS with Create React App**

1. Go to [*https://tailwindcss.com/docs/guides/create-react-app*](https://tailwindcss.com/docs/guides/create-react-app)
2. Open Window Poweshell terminal in VS Code.
3. Type **npm install -D tailwindcss postcss autoprefixer**
4. Then run **npxtailwindcssinit-p** to create *tailwindconfig.js* file as well as *postcss.config.js* files under the project.
5. Now, copy the **template paths**and paste it to the tailwind cssconfig file i.e. (tailwind.config.js).
6. Now, copy the tailwind directives to our css file under src file. i.e. ./src/index.css

And finally Again run npm command in order to execute the project and see the tailwindcss result. Run *npm run dev* (If react app created using vet) or *npm run start* (if react app created using default npx )