* npm i tailwindcss
  + créer le fichier style et y coller le code suivant :
  + @tailwind base;
  + @tailwind components;

@tailwind utilities;

* Configuration de tailwind
  + npx tailwindcss init
* npm i autoprefixer postcss-cli watch
* creer le fichier postcss.config.js dans la racine du projet et y coller le code suivant

module.exports = {

plugins: [

require('tailwindcss'),

require('autoprefixer'),

],}

* ajouter la commade suivant dans package.json :"build": "postcss public/css/style.css -o dist/css/style.css"

TEST :

Lancer la commande pour voir le resultat : le fichier sera crée dans dist/css

Remarque : Pas de css appliqué.

Solution : mettre la valeur de CONTENT dans le fichier de configuration de tailwind, pour lui dire quels sont les fichiers qui utilisent tailwind :"./src/\*\*/\*.{html,js}","./index.html"

**REMARQUE :**

Il faut, a chaque fois avoir ajouter une classe, compiler encore le css.

**POURQUOI ?**

Parce que le fichier generé apres la compilation de tailwind (dist/css/style.css) regardes les fichier html et **JS** (ce qu’on preciser dans content de tailwind.cnfig) et ajoute les classe utiliser dans le fichier generé.

**SOLUTION** : donc il faut le faire une seule fois en ajouter –watch (pour regener a chaque modification) dans la commande build

**NB :**

* On peut créer une commande pour lancer en production(dans laquelle il ya moins de commande) et une commande pour le dev (dans laquelle on combine toute les commandes separee par &).

Exemple : "dev": "lite-server & tsc -w & postcss public/css/style.css -o dist/css/style.css -w"

* Alternative: la commande watch (npm install watch)

Il suffit de dire a **watch** de relancer une commande (de notre choix) à chaque fois que il  y a changement dans un fichier.

Exemple :

"aly": "postcss public/css/style.css -o dist/css/style.css",

"watch": "watch \"npm run aly\" ./dist"

LES MODULES:

* 1ere approche : LES NAMESPACES
  + Créer un fichier avec un un namescpace
  + Importer le namespace dans le fichier destinataire : ///<reference path= ‘…’/>
  + Tsconfig :
    - Activer l’option outFile pour créer le fichier bundle
    - Mettre la valeur de l’option module à AMD
* 2eme approche : ES6 MODULES :
  + Tsconfig :
    - Mettre la valeur de l’option module à ES6
    - Mettre la valeur de l’option target à es2016