Chapitre1 : Introduction

I. Qu'est-ce qu'Angular ?

1. Définition

Angular est un Framework JavaScript pour clients, développé par Google, qui vous permet de développer des applications.

Un Framework logiciel est un ensemble d'outils et de composants à partir desquels on peut développer des applications.

Le développement Angular passe par trois langages principaux :

- Le HTML pour structurer : toutes vos connaissances avec ce langage vous seront utiles, et Angular viendra vous ajouter quelques nouveautés ;
- Le SCSS pour les styles : le SCSS est une surcouche du CSS qui y apporte des fonctionnalités supplémentaires, mais qui permet également d'écrire du CSS pur si on le souhaite ;
- le TypeScript pour tout ce qui est dynamique, comportement et données , un peu comme le JavaScript sur un site sans framework.

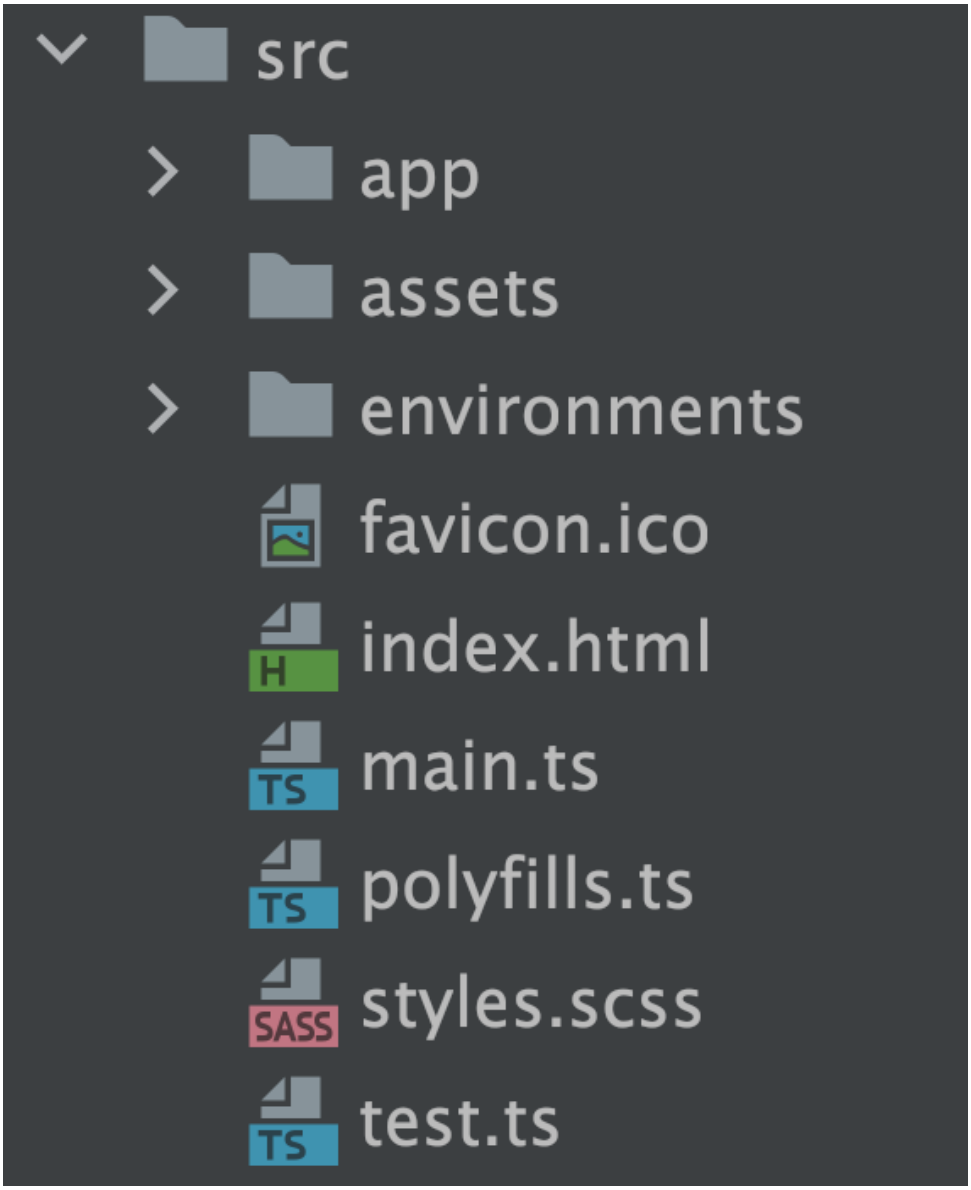
2. Type script :

TypeScript est un langage de programmation libre et open source. Il vient ajouter des syntaxes supplémentaires au JavaScript ; notamment, comme son nom l'indique, **le typage strict**. Le typage strict nous oblige, entre autres, à spécifier le type de toutes les variables, contrairement au typage dynamique de JavaScript. Cette contrainte peut paraître gênante, mais elle permet de réduire considérablement le nombre d'erreurs au runtime, et facilite énormément le développement avec un IDE comme VS Code ou WebStorm.

3. Installation

- Pour installer le CLI d'Angular, il vous faut une version LTS de **Node et npm.**
- On installe le CLI avec la commande npm **i -g @angular/cli**
- On peut vérifier la version installée du CLI avec ng **v**
- Une nouvelle application Angular se crée avec ng **new**
- L'exécution du le serveur de développement avec **ng serve**

Une application Angular contient beaucoup de fichiers (notamment de configuration). Nous allons majoritairement travailler dans le dossier src :

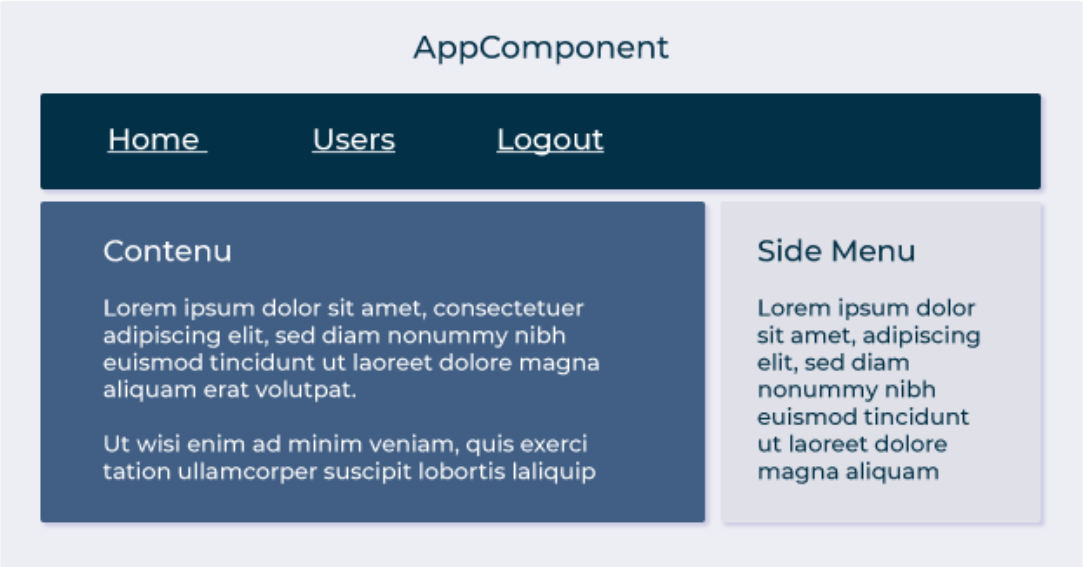


Index.html est le fichier principal de votre projet, qui ne contient qu'une balise <app-root>. C'est la racine de votre application

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Snapface</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

II. Les composants

Les components sont les **composants de base** d'une application Angular. Une application Angular peut être vue comme une **arborescence** de components avec AppComponent comme component racine.



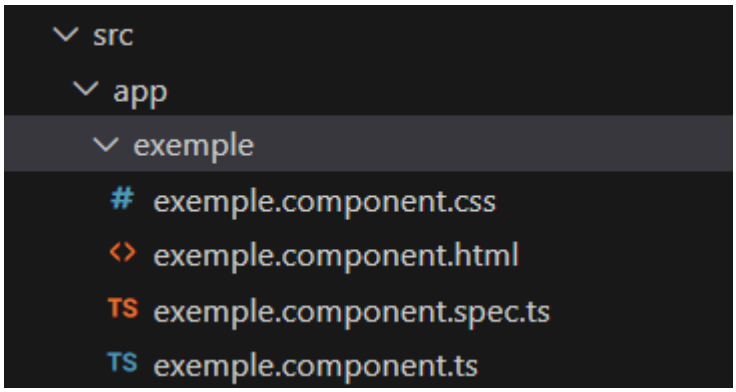
1.Génération composant

Le CLI (Commande line interface) d'Angular propose des commandes de création pour faciliter le développement.
ng generate component nom_composant => **ng g c** nom_composant

ng g c exemple

```
CREATE src/app/exemple/exemple.component.html (22 bytes)
CREATE src/app/exemple/exemple.component.spec.ts (566 bytes)
CREATE src/app/exemple/exemple.component.ts (206 bytes)
CREATE src/app/exemple/exemple.component.css (0 bytes)
UPDATE src/app/app.module.ts (479 bytes)
```

Il a **créé**, dans un sous-dossier exemple sous dossier app, creation les trois fichier du component, et il a mis à jour **app.module.ts**



Contenu fichier app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ExempleComponent } from './exemple/exemple.component';

@NgModule({
  declarations: [
    AppComponent,
    ExempleComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
```

```
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Pour utiliser un component dans une application Angular il faut le **déclarer** dans un **module** – AppModule étant le module principal de l'application, tout comme AppComponent est le component principal. Tous les components que vous créerez dans ce cours seront déclarés ici.

2. Contenu d'un composant Angular

Un fichier TypeScript contenant :

- Les donnees du composant
- La logique/le comportement du composant

Un fichier html contenant

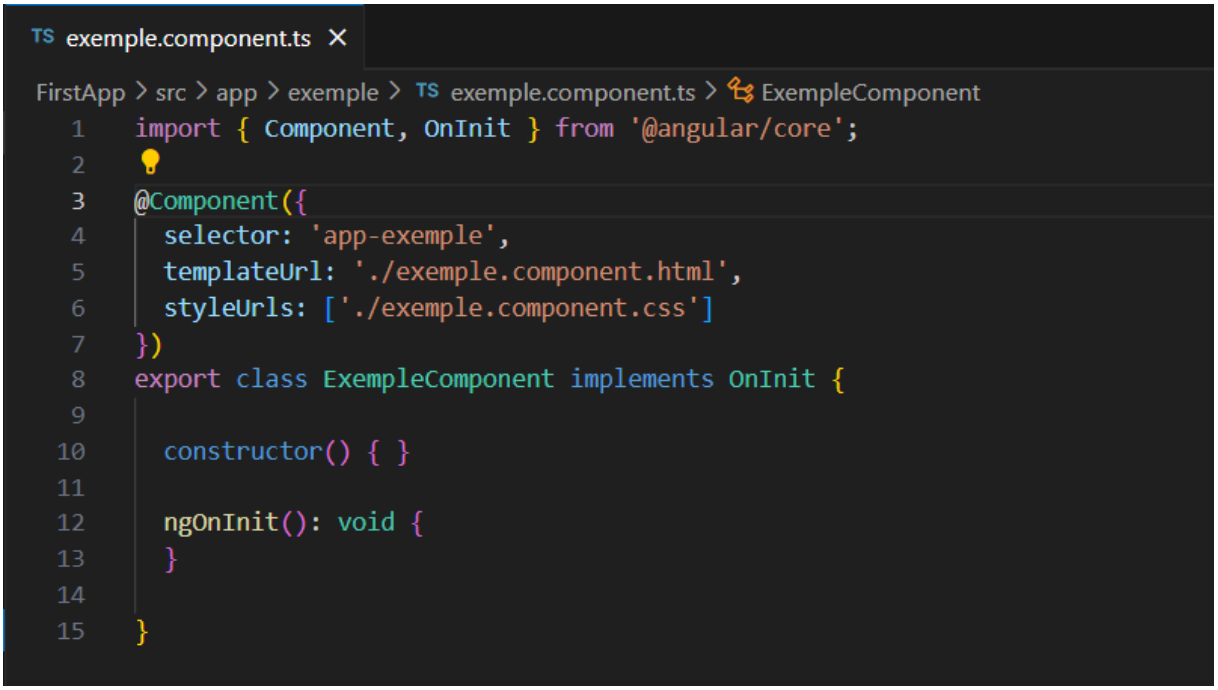
- Le code HTML affiche par le browser
- Des instructions pour interagir avec le code TypeScript

Un fichier css contenant

- Le style propre au composant

Un fichier de test unitaire (fichier .spec.ts)

Dans le Fichier ts qui s'appelle dans cas exemple.component.ts



On voit ici la construction du component. On voit une **classe** TypeScript qui s'appelle ExempleComponent, avec un constructor et une méthode ngOnInit.

Cette classe est déclarée avec un décorateur @Component à qui on passe un objet de configuration avec un **sélecteur**, un fichier de **template** et un fichier de **styles**.

Un décorateur en TypeScript permet, entre autres, d'apporter des modifications à une classe. Ici, le décorateur @Component vient ajouter tous les comportements nécessaires à l'utilisation de ce component dans l'application.

Pour le template et les styles, c'est plutôt simple : on dit à Angular quels fichiers utiliser pour afficher notre component. Le sélecteur (avec le préfixe app- par défaut), c'est ce qui va nous permettre d'insérer ce component dans notre application.

L'application angular est une arborescence de components avec AppComponent comme racine : c'est donc dans app.component.html qu'on va venir ajouter le **sélecteur** de notre nouveau component comme balise HTML :

```
<app-exemple></app-exemple>
```

exemple works!