**Angular :** est un Framework JavaScript qui permet de créer des applications SPA (Single Page Application), c’est-à-dire une application qui tourne dans une seule page HTML, bref Une application Angular peut être vue comme une **arborescence** de components avec AppComponent comme component racine de l'application.

**npm i -g @angular/cli** 🡪 permet d'installer Angular en utilisant NodeJS ,i signifie installe, -g d’une manière globale sur la machine.

**ng v** 🡪 indique la version d’Angular utilisé.

**ng new projectName --style=scss --skip-tests=true** 🡪 permet de créer une application Angular, le SCSS sera le type de styles á appliquer, et les tests d’angular seront ignorés.

L’application est lancée dans le localhost sur le port 4200.

Chaque composant de Angular est représenté par 3 fichier, fichier Template de type html, fichier de style de type scss, fichier de script de type typeScript.

Index.html est le point d'entrer de l'application, contient une seul balise <app-root> qui fait référence au template de appComponent.

**ng generate component component-name** 🡪 permet de créer un composant, un dossier portant le nom du composant et contenant les fichiers (ts, html, scss) sera ajouter dans le répertoire src/app

Tous les composants sont déclarés dans AppModule (Module principale de l'application).

Le fichier ts du composant contient une annotation (décorateur) @Component qui indique que cette classe s'agira d'un composant, et indique aussi les différents fichiers (selector, template, et SCSS) utilisable par ce composant.

**Selector**: est la balise ou bien l'élément html qui va être utiliser pour injecter le contenu du composant courant dans un autre composant.

Pour afficher la template du composant, vous devez ajouter son selector dans le template de appComponent.

**ngOnInit:** est une méthode (de l'interface OnInit ) utilisée pour initialisé un composant, jamais initialisé les variables du composant dans le constructeur (utilisé que pour l'injection des dépendances).

**String interpolation** : {{ componentVariable}} utilisé pour afficher une variable de composant dans la template du composant.

**Attribut Biding :** [attribute] = "componentVariable" utilisé pour affecter á un attribut d'un élément html une variable du composant.

**Événement Angular :** (event) = "componentMethod" permet d'affecter á un événement d'Angular, une méthode du composant. (Click) = "onAddSnap" 🡪 événement appliquer sur un button élément.

**@Input () :** est un décorateur qui permet de dire que cette propriété du composant est injectable depuis l'extérieur (depuis un composant parent).

**<app-component-name [variableWithInputDecorator] = "variableFromAParentComponent" > </app-component-name>** 🡪 permet d'injecter une variable du composant actuelle dans un composant enfant.

**\*ngIf:** utiliser comme attribut dans un élément html, et elle reçoi une condition qui indique est ce que l'élément html sera afficher ou non.

**\*ngFor:** utiliser comme attribut sur un élément html, pour le répéter tant de fois que le nombre d'itération existes dans la liste 🡪 \*ngFor = "let x of integerList".

**[ngStyle]:** utiliser comme attribut sur un élément html, pour ajouter du style dynamique 🡪 [ngStyle] = "{color: 'rgb(0, '+ facesnap.snaps + ',0)'}"

[ngClass]: utiliser comme attribut sur un élément html, recoit un pair clé valeur, la clé est le nom de la classe css á appliquer, et la valeur est une condition qui indique quand le style de la classe sera ajouté á l'élément html 🡪 [ngClass] = "{ classCss : condition }"

**@Injectable({ providedIn : 'root'})** : utilisable sur une classe ( service, dao, …) pour dire á Angular d'enregistrer une seul instance de cette classe á la racine de l'application. Cette instance sera partagée par les parties intéressés.

**Injection de dépendance dans Angular :** pour injecter une instance dans un composant, il suffit de créer un constructeur qui a comme paramètre l'instance voulue d'être injecter 🡪

Consturctor( private instance: instanceClass) {}

Ajouter un **modificateur d'accès** comme  public  ou  private  à un argument du  constructor  crée une propriété avec ce même nom dans la classe.

**Configuration de router** :

1. Création d'une classe AppRoutingModule avec le décorateur @NgModule.
2. Initialiser dans la classe AppRoutingModule une tableau **const routes** de types Routes, qui va lier les **routes** de votre application aux **components** correspondants
3. Importer l'objet RouterModule.forRoot(routes) est configurer la racine de l'application avec les routes récemment crées, et réexporter le RouterModule configuré pour l'utiliser, En utilisant les tableaux import et export de décorateur @NgModule de AppRoutingModule.
4. Il reste á ajouter AppRoutingModule aux **imports** de AppModule, le module principal de l'application.
5. Finalement dans la template de appComponent remplacer les balises des composants par la baliset <router-outlet> </router-outlet>

Const routes : Routes = [

{ path : '', component: landingComponent}

{ path : 'link1', component: componant2},

{ path : 'link2', component: componant1}

] 🡪 Exemple d'un tableau des routes qui fait la correspondance entre les composants est lin URL.

**<a routerLink = "link1"> aller vers composant1 </a>** 🡪 aller vers le composant1 en utilisant l'attribut routerLink,

**This.router.navigateByUrl('link2')** 🡪 utiliser dans une méthode qui sera appeler dans un évènement angular pour mener vous au composant2. L'objet router doit être injecter par constructeur au composant puis qu'on puisse l'utiliser.

This.activatedRoute.route.snapshot.params['parm1'] 🡪 permet d'extraire la valeur de param1 passer dans la requetes, afin de l'utiliser pour naviguer

Remarque:

**var!: typVar** 🡪 dit au compilateur de typeScript d'ignorer l'initialisation de cette variable.

**Var?: varType** 🡪 il s'agit d'une variable optionnelle.