CHAPITRE 1: INITIATION À ANGULAR

CHEBBI SAFA

SAFA.CHEBBI@SUPCOM.TN

Objectifs pédagogiques

À la fin de ce cours, on sera capable de:

- ✓ créer une application Angular
- ✓ construire des components Angular
- ✓ structurer un document avec les directives
- ✓ modifier l'affichage des données avec les pipes
- ✓ améliorer la structure de votre application avec les services et le routing

Prérequis

- Bonnes connaissances en HTML, CSS et javascript.

- **Outils nécessaires:**
  - Editeur de code (de préférence un IDE comme VS Code ou WebStorm)
  - Navigateur pour tester l'application

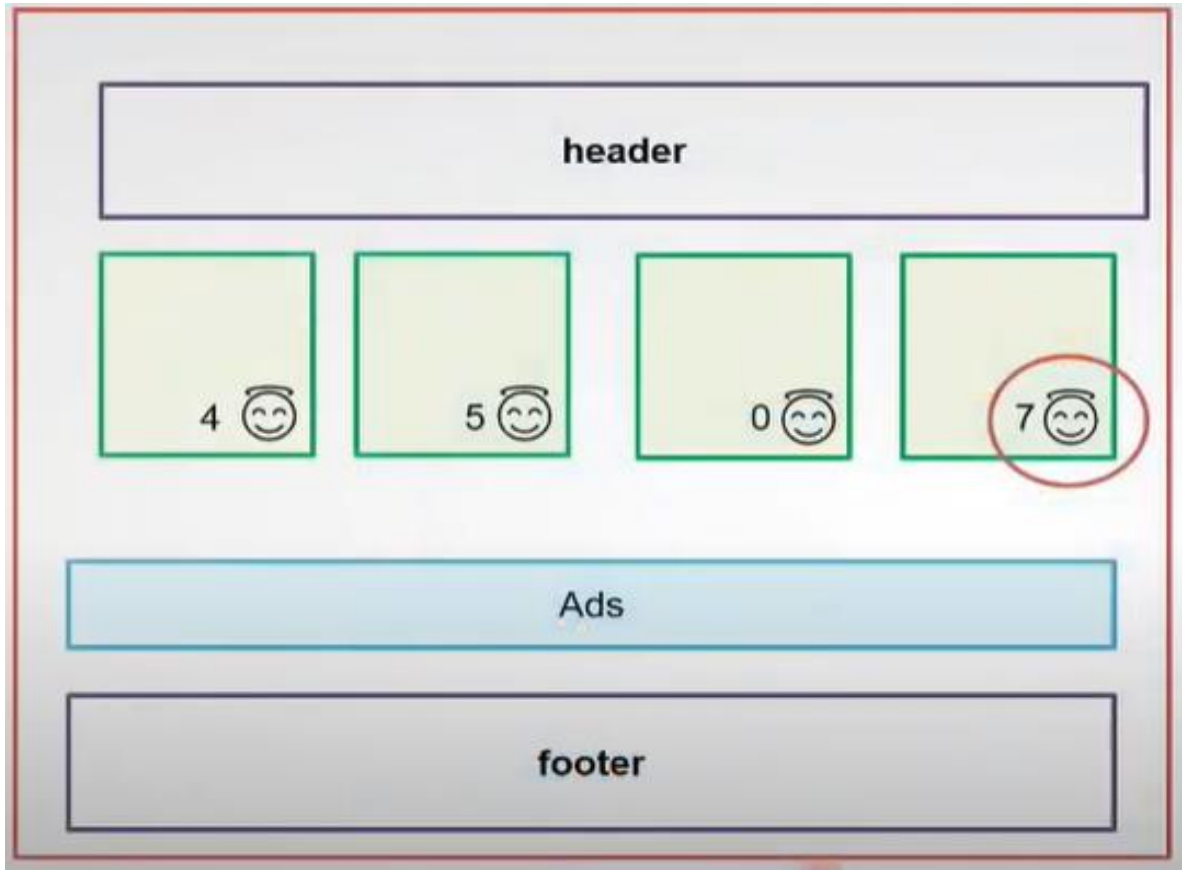
Introduction générale:
Architecture: Front-End & Back-End

- La tendance actuelle est de vouloir séparer entre les deux parties d'un site web :
  - ✓ La partie cliente (Front-End) : composée de fichiers HTML, CSS et JavaScript qui sont interprétés par le navigateur
  - ✓ La partie serveur (Back-End) : composée de fichiers PHP, JS(node) ou Java ou autres qui sont interprétés coté serveur et SGBD

Introduction générale:
Un Framework , c'est quoi ?

- Un framework == un cadre de travail
- Un ensemble de composants qui structure l'application et qui contraint la manière dont vous allez la développer
- Un framework est toujours associé à un langage de programmation : Java, PHP, JavaScript… et aide dans le développement d'applications web.

Introduction générale:
Pourquoi utiliser un Framework en front end? (1)



Index.html

- Complexité du projet
- Structure du projet dépendant du développeur
- Pas de structure cohérente et logique
- Difficulté de maintenance

Introduction générale:
Pourquoi utiliser un Framework en front end? (2)

- Réduire la complexité de développement
- Gain de temps et d'efficacité
- Richesse en animations
- Meilleure structuration du code
- Maintenance simplifiée

C’est quoi Angular?

- **Angular:** framework de développement javascript
- open source
- Développé par Google.
- Créer des applications client (Client-side application) dynamiques et interactives.
- Utilise les langages HTML, CSS et **TypeScript/**Javascript.

Pourquoi Angular?

- ❑ Angular est une solution complète
- ❑ Angular est développé en Typescript
- ❑ Angular est soutenu par Google
- ❑ Composants réutilisables
- ❑ Facilité d'intégration avec d'autres technologies

Historique d'Angular

- **Angular 1 (AngularJS):**
  - Première version de Angular qui était très populaire
  - Développée dans les locaux de Google en 2009
  - Les applications Angular JS sont écrites en JS et basées sur une architecture MVC coté client.
  - Utilise une syntaxe spécifique, qui est différente de celle d'Angular 2 et supérieures.
- **Angular 2 (Angular):**
  - Refonte complète de la syntaxe et de l'architecture d'AngularJS
  - Version officielle sortie en 2016
  - Utilise une architecture basée sur les composants
  - Utilise TypeScript comme langage de programmation
  - Angular 4,5,6,7,8 ,9 ,10 , 11, 12 et 13 sont des simples mises à jours de Angular 2 avec des améliorations au niveau de performance (optimisation du code, prise en charge du lazy loading, compilation partielle, facilité de migration depuis AngularJS,…)
  - Angular 13 est la dernière version stable lancée en Novembre 2021

Caractéristiques d'Angular (1)

1. **Architecture MVVM :** Angular est basé sur le modèle MVVM (Modèle-Vue-VueModèle) permettant une séparation claire des responsabilités et une organisation du code en couches.

2. **Expressive HTML:**
   - ✓ les templates HTML sont étendus avec des directives Angular rajoutant des comportements dynamiques aux éléments du DOM.
   - ✓ On n'as pas besoin de manipuler directement le DOM.
   - ✓ Offre une interface utilisateur claire, lisible et dynamique.

3. **Modularité:** Angular est un framework modulaire:
   - ✓ Meilleure organisation du code
   - ✓ Réutilisation des composants dans différents projets
   - ✓ Réduction du coût et du temps de développement

4. **Routage :** Angular dispose d'un système de routage intégré qui permet la navigation entre les différentes vues de l'application.

Caractéristiques d'Angular (2)

5. Dependency Injection: Angular utilise un système d'injection des dépendances pour fournir des instances de classes et de services aux composants de l'application, au lieu de les instancier directement.

- ✓ Facilité de la réutilisation, la maintenance et les tests du code
- ✓ Réduction de la complexité liée à la gestion des dépendances.

6. **Binding bidirectionnel :** Angular offre un mécanisme de liaison bidirectionnelle et synchronisation des données entre un composant et son template HTML. Cela signifie que toute modification de la valeur d'une propriété du composant est automatiquement reflétée dans le template HTML, et vice versa.

7. **DOM Virtuel:**
   - ✓ Représentation en mémoire du DOM physique
   - ✓ Les modifications se font sur ce DOM virtuel ensuite Angular s'occupe de les synchroniser vers le DOM physique.
   - ✓ Amélioration de la performance et la réactivité de l'application

Caractéristiques d'Angular (3)

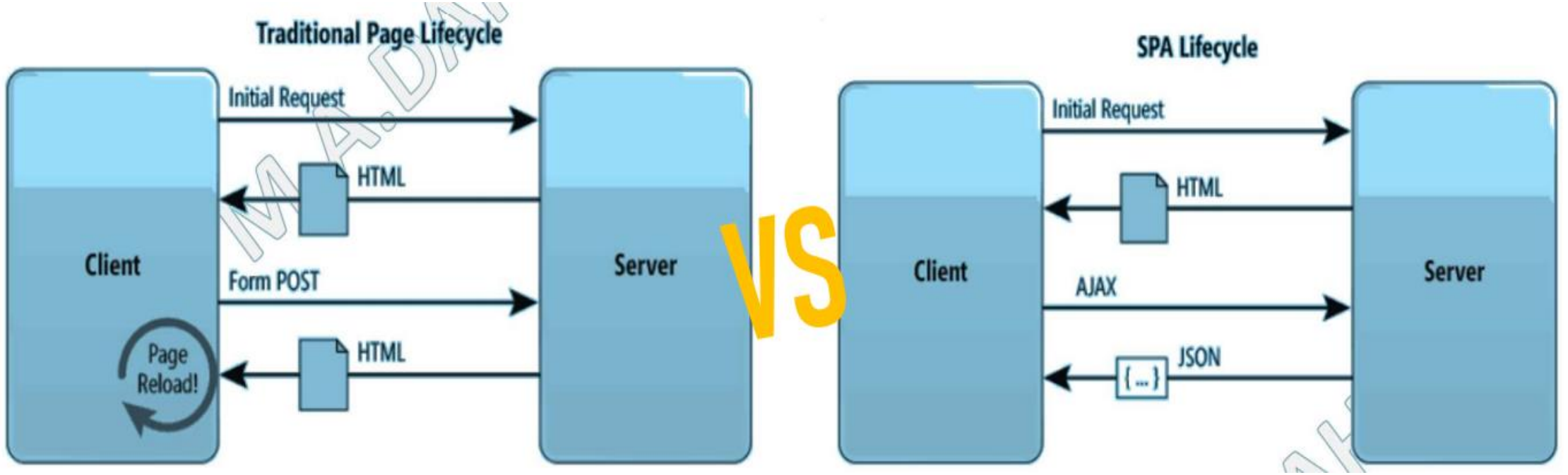
8. Change detector:

- ✓ Utilisé pour synchroniser la vue avec le composant
- ✓ Chaque composant dispose d'un change detector qui surveille les modifications du composant et met à jour la vue correspondante

9. Performance: Angular est conçu pour offrir de bonnes performances grâce à une optimisation du rendu, une compilation AOT (Ahead of Time) et une gestion efficace de la mémoire.

10. Single Page Applications: ne nécessite pas le rechargement de la page lorsqu'un utilisateur interagit avec elle, grâce à la manipulation dynamique du DOM. Seulement les parties modifiées du DOM sont mises à jour.

Application web traditionnelle vs SPA



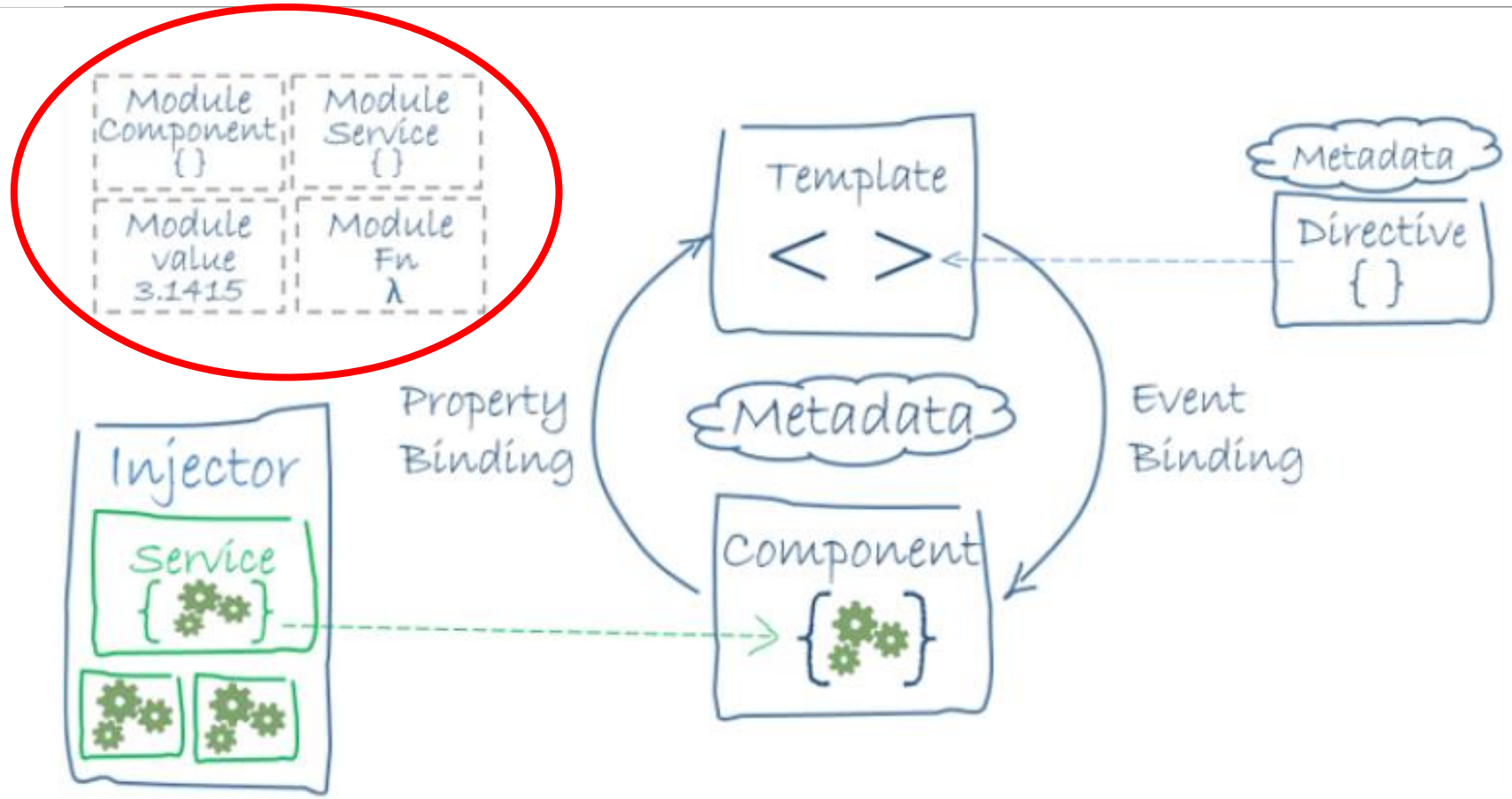
- ✓ La page est rechargée de nouveau **à partir du serveur** pour chaque interaction de l'utilisateur avec l'application.
- ✓ Recharge **complète** de la page pour chaque requête

=> Latence relativement lourde

- ✓ La page web est chargée **une seule fois lors du chargement initial**
- ✓ Lorsque l'utilisateur interagit avec l'application, la SPA utilise JavaScript pour mettre à jour dynamiquement l'interface sans avoir besoin de recharger la page.

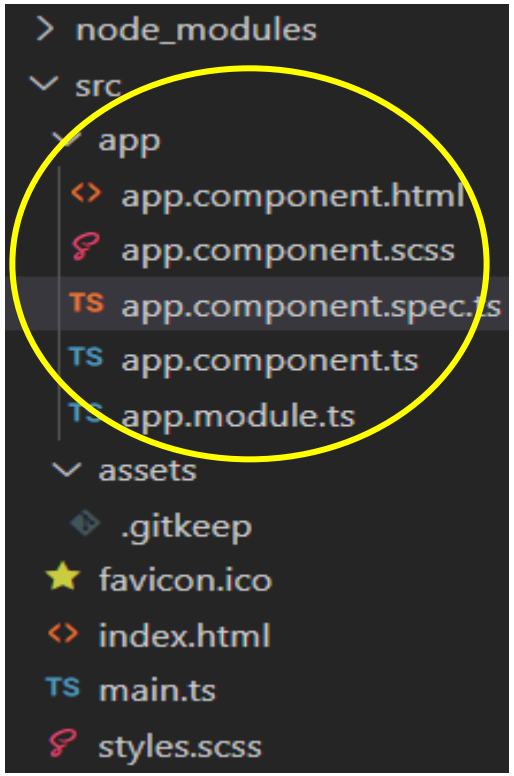
=> Expérience utilisateur plus fluide et réactive

Architecture Angular



Angular est modulaire

- Les modules en Angular sont des conteneurs logiques regroupant des composants, des directives, des services et d'autres fonctionnalités d'une application Angular.
- Les modules permettent de découper une application en plusieurs parties fonctionnelles, ce qui facilite la gestion et le développement de l'application.
- Elle possède au moins un module appelé « module racine » ou « root module ».
- Elle peut contenir d'autres modules à part le module racine.
- Par convention, le module racine s'appelle « AppModule », représenté par le fichier app,module!.ts sous le dossier app.
- Il est créé par défaut lors de la création d'une application angular, et appelé automatiquement lors du lancement de l'application.



Module / NgModule

- Le système de modularité dans Angular est appelé NgModules.
- Un module peut être exporté sous forme de classe.
- La déclaration d'un module en Angular est représentée par une classe décorée par @NgModule.
- Exemples: FormsModule, HttpClientModule, RouterModule

Les décorateurs en angular sont des fonctions qui permettent de modifier des classes Typescript. Ils sont essentiellement utilisés pour attacher des métadonnées à des classes afin que le système connaisse la configuration de ces classes et leurs fonctionnement.

app.module.ts

src > app > app.module.ts > ...

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

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Module / NgModule

- **Déclarations**: les composants, directives, pipes utilisés par ce module.
- **Imports**: les modules internes ou externes utilisés dans ce module
- **Providers**: les services utilisés
- **Bootstrap**: déclare la vue principale de l'application (celle du composant racine).

app.module.ts

src > app > app.module.ts > ...

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

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Préparation de l'environnement

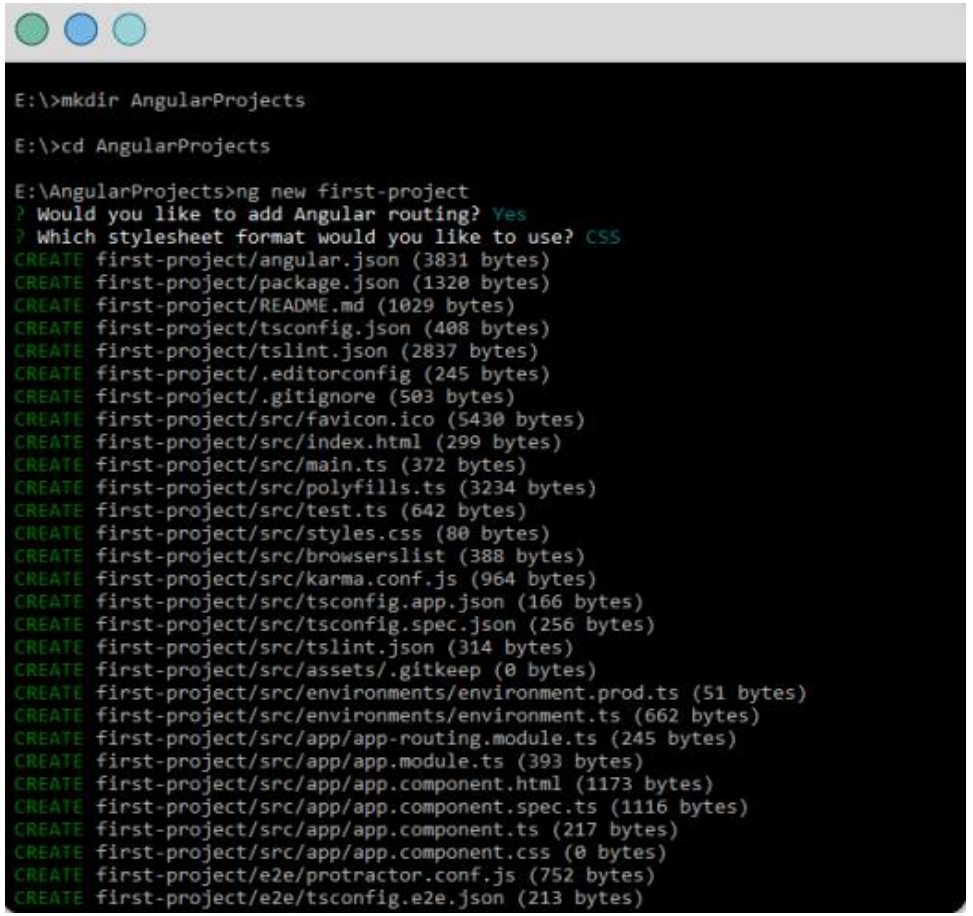
Node Js	www.NodeJs.org
npm	npm install npm@latest
Angular CLI	npm install -g @angular/cli
Editeur	Visualcode, SublimeText, Atom ….

Création d'un projet Angular

1. A un emplacement pratique sur votre ordinateur, créez un dossier nommé AngularProjects et accédez à ce dossier à l'aide de l'invite de commandes.

2. Pour créer une nouvelle application nommée first-project, il suffit d'exécuter la commande suivante :

   ng new first-project



Lancer le projet

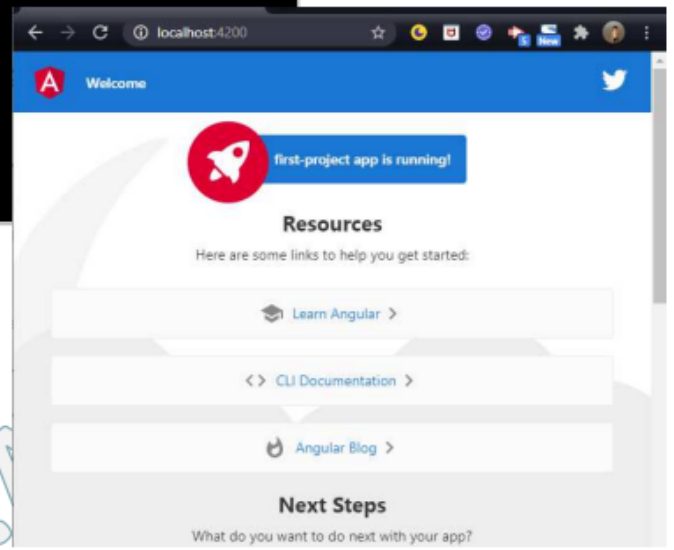
Se déplacer dans le projet

```
E:\AngularProjects>cd first-project

E:\AngularProjects\first-project>ng serve --open
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Date: 2020-10-08T16:33:17.937Z
Hash: 15288ef9618a4cb33766
Time: 8648ms
chunk {main} main.js, main.js.map (main) 12.7 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 236 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.08 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 16.2 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.55 MB [initial] [rendered]
i ｢wdm｣: Compiled successfully.
```

Lancer le projet --open ouvre le navigateur par défaut

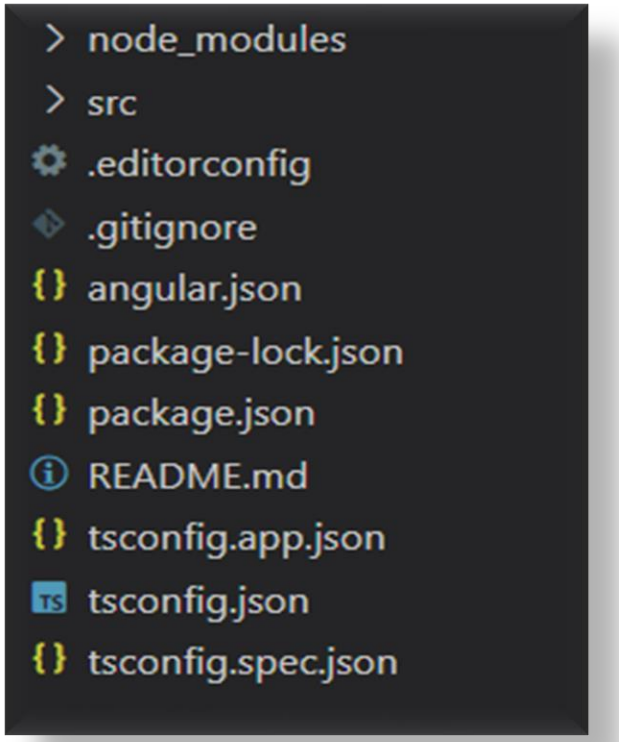


Info

Pour créer un projet sans les fichiers de test (.spec.ts), utiliser la commande: `$ng new ProjectName --skip-tests=true`

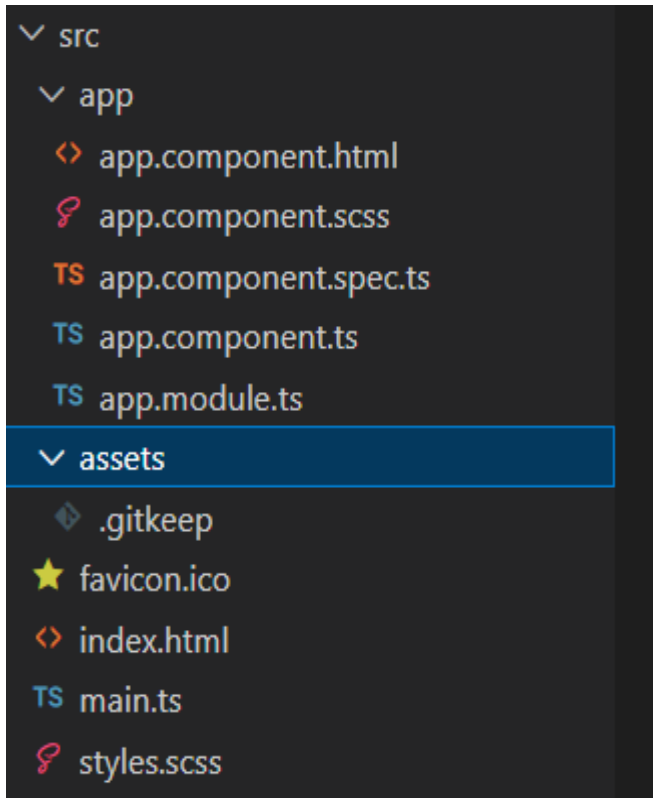
La commande `ng-serve` ouvre un onglet dans votre navigateur par défaut à l'adresse: `http: // localhost: 4200`.

Structure d'un projet Angular



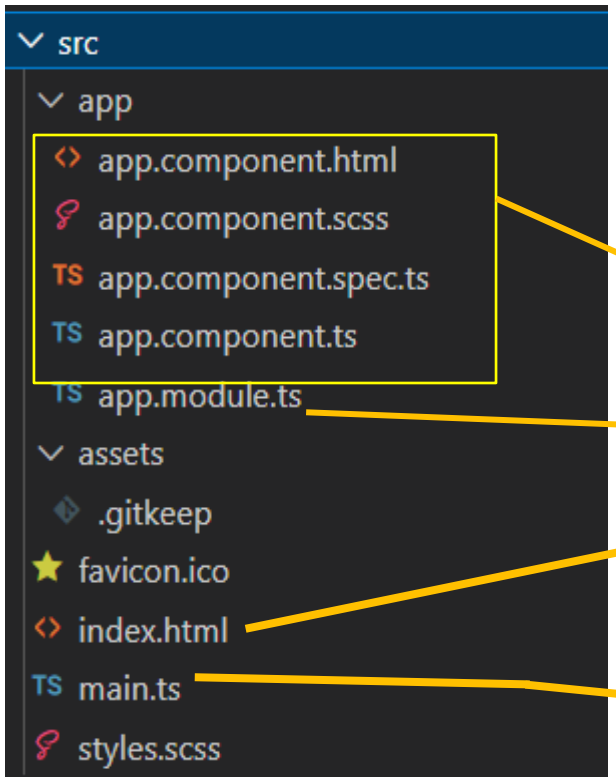
- **Node-modules:** ensemble de bibliothèques et dépendances indispensable pour le bon fonctionnement d'une application angular (forms, http,cli,…)
- **src :** pour les fichiers sources relatifs au projet
- **.gitignore:** définir les dossiers ou fichiers à ignorer lors du chargement du projet sous Git.
- **package.json :** déclaration de la liste de dépendances de l'application
- **angular.json** : contenant les données concernant la configuration du projet (l'emplacement des fichiers de démarrage...)
- **tsconfig.json** : contenant les données de configuration de TypeScript

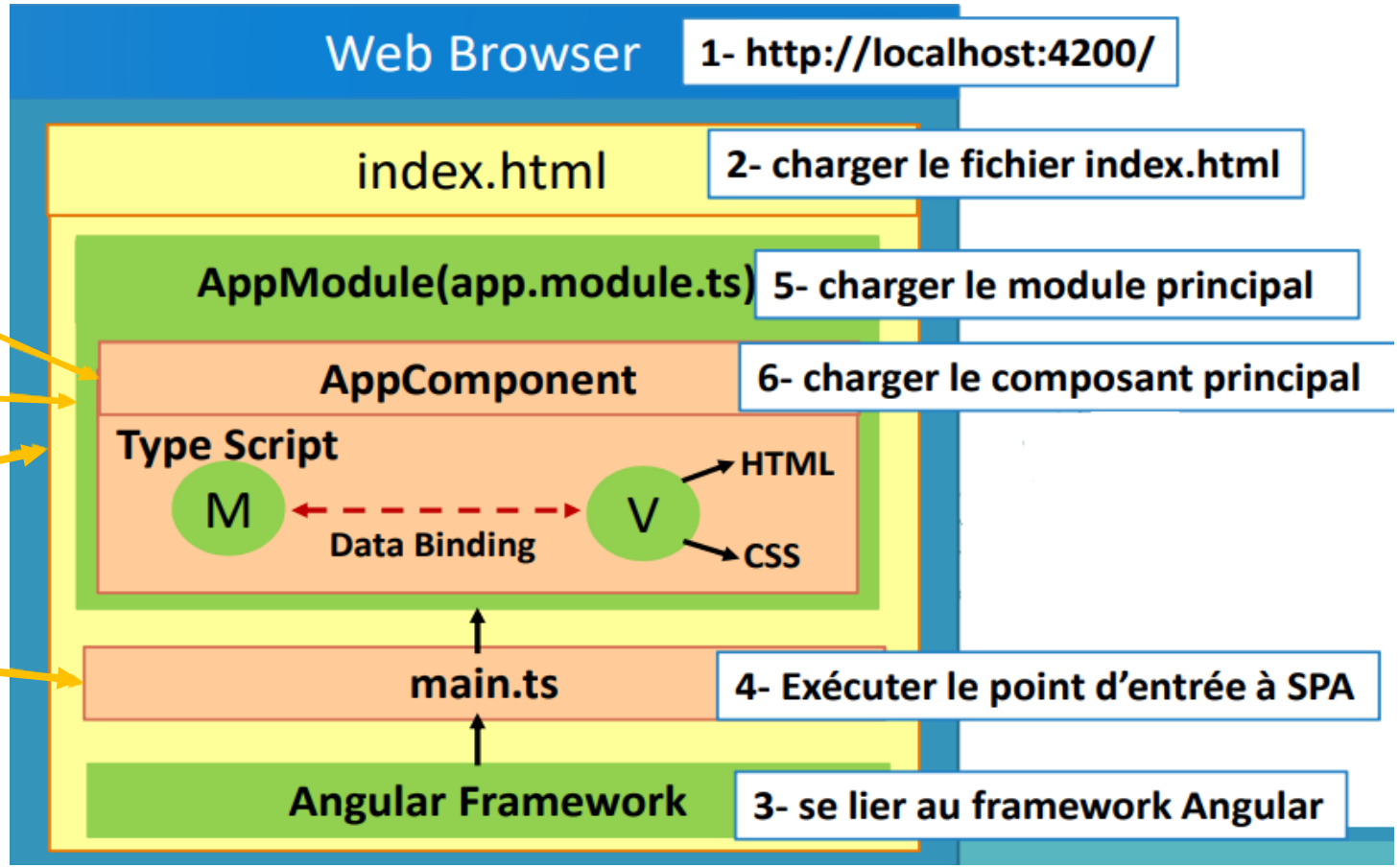
Structure d'un projet Angular: Dossier /Src



- **assets** : contenant les images, les fichiers css, les sons,...
- **index.html** : 1ère page HTML et le fichier de démarrage de l'application
- **styles.css** : la feuille de style commune de tous les composants web de l'application
- **main.ts:** le point d'entrée typescript à l'application
- **app** : contient tous les composants web de l'application. Initialement les 5 fichiers du module principal
  - ✓ **app.module.ts :** la classe correspondante au module principal
  - ✓ **app.component.ts** : la classe associé au composant web
  - ✓ **app.component.html** : le fichier contenant le code HTML associé au composant web
  - ✓ **app.component.css** : le fichier contenant le code CSS associé au composant web
  - ✓ **app.component.spec.ts** : le fichier de test du composant web

Etapes d'exécution d'un projet Angular





Scénario complet (Résumé)

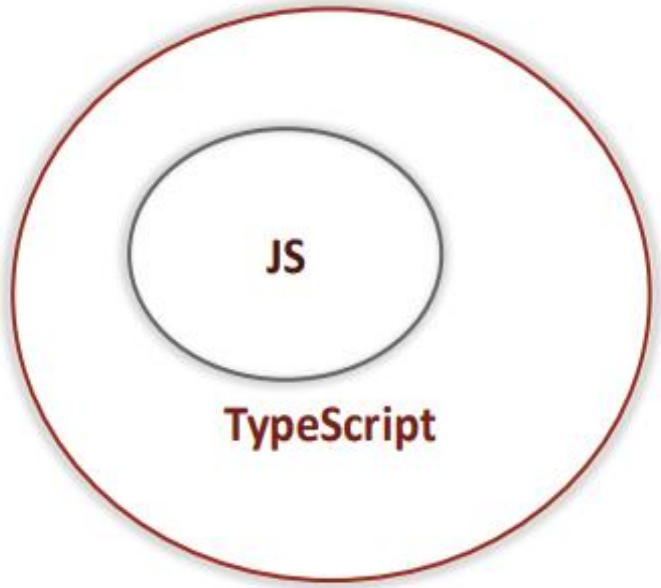
1. La première page à charger c'est la page **index.html**,
2. Angular cherche **main.ts**,
3. À l'intérieur de **main.ts**, il cherche le module principal à déclencher (Dans notre cas: **AppModule**),
4. Il cherche le fichier de ce module (dans notre cas c'est **app.module.ts**)
5. À l'intérieur de **app.module.ts** il cherche le composant principapal à déclencher (dans notre cas c'est **appComponent**)
6. comme on a déjà vu, **appComponent** est composé par une classe et template
7. **appComponent** a la propriété **selector = "app-root"** qui correspond à la balise **`<app-root>`** trouvée dans le fichier **index.html**
8. … tout ça donne le rendu visuel final.

Commandes CLI utilisées fréquemment

Commande	Besoin
ng new firstApp	Créer un projet angular
ng serve –open (oubien -o)	Exécuter un projet (-o pour l'ouvrir automatiquement)
ng g component componentName	Créer un composant
ng g directive directiveName	Créer une directive
ng g service serviceName	Créer un service
ng g class className	Créer une classe

Angular et Typescript

https://www.typescriptlang.org/



Attention, l'extension des fichiers TypeScript est .ts et non .js !

- **TypeScript** est un langage de programmation libre et open-source, développé par Microsoft.

- **Avantages:**
  - **Transpilé**: converti en JS avant d'être exécuté pour qu'il soit supporté par les navigateurs
  - **Typage strict:** oblige à spécifier le type de toutes les variables, contrairement au typage dynamique de JavaScript
  - **Transtypé** : qui permet de s'assurer qu'une variable ou une valeur passée vers ou retournée par une fonction soit du type prévu
  - **POO**: prend en charge les notions d'orientée objet
  - **Riche en fonctionnalités**: fonctions lambda par exemple
  - **Débogage**: il permet de détecter les erreurs syntaxiques à la compilation => réduire considérablement le nombre d'erreurs au runtime

- Angular est développé avec TypeScript.

Un Peu du Typescript
Déclaration des types

https://www.typescriptlang.org/

- **1ère Méthode:** Déclaration explicite de la variable
  - ***Exemple:***
    - let X: number=5; *//déclaration + initialisation*
    - let test:Boolean;

- **2ème Méthode:** la déclaration est détectée via l'affectation
  - ***Exemple:***
    - let X=50; *//le type de X est détecté number via la valeur affectée à X*
    - X=" Hello" //ERREUR
    - Si le type est any, la variable peut prendre n'importe quelle valeur:
    - Exemple: let Y;
      Y=12;
      Y=True;

Un Peu du Typescript
Types de base

https://www.typescriptlang.org/

- Les types primitifs:
  - ✓ **Number:** représente des nombres. Typescript ne différencie pas entre entier, float, décimal… Ils sont tous vus comme étant des nombres simplement.
  - ✓ **Boolean:** pour les deux valeurs true et false
  - ✓ **String**: représente des valeurs de chaîne telles que "Hello, world"
- Les types complexes:
  - ✓ Array
  - ✓ Tuple
  - ✓ Enum
- Le type Any

Un Peu du Typescript
Types de base: les tableaux

https://www.typescriptlang.org/

```typescript
let tab: Array<number> = [1, 2, 3];//Type générique array
let tab1: number[]=[1,2,3] //INDICES ALLANT DE 0 -> TAILLE-1
tab1[0]="toto" //ERRUR!

let tab2: (number|string)[]=[];//TAB2 peut contenir des nombres
                               //ou bien des chaines

tab2[0]="hello" //OK
tab2[1]=15      //OK
tab2[3]=false //ERREUR!
```

Un Peu du Typescript
Types de base: les Tuples

https://www.typescriptlang.org/

Les tuples: permettent d'exprimer un tableau avec un nombre fixe d'éléments dont les types sont connus, mais pas nécessairement identiques.

```
// DECLARATION TUPLE
let x: [string, number];
// INITIALISATION
x = ["hello", 10]; // OK
// INITIALISATION INCORRECTE
x = [10, "hello"]; // ERREUR!!
x[3] = "world";//ERREUR: Accès à un élément en dehors de l'ensemble des index connus
```

Un Peu du Typescript
Types de base: les énumérations

https://www.typescriptlang.org/

Les énumérations: fonctionnalité ajoutée à JavaScript par TypeScript permettant de donner des noms plus conviviaux à des ensembles de valeurs numériques.

```
//Définir une énumération appelée Color
enum Color {
    Red,
    Green,
    Blue,
  }
  let c: Color = Color.Green;//donne 1
//=>Par défaut, les énumérations commencent à numéroter leurs membres à partir de 0.
//On pourra le modifier cela en définissant manuellement les valeurs:
enum Couleur {
    Red = 1,
    Green = 2,
    Blue = 4,
  }
  let c1: Couleur = Couleur.Green;//donne 2
```

Un Peu du Typescript
Les Fonctions

https://www.typescriptlang.org/

Les fonctions: le principal moyen de transmettre des données en JavaScript. TypeScript permet de spécifier les types des valeurs d'entrée et de sortie des fonctions pour garder le principe du typage strict.

```
  // Annotation pour le type de paramètre
  function greet(name: string) {
    console.log("Hello, " + name.toUpperCase() + "!!");
  }
greet("Alex") //OK
greet(14) //ERREUR!

// Annotation pour le type de retour de la fonction
function getFavoriteNumber(): number {
    return 26;
  }
let   z:number =getFavoriteNumber(); //Retourne 26
let ch:string =getFavoriteNumber(); //ERREUR!
```

Un Peu du Typescript
Les Fonctions fléchées

https://www.typescriptlang.org/

Fonctions fléchées: syntaxe plus concise et expressive pour définir des fonctions. Elles sont également connues sous le nom de "lambda functions" dans d'autres langages de programmation.

```
//La syntaxe générale d'une fonction fléchée est la suivante :
(paramètres) => { instructions; } //où paramètres est une liste de paramètres séparés
//par des virgules, expression est une expression qui sera évaluée et renvoyée, et { instructions; }
//est un bloc d'instructions à exécuter.

//Déclaration fonction fléchée
let fct = (x: number) => {x * 2;}
```

Un Peu du Typescript
Fonctions fléchées et tableaux (1)

https://www.typescriptlang.org/

```
//Utiliser forEach pour afficher le contenu d'un tableau
var tab = [2, 3, 5];
tab.forEach(elt => console.log(elt));
// affiche 2 3 5

//Dans forEach, on peut aussi appeler une fonction afficher
tab.forEach(elt => afficher(elt));
function afficher(value) {
console.log(value);
}
// affiche 2 3 5
```

Un Peu du Typescript
Fonctions fléchées et tableaux (2)

https://www.typescriptlang.org/

```
/*On peut utiliser map pour effecteur un traitement sur chaque elément
du tableau puis forEach pour afficher le nouveau tableau */
tab.map(elt => elt + 3)
.forEach(elt => console.log(elt));
// affiche 5 6 8

/*On peut aussi utiliser filter pour filtrer des eléments*/
tab.map(elt => elt + 3)
.filter(elt => elt > 5)
.forEach(elt => console.log(elt));
// affiche 6 8

/*Attention, selon l'ordre d'appel de ces methodes,
le résultat peut changer. */
```

Un Peu du Typescript
Les interfaces

https://www.typescriptlang.org/

les interfaces: structures permettant de définir la forme ou la structure d'un objet sans passer par les classes. Elles sont souvent utilisées pour décrire les types d'objets que les fonctions ou les classes s'attendent à recevoir ou à retourner.

EXEMPLE 1

```
interface Person {
  firstName: string;
  lastName: string;
  age?: number;
}
```

```
let person: Person = {
    firstName: 'John',
    lastName: 'Doe'
  };
```

EXEMPLE 2

```
interface Contact {
  nom :string
  prenom: string
  numero: number
  photo: string
  email: string
  }
```

```
let createContact = (contact: Contact) => { }
createContact({
nom: "ben Foulen",
prenom: "Foulen",
numero : 123456,
photo: "url",
email: "hhj@gjhcom"
});
```

Un Peu du Typescript
Les Classes

https://www.typescriptlang.org/

les classes: Regroupement des méthodes et des attributs dans le même endroit => C'est le concept d'Orienté Objet

```
//Définir Classe Contact
class Contact {
    //Propriétés
    nom :string; //public par défaut
    prenom: string;
    //Méthodes
    afficheContact(){
    console.log( "nom: " + this.nom, "prenom : " + this.prenom);
    }
    }
```

```
//Instanciation d'un objet de type Contact
let contact = new Contact ;
contact.prenom= "Alex";
contact.nom= "Smith";
contact.afficheContact();
```

Résumé

- Une application Angular possède au moins un module, le root module, nommé par convention AppModule.

- Un module Angular est défini simplement avec une classe (généralement vide) et le décorateur NgModule.

- Un module Angular est un mécanisme permettant de:
  - ✓ Regrouper des composants (mais aussi des services, directives, pipes…)
  - ✓ Définir leurs dépendances
  - ✓ Définir leur visibilité