Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Centre Universitaire de Mila
Institut des Sciences et de la Technologie

Développement Web Avancé

Chapitre 2 : Les frameworks de développement Web

Département MI

s.meghzili@centre-univ-mila.dz

Plan

- **Chapitre 2 : Les frameworks de développement Web**

  - *Introduction*
  - *Modèle Vue Contrôleur (MVC)*
  - *Framework Angular*
  - *Etude de cas* 1
  - *Framework Spring Boot*
  - *Framework Hibernate (Gestion des bases de données)*
  - *Etude de cas 2*

Qu'est-ce qu'un framework Web ?

- **Un framework web est un framework logiciel conçu pour simplifier votre vie de développement web.**

- **Des cadres existent pour vous éviter d'avoir à réinventer la roue et aider à atténuer certains des frais généraux lorsque construction d'un nouveau site.**

Framework du développement web

Le framework

≈

« cadre de travail » ou « cadre de développement »

≈

« architecture prête à l'emploi »

≈

« ensemble d'outils constituant les fondations d'une application »

≈

« c'est à la fois une sorte de boîte à outils et une méthodologie de travail »

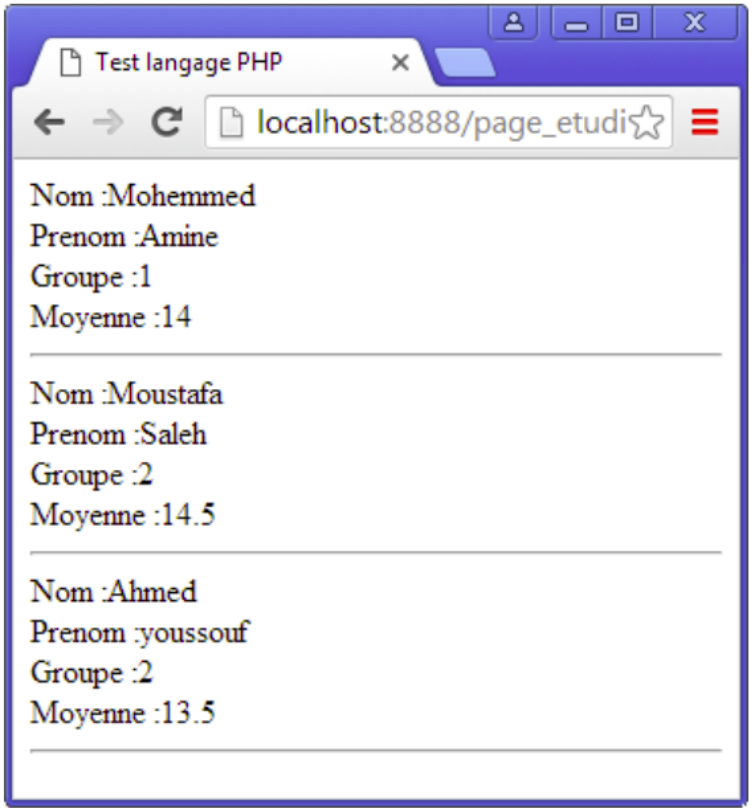
Framework du développement web

Avantages des frameworks

- *La rapidité de développement des projets grâce aux outilles disponibles*
- *Une très bonne organisation des projets*
- *Plusieurs frameworks sont basé sur l'architecture MVC, ils bénéficient donc de ses avantages*
- *Composants réutilisables*
- *Respecter les norme actuelles (sécurité , nouvelles technologies)*

Framework MVC : exemple introductif (1)

Pour commencer nous allons prendre un exemple d'un programme PHP qui permet d'afficher une liste d'étudiants a partir d'une base de données. Nous allons voir plusieurs version du programme et déterminer la version la plus efficace.



Framework MVC : exemple introductif (2)

```php
<?php
    $ma_connexion = mysql_connect ('localhost', 'root', '');   //Connexion au SGBD MySQL
    mysql_select_db ('universite', $ma_connexion) ; // selection de la base université
    //Préparer la requette SQL
    $sql = 'SELECT * FROM etudiants;';
    //Exécuter la requette SQL
    $resultat = mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error());

?>


<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
        <head>
            <title>Test langage PHP</title>
            <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        </head>
<body>
    <?php
    //Percourir chaque tuple du résultat et l'afficher
    while($etud = mysql_fetch_row($resultat))
    {
    echo 'Nom :'.$etud['0'].'<br/>';
    echo 'Prenom :'.$etud['1'].'<br/>';
    echo 'Groupe :'.$etud['2'].'<br/>';
    echo 'Moyenne :'.$etud['3'].'<br/>';
    echo '<hr/>';
    }
    ?>
```

Version 1 : Un seul fichier

Framework MVC : exemple introductif (3)

```php
<?php
    $ma_connexion = mysql_connect ('localhost', 'root', '');   //Connexion au SGBD MySQL
    mysql_select_db ('universite', $ma_connexion) ; // selection de la base université
    //Préparer la requette SQL
    $sql = 'SELECT * FROM etudiants;';
    //Exécuter la requette SQL
    $resultat = mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error());
    mysql_close(); //fermer la connexion
    require ('vue1.php');
?>
```

« page_etudiants.php »

```php
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
        <head>
            <title>Test langage PHP</title>
            <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        </head>
<body>
    <?php
    //Percourir chaque tuple du résultat et l'afficher
    while($etud = mysql_fetch_row($resultat))
    {
    echo 'Nom :'.$etud['0'].'<br/>';
    echo 'Prenom :'.$etud['1'].'<br/>';
    echo 'Groupe :'.$etud['2'].'<br/>';
    echo 'Moyenne :'.$etud['3'].'<br/>';
    echo '<hr/>';
    }
    ?>
</body>
</html>
```

« vue1.php »

Version 2 : deux fichiers

Framework MVC : exemple introductif (4)

```php
<?php
    function get_etudiants()
    {
    $ma_connexion = mysql_connect ('localhost', 'root', '');   //Connexion au SGBD MySQL
    mysql_select_db ('universite', $ma_connexion) ; // selection de la base université
    //Préparer la requette SQL
    $sql = 'SELECT * FROM etudiants;';
    //Exécuter la requette SQL
    $data = mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error());

    return $data;    // retourner le resultat extrait de la base de données
    }
?>
```

« modele_etudiants.php »
le modèle

Version 3 : trois fichiers

```php
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
        <head>
            <title>Test langage PHP</title>
            <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        </head>
<body>
    <?php
    //Percourir chaque tuple du résultat et l'afficher
    while($etud = mysql_fetch_row($resultat))
    {
    echo 'Nom :'.$etud['0'].'<br/>';
    echo 'Prenom :'.$etud['1'].'<br/>';
    echo 'Groupe :'.$etud['2'].'<br/>';
    echo 'Moyenne :'.$etud['3'].'<br/>';
    echo '<hr/>';
    }
    ?>
</body>
</html>
```

« vue1.php »
la vue

```php
<?php
    require 'modele_etudiants.php';
    $resultat = get_etudiants();
    require 'vue1.php';
?>
```

« page_etudiants.php »
le contrôleur

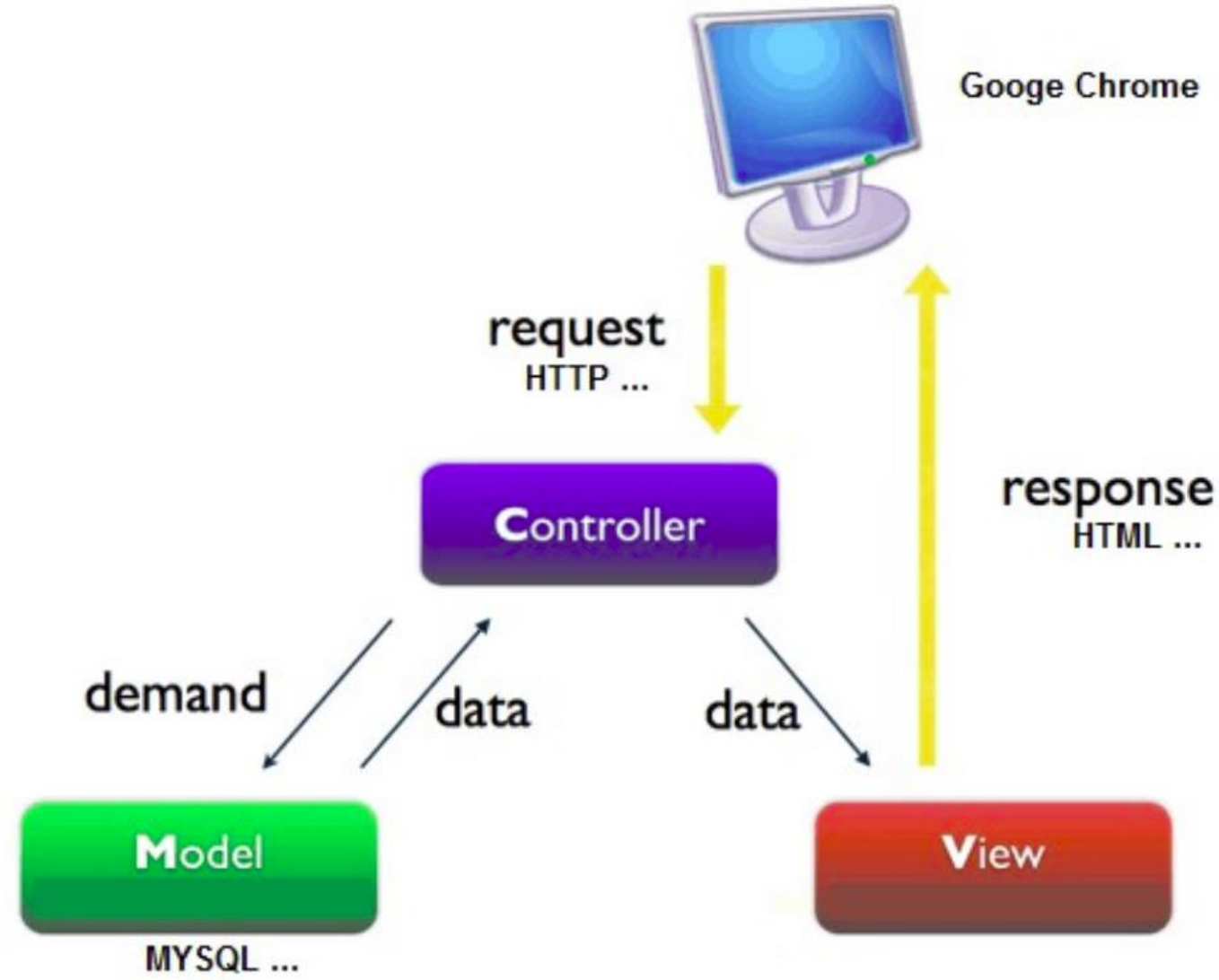
Framework MVC

Modèle : gestion des données de l'application

Vue : Affichage et interaction avec l'utilisateur

Contrôleur : gestion de la logique du code (il prend les décisions)

Modèle MVC



L'architecture MVC (Modèle-Vue-Contrôleur)

Avantages du MVC

- **Une conception claire et efficace**
- **Un gain de temps de maintenance et d'évolution du site (l'ajout et mise à jour des fonctionnalités est très facile)**
- **Une modularité qui offre une grande souplesse pour organiser le développement du site entre différents développeurs**
- **Une rapidité de développement**

Framework Angular :

1. **Introduction**
2. **Installation et configuration**
3. **Data binding**
4. **Directives Angular**
5. **Exemple d'une application Angular:**
   - **Composants**
   - **Routes**
   - **Liens**
   - **Passage de paramètres**
   - **Exemple sur les opérations arithmétiques**
   - **Formulaires**
   - **Services**

Angular : introduction

- **Framework front end de Google permettant de créer des applications clientes web et mobile en HTML et en TypeScript**

Facilite la création de Single Page Application (SPA)

Rends l'HTML dynamique

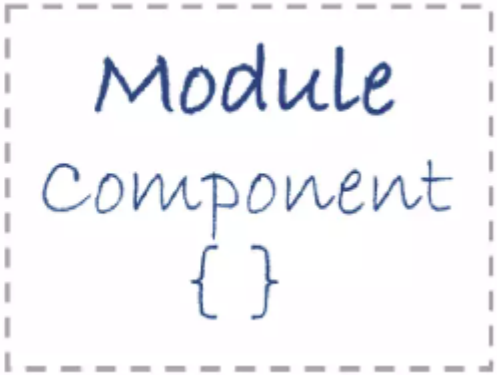
L'architecture d'Angular

Module

- C'est la base d'une application Angular
- Un module peut représenter le tout ou une partie de votre application

- Un module peut contenir les éléments suivants :
  - Component
  - Service
  - Directive
  - Pipe

- Un module peut être dépendant d'un ou plusieurs autre(s) module(s)
- Un module peut être partagé à d'autres modules



L'architecture d'Angular

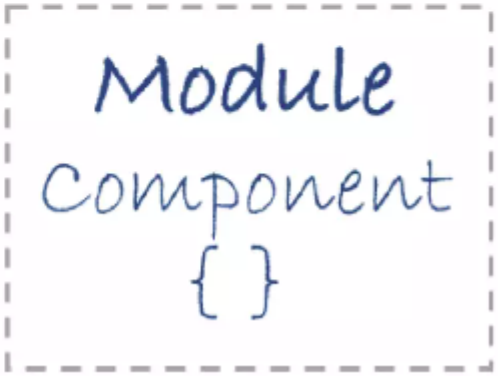
Component

- Dans Angular, tout est composant
- Un composant est une balise HTML personnalisée (ex: app-root, app-shop, app-login etc.)

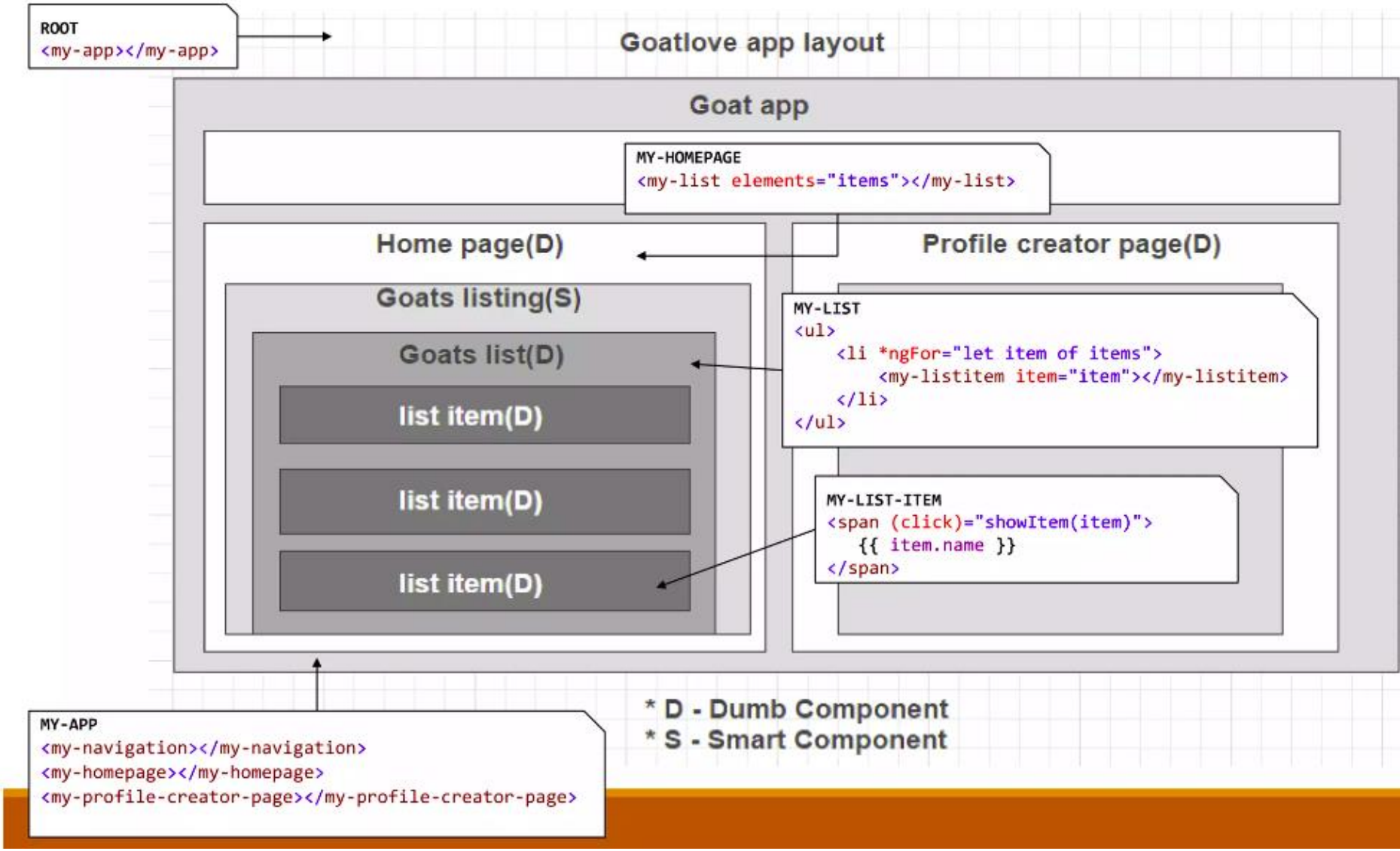
- Définit par :
  - 1 sélecteur (le nom de la balise)
  - 1 template
  - *1 ou plusieurs fichiers de style CSS (facultatif)*

- Représenté par :
  - 1 fichier Typescript (.ts)
  - 1 fichier HTML (.html)
  - *1 ou plusieurs fichiers de style CSS (facultatif)*

- Un component peut utiliser d'autres components



Angular: introduction



Outillage

```
npm install -g typescript
```

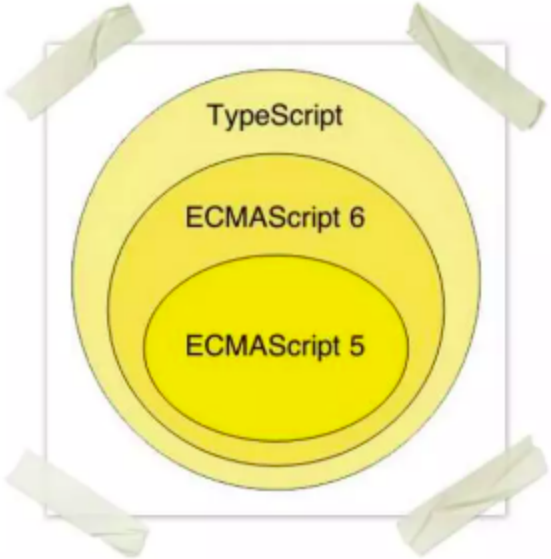
Typescript

Language de programmation libre et open-source développé par Microsoft

But initial : simplifier la production de code Javascript

Permet d'utiliser tous les concepts orienté objets
(classes, interfaces, héritages, public/private etc.)

Le code écrit en Typescript est transpilé en Javascript



Site officiel : https://www.typescriptlang.org

Qu'apporte TypeScript par rapport à Javascript : http://bit.ly/2rMQups

```
npm install -g @angular/cli
```

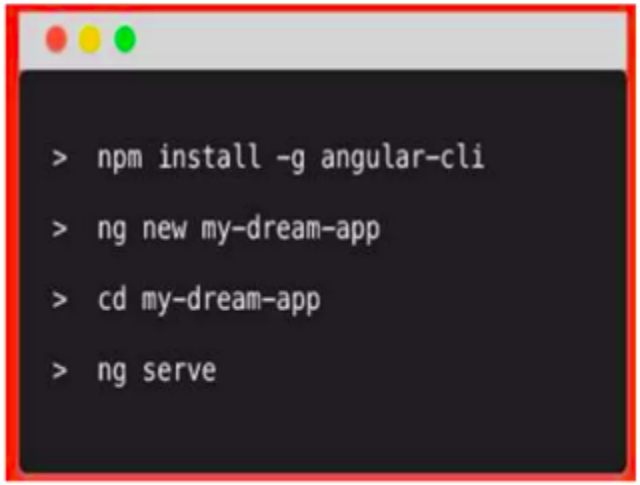
Outillage

Angular CLI

Outil en ligne de commande pour simplifier les tâches de développement avec Angular.

Fonctionnalités :

création d'un projet, génération de composants, exécution des tests, lancement du serveur, déploiement en production…



- **Installation et configuration:**

- Pour créer un nouveau projet nous devons bien évidemment disposer de certains éléments :

  - **Node.js** : La plateforme javascript
  - **Angular CLI** : L'outil fourni par Angular.
  - **Visual Studio code** : Un éditeur de code.

• Installation et configuration:

- **Etapes:**

1. **Installer le NodeJS en exécutant le fichier node-v16.14.2-x64.msi**
2. **Tester l'installation dans l'invite de commande, vous tapez : *npm -v***
3. **Installer l'éditeur du texte VS code**
4. **Installer Angular en utilisant la commande : npm install -g @angular/cli**
5. **Voir la version de Angular, vous tapez : *ng --v***
6. **Créer le premier projet Angular, par exemple : ng new *marchapp***
7. ***Exécuter ce projet en utilisant les commandes : cd / marchapp, ng serve***
8. **Ouvrir le navigateur puis vous tapez : *http://localhost:4200***
9. **Ouvrir ce projet en utilisant VSCode et explorer sa structure**

Angular : Configuration

- **.editorconfig** : configuration pour les éditeurs de code

- .**gitignore** : spécifie les fichiers intentionnellement non suivis que **Git** doit ignorer.
- **README.md** : documentation d'introduction pour l'application racine.

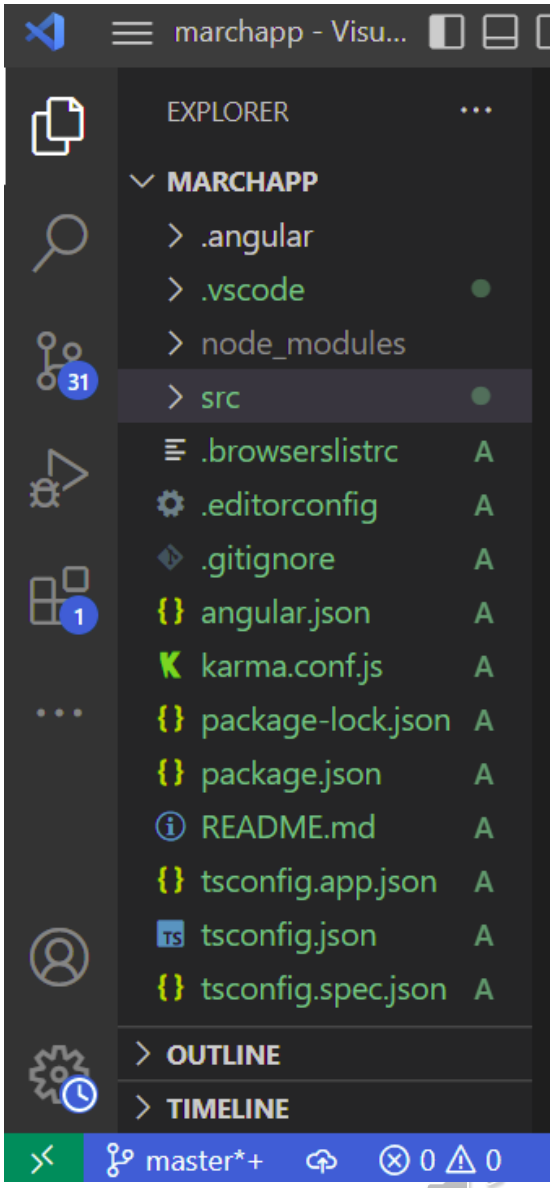
- **angular.json** : configuration par défaut de la CLI pour tous les projets de l'espace de travail, y compris les options de configuration pour les outils de génération, de service et de test utilisés par la CLI

- **package.json** : configure les dépendances de package ***npm*** qui sont disponibles pour tous les projets de l'espace de travail.

- **package-lock.json** : fournit des informations sur la version de tous les packages installés dans **node_modules** par le client npm.

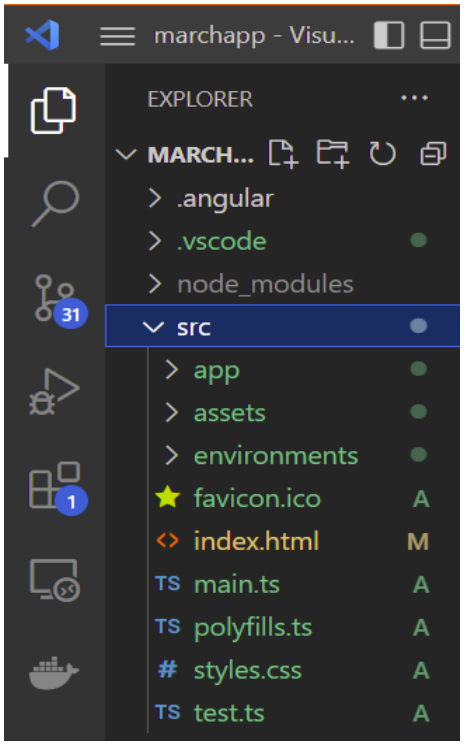
- **src/** : fichiers source pour le projet d'application de niveau racine.
- **node_modules/** : Fournit des packages npm à l'ensemble de l'espace de travail. Les dépendances **node_modules** à l'échelle de l'espace de travail sont visibles pour tous les projets.

- **tsconfig.json** : La configuration **TypeScript** de base pour les projets dans l'espace de travail. Tous les autres fichiers de configuration héritent de ce fichier de base.



Angular : dossier src/

- **app/** : contient les fichiers de **composants** dans lesquels la logique et les données de votre application sont définies.



- **assets/ :** Contient des **images** et d'autres fichiers d'actifs à copier tels quels lorsque vous créez votre application.
- **favicon.ico:** Une icône à utiliser pour cette application dans la barre de favoris.

- **index.html :** la page **HTML principale** qui est servie lorsqu'un internaute visite votre site. La CLI ajoute automatiquement tous les fichiers JavaScript et CSS lors de la création de votre application, vous n'avez donc généralement pas besoin d'ajouter manuellement des balises <script> ou <link> ici.

- **main.ts :** le point d'entrée principal de votre application. Compile l'application avec le compilateur JIT et démarre le module racine de l'application (**AppModule**) pour qu'il s'exécute dans le navigateur.

- **styles.css** : répertorie les fichiers CSS qui fournissent des styles pour un projet.

Angular : dossier /src/app/

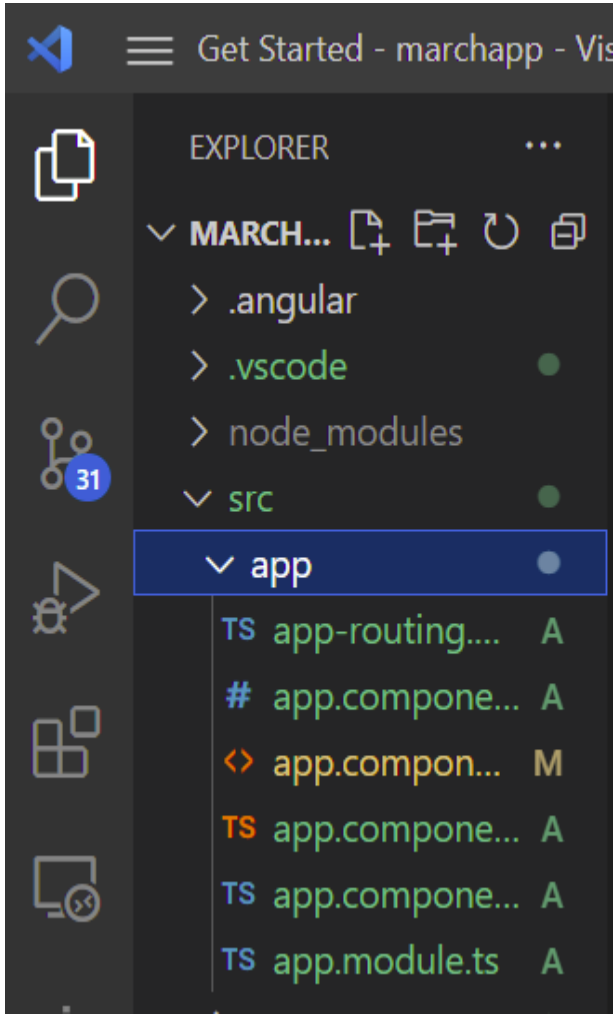
- ***app/app.component.ts*** : définit la logique du composant racine de l'application, nommé **AppComponent**. La vue associée à ce composant racine devient la racine de la hiérarchie des vues lorsque vous ajoutez des composants et des services à votre application.

- ***app/app.component.html*** : définit le modèle **HTML** associé à l'AppComponent racine.

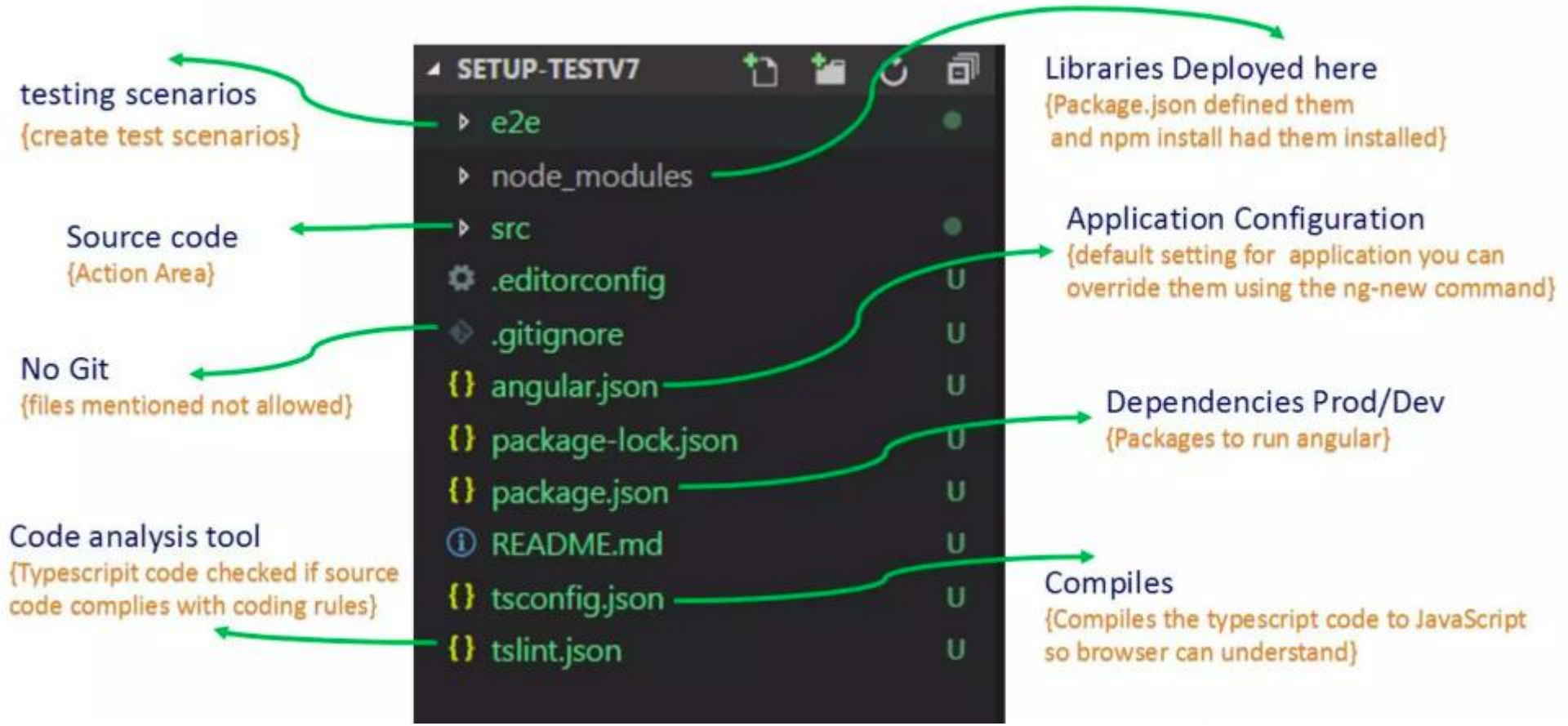
- ***app/app.component.css*** : définit la feuille de style **CSS** de base pour l'AppComponent racine.

- ***app/app.component.spec.ts*** : définit un test unitaire pour l'AppComponent racine.

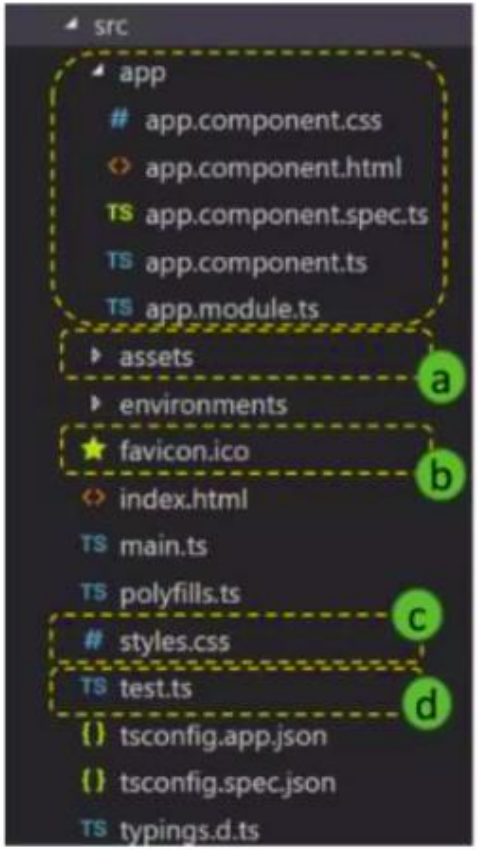
- ***app/app.module.ts*** : définit le module racine, nommé **AppModule**, qui indique à Angular comment assembler l'application. Déclare initialement uniquement AppComponent. Au fur et à mesure que vous ajoutez des composants à l'application, ils doivent **<u>être déclarés ici</u>**.

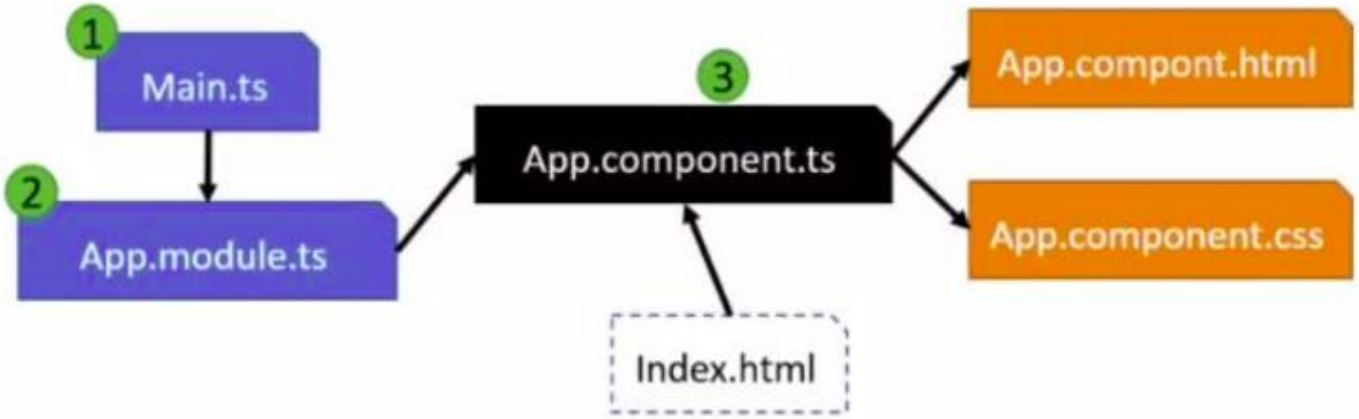


- **Structure du projet**

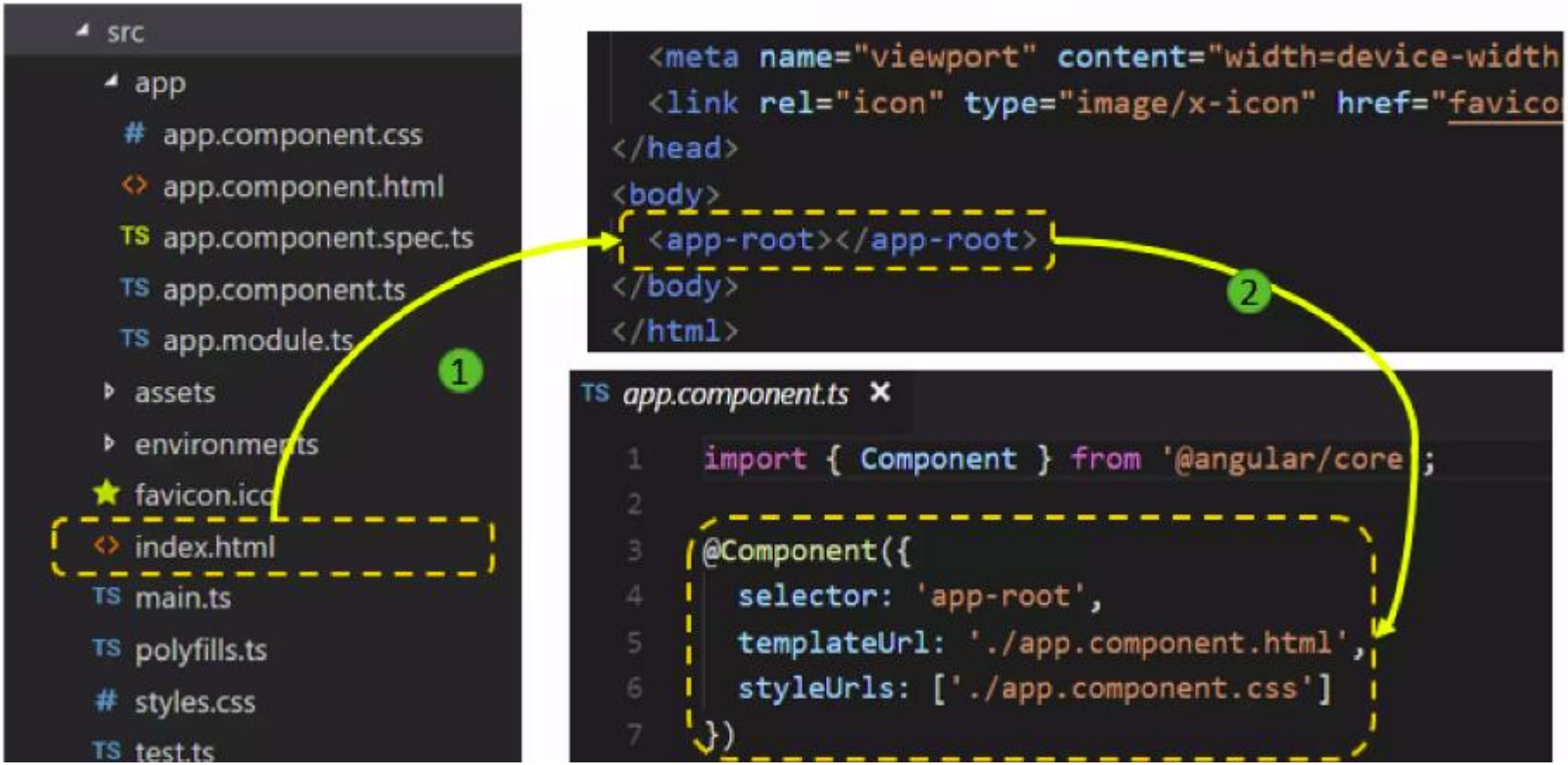


STRUCTURE PROJET ANGULAR DETAILLÉE

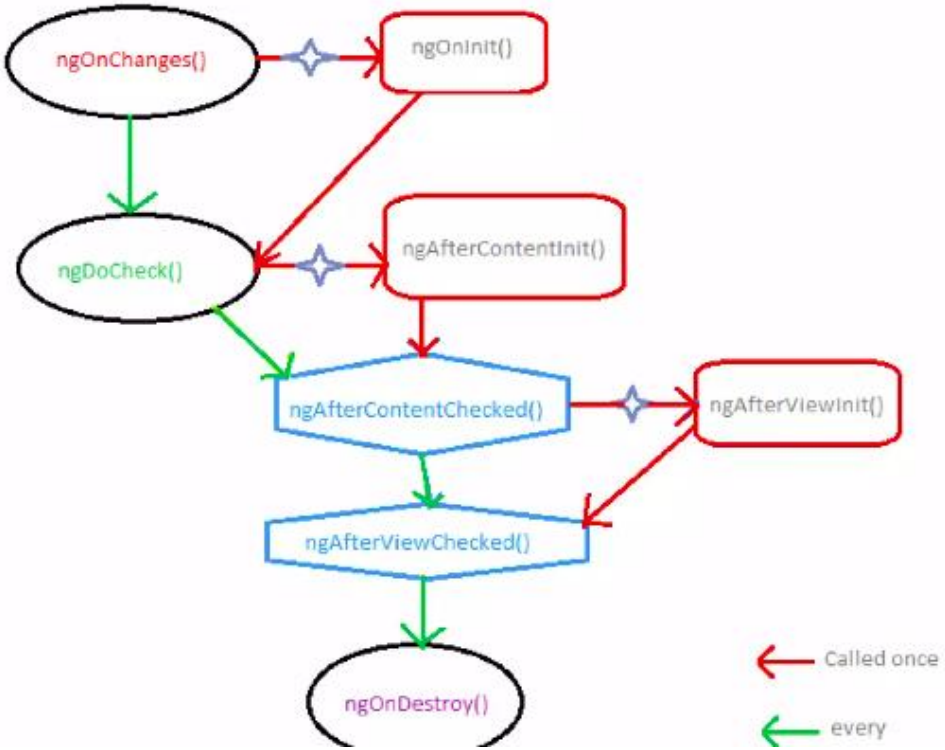




SELECTEUR


src
app
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
assets
environments
favicon.ico
index.html
main.ts
polyfills.ts
styles.css
test.ts
1
<meta name="viewport" content="width=device-width
<link rel="icon" type="image/x-icon" href="favico
</head>
<body>
<app-root></app-root>
</body>
</html>
2
app.component.ts
import { Component } from '@angular/core';
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})

Methode prédefinie dans l'API @angular/core



Angular: Data Binding

- *Insertion dynamique de donnée dans l'application*
- ***One-way** data binding injecte du contenu dans la **vue** : <h1>{{name }}</h1>*
- ***Property** binding : <button [disabled] = « ! **isValid** »> Valider </button>*
- ***Two-way** data binding : <input [(name)] = « newName » />*
- ***Event** binding : <button (click) = 'envoyer()' > Envoyer </button>*

Directives Angular

- *Interaction direct avec le DOM de la page **HTML***
- ***Ajout – suppression – modification** des éléments au cours de l'exécution de la page*
- ***NgIf** rend **oui** ou **non** un élément **HTML***

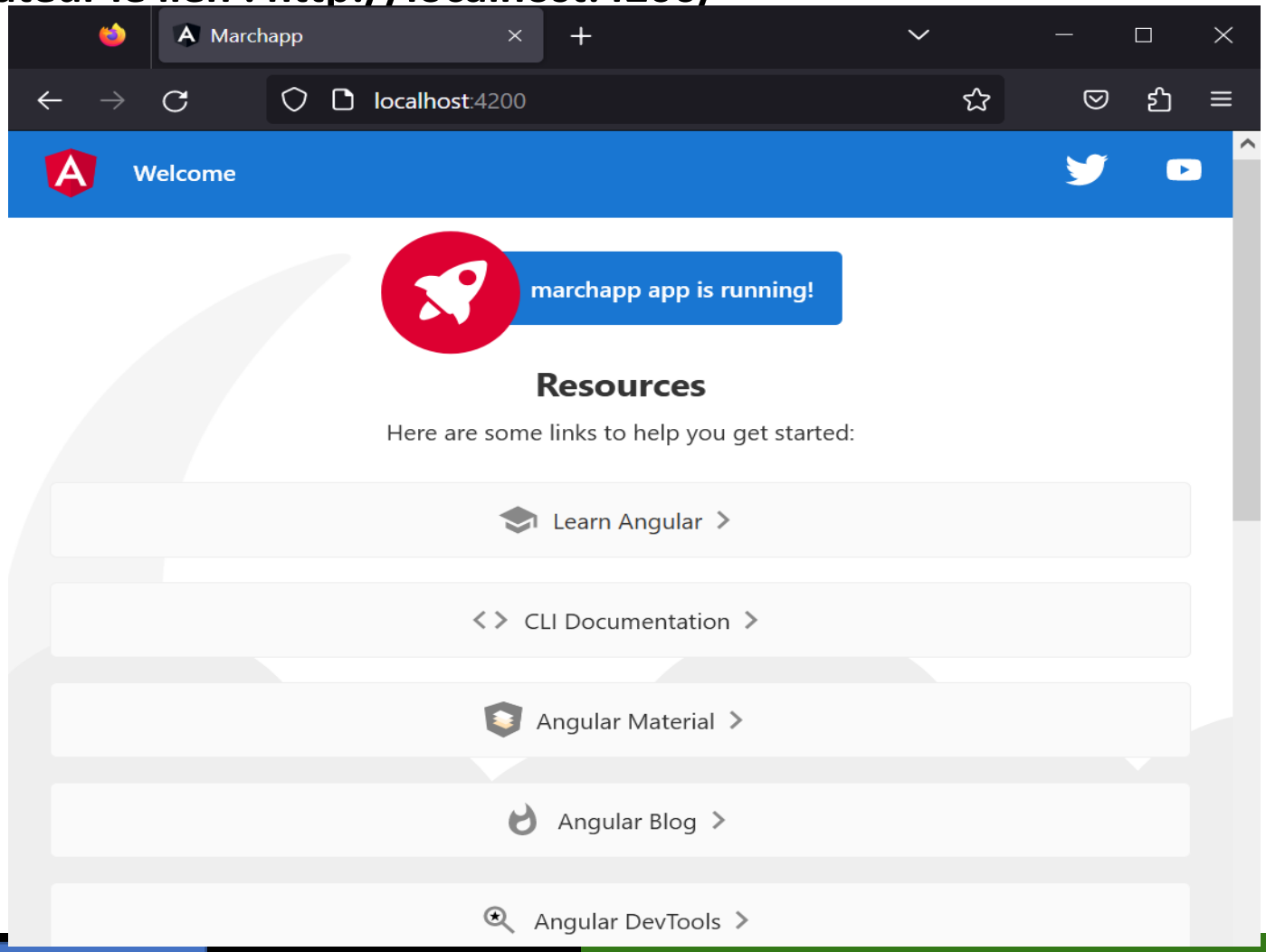
```
<div *NgIf= « condition »> Hello World </div>
```

- ***NgFor** itère sur une **collection** afin d'appliquer un template*

```
<li *ngFor = « let idea of ideas » > …… </li>
```

- **Démarrage du serveur:** *ng serve*

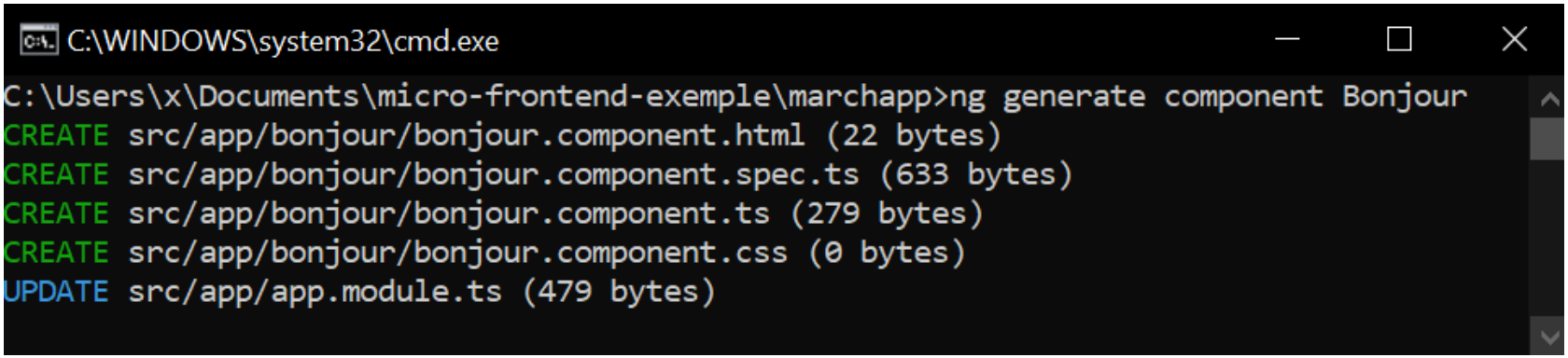
Pour vérifier que l'installation s'est bien déroulé, on ouvre avec un navigateur le lien : http://localhost:4200/



Angular: les composants

- *Exemple:*

Création du composant Bonjour : **ng generate component Bonjour**



Angular: les routes

- **Système de routage : Les routes dans Angular sont gérés par le fichier «app-routing.module.ts »**

app-routing.module.ts M

src > app > app-routing.module.ts > routes

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BonjourComponent } from './bonjour/bonjour.component';
const routes: Routes = [
  { path: 'bonjour', component: BonjourComponent},
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

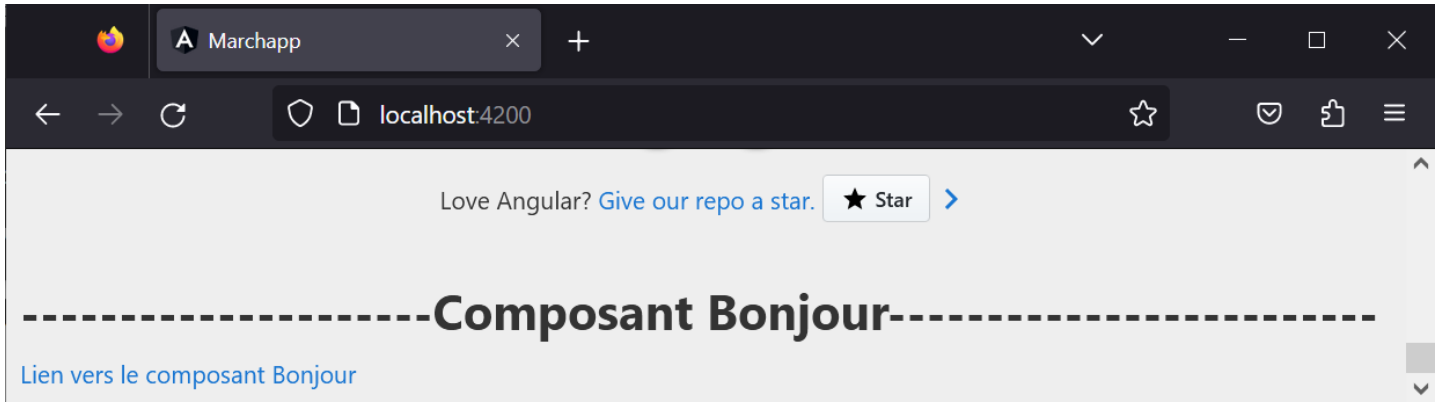
Angular: les liens

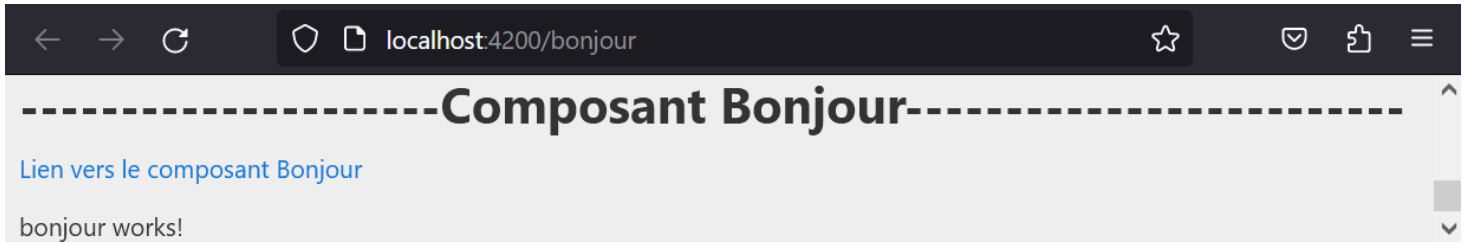
Exemple : On veut créer un lien vers le composant Bonjour : « /bonjour », pour cela on ajoute au fichier « app.component.html » le code suivant :

app.component.html M

src > app > app.component.html > ...

```
474   </div>
475   <div><h1>--------------------Composant Bonjour-----------------------</h1>
476     <a href="/bonjour" target="_blank" rel="noopener"> Lien vers le composant Bonjour
477
478     </a>
479   </div>
```





Angular : passage de paramètres (1)

- **Système de routage (passage de paramètres) :**

On peut passer des paramètre avec le lien URI comme suit :

app-routing.module.ts M

src > app > app-routing.module.ts > AppRoutingModule

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BonjourComponent } from './bonjour/bonjour.component';
const routes: Routes = [
  { path: 'bonjour/:param1/:param2', component: BonjourComponent},
];
```

On veut calculer la somme de param1+param2, on modifier les deux fichier Bonjour.component.ts (contrôleur) et Bonjour.component.html (Vue):

bonjour.component.ts U | bonjour.component.html U

src > app > bonjour > bonjour.component.html > div > p

```html
<div>
    <p>Somme = {{somme}}</p>
</div>
```

Angular : passage de paramètres (2)

bonjour.component.ts U

src > app > bonjour > TS bonjour.component.ts > BonjourComponent > constructor

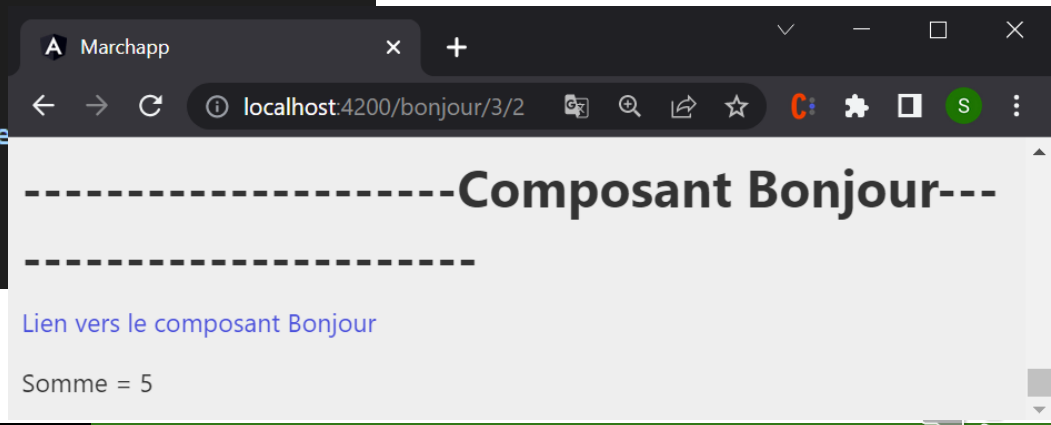
```ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-bonjour',
  templateUrl: './bonjour.component.html',
  styleUrls: ['./bonjour.component.css']
})

export class BonjourComponent implements OnInit {
somme :number=0;

constructor(private activatedRoute: ActivatedRoute) {
    this.activatedRoute.params.subscribe(params => {
      let p1:number = params['param1'];
      let p2:number = params['param2'];

        this.somme= +p1 + +p2;
        console.log("La some = "+this.somme
      }
      );
    }
```



Angular : exemple des opération arithmétique

- *On veut améliorer notre exemple en ajoutant d'autres types d'opération : Soustraction (-), multiplication (*) et la division (/) de deux nombres.*

- *La Première modification est dans le fichier app-routing.module.ts en ajoutant le paramètre 3: typeop:*

```
4  const routes: Routes = [
5    { path: 'bonjour/:param1/:param2/:typeop', component: BonjourComponent},
6  ];
```

- *Puis, on modifie les deux fichier Bonjour.component.ts (contrôleur) et Bonjour.component.html (Vue):*

TS bonjour.component.ts U <> bonjour.component.html U

src > app > bonjour > <> bonjour.component.html > div

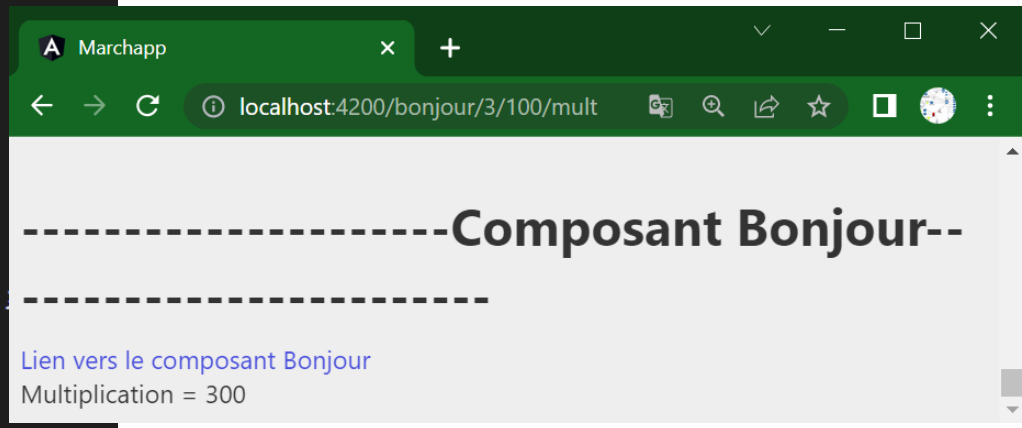
```
1  <div *ngIf="isSome">Somme = {{resultat}}</div>
2  <div *ngIf="isDiv">Division = {{resultat}}</div>
3  <div *ngIf="isSous">Soustraction = {{resultat}}</div>
4  <div *ngIf="isMult">Multiplication = {{resultat}}</div>
```

Angular : exemple des opération arithmétique

TS bonjour.component.ts U

src > app > bonjour > TS bonjour.component.ts > BonjourComponent >

```
9
10  export class BonjourComponent implements OnInit {
11  resultat :number=0;
12  isSome : boolean=false;
13  isSous : boolean=false;
14  isMult : boolean=false;
15  isDiv : boolean=false;
16  typeoperation:any;
17
18  constructor(private activatedRoute: ActivatedRoute) {
19      this.activatedRoute.params.subscribe(params => {
20        let p1:number = params['param1'];
21        let p2:number = params['param2'];
22        let operation = params['typeop'];
23
24        if (operation =="some")
25       { this.resultat= this.Some(p1, p2);
26        this.isSome=true ;
27       }
28       if (operation =="sous")
29       { this.resultat= this.soustraction(p1, p2);
30        this.isSous=true ;
31      }
32      if (operation =="mult")
33      { this.resultat= this.Multiplication(p1, p2);
34       this.isMult=true ;
35     }
36     if (operation =="div")
37     { this.resultat= this.Division(p1, p2);
38      this.isSous=true ;
```



Angular: les formulaires

Exemple : On veut créer une FORM qui contient le champ nom dans le composant Bonjour : « /bonjour », pour cela on ajoute au fichier « Bonjour.component.html » le code suivant :

bonjour.component.html U

src > app > bonjour > bonjour.component.html > div > label

```
5   <div>
6   <label for="name">Nom: </label>
7   <input id="name" type="text" [formControl]="name">
8   <p>Valeur: {{ name.value }}</p>
9   <button type="button" (click)="updateName()">Modifier Nom</button>
10  </div>
```

On import : `import { FormControl } from '@angular/forms';`

Et on ajoute au fichier Bonjour.component.ts » le code suivant :

```
51  name = new FormControl('');
52    updateName() {
53    this.name.setValue('samir');
54  }
```