

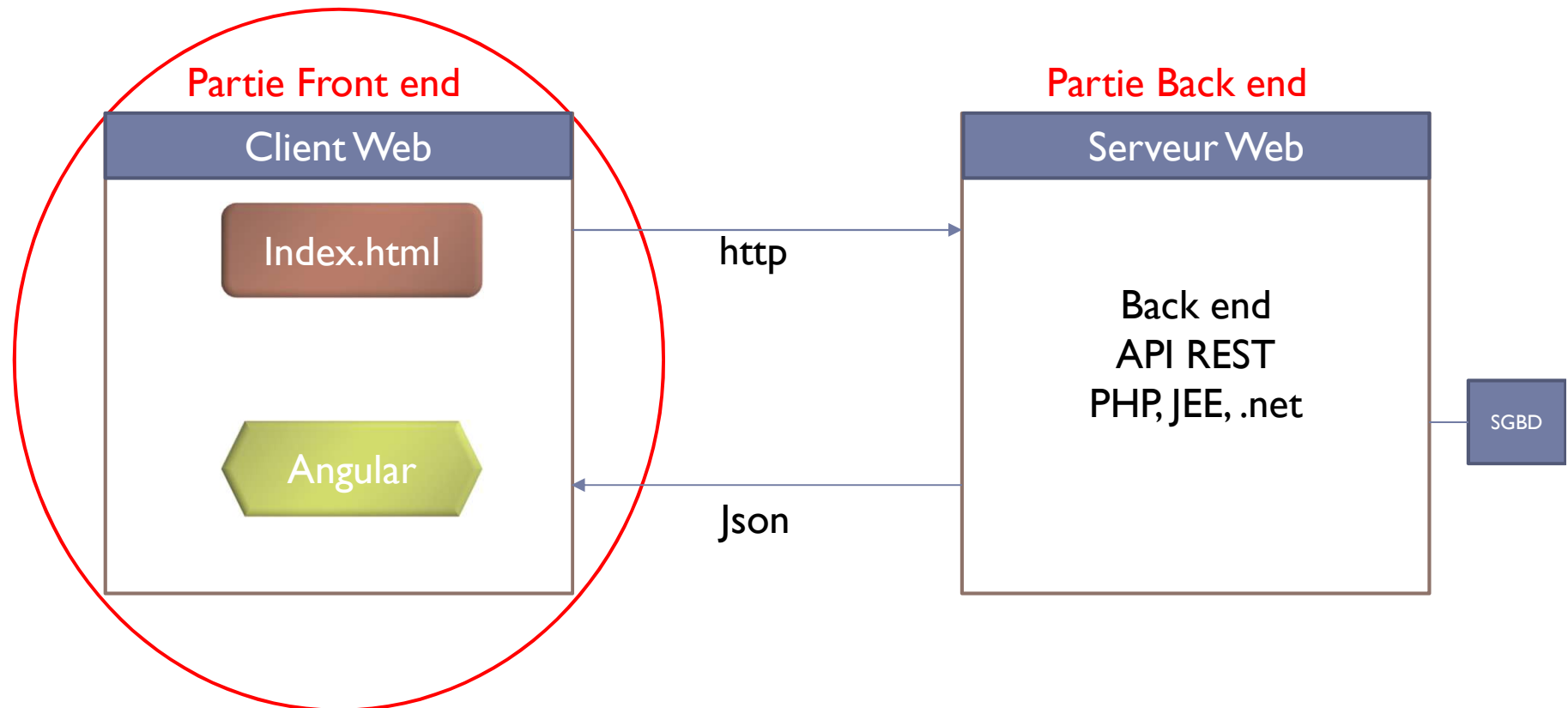


Développement Web avancée

Le Framework Angular

Imene Lahyani

Introduction



Angular

- ▶ Angular permet de créer la partie Front-end des applications Web de type **SPA (Single Page Application reactive)**
- ▶ Une SPA est une application qui contient une seule page HTML (index.html) récupérée du serveur
- ▶ Pour naviguer entre les différentes parties de l'application, Javascript est utilisé pour envoyer des **requetes http** vers le serveur pour **recuperer du contenu dynamique au format JSON** (generalement)
- ▶ Ce contenu JSON est affiché par la suite coté client au format HTML sur la meme page.

Différentes versions de Angular

- ▶ Angular 1 (Angular JS) (juin 2012)
 - ▶ Première version de Angular
 - ▶ Elle est basé sur une architecture MVC coté client. Les applications Angular 1 sont écrites en **Javascript**.
- ▶ Angular 2 (Angular) (septembre 2016)
 - ▶ Est une réécriture de Angular 1 qui est plus performante, mieux structuré et représente le futur de Angular
 - ▶ Les applications de Angular 2 sont écrites en **Typescript** qui est compilé et traduit en javascript avant d'être exécutés par les browsers Web,
 - ▶ Angular 2 est basé sur une programmation à base de composants Web (web components)
- ▶ Angular 4 (Mars 2017)
 - ▶ Est une simple mise à jour de Angular 2

Différentes versions de Angular

- ▶ **Angular 5 : Novembre 2017**
 - ▶ est une simple mise à jour de Angular 4
- ▶ **Angular 6 : Mai 2018**
 - ▶ est une simple mise à jour de Angular 5
- ▶ **Angular 7 : Septembre 2018**
 - ▶ Est une simple mise à jour de Angular 6
- ▶ **Angular 8**
- ▶ **Angular 9**
- ▶ **Angular 10**
- ▶ **Angular 11**
- ▶ **Angular 12**

Angular avec Typescript

- ▶ Pour développer une application Angular, il est recommandé d'utiliser **Typescript** qui sera compilé et traduit en **Javascript**.
- ▶ Typescript est un langage de script structuré et orienté objet qui permet de simplifier le développement d'applications Javascript



Démarrer avec Angular

<http://angular.io/guide/quickstart>

The screenshot shows the Angular.io website's quickstart guide. The browser's address bar displays the URL <https://angular.io/guide/quickstart>. The website's header is blue with the Angular logo and navigation links: FEATURES, DOCS, RESOURCES, EVENTS, and BLOG. A left sidebar contains a menu with categories like GETTING STARTED, TUTORIAL, FUNDAMENTALS, TECHNIQUES, SETUP & DEPLOYMENT, RELEASE INFORMATION, QUICK REFERENCE, CLI COMMANDS, and API. The 'stable (v7.0.0)' version is selected. The main content area is titled 'Getting started' and includes an introduction, prerequisites, and a section for Node.js. The introduction states that Angular helps build modern applications and that the guide shows how to build and run a simple Angular app using the Angular CLI tool. The prerequisites section mentions that Node.js and an npm package manager are required. The Node.js section specifies that Angular requires Node.js version 8.x or 10.x and provides instructions on how to check the version and where to get Node.js.

Getting started

Welcome to Angular! Angular helps you build modern applications for the web, mobile, or desktop.

This guide shows you how to build and run a simple Angular app. You'll use the [Angular CLI tool](#) to accelerate development, while adhering to the [Style Guide](#) recommendations that benefit every Angular project.

This guide takes less than 30 minutes to complete. At the end of this guide—as part of final code review—there is a link to download a copy of the final application code. (If you don't execute the commands in this guide, you can still download the final application code.)

Prerequisites

Before you begin, make sure your development environment includes Node.js® and an npm package manager.

Node.js

Angular requires Node.js version 8.x or 10.x.

- To check your version, run `node -v` in a terminal/console window.
- To get Node.js, go to nodejs.org.

Installation des outils nécessaires

- ▶ Pour faciliter le développement d'une application angular, les outils suivants doivent être installés:
 - **Node JS** : installe l'outil npm (Node package manager) qui permet de télécharger et installer des bibliothèques JavaScript
<https://nodejs.org/en/download/>
 - Installer Angular CLI (Command Line Interface) qui permet de générer , compiler, tester, deployer des projets angular
<http://cli.angular.io> en utilisant :

npm install -g @angular/cli

Création d'un nouveau projet Angular

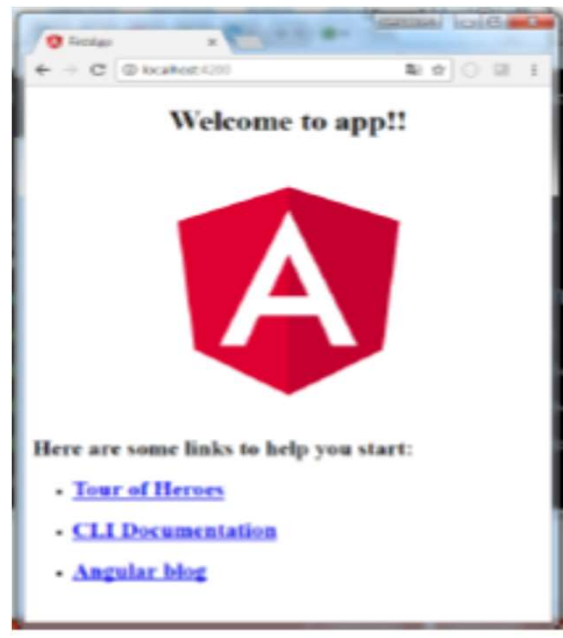
- ▶ Afin de générer la structure d'un projet Angular, on utilise Angular CLI via la commande **ng** suivi de l'option new et du nom du dossier
 - ▶ **ng** new FirstApp
 - ▶ ng: genere les différents fichiers requis par une application basique Angular et installe toutes les dépendances requises par ce projet.

Exécuter un projet Angular

- ▶ Afin **d'exécuter** le projet Angular, on exécute la commande
- ▶ **ng serve**
- ▶ Cette commande :
 - ▶ compile le code source du projet pour transpiler le code TypeScript en Javascript
 - ▶ En meme temps , démarre un serveur web local basé sur node Js pour déployer l'application localement (port par défaut du serveur : 4200)

Tester un projet Angular

- ▶ Afin de tester un projet Angular, il suffit d'ouvrir votre navigateur et taper : <http://localhost:4200>

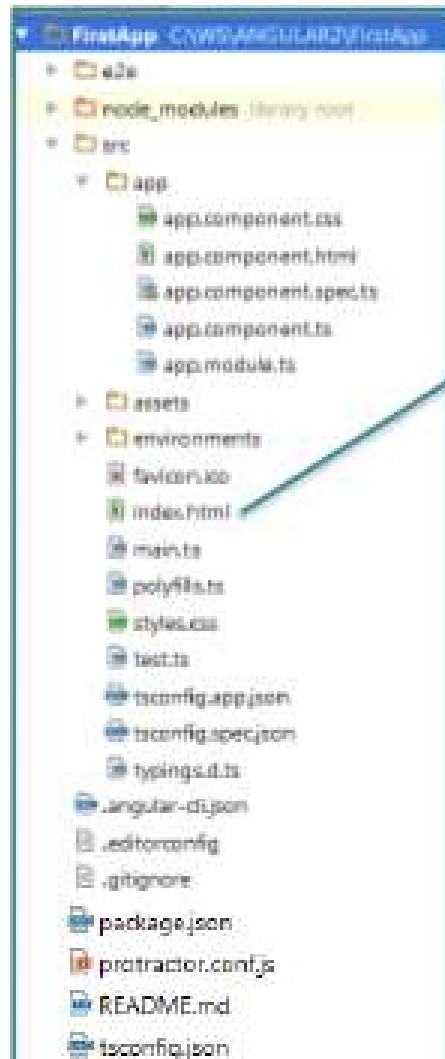


Editeur de projets

- ▶ Plusieurs editeurs professionnels peuvent etre utilisés pour editer le code:
 - ▶ Web Storm, PHP Storm,
 - ▶ Visual Studio Code
 - ▶ Eclipse avec plugin Angular
- ▶ D'autres editeurs classiques peuvent etre utilisés :
 - ▶ Atom
 - ▶ Sublime Text

Installer votre éditeur de projets

Structure du projet Angular



index.html

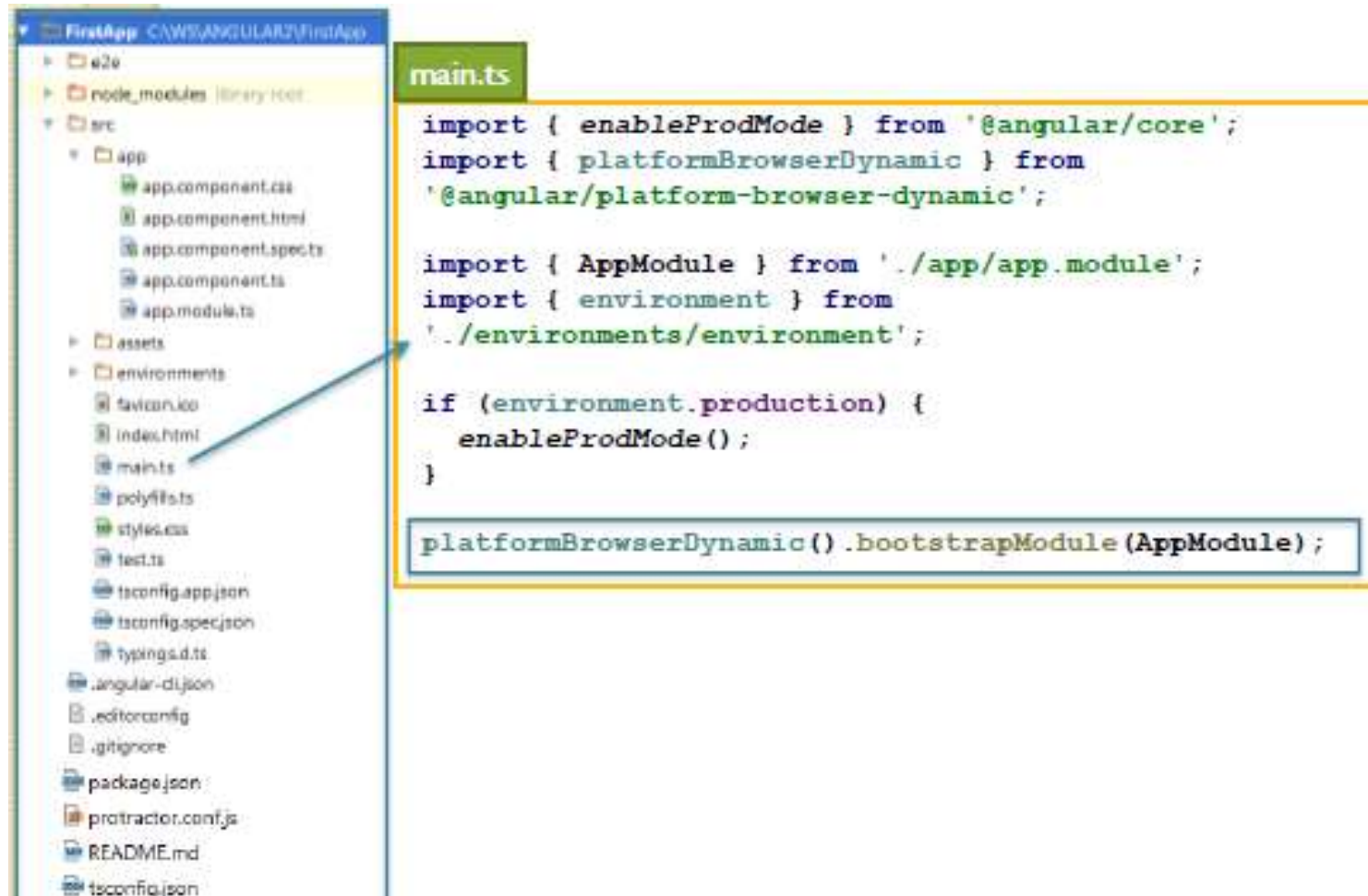
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>FirstApp</title>
  <base href="/">

  <meta name="viewport" content="width=device-
width, initial-scale=1">
  <link rel="icon" type="image/x-icon"
href="favicon.ico">
</head>
<body>

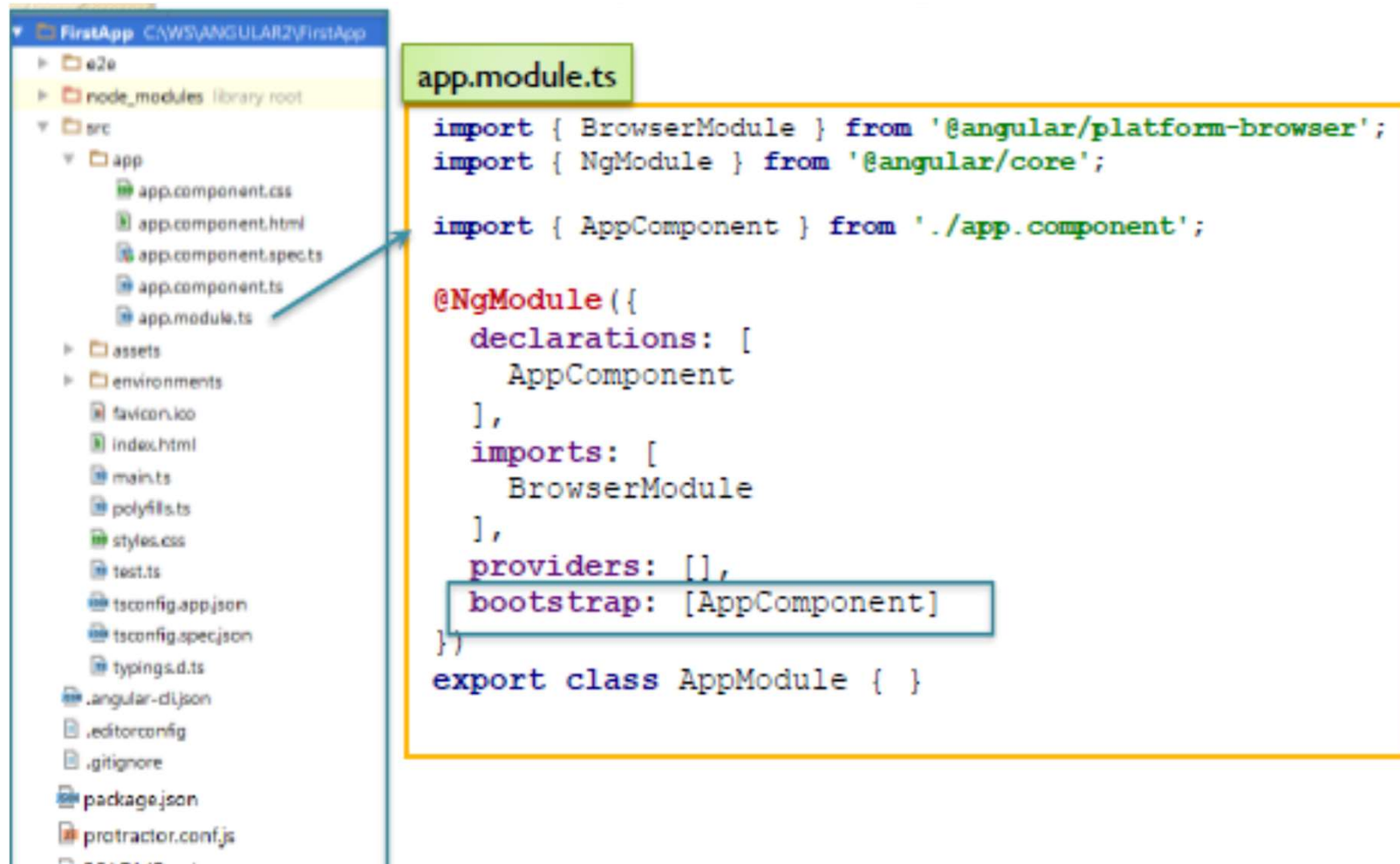
  <app-root> </app-root>

</body>
</html>
```

Structure du projet Angular



Structure du projet Angular



The image shows a file explorer on the left and a code editor on the right. The file explorer displays the project structure for 'FirstApp' at 'C:\WS\ANGULAR2\FirstApp'. The 'src' directory is expanded, showing 'app' and 'assets' subdirectories. The 'app' directory contains 'app.component.css', 'app.component.html', 'app.component.spec.ts', 'app.component.ts', and 'app.module.ts'. An arrow points from 'app.module.ts' in the file explorer to the code editor. The code editor shows the content of 'app.module.ts', which is highlighted with an orange border. The code defines the 'AppModule' using the '@NgModule' decorator, importing 'BrowserModule' and 'AppComponent', and bootstrapping 'AppComponent'. The 'bootstrap' property is highlighted with a blue box. The code is as follows:

```
app.module.ts

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

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

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

Structure du projet Angular

The diagram illustrates the structure of an Angular project. On the left, a file explorer shows the project hierarchy. The main part of the diagram consists of three yellow boxes, each containing code for a specific file. Arrows point from the file explorer to these boxes. A small browser window on the right shows the rendered output.

File Explorer:

- FirstApp C:\WS\ANGULAR2\FirstApp
 - e2e
 - node_modules library root
 - 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
 - tsconfig.app.json
 - tsconfig.spec.json
 - typings.d.ts
 - .angular-cli.json
 - .editorconfig
 - .gitignore
 - package.json
 - protractor.conf.js
 - README.md
 - tsconfig.json

app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
}
```

app.component.html

```
<div style="text-align:center">
  <h1>
    Welcome to {{title}}!!
  </h1>
</div>
```

app.component.css

Browser Preview:

Welcome to app!!

Architecture de Angular

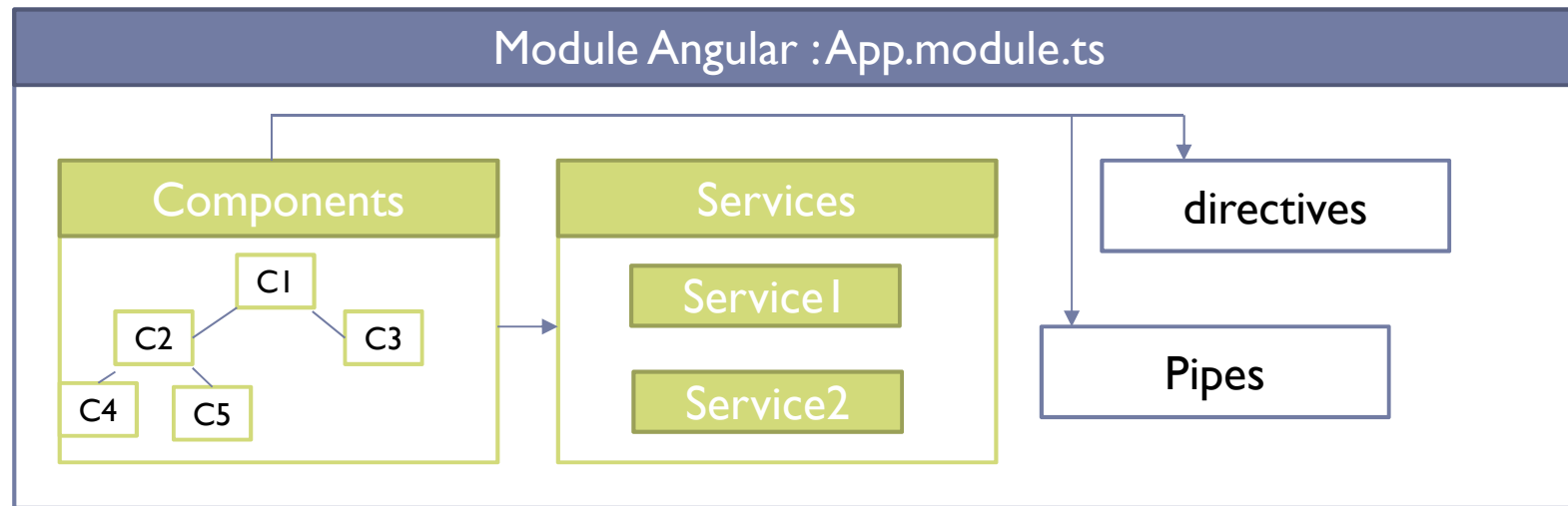
- ▶ Une application Angular se compose de :

- ▶ Un ou plusieurs modules dont un est principal

Chaque module peut inclure:

- Des composants Web: la partie visible de l'application IHM
 - Des services pour la logique applicative
 - Les directives : Un composant peut utiliser des directives
 - Les pipes : utilisés pour formater l'affichage des données dans les composants.

Architecture de Angular



Modules

- ▶ Les applications Angular sont modulaires
- ▶ Angular possède son propre système de modularité appelés modules angulaires ou NgModules
- ▶ Chaque application possède au moins une classe de module racine appelé classiquement **AppModule**
- ▶ Un module angulaire est une classe avec un décorateur **@NgModule**
- ▶ Les décorateurs sont des classes qui modifient les classes Javascript

Modules

Src/app/app.module.ts

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

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

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

@NgModule

- ▶ @NgModule est un décorateur qui prend en parametre un objet Javascript qui contient les metadonnées dont les proprietes decrivent le module.
- ▶ Les propriétés les plus importantes sont :
 - ▶ **imports**: importer d'autres modules
 - ▶ **exportes**: importer des classes utilisable dans d'autres modules
 - ▶ **Declarations**: declarer les composants qui vont etre utilisés dans ce module
 - ▶ **Providers** : declarer les services
 - ▶ **Bootstrap** : declarer le composant racine du module
(seul le composant racine doit définir cette propriété)

Démarrage de l'application

- ▶ La module racine est démarré dans main.ts
- ▶ Par défaut le module racine s'appelle AppModule

main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

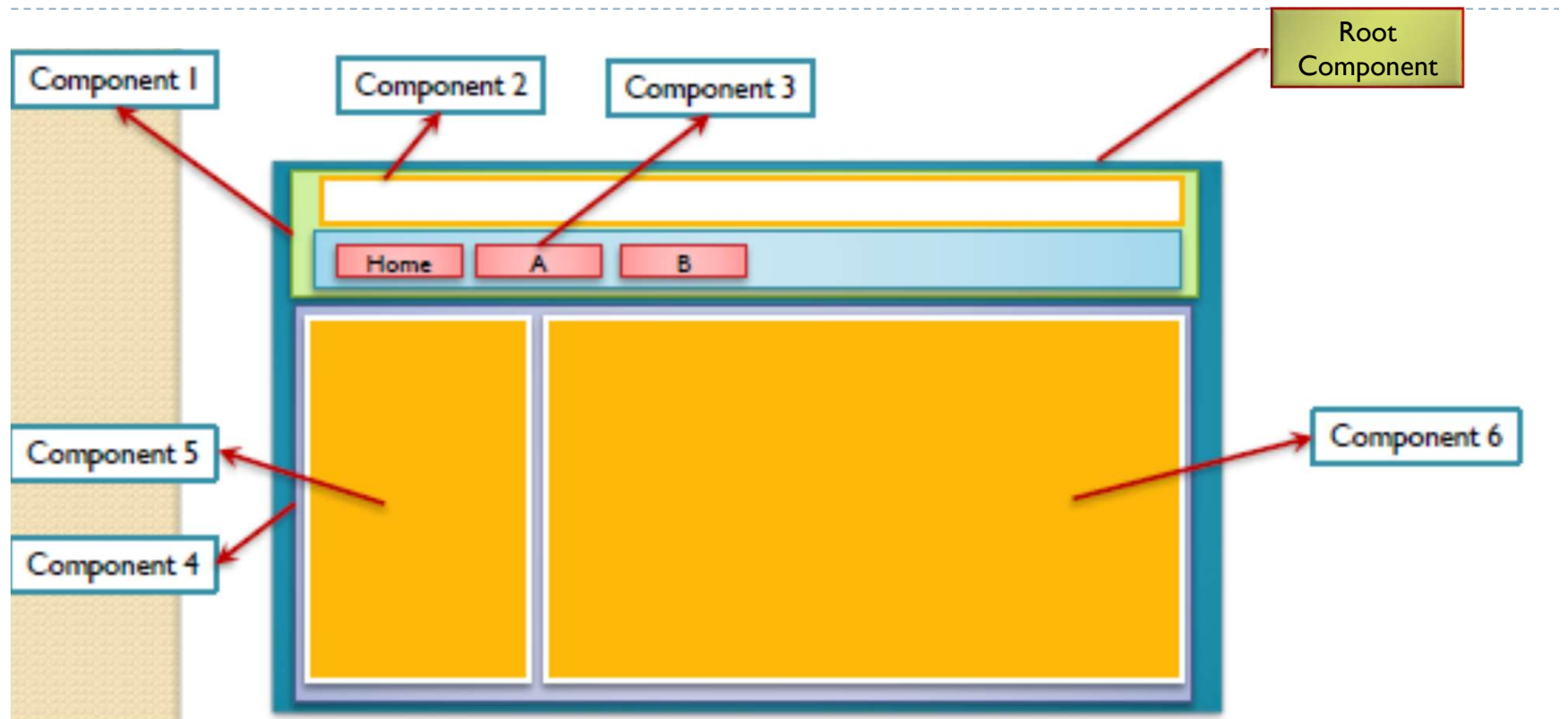
if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));
```

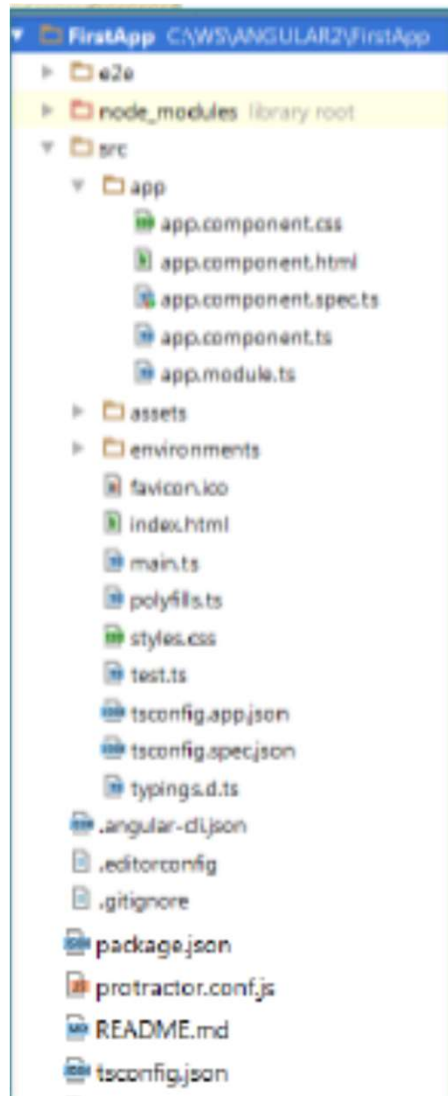
Les composants

- ▶ Les composants sont des elements importants dans Angular
- ▶ L'application est formé par un ensemble de composants
- ▶ Chaque composant peut imbriquer d'autres composants définissant ainsi une structure hierarchique
- ▶ Le composant racine s'appelle Root component

Les composants



Les composants



- Chaque composant se compose principalement des éléments suivants:
 - HTML Template : représentant sa vue
 - Une classe représentant sa logique métier
 - Une feuille de style CSS
- Les composants sont facile à mettre à jour et à échanger entre les différentes parties des applications.

app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
}
```

app.component.html

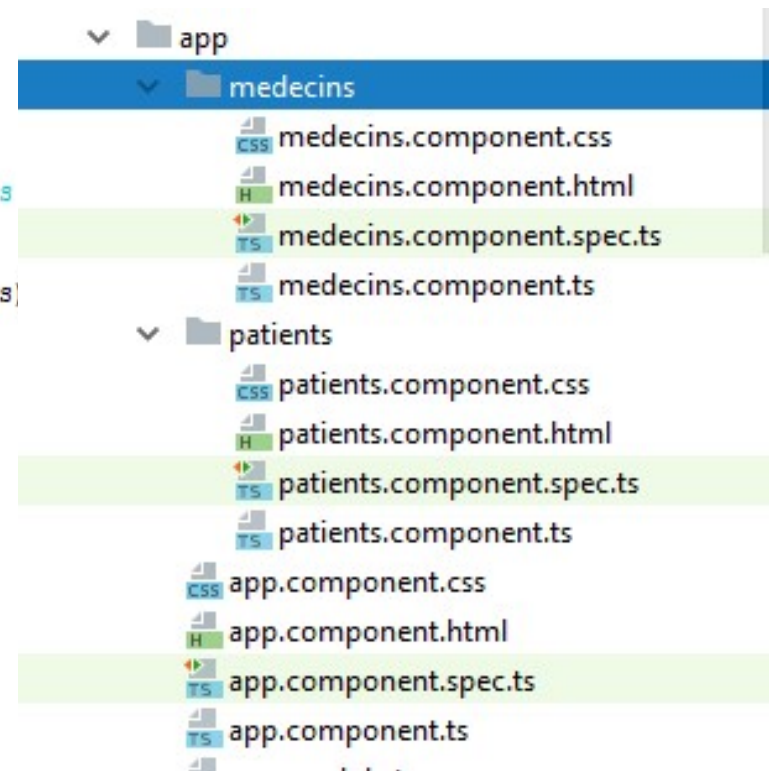
```
<div style="text-align:center">
  <h1>
    Welcome to {{{title}}}!!
  </h1>
</div>
```

app.component.css

Création de nouveaux composants

- ▶ Pour créer facilement des composants angular, on peut utiliser la commande **ng** comme suit :
- ▶ **ng generate component** nom_composant

```
C:\Angular3\FirstA>ng g c
? What name would you like to use for the component? patients
CREATE src/app/patients/patients.component.html (27 bytes)
CREATE src/app/patients/patients.component.spec.ts (642 bytes)
CREATE src/app/patients/patients.component.ts (277 bytes)
CREATE src/app/patients/patients.component.css (0 bytes)
UPDATE src/app/app.module.ts (495 bytes)
```



Création de nouveaux composants

medecins.component.ts

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

@Component({
  selector: 'app-medecins',
  templateUrl: './medecins.component.html',
  styleUrls: ['./medecins.component.css']
})
export class MedecinsComponent implements OnInit {

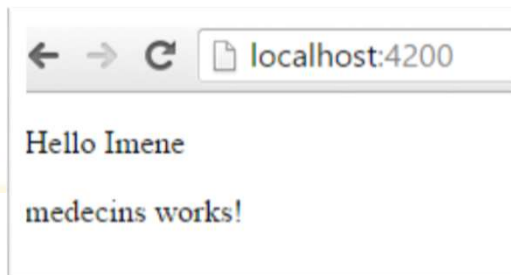
  constructor() { }

  ngOnInit() {
  }

}
```

medecins.component.html

```
<p>
  medecins works!
</p>
```



ng g c --module=app.module

Création de nouveaux composants

- ▶ Le nouveau composant est une classe qui utilise le décorateur `@component`
- ▶ Le décorateur a les propriétés suivantes :
 - **selector** : indique la déclaration qui permet d'insérer le composant dans le document HTML. Cette déclaration peut être :
 - Le nom de la balise associé à ce composant
 - selector : `app-medecins`
 - Dans ce cas le composant sera inséré par : `<app-medecins></app-medecins>`
 - Le nom de l'attribut associé à ce composant :
 - selector : `[app-medecins]`
 - Dans ce cas le composant sera inséré par : `<div app-medecins</div>`
 - Le nom de la classe associé à ce composant :
 - selector : `app-medecins`
 - Dans ce cas le composant sera inséré par : `<div class="app-medecins"></div>`
 - **template ou templateUrl** :
 - **template** : permet de définir dans à l'intérieur du décorateur le code HTML représentant la vue du composant
 - **templateUrl** : permet d'associer un fichier externe HTML contenant la structure de la vue du composant
 - **styleUrls** : spécifier les feuilles de styles CSS associées à ce composant

Déclaration d'un nouveau composant

- Pour utiliser un composant, ce dernier doit être déclaré dans le module principal `app.module.ts` (automatique avec ng g c)

App.module.ts

```
import { NgModule } from '@angular/core';

import { AppComponent } from '../app.component';
import { MedecinsComponent } from '../medecins/medecins.component';
import { PatientsComponent } from '../patients/patients.component';

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

Hello Imene

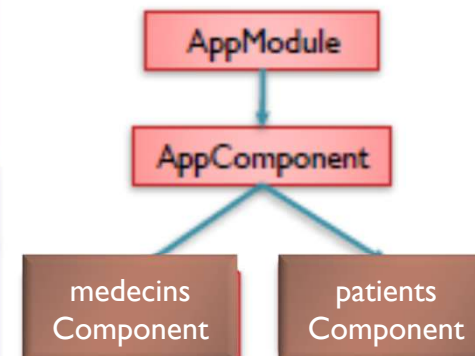
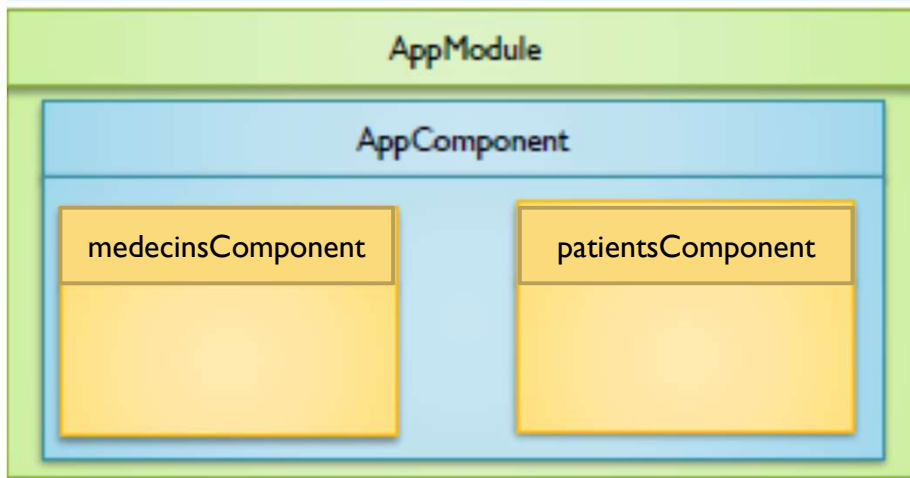
medecins works!

patients works!

Utilisation d'un nouveau composant

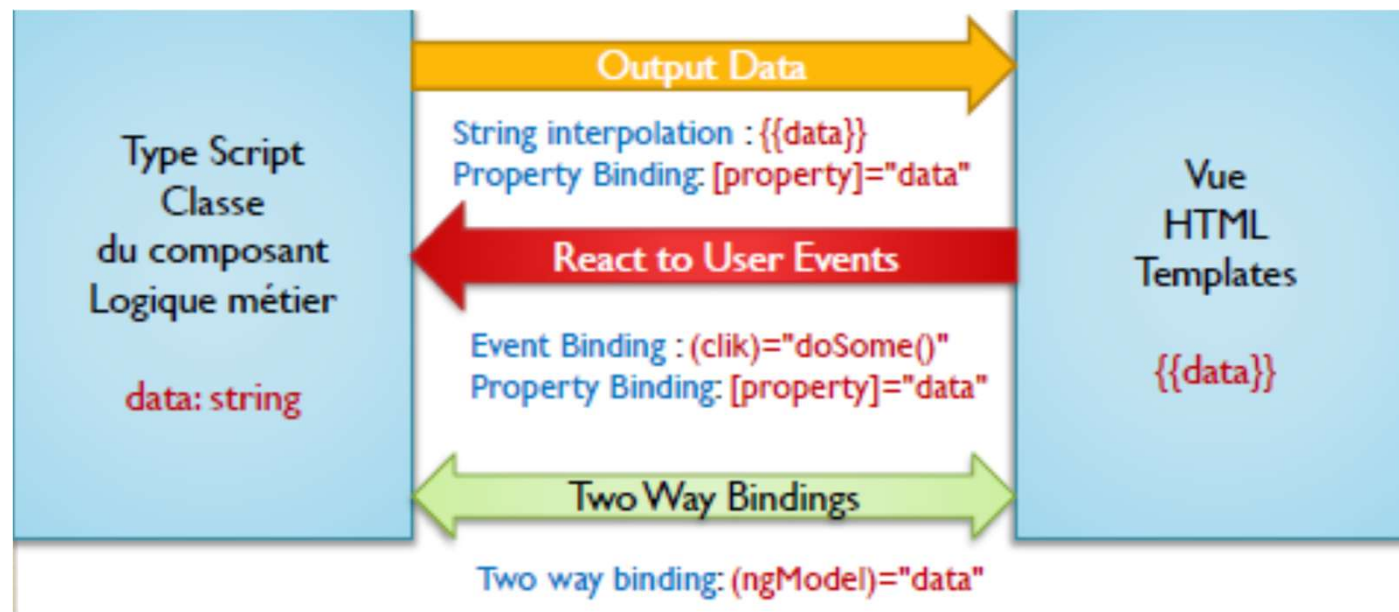
- Un composant peut être inséré dans n'importe quelle partie HTML de l'application en utilisant son selector associé

```
<p> Hello Imene </p>  
<div><app-medecins> </app-medecins></div>  
<div><app-patients></app-patients></div>
```



Data Binding

- ▶ Pour insérer dynamiquement les données de l'application dans les vues du composant, Angular définit des techniques pour assurer la liaison des données.
- ▶ Data Binding = communication



Exercice

- Insérer des informations concernant le nom et l'email d'un médecin, Affichez ensuite ces informations dans la page html . [utilisez le composant medecins]

medecin.component.ts

```
export class MedecinsComponent implements OnInit {  
  Agenda = [  
    {date : new Date(), message : 'Operation_Bloc A' },  
    {date : new Date(), message : 'Operation_Bloc B' },  
  ];  
  info_medecin = {nom : 'Docteur',  
    prenom : 'imen',  
    telephone : '0124587',  
  };  
};
```

medecin.component.html

```
L1>>  
medecins works!  
</p>  
<div>  
  <ul>  
    <li> Nom: {{info_medecin.nom}} </li>  
    <li> Nom: {{info_medecin.prenom}} </li>  
    <li> Nom: {{info_medecin.telephone}} </li>  
  </ul>  
</div>  
  
  <li *ngFor="let element of Agenda">  
    {{ element.date }} : {{ element.message }}  
  </li>
```

String interpolation

Diapositive 32

L1

LENOVO; 13/09/2021

Exercise

Hello Imene

medecins works!

- Nom : Docteur
- prenom : imen
- telephone : 0124587
- Tue Nov 06 2018 10:46:46 GMT+0100 (Afr. centrale Ouest) : Operation_Bloc A
- Tue Nov 06 2018 10:46:46 GMT+0100 (Afr. centrale Ouest) : Operation_Bloc B

patients works!

Exercice

- Améliorer l'affichage en utilisant les **pipes** (pipe date)

solution :

```
{{c.date|date: "dd/MM/yyyy:HH:mm:ss "}}
```

```
Hello Imene
```

```
medecins works!
```

- Nom : Docteur
- prenom : imen
- telephone : 0124587
- 06/11/2018:10:45:53 : Operation_Bloc A
- 06/11/2018:10:45:53 : Operation_Bloc B

```
patients works!
```