

Institut Supérieur des Etudes Technologiques de Mahdia

# FRAMEWORK COTÉ CLIENT

Mme KHAYATI Alya

Ingénieur en génie logiciel

Alya.khayati@hotmail.com



# OBJECTIFS

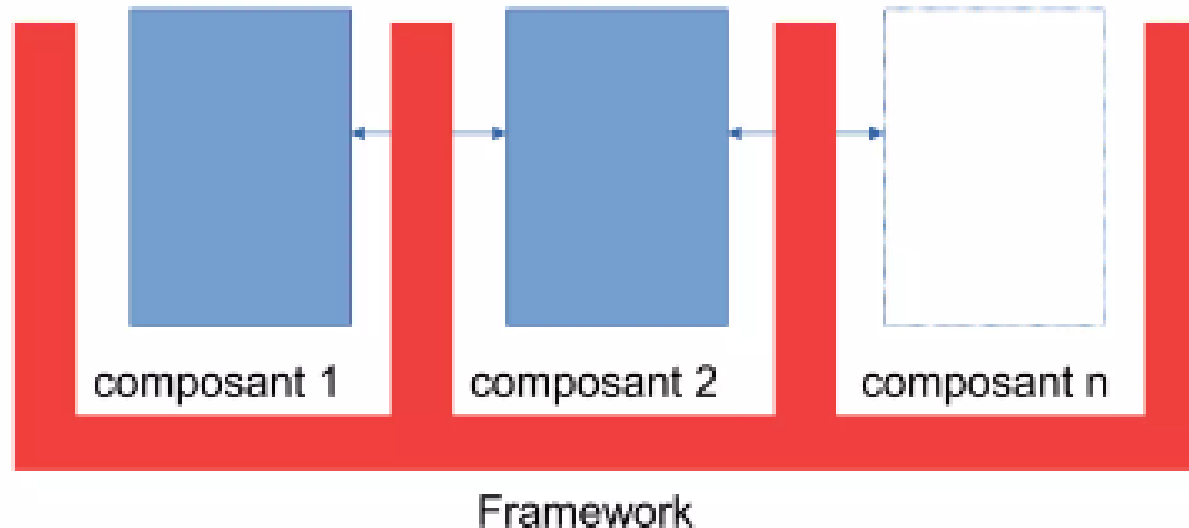
- Comprendre les fondamentaux et les concepts avancés d'Angular.
- créer une application Angular
- Maîtriser le routage, les services, et les modules en Angular.
- Comprendre comment travailler avec des observables et des événements.

# PRÉ-REQUIS

- Connaissance de base en programmation.
- de bonnes connaissances en HTML, en CSS et en JavaScript.

# NOTION DE FRAMEWORK

- Un Framework désigne un ensemble cohérent de composants logiciels structurels.
- Un framework sert à créer les **fondations** ainsi que **les grandes lignes** de tout ou d'une partie d'un logiciel(architecture)



# QU'EST-CE QU'ANGULAR ?



**Angular** est un **framework de développement** pour créer des **applications web dynamiques**. Il a été développé par **Google** et il est écrit en **TypeScript**.

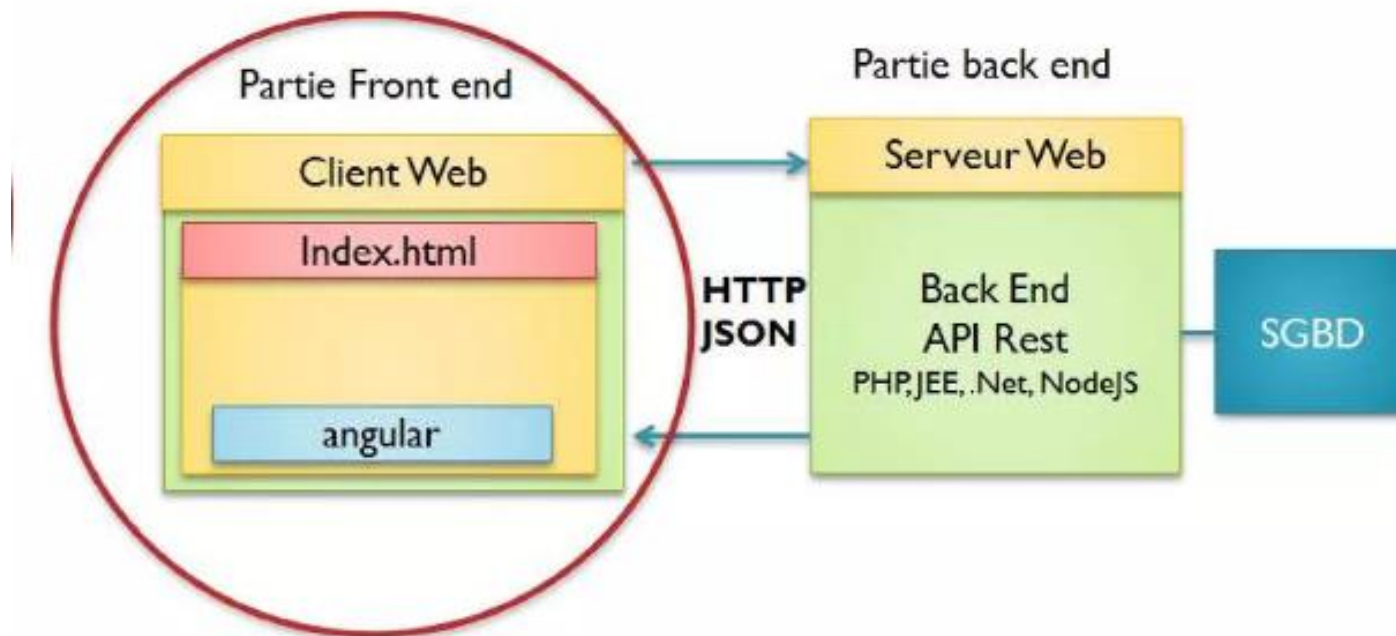
Version Actuelle	Angular 16
Date Lancement	Mai 2023
NodeJs Compatible	^16.14.0    ^18.10.0
TypeScript	>=4.9.3 <5.1.0



## Angular

Angular est un Framework front end de google qui aide à créer des applications à page unique (SPA) interactives et dynamiques avec ses fonctionnalités convaincantes, notamment la création de modèles, la liaison bidirectionnelle, la modularisation, la gestion de l'API RESTful, l'injection de dépendances et la gestion AJAX.

- Il permet notamment de créer ce qu'on appelle des **Single Page Applications** (ou **SPA**) : des applications entières qui tournent dans une seule page HTML .
- La seule page contient différents composants web.



# POURQUOI ANGULAR ?

**Angular** offre plusieurs avantages qui en font un choix populaire parmi les développeurs.

Angular est un framework **complet** – on peut créer des applications web complètes sans avoir besoin de librairies tierces supplémentaires. C'est notamment ce qui différencie un *framework* d'une *library*.

Il permet une **modularité**, ce qui signifie que les différentes parties de votre application peuvent être **réutilisées** et **testées séparément**. De plus, sa **communauté active** et les **régulières mises à jour** contribuent à le garder pertinent et à jour.



# LES TROIS PILIERS D'ANGULAR

- **Composants:** Ce sont les éléments de base d'une application Angular. Ils contiennent le code **HTML**, **CSS**, et **TypeScript** pour rendre les pages.
- **Modules:** Angular utilise un système modulaire pour organiser votre code. Un module peut encapsuler des **composants**, **directives** et **services** qui sont liés fonctionnellement.
- **Services et Injection de Dépendance:** Les services sont des objets qui sont injectés dans des composants et autres services. Ils permettent de partager des **fonctions** et des **données** à travers l'application.

# QUAND UTILISER ANGULAR ?

**Angular** est particulièrement utile lorsque vous développez des **applications web complexes** avec des **fonctionnalités riches**. Il est souvent utilisé pour créer des **applications d'entreprise**, des **dashboards**, et même des **applications mobiles** grâce à des solutions comme **Ionic**.

Angular utilise TypeScript, qui fournit un excellent support pour la vérification de type et d'autres outils externes.

Angular est pris en charge par Google, ce qui signifie qu'il est soutenu par une organisation fiable. Ils travaillent avec une documentation détaillée et une grande communauté, ce qui en fait un cadre d'apprentissage fiable.

Angular-language-service permet la saisie semi-automatique à l'intérieur des fichiers de modèle HTML externes des composants, vous permettant d'accélérer votre développement.

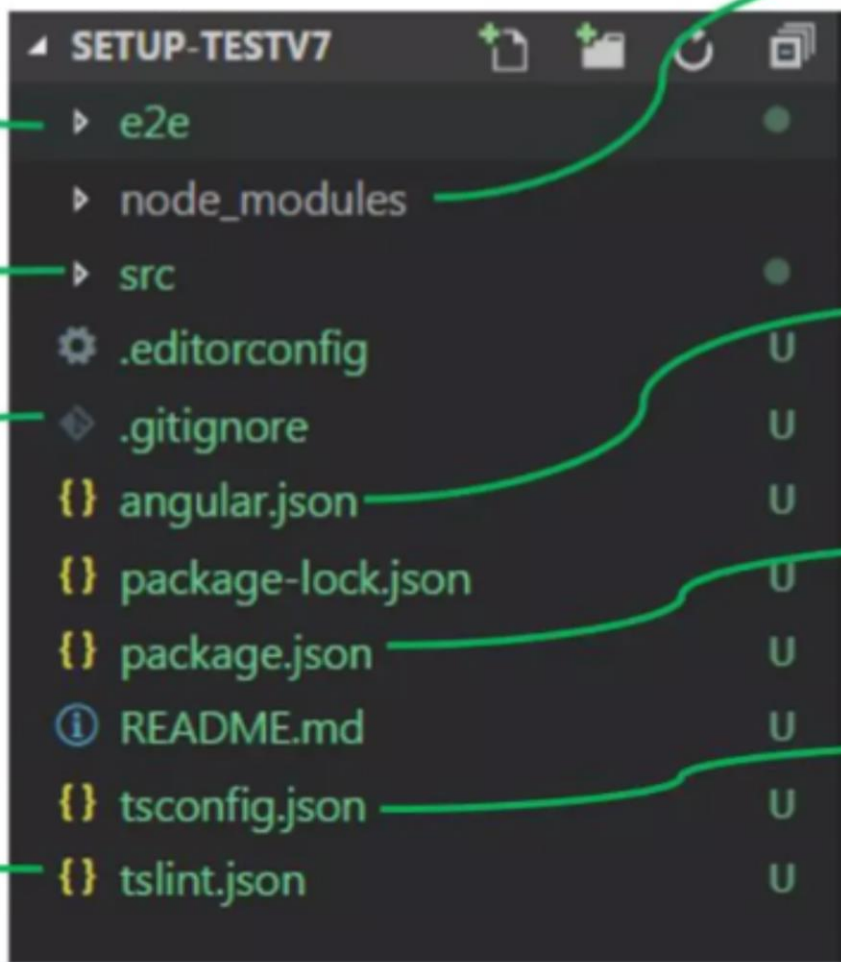
# STRUCTURE PROJET ANGULAR

testing scenarios  
{create test scenarios}

Source code  
{Action Area}

No Git  
{files mentioned not allowed}

Code analysis tool  
{Typescript code checked if source  
code complies with coding rules}



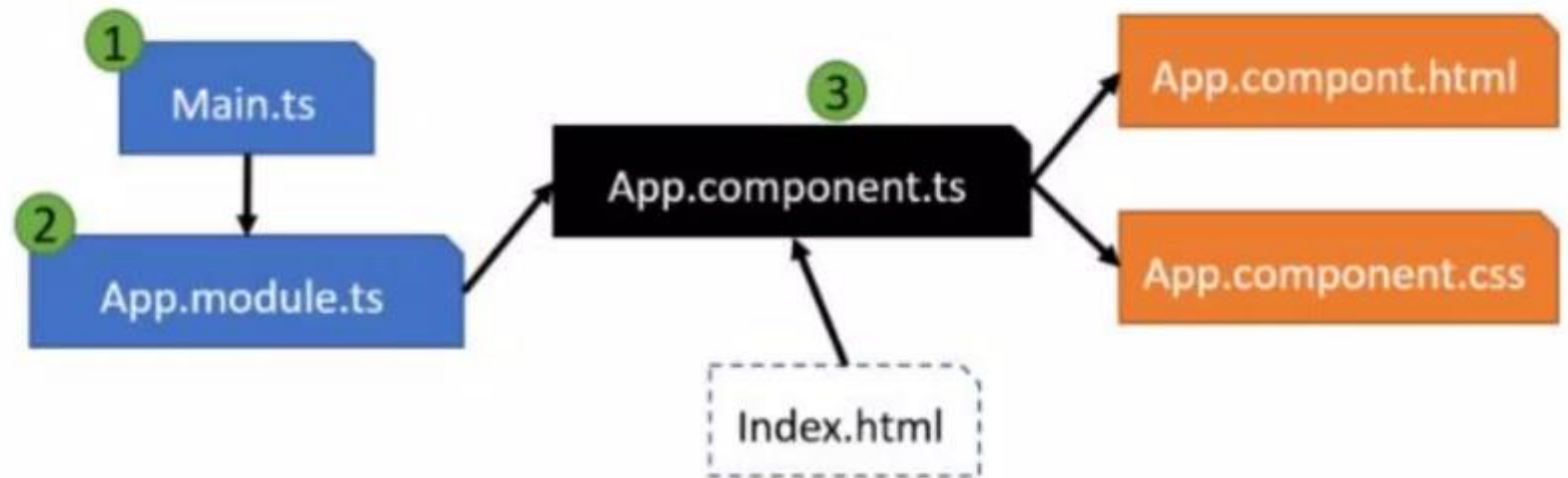
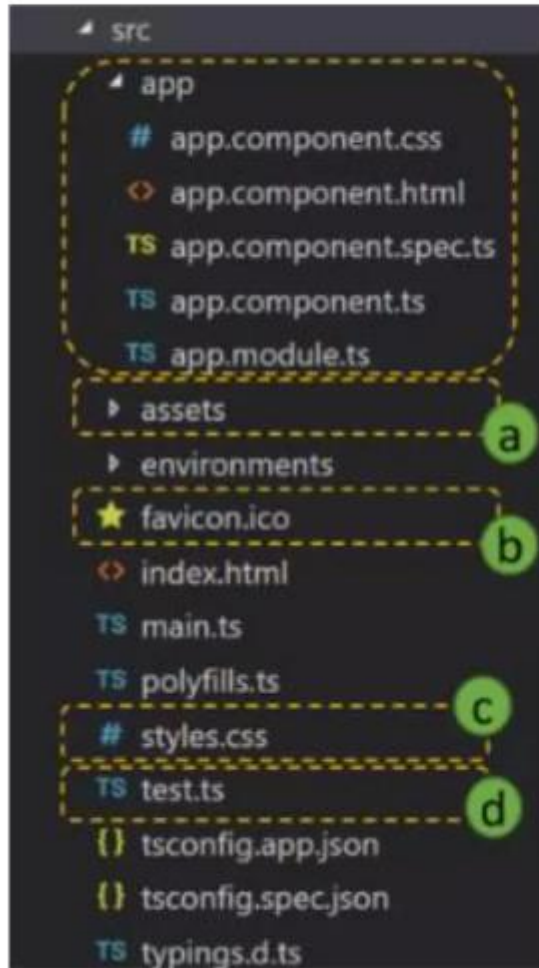
Libraries Deployed here  
{Package.json defined them  
and npm install had them installed}

Application Configuration  
{default setting for application you can  
override them using the ng-new command}

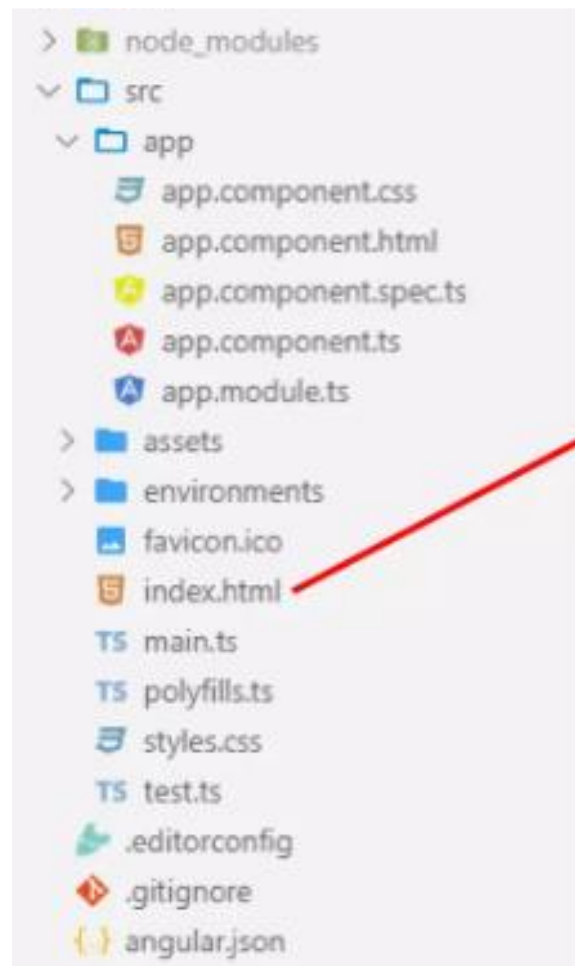
Dependencies Prod/Dev  
{Packages to run angular}

Compiles  
{Compiles the typescript code to JavaScript  
so browser can understand}

# STRUCTURE PROJET ANGULAR DETAILLÉE





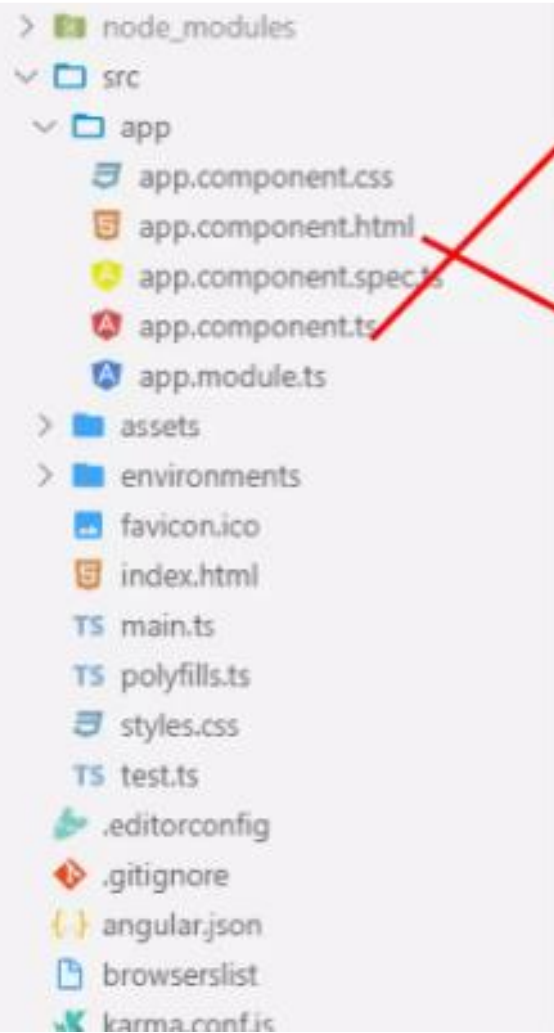


index.html X

src > index.html > ...

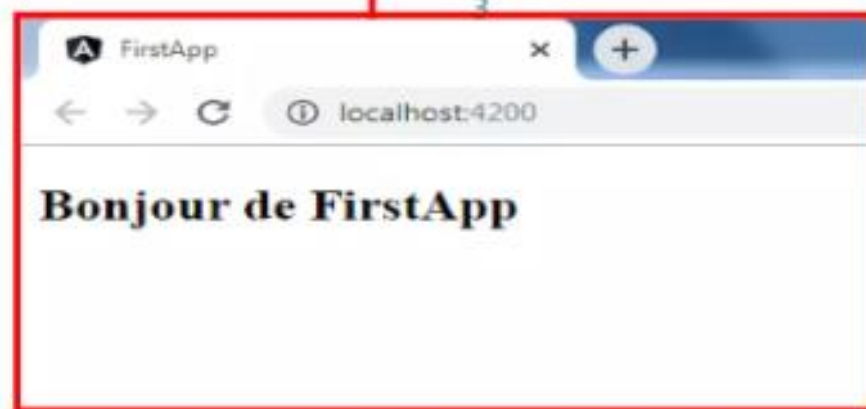
```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>FirstApp</title>
6   <base href="/">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="icon" type="image/x-icon" href="favicon.ico">
9 </head>
10 <body>
11   <app-root></app-root>
12 </body>
13 </html>
```

Sélecteur du root component



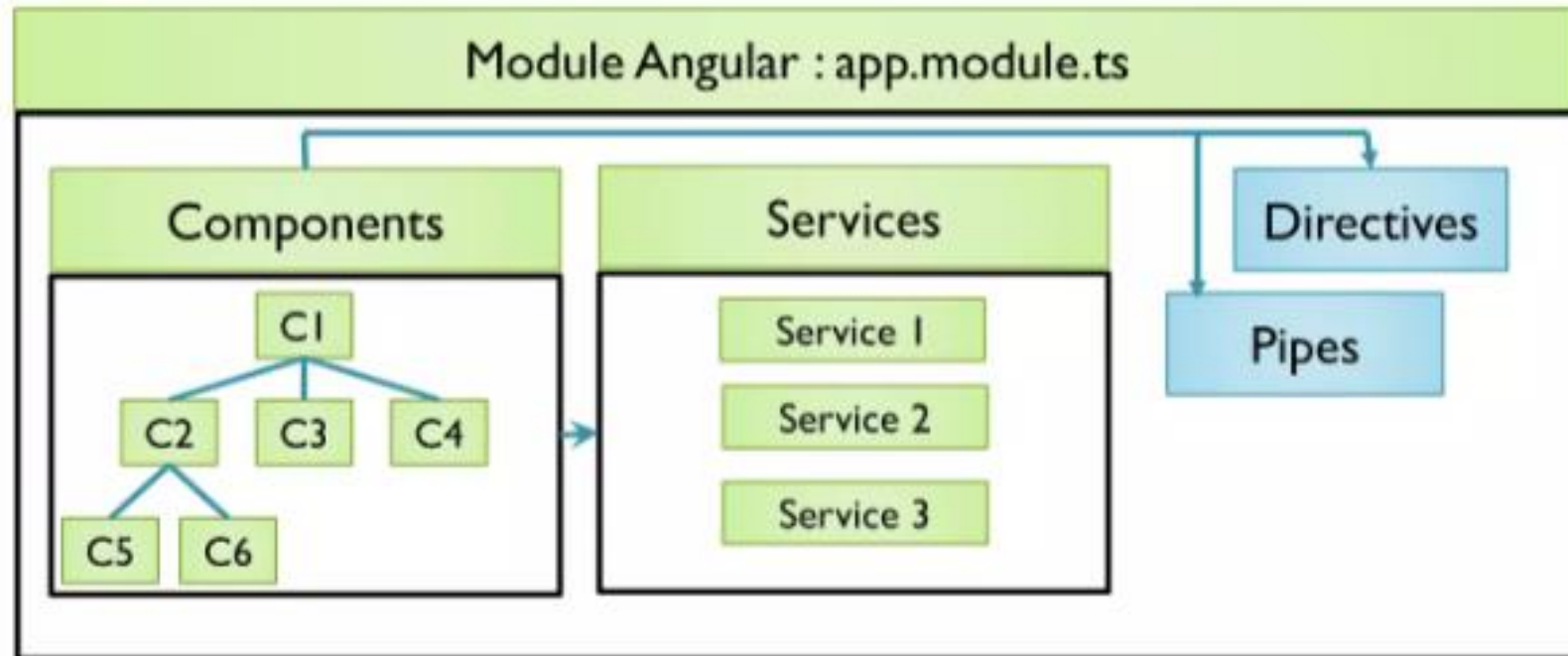
```
app.component.ts x
src > app > app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'FirstApp';
10 }
```

```
app.component.html x
src > app > app.component.html > ...
1
2  <h2>Bonjour de {{title}}/h2>
3
```



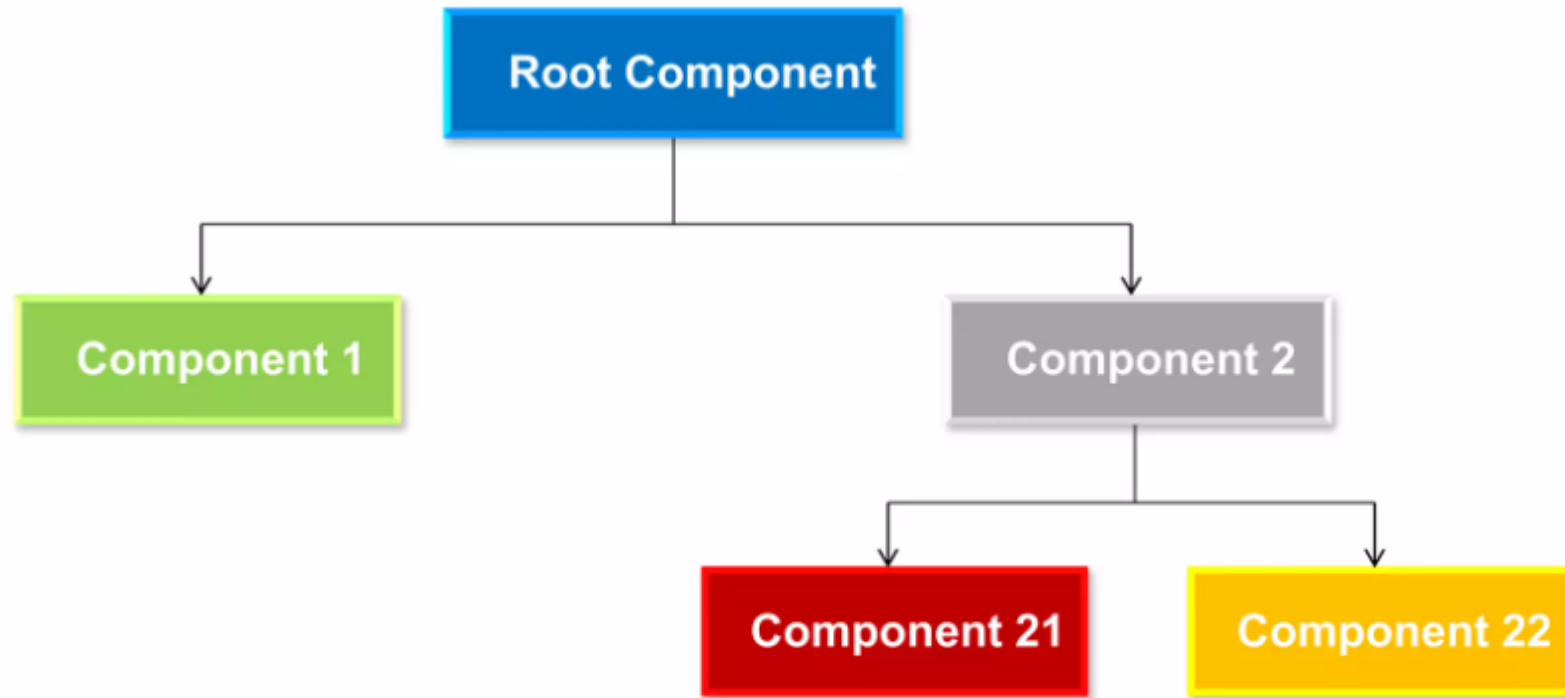
# ARCHITECTURE DU FRAMEWORK ANGULAR

- Une application Angular se compose de :
  - Un à plusieurs modules dont un est principal.
  - Chaque module peut inclure :
    - Des composants web : La partie visible de la 'application Web (IHM)
    - Des services pour la logique applicative. Les composants peuvent utiliser les services via le principe de l'injection des dépendances.
    - Les directives : un composant peut utiliser des directives
    - Les pipes : utilisés pour formater l'affichage des données dans els composants.

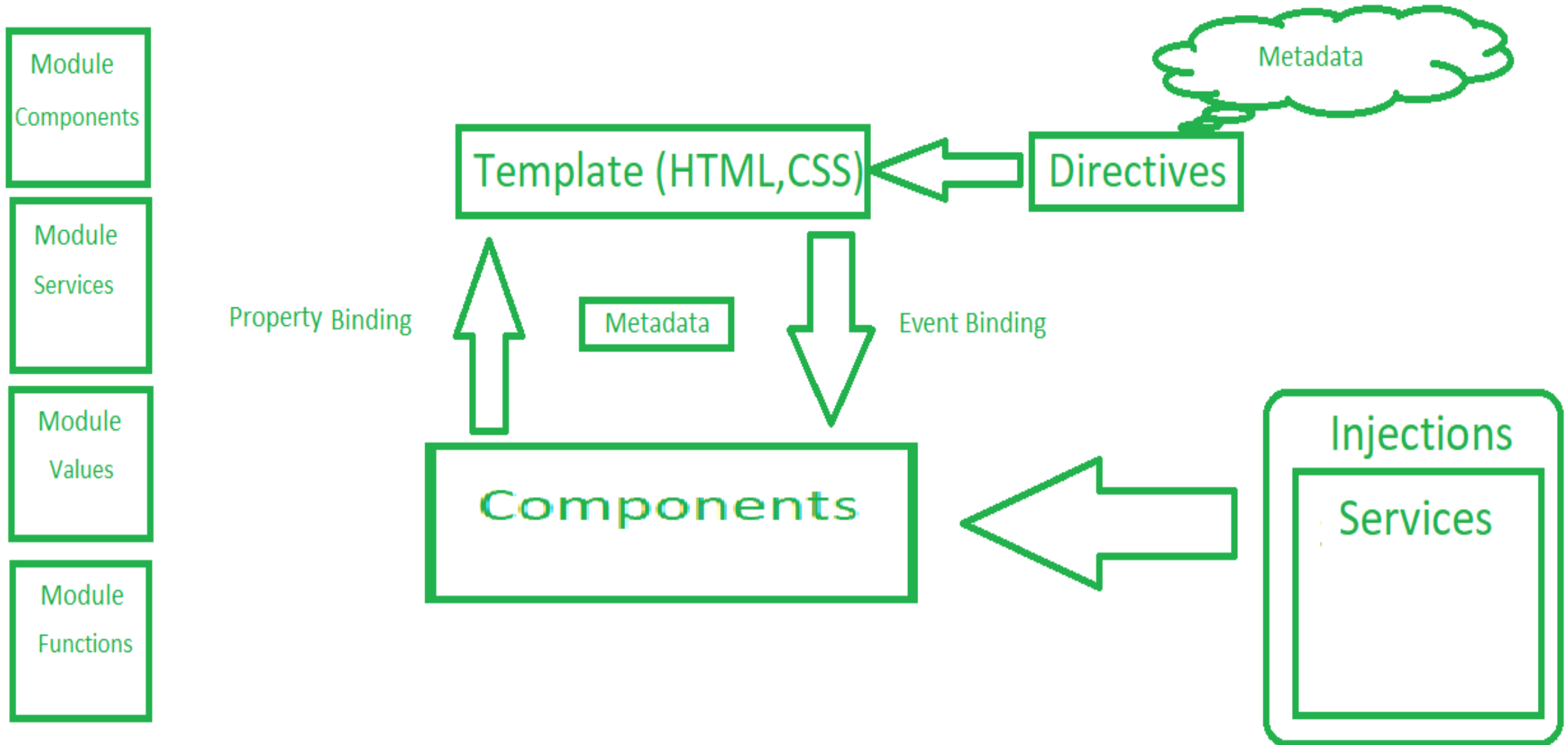




# ARCHITECTURE DU FRAMEWORK ANGULAR



# ARCHITECTURE DE BASE DE ANGULAR



# LES COMPSANTS

Chaque composant se compose de:

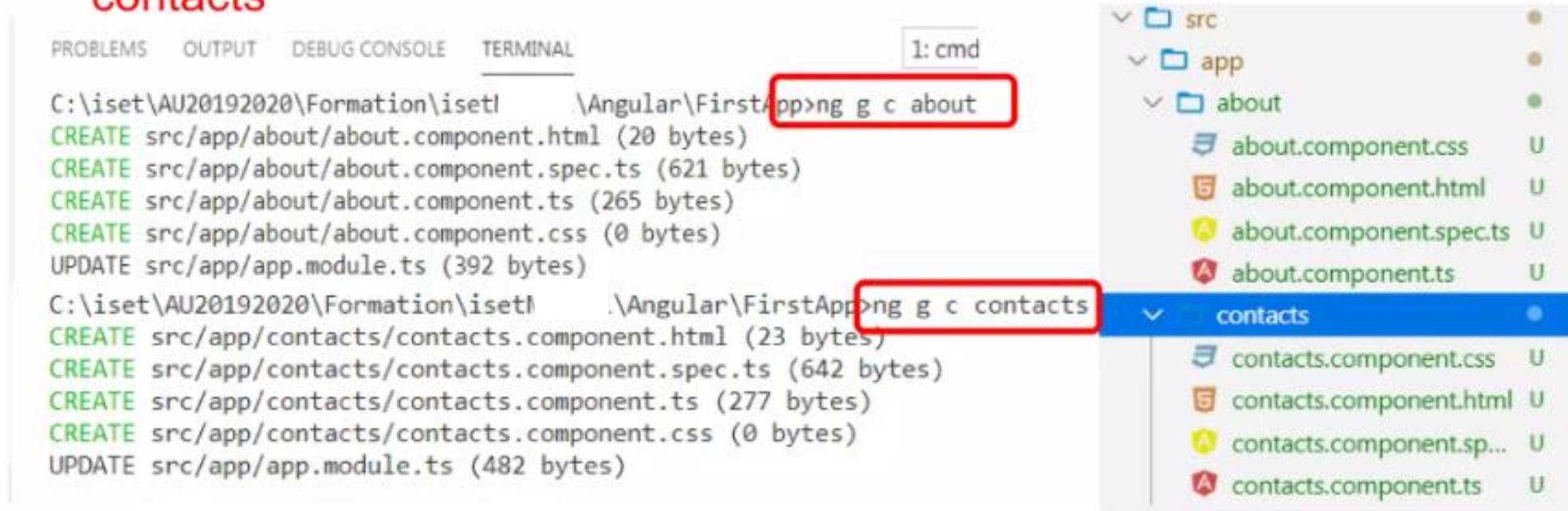
- ☐ Une classe représentant sa logique métier
- ☐ Une template HTML: représentant sa vue
- ☐ Une feuille de style css
- ☐ Un fichier de test unitaire

# CRÉATION DE NOUVEAUX COMPOSANTS

- Pour créer facilement des composants Angular, on peut utiliser la commande ng comme suit :

**ng generate component NomComposant**

- Dans notre exemple, nous allons créer deux composants : **about** et **contacts**



The screenshot displays an IDE interface with a terminal window and a file explorer. The terminal shows the execution of two Angular CLI commands to generate components. The first command, `ng g c about`, is highlighted with a red box. The second command, `ng g c contacts`, is also highlighted with a red box. The file explorer on the right shows the resulting directory structure, with the `about` and `contacts` folders expanded to show the generated files.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: cmd
C:\iset\AU20192020\Formation\isetl \Angular\FirstApp>ng g c about
CREATE src/app/about/about.component.html (20 bytes)
CREATE src/app/about/about.component.spec.ts (621 bytes)
CREATE src/app/about/about.component.ts (265 bytes)
CREATE src/app/about/about.component.css (0 bytes)
UPDATE src/app/app.module.ts (392 bytes)
C:\iset\AU20192020\Formation\isetl \Angular\FirstApp>ng g c contacts
CREATE src/app/contacts/contacts.component.html (23 bytes)
CREATE src/app/contacts/contacts.component.spec.ts (642 bytes)
CREATE src/app/contacts/contacts.component.ts (277 bytes)
CREATE src/app/contacts/contacts.component.css (0 bytes)
UPDATE src/app/app.module.ts (482 bytes)
```

File Explorer Structure:

- src
  - app
    - about
      - about.component.css U
      - about.component.html U
      - about.component.spec.ts U
      - about.component.ts U
    - contacts
      - contacts.component.css U
      - contacts.component.html U
      - contacts.component.sp... U
      - contacts.component.ts U

about.component.ts ✕

src > app > about > about.component.ts > ...

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-about',
5    templateUrl: './about.component.html',
6    styleUrls: ['./about.component.css']
7  })
8  export class AboutComponent implements OnInit {
9
10     constructor() { }
11
12     ngOnInit() {
13     }
14
15  }
16
```

**Décorateur:** indique à Angular que cette classe joue le rôle d'un composant avec:

- Un sélecteur 'app-about'
- Un template HTML  
'./about.component.html'
- Un seul fichier de style css  
'./about.component.css'

## Un composant doit être déclaré dans un module

app.module.ts

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

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { AboutComponent } from './about/about.component';
6 import { ContactsComponent } from './contacts/contacts.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     AboutComponent,
12     ContactsComponent
13   ],
14   imports: [
15     BrowserModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
21
```

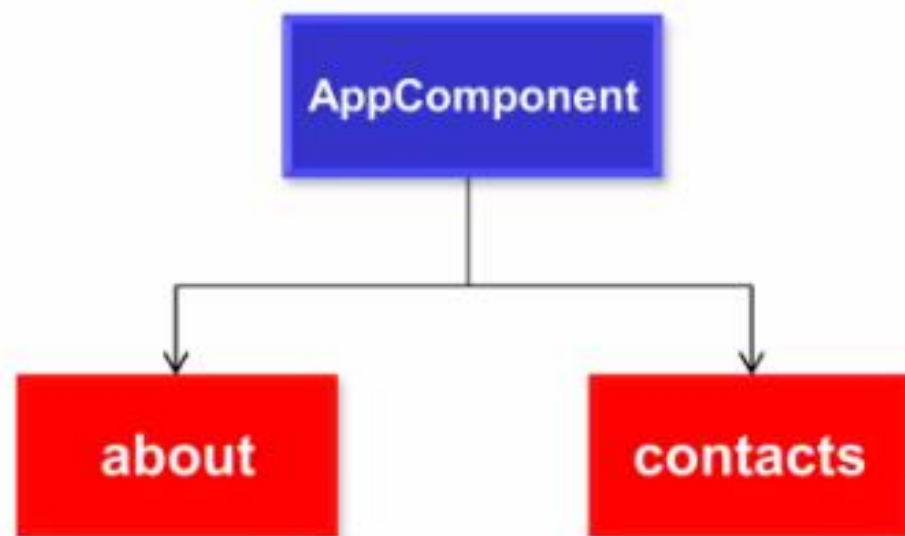


- Un composant peut être inséré dans n'importe quel partie HTML de l'application en utilisant son **sélecteur**
- Dans cet exemple, les deux composants **about** et **contacts** sont insérés à l'intérieur du composant racine **AppComponent**

app.component.html X

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

```
1
2 <h2>Bonjour de {{title}}</h2>
3 <div>
4 <app-about></app-about>
5 </div>
6 <div>
7 <app-contacts></app-contacts>
8 </div>
9
```



# LIAISON DE DONNÉES



# DOM

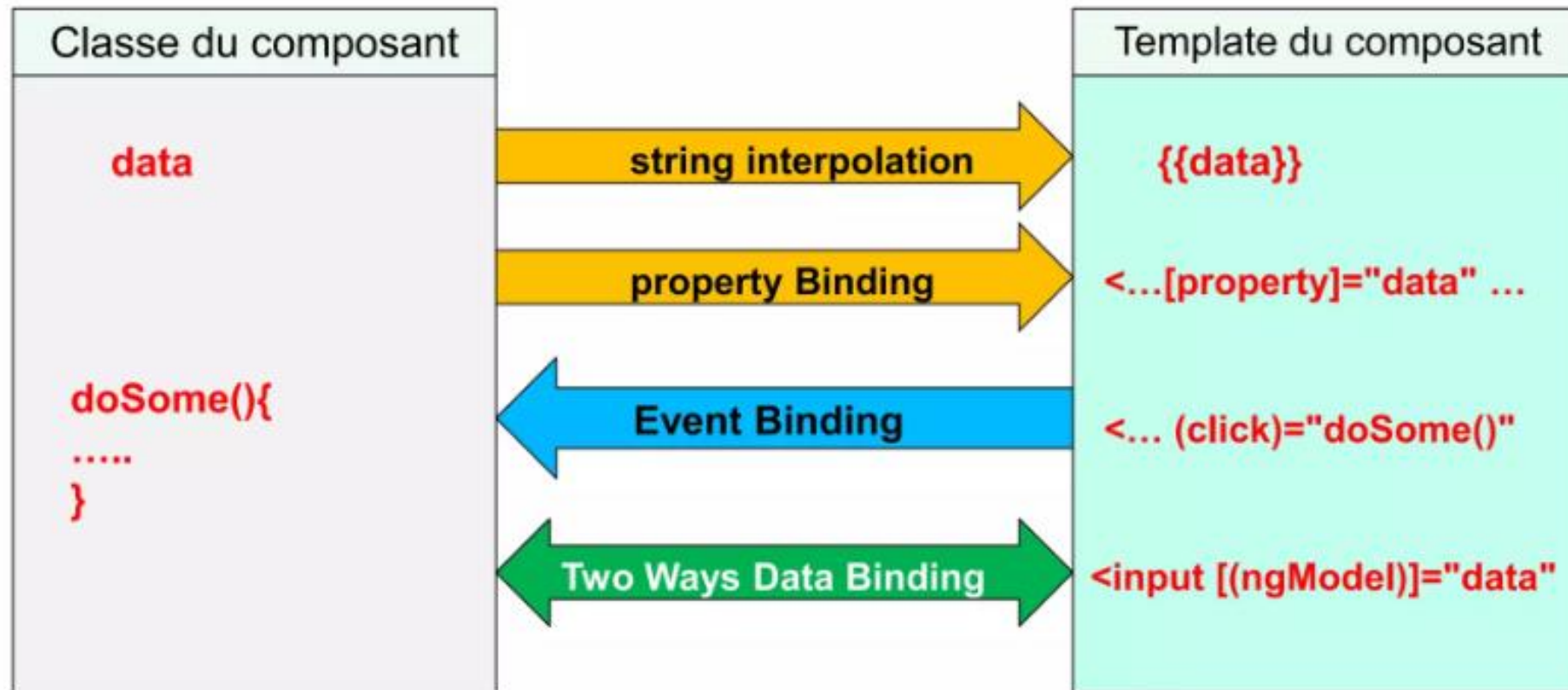
## DOM

La structure logique des documents et des documents accessibles et manipulés est définie à l'aide d'éléments DOM. Il définit les événements, les méthodes et les propriétés de tous les éléments HTML en tant qu'objets. DOM dans Angular agit comme une API (interface de programmation) pour javascript.

Chaque fois qu'une page Web est chargée, le navigateur crée un objet de modèle de document (DOM) de cette page.

# DATA BINDING

- Permet de faire un lien entre les données de la classe du composant et son Template associé



# DATA BINDING : EXEMPLE

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-firsdtdcomp',
5   templateUrl: './firsdtdcomp.component.html',
6   styleUrls: ['./firsdtdcomp.component.css'],
7 })
8 export class FirsdtdcompComponent {
9   info = { nom: 'Alya',
10            email: 'alya.khayati@hotmail.com',
11            age: 30 };
12   bgColor = 'red';
13
14   Comments: any[] = [];
15
16   comment = { id: 0, message: '' };
17
18   newComment = false;
19   addComment() {
20     if (this.comment.message !== '') {
21       this.comment.id = this.Comments.length + 1;
22       this.Comments.push({
23         id: this.comment.id,
24         message: this.comment.message,
25       });
26       this.comment.message = '';
27     }
28   }
29 }
30
```

Bonjour

1. Alya
2. alya.khayati@hotmail.com
3. 30

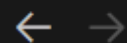
AjouterCommentaire

Liste des messages:

- 1 : Bonjour



File Edit Selection View Go Run ...



tp1



<> firstdtcomp.component.html X

TS firstdtcomp.component.ts

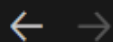
src > app > firstdtcomp > <> firstdtcomp.component.html > div > ul > li



```
5      <!-- {{ into.age }} -->
6    </ol>
7    <div>
8
9      <input type="text" [(ngModel)]="comment.message">
10     <button (click)="addComment()" [disabled]="newComment">Ajouter</button>
11   </div>
12   <div *ngIf="Comments.length>0; else noComments">
13     <h3>Liste des messages:</h3>
14     <ul>
15       <li *ngFor="let c of Comments">
16         {{c.id}} : {{c.message}}
17       </li>
18     </ul>
19   </div>
20   <ng-template #noComments>
21     <p [style.background]="bgColor">Liste est vide</p>
22   </ng-template>
```



File Edit Selection View Go Run ...



tp1



EXPLORER



TP1

> node\_modules

src

app

> div-comp

firstdtcomp

# firstdtcomp.component.css

<> firstdtcomp.component.html

TS firstdtcomp.component.spec.ts

TS firstdtcomp.component.ts

TS app-routing.module.ts

# app.component.css

<> app.component.html

TS app.component.spec.ts

TS app.component.ts

TS app.module.ts

assets

<> firstdtcomp.component.html

TS app.module.ts

TS firstdtcomp.component.ts

src > app > TS app.module.ts > AppModule

```
1  import { NgModule } from '@angular/core';
2  import { FormsModule } from '@angular/forms';
3  import { BrowserModule } from '@angular/platform-browser';
4
5  import { AppRoutingModule } from './app-routing.module';
6  import { AppComponent } from './app.component';
7  import { FirstdtcompComponent } from './firstdtcomp/firstdtcomp.component';
8  import { DivCompComponent } from './div-comp/div-comp.component';
9
10 @NgModule({
11   declarations: [AppComponent, FirstdtcompComponent, DivCompComponent],
12   imports: [BrowserModule,
13             AppRoutingModule,
14             FormsModule],
15   providers: [],
16   bootstrap: [AppComponent],
17 })
18 export class AppModule {}
19
```

# LES DIRECTIVES

# LES DIRECTIVES

Les directives sont des classes permettant d'enrichir et modifier la vue par simple ajout d'attributs Html sur le template

Il existe deux types de directives:

- **Les directives structurelles:** Elles ont pour but de modifier le DOM en ajoutant, enlevant ou remplaçant un élément du DOM.
- **Les attribute directives:** Elles ont pour but de modifier l'apparence ou le comportement d'un élément.

# LES DIRECTIVES STRUCTURELLES

ngIf

ngFor

NgSwitch



# NGIF

- Permet de supprimer ou de recréer l'élément courant suivant l'expression passée en paramètre
- Exemple:

`<div *ngIf="1 > 0">` Afficher la div`</div>`

`<div *ngIf="1 < 0">` N'affiche pas la div`</div>`

```
<div *ngIf="afficherNom; else elseBlock">
```

**Alya Khayati**

```
</div>
```

```
<ng-template #elseBlock>
```

**\*\*\*\*\***

```
</ng-template>
```

```
<div *ngIf="afficherNom;then thenBlock; else elseBlock"></div>
```

```
<ng-template #thenBlock>
```

**Alya Khayati**

```
</ng-template>
```

```
<ng-template #elseBlock>
```

**\*\*\*\*\***

```
</ng-template>
```

# NGFOR

```
export class TestComponent implements OnInit {  
    public couleurs = ['rouge', 'vert', 'bleu'];  
  
    constructor() { }  
  
    ngOnInit() {  
    }  
}
```

Liste des couleurs:

```
<div *ngFor="let c of couleurs">  
    {{c}}  
</div>
```

Liste des couleurs:

- rouge
- vert
- bleu

# NGSWITCH

```
<div [ngSwitch]="couleur">  
  <div *ngSwitchCase="'rouge'">couleur rouge</div>  
  <div *ngSwitchCase="'bleu'">couleur bleu</div>  
  <div *ngSwitchCase="'vert'">couleur vert</div>  
  <div *ngSwitchDefault>choisir une couleur</div>  
</div>
```

# LES ATTRIBUTE DIRECTIVES

ngStyle

ngClass

# NGSTYLE

- permet de modifier le style d'un élément HTML
- s'utilise conjointement avec le **property binding** pour récupérer des valeurs définies dans la classe

```
public styleCorrect={'background-color':'green'};
```

```
<div [ngStyle]="styleCorrect">Réponse</div>
```

contacts works!

Réponse

# NGCLASS

---

- permet d'attribuer de nouvelles classes d'un élément HTML
- s'utilise conjointement avec le **property binding** pour récupérer des valeurs définies dans la classe ou dans la feuille de style

```
<div [ngClass]="{'text-success':question.isCorrect}">Réponse</div>
```



# QUIZZ

**1) Dans Angular CLI, que signifie CLI ?**

- a) Control Link Interface
- b) Control Layer Interface
- c) Command Layer Interface
- d) Command Line Interface

**2) Dans le fichier index.html, à quoi correspond la balise `<app-root>` ?**

- a) Au component AppComponent
- b) Au module AppModule
- c) Au contenu de main.ts
- d) Au component de test TestComponent

### **3)Qu'est-ce que la string interpolation ?**

- a) La string interpolation permet de concaténer plusieurs chaînes de caractères ensemble.
- b) La string interpolation remplace un chiffre par une chaîne de caractères.
- c) La string interpolation permet d'afficher la valeur d'une variable dans le DOM.
- d) La string interpolation permet de récupérer une chaîne de caractères du DOM pour la traiter dans le fichier TypeScript.

#### **4)Quels sont les fichiers principaux d'un component Angular ?**

- a) Un fichier HTML, un fichier SCSS et un fichier TS.
- b) Un fichier HTML, un fichier SCSS et un fichier JS.
- c) Un fichier XML, un fichier CSS et un fichier TS.
- d) Un fichier HTML, un fichier TS et un fichier JS.

## **5)Qu'est-ce que la liaison par événement, ou event binding ?**

- a) Elle permet de lier la valeur d'une variable à un attribut d'un élément du DOM.
- b) Elle permet de lier une méthode TypeScript à un événement du DOM.
- c) Elle permet de remplacer une valeur par une autre dans le DOM.
- d) Elle permet de réagir à des changements de valeur d'un élément dans le DOM.

**6) Mon component `CoffeeCupComponent` a une propriété injectable `size` . Quelle syntaxe est correcte pour lui passer une variable appelée `"largeSize"` ?**

- a) `<app-coffee-cup size="[largeSize]"></app-coffee-cup>`
- b) `<app-coffee-cup (size)="largeSize"></app-coffee-cup>`
- c) `<app-coffee-cup [size]="largeSize"></app-coffee-cup>`
- d) `<app-coffee-cup [largeSize]="size"></app-coffee-cup>`

# ROUTAGE ET NAVIGATION