DATABINDING

ULT – DÉPARTEMENT GÉNIE INFORMATIQUE

Module: Projet Framework et Développement

Public cible: 5° Génie Logiciel

Houcem Hedhly

houssem.hedhli@ult-tunisie.com

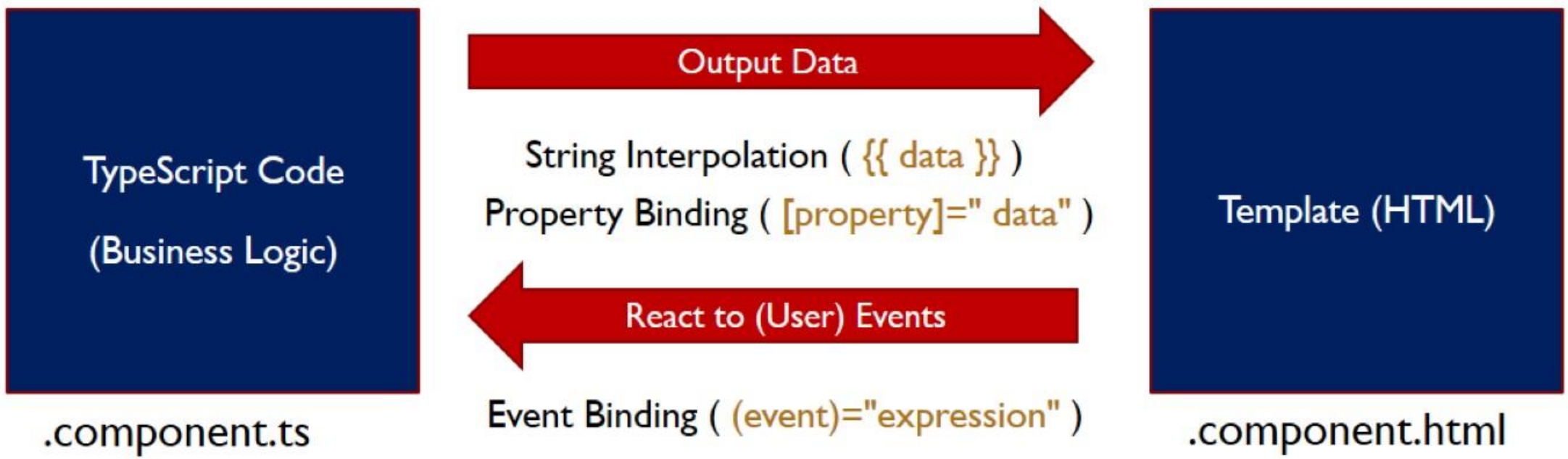
2021/2022

PLAN

I. Principe de base
II. String Interpolation
III. Property Binding
IV. Event Binding
V. Two-way binding
VI. Pipes

DATABINDING ?

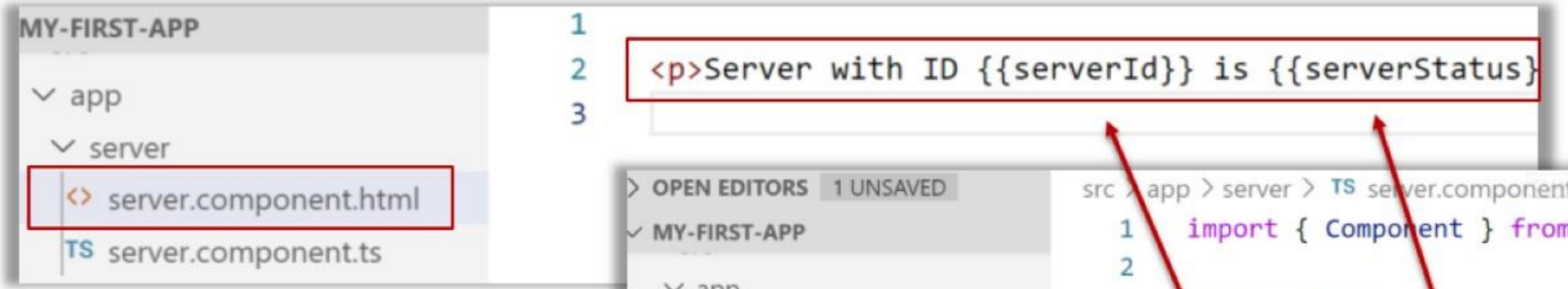
Databinding = Liaison de donnés

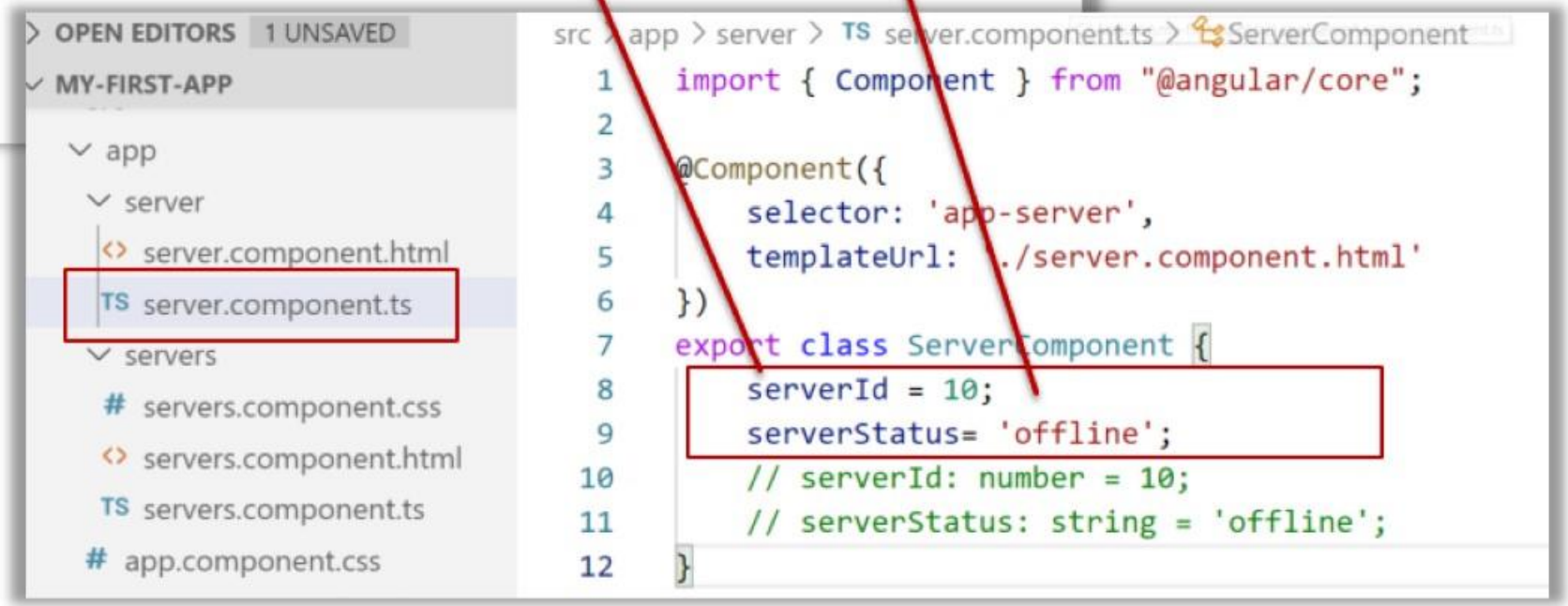


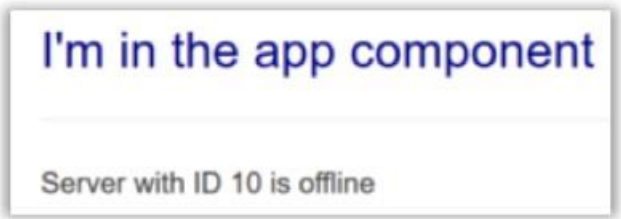
Combination of Both: Two-Way Binding ([(ngModel)]="data")

STRING INTERPOLATION

- On veut afficher un contenu variable dans les Templates
- Pour cela on peut créer des variables au niveau du component







STRING INTERPOLATION

- **{{ serverId }}** // le contenu d'une variable
- **{{ 'server' }}** // une constante
- **NB**: le résultat final des {{ }} doit être une chaine de caractères

- On peut utiliser des fonctions aussi

```
export class ServerComponent {
    serverId = 10;
    serverStatus= 'offline';
    // serverId: number = 10;
    // serverStatus: string = 'offline';

    getServerStatus() {
        return this.serverStatus;
    }
}
```

```
<p>{{ 'Server' }} with ID {{ serverId }} is {{ getServerStatus() }}</p>
```

I'm in the app component

Server with ID 10 is offline

STRING INTERPOLATION (EXEMPLES)

- public **name** = "Foulen";
- **greetUser()** { return "Hello " + this.name; }

- Welcome {{ name }} // Affichage du contenu d'une variable
- {{ 2+2 }} // Evaluation d'une expression
- {{ "Welcome " + name }} // Concaténation de plusieurs expressions
- {{ name.length }} // Utilisation de propriétés JavaScript
- {{ name.toUpperCase() }} // Utilisation de méthodes JavaScript usuelles
- {{ greetUser() }} // Utilisation de méthodes JavaScript personnalisés

- {{ a = 2+2 }} // ERROR pas d'affectation
- {{ window.location.href }} // ERROR On ne peut pas accéder aux variables globales JavaScript
  // Solution : Utiliser des méthodes

PROPERTY BINDING

- Property binding : liaison de propriété.
- Une autre façon de créer de la communication dynamique entre le TypeScript et les propriétés des différents éléments du Template.
- Lier dynamiquement une propriété d'un élément du DOM avec une variable dans le code TypeScript.
- La liaison va dans un seul sens: depuis le TS vers le DOM.
- On met la propriété entre [ ], ce qui la rend une propriété cible.

PROPERTY BINDING

- public **myID** = "testID";
- <input **[id]**="**myID**" type="text" value="Ouali">

On a déclaré une variable myID

Property Binding
On a lié l'attribut id avec la variable myID

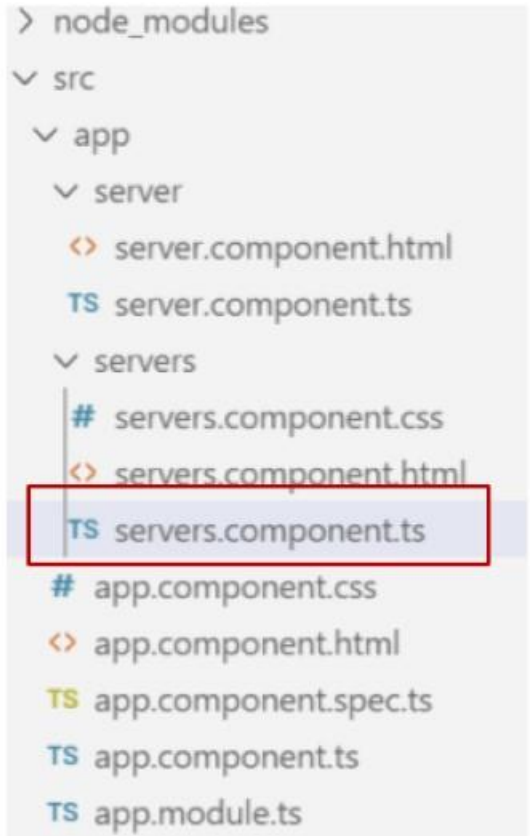
```
<input id="testID" _ngcontent-jnl-c1="" type="text" value="Ouali">
```

EVENT BINDING

- Avec le string interpolation et le property binding, vous savez communiquer depuis votre code TypeScript vers le Template HTML.
- Maintenant, on verra comment réagir dans votre code TypeScript aux événements venant du Template HTML.
- La technique de liaison d'évènement permet de rester à l'écoute des actions faites par l'utilisateur (clic, appui sur une touche du clavier, mouvements de la souris, etc.), et de prévoir des comportements.

EVENT BINDING

- **NB** : onCreateServer () par convention on ajoute « **on** » au début de la méthode pour spécifier qu'elle va être déclenchée à partir de l'interface. (Ce n'est pas obligatoire)




```
10  export class ServersComponent implements OnInit {
11    allowNewServer = false;
12    serverCreationStatus = 'No server was created!';
13
14    constructor() {
15      setTimeout(() => {
16        this.allowNewServer = true;
17      }, 2000);
18    }
19
20    ngOnInit() {
21    }
22
23    onCreateServer () {
24      this.serverCreationStatus = 'Server was created!';
25    }
26  }
```

EVENT BINDING

- Pour associer notre méthode ***onCreateServer( )*** au bouton de l'interface

```
1   <button class="btn btn-primary"
2       [disabled]="!allowNewServer"
3       (click)="onCreateServer()">Add Server</button>
4
5   <!-- <p>{{ allowNewServer }}</p> -->
6   <!-- <p [innerText]="allowNewServer"></p> -->
7   <p>{{ serverCreationStatus }}</p>
8   <app-server></app-server>
9   <app-server></app-server>
```

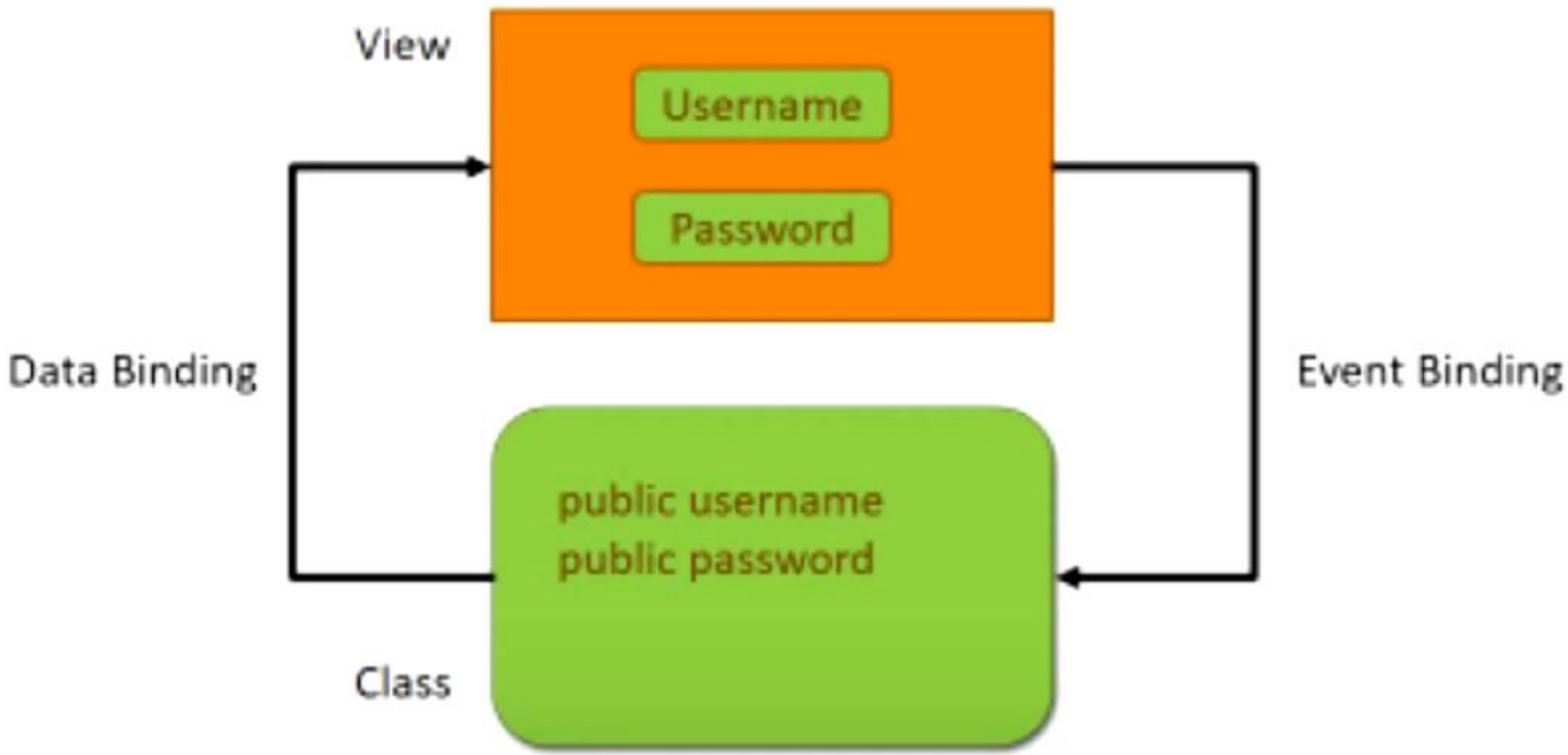
Evénement déclencheur.
Entre () et sans le « on » de javascript

Un code ou une fonction qui va s'exécuter lorsque l'événement est déclenché.
La fonction doit s'écrire avec les ()

TWO-WAY BINDING

- La liaison à double sens (ou two-way binding) utilise la liaison par propriété et la liaison par événement en même temps.
- On l'utilise, par exemple, pour les formulaires, afin de pouvoir déclarer et de récupérer le contenu des champs, entre autres.
- Avec la méthode Two-way binding on mélange les deux autres méthodes Property Binding et Event Binding en utilisant la notation **[( )]**.

Two Way Binding



TWO-WAY BINDING

```
<input
    type="text"
    class="form-control"
    [(ngModel)]="serverName">
    <p>{{ serverName }}</p>
```

[(ngModel)] : ça fonctionne dans les deux sens.
Si « serverName » change ça sera affiché dans l'interface et inversement si l'input change la valeur de « serverName » change aussi.

TWO-WAY BINDING

- REMARQUE:
  - Pour que le Two-way binding fonctionne, on doit activer la directive `ngModel`.
  - On doit importer `FormsModule` depuis `@angular/forms` dans `AppModule`.
  - On doit ajouter, ensuite, le `FormsModule` dans le tableau des `imports[]`.

PIPES

- Les Pipes nous permettent de transformer les données avant de les afficher dans l'interface.
- On peut transformer l'affichage de variables contenant:
  - string
  - monétique
  - date
  - etc.
- On déclare un Pipe une seule fois et on l'utilise dans toute l'application.
- Les données restent dans leurs formats standards.

PIPES

- Soient les variables suivantes:

```
public nom = "Angular";
public message = "Welcome to ISET Bizerte";
public person = {
  firstname: "James",
  lastname : "Bond"
};
public date = new Date();
```

PIPES

- <h2>{{ nom }}</h2> // Angular
- <h2>{{ nom | **lowercase** }}</h2> // angular
- <h2>{{ nom | **uppercase** }}</h2> // ANGULAR
- <h2>{{ message | **titlecase** }}</h2> // Welcome To Iset Bizerte
- <h2>{{ nom | **slice:3** }}</h2> // ular
- <h2>{{ nom | **slice:3:5** }}</h2> // ul
- <h2>{{ person | **json** }}</h2> // { "firstName": "James", "lastName": "Bond" }
- <h2>{{ 5.678 | **number:'1.2-3'** }}</h2> // 5.678
- <h2>{{ 5.678 | **number:'3.4-5'** }}</h2> // 005.6780
- <h2>{{ 5.678 | **number:'3.1-2'** }}</h2> // 005.68

PIPES

- <h2>{{ 0.25 | **percent** }}</h2> // 25%
- <h2>{{ 0.25 | **currency** }}</h2> // $0.25
- <h2>{{ 0.25 | **currency: 'EUR'** }}</h2> // €0.25
- <h2>{{ date }}</h2> // Fri Sep 20 2019 09:54:07 GMT+0200 (heure d'été d'Europe centrale)
- <h2>{{ date | **date:'short'** }}</h2> // 9/20/19, 9:54 AM
- <h2>{{ date | **date:'shortDate'** }}</h2> // 9/20/19
- <h2>{{ date | **date:'shortTime'** }}</h2> // 9:54 AM
- <h2>{{ date | **date:'medium'** }}</h2> // Sep 20, 2019, 9:54:07 AM
- <h2>{{ date | **date:'long'** }}</h2> // September 20, 2019 at 9:54:07 AM GMT+2