

# Cours Angular : Composants et Data Binding

## 1. Introduction

Dans Angular, **les composants** sont les briques fondamentales de l'application. Ils contrôlent une partie de l'interface utilisateur (UI) et gèrent l'affichage des données grâce au **data binding**.

Objectifs de ce cours : - Comprendre ce qu'est un composant Angular - Créer et structurer un composant - Maîtriser les différents types de data binding - Utiliser ces notions dans des cas pratiques concrets

---

## 2. Les composants Angular

### 2.1 Définition

Un composant Angular est constitué de : - Une **classe TypeScript** (logique) - Un **template HTML** (vue) - Des **styles CSS/SCSS** (apparence) - Un **décorateur** `@Component` (métadonnées)

Chaque composant représente une portion réutilisable de l'interface.

---

### 2.2 Structure d'un composant

Exemple de composant `user.component.ts` :

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

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css']
})
export class UserComponent {
  name: string = 'Jean';
  age: number = 25;
}
```

Template HTML associé (`user.component.html`) :

```
<h2>Profil utilisateur</h2>
<p>Nom : {{ name }}</p>
<p>Âge : {{ age }}</p>
```

Utilisation du composant dans un autre template :

```
<app-user></app-user>
```

## 2.3 Le décorateur `@Component`

Propriétés principales : - `selector` : nom de la balise HTML - `template` ou `templateUrl` : vue HTML - `styles` ou `styleUrls` : styles CSS

## 3. Le Data Binding

Le **data binding** permet de synchroniser les données entre la classe TypeScript et le template HTML.

Angular propose **4 types de data binding**.

## 4. Interpolation (One-way binding)

### 4.1 Principe

L'interpolation permet d'afficher une donnée TypeScript dans le HTML.

Syntaxe :

```
{{ expression }}
```

### 4.2 Exemple

```
export class ProductComponent {  
  productName = 'Ordinateur';  
  price = 450000;  
}
```

```
<p>Produit : {{ productName }}</p>  
<p>Prix : {{ price }} FCFA</p>
```

👉 Sens : **TypeScript → HTML**

## 5. Property Binding

### 5.1 Principe

Le property binding permet de lier une **propriété HTML** à une variable TypeScript.

Syntaxe :

```
[propriete]="expression"
```

## 5.2 Exemple

```
export class ImageComponent {  
    imageUrl = 'https://via.placeholder.com/150';  
    isEnabled = true;  
}
```

```
<img [src]="imageUrl">  
<button [disabled]="isEnabled">Valider</button>
```

👉 Sens : TypeScript → HTML

---

## 6. Event Binding

### 6.1 Principe

L'event binding permet de réagir à une action de l'utilisateur (clic, saisie, survol...).

Syntaxe :

```
(event)="methode()"
```

### 6.2 Exemple

```
export class CounterComponent {  
    count = 0;  
  
    increment() {  
        this.count++;  
    }  
}
```

```
<p>Compteur : {{ count }}</p>  
<button (click)="increment()>+1</button>
```

👉 Sens : HTML → TypeScript

---

## 7. Two-Way Binding

### 7.1 Principe

Le two-way binding permet une synchronisation **dans les deux sens** entre la vue et la classe.

Syntaxe :

```
[(ngModel)]="variable"
```

👉 Nécessite l'import de `FormsModule`.

### 7.2 Configuration

Dans `app.module.ts` :

```
import { FormsModule } from '@angular/forms';

@NgModule({
  imports: [FormsModule]
})
export class AppModule {}
```

### 7.3 Exemple

```
export class FormComponent {
  username = '';
}
```

```
<input type="text" [(ngModel)]="username">
<p>Nom saisi : {{ username }}</p>
```

👉 Sens : **HTML ⇌ TypeScript**

## 8. Cas pratique complet

### 8.1 Objectif

Créer un composant qui : - Affiche un nom - Permet de le modifier - Active/désactive un bouton

## 8.2 Composant

```
export class ProfileComponent {  
  name = 'Atene';  
  isActive = false;  
  
  toggle() {  
    this.isActive = !this.isActive;  
  }  
}
```

```
<h3>Profil</h3>  
<input [(ngModel)]="name">  
<p>Nom : {{ name }}</p>  
<button (click)="toggle()">Activer</button>  
<button [disabled]="!isActive">Envoyer</button>
```

## 9. Bonnes pratiques

- Un composant = une responsabilité
- Éviter la logique complexe dans le template
- Utiliser le data binding avec parcimonie
- Préférer les méthodes simples dans les events

## 10. Résumé

- Les composants sont le cœur d'Angular
- Le data binding connecte la logique à la vue
- 4 types : interpolation, property, event, two-way
- Essentiel pour construire des interfaces dynamiques

👉 Prochaine étape recommandée : **@Input / @Output et communication entre composants**