
Welcome to the Apprenticeship
Program – Session 26

Agenda

- What are Angular Components?
- Template Syntax Basics
- Data Binding in Angular
- Property Binding vs Event Binding

What is a Component

- **Definition:** A component controls a section of the screen (called a view).
- **It consists of:**
  - **Class –** Contains logic & data.
  - **Template –** Defines HTML view.
  - **Style –** Optional CSS for the view.

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

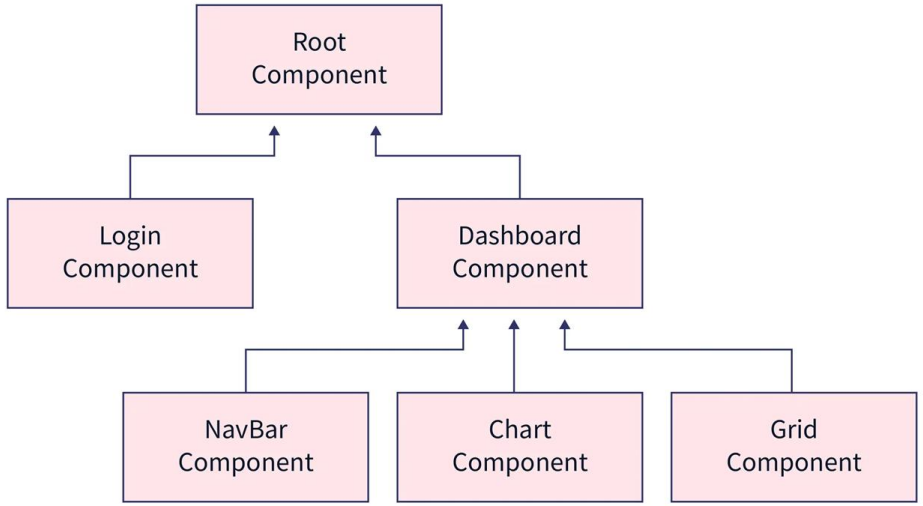
Anatomy of a Component

📁 **A typical component has:**

- hero.component.ts – logic & properties
- hero.component.html – view markup
- hero.component.css – styles

Declared in app.module.ts

✅ Each component is reusable and self-contained.



Component Class Example

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

@Component({
  selector: 'app-greeting',
  template: `<h1>Hello, {{name}}!</h1>`
})
export class GreetingComponent {
  name = 'Apprentices';
}
```

DEMO

What is a Template?

- A template defines how the component is rendered.
- It's just HTML + Angular template syntax.
- Templates can include:
  - Interpolation ({{ }})
  - Event bindings
  - Property bindings.
  - Directives (*ngIf, *ngFor)

```
<h2>Welcome, {{ user }}!</h2>
```

Interpolation ({{ }})

✅ Used to bind component data into HTML.

Basic interpolation:

```
<h1>Hello {{ userName }}</h1>
```

Expression Support: you can perform simple logic and method calls inside {{ }}.

```
<p>{{ 5 + 5 }}</p>
<p>{{ userName.toUpperCase() }}</p>
<p>{{ isLoggedIn ? 'Logout' : 'Login' }}</p>
```

Property Binding

✅ Pass data from component to template. Binds class properties to element attributes

Component:

```
profilePicUrl = 'assets/user.png';
isDisabled = true;
```

Template: Button will be disabled if isDisabled is true.

```
<img [src]="profilePicUrl">
<button [disabled]="isDisabled">Submit</button>
```

Event Binding

✅ Capture user interactions like clicks

Example:

```
<button (click)="showMessage()">Click Me</button>
<input (input)="logChange($event)">
```

Component:

```
showMessage() {
  alert("Button clicked!");
}

logChange(event: any) {
  console.log(event.target.value);
}
```

Event Binding – Mouse & Keyboard Events

✅ Angular supports all native DOM events.

Example:

```
<input (keyup.enter)="onEnterKey()">
<div (mouseover)="hoverEffect()">Hover here</div>
```

- Use Cases:
  - Form submissions
  - Keyboard shortcuts
  - Hover tooltips:

Two-Way Data Binding

✅ Combine property and event binding.

Uses [(ngModel)] (requires FormsModule).

```
<input [(ngModel)]="username">
<p>You typed: {{username}}</p>
```

- Updates in real-time as the user types. In above example
  - Every keystroke updates userName and displays it in real-time.
  - Requires FormsModule in app.module.ts.

Event-Driven Templates vs Template Driven Forms

Event Driven:

- Manually handle events using functions in component class
- Good for fine control.

```
<button (click)="submitData()">Submit</button>
```

```
submitData() {
  // Manual form handling logic
}
```

Template Driven:

- Leverages Angular's built-in form features
- Minimal code in component
- Good for simpler forms.

```
<form #userForm="ngForm" (ngSubmit)="onSubmit(userForm)">
  <input name="username" ngModel required>
</form>
```

```
onSubmit(form: NgForm) {
  console.log(form.value);
}
```

Component Hierarchy

- Angular apps are component trees
  - Parent → Child communication using:
    - @Input() – Pass data down
    - @Output() – Emit events up
- Break UI into smaller, manageable pieces..

```
@Component({
  selector: 'app-parent',
  template: `<app-child [childMessage]="parentMessage"></app-child>`
})
export class ParentComponent {
  parentMessage = "Hello from Parent!";
}
```

```
@Component({
  selector: 'app-child',
  template: `<p>{{ childMessage }}</p>`
})
export class ChildComponent {
  @Input() childMessage: string;
}
```

```
@Component({
  selector: 'app-child',
  template: `<button (click)="sendMessage()">Send Message</button>`
})
export class ChildComponent {
  @Output() messageSent = new EventEmitter<string>();

  sendMessage() {
    this.messageSent.emit("Hello from Child!");
  }
}
```

```
@Component({
  selector: 'app-parent',
  template: `<app-child (messageSent)="receiveMessage($event)"></app-child>`
})
export class ParentComponent {
  receiveMessage(message: string) {
    console.log(message); // "Hello from Child!"
  }
}
```

Conclusion and Q&A

- **Key Takeaways:**
  - Angular apps are built using components.
  - Templates control what is displayed to the user.
  - Data Binding connects the class logic to the UI.
  - Two-way binding allows live user interaction!

Questions? 👀