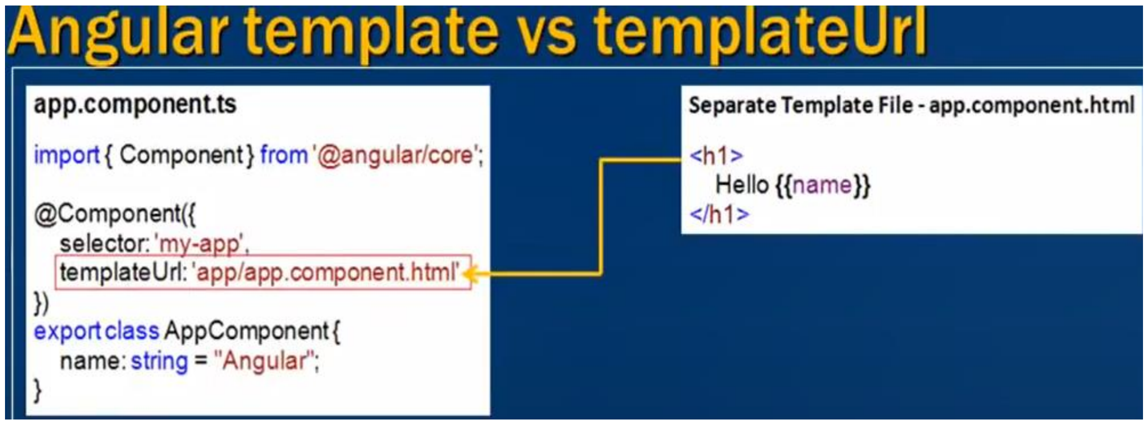
Angular 2(Opensource JS Framework)

Components – (template + class + metadata)

- **Template** – This is used to render the view for the application. This contains the HTML that needs to be rendered in the application. This part also includes the binding and directives.
- **Class** – This is like a class defined in any language such as C. This contains properties and methods. This has the code which is used to support the view. It is defined in TypeScript.
- **Metadata** – This has the extra data defined for the Angular class. It is defined with a decorator.

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

@Component ({
   selector: 'my-app',
   template: ` <div>
      <h1>{{appTitle}}</h1>
      <div>To Tutorials Point</div>
   </div> `,
})

export class AppComponent {
   appTitle: string = 'Welcome';
}
```



Angular template vs templateUrl

app.component.ts

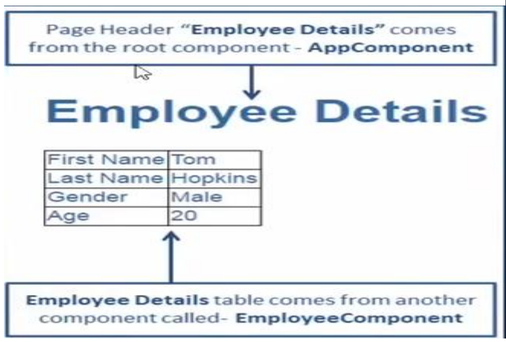
```
import { Component } from '@angular/core';

@Component({
   selector: 'my-app',
   templateUrl: 'app/app.component.html'
})
export class AppComponent {
   name: string = "Angular";
}
```

Separate Template File - app.component.html

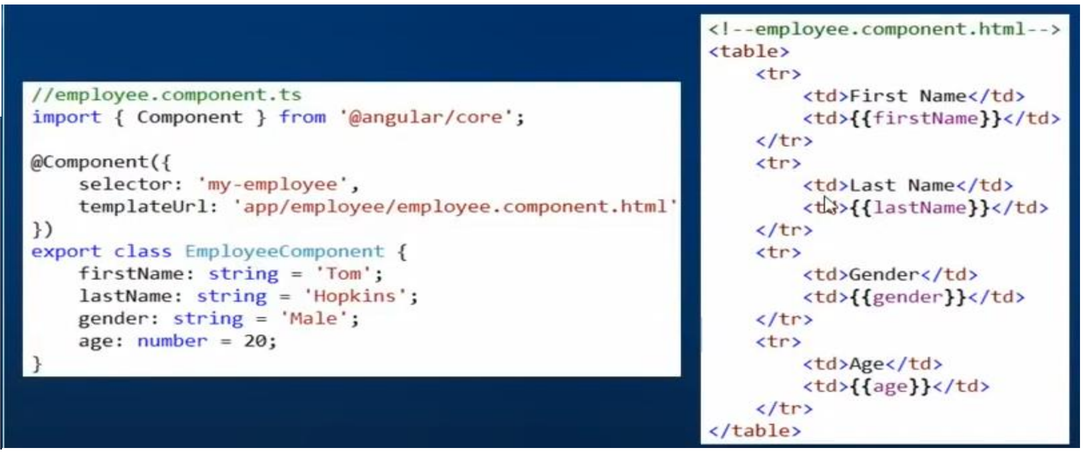
```
<h1>
   Hello {{name}}
</h1>
```

Nested Components :-

AppComponent is the root components.

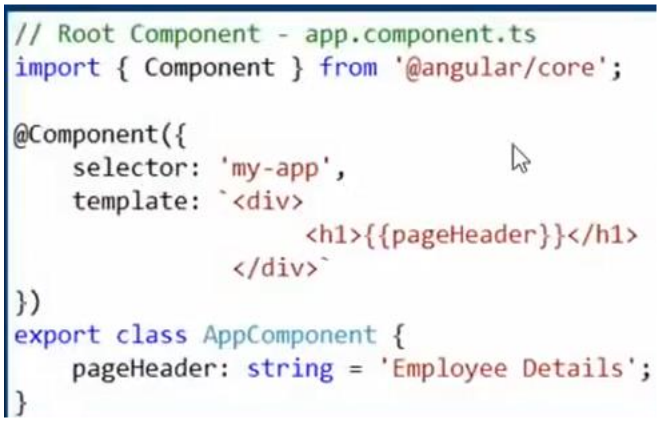

Page Header "Employee Details" comes from the root component - AppComponent
Employee Details
Employee Details table comes from another component called- EmployeeComponent

First Name	Tom
Last Name	Hopkins
Gender	Male
Age	20

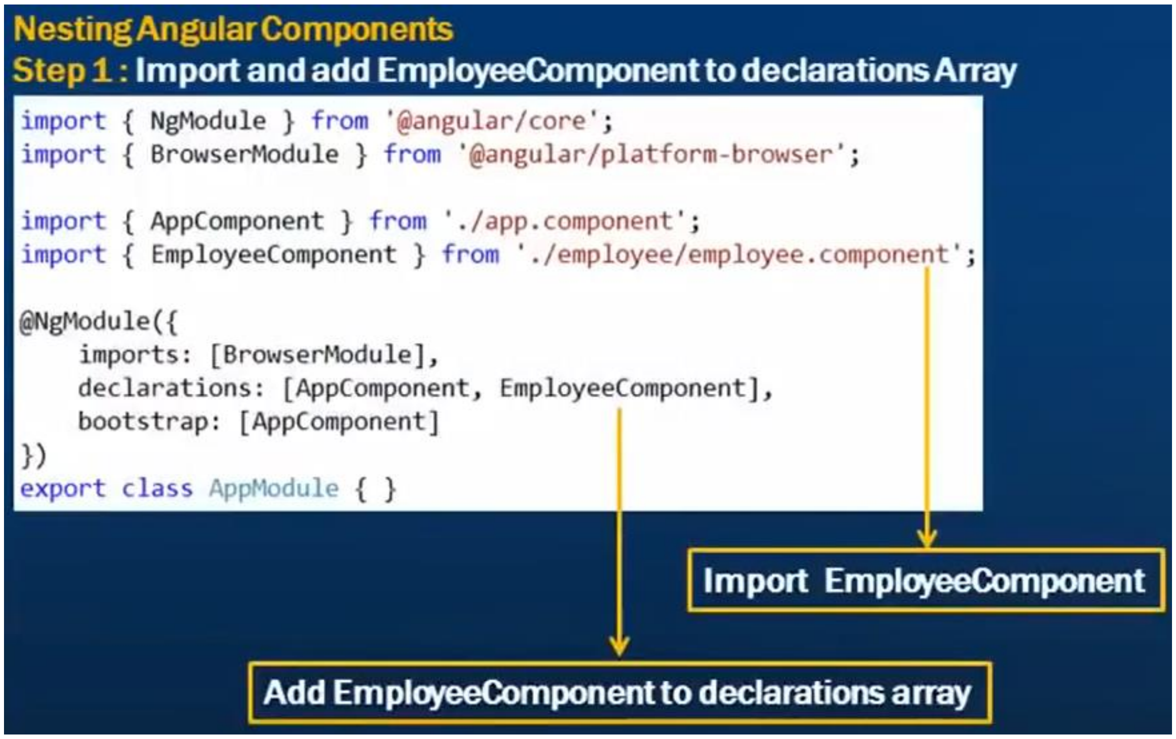


```
//employee.component.ts
import { Component } from '@angular/core';

@Component({
    selector: 'my-employee',
    templateUrl: 'app/employee/employee.component.html'
})
export class EmployeeComponent {
    firstName: string = 'Tom';
    lastName: string = 'Hopkins';
    gender: string = 'Male';
    age: number = 20;
}
```

```
<!--employee.component.html-->
<table>
    <tr>
        <td>First Name</td>
        <td>{{firstName}}</td>
    </tr>
    <tr>
        <td>Last Name</td>
        <td>{{lastName}}</td>
    </tr>
    <tr>
        <td>Gender</td>
        <td>{{gender}}</td>
    </tr>
    <tr>
        <td>Age</td>
        <td>{{age}}</td>
    </tr>
</table>
```



```
// Root Component - app.component.ts
import { Component } from '@angular/core';

@Component({
    selector: 'my-app',
    template: `<div>
                    <h1>{{pageHeader}}</h1>
               </div>`
})
export class AppComponent {
    pageHeader: string = 'Employee Details';
}
```

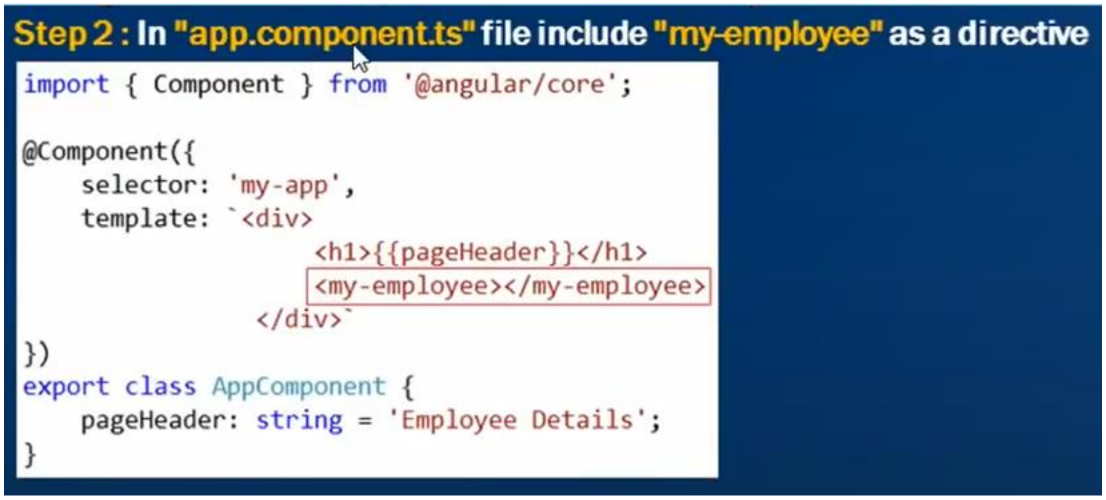


Nesting Angular Components

Step 1 : Import and add EmployeeComponent to declarations Array

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

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

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

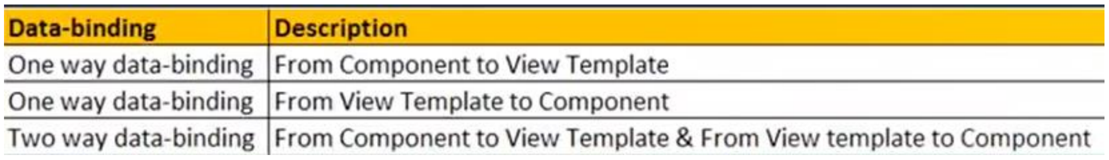
- Appmodule is the root module which bootstraps and launches the angular application.
- BrowserModule is required by all angular applications which run on browser. Also provides ng-for and ng-if directives.



Step 2 : In "app.component.ts" file include "my-employee" as a directive

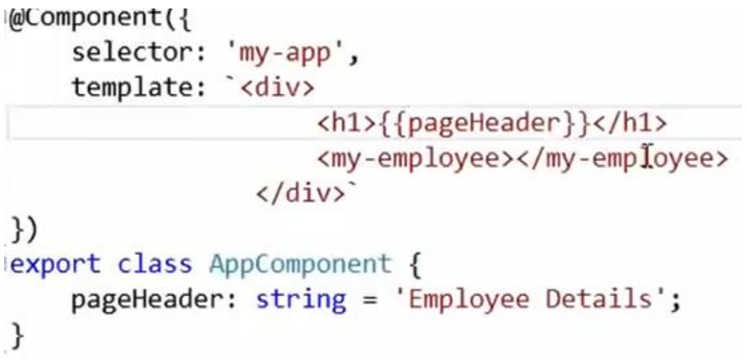
```
import { Component } from '@angular/core';

@Component({
    selector: 'my-app',
    template: `<div>
                    <h1>{{pageHeader}}</h1>
                    <my-employee></my-employee>
                </div>`
})
export class AppComponent {
    pageHeader: string = 'Employee Details';
}
```

Angular Interpolation :-



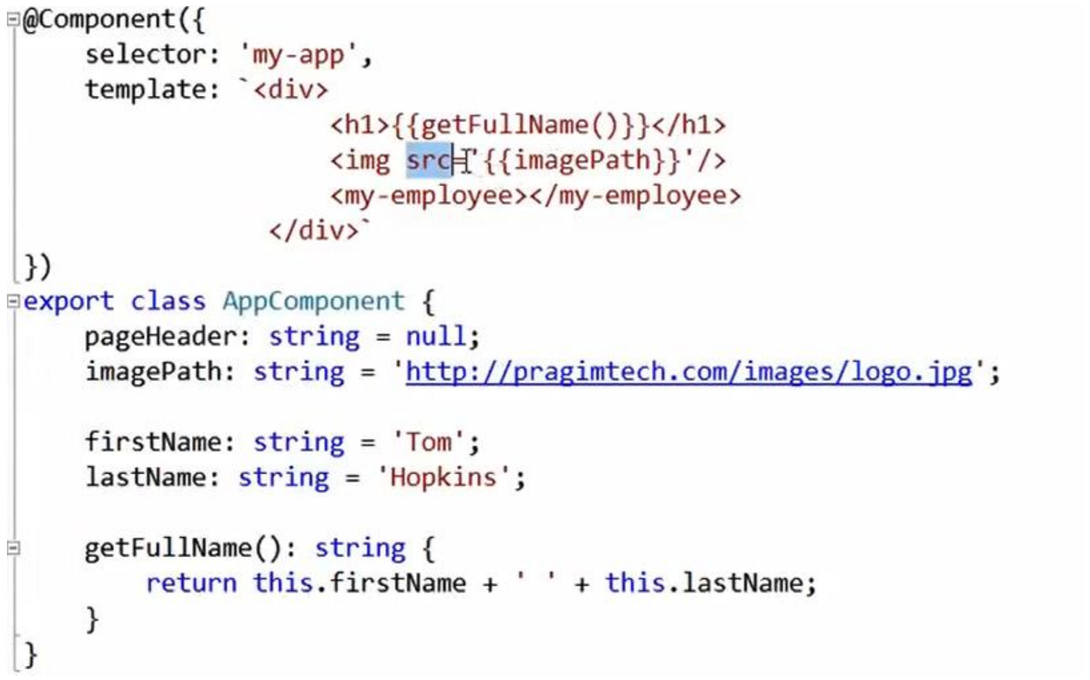
Data-binding	Description
One way data-binding	From Component to View Template
One way data-binding	From View Template to Component
Two way data-binding	From Component to View Template & From View template to Component

From Component to view Template: Interpolation example



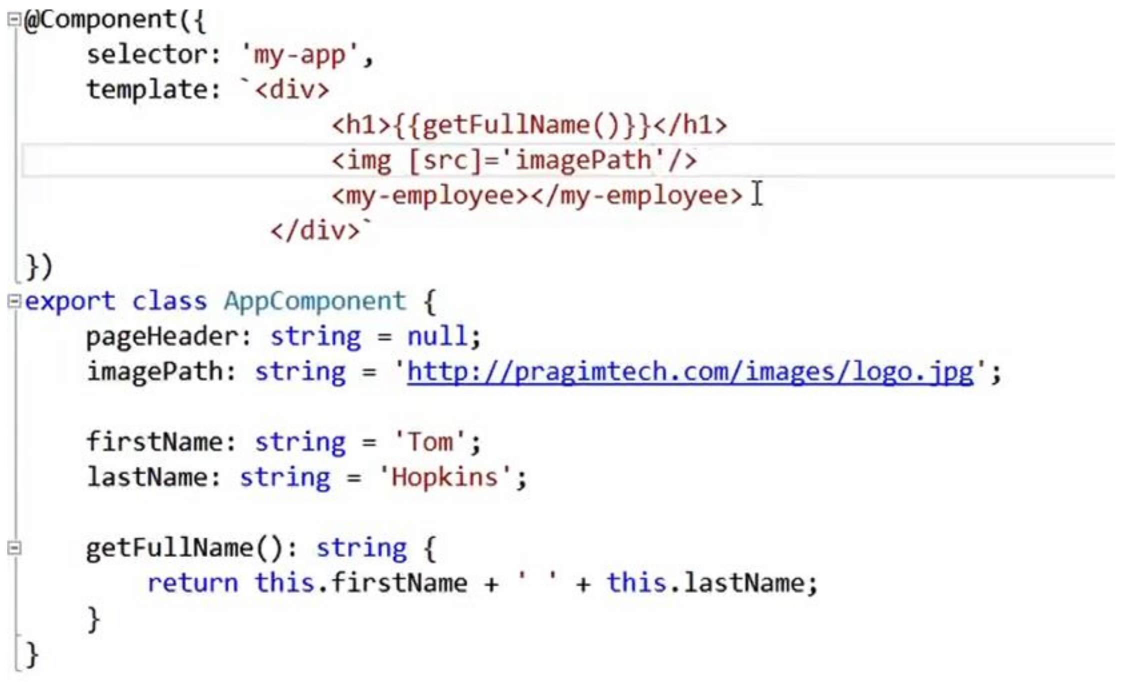
```
@Component({
    selector: 'my-app',
    template: `<div>
                    <h1>{{pageHeader}}</h1>
                    <my-employee></my-employee>
                </div>`
})
export class AppComponent {
    pageHeader: string = 'Employee Details';
}
```

pageHeader being used for one way binding.



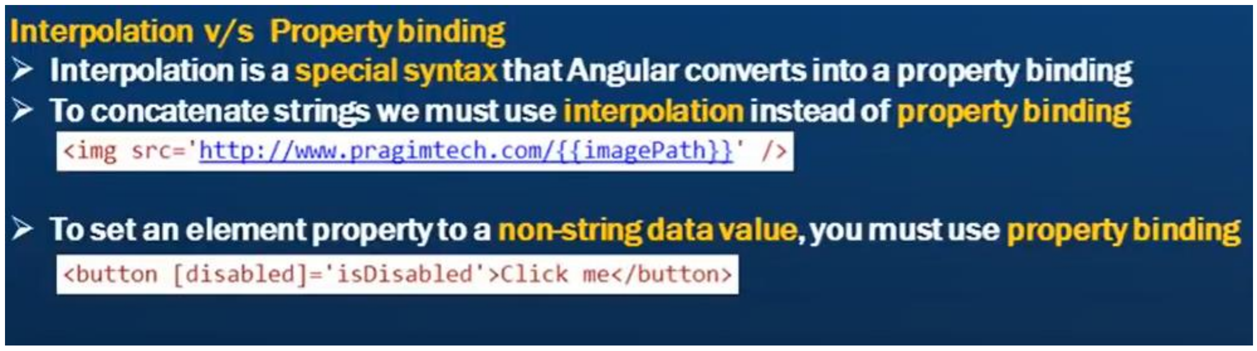
```
@Component({
    selector: 'my-app',
    template: `<div>
                    <h1>{{getFullName()}}</h1>
                    <img src='{{imagePath}}'/>
                    <my-employee></my-employee>
                </div>`
})
export class AppComponent {
    pageHeader: string = null;
    imagePath: string = 'http://pragimtech.com/images/logo.jpg';

    firstName: string = 'Tom';
    lastName: string = 'Hopkins';

    getFullName(): string {
        return this.firstName + ' ' + this.lastName;
    }
}
```

Property Binding :-



```
@Component({
    selector: 'my-app',
    template: `<div>
                    <h1>{{getFullName()}}</h1>
                    <img [src]='imagePath'/>
                    <my-employee></my-employee>
                </div>`
})
export class AppComponent {
    pageHeader: string = null;
    imagePath: string = 'http://pragimtech.com/images/logo.jpg';

    firstName: string = 'Tom';
    lastName: string = 'Hopkins';

    getFullName(): string {
        return this.firstName + ' ' + this.lastName;
    }
}
```

Interpolation and Property value both are ways of one way data binding.



Interpolation v/s Property binding

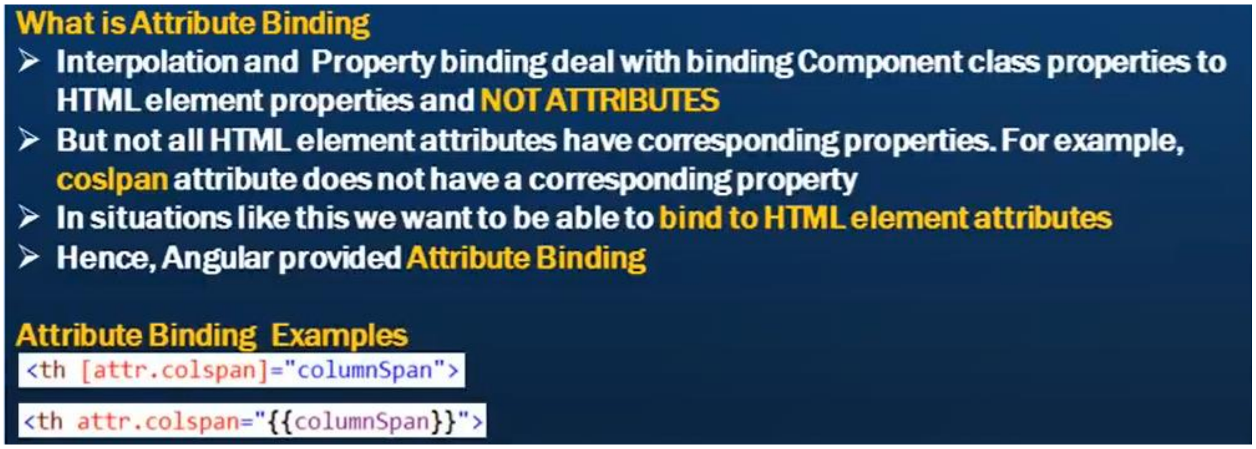
- Interpolation is a special syntax that Angular converts into a property binding
- To concatenate strings we must use interpolation instead of property binding

```
<img src='http://www.pragimtech.com/{{imagePath}}' />
```

- To set an element property to a non-string data value, you must use property binding

```
<button [disabled]='isDisabled'>Click me</button>
```

Attribute Binding :-



What is Attribute Binding

- Interpolation and Property binding deal with binding Component class properties to HTML element properties and NOT ATTRIBUTES
- But not all HTML element attributes have corresponding properties. For example, coslpan attribute does not have a corresponding property
- In situations like this we want to be able to bind to HTML element attributes
- Hence, Angular provided Attribute Binding

Attribute Binding Examples

```
<th [attr.colspan]="columnSpan">
```

```
<th attr.colspan="{{columnSpan}}">
```