Learning and Knowledge Management

Angular Components

Copyright © 2020 Accenture. All rights reserved.

Module Objectives

At the end of this module, you should be able to understand:

- Component Basics
- Interpolation in Angular
- Data Binding in Angular(OneWay and TwoWay)
- Nesting of Components

Copyright © 2020 Accenture. All rights reserved.

Topic List

#No	Module Topics
1	Component Basics
2	Data Binding
3	Nesting of Components

Copyright © 2020 Accenture. All rights reserved.

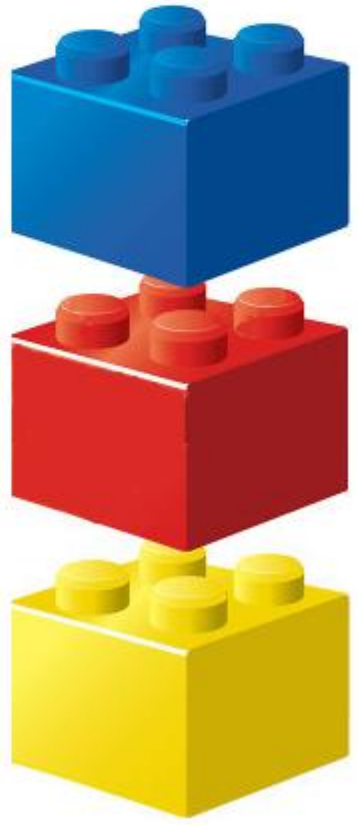
Component Basics

Copyright © 2020 Accenture. All rights reserved.

Components

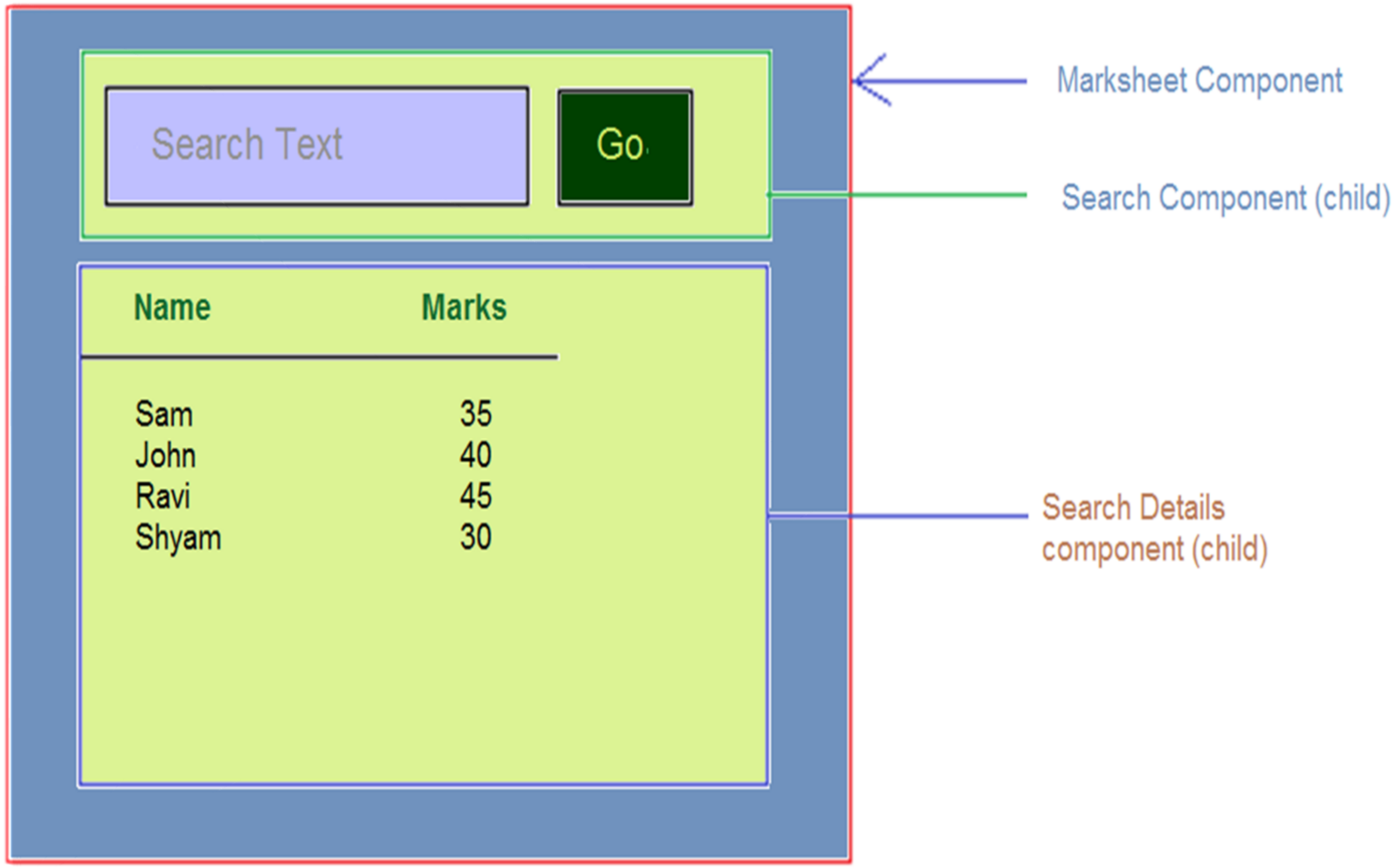
Component Basics

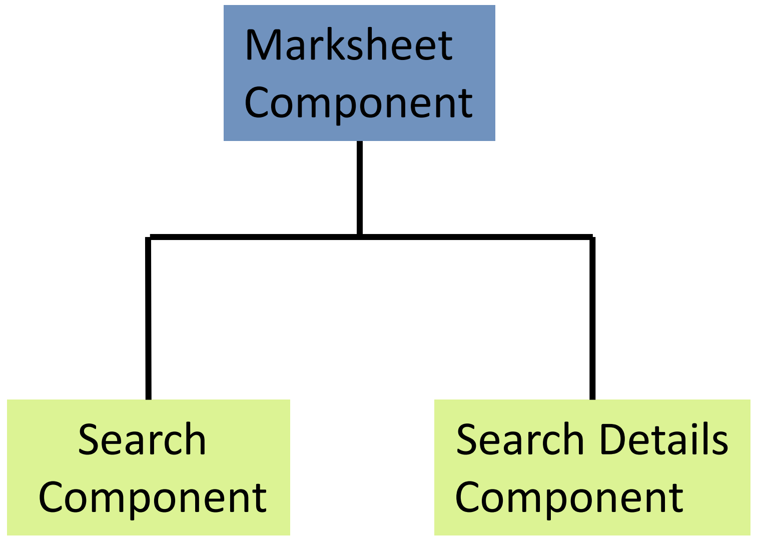
- A component controls a patch of screen real estate that we could call a view and declares reusable UI building blocks for an application
- The core concept of any Angular application is the component
- The whole application can be modeled as a tree of these components



Components

Component Basics: Components Hierarchy





3

Components

Component Basics: Creating component

- To create a Component in Angular using Angular CLI

  ng g component <new-component-name>

- Example: ng g component Product
- **Note:** A folder by name 'product' is created
- This folder product contains the following four files

File Name	Description
product.component.ts	TypeScript Class for product Component
product.component.html	Template (View) of product Component
product.component.css	Styling for product Component
product.component.spec.ts	Test specification for product Component

Copyright © 2020 Accenture. All rights reserved.

Components

Component Basics: Creating component

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

@Component({
        selector: 'app-root',
        templateUrl: './app.component.html',
        styleUrls: ['./app.component.css']
})
export class AppComponent {
        pageHeader:String = ‘Home Pagw';
}
```

```
<div class=“header”>
        <h1> {{PageHeader}} </h1>
</div>
```

```
import { BrowserModule } from '@angular/platform-browser’;
import { NgModule } from '@angular/core’;
import { AppComponent } from './app.component’;

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

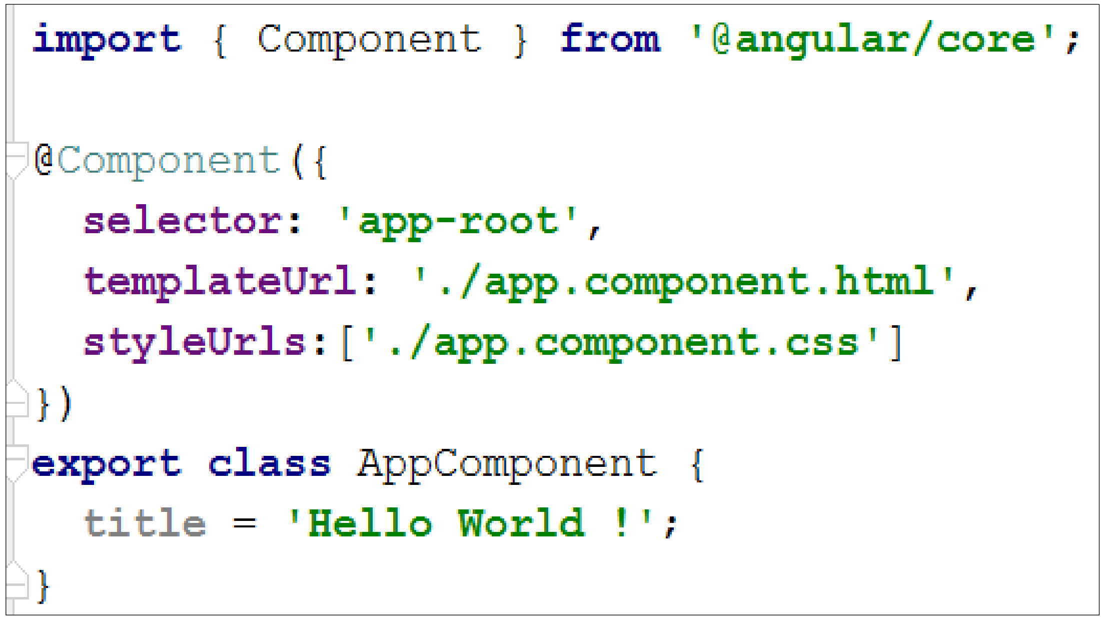
3

Copyright © 2020 Accenture. All rights reserved.

Components

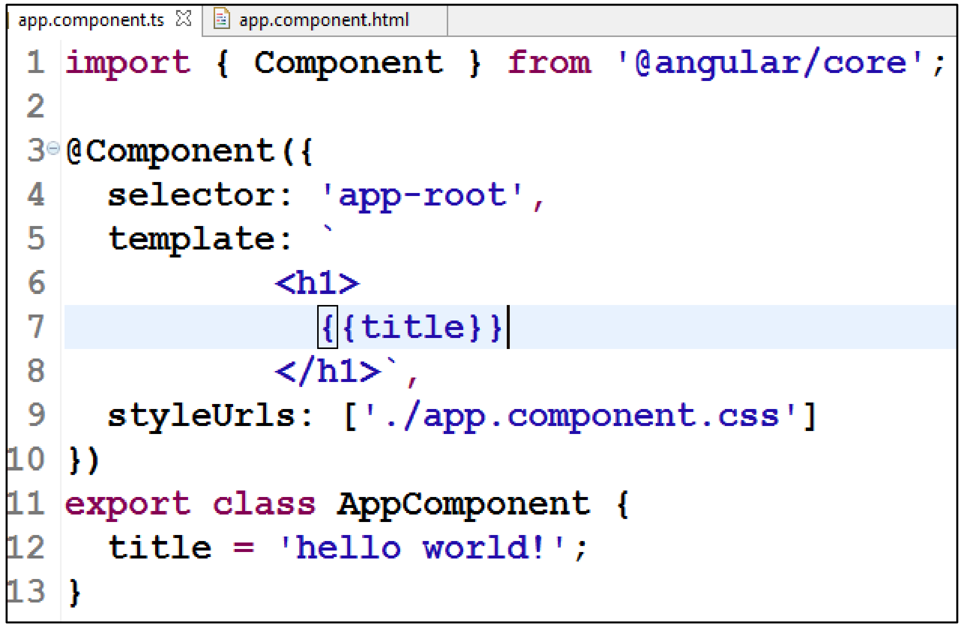
Component Basics: Creating component

- We define a Component's view with a template
- A template is a form of HTML that tells Angular how to render the Component



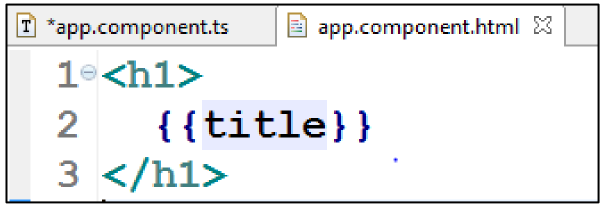
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls:['./app.component.css']
})
export class AppComponent {
  title = 'Hello World !';
}
```



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

@Component({
  selector: 'app-root',
  template: `
          <h1>
            {{title}}
          </h1>`,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'hello world!';
}
```



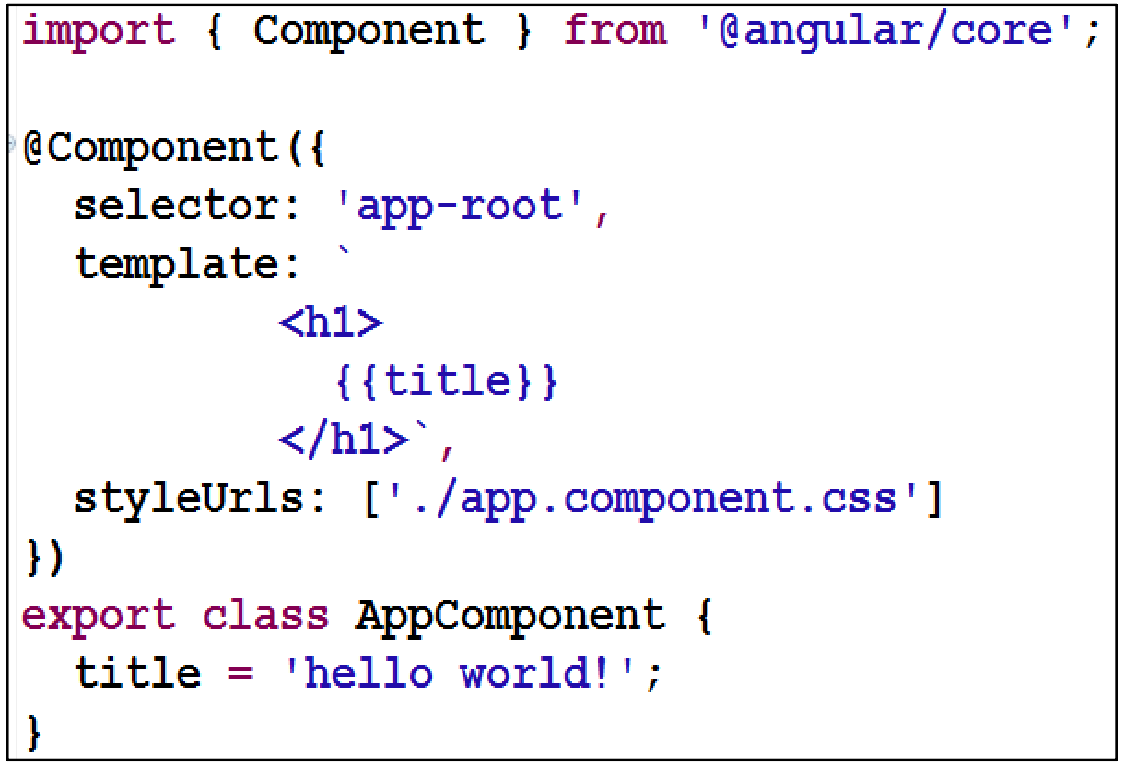
```
<h1>
  {{title}}
</h1>
```

3

Components

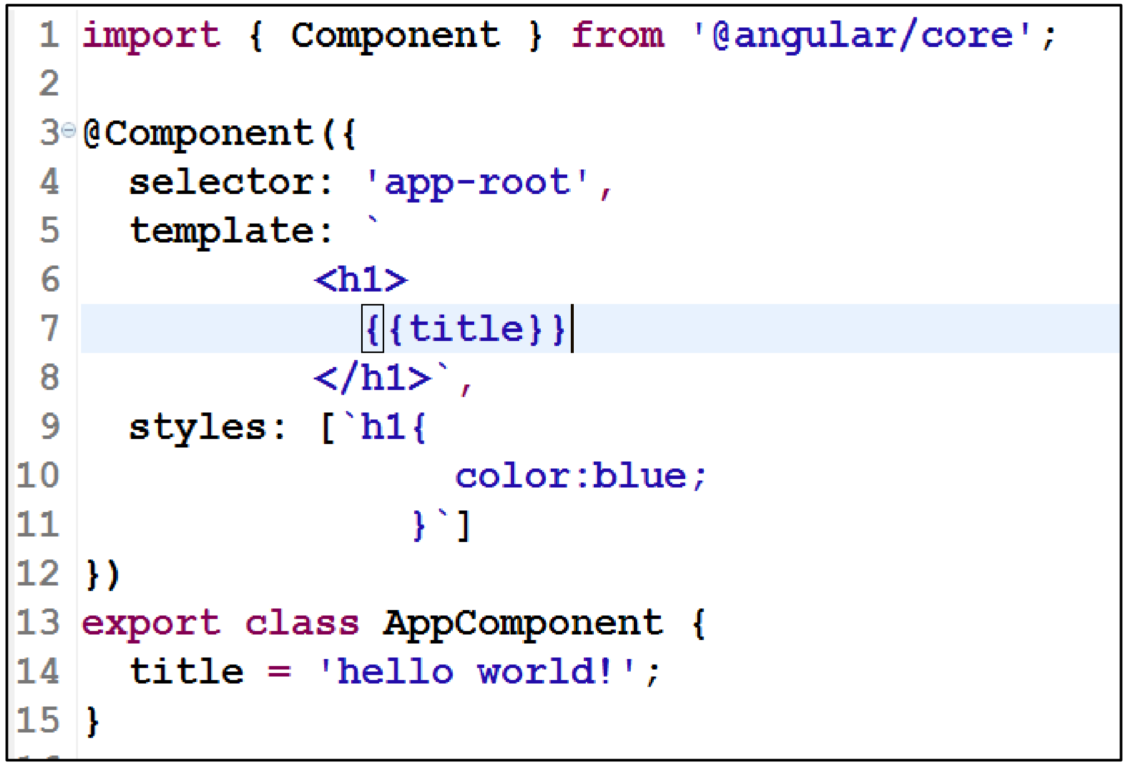
Component Basics: Creating component

- Component's style is defined using **styles** or **styleUrls**



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

@Component({
  selector: 'app-root',
  template: `
          <h1>
            {{title}}
          </h1>`,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'hello world!';
}
```

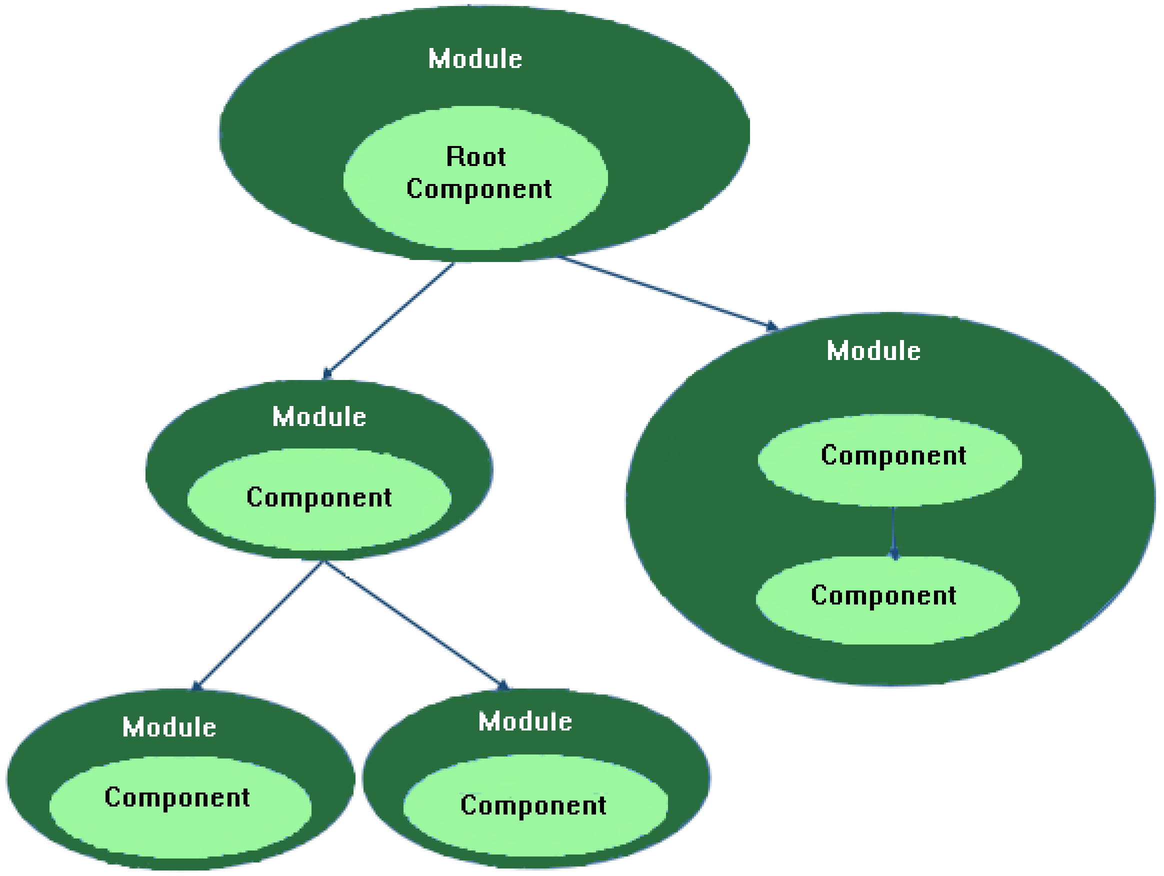


```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
          <h1>
            {{title}}
          </h1>`,
  styles: [`h1{
              color:blue;
            }`]
})
export class AppComponent {
  title = 'hello world!';
}
```

Components

Component Basics: Component Hierarchy



3

Components

Component Basics: Component Hierarchy

product.component.ts

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

@Component({
        selector: 'product-details',
        templateUrl: './product.component.html',
        styleUrls: ['./product.component.css']
})
export class ProductComponent implements OnInit {
        productId:String="P1001";
        productName:String = "IPhone";
        productPrice:Number = 2345.55;
}
```

```
<div class="header">
        <h1> {{PageHeader}} </h1>
        <product-details></product-details>
</div>
```

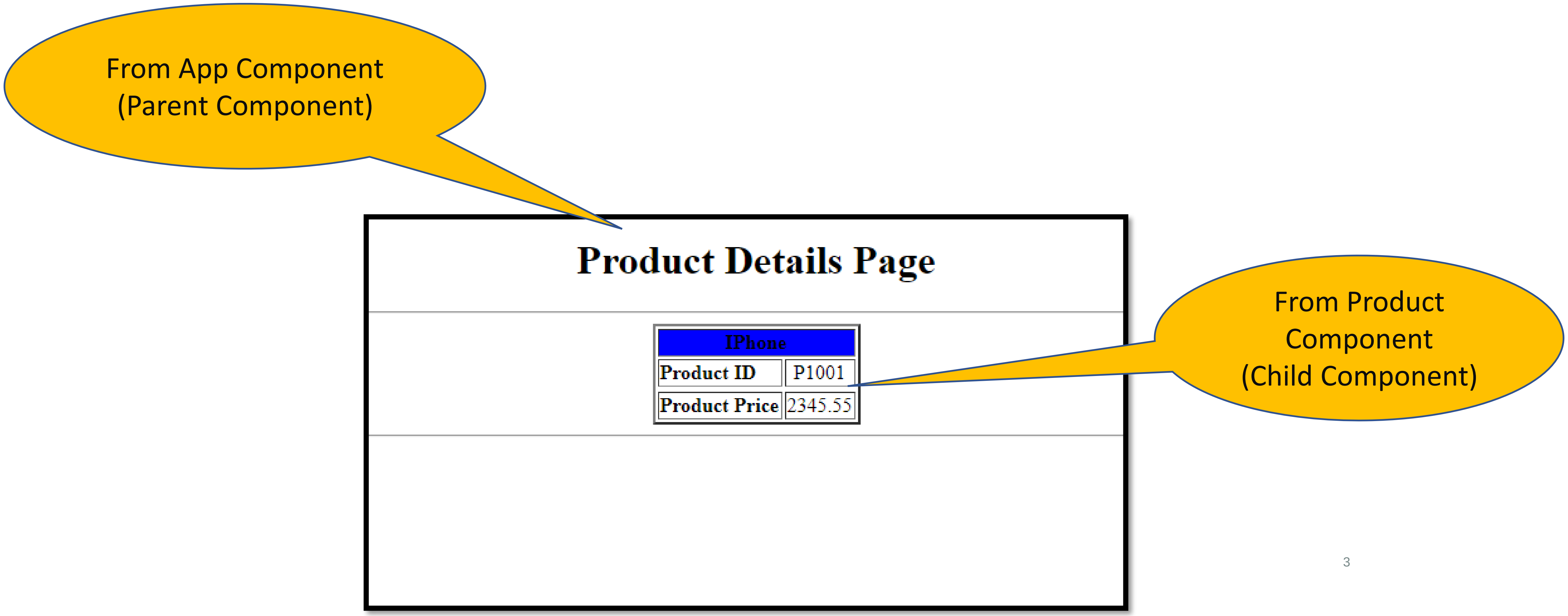
```
<table border=2px solid>
        <thead>
        <td colspan="2" align="center" bgColor="blue">
                <b>{{productName}}</b>
        </td>
        </thead>
        <tbody>
        <tr>
                <td><b>Product ID</b></td>
                <td align="center"> {{productId}} </td>
        </tr>
        <tr>
                <td><b>Product Price</b></td>
                <td align="center"> {{productPrice}} </td>
        </tr>
        </tbody>
</table>
```

Interpolation Operator {{ }}

Copyright © 2020 Accenture. All rights reserved.

Components

Component Basics: Component Hierarchy



Product Details Page

IPhone	
Product ID	P1001
Product Price	2345.55

3

Data Binding

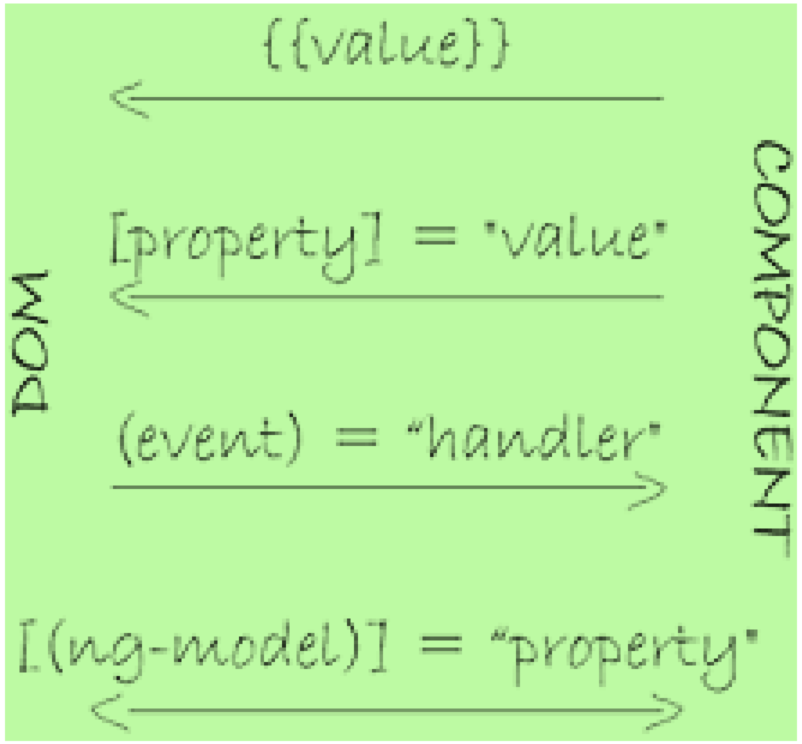
Copyright © 2020 Accenture. All rights reserved.

Components

Data Binding

- Data binding in Angular apps allows the automatic synchronization of data between the model and view

Data Binding	Description
Interpolation	Component to View
Property Binding	Component to View
Event Binding	View to Component
Two-way Data Binding	Bidirectional From Component to View and View to Component



Components

Data Binding: One-Way Binding using Interpolation

- One way binding from a Component class property to an element
- Use curly brackets {{ }} for template expression
- **Example**: {{propertyName}}

```
import { Component, OnInit } from '@angular/core';
@Component({
        selector: 'product-details',
        templateUrl: './product.component.html',
        styleUrls: ['./product.component.css']
})
export class ProductComponent  {

        productId:String="P1001";
        productName:String = "IPhone";
        productPrice:Number = 2345.55;

}
```

```
Product Name : {{productName}}
Product ID : {{productId}}
Product Price : {{productPrice}}
```

Components

Data Binding: Property Binding

- Property binding is another way of one-way data binding from Component to View Template
- It is done by enclosing the property by square brackets and assigning the component's property to it

product.component.html

```
<div [align]="position">
        <h1> {{pageHeader}} </h1>
</div>
```

product.component.ts

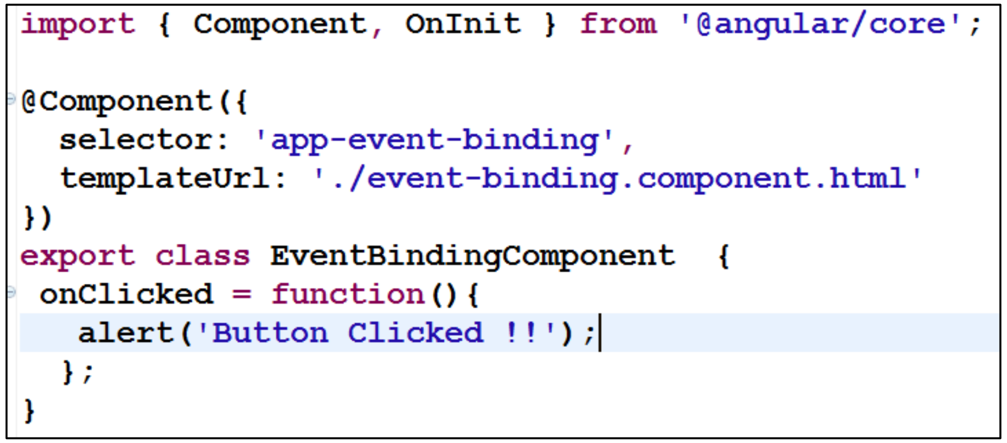
```
import { Component } from '@angular/core';

@Component({
        selector: 'app-produt',
        templateUrl: './product.component.html',
        styleUrls: ['./product.component.css']
})
export class ProductComponent {
        pageHeader:string = 'Product Details Page';
        position:string ="left";
}
```

Components

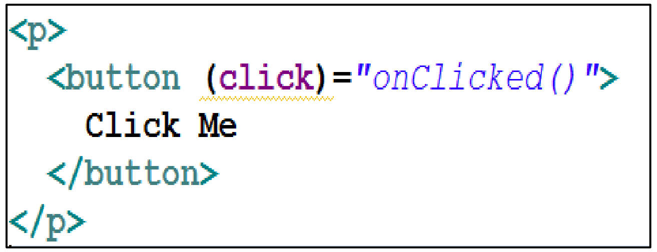
Data Binding: Event Binding

- Event binding is a data binding from a DOM element to the component
- Components listen to user actions using an event binding



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

@Component({
  selector: 'app-event-binding',
  templateUrl: './event-binding.component.html'
})
export class EventBindingComponent  {
 onClicked = function(){
   alert('Button Clicked !!');
  };
}
```



```
<p>
  <button (click)="onClicked()">
    Click Me
  </button>
</p>
```

Components

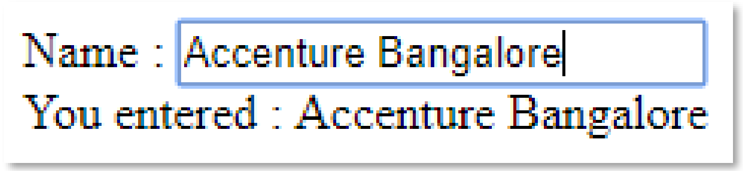
Data Binding: Two-Way Data Binding

- Two Way data binding is the ability to flow data in both the directions
- From component to View Template and from View template to Component

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

@Component({
        selector: 'app-root',
        templateUrl: './app.component.html',
        styleUrls: ['./app.component.css']
})
export class AppComponent {
        companyName:string="Accenture";
}
```

```
Company Name : <input [(ngModel)]=‘companyName'>
<br>
You entered : {{companyName}}
```



```
Name : Accenture Bangalore
You entered : Accenture Bangalore
```

Components

Data Binding: Passing data to the component

- The **@Input()** decorator defines a set of parameters that can be passed from the component's parent

hello.component.html

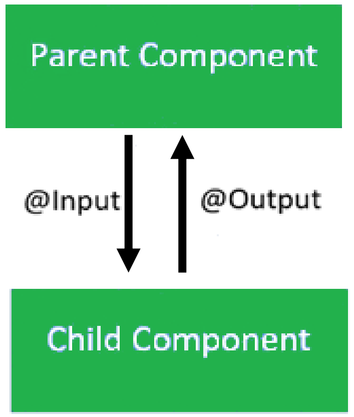
```
 import { Component, Input } from '@angular/core';

@Component({
  selector: ‘app-hello',
  template: '<p>Hello, {{name}}!</p>',
})
export class HelloComponent {
  @Input() name: string;
}
```

hello.component.html

```
<!-- To bind to a raw string -->
<app-hello name="World"></app-hello>

<!-- To bind to a variable in the parent scope -->
<app-hello [name]="helloName"></app-hello>
```



Module Summary

Now, you should be able to understand:

- Angular Modules
- Decorators
- Custom Modules

Copyright © 2020 Accenture. All rights reserved.

Thank You

Copyright © 2020 Accenture. All rights reserved.