Created with an evaluation copy of Aspose.Words. To remove all limitations, you can use Free Temporary License https://products.aspose.com/words/temporary-license/

Angular

How Angular project is executed?

When we run an Angular project, the main.ts file gets executed. From main.ts the Anguler comes to know about the AppModule. From here, it goes to app.module.ts file. There, it comes to know about AppComponent. SO it goes to app.component.ts file. From here, it comes to know about the app-root which is in the selector property of the component and also know about the view from the templateUrl property. After this the index.html file will be loaded in the browser where the <app-root></app-root> in index.html will be replaced by the content in the template provided by app component.

When we start the application by hitting the ng-serve command, it rebuilds the project then it will create javascript bundles automatically to the index.html file. So, the script files are injected into the index.html by the angular cli.

Component

Component is a TypeScript class decorated with @Component decorator and it contains methods & properties which can use in html. The AppComponent (Root Component)is the main component in any Angular application. The root component can have several nested components and at the end we will have a complete application consisting of different components.

We can have components for header, navbar, main body, sidebar, footer,etc. We can also have components for home, about, contact present in the navbar.

ASPOSE Your File Format APIs

```
@Component({
selector: 'app-component-overview',
templateUrl: './component-overview.component.html',
styleUrls: ['./component-overview.component.css'] })
export class AppComponet{
title=' AngularApp' ;
message =  'Angular is the app of apps' ;
}
```

So a component has :

A Class, that contains the code required for the view template i.e. contains the UI logic

A View template, defines the user interface. It contains the Html, directives and data binding.

A decorator, it adds meta data to the class, making it a component.

In order to use a component, we need to export it, and register it in app module.

Evaluation Only. Created with Aspose.Words. Copyright 2003-2024 Aspose Pty Ltd.

We can use the value in selector property as an html tag.

Template property of Component:

With template property, we can write html in the component file. It won't need the html file.

```
@Component({
selector: 'app-component-overview',
template: '<div><p>This web takes cookies</p></div>',
styleUrls: ['./component-overview.component.css'] })
export class AppComponet{
title=' AngularApp' ;
message = 'Angular is the app of apps' ;
}
```

In templateUrl property, we specify the path to a html file to use as a view file. While for template property we specify the html inside a string. For multiline html code using template property, user``(tactiks). Its good practice to write the html code in separate file, if we have 3 or more lines of html code.

Disadvantage: We know about errors in html at runtime only. We are mixing typescript code with html code.

Styles property of Component:

With style property, we can add css style in the component file. No need for css file.

```
@Component({
selector: 'app-component-overview',
template: '<div><p>This web takes cookies</p></div>',
styles: [ "div{margin: 10px 0px; padding: 10px 20px;}" ,
" p{font-size:40px}" ]})
export class AppComponet{
title=' AngularApp' ;
message = 'Angular is the app of apps' ;
}
```

It is similar to template property and has similar disadvantages.

Evaluation Only. Created with Aspose.Words. Copyright 2003-2024 Aspose Pty Ltd.

Selectors in Angular:

There are different selector in Angular:

1. We can use a selector like an **HTML tag**.
   We do so by using this syntax.
   Selector : ‘app-nav’,
   Example: <app-nav></app-nav>
2. We can use a selector like a **HTML attribute**.
   We do so by using this syntax: Write the value between []
   Selector : ‘[app-nav]’,
   Example: <div app-nav></div>
3. We can use a selector like a **CSS class**.
   We can do so by using this syntax: Add a dot(.) before the value.
   Selector: ‘.app-nav’,
   Example : <div class=”app-nav”></div>

Data binding:

It allows us to communicate between a component class and its corresponding view template or vice versa.

ASPOSE
Your File Format APIs

One-way data binding:

It is when we can access a component class property in its corresponding view template Or when we can access a value form view template in corresponding component class property. I.E only in one direction.

Two-way data binding:

Binds data from component to view template and view template to component class. This is a combination of property binding and event binding i.e. change in component will be seen in view and change in view will be seen in component. In Both Directions.

Property binding is done by [].

Event binding is done by ().

So for two way binding, [()]. This is banana box syntax. Then we use a directive called ngModel and assign this property. To use ngModel you need to import FormModules.

<input type=”text” [(ngModel)] = “searchValue”>

searchValue is a property in the component/

String Interpolation:

Used to bind data from component class to view template. I.E. data flows from component to view. It is used to achieve one way data binding. Using string interpolation we can use property or methods of component class.

Evaluation Only. Created with Aspose.Words. Copyright 2003-2024 Aspose Pty Ltd.

We use {{}} to perform string interpolation in html file. Inside {{}}, we can write any typescript code. We can also call typescript method using string interpolation. Example: <h1> {{ getName() }} </h2>

We use Angular to dynamically render data in the webpage. When we write static content in html, angular isn't need. Also, if the same static data is used in multiple parts, then it is prone to have issue like not rendering consistent data. To solve this, we can create a property in th component where we can define the data we want to render.

Inside component:

export class HeaderComponent{

    slogan: string = "Hello world."

}

In Html:

<h1>{{slogan}}</h1>

We using slogan property in the component to render the value in the view dynamically using data binding.

Property binding:

With property binding, we are binding the properties of html elements with the property or method ofc component class.

In component class,

export class HeaderComponent{

    source : string = "/assets/shopping.jpg';

}

In Html,

<img [src]="source" height="240" width="320">

Here data flows from component to view.

Event binding:

Binds webpage events to a components property or method. Using event binging we can pass data from view to component.

In view,

```
<div class="search-div">
    <span><b>Search:</b></span>
    <input type="text" (input)="changeSearchValue($event)">
</div>
```
The $event stores all the event data related to the particular event.

ASPOSE
Your File Format APIs

Evaluation Only. Created with Aspose.Words. Copyright 2003-2024 Aspose Pty Ltd.

In Component,

```
@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.scss']
})
export class SearchComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

  searchValue :string = ""

  changeSearchValue(eventData : Event){
    this.searchValue = (<HTMLInputElement>eventData.target).value
  }
}
```

ASPOSE
Your File Format APIs

So whenever user enter the value in the search box, the input event will happen. And when the input methods happens, it'll call the changeSearchValue($event) method. Then we pass the event data to the method. From the event data, we get the user provided value.

Directives:

Directives are simply an instruction to the DOM. Used to tell Dom what to add to the webpage and what not. Also tell DOM how to render html elements.

Components:

Are kind of one such instruction in DOM. Using component we tell DOM what to add to the webpage. We can use directive as a css class or html element. But we mostly use directives as html attribute.

Custom directive:

```
@Directive({
  Selector: '[changeDivGreen]'
})
Export class ChangeDivGreen{
}
```

Types of directives:

Evaluation Only. Created with Aspose.Words. Copyright 2003-2024 Aspose Pty Ltd.

1. **Structural directives:**

   Changes the view of a webpages by adding or removing DOM elements from a webpage.

   Note: Before using any structural derivative in Angular, we need to star with *. Example: *ngFor, *ngIf,*ngSwitch

2. **Attribute directives:**

   Used like an attribute on a existing webpage element to change its look and behaviour. Example: [ngStyle], [ngClass],[ngModel]

Built-in Directives:

1. **ngFor:**
   It is a structural directive. That means, ngFor manipulates the DOM by adding or removing elements from the DOM. It is used to repeat a portion of HTML template once per each item from an itterable list.

```
@Component({
  selector: 'app-products',
  templateUrl: './products.component.html',
  styleUrls: ['./products.component.scss']
})
export class ProductsComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

  products = [
    {id:1, name:"Watch",price:"1200",color:'black',available:'available'},
    {id:2, name:"TV",price:"100000",color:'black',available:'not available'},
    {id:3, name:"Phone",price:"120000",color:'blue',available:'available'},
    {id:4, name:"Laptop",price:"87000",color:'midnight black',available:'not 
available'},
    {id:5, name:"Book",price:"1450",color:'black',available:'available'}
  ]

}
```

we can assign multiple variable in ngFor like below.

```
<div *ngFor="let item of products; let i = index">
    <p>{{i}}</p>
    <div class="product-container">
        <div>
            <div class="name-container">
```

Evaluation Only. Created with Aspose.Words. Copyright 2003-2024 Aspose Pty Ltd.

```
                <h6>{{item.name}}</h6>
            </div>
            <div class="detailContainer">
                <div class = "details">{{item.available}}</div>
                <div class = "details"><b>Price:</b>{{item.price}}</div>
                <div class = "details"><b>Color:</b>{{item.color}}</div>
            </div>

        </div>
        <div class="options">
            <button class="btn btn-success">Show Details</button>
            <button class="btn btn-warning">Buy Now</button>
        </div>
    </div>
</div>
```

2. **ngStyle:**

   It is an attribute directive. ngStyle changes the look and behavior of an HTML element. The ngStyle directive is used to set a CSS style dynamically for an HTML element based on a given typescript expression.

```
<div class="detailContainer">
                <div class = "details" [ngStyle]="{color: item.available
==='available' ? 'Green' : 'Red'}">{{item.available}}</div>
                <div class = "details"><b>Price:</b>{{item.price}}</div>
                <div class = "details"><b>Color:</b>{{item.color}}</div>
            </div>
```

   ngStyle is used as html attribute to change the color of available text. We put ngStyle within [] for property binding with the value in component. Here, text color is set up dynamically.

3. **ngIf:**

   It is an structural directive. ngIf is used to add or remove element from a webpage based on a given condition. If the condition assigned to ngIf returns true, it will add the element on which it is used to the webpage. Otherwise, if the condition returns false, it will remove that element from the webpage.

```
<div class="search-div">
    <span><b>Search:</b></span>
    <!-- <input type="text" (input)="changeSearchValue($event)"> -->
```

This document was truncated here because it was created in the Evaluation Mode.

Evaluation Only. Created with Aspose.Words. Copyright 2003-2024 Aspose Pty Ltd.