Angular Directive Notes

Angular Directive is a class with a **@Directive** decorator. **Decorators** are functions that modify JavaScript classes. The **decorators** are used for attaching metadata to classes as it knows the configuration of those classes and how they should work. You will be surprised to know that the component is also a directive-with-a-template. The **@Component *decorator*** is a **@Directive** decorator extended with template-oriented features.

Angular Directive Example

An **Angular Directive** can be divided into main two types but consider component also:

- **Components**
- **Structural**
- **Attribute**

#Structural Directives

Structural directives reconstruct the layout by adding, removing, and replacing elements in **DOM**. The **structural directives** are responsible for shape or reshape the DOM's structure, typically by adding, deleting, or modifying elements. Similar to the other directives, you apply the structural Directive to a *host* element. The Directive then performs whatever it is intended to do with that host element. The structural directives are straightforward to recognize. An **asterisk (*)** precedes the directive attribute name. It does not require brackets or parentheses like attribute directive.

Let us take an example. Generally, we loop through data from the backend and display inside the table. It is the general case scenario in the web application. At that time, we use the ***ngFor Directive** to loop through the data.

***ngFor Directive** Example:

```
<table class="table table-hover">
 <thead>
 <tr>
    <td>Ad Unit Name</td>
```

Angular Directive Notes

```
    <td>Ad Unit Price</td>
    <td colspan="2">Actions</td>
  </tr>
  </thead>

  <tbody>
    <tr *ngFor="let adunit of adunits">
      <td>{{ adunit.unit_name }}</td>
      <td>{{ adunit.unit_price }}</td>
    </tr>
  </tbody>
</table>
```

So, in the above example, we have a loop through the **ad units** and display the ad unit one by one. Three of the standard, built-in structural directives are **NgIf**, **NgFor**, and **NgSwitch**. The Directives can be written in both the **UpperCamelCase** and **lowerCamelCase** because **NgIf** refers to a directive class & **ngIf** refers to a directive's **attribute** name.

Conditional rendering means elements are inserted into the DOM only when a condition is meet.

***ngIf Directive**

Angular provides the *ngIf directive which allows you to render elements conditionally in your Angular templates.

Let's see this with a simple example.

Open the src/app/app.component.ts file in your project define a new class variable called displayElement and gives it an initial value of false:

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

@Component({

  selector: 'app-root',
```

Angular Directive Notes

```
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  displayElement = false;
}
```

Next, open the src/app/app.component.html file and update it as follows:

```
<div>
   <p *ngIf="displayElement">Magic element!</p>
</div>
```

In this case, the <p> element and its contents will not be rendered in the DOM because we applied the *ngIf directive with a false value

If you go back to your component's class and assign a true value to the displayElement variable the element will be rendered.

The Else block

Just like typical programming languages the *ngIf directive can have an else block which is shown if the statement defined in the main block is false.

Go back to the src/app/app.component.html file and update it as follows:

```
<div>
   <!-- notActive is a reference to else -->
   <p *ngIf="displayElement; else showThis">Magic element!</p>
   <ng-template #showThis>
      Another magic element!
   </ng-template>
```

Angular Directive Notes

Here, we used the else show This with the *ngIf directive to provide a partial template that will be rendered instead if the <p> element.

The else block has to be an ng-template.

The Angular template is referenced using a template reference that we've called show This.

Another example

Open the src/app/app.component.html file and replace the contents with the following code:

<input [(ngModel)]="showContent" type="checkbox"/> Show My Secret Message

<hr />

<div *ngIf="showContent; else message">

Hello Angular 9!

<ng-template #message>

Click the checkbox above to read the secret message!

</ng-template>

Example will help you ng switch condition in angular. Here you will learn *ngSwitch condition in angular example.

ngSwitch directive provide you to write simple switch case of javascript. you can see bellow simple example that will help you more:

src/app/app.component.ts

Angular Directive Notes

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


@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: [ './app.component.css' ]
})
export class AppComponent  {


  mySwitchVar = "one";


}
```

src/app/app.component.html

```
<h1>Angular NgSwitch Directive Example - Chandu Full Stack Training @91-9866037742</h1>
<div [ngSwitch]="mySwitchVar">
   <div *ngSwitchCase="'one'">
     <p>Switch Case One</p>
   </div>
   <div *ngSwitchCase="'two'">
     <p>Switch Case Two</p>
   </div>
```

Angular Directive Notes

```
  <div *ngSwitchCase="'three'">
    <p>Switch Case Three</p>
  </div>
  <div *ngSwitchDefault>
    <p>Switch Case Default</p>
  </div>
</div>
```


#Attribute Directives

An **Attribute** directive changes the appearance or behavior of a **Document Object Model(DOM)** element. The attribute directives are used as attributes of elements. The built-in NgStyle Directive in the Template Syntax guide, for example, can change several element styles at the same time.

You can easily use ng style in angular 6, angular 7, angular 8 and angular 9 example.

*ngStyle allows you to write css on element. In this example i will give you two example one will be simple add css using object and another example of adding css using component function.

Let's see bellow example angular ngStyle directive example.

Example 1: ngStyle with a Object

src/app/app.component.html

```
<h1>Angular ngStyle Directive Example - Chandu Java Full Stack @91-9866037742</h1>
  <button [ngStyle]="{background: 'red'}">Click Me!</button>
```

Example 2: ngStyle with a Function

src/app/app.component.html

```
<h1>Angular ngStyle Directive Example - Chandu Java Full Stack @91-9866037742</h1>
  <button [ngStyle]="addButtonStyles()">Click Me!</button>
```

src/app/app.component.ts

Angular Directive Notes

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


@Component({
 selector: 'my-app',
 templateUrl: './app.component.html',
 styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
 addButtonStyles(){
  return {
    'background': 'blue',
    'padding': '12px'
  }
 }
}
```

example of how to use ngmodel with input field in form with angular 6, angular 7, angular 8 and angular 9 application.

*ngModel will help to bind input fiel. you can create form control instance using ngModel. ngModel will help to access form field value, status and error status.

Angular Directive Notes

Here, i will give you two example so you can understand how to use ng model in angular and what is ng model in angular.

1) NgModel Simple Example

2) NgModel with Form Example

1) NgModel Simple Example

Before we use ng model, we must need to import "FormsModule" from '@angular/forms'; in module.ts file as bellow:

src/app/app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
 @NgModule({
 imports:      [ BrowserModule, FormsModule ],
 declarations: [ AppComponent],
 bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

src/app/app.component.ts

```
import { Component } from '@angular/core';
 @Component({
 selector: 'my-app',
```

Angular Directive Notes

```
 templateUrl: './app.component.html',
 styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
 name: string = 'Hardik';

 setValueName() {
  this.name = 'Savani';
  }
}
```

src/app/app.component.html

```
<h1>Angular NgModel Example - Chandu Java Full Stack @91-9866037742</h1>
 <input [(ngModel)]="name" #ctrlName="ngModel" required>
 <p>Value: {{ name }}</p>
<p>For Validation: {{ ctrlName.valid }}</p>
<button (click)="setValueName()">Set Value!</button>
```

2) NgModel with Form Example

Before we use ng model, we must need to import "FormsModule" from '@angular/forms'; in module.ts file as bellow:

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

Angular Directive Notes

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

src/app/app.component.ts

```
import { Component } from '@angular/core';
import {NgForm} from '@angular/forms';
 @Component({
 selector: 'my-app',
 templateUrl: './app.component.html',
 styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
   onSubmit(myForm: NgForm) {
    console.log(myForm.value);
     console.log(myForm.valid);
   }
}
```

src/app/app.component.html

```
<h1>Angular NgModel Example - Chandu Java Full Stack @91-9866037742</h1>
 <form #myForm="ngForm" (ngSubmit)="onSubmit(myForm)" novalidate>
```

Angular Directive Notes

```
  <input name="name" ngModel required #name="ngModel">
  <input name="email" ngModel required #email="ngModel">
  <button>Submit</button>
</form>
 <p>Name Field Value: {{ name.value }}</p>
<p>Name Field Is Valid?: {{ name.valid }}</p>

<p>Email Field Value: {{ email.value }}</p>
<p>Email Field is Valid?: {{ email.valid }}</p>
<p>Form value: {{ myForm.value | json }}</p>
<p>Form valid: {{ myForm.valid }}</p>
```

***ngClass**

let's see example of angular ng class conditional example. you will learn ngclass with condition angular. you can see ngclass with condition in angular. we will help you to give example of *ngClass condition in angular example. You just need to some step to done angular ngClass directive example.

Example 1: ngClass with a String

In this example we can simply use ngClass and add two calss "btn" and "btn-success". you don't need to use any logic.

src/app/app.component.html

```
<h1>Angular ngClass Directive Example - Chandu Java Full Stack @91-9866037742</h1>
 <button [ngClass]="'btn btn-success'">Click Me!</button>
```

Example 2: ngClass with a Array

Here, we will add class array with ngClass.

Angular Directive Notes

src/app/app.component.html

```
<h1>Angular ngClass Directive Example - Chandu Java Full Stack @91-9866037742</h1>
  <button [ngClass]="['btn', 'btn-success']">Click Me!</button>
```

Example 3: ngClass with a Object

Here, we will add class object with ngClass.

src/app/app.component.html

```
<h1>Angular ngClass Directive Example - Chandu Java Full Stack @91-9866037742</h1>
  <button [ngClass]="{'btn': isButtonClass, 'btn-success': true}">Click Me!</button>
```

src/app/app.component.ts

```
import { Component } from '@angular/core';
  @Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
  isButtonClass = true;
}
```

ng-content with ng-container. we will see simple example of angular ng-container. It's very simple to use with angular 6, angular 7, angular 8 and angular 8 for ng-content. There is a another solution for reusable components using ng-content and ng-container. ng-content and ng-container will help you to create reusable components with your angular application. you can pass data dynamically in your components.

Angular Directive Notes

In this example, we will simple create my-card component and we will use ng-content with ng-container. you can also use id or dom element. we will use bootstrap 4 cards with title, content and footer text. we will call our component every where with passing those parameters dynamically. So you can see bellow example with layout.

Let's follow bellow step to create simple example of ng-content in angular application.

Step 1: Create New App

You can easily create your angular app using bellow command:

```
ng new appNgContent
```

Step 2: Create New Component

Here, we will just create new my-card component and use bootstrap card. we also update view file.

```
ng g component my-card
```

Let's update code of my-card.component.ts file.

src/app/my-card/my-card.component.ts

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

 @Component({

 selector: 'my-card',

 templateUrl: './my-card.component.html',

 styleUrls: ['./my-card.component.css']

})

export class MyCardComponent implements OnInit {

   constructor() { }

  ngOnInit() {
```

Angular Directive Notes

```
 }
}
```

Here, we will use bootstrap 4 cards layout. Now let's just update html view file. as bellow:

src/app/my-card/my-card.component.html

```
<div class="card">
   <div class="card-header">
      <ng-content select=".header"></ng-content>
   </div>
   <div class="card-body">
      <ng-content select=".content"></ng-content>
   </div>
   <div class="card-footer">
      <ng-content select=".footer"></ng-content>
   </div>
</div>
```

Step 3: Use Component

Now use can see how we can call our component with dynamic data.

src/app/app.component.html

```
<my-card>
  <ng-container class="header">
    Angular ng-content and ng-container example
  </ng-container>
```

Angular Directive Notes

<ng-container class="content">

Let's talk about what is ng-content in angular? and how to use ng-content in angular?, here we will learn example of angular ng-content select by class. we will see simple....

</ng-container>

<ng-container class="footer">

Tutorial by ItSolutionStuff.com

</ng-container>

</my-card>

Now we are ready to run our example, you can run by following command:

ng serve