**Angular ngIf Directive**

**Angular ngIf Directive:**

The ngIf is a structural directive and it is used to add or removes the HTML element and its descendant elements from the DOM layout at runtime conditionally. That means it conditionally shows the inline template.

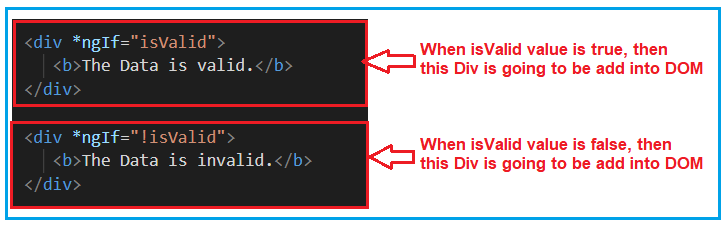
The ngIf directive works on the basis of a boolean true and false results of a given expression. If the condition is true, the elements will be added into the DOM layout otherwise they simply removed from the DOM layout.

The basic syntax of the ngIf directive is very simple, all we need to do is prefix the directive name with an asterisk (\*) as shown below and add it anywhere inside your template. Here if the given “expression” result is false or null, then the elements HTML element and its descendant elements will not be added to the DOM.

**\*ngIf = “expression”**

**Example to understand Angular ngIf Directive:**

Please have a look at the following image. As you can, here we have two div Element and we are using the ngIf directive to add or remove the div from the DOM. If the isValid (this is a variable defined in the component class) value is true, then the first div is going to be added into the DOM and if the value is false, then the second div is going to be added into the DOM.



So, first, modify the **app.component.html** file as shown below.

**<div** \*ngIf="isValid"**>**

**<b>**The Data is valid.**</b>**

**</div>**

**<div** \*ngIf="!isValid"**>**

**<b>**The Data is invalid.**</b>**

**</div>**

Then modify the **app.component.ts** file as shown below.

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

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

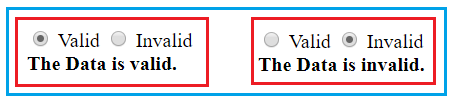
isValid: **boolean** = **true**;

**}**

At this moment, change the value of isValid Boolean property (true and false) and check the output in the browser and based on the value, the respective div will be added into the DOM and that you can in the webpage.

**Another Example For Better Understanding:**

Let us enhance the same example. Now, instead of hardcoding the isValid Boolean property value, we will set this value from the radio buttons. We will create two radio buttons and based on the selected radio button value we will set the IsValid property value. For better understanding please have a look at the below image.



When the Valid radio button is checked, we need to add the first Div to the DOM and when the InValid radio button is checked we need to add the second Div into the DOM. Let’s see how to implement this.

**Modify the app.component.ts file:**

Please modify the app.component.ts file as shown below. By default, we set the Boolean isvalid property to true. We have also created one method i.e. ChangeData which accepts a Boolean parameter (in this case valid) and then it set the isValid property value with the incoming parameter value (valid).

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

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

isValid: **boolean** = **true**;

ChangeData**(**valid: **boolean)** **{**

this.isValid = valid;

**}**

**}**

**Modify the app.component.html file:**

Please modify the **app.component.html** file as shown below. Here we have created two radio buttons with the name rb. Then we attach our method ChangeData to the click event of the radio button. For the Valid radio button, we are passing true to the ChangeData method and for the Invalid radio button, we are passing false to the ChangeData method.

**<div>**

**<input** type="radio" name="rb" (click)= "ChangeData(true)" checked **>** Valid

**<input** type="radio" name="rb" (click)= "ChangeData(false)"**>** Invalid

**</div>**

**<div** \*ngIf="isValid"**>**

**<b>**The Data is valid.**</b>**

**</div>**

**<div** \*ngIf="!isValid"**>**

**<b>**The Data is invalid.**</b>**

**</div>**

With the above changes in place, now run the application and you should get the output as expected.

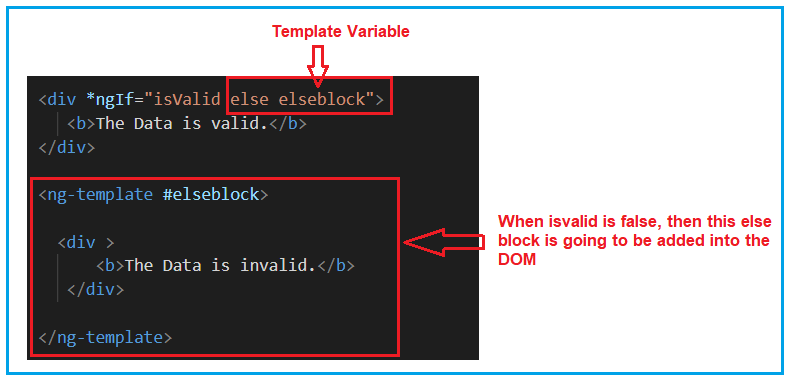
**Angular NgIf directive with else block:**

Like other programming languages such as C, Java, and C#, the else is also used when you want to display something for the false condition of NgIf block. The syntax to use the ngIf block with the else block as follows:

**<div \*ngIf = “condition; else elseBlock”>…</div>**  
**<ng-template #elseblock>….</ng-tempalte>**

Here, for else block you need to use <ng-template> element. It is referred by a template reference variable. NgIf will use that template reference variable to display the else block when the condition is false.

Let us work with the same example. Please have a look at the following image. In the else block we have created a template variable with the name elseblock (you can give any name as per your choice). So, what will happen here is, it will check the ngIf condition and if the condition is true then it will execute if block. On the other hand, if the condition is false, then it will look for the else block. As we created the else block with the name elseblock, so it searches for ng-template tag followed by template variable name (<ng-template #elseblock>) and add that block to the DOM.



So, modify the **app.component.html** file as shown below and then see the output in the browser.

**<div>**

**<input** type="radio" name="rb" (click)= "ChangeData(true)" checked **>** Valid

**<input** type="radio" name="rb" (click)= "ChangeData(false)"**>** Invalid

**</div>**

**<div** \*ngIf="isValid else elseblock"**>**

**<b>**The Data is valid.**</b>**

**</div>**

**<ng-template** #elseblock**>**

**<div** **>**

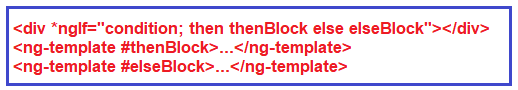
**<b>**The Data is invalid.**</b>**

**</div>**

**</ng-template>**

**NgIf with Then and else:**

The NgIf with then and else is used as follows. When the condition is true then the <ng-template> with the reference variable then block is executed and when the condition is false then the <ng-template> with the reference variable else block is executed. For better understanding please have a look at the following image.



We can have more than one <ng-template> for then and else block and at runtime, we can switch to that ng-template by changing the value of then block and else block. At any given point of time, one ng-template is going to be executed.

So, modify the **app.component.html** file as shown below and then see the output in the browser.

**<div>**

**<input** type="radio" name="rb" (click)= "ChangeData(true)" checked **>** Valid

**<input** type="radio" name="rb" (click)= "ChangeData(false)"**>** Invalid

**</div>**

**<div** \*ngIf="isValid then thenblock else elseblock"**>** **</div>**

**<ng-template** #thenblock**>**

**<div>**

**<b>**The is Then Block**</b>**

**</div>**

**</ng-template>**

**<ng-template** #elseblock**>**

**<div** **>**

**<b>**The is Else Block**</b>**

**</div>**

**</ng-template>**

**NgSwitch Directive**

**What is the Angular ngSwitch directive?**

The Angular ngSwitch directive is actually a combination of two directives i.e. an attribute directive and a structural directive. It is very similar to the switch statement of other programming languages like Java and C# but within a template.

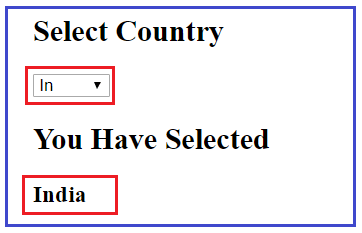
The ngSwitch directive lets you hide or show the HTML elements based on an expression. Here, you can also define a default section using the ng-switch-default directive to show a section if no other sections get a match. So, while working with ngSwitch directive, you need three things to keep in mind, they are ngSwitch, ngSwitchCase and ngSwithDefault.

**Example to understand Angular ngSwitch Directive:**

Let us understand the Angular ngSwitch Directive with an example. What we are going to do is, we will provide a dropdown list to the user to select the country code and based on the selected country code we will provide the full country name. Whenever no country code is selected then it will display you have not selected any country. For example, when you have not selected the country code, then by default the select option is selected from the drop dropdown list and You have not selected any country is displayed as sown in the below image.



But when selected a particular country code let say IN, then India should display as shown in the below image.



Let us see how to implement the above using Angular ngSwitch directive step by step.

**Step1: Modify the app.component.ts file**

First, modify the **app.component.ts** file as shown below. As you can see here, we have created one property i.e. **dropDownValue** and one method i.e. SetDropDownValue. And we are setting the property value using the methods parameter value. This method is going to be called by the dropdown list **change** event.

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

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

**public** dropDownValue = "";

SetDropDownValue**(**drpValue : **any)** **{**

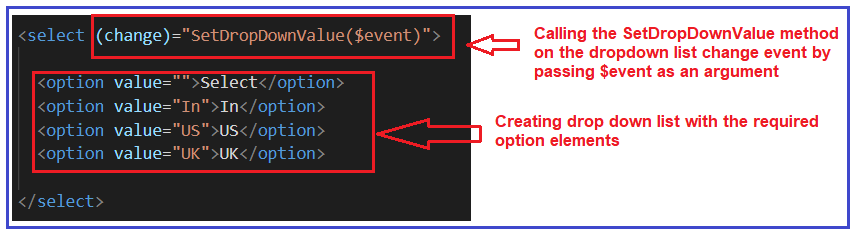
this.dropDownValue = drpValue.target.value;

**}**

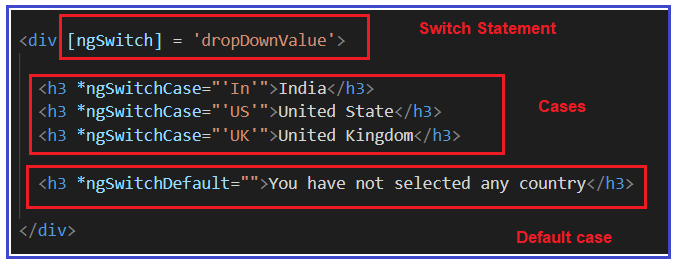
**}**

**Step2: Modify the app.component.html file**

This is the important part and here you need to understand two things. First, we need to create the drop-down list and then on the change event of the dropdown list, we need to call the **SetDropDownValue** method that we created within our component class. For better understanding please have a look at the following image.



In the second step, we need to use the ngSwitch directive and in the ngSwitch directive, we need to bind the property (**dropDownValue**) and whose value is set by the drop-down list change event. Then we need to write the required ngSwitchCase statements. As we have three options in the drop-down list and so we have one ngSwitchCase one per each option value. Finally, we have the default switch case i.e. ngSwitchDefault which will execute when no options are selected from the dropdown list. For better understanding please have a look at the following image.



So, modify the **app.component.html** file as shown below and check in the browser and it should work as expected.

**<h2>**Select Country**</h2>**

**<select** (change)="SetDropDownValue($event)"**>**

**<option** value=""**>**Select**</option>**

**<option** value="In"**>**In**</option>**

**<option** value="US"**>**US**</option>**

**<option** value="UK"**>**UK**</option>**

**</select>**

**<h2>**You Have Selected**</h2>**

**<div** [ngSwitch] = 'dropDownValue'**>**

**<h3** \*ngSwitchCase="'In'"**>**India**</h3>**

**<h3** \*ngSwitchCase="'US'"**>**United State**</h3>**

**<h3** \*ngSwitchCase="'UK'"**>**United Kingdom**</h3>**

**<h3** \*ngSwitchDefault=""**>**You have not selected any country**</h3>**

**</div>**

## **Angular ngStyle Directive**

**What is the Angular ngStyle Directive?**

The ngStyle directive is used to set the DOM element style properties. For example, if you want to create a button with font size 20, the color red and font-weight bold, then you could do the same using the Angular ngStyle directive as shown in the below image.

What is the Angular ngStyle Directive?

Let’s implement this and see the output in the browser.

**Step1: Modify the app.component.ts file**

Please copy and paste the following code in your **app.component.ts** file.

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

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

**}**

**Step2: Modify the app.component.html file**

Please modify the **app.component.html** file as shown below.

**<div>**

**<button** [ngStyle]="{'color':'red', 'font-weight': 'bold', 'font-size.px':20}"**>**Click Me **</button>**

**</div>**

Now run the application and see the output in the browser and you should get a button with red color, font size 20 and font-weight bold as shown in the below image.

Angular ngStyle Directive

Instead of hard coding the styles directly in the ngStyle directive, you can also create a method in your component and then bind that method to the ngStyle directive. Let us implement this.

**Modify app.component.ts file:**

Please modify the app.component.ts class file as shown below. Here, we created one method i.e. AddButtonCSSStyles which returns an object with three key-value pair properties. The key here is nothing but the style name and the value is the value of that respective style property.

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

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

AddButtonCSSStyles**()** **{**

let CssStyles = **{**

'color':'red',

'font-weight': 'bold',

'font-size.px': 20

**}**;

**return** CssStyles;

**}**

**}**

**Modify app.component.html file:**

Please modify the app.component.html file as shown below. As you can see, now the ngStyle directive is bounded to the **AddButtonCSSStyles ()** method of the AppComponent class.

**<div>**

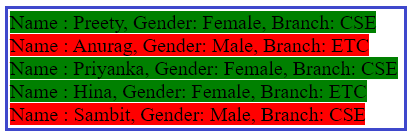
**<button** [ngStyle]="AddButtonCSSStyles()"**>**Click Me **</button>**

**</div>**

Now, run the application and you should the output as expected.

**Angular ngStyle directive with dynamic values:**

The ngStyle directive becomes more effective when the value is dynamic. For example, let say we have a list of students and we want to display the student list on a webpage as shown below. When the student gender is male we want to display that record in red color else we want to display the student in green color.



Let see how to implement this.

**Step1: Modify the app.component.ts file**

Please modify the **app.component.ts** file as shown below.

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

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

students: **any[]** = **[**

**{**

Name: 'Preety', Branch: 'CSE', Gender: 'Female'

**}**,

**{**

Name: 'Anurag', Branch: 'ETC', Gender: 'Male'

**}**,

**{**

Name: 'Priyanka', Branch: 'CSE', Gender: 'Female'

**}**,

**{**

Name: 'Hina', Branch: 'ETC', Gender: 'Female'

**}**,

**{**

Name: 'Sambit', Branch: 'CSE', Gender: 'Male'

**}**

**]**;

**}**

**Step2: Modify the app.component.html file**

Now modify the **app.component.html** file as shown below.

**<div** \*ngFor='let student of students'**>**

**<span** [ngStyle] ="{'background-color':student.Gender === 'Male' ? 'red' : 'green'}"**>**Name : {{student.Name}}, Gender: {{student.Gender}}, Branch: {{student.Branch}}**</span>**

**</div>**

Now, if you run the application, then you should get the output as expected.

## **Angular ngClass Directive**

##### ****What is Angular ngClass Directive?****

The Angular ngClass directive is used to add or remove CSS classes dynamically (run-time) from a DOM Element. Here we will discuss the different mechanisms or methods to use the ngClass directive. They are as follows:

1. **ngClass with string**
2. **ngClass with array**
3. **The ngClass with object**
4. **ngClass with component method**

##### ****ngClass as a string:****

Let us see an example to use ngClass as a string.

###### **Modify app.component.css file:**

First open the **app.component.css** file and then copy and paste the following CSS styles. These styles are going to be used in our HTML pages.

**.one{**

color: red;

**}**

**.two{**

font-size: 30px;

**}**

**.three{**

font-weight: bold;

**}**

**.four{**

font-style: italic;

**}**

**.five{**

color: green;

**}**

###### **Step2: Modify app.component.ts file**

Open **app.component.ts** file and then copy and paste the following code in it.

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

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

**}**

###### **Step3: Modify the app.component.html file**

The syntax to use the ngClass directive as a string is given below. You have to apply the class in between the single quote which is in between a pair of double quotes.

Angular ngClass Directive

Let us understand this with an example. Please modify the **app.component.html** file as shown below. As you can see here we are applying the class one as string to the ngClass directive.

**<h3** [ngClass]="'one'"**>**

Angular ngClass with String

**</h3>**

As the css class one holds one property color with value red, so when you browse you should get the message “Angular ngClass with String” in red color as shown in the below image.

What is Angular ngClass Directive?

##### ****Applying multiple css classes with ngClass Directive:****

The syntax to use ngClass as a string with multiple css classes is given below. Here, you need to specify multiple classes separated by a space.

Applying multiple css classes with NgClass Directive:

Let us see how to apply multiple css classes to ngClass directive with an example. Suppose, you want to apply red color along with you want display it in italic. That means now you need to apply two classes i.e. one and four. In order to do this, please modify the **app.component.htm**l file as shown below.

**<h3** [ngClass]="'one four'"**>**

Angular ngClass with String

**</h3>**

Now if you browse application, then you should get the message with red color italic font as shown in the below image.

Applying multiple css classes with ngClass

##### ****ngClass with Array:****

This is very much similar to the previous example i.e. applying multiple css classes. The difference is only the way you applying the css classes. The syntax to use ngClass with Array is given below. Here, you need to use one square bracket and then you need to specify the classes in a single quotes separated by a comma.

ngClass with Array

For example, suppose you want to apply three css classes (three, four, five) to the ngClass directive as array. Then you need to modify the **app.component.html** file as shown below.

**<h3** [ngClass]="['three', 'four', 'five']"**>**

Angular ngClass with String

**</h3>**

When you run the application, you should get the output as expected as as shown in the below image.

ngClass with Array

##### ****ngClass as object:****

The syntax to use ngClass as object is given below. If you have worked with JSON then you can easily understand this syntax. In JSON we need to specify the data as key value pair. Here, the key is the class name and then value can be true or false. If you specify true then that class will be applied and value false means the class will not be applied.

ngClass as object

Let us understand this with an example. Please modify the app.component.html file as shown below. As you can see we have applied three classes among which we have set the value to true for the classes one and three.

**<h3** [ngClass]="{'one':true, 'two':false, 'three':true}"**>**

Angular ngClass with String

**</h3>**

Now if you browse the application, then you can see the classes one and three are applied while the class two is not applied to the DOM element.

##### ****ngClass with Method:****

Let us understand this with an example.

First we need to create one method in the **AppComponent**class which will return the required CSS classes as an object. So, modify the **app.component.ts** file as shown below.

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

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

AddCSSClasses**(**flag:**string)** **{**

let Cssclasses;

**if(**flag == "type1"**)**

**{**

Cssclasses = **{**

'one' : **true**,

'two' : **true**

**}**

**}**

**else**

**{**

Cssclasses = **{**

'four' : **true**,

'five' : **true**

**}**

**}**

**return** Cssclasses;

**}**

**}**

Modify the **app.component.html** file as shown below to use method with ngClass directive.

**<h3** [ngClass]="AddCSSClasses('type1')"**>**

Angular ngClass with String

**</h3>**