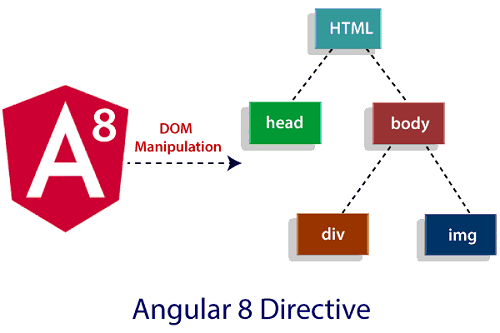
Angular Directives

The Angular 8 directives are used to manipulate the DOM. By using Angular directives, you can change the appearance, behavior or a layout of a DOM element. It also helps you to extend HTML.



**Angular directives can be classified in 3 categories based on how they behave:**

* Component Directives
* Structural Directives
* Attribute Directives

**Component Directives:** Component directives are used in main class. They contain the detail of how the component should be processed, instantiated and used at runtime.

**Structural Directives:** Structural directives start with a \* sign. These directives are used to manipulate and change the structure of the DOM elements. For example, \*ngIf directive, \*ngSwitch directive, and \*ngFor directive.

* **\*ngIf Directive:** The ngIf allows us to Add/Remove DOM Element.
* **\*ngSwitch Directive:** The \*ngSwitch allows us to Add/Remove DOM Element. It is similar to switch statement of C#.
* **\*ngFor Directive:** The \*ngFor directive is used to repeat a portion of HTML template once per each item from an iterable list (Collection).

**Attribute Directives:** Attribute directives are used to change the look and behavior of the DOM elements. For example: ngClass directive, and ngStyle directive etc.

* **ngClass Directive:** The ngClass directive is used to add or remove CSS classes to an HTML element.
* **ngStyle Directive:** The ngStyle directive facilitates you to modify the style of an HTML element using the expression. You can also use ngStyle directive to dynamically change the style of your HTML element.

# Angular 8 ngIf Directive

The ngIf Directives is used to add or remove HTML Elements according to the expression. The expression must return a Boolean value. If the expression is false then the element is removed, otherwise element is inserted. It is similar to the ng-if directive of AngularJS.

### ngIf Syntax

1. <p \*ngIf="condition">
2. condition is **true** and ngIf is **true**.
3. </p>
4. <p \*ngIf="!condition">
5. condition is **false** and ngIf is **false**.
6. </p>

### The \*ngIf directive form with an "else" block

1. <div \*ngIf="condition; else elseBlock">
2. Content to render when condition is **true**.
3. </div>
4. <ng-template #elseBlock>
5. Content to render when condition is **false**.
6. </ng-template>

NoteL: The \*ngIf directive is most commonly used to conditionally show an inline template.

1. @Component({
2. selector: 'ng-if-simple',
3. template: `
4. <button (click)="show = !show">{{show ? 'hide' : 'show'}}</button>
5. show = {{show}}
6. <br>
7. <div \*ngIf="show">Text to show</div>
8. `
9. })
10. export **class** NgIfSimple {
11. show: **boolean** = **true**;
12. }

# Angular 8 \*ngFor Directive

The \*ngFor directive is used to repeat a portion of HTML template once per each item from an iterable list (Collection). The ngFor is an Angular structural directive and is similar to ngRepeat in AngularJS. Some local variables like Index, First, Last, odd and even are exported by \*ngFor directive.

## Syntax of ngFor

See the simplified syntax for the ngFor directive:

1. <li \*ngFor="let item of items;"> .... </li>

## How to use ngFor Directive?

To Use ngFor directive, you have to create a block of HTML elements, which can display a single item of the items collection. After that you can use the ngFor directive to tell angular to repeat that block of HTML elements for each item in the list.

### Example for \*ngFor Directive

First, you have to create an angular Application. After that open the app.component.ts and add the following code.

The following Code contains a list of Top 3 movies in a movies array. Let's build a template to display these movies in a tabular form.

1. **import** { Component } from '@angular/core';
2. @Component({
3. selector: 'movie-app',
4. templateUrl:'./app/app.component.html',
5. styleUrls:['./app/app.component.css']
6. })
7. export **class** AppComponent
8. {
9. title: string ="Top 10 Movies" ;
10. movies: Movie[] =[
11. {title:'Zootopia',director:'Byron Howard, Rich Moore',cast:'Idris Elba, Ginnifer Goodwin, Jason Bateman',releaseDate:'March 4, 2016'},
12. {title:'Batman v Superman: Dawn of Justice',director:'Zack Snyder',cast:'Ben Affleck, Henry Cavill, Amy Adams',releaseDate:'March 25, 2016'},
13. {title:'Captain America: Civil War',director:'Anthony Russo, Joe Russo',cast:'Scarlett Johansson, Elizabeth Olsen, Chris Evans',releaseDate:'May 6, 2016'},
14. {title:'X-Men: Apocalypse',director:'Bryan Singer',cast:'Jennifer Lawrence, Olivia Munn, Oscar Isaac',releaseDate:'May 27, 2016'},
15. ]
16. }
17. **class** Movie {
18. title : string;
19. director : string;
20. cast : string;
21. releaseDate : string;
22. }

Now, open the app. component.html and add the following code:

1. <div **class**='panel panel-primary'>
2. <div **class**='panel-heading'>
3. {{title}}
4. </div>
5. <div **class**='panel-body'>
6. <div **class**='table-responsive'>
7. <table **class**='table'>
8. <thead>
9. <tr>
10. <th>Title</th>
11. <th>Director</th>
12. <th>Cast</th>
13. <th>Release Date</th>
14. </tr>
15. </thead>
16. <tbody>
17. <tr \*ngFor="let movie of movies;">
18. <td>{{movie.title}}</td>
19. <td>{{movie.director}}</td>
20. <td>{{movie.cast}}</td>
21. <td>{{movie.releaseDate}}</td>
22. </tr>
23. </tbody>
24. </table>
25. </div>
26. </div>
27. </div>

# Angular 8 ngSwitch Directive

In Angular 8, ngSwitch is a structural directive which is used to Add/Remove DOM Element. It is similar to switch statement of C#. The ngSwitch directive is applied to the container element with a switch expression.

## Syntax of ngSwitch

1. **<container\_element** [ngSwitch]="switch\_expression"**>**
2. **<inner\_element** \*ngSwitchCase="match\_expresson\_1"**>**...**</inner\_element>**
3. **<inner\_element** \*ngSwitchCase="match\_expresson\_2"**>**...**</inner\_element>**
4. **<inner\_element** \*ngSwitchCase="match\_expresson\_3"**>**...**</inner\_element>**
5. **<inner\_element** \*ngSwitchDefault**>**...**</element>**
6. **</container\_element>**

## ngSwitchCase

In Angular ngSwitchCase directive, the inner elements are placed inside the container element. The ngSwitchCase directive is applied to the inner elements with a match expression. Whenever the value of the match expression matches the value of the switch expression, the corresponding inner element is added to the DOM. All other inner elements are removed from the DOM

If there is more than one match, then all the matching elements are added to the DOM.

## ngSwitchDefault

You can also apply the ngSwitchDefault directive in Angular 8. The element with ngSwitchDefault is displayed only if no match is found. The inner element with ngSwitchDefault can be placed anywhere inside the container element and not necessarily at the bottom. If you add more than one ngSwitchDefault directive, all of them are displayed.

Any elements placed inside the container element, but outside the ngSwitchCase or ngSwitchDefault elements are displayed as it is.

## ngSwitch Directive Example

**Use the following code in app.component.ts file of your application:**

1. class item {
2. name: string;
3. val: number;
4. }
5. export class AppComponent
6. {
7. items: item[] = [{name: 'One', val: 1}, {name: 'Two', val: 2}, {name: 'Three', val: 3}];
8. selectedValue: string= 'One';
9. }

**Use the following code in the app.component.html file of your application:**

1. **<select** [(ngModel)]="selectedValue"**>**
2. **<option** \*ngFor="let item of items;" [value]="item.name"**>**{{item.name}}**</option>**
3. **</select>**
4. **<div** class='row' [ngSwitch]="selectedValue"**>**
5. **<div** \*ngSwitchCase="'One'"**>**One is Pressed**</div>**
6. **<div** \*ngSwitchCase="'Two'"**>**Two is Selected**</div>**
7. **<div** \*ngSwitchDefault**>**Default Option**</div>**
8. **</div>**

**NgStyle & NgClass**

## [NgStyle](https://codecraft.tv/courses/angular/built-in-directives/ngstyle-and-ngclass/#_ngstyle)

The NgStyle directive lets you set a given DOM elements style properties.

One way to set styles is by using the NgStyle directive and assigning it an object literal, like so:

HTML

Copy<div [ngStyle]="{'background-color':'green'}"></<div>

This sets the background color of the div to green.

ngStyle becomes much more useful when the value is *dynamic*. The *values* in the object literal that we assign to ngStyle can be JavaScript expressions which are evaluated and the result of that expression is used as the value of the CSS property, like this:

HTML

Copy<div [ngStyle]="{'background-color':person.country === 'UK' ? 'green' : 'red' }"></<div>

## [NgClass](https://codecraft.tv/courses/angular/built-in-directives/ngstyle-and-ngclass/#_ngclass)

The NgClass directive allows you to set the CSS class dynamically for a DOM element.

#### **Tip**

The NgClass directive will feel very similar to what ngClass used to do in Angular 1.

There are two ways to use this directive, the first is by passing an object literal to the directive, like so:

HTML

Copy[ngClass]="{'text-success':true}"

When using an object literal, the keys are the classes which are added to the element if the value of the key evaluates to true.

So in the above example, since the value is true this will set the class text-success onto the element the directive is attached to.

The value can also be an *expression*, so we can re-write the above to be.

HTML

Copy[ngClass]="{'text-success':person.country === 'UK'}"

<h4>NgClass</h4>

<ul \*ngFor="let person of people">

<li [ngClass]="{

'text-success':person.country === 'UK',

'text-primary':person.country === 'USA',

'text-danger':person.country === 'HK'

}">{{ person.name }} ({{ person.country }})

</li>

</ul>