Angular Directives

The Angular directives are used to manipulate the DOM(html document/element).

Directives are instructions in the DOM. They specify how to place your components and business logic in the Angular.

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:** The Component is also a type of directive in angular with its own template, styles, and logic needed for the view. The Component Directive is the most widely used directive in the angular application and you cannot create an angular application without a component.

**Structural Directives:** Structural directives start with a \* sign. These directives are used to manipulate and change the structure of the DOM elements. \*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 (Array/Collection).

**Attribute Directives:** Attribute Directive is basically used to modify or alter the appearance and behavior of an element. 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.

# Angular \*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. Some local variables like Index, First, Last, odd and even are exported by \*ngFor directive.

## Syntax of ngFor

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

# Angular ngSwitch Directive

ngSwitch is a structural directive which is used to Add/Remove DOM Element. 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**>**...**</inner\_element>**
6. **</container\_element>**

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.

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

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

<div [ngStyle]="{'background-color':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.

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

[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.

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

<h4>NgClass</h4>

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

<li [ngClass]="{

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

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

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

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

</li>

</ul>