

# Angular 2.0 for JEE

Lesson 05 : Directives



# Lesson Objectives

- What are directives?
- Types of directives - component, structural and attribute
- Creating a basic directive
- Creating your own structural directive





# Directives

- Directives are the most fundamental unit of Angular applications.
- Components are high-order directives with templates and serve as building blocks of Angular applications.
- Used to attach behavior to element in DOM
- There are three kinds of directives in Angular:
  - **Components** — directives with a template.
  - **Structural directives** — change the DOM layout by adding and removing DOM elements.
  - **Attribute directives** — change the appearance or behavior of an element, component, or another directive.



# Demo

## ➤ Component Directive





# Structural Directives

- Structural directives are responsible for HTML layout.
- They shape or reshape the DOM's *structure*, typically by adding, removing, or manipulating elements
- Structural directives are easy to recognize. An asterisk (\*) precedes the directive attribute name. Examples
  - \*ngIf displaying the hero's name if hero exists.
  - `<div *ngIf="hero" class="name">{{hero.name}}</div>`



# Structural Directives (Contd...)

- In angular we have three built-in structural directives
  - **ngIf** : ngIf directive inserts or removes an element based on a truthy/falsey condition.
  - **ngFor** : ngFor directive is used to iterate an array of items
  - **ngSwitch**: ngSwitch directive is used to conditionally swap DOM structure on template based on an expression.



# Demo

- Demo ngIf Directive
- Demo ngFor Directive
- Demo ngSwitch Directive





# Creating own structural directive

- We can create custom attribute directives and custom structural directives using @Directive decorator.
- Structural directives are responsible for HTML layout.
- We can add and remove elements in DOM layout dynamically.
- The HTML element using directive is called host element for that directive.
- To add and remove host elements from DOM layout we can use TemplateRef and ViewContainerRef classes in our structural directive.
- To change DOM layout we should use TemplateRef and ViewContainerRef in our structural directive.
- **TemplateRef** : It represents an embedded template that can be used to instantiate embedded views.
- **ViewContainerRef** : It represents a container where one or more views can be attached.



# Demo



➤ Demo Own structural Directive





# Attribute Directives

- Attribute directives alter the appearance or behavior of an existing element.
- In templates they look like regular HTML attributes.
- Some important angular in-built attribute directives are :
  - **ngModel** : Implements two-way data binding, which modifies the behavior of an existing element (typically an `<input>`) by setting its display value property and responding to change events.
  - **ngStyle** : Changes the style based on a result of expression evaluation.
  - **ngClass** : Conditionally adds and removes CSS classes on an HTML element based on an expression's evaluation result



# Demo

- Demo ngStyle Directive
- Demo ngClass Directive





# Demo

➤ Demo Custom Directive





# Summary

- Component Directives is a directive with a template.
- Directives can't be bootstrapped
- Structural directives change the DOM layout by adding and removing DOM elements.
- Attribute directives changes the appearance or behavior of a DOM element.
- In order to use *ngModel* in the application components, we need to compulsorily add FormsModule in the Imports array of Application Module class
- *ngNonBindable* tells the Angular not to compile or bind a particular section of a DOM.
- Using *ngStyle* directive we can set CSS properties for the DOM element from Angular expressions
- *ngClass* directive allows us to dynamically set and change the CSS classes for a given DOM element



# Lab



## ➤ Lab 2

