Angular Training

(Intermediate to Advanced)

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

Schedule for React JS Training

Day1	:	Custom Pipes in Angular
Day2	:	Parent-Child Communication
Day3	**:**	**Custom Directives in Angular**
Day4	:	Working with Reactive Forms
Day5	:	Dependency Injection and Services in Angular
Day6	:	RxJS Library – Observables
Day7	:	Http Client – Server calls in Angular
Day8	:	Routing and Security in Angular

Day3
Custom Directives in Angular

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

Index – Day3

Introduction to Angular Directives

Narasimha

Sr. IT Trainer/Consultant

Introduction to Angular Directives

- Directives are classes that add additional behavior to elements in your Angular applications.
- Use Angular's built-in directives to manage forms, lists, styles, and what users see.
- Angular supports different types of directives to address corresponding requirements.
- Eg: ngModel, ngClass ngFor, ngIf, etc…

Different types of Angular directives

Narasimha

Sr. IT Trainer/Consultant

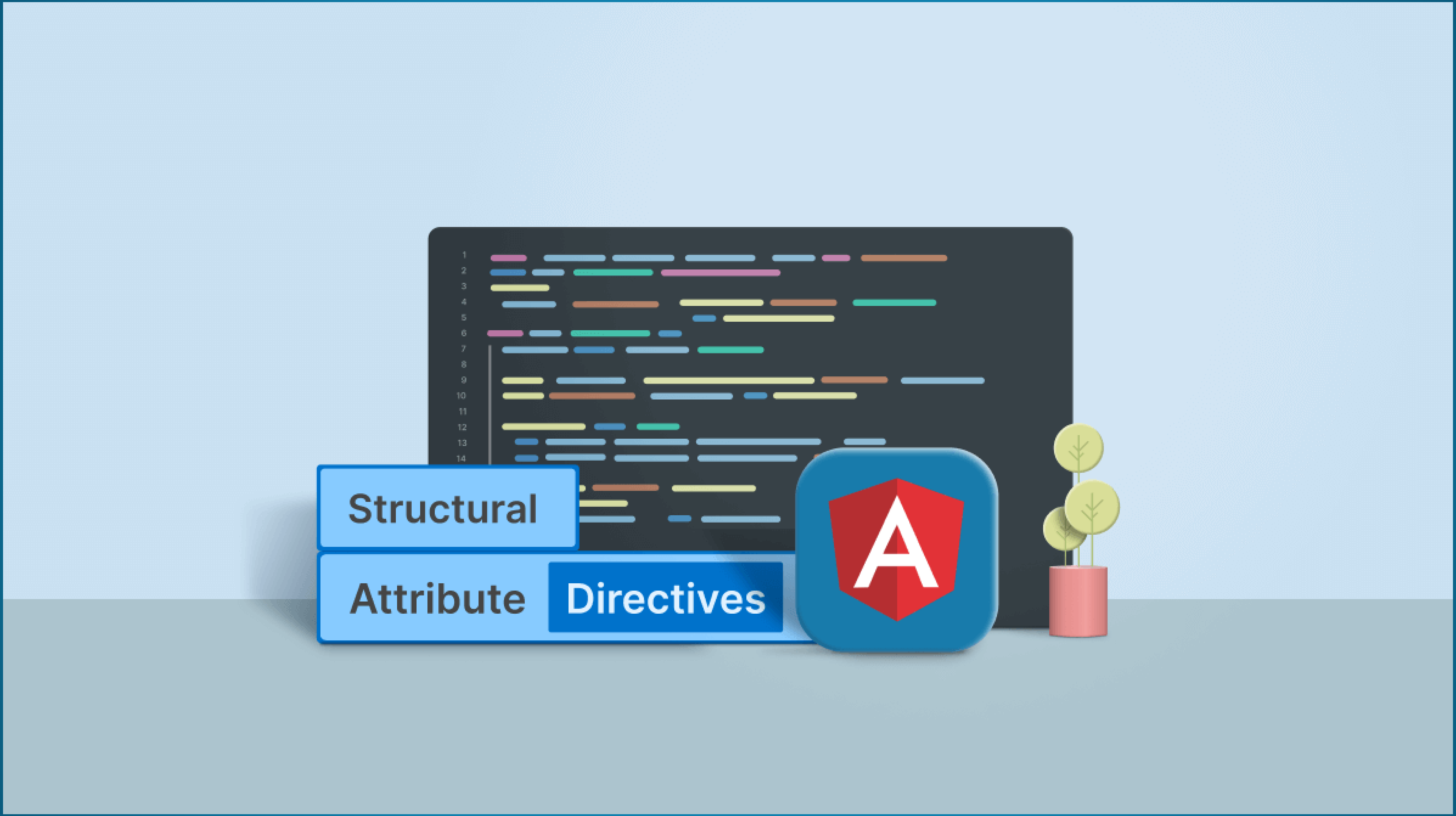
Different types of Angular directives

Angular framework supports three different types of directives.

1. Component Directives
2. Attribute Directives
3. Structural Directives

Different types of Angular directives

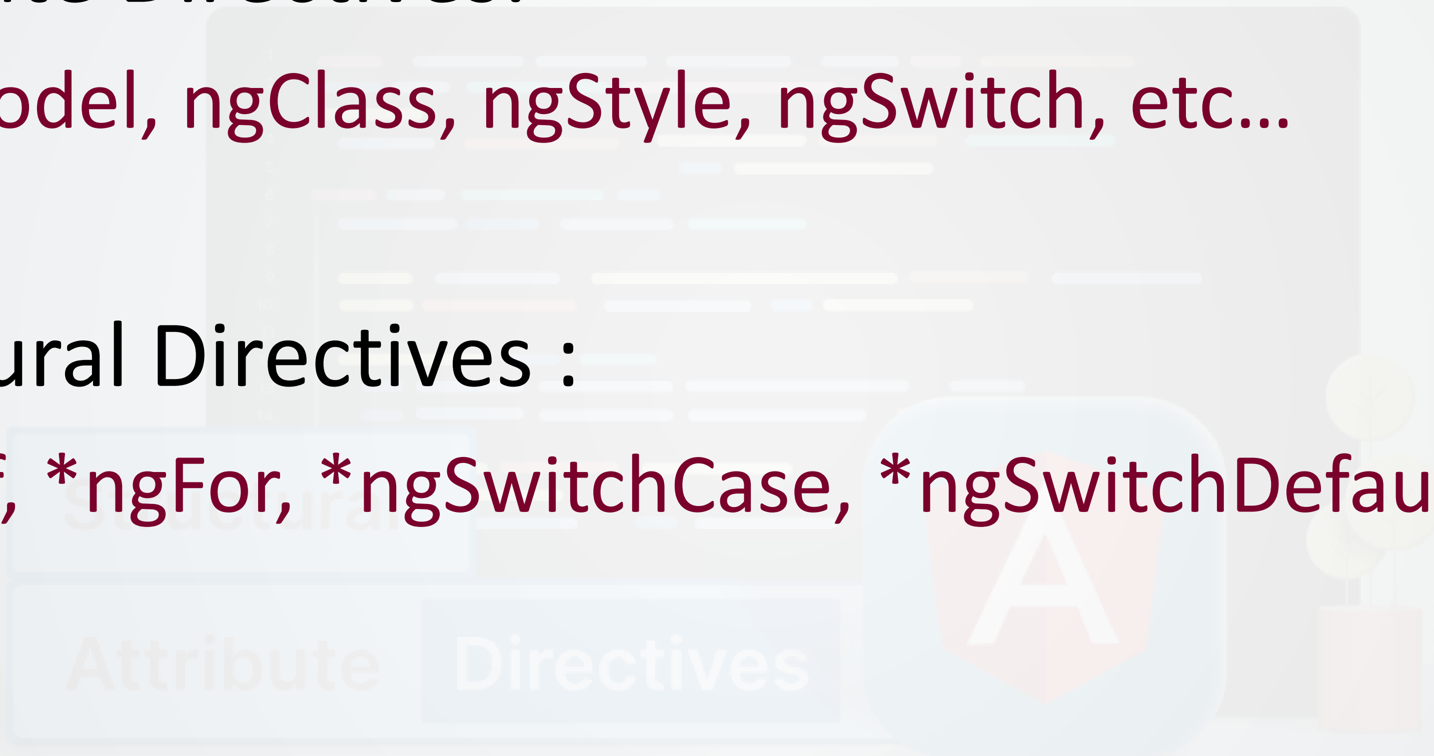
1. **Component** : This type of directive is the most common directive type.
2. **Attribute** : Change the appearance or behavior of an element.
3. **Structural** : Change the DOM layout by adding and removing DOM elements.


Structural
Attribute
Directives

Attribute and Structural

1. Attribute Directives:
   – ngModel, ngClass, ngStyle, ngSwitch, etc…

2. Structural Directives :
   – *ngIf, *ngFor, *ngSwitchCase, *ngSwitchDefault



Usage of ngSwitch

```
<tag  [ngSwitch]=“variable">
          <tag *ngSwitchCase=" value ">   </tag>
          <tag *ngSwitchCase=" value ">   </tag>
          ........
          ........
          <tag *ngSwitchDefault> </tag>
</tag>
```

Practical HandsOns

What is Custom Directives?

Narasimha

Sr. IT Trainer/Consultant

Custom Directives

- Custom Directives are used in Angular to extend the functionality of HTML.
- Custom directives also created as class with corresponding rules.
- @Directive() decorator is used to mark the Custom Directive class.
- Custom Directives may be attribute or structural based on the requirement.

How to Create Custom Directives?

Narasimha

Sr. IT Trainer/Consultant

How to Create Custom Directives?

1. ng **g**enerate **d**irective highlight
2. @Directive() from @angular/core
3. Inject required items in constructor. It will be depends on the directive type.

   → constructor(private el: ElementRef)
4. Define the required behaviour in the class
5. Applying the directive on Html Element / Component.

Developing
Attribute Custom Directive

Narasimha

Sr. IT Trainer/Consultant

Attribute Custom Directives

- Import **ElementRef** from @angular/core.
- ElementRef grants direct access to the host DOM element through its **nativeElement** property.
- Add ElementRef in the directive's **constructor**() to inject a reference to the host DOM element
- Add logic to your Directive class that change the behavior.

Attribute Custom Directives

```
import { Directive, ElementRef } from '@angular/core';

@Directive({
            selector: '[appHighlight]'
  })
export class HighlightDirective {

    constructor(private el: ElementRef) {
            this.el.nativeElement.style.backgroundColor = 'yellow';
     }

  }
```

Passing Values to Attribute Directives

1. import Input from @angular/core.
2. Add an appHighlight @Input() property.
3. Use property binding with the appHighlight directive selector:

```
<p [appHighlight]="color">Highlight me!</p>
```

Note: The [appHighlight] attribute binding performs two tasks:

a. Applies the highlighting directive to the <p> element

b Sets the directive's highlight color with a property binding

Practical HandsOns

Developing Structural Custom Directive

Narasimha

Sr. IT Trainer/Consultant

Structural Custom Directive

- Import Input, TemplateRef, and ViewContainerRef
- Inject TemplateRef and ViewContainerRef in the directive constructor as private variables.
- Apply Input() decorator on the property
- Define the required functionality in the class
- Apply on the html element / component

TemplateRef

1. TemplateRef:
   - Refers the current tag on which we apply custom directive.
   - TemplateRef is injected into the constructor of the custom directive class.

```
constructor(private templateRef: TemplateRef<any>)  {

        }
```

ViewContainerRef

- It represents a container where one or more views can be attached.
- It can contain embedded views (created by instantiating a TemplateRef).
- We can organize the TemplateRef with the createEmbeddedView() method.

ViewContainerRef

```
constructor(private viewContainer: ViewContainerRef,
    private templateRef: TemplateRef<any>)  {


  }
```

```
this.viewContainer.createEmbeddedView(this.templateRef);
```

```
this.viewContainer.clear();
```

Practical HandsOns

Q & A