Angular 16 : Online Class

ElementRef

By: Sahosoft Solutions

Presented by : Ajeet Kumar

ElementRef

ElementRef

ElementRef is a class that can hold a reference to a DOM element. We use ElementRef class to access DOM to change host element appearance.

If **ElementRef** is injected to a component, the injected instance is a reference to the host element of the current component.

By using **ElementRef** we are able to easily get a reference to an element and its properties, and we can modify the element's properties through the ElementRef.

If you observe its class structure, you'll see that it only holds the native element it's associated with. It's useful for accessing native DOM element as we can see here:

```
constructor(private elRef: ElementRef) {
    this.elRef.nativeElement.style.color = 'blue';
    this.elRef.nativeElement.style.fontSize = '40px';
  }
```

Angular 16 : Online Class

Custom Directives

By: Sahosoft Solutions

Presented by : Ajeet Kumar

Custom Directives

Custom Directives

Angular provides three types of directive: **component directive**, **attribute directive** and **structural directive**.

- ✓ Component is used to create HTML template.
- ✓ Attribute directive changes the appearance and behavior of DOM element.
- ✓ Structural directive changes the DOM layout by adding and removing DOM elements.

Angular also provides built-in directives.

- ✓ The built-in attribute directives are **NgStyle, NgClass** etc. These directives change the appearance and behavior of HTML elements.
- ✓
- ✓ The built-in structural directives are **NgFor** and **NgIf** etc. These directives can add and remove HTML elements from the DOM layout.

Custom Directives

Custom Directives

we will see how to create custom attribute and structural directives.

In angular we create these directives using **@Directive()** decorator. It has a **selector** metadata that defines the custom directive name.

Custom directives are created using following syntax.

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

@Directive({
  selector: '[appDir]'
})
export class appDirective {

  constructor() { }

  }
}
```

Custom Directives

Custom Directives

The directive name is **appDir** here. It should be enclosed within bracket **[]**. We can keep directive name as we want but it should be started with your custom name or any other keyword but not with Angular keyword such as ng.

- ✓ To behave our directive like **attribute directive**, we can use **ElementRef** to change appearance.
- ✓ To behave our directive like **structural directive**, we can use **TemplateRef** and **ViewContainerRef**.

Angular 16 : Online Class

@HostBinding() and @HostListener()

By: Sahosoft Solutions

Presented by : Ajeet Kumar

@HostBinding() and @HostListener()

To understand **@HostListener** and **@HostBinding**, you should have basic knowledge about directives in Angular. There are three types of directives in Angular:

1. Component Directive
2. Attribute Directive
3. Structural Directive

@HostBinding and *@HostListener* are two decorators in Angular that can be really useful in custom directives.

✓ **@HostBinding** lets you set properties on the element or component that hosts the directive
✓ **@HostListener** lets you listen for events on the host element or component.

@HostBinding() and @HostListener()

@HostBinding() Decorator

In Angular, the **@HostBinding()** function decorator allows you to set the properties of the host element from the directive class.

Let's say we want to change the style properties such as height, width, color, margin, border, etc., or any other internal properties of the host element in the directive class.

Here, we need to use the **@HostBinding()** decorator function to access these properties on the host element and assign a value to it in directive class.

The **@HostBinding()** decorator takes one parameter, the name of the host element property which value we want to assign in the directive.

@HostBinding() and @HostListener()

@HostListener() Decorator

In Angular, the @HostListener() function decorator allows you to handle events of the host element in the directive class. In Angular, the @HostListener() function decorator makes it super easy to handle events raised in the host element inside the directive class.

Let's take the following requirement: when we hover our mouse over the host element, only the color of the host element should change.

In addition, when the mouse is gone, the color of the host element should change to its default color.

To do this, we need to handle events raised on the host element in the directive class. In Angular, we do this using **@HostListener()** .

To understand **@HostListener()** in a better way, consider another simple scenario:

on the click of the host element, we want to show an alert window. To do this in the directive class, add @HostListener() and pass the event 'click' to it. Also, associate a function to raise an alert as shown in the listing below:

```
@HostListener('click') onClick() {
   window.alert('Host Element Clicked');
}
```