What Are @HostBinding() and @HostListener() in Angular?

In this article, we go over how you can incorporate these two decorators into the TypeScript code you use to build your Angular web app.

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

1. Component
2. Attribute Directive
3. Structural Directive

The basic difference between a component and a directive is that a component has a template, whereas an attribute or structural directive does not have a template. To understand these two concepts, let us start by creating a simple custom attribute directive. The directive below changes the background color of the host element:

```
import { Directive, ElementRef, Renderer } from '@angular/core';
@Directive({
    selector: '[appChbgcolor]'
})
export class ChangeBgColorDirective {
    constructor(private el: ElementRef, private renderer: Renderer) {
        this.ChangeBgColor('red');
    }
    ChangeBgColor(color: string) {
        this.renderer.setElementStyle(this.el.nativeElement, 'color', color
);
    }
}
```

To create a custom attribute directive, you need to create a class and decorate it with @Directive. In the constructor of the directive class, inject the objects **ElementRef** and **Renderer**. Instances of these two classes are needed to get the reference of the host element and of the renderer.

You can use the above attribute directive on a component template as shown in the code block below:

```
<div appChbgcolor>
    <h3>{{title}}</h3>
</div>
```

Here, the component class holding the host element is created as below:

```
import { Component } from '@angular/core';
@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']
})
export class AppComponent {
    title = 'Hey ng Developer ! ';
}
```

Right now, the **appChbgcolor** directive will change the color of the host element.

@HostListener() Decorator

In Angular, the @HostListener() function decorator allows you to handle events of the host element in the directive class.

Let's take the following requirement: when you hover you 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, you need to handle events raised on the host element in the directive class. In Angular, you do this using **@HostListener()** .

To understand **@HostListener()** in a better way, consider another simple scenario: on the click of the host element, you 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');
}
```

In Angular, the @HostListener() function decorator makes it super easy to handle events raised in the host element inside the directive class. Let's go back to our requirement that says you must change the color to red only when the mouse is hovering, and when it's gone, the color of the host element should change to black. To do this, you need to handle the **mouseenter** and **mouseexit** events of the host element in the directive class. To achieve this, modify the **appChbgcolor** directive class as shown below:

```
import { Directive, ElementRef, Renderer, HostListener } from '@angular/cor
e';
@Directive({
    selector: '[appChbgcolor]'
})
export class ChangeBgColorDirective {
    constructor(private el: ElementRef, private renderer: Renderer) {
        // this.ChangeBgColor('red');
    }
    @HostListener('mouseover') onMouseOver() {
        this.ChangeBgColor('red');
    }
    @HostListener('click') onClick() {
        window.alert('Host Element Clicked');
    }
    @HostListener('mouseleave') onMouseLeave() {
        this.ChangeBgColor('black');
    }
    ChangeBgColor(color: string) {
        this.renderer.setElementStyle(this.el.nativeElement, 'color', color
);
    }
}
```

In the directive class, we are handling the **mouseenter** and **mouseexit** events. As you see, we are using @HostListener() to handle these host element events and assigning a function to it.

So, let's use **@HostListener()** function decorator to handle events of the host element in the directive class.

@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 you 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, you'd 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.

In our example, our host element is an HTML div element. If you want to set border properties of the host element, you can do that using @HostBinding() decorator as shown below:

```
@HostBinding('style.border') border: string;
@HostListener('mouseover') onMouseOver() {
    this.border = '5px solid green';
}
```

Using this code, on a mouse hover, the host element border will be set to a green, solid 5-pixel width. Therefore, using the @HostBinding decorator, you can set the properties of the host element in the directive class.