@HostBinding and @HostListener in Angular

Leave a Comment / 4 minutes of reading / March 9, 2023

← ngTemplateOutlet | Angular Tutorial | ViewChild,ViewChildren & QueryList →

The HostBinding & HostListener are decorators in Angular. HostListener listens to host events, while HostBinding allows us to bind to a property of the host element. The host is an element on which we attach our component or directive. This feature allows us to manipulate the host styles or take some action whenever the user performs some action on the host element.

Table of Contents

Host Element

The host element is the element on which we attach our directive or component. Remember components are directives with a view (template).

For Example

Consider the following `ttToggle` directive. We built this directive in our tutorial custom directive in Angular. We attach it to a `button` element. Here the `button` element is the host element.

```
<button ttToggle>Click To Toggle</button>
```

In the following example for the `apphighlight` directive, `p` element is the host element

```
<div>
  <p apphighlight>
    <div>This text will be highlighted</div>
  </p>
</div>
```

HostBinding

Host Binding binds a Host element property to a variable in the directive or component

HostBinding Example

The following `appHighLight` directive, uses the `HostBinding` on `style.border` property of the parent element to the `border` property.

Whenever we change the value of the `border`, the angular will update the `border` property of the host element.

```
import { Directive, HostBinding, OnInit } from '@angular/core'

@Directive({
  selector: '[appHighLight]',
})
export class HighLightDirective implements OnInit {

  @HostBinding('style.border') border: string;

  ngOnInit() {
    this.border="5px solid blue"
  }

}
```

We apply `appHighLight` directive to a host element (`p` in the example) as shown below.

```
<div>
  <h2>appHighLight Directive</h2>
  <p appHighLight>
    This Text has blue Border
  </p>
</div>
```

HostListener

HostListener Decorator listens to the DOM event on the host element. It also provides a handler method to run when that event occurs.

HostListener Example

For example, in the following code HostListener listens to the mouseover & mouseleave event. We use the HostListner decorator to decorate functions onMouseOver & onMouseLeave .

```
1
2  import { Directive, HostBinding, OnInit, HostListener } from '@angular/core'
3
4  @Directive({
5    selector: '[appHighLight]',
6  })
7  export class HighLightDirective implements OnInit {
8
9    @HostBinding('style.border') border: string;
10
11   ngOnInit() {
12   }
13
14   @HostListener('mouseover')
15   onMouseOver() {
16     this.border = '5px solid green';
17     console.log("Mouse over")
18   }
19
20   @HostListener('mouseleave')
21   onMouseLeave() {
22     this.border = '';
23     console.log("Mouse Leave")
24   }
25
26 }
27
```

We apply the directive on a host element (p in the example) as shown below.

Whenever the mouse is moved over the p element, the mouseover event is captured by the HostListener. It runs the onMouseOver method which we have attached to it. This method adds a green border to the p element using the HostBinding.

Similarly, on mouseleave event, the border is removed.

```
<div>
  <h2>appHighLight Directive</h2>
  <p appHighLight>
    This Text has blue Border
  </p>
</div>
```

Attaching Classes using HostBinding

Attaching a class to the host element is one of the common use cases of the HostBinding decorator.

For Example the following example adds the `highlight & box` class to the host element

```
  @HostBinding('class') class: string;

  ngOnInit() {
    this.class="highlight box"
  }
```

You can also Use the `getter` method.

```
  @HostBinding('class')  get class() {  return "highlight box"  }
```

Another example

```
  @HostBinding('class.highlight') get hasHighlight () { return true; }
  @HostBinding('class.box') get hasBox () { return false }
```

The Classes that HostBinding adds must exist in the scope of the host. i.e. both `highlight & box` must exist in the global styles or in the Component where we add the directive.

HostBinding & HostListner in Components

The components are nothing but directives with a template. Hence we can make use of both HostBinding & HostListner in components also.

The following is a `BoxComponent`, which applies the `highlight & box` classes to the host element. The classes `highlight & box` are defined in the component.

```
1
2  import { Component, HostBinding, HostListener } from '@angular/core';
3
4  @Component({
5    selector: 'app-box',
6    template: `
7      <h2> This is Box Component</h2> `,
8    styles: [
9      `
10     .highlight {
11       color:green;
12       display: block;
13     }
14
15     .box {
16       border: 1px dashed green;
17     }
18     `
19   ],
20
21
22 })
23 export class BoxComponent {
24   title = 'hostBinding';
25
26   @HostBinding('class.highlight') get hasHighlight() { return true; }
27   @HostBinding('class.box') get hasBox() { return true }
```

```
}

```

Now, open the app.component.ts and insert the above component.

```

<app-box></app-box>

```

Run the app and *you will not see any border neither the text is highlighted*. i.e. because the host element exists in the parent components (AppComponent) scope and not in the BoxComponent scope. So any CSS styles in the BoxComponent will have no effect

Open the app.component.css and add the styles. These styles are now applied correctly.

```

.highlight {
  color:blue;
  display: block;
}

.box {
  border: 1px solid red;
}

```

References

1. HostBinding API
2. HostListner API

Read More