@HostListener

`@HostListener` is a decorator in Angular that allows you to listen for events on the host element of a directive. It is commonly used with directives to define behavior based on user interactions or other events. Here's an explanation and example of using `@HostListener` in Angular:

Usage:

Let's say you want to create a directive that changes the background color of an element when the user hovers over it. You can use `@HostListener` to listen for the `mouseenter` and `mouseleave` events:

```
import { Directive, ElementRef, Renderer2, HostListener } from '@angular/core';

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

  constructor(private el: ElementRef, private renderer: Renderer2) {}

  @HostListener('mouseenter') onMouseEnter() {
    this.highlight('yellow');
  }

  @HostListener('mouseleave') onMouseLeave() {
    this.highlight(null);
  }

  private highlight(color: string | null) {
    this.renderer.setStyle(this.el.nativeElement, 'background-color', color);
  }
}
```

In this example:

- The `@Directive` decorator is used to define a directive.
- The `selector` property (`'[appHighlight]'`) specifies that this directive should be applied to elements with the `appHighlight` attribute.
- The `@HostListener` decorator is used to listen for the `mouseenter` and `mouseleave` events on the host element.
- The `onMouseEnter` method is called when the mouse enters the element, and it calls the `highlight` method to set the background color to yellow.
- The `onMouseLeave` method is called when the mouse leaves the element, and it calls the `highlight` method with `null` to reset the background color.

- The `highlight` method uses `Renderer2` to safely manipulate the DOM by setting the background color of the host element.

Now, you can use this directive in your templates:

```
<div appHighlight>
  Hover me to highlight!
</div>
```

When you hover over the element, the background color changes to yellow, and it returns to its original state when you move the mouse away.

This is a simple example, but `@HostListener` is versatile and can be used in various scenarios to respond to events on the host element of a directive.