Building an attribute directive

This section walks you through creating a highlight directive that sets the background color of the host element to yellow.

1. To create a directive, use the CLI command [ng generate directive](https://angular.io/cli/generate).

ng generate directive highlight

The CLI creates src/app/highlight.directive.ts, a corresponding test file src/app/highlight.directive.spec.ts, and declares the directive class in the AppModule.

The CLI generates the default src/app/highlight.directive.ts as follows:

src/app/highlight.directive.ts

import { [Directive](https://angular.io/api/core/Directive) } from '@angular/core';

@[Directive](https://angular.io/api/core/Directive)({

selector: '[appHighlight]'

})

export class HighlightDirective {

constructor() { }

}

The @[Directive](https://angular.io/api/core/Directive)() decorator's configuration property specifies the directive's CSS attribute selector, [appHighlight].

1. Import [ElementRef](https://angular.io/api/core/ElementRef) from @angular/core. [ElementRef](https://angular.io/api/core/ElementRef) grants direct access to the host DOM element through its nativeElement property.
2. Add [ElementRef](https://angular.io/api/core/ElementRef) in the directive's constructor() to [inject](https://angular.io/guide/dependency-injection) a reference to the host DOM element, the element to which you apply appHighlight.
3. Add logic to the HighlightDirective class that sets the background to yellow.

src/app/highlight.directive.ts

content\_copyimport { [Directive](https://angular.io/api/core/Directive), [ElementRef](https://angular.io/api/core/ElementRef) } from '@angular/core';

@[Directive](https://angular.io/api/core/Directive)({

selector: '[appHighlight]'

})

export class HighlightDirective {

constructor(el: [ElementRef](https://angular.io/api/core/ElementRef)) {

el.nativeElement.style.backgroundColor = 'yellow';

}

}

Directives *do not* support namespaces.

src/app/app.component.avoid.html (unsupported)

content\_copy<p app:Highlight>This is invalid</p>

Applying an attribute directive

1. To use the HighlightDirective, add a <p> element to the HTML template with the directive as an attribute.

src/app/app.component.html

content\_copy<p appHighlight>Highlight me!</p>

Angular creates an instance of the HighlightDirective class and injects a reference to the <p> element into the directive's constructor, which sets the <p> element's background style to yellow.

Handling user events

This section shows you how to detect when a user mouses into or out of the element and to respond by setting or clearing the highlight color.

1. Import [HostListener](https://angular.io/api/core/HostListener) from '@angular/core'.

src/app/highlight.directive.ts (imports)

content\_copyimport { [Directive](https://angular.io/api/core/Directive), [ElementRef](https://angular.io/api/core/ElementRef), [HostListener](https://angular.io/api/core/HostListener) } from '@angular/core';

1. Add two event handlers that respond when the mouse enters or leaves, each with the @[HostListener](https://angular.io/api/core/HostListener)() decorator.

src/app/highlight.directive.ts (mouse-methods)

content\_copy@[HostListener](https://angular.io/api/core/HostListener)('mouseenter') onMouseEnter() {

this.highlight('yellow');

}

@[HostListener](https://angular.io/api/core/HostListener)('mouseleave') onMouseLeave() {

this.highlight('');

}

private highlight(color: string) {

this.el.nativeElement.style.backgroundColor = color;

}

With the @[HostListener](https://angular.io/api/core/HostListener)() decorator, you can subscribe to events of the DOM element that hosts an attribute directive, the <p> in this case.

The handlers delegate to a helper method, highlight(), that sets the color on the host DOM element, el.

The complete directive is as follows:

src/app/highlight.directive.ts

content\_copy@[Directive](https://angular.io/api/core/Directive)({

selector: '[appHighlight]'

})

export class HighlightDirective {

constructor(private el: [ElementRef](https://angular.io/api/core/ElementRef)) { }

@[HostListener](https://angular.io/api/core/HostListener)('mouseenter') onMouseEnter() {

this.highlight('yellow');

}

@[HostListener](https://angular.io/api/core/HostListener)('mouseleave') onMouseLeave() {

this.highlight('');

}

private highlight(color: string) {

this.el.nativeElement.style.backgroundColor = color;

}

}

The background color appears when the pointer hovers over the paragraph element and disappears as the pointer moves out.



Passing values into an attribute directive

This section walks you through setting the highlight color while applying the HighlightDirective.

1. In highlight.directive.ts, import [Input](https://angular.io/api/core/Input) from @angular/core.

src/app/highlight.directive.ts (imports)

content\_copyimport { [Directive](https://angular.io/api/core/Directive), [ElementRef](https://angular.io/api/core/ElementRef), [HostListener](https://angular.io/api/core/HostListener), [Input](https://angular.io/api/core/Input) } from '@angular/core';

1. Add an appHighlight @[Input](https://angular.io/api/core/Input)() property.

src/app/highlight.directive.ts

content\_copy@[Input](https://angular.io/api/core/Input)() appHighlight = '';

The @[Input](https://angular.io/api/core/Input)() decorator adds metadata to the class that makes the directive's appHighlight property available for binding.

1. In app.component.ts, add a color property to the AppComponent.

src/app/app.component.ts (class)

content\_copyexport class AppComponent {

color = 'yellow';

}

1. To simultaneously apply the directive and the color, use property binding with the appHighlight directive selector, setting it equal to color.

src/app/app.component.html (color)

content\_copy<p [appHighlight]="color">Highlight me!</p>

The [appHighlight] attribute binding performs two tasks:

* + applies the highlighting directive to the <p> element
  + sets the directive's highlight color with a property binding

Setting the value with user input

This section guides you through adding radio buttons to bind your color choice to the appHighlight directive.

1. Add markup to app.component.html for choosing a color as follows:

src/app/app.component.html (v2)

<h1>My First [Attribute](https://angular.io/api/core/Attribute) [Directive](https://angular.io/api/core/Directive)</h1>

<h2>Pick a highlight color</h2>

<div>

<input type="radio" name="colors" (click)="color='lightgreen'">Green

<input type="radio" name="colors" (click)="color='yellow'">Yellow

<input type="radio" name="colors" (click)="color='cyan'">Cyan

</div>

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

1. Revise the AppComponent.color so that it has no initial value.

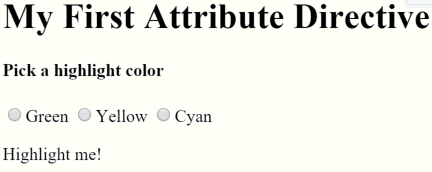
src/app/app.component.ts (class)

export class AppComponent {

color = '';

}

1. Serve your application to verify that the user can choose the color with the radio buttons.



Binding to a second property

This section guides you through configuring your application so the developer can set the default color.

1. Add a second [Input](https://angular.io/api/core/Input)() property to HighlightDirective called defaultColor.

src/app/highlight.directive.ts (defaultColor)

content\_copy@[Input](https://angular.io/api/core/Input)() defaultColor = '';

1. Revise the directive's onMouseEnter so that it first tries to highlight with the highlightColor, then with the defaultColor, and falls back to red if both properties are undefined.

src/app/highlight.directive.ts (mouse-enter)

content\_copy@[HostListener](https://angular.io/api/core/HostListener)('mouseenter') onMouseEnter() {

this.highlight(this.highlightColor || this.defaultColor || 'red');

}

1. To bind to the AppComponent.color and fall back to "violet" as the default color, add the following HTML. In this case, the defaultColor binding doesn't use square brackets, [], because it is static.

src/app/app.component.html (defaultColor)

content\_copy<p [appHighlight]="color" defaultColor="violet">

Highlight me too!

</p>

As with components, you can add multiple directive property bindings to a host element.

The default color is red if there is no default color binding. When the user chooses a color the selected color becomes the active highlight color.

