Directivas de atributos

Una directiva **Attribute** cambia la apariencia o el comportamiento de un elemento DOM.

Pruebe el Ejemplo de Directiva de atributos / ejemplo de descarga .

Descripción general de las directivas

Hay tres tipos de directivas en Angular:

1. Componentes: directivas con una plantilla.
2. Directivas estructurales: cambie el diseño DOM agregando y eliminando elementos DOM.
3. Directivas de atributo: cambie la apariencia o el comportamiento de un elemento, componente u otra directiva.

Los componentes son los más comunes de las tres directivas. Ha visto un componente por primera vez en el tutorial de introducción.

Las directivas estructurales cambian la estructura de la vista. Dos ejemplos son NgFor y NgIf. Aprenda sobre ellos en la guía Directivas estructurales.

Las directivas de atributo se utilizan como atributos de elementos. La directiva NgStyle integrada en la guía de sintaxis de plantilla, por ejemplo, puede cambiar varios estilos de elemento al mismo tiempo.

Cree una directiva de atributo simple

Una directiva de atributo requiere mínimamente la creación de una clase de controlador anotada con , que especifica el selector que identifica el atributo. La clase controller implementa el comportamiento de directiva deseado.`@Directive`

En esta página se muestra cómo crear una directiva de atributo *appHighlight* simple para establecer el color de fondo de un elemento cuando el usuario pasa el cursor sobre ese elemento. Puedes aplicarlo así:

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

```
<p appHighlight>Highlight me!</p>
```

Tenga en cuenta que las directivas *no* admiten espacios de nombres.

src/app/app.component.avoid.html (no compatible)

```
<p app:Highlight>This is invalid</p>
```

Escriba el código de directiva

Cree el archivo de clase de directiva en una ventana de terminal con el comando CLI `ng generate directive`.

```
ng generate directive highlight
```

La CLI crea , un archivo de prueba correspondiente y *declara* la clase de directiva en la raíz. `src/app/highlight.directive.ts` `src/app/highlight.directive.spec.ts` `AppModule`

> *Las directivas* deben declararse en módulos angulares de la misma manera que *los componentes.*

El generado es el siguiente: `src/app/highlight.directive.ts`

src/app/highlight.directive.ts

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

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

El símbolo importado proporciona Angular el decorador. `Directive` `@Directive`

La propiedad de configuración solitaria del decorador especifica el selector de atributos CSS de la directiva, .`@Directive` `[appHighlight]`

Son los corchetes () los que lo convierten en un selector de atributos. Angular localiza cada elemento de la plantilla que tiene un atributo denominado y aplica la lógica de esta directiva a ese elemento.`[]` `appHighlight`

El patrón *de selector de atributos* explica el nombre de este tipo de directiva.

¿Por qué no "resaltar"?

Aunque *highlight* sería un selector más conciso que *appHighlight* y funcionaría, la mejor práctica es prefijar nombres de selector para asegurarse de que no entren en conflicto con los atributos HTML estándar. Esto también reduce el riesgo de colisión con nombres de directivas de terceros. El CLI agregó el prefijo para usted.`app`

Asegúrese de **no** prefijar el nombre de la directiva con `ng` porque ese prefijo está reservado para Angular y usarlo podría causar errores que son difíciles de diagnosticar.`highlight`

Después de los metadatos viene la clase de controlador de la directiva, llamada , que contiene la lógica (actualmente vacía) para la directiva. La exportación hace que la directiva sea accesible.`@Directive` `HighlightDirective` `HighlightDirective`

Ahora edite el aspecto generado para que se vea de la siguiente manera:`src/app/highlight.directive.ts`

src/app/highlight.directive.ts

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

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
    constructor(el: ElementRef) {
       el.nativeElement.style.backgroundColor = 'yellow';
    }
}
```

La instrucción especifica un símbolo adicional de la biblioteca Angular:`import` `ElementRef` `core`

Utilice el constructor de la directiva para insertar una referencia al elemento DOM host, el elemento al que aplicó .`ElementRef` `appHighlight`

`ElementRef` concede acceso directo al elemento DOM host a través de su propiedad.`nativeElement`

Esta primera implementación establece el color de fondo del elemento host en amarillo.

Aplicar la directiva de atributos

Para utilizar el nuevo , agregue un elemento de párrafo () a la plantilla de la raíz y aplique la directiva como atributo.`HighlightDirective` `<p>` `AppComponent`

src/app/app.component.html

```
<p appHighlight>Highlight me!</p>
```

Ahora ejecute la aplicación para ver la acción.`HighlightDirective`

```
ng serve
```

En resumen, Angular encontró el atributo en el elemento **host.** Creó una instancia de la clase e inyectó una referencia al elemento en el constructor de la directiva que establece el estilo de fondo del elemento en amarillo.`appHighlight` `<p>` `HighlightDirective` `<p>` `<p>`

Responder a eventos iniciados por el usuario

Actualmente, simplemente establece un color de elemento. La directiva podría ser más dinámica. Podría detectar cuándo el usuario entra o sale del elemento y responde configurando o borrando el color de resaltado.`appHighlight`

Comience agregando a la lista de símbolos importados.`HostListener`

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

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

A continuación, agregue dos controladores de eventos que respondan cuando el mouse entre o se vaya, cada uno adornado por el decorador.`HostListener`

src/app/highlight.directive.ts (métodos de ratón)

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

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

private highlight(color: string) {
  this.el.nativeElement.style.backgroundColor = color;
}
```

El decorador le permite suscribirse a eventos del elemento DOM que hospeda una directiva de atributo, en este caso.`@HostListener` `<p>`

> Por supuesto, podría llegar al DOM con JavaScript estándar y adjuntar detectores de eventos manualmente. Hay al menos tres problemas con *ese* enfoque:
>
> 1. Tienes que escribir los oyentes correctamente.
> 2. El código debe *separar* el agente de escucha cuando se destruye la directiva para evitar pérdidas de memoria.
> 3. Hablar con la API de DOM directamente no es una práctica recomendada.

Los controladores delegan en un método auxiliar que establece el color en el elemento DOM host, .`el`

El método auxiliar, , se extrajo del constructor. El constructor revisado simplemente declara la inserción .`highlight` `el: ElementRef`

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

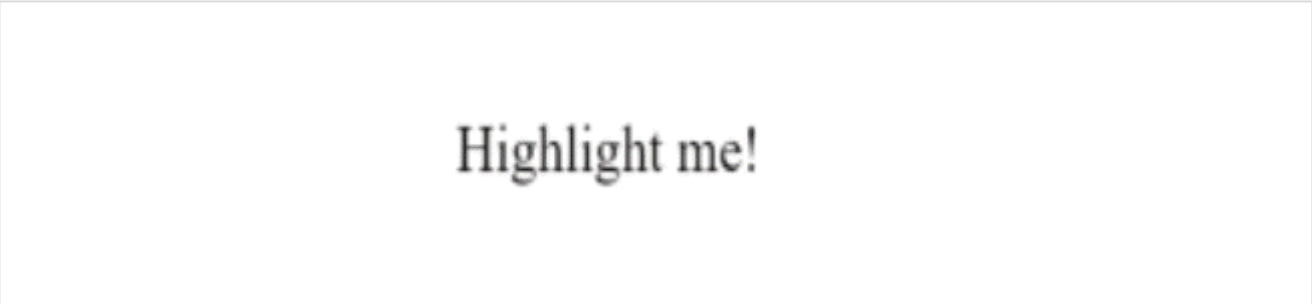
```
constructor(private el: ElementRef) { }
```

Aquí está la directiva actualizada en su totalidad:

src/app/highlight.directive.ts

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

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
  constructor(private el: ElementRef) { }

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

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

  private highlight(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
}
```

Ejecute la aplicación y confirme que el color de fondo aparece cuando el puntero se sitúa sobre el elemento de párrafo y desaparece a medida que el puntero se mueve hacia fuera.



Pasar valores a la directiva con un *enlace de* datos @Input

Actualmente, el color de resaltado está codificado de forma *rígida dentro* de la directiva. Eso es inflexible. En esta sección, se proporciona al desarrollador el poder de establecer el color de resaltado al aplicar la directiva.

Comience agregando a la lista de símbolos importados de . `Input` `@angular/core`

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

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

Agregue una propiedad a la clase de directiva de la siguiente manera:`highlightColor`

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

```
@Input() highlightColor: string;
```

Enlace a una propiedad *@Input*

Fíjate en el decorador. Agrega metadatos a la clase que hace que la propiedad de la directiva esté disponible para el enlace.`@Input` `highlightColor`

Se denomina propiedad *input* porque los datos fluyen de la expresión de enlace *a* la directiva. Sin esos metadatos de entrada, Angular rechaza el enlace; ver a continuación para obtener más información sobre eso.

Pruébelo agregando las siguientes variaciones de enlace de directiva a la plantilla:`AppComponent`

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

```
<p appHighlight highlightColor="yellow">Highlighted in yellow</p>
<p appHighlight [highlightColor]="'orange'">Highlighted in orange</p>
```

Agregue una propiedad al archivo .`color` `AppComponent`

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

```
export class AppComponent {
  color = 'yellow';
}
```

Deje que controle el color de resaltado con un enlace de propiedad.

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

```
<p appHighlight [highlightColor]="color">Highlighted with parent component's
color</p>
```

Eso es bueno, pero sería bueno aplicar *simultáneamente* la directiva y establecer el color *en el mismo atributo* como este.

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

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

El enlace de atributo aplica la directiva de resaltado al elemento y establece el color de resaltado de la directiva con un enlace de propiedad. Está reutilizando el selector de atributos de la directiva () para realizar ambos trabajos. Es una sintaxis nítida y compacta.`[appHighlight]``<p>` `[appHighlight]`

Tendrá que cambiar el nombre de la propiedad de la directiva a porque ahora es el nombre de enlace de propiedad de color.`highlightColor``appHighlight`

src/app/highlight.directive.ts (renombrado como selector de directivas de coincidencia)

```
@Input() appHighlight: string;
```

Esto es desagradable. La palabra, , es un terrible nombre de la propiedad y doesn't transmitir la intención de la propiedad.`appHighlight`

Enlazar a un alias *de @Input*

Afortunadamente, puede asignar a la propiedad de directiva el nombre que desee ***y asignarle alias*** con fines de enlace.

Restaure el nombre de propiedad original y especifique el selector como alias en el argumento en .`@Input`

src/app/highlight.directive.ts (propiedad de color con alias)

```
@Input('appHighlight') highlightColor: string;
```

Dentro de la directiva, la propiedad se conoce como . *Fuera* de la directiva, donde se enlaza a ella, se conoce como .`highlightColor``appHighlight`

Obtiene lo mejor de ambos mundos: el nombre de propiedad que desea y la sintaxis de enlace que desea:

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

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

Ahora que está enlazando a través del alias a la , modifique el método para usar esa propiedad. Si alguien descuida enlazar a , resalte el elemento host en rojo:`highlightColor` `onMouseEnter()` `appHighlight`

src/app/highlight.directive.ts (entrada del ratón)

```
@HostListener('mouseenter') onMouseEnter() {
  this.highlight(this.highlightColor || 'red');
}
```

Esta es la última versión de la clase de directiva.

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

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

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

  constructor(private el: ElementRef) { }

  @Input('appHighlight') highlightColor: string;

  @HostListener('mouseenter') onMouseEnter() {
    this.highlight(this.highlightColor || 'red');
  }

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

  private highlight(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
}
```

Escribe un arnés para probarlo

Puede ser difícil imaginar cómo funciona realmente esta directiva. En esta sección, se convertirá en un arnés que le permite elegir el color de resaltado con un botón de opción y enlazar su elección de color a la directiva.`AppComponent`

Actualice de la siguiente manera:`app.component.html`

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

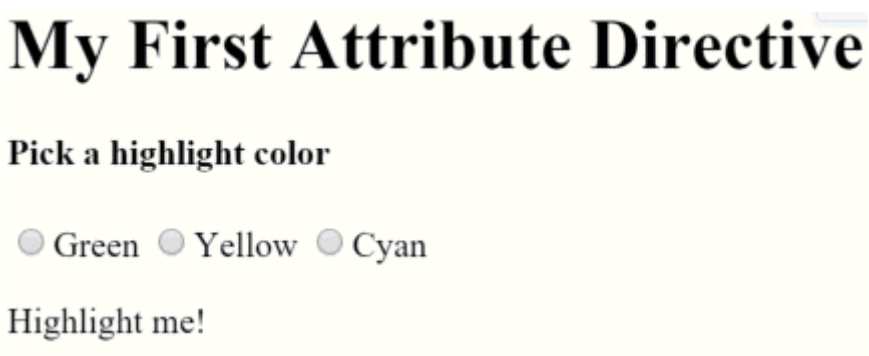
```
<h1>My First Attribute Directive</h1>

<h4>Pick a highlight color</h4>
<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>
```

Revise el para que no tenga ningún valor inicial.`AppComponent.color`

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

```
export class AppComponent {
  color: string;
}
```

Aquí están el arnés y la directiva en acción.



Enlazar a una segunda propiedad

Esta directiva de resaltado tiene una sola propiedad personalizable. En una aplicación real, puede necesitar más.

Por el momento, el color predeterminado (el color que prevalece hasta que el usuario elige un color de resaltado) está codificado de forma rígida como "rojo". Deje que el desarrollador de plantillas establezca el color predeterminado.

Agregue una segunda propiedad **de entrada** a la llamada:`HighlightDirective` `defaultColor`

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

```
@Input() defaultColor: string;
```

Revise la directiva para que primero intente resaltar con el , a continuación, con el , y vuelve a "rojo" si ambas propiedades son indefinidas.`onMouseEnter` `highlightColor` `defaultColor`

src/app/highlight.directive.ts (entrada del ratón)

```
@HostListener('mouseenter') onMouseEnter() {
  this.highlight(this.highlightColor || this.defaultColor || 'red');
}
```

¿Cómo se enlaza a una segunda propiedad cuando ya está enlazando al nombre de atributo? `appHighlight`

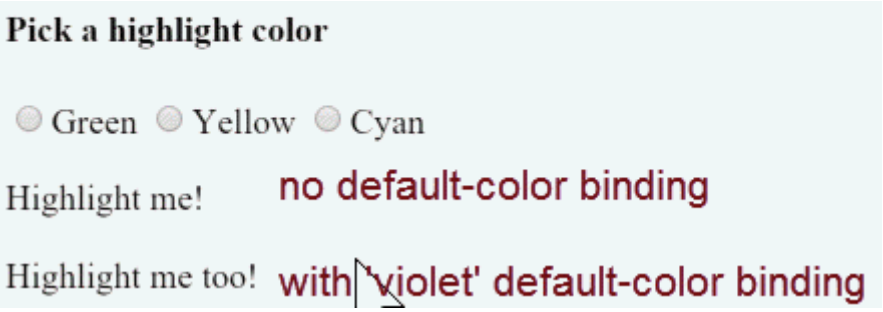
Al igual que con los componentes, puede agregar tantos enlaces de propiedad de directiva como necesite encadenarlos en la plantilla. El desarrollador debe ser capaz de escribir la siguiente plantilla HTML para enlazar a la y volver a "violeta" como el color predeterminado.`AppComponent.color`

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

```
<p [appHighlight]="color" defaultColor="violet">
  Highlight me too!
</p>
```

Angular sabe que la encuadernación pertenece a la porque lo hizo *público* con el decorador.`defaultColor` `HighlightDirective` `@Input`

Así es como el arnés debe funcionar cuando haya terminado de codificar.



Resumen

Esta página trata cómo:

- Cree una **directiva de atributo** que modifique el comportamiento de un elemento.
- Aplique la directiva a un elemento de una plantilla.
- Responda a **eventos** que cambien el comportamiento de la directiva.
- **Enlazar** valores a la directiva.

El código fuente final sigue:

app/app.component.ts *app/app.component.html* *app/highlight.directive.ts* *app/*

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  color: string;
}
```

También puede experimentar y descargar el Ejemplo de Directiva de atributos / ejemplo de descarga .

Apéndice: ¿Por qué añadir *@Input?*

En esta demostración, la propiedad es una propiedad de ***entrada*** del archivo . Lo has visto aplicado sin un alias:`highlightColor` `HighlightDirective`

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

```
@Input() highlightColor: string;
```

Lo has visto con un alias:

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

```
@Input('appHighlight') highlightColor: string;
```

De cualquier manera, el decorador indica a Angular que esta propiedad es *pública* y está disponible para el enlace por un componente primario. Sin , Angular se niega a enlazar a la propiedad.`@Input``@Input`

Ha enlazado la plantilla HTML a las propiedades de componente antes y nunca ha utilizado . ¿Qué es diferente?`@Input`

La diferencia es una cuestión de confianza. Angular trata la plantilla de un componente como *una pertenencia* al componente. El componente y su plantilla confían entre sí implícitamente. Por lo tanto, la propia plantilla del componente puede enlazarse a *cualquier* propiedad de ese componente, con o sin el decorador.`@Input`

Pero un componente o directiva no debe confiar ciegamente en *otros* componentes y directivas. Las propiedades de un componente o directiva se ocultan del enlace de forma predeterminada. Son *privados* desde una perspectiva de enlace angular. Cuando se adorna con el decorador, la propiedad se hace *pública* desde una perspectiva de enlace angular. Sólo entonces puede estar vinculado por algún otro componente o directiva.`@Input`

Puede saber si es necesario por la posición del nombre de propiedad en un enlace.`@Input`

- Cuando aparece en la expresión de plantilla a la ***derecha*** de los iguales (-), pertenece al componente de la plantilla y no requiere el decorador.`@Input`
- Cuando aparece **entre corchetes** ([ ]) a la **izquierda** de los iguales (-), la propiedad pertenece a algún *otro* componente o directiva; esa propiedad debe estar adornada con el decorador.`@Input`

Ahora aplique ese razonamiento al siguiente ejemplo:

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

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

- La propiedad de la expresión de la derecha pertenece al componente de la plantilla. La plantilla y su componente confían entre sí. La propiedad no requiere el decorador.`color` `color` `@Input`
- La propiedad de la izquierda hace referencia a una propiedad *con alias* de la propiedad , no a una propiedad del componente de la plantilla. Hay problemas de confianza. Por lo tanto, la propiedad de directiva debe llevar el decorador.`appHighlight` `HighlightDirective` `@Input`