Conociendo las directivas 17/20

RECURSOS MARCADORES

Angular utiliza el concepto de directiva para **cambiar la apariencia o el comportamiento de un elemento** en el HTML. Acá estaremos **conociendo las directivas**.

Tu primera directiva

Para crear tu primera directiva, es necesario usar el CLI de Angular con el comando `ng generate directive test-name` o en su forma corta `ng g d test-name` .

Al igual que con los servicios y los pipelines, el comando CLI creará un archivo `.ts` con el código de tu directiva y un archivo `.spec.ts` para sus respectivas pruebas unitarias.

El CLI también actualizará el archivo `app.module.ts` importando la directiva en las `declarations[]` . No olvides de asegurarte que se esté importando correctamente. De lo contrario, Angular no reconocerá la directiva.

Las directivas por defecto tienen el siguiente aspecto:

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

```
@Directive({
  selector: '[appTestName]'
})
export class TestNameDirective {
  constructor() { }
}
```

Utilizan el decorador `@Directive()` que contiene el nombre que utilizarás en el HTML para utilizar la directiva.

Manipulando estilos

En las directivas, puedes capturar el elemento HTML importando el servicio `ElementRef` y de esta manera poder cambiar los estilos de dicho elemento:

```
// directives/change-color.directive.ts
import { Directive, ElementRef } from '@angular/core';

@Directive({
  selector: '[appChangeColor]'
})
export class ChangeColorDirective {
  constructor(
    private element: ElementRef
  ) {
    this.element.nativeElement.style.color = 'blue';
```

```
  }
}
```

```
<div>
    <p appChangeColor>Texto color azul.</p>
</div>
```

Escuchando eventos

Otra posibilidad que ofrecen las directivas, es la escucha de eventos. Haciendo uso del decorador `@HostListener()` e importado desde `@angular/core` puedes ejecutar una función cada vez que se produce un clic, hover, scroll o cualquier otro evento.

```
// directives/change-color.directive.ts
import { Directive, ElementRef, HostListener } from '@angular/core';

@Directive({
  selector: '[appChangeColor]'
})
export class ChangeColorDirective {

  @HostListener('mouseenter') onMouseEnter() {
    this.element.nativeElement.style.color = 'blue';
  }
  @HostListener('mouseleave') onMouseLeave() {
```

```
    this.element.nativeElement.style.color = '';
  }

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

```
<div>
    <p appChangeColor>Texto color azul al hacer hover.</p>
</div>
```

Pasando datos a una directiva

Finalmente, si tienes la necesidad de que tu directiva reciba algún tipo de valor, lo mejor es apoyarte en el decorador que ya conoces `@Input()` .

```
// directives/change-color.directive.ts
import { Directive, Input, ElementRef } from '@angular/core';

@Directive({
  selector: '[appChangeColor]'
})
export class ChangeColorDirective {

  @Input() color!: string;
```

```
  constructor(
    private element: ElementRef
  ) {
    this.element.nativeElement.style.color = this.color;
  }
}
```

```
<div>
    <p appChangeColor [color]="'blue'">Texto color azul.</p>
</div>
<div>
    <p appChangeColor [color]="'red'">Texto color rojo.</p>
</div>
<div>
    <p appChangeColor [color]="'green'">Texto color verde.</p>
</div>
```

Puede ser algo difícil si recién estás comenzando con Angular imaginar un buen uso para una directiva propia. De momento, es importante saber que existen para poder implementarlas cuando llegue ese momento.

Ver código fuente del proyecto

Contribución creada con los aportes de Kevin Fiorentino.

Lecturas recomendadas

GitHub - platzi/angular-componentes at 14-step
https://github.com/platzi/angular-componentes/tree/14-step

Clases relacionadas

Creando la Directiva

Curso de Angular: Unit Testing para Componentes