






Angular 17


Tema 10. Directivas y pipes

Objetivos

- Conocer las directivas y pipes más usados
- Saber crear directivas y pipes propios

Contenidos

-  Pipes y directivas más utilizadas
-  Creación de pipes
-  Creación de directivas. Manipulación del DOM

-  ¿Y después?

Pipes

Un pipe es una función que modifica un valor en la plantilla del componente

Algunos habituales (muchos ya los hemos usado):

number	Formatea una expresión numérica
date	Formateo de fechas
lowercase, uppercase, titlecase	Mayúsculas, minúsculas, títulos
slice	Corta un texto o un array
keyvalue	Convierte un objeto en un array de objetos {key, value}
async	Se suscribe a una promesa o un observable (sólo para pruebas)
json	Muestra los campos y valores de un JSON

La mayoría están definidos en «CommonModule».

```
{{p.title | uppercase | slice:0:20 }}
```

```
@for(elem of producto | keyvalue; track $index) {  
  {{elem.key}} = {{elem.value}} <br>  
}
```

```
{{p.price | number:'1.2-2':'es'}}
```

Directivas de atributo

Elementos de Angular que se aplican a las etiquetas de HTML como un atributo estándar. Modifican el aspecto, contenido o comportamiento de la etiqueta.

Una **directiva** se sobrentiende **de atributo**. Tradicionalmente existían tres tipos de directivas: componentes, directivas estructurales y de atributo, pero ya no se habla de ese modo.

Directivas de atributo habituales:

ngClass Activa o desactiva clases (CommonModule)

ngStyle Permite definir estilos de CSS usando el estado del componente (CommonModule)

ngForm Define un formulario dirigido por plantilla (FormsModule)

ngModel Permite usar «two-way-binding» en un control de formulario (FormsModule)

```
<p [ngClass]="{estiloUno:activadoUno, estiloDos:valor}>100">  
<p [ngStyle]="{'fontWeight':negrita, 'textDecoration':subrayado, 'fontStyle':cursiva}">  
<input type="text" [(ngModel)]="campoNombre">
```

Pipes personalizados

Son clases de TypeScript decoradas con **@Pipe** y que implementan la interfaz **PipeTransform**:

```
@Pipe({
  name: 'elevar',
  standalone: true
})
export class ElevarPipe implements PipeTransform {
  transform(valor: number, ...argumentos: number[]): number {
    if (argumentos.length!=1) throw new Error("Es obligatorio indicar el exponente")
    return Math.pow(valor, argumentos[0]);
  }
}
```

<p>Siete al cubo vale {{7 | elevar:3}}</p>

Lo más habitual es usar Angular CLI para crear los pipes:

```
ng g p elevar
ng generate pipe elevar
```

```
@Pipe({
  name: 'elevar',
  standalone: true
})
export class ElevarPipe implements PipeTransform {
  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }
}
```

Ejercicio 10 A

Escribe una modificación del pipe «traducir» del vídeo anterior. Debe poder recibir tanto números como textos compuestos de cifras, que convertirá a una «cadena de nombres de números»:

```
<p>El texto "1230": {{'1230' | traducir}}</p>
<p>El número 7136: {{7136 | traducir}}</p>
<p>El texto "MAL1230": {{'MAL1230' | traducir}}</p>
```

```
El texto "1230": uno - dos - tres - cero
El número 7136: siete - uno - tres - seis
El texto "MAL1230": error
```

Quiero que admita un parámetro «h» que muestre el resultado traduciéndolo todo a «hexadecimal», por ejemplo usando «*let texto=numero.toString(16)*»:

```
<p>El texto "1230": {{'1230' | traducir:'h'}}</p>
<p>El número 7136: {{7136 | traducir:'h'}}</p>
<p>El texto "MAL1230": {{'MAL1230' | traducir:'h'}}</p>
```

```
El texto "1230": cuatro - C - E
El número 7136: uno - B - E - cero
El texto "MAL1230": error
```

Directivas de atributo personalizadas

Son clases de TypeScript decoradas con **@Directive**. Se usan para modificar el aspecto o el comportamiento del elemento de HTML sobre el que se aplican. Tienen mucho en común con los componentes (son directivas).

```
@Directive({
  selector: '[mayusculas]',
  standalone: true
})
export class MayusculasDirective implements OnInit {
  @Input() mayusculas=true;
  private elemento=inject(ElementRef);
  private renderizador=inject(Renderer2);

  ngOnInit(): void {
    let texto=this.elemento.nativeElement.textContent;
    if (this.mayusculas) texto=texto.toUpperCase();
    else texto=texto.toLowerCase();
    this.renderizador.setProperty(this.elemento.nativeElement, "textContent", texto);
  }
}
```

```
<p [mayusculas]="true">Ejemplo de aplicación de la directiva "Mayúsculas" (true)</p>
<p [mayusculas]="false">Ejemplo de aplicación de la directiva "Mayúsculas" (false)</p>
```

EJEMPLO DE APLICACIÓN DE LA DIRECTIVA "MAYÚSCULAS" (TRUE)

ejemplo de aplicación de la directiva "mayúsculas" (false)

Se suele utilizar Angular CLI para crear las directivas:

```
ng g d mayusculas
ng generate directive mayusculas
```


Ejercicio 10 B

Crea la directiva **aumentar**, que incrementa el tamaño de letra de una etiqueta cada vez que haces «click» sobre ella y que lo restaura al hacer «dblclick». Por simplificar, suponemos que siempre se le pasa el tamaño inicial en puntos:

```
<p [aumentar]='12pt'>
  Este párrafo aumentará de tamaño cada vez que hagas click sobre él.
</p>
```

Escribe también la directiva **rotar**. Cada vez que hagas «click» sobre la etiqueta incrementará el giro de la misma en la cantidad indicada en la directiva:

```
<p>
  La imagen rotará sobre su centro cada vez que le hagas click.<br/>
  
</p>
```

Quiero que la etiqueta gire sobre su eje. Para conseguirlo puedes usar los estilos:

```
transform-origin:50% 50%;
transform: rotate(45deg);
```

¿Qué hemos aprendido?

- Recordatorio de pipes y directivas estándares
- Creación de pipes
- Creación de directivas de atributo

Resumen de comandos

npm

npm install [-g] paquete

npm install

npm uninstall paquete

npm -version

ng (1)

ng version

ng help

ng serve [-o]

ng build

ng new nombre_proyecto

--routing false

--skip-tests

--skip-git

--no-standalone

ng (2)

ng generate component nombre_componente / ng g c

ng generate service nombre_servicio / ng g s

ng generate module nombre_módulo / ng g m

ng generate pipe nombre_pipe / ng g p

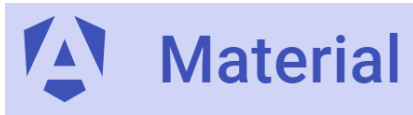
ng generate directive nombre_directiva / ng g d

Temas adicionales de Angular

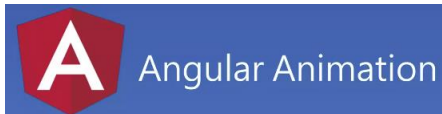
- Hojas de estilo con **SASS/SCSS**



- Componentes de **Angular Material**



- Animaciones con **Angular Animations**



- **Pruebas unitarias** con Angular



- **JavaScript** estándar

