

Welcome to the fullStack developer bootcamp



Module frontend with angular - class 3


✨ Today's agenda - Feb 5th ✨


Table of contents

- [Repository](#)
- 🦖 [Angular directives](#)
- 🖐️ [Angular pipes](#)
- 🕸️ [Angular routing](#)
- [Let's review](#)
- [Thank you](#)


Repository

dfloresgonz/**angular-bootcamp**




 1


Contributor

 0


Issues


 0


Stars

 0

Forks



 GitHub



GitHub - dfloresgonz/angular-bootcamp

Contribute to dfloresgonz/angular-bootcamp development by creating an account on GitHub.

<https://github.com/dfloresgonz/angular-bootcamp/>



Angular directives



Permiten crear interfaces dinámicas y reutilizables

```
// Directiva *ngIf para renderizado condicional
@Component({
  template: `
    <div *ngIf="isVisible">
      Este contenido aparece solo si isVisible es true
    </div>

    <!-- También podemos usar else -->
    <div *ngIf="isLoggedIn; else loginTemplate">
      Bienvenido usuario
    </div>
    <ng-template #loginTemplate>
      Por favor, inicia sesión
    </ng-template>
  `,
})
export class MiComponente {
  isVisible = true;
  isLoggedIn = false;
}
```

```
@Component({
  template: `
    <!-- Iteración básica -->
    <ul>
      <li *ngFor="let item of items">{{ item }}</li>
    </ul>

    <!-- Con índice y otras variables -->
    <div *ngFor="let user of users;
      let i = index;
      let isFirst = first;
      let isLast = last">
      Usuario {{i + 1}}: {{user.name}}
      <span *ngIf="isFirst">(Primer usuario)</span>
      <span *ngIf="isLast">(Último usuario)</span>
    </div>
  `,
})
export class ListaComponent {
  items = ['Manzana', 'Naranja', 'Plátano'];
  users = [
    { name: 'Ana' },
    { name: 'Juan' },
    { name: 'María' }
  ];
}
```

```
// Directiva ngClass para manipular clases CSS
@Component({
  template: `
    <div [ngClass]="{'active': isActive,
      'error': hasError,
      'highlight': isHighlighted}">
      Contenido con clases dinámicas
    </div>

    <!-- ngStyle para estilos inline -->
    <div [ngStyle]="{'color': textColor,
      'font-size': fontSize + 'px',
      'background-color': bgColor}">
      Estilos dinámicos
    </div>
  `,
})
export class EstilosComponent {
  isActive = true;
  hasError = false;
  isHighlighted = true;
  textColor = 'blue';
  fontSize = 16;
  bgColor = '#f0f0f0';
}
```

```
// Componente con múltiples directivas
@Component({
  template: `
    <div class="task-list">
      <!-- Usar ngIf para mostrar mensaje cuando no hay tareas -->
      <div *ngIf="tasks.length === 0" class="empty-message">
        No hay tareas pendientes
      </div>

      <!-- Lista de tareas con ngFor y múltiples directivas -->
      <div *ngFor="let task of tasks; let i = index"
        [ngClass]="{'completed': task.completed,
          'urgent': task.priority === 'alta'}"
        [ngStyle]="{'opacity': task.completed ? '0.6' : '1'}"
        appResaltado="lightyellow">

        <span class="task-number">{{i + 1}}.</span>
        <span class="task-title">{{task.title}}</span>

        <!-- Botones condicionales -->
        <button *ngIf="!task.completed"
          (click)="completeTask(task)">
          Completar
        </button>

        <button *ngIf="task.completed"
          (click)="undoComplete(task)">
          Deshacer
        </button>
      </div>
    </div>
  `,
})
export class TaskListComponent {
  tasks = [
    { title: 'Comprar víveres', completed: false, priority: 'normal' },
    { title: 'Pagar facturas', completed: false, priority: 'alta' },
    { title: 'Llamar al médico', completed: true, priority: 'alta' }
  ];

  completeTask(task: any) {
    task.completed = true;
  }

  undoComplete(task: any) {
    task.completed = false;
  }
}
```

```
@Component({
  template: `
    <div [ngSwitch]="userRole">
      <div *ngSwitchCase="'admin'">
        Panel de administrador
      </div>
      <div *ngSwitchCase="'editor'">
        Panel de editor
      </div>
      <div *ngSwitchCase="'user'">
        Panel de usuario
      </div>
      <div *ngSwitchDefault>
        Acceso denegado
      </div>
    </div>
  `,
})
export class PanelComponent {
  userRole = 'admin';
}
```



Angular pipes



Los pipes son herramientas para transformar datos directamente en las plantillas. Se usan con el operador `|`.

```
{{ valor | uppercase }}: Convierte el texto a mayúsculas.
{{ valor | lowercase }}: Convierte el texto a minúsculas.
{{ valor | date:'dd/MM/yyyy' }}: Formatea una fecha.
{{ valor | currency:'USD' }}: Formatea un número como moneda.
{{ valor | json }}: Convierte un objeto a formato JSON.
```

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'miPipe' })
export class MiPipe implements PipeTransform {
  transform(value: string): string {
    return value.split('').reverse().join('');
  }
}
```

```
{{ 'Hola' | miPipe }} <!-- Resultado: "aloH" -->
```

```
<div *ngIf="usuarios.length > 0">
  <ul>
    <li *ngFor="let usuario of usuarios">
      {{ usuario.nombre | uppercase }} - {{ usuario.fechaRegistro | date:'dd/MM/yyyy' }}
    </li>
  </ul>
</div>
```

```
@Component({
  template: `
    <!-- DatePipe - Formatea fechas -->
    <p>Fecha: {{ fecha | date }}</p>
    <p>Fecha personalizada: {{ fecha | date:'dd/MM/yyyy' }}</p>
    <p>Hora: {{ fecha | date:'shortTime' }}</p>

    <!-- UpperCasePipe y LowerCasePipe - Transformación de texto -->
    <p>Mayúsculas: {{ nombre | uppercase }}</p>
    <p>Minúsculas: {{ nombre | lowercase }}</p>

    <!-- DecimalPipe - Formato de números -->
    <p>Número: {{ precio | number:'1.2-2' }}</p>
    <p>Moneda: {{ precio | currency:'EUR' }}</p>
    <p>Porcentaje: {{ valor | percent:'1.0-2' }}</p>

    <!-- SlicePipe - Recorta arrays o strings -->
    <p>Texto recortado: {{ texto | slice:0:10 }}</p>
    <p>Array recortado: {{ array | slice:1:3 }}</p>

    <!-- JsonPipe - Útil para debugging -->
    <pre>{{ objeto | json }}</pre>
  `,
})
export class EjemploPipesComponent {
  fecha = new Date();
  nombre = 'Juan Pérez';
  precio = 23.5678;
  valor = 0.8543;
  texto = 'Este es un texto largo de ejemplo';
  array = [1, 2, 3, 4, 5];
  objeto = { nombre: 'Juan', edad: 25 };
}
```

```
// Pipe personalizado para filtrar una lista
@Pipe({
  name: 'filtrar'
})
export class FiltrarPipe implements PipeTransform {
  transform(items: any[], searchText: string): any[] {
    if (!items) return [];
    if (!searchText) return items;

    searchText = searchText.toLowerCase();

    return items.filter(item => {
      return item.nombre.toLowerCase().includes(searchText);
    });
  }
}

// Componente que usa el pipe personalizado
@Component({
  template: `
    <input [(ngModel)]="busqueda" placeholder="Buscar...">

    <ul>
      <li *ngFor="let item of items | filtrar:busqueda">
        {{ item.nombre }}
      </li>
    </ul>
  `,
})
export class ListaComponent {
  busqueda = '';
  items = [
    { nombre: 'Manzana' },
    { nombre: 'Banana' },
    { nombre: 'Naranja' }
  ];
}
```

```
@Component({
  template: `
    <!-- Encadenamiento de pipes -->
    <p>{{ texto | lowercase | titlecase }}</p>
    <p>{{ precio | currency:'EUR' | uppercase }}</p>

    <!-- Pipes con parámetros -->
    <p>{{ fecha | date:'fullDate' | uppercase }}</p>
  `,
})
export class PipesEncadenadosComponent {
  texto = 'HOLA MUNDO';
  precio = 99.99;
  fecha = new Date();
}
```



Angular routing



Mecanismo que permite a una app web dirigir y gestionar las vistas que se muestran al usuario en función de la URL o la ubicación dentro del sitio web.

- Basic routing
- Route parameters
- Lazy loading
- Route guards
- Child Routes



Let's review

Did you learn?



Thank you

Thanks for paying attention!