Imagina

Curso de Angular 14

Enrique de la Calle Santa Ana

29 Febrero 2,024

Índice

Índice

01

Introducción a Angular

Introduccion Angular

- Introducción:
- https://docs.angular.lat/
- Ventajas y desventajas
- Novedades sobre A14
- Creación de entorno trabajo
- https://angular.io/guide/setup-local
- Creamos nuestra primera App
- Analizamos los ficheros

Revisión de puntos

- ☐¿Por qué Angular?
- ☐ Ventajas y desventajas principales
- ☐ Características de Angular 14.2
- ☐ Diferencias más destacables frente a las versiones actuales de Angular
- ☐¿Qué es el renderizador de Angular?
- ☐ Instalación de Angular CLI a través de NPM
- ☐ Diferenciando entre instalaciones locales y globales de NPM
- ☐¿Qué es NPX y por qué usarlo?
- ☐ Creación de tu primer proyecto Angular ☐

Revisión de puntos

- ☐ Análisis de la estructura del proyecto y sus archivos de configuración
- ☐ Desplegando nuestra aplicación localmente con Angular CLI

02

Comandos Ng-cli

Revisión de puntos

- ☐ Introducción Comandos CLI esenciales
- ☐ ng new
- ☐ ng serve
- ☐ ng generate
- ☐ ng add
- ☐ ng build
- ☐ ng update
- ☐ Otros comandos existentes

03

Componentes / Material

Links Interés

- https://codingpotions.com/angular-componentes
- https://ngchallenges.gitbook.io/project/componentes
- https://codingpotions.com/angular-comunicacion-componentes/
- https://platzi.com/clases/2486-angular-componentes/41180-ciclo-de-vida-de-componentes/#:~:text=Ciclo%20de%20vida%20en%20Angular,los%20inputs%20en%20todo%20momento
- https://material.angular.io/guide/getting-started

Componentes

- Componente es un decorador Typescript (personaliza visual)
- Componente como unidad visual mínima en Angular
- Respeta MVC (template = vista = html, css) , (Modelo = prop clase) , (Controlador = Typescript o clase)
- Metadatos en un componente (selector , templateUrl , styleUrl)
- Recomendación:
- las Clases CamelCase y terminadas en Component (Si es componente)
- El selector en Minuscula y separado por guiones (html no dife may)

Componentes

- Creación Componente
- Se puede generar a mano creando todos los ficheros o
- Ng generate component components/contador
- Revisamos los ficheros generados
- Enlace a datos
- {{cliente.nombre}} interpolación solo en un sentido
- [property] = “value” Propiedad o atributo
- (event) = “handler” Evento o controlador
- [(ng-model)] = “property” propiedad (two binding)

Practica Componentes

- Creamos un proyecto nuevo Componentes / No Router / CSS
- C:\Ang14\npx ng new 03_Componentes
- C:\Ang14\npx generate component components/contador
- Revisamos la estructura y vemos como app.module.ts se ajusta
- Vemos la jerarquía de estilos css en cada parte de la app
- Incorporo en la pagina app.component.html el contador (varios fijos)
- Le paso el atributo mediante [] en varios contadores
- `@Input() titulo:string = 0;`

Practica Componentes

- En app.component.ts incorporamos un array de contadores

```
interface ContadorInterface {
 id:number,
 title:string,
 inicio:number,
 valor:number
}
aContadores:ContadorInterface[] = [
   {id:1, title:"Contador 1", inicio: 0  },
   {id:2, title:"Contador 2", inicio: 5  },
   {id:3, title:"Contador 3", inicio: 4 },
 ]
```

Practica Componentes

- Quiero ver el array de contadores en la app (introduzco el concepto pipe | json)
- Prueba {{ aContadores | json }}
- Adapto la estructura con un bucle json
- <table border="1">
- <thead>
- <tr>
- <th *ngFor="let contador of aContadores" >{{contador.title}}</th>
- </tr>
- </thead>
- <tbody>
- <tr>
- <td *ngFor="let contador of aContadores" ><app-contador></app-contador></td>
- </tr>
- </tbody>
- </table>

Practica Componentes

- He introducido el concepto directiva de estructura (*ngFor="let contador of aContadores)
- Directiva de Atributo (modifican las características de un componente como title ,etc [])
- Directivas de estructura (modifican el dom *ngFor ,etc)
- Directivas de plantilla que son los componentes en si
- Le pasamos como directivas de atributo , ID , title , contador y presentamos en el contador su valor
- Incorporamos un timer: (podríamos hacer un timerinterval de JS pero aprovechamos y ponemos)
- `import { interval, Observable, Subscription } from 'rxjs';`
- `const unseg:Observable<number> = interval(1000);`
- Adelantamos la comunicación Bidireccional con ngModel
- app.module: `import { FormsModule } from '@angular/forms'`
- App.component: `<input [ngModel]="currentItem.name" (ngModelChange)="currentItem.name=$event" id="example-change">`

Imagina

Codigo contador.component.ts

```
import { Component, Input, Output, OnChanges, OnInit, OnDestroy,   SimpleChanges, EventEmitter } from '@angular/core';
import { interval, Observable, Subscription } from 'rxjs';
const unseg:Observable<number> = interval(1000);
@Component({  selector: 'app-contador',  templateUrl: './contador.component.html',  styleUrls: ['./contador.component.css']})
export class ContadorComponent implements OnInit, OnDestroy, OnChanges {
   @Input() id:number = 0;  @Input() valor:number = 0;  @Input() parentMessage:string = '';
   @Output() messageEvent = new EventEmitter<string>();
   obs?: Subscription;
   constructor() {
            this.obs =  unseg.subscribe(x=>{
                        console.log("temporizador un seg", x)
                        this.valor++;
                        this.messageEvent.emit(`Papa: soy ${this.id} con el valor ${this.valor} ` );
             })
    }
```

Imagina

Codigo contador.component.ts

```
ngOnInit(): void {
                    console.log("init elemento")
}
ngOnDestroy() {
  console.log("destruyo elemento")
  this.obs?.unsubscribe();
}
ngOnChanges(changes: SimpleChanges) {
   console.log("cammbios ", changes)
}
ngDoCheck(){
 //console.log("do check")
}
}
```

Ciclo de Vida del componente

https://platzi.com/clases/2486-angular-componentes/41180-ciclo-de-vida-de-componentes/#:~:text=Ciclo%20de%20vida%20en%20Angular,los%20inputs%20en%20todo%20momento

- constructor
- ngOnChanges
- ngOnInit
- ngDoCheck
- ngAfterContentInit
- ngAfterContentChecked
- ngAfterViewInit
-
- ngOnDestroy

Revisión de puntos

- **3. COMPONENTES**
  - ☐ Metadatos de componentes
  - ☐ Creación de un componente
  - ☐ Instanciando componentes en archivos HTML
  - ☐ Introducción al ngModel
  - ☐ Data binding
  - ☐ Operador de coalescencia nula
  - ☐ Anidado de componentes
  - ☐ Pasando datos al componente a través de @Inputs
  - ☐ Respondiendo a eventos con @Outputs
  - ☐ Ciclo de Vida de los componentes
  - ☐¿Cuándo usar el ciclo de vida de los componentes en aplicaciones reales?
  - ☐ Aplicando estilos a los componentes desde una hoja CSS o SCSS

03 (Bis)

Material

Angular Material

Librería Grafica (Google) para incorporar a nuestros Proyectos

Alternativas: BootStrap , PrimeNg , etc

- Mkdir c:\Ang14\demo_material
- Cd , copio proto0 y npm i
- Npx ng add @angular/material@14 (Trampa Aviso)
- Revisamos app.module

```
import { MatSlideToggleModule } from '@angular/material/slide-toggle';
            @NgModule ({
              imports: [
                MatSlideToggleModule,
              ]
            })
class AppModule {}
```

Imagina

Angular Material (Cont. Y Schematics)

- App.component.html
- <mat-slide-toggle>Toggle me!</mat-slide-toggle>
- Angular Material tiene distintos templates y macros
- Un ejemplo seria un marco de navegación
- ng generate @angular/material:navigation navega
- (Si error versión)
- “npx ng update @angular/material@14”

Revisión de puntos

- **3. COMPONENTES**
  - ☐ Introducción a Angular Material como framework de componentes
  - ☐ Breve introducción a los componentes más destacables de Angular Material
  - ☐ Instalación y configuración de Angular Material en un proyecto Angular
  - ☐¿Qué son los schematics?
  - ☐ Comentarios acerca de los schematics de Angular Material
  - ☐ Empleando componentes de Angular Material en un proyecto Angular

04

Directivas/Modulos

Links

- https://medium.com/notasdeangular/directivas-en-angular-efb8a8cf78e0 (Directivas)
- https://codigoencasa.com/que-son-las-directivas-en-angular/ (Directivas)
- https://www.freecodecamp.org/espanol/news/como-usar-y-crear-directivas-personalizadas-en-angular/#:~:text=%C2%BFQu%C3%A9%20es%20una%20directiva%20angular,adjuntar%20comportamientos%20personalizados%20al%20DOM
- https://www.tutorialesprogramacionya.com/angularya/detalleconcepto.php?punto=14&codigo=14&inicio=0 (Módulos)
- https://www.youtube.com/watch?v=KzTAvLxN60U (Módulos)
- https://angular.io/guide/file-structure (Estructura carpetas y multiproyectos)
- https://material.angular.io/guide/getting-started (Instalación Angular Material)
- https://ng-bootstrap.github.io/#/getting-started (Instalacion Bootstrap)

Directivas

- Directivas son funciones que el renderizador ejecutara cuando las encuentre
- Las directivas pueden ser de angular o definidas por nosotros
- Directivas de atributo
- Manipulan la apariencia y el comportamiento (p.e. ngClass , ngStyle , ngMode l)
- Directivas estructurales

  Cambian la estructura del DOM , bucles , condicionales , etc

  Comienzan siempre por un asterisco (*NgIf , * NgFor , etc)
- Directivas de componente (Componentes)

  Es una directiva que proporciona el factor visual de nuestra clase/objeto

```
@Component({selector: 'app-contador',templateUrl:
'./contador.component.html’,   styleUrls: ['./contador.component.css’]
})
export class ContadorComponent implements OnInit, OnDestroy, OnChanges {}
```

Imagina

Directivas

- *ngIf
- *ngFor
- ngSwitch / * ngSwitchCase
- *ngPlural
- ngTemplate
- ngComponentOutlet

Directivas

- Creación de una directiva
- ng generate directive

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

@Directive({ selector: '[Mifondo]’})
export class MifondoDirective {
            constructor(private elementRef:ElementRef) {
                        elementRef.nativeElement.style.background = 'red';
            }}
```

- Pasar Parámetros a Directiva
- Añade @ Input() en la clase directiva con el mismo nombre que la directiva (@Input() highlight;) y pasar el valor así Highlight Directive
- Añade @ Input() en la clase directiva con cualquier nombre de variable (@Input() colorName;) y pasar el valor así Highlight Directive

Imagina

Revisión de puntos

- **4. DIRECTIVAS**
  - ☐ Directivas de atributo
  - ☐ Buenas prácticas en el uso de directivas de atributo
  - ☐ Directivas estructurales (condicionales y bucles)
  - ☐ Creación de directivas personalizadas
  - ☐ Buenas prácticas en el uso de directivas estructurales

Modulos

- https://www.tutorialesprogramacionya.com/angularya/detalleconcepto.php?punto=14&codigo=14&inicio=0
- Modularizar , librerías , etc. Lo veremos mas en profundidad en ejerc.
- Agrupamiento de recursos en un modulo
- npx ng generate module mimodulo
- Genera una carpeta mimodulo con los ficheros correspondientes
- Para generar un componente en ese modulo
- npx ng generate component mimodulo/contador

Revisión de puntos

- **3. (BIS) MODULOS**
  - ☐¿Qué son los módulos?
  - ☐ La organización de un proyecto mediante módulos
  - ☐ Creación de módulos en un proyecto Angular
  - ☐ Inyección de dependencias en Angular
  - ☐ Creación de una plantilla HTML inicial

05

Formularios Reactivos

05 Forms y Reactive Forms

- Definición
  - Un formulario es un control o conjunto de controles que esperan la interacción del usuario
  - Los formularios en Angular están construidos sobre el formulario HTML estándar, para ayudarte a crear controladores personalizados y simplificar la experiencia de validación de campos
- Pueden ser de dos tipos
  - Basado en plantillas
    - Son los basados principalmente en los controles del HTML
    - Usamos el componente ngModel para enlazar bidireccionalmente
    - TIPS:
      - Si lo encapsulamos en un componente standard form hay que detallar “name”.
      - No olvidar import { FormsModule } from '@angular/forms' en app.module
    - Ejercicio 1
      - Incorporamos un simple input con la directiva ngModel , vemos como el dato se transmite en ambas direcciones

05 Forms y Reactive Forms

- Reactivos
  - Mas robusto, mas escalable , incorporamos funcionalidades de validación
  - Incorpora componentes nuevos como FormControl , FormGroup , Validación , etc que nos ayuda a tener un mayor control sobre el formulario
  - TIPS:
    - No olvidar import { ReactiveFormsModule } from '@angular/forms' en app.module
  - Ejercicio 2
    - Igual que el ejercicio 1 pero demostrando que tiene mas funcionalidad al usar el componente FormControl

- Con reactive Forms podemos agrupar controles
  - Un solo control es fácilmente gestionable con FormControl, pero quizás nos interese tener un agrupamiento de controles
  - Los agrupamos y tenemos un objeto values con toda la info
  - Ejercicio 3
    - Agrupo componentes e incluyo select , check , radio

05 Forms y Reactive Forms

- Validaciones
  - Reactive Forms tiene componentes de validación llamado “Validators”
  - Se incluyen en el Formcontrol para verificar su contenido
  - Validacion simple como un require o un email en Ejercicio 4
  - Se pueden incluir mas de una condición en un array de Validators . p.e. required y mayor que 10 , etc
  - Hay múltiples validadores
    - Valor Requerido
    - Numéricos , Max , Min,
    - Texto: formato email , min long , max long , pattern(Expresión regular)
    - Se pueden agrupar en un Validator.compose
  - Ejercicio 5
    - Validaciones múltiples
    - Aprovecho el ciclo de vida para que si cambia un dato , rechequea la validación
    - Utilizo un compose con pattern para solo números y varias mas

Revisión de puntos

- **5. FORMULARIOS REACTIVOS**
  - ☐ Introducción a los Formularios Reactivos
  - ☐ Creación de Formularios Reactivos
  - ☐ Introducción a la validación de campos en formularios reactivos
  - ☐ Introducción a FormControl y FormGroup
  - ☐ Validando campos obligatorios
  - ☐ Validando campos numéricos
  - ☐ Validando campos booleanos
  - ☐ Introducción a FormArray
  - ☐ Validando campos de tipo lista

Revisión de puntos

- **5. FORMULARIOS REACTIVOS**
  - ☐ Validando expresiones regulares
  - ☐ Accediendo a los errores
  - ☐ Mostrar mensajes de error de validación
  - ☐ Anidación de validaciones
  - ☐ Validando que dos campos sean iguales
  - ☐ Comprobando el estado del formulario
  - ☐ Accediendo al contenido del formulario
  - ☐ Creación de un formulario de Login reactivo
  - ☐ Creación de un formulairo de Registro reactivo
  - ☐¿Qué son los formularios estrictamente tipados?
  - ☐ Haciendo uso de los formularios tipados

06

Servicios

Links Servicios

- https://imaginaformacion.com/tutoriales/servicios-llamadas-api-angular
- https://desarrolloweb.com/articulos/servicios-angular.html
- https://codigoencasa.com/angular-promesas-vs-observables-elige-tu-destino/
- https://codingpotions.com/angular-servicios-llamadas-http/

Servicios

- Definición
  - Un servicio es una Directiva de Angular que mantiene la lógica de acceso a los datos , los servicios serán consumidos por los componentes , y estos delegaran en ellos la responsabilidad de acceder a la información y a la realización de operaciones con los datos
  - Dicho de otra manera , un servicio es el responsable de acceder y gestionar los datos , encapsulando su proceder
  - Un servicio se crea en Angular 14 con el siguiente comando
    - ng generate service path/servicio
  - Ejemplos de servicios
    - Servicio de acceso a datos de un cliente
    - Servicio de Api /REST con datos remotos
    - Servicio de Autentificacion , etc
- Un servicio puede hacer llamadas a terceros y estos provocar cierta latencia , por ello los servicios se pueden consumir con diferentes tipos de llamadas
  - Servicios simples (solo accedo a datos de memoria de mi app)
  - Servicios HTTP (peticiones cliente http)

Creamos un servicio simple

- Objetivo: Tenemos un array con los alumnos , queremos consumir un elemento lista con esos datos
  - Primero creamos el componente lista
    - npx ng generate component components/lista
  - Después creamos el servicio curso
    - npx ng generate service services/curso
- Vemos en app.modules.ts los cambios
  - Importa la lista y el servicio
    - import { ListaComponent } from ./components/lista/lista.component';
    - **import { CursoService } from './services/curso.service';**
  - En la marca providers inyecta el servicio
    - **providers: [CursoService],**
- Analizamos curso.service.ts
  - import { Injectable } from '@angular/core';
  - @Injectable({ providedIn: 'root'})
  - El resto como si fuera una clase cualquiera con sus métodos

07

Peticiones HTTP