



# Presentació Angular

Primeres passes amb Angular

**IT Academy**

09/2024

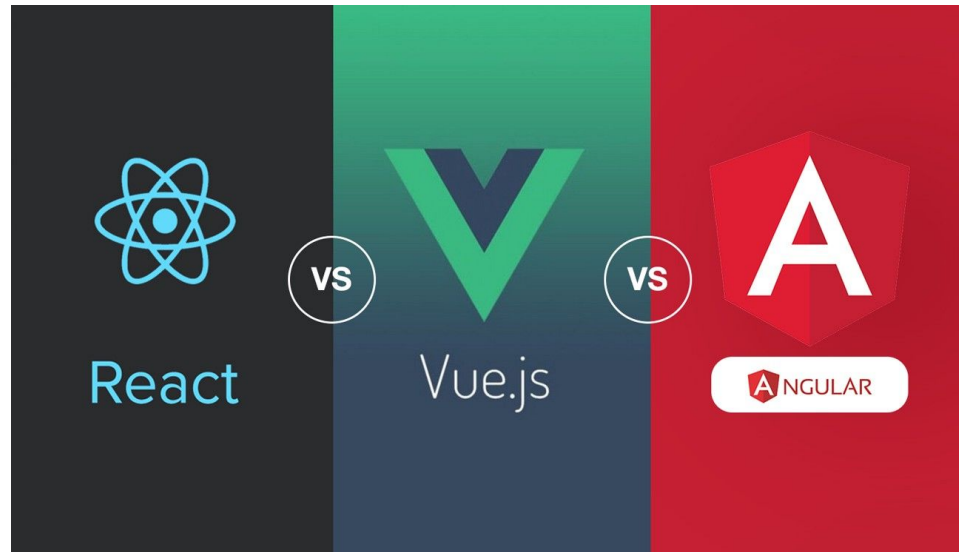


# 01

## Introducció amb Angular 18



# ¿Quin Framework escollir?





# Avantatges d'Angular



## Alguns d'ells...

- Estalvi de temps
- Estructura predeterminada. Ens grans empreses/projectes els programadors/es han de posar cada cosa al seu lloc.
- Ofereix més opcions de configuració al començament del projecte
- Utilitza tota la potència de Typescript, com ara el tipatge.
  - `let decimal: number = 6;`
  - `let fullName: string = `Bob Bobbington`;`
- Millors sous → grans empreses
- Aplicacions més fàcils de mantenir en tenir una estructura més marcada.
- Utilitza la lògica de components, permet la reutilització
- Angular és de Google i React és de Facebook
- Framework complet



En el powershell: set-executionpolicy unrestricted -scope CurrentUser

# Com Instal·lem Angular

- Versió actual amb la que treballem es Angular 18
- Per poder instal·lar Angular, necessitem 2 programes:
  - Node.js
    - Gran quantitat de programes per al desenvolupament a Angular estan implementades a Node.
    - Descarregar: <https://nodejs.org/en>
    - Tenim Node instal·lat? **node -v**
  - Angular CLI
    - Instal·lació d'Angular: **npm install -g @angular/cli**
    - Tenim Angular instal·lat? **ng version**
    - Si volem actualitzar: **npm install -g @angular/cli@latest**



# Plugin Angular Language Service

The screenshot shows the Visual Studio Code interface with the 'Angular Language Service' extension installed and active. The left sidebar displays the 'EXTENSIONS: MARKETPLACE' view, listing several Angular-related extensions. The main editor area shows the 'Extension: Angular Language Service' details, including the Angular logo, the extension's name, description, and version information. The extension is currently installed, and the 'Disable' and 'Uninstall' buttons are visible. The search bar at the top right shows 'typescript-project'.

File Edit Selection View Go Run Terminal Help

EXTENSIONS: MARKETPLACE

angular language Service

**Angular Language Service** 486ms  
Editor services for Angular templates  
Angular

**Angular Snippets (Version 16)** 4.5M ★ 5  
Angular version 16 snippets by John Papa  
John Papa Install

**Angular 17 Snippets - TypeScript, HTML, Angular** 2.4M ★ 4.5  
258 Angular Snippets (TypeScript, HTML, Angular)  
Mikael Morlund Install

**Angular Language Service**  
Angular angular.io | 6,873,147 | ★★★★★  
Editor services for Angular templates  
Disable Uninstall

DETAILS FEATURES CHANGELOG

Angular Language Service

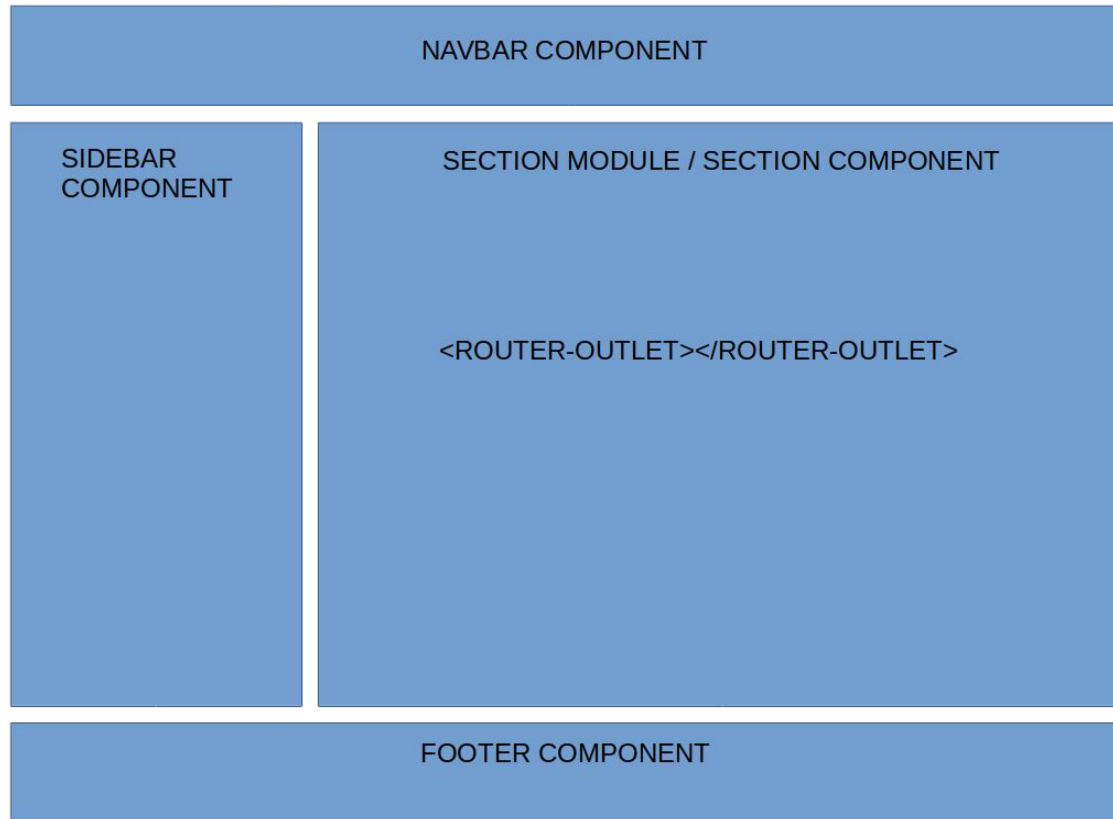


# Creació d'un Projecte amb Angular

1. **ng new projecte\_0** (creació del projecte)
2. Which stylesheet format would you like to use?
  - a. Definir el motor de CSS que volem utilitzar (CSS o SCSS)
3. Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)?
  - a. Habilitar SSR: Millor rendiment inicial i SEO, pero genera més complexitat i càrrega al servidor.
  - b. **No habilitar SSR**: Configuració més simple. Menys càrrega al servidor.  
Temps de càrrega inicial més lent. (llavors SSG activat)
4. **cd projecte\_0** (entro a la carpeta del projecte)
5. **ng serve -o** (activació del servidor)



# ¿Com són els components?







# Estructura d'un Projecte en Angular

Les aplicacions amb angular funcionen amb Components.

- Permeten una reutilització molt ràpida
- Ex: una web d'una gestoria tindria el components:
  - El component "HEADER", "FOOTER" i "PAGES"
- Un component conté:
  - **app.component.ts** per posar la lògica
  - **app.component.html** per posar el html
  - **app.component.css** per posar el css
  - **app.component.spec.ts** per posar el test unitaris

```
> node_modules
> public
✓ src
  ✓ app
    app.component.html
    app.component.scss
    app.component.spec.ts
    app.component.ts
    app.config.ts
    app.routes.ts
    index.html
    main.ts
    styles.scss
    .editorconfig
    angular.json
```



# Modificarem el nostre primer Component

1. Obrim el fitxer app.component.html i eliminem les dades de prova i escrivim:

```
projecte_1 > src > app > app.component.html  
Go to component  
1 <h1 style="text-align:center">  
2   Benvingut al {{ title }}  
3 </h1>
```

2. Obrim el fitxer app.component.scss i escrivim els nostre estil CSS

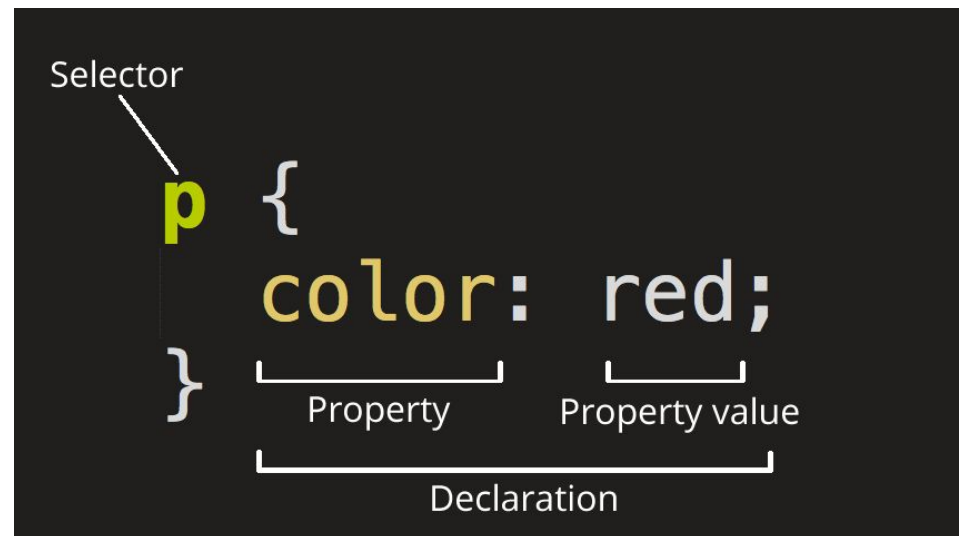
```
projecte_1 > src > app > app.component.scss > h1  
1 h1{  
2   color: red;  
3 }
```

3. Obrim el fitxer app.component.ts i inicialitzem una nova variable

```
projecte_1 > src > app > app.component.ts > ...  
1 import { Component } from '@angular/core';  
2  
3 @Component({  
4   selector: 'app-root',  
5   standalone: true,  
6   imports: [],  
7   templateUrl: './app.component.html',  
8   styleUrls: ['./app.component.scss']  
9 })  
10 export class AppComponent {  
11   title = 'projecte_1';  
12 }
```



## ...una miqueta de CSS





# Afegir una imatge

- Busquem una imatge i la col·loquem a la nostra nova carpeta anomenada assets

```
> node_modules 3 </h1>
✓ public 4
  assets 5
    logo.svg 6 
    favicon.ico 7
  src 8
  app 9
10
```



# Instal·lació de Bootstrap 5

CMD per instal·lar la dependència de bootstrap al nostre projecte:

- **npm install bootstrap**

Afegim en el fitxer **angular.json** els nostre fitxers necessaris per fer funcionar a bootstrap 5

```
    "input": "public"
  },
],
"styles": [
  "./src/styles.scss",
  "./node_modules/bootstrap/scss/bootstrap.scss"
],
"scripts": [
  "./node_modules/bootstrap/dist/js/bootstrap.bundle.js"
]
},
```



# 02

## Elements bàsics d'Angular 18



# Estructura de carpetes

**public:** Aquesta carpeta conté els elements estàtics de la nostra web

**components:** Aquesta carpeta conté els components de l'aplicació. Cada component és una peça reutilitzable de la interfície d'usuari amb la seva pròpia plantilla HTML, full d'estils CSS i lògica TypeScript.

**interfaces:** Aquí s'emmagatzemen les definicions d'interfícies TypeScript que descriuen la forma d'objectes de dades utilitzats a l'aplicació.

**services:** Aquesta carpeta conté els serveis que s'utilitzen per compartir dades i funcionalitats entre diferents parts de l'aplicació.

**app.config.ts:** Configuració general de l'aplicació.

**app.routes.ts:** Definició de les rutes de navegació.

```
> node_modules
✓ public
  > assets
    ★ favicon.ico
  ✓ src
    ✓ app
      > components
      > interfaces
      > services
      app.component.html
      app.component.scss
      app.component.spec.ts
      app.component.ts
      app.config.ts
      app.routes.ts
      index.html
      main.ts
      styles.scss
      .editorconfig
      angular.json
      package-lock.json
      package.json
```



# Previ 1: Comandos d'Angular

## Creació d'un Component:

- ng g component nombre-componente
- ng g c nombre-componente

## Creació d'una Interface

- ng g interface nombre-interface
- ng g i nombre-interface

## Creació d'un Servei

- ng g service nombre-servicio
- ng g s nombre-servicio





# Previ 2: Standalone Components

```
app.component.ts X
curs_angular > src > app > app.component.ts > ...
1 import { CommonModule } from '@angular/common';
2 import { Component } from '@angular/core';
3 import { FormsModule, ReactiveFormsModule } from '@angular/forms';
4 import { RouterOutlet } from '@angular/router';
5
6 @Component({
7   selector: 'app-root',
8   standalone: true,
9   imports: [CommonModule, FormsModule, ReactiveFormsModule, RouterOutlet],
10  templateUrl: './app.component.html',
11  styleUrls: ['./app.component.scss']
12 })
13 export class AppComponent {
14   title = 'curs_angular';
15 }
```

Que fa cada  
mòdul importat?

- **CommonModule:**
  - Directives comunes: Com ngIf, ngFor, ngClass, ngStyle, entre altres.
  - Pipes comunes: Com DatePipe, UpperCasePipe, LowerCasePipe, JsonPipe, entre altres.
- **FormsModule y ReactiveFormsModule:** per treballar amb formularis en Angular
- **RouterOutlet:** permet que l'aplicació canviï de vista dinàmicament segons la URL.



# Tipus de variables

- Creem algunes variables i les enviem cap al **app.component.html**

```
export class AppComponent {  
  title: string = 'projecte_1';  
  nom: string = 'Ismael Kale';  
  edat: number = 55;  
  email: string = 'ismael@gmail.com';  
  cursos: string[] = ['PHP', 'Angular', 'NodeJs'];  
  notas: number[] = [5, 6.5, 8.33];  
  actiu: boolean = true;  
  nivell_cursos: 'Facil' | 'Dificil' | 'Mig' = 'Facil';  
  code: number | string = 45  
}
```



# Activitat 1: Pas de paràmetres a la vista

- Creem algunes variables i les enviem cap al **app.component.html**

```
export class AppComponent {  
  title: string = 'projecte_1';  
  nom: string = 'Ismael Kale';  
  edat: number = 55;  
  email: string = 'ismael@gmail.com';  
  cursos: string[] = ['PHP', 'Angular', 'NodeJs'];  
  notas: number[] = [5, 6.5, 8.33];  
  actiu: boolean = true;  
  nivell_cursos: 'Facil' | 'Dificil' | 'Mig' = 'Facil';  
  code: number | string = 45  
}
```

- Exemple del **app.component.html**

```
<h1 class="text-center">  
  Benvingut al {{ title }}  
</h1>  

```



# @if / @else - @for - @switch/@case/@default

## Exemple d'un @if

```
@if (edat>=18) {  
|   <p>Es major d'edat.</p>  
} @else {  
|   <p>Es menor d'edat.</p>  
}
```

## Exemple d'un @switch

```
@switch (nivell_cursos) {  
|   @case ('Facil') {  
|       <p>El nivell dels cursos es Fàcil</p>  
|   }  
|   @case ('Mig') {  
|       <p>El nivell dels cursos es Mig</p>  
|   }  
|   @case ('Dificil') {  
|       <p>El nivell dels cursos es Difícil</p>  
|   }  
|   @default {  
|       <p>Aquest nivell es desconegut.</p>  
|   }  
}
```

## Exemple d'un @for

```
<ul>  
@for(curso of cursos; track curso) {  
|   <li>{{curso}}</li>  
}  
</ul>
```



## Activitat 2: @if

```
@if (edat>=18) {  
  <p>Es major d'edat.</p>  
} @else {  
  <p>Es menor d'edat.</p>  
}
```

Donades les variables nom i nota, indica si l'alumne ha aprovat o ha suspès.

Exemple:

- L'alumne **XXXX** ha aprovat l'examen amb un **XX**



## Activitat 3: @for

Mostra una taula amb la següent informació:

- Ismael Kale amb edat: 52
- Laura Martinez amb edat: 18
- Pepe Martorell amb edat: 42

```
<ul>
@for(curso of cursos; track curso) {
  <li>{{curso}}</li>
}
</ul>
```

persones: any = [{

id: 1,

nom: 'Ismael',

cognom: 'Kale',

edad: 52

}, {

id: 2,

nom: 'Laura',

cognom: 'Martinez',

edad: 18

}, {

id: 3,

nom: 'Pepe',

cognom: 'Martorell',

edad: 42

}]





## Activitat 4: @switch/@case/@default

Crea una variable animal que només pot tenir 3 valors (Gos, Gat i Tortuga). En cas de ser un altre animal s'ha de mostrar el missatge “Aquest animal és desconegut”

Exemple si el valor es un “Gos”:

- Hola sóc un Gos

Exemple si el valor es un “Serp”:

- Una Serp no es un animal vàlid

```
@switch (nivell_cursos) {  
  @case ('Facil') {  
    <p>El nivell dels cursos es Fàcil</p>  
  }  
  @case ('Mig') {  
    <p>El nivell dels cursos es Mig</p>  
  }  
  @case ('Dificil') {  
    <p>El nivell dels cursos es Difícil</p>  
  }  
  @default {  
    <p>Aquest nivell es desconegut.</p>  
  }  
}
```





# ¿Com podem capturar event?

Un event pot ser:

- Al fer un clic a un botó vull que canviï un valor.

El número actual es: 6

Incrementar

Decrementar

Solució:

```
contador: number = 1;

incrementar() {
  this.contador++;
}

decrementar() {
  this.contador--;
}
```

```
El número actual es: {{contador}}
<button class="btn btn-primary w-auto m-1" (click)="incrementar()">Incrementar</button>
<button class="btn btn-primary w-auto m-1" (click)="decrementar()">Decrementar</button>
```





## Activitat 5: Calcula un número aleatori

Crea un botó que al fer clic, es mostri un numero aleatori entre el 0 y el 13. El valor inicial es 1

```
getRandomNumber(max: number) {  
    this.num_aleatori = Math.floor(Math.random() * max);  
}
```

Exemple:

Dona'm un número aleatori 1

Nou número



# ¿Qué es un Pipe?

- Son funciones que se criden desde la vista (html) i tenen com a objectiu transformar una dada per millorar la seva visualització.

Exemples:

- Nom del client {{ nombre | uppercase }}
- El salari es: {{ salari | currency:'\$' }}
- Data actual: {{ fechaActual | date:'d/M/y' }}
- Llistat de persones en format JSON {{ persones | json }}

```
templateUrl: './app.component.  
styleUrl: './app.component.scs  
imports: [CommonModule, ListAr
```



# Activitat 5

Converteix el nom a majúscules i el cognom a minúscules




# ¿Qué es un interfície?


En aquest cas haurem de crear un interfície Persona:

- **ng g i interfaces/persona** (ng generate interface persona)

```
export interface Persona {  
  id: number;  
  nom: string;  
  cognom: string;  
  edad: number;  
  salari?: number; //campo opcional  
}
```



```
persones: any = [{  
  id: 1,  
  nom: 'Ismael',  
  cognom: 'Kale',  
  edad: 52  
}, {  
  id: 2,  
  nom: 'Laura',  
  cognom: 'Martinez',  
  edad: 18  
}, {  
  id: 3,  
  nom: 'Pepe',  
  cognom: 'Martorell',  
  edad: 42  
}];
```



```
persones: Persona[] = [{  
  id: 1,  
  nom: 'Ismael',  
  cognom: 'Kale',  
  edad: 52  
}, {  
  id: 2,  
  nom: 'Laura',  
  cognom: 'Martinez',  
  edad: 18  
}, {  
  id: 3,  
  nom: 'Pepe',  
  cognom: 'Martorell',  
  edad: 42  
}];
```



# Creació d'un component Fill d'Articles

Les aplicacions d'Angular estan creades amb molts components.

- **ng g c components/list-articles** (ng generate component components/articles)
- **ng g i interfaces/article** (ng generate interface interfaces/articles)

La nostra aplicació comença amb el app.component

Des d'aquest component cridarem al nostre component fill anomenat **list-articles**

```
app.component.html × list-articles.component.ts
projecte_1 > src > app > app.component.html > ...
36
37 <app-list-articles></app-list-articles>
38
```





# Creació d'un servei Articles

Un servei en Angular és una classe que s'utilitza per compartir dades, lògica de negoci i funcionalitats entre diferents components d'una aplicació.

```
import { Injectable } from '@angular/core';
import { Article } from '../interfaces/article';

@Injectable({
  providedIn: 'root'
})
export class ArticleService {

  articulos: Article[] = [
    { id: 1, nombre: 'Servicio Aceite de oliva', precio: 8.50, foto: 'cookingOil.jpg' },
    { id: 2, nombre: 'Servicio CupCake', precio: 0.90, foto: 'instantCupcakeMixture.jpg' },
    { id: 3, nombre: 'Servicio Pasta', precio: 2.50, foto: 'pasta.jpg' },
    { id: 4, nombre: 'Servicio Jersey', precio: 15.90, foto: 'sweaters.jpg' }
  ];

  getArticles(): Article[] {
    return this.articulos;
  }
}
```

Actualment el nostre **ListArticlesComponent** conté el llistat del Articles, pero en realitat hauren d'estar a un Service.

```
export class ListArticlesComponent {
  articulos: Article[] = [];

  artcileService = inject(ArticleService);

  constructor() {
    this.loadArticles();
  }

  loadArticles() {
    this.articulos = this.artcileService.getArticles();
  }
}
```

**ng g s service/article** (ng generate service articles)



# Copy & Paste

```
articulos: Article[] = [  
    { id: 1, nombre: 'Servicio Aceite de oliva', precio: 8.50, foto: 'cookingOil.jpg' },  
    { id: 2, nombre: 'Servicio CupCake', precio: 0.90, foto: 'instantCupcakeMixture.jpg' },  
    { id: 3, nombre: 'Servicio Pasta', precio: 2.50, foto: 'pasta.jpg' },  
    { id: 4, nombre: 'Servicio Jersey', precio: 15.90, foto: 'sweaters.jpg' }  
];
```



## Activitat 5

Fes el mateix que a l'activitat anterior pero amb l'array de Persones





# Directiva: ngClass

Es un directiva per modificar les classes de l'HTML

Exemple:

**color: string = 'azul'**

Línea 1: Si la variable color és 'azul' es mostrarà per pantalla la classe 'color-azul', si no es mostrarà la classe 'color-verde'

```
<p [ngClass]="(color == 'azul') ? 'color-azul' : 'color-verde'">  
  Soy un texto pintado1  
</p>
```

Línea 2: Si la variable color és 'azul' es mostrarà per pantalla la classe 'color-azul'. Si la variable és 'verde' es mostrarà la classe 'color-verde'

```
<p [ngClass]="{'color-azul': (color=='azul'), 'color-verde': (color=='verde')}">  
  Soy un texto pintado2  
</p>
```



# 03

## Observables & API



# Creació d'un servei per consumir una API

¿Qué es una API?

- Es defineix com una interfície que afavoreix la comunicació entre dos sistemes o plataformes diferents.

Exemples de API:

- APIs de Google Maps
- API de Pokemon (<https://pokeapi.co/api/v2/pokemon/>)
- **API d'Acudits** (<https://api.chucknorris.io/jokes/random>)
- API de BarcelonActiva

El programa **POSTMAN** ens permet realitzar peticions d'una manera simple per testejar API.



# Fem una prova amb la APi d'Acudits

The screenshot shows a web browser interface for testing HTTP requests. The address bar displays the URL `https://api.chucknorris.io/jokes/random`. The method is set to `GET`. The response body is displayed in JSON format, showing a random joke by Chuck Norris.

```
{
  "categories": [],
  "created_at": "2020-01-05 13:42:25.352697",
  "icon_url": "https://api.chucknorris.io/img/avatar/chuck-norris.png",
  "id": "ML01flytQral59tQ0KytWg",
  "updated_at": "2020-01-05 13:42:25.352697",
  "url": "https://api.chucknorris.io/jokes/ML01flytQral59tQ0KytWg",
  "value": "Chuck Norris can smell how tight your wife is."
}
```



# ¿Qué es un observable i una subscripció?

**Observable:** Sería com un canal de Youtube. Només es podran rebre la informació aquells components que s'hi hagin subscript.

En Angular és molt típic tenir un servei que emet dades a través d'un observable.



## Activitat 6

1. Creació d'un component anomenat **joke** a la carpeta Components
2. Creació d'una interfície anomenada **joke** a la carpeta Interfaces
3. Creació d'un servei anomenat **joke** a la carpeta services
4. Afegir el provider httpClient al fitxer el app.config.ts
5. Crear una subscripció a un observable i fer un bucle al HTML

```
export class JokesComponent {  
  
  jokeService = inject(JokeService);  
  joke!: Joke;  
  
  getJoke() {  
    this.jokeService.getJokes().subscribe((data: Joke) => {  
      this.joke = data;  
    });  
  }  
}
```

### Interfaces

```
"categories": [],  
"created_at": "202",  
"icon_url": "https",  
"id": "CJ_39dM9TLy",  
"updated_at": "202",  
"url": "https://ap",  
"value": "Chuck No",  
on the screen"
```

```
import { Injectable, inject } from '@angular/core';  
import { Observable } from 'rxjs';  
import { HttpClient } from '@angular/common/http';  
import { Joke } from '../interfaces/joke';
```

```
@Injectable({  
  providedIn: 'root'  
})
```

```
export class JokeService {
```

```
  httpClient = inject(HttpClient);
```

```
  getJokes(): Observable<Joke> {  
    return this.httpClient.get<Joke>(`https://api.chucknorris.io/jokes/random`);  
  }  
}
```

```
src > app > ts app.config.ts > ...
```

```
1 import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';  
2 import { provideRouter } from '@angular/router';  
3  
4 import { routes } from './app.routes';  
5 import { provideHttpClient } from '@angular/common/http';  
6  
7 export const appConfig: ApplicationConfig = {  
8   providers: [  
9     provideZoneChangeDetection({ eventCoalescing: true }),  
10    provideRouter(routes),  
11    provideHttpClient(),  
12  ]  
}
```





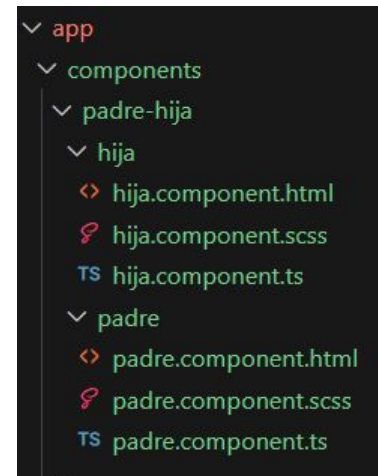
# 04

## **Pas de paràmetres entre pare-filla i filla-pare**



# Passar paràmetres de Pare a Filla

```
ng new projecte_experiment
cd projecte_experiment
ng g c components/padre-hija/padre
ng g c components/padre-hija/hija
```



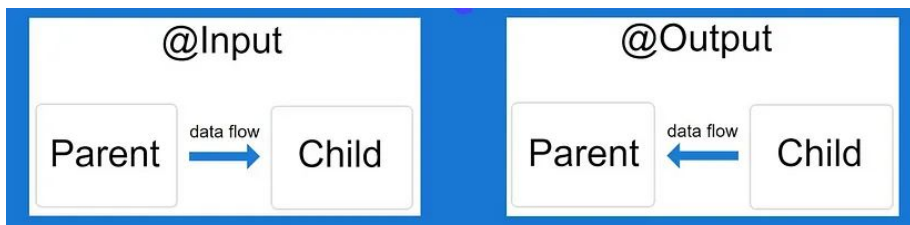
```
@Component({
  selector: 'app-padre',
  standalone: true,
  imports: [HijaComponent],
  templateUrl: './padre.component.html',
  styleUrls: ['./padre.component.scss']
})

export class PadreComponent {
  mensajeParaHija: string = 'Hola desde el componente Padre';
}
```

```
export class HijaComponent {
  @Input({required: true }) mensaje: string = '';
}
```

```
<div class="bg-red">
  <h1>Componente Padre</h1>
  <!-- Pasar el parámetro mensaje -->
  <app-hija [mensaje]="mensajeParaHija"></app-hija>
</div>
```

```
<div class="bg-green">
  <p>Componente Hija</p>
  <p>{{ mensaje }}</p>
</div>
```



## Componente Padre

Componente Hija

Hola desde el componente Padre





# ¿Com arriben al component app-padre?

...doncs amb el routing d'Angular

```
app.component.html  TS app.routes.ts X
c > app > TS app.routes.ts > ...
1  import { Routes } from '@angular/router';
2  import { PadreComponent } from './components/padre-hija/padre/padre.component';
3
4  export const routes: Routes = [
5    { path: '', component: PadreComponent },
6    { path: 'padre-hija', component: PadreComponent },
7    { path: '**', redirectTo: '' } // Ruta de "no encontrado"
8  ];
```

```
app > <> app.component.html > ...
<router-outlet></router-outlet>
```



# Passar paràmetres de Filla a Pare

Per fer-ho necessitem el decorador **@Output** i un **EventEmitter**

```
export class PadreComponent {
  mensajeParaHija: string = 'Hola desde el componente Padre';
  mensajeDesdeHija: string = '';

  recibirMensaje(mensaje: string) {
    this.mensajeDesdeHija = mensaje;
  }
}
```

```
export class HijaComponent {

  @Input({required: true }) mensaje: string = '';
  @Output() mensajeEmitido = new EventEmitter<string>();

  enviarMensajeAlPadre() {
    this.mensajeEmitido.emit('Mensaje desde el componente Hija');
  }
}
```

```
<div class="bg-green">
  <p>Componente Hija</p>
  <p>{{ mensaje }}</p>
  <button (click)="enviarMensajeAlPadre()">Enviar mensaje al Padre</button>
</div>
```

```
<div class="bg-red">
  <h1>Componente Padre</h1>
  <!-- Pasar el parámetro mensaje a la hija y recibir el evento de la hija -->
  <app-hija [mensaje]="mensajeParaHija" (mensajeEmitido)="recibirMensaje($event)"></app-hija>
  <!-- Mostrar el mensaje recibido de la hija -->
  <p>{{ mensajeDesdeHija }}</p>
</div>
```



# Passar paràmetres de Filla a Pare (amb signals)

Per fer-ho amb signals necessitem un **Output** i un **Input**

```
export class CpadreComponent {  
  mensajeParaHija = signal<string>('Hola desde el componente Padre');  
  mensajeDesdeHija = signal<string>('');  
  
  recibirMensaje(mensaje: string) {  
    this.mensajeDesdeHija.update((value: string) => mensaje);  
  }  
}
```

```
export class ChijaComponent {  
  
  mensaje = input.required<string>();  
  mensajeEmitido = output<string>();  
  
  enviarMensajeAlPadre() {  
    this.mensajeEmitido.emit('Mensaje desde el componente Hija');  
  }  
  
  constructor() {  
    effect(() => {  
      console.log("Acabo de recibir un maneje:" + this.mensaje());  
    });  
  }  
}
```

```
<div class="bg-red">  
  <h1>Componente Padre</h1>  
  <!-- Pasar el parámetro mensaje a la hija y recibir el evento de la hija -->  
  <app-chija [mensaje]="mensajeParaHija()" (mensajeEmitido)="recibirMensaje($event)"></app-chija>  
  <!-- Mostrar el mensaje recibido de la hija -->  
  <p>{{ mensajeDesdeHija() }}</p>  
</div>
```

```
<div class="bg-green">  
  <p>Componente Hija</p>  
  <p>{{ mensaje() }}</p>  
  <button (click)="enviarMensajeAlPadre()">Enviar mensaje al Padre</button>  
</div>
```



# ¿Què son els signals?

ng g c components/signals/send

ng g c components/signals/receive

ng g s service/data\_transfer

```
export class SendComponent {  
  transfer = inject(DataTransferService);  
  
  send() {  
    this.transfer.updateCount();  
  }  
}
```

**Effect:** es crida a la funció cada cop que un valor d'un signal canviï. Sempre va al Constructor

```
export class ReceiveComponent {  
  transfer = inject(DataTransferService);  
  count: number = 0;  
  
  constructor() {  
    effect(() => {  
      this.count = this.transfer.count();  
    });  
  }  
}
```

```
export class DataTransferService {  
  
  count = signal<number>(0);  
  
  updateCount(){  
    this.count.update(value => value + 1);  
  }  
}
```



# Signals VS Observables

En Angular 18, tant els signals (senyals) com els observables són mecanismes per **gestionar dades reactives**. Compararem tots dos i veurem quan és més apropiat utilitzar un o l'altre.

Característica	Observables	Signals
Creació	<code>Observable`</code> de RxJS	<code>signal`</code> d'Angular
Subscripcions	Necessàries ( <code>.subscribe()</code> )	No es necessiten subscripcions
Gestió d'estat	Complex amb operadors RxJS	Senzill i directe
Reactivitat	Manual mitjançant <code>.subscribe()</code>	Automàtica
Interoperabilitat	Ampli suport i operadors RxJS	Pot interoperar amb observables
Cancel·lació	Necessària per evitar fuites de memòria	No aplica
Complexitat	Més complex per aprendre i utilitzar	Més simple per gestionar estats bàsics
Ús típic	Dades asíncrones, fluxos d'esdeveniments complexos	Estat local reactiu i senzill



# 05

## Formularis Reactius





# Descarreguem el projecte\_empleats

1. Descarregar el fitxer de l'aplicació Angular [Link projecte](#)
2. npm install
3. ng serve -o



# API: Llistat d'empleats (GET)

HTTP <https://api.beez.es/empleado>

GET <https://api.beez.es/empleado>

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON** ▾

1 ↕

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON ▾ ≡

```
1 {
2   "code": 0,
3   "error": false,
4   "message": "Show List Employee ",
5   "data": [
6     {
7       "id_empleado": 1,
8       "dni": "62924263N",
9       "nombre": "Juan",
10      "apellidos": "Pérez",
11      "email": "juan.perez@example.com",
12      "departamento": "Ventas",
13      "salario": "50000.00",
14      "telefono": "123456789",
15      "created_at": "2024-06-26 10:25:08",
16      "updated_at": "2024-06-26 10:25:08"
17    },
18    {
19      "id_empleado": 2,
20      "dni": "24526584M",
21      "nombre": "María",
22      "apellidos": "López",
```





# API: Detall de l'empleat id:2 (GET)

HTTP <https://api.beez.es/empleado/2>

GET <https://api.beez.es/empleado/2>

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON**

1 5

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize **JSON**

```
1
2  "code": 0,
3  "error": false,
4  "message": "Show Employee 2",
5  "data": {
6    "id_empleado": 2,
7    "dni": "24526584M",
8    "nombre": "María",
9    "apellidos": "López",
10   "email": "maria.lopez@example.com",
11   "departamento": "Marketing",
12   "salario": "45000.00",
13   "telefono": "123456789",
14   "created_at": "2024-06-26 10:25:08",
15   "updated_at": "2024-06-26 10:25:08"
16 }
17
```



# API: Detall de l'empleat id:2 (GET)

HTTP <https://api.beez.es/empleado/2>

GET <https://api.beez.es/empleado/2>

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON**

1 5

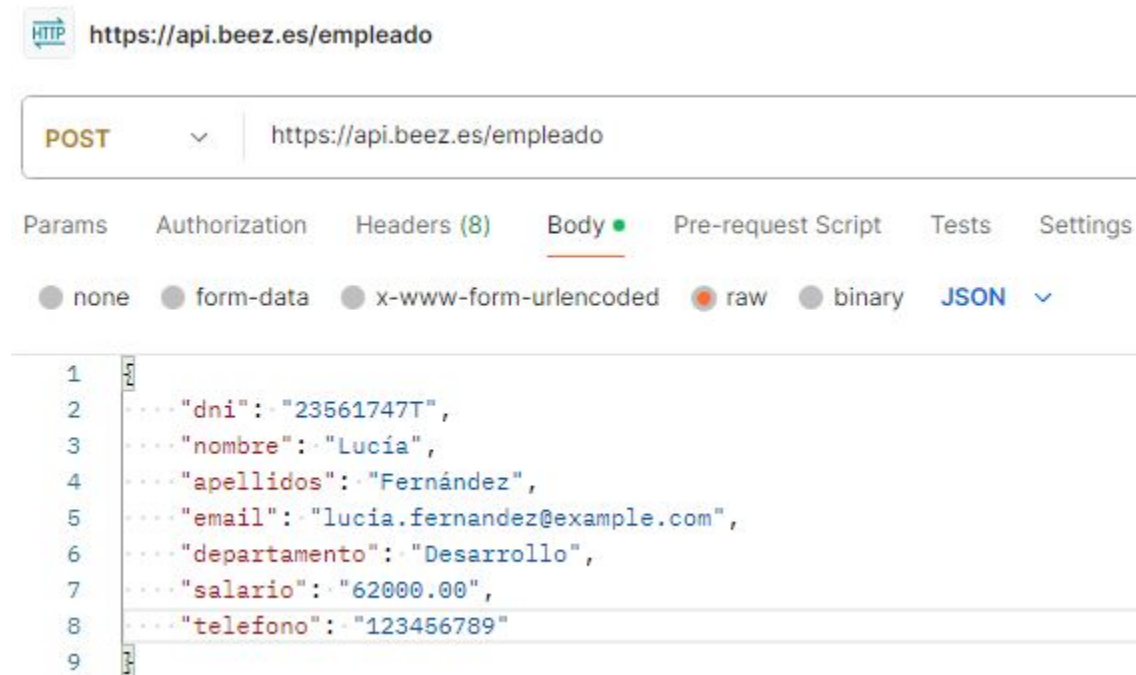
Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize **JSON**

```
1
2  "code": 0,
3  "error": false,
4  "message": "Show Employee 2",
5  "data": {
6    "id_empleado": 2,
7    "dni": "24526584M",
8    "nombre": "María",
9    "apellidos": "López",
10   "email": "maria.lopez@example.com",
11   "departamento": "Marketing",
12   "salario": "45000.00",
13   "telefono": "123456789",
14   "created_at": "2024-06-26 10:25:08",
15   "updated_at": "2024-06-26 10:25:08"
16  }
17
```



# API: Afegir un empleat (POST)





# API: Actualizar l'empleat id:20 (PATCH)

HTTP <https://api.beez.es/empleado/20>

PUT [▼](#) <https://api.beez.es/empleado/20>

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON** ▼

```
1  {
2    "dni": "32208688V",
3    "nombre": "Pepe",
4    "apellidos": "Rivas",
5    "email": "gloria.rivas@example.com",
6    "departamento": "Finanzas",
7    "salario": "58000.00",
8    "telefono": "123456780"
9  }
```



# Reactive forms: El trabajo de validación se realiza en el .ts

---

**FormBuilder:** Se usa para construir un formulario creando un FormGroup, (un grupo de controles) que realiza un seguimiento del valor y estado de cambio y validez de los datos.

## Del FormBuilder derivan (parte TS):

- FormControl: controles de formulario
- FormGroup: estructuras complejas de grupos de controles anidados
- FormArray: gestionar grupos de controles como si fueran parte de un array

## En la plantilla, usamos (parte Vista):

- La directiva “formGroup” para asociar la etiqueta form al objeto FormGroup principal
- La directiva “formControlName” para asociar cada control al objeto que lo representa en el código.



# Ej. Reactive forms

```

1 import { Component } from '@angular/core';
2 import { FormBuilder, Validators, FormArray } from
3
4 @Component({
5   selector: 'app-profile-editor',
6   templateUrl: './profile-editor.component.html',
7   styleUrls: ['./profile-editor.component.css']
8 })
9 export class ProfileEditorComponent {
10   profileForm = this.fb.group({
11     firstName: ['', Validators.required],
12     lastName: ['', Validators.required],
13     address: this.fb.group([
14       street: ['',],
15       city: ['',],
16       state: ['',],
17       zip: ['']
18     ]),
19     aliases: this.fb.array([
20       this.fb.control('')
21     ])
22   });
23
24   get aliases() {
25     return this.profileForm.get('aliases') as FormArray;
26   }
27
28   constructor(private fb: FormBuilder) { }
29

```

```

1 <form [formGroup]="profileForm" (ngSubmit)="onSubmit()">
2   <label for="first-name">First Name: </label>
3   <input id="first-name" type="text" formControlName="firstName" required>
4
5   <label for="last-name">Last Name: </label>
6   <input id="last-name" type="text" formControlName="lastName">
7
8   <div formGroupName="address">
9     <h2>Address</h2>
10
11     <label for="street">Street: </label>
12     <input id="street" type="text" formControlName="street">
13
14     <label for="city">City: </label>
15     <input id="city" type="text" formControlName="city">
16
17     <label for="state">State: </label>
18     <input id="state" type="text" formControlName="state">
19
20     <label for="zip">Zip Code: </label>
21     <input id="zip" type="text" formControlName="zip">
22   </div>
23
24   <div formArrayName="aliases">
25     <h2>Aliases</h2>
26     <button type="button" (click)="addAlias()">+ Add another alias</button>
27
28     <div *ngFor="let alias of aliases.controls; let i=index">
29       <!-- The repeated alias template -->
30       <label for="alias-{{ i }}">Alias:</label>
31       <input id="alias-{{ i }}" type="text" [formControlName]="i">
32     </div>

```

Tener en cuenta que la directiva **"formGroup"** se asigna con binding de propiedades (con corchetes), mientras que la directiva **"formControlName"** se asigna directamente, sin binding.

[Link projecte 1](#)  
[Link projecte experiment](#)  
[Link projecte empleats](#)



**[barcelona.cat/barcelonactiva](http://barcelona.cat/barcelonactiva)**