




How to be a Hero in Angular

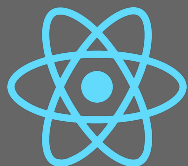
codehouse
{ ACADEMY }



Software Engineer at Technest

 /JoseAntpr

 @JoseAntPR



codehouse
{ ACADEMY }

Angular Introducción

</> Angular Introducción

- Framework para crear SPA apps con HTML y JS(TS).
- Son aplicaciones Reactivas y no recargan el navegador.
- Comunidad muy grande detrás y Google.
- Multitud de librerías (Angular material, ngrx ...)



Angular Introducción

¿Cómo empiezo con Angular?

</> Angular Introducción

Usemos el CLI y creemos nuestro primer proyecto

<https://cli.angular.io/>



</> Primer ejercicio

- Crea tu aplicación llamada examples
- Corre la aplicación



`</>` TypeScript

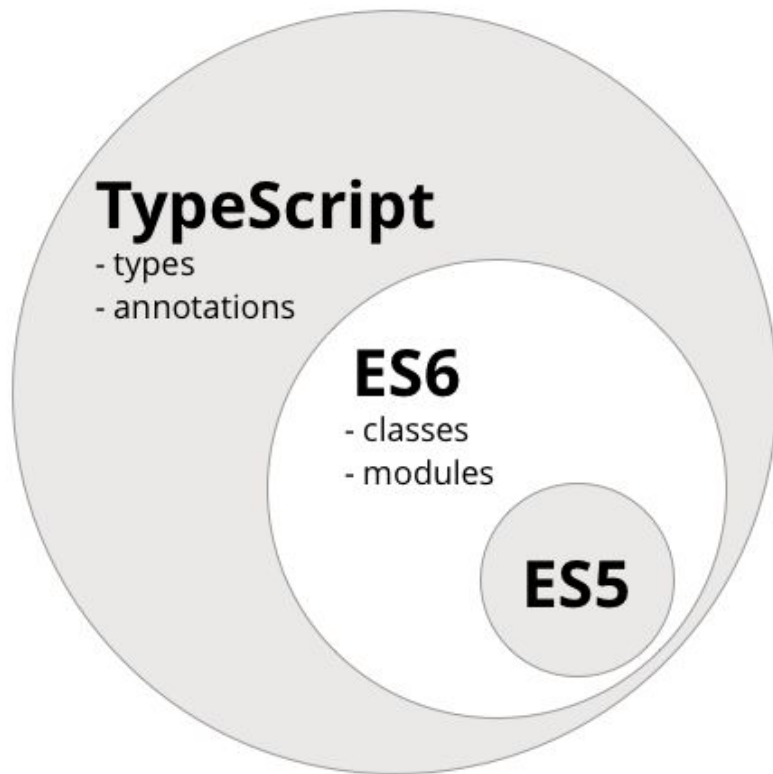


</> TypeScript

- Extensión de JS
- Añade características de POO: tipado, herencia, clase, etc.
- Compila a JS (Transpile)
- Motor ECMAScript3 



TypeScript





TypeScript - Tipado de variables

```
// Type annotation  
let message: string = 'Hello World';  
  
// Type inference  
let message = "Hello World"
```



Pregunta

```
let message = 'Hello World';
```

```
message = 10;
```

```
console.log(message);
```

```
const n = 5
```

```
n = n + 1
```



TypeScript - Tipado any,void & type alias

```
let things: any[] = ['Hello World', 10, false];
```

```
testFunction(): void {  
}
```

```
type ownType = string | number | boolean
```



TypeScript - Interfaces

```
export interface videoResource {  
  name: string;  
  
  duration: number;  
  
  casting: Casting[];  
  
  method1(): string; {}  
}  
  
export interface Casting {  
  name: string;  
  surname: string;  
  image: string;  
}  
  
class Movie implements videoResource {}
```



TypeScript - Clases y Herencia

```
export class Vehicle {  
  constructor(  
    private id: number,  
    public type: string  
  ) {}  
  
  getSpeed(): number {  
    return 5;  
  }  
}  
  
class Car extends Vehicle {}
```



TypeScript - Decoradores

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.scss']  
})  
export class AppComponent {  
  title = 'movies';  
}
```


`</>` Angular Template syntax





Angular Binding - Interpolación

```
<!--Template-->  
<h1>  
  Welcome to {{ title }}!  
</h1>  
  
@Component({...})  
export class AppComponent {  
  title = 'movies';  
}
```



Angular Binding - Enlace de propiedades

```
<!--Template-->  
<img width="300" alt="Angular Logo" [src]="img" />  
  
@Component({...})  
export class AppComponent {  
  img = "...";  
}
```



Angular Binding - Enlace de clases

```
<!-- Template -->  
<button [class.like]="isLike">Favourite</button>  
  
@Component({...})  
export class AppComponent {  
  isLike = false;  
}
```



Angular Binding - Enlace de eventos

```
<!-- Template -->  
<button (click)="save()">Save</button>  
  
@Component({...})  
export class AppComponent {  
  save(): void {  
    console.log("Saving ...");  
  }  
}
```

`</>` Angular Directivas





Angular Directivas

- Facilitan la manipulación del DOM
- Se apoyan en un motor de templates
- Tres tipos:
 - Components
 - Directivas de atributo
 - Directivas estructurales



Angular Directivas - Directivas de atributo

- Cambian estilo y comportamiento
- Se usan como atributos de los elementos

</> Angular Binding - Directivas de atributo [ngClass]

```
<div [ngClass]="{'modal-opened': open, 'modal-closed':  
!open}"></div>
```

</> Angular Binding - Directivas de atributo [ngStyle]

```
<div [ngStyle]="{ color: red }"></div>
```

</> Angular Directivas - Directivas de Estructura

- Cambian la estructura
- Ofrecen syntactic sugar * para facilitar uso.



Angular Directivas - Directivas de Estructura *ngIf & *ngFor

```
<div *ngIf="movie.theater"><button>Ver en  
cines</button></div>
```

```
<ul>
```

```
<li *ngFor="let movie of movies">{{ movie.name }}</li>
```

```
</ul>
```



codespace

FORMACIÓN TECNOLÓGICA

</> Angular Directivas - Directivas de Estructura *ngSwitch

```
<div [ngSwitch]="movie.service">  
  <span *ngSwitchCase="cinema">En cines</span>  
  <span *ngSwitchCase="netflix">En netflix</span>  
  <span *ngSwitchDefault>En ningún canal</span>  
</div>
```



codespace

FORMACIÓN TECNOLÓGICA

`</>` Angular Components

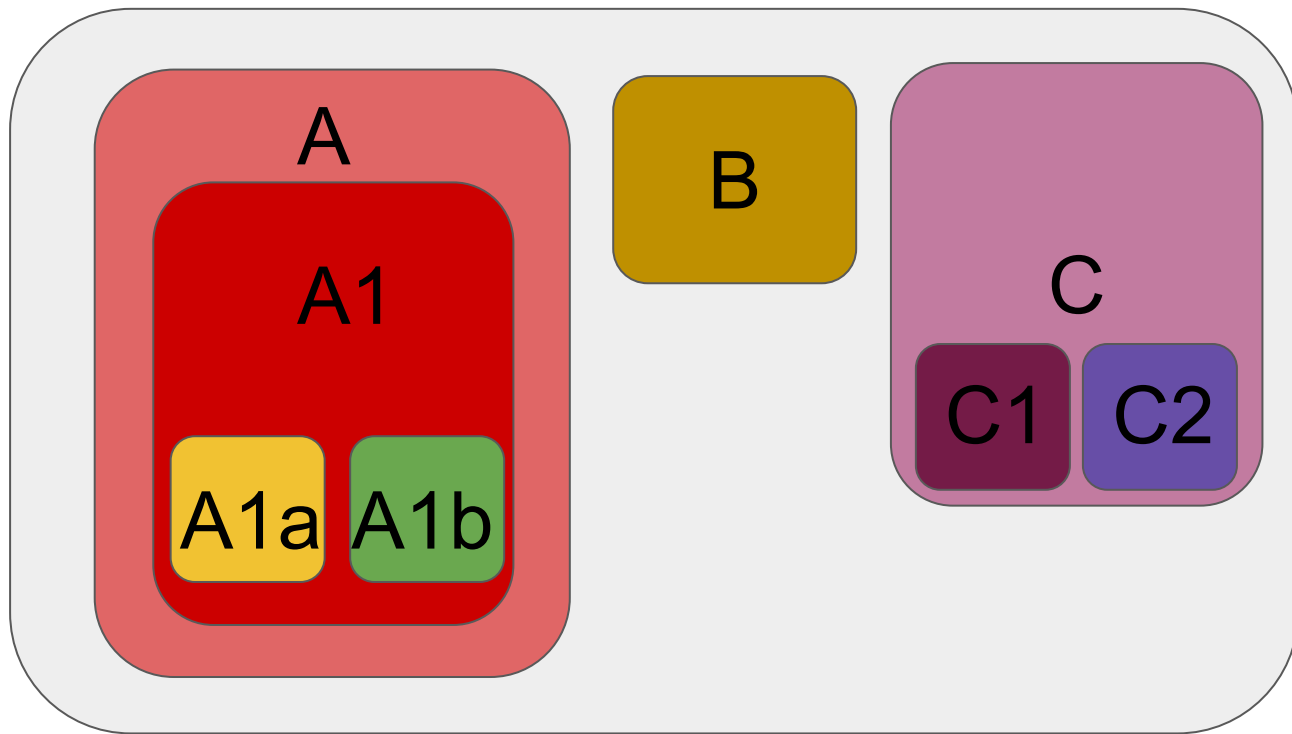


</> Angular Components

- Bloques de construcción de UI.
- Las apps son árboles de componentes.
- Clases decoradas con `@Component`.
- Nuestro componente raíz es `AppComponent`



Angular Components - Estructura





Angular Components - Creación

```
// Por defecto  
ng generate component <name> --options
```

```
// Simplificado  
ng g c <name>
```



Angular Components - Metadatos

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.scss']  
})  
export class AppComponent {  
}
```

</> Angular Components - Ciclos de vida

- Importar:

```
import { Component, OnInit } from '@angular/core';
```

- Implementar en la clase:

```
export class AppComponent implements OnInit {  
  ngOnInit(): void {}  
}
```



Angular Components - Ciclos de vida

Hook	Purpose and Timing
<code>ngOnChanges()</code>	Respond when Angular (re)sets data-bound input properties. The method receives a <code>SimpleChanges</code> object of current and previous property values. Called before <code>ngOnInit()</code> and whenever one or more data-bound input properties change.
<code>ngOnInit()</code>	Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties. Called <i>once</i> , after the <i>first</i> <code>ngOnChanges()</code> .
<code>ngDoCheck()</code>	Detect and act upon changes that Angular can't or won't detect on its own. Called during every change detection run, immediately after <code>ngOnChanges()</code> and <code>ngOnInit()</code> .
<code>ngAfterContentInit()</code>	Respond after Angular projects external content into the component's view / the view that a directive is in. Called <i>once</i> after the first <code>ngDoCheck()</code> .
<code>ngAfterContentChecked()</code>	Respond after Angular checks the content projected into the directive/component. Called after the <code>ngAfterContentInit()</code> and every subsequent <code>ngDoCheck()</code> .
<code>ngAfterViewInit()</code>	Respond after Angular initializes the component's views and child views / the view that a directive is in. Called <i>once</i> after the first <code>ngAfterContentChecked()</code> .
<code>ngAfterViewChecked()</code>	Respond after Angular checks the component's views and child views / the view that a directive is in. Called after the <code>ngAfterViewInit</code> and every subsequent <code>ngAfterContentChecked()</code> .
<code>ngOnDestroy()</code>	Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks. Called <i>just before</i> Angular destroys the directive/component.

`</>` Angular
RxJS





Angular RxJS

- Facilitan trabajo con flujos de datos(streams)
- Proporciona sistema push de datos
- Ofrece operadores para transformar, Filtrar o combinar.



Angular RxJS

```
import { Observable } from 'rxjs/Observable';  
import { map } from 'rxjs/operators';
```



Angular RxJS - Observables

```
const observable$: Observable<string> =  
Observable.create(  
  (observer: Observer<string>) => {  
    observer.next('Hello');  
    observer.error('error');  
    observer.complete();  
  }  
);  
  
observable$.subscribe(  
  (data) => console.log(data),  
  (err) => console.error(err)  
);
```


y muchos muchos más





Angular



Inyección de
dependencias

</> Angular Inyección de dependencias

- Patrón de diseño
- Las dependencias se suministran ya resueltas.
- La clase no se ocupa de instanciar las dependencias.
- Angular Injector es nuestro Héroe.

</> Angular Inyección de dependencias - Ejemplo sin inyección

```
@Component({})  
export class NameComponent {  
  private _nameService: NameService;  
  constructor() {  
    this._nameService = new NameService();  
  }  
}
```

</> Angular Inyección de dependencias - Ejemplo con inyección

```
@Component({})  
export class NameComponent {  
  constructor(private _nameService: NameService) {}  
}
```

</> Angular Inyección de dependencias - Registro de providers

```
@NgModule({  
  ...  
  providers: [NameService]  
  ...  
})  
export class AppModule {}
```

`</>` Angular Routing



</> Angular Routing

- Aporta SPA(Single Page Application).
- Angular se encarga de manipular el DOM.
- Mejor UX



Angular Routing - Establecer / como URL

En en el <header> de nuestro index.html añadir:

```
<base href="/">
```

Debiendo ser el primer documento antes que el CSS, favicon o scripts.



Angular Routing - Crear modulo

```
// Por defecto  
ng generate module app-routing
```

```
// Simplificado  
ng g m app-routing
```

</> Angular Routing - Definición de rutas

```
const routes: Routes = [  
  {  
    path: "",  
    component: Componente  
  },  
  {  
    path: 'admin',  
    component: AdminComponent  
  }  
]
```

</> Angular Routing - Registrar rutas

```
@NgModule({  
  imports:[  
    RouterModule.forRoot(routes)  
  ],  
  exports: [ RouterModule ]  
})  
export class AppRoutingModule {}
```

</> Angular Routing - Importar el modulo Routing

```
@NgModule({  
  imports:[  
    AppRoutingModule  
  ],  
})  
export class AppModule {}
```

</> Angular Routing - Router Outlet & RouterLink

```
//RouterOutlet
<div>
  <router-outlet></router-outlet>
</div>
//RouterLink
<nav>
  <a routerLink="/">Home</a>
</nav>
```

</> Angular Routing - Rutas parametrizadas

```
const routes: Routes = [  
  {  
    path: 'movie/:id',  
    component: MovieComponent  
  },  
]
```


</> Angular Routing - Rutas parametrizadas

//RouterLink

```
<li *ngFor="let movie of movies">  
  <a routerLink="['movie', movie.id]">{{ movie.name }}</a>  
</li>
```

//Obteniendo los parámetros

```
constructor( private _route: ActivatedRoute) { }
```

```
ngOnInit() {  
  this._route.params.subscribe(( params) => { ...});  
}
```



Angular

Formularios



</> Angular Formularios

- Dos tipos de formularios: Reactivos o por Template.
- Ambas tienen pros y contras
- Reactiva mucho más eficiente y potente.
- Template mucho más intuitiva



Angular Formularios

Para usar unos u otros deberemos importar los módulos correspondientes en el app.module.

ReactiveFormsModule

FormsModule



Angular Formularios - Template

```
<form ngForm #login="ngForm" (ngSubmit)="onSubmit(login)">
```

```
...
```

```
<button type="submit" [disabled]="login.invalid"></button>
```

```
</form>
```

```
<input
```

```
  type="text"
```

```
  [(ngModel)]
```

```
  name="username"
```

```
  #user="ngModel"
```

```
  required />
```



Angular Formularios - Reactivos

```
form: FormGroup;
```

```
ngOnInit() {  
  this.form = new FormGroup({  
    title: new FormControl('', [validators]),  
    overview: new FormControl('', [validators])  
  });  
}
```



Angular Formularios - Reactivos

```
<form [formGroup]="form">
```

```
  Title: <input type="text" formControlName="title" />
```

```
  Overview: <input type="text" formControlName="overview" />
```

```
</form>
```



Angular Formularios - Validaciones

	Si	No
Si el control se ha visitado	ng-touched	ng-untouched
Si el valor ha cambiado	ng-dirty	ng-pristine
Si el valor es válido	ng-valid	ng-invalid



Angular Formularios - Template

```
<input
  type="text"
  [(ngModel)]
  name="username"
  #user="ngModel"
  required />
<div ngIf="user.touched && user.invalid">
  User is not valid
</div>

<div ngIf="form[user].touched && form[user].invalid">
  User is not valid
</div>
```

`</>` Angular Servicios



</> Angular servicios

- Bloques de código reusables
- Son singletons.
- Necesitan inyectarse como dependencias.
- Decoradas con `@Injectable`.



Angular Servicios - Crear servicio

```
// Por defecto  
ng generate service nameService
```

```
// Simplificado  
ng g s nameService
```



Angular Routing - Uso del servicio

```
@component({})  
export class NameComponent {  
    constructor(private _nameService: NameService)  
}
```



Angular

HTTP Client



</> Angular - Http Client

- Facilita el trabajo con peticiones HTTP
- Se apoya en observables.



Angular - Http Client

Necesitamos el modulo de HttpClient Modulo y rxjs.

</> Angular - Http Client

Podemos hacer clases con cualquier verbo http

```
this._http.get('url')
```

```
this._http.post('url', item)
```

```
this._http.put('url', item)
```

```
this._http.delete('url')
```

`</>` Angular Pipes



</> Angular Pipes

- Transforman los datos
- El dato original no se altera.
- Clases decoradas con @Pipe



Angular Pipe - Creacion

// Por defecto
ng generate pipe filter

// Simplificado
ng g p filter



Angular Pipe - Pipe propio

```
@pipe({  
  name: 'filter'  
})  
export class FilterPipe implements PipeTransform {  
  transform( param: any[], property: string, value: any) {  
    return list.filter( item => item[property] === value);  
  }  
}
```

`</>` Angular Guards



</> Angular Guards

- Nos permiten gestionar los permisos de los usuarios
- Permite acceder a las páginas en cuestión
- Varios tipos:
 - CanActive
 - CanActivateChild
 - CanDeactivate
 - Resolve
 - CanLoad



Angular Guards - CanActivate

- Decorada como Injectable
- Retorna true o false o un observable/Promesa.



Angular Guards - CanActivate

```
class AuthGuard implements CanActivate {  
    canActivate(route: ActivatedRouteSnapshot, state:  
RouterStateSnapshot) {  
        if( this.userService.IsLoggedIn()){  
            return true;  
        }else{  
            router.navigate(...)  
            return false;  
        }  
    }  
}
```



Angular Guards - CanActivate

```
{  
  path: '  
  component: HomeComponent  
  canActivate: [ AuthGuard ]  
}
```

`</>` Angular Lazy Load



</> Angular Lazy Load

- Técnica para cargar sólo los componentes o módulos que necesitemos.
- Tendremos que tener rutas hijas para ello.



Angular Lazy load - CanActivate

```
const routes: Routes = [  
  { path: "", component: MovieComponent }  
]
```

```
....  
imports: [ RouterModule.forChild(routes]
```

```
cons routes: Routes = [  
  { path: 'home', loadChildren:  
    './paginas/paginas.module#PaginaModule'}  
]
```

GRACIAS



MAKE GIFS AT GIFSOUP.COM