Historia de Angular(.js)



- Miško Hevery creó 'GetAngular' en 2009 como side project.
- Trabajaba en una herramienta llamada Google Feedback.
- Hizo una apuesta con su manager de reescribirlo en 2 semanas.
- Tardó 3 pero pasó de 17.000 lineas a solo 1.500.
- Miško junto a su manager (Brad Green) convirtieron `GetAngular` en Angular.js.
- Vamos, un refactor en toda regla.



Angular Renaissance





Modern Angular

Jesús Rodríquez Rodríquez













Standalone

```
@NgModule({
 declarations:
   AppComponent,
   HeaderComponent,
    FooterComponent,
    HomeComponent,
    FormatPipe,
  imports: [BrowserModule, RouterModule, SharedModule],
  providers: [UserService],
 bootstrap: [AppComponent],
export class AppModule {}
```

Standalone

```
@Injectable({
   providedIn: 'root'
})
export class UserService {
   // Código
}
```

```
@Component((
    standalone: true,
    imports: [NxWelcomeComponent, RouterOutlet],
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrl: './app.component.scss',
))
export class AppComponent (
    title = 'modules';
)
```

```
export const appConfig: ApplicationConfig = {
    providers: {
        provideZoneChangeDetection({ eventCoalescing: true }),
        provideRouter(appRoutes),
        l,
    };
```

Standalone - v19

```
@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  template: '<router-outlet></router-outlet>',
  changeDetection: ChangeDetectionStrategy.OnPush,
  encapsulation: ViewEncapsulation.None
export class AppComponent {
 title = 'Hola DotNetMalaga';
```

Inject()

```
export class MyComponent {
  constructor(private dataService: DataService) {}
  changeLanguage() {
    this.dataService.currentLanguage.set('fr');
  }
}
```

Inject()

```
export class MyComponent {
    #dataService = inject(DataService);

changeLanguage() {
    this.#dataService.currentLanguage.set('fr');
    }
}
```

Inject()

```
import { inject } from '@angular/core';
import { DataService } from './data.service';
export const injectCurrentLanguage = () => inject(DataService).currentLanguage;
```

```
export class MyComponent {
   currentLanguage = injectCurrentLanguage();

   changeLanguage() {
     this.currentLanguage.set('fr');
   }
}
```

Control de Flujo - @if

Control de Flujo - @elseif

```
@if(a > b) {
  {{a}} is greater than {{b}}
 Qelse if (b > a) {
  {{a}} is less than {{b}}
  @else {
  {{a}} is equal to {{b}}
```

```
@for (framework of frameworks(); track framework.id) {
    {{ framework.name }}
} @empty {
    No frameworks web(mentira!)
}
```

Control de Flujo - @switch

```
@switch (user().role) {
  @case (role.ADMIN) {
   <dotnet-admin />
  @case (role.USER) {
   <dotnet-user />
```

Control de Flujo - @let

Control de Flujo - @let

```
@let products = products$ | async;
@if (products) {
<u1>
 @for (product of products; track product) {
   {| product.name | } - {{ product.price | }
@else {
 No products
```

- Permite cargar componentes de manera lazy
- Puede retrasar los componentes que no están visible en pantalla
- Muestra "placeholders" mientras no está cargada esa porción de template
- Cargan bajo ciertos eventos:
 - Idle
 - Viewport
 - Interaction
 - Hover
 - Immediate
 - Timer

Initial chunk files	Names	Raw size	Estimated transfer size
chunk-WUPHH373.js		106.13 kB	32.08 kB
polyfills-FFHMD2TL.js	polyfills	34.52 kB	11.28 kB
main-QDOXTWRA.js	main	17.75 kB	5.69 kB
styles-5INURTSO.css		0 bytes	0 bytes
	Initial total	158.40 kB	49.05 kB
Lazy chunk files chunk-MRRY4GLD.js	Names lento-component	Raw size 351 bytes	Estimated transfer size 351 bytes

```
@defer (on viewport) {
  <componente-lento-de-la-muerte />
@placeholder {
 <div>Cargando...</div>
```

```
@defer (on interaction) {
 <componente-lento-de-la-muerte />
@loading (after 100ms; minimum 1s) {
 <div>Lo que se ve mientras se procesa</div>
@placeholder {
 <div>Lo que se ve mientras no se procesa</div>
```

```
@Injectable(|
export class AuthGuard implements CanActivate {
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): boolean | UrlTree | Observable<boolean | UrlTree> | Promise<boolean | UrlTree>
    return !!localStorage.getItem('token');
```

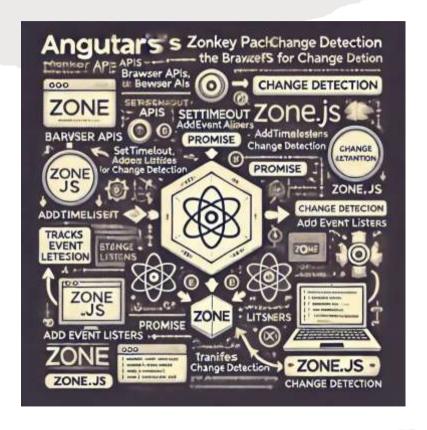
```
import { CanActivateFn } from '@angular/router';

export const authGuard: CanActivateFn = (route, state) => {
    // Example logic: Check if the user is authenticated
    return !!localStorage.getItem('token');
};
```

```
@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  intercept(reg: HttpReguest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const authToken = localStorage.getItem('token');
    if (authToken) {
      const clonedRequest = req.clone({
         Authorization: Bearer $(authToken),
      1);
      return next.handle(clonedRequest);
```

```
export const authInterceptor: HttpInterceptorFn = (req, next) => {
  const authToken = localStorage.getItem('token');
  if (authToken)
    const clonedRequest = req.clone({
      setHeaders: {
        Authorization: `Bearer ${authToken}`,
    });
    return next(clonedRequest);
  return next (req);
```

Zone - Zoneless



Signals

```
@Component({
  selector: 'app-root',
  template:
  <h1>{{title()}}</h1>
  changeDetection: ChangeDetectionStrategy.OnPush,
  encapsulation: ViewEncapsulation.None
export class AppComponent {
  title = signal('Hola DotNetMalaga') // -> WritableSignal<string>
```

Signals

```
@Component({
  selector: 'app-root',
  template: `
  <h1>{{title()}}</h1>
  <button (click)="title.set('Algo ha cambiado')">Cambiar título</button>
  changeDetection: ChangeDetectionStrategy.OnPush,
  encapsulation: ViewEncapsulation.None
export class AppComponent {
  title = signal('Hola DotNetMalaga') // -> WritableSignal<string>
```

Signals - Computed

```
@Component({
  selector: 'app-root',
  template:
  {{nombreCompleto()}}
  changeDetection: ChangeDetectionStrategy.OnPush,
  encapsulation: ViewEncapsulation.None
export class AppComponent {
  nombre = signal('Jesús');
  apellido = signal('Rodriguez');
  nombreCompleto = computed(() => `${this.nombre()} ${this.apellido()}`);
```

Signals - Computed

```
export class AppComponent {
 pedidos = signal([
    { fecha: new Date('2024-03-01'), producto: 'Portatil' },
    { fecha: new Date('2024-03-15'), producto: 'Monitor' },
    { fecha: new Date('2024-03-20'), producto: 'Teclado' }
  1);
 ultimoPedido = computed(() => {
    const listaPedidos = this.pedidos();
    if (listaPedidos.length === 0) return null;
    return listaPedidos.reduce((pedidoMasReciente, pedidoActual) =>
      pedidoActual.fecha > pedidoMasReciente.fecha ? pedidoActual : pedidoMasReciente
   );
  });
```

Signals - linkedSignal

```
@Component({
 imports: [NgFor],
   <select (change) = "changeShippingMethod($event)">
     <option *nqFor="let method of shippingMethods()" [value]="method">{{method}}/option>
   </select>
   {{selectedMethod()}}
 changeDetection: ChangeDetectionStrategy.OnPush,
 encapsulation: ViewEncapsulation.None
export class AppComponent {
 shippingMethods = signal(['DHL', 'FedEx', 'UPS']);
   return this.shippingMethods()[0];
 changeShippingMethod(event: Event) {
   this.selectedMethod.set((event.target as HTMLSelectElement).value);
```

Signals - Effect

```
export class AppComponent {
  shippingMethods = signal(['DHL', 'FedEx', 'UPS']);
  selectedMethod = linkedSignal(() => {
    return this.shippingMethods()[0];
  changeShippingMethod(event: Event) {
    this.selectedMethod.set((event.target as HTMLSelectElement).value);
  constructor() {
    effect(() => {
      window.localStorage.setItem('shippingMethod', this.selectedMethod());
    1);
```

Signals - Inputs

```
export class MyComponent {
  name = input.required<string>();
  selected = output<boolean>();
  twoway = model<string>();
}
```

¿Qué pasa con RXJS? ¿Lo reemplaza Signals?

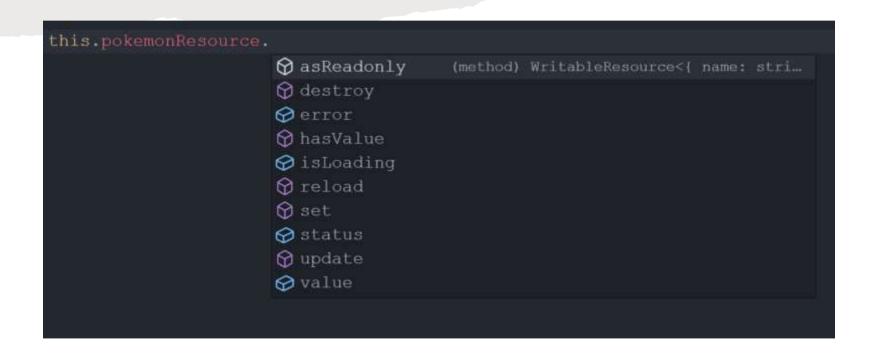
NO

Resource

```
pokemonResult = input.required<PokemonListResult>({ alias: 'pokemon' });
loaded = output<void>();

pokemonResource = resource({
   request: this.pokemonResult,
   loader: ({ request: pokemon, abortSignal }) =>
        fetch(pokemon.url, { signal: abortSignal }).then(res => res.json()),
});
```

Resource



¿Qué más?

- nxResource
- toSignal
- toObservable

```
busqueda = signal('');
busqueda$ = toObservable(this.busqueda).pipe(
   debounceTime(500),
   distinctUntilChanged()
);
```

¿Futuro?

- Adios Karma (por fin)
- Formularios con Signals
- Componentes con Signals
- Router con Signals
- ¿Nueva sintaxis para componentes?



DEMO













