

ANGULAR 17

Nuevas funcionalidades en Angular:

- **Estructura:**
 - No hay app.module.
 - Todos los componentes y aplicaciones son standalone por defecto.
 - Se crea automáticamente el app.routes.ts.
 - Importación de módulos: para importar módulos desde la raíz ya no tenemos el "app.module", así que ahora esas importaciones se hacen desde los "providers" de "app.config.ts".

```
export const appConfig: ApplicationConfig = {  
  providers: [  
    provideRouter(  
      routes, You, 5 hours ago • transitions add  
      withViewTransitions({  
        skipInitialTransition: true,  
        // onViewTransitionCreated(transitionInfo) {  
        //   console.log({ transitionInfo });  
        // }  
      })),  
    importProvidersFrom(  
      HttpClientModule  
    )  
  ],  
};
```

- Lazy Load

- Antes:

```
src > app > app.routes.ts > routes > loadComponent > then() callback
You, 6 minutes ago | 1 author (You)
1 import { Routes } from '@angular/router';
2
3 export const routes: Routes = [
4
5   {
6     path: "dashboard",
7     loadComponent: () => import("./dashboard/dashboard.component").then(c => c.DashboardComponent),
8   }
9 ];
```

- Ahora:

```
src > app > dashboard > dashboard.component.ts > ...
You, 15 seconds ago | 1 author (You)
1 import { CommonModule } from '@angular/common';
2 import { Component } from '@angular/core';
3
4 @Component({
5   standalone: true,
6   imports: [
7     CommonModule,
8   ],
9   templateUrl: './dashboard.component.html',
10  styles: [],
11 })
12 export default class DashboardComponent { }
13
```

```
src > app > app.routes.ts > routes > loadComponent
You, 3 minutes ago | 1 author (You)
1 import { Routes } from '@angular/router';
2
3 export const routes: Routes = [
4
5   {
6     path: "dashboard",
7     // loadComponent: () => import("./dashboard/dashboard.component").then(c => c.DashboardComponent ),
8     loadComponent: () => import("./dashboard/dashboard.component"),
9   }
10 ];
```

- **Señales:** en Angular las señales son wrappers que envuelven otros objetos o tipos de datos, pero para acceder a su valor hay que poner los paréntesis () como si fuera una función. [Más info](#)

```
<h2 class="text-2xl font-bold mb-5">if: {{ showContent() }}</h2>
```

- **Observables a señales:** `toSignal()` es una función que convierte un observable en una señal. Cada vez que el observable emite un valor la señal se actualiza.

```
import { toSignal } from '@angular/core/rxjs-interop';
```

```
public user = toSignal(
  this.route.params.pipe(
    switchMap( ( {id} ) => this.userService.getUserById( id ) )
  )
);
```

- **Señales de sólo lectura (asReadOnly):** se le asigna un valor inicial que ya no puede cambiarse.

```
public showContent = signal(false).asReadOnly;
```

- **Señales para estados en servicios:**

```
#state = signal<State>({
// la almohadilla indica que siempre es privado (en ES6) a diferencia de "private" que desaparece al compilar
  loading: true,
  users: []
});
```

You, 5 seconds ago • Uncommitted changes

- **Señales computadas:** son señales de sólo lectura, para que no puedan ser modificadas de ninguna forma. En la imagen de abajo se usan para acceder a propiedades de la señal de estado anterior, que era privada.

```
public users = computed( () => this.#state().users );
public loading = computed( () => this.#state().loading );
```

- **Defer:** es como un Lazy Load de componentes (sólo si son standalone). Se usa para programar como debe comportarse un componente o si debe aparecer otro cuando este tarda en cargar (mediante `@placeholder (minimun 500ms){ }` “minimun” puede medirse en segundos o milisegundos y es el tiempo mínimo que debe verse) o al darle carga diferida mediante triggers (se pueden usar varios a la vez).

- **Funcionalidad:**

```
@defer ( on idle ) {
  You, 1 second ago • Uncommitted changes
  <app-heavy-loaders-slow cssClass="bg-green-500" />
}@placeholder {

  <p class="h-[600px] w-full bg-gradient-to-r from-gray-200 to-gray-400 animate-pulse">Cargando...</p>
}

@defer ( on viewport ) {

  <app-heavy-loaders-slow cssClass="bg-purple-500" />
}@placeholder {

  <p class="h-[600px] w-full bg-gradient-to-r from-gray-200 to-gray-400 animate-pulse">Cargando...</p>
}
```

- **Triggers:**

When <condition>

Similar a una condición aplicable a un `@if`, puedes añadir la condición de carga

```
@defer (when cond) {
  <calendar-cmp />
}
```

Recuerda que se pueden mezclar.

```
@defer (on viewport; when cond) {
  <calendar-cmp />
} @placeholder {
  
}
```

Aquí se cargará cuando cualquiera de las dos condiciones se cumpla.

Posibles opciones “on”

Estas son las posibles opciones en el defer con “on”

Pipe	Descripción
on_idle	Se dispara cuando el navegador llega a un estado inactivo “idle”
on_viewport	Se dispara cuando el bloque entra al punto de vista del usuario. Por defecto se puede conectar el placeholder y otro elemento
on_interaction	Se dispara cuando el usuario interactúa con un elemento específico mediante un click o keydown
on_hover	Se dispara cuando el mouse pasa sobre el elemento o la referencia.
on_immediate	Se dispara tan pronto el cliente termina de renderizar la pantalla.
on_timer	Se dispara después de cierta duración de tiempo en MS milliseconds.

- Prefetch:

Prefetching

@defer permite precargar el código componente (no lo ejecuta, carga el código) y así tenerlo listo tan pronto sea el momento de llamarlo.

```
@defer (on interaction; prefetch on idle) {  
  <calendar-cmp />  
} @placeholder {  
    
}
```

- Control Flow

- @if / @else:

```
@if ( showContent()) {  
  <p>Hola Mundo</p>  
} @else {  
  <p>*****</p>  
}
```

- **@for:** track = es una bandera que se usa para definir al “ciclo for” cómo darle seguimiento a los elementos contenidos. Se puede usar el índice o cualquier propiedad única del item, incluso el propio item si este no se repite.

No sólo se pueden sacar los índices sino también el primero, el último, los pares, impares y el conteo total.

```
<ul>

@for (framework of frameworks(); track framework;
  let i = $index, first = $first, last = $last, even = $even, odd = $odd, total = $count ) {

  <li [class]="{
    'bg-blue-100': even && !first && !last,
    'bg-green-100': odd && !first && !last,
    'bg-red-100': first || last,
  }">
    {{ i + 1 }}/{{ total }} - {{ framework }}
  </li>
}

</ul>
```

- **@empty:** se muestra su contenido cuando “frameworks2()” esté vacío.

```
<ul>

  @for (framework of frameworks2(); track $index) {
    <li>{{ framework }}</li>
  } @empty {
    <li>No hay frameworks agregados</li>
  }

</ul>
```

- **@switch @case @default:**

```
<h2 class="text-3xl">{{ grade() }}</h2>

@switch ( grade() ) {

  @case ( "A" ) {

    <p>Más de 90</p>

  }

  @case ( "B" ) {

    <p>Más de 80</p>

  }

  @case ( "F" ) {

    <p>Suspendido</p>

  }

  @default {

    <p>Aprobado</p>

  }

}

</div>
```

- **@Input:** Ahora se le puede poner restricciones como “required” e incluso un alias. También se puede usar el “transform”, propiedad con la que transformar el valor de entrada antes de asignarlo a la instancia de directiva.

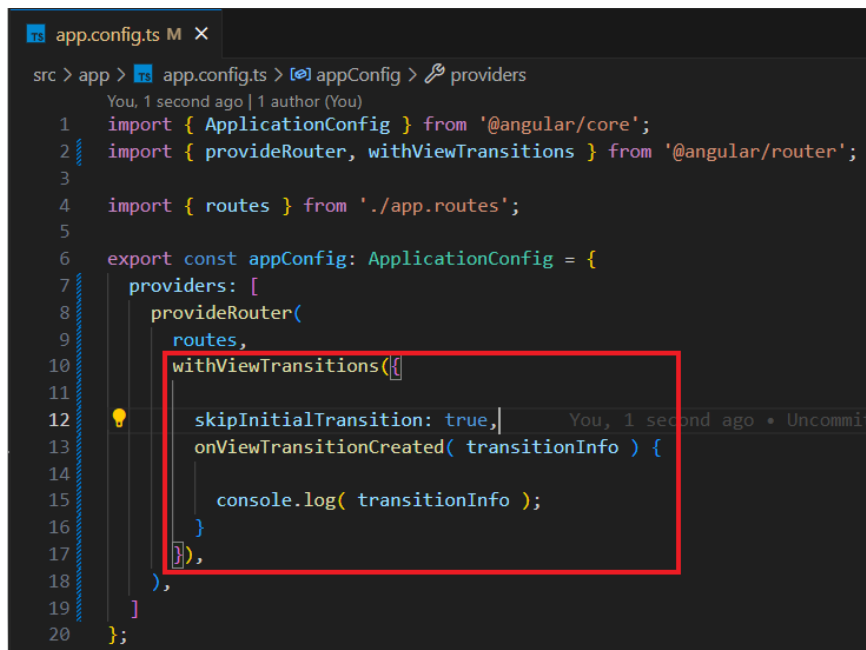
```
src > app > shared > title > title.component.ts > ...
You, 6 minutes ago | 1 author (You)
1 import { Component, Input, booleanAttribute } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3
You, 6 minutes ago | 1 author (You)
4 @Component({
5   selector: 'app-title',
6   standalone: true,
7   imports: [CommonModule],
8   template: '<h1 class="text-3xl mb-5">{{ pageName }} - {{ withShadow }}</h1>',
9   styles: [``]
10 })
11 export class TitleComponent {
12
13   @Input( { required: true, alias: "title" } ) pageName!: string;
14   @Input( { transform: booleanAttribute } ) withShadow: boolean = false;
15 }
```

El “booleanAttribute” devuelve “true” si la directiva está presente en el selector del componente y si no devuelve el valor de la variable.

```
<app-title title="Control Flow" withShadow />
```

Control Flow - true

- **View Transition API** (Aún no compatible con todos los navegadores):
 - Modificamos el “app.config.ts” para ver transiciones en los cambios de página con “**withViewTransitions**”, y además los configuramos para omitir la primera transición y ver información de las transiciones.



```
src > app > app.config.ts > appConfig > providers
You, 1 second ago | 1 author (You)
1 import { ApplicationConfig } from '@angular/core';
2 import { provideRouter, withViewTransitions } from '@angular/router';
3
4 import { routes } from './app.routes';
5
6 export const appConfig: ApplicationConfig = {
7   providers: [
8     provideRouter(
9       routes,
10      withViewTransitions({
11        skipInitialTransition: true,
12        onViewTransitionCreated( transitionInfo ) {
13          console.log( transitionInfo );
14        }
15      })
16    ],
17  ],
18 };
19
20
```

- **Hero Animation:** reconoce el mismo elemento en 2 páginas distintas y si están en distintas posiciones realiza una transición al cambiar entre ambas.


```

template: `
11
12
13 <app-title title="View Transition 1" />
14
15 <section class="flex justify-start">
16
17   <img
18     srcset="https://picsum.photos/id/237/200/300"
19     alt="Picsum"
20     width="200"
21     height="300"
22     style="view-transition-name: hero1"
23   >
24
25   <div
26     class="bg-blue-500 w-56 h-56"
27     style="view-transition-name: hero2"
28   ></div>
29 </section>
30
31
32

```

- **Nuevo sistema de detección de cambios:** el modo “changeDetection” ahora puede ponerse bajo demanda, en lugar del modo por defecto que busca cambios constantemente.

```
app > dashboard > pages > change-detection > change-detection.component.ts > ChangeDetectionComponent
```

```
You, 2 (property) Component.changeDetection?: ChangeDetectionStrategy | undefined
```

```
impo
```

```
impo The change-detection strategy to use for this component.
```

```
impo When a component is instantiated, Angular creates a change detector, which is responsible for propagating the component's bindings. The
```

```
impo strategy is one of:
```

```
You, 2
```

```
@Com
```

- `ChangeDetectionStrategy#OnPush` sets the strategy to `CheckOnce` (on demand).
- `ChangeDetectionStrategy#Default` sets the strategy to `CheckAlways`.

```
changeDetection: ChangeDetectionStrategy.OnPush,
```

```
template: `
```

- **Configuración de Path alias en TypeScript:** para evitar paths relativos.

```
tsconfig.json > { } compilerOptions > { } paths > [ ] @shared/* > [ ] 0
You, 1 second ago | 1 author (You)
/* To learn more about this file see: https://angular
{
  "compileOnSave": false,
  "compilerOptions": {
    "paths": {
      "@shared/*": [ "./src/app/shared/*" ]
    },
    "outDir": "./dist/out-tsc",
    "forceConsistentCasingInFileNames": true,
```

Así pasamos de la ruta relativa:

```
import { SidemenuComponent } from '../shared/sidemenu/sidemenu.component';
```

al alias:

```
import { SidemenuComponent } from '@shared/sidemenu/sidemenu.component';
```