ngDestroy and SetInput 7/20

RECURSOS MARCADORES

El hook `ngOnDestroy() & SetInput` tiene una importancia clave para el cuidado de nuestra aplicación. **Su funcionalidad más importante es la liberación de espacio en memoria** de variables para que no se acumule. Si esto llegara a suceder en tu aplicación, la misma podría volverse lenta y tosca a medida que toda la memoria del navegador es ocupada.

Liberando espacio de memoria

Todo el ecosistema Angular está basado en observables para el manejo asincrónico.

Cada vez que utilices un `subscribe()` para escuchar la respuesta de algún evento asincrónico (por ejemplo, el llamado a una API), es relevante realizar el respectivo `unsubscribe()` para liberar ese espacio en memoria.

RxJS

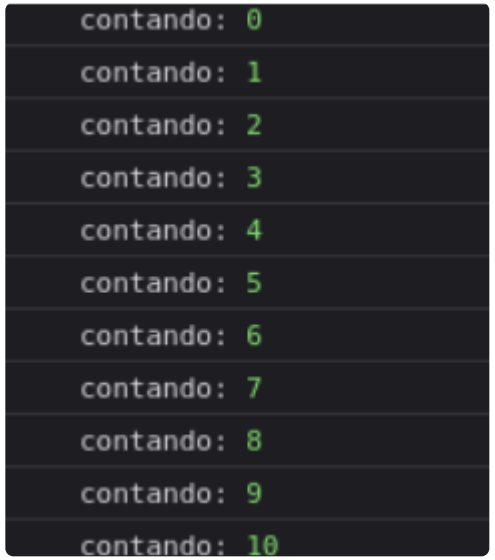
RxJS (Reactive Extensions Library for JavaScript) es una popular librería de Javascript para el manejo de observables. Si trabajas con Angular esta librería será tu mejor amiga.

Observa el siguiente ejemplo donde primero se importa `Subscription` desde `rxjs` para tipar la variable `suscription` . Guardamos el observable para posteriormente darlo de baja. También importamos

`interval` que devuelve el observable y genera un contador que emite una pulsación, en este ejemplo, cada 1000 milisegundos.

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Subscription, interval } from 'rxjs';

@Component({
  selector: 'app-test-name',
  templateUrl: './test-name.component.html',
  styleUrls: ['./test-name.component.less']
})
export class TestNameComponent implements OnDestroy, OnInit {

    count = interval(1000);
    suscription!: Subscription;

    ngOnInit(): void {
        this.suscription = this.count.subscribe(d => {
          console.log("contando:", d);
        })
    }

    ngOnDestroy(): void {
        this.suscription.unsubscribe();
    }

}
```

En el `ngOnInit()`, se está suscribiendo a la propiedad `this.count` para imprimir por consola, cada 1000 milisegundos, el contador. Podrás observar en la consola del navegador el contador corriendo:



Si nuestro código acabara aquí, cuando el componente es destruido, el contador continuaría ocupando memoria que ya no debería ser utilizada.

Para solucionar esto, guardamos en `this.suscription` el observable del contador y en `ngOnDestroy()` y llamamos al método `.unsubscribe()` para detener el contador.

Contribución creada con los aportes de Kevin Fiorentino.

Lecturas recomendadas

GitHub - platzi/angular-componentes at 6-step
https://github.com/platzi/angular-componentes/tree/6-step