DWC

(Desarrollo Web en entorno cliente)

Índice

1.- Comunicación entre componentes con Observable

Vamos a realizar un proyecto en el que tengamos dos componentes que compartan un valor contador. Ambos componentes accederán al valor del contador y ambos podrán modificar el contador reflejándose el cambio en los dos componentes.
Además cada componente dispondra de un boton para poder cancelar la suscripcion y otro para completar el observable El contador se incrementara en una unidad cada vez que se le indique desde cualquier componente

Para realizar este proyecto se debe crear:

- **Crear un servicio** con un observable a una variable numérica contador y con los métodos para notificar cambios cada vez que se incremente y un método para completar el observable
- **Crear dos componentes** componente1 y componente2. Cada componente mostrará el valor del contador y dispondrá de un botón para incrementar el contador, otro botón para cancelar la suscripción y un botón para completar el observable
- **Llamar a los componentes de forma independiente** desde el app-component

2.- Creación del servicio

Creamos un servicio dentro de la carpeta Services

```
ng g s Services/observables
```

En el ts del servicio:

- Importamos clase BehaviourSubject
- Declaramos el subject, en este caso una variable llamada contador. Además, la inicializamos a 0
- Definimos los métodos del servicio que afectan al observable:
  - incrementarContador, será el encargado de emitir cada valor nuevo del observable
  - completarObservable, será el encargado de finalizar la emisión de valores del observable.

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

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

  constructor() { }

  contador:BehaviorSubject<number>=new BehaviorSubject(0)

  /* getContador(): Observable<number> {
    return this.contador.asObservable();
  } */

  incrementarContador(){
    console.log('Emitimos valor con incremento del contador')
    this.contador.next(this.contador.value + 1)
  }

  completarObservable(){
    console.log('Completamos Observable, deja de emitir')
    this.contador.complete()
  }
}
```

3.- Creación de los componentes

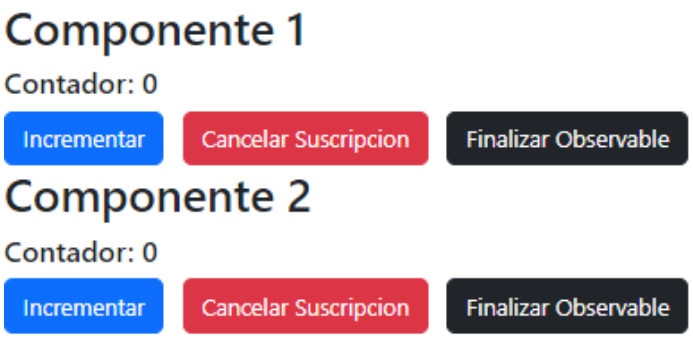
Creamos los dos componentes en la carpeta Components. Ambos componentes tendrán la misma interface y el mismo código en el controlador

```
ng g c Components/componente1
ng g c Components/componente2
```

En el html

```
<h5>Contador: {{ contador }}</h5>
<button (click)="incrementarContador()" class="btn btn-primary">
    Incrementar
</button>
 
<button (click)="cancelarSuscripcion()" class="btn btn-danger">
    Cancelar Suscripcion
</button>
 
<button (click)="completarObservable()" class="btn btn-dark">
    Finalizar Observable
</button>
```

El resultado



Ahora definimos **en el controlador** los métodos asociados a los clicks de los botones.

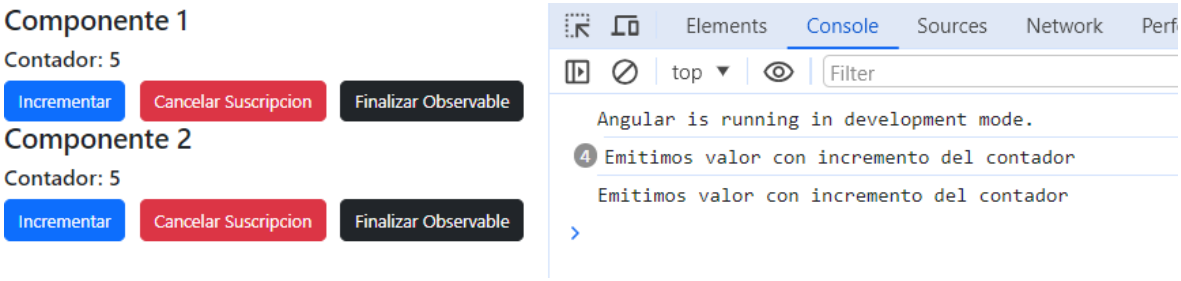
Deberemos hacer lo siguiente:

- Importamos el servicio
- Inyectar el servicio en el constructor
- Declaramos las variables:
  - contador de tipo number para almacenar el valor del observable e interpolar en la vista
  - miSuscripcion de tipo any para almacenar la referencia a la suscripción del observable y así poder cancelar en cualquier momento
- Implementamos el método ngOnInit para suscribirnos al observable contador desde la carga del componente y asi estar ya siempre suscritos para recibir los cambios.
- Implementamos los métodos asociados a los botones:
  - incrementarContador, será el encargado de llamar al servicio para que se emita un nuevo valor con el incremento del contador.
  - cancelarSuscripcion, será el encargado de cancelar la suscripción del componente al observable
  - completarObservable, será el encargado de llamar al servicio para completar el observable. Dejará de emitir datos

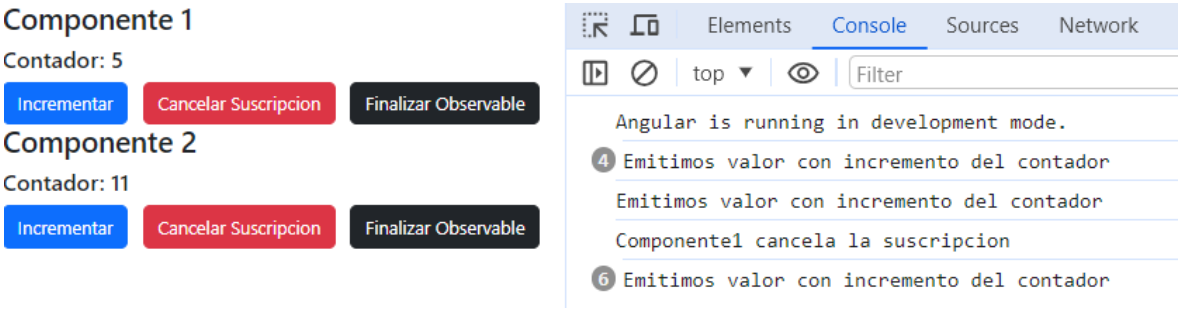
```typescript
import { Component } from '@angular/core';
import { ObservablesService } from '../../Services/observables.service';

@Component({
  selector: 'app-componente1',
  standalone: true,
  imports: [],
  templateUrl: './componente1.component.html',
  styleUrl: './componente1.component.css'
})
export class Componente1Component {

  constructor(private miServicio:ObservablesService){}

  contador!:number
  miSuscripcion:any

  ngOnInit(){
    this.miSuscripcion=this.miServicio.contador.subscribe({
      next: (dato) => this.contador=dato,
      error: (err) => console.error(err),
      complete: () => console.log('Contador completado'),
    })
  }

  incrementarContador(){
    this.miServicio.incrementarContador()
  }

  cancelarSuscripcion(){
    console.log('Componente1 cancela la suscripcion')
    this.miSuscripcion.unsubscribe()
  }

  completarObservable(){
    this.miServicio.completarObservable()
  }
}
```

El resultado

Botón de incrementar



Botón de cancelar suscripción para componente1



Botón de finalizar observable