Formación Interna

Formación I: Observables y Suscripciones.

1: Observables para Cargar Datos Asíncronos.

Los observables en Angular permiten manejar flujos de datos asíncronos. Esto es especialmente útil para cargar datos desde APIs u otras fuentes externas, ya que permite suscribirse y recibir los datos en el momento en que están disponibles.

Un componente A solicita los datos a un servicio que realiza la llamada http.

En el ejemplo al hacer click en `Ejecutar ejemplo` llamamos a loadData().

IMPORTANTE:

- En el HTML tendremos la condición de mostrar la sección solo si data es distinto de nulo para ello es importante tener en cuenta su inicialización y el control de su valor: **data: string | null = null;**

```
// Ejemplo 1: Cargar datos asíncronos
loadData() {
  this.dataService.getMockData().subscribe(data => {
    this.data = data;
    this.dataService.onShowLoading.next(false);
  });
}
```

En este ejemplo cuando recibimos los datos, data deja de ser nulo por lo que el HTML mostrará la sección correspondiente.

Además para este caso estamos seteando una suscripción a false para quitar el spinner.

2. Mostrar / Ocultar componentes dinámicamente.

Subject permite emitir eventos en tiempo real sin preocuparte de que los nuevos suscriptores reciban valores previos.

BehaviorSubject permite que los nuevos suscriptores reciban de inmediato el valor actual (como un estado compartido en un servicio Angular).

Inicializamos en el pop up las variables:

```
public onSendNotification = new Subject<void>();
public onSendNotification$ = this.onSendNotification.asObservable();
```

Al hacer click en el botón enviar dentro del pop up emitimos el evento a quien se haya suscrito:

```
sendNotification() {
  this.onSendNotification.next();
}
```

El componente que se suscribe cada vez que recibe la señal cambia le valor de la variable ‘showNotification` lo que permite mostrar u ocultar esa sección desde el pop up.

```
this.popUpSub = this.popUpComponent.onSendNotification$.subscribe(() => {
  this.showNotification = !this.showNotification;
});
```

Resumen de las Diferencias

Característica	**Subject**	**BehaviorSubject**
Valor inicial	No	Sí (se requiere un valor inicial)
Último valor emitido	No lo guarda	Lo guarda y lo emite a nuevos suscriptores
Emisión a suscriptores nuevos	Solo valores emitidos tras la suscripción	Último valor emitido y valores futuros
Ideal para	Eventos, flujos de datos en tiempo real	Estado compartido, almacenamiento de estados

Cuándo Usar Subject o BehaviorSubject:

- **Usa Subject cuando:**
  - No necesitas mantener un valor inicial o actual.
  - Quieres emitir eventos en tiempo real sin preocuparte de que los nuevos suscriptores reciban valores previos.
- **Usa BehaviorSubject cuando:**
  - Necesitas almacenar el último valor emitido.
  - Quieres que los nuevos suscriptores reciban de inmediato el valor actual (como un estado compartido en un servicio Angular).

3. Manejo de Suscripciones y Ciclo de Vida.

Es importante cancelar las suscripciones de los observables para evitar fugas de memoria en Angular. Esto se logra usualmente usando `takeUntil` junto con un `Subject` o manejadores de ciclo de vida como `OnDestroy`. También podemos usar un objeto de tipo `Subscription` y añadir a este objeto todos los eventos que necesitemos. En el `OnDestroy` debemos liberar este objeto al igual que hacemos con el `Subject`.

```
  startDataStream() {
    interval(1000)
      .pipe(takeUntil(this.destroy$))
      .subscribe((value) => {
        this.streamData = value;
      });
  }

  stopDataStream() {
    this.destroy$.next();
  }

  ngOnDestroy() {
    this.destroy$.next();
    this.destroy$.complete();
    this.popUpSub?.unsubscribe();
  }
}
```