Observables et RxJS

Angular utilise **RxJS** (Reactive Extensions for JavaScript) pour la gestion asynchrone avec les **Observables**.

Qu'est-ce qu'un Observable ?

Un **Observable** est un flux de données asynchrones pouvant émettre plusieurs valeurs au fil du temps.
Des exemples :
- Les requêtes HTTP (`HttpClient`).
- La gestion des événements utilisateur.
- La communication entre composants.

Création et Souscription à un Observable

Création d'un Observable

```
import { Observable } from 'rxjs';

const monObservable = new Observable(observer => {
  observer.next('Première valeur');
  observer.next('Deuxième valeur');
  observer.complete(); // Fin du flux
});
```

Souscription

```
monObservable.subscribe({
  next: valeur => console.log(valeur), // Réception des valeurs
  complete: () => console.log('Observable terminé'),
});
```

Opérateurs RxJS essentiels

Les opérateurs permettent de **manipuler les flux de données** dans RxJS.

Opérateurs de transformation

- `map(val => ...)` : Transforme chaque valeur.

  ```
  observable.pipe(map(val => val * 2));
  ```

- `filter(val => ...)` : Filtre les valeurs.

  ```
  observable.pipe(filter(val => val > 10));
  ```

- `tap(val => console.log(val))` : Effectue une action sans modifier la valeur.

Opérateurs de combinaison

- `merge(observable1, observable2)` : Fusionne plusieurs observables.
- `concat(observable1, observable2)` : Exécute les observables l'un après l'autre.

Opérateurs temporels

- `debounceTime(500)` : Ignore les valeurs rapides, utile pour limiter les appels API.
- `throttleTime(1000)` : Limite la fréquence d'émission des valeurs.

Observables vs Promises

Observables	**Promises**
Émettent **plusieurs valeurs**	Résolvent **une seule valeur**
Annulables avec `unsubscribe()`	Non annulables
Fonctionnent **par paresse** (lazy)	Exécutées immédiatement
Opérateurs RxJS pour la transformation	`then()` et `catch()` pour la gestion

Conversion entre Observables et Promises

- **Observable → Promise** :

```
monObservable.toPromise().then(val => console.log(val));
```

- **Promise → Observable** :

```
import { from } from 'rxjs';

const monObservable = from(fetch('https://api.exemple.com/data'));
```

Exemple : Utilisation avec HttpClient

Requête HTTP avec HttpClient et Observables

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

export class MonService {
  constructor(private http: HttpClient) {}
```

```
  getData(): Observable<any> {
    return this.http.get('https://jsonplaceholder.typicode.com/posts');
  }
}
```

Souscription dans un composant

```
this.monService.getData().subscribe(data => {
  console.log('Données reçues :', data);
});
```

Annulation d’un Observable (unsubscribe)

```
import { Subscription } from 'rxjs';

let subscription: Subscription = this.monService.getData().subscribe();
subscription.unsubscribe(); // Stoppe l'écoute de l'Observable
```