**Observable**

An Observable is a core concept in RxJS (Reactive Extensions for JavaScript) and Angular. It's used to handle asynchronous data streams — like data from APIs, user inputs, or timers.

**Key Features:**

1)It can emit multiple values over time.

2)It is lazy: It doesn't do anything until you subscribe to it.

3)It is unicast: Each subscription gets its own execution.

4)Comparison with Promises: Supports features like cancellation and operators.

**Example:**

import { Observable } from 'rxjs';

const observable = new Observable(observer => {

observer.next('Hello');

observer.next('World');

observer.error('Something went wrong!');

observer.complete();

});

observable.subscribe({

next: value => console.log(value),

error: err => console.error('Error:', err),

complete: () => console.log('Done')

});

**Subject**

A Subject is a special type of Observable that allows multicasting — i.e., it shares the same execution among multiple subscribers.

**Key Features:**

1)Acts as both Observable and Observer.

2)You can emit values manually using .next().

3)It is starts emitting immediately.

**Example:**

import { Subject } from 'rxjs';

const subject = new Subject<string>();

subject.subscribe(value => console.log('Subscriber A:', value));

subject.subscribe(value => console.log('Subscriber B:', value));

subject.next('Hello from Subject');

**Behavior Subject**

A Behavior Subject holds a current value and emits it immediately to new subscribers. It's great when you want to maintain and access the latest state.

**Key Features:**

1)Requires an initial value.

2)Emits the latest value to new subscribers.

**Example:**

import { BehaviorSubject } from 'rxjs';

const behaviorSubject = new BehaviorSubject<string>('Initial Value');

behaviorSubject.subscribe(val => console.log('Subscriber A:', val));

behaviorSubject.next('New Value');

behaviorSubject.subscribe(val => console.log('Subscriber B:', val));

**Replay Subject**

A Replay Subject takes a specified number of emitted values and replays them to new subscribers.

**Key Features:**

1)Can replay past values to new subscribers.

2)Useful when you want subscribers to catch up with previously emitted values.

**Example:**

import { ReplaySubject } from 'rxjs';

const replaySubject = new ReplaySubject<string>(2);

replaySubject.next('Value 1');

replaySubject.next('Value 2');

replaySubject.next('Value 3');

replaySubject.subscribe(val => console.log('Subscriber A:', val));