RxJs

RxJS: Reactive Extension for JavaScript

# Introduction

**RxJS (Reactive Extensions for JavaScript)** is a library that allows you to **work with asynchronous data as streams**, using **Observables** to emit data over time and **operators** to transform, filter, and combine these streams efficiently.

Example:

"In Angular, HTTP requests using HttpClient return an Observable, which represents a stream of data over time. We can use RxJS operators like map to transform the response, for example extracting only the user names, and catchError to handle any errors gracefully. In the component, we subscribe to the Observable to actually execute the request and receive the data. This approach keeps the code declarative, reactive, and easy to manage for asynchronous operations."

# Stream:

"A stream is a sequence of data or events over time. In RxJS, we represent streams using Observables, which can emit multiple values asynchronously. You can listen to a stream, transform its data using operators like map or filter, and handle events as they occur, such as clicks, API responses, or timers."

# Difference between Array and Streams

| **Aspect** | **Array** | **Stream (Observable)** |
| --- | --- | --- |
| **Nature** | Collection of data **available all at once** | Sequence of data/events **over time** |
| **Execution** | **Eager** – data exists immediately | **Lazy** – emits data only when subscribed |
| **Size** | Fixed (or pre-defined) | Can be infinite (e.g., timer, clicks) |
| **Access** | Indexed access, synchronous | Event-driven, asynchronous |
| **Modification** | Can directly push/pop elements | Cannot modify directly; operators transform streams |
| **Use Case** | Static data | Dynamic or asynchronous data like clicks, HTTP requests, WebSocket messages |

"An array is a static collection of values available all at once, while a stream (Observable) represents a sequence of values that arrive over time. Arrays are synchronous, streams are asynchronous and lazy—they only produce values when you subscribe. Streams are ideal for events, HTTP responses, and timers, whereas arrays are for static data."

# Key elements: Observable, observer, subscription

Observable:

An **Observable** is a **stream of data or events** that can emit **multiple values over time**. Think of it as a blueprint for a sequence of asynchronous values.

import { of } from 'rxjs';

const numbers$ = of(1, 2, 3); // Observable emitting 3 numbers

Observer:

An **Observer** is an object that **reacts to values** emitted by an Observable. It defines **how to handle data, errors, and completion**.

Has three callbacks:

* next → handles emitted values
* error → handles errors
* complete → handles completion of the stream

const observer = {

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

error: (err: any) => console.error(err),

complete: () => console.log('Stream completed')

};

Subscription:

A **Subscription** represents the **execution of an Observable**. It allows you to **start receiving values** and **unsubscribe** to stop listening.

Created when you call subscribe() on an Observable.

Provides a way to **clean up resources** and **prevent memory leaks**.

const numbers$ = of(1, 2, 3);

const subscription = numbers$.subscribe(observer);

// To stop receiving values

subscription.unsubscribe();

"In RxJS, an Observable defines a stream of data that can emit multiple values over time. An Observer is an object that reacts to the values, errors, and completion of that stream. When we subscribe to an Observable, it returns a Subscription, which allows us to start receiving data and also unsubscribe to clean up resources."

# Types of Observables: