**Reactive Programming:**

* reactive programming is a way of writing code where your program reacts to changes or new data as soon as it happens.
* Your code responds to data/events as they come, instead of constantly checking for updates.
* Ex/- if we subscribe to particular channel in YouTube the notification we received from the channel if any updates comes instead of waiting.

**Why Use Reactive Programming**

* Handles lots of data/events efficiently (good for high-traffic apps).
* Makes your program more responsive and less likely to “freeze”.
* Better at using computer resources, especially with things like network calls, files, or databases.

**Workflow with example**

* Publisher: The newspaper company (data source).
* Subscriber: You (the person who wants the data).
* Subscription: The link between you and the publisher.
* Event/Data: The newspaper itself.

The Flow:

* You “subscribe” to data/events.
* When the data is ready, the “publisher” sends it to you.
* You react to it (read, process, etc.).
* If there’s an error, the publisher can tell you.
* When there’s no more data, the publisher says “completed”.

**Popular libraries in Reactive Stream API**

* RxJava
* Project Reactor (used in Spring WebFlux)
* built-in java.util.concurrent.Flow.

**Core Types:**

* **Publisher:** Sends data/events.
* **Subscriber:** Receives and reacts to data/events.
* **Subscription:** Manages the connection, allows you to say “stop” if you want.
* **Processor:** Both a subscriber (can receive) and publisher (can send).

**Comparison:**

| **Library** | **Best For** | **Features & Ecosystem** | **Ease of Use** | **When to Avoid** |
| --- | --- | --- | --- | --- |
| **Project Reactor** | Spring/modern Java apps | Rich, modern, Spring-native | Easiest | Not for < Java 8, non-Java |
| **RxJava** | Android, cross-platform | Huge, cross-language | Moderate | Overkill for simple apps |
| **java.util.concurrent.Flow** | Building libraries, basics | Just interfaces, no tools | Hardest | Complex reactive needs |

* Spring Boot? Modern Java? → Use Project Reactor
* Android? Already know Rx? → Use RxJava
* Just want basic interfaces? → Use java.util.concurrent.Flow

**Types in Reactive Programming**

* **Flux**: 0 to many items (like a stream of events).
* **Mono**: 0 or 1 item (like a single response, e.g., HTTP request).

**Which to use?**

* Use **Mono** if you expect a single result (e.g., fetch one user).
* Use **Flux** if you expect many results (e.g., stream of messages).

**Event looping** = a program that sits, waits for something to happen, reacts, and then waits again.