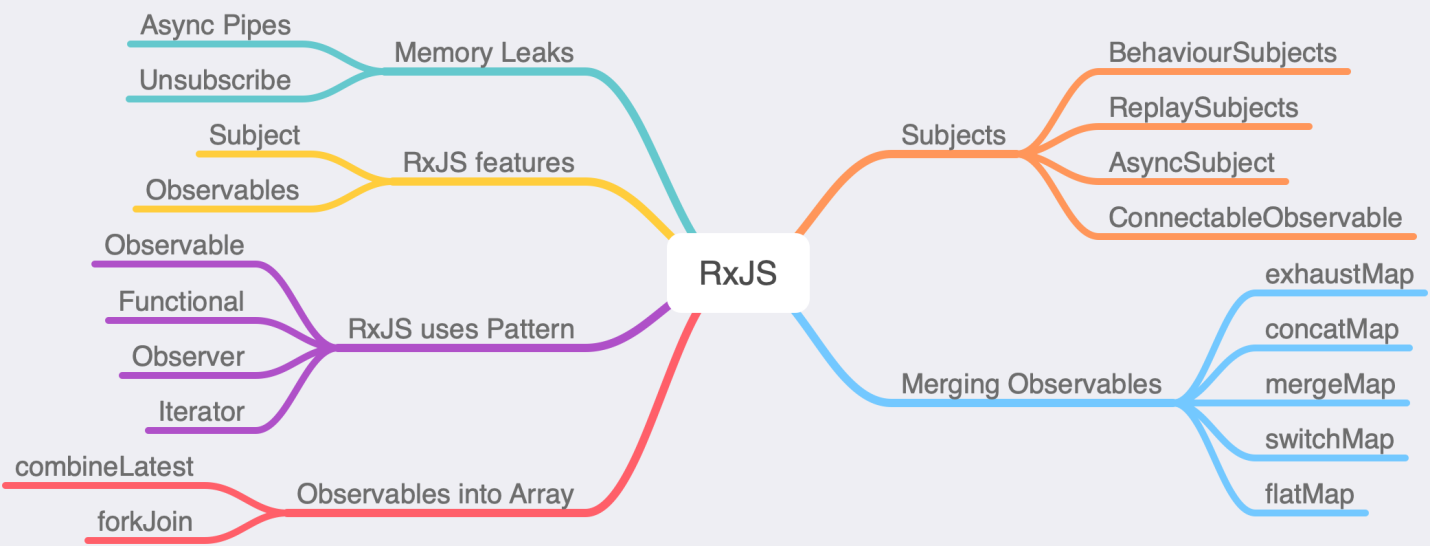
**What and Why RxJS?**

ReactiveX is a combination of the **observer pattern, iterator pattern, and functional programming**.



* **Observable pattern:** Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
* **Iterator Pattern:** Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.
* **Functional Programming:** Functional programming is a programming paradigm where we use a pure function, avoiding the shared state, and mutable data. **Currying in JavaScript is one of the great examples of functional programming.**

RxJS is an event-based and asynchronous program that uses the observable sequence.

**Observable vs Promise:**

In general, promises handle single events that fail or succeed.

Observable is a set of the stream that can handle no to many events. Observables are preferred over promises because they can be **aborted**, **retry** in case if it fails, and much more!

The good part is a promise can be converted to Observables when required!

RxJS operators that accept Observables can also accept a promise as well. You just need to change promise to observable.

import { from } from 'rxjs';  
const observable = from(promise)

or

import { defer, fromEvent, interval } from 'rxjs';  
Observable.defer(p => Promise())