Table of Contents

**Type chapter title (level 1)1**

Type chapter title (level 2)2

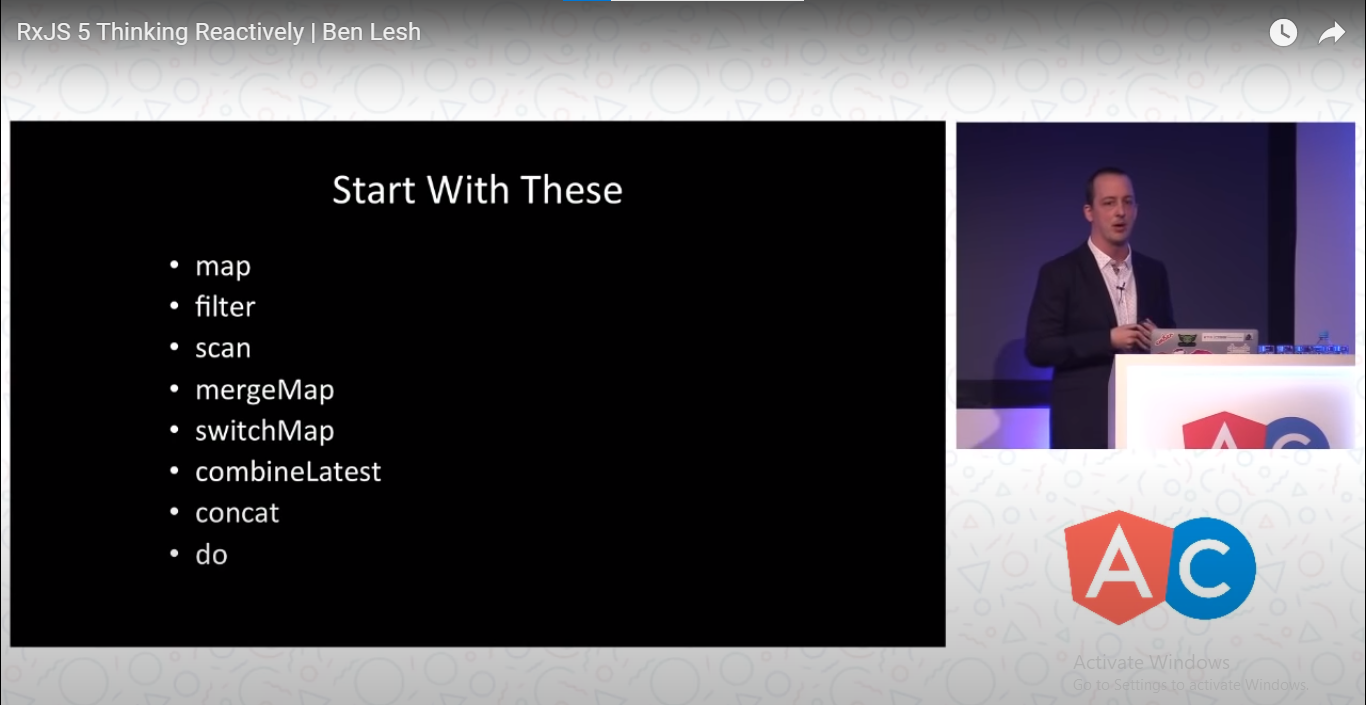
Type chapter title (level 3)3

**Type chapter title (level 1)4**

Type chapter title (level 2)5

Type chapter title (level 3)6

# Các operators hay dùng trong RxJS



Hình 1 Revial operations

A black screen with white text

Description automatically generated

Hình 2 Declarative Subscriptions

# Chú ý về RxJS và Redux Obseverable

## Cần gì để học Obseverable

* Redux
* RxJS
* Reactive Programing

## Obseverable là gì

RxJS là “lodash for async”

Ý tưởng đầu tiên của Obseverable ra đời 15 năm trước, từ 1 dự án của Microsoft với ý tưởng tích hợp C# vào trong JS

[RxJS + React + React is amazing combo](https://www.youtube.com/watch?v=AslncyG8whg&ab_channel=NetflixEngineering)

Hiện đang được ứng dụng bởi các dự án của Neflix

Bản chất của RxJS là sự kết hợp của Observer pattern *where Observables or Subjects notify Observers of state change* với functional program

Tương tự như Redux cũng sử dụng Observer pattern, subcriber và listener để update state

### Đặt vấn đề

Quản lý state trong dự án React là 1 vấn đề phức tạp, vì vậy các dev đã phát triển Redux (tham khảo thêm về Dan Abramov)

Redux giúp cho việc quản lý state đơn giản hơn(nhưng không có nghĩa là dễ hơn)

Tùy vậy Redux không hỗ trợ xử lý async

Một số đặc điểm của Redux:

* Cung cấp các phương thức để thao tác với *state* thông qua **actions** và  **reducers**
* **Action:** là 1 kiểu dữ liệu, action trả lời câu hỏi “what” has (or should) happen mà không mô tả “how it should happen”

// Ví dụ action

let count;

let action = {

    type: "INCREMENT",

    payload: "Increase count by 1"

}

Như ở ví dụ trên chúng ta có thể hiểu được action là gì và làm gì nhưng ta không hiểu được nó được làm như thế nào

* **Reducer**: là 1 pure function với input là *previous state* và *action,* output là 1 state đã được xử lý logic

### Một số tác vụ với async

* User interactions
* AJAX
* Timers/ Animations (Thư viện React-spring)
* Web Sockets
* Work Workers, etc

Một số tác vụ trên có thể xử lý synchronously, tuy vậy khi cần nhiều tính năng hơn, ví dụ:

* AJAX cancellation/composing
* Debounce/throttle/bufer
* Drag and drop
* Web Sockets

Với những tác vụ trên chúng ta không thể xử lý sync, mà cần async, tuy vậy async của JS chỉ cung cấp callback và Promise để xử lý async( async await về bản chất cũng trả về promise)

Promise có 1 số vấn đề nếu chúng ta sử dụng plain JS sẽ rất dài và khó maintain, đồng thời Promise có 4 đặc điểm :

* **Guaranteed future** : Promise sẽ **luôn luôn trả về 1 kết quả** và không thể bị cancel no matter what
* Immutable
* **Single value**
* Caching

Lấy ví dụ nếu user click vào 1 phim nhưng khi promise chưa trả về kết quả, user đã đổi sang 1 phim khác

Chúng ta cần 1 method để xử lý những trường hợp như vậy

Vấn đề thứ 2 là single value , trong 4 tác vụ xử lý async đã nếu trên thì chỉ có AJAX là single Value

Đây là lý do tại sao chúng ta sử dụng Observables

## Các tính chất của Observables

* Stream of zero, one or more values
* Over any amount of time: có nghĩa là dimension của Observables là time chứ không phải space
* Cancellable

Stream are a set, with a dimension of time

* Observables are *lazy:* Về bản chất thì Observer là các function, và function chỉ chạy khi được call, vì vậy nên chúng *lazy,* nói cách khác chỉ khi có 1 subcribe gọi tới chúng thì chúng mới hoạt động
* Khi Observables kết hợp với Redux thì “nhận vào 1 action và trả ra 1 actions”

### Khi nào sử dụng Observables

Việc lập trình và kiểm soát các tác vụ bất đồng bộ dựa trên thời gian sẽ dễ hơn nếu sử dụng redux-observable

### Sử dụng Observables

Observables sẽ ngưng truyền vào values khi

* Error() is called
* Complete() is called
* Unsubscribe is called