nguyen minh quan 20158325

[Company name]  [Company address]

RxJS

Table of Contents

[1. Rebuild Observer 2](#_Toc143617527)

[2. Các operators 3](#_Toc143617528)

[a. Map() 3](#_Toc143617529)

[b. Bóp giá trị trả lại với throttleTime() 5](#_Toc143617530)

[c. Merge map () 6](#_Toc143617531)

[d. Switch map() 7](#_Toc143617532)

[e. Of và from 9](#_Toc143617533)

[f. Pipe() 9](#_Toc143617534)

# Rebuild Observer

const observer = {

    next: function (value) {

        console.log(value)

    },

    error: function (error) {

        console.log(error)

    },

    complete: function () {

        console.log('complete')

    }

}

Các method của object observer

const button = document.querySelector("button")

const subscription = Rx.Observable.create(obs => {

    // obs.next('A Value');

    // setTimeout(() => {

    //     obs.error()

    // }, 2000);

    // obs.next('2nd value')

    button.onclick= function (event) {

        obs.next(event)

    }

})

.subscribe(observer);

Select button và nhúng event “click” vào để gọi method next()

Chú ý nếu không có sự kiện unsubscribe thì với 1 streams infinity sẽ có thể xảy ra hiện tượng memory leek khi ứng dụng trở nên phức tạp hơn, trong trường hợp này có thể gắn 1 biến subscription và unsubscribe

// setTimeout(() => {

//     subscription.unsubscribe()

// }, 5000);

# Các operators

## [Map](https://www.youtube.com/watch?v=-nYQJkMpOHU&list=PL55RiY5tL51pHpagYcrN9ubNLVXF8rGVi&index=3&pp=iAQB&ab_channel=Academind)()

const observable = Rx.Observable.interval(1000)

Khởi tạo observable tự động đếm mỗi 1s

const observer = {

    next: function (value) {

        console.log(value);

    }

}

Tạo 1 observer log value ra console

observable

.map((value) => {

    return value\*3

})

.subscribe(observer)

Subscribe observer vào observable, ta được bộ đếm trên console trả lại thời gian từ lúc bắt đầu chạy chương trình

A screenshot of a computer

Description automatically generated

Hình Kết quả hàm đếm thời gian

Dùng hàm map() nhân value nhận vào từ subcription lên 3

observable

.map((value) => {

    return value\*3

})

.subscribe(observer)

Trả lại giá trị

A screenshot of a computer

Description automatically generated

Hình Thay đổi value trả về vói map()

## Bóp giá trị trả lại với throttleTime()

observable

.map((value) => {

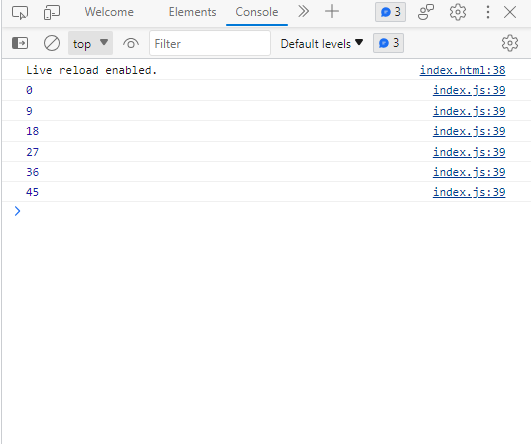
    return value\*3

})

.throttleTime(2000)

.subscribe(observer)

Sau mỗi 2s thì mới nhận kết quả gửi đến từ subscriber



Hình Kết quả trả về với throttletime

## [Merge map ()](https://www.youtube.com/watch?v=b59tcUwfpWU&list=PL55RiY5tL51pHpagYcrN9ubNLVXF8rGVi&index=11&ab_channel=Academind)

mergeMap() kết hợp 2 observable lại thành 1 và có khả năng xử lý 2 luồng data khác nhau, method này come in handy khi chúng ta quyết định xử lý 1 data dựa trên 1 data còn lại

Tạo 2 input và 1 <p> để in giá trị ra màn hình

Khai báo 2 observable

const obs1 = Rx.Observable.fromEvent(fistName,'input')

const obs2 = Rx.Observable.fromEvent(lastName,'input')

Để sử dụng data từ 2 obs không có scope liên quan thì ta dùng mergeMap kết hợp với gọi chồng giá trị

obs1.mergeMap((event1) => {

    return obs2.map((event2) => {

        return `${event1.target.value}  ${event2.target.value}`

    })

}).subscribe((combinedValue) => {

    span.textContent = combinedValue

})

Chú ý:

return obs2.map((event2) => {

Return giá trị obs2 bằng method map là 1 trick để thay đổi value đầu ra, giúp kết hợp value 1 và value 2

Cuối cùng subscibe và trả lại combinedValue ra span , in ra màn hình

## Switch map()

Switch map giúp chúng ta đổi data giữa 2 luồng chạy với nhau và cancel luồng còn lại

Ví dụ nếu subscribe obs2 từ event click obs1 để tạo tính năng khi click button thì bắt đầu đếm, ta sẽ được kết quả sau

A screenshot of a computer

Description automatically generated

Hình kết quả without switchmap

const button = document.querySelector("button")

const obs1 = Rx.Observable.fromEvent(button,'click')

const obs2 = Rx.Observable.interval(1000)

obs1.subscribe((event) => {

    return obs2.subscribe((value) => {

        console.log(value);

    })

})

Các vòng chạy sẽ bị conflict với nhau vì không có sự kiện unsubcribe

Vì vậy ta có thể dùng mergeMap để switch giữa 2 streams và cancel stream còn lại

obs1.switchMap((event) => {

    return obs2

}).subscribe((value) => {

    console.log(value);

})

switchMap trả lại obs2, sau đó subscribe sẽ nhận value là obs2 và in ra màn hình

A white sheet with black lines

Description automatically generated

Hình switchmap

Kết quả không còn bị conflict sau nhiều lần click

## Of và from

Đều để xử lý các sequence data

Of tạo ra các observable từ string, objects hoặc array, thường dùng khi làm việc với Models hoặc interfaces

From nhận vào 1 array hoặc object dạng array, promise, vòng lặp

## Pipe()

Giống compose trong lodash nhưng duyệt toán tử từ trái sang phải