Report

Angular tutorial for beginners  
(19/07/2023)

# RxJS:

## Observable:

Một tập các lazy push cấu thành từ nhiều giá trị.

### Pull & Push:

### Pull: Consumer xác định khi nào nó nhận được giá trị từ producer. Điều này được thực hiện bằng cách gọi một phương thức trên consumer.

### Push: Producer xác định khi nào để gửi giá trị đến consumer. Consumer không biết khi nào sẽ nhận được giá trị tiếp theo.

### Observable trong Angular sử dụng cơ chế push để gửi dữ liệu liên tục giữa các thành phần trong ứng dụng. Các đối tượng Observable phát ra các giá trị dữ liệu theo thời gian, trong khi các đối tượng Observer đăng ký để nhận được các giá trị này

### Observable như là sự khái quát hóa của Function:

### func.call() có nghĩa là "cho tôi một giá trị sync"

### observable.subscribe() có nghĩa là "cung cấp cho tôi bất kỳ số lượng giá trị nào, sync hoặc async".

### Các giai đoạn của Observable:

### Creating Observables

### Subscribing to Observables

### Executing Observables:

### Trong một Executing Observables, các thông báo Next từ 0 đến vô hạn có thể được gửi. Nếu một thông báo Error hoặc Complete được gửi, thì không có gì khác có thể gửi được sau đó.

### Disposing Observable Executions:

### Unsubscribe

## Observer:

Observer là đối tượng sử dụng giá trị của Observable

Observer hiểu đơn giản là một tập các call back (next, error, complete)

## Operators:

### Khái niệm

Operator là Function và có 2 kiểu sau:

### Pipeable Operators

### Pipeable Operator là một function lấy một Observable làm đầu vào và trả về một Observable khác.

### Pipeable Operator là một pure operator (Observable trước đó không bị sửa đổi).

### Operator factory functions gồm: [filter(...)](https://rxjs.dev/api/operators/filter) và [mergeMap(...)](https://rxjs.dev/api/operators/mergeMap).

### Creation Operators

### Creation Operators có thể được gọi là standalone functions dùng để tạo Observable mới

### Piping:

### Chúng ta có thể kết hợp nhiều Pipeable Operators bằng cách sử dụng phương thức pipe của observable: observable.pipe(pipeOp1(), pipeOp2(), pipeOp3(), pipeOp4());

### pipeOp()(observable) không bao giờ được sử dụng ngay cả khi chỉ có một toán tử. Thay vào đó người ta sẽ sử dụng observable.pipe(pipeOp()).

### Higher-order Observables

### Một loại Observable đặc biệt, nó phát ra các Observable khác thay vì các giá trị thông thường, trường hợp Observables lồng vào nhau nhiều lớp sẽ rất khó để xử lý.

### Các operater thông dụng:

### concatAll() Chuyển đổi một Observable bậc cao thành một Observable bậc nhất bằng cách nối các Observable bên trong theo thứ tự.

## Marble diagrams