**Cấu trúc RxJs**

**Observabale** : 1 function , tập hợp các giá trị trả về tương lai và trả về Obsever ( Khai báo dữ liệu ) – Giống khởi tạo promise

**Obsever** : Lắng nghe giá trị ( next, error, complete) được trả về bởi Observabale

**Subscription (.subscribe)**  : là trả về kết quả sau mỗi lần thực hiện Observabale

**Oparators** : function có nhiệm vụ tạo ra Observabale mới, change data, fitter data

**Subject** : thực hiện gửi dữ liệu đến Observabale

**Schedulers** : điều khiển khi nào một Subscription được thi và khi nào tín hiệu được gửi đi

Cú pháp tạo ra một **Observabale :**

**Cú pháp tạo Observabale**

**const tên\_biến = new Observabale(function tên\_hàm(obsever){**

**//Thực hiện logic// Lắng nghe dữ liệu**

**obsever.next(data)**

**obsever.error(data)**

**obsever.complete(data)**

**})**

**const tên\_biến\_2 = tên\_biến.subscribe(next, error, complete)**

**Hoặc viết thẳng :**

**const tên\_biến\_2 = tên\_biến.subscribe(val =>{console.log(val)}, )**

**Hoặc**

**const tên\_biến\_2 = tên\_biến.subscribe({**

**next: (x ) => {**

**console.log(x);**

**},  
error: (x ) => {**

**console.log(x);**

**},**

**complete: (x ) => {**

**console.log(x);**

**}**

**})**

**Ví dụ:**

**const observabale = new Observabale(function subscribe(obsever){**

**const id = setTimeout(()=>{**

**obsever.next(‘Hello Rxjs’);**

**obsever.complete()**

**},1000);**

**return function unsubscribe(){**

**clearTimeout(id)**

**}**

**const subscription = observabale.subscribe(console.log(value))**

**Tạo Observabale với cú pháp gọn of() và from():**

**Lưu ý : cú pháp Of() và From() tạo Observabale tự động có complete**

**Of():**

**Cú pháp :**

**of (‘’, string, number,[], {}, null, Boolean, promise ).subscribe(obsever)**

***//Nhận tất cả giá trị***

**Luồng chạy :**

**Nhận giá trị được truyền vào**

**chạy .subscribe(obsever) -> nhận vào giá trị được truyền next và complete**

**Điểm khác biệt : Chạy từng dữ liệu được truyền vào theo mảng, array, string,….**

**Trường hợp đặc biệt:**

**Promise trả về promise**

**From():**

**From ([], map, set, string )**

***//Chỉ nhận dữ liệu được lọc qua***

**Luồng chạy :**

**Nhận giá trị được truyền vào**

**chạy .subscribe(obsever) -> nhận vào giá trị được truyền next và complete**

**Điểm khác biệt: Trả về lọc qua từng index của array, từng kí tự string, object**

**Trường hợp đặc biệt:**

**Promise trả về kết quả**

**fromEvent()**

**Lưu ý : Tạo Observabale không có complete**

**Cú pháp tạo Observabale xử lý:**

**Khai báo obsever**

**const obsever = {**

**next: (tham so) =>{//logic}**

**}**

**fromEvent(element object HTML, tên\_hành\_động) .subscribe(obsever)**

**fromEventPattern() - Nâng cao fromEvent()**

**Cú pháp tạo Observabale xử lý :**

**Khai báo obsever**

**fromEventPattern(**

**(handler) =>{**

**element object HTML.addEventListener( ‘tên\_hành\_động’, handler),**

**},**

**(handler) =>{**

**element object HTML.removeEventListener(‘tên\_hành\_động’, handler)**

**}**

**).** **subscribe(observer)**

**interval**

**Khởi tạo số sau khoảng thời gian xác định**

**Luồng : Tạo ra Observabale**

**Cú pháp :**

**Interval(time)**

**Timer**

**Luồng chạy : Tạo ra Observabale và tự động complete không cần clear Timeout**

**Cú pháp :**

**Timer(1000).subscribe(obsever); Tương đương setTimeout**

**Sau thời gian delay mới thực hiện interval**

**Timer(1000, 1000).subscribe(obsever);**

**throwError**

**Luồng chạy: bắn ra lỗi thường dùng dể xử lý lỗi của Observabale**

**Cú pháp :**

**throwError(‘error’). subscribe(obsever);**

**BÀI 21 :**

**Pipe**

**Nhiệm vụ: \* Nhận đầu vào là một Observable và returns một Observable khác**

**\* Gộp function và không làm biến đổi**

**Cú pháp :**

**tên\_observable.pipe(operator1(), operator2());**

**Map ( function )**

**Cú pháp:**

**.pipe(map((data) => {**

**// logic xử lý**

**})**

**Pluck (funciton)**

**Cú pháp : pluck(‘id của array, key của object’)**

**Maptop**

**Kiến thức quan trọng Scan,**

**RxJS Filtering Operators**