30 天精通 RxJS (07)：Observable Operators & Marble Diagrams

2016年12月23日。4 分钟阅读

> Observable 的Operators 是实务应用上最重要的部份，我们需要了解各种Operators 的使用方式，才能轻松实作各种需求！

这是【30天精通RxJS】的07篇，如果还没看过06篇可以往这边走： 30天精通RxJS (06)：建立Observable(二)

昨天我们把所有建立Observable 实例的operators 讲完了，接下来我们要讲关于转换(Transformation)、过滤(Filter)、合并(Combination)等操作方法。先来让我们看看什么是Operator

什么是Operator？

Operators 就是一个个被附加到Observable 型别的函式，例如像是map, filter, contactAll... 等等，所有这些函式都会拿到原本的observable 并回传一个新的observable，就像有点像下面这个样子

```
var people = Rx.Observable.of('Jerry', 'Anna');

function map(source, callback) {
    return Rx.Observable.create((observer) => {
        return source.subscribe(
            (value) => {
                try{
                    observer.next(callback(value));
                } catch(e) {
                    observer.error(e);
                }
            },
            (err) => { observer.error(err); },
            () => { observer.complete() }
```





```
var helloPeople = map(people, (item) => item + ' Hello~');

helloPeople.subscribe(console.log);
// Jerry Hello~
// Anna Hello~
```

JSBin | 的jsfiddle

这里可以看到我们写了一个map的函式，它接收了两个参数，第一个是原本的observable，第二个是map的callback function。map内部第一件事就是用 `create` 建立一个新的observable并回传，并且在内部订阅原本的observable。

当然我们也可以直接把map 塞到 `Observable.prototype`

```
function map(callback) {
    return Rx.Observable.create((observer) => {
        return this.subscribe(
            (value) => {
                try{
                    observer.next(callback(value));
                } catch(e) {
                    observer.error(e);
                }
            },
            (err) => { observer.error(err); },
            () => { observer.complete() }
        )
    })
}
Rx.Observable.prototype.map = map;
var people = Rx.Observable.of('Jerry', 'Anna');
var helloPeople = people.map((item) => item + ' Hello~');

helloPeople.subscribe(console.log);
// Jerry Hello~
// Anna Hello~
```

這裡有兩個重點是我們一定要知道的，每個 operator 都會回傳一個新的 observable，而我們可以透過 `create` 的方法建立各種 operator。





> operator，記錄原本的資料源跟當前使用的 operator。

> 其實 lift 方法還是用 new Observable(跟 create 一樣)。至於為什麼要獨立出這個方法，除了更好的封裝以外，主要的原因是為了讓 RxJS 5 的使用者能更好的 debug。關於 RxJS 5 的除錯方式，我們會專門寫一篇來講解！

> 這裡我們只是簡單的實作 operator。如果之後實務上，想要不影響原本的 Observable 又能夠自訂 operator 可以參考官方的這份 文件。(現在先不用看)

其實 RxJS 提供的各種 operators 已經非常夠用了，不太需要我們自己創造 operator，這裡只是想讓大家先對 operator 的建立有個基本的觀念，之後在學習的過程中會比較輕鬆。

在我們開始介紹 RxJS 的 operators 前，為了能讓我們更好地理解各種 operators，我們需要先訂定一個簡單的方式來表達 observable！

Marble diagrams

我們在傳達事物時，文字其實是最糟的手段，雖然文字是我們平時溝通的基礎，但常常千言萬語也比不過一張清楚的圖。如果我們能訂定 observable 的圖示，就能讓我們更方便的溝通及理解 observable 的各種 operators！

我們把描繪 observable 的圖示稱為 Marble diagrams，在網路上 RxJS 有非常多的 Marble diagrams，規則大致上都是相同的，這裡為了方便撰寫以及跟讀者的留言互動，所以採用類似 ASCII 的繪畫方式。

我們用 - 來表達一小段時間，這些 - 串起就代表一個 observable。

```
----------------
```

x (大寫 X)則代表有錯誤發生

```
---------------x
```

| 則代表 observable 結束

```
----------------|
```

在這個時間序當中，我們可能會發送出值(value)，如果值是數字則直接用阿拉伯數字取代，其他的資料型別則用相近的英文符號代表。這裡我們用 [illegible] 舉例

source 的圖形就會長像這樣

```
-----0-----1-----2-----3--...
```

當 observable 是同步送值的時候，例如

```
var source = Rx.Observable.of(1,2,3,4);
```

source 的圖形就會長像這樣

```
(1234)|
```

小括號代表著同步發生。

另外的 Marble diagrams 也能夠表達 operator 的前後轉換，例如

```
var source = Rx.Observable.interval(1000);
var newest = source.map(x => x + 1);
```

這時 Marble diagrams 就會長像這樣

```
source: -----0-----1-----2-----3--...
            map(x => x + 1)
newest: -----1-----2-----3-----4--...
```

最上面是原本的 observable，中間是 operator，下面則是新的 observable。

以上就是 Marble diagrams 如何表示 operator 對 observable 的操作，這能讓我們更好的理解各個 operator。

> Marble Diagrams 相關資源： http://rxmarbles.com/

最後讓我們來看幾個簡單的 Operators！

Operators

```
var source = Rx.Observable.interval(1000);
var newest = source.map(x => x + 2);

newest.subscribe(console.log);
// 2
// 3
// 4
// 5..
```

用 Marble diagrams 表達就是

```
source: -----0-----1-----2-----3--...
            map(x => x + 1)
newest: -----1-----2-----3-----4--...
```

我們有另外一個方法跟 map 很像，叫 mapTo

mapTo

mapTo 可以把傳進來的值改成一個固定的值，如下

```
var source = Rx.Observable.interval(1000);
var newest = source.mapTo(2);

newest.subscribe(console.log);
// 2
// 2
// 2
// 2..
```

mapTo 用Marble diagrams 表达

```
source: -----0-----1-----2-----3--...
               mapTo(2)
newest: -----2-----2-----2-----2--...
```





下

```
var source = Rx.Observable.interval(1000);
var newest = source.filter(x => x % 2 === 0);

newest.subscribe(console.log);
// 0
// 2
// 4
// 6..
```

filter 用Marble diagrams 表达

```
source: -----0-----1-----2-----3-----4-...
            filter(x => x % 2 === 0)
newest: -----0-----------2-----------4-...
```

> 读者应该有发现map, filter 这些方法其实都跟阵列的相同，因为这些都是functional programming 的通用函式，就算换个语言也有机会看到相同的命名及相同的用法。

> 实际上Observable 跟Array 的operators(map, filter)，在行为上还是有极大的差异。当我们的资料量很大时，Observable 的效能会好上非常多。我们会有一天专门讲这个部份！

今日小结

今天我们讲了Observable Operators 的相关知识，有以下几个重点

- 什么是Operators
    - 如何建立 operator
- 大理石图
- 运营商
    - 地图
    - mapTo
    - 过滤

JavaScript RxJS 操作符 可观察的 大理石图 RxJS 30天

上一页 下一个