RxSwift - step by step

Todo VI - 更好的处理授权提示 (https://www.boxueio.com/series/rxswift-101/ebook/244)

为什么RxSwift也需要flatMap (https://www.boxueio.com/series/rxswift-101/ebook/267)

了解常用的transform operators

Back to series (/series/rxswift-101)

字号
字号
默认主题
金色主题
暗色主题

在接下来的两节里，我们讨论RxSwift中最重要的一类操作符，叫做*Transform operators*它们用来把一个Observable中的事件，变成另外一种形式。可以说，任何一个基于RxSwift开发的项目，都会使用这一类操作符。作为这个话题的第一部分，我们来看一些直观易懂的*Transform operators*。

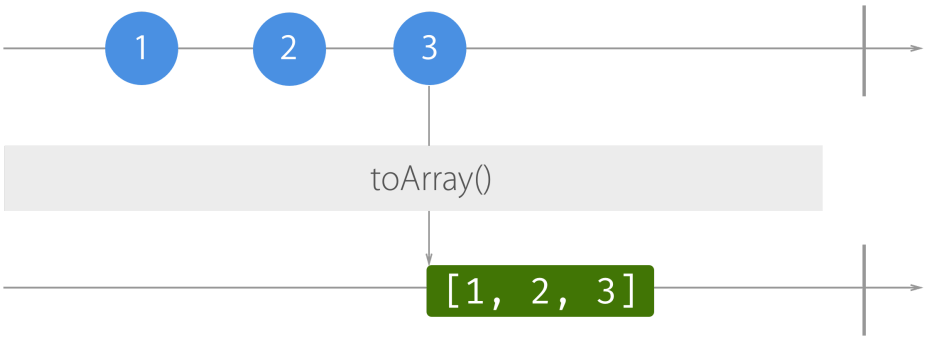
toArray

首先，是 `toArray`，这可以说是最简单的*Transform operator*。它把 `Observable<T>` 中所有的事件值，**在订阅的时候**，打包成一个 `Array<T>` 返回给订阅者。例如：

```
let bag = DisposeBag()

Observable.of(1, 2, 3)
    .toArray()
    .subscribe(onNext: {
        // Array<Int>
        print(type(of: $0))
        // [1, 2, 3]
        print($0)
    }).addDisposableTo(bag)
```

把上面这段代码用序列图表示，就是这样的：



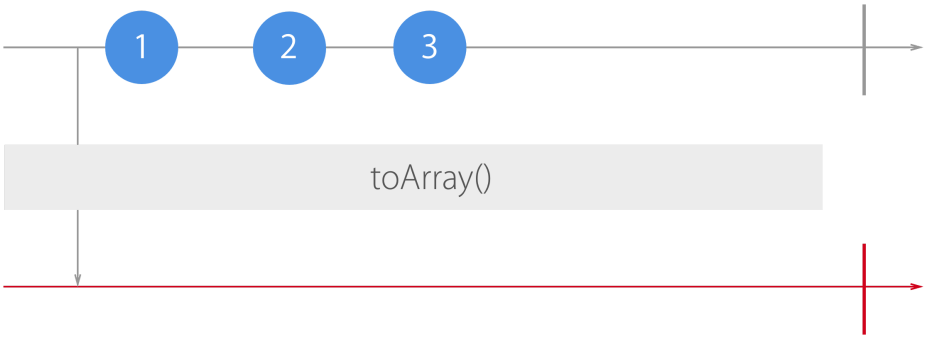
于是，在订阅的代码里，`$0` 的类型，就是 `Array<Int>`，`$0` 的值就是 `[1, 2, 3]`。这很好理解，但有一点要注意的是，`toArray` 的转换，是在订阅的时候，根据当前Observable中的值一次性完成转换的，后续的事件订阅则不会再进行转换。可能听着有点儿晕，我们来看个例子：

```
let numbers = PublishSubject<Int>()

numbers.asObservable()
    .toArray()
    .subscribe(onNext: {
        print($0)
    }).addDisposableTo(bag)


numbers.onNext(1)
numbers.onNext(2)
numbers.onNext(3)
```

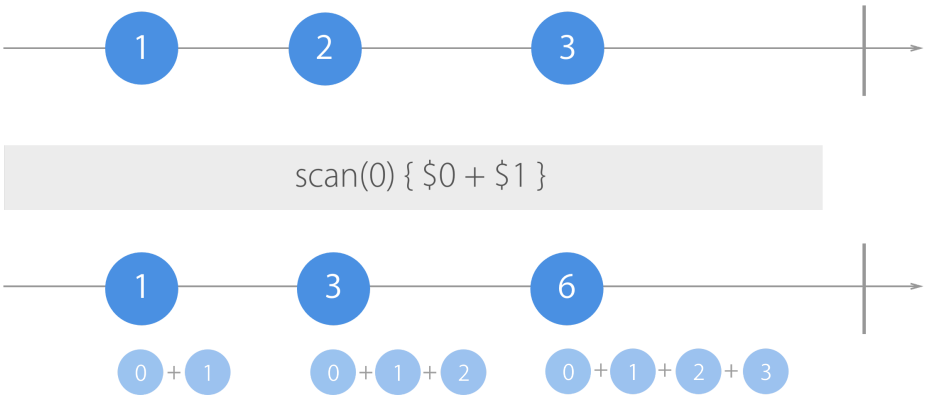
对于这个例子来说，在订阅的时候，使用了 `toArray`，但此时，`numbers` 中没有任何值，`toArray` 变换出来的，就是个空数组。即便之后 `numbers` 中发生了事件123，但是，我们订阅的，已经不是 `numbers`，而是 `numbers` 在订阅的时候转换成的 `Observable<Array<Int>>`，用序列图来表示是这样的：



我们订阅的代码是图中红色的Observable，因此，也就不会打印任何内容了。

scan

第二个*Transform operator*是 scan ，在之前改进用户授权的代码里，我们已经用过它了。给它设定一个初始值之后，它可以对Observable序列中的每一个事件进行“累加”，最终订阅到的，是“累加”之后的结果，如下图所示：



于是，下面的代码：

```
Observable.of(1, 2, 3).scan(0) {
        accumulatedValue, value in
         accumulatedValue + value
    }.subscribe(onNext: {
        print($0)
    }).addDisposableTo(bag)
```

打印出来的结果，就分别是1、3、6了。但是，就想上面图中展示的那样，**和 toArray 不同的是，scan 在Observable每次有事件的时候都会执行**。因此，如果我们把之前的 numbers 使用 scan 变换：

```
let numbers = PublishSubject<Int>()

numbers.asObservable()
    .scan(0) { $0 + $1 }
    .subscribe(onNext: {
        print("Scan: \($0)")
    }).addDisposableTo(bag)
```

就会看到“Scan: 1”和“Scan: 2”两个结果，表示，每次有事件发生时， scan 都会进行“累加”。

转换事件类型的map

除了把事件进行“累加”之外，我们也可以更自由的定义事件变换的行为。就像我们对集合中的元素进行变换一样，RxSwift也提供了一个 map operator：

```
Observable.of(1, 2, 3).map {
    value in value * 2
}.subscribe(onNext: {
    print($0)
}).addDisposableTo(bag)
```

这样，在订阅的时候，我们会得到“2 4 6”，也就是说，我们订阅到的，是事件被 map 之后的Observable。 map 接受一个closure，而这个closure的参数，就是原Observable中的事件值。

另外，和我们在之前提到过滤型operators时讲过的 `takeWhileWithIndex` 一样， `map` 也提供了一个 `withIndex` 的版本，像这样：

```
Observable.of(1, 2, 3)
    .mapWithIndex {
        value, index in
        index < 1 ? value * 2 : value
    }.subscribe(onNext: {
        print($0)
    }).addDisposableTo(bag)
```

`mapWithIndex` 的closure接受两个参数，第一个表示事件本身，第二个表示事件在序列中的位置。因此，在上面的例子里，当把第一个发生的事件值乘以2，之后的都返回事件值本身。这样，就可以得到“2 2 3”这样的结果了。

What's next?

以上，就是一些常用并且简单直观的的*Transform operators*，理解了它们之后，下一节，我们来看一个不那么容易理解的operator： `flatMap` 。

Todo VI - 更好的处理授权提示
(https://www.boxueio.com/series/rxswift-101/ebook/244)

为什么RxSwift也需要flatMap
(https://www.boxueio.com/series/rxswift-101/ebook/267)