Todo V - 理解重复订阅Observable的行为

RxSwift - step by step

← 返回视频列表 | 预计阅读时间: *16分钟* | ‹ PREVIOUS | NEXT ›

这一节，我们只解决根据用户选择的图片修改section header text的问题。它的内容比我们想象的要复杂一些。在新的项目起始模板里，我们唯一的改动，就是实现了之前的`setMemoSectionHederText`方法。大家也可以在这里下载项目的完成模板。

share() - 不要反复订阅同一个Observable

为了在用户选择完图片后把对应的Section Header文字修改成类似*4 MEMOS*的样子。我们可能会直接在`TodoDetailViewController.prepare(segue:sender:)`里添加下面的代码：

```
// In TodoDetailViewController
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    // ...
    _ = selectedPhotos.subscribe(onNext: { image in
            self.images.value.append(photos)
            self.setMemoSectionHederText()
        }, onDisposed: {
            print("Finished choose photo memos.")
        })
    // ...
}
```

但仔细想一下，其实我们并不用每次用户选择图片之后，都更新这个section header，只要在从图片选择界面返回的时候，根据当前选中的图片总数计算一次就好了。于是，我们很自然的把代码改成了下面这样：

```
// In TodoDetailViewController
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    // ...
    _ = selectedPhotos.subscribe(onNext: { image in
            self.images.value.append(photos)
        }, onDisposed: {
            print("Finished choose photo memos.")
        })

     _ = selectedPhotos.ignoreElements()
        .subscribe(onCompleted: { self.setMemoSectionHederText() })
    // ...
}
```

先来看这段代码的后半部分，我们用`ignoreElements`忽略了所有选择图片的事件。在从图片选择界面返回时，`selectedPhotos`会发送`.completed`事件，我们直接订阅这个事件，然后更新一次section header上的文字就好了。

这时，细心的你可能会发现，在上面的代码里，我们重复订阅了`selectedPhotos`。但是，这样的重复订阅会发生什么呢？为了回答这个问题，我们来看个例子。为了方便看到结果，我们直接在`ToDoListViewController.viewDidLoad`方法里，添加下面的代码：

```
let numbers = Observable<Int>.create { observer in
    observer.onNext(1)
    observer.onNext(2)
    observer.onNext(3)
    observer.onNext(4)
    observer.onNext(5)

    return Disposables.create()
}

let bag = DisposeBag()

numbers.subscribe(onNext: { print($0) }).disposed(by: bag)
numbers.subscribe(onNext: { print($0) }).disposed(by: bag)
```

假设，我们希望这两次订阅实际上使用的是同一个Observable，但执行一下就会在控制台看到，打印了两次*1 2 3 4 5*，也就是说**每次订阅，都会产生一个新的Observable对象**，多次订阅的默认行为，并不是共享同一个序列上的事件。

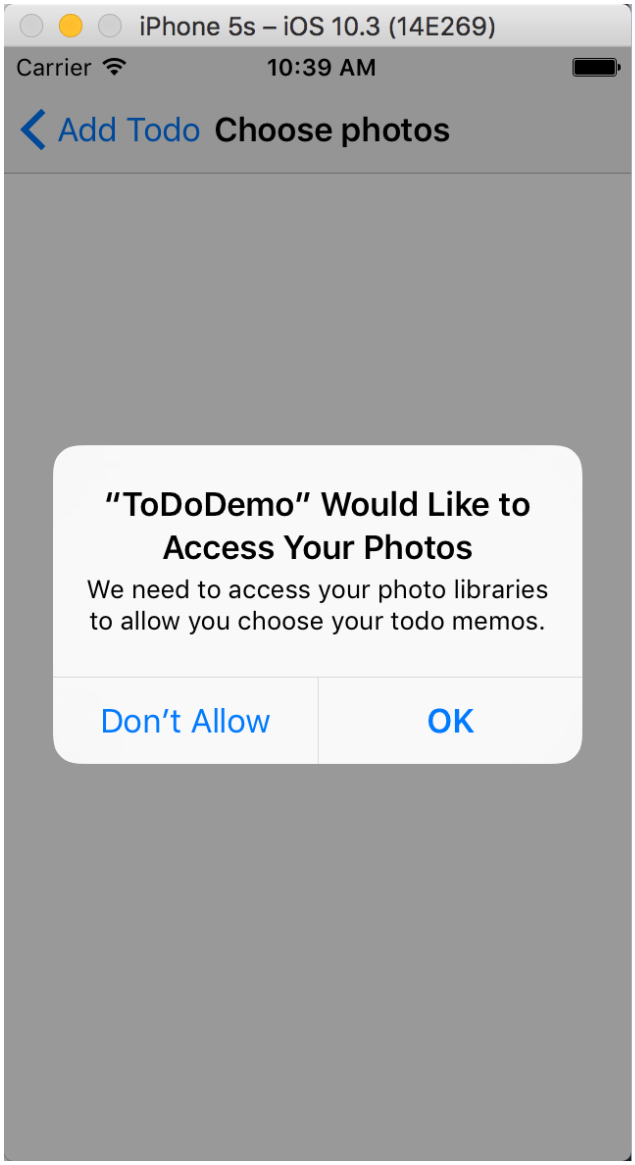
为了在多次订阅的时候共享事件，我们可以使用`share` operator，为了观察这个效果，我们把`numbers`的定义改成这样：

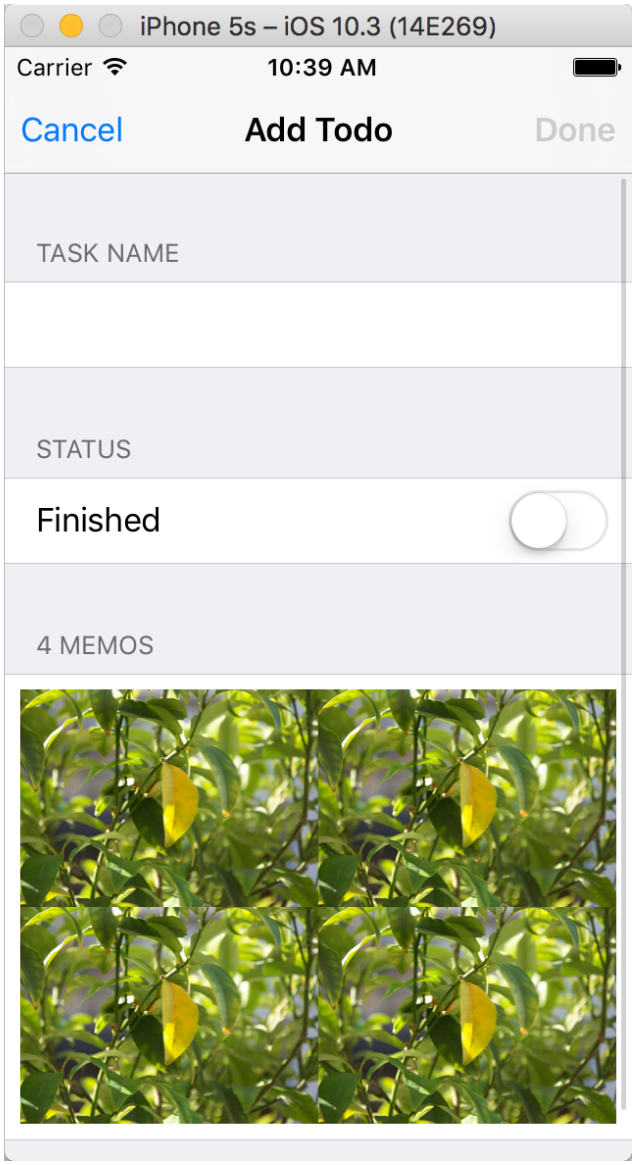
```
let numbers = Observable<Int>.create { observer in
    observer.onNext(1)
    observer.onNext(2)
    observer.onNext(3)
    observer.onNext(4)
    observer.onNext(5)

    return Disposables.create()
}.share()
```

重新再执行一下，就会发现，虽然订阅了两次，但我们只能看到打印了一次*1 2 3 4 5*。理解了共享订阅的概念之后，我们回到刚才的例子，重复订阅`selectedPhotos`也不是我们想要的行为，我们同样需要共享这个序列上的事件。因此，我们把`selectedPhotos`的定义改成这样就好了：

```
let selectedPhotos =
    photoCollectionViewController.selectedPhotos.share()
```

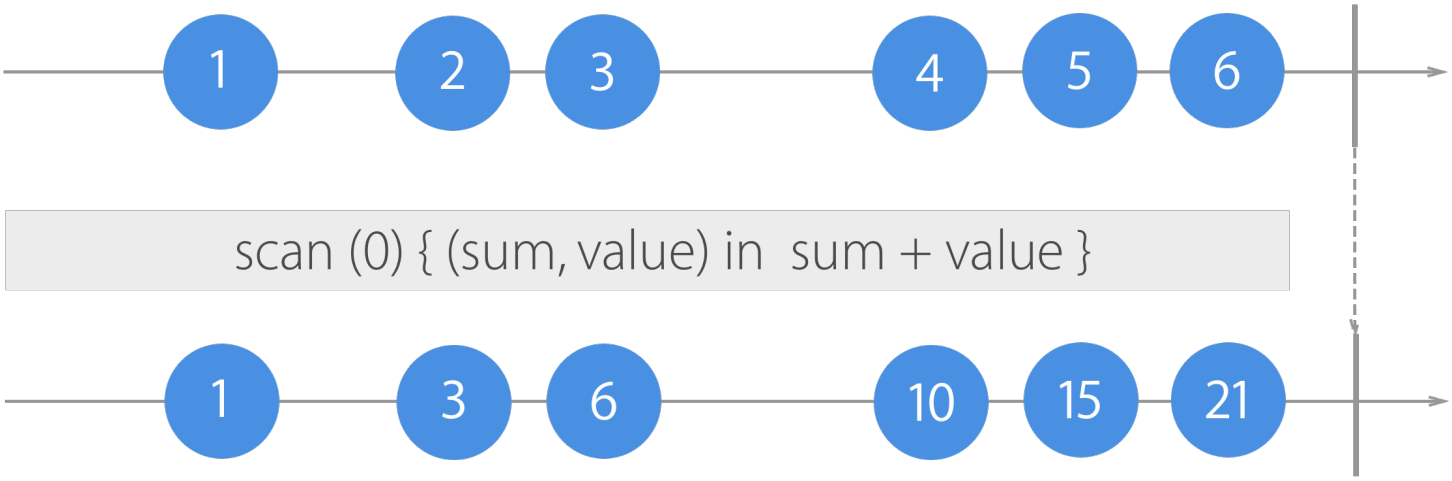
重新执行下TodoDemo，会发现，结果和之前是一样的。但目前的版本还有一个bug：尽管我们可以选中/反选照片库中的图片，但是这个行为并不会对应到图片的合成上，只要我们点了一次图片，图片就会被合成进来；





过滤重复选择的图片

为了解决这个问题，我们先介绍一个之前没提过的operator：`scan`，它有点儿类似集合中的`reduce`，可以把一个序列中所有的事件，通过一个自定义的closure，最终归并到一个事件，用序列图表示，就是这样的：



在上面这个图里，我们指定合并的初始值是0，合并动作是把历史和并结果和新的事件值相加。于是，在事件2的时候订阅，订阅到的结果就是3，3的时候订阅，订阅的结果就是6，以此类推。

有了这个operator之后，我们就可以解决图片重复添加的问题了。基本的思路是这样的：我们可以把来自`selectedPhotos`中的每一个`UIImage`事件合并成一个`[UIImage]`事件，然后在合并的过程中，如果遇到之前已经添加过的，表示这是用户反选的图片，我们就从`[UIImage]`中删除，否则，就添加到`[UIImage]`。这样，当我们再订阅这个`[UIImage]`的时候，就一定是用户选中的图片了。

理解了这个思路之后，我们来看代码，把前一个`selectedPhotos`的订阅改成这样：

```
_ = selectedPhotos.scan([]) {
    (photos: [UIImage], newPhoto: UIImage) in
        var newPhotos = photos

        if let index = newPhotos.index(where: { UIImage.isEqual(lhs: newPhoto, rhs: $0) }) {
            newPhotos.remove(at: index)
        }
        else {
            newPhotos.append(newPhoto)
        }

        return newPhotos
    }.subscribe(onNext: { (photos: [UIImage]) in
        self.images.value = photos
    }, onDisposed: {
        print("Finished choose photo memos.")
    })
```

可以看到，`scan`的初始值是一个空的数组，然后`selectedPhotos`中每发生一次图片选中事件，我们就检查图片是否已经添加过了，如果加过就删掉，否则就添加进来。处理完之后，我们把当前所有合并的`[UIImage]`返回。

这样，在接下来的订阅里，我们订阅到的就是一个已经处理好的`[UIImage]`。所以，直接把它赋值给`self.images.value`，之后所有的逻辑，就都是一样的了。

What's next?

至此，保存图片备忘的全部功能就实现了。但一开始我们申请用户授权访问照片库的时候，还有一个交互上的小问题，第一次打开照片库的界面是全白的。在下一节，我们就来解决这个问题。

‹ Prev: Todo IV - 进一步理解Subject的实际应用

Todo V - 理解重复订阅Observable的行为

Next: Todo VI - 更好的处理授权提示 ›