

## ☰ Reactive Programming in Swift

◀ 理解Reactive编程思想

理解Observables and Observer ▶

(<https://www.boxueio.com/series/reactive-programming-in-swift/ebook/73>)

(<https://www.boxueio.com/series/reactive-programming-in-swift/ebook/75>)

# Hello world in RxSwift

⌕ Back to series (</series/reactive-programming-in-swift/>)

理解了Reactive Programming的编程思想之后，在这段视频里，我们使用RxSwift (<https://github.com/ReactiveX/RxSwift>)来实现上个视频中“筛选用户输入偶数”的例子，以此来进一步了解Reactive Programming中的各种思想的具体实现。

## 安装RxSwift

首先，我们使用CocoaPods (<https://cocoapods.org/>)在之前的项目中安装RxSwift (<https://github.com/ReactiveX/RxSwift>)。

在终端里，进入项目目录，执行：

```
pod init
```

创建一个Podfile，然后，给它添加下面的内容：

```
# Uncomment this line to define a global platform for your project
platform :ios, '8.0'
# Uncomment this line if you're using Swift
use_frameworks!

target 'ThinkingInRxSwift' do
  pod 'RxSwift', '~> 2.0'
  pod 'RxCocoa', '~> 2.0'
end
```

执行 `pod install` 完成RxSwift (<https://github.com/ReactiveX/RxSwift>)安装。成功之后，CocoaPods (<https://cocoapods.org/>)会给我们提示：使用.xcworkspace扩展名的文件打开项目：

```
Analyzing dependencies
Downloading dependencies
Installing RxCocoa (2.4)
Installing RxSwift (2.4)
Generating Pods project
Integrating client project
```

## 依旧是过滤用户输入的偶数

在Finder里打开CocoaPods (<https://cocoapods.org/>)生成的.xcworkspace文件。

在Main.storyboard里，我们新添加一个 UITextField，并且，用同样的方式让Xcode生成约束：

⊕ 字号

● 字号

🖌️ 默认主题

🖌️ 金色主题

🖌️ 暗色主题



然后，按 `Option + Command + Enter`，打开Assistant View，按住Ctrl把 UITextField 拖拽到 Assistant View里，在弹出的窗口里给控件设置个属性，例如：rxUserInput：



接下来，我们修改下之前实现的 UITextFieldDelegate，让它只针对userInput对象生效：

```
func textField(textField: UITextField,
               shouldChangeCharactersInRange range: NSRange,
               replacementString string: String) -> Bool {

    if textField == self.userInput {
        // 1. Map user input string to Int
        if let n = Int(string) {
            // 2. Filter even numbers
            if n % 2 == 0 {
                print(n)
            }
        }
    }

    return true
}
```

这样，一切准备工作就就绪了。接下来我们就通过RxSwift (<https://github.com/ReactiveX/RxSwift>)来实现筛选输入偶数的功能。尽管你还没有开始，但是，相信我，它一定比你想象中简单得多。

## 如何定义一个“以时间为维度的数组”

RxSwift (<https://github.com/ReactiveX/RxSwift>)给 UITextField 添加了一个新的属性： `rx_text`，我们暂时忽略掉它的具体类型，把它理解为就是那个“以时间为维度的数组”就可以了。

而这个数组中事件的值，就是每一次用户输入之后，当前 UITextField 中的字符串（注意：这里指的是输入之后， UITextField 中的字符串，不是每一次用户输入的单个字符）。

## 如何定义发生的事件

对于每一个添加到“数组”中的事件，都存在三种可能：

- 成功：对于我们的例子来说，就是用户输入了一个字符，于是让 UITextField 有了新的值；
- 失败：在我们的这个例子里，还不会有失败的情况，但是在后面我们处理网络请求事，就会处理这种事件；
- 完成：表示“数组”中再也不会添加新的事件，例如，我们切换到了另外一个View；

为了封装这个事件，我们很自然会想到使用Swift中的 `enum` 来表示它，但是对于不同类型的成功事件，它包含的值有可能是不相同的，因此，我们使用一个泛型的 `enum` 来表达一个发生的事件：

```
enum Event<Element> {
    case Next(Element)      // next element of a sequence
    case Error(ErrorType)   // sequence failed with error
    case Completed          // sequence terminated successfully
}
```

为什么要使用Next表示事件成功时的值呢？强行记住它也没什么问题。

这里分享一个理解方式：

因为对于“事件数组”来说，每一次添加进来的值，相对于事件发生之前来说，都是它的“下一次”事件。因此，把“成功的下一次事件”，定义为Next。

## 如何订阅事件的发生

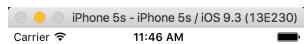
此时，我们已经有了“事件数组”、有了事件对象。接下来，我们该订阅事件了。在 viewDidLoad 方法里，添加下面的代码：

```
self.rxTextInput.rx_text.subscribeNext { print($0) }
```

上面这行代码很好理解，我们使用 subscribeNext 方法订阅了 rx\_text “数组”中的成功事件。

subscribeNext 接受一个Closure参数，这个Closure针对 UITextField 来说，接受一个 String 参数，表示当前事件发生后， UITextField 中字符串的值，返回 Void。在它的实现里，我们只是把这个值打印在了控制台上，方便我们观察。

完成之后，Command + R 编译执行，在第二个输入框里输入“1A2B”，我们就能在控制台看到：每一次输入，控制上都打印出了当前 UITextField 的值：



这说明，我们的订阅行为已经成功了，接下来，我们要筛选中其中的偶数。怎么办呢？我们有两种选择：

- 大部分人的直觉都会是直接在 subscribeNext 的 closure 参数中判断一下最后一个字符是否是偶数不就好了么？如果是这样，我们就又不知不觉回到了传统的基于状态的编程方式；
- 在 reactive programming 里，故事是这样的：每一次输入事件发生时， UITextField 的值，就被放在了时间轴上，我们要做的事情，不是处理每一个事件，然后然后根据事件值的不同做不同的处理。而是，我们可以对发生的事件进行“二次加工”，形成新的“事件数组”，并在新的事件数组中进行订阅。

这听起来有点儿抽象，我们来看代码。

## 如何对发生的事件进行“二次加工”

在 viewDidLoad 里，修改我们订阅事件的代码。首先，我们把值是字符串的“事件数组”变成一个值是 Int 的“事件数组”：

```
self.rxTextInput.rx_text
    .map { (input: String) -> Int in
        if let lastchar = input.characters.last {
            if let n = Int(String(lastchar)) {
                return n
            }
        }
        return -1
    }
}
```

这里，我们使用的 map 方法和函数式编程中我们对数组使用的 map 方法作用是类似的。它接受一个 closure 参数，这个 closure 自身的参数则是原事件数组中值的类型：String，并且，我们让它返回一个 Int。

这样，经过 `map` 映射之后：

1. 我们先判断 `UITextField` 中是否有值；
2. 把最后一次用户输入的字符转换成整数；
3. 转换成功后，我们就返回它；

上面任何一个条件失败，我们就返回-1（注：实际上任何一个不能被2整除的数都可以）。这样，我们就把新的“事件数组”中的值变成了`Int`。

接下来，我们要对得到的新“事件数组”再做一次转换，筛选出所有的偶数。在 `map` 后面，添加下面的代码：

```
self.rxTextInput.rx_text
    .map { (input: String) -> Int in
        if let lastchar = input.characters.last {
            if let n = Int(String(lastchar)) {
                return n
            }
        }

        return -1
    }
    .filter {
        $0 % 2 == 0
    }
```

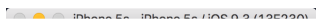
`filter` 用于逐个过滤“事件数组”中的元素，它也有一个 `closure` 参数。而这个 `closure` 的参数是“原数组”中元素的类型，在我们的例子里是`Int`，返回一个`Bool`值，表示是否满足过滤条件。

由于我们要筛选偶数，因此我们使用 `$0 % 2 == 0`。这样，我们就得到了只包含偶数值的“事件数组”。接下来，我们直接使用之前的 `subscribeNext` 订阅就可以了。

```
self.rxTextInput.rx_text
    .map { (input: String) -> Int in
        if let lastchar = input.characters.last {
            if let n = Int(String(lastchar)) {
                return n
            }
        }

        return -1
    }
    .filter {
        $0 % 2 == 0
    }
    .subscribeNext {
        print($0)
    }
```

完成之后，`Command + R` 编译执行，我们再输入1A2B，就可以在控制台看到，只有偶数被输出到控制台了：

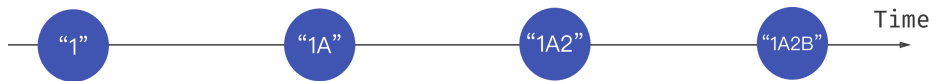


这样，我们就使用RxSwift完成了从用户输入中过滤偶数的例子。和我们之前通过`Delegate`实现最根本的一个差别就是，我们从事件处理函数中设置诸多状态判断，变成了我们根据自己的需要去生成对应的事件队列。接下来，我们再回顾一下整个过程。

---

## 小结

对于用户输入的“1A2B”来说，我们的“事件数组”是这样的：



然后，我们先用 `map` 把数组中的每一个元素的值从String变成Int，把无法转换成Int的值变成-1：

然后，再使用 `filter` 过滤出所有的偶数：

这样，我们的“事件数组”就变成了只有当用户输入偶数时，才添加元素，最后，我们使用`subscribeNext` 订阅其中的事件：

## Next?

通过这段视频，我们对于reactive programming中用到的概念和编程思想应该有一个更具体的认识了，在下一段视频中，我们将进一步探索被反复提及的“事件数组”。在RxSwift (<https://github.com/ReactiveX/RxSwift>)里，它们有自己的名字，叫做**Observables**，并且了解管理Observables的**DispatchBag**的用法。

### ◀ 理解Reactive编程思想

(<https://www.boxueio.com/series/reactive-programming-in-swift/ebook/73>)

### 理解Observables and Observer ▶

(<https://www.boxueio.com/series/reactive-programming-in-swift/ebook/75>)



职场漂泊的你，每天多学一点。

从开发、测试到运维，让技术不再成为你成长的绊脚石。我们用打磨产品的精神去传播知识，把最新的移动开发技术，通过简单的图表，清晰的视频，简明的文字和切实可行的例子——向你呈现。让学习不仅是一种需求，也是一种享受。

## 泊学动态

一个工作十年PM终创业的故事（二）(<https://www.boxueio.com/after-the-full-upgrade-to-swift3>)  
Mar 4, 2017

人生中第一次创业的“10有”(<https://www.boxueio.com/founder-chat>)  
Jan 9, 2016

猎云网采访报道泊学 (<http://www.lieyunwang.com/archives/144329>)  
Dec 31, 2015

What most schools do not teach (<https://www.boxueio.com/what-most-schools-do-not-teach>)  
Dec 21, 2015

一个工作十年PM终创业的故事（一）(<https://www.boxueio.com/founder-story>)  
May 8, 2015

## 泊学相关

关于泊学 >

加入泊学 >

泊学用户隐私以及服务条款 ([HTTPS://WWW.BOXUEIO.COM/TERMS-OF-SERVICE](https://www.boxueio.com/terms-of-service))

版权声明 ([HTTPS://WWW.BOXUEIO.COM/COPYRIGHT-STATEMENT](https://www.boxueio.com/copyright-statement))

## 联系泊学