**<https://www.jianshu.com/p/674e2f50684f>**

**RxJS的Observable概念详解**

[](https://www.jianshu.com/u/57bc723d246c)

[小溪的生活](https://www.jianshu.com/u/57bc723d246c)

2017.08.29 17:04\* 字数 1757 阅读 1531评论 3喜欢 5

文章主要介绍Observable，但是它是来自于RxJS这个强大的库。

简单了解一下RxJS以及RxJs和Observable的关系。

Reactive Extensions Library for JavaScript.

https://upload-images.jianshu.io/upload_images/5981091-da08b36841d7da48.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/700

官方定义

下面主要介绍Observable。

这里只是对一些概念进行解析。因为现在的工具太多，大家一开始总是想着怎么用起来（我经常也是这么做的，所以经常会有用了某个工具很长时间，却不知道它的概念是什么。），至于概念是什么？为什么这么定义？有什么好处？似乎想的很少。

盲目的使用会带来下面的一些问题：

第一，当有新问题的时候，不能马上建立相应的模型；

第二，用了之后，发现稍微深入一点的问题，不知道如何去用了；

第三，出了问题，漫天的百度找答案。

废话不多说，先来看看Observable与设计模式中的观察者模式是什么关系吧？

**1. Observable与观察者模式的关系**

其实这里讲的Observable就是一种观察者模式，只不过RxJS把Observable结合了迭代模式以及很多的operator，让他变得很强大，同时也增添了一些神秘的色彩。

那么设计模式中的观察者模式，其实是非常简单的，可以用生活中的订牛奶的示例来说明，

你订阅了某订奶机构的牛奶，那么付了钱之后，在预定的时间内都会给你送牛奶，如果你取消订阅，那么第二天就收不到新鲜的牛奶了。

其实，观察者模式的模型在生活中很多，突然想到订牛奶，是因为我前段时间订牛奶，要装奶箱，订奶机构的工作人员省事到直接用两根铁丝绑了门上了，结果第二天我就被小区物业说了，我给他发完短信，一怒之下想给客服打电话投诉他，结果装奶箱的可爱人给我发了一条短信：

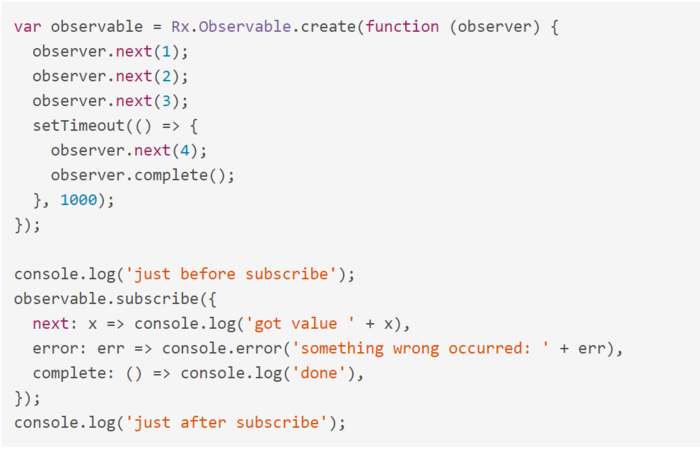
对不起啊，我明天给你重装，不要给客服打电话，他们会扣我的毛爷爷的。

于是乎，我就放下了电话，只能说大家赚钱都不容易。

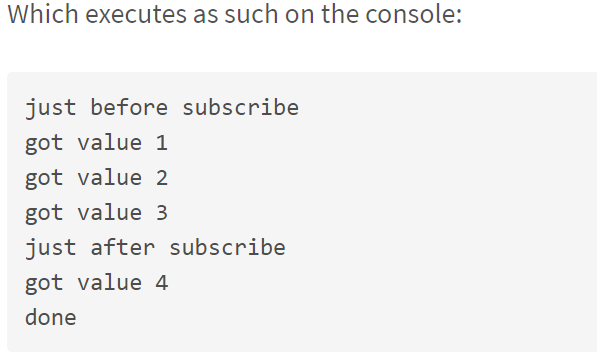
闲话莫谈，下面通过具体的实现来一起理解相应的概念。

**2. Observable的具体实现**

我们先看一下如何去使用Observable，下面一段代码是RxJS官网中的例子。



官网中的例子



运行结果

我们一起来分析一下上面的例子，并把理解的重点之处指出来。

**2.1 Observable是一个函数**

首先通过RxJS创建一个Observable的实例，Observable解释为被观察者。

var observable = Rx.Observable.create(function(observer){}

重点：Observable理解为一个函数，这很重要

这样在下面的每次订阅中就可以知道为什么是从头执行，因为一个函数调用都是从头开始执行的。

**这就和我们普通理解到的报纸订阅不太一样了。**

**报纸订阅：**如果你今天订了报纸，那么明天只会收到明天以及明天以后的报纸，之前的报纸是不会收到的。

**Observable：**相当于你今天订了报纸，那么明天会收到报纸机构从第一天发报到明天的所有的报纸。

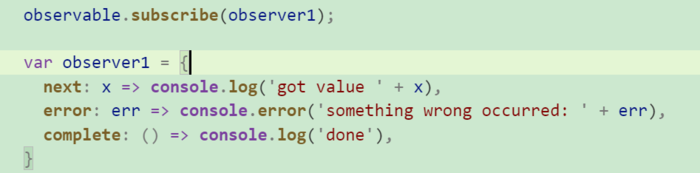
**2.2 订阅的过程**

*怎么订阅的呢？*

是通过Observable的subscribe函数进行订阅的，subscribe函数的函数体就是刚才的Observable的函数体，那么subscribe的参数就是订阅者，也叫观察者observer。

代码中是一个观察者observer1去订阅了此Observable对象observable。

那么只要观察者订阅服务就马上执行Observable的函数体。



observer1订阅

**2.3 观察者实现过程**

首先观察者是一个对象，包含三个回调函数。

**next**：表示一个接一个的任务。

**error**：表示Observable对象执行流程出错了，订阅会立即结束。

**complete**：表示订阅过程完成了。

所以Observable对象的函数体内执行的任务其实就是在调用观察者的next以及complete回调函数。

当Observable对象的函数体执行出错时，会调用观察者的error回调函数。

**2.3 取消订阅**

那么如何去取消订阅呢？

在解决此问题之前，先列出一个对象subscription，它是干嘛的呢？

subscription是用来描述观察者和被观察者的订阅关系。

var subscription1 = observable.subscribe(observer1);

取消订阅就很简单，直接采用：

 subscription1.unsubscribe();

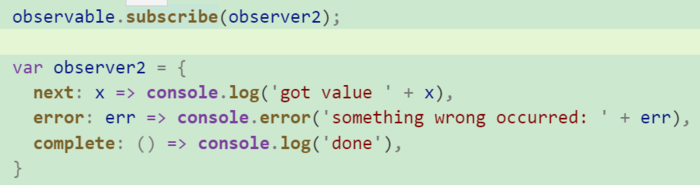
**2.4 多人订阅**

可以进行多个观察者订阅。

像下面的例子，再创建一个observer2的对象，执行下面的操作，observer2就订阅了此服务，那么observer2的执行过程和observer1是没有任何关系的。

**注意：并不是共享订阅的Observable对象。**

observable.subscribe(observer2);



observer2订阅

**3. 总结概念**

现在总结一下，涉及到的名字，我们以家里订牛奶为例来进一步说明。

**3.1 Observable**

**被观察者，相当于送奶机构。**

实现上是一个函数对象，谁订阅了，就执行此函数的函数体。

**3.2 Observer**

**观察者，相当于某一个订牛奶的个体。**

实现上是一个对象，里面注册了三个回调函数，分别是next、error、complete。

Observable的函数体会执行Observer的回调函数。

**3.3 subscribe**

Observable和Observer怎么建立关系？送奶机构怎么知道给谁送牛奶？

通过subscribe函数，这样observer就和observable建立了观察和被观察的关系了。

**3.4 subscription**

Observable和Observer之间的关系，谁去维护呢？订阅牛奶的订单谁去管呢？

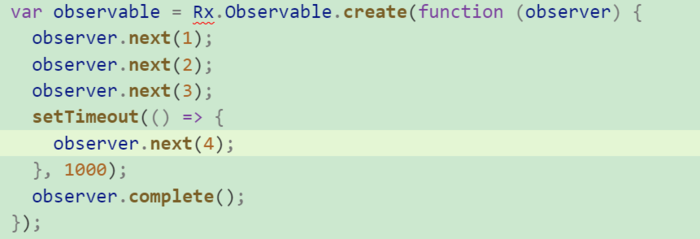
那么就是subscription来管理Observable和Observer之间的关系。

**3.5 unsubscribe**

unsubscribe解除Observable和Observer之间关系，牛奶不再送了。

**3.6 中断的概念**

Observable的函数体如果出错或complete，那么函数体就会停止执行，再有next也不会执行，如果函数体是下面代码这样执行的，因为setTimeout里面的next(那么observer.next(4)是不会执行的。



**最后用一句话总结一下：**

**Observable的好处在于模型简单、过程有多个输出、中间可以中断和RxJS提供的强大的operators。**

如有表达上的不符，还请不吝赐教。

参考文献：http://reactivex.io/rxjs/manual/overview.html

<完>