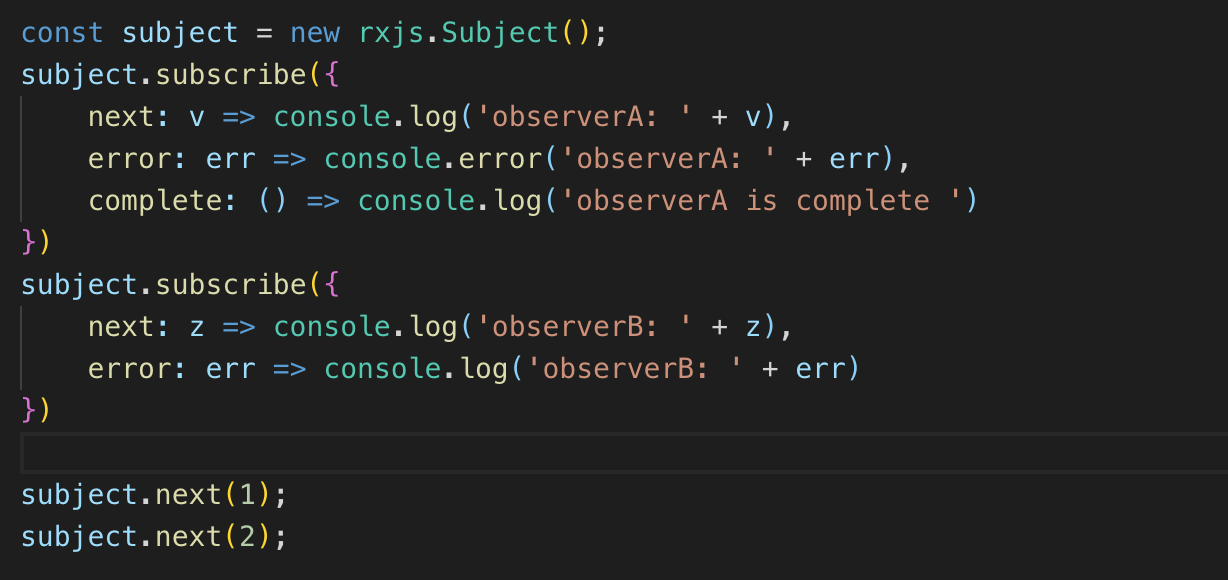
RxJs--基础概念篇（下）

承上启下，今天是RxJs基础概念的后半部分。主要聊一聊RxJs的核心概念中剩下的三个概念，Subjects（主体）、Operators（操作符）以及Schedulers（调度器）。如果回顾之前聊到的概念，请移步《RxJs--基础概念篇》以及《初识RxJs--基础铺垫篇》。

在《RxJs--基础概念篇》中，我们讲到了Observer（观察者）通过Subscription（订阅）的方式，可以获得Observable（可观察对象）推送的数据。可以说，一个Observable（可观察对象）对应着一个Observer（观察者），它们之间的联系是一个单向的。假设，我现在有一个Observer（可观察对象），但是我有很多个Observer（观察者）要用到它的数据，难道我要每一个Observer（观察者）对应着创建一个Observable（可观察对象）吗？那这未免有点麻烦了。RxJs的设计者也考虑到了这种多播的情况，也就是Subjects（主体）。

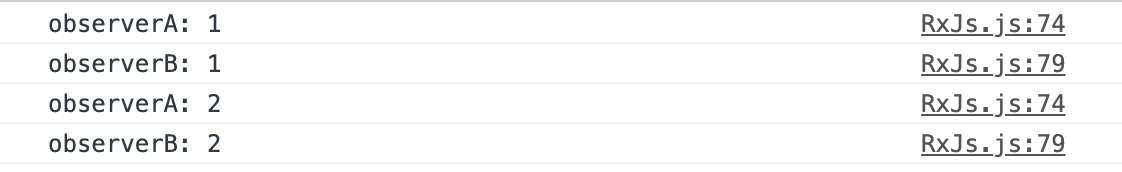
Subjects（主体）是一种特殊类型的Observable，但是它和Observable（可观察对象）不同的是，它可以多播给多个Observer（观察者）。它是将值或者事件发送给多个Observer（观察者）的唯一方式。我们来看个例子。



首先，和创建Observable不同的是，Subject是一个RxJs下的对象，我们需要通过new 关键字对其进行一个实例化操作。在实例化操作完成后，我们就获得了一个subject对象。

通过Subject对象下的subscribe方法，我们可以将观察者注册到Subject观察者列表中。在Subject对象中，subscribe方法并不是对Subject进行的一种订阅，而是将给定的观察者注册到了注册列表中。

既然将观察者注册到了自身的观察者列表中，那么也可以说每个Subject对象都是一个观察者。每一个Subject对象下都有next()、error()、complete()方法。所以，如果要给Subject对象提供新值的话，只需要调用next()方法即可。上面的例子最终输出的结果如下：



通过结果我们可以看出，先是两个观察者输出了1，之后两个观察者输出了2.这也就是Subject多播的功能。可以拥有多个观察者。

之前我们也一直在说关于操作符的内容，在这里，就简单的聊一下什么事操作符。操作符其实就是对于Observable对象的数据集合进行的一些数据操作方法，例如map操作符，其实就类似与我们Array（数组）中的map循环方法。我们可以将其理解为Observable对象的一些数据处理方法。我们之前说到，RxJs的一个核心思想就是函数式编程，将一个复杂的数据逻辑进行步骤拆分，每一步处理完的结果返回给下一步的函数，最终得出我们想要的结果。Operators（操作符）就是这样的一种概念。

在操作符被调用的时候，它们不会改变已经存在的Observable对象，而是会创建一个新的Observable对象并返回。这样保证每一步的结果都是可以追踪的。它的本质是一个纯函数，接收一个Observable作为输入，输出一个新的Observable作为输出。但是要明确一点，它的subscription逻辑是基于第一个Observable对象的。

在这里明确两个有关操作符的概念。那就是实力操作符与静态操作符。

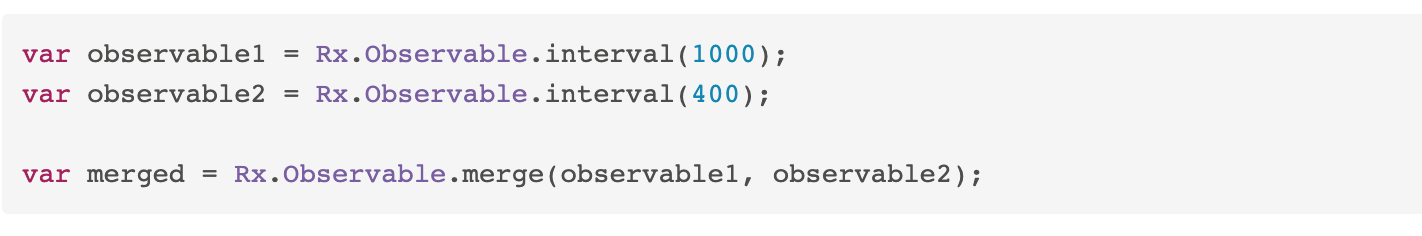
实例操作符，是Observable实例上的方法，函数的内部不依赖于外部传递的参数，函数内部的this指向是当前调用它的Observable类。静态操作符，是被附加到Observable类上的方法，函数内部依赖于外部传递进来的参数，在内部不使用this关键字，并且会接收一个非Observable对象，返回一个新的Observable对象。这里我们用官方提供的两个例子来做一下对比。





实例操作符

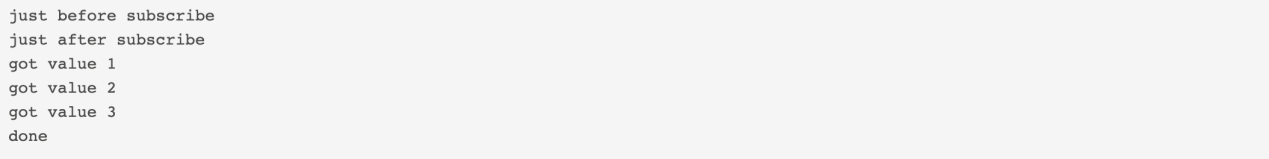




静态操作符

最后我们说一下Schedulers（调度器）。官方对此的解释也并非很多，总结下来，Schedulers（调度器）其实是控制何时启动subscription和何时发送通知，有点类似于我们的延迟加载或者说延迟通知吧。在前面的例子中我们可以看出来，当我们订阅了Observable对象后，订阅会立即出发，Observable对象会立即向我们广播数据。而Schedulers（调度器）则可以阻止这种同步发送。我们来看个官方的例子。

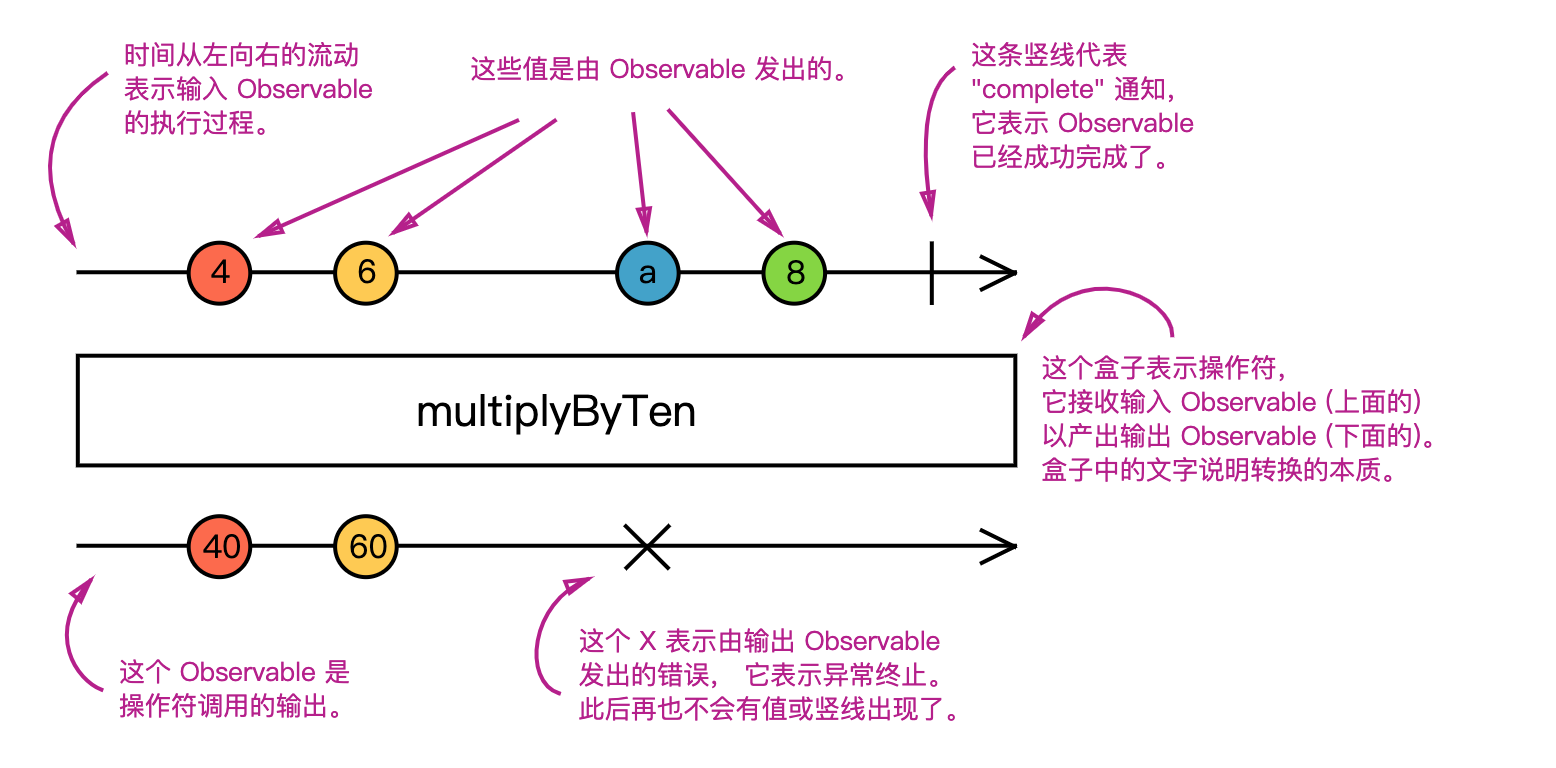




到此，RxJs的涉及到的核心概念也已经完成了。通过对这些基础概念的了解下，我对RxJs也算是有了新的认识。之前觉得比较鸡肋，但是现在觉得还是很有优势的。主要有以下几点：

* 相比较函数来说，数据获取方式由被动变为主动，也就是由pull拉取模式变成push推送模式，对于我们在获取数据的时候，就不需要增加定时器来定时获取了。
* 相比较推送模式下的promise对象来说，它可以推送多个值，而减少当我们需要传递多值时，需要创建多个promise对象的麻烦。
* 函数式编程，让整个的代码清晰明了，一目了然。

最后附上一张官方对于RxJs的执行弹珠图。



如果当你了解了RxJs后，觉得我所说的有些出入的话，可以留言哦~赞一个吧。谢谢~