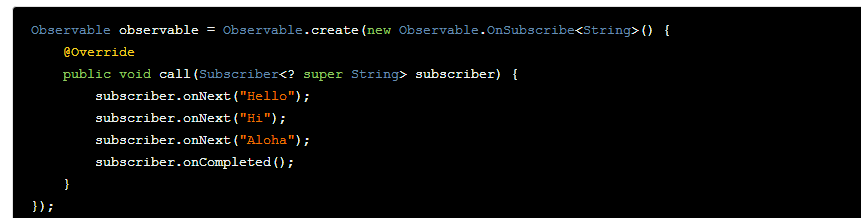
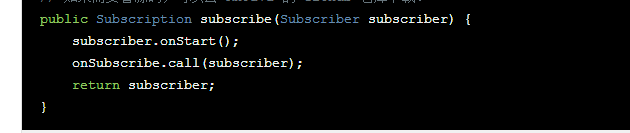
Observable 在创建的时候传入.OnSubscribe：

--》OnSubscribe里面的call方法在订阅的时候执行是因为看订阅源码：



也就是说先执行subscriber的onStart方法，然后才开始事件分发也就是执行onSubscribe的call方法。

精华：



1. onStart(): 这是 Subscriber 增加的方法。它会在 subscribe 刚开始，而事件还未发送之前被调用，可以用于做一些准备工作，例如数据的清零或重置。这是一个可选方法，默认情况下它的实现为空。

\*\*\*\*需要注意的是，如果对准备工作的线程有要求（例如弹出一个显示进度的对话框，这必须在主线程执行）， onStart() 就不适用了，因为它总是在 subscribe 所发生的线程被调用，而不能指定线程。要在指定的线程来做准备工作，可以使用 doOnSubscribe() 方法，具体可以在后面的文中看到\*\*\*\*

有了这几个 Scheduler ，就可以使用 subscribeOn() 和 observeOn() 两个方法来对线程进行控制了。 \* subscribeOn(): 指定subscribe() 所发生的线程，即 Observable.OnSubscribe 被激活时所处的线程。或者叫做事件产生的线程。 \* observeOn(): 指定Subscriber 所运行在的线程。或者叫做事件消费的线程。

当 Observable 被订阅的时候，OnSubscribe 的 call() 方法会自动被调用，事件序列就会依照设定依次触发，这样，由被观察者调用了观察者的回调方法，就实现了由被观察者向观察者的事件传递，即观察者模式。

just(T...): 将传入的参数依次发送出来。

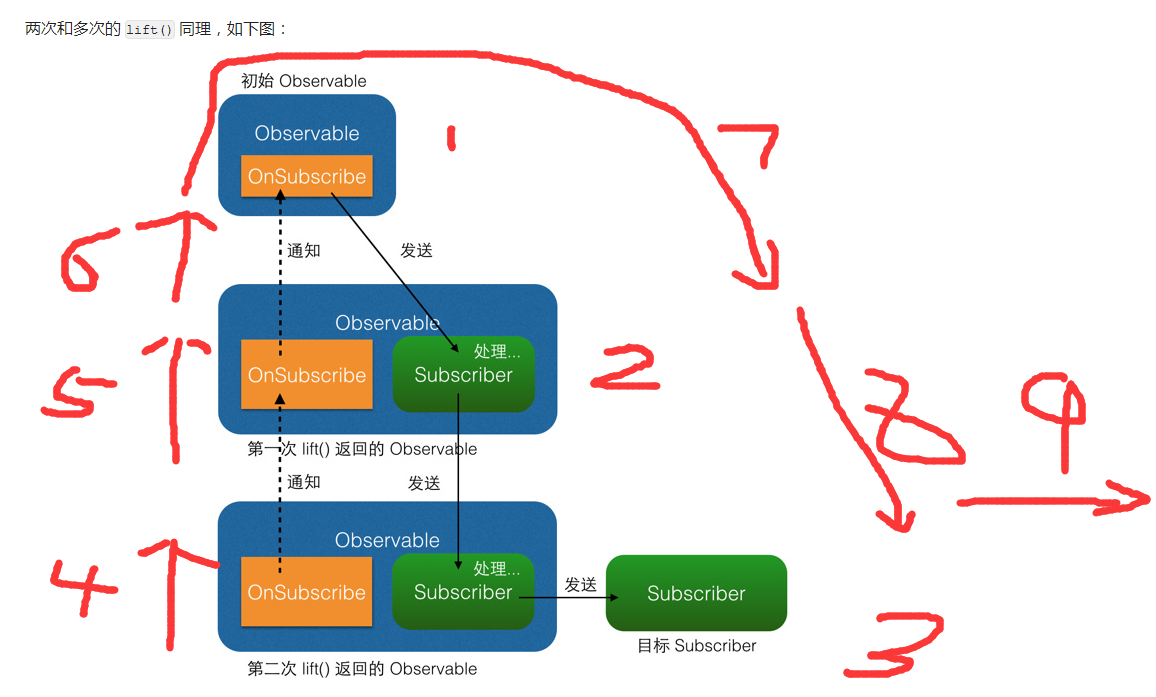
from(T[]) / from(Iterable<? extends T>) : 将传入的数组或 Iterable 拆分成具体对象后，依次发送出来。

map(): 事件对象的直接变换 ：例如把一个传入String对象变换成一个Bitmap对象，然后再传给观察者

flatMap() 中返回的是个 Observable 对象，并且这个 Observable 对象并不是被直接发送到了 Subscriber 的回调方法中。

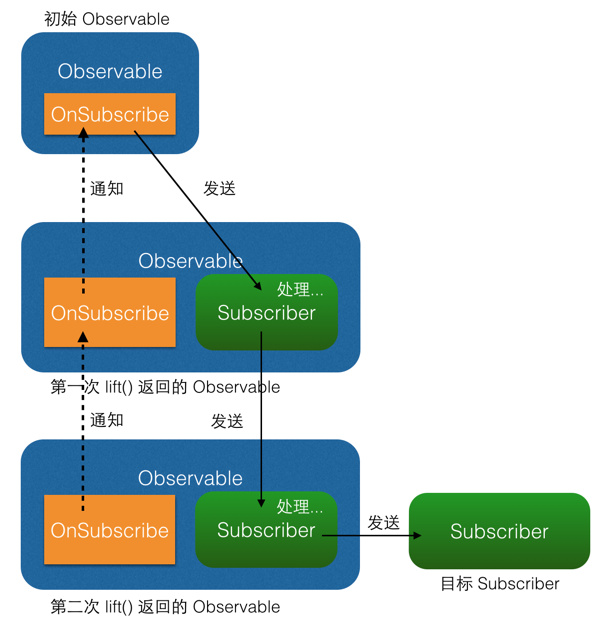
问题：

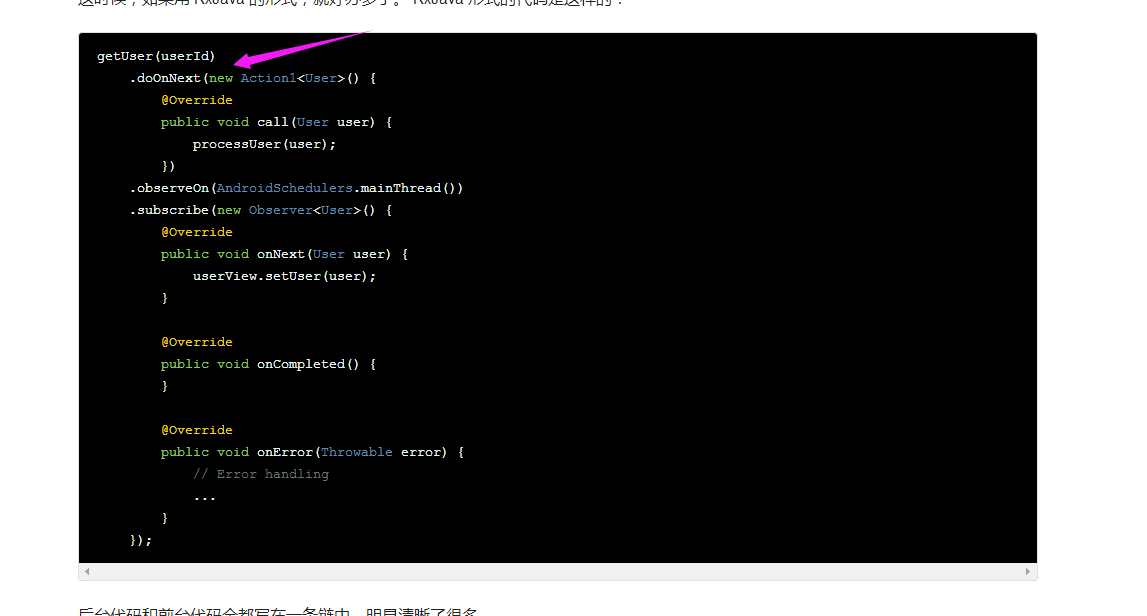
两次和多次的 lift() 同理



这是自己的理解不知道对不对？？？

原图：





这个方法名字可以随便写吗？？？