

如何合并Observables中的事件

RxSwift - step by step

← [返回视频列表](#)

预计阅读时间: 15分钟

< [PREVIOUS](#)

[NEXT](#) >

了解了如何合并Observables之后，这一节，我们来看如何合并Observables中的事件。

把多个Observables中的事件合并为一个

首先要介绍的operator，是`combineLatest`，它把多个Observables中的**当前事件**合并成一个事件。为什么我们使用了**当前事件**而不是**最新事件**呢？稍后，我们就会明白了。

现在，先来看它的用法：

```
let sequence =
    Observable.combineLatest(queueA, queueB) {
        eventA, eventB in
        eventA + "," + eventB
    }.subscribe(onNext: {
        dump($0)
    })
```

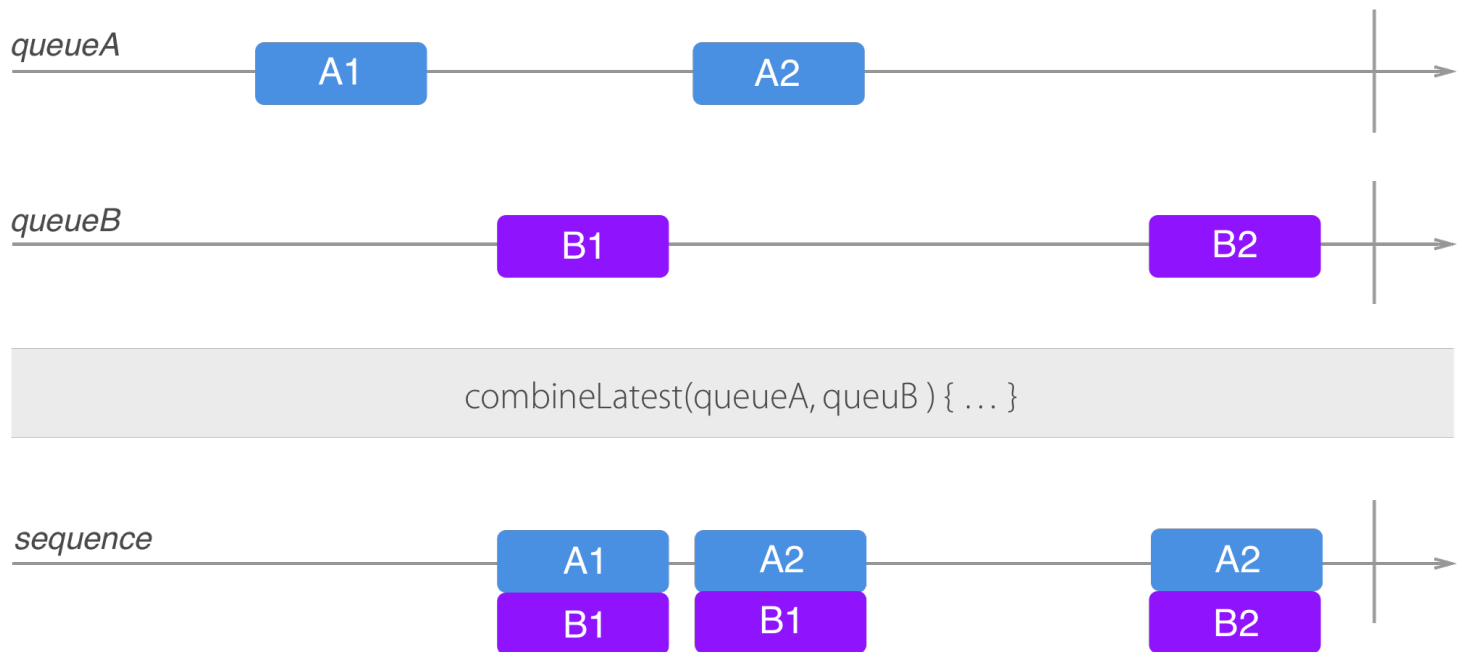
从我们的例子里可以看到，`combineLatest`的前两个参数表示要合并事件的Sub-observables，最后一个closure表示合并的过程。

实际上，`combineLatest`不止接受2个Observables，最多可以可以给它传递7个。

然后，我们在`queueA`和`queueB`中生成一些事件：

```
queueA.onNext("A1")
queueB.onNext("B1")
queueA.onNext("A2")
queueB.onNext("B2")
```

为了理解合并后的订阅过程，我们通过序列图来看一下：



在这个图中，有以下几点是需要关注的：

1. 当`queueA`中发生A1时，由于`queueB`中还没有任何事件，此时，不会发生任何combine的动作。只有在每一个Sub-observable中都发生过一个事件之后，`combineLatest`才会执行我们定义的closure；
2. 于是，当`queueB`中发生B1时，我们就订阅到了第一个合并后的事件，值是“A1,B1”；
3. 接下来，当`queueA`中发生A2时，对于`queueA`来说，当前的事件就是A2，对于`queueB`来说，当前的事件仍就是B1，因此，我们会订阅到“A2,B1”；
4. 最后，当B2发生时，我们会订阅到“A2,B2”；

因此，对于这个例子来说，我们一共会订阅到三次合并后的事件。在控制台里执行一下，就能看到这个结果了。

除了把要合并的Sub-observable一个个传递给`combineLatest`之外，和`concat`类似，我们也可以把多个Observable放在一个数组里：

```
let sequence =
Observable.combineLatest([queueA, queueB]) {
  events in
  events.joined(separator: ",")
}.subscribe(onNext: {
  dump($0)
})
```

这次，我们在合并的closure里，会收到一个`Array<T>`参数，其中`T`就是Sub-observables中的事件类型。因此，这种数组参数的用法，要求`combineLatest`的所有Sub-observables的事件类型都相同。

合并事件类型不同的Sub-observables

言外之意，非数组参数的`combineLatest`是可以组合事件类型不同的Observables的。而它也是RxSwift中为数不多的可以组合不同事件类型的operators之一。例如，我们把`queueB`的事件类型，改成`Int`。

```
let queueB = PublishSubject<Int>()
```

然后，修改对应的订阅代码，把`queueB`中的事件值手动转换成`String`：

```
let sequence =
    Observable.combineLatest(queueA, queueB) {
        eventA, eventB in
        eventA + "," + String(eventB)
    }.subscribe(onNext: {
        dump($0)
    })
```

接下来，把用于测试的事件序列改成这样：

```
queueA.onNext("A1")
queueB.onNext(1)
queueA.onNext("A2")
queueB.onNext(2)
```

重新执行下，就会看到下面这样的结果了：

```
- "A1,1"
- "A2,1"
- "A2,2"
```

当然，道理和之前是一样的。

combineLatest的生命周期

最后，来看`combineLatest`合并之后的Observable的生命周期。简单来说，只有所有的Sub-observable都完成之后，合并后的Observable才会发生`Completed`事件。如果其中某个Sub-observable提前结束了，`combineLatest`会一直把最后一次发生的事情，作为这个Sub-observable的“当前事件”。例如，这次，我们让`queueA`提前完成：

```
queueA.onNext("A1")
queueB.onNext(1)
queueA.onNext("A2")
queueA.onCompleted()
queueB.onNext(2)
queueB.onNext(3)
```

这样，当`queueB`发生事件2和3时，`combineLatest`就会用`queueA`中的最后一次Next事件，也就是A2，和`queueB`中的事件进行合并，于是，我们就能订阅到下面这样的结果了：

```
- "A1,1"
- "A2,1"
- "A2,2"
- "A2,3"
```

但是，如果在合并的过程中有Sub-observable发生`Error`事件，`combineLatest`合成的Observable就会立即结束，例如这样：

```
queueA.onNext("A1")
queueB.onNext(1)
queueA.onNext("A2")
queueA.onError(E.demo)
queueB.onNext(2)
queueB.onNext(3)
```

我们就只能订阅到“A1,1”和“A2,1”了。

看到这里，尤其是`combineLatest`对提前完成的Sub-observable的处理，你应该就能明白为什么我们在开始说它合并的是当前事件，而不是真正的最新事件了。从某种意义上说，它叫做`combineCurrent`似乎更合理一些 :-)

真正只合并最新事件的operator

那么，如果我们真正要合并Sub-observable中的最新事件该怎么办呢？为此，RxSwift提供了另外一个operator，叫做`zip`，它的用法和`combineLatest`几乎是相同的，我们可以把之前的合并代码改成这样试一下：

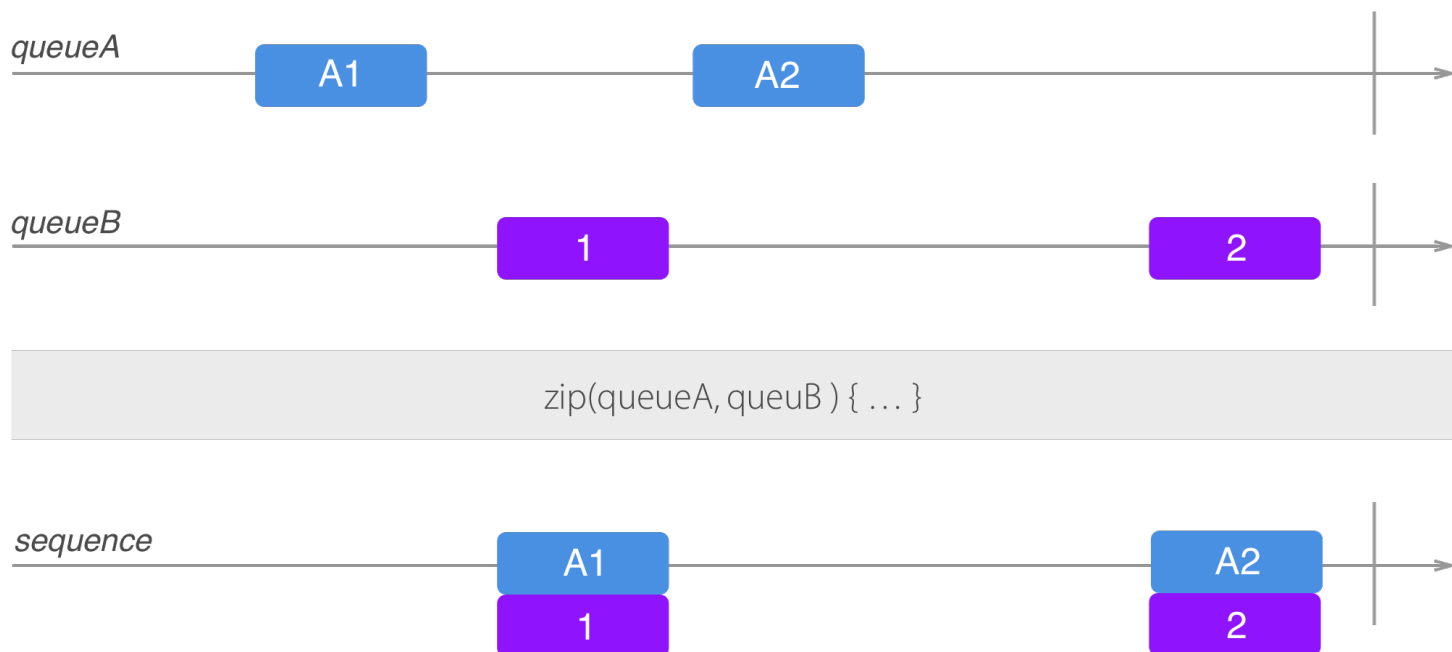
```
let sequence = Observable.zip(queueA, queueB) {
    eventA, eventB in
    eventA + "," + String(eventB)
}.subscribe(onNext: {
    dump($0)
})
```

可以看到，除了改用`zip` operators之外，没有任何变化。

然后，把测试的事件序列改成这样：

```
queueA.onNext("A1")
queueB.onNext(1)
queueA.onNext("A2")
queueB.onNext(2)
queueB.onNext(3)
```

用序列图表示，就是这样的：



每次在完成合并之后，所有Sub-observables中的事件就可以理解为被消费掉了。只有当下一次所有序列中都产生新事件的时候，才会进行下一次合并。于是，我们就只能订阅到“A1,1”和“A2,2”了。

另外一点和`combineLatest`不同的是，`zip`合成的Observable中，其中任何一个Sub-observable发生了`Completed`事件，整个合成的Observable就完成了。

What's next?

了解了如何合并多个Observables的事件之后，下一节，我们来讨论Observables之间更复杂的关系，如何根据事件，在多个Observables之间进行跳转。

[< Prev: 如何合并Observables](#)

[☰ 如何合并Observables中的事件](#)

[Next: 如何在不同的Observables之间跳转 >](#)

关于我们

想循序渐进的跟上最新的技术趋势？想不为了学点东西到处搜索？想找个伙伴一起啃原版技术经典书？技术之外，还想了解高效的工作流技巧？甚至，工作之余，想找点儿东西放松心情？没问题，我们用4K开发视频，配以详尽的技术文档，以及精心准备的广播节目，让你渴望成长的技术需求，也是一种享受。

Email Address

10@boxue.io

客户服务

📞 2085489246

相关链接

- › 版权声明
- › 用户隐私及服务条款
- › 京ICP备15057653号-1
- › 京公网安备 11010802020752号

关注我们

在任何你常用的社交平台上关注我们，并告诉我们你的任何想法和建议！



邮件列表

订阅泊学邮件列表以了解泊学视频更新以及最新活动，我们不会向任何第三方公开你的邮箱！

Email address

立即订阅