30 天精通 RxJS (20)：Observable Operators - window, windowToggle, groupBy

2017年1月1日。6 分钟阅读

前几天我们讲完了能把Higher Order Observable 转成一般的Observable 的operators，今天我们要讲能够把一般的Observable 转成Higher Order Observable 的operators。其实前端不太有机会用到这类型的Operators，都是在比较特殊的需求下才会看到，但还是会有遇到的时候。

运营商

窗口

window 是一整个家族，总共有五个相关的operators

- 窗口
- windowCount
- windowTime
- windowToggle
- windowWhen

这里我们只介绍window 跟windowToggle 这两个方法，其他三个的用法相对都简单很多，大家如果有需要可以再自行到官网查看。

基本的HTML 跟CSS 笔者已经帮大家完成，大家可以直接到下面的连结接着实作：

```
Observable<T> => Observable<Array<T>>
```

这里我们会用到#search 以及#suggest-list 这两个DOM

```
var click = Rx.Observable.fromEvent(document, 'click');
var source = Rx.Observable.interval(1000);
var example = source.window(click);

example
  .switch()
  .subscribe(console.log);
// 0
// 1
// 2
// 3
// 4
// 5 ...
```

首先window 要传入一个observable，每当这个observable 送出元素时，就会把正在处理的observable 所送出的元素放到新的observable 中并送出，这里看Marble Diagram 会比较好解释

```
click  : -----------c----------c------------c--
source : ----0----1----2----3----4----5----6---..
                  window(click)
example: o----------o----------o------------o--
         \          \          \
          ---0----1-|--2----3--|-4----5----6|
                  switch()
       : ----0----1----2----3----4----5----6---...
```

这里可以看到example 变成发送observable 会在每次click 事件发送出来后结束，并继续下一个observable，这里我们用switch 才把它摊平。

当然这个范例只是想单存的表达window 的作用，没什么太大的意义，实务上window 会搭配其他的operators 使用，例如我们想计算一秒钟内触发了几次click 事件

```
var click = Rx.Observable.fromEvent(document, 'click');
var source = Rx.Observable.interval(1000);
var example = click.window(source)
```

JSBin | 的jsfiddle

注意这里我们把source跟click对调了，并用到了observable的一个方法 `count()` ，可以用来取得observable总共送出了几个元素，用Marble Diagram表示如下

```
source : ---------0---------1---------2--...
click  : --cc---cc----c-c---------------...
                  window(source)
example: o--------o---------o---------o--..
         \        \         \         \
          -cc---cc|---c-c---|---------|--..
                  count()
       : o--------o---------o---------o--
         \        \         \         \
          -------4|--------2|--------0|--..
                  switch()
       : ---------4---------2---------0--...
```

从Marble Diagram 中可以看出来，我们把部分元素放到新的observable 中，就可以利用Observable 的方法做更灵活的操作

windowToggle

windowToggle 不像window 只能控制内部observable 的结束，windowToggle 可以传入两个参数，第一个是开始的observable，第二个是一个callback 可以回传一个结束的observable，让我们来看范例

```
var source = Rx.Observable.interval(1000);
var mouseDown = Rx.Observable.fromEvent(document, 'mousedown');
var mouseUp = Rx.Observable.fromEvent(document, 'mouseup');

var example = source
  .windowToggle(mouseDown, () => mouseUp)
  .switch();

example.subscribe(console.log);
```

```
source   : ----0----1----2----3----4----5--...

mouseDown: -------D------------------------...
mouseUp  : ---------------------------U----...

        windowToggle(mouseDown, () => mouseUp)

         : -------o-------------------------...
                  \
                   -1----2----3----4--|
                   switch()
example  : ---------1----2----3----4---------...
```

从Marble Diagram 可以看得出来，我们用windowToggle 拆分出来内部的observable 始于mouseDown 终于mouseUp。

通过...分组

最后我们来讲一个实务上比较常用的operators - groupBy，它可以帮我们把相同条件的元素拆分成一个Observable，其实就跟平常在下SQL 是一样个概念，我们先来看个简单的例子

```
var source = Rx.Observable.interval(300).take(5);

var example = source
              .groupBy(x => x % 2);

example.subscribe(console.log);

// GroupObservable { key: 0, ...}
// GroupObservable { key: 1, ...}
```

JSBin | 的jsfiddle

上面的例子，我们传入了一个callback function 并回传groupBy 的条件，就能区分每个元素到不同的Observable 中，用Marble Diagram 表示如下

```
source : ---0---1---2---3---4|
```

```
            0-------2-------4|
```

在实务上，我们可以拿groupBy 做完元素的区分后，再对inner Observable 操作，例如下面这个例子我们将每个人的分数作加总再送出

```
var people = [
    {name: 'Anna', score: 100, subject: 'English'},
    {name: 'Anna', score: 90, subject: 'Math'},
    {name: 'Anna', score: 96, subject: 'Chinese' },
    {name: 'Jerry', score: 80, subject: 'English'},
    {name: 'Jerry', score: 100, subject: 'Math'},
    {name: 'Jerry', score: 90, subject: 'Chinese' },
];
var source = Rx.Observable.from(people)
                                                    .zip(
                                                      Rx.Observable.inte
                                                      (x, y) => x);

var example = source
  .groupBy(person => person.name)
  .map(group => group.reduce((acc, curr) => ({
            name: curr.name,
            score: curr.score + acc.score
        })))
        .mergeAll();

example.subscribe(console.log);
// { name: "Anna", score: 286 }
// { name: 'Jerry', score: 270 }
```

JSBin | 的jsfiddle

这里我们范例是想把Jerry 跟Anna 的分数个别作加总，画成Marble Diagram 如下

```
source : --o--o--o--o--o--o|

  groupBy(person => person.name)
```

```
             o--o--o--|

            map(group => group.reduce(...))

        : --i---------i------|
            \         \
            o|        o|

              mergeAll()
example: --o---------o------|
```

今日小结

今天讲了两个可以把元素拆分到新的observable 的operators，这两个operators 在前端比较少用到，但在后端或是比较复杂了前端应用才比较有机会用到。不知道读者有没有收获呢？如果有任何问题欢迎留言给我，谢谢。

标签

JavaScript RxJS 可观察的 运算符 RxJS 30天

上一页 下一个