30 天精通RxJS (13)：Observable Operator - delay, delayWhen

Dec 28th, 2016 . 4 mins read

> 在所有非同步中行为中，最麻烦的大概就是UI 操作了，因为UI 是直接影响使用者的感受，如果处理的不好对使用体验会大大的扣分！

UI 大概是所有非同步行为中最不好处理的，不只是因为它直接影响了用户体验，更大的问题是UI 互动常常是高频率触发的事件，而且多个元件间的时间序需要不一致，要做到这样的UI 互动就不太可能用Promise 或async/await，但是用RxJS 仍然能轻易地处理！

今天我们要介绍的两个Operators，delay 跟delayWhen 都是跟UI 互动比较相关的。当我们的网页越来越像应用程式时，UI 互动就变得越重要，让我们来试试如何用RxJS 完成基本的UI 互动！

Operators

delay

delay 可以延迟observable 一开始发送元素的时间点，范例如下

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

var example = source.delay(500);

example.subscribe({
    next: (value) => { console.log(value); },
    error: (err) => { console.log('Error: ' + err); },
    complete: () => { console.log('complete'); }
});
// 0
// 1
// 2
// 3
```

~~当然直接从log 出的讯息看，是完全看不出差异的~~

让我们直接看Marble Diagram

```
source : --0--1--2--3--4|
         delay(500)
example: -------0--1--2--3--4|
```

从Marble Diagram 可以看得出来，第一次送出元素的时间变慢了，虽然在这里看起来没什么用，但是在UI 操作上是非常有用的，这个部分我们最后示范。

delay 除了可以传入毫秒以外，也可以传入Date 型别的资料，如下使用方式

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

var example = source.delay(new Date(new Date().getTime() + 1000));

example.subscribe({
    next: (value) => { console.log(value); },
    error: (err) => { console.log('Error: ' + err); },
    complete: () => { console.log('complete'); }
});
```

JSBin | JSFiddle

~~这好像也能用在预定某个日期，让程式挂掉~~

delayWhen

delayWhen 的作用跟delay 很像，最大的差别是delayWhen 可以影响每个元素，而且需要传一个callback 并回传一个observable，范例如下

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

var example = source
              .delayWhen(
                  x => Rx.Observable.empty().delay(100 * x * x)
              );
```

```
    complete: () => { console.log('complete'); }
});
```

JSBin | JSFiddle

这时我们的Marble Diagram 如下

```
source : --0--1--2--3--4|
    .delayWhen(x => Rx.Observable.empty().delay(100 * x * x));
example: --0---1----2-----3-----4|
```

这里传进来的x 就是source 送出的每个元素，这样我们就能对每一个做延迟。

这里我们用delay 来做一个小功能，这个功能很简单就是让多张照片跟着滑鼠跑，但每张照片不能跑一样快！

首先我们准备六张大头照，并且写进HTML

```
<img src="https://res.cloudinary.com/dohtkyi84/image/upload/c_scale,w_5
<img src="https://res.cloudinary.com/dohtkyi84/image/upload/c_scale,w_5
<img src="https://res.cloudinary.com/dohtkyi84/image/upload/c_scale,w_5
<img src="https://res.cloudinary.com/dohtkyi84/image/upload/c_scale,w_5
<img src="https://res.cloudinary.com/dohtkyi84/image/upload/c_scale,w_5
<img src="https://res.cloudinary.com/dohtkyi84/image/upload/c_scale,w_5
```

用CSS 把img 改成圆形，并加上边筐以及绝对位置

```
img{
  position: absolute;
  border-radius: 50%;
  border: 3px white solid;
  transform: translate3d(0,0,0);
}
```

再来写JS，一样第一步先抓DOM

```
var imgList = document.getElementsByTagName('img');
```

第二步建立observable

这里的范例source2 是每一秒就会送出一个元素，我们可以改用bufferTime 简洁的表达，如下

```
function followMouse(DOMArr) {
  const delayTime = 600;
  DOMArr.forEach((item, index)

example => {
    movePos
      .delay(delayTime * (Math.pow(0.65, index) + Math.cos(index / 4))
      .subscribe(function (pos){
        item.style.transform = 'translate3d(' + pos.x + 'px, ' + pos.y
      });
  });
}

followMouse(Array.from(imgList))
```

这里我们把imgList从Collection转成Array后传入 `followMouse()` ，并用forEach把每个omg取出并利用index来达到不同的delay时间，这个delay时间的逻辑大家可以自己想，不用跟我一样，最后subscribe就完成啦！

最后完整的范例在 这里

今日小结

今天我们介绍了两个operators 并带了一个小范例，这两个operators 在UI 操作上都非常的实用，我们明天会接着讲几个operators 可以用来做高频率触发的事件优化！

Tags

JavaScript RxJS Observable Operator RxJS 30 Days

Series / 30 天精通RxJS