Rxjs 异步数据流编程-Rxjs 快速入门教程

主讲教师：（大地）

合作网站：www.itying.com （IT 营）

我的专栏：https://www.itying.com/category-79-b0.html

目录

一、 Rxjs 介绍.....1
二、 Promise 和 RxJS 处理异步对比.....2
三、 Rxjs unsubscribe 取消订阅.....3
四、 Rxjs 订阅后多次执行.....3
五、 Angualr6.x 之前使用 Rxjs 的工具函数 map filter.....4
六、 Angualr6.x 以后 Rxjs6.x 的变化以及使用.....5
七、 Rxjs 延迟执行.....6

一、 Rxjs 介绍

参考手册：https://www.npmjs.com/package/rxjs

中文手册：https://cn.rx.js.org/

RxJS 是 ReactiveX 编程理念的 JavaScript 版本。ReactiveX 来自微软，**它是一种针对异步数据流的编程**。简单来说，它将一切数据，包括 HTTP 请求，DOM 事件或者普通数据等包装成流的形式，然后用强大丰富的操作符对流进行处理，使你能以同步编程的方式处理异步数据，并组合不同的操作符来轻松优雅的实现你所需要的功能。

RxJS 是一种针对异步数据流编程工具，或者叫响应式扩展编程；可不管如何解释 RxJS 其目标就是异步编程，Angular 引入 RxJS 为了就是让异步可控、更简单。

RxJS 里面提供了很多模块。这里我们主要给大家讲 RxJS 里面最常用的 Observable 和 fromEvent。

目前常见的异步编程的几种方法：

1、回调函数
2、事件监听/发布订阅
3、Promise
4、Rxjs

二、 Promise 和 RxJS 处理异步对比

Promise 处理异步:

```
let promise = new Promise(resolve => {
                setTimeout(() => {
                    resolve('---promise timeout---');
                }, 2000);
});
promise.then(value => console.log(value));
```

RxJS 处理异步：

```
import {Observable} from 'rxjs';

let stream = new Observable(observer => {
   setTimeout(() => {
          observer.next('observable timeout');
   }, 2000);
});

stream.subscribe(value => console.log(value));
```

从上面列子可以看到 RxJS 和 Promise 的基本用法非常类似，除了一些关键词不同。Promise 里面用的是 then() 和 resolve()，而 RxJS 里面用的是 next() 和 subscribe()。

从上面例子我们感觉 Promise 和 RxJS 的用法基本相似。其实 Rxjs 相比 Promise 要强大很多。比如 Rxjs 中可以中途撤回、Rxjs 可以发射多个值、Rxjs 提供了多种工具函数等等。

三、 Rxjs unsubscribe 取消订阅

Promise 的创建之后，动作是无法撤回的。Observable 不一样，动作可以通过 unsbscribe() 方法中途撤回，而且 Observable 在内部做了智能的处理.

Promise 创建之后动作无法撤回

```
let promise = new Promise(resolve => {
                setTimeout(() => {
                      resolve('---promise timeout---');
                }, 2000);
          });
promise.then(value => console.log(value));
```

Rxjs 可以通过 unsubscribe() 可以撤回 subscribe 的动作

```
  let stream = new Observable(observer => {
       let timeout = setTimeout(() => {
            clearTimeout(timeout);
            observer.next('observable timeout');
       }, 2000);
});
let disposable = stream.subscribe(value => console.log(value));
setTimeout(() => {
            //取消执行
       disposable.unsubscribe();
}, 1000);
```

四、 Rxjs 订阅后多次执行

如果我们想让异步里面的方法多次执行，比如下面代码。

这一点 Promise 是做不到的，对于 Promise 来说，最终结果要么 resole（兑现）、要么 reject（拒绝），而且都只能触发一次。如果在同一个 Promise 对象上多次调用 resolve 方法，则会抛异常。而 Observable 不一样，它可以不断地触发下一个值，就像 next() 这个方法的名字所暗示的那样。

```
 let promise = new Promise(resolve => {
   setInterval(() => {
         resolve('---promise setInterval---');
   }, 2000);
});
promise.then(value => console.log(value));
```

Rxjs

```
let stream = new Observable<number>(observer => {
       let count = 0;
       setInterval(() => {
            observer.next(count++);
       }, 1000);

});
stream.subscribe(value => console.log("Observable>"+value));
```

五、 Angualr6.x 之前使用 Rxjs 的工具函数 map filter

注意：Angular6 以后使用以前的 rxjs 方法，必须安装 rxjs-compat 模块才可以使用 map、filter 方法。

angular6 后官方使用的是 RXJS6 的新特性，所以官方给出了一个可以暂时延缓我们不需要修改 rsjx 代码的办法。

```
npm install rxjs-compat
```

```
import {Observable} from 'rxjs';
import 'rxjs/Rx';
```

```
let stream= new Observable<any>(observer => {
    let count = 0;
    setInterval(() => {
    observer.next(count++);
    }, 1000);
});

stream.filter(val=>val%2==0)
    .subscribe(value => console.log("filter>"+value));

stream
    .map(value => {
        return value * value
    })
.subscribe(value => console.log("map>"+value));
```

六、 Angualr6.x 以后 Rxjs6.x 的变化以及使用

Rxjs 的变化参考文档：http://bbs.itying.com/topic/5bfce189b110d80f905ae545

RXJS6 改变了包的结构，主要变化在 import 方式和 operator 上面以及使用 pipe()

```
import {Observable} from 'rxjs';
import {map,filter} from 'rxjs/operators';
```

```
let stream= new Observable<any>(observer => {
```

```
    let count = 0;
    setInterval(() => {
    observer.next(count++);
    }, 1000);
});
stream.pipe(
        filter(val=>val%2==0)
     )
    .subscribe(value => console.log("filter>"+value));
stream.pipe(
    filter(val=>val%2==0),
    map(value => {
          return value * value
})
    )
.subscribe(value => console.log("map>"+value));
```

七、 Rxjs 延迟执行

```
import {Observable,fromEvent} from 'rxjs';
import {map,filter,throttleTime} from 'rxjs/operators';
var button = document.querySelector('button');
fromEvent(button, 'click').pipe(
throttleTime(1000)
)
.subscribe(() => console.log(`Clicked`));
```