使用RxJS构建流式前端应用

@ 郭林烁 (joeyguo)

关于我

郭林烁

joeyguo

腾讯前端 AlloyTeam

技术钻研

性能优化

blog: https://github.com/joeyguo/blog

大纲

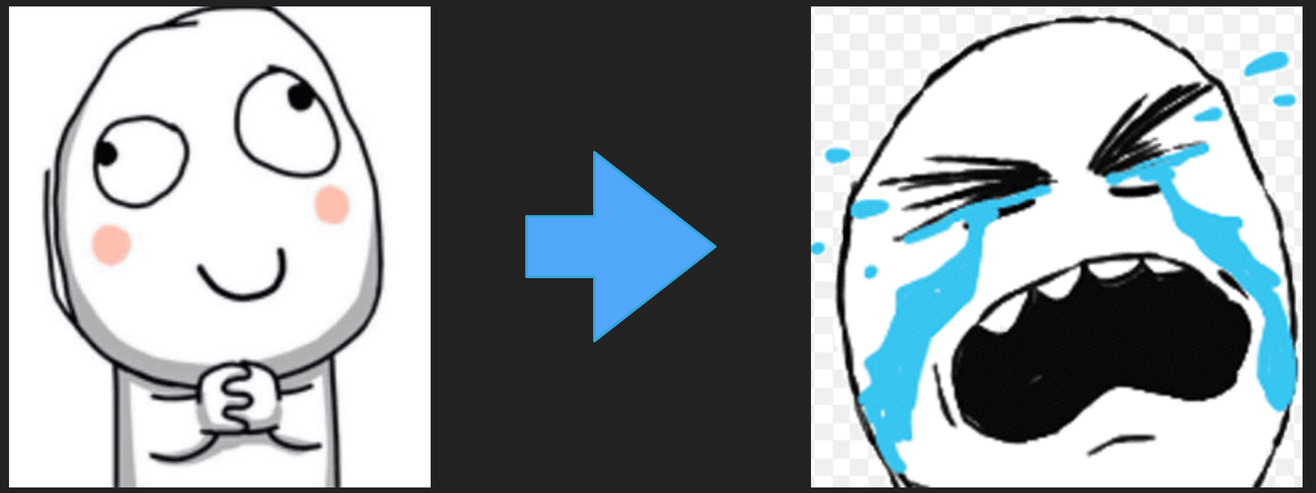
- 走进 Rx 与流
- 从入门到理解
- 应用实践
- 编码体会&总结

搜索功能

AC2016

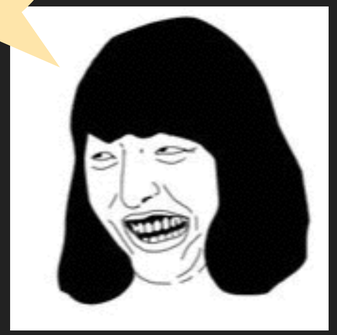

```
$.ajax({
    url: `search.qq.com/${text}`,
    success: data => {
        render(data);
    }
});
```

用户反馈

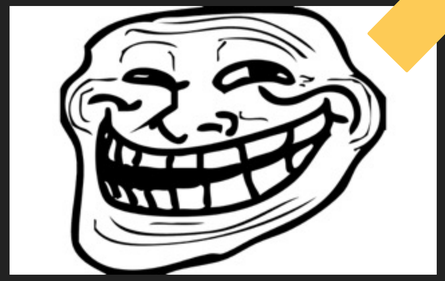


场景重现

想干嘛



上网搜 成人



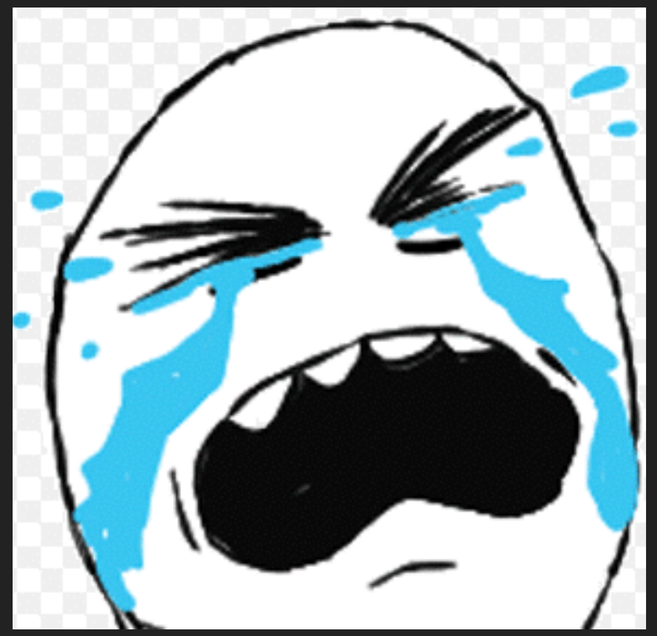
改搜 读书

理论分析

request 成人

request 书籍

response 成人



response 书籍

优化方案

1.数据约定

返回当前搜索的标志

2.发起前

声明一个当前搜索状态

3.返回后

判断返回标志与当前搜索状态是否一致

```
var currentSearch = '书';
$.ajax({
    url: `search.qq.com/${text}`,
    success: data => {
        if (data.search ===
                currentSearch) {
            render(data);
        } else {
            // ..
        }
    }
});
```

最终代码

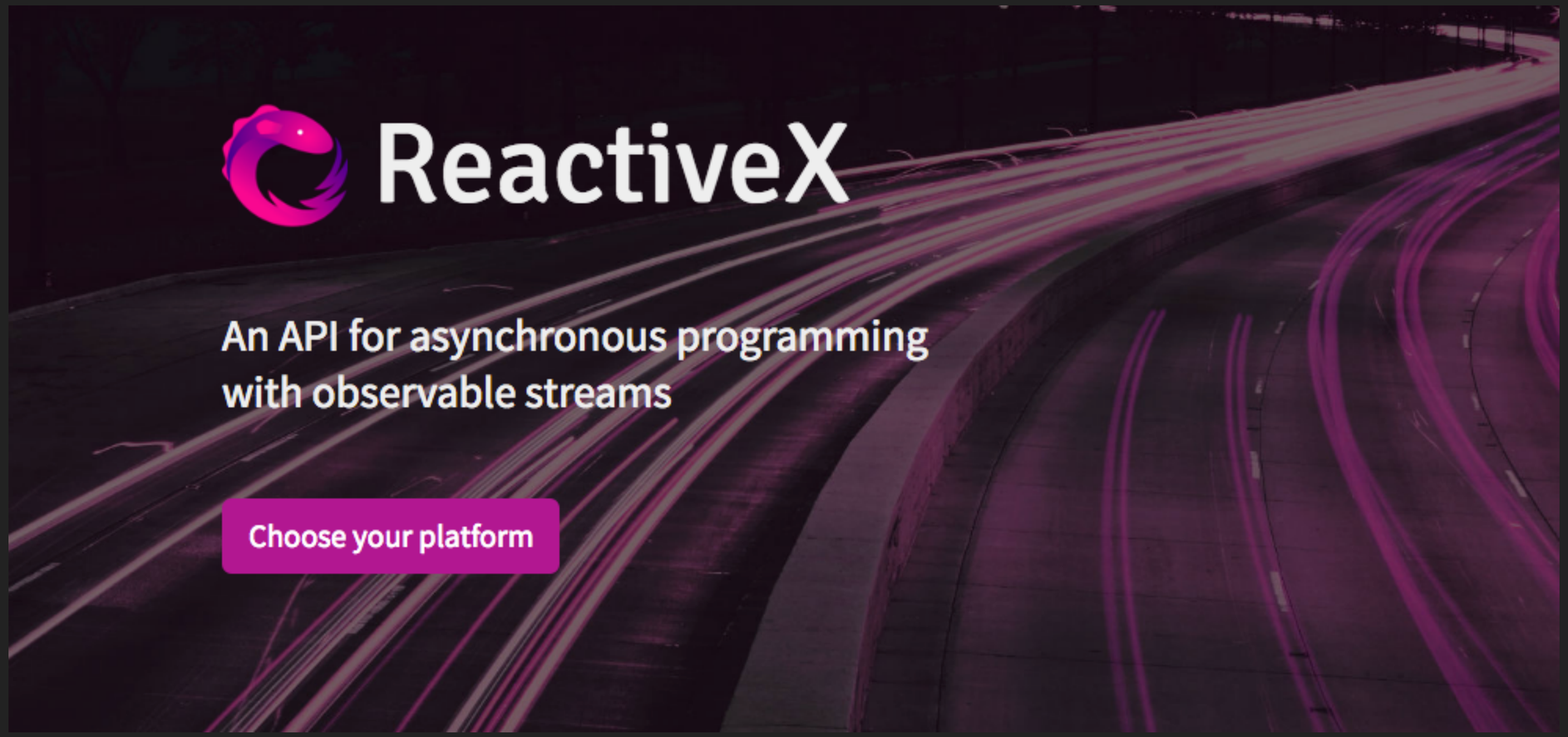
```
var text = document.querySelector('#text'),
    timer = null,
    currentSearch = '';
text.addEventListener('keyup', () =>{
    clearTimeout(timer)
    timer = setTimeout(() => {
        currentSearch = '书';
        $.ajax({
            url: `search.qq.com/${text}`,
            success: data => {
                if (data.search === currentSearch) {
                    render(data);
                } else {
                    // ..
                }
            }
        });
    },250)
})
```

有没有一种 优雅 的方式?

RxJS

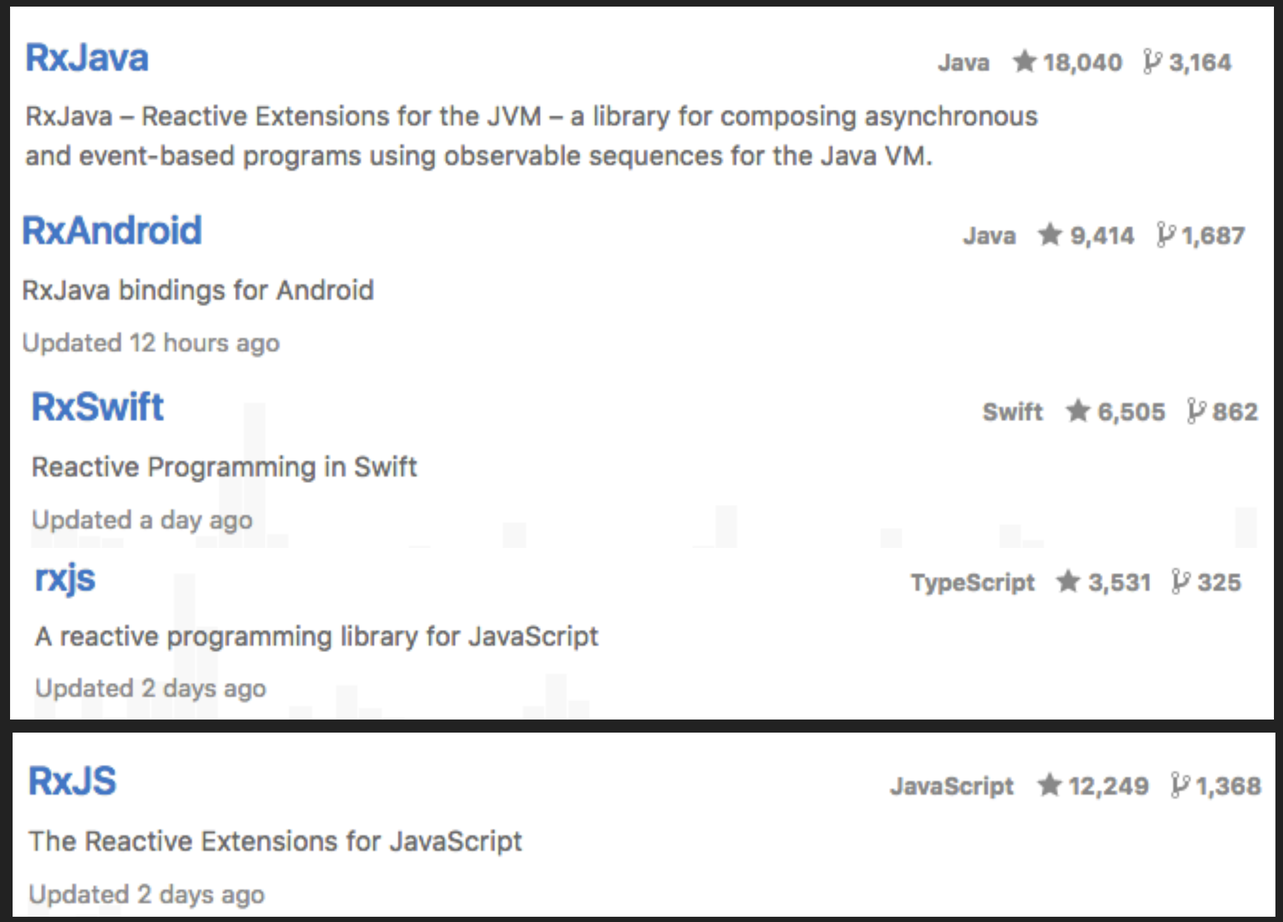
```
var text = document.querySelector('#text');
var inputStream = Rx.Observable.fromEvent(text, 'keyup')
                  .pluck('target', 'value')
                  .flatMapLatest(url => Http.get(url))
                  .subscribe(data => render(data))
```

Reactive Extensions



http://reactivex.io/rxjs/

Rx Github


RxJava
Java ★ 18,040 3,164
RxJava – Reactive Extensions for the JVM – a library for composing asynchronous and event-based programs using observable sequences for the Java VM.
RxAndroid
Java ★ 9,414 1,687
RxJava bindings for Android
Updated 12 hours ago
RxSwift
Swift ★ 6,505 862
Reactive Programming in Swift
Updated a day ago
rxjs
TypeScript ★ 3,531 325
A reactive programming library for JavaScript
Updated 2 days ago
RxJS
JavaScript ★ 12,249 1,368
The Reactive Extensions for JavaScript
Updated 2 days ago

流　随时间流逝的一系列事件

时间

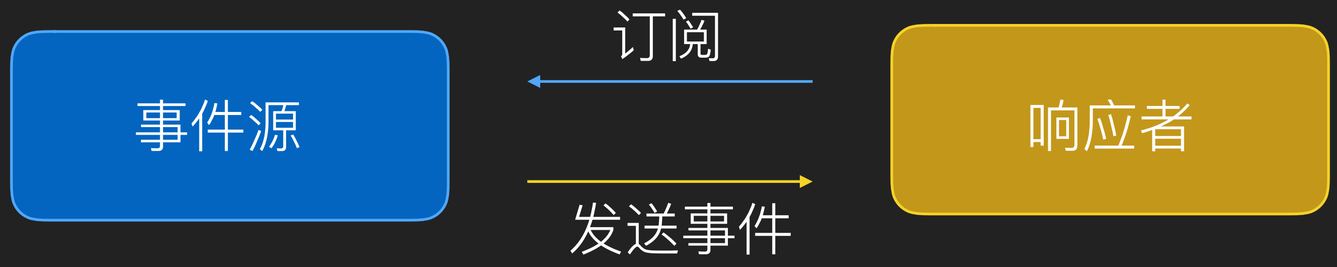
事件

鼠标点击、键盘点击、 数据集合...

入门

Observables（事件源，被观察者）

Observer（响应者，观察者）



入门 show the code

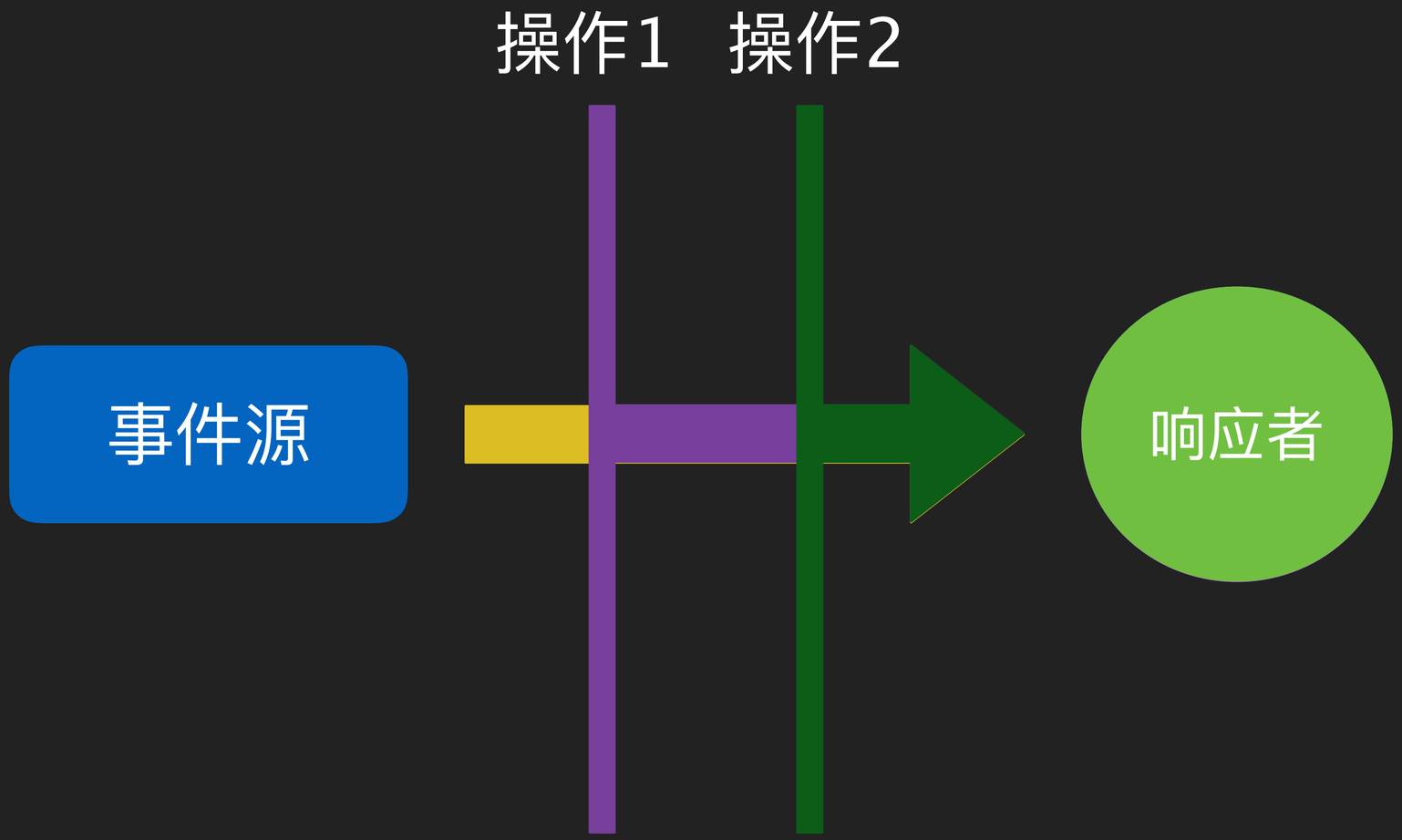
事件源 `var streamA$ = Rx.Observable.just(2);`

订阅与响应 `streamA$.subscribe(v => console.log(v));`

```
var streamA$ = Rx.Observable.just(2)
                    .map(s => s * 2)
                    .subscribe(v => console.log(v))
```

Operator 操作

基础组成



操作 Rx 提供了许多接口

创建　　转变　　过滤　　组合　　错误处理　　...

http://reactivex.io/documentation/operators.html

操作 Rx 提供了许多接口

创建　转变　过滤　组合　错误处理　...

Just

```
Rx.Observable.just(2)
    .subscribe(v => console.log(v));
```

fromEvent

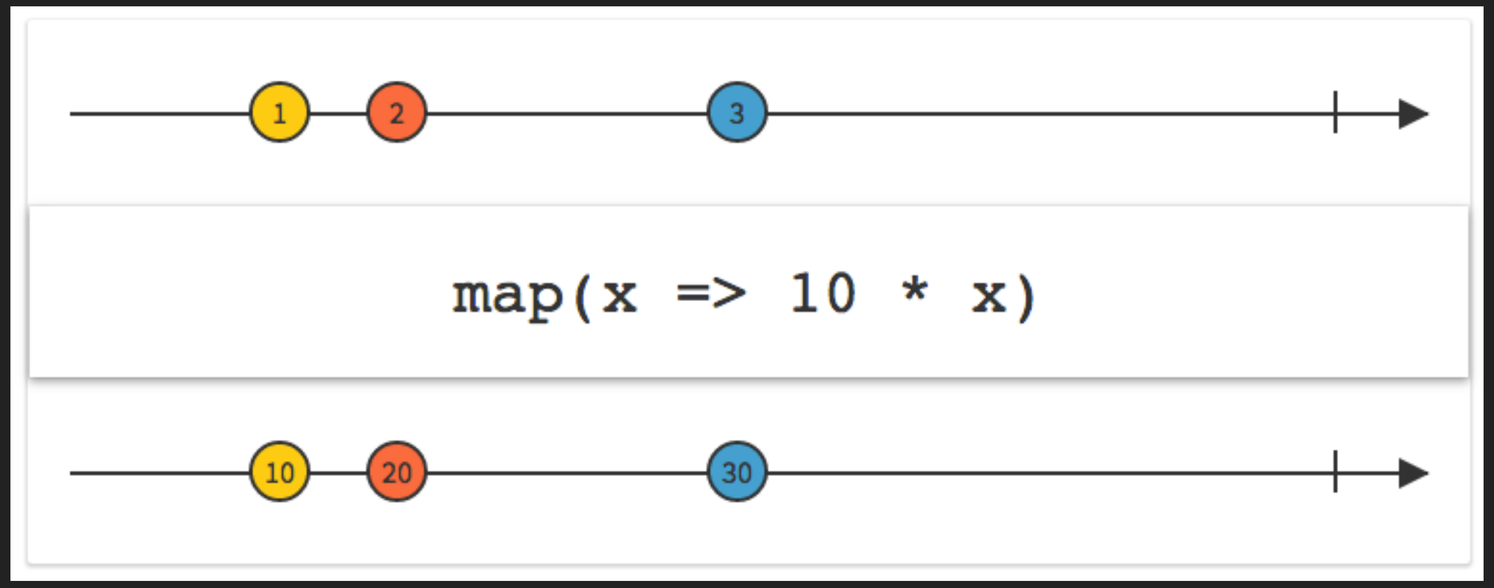
```
Rx.Observable.fromEvent($text, 'keyup')
                    .subscribe(e => console.log(e));
```

操作 Rx 提供了许多接口

创建　　转变　　过滤　　组合　　错误处理　　...

Map

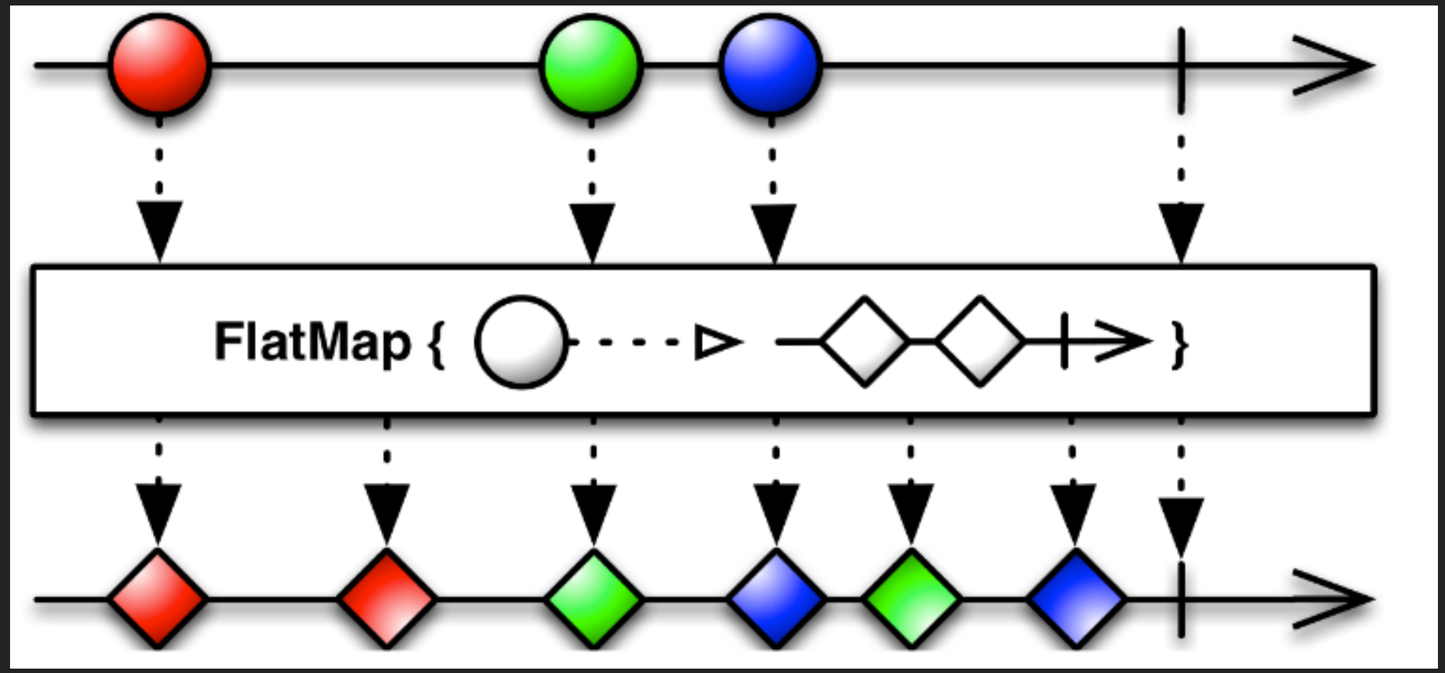
```
Rx.Observable.just(2)
    .map(v => 10 * v)
    .subscribe(v => console.log(v));
```



操作 Rx 提供了许多接口

创建　　转变　　过滤　　组合　　错误处理　　...

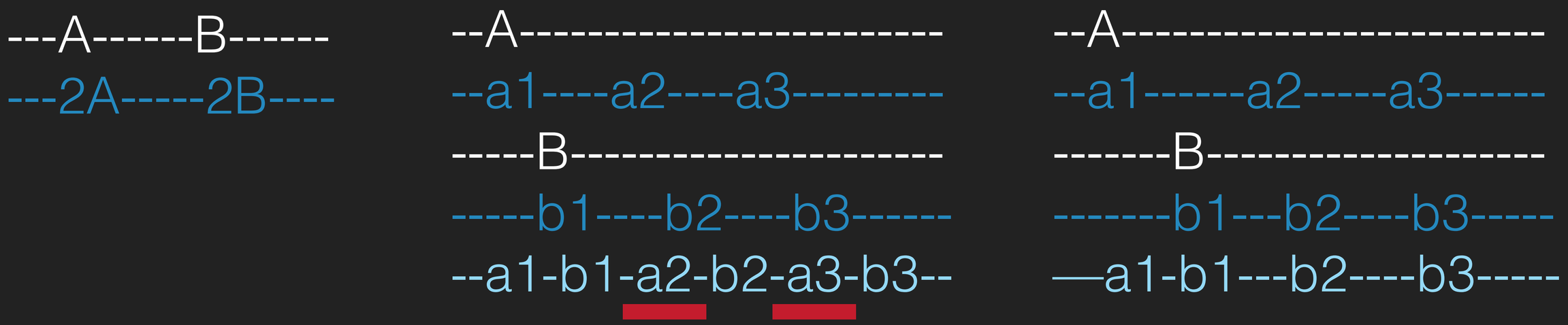
FlatMap



操作 Rx 提供了许多接口

创建　转变　过滤　组合　错误处理　...

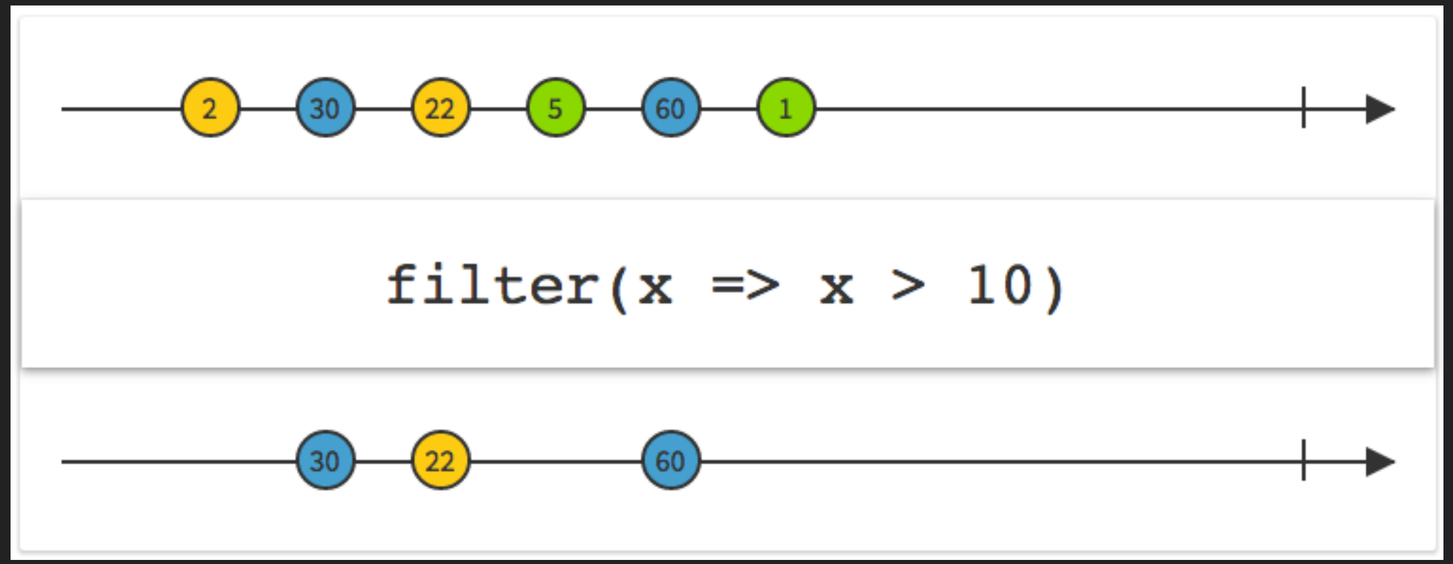
Map vs FlatMap vs FlatMapLatest



操作 Rx 提供了许多接口

创建　　转变　　过滤　　组合　　错误处理　　...

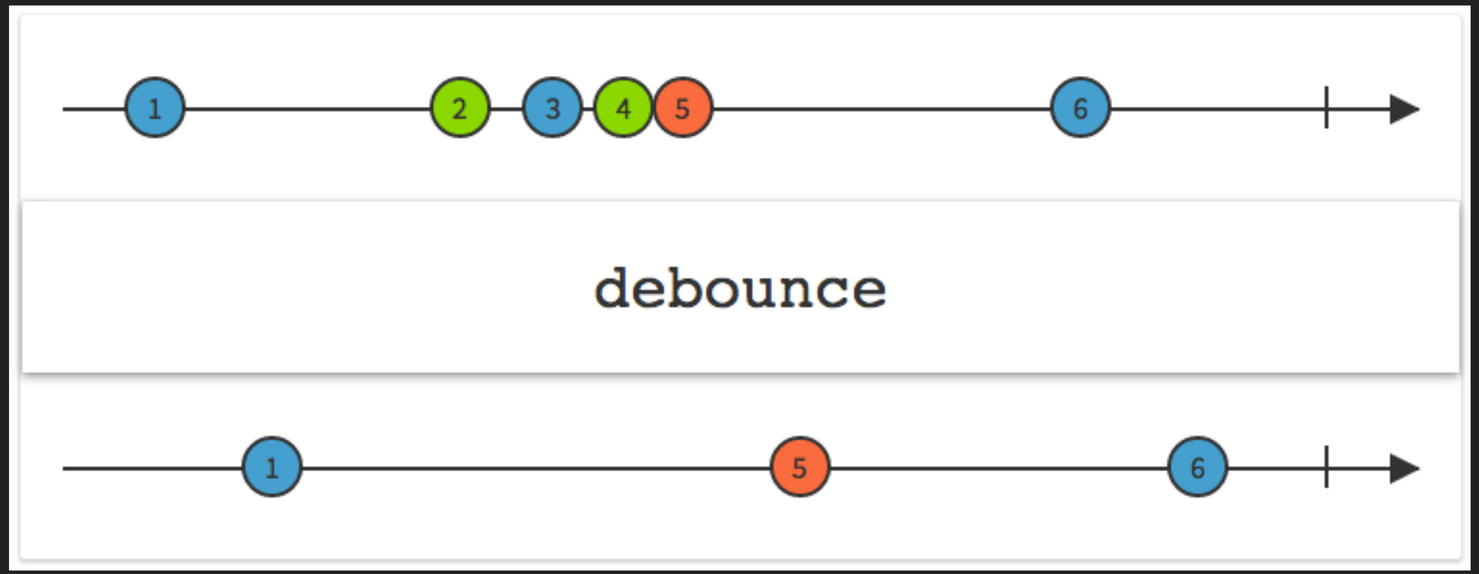
Filter



操作 Rx 提供了许多接口

创建　　转变　　过滤　　组合　　错误处理　　...

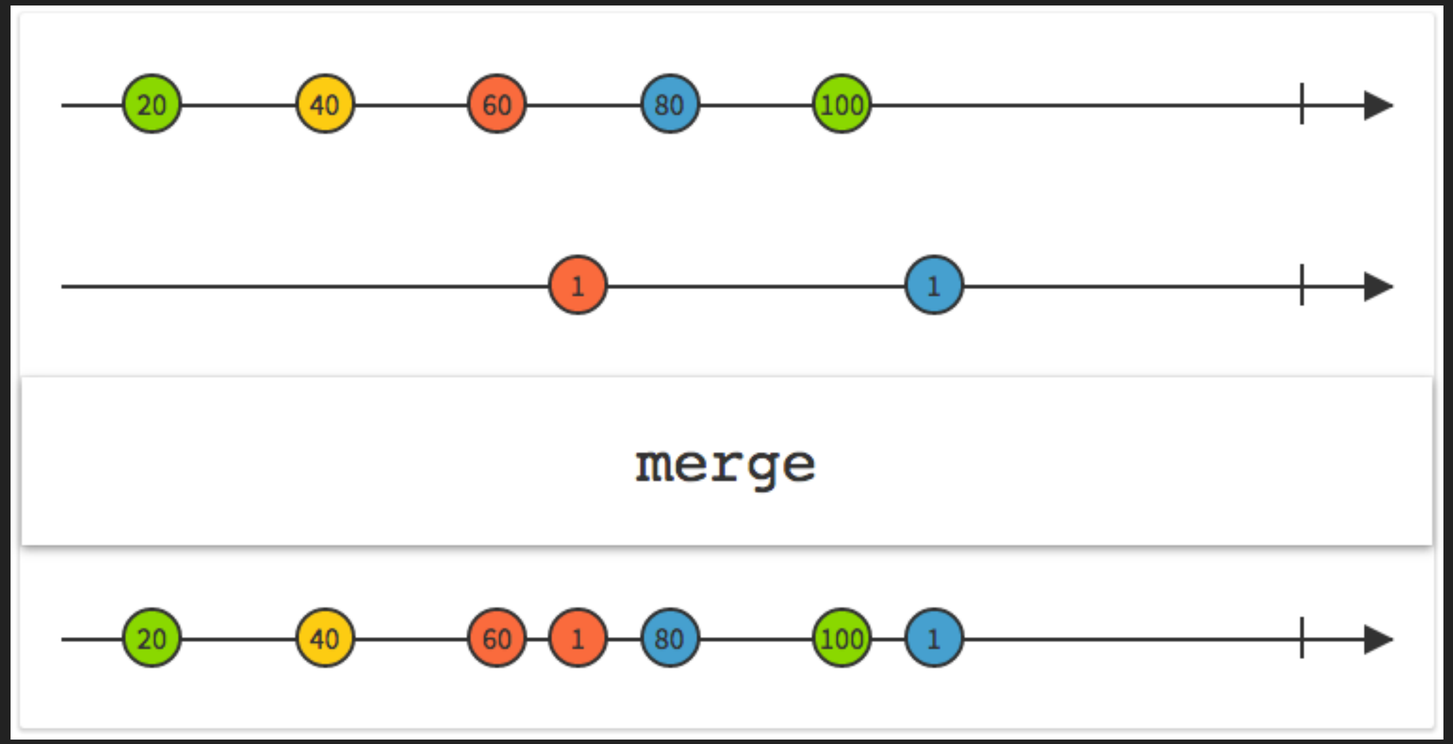
Debounce



操作 Rx 提供了许多接口

创建　　转变　　过滤　　组合　　错误处理　　...

Merge



操作 Rx 提供了许多接口

创建　　转变　　过滤　　组合　　错误处理　　...

Catch

```
var anotherStream =  Rx.Observable
                        .just(`search.2.qq.com`);

var oneStream =  Rx.Observable.just(`search.qq.com`)
                    .flatMap(url => Http.get(url))
                    .catch(anotherStream)
                    .subscribe(
                        data => render(data)
                        err => console.log(`wrong: ${err}`),
                        () => console.log(`complete`));
```

示例 回到一开始的例子

```
var text = document.querySelector('#text');
var inputStream = Rx.Observable.fromEvent(text, 'keyup')
                    .pluck('target', 'value')
                    .flatMapLatest(url => Http.get(url))
                    .subscribe(data => render(data))
```

示例 完善与增加功能

```
var text = document.querySelector('#text');
var inputStream = Rx.Observable.fromEvent(text, 'keyup')
                    .pluck('target', ‘value')
                    .filter(text => text.length > 1)
                    .flatMapLatest(url => Http.get(url))
                    .catch(anotherStream)
                    .map(v => v * 2)
                    .subscribe(
                        data => render(data)
                        err => console.log(`wrong: ${err}`),
                        () => console.log(`complete`))
```

示例 实现 N 次点击监听

```
var button = document.querySelector('.btn');
var clickStream = Rx.Observable.fromEvent(button, 'click');

var multiClickStream = clickStream
                            .buffer(() => clickStream.throttle(250))
                            .map(list => list.length)
                            .filter(x => x >= 2);

multiClickStream.subscribe(numclicks => render(numclicks));
```

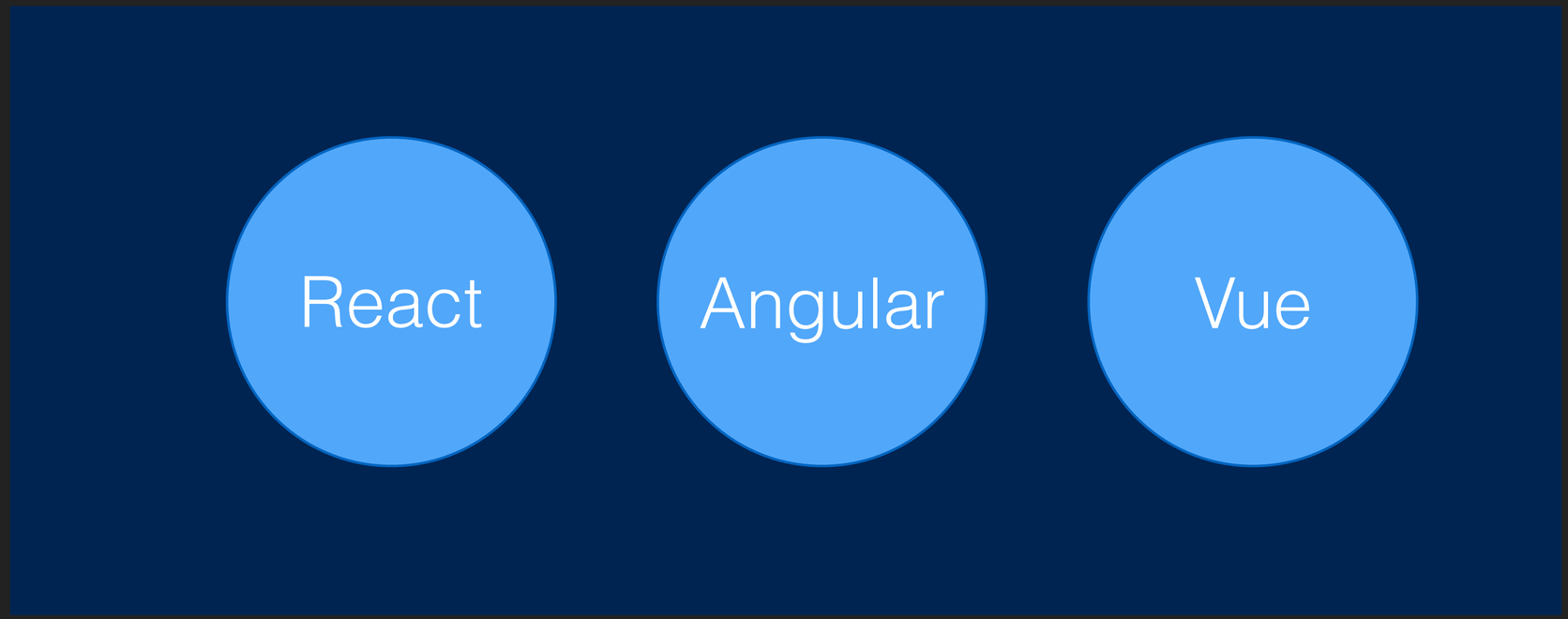
so easy :D

编码体会&总结

- 代码简洁
- 函数式
- 可组合
- 伸缩性强
- 错误处理
- 并发处理
- 局部到整体
- 同步与异步
- 事件与数据
- 前端与后端

与其他框架结合

```
anyStream.subscribe(data => {
```




```
});
```

更多

Observable
数据源

Observer
观察者

Operators
操作

Subscription
订阅

Subject
发射器

Schedulers
调度

Subject.complete();

@ 郭林烁 (joeyguo)

Q & A