预习资料：

预习资料名称	链接	备注
Rxjs的思维导图	请在课程学习目录页下载《【复习】异步编程-Rxjs知识点思维导图》（在《异步编程》3.7、3.8课程后）	可以借助思维导图了解视频的大致内容，后续可以借助思维导图复习巩固
Rxjs的官方指导网站	https://rxjs.dev/guide/overview	网站是英文版的，英文基础好的同学可以看下
Rxjs的中文学习网站	https://cn.rx.js.org/	中文版的学习网站，可以结合视频翻阅
Rxjs的操作符学习网站	https://github.com/RxJS-CN/learn-rxjs-operators	可以在课后翻阅一下
Rxjs的操作符行为网站	https://reactive.how	可以在课后自行实现几个例子，对照操作符行为理解操作符
流动的数据——使用 RxJS 构造复杂单页应用的数据逻辑	https://zhuanlan.zhihu.com/p/23305264	使用Rxjs管理视图数据，可以看看文章介绍的使用场景

一、ReactiveX介绍

- Reactive Extensions的缩写，一般简写为Rx，它是一个编程的模型，它的目标是提供一致的编程接口，帮助我们开发者更方便的处理异步数据流，它属于响应式的一种编程方案
- Rxjs是JavaScript的语言实现
- 使用可观察的序列（Observable）来编写异步和基于事件的程序

什么是响应式编程?

与响应式编程对应的是命令式编程，例如：

```
// a b
a = 1;
b = a + 1;
a = 2;
```

上面代码中定义了a、b两个变量，a的值为1，b的值为a + 1，然后将a的值重新赋值为2，可以看到a的值为2时，b的值是不变的，b这一行的a还是取的第一行代码赋值的1，也就是说b在赋值的这一刻就计算好了，后续a的变化不会影响到b，这种我们叫命令式编程。

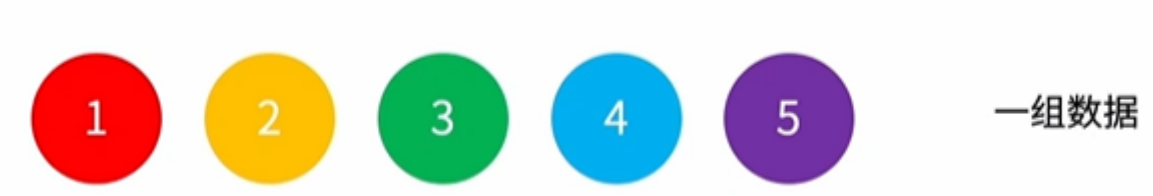
响应式编程是b会跟随a的变化而发生变化，也就是说在响应式编程下b和a之间维护的是一种关系，本例中b和a的关系就是b比a大1，不管a怎么变化，b永远都保持比a大1的这种关系，这种被称为响应式编程，因为a的变化会反应到b上面去。

这种在同步下可能没什么用，在异步情况下，a和b这两个数据不是同步产生的，它们是在不同的时间下由不同的行为产生的数据，如果不间断的产生数据，它就形成了一个数据流，这个数据流最终会反馈在我们的view层。

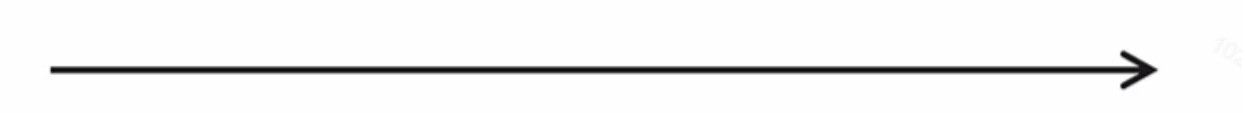
能不能对这个数据流做一层抽象呢?

这里就出现了Rx里面一个最终要的概念——可观察的序列（Observable），它是整个Rx的核心思想。

Observable是什么?



我们把上面这几个圈当做一组数据，接着往下看



这个箭头是一个时间线，上面的这组数据不是一次性产生的，它是随着时间的推移产生的

Observable：数据+时间

这五个数组就形成了一个数据流，在Rxjs的编程模型中就叫做Observable，翻译过来就是可观察的数据序列，所以Observable就是数据和时间的一个抽象。我们可以对数组进行一系列操作，比如拼接、过滤之类的，同样的，我们也可以对抽象出来的这个数据流进行一些操作，比如说我们可以创建、组合和过滤Observable。

数据产生之后最终要被消费掉，才能让这个数据真正产生了作用，所以从生产到消费就产生了一个数据传输。

数据传输的方式有哪些?

- 拉取：生产者只负责生产，不关心你什么时候用，由消费者自己决定

例如：有一个函数，我想用这个函数产生的数据的话，就去执行一下这个函数，拿到我想要的数据。

- 推送：生产完成之后就把数据推出去

例如：promise决议完成之后就把它的值推送给then函数里面的回调函数，因为消费者不知道什么时候能拉取到数据，所以它采用的是推送的方式。

Rx里面的Observable也是采用推送体系的。为什么用推送呢？因为Observable是包含时间信息的，如果是拉取的话，消费者怎么知道该什么时间去拉什么数据呢，所以Observable

索性就产生数据后推送给消费者。因为它是一个可观察的数据序列，它的底层是观察者模式实现的，所以它的消费者也就是它的观察者。

Rx编程模型的特点：

- 函数式风格：可以对Observable做一个无副作用的输入输出，避免了一些副作用的情况，这样输入输出都是可预测的，不容易出现bug
- 简化代码：Rx有一些操作符，这些操作符可以将复杂的问题简单化，也就是简化代码
- 异步错误处理：异步错误使用传统的try...catch是没办法处理的，Rx提供了一些异步错误处理的机制
- 轻松使用并发：Rx不仅有Observable的概念，还有Schedulers的概念，Schedulers可以帮助我们轻松实现并发

二、Rxjs用法

- Rxjs是ReactiveX编程模型的JavaScript语言实现
- 使用可观察的序列（Observable）来编写异步和基于事件的程序

我们知道Rx是一种编程模型，响应式编程还有别的实现方式，它是响应式编程的一种实现方式。说白了它是一个套路，这个套路必须要靠语言实现它才能使用，Rxjs就是Rx的JavaScript实现版，除了js语言还有其他语言的实现，比如java等，实现这个套路的语言有很多。我们知道Rx有个最重要的概念就是可观察的序列Observable，所以Rxjs中也必须要实现这个概念。

它是怎么实现的呢?

Rxjs结合了观察者模式、迭代器模式和函数式编程来实现Observable。

使用Rxjs的方式有：

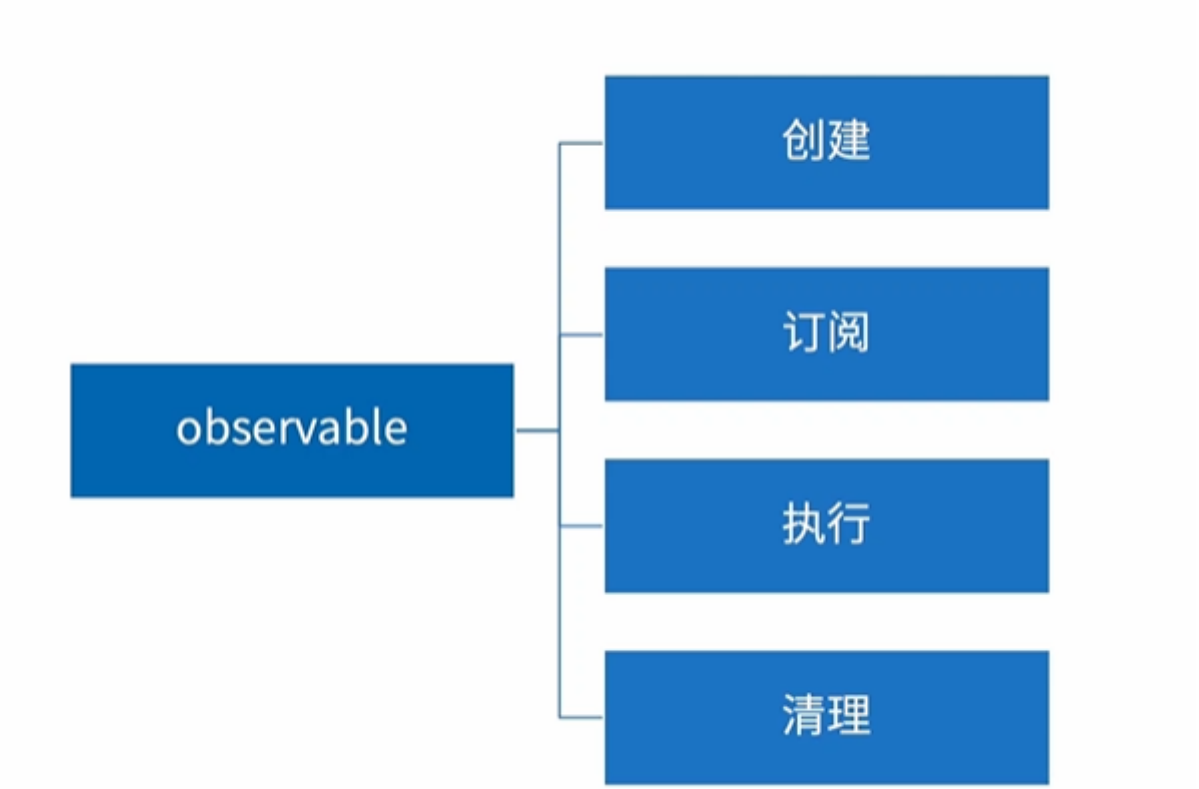
1、npm

- npm install rxjs
- import * as rxjs from 'rxjs'

2、cdn

- https://unpkg.com/@reactivex/rxjs@version/dist/global/rxjs.umd.js

Observable是Rx里面最核心的东西，在Rxjs里面也是一样的，它有哪些操作呢?



创建和订阅比较好理解，类似于观察者模式的概念，清理类似于取消订阅的意思，执行比较复杂，看一段代码

```
const docElm = document.documentElement;

const mouseMove$ = Rx.Observable
  .fromEvent(docElm, 'mousemove');

mouseMove$.subscribe(event => {
  docElm.innerHTML =
    `${event.clientX}, ${event.clientY}`
});

```

三、使用场景