Observable (可观察对象): 表示一个概念，这个概念是一个可调用的未来值或事件的集合。

Observer (观察者): 一个回调函数的集合，它知道如何去监听由 Observable 提供的值。

Subscription (订阅): 表示 Observable 的执行，主要用于取消 Observable 的执行。

Operators (操作符): 采用函数式编程风格的纯函数 (pure function)，使用像 map、filter、concat、flatMap 等这样的操作符来处理集合。

Subject (主体): 相当于 EventEmitter，并且是将值或事件多路推送给多个 Observer 的唯一方式。

Schedulers (调度器): 用来控制并发并且是中央集权的调度员，允许我们在发生计算时进行协调，例如 setTimeout 或 requestAnimationFrame 或其他。

订阅：Observer 通过 Observable 提供的 subscribe() 方法订阅 Observable。

发布：Observable 通过回调 next 方法向 Observer 发布事件。

Observable 就是一个函数，它接受一个 Observer 作为参数然后返回另一个函数。

创建一个 Observable 的实例，其中定义了一个订阅者（subscriber）函数。当有消费者调用 subscribe() 方法时，这个函数就会执行。订阅者函数用于定义“如何获取或生成那些要发布的值或消息”。

要执行所创建的可观察对象，并开始从中接收通知，你就要调用它的 subscribe() 方法，并传入一个观察者（observer）。这是一个 JavaScript 对象，它定义了你收到的这些消息的处理器（handler）。subscribe() 调用会返回一个 Subscription 对象，该对象具有一个 unsubscribe() 方法。当调用该方法时，你就会停止接收通知。

// Create an Observable that will start listening to geolocation updates

// when a consumer subscribes.

const locations = new Observable((observer) => {

// Get the next and error callbacks. These will be passed in when

// the consumer subscribes.

const {next, error} = observer; //next=observer.next, error=observer.error}

let watchId;

// Simple geolocation API check provides values to publish

if ('geolocation' in navigator) {

watchId = navigator.geolocation.watchPosition(next, error);

} else {

error('Geolocation not available');

}

// When the consumer unsubscribes, clean up data ready for next subscription.

return {unsubscribe() { navigator.geolocation.clearWatch(watchId); }};

});

// Call subscribe() to start listening for updates.

const locationsSubscription = locations.subscribe({

next(position) { console.log('Current Position: ', position); },

error(msg) { console.log('Error Getting Location: ', msg); }

});

// Stop listening for location after 10 seconds

setTimeout(() => { locationsSubscription.unsubscribe(); }, 10000);

Observable一些静态方法创建常用的可观察对象：

* of(...items) —— 返回一个 Observable 实例，它用同步的方式把参数中提供的这些值发送出来。
* from(iterable) —— 把它的参数转换成一个 Observable 实例。 该方法通常用于把一个数组转换成一个（发送多个值的）可观察对象。

// Create simple observable that emits three values

const myObservable = of(1, 2, 3);

// Create observer object

const myObserver = {

next: x => console.log('Observer got a next value: ' + x),

error: err => console.error('Observer got an error: ' + err),

complete: () => console.log('Observer got a complete notification'),

};

// Execute with the observer object

myObservable.subscribe(myObserver);

// Logs:

// Observer got a next value: 1

// Observer got a next value: 2

// Observer got a next value: 3

// Observer got a complete notification

订阅可观察对象： observable.subscribe(观察者)