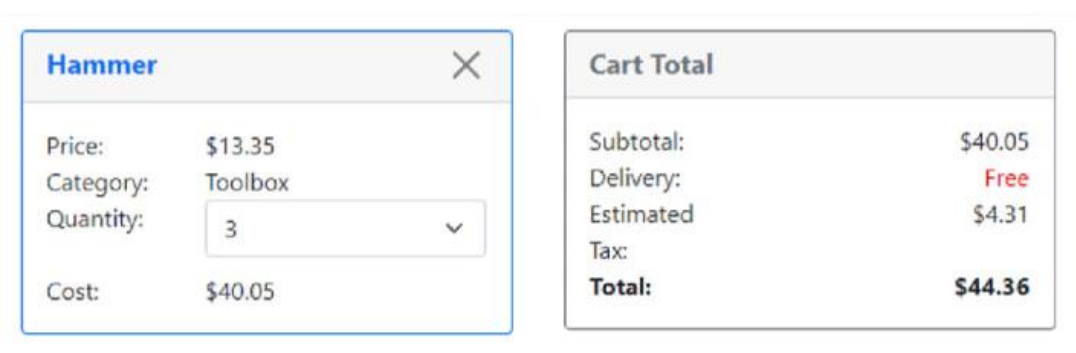
Reactive programming with RxJs

www.andersenlab.com

What is Reactive Programming?

> Reactive programming is a programming paradigm for writing code, mainly concerned with asynchronous data streams.

Code is reactive when an input change leads to an automatic change in output



Characteristics of Reactive Programming

- oriented on interaction with asynchronous data streams
- streams have the ability to transfer other streams
- data transmitted over the stream is listened by subscribers
- subscribers respond to data changes

What is RxJs ?

A library for reactive programming using Observables, to make it easier to compose asynchronous or callback-based code.

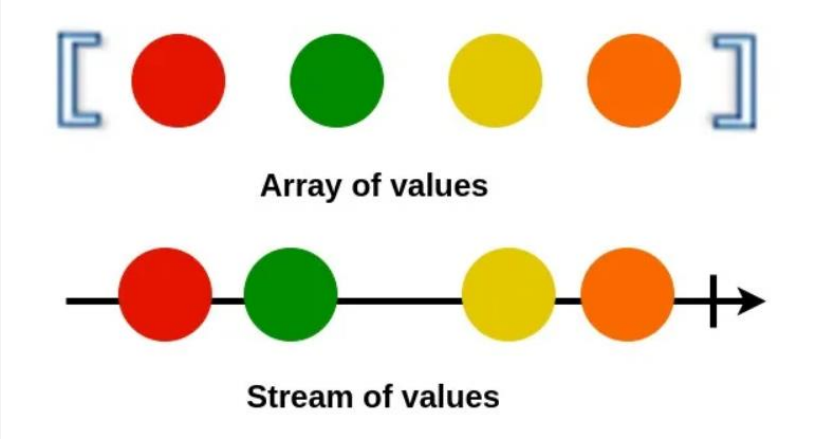
Observer

- is a behavioral design pattern that lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they’re observing.

```
const observer = {
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’),
};
```

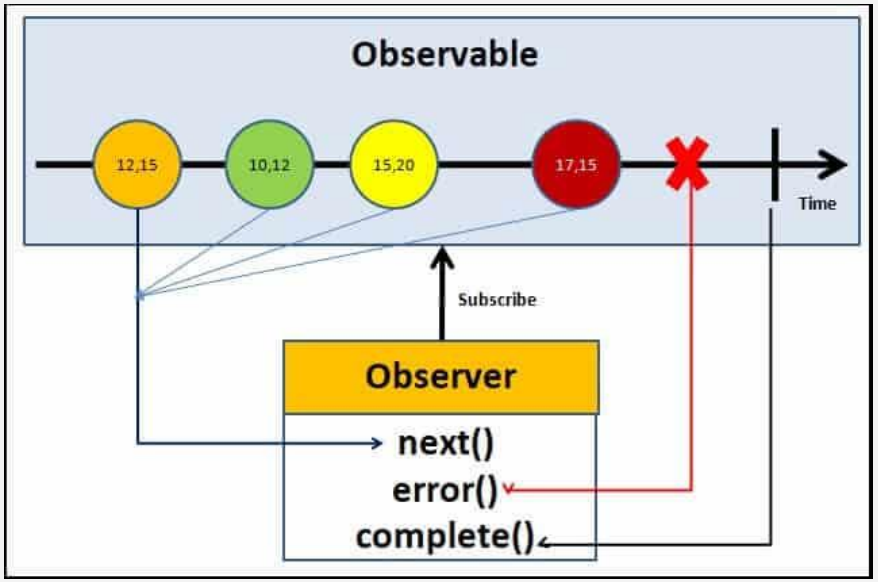
Stream

- source of data that can arrive over time



Observable

- an object that can be observed; they are lazy collections of multiple values over time



The Observable can emit the following event:

- next: Event is used to emit the next value from the Observable.
- error: Following event is used to notify Observer about some error.
- complete: Notifies that Observable has completed emitting data.

Creating Observables

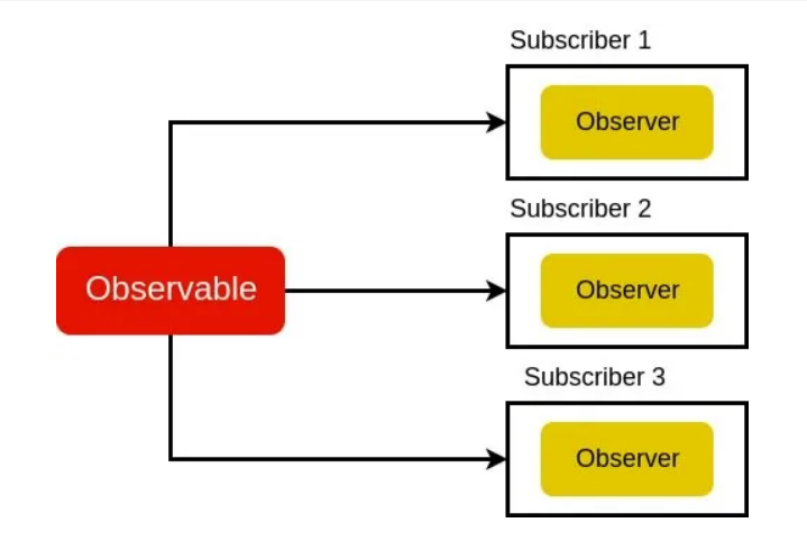
1. Observable class

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

const observable = new Observable(function
subscribe(subscriber) {
  const id = setInterval(() => {
    subscriber.next('hi');
  }, 1000);
});
```

2. Creation operator

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

of(1, 2, 3).subscribe({
next: value => console.log('next:', value),
error: err => console.log('error:', err),
complete: () => console.log('the end'),
});
```

Subscription



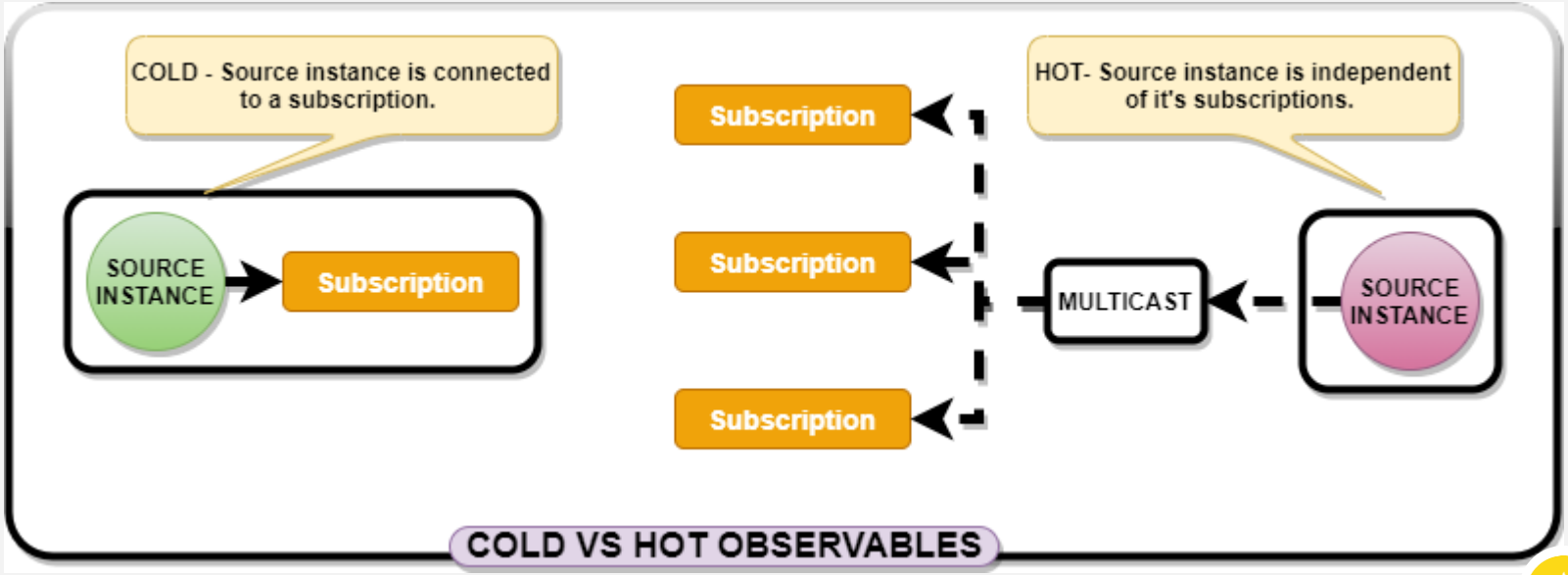
- An Observable instance begins publishing values only when someone subscribes to it:

```
// const subscription =  observable.subscribe(observer);
```

- A Subscription has one important method, unsubscribe(), that takes no argument and just disposes of the resource held by the subscription.

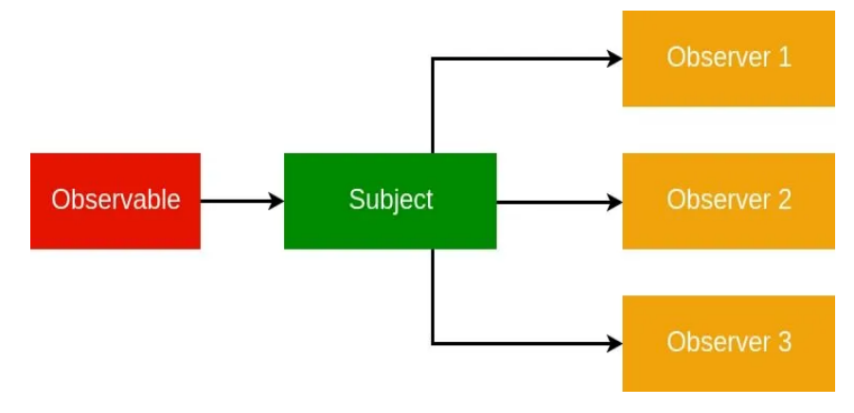
```
// subscription.unsubscribe();
```

Hot and Cold Observables



Subject

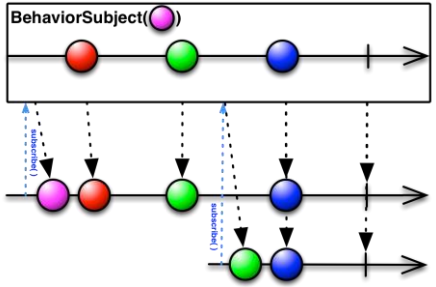
- special type of Observable that allows values to be multicasted to many Observers



- Subject is hot observable
- It can act as both an Observable and an Observer at the same time.
- Subject is another way to create a stream

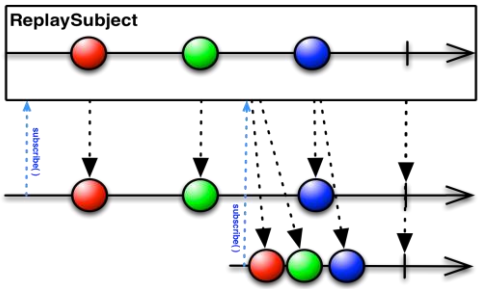
```
const source$ = new Subject();
```

Behavior Subject



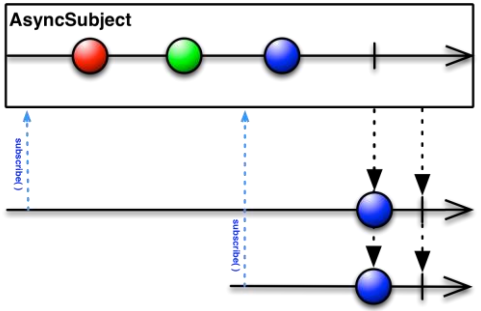
- allows you to set the initialization value in the constructor function

Replay Subject



- record a part of the Observable execution

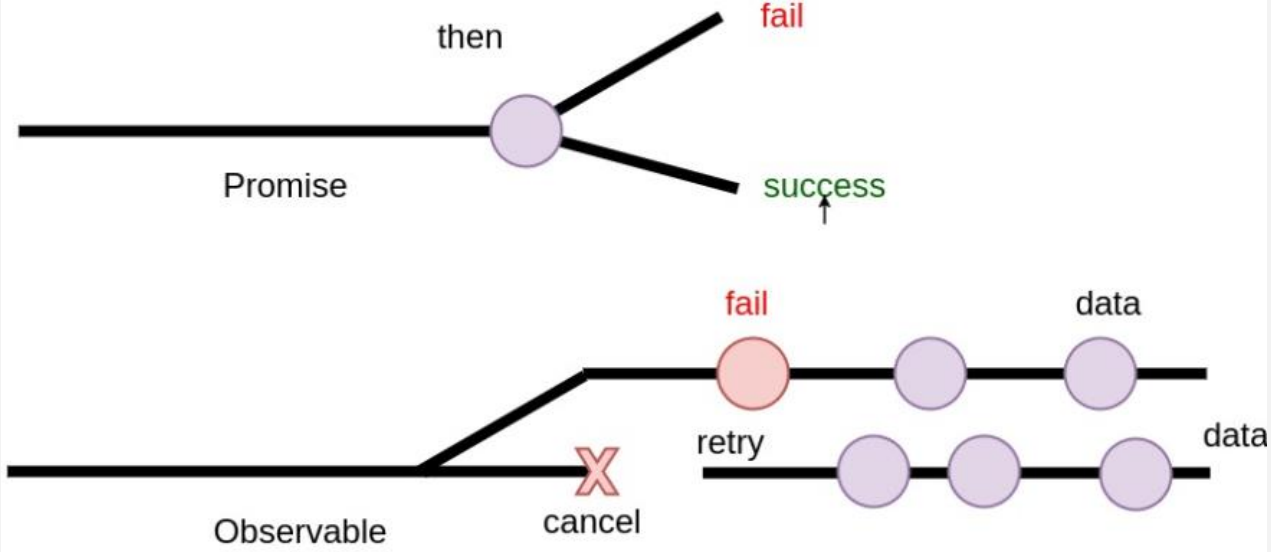
Async Subject



- the last value is sent when the execution completes

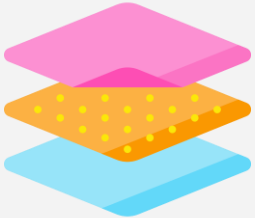
Why Use RxJs?

For async operations we can use: Callbacks, Promises, async/await



Benefits of using RxJs

One technique
to rule all data



Compositional

Watchful

Lazy

Handles errors

Cancellable

THANK YOU

www.andersenlab.com

@zelinskayaL