RXJS

Akila Iroshan

12/10/2017

REACTIVE

- Reactive： readily responsive to a stimulus
  - React to events (event-driven)
  - React to load (scalable)
  - React to failures (resilient)
  - React to users (responsive)

WHAT IS REACTIVE PROGRAMMING

- Programming paradigm that works with **asynchronous data streams**
- Data streams can be created from many things
  - UI events
  - Http requests
  - File Systems
  - Array-like objects
  - Memory/cache

REACTIVE EXTENSIONS/REACTIVEX

- A library for composing asynchronous programs by using observable sequences
- Provides a long list of operators which allow us to filter, select, transform, combine and compose observables

REACTIVEX

- Java: RxJava
- JavaScript: RxJS
- C#: Rx.NET
- C#(Unity): UniRx
- Scala: RxScala
- Clojure: RxClojure
- C++: RxCpp
- Lua: RxLua
- Ruby: Rx.rb
- Python: RxPY
- Go: RxGo
- Groovy: RxGroovy
- JRuby: RxJRuby
- Kotlin: RxKotlin
- Swift: RxSwift
- PHP: RxPHP
- Elixir: reaxive
- Dart: RxDart


Google

slack

facebook

Microsoft

NETFLIX

American Airlines

AMERICAN
EXPRESS

redhat

BANCO DE MÉXICO

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

infoComm16

GROUPON

DigitalOcean

criteo

MOTOROLA
SOLUTIONS

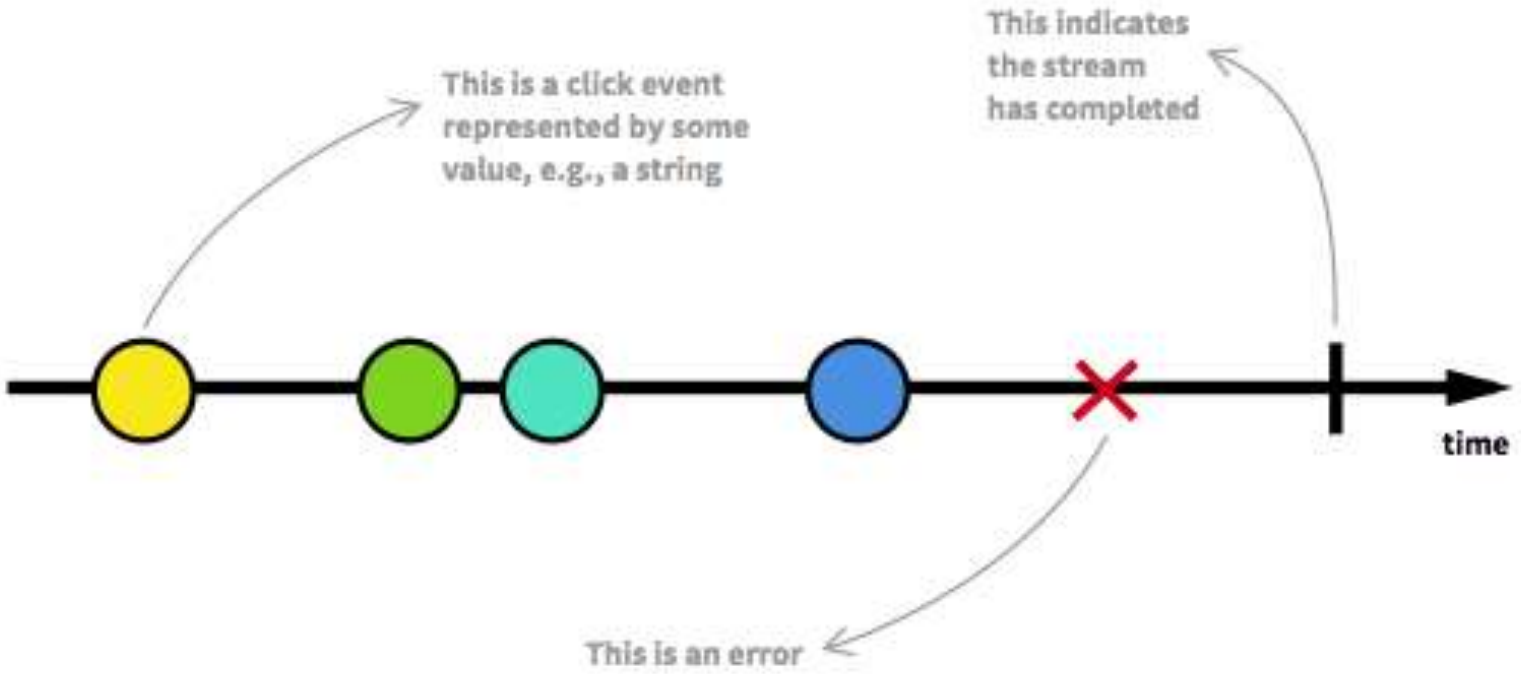
PETROL

Roche

ING

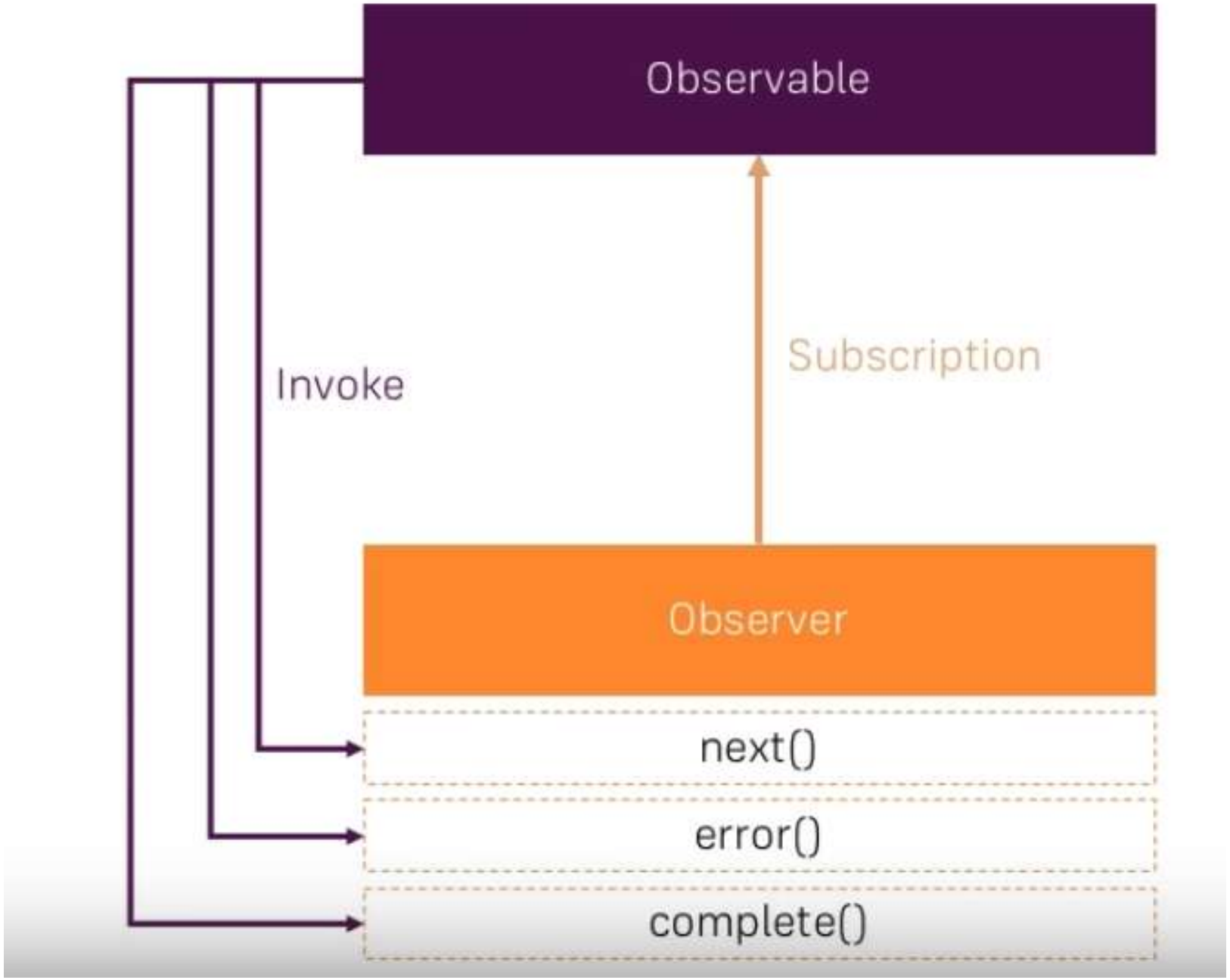

A STREAM?

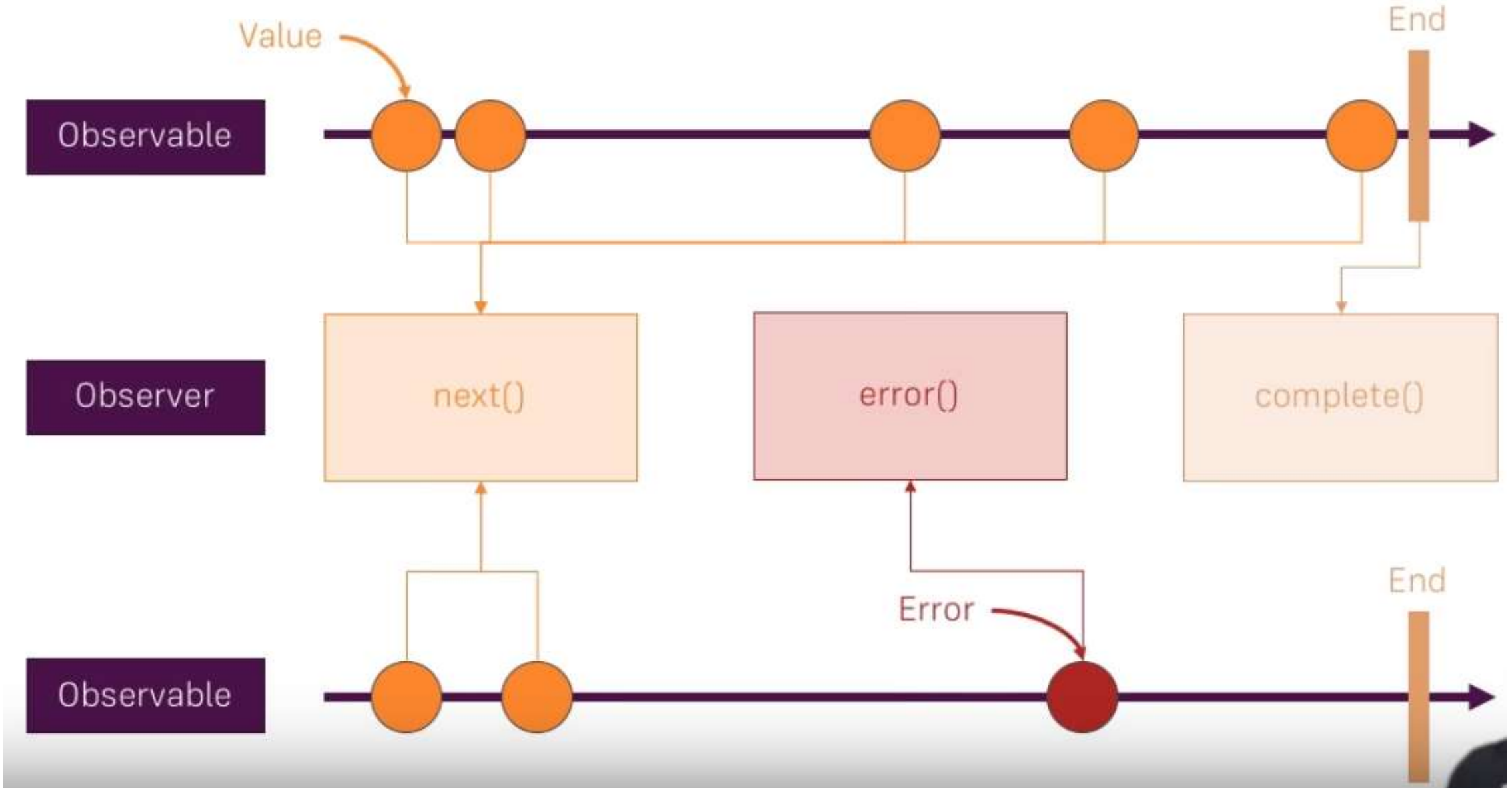
- A sequence of ongoing events ordered in time
- Emits a value, error and complete signal



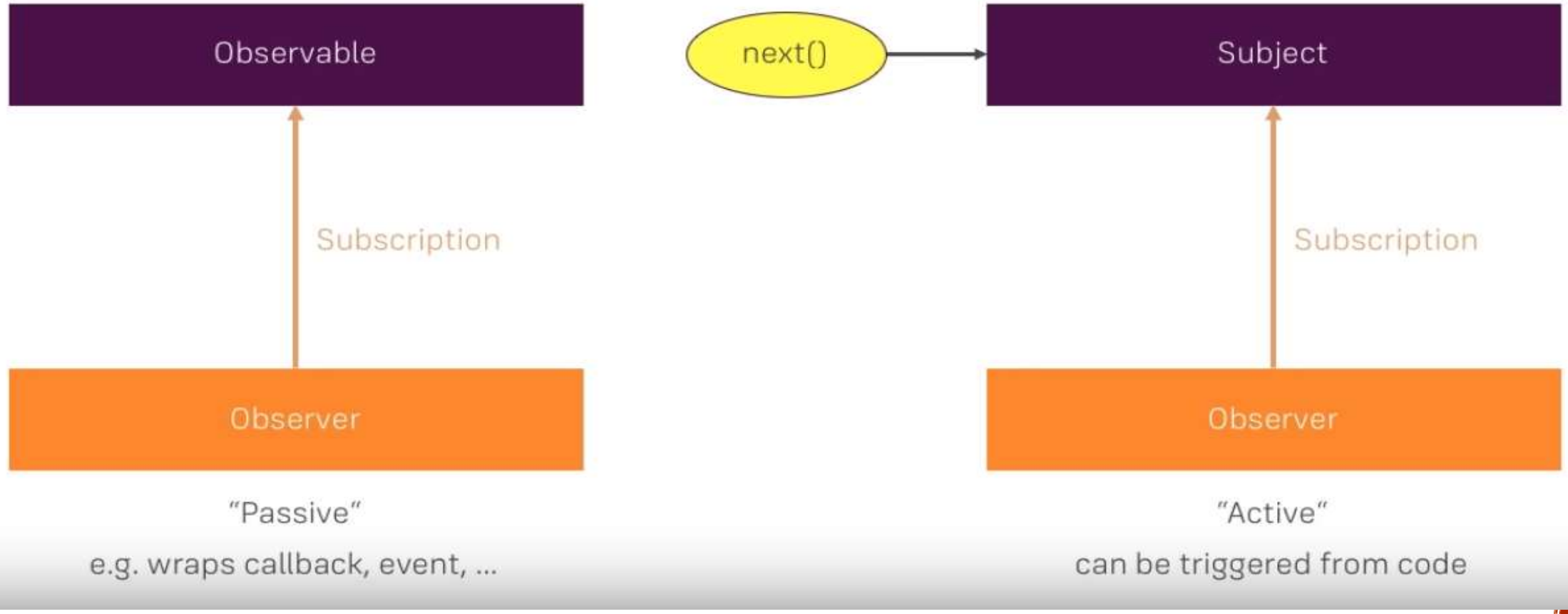
OBSERVABLES

- Observables are used to watch these streams and emit functions when a value, error or completed signal is returned
- **Observables** can be subscribed to by an **observer**
- Observables will constantly watch streams and will update accordingly
- We can interact with data streams as any regular array

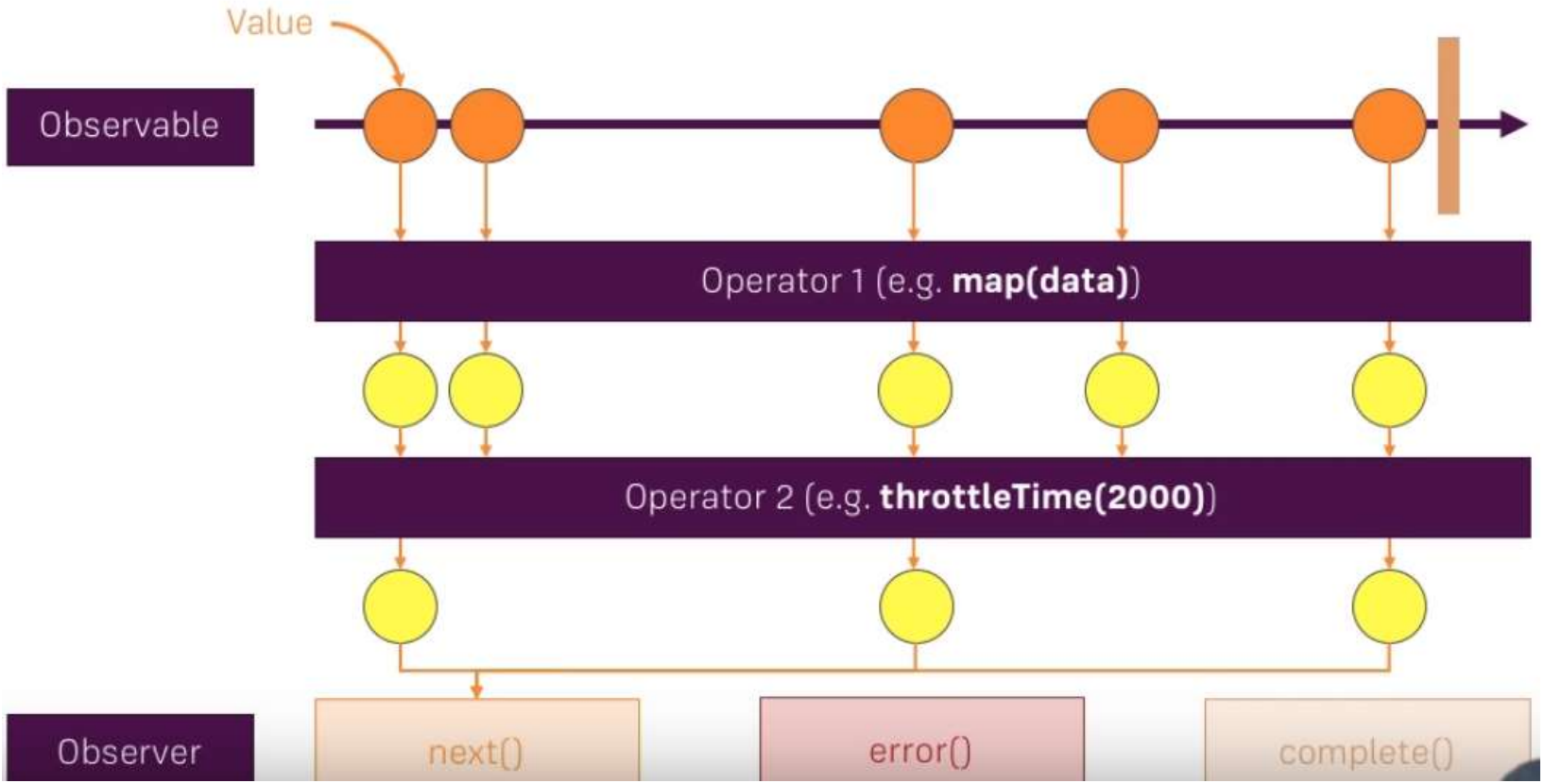

Observable
Invoke
Subscription
Observer
next()
error()
complete()


Value
End
Observable
Observer
next()
error()
complete()
Error
End
Observable

SUBJECT


Observable
next()
Subject
Subscription
Subscription
Observer
Observer
"Passive"
e.g. wraps callback, event, ...
"Active"
can be triggered from code

OPERATORS


Value
Observable
Operator 1 (e.g. map(data))
Operator 2 (e.g. throttleTime(2000))
Observer
next()
error()
complete()

CONCLUSION

- Observables are a POWERFUL abstraction
- Require a bit of mental rewiring
- The RX API is HUGE, take baby steps