Reactive Programming

wizeline.com | nicole.terc@wizeline.com

WIZELINE®

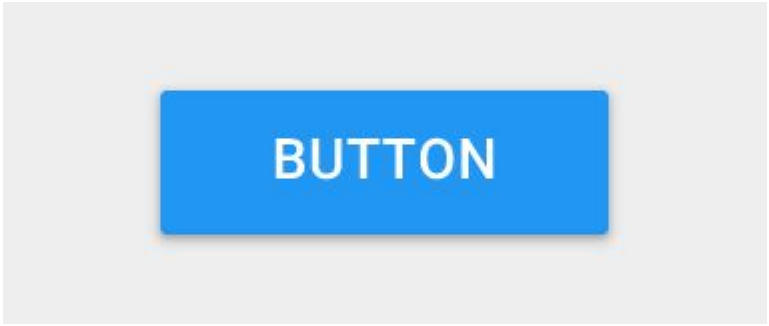
Reactive Programming

What is Reactive Programming?

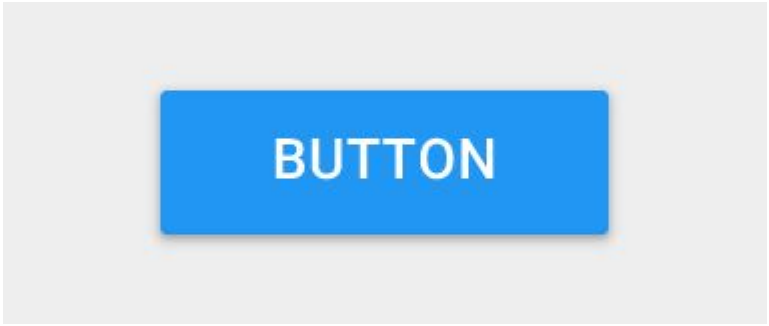
“Reactive programming is **programming** with asynchronous data **streams**.”

- André Staltz

What is a stream?

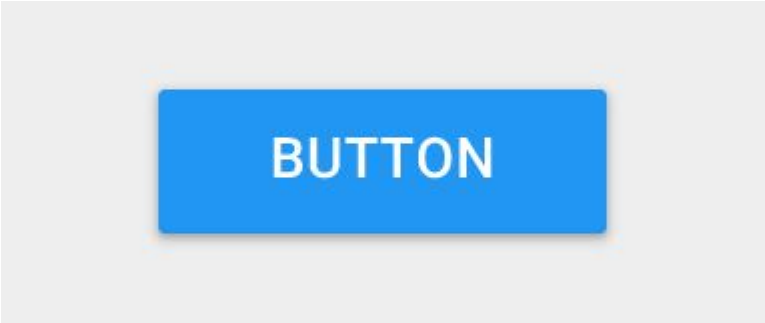


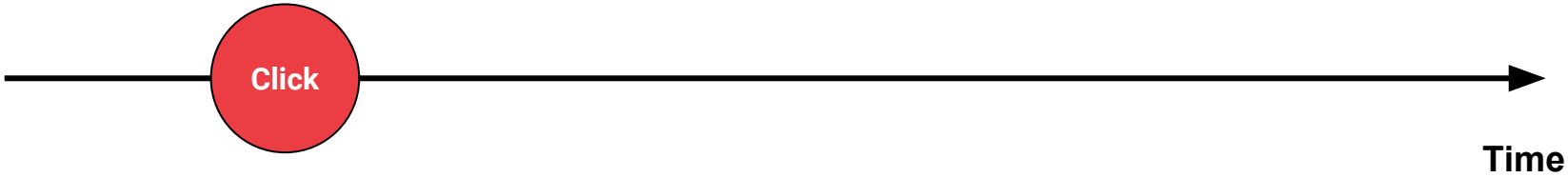
What is a stream?



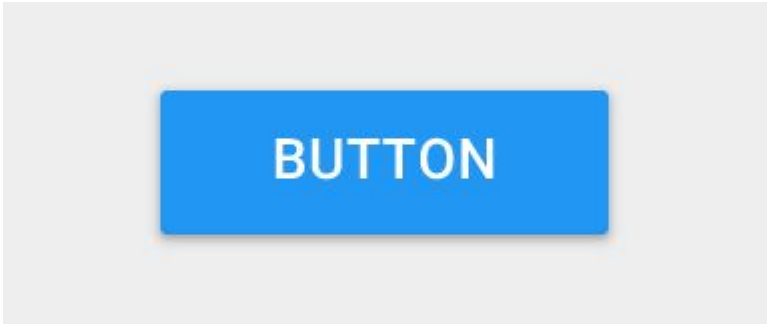
Click

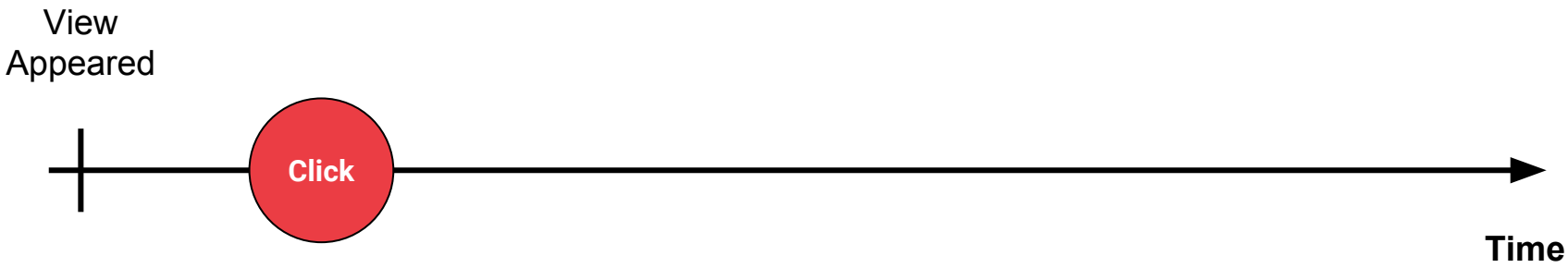
What is a stream?



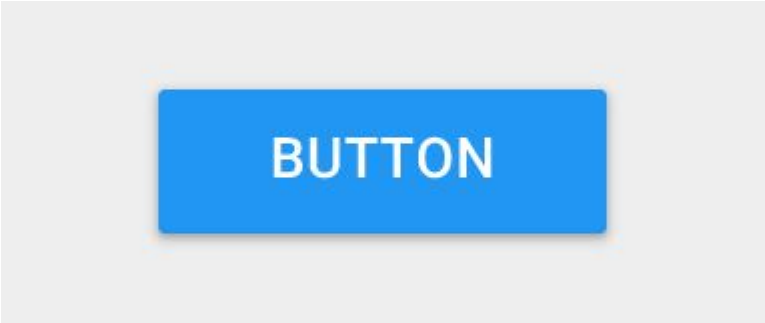


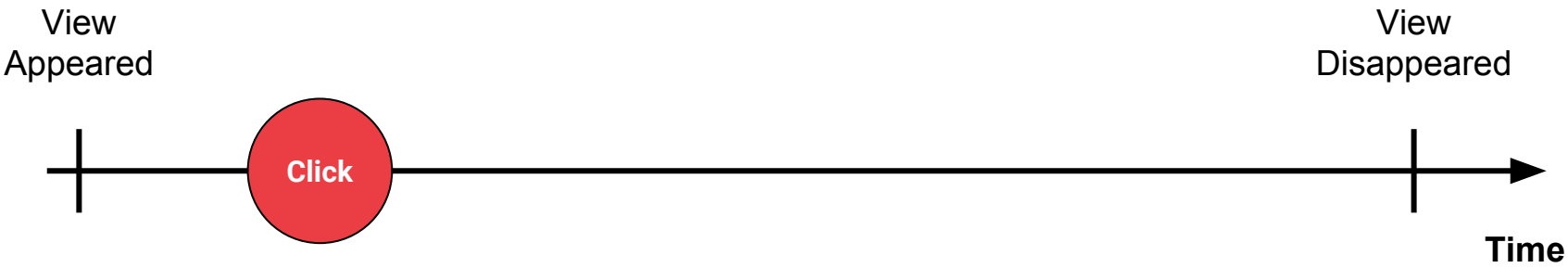
What is a stream?



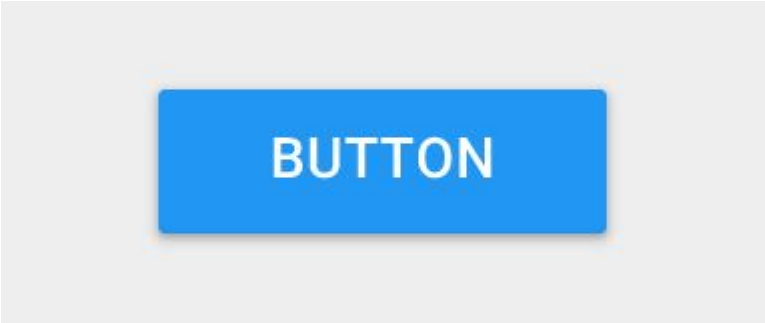


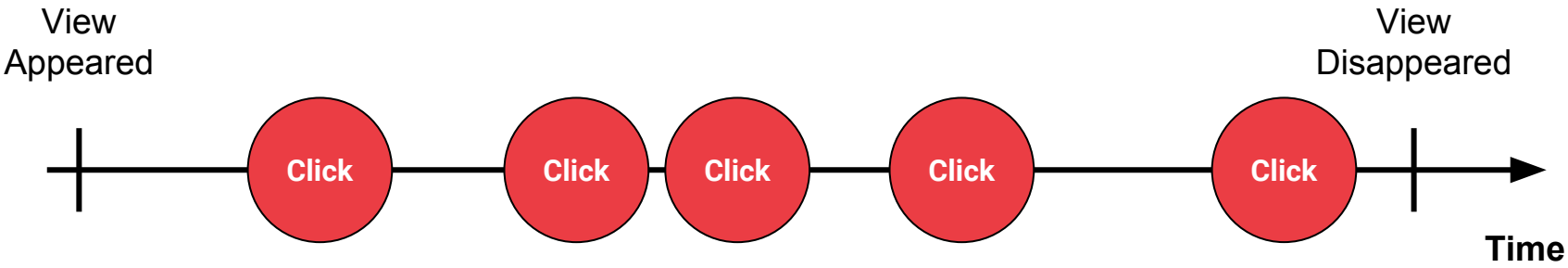
What is a stream?



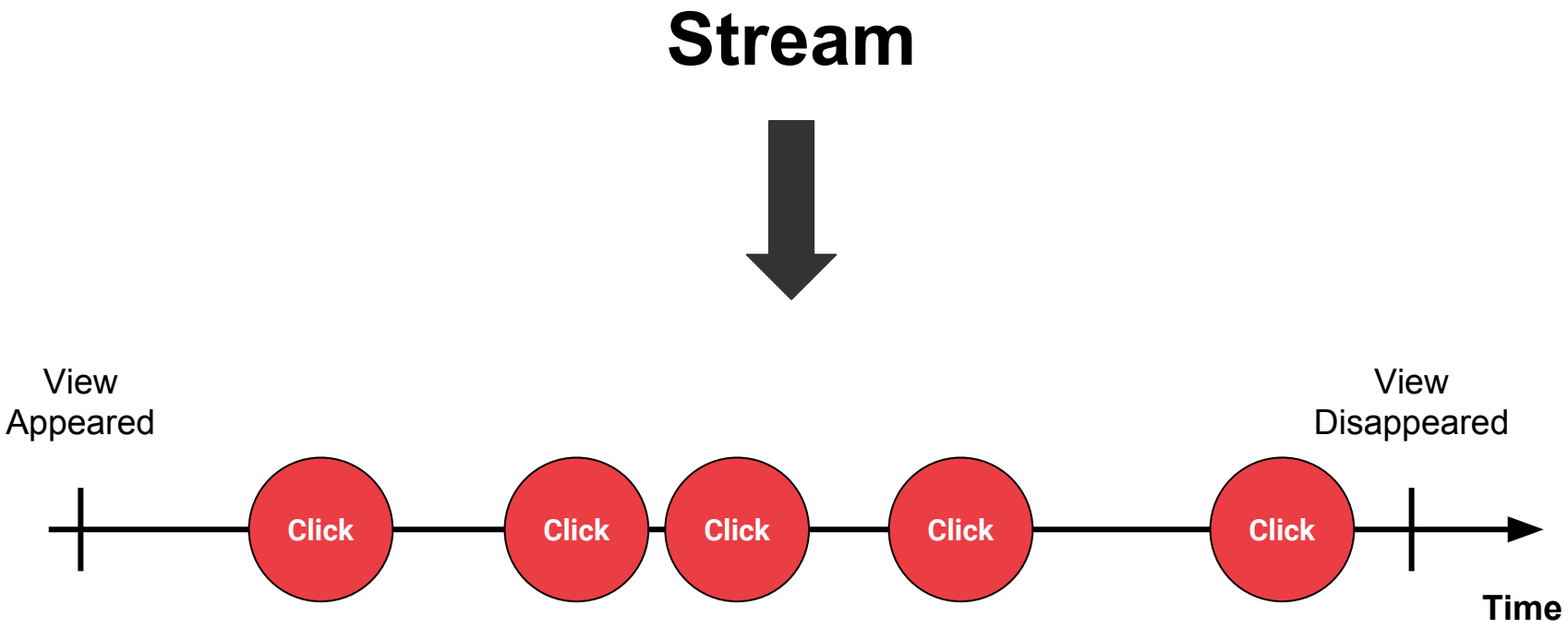


What is a stream?

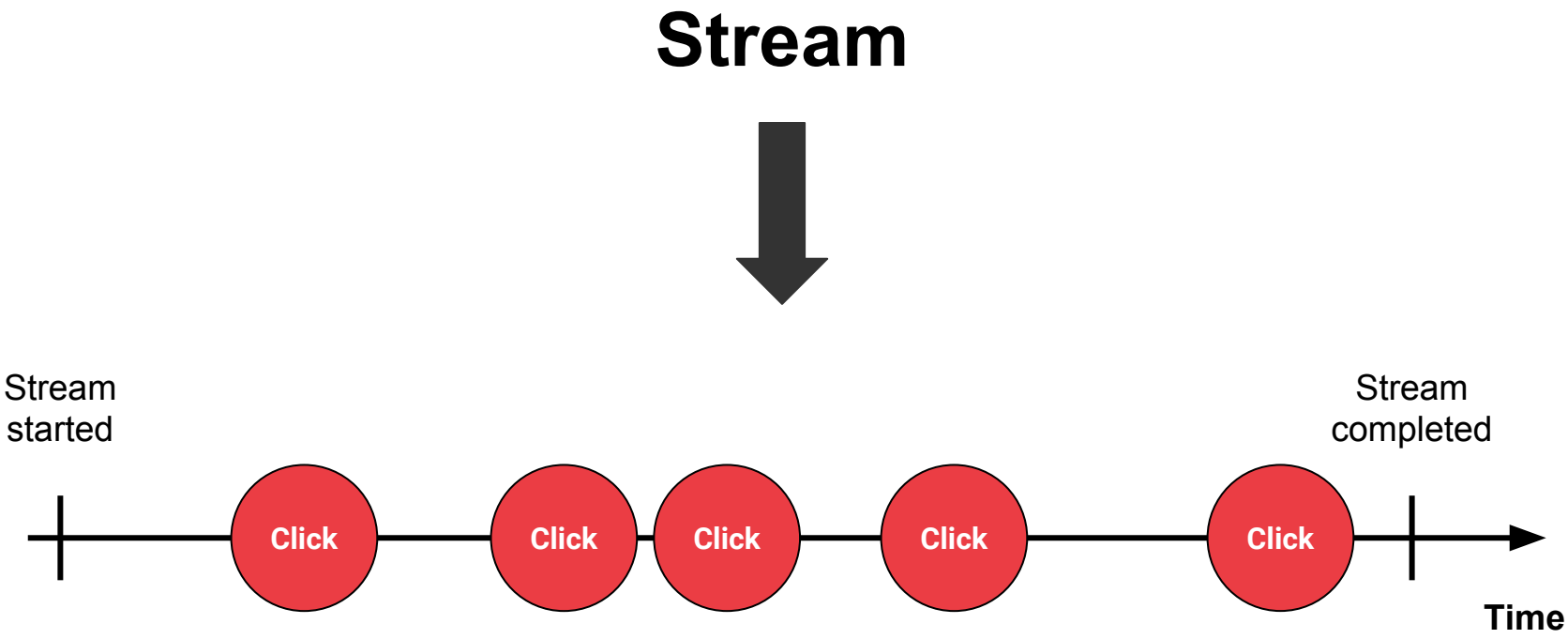




What is a stream?

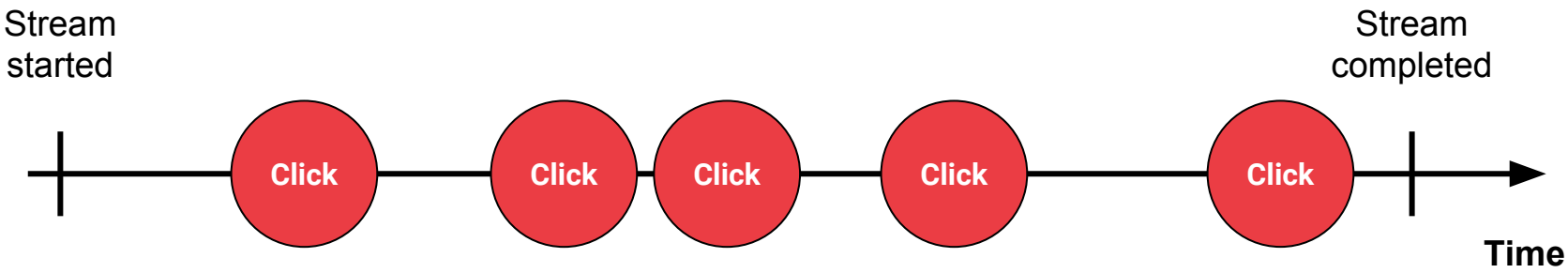


What is a stream?



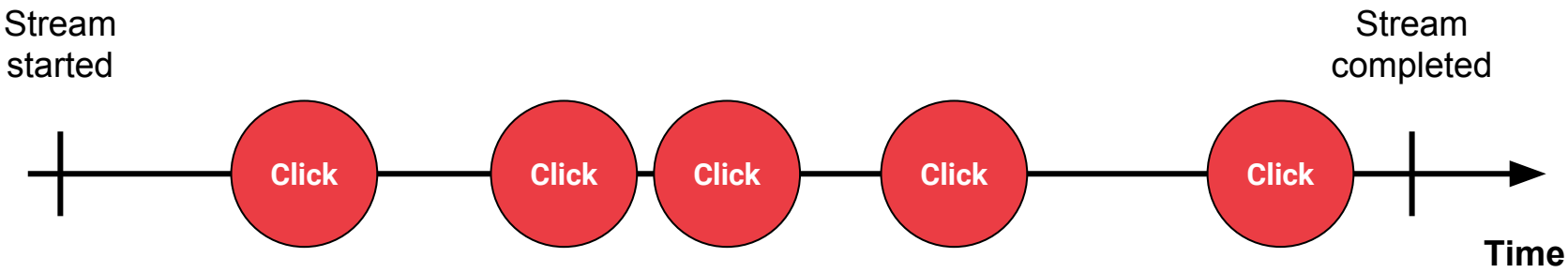
What is a stream?

- A stream is a **sequence** of ongoing **events** ordered **in time**.



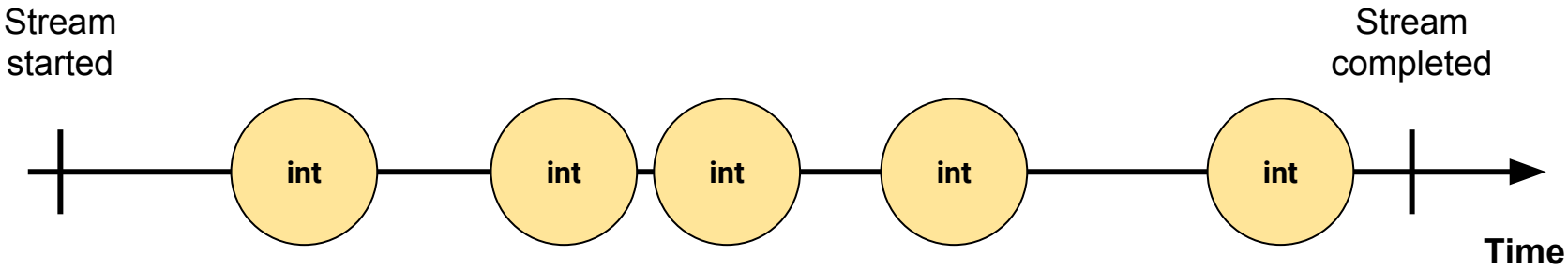
What is a stream?

- A stream is a **sequence** of ongoing **events** ordered **in time**.
- **Everything** can be streamed.



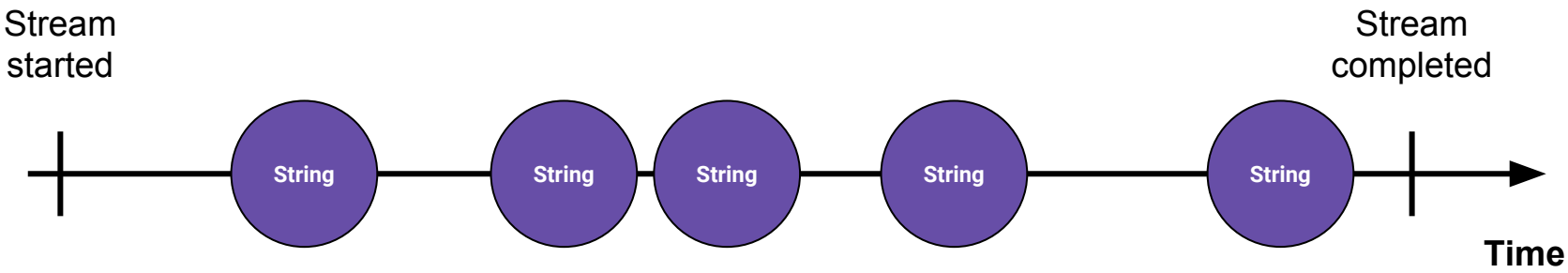
What is a stream?

- A stream is a **sequence** of ongoing **events** ordered **in time**.
- **Everything** can be streamed.



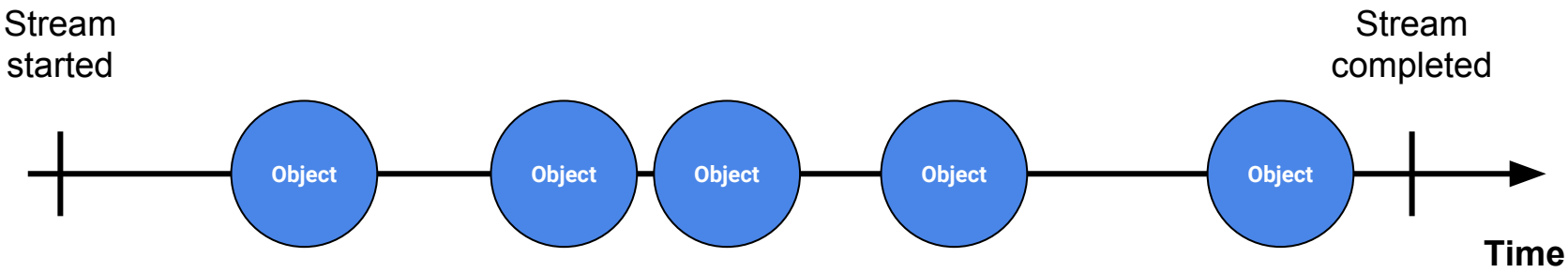
What is a stream?

- A stream is a **sequence** of ongoing **events** ordered **in time**.
- **Everything** can be streamed.



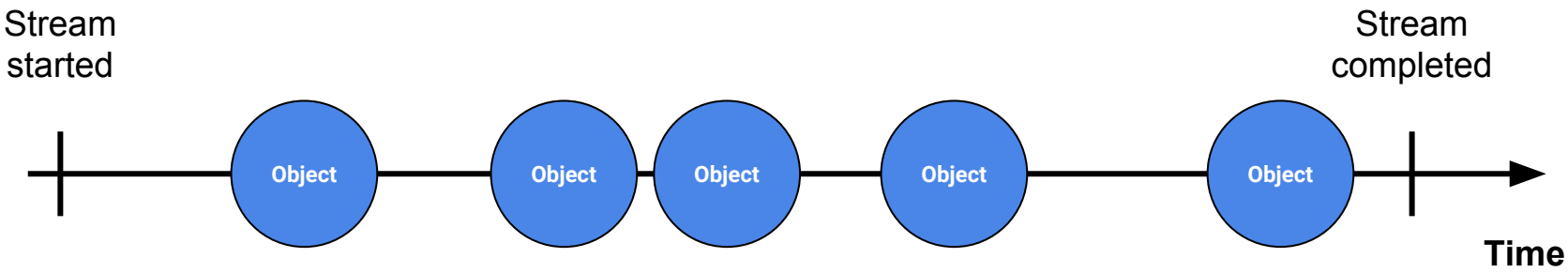
What is a stream?

- A stream is a **sequence** of ongoing **events** ordered **in time**.
- **Everything** can be streamed.



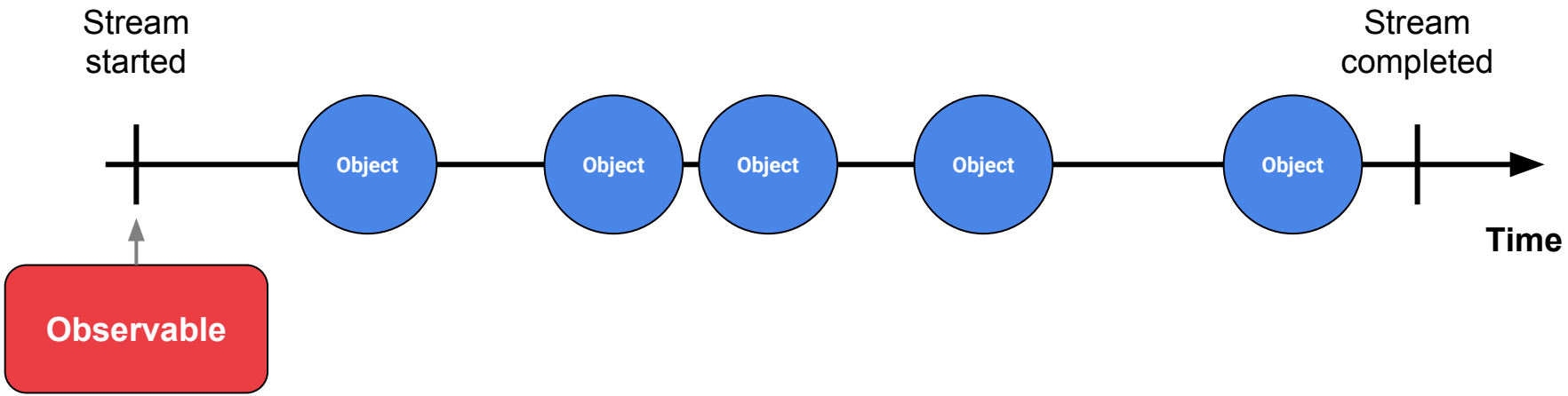
What is a stream?

- A stream is a **sequence** of ongoing **events** ordered **in time**.
- **Everything** can be streamed.
- You need something to **emit the data**.

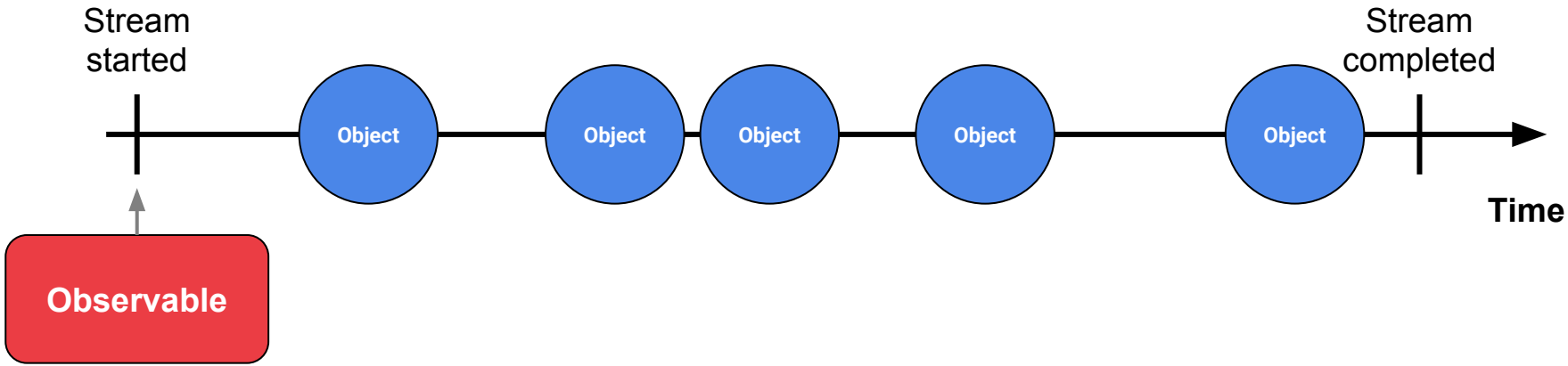


What is a stream?

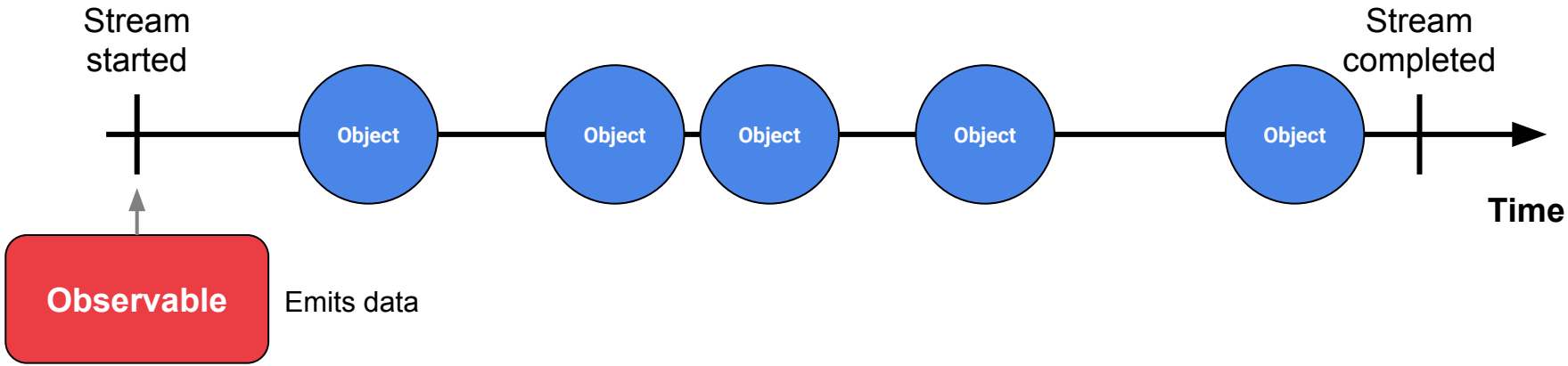
- A stream is a **sequence** of ongoing **events** ordered **in time**.
- **Everything** can be streamed.
- You need something to **emit the data**.



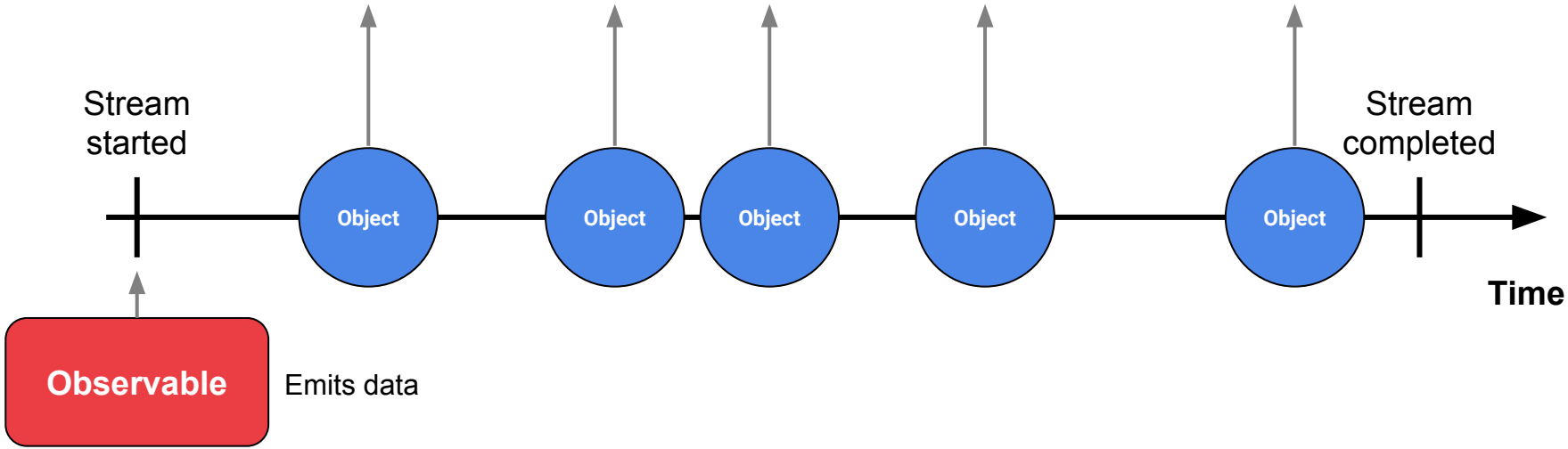
Now what?


Stream
started
Object
Object
Object
Object
Object
Stream
completed
Time
Observable

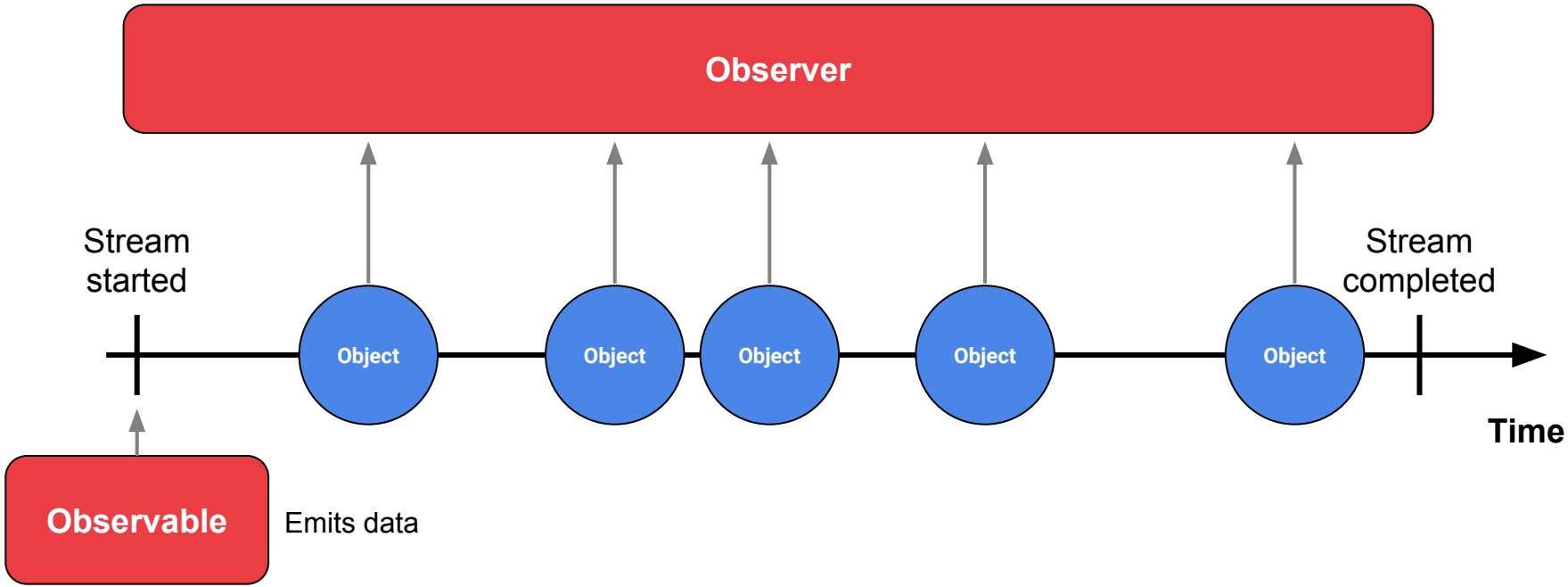
Now what?


Stream started
Object
Object
Object
Object
Object
Stream completed
Time
Observable
Emits data

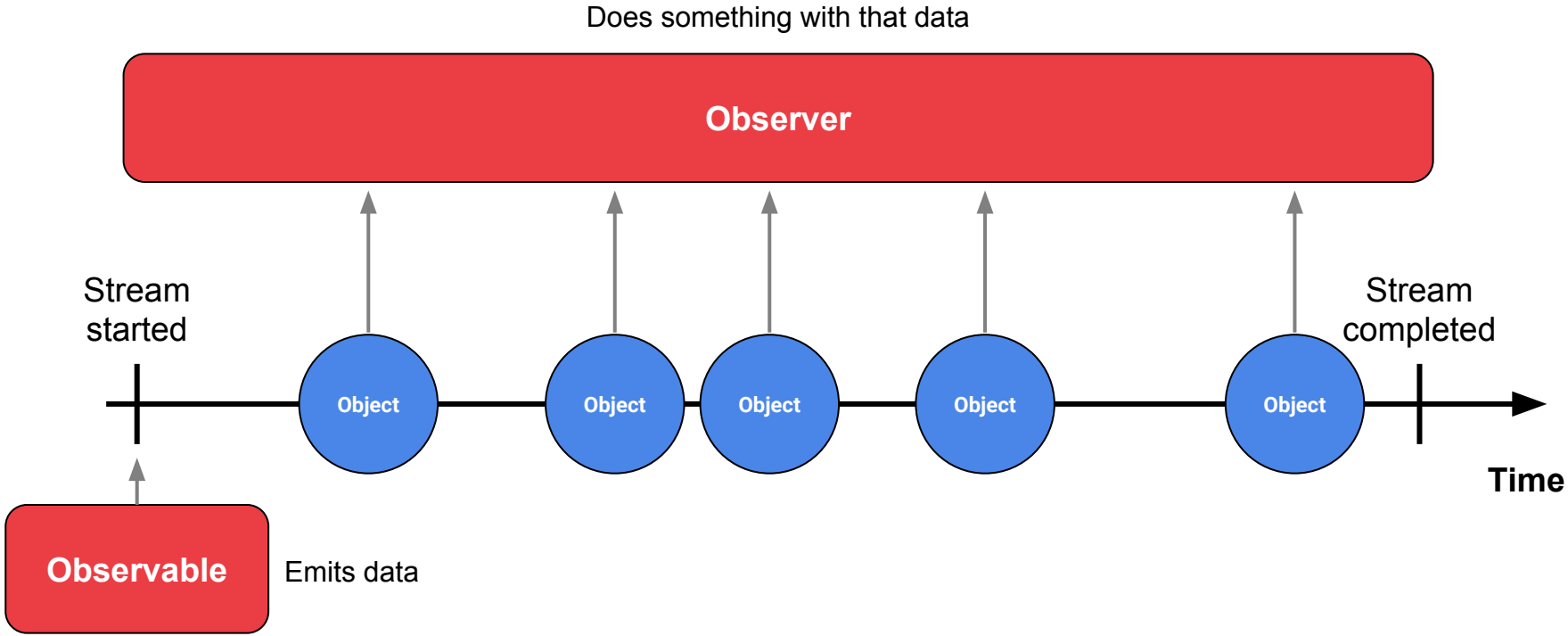
Now what?


Stream started
Object
Object
Object
Object
Object
Stream completed
Time
Observable
Emits data

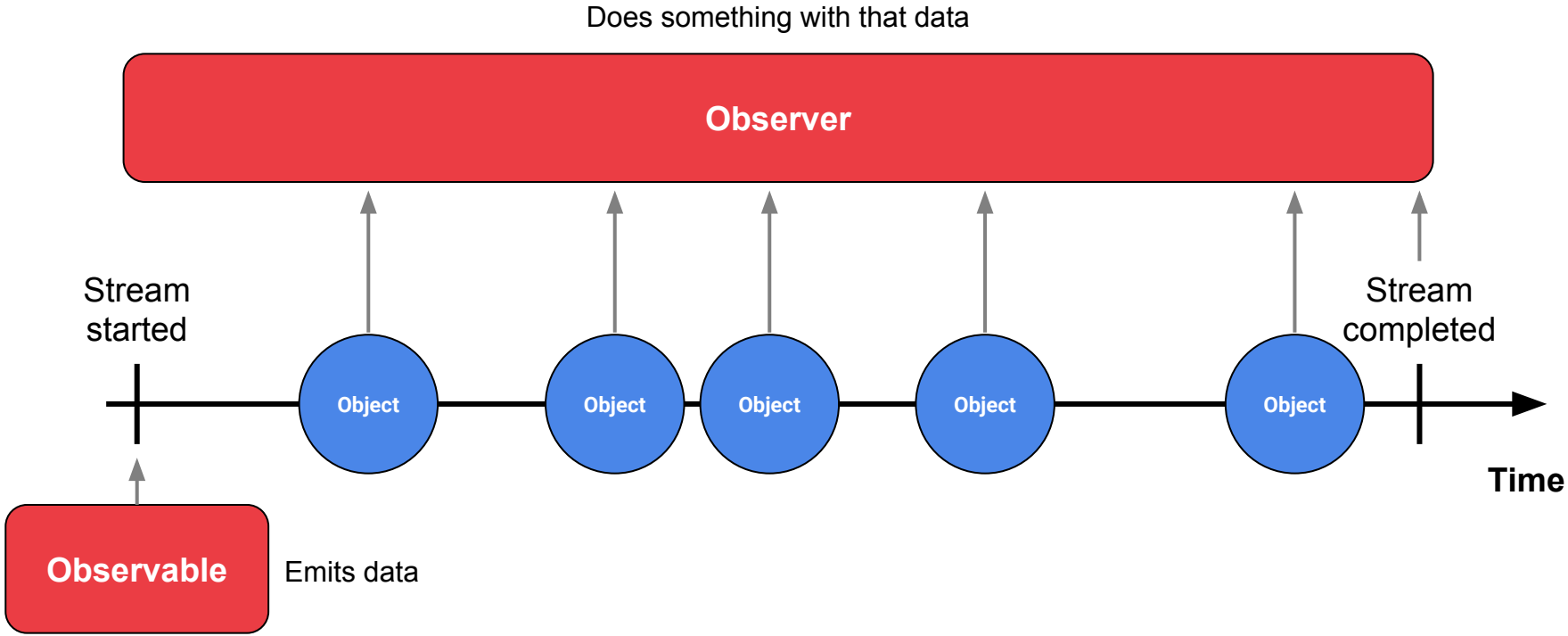
Now what?


Observer
Stream started
Stream completed
Object
Object
Object
Object
Object
Time
Observable
Emits data

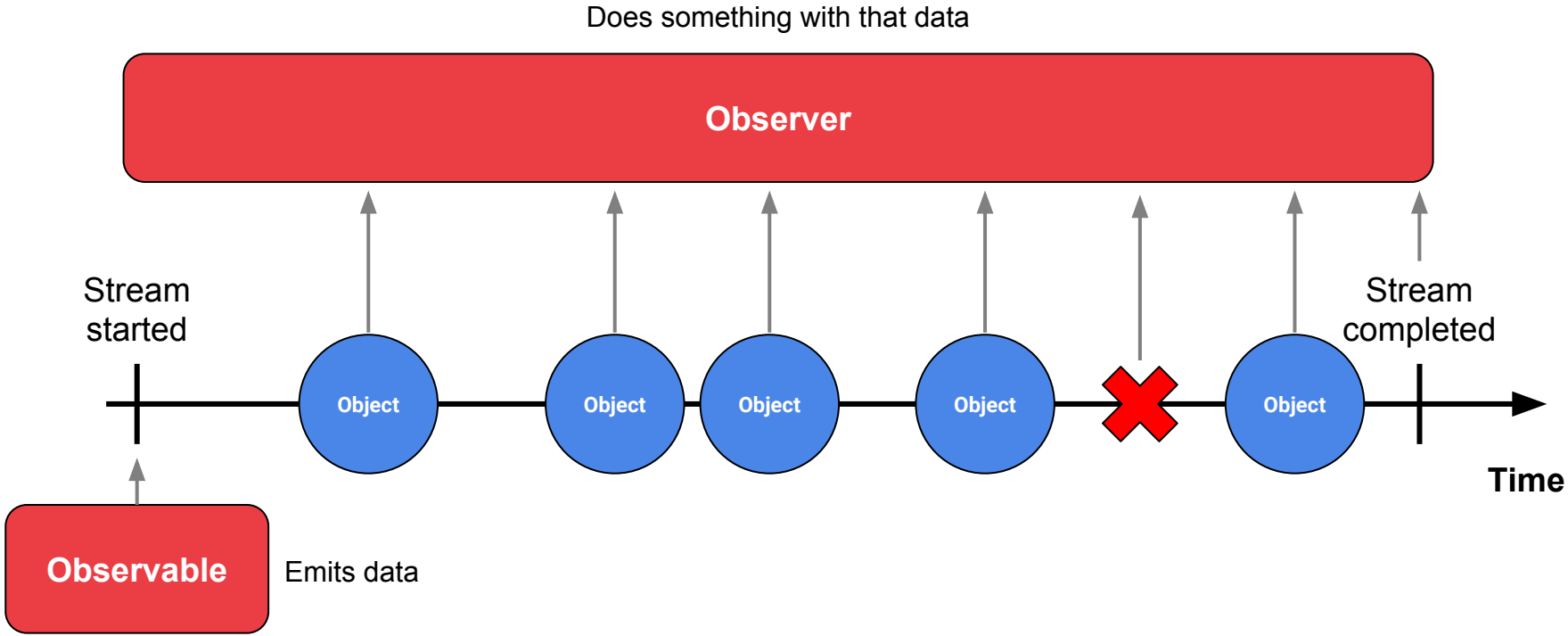
Now what?


Does something with that data
Observer
Stream started
Stream completed
Object
Object
Object
Object
Object
Time
Observable
Emits data

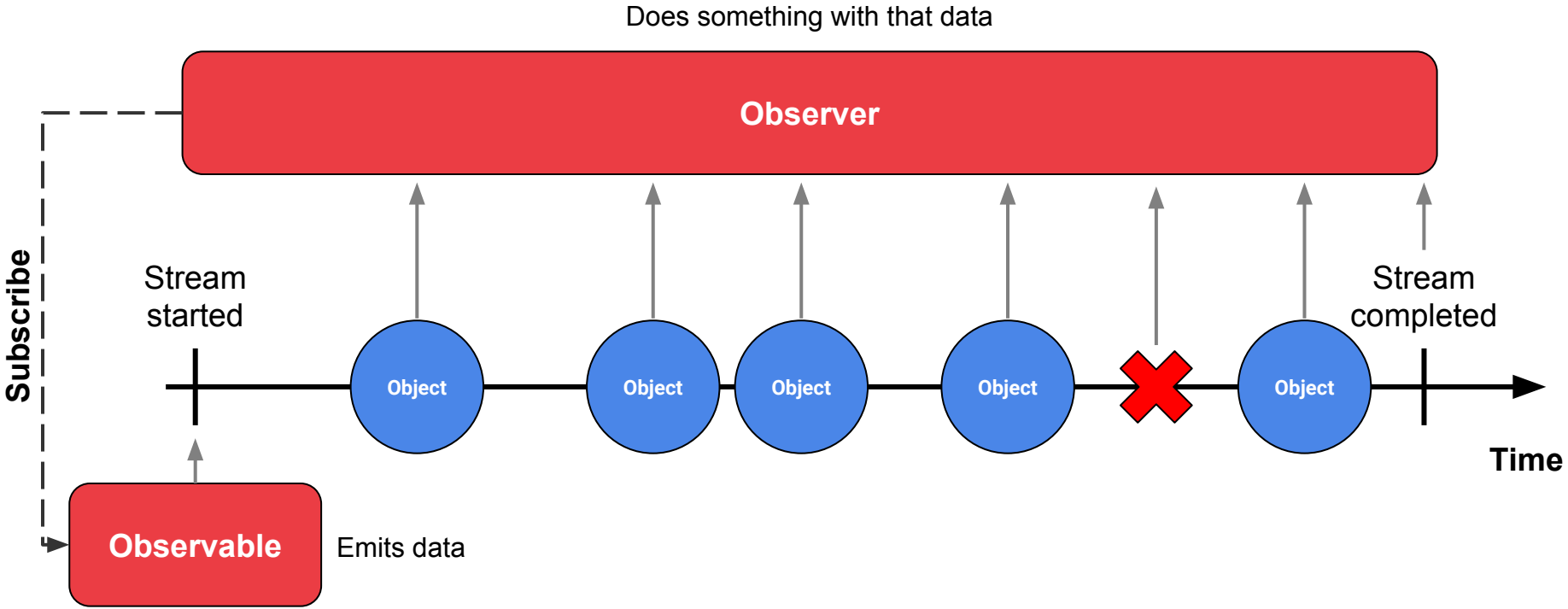
Now what?


Does something with that data
Observer
Stream started
Stream completed
Object
Object
Object
Object
Object
Time
Observable
Emits data

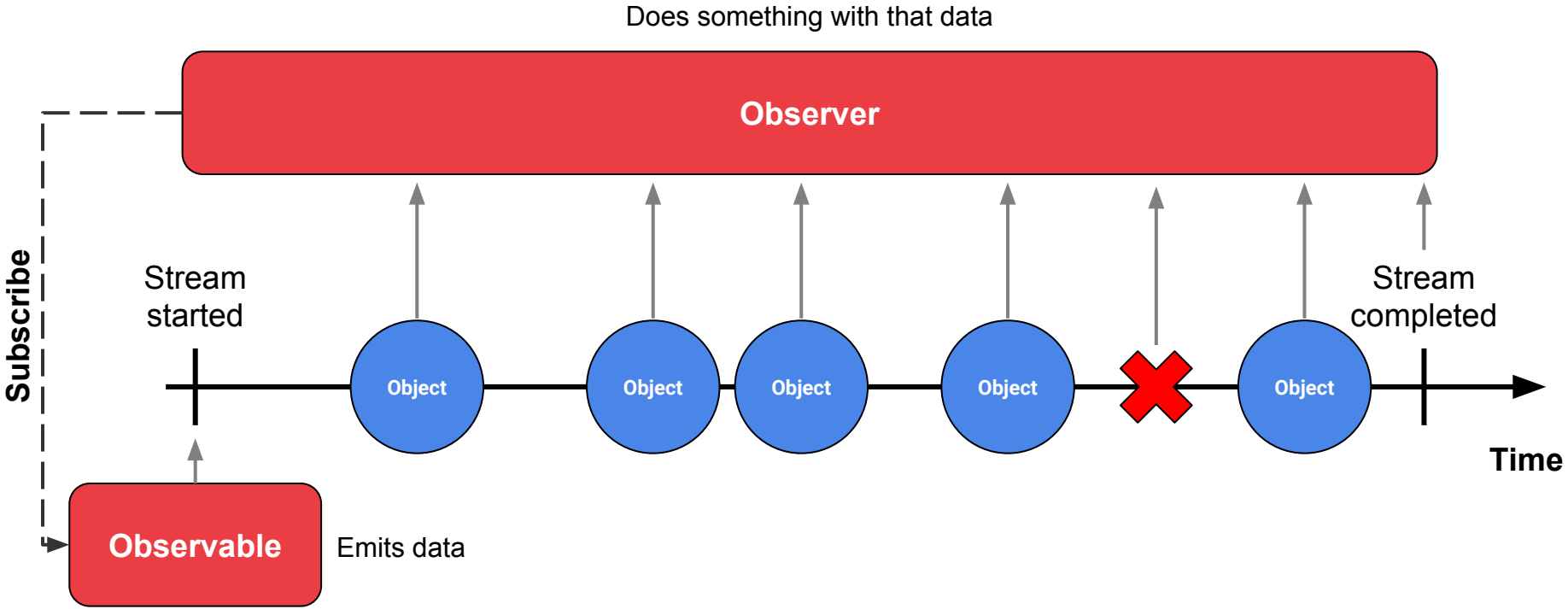
Now what?


Does something with that data
Observer
Stream started
Object
Object
Object
Object
Object
Stream completed
Time
Observable
Emits data

Now what?


Does something with that data
Observer
Subscribe
Stream started
Object
Object
Object
Object
Object
Stream completed
Time
Observable
Emits data

Observable Pattern



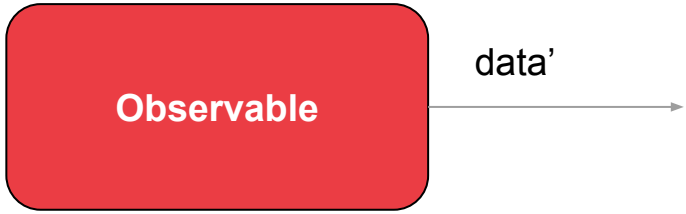
Reactive Programming

ReactiveX & RxJava

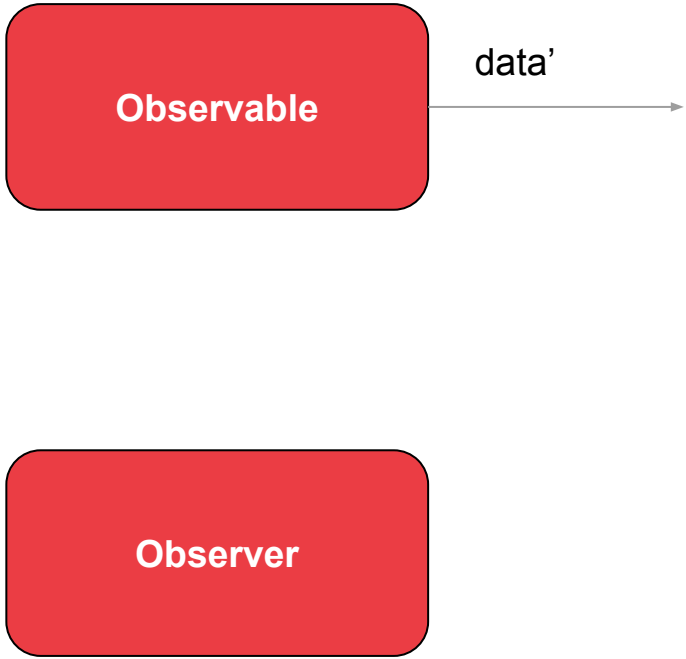
ReactiveX

ReactiveX is a library for composing **asynchronous** and **event-based** programs by using **observable** sequences.

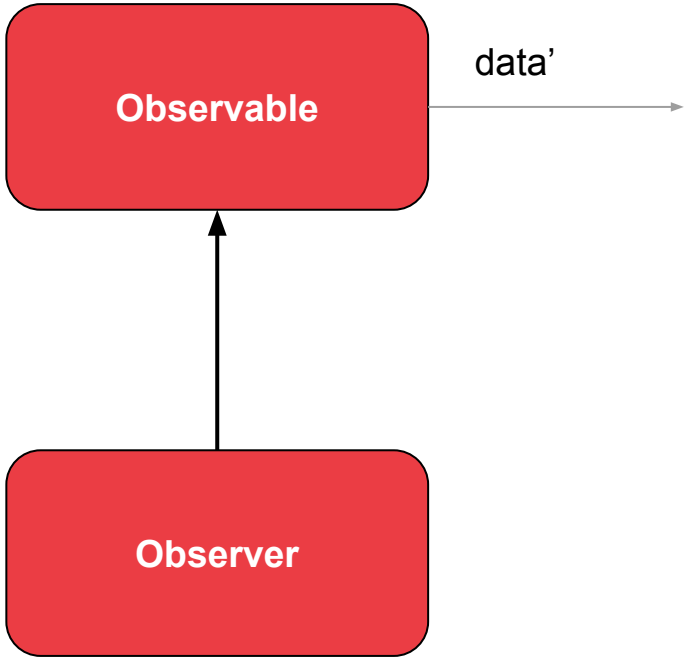
ReactiveX Components



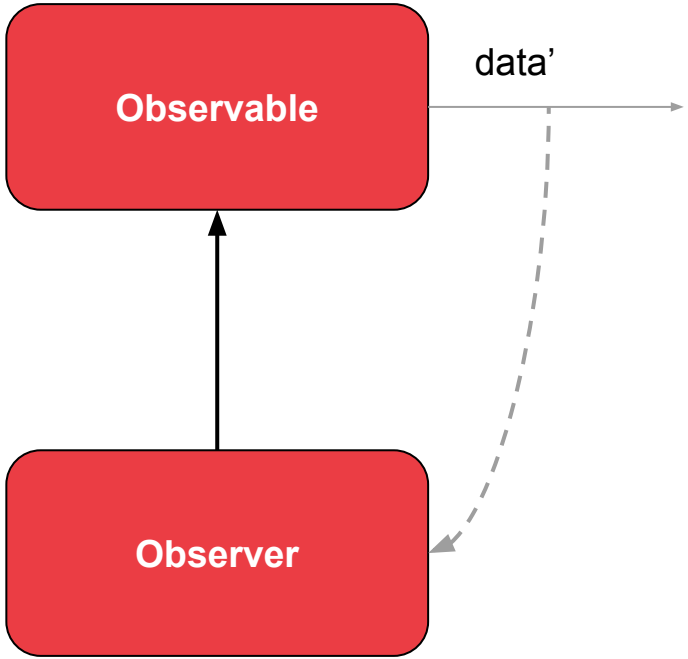
ReactiveX Components



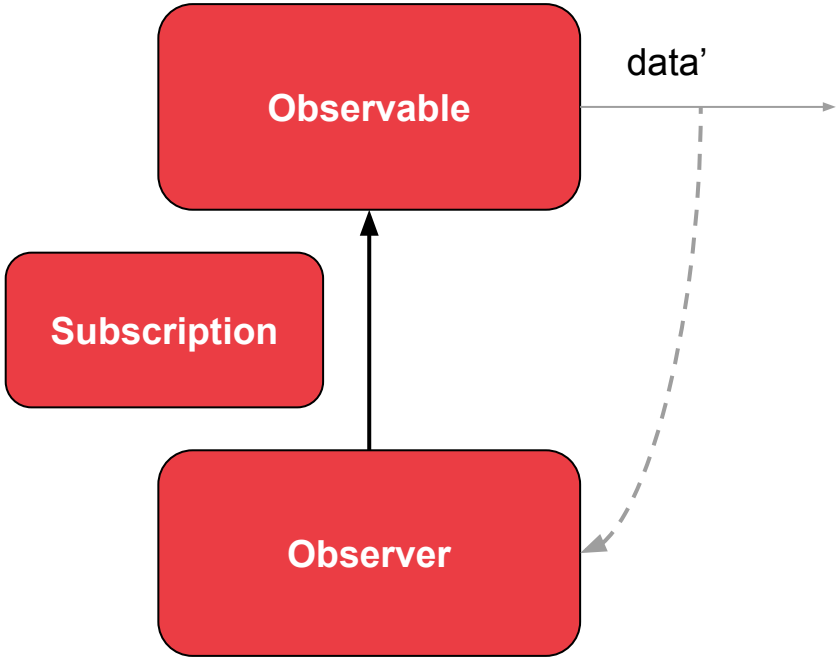
ReactiveX Components



ReactiveX Components



ReactiveX Components



ReactiveX Components

- **Observables:** Emits data/items.
- **Observers:** Watches the Observable and reacts to its data emissions.
  - onError() - Observable encountered an error.
  - onNext() - Observable sent an item.
  - onComplete() - Observable sent the last item and there were no errors.
- **Subscribers/Disposables:** Link between Observables and Observers.

RxJava Android example

```
ArrayList<String> getSuggestionsData() {
        ArrayList<String> suggestionsData = new ArrayList<>();
        for (int i = 0; i < 10; i++) {
                suggestionsData.add("Suggestion #" + i);
         }
        return suggestionsData;
}
```

RxJava Android example

```
ArrayList<String> getSuggestionsData() {
        ArrayList<String> suggestionsData = new ArrayList<>();
        for (int i = 0; i < 10; i++) {
                suggestionsData.add("Suggestion #" + i);
         }
        return suggestionsData;
}
```

RxJava Android example

```
Observable.fromIterable(getSuggestionsData()) // Converts an ArrayList to an Observable
        .subscribe( //Create Subscription
                new Observer<String>() {
                    @Override public void onSubscribe(Disposable d) {
                        //Do something when process starts
                        //Ej: show loading message
                    }
                    @Override public void onNext(String s) {
                        //New item was received, do something with it
                        //Ej: add item to a list adapter
                    }
                    @Override public void onError(Throwable e) {
                        //An error was encountered, deal with it
                        //Ej: show an error message
                    }
                    @Override public void onComplete() {
                        //All items were sent, clean up
                        //Ej: hide loading message, clear variables
                    }
                });
```

RxJava Android example

```
Observable.fromIterable(getSuggestionsData()) // Converts an ArrayList to an Observable
        .subscribe( //Create Subscription
                new Observer<String>() {
                    @Override public void onSubscribe(Disposable d) {
                        //Do something when process starts
                        //Ej: show loading message
                    }
                    @Override public void onNext(String s) {
                        //New item was received, do something with it
                        //Ej: add item to a list adapter
                    }
                    @Override public void onError(Throwable e) {
                        //An error was encountered, deal with it
                        //Ej: show an error message
                    }
                    @Override public void onComplete() {
                        //All items were sent, clean up
                        //Ej: hide loading message, clear variables
                    }
                });
```

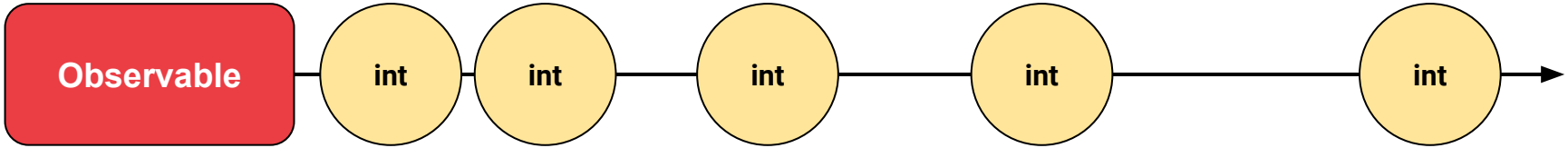
RxJava Android example

```
Observable.fromIterable(getSuggestionsData()) // Converts an ArrayList to an Observable
    .subscribe( //Create Subscription
        new Observer<String>() {
          @Override public void onSubscribe(Disposable d) {
            //Do something when process starts
            //Ej: show loading message
          }
          @Override public void onNext(String s) {
            //New item was received, do something with it
            //Ej: add item to a list adapter
          }
          @Override public void onError(Throwable e) {
            //An error was encountered, deal with it
            //Ej: show an error message
          }
          @Override public void onComplete() {
            //All items were sent, clean up
            //Ej: hide loading message, clear variables
          }
        });
```

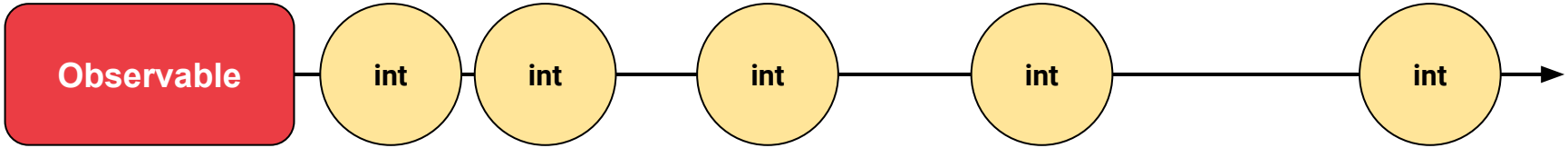
RxJava Android example

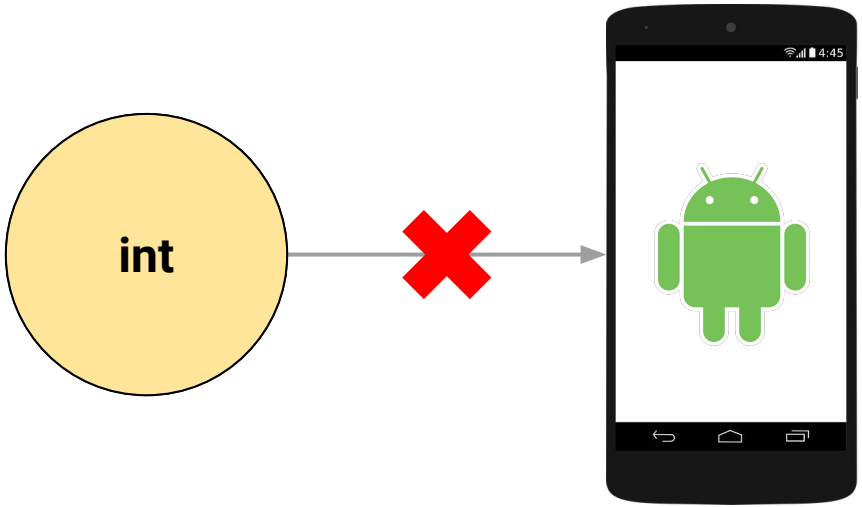
```
Observable.fromIterable(getSuggestionsData()) // Converts an ArrayList to an Observable
        .subscribe( //Create Subscription
                new Observer<String>() {
                    @Override public void onSubscribe(Disposable d) {
                        //Do something when process starts
                        //Ej: show loading message
                    }
                    @Override public void onNext(String s) {
                        //New item was received, do something with it
                        //Ej: add item to a list adapter
                    }
                    @Override public void onError(Throwable e) {
                        //An error was encountered, deal with it
                        //Ej: show an error message
                    }
                    @Override public void onComplete() {
                        //All items were sent, clean up
                        //Ej: hide loading message, clear variables
                    }
                });
```

ReactiveX Components

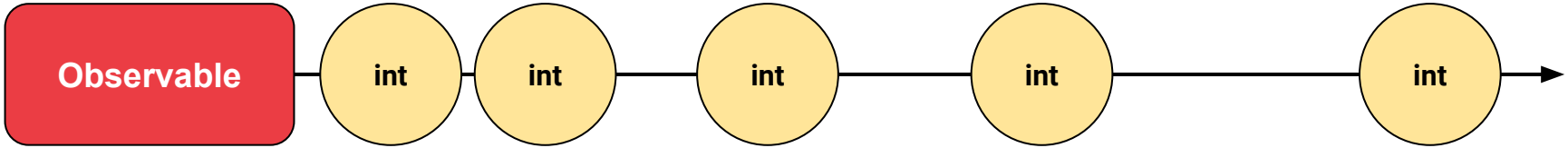


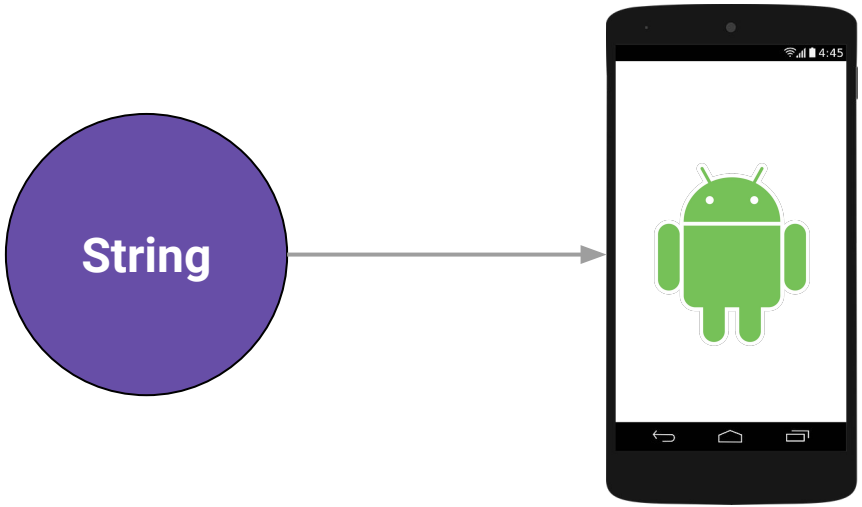
ReactiveX Components



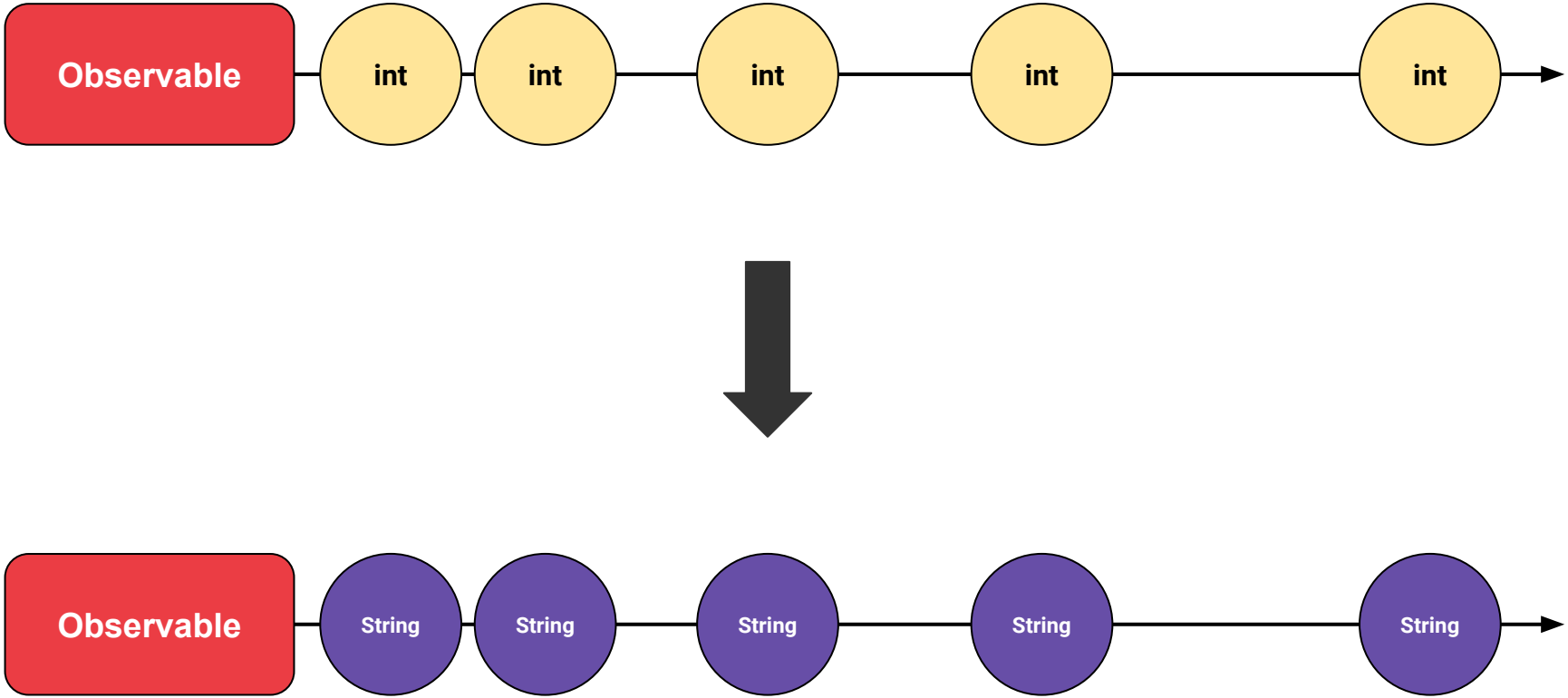


ReactiveX Components

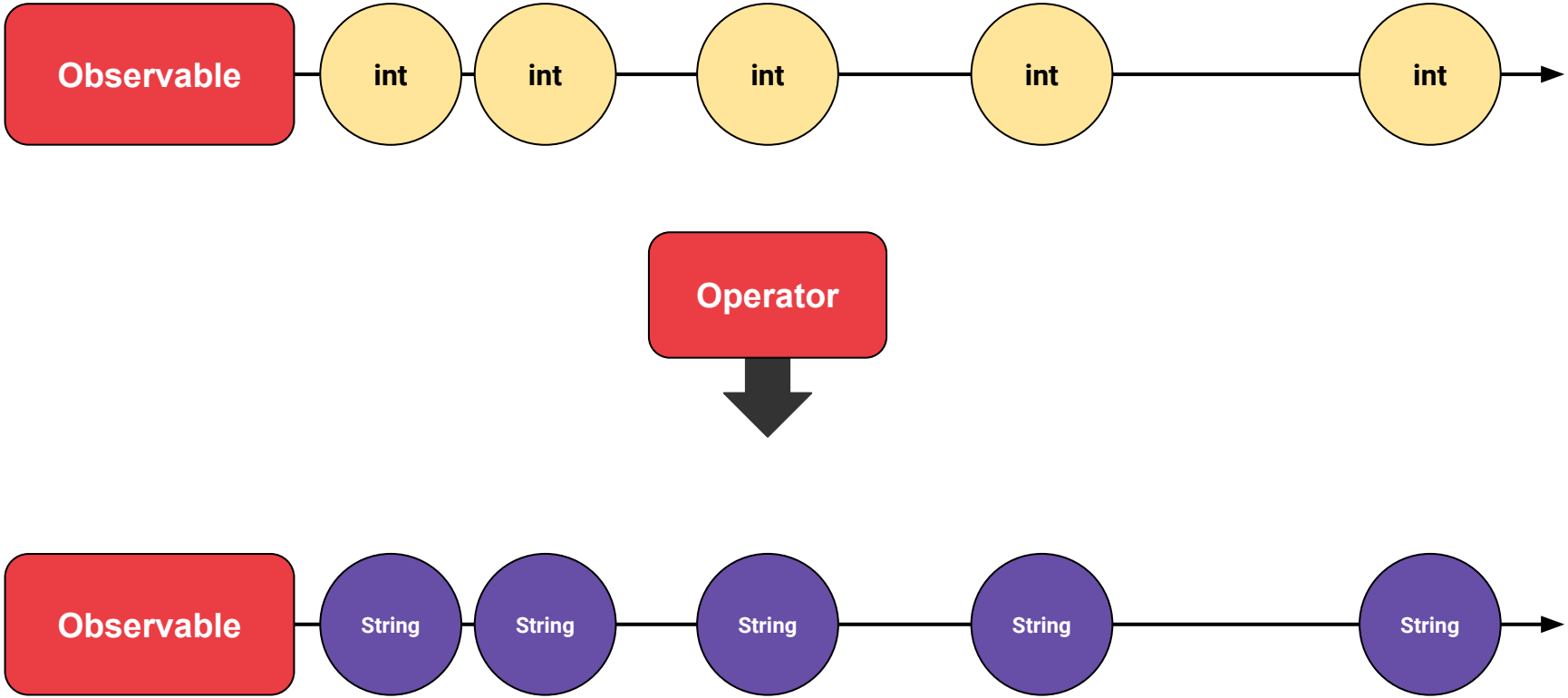




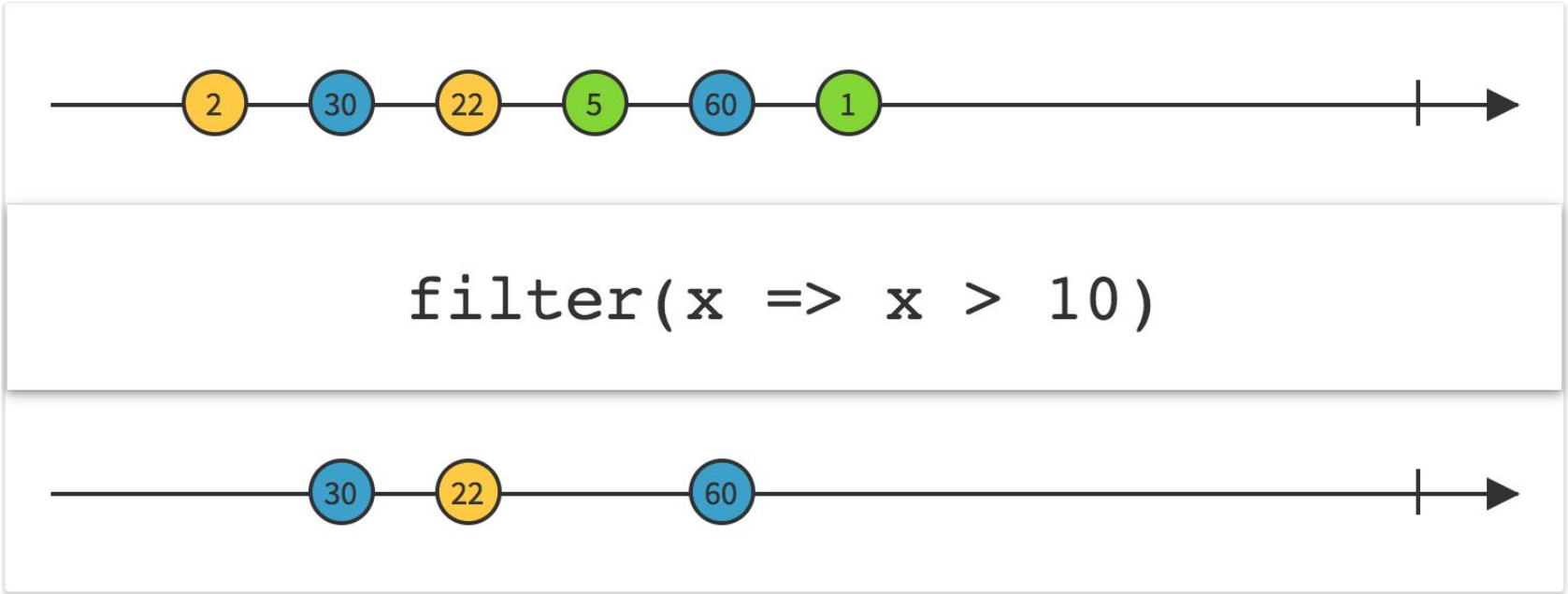
ReactiveX Components



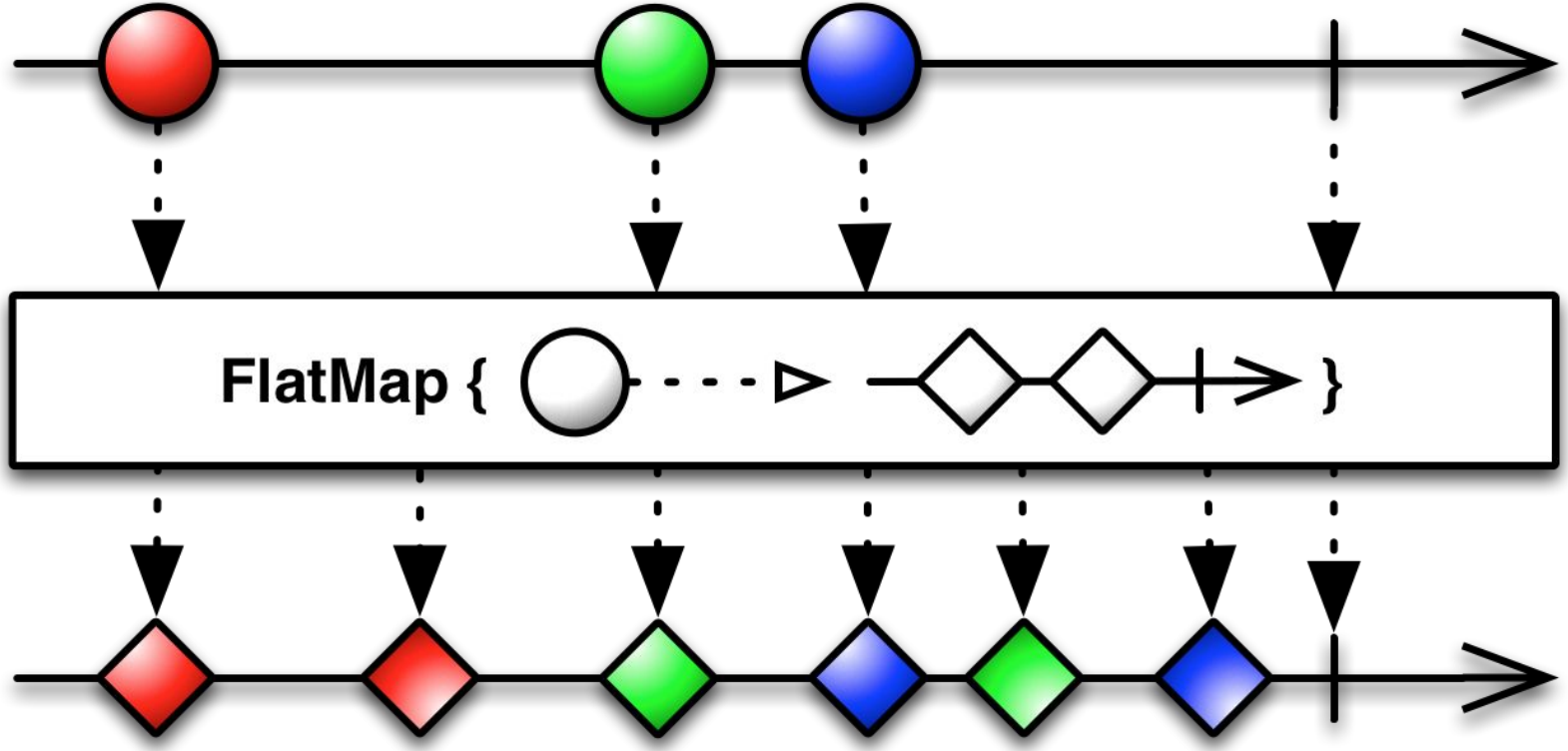
ReactiveX Components



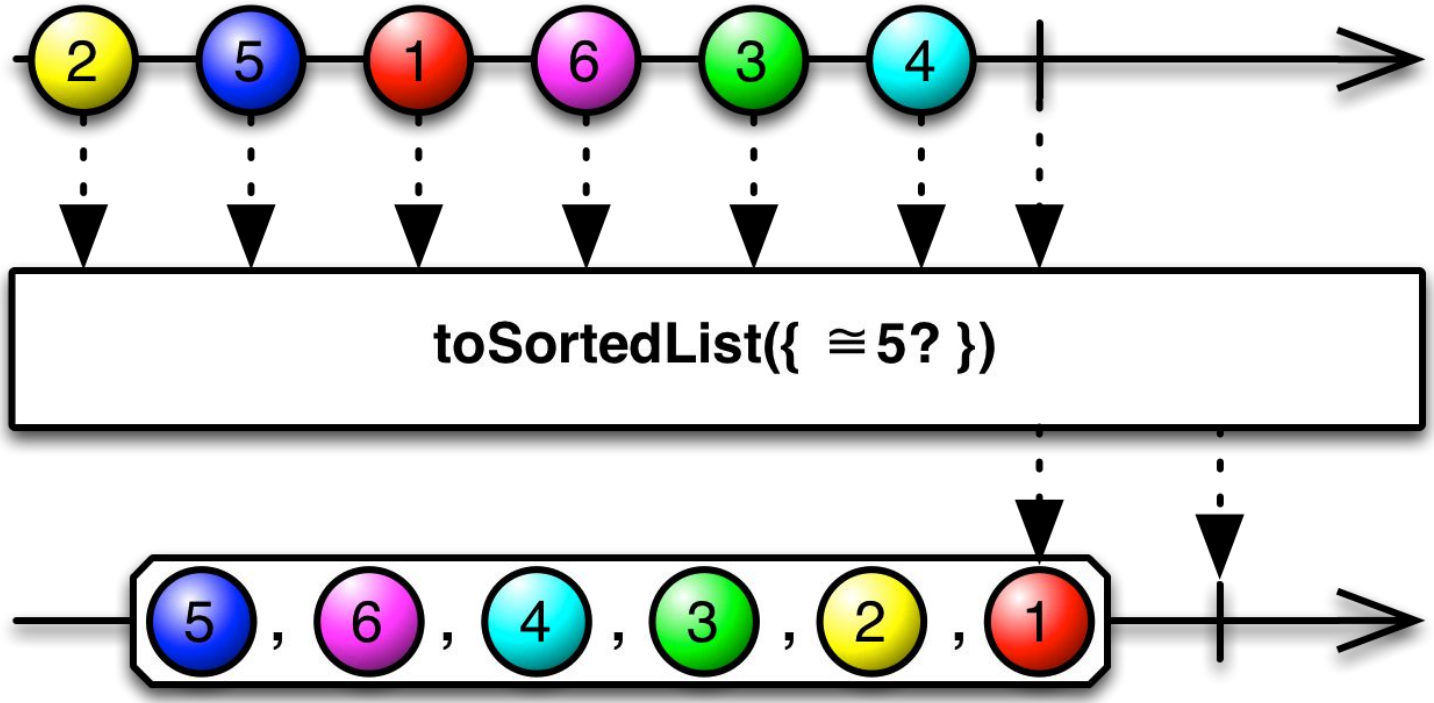
Operators



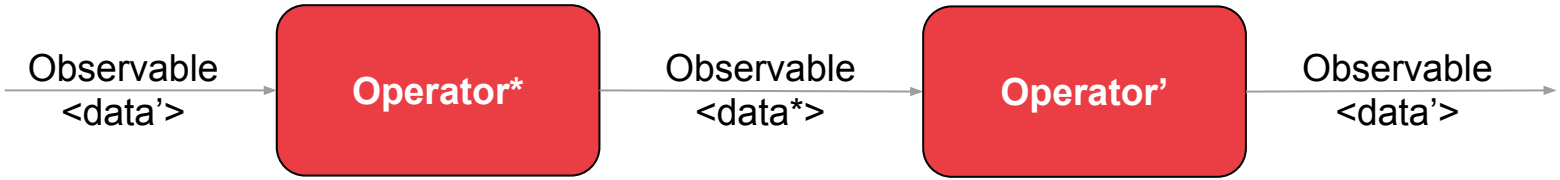
ReactiveX marble diagrams



ReactiveX marble diagrams



Operators


Observable
<data'>
Operator*
Observable
<data*>
Operator'
Observable
<data'>

Operators

- **fromIterator()** creates observable from iterable objects.
- **fromArray()** creates observable from array.
- **map()** transforms one item to another
- **flatMap()** transforms one item to several
- **filter()** filter out items
- **toSortedList()** combines data and returns a **Single**<List<data>>
- **toObservable()** various objects to observable.
- **debounce()** only pass items after x time without emitting anything
- **merge()** combine multiple Observables into one

And much more: http://reactivex.io/documentation/operators.html

Threading

- **subscribeOn()** -> in which thread are we going to subscriber

- **observeOn()** -> in which thread are we going to observe

Threading

- **subscribeOn()** -> in which thread are we going to subscriber
  - subscribeOn(**Schedulers.io()**)

- **observeOn()** -> in which thread are we going to observe
  - observeOn(**AndroidSchedulers.mainThread()**)


CODE TIME!

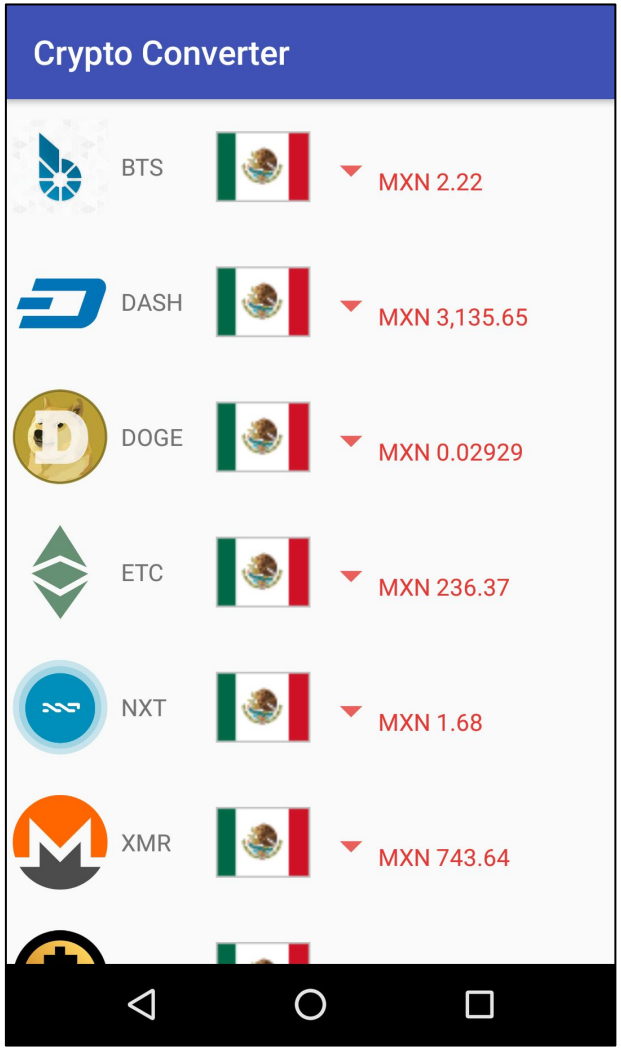
CryptoCurrency

Challenge 1

Filter results and show only currency with price **lower than 1000 mxn.**

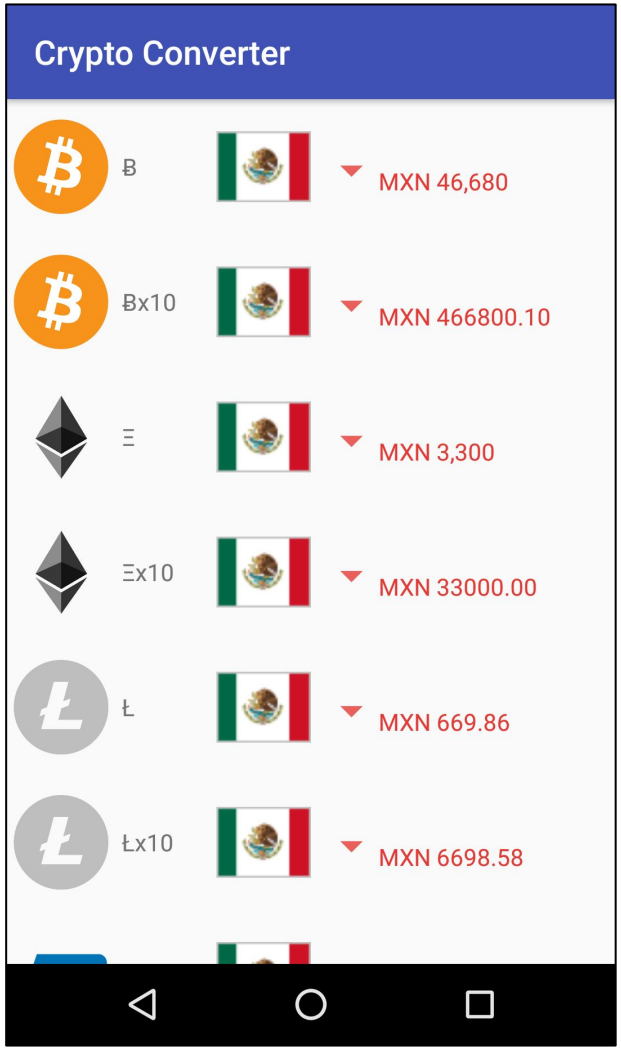
Challenge 2

Sort currencies alphabetically.



Challenge 3

For each currency, show it’s original value and one item with **value x 10.**



Why ReactiveX?

- Work with async data/event streams as easy as if them were arrays
- Compose your observables to fit your needs (operators, yai!)
- Easy threading

Want to learn more?

- **ReactiveX**
  - http://reactivex.io
- **RxJava**
  - https://github.com/ReactiveX/RxJava
- **RxAndroid**
  - https://github.com/ReactiveX/RxAndroid


THANK
YOU
WIZELINE®