Reactive Programming

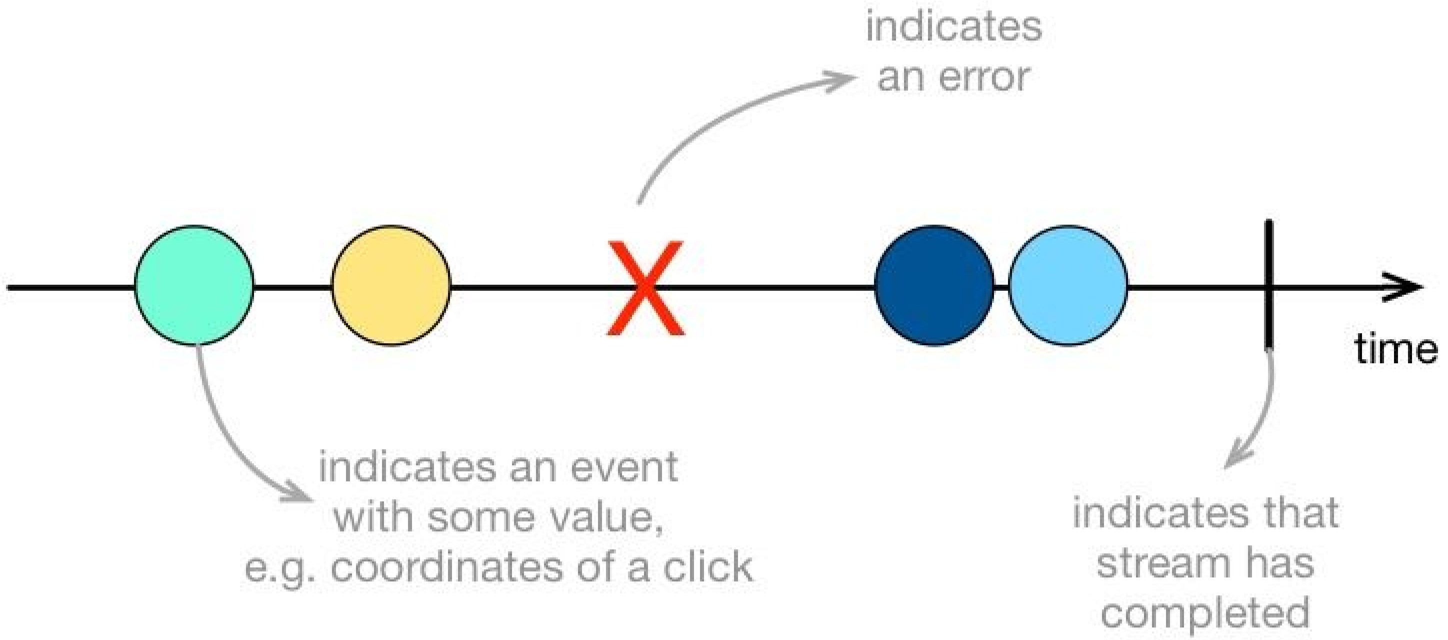
with Rx.js

Rx.Observable.prototype.flatMapLatest(selector, [thisArg])

Projects each element of an observable sequence into a new sequence of observable sequences by incorporating the element's index and then transforms an observable sequence of observable sequences into an observable sequence producing values only from the most recent observable sequence.

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

Observable

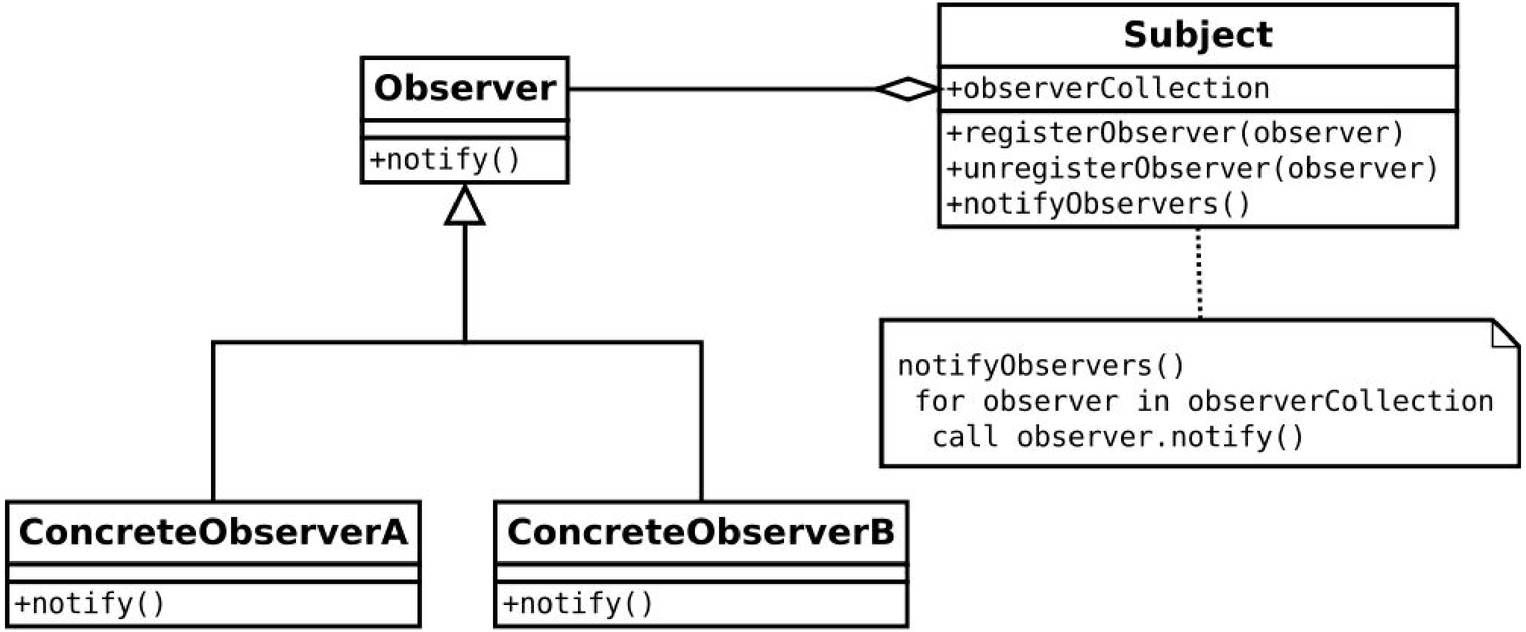

indicates an error
time
indicates an event with some value, e.g. coordinates of a click
indicates that stream has completed

A simple *Observable*

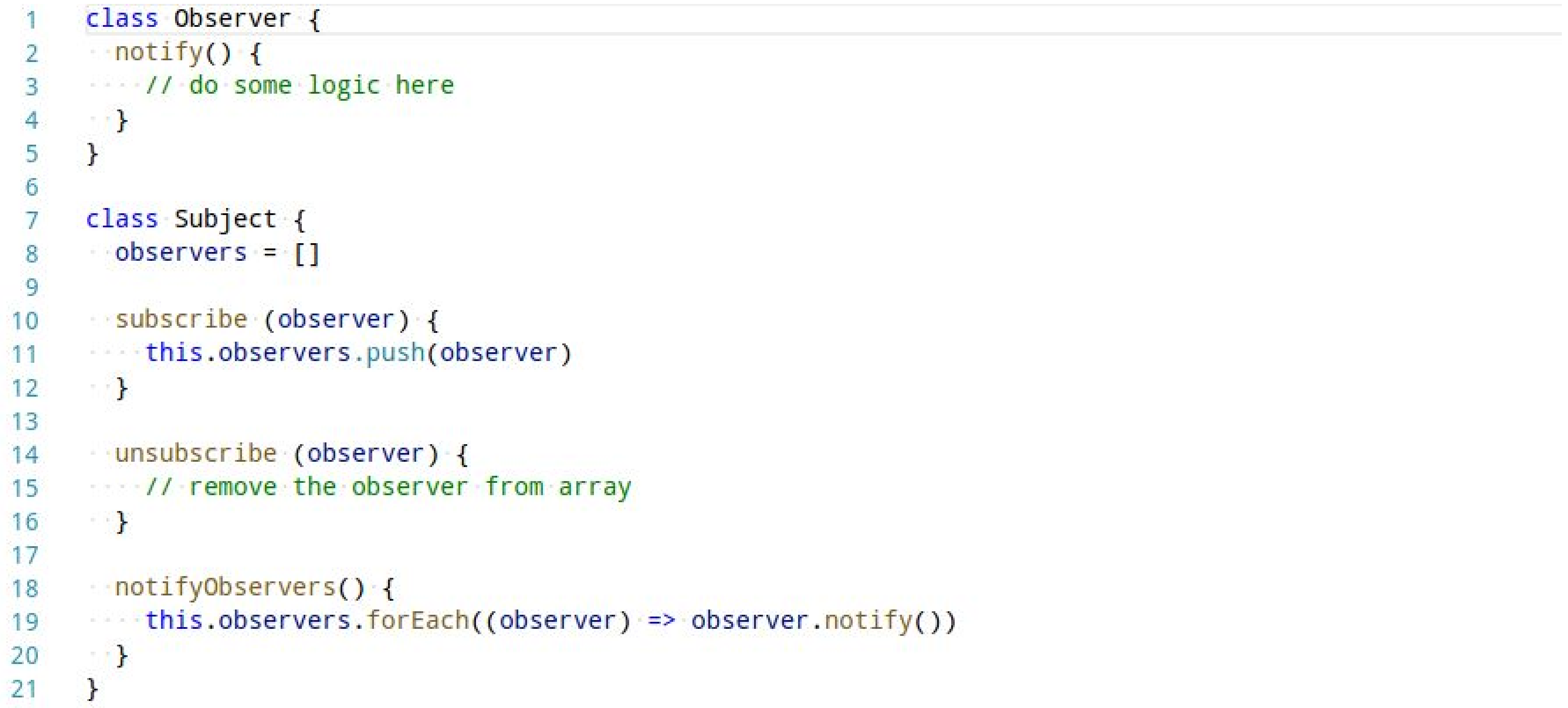
```
let observable$ = Rx.Observable.from([1, 2, 3, 4, 5])

observable$.subscribe(
    (value) => {}, // 1, 2, 3, 4, 5
    (error) => {}, // if any errors
    (completed) => {} // when the iterator has finished
)
```

Observable = Observer + Iterator

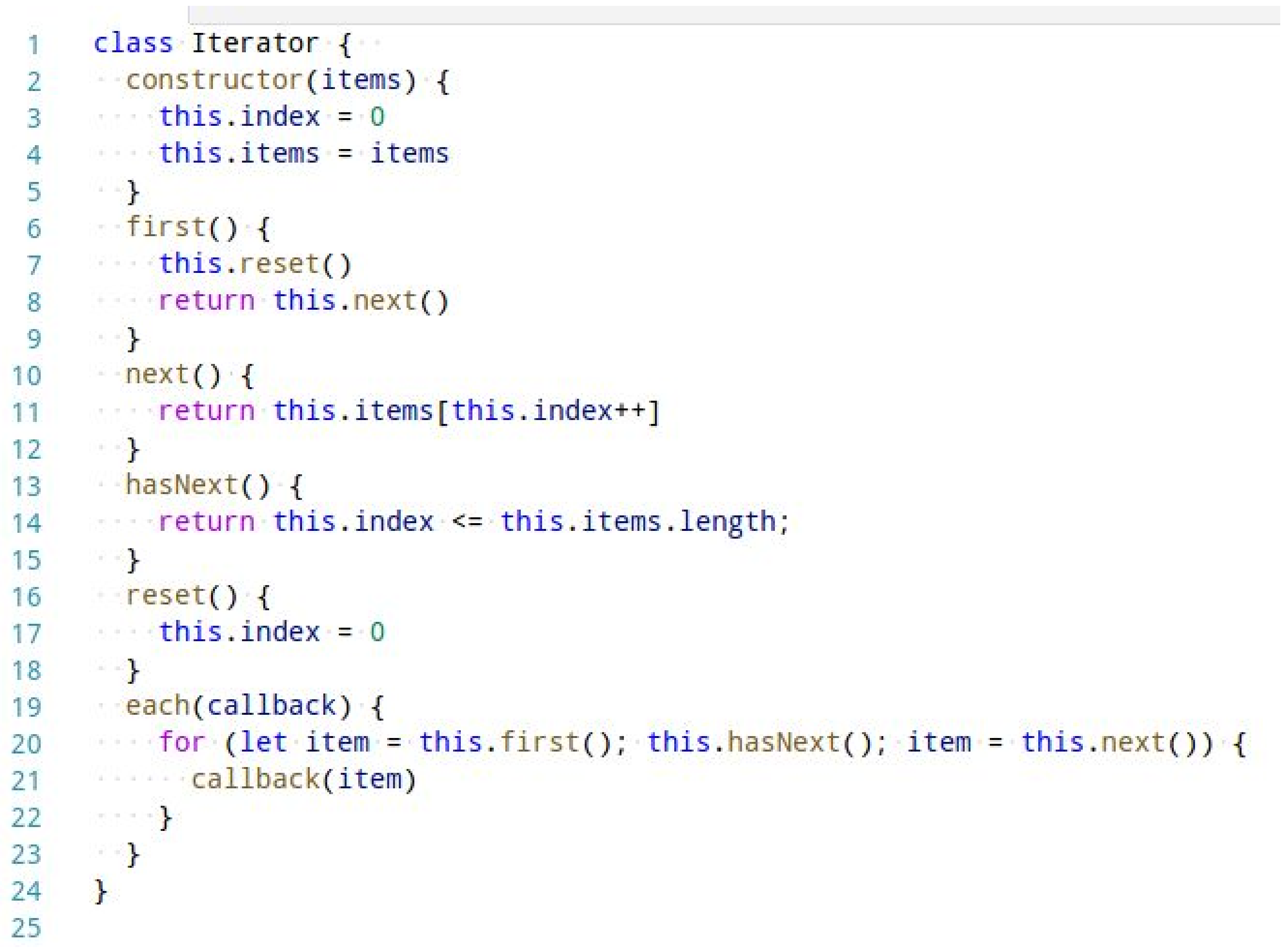
Observer Pattern



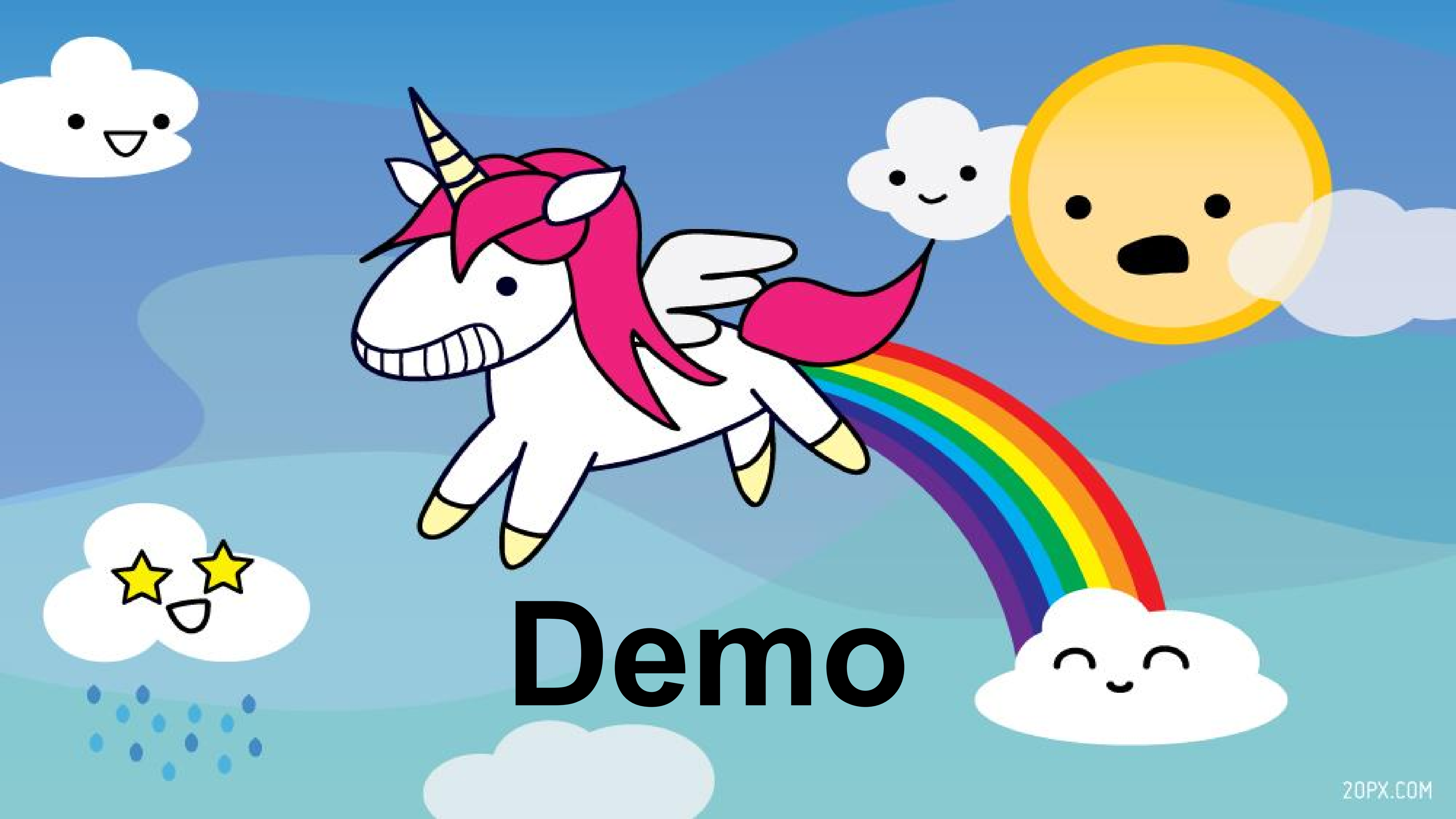
```
notifyObservers()
 for observer in observerCollection
  call observer.notify()
```



```
class Observer {
  notify() {
    // do some logic here
  }
}

class Subject {
  observers = []

  subscribe (observer) {
    this.observers.push(observer)
  }

  unsubscribe (observer) {
    // remove the observer from array
  }

  notifyObservers() {
    this.observers.forEach((observer) => observer.notify())
  }
}
```



```
class Iterator {
  constructor(items) {
    this.index = 0
    this.items = items
  }
  first() {
    this.reset()
    return this.next()
  }
  next() {
    return this.items[this.index++]
  }
  hasNext() {
    return this.index <= this.items.length;
  }
  reset() {
    this.index = 0
  }
  each(callback) {
    for (let item = this.first(); this.hasNext(); item = this.next()) {
      callback(item)
    }
  }
}
```



Demo

Thank you