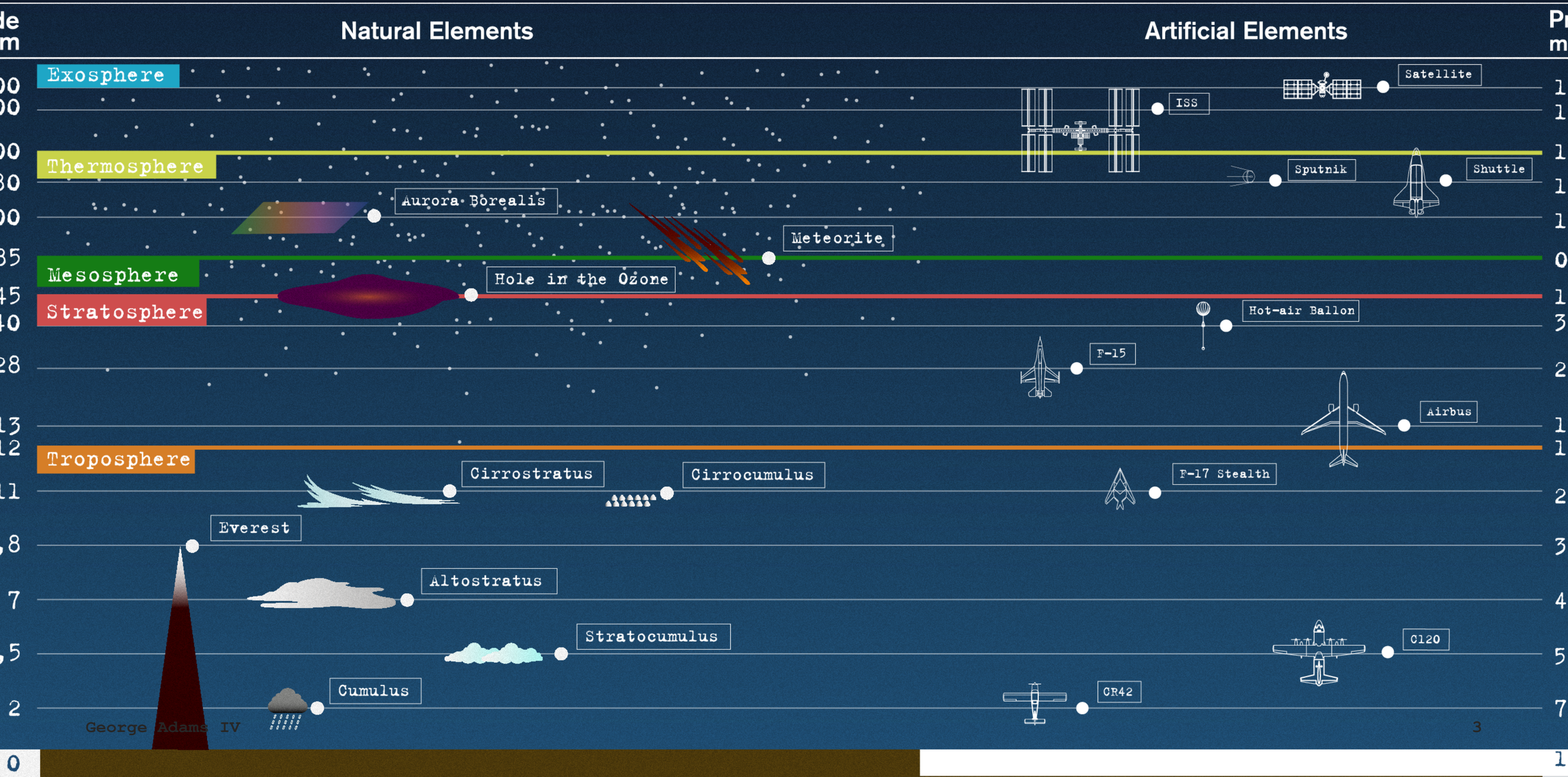
Reactive Extensions for JavaScript

RxJS

Atmosphere of Earth



Multilingual

- .NET
- Java
- Android
- C/C++
- Python
- JavaScript

Let's play a game.

Data-First Development

Types

- Observable
- Observer
- Subject

Observable

```
stream
   .filter(x => x > 0)
   .map(x => x * 2)
   .bufferWithTime(500)
   .subscribe(onNext, onError, onComplete)
```

Observer

```
stream
   .filter(x => x > 0)
   .map(x => x * 2)
   .bufferWithTime(500)
   .subscribe(myObserver)
```

Observer

```
myObserver.onNext(42)
myObserver.onComplete()
```

Subject

```
subject = new BehaviorSubject(42)
subject.subscribe(x => console.log(x))
// 42
subject.onNext(56)
// 56
console.log(subject.getValue())
// 56
```

Bridging to Imperative Code

```
list = subject.getValue()
newList = list.push(newItem)
subject.onNext(newList)
```

Bridging to Events

```
var result = document.getElementById('result')

var source = Rx.Observable.fromEvent(document, 'mousemove')

var subscription = source.subscribe(function (e) {
  result.innerHTML = e.clientX + ', ' + e.clientY
})
```

Bridging to Callbacks

```
var Rx = require('rx'),
    fs = require('fs')

// Wrap the exists method
var exists = Rx.Observable.fromCallback(fs.exists)

var source = exists('file.txt')

// Get the first argument only which is true/false
var subscription = source.subscribe(
  function (x) { console.log('onNext: %s', x) },
  function (e) { console.log('onError: %s', e) },
  function ()  { console.log('onCompleted') }
)

// => onNext: true
// => onCompleted
```

Bridging to Node Callbacks

```
var fs = require('fs'),
    Rx = require('rx')

// Wrap fs.rename
var rename = Rx.Observable.fromNodeCallback(fs.rename)

// Rename file which returns no parameters except an error
var source = rename('file1.txt', 'file2.txt')

var subscription = source.subscribe(
  function (x) { console.log('onNext: success!') },
  function (e) { console.log('onError: %s', e) },
  function ()  { console.log('onCompleted') }
)

// => onNext: success!
// => onCompleted
```

Bridging to Promises

```
// Create a promise which resolves 42
var promise1 = new Promise(function (resolve, reject) {
  resolve(42)
})

var source1 = Rx.Observable.fromPromise(promise1)

var subscription1 = source1.subscribe(
  function (x) { console.log('onNext: %s', x) },
  function (e) { console.log('onError: %s', e) },
  function () { console.log('onCompleted') }
)

// => onNext: 42
// => onCompleted
```

Bridging to Promises

```
// Create a promise which rejects with an error
var promise2 = new Promise(function (resolve, reject) {
  reject(new Error('reason'))
})

var source2 = Rx.Observable.fromPromise(promise2)

var subscription2 = source2.subscribe(
  function (x) { console.log('onNext: %s', x) },
  function (e) { console.log('onError: %s', e) },
  function () { console.log('onCompleted') }
)

// => onError: reject
```

Bridging to Promises

```
var source1 = Rx.Observable.just(1).toPromise()

source1.then(
  function (value) {
    console.log('Resolved value: %s', value)
  },
  function (reason) {
    console.log('Rejected reason: %s', reason)
  }
)

// => Resolved value: 1
```

Bridging to Promises

```
var source = Rx.Observable.range(0, 3)
  .flatMap(function (x) { return Promise.resolve(x * x) })

var subscription = source.subscribe(
  function (x) { console.log('onNext: %s', x) },
  function (e) { console.log('onError: %s', e) },
  function () { console.log('onCompleted') })

// => onNext: 0
// => onNext: 1
// => onNext: 4
// => onCompleted
```

Bridging to React

RxReact

Bridging to React

```
var RxReact = require('rx-react')
var Rx = require('rx')

class MyComponent extends RxReact.Component {
  getStateStream() {
    return Rx.Observable.interval(1000).map(function (interval) {
      return {
        secondsElapsed: interval
      }
    })
  }

  render() {
    var secondsElapsed = this.state ? this.state.secondsElapsed : 0
    return (
      <div>Seconds Elapsed: {secondsElapsed}</div>
    )
  }
}
```

Bridging to React

```
var FuncSubject = require('rx-react').FuncSubject
var React = require('react')
var Rx = require('rx')


var Button = React.createClass({
  componentWillMount: function () {
    this.buttonClicked = FuncSubject.create()

    this.buttonClicked.subscribe(function (event) {
      alert('button clicked')
    })
  },
  render: function() {
    return <button onClick={this.buttonClicked} />
  }
})
```

Into the Troposphere

- Schedulers
- Testing and Debugging
- Backpressure
- Dozens of operators

Summary

- Rx is huge.
- But you can start small.
- Once you know it, you can use it almost anywhere.