Our Asynchronous Past Present And Future

github.com/mattpodwysocki/nodeconfco-2019

Matthew Podwysocki
@mattpodwysocki

@mattpodwysocki

@mattpodwysocki
@mpodwysocki

BluerThanBlueFalcon


MICROSOFT

Visual Studio App Center

appcenter.ms

@VSAppCenter


2009


Async is
Awful


node
JS

Callbacks

+

Events

JS

Callbacks are confusing...

```
fs.open('message.txt', 'a', (err, fd) => {
  if (err) throw err;
  fs.appendFile(fd, 'data to append', 'utf8', (err) => {
    fs.close(fd, (err) => {
      if (err) throw err;
    });
    if (err) throw err;
  });
});
```

Callback Hell is Real...

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height)
              .write(dest + 'w' + width + '_' + filename, function(err) {
                if (err) console.log('Error writing file: ' + err)
              })
          }.bind(this))
        }
      })
    })
  }
})
```

Events are non-compositional...

```
let mouseDown = false;
let last = null;

const onMouseDown = (e) => {
  mouseDown = true;
};

const onMouseUp = (e) {
  mouseUp = true;
};

const onMouseMove = (e) => {
  if (mouseDown) {
    const current = { x: e.target.clientX, y = e.target.clientY };
    const delta = { x: current.x - last.x, y: current.x - last.y };
    // Do something with the delta
    last = current;
  }
};

document.addEventListener('mousedown', onMouseDown, false);
document.addEventListener('mouseup', onMouseUp, false);
document.addEventListener('mosemove', onMouseMove, false);
```


There has
to be a
better way


Today

Promises,
Promises...

then

Node.js 8.0 made things better...

```
const util = require('util');
const fs = require('fs');

const stat = util.promisify(fs.stat);
stat('.').then((stats) => {
  console.log(`This directory is owned by ${stats.uid}`);
}).catch((error) => {
  console.log(`Error getting fs.stat: ${error.message}`);
});
```

async/await

JS

Node.js 10.0 added native API promises...

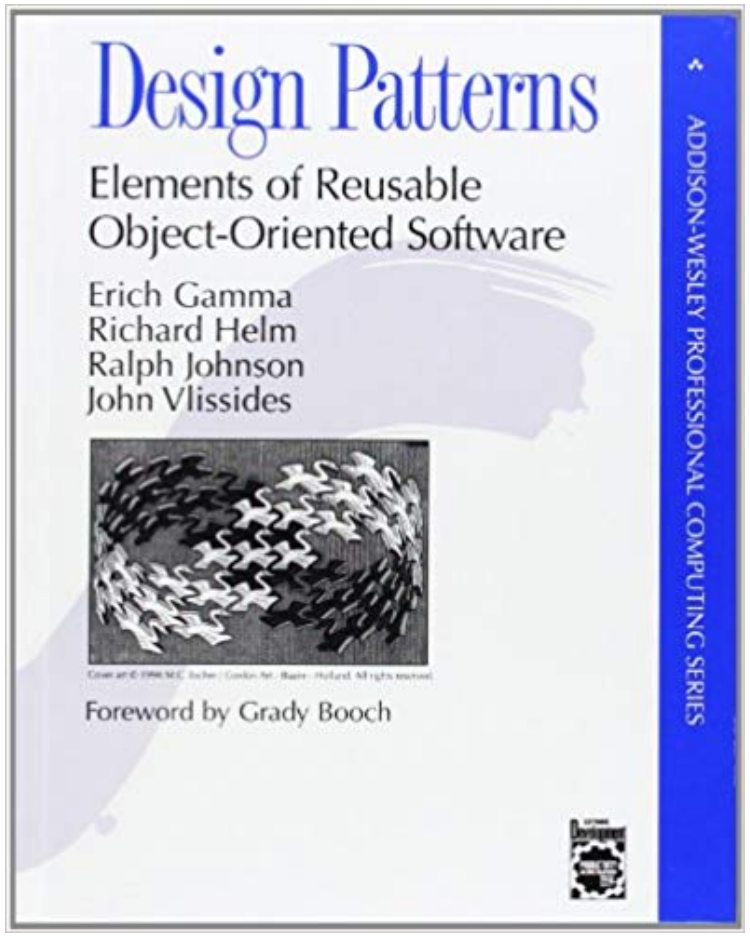
```
const fs = require('fs').promises;

async function openAndClose() {
  let filehandle;
  try {
    filehandle = await fs.open('thefile.txt', 'a+');
    await fs.appendFile(filehandle, 'data to append', 'utf8');
  } finally {
    if (filehandle)
      await filehandle.close();
  }
}
```

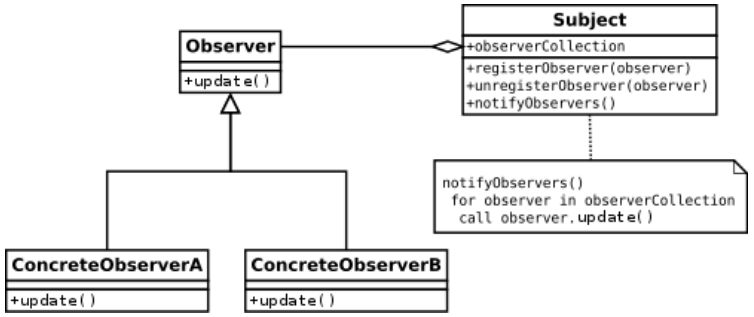
Events?

JS

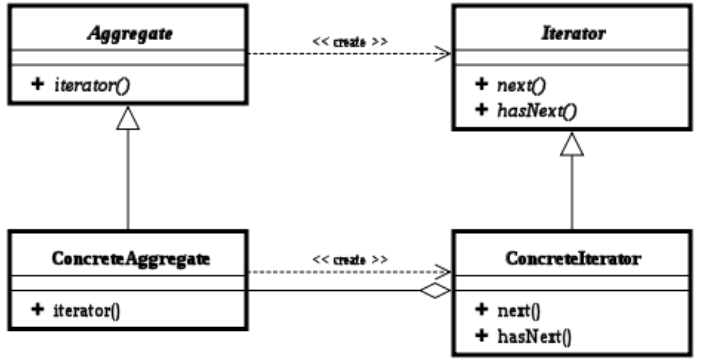
1994 - Design Patterns Emerge



Subject/Observer

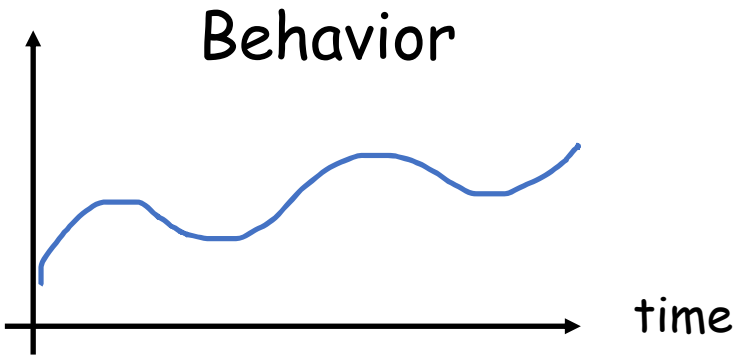


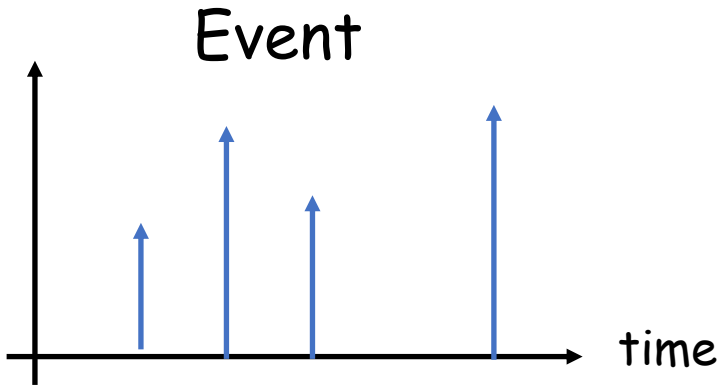
Iterator



Duality of Push to Pull Collections

1997 - Functional Reactive Programming





Functional Reactive Animation
Conal Elliott / Paul Hudak
http://conal.net/papers/icfp97/

Your Mouse is a Database

Web and mobile applications are increasingly composed of asynchronous and realtime streaming services and push notifications.

Erik Meijer

https://queue.acm.org/detail.cfm?id=2169076

Joe Armstrong @joeerl · Apr 4

One on the disadvantages of having a PhD in computer science is that I get asked really difficult questions.

Like - "In gmail on my iPhone I press archive - can I get my mail back?"

and "Why have they changed the interface?"

Why no easy questions like what's a monad?

59 677 3.8K

Ahmad @sudoreality · Apr 4

What's a monad?

4 15

Joe Armstrong
@joeerl

Follow

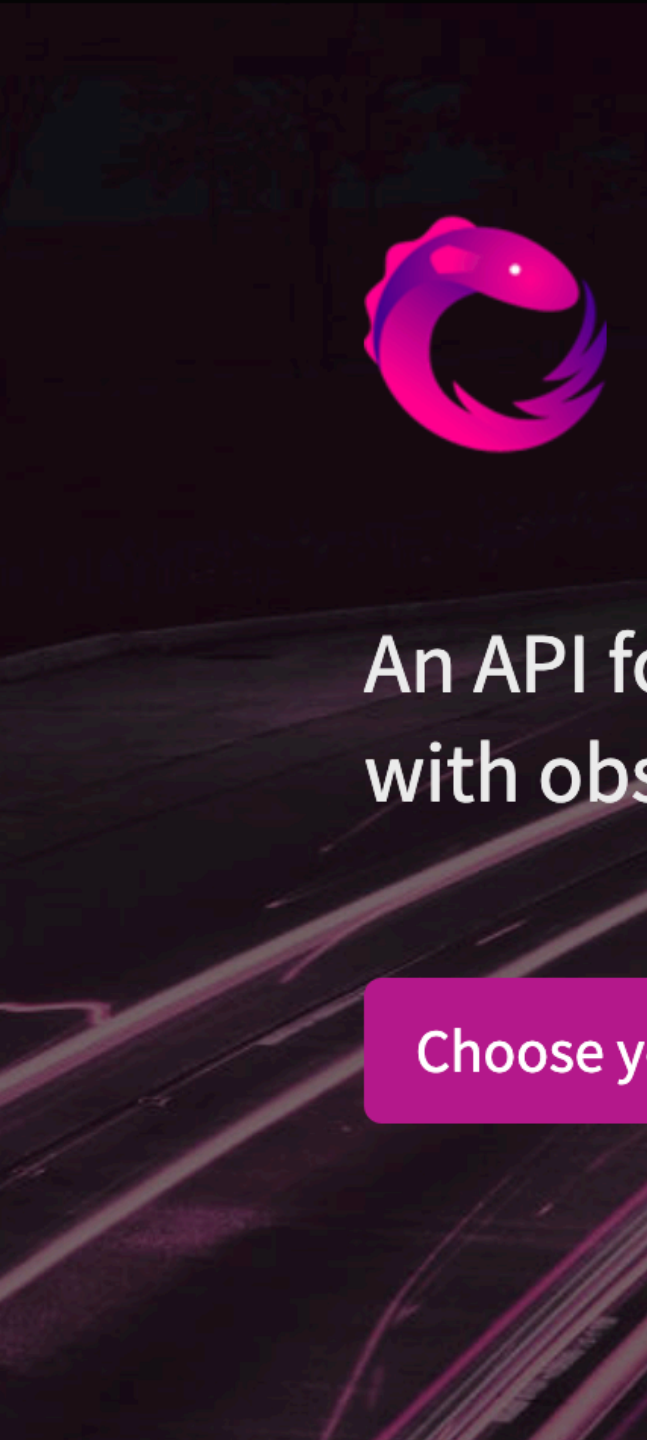
Replying to @sudoreality

it's a thingy - see
en.wikipedia.org/wiki/Monad

11:59 AM - 4 Apr 2019

3 Retweets **60** Likes

Monads???

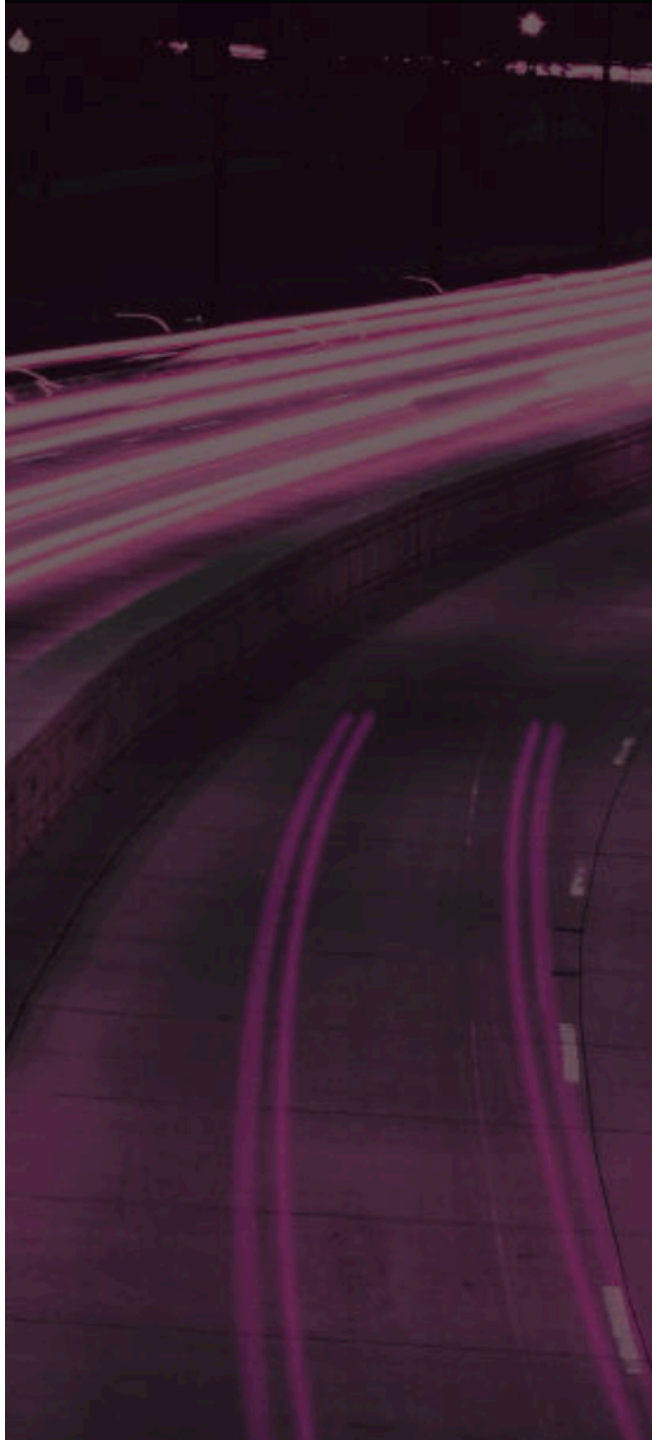


Languages

- 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

ReactiveX for platforms and frameworks

- RxNetty
- RxAndroid
- RxCocoa



What is an Observable?

```
interface Subscription {
    unsubscribe: () => void;
}

interface Observer<T> {
    next: (value: T) => void;
    error: (error: any) => void;
    complete: () => void;
}

interface Observable<T> {
    subscribe: (observer: Observer<T>) => Subscription;
}
```

If you know arrays...

```
array
  .map(x => x * 10)
  .filter(x => x % 3 === 0)
  .forEach(x => ...)
```

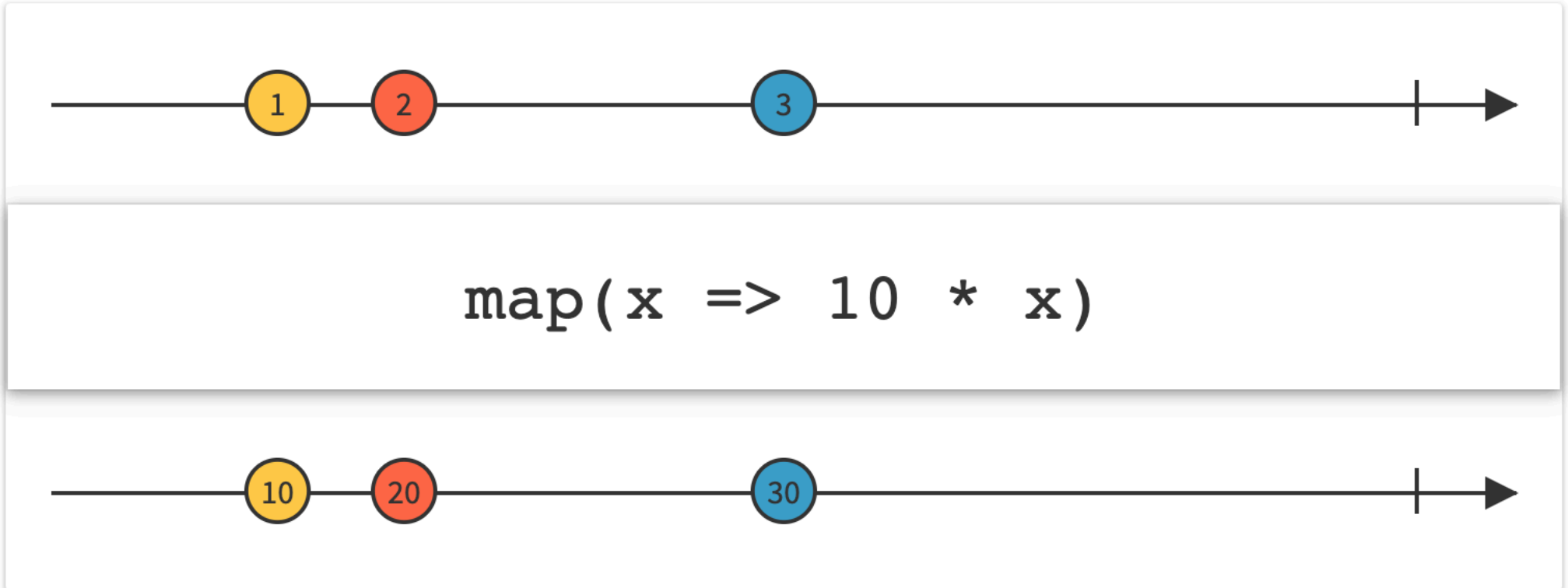
JS

Then you know Observables...

```
observable.pipe(
  map(x => x * 10),
  filter(x => x % 3 === 0)
)
.subscribe(x => ...)
```

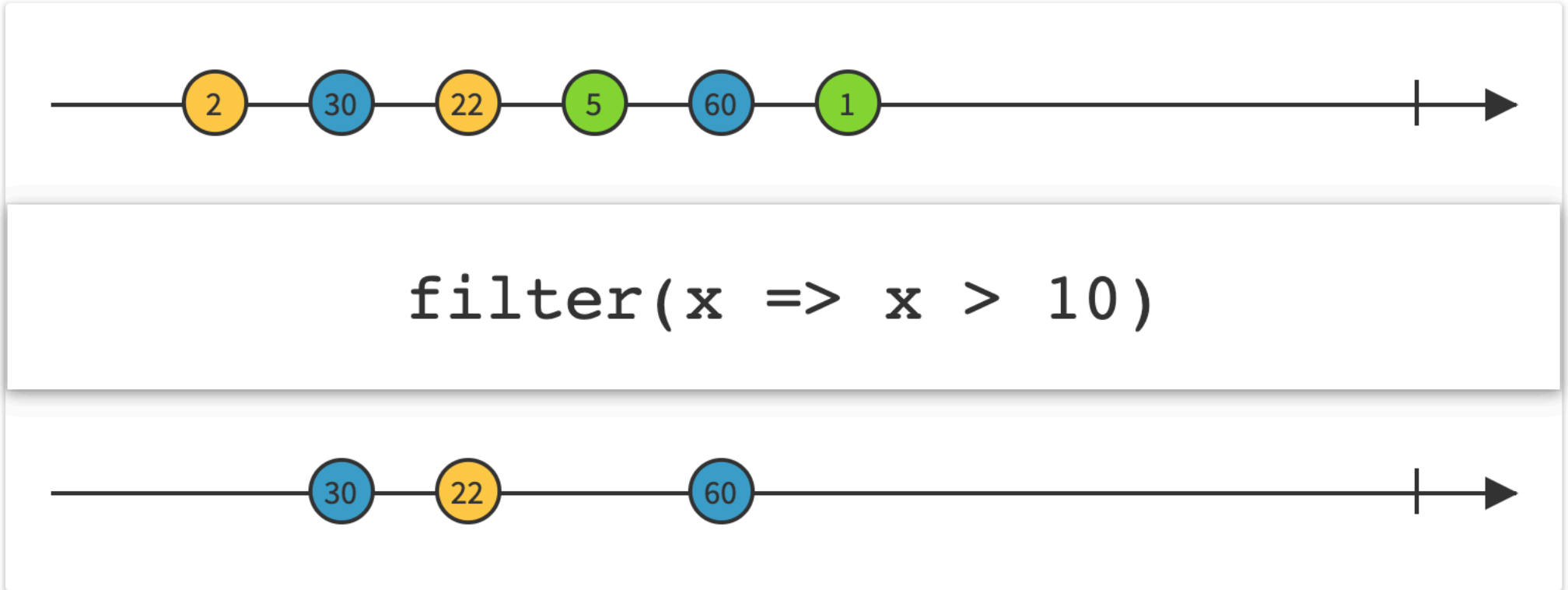
Map

transform the items emitted by an Observable by applying a function to each item



Filter

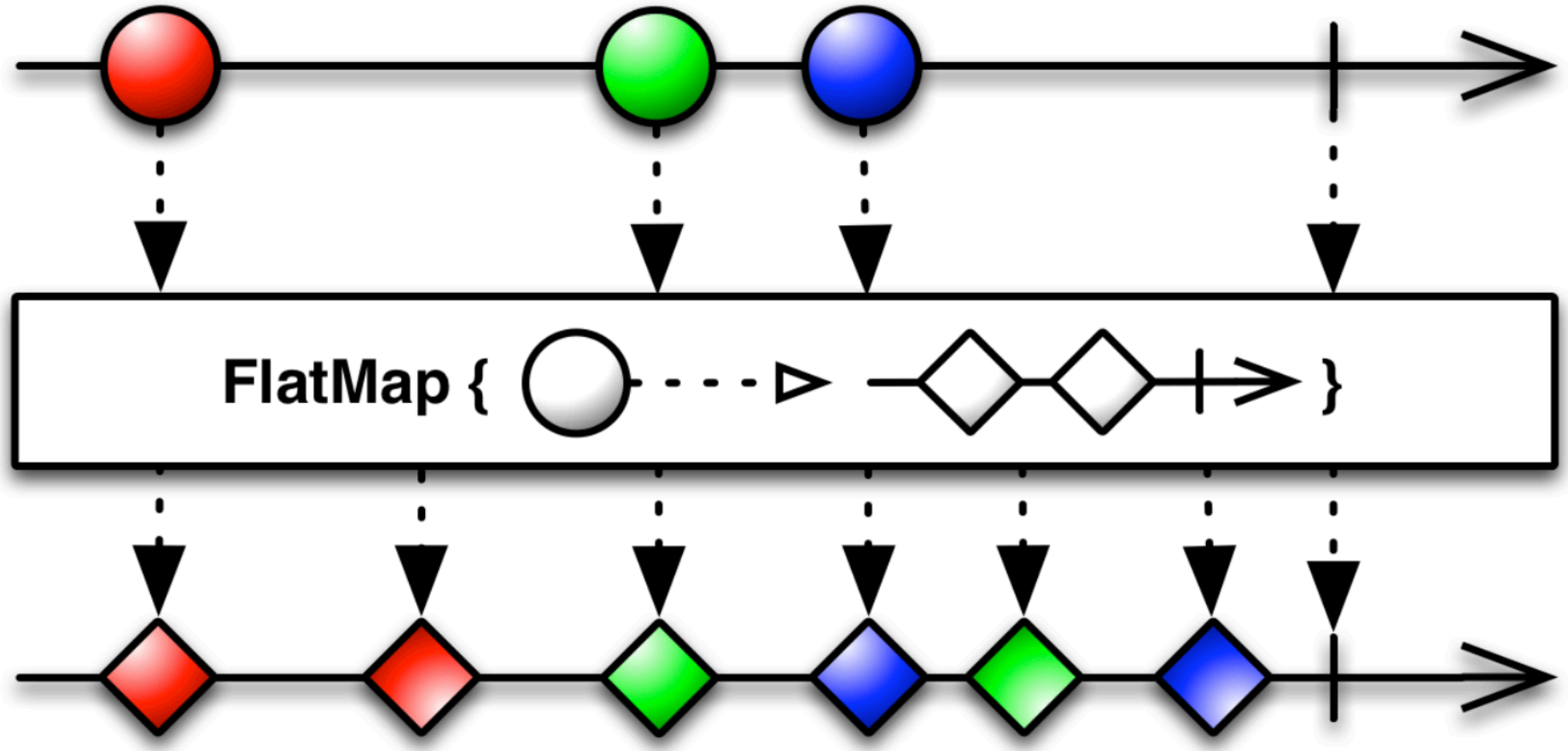
emit only those items from an Observable that pass a predicate test



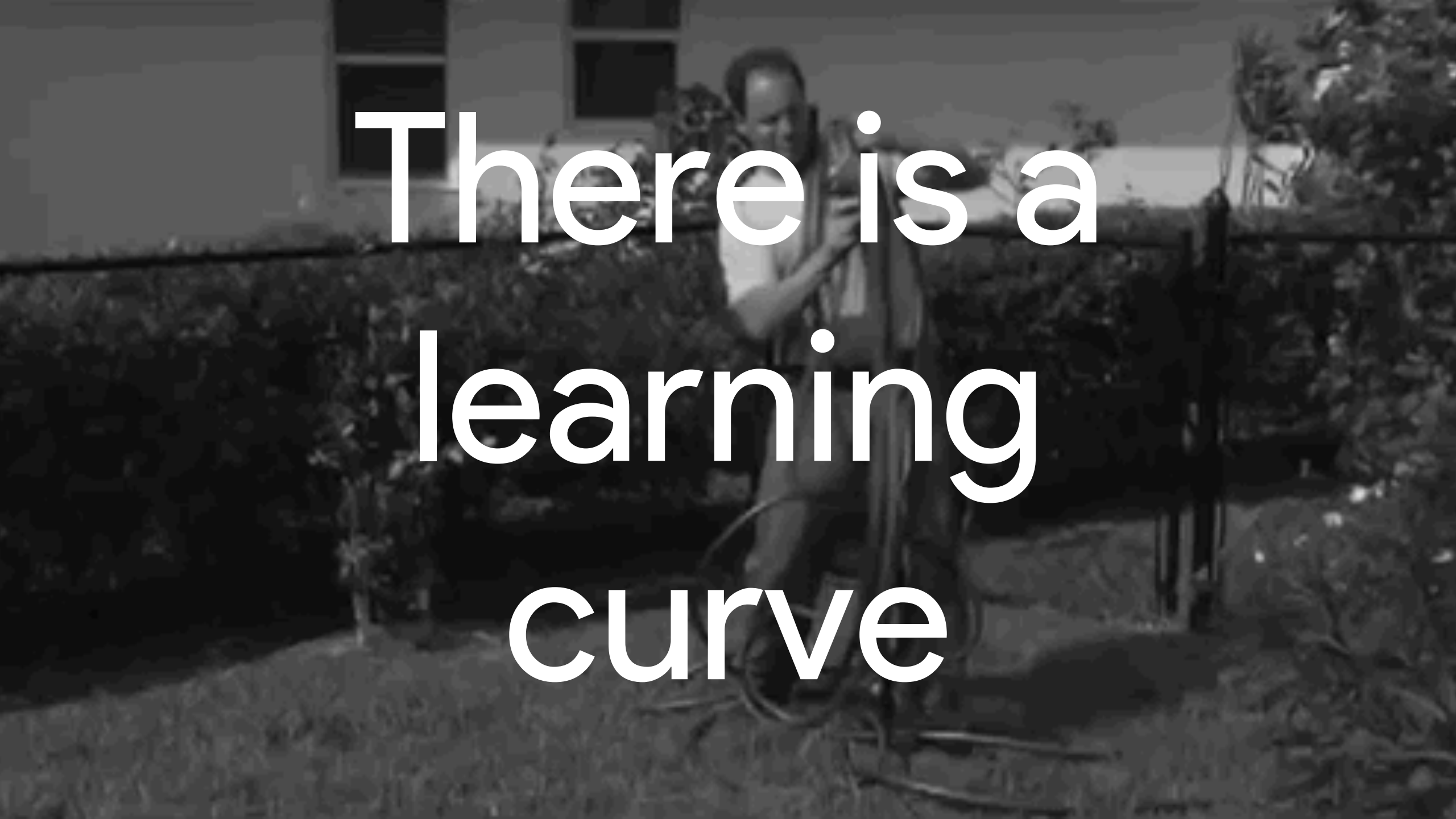
The `Filter` operator filters an Observable by only allowing items through that pass a test that you specify in the form of a predicate function.

FlatMap

transform the items emitted by an Observable into Observables, then flatten the emissions from those into a single Observable



The `FlatMap` operator transforms an Observable by applying a function that you specify to each item emitted by the source Observable, where that function returns an Observable that itself emits items. `FlatMap` then merges the emissions of these resulting Observables, emitting these merged results as its own sequence.


There is a
learning
curve

An Observable Community...

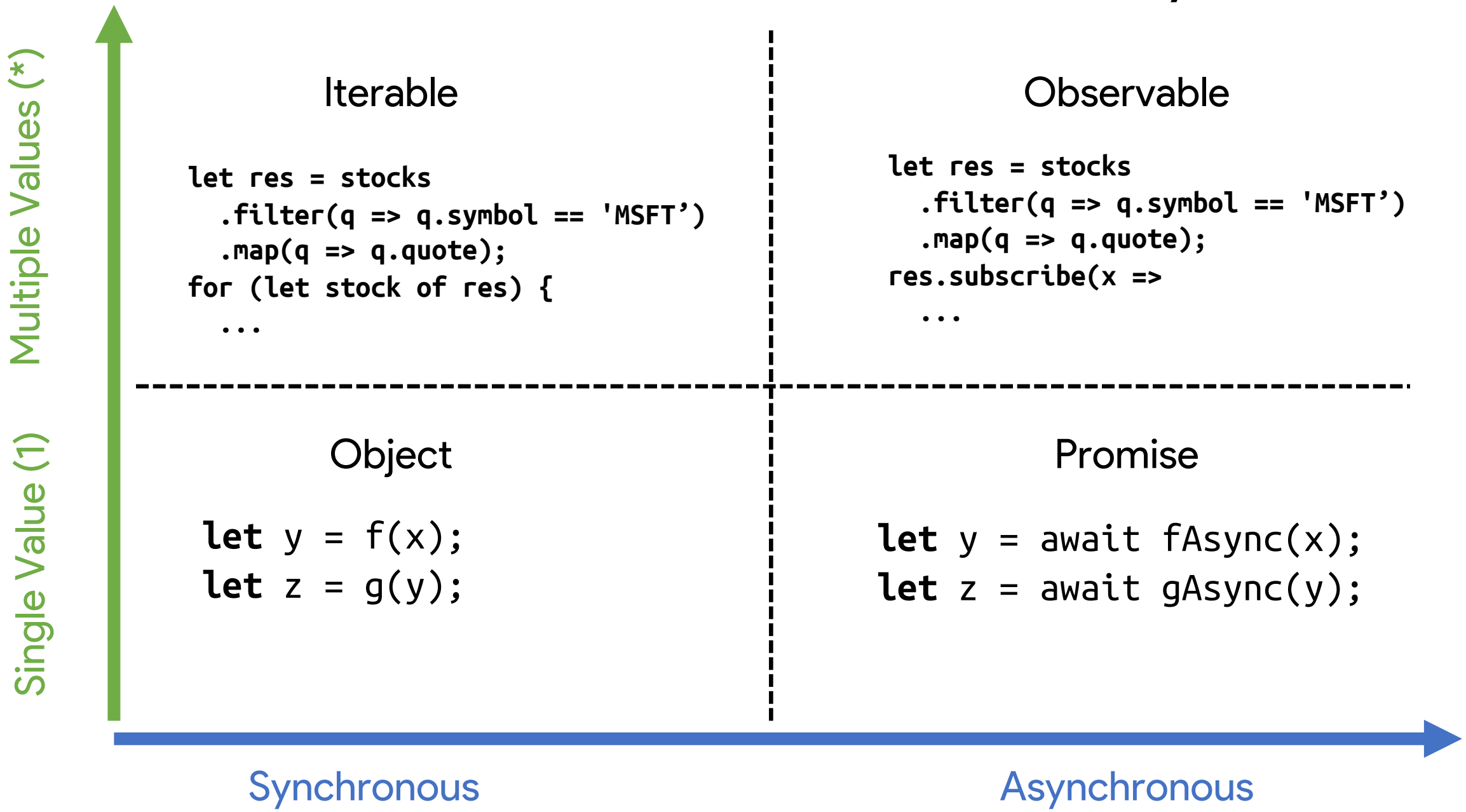
Relay

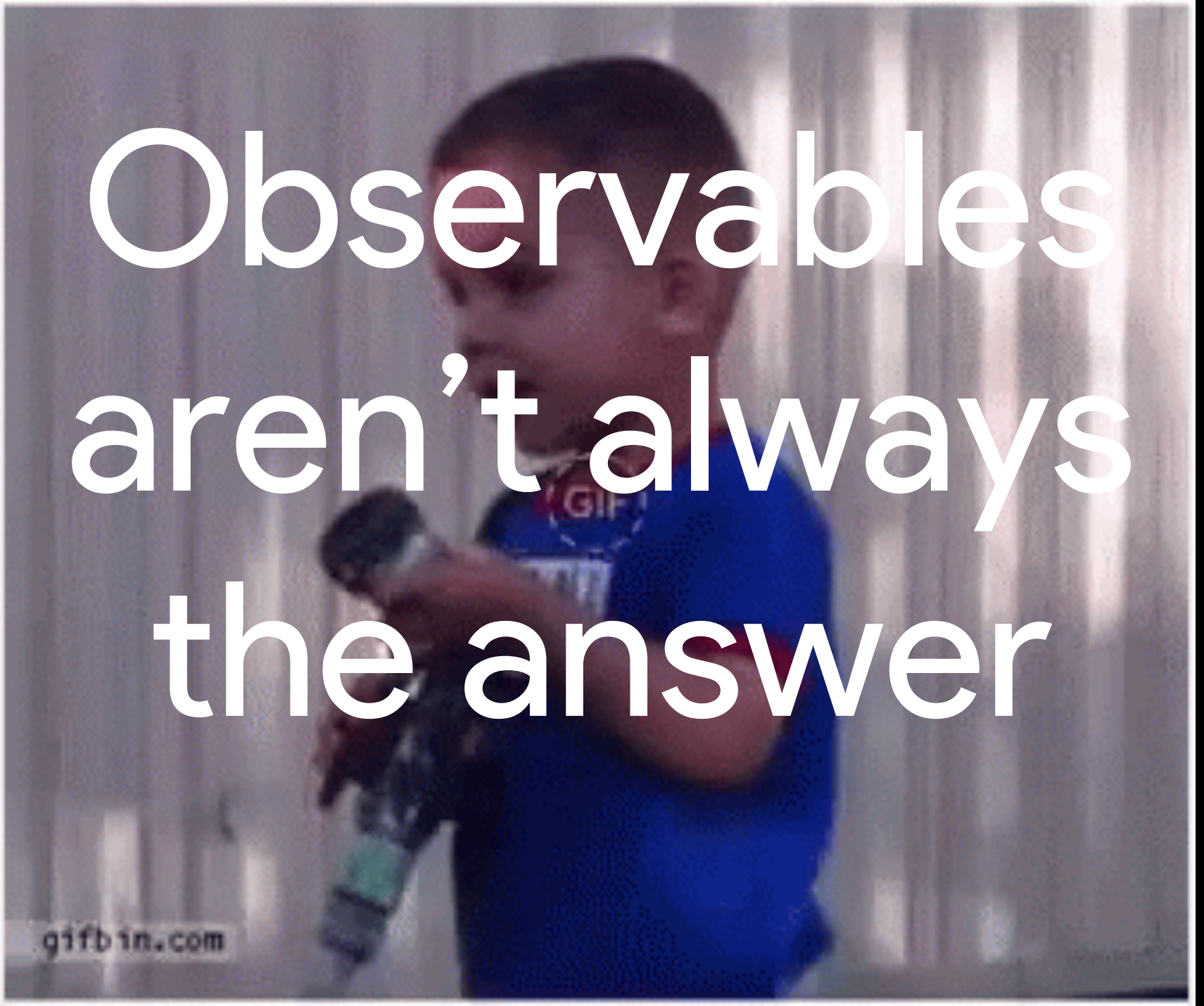
A JavaScript framework for building data-driven React applications

Redux

A predictable state container for JavaScript apps.

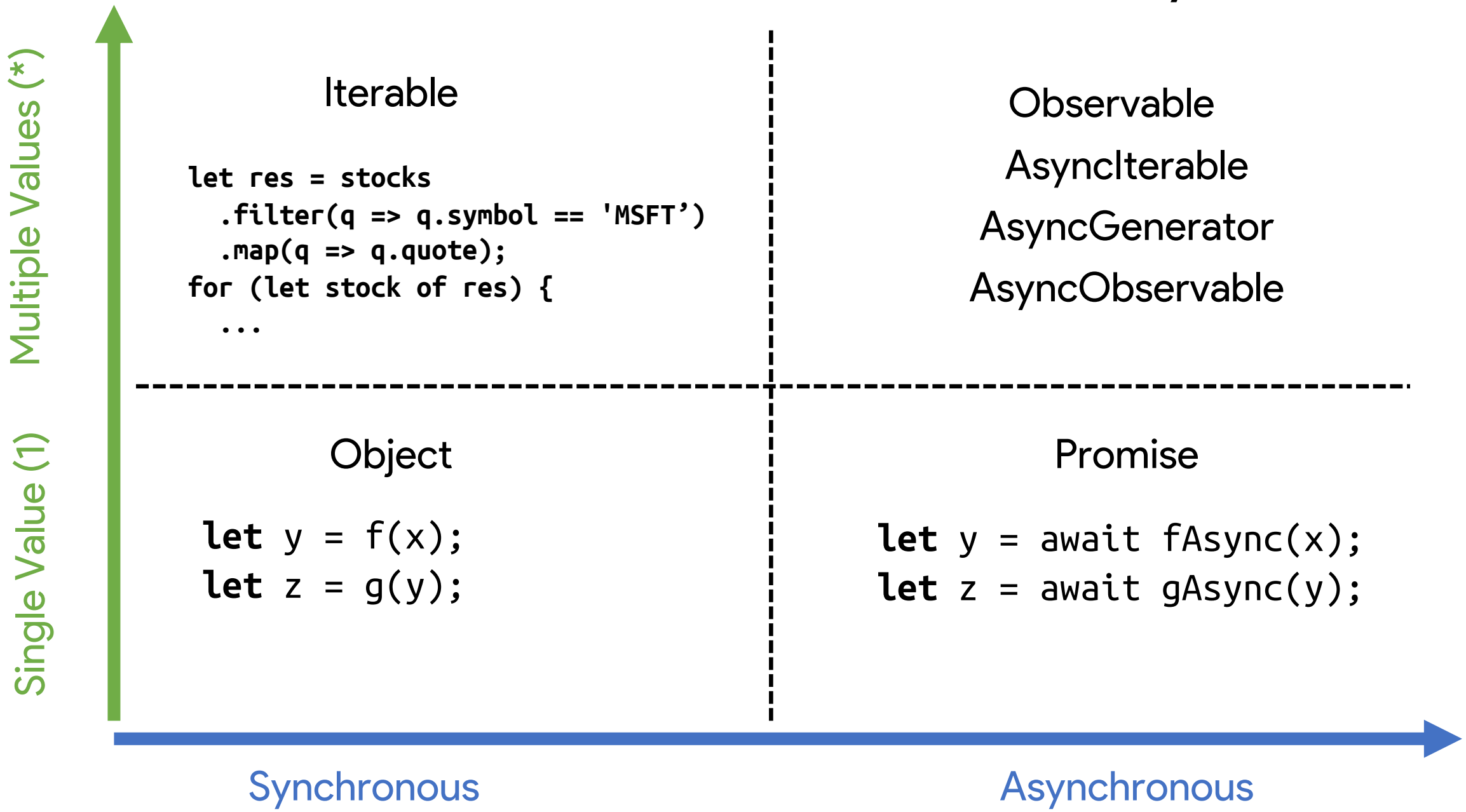
General Theory of Reactivity




Observables
aren't always
the answer
gifbin.com

Things can overload

General Theory of Reactivity



AsyncGenerators
AsyncIterables
JS

What is an AsyncIterable?

```
// Iterator
const iterator = createIterator();
iterator.next(); // Object {value: 1, done: false}
iterator.next(); // Object {value: 2, done: false}
iterator.next(); // Object {value: 3, done: false}
iterator.next(); // Object {value: undefined, done: true}

// AsyncIterator
const asyncIterator = createAsyncIterator();
await asyncIterator.next(); // Object {value: 1, done: false}
await asyncIterator.next(); // Object {value: 2, done: false}
await asyncIterator.next(); // Object {value: 3, done: false}
await asyncIterator.next(); // Object {value: undefined, done: true}
```

Gimme an example!

```
async function* asyncDoggos() {
  const url = 'https://dog.ceo/api/breeds/image/random';

  while (true) {
    const response = await fetch(url);
    const json = await response.json();
    yield json.message;
  }
}

for await (let dog of asyncDoggos()) {
  showDoggo(dog);
}
```

Node Streams are AsyncIterables...

```
const fs = require('fs');

async function print(readable) {
  readable.setEncoding('utf8');
  let data = '';
  for await (const k of readable) {
    data += k;
  }
  console.log(data);
}

print(fs.createReadStream('file')).catch(console.log);
```

AsyncIterables made easy with IxJS

```
import { as } from 'ix/asynciterable';
import { map } from 'ix/asynciterable/operators';

async function* getData() { ... }

let data = getData()
  .pipe(map(transformData))
  .pipe(domEncodeStream);

for await (let item of data) {
  ...
```

https://github.com/reactivex/ixjs


Where will
we go
next?

ECMAScript Observable

This proposal introduces an **Observable** type to the ECMAScript standard library. The **Observable** type can be used to model push-based data sources such as DOM events, timer intervals, and sockets. In addition, observables are:

- *Compositional*: Observables can be composed with higher-order combinators.
- *Lazy*: Observables do not start emitting data until an **observer** has subscribed.

Example: Observing Keyboard Events

Using the **Observable** constructor, we can create a function which returns an observable stream of events for an arbitrary DOM element and event type.

```
function listen(element, eventName) {
    return new Observable(observer => {
        // Create an event handler which sends data to the sink
        let handler = event => observer.next(event);

        // Attach the event handler
        element.addEventListener(eventName, handler, true);

        // Return a cleanup function which will cancel the event stream
        return () => {
            // Detach the event handler from the element
            element.removeEventListener(eventName, handler, true);
        };
    });
}
```

https://github.com/tc39/proposal-observable

ECMAScript proposal: `Emitter`

- JavaScript's most general container, over single and multiple, sync and async values
- Operators form the Generics, a large part of the eventual standard library
- Composable algorithmic transformations, decoupled from their input and outputs
- Well-integrated with language rather than inventing lots of new concepts, (i.e. Promises, iteration, async iteration, etc)
- Backwards-compatible unification of platform API (interfaces like EventTarget and EventEmitter)
- Ultra-High Performance (faster than most.js) and memory efficient

Reactive programming examples

Log XY coordinates of click events on `<button>` elements:

```
const { on, map, filter, run } = Emitter

run(
  on(document, 'click')
, filter(ev => ev.target.tagName === 'BUTTON')
, map(ev => ({ x: ev.clientX, y: ev.clientY }))
, coords => console.log(coords)
)
```

https://github.com/pemrouz/proposal-emitter

AsyncObservables?

```
import { AsyncObservable } from 'ix/asyncobservable';

const data$ = new AsyncObservable(async obs => {
  const socket = new Socket();

  const onReceive = async function (data) {
    await obs.nextAsync(data);
  };

  await socket.openAsync();
  socket.receiveAsync(onReceive);

  return async function () {
    await socket.closeAsync();
  };
});

const subscription = await data$
  .subscribeAsync(...);
```

https://github.com/reactivex/ixjs

¡Muchas gracias a todos!

github.com/mattpodwysocki/nodeconfco-2019

Matthew Podwysocki
@mattpodwysocki