

NETFLIX



Async JavaScript at Netflix, Microsoft and the World



Matthew Podwysocki

@mattpodwysocki

github.com/mattpodwysocki/jsconfasia-2015





**OR:
HOW I LEARNED TO STOP WORRYING ABOUT
ASYNCHRONOUS PROGRAMMING AND LOVE THE
OBSERVABLE**

Or "I thought I had a problem. I thought to myself, "I know, I'll solve it with promises and events!". have Now problems. two I



**Principal SDE
Open Sourcerer**
@mattpodwysocki
github.com/mattpodwysocki

@ThaliProject
github.com/thaliproject

MICROSOFT



@ThaliProject

<http://thaliproject.org>

<http://github.com/thaliproject>



Reactive Extensions (Rx)

@ReactiveX
<http://reactivex.io>

The Netflix logo, featuring the word "NETFLIX" in its signature red, bold, sans-serif font, centered within a white rounded rectangle.

Stream Movies From Any Device

1/3 of US Broadband Traffic

This is the story of how Netflix, Microsoft and others solved

BIG async problems

by thinking differently about

Events.

The Netflix App is Asynchronous

- App Startup
- Player
- Data Access
- Animations
- View/Model Binding



Asynchronous Nightmares

- Memory Leaks
- Race Conditions
- Callback Hell
- Complex State Machines
- Disjointed Error Handling



2014

Real-Time is Everywhere...



PHOTO: LEATHER

Let's Face It, Asynchronous Programming is Awful!



“We choose to go to solve asynchronous programming and do the other things, not because they are easy, but because they are hard”



**Former US President John F. Kennedy - 1962
[citation needed]**

Callback Hell

```
function play(movieId, callback) {  
  let movieTicket, playError,  
    tryFinish = () => {  
      if (playError) {  
        callback(playError);  
      } else if (movieTicket && player.initialized) {  
        callback(null, ticket);  
      }  
    };  
  if (!player.initialized) {  
    player.init( error => {  
      playError = error;  
      tryFinish();  
    }  
  }  
  authorizeMovie( (error, ticket) => {  
    playError = error;  
    movieTicket = ticket;  
    tryFinish();  
  });  
}
```





culturepub.fr

next
ad ➔

Events and the Enemy of the State





First Class Async with Promises

then

```
player.initialize()  
  .then(authorizeMovie, loginError)  
  .then(playMovie, unauthorizedMovie)
```

Breaking the Promise...

then

```
let promise;

input.addEventListener('keyup', (e) => {

  if (promise) {
    // Um, how do I cancel?
  } else {
    promise = getData(e.target.value).then(populateUI);
  }
}, false);
```

Aborting a fetch #27

[New issue](#)

! Open **annevk** opened this issue on Mar 26 · 204 comments



annevk commented on Mar 26

Owner

Goal

Provide developers with a method to abort something initiated with `fetch()` in a way that is not overly complicated.

Previous discussion

- [#20](#)
- [slightlyoff/ServiceWorker#592](#)
- [slightlyoff/ServiceWorker#625](#)
- [whatwg/streams#297](#)

Viable solutions

We have two contenders. Either `fetch()` returns an object that is more than a promise going forward or `fetch()` is passed something, either an object or a callback that gets handed an object.

A promise-subclass

Labels

None yet

Milestone

No milestone

Assignee



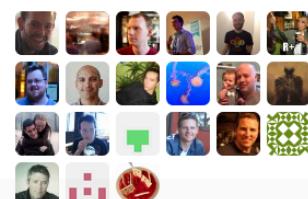
jakearchibald

Notifications

► **Subscribe**

You're not receiving notifications from this thread.

21 participants



then

```
var resou  
try {  
    var res  
} catch (  
    throw e  
} finally  
resource  
}
```

```
e) {
```



It's the final countdown

// How do I clean up my resource?



then



UNSAFE AT ANY SPEED

The Designed-In Dangers
of The American Automobile
By Ralph Nader

MAKE GIFS AT GIFSOUP.COM



stroom prosesing



Let's Face it, Streams¹ were terrible...

- The pause method didn't
- The 'data' event started immediately, ready or not!
- Can't just consume a specified number of bytes
- Pause and resume were impossible to get right...





Streams 2 Electric Boogaloo



Streams 33 1/3



1994



Design Patterns

Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

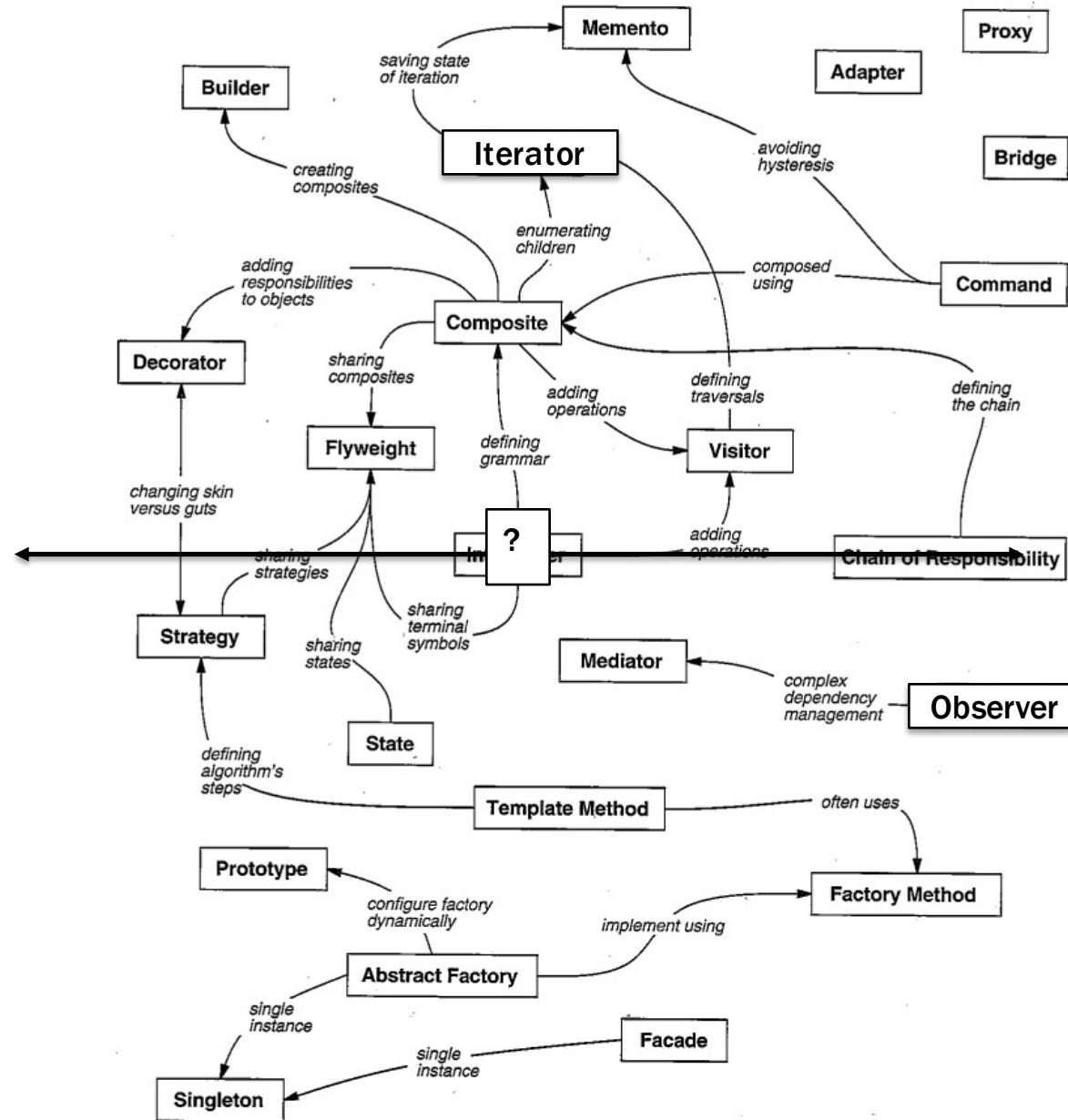


Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES





Design Pattern Relationships

Iterator Pattern with ES2015

```
> let iterator = getNumbers(); █  
> console.log(iterator.next()); █  
> { value: 1, done: false }  
> █onsole.log(iterator.next()); █  
> { value: 2, done: false }  
> █onsole.log(iterator.next()); █  
> { value: 3, done: false }  
> █onsole.log(iterator.next()); █  
> { done: true }  
> █
```

Subject/Observer Pattern with the DOM

```
> document.addEventListener(  
  'mousemove',  
  e =>  
    console.log(e);  
); ■  
  
> { clientX: 425, clientY: 543 }  
> { clientX: 450, clientY: 558 }  
> { clientX: 455, clientY: 562 }  
> { clientX: 460, clientY: 743 }  
> { clientX: 476, clientY: 760 }
```

“What’s the
difference between
an **Array**...

[{x: 23, y: 44}, {x:27, y:55}, {x:27, y:55}]

... and an Event?

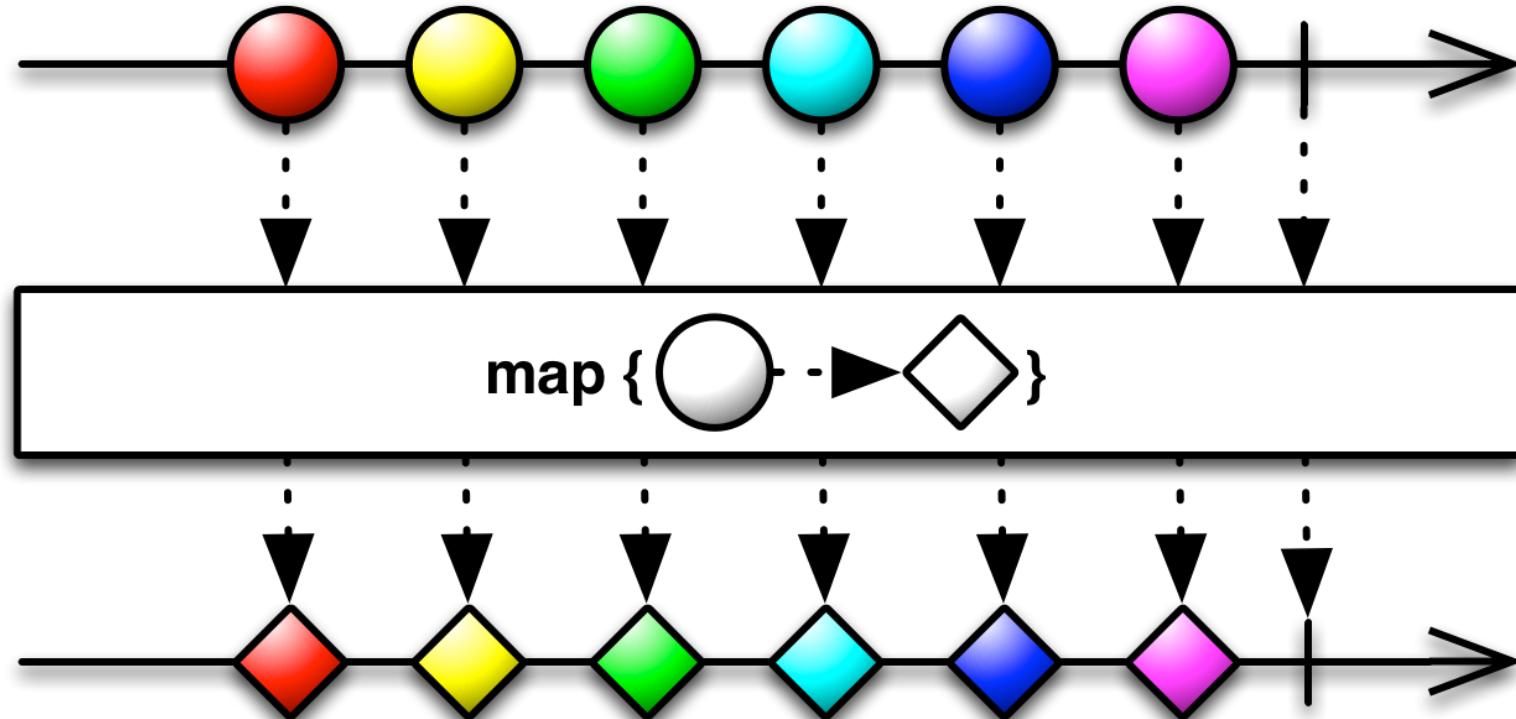


Events and Arrays are *both*
collections.

The majority of asynchronous
code from Netflix, Microsoft and
others is written with just a few
flexible functions.

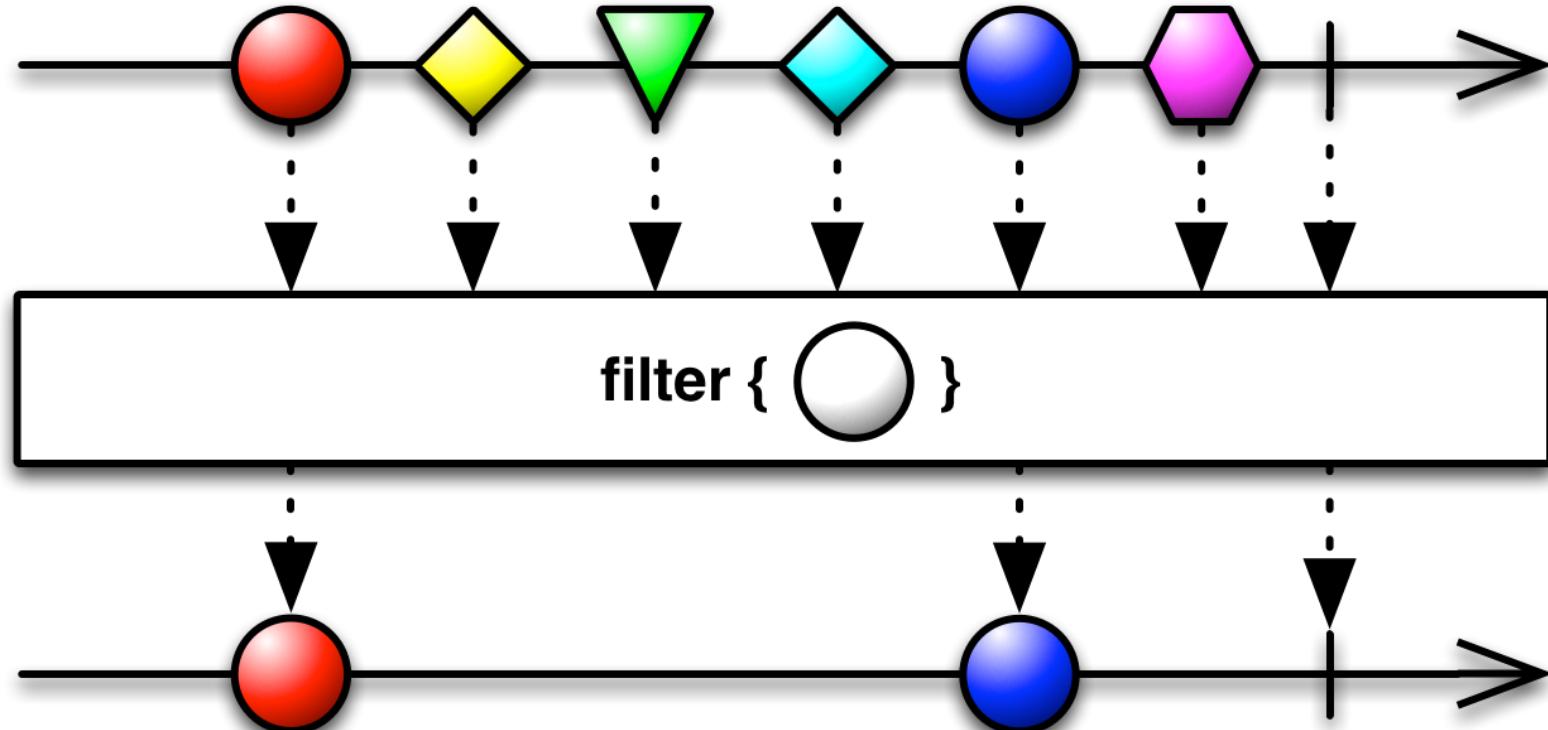
map(selector, thisArg)

Transform the items emitted by an Collection by applying a function to each of them



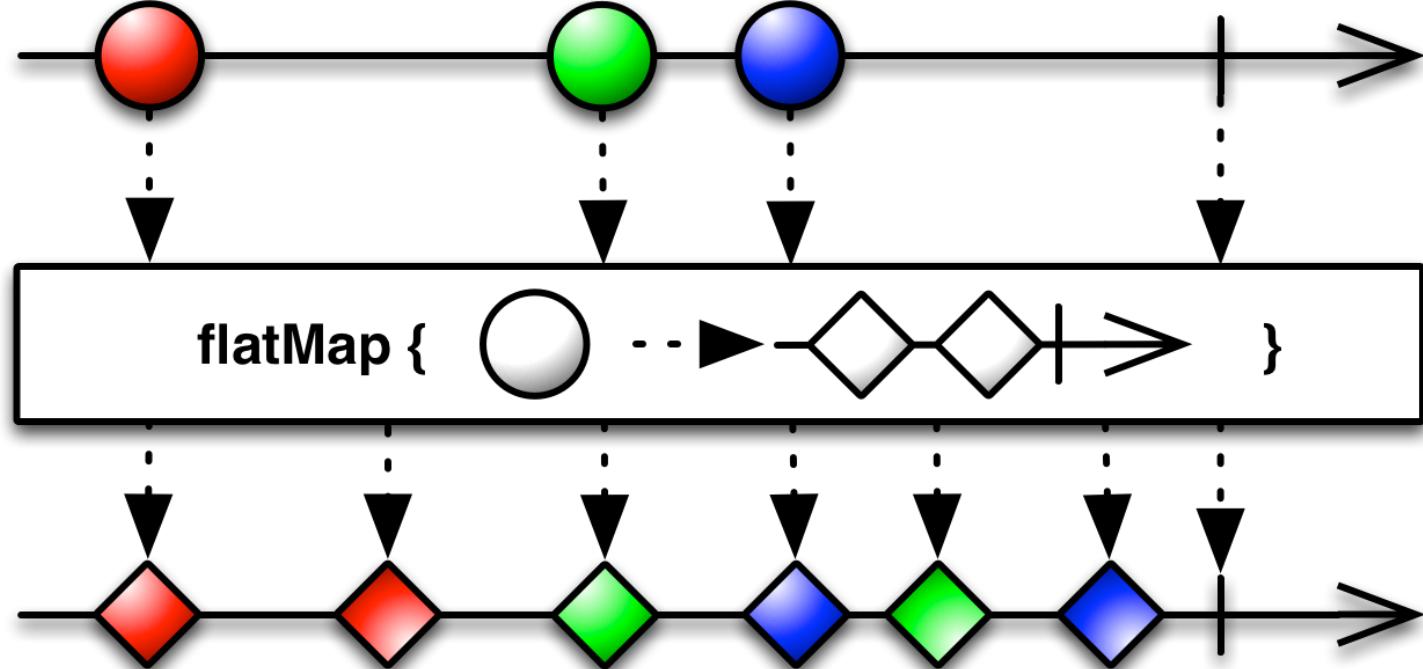
`filter(predicate, thisArg)`

Filter items emitted by a Collection



flatMap(selector)

Transform the items emitted by a Collection into Collections, then flatten this into a single Collection



MAYBE WE CAN
ITERATE OVER...

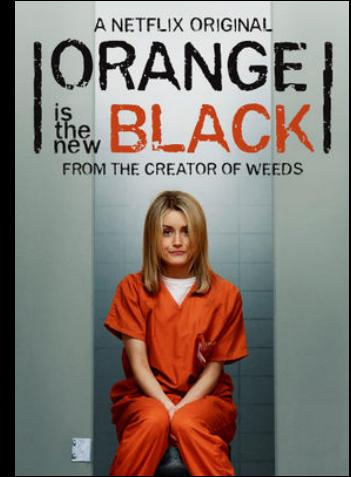
FLATMAP
THAT SHIT!



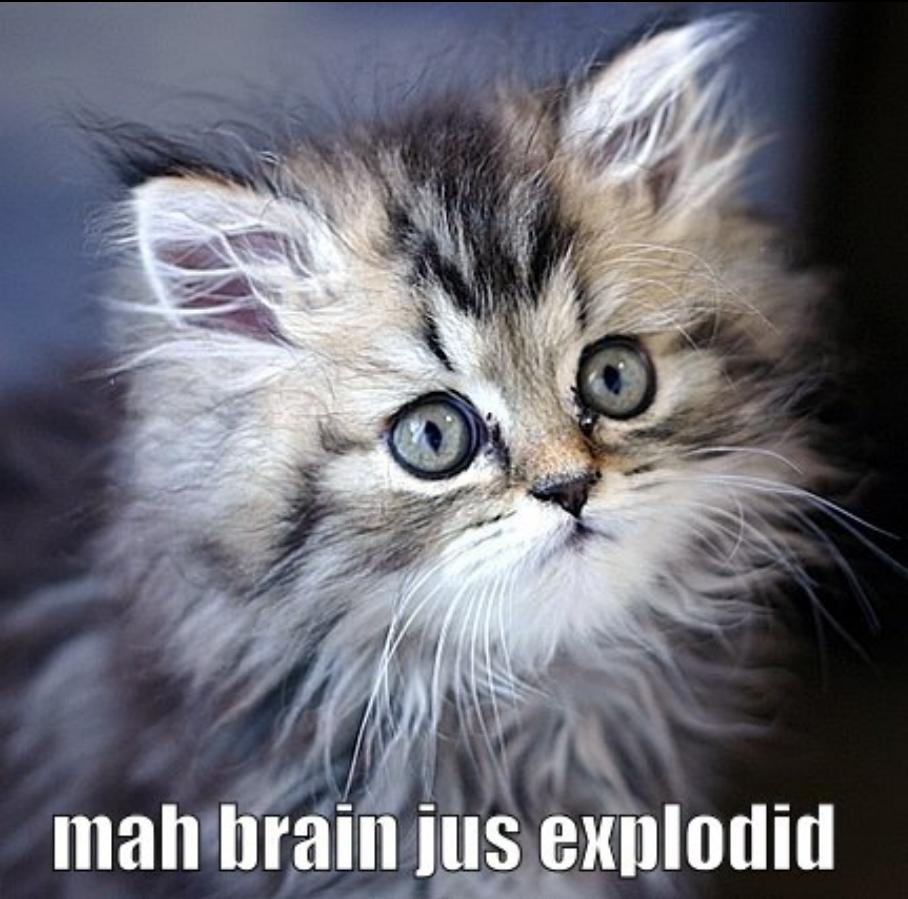
Top-rated Movies Collection

```
const getTopRatedFilms = user => {
  user.videoLists
    .flatMap( videoList =>
      videoList.videos
        .filter( v => v.rating === 5)
    )
};
```

```
getTopRatedFilms(me)
  .forEach(displayMovie);
```



What if I told you...



...that you could create a drag event...
...with the almost the **same code**

mah brain jus explodid

Mouse Drags Collection

```
const getElementDrags = elmt => {
  DOM.mousedown(elmt)
    .flatMap( md =>
      dommousemove(document)
        .filter .takeUntil(DOM.mouseup(elmt))
    )
};
```

```
getElementDrags(image)
  .forEach(moveImage)
```





Everything is a stream

First-Class Asynchronous Values

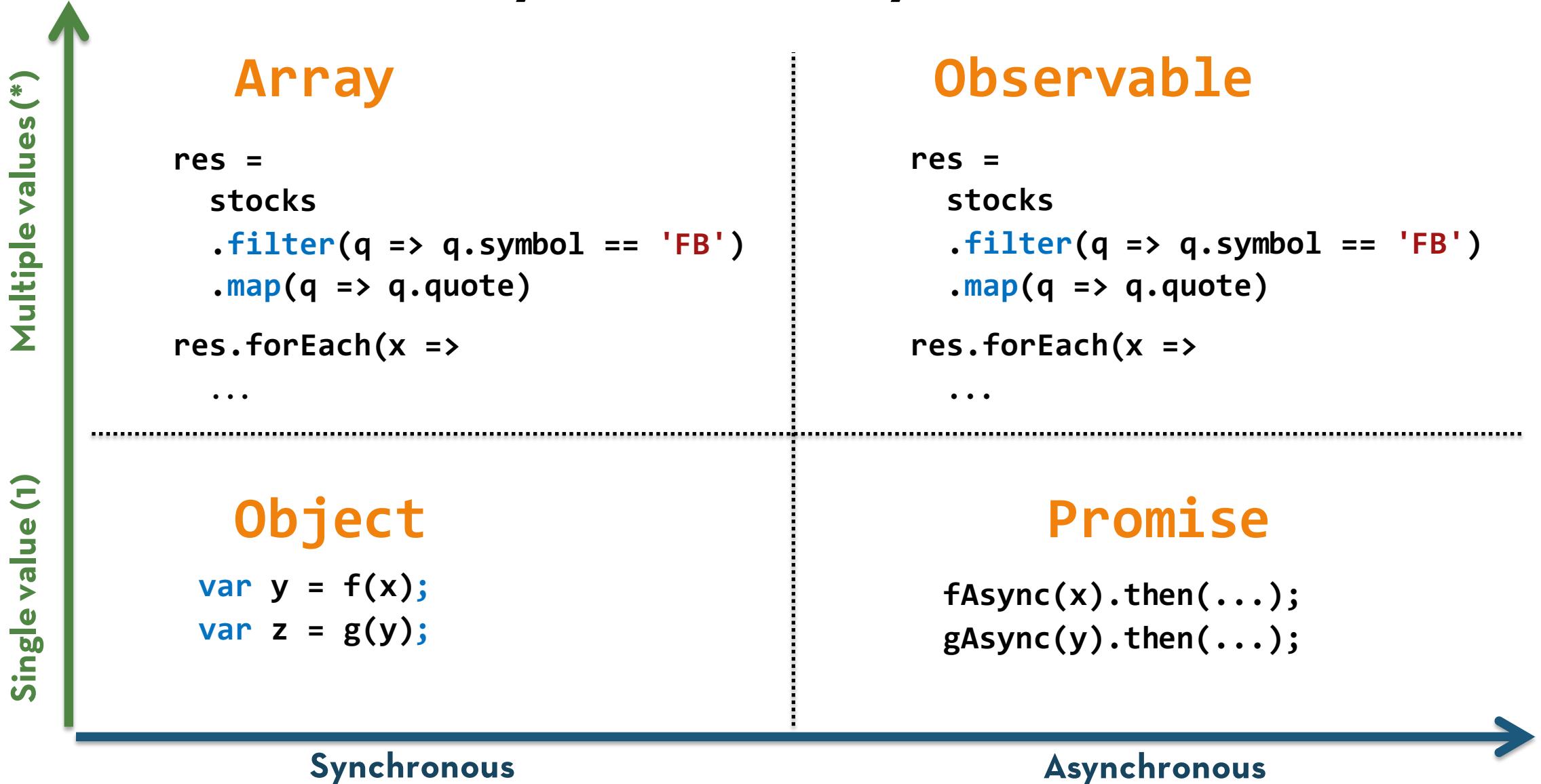
An object is **first-class** when it:^{[4][5]}

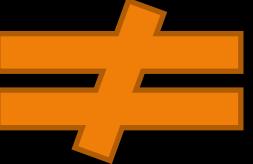
- can be stored in variables and data structures
- can be passed as a parameter to a subroutine
- can be returned as the result of a subroutine
- can be constructed at runtime
- has intrinsic identity (independent of any given name)



WIKIPEDIA
The Free Encyclopedia

The General Theory of Reactivity



Observable 

Object.observe
EventEmitter
EventTarget

An update on Object.observe



Adam Klein (16 days ago)

[Reply](#) [View Original](#) [Go to Source](#) [Edit](#)

Over three years ago, Rafael Weinstein, Erik Arvidsson, and I set out to design and implement what we believed to be the primitive underlying the data-binding system of MDV ("model-driven views"). We prototyped an implementation in a branch of V8, then got agreement from the V8 team to build a real version upstream, while pushing `Object.observe ("O.o")` as a part of the upcoming ES7 standard and working with the Polymer team to build their data-binding system on top of O.o.

Three years later, the world has changed in a variety of ways. While other data-binding frameworks (such as Ember and Angular) showed interest, it was difficult to see how they could evolve their existing model to match that of O.o. Polymer rewrote from the ground up for its 1.0 release, and in that rebuilding did not utilize O.o. And React's processing model, which tries to avoid the mutable state inherent in data-binding systems, has become quite popular on the web.

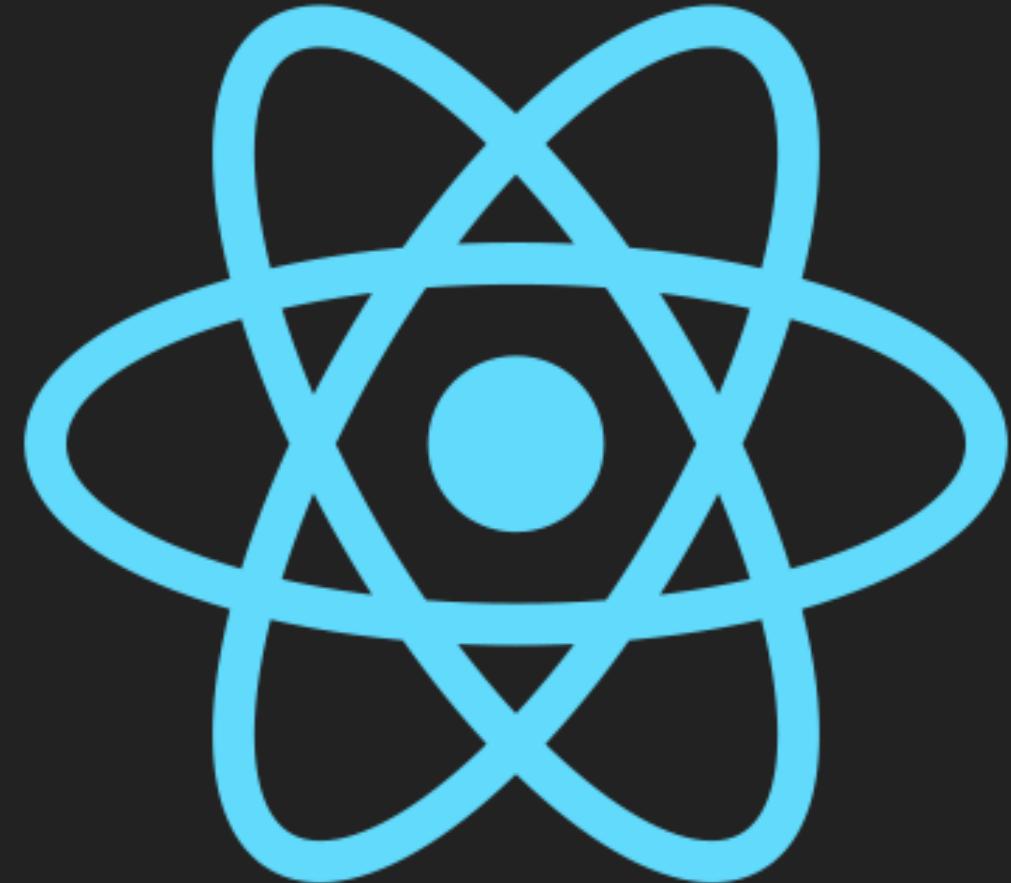
After much discussion with the parties involved, I plan to withdraw the `Object.observe` proposal from TC39 (where it currently sits at stage 2 in the ES spec process), and hope to remove support from V8 by the end of the year (the feature is used on 0.0169% of Chrome pageviews, according to chromestatus.com).

For developers who have been experimenting with O.o and are seeking a transition path, consider using a polyfill such as [MaxArt2501/object-observe](#) or a wrapper library like [polymer/observe-js](#).

<https://esdiscuss.org/topic/an-update-on-object-observe>

Reactive Programming





React

What is Reactive Programming Anyhow?

Merriam-Webster defines reactive as “*readily responsive to a stimulus*”, i.e. its components are “active” and always ready to receive events.

```
$( 'p' ).click(function() {  
  $( this ).slideUp();  
});
```

Clipboard

Font

Alignment

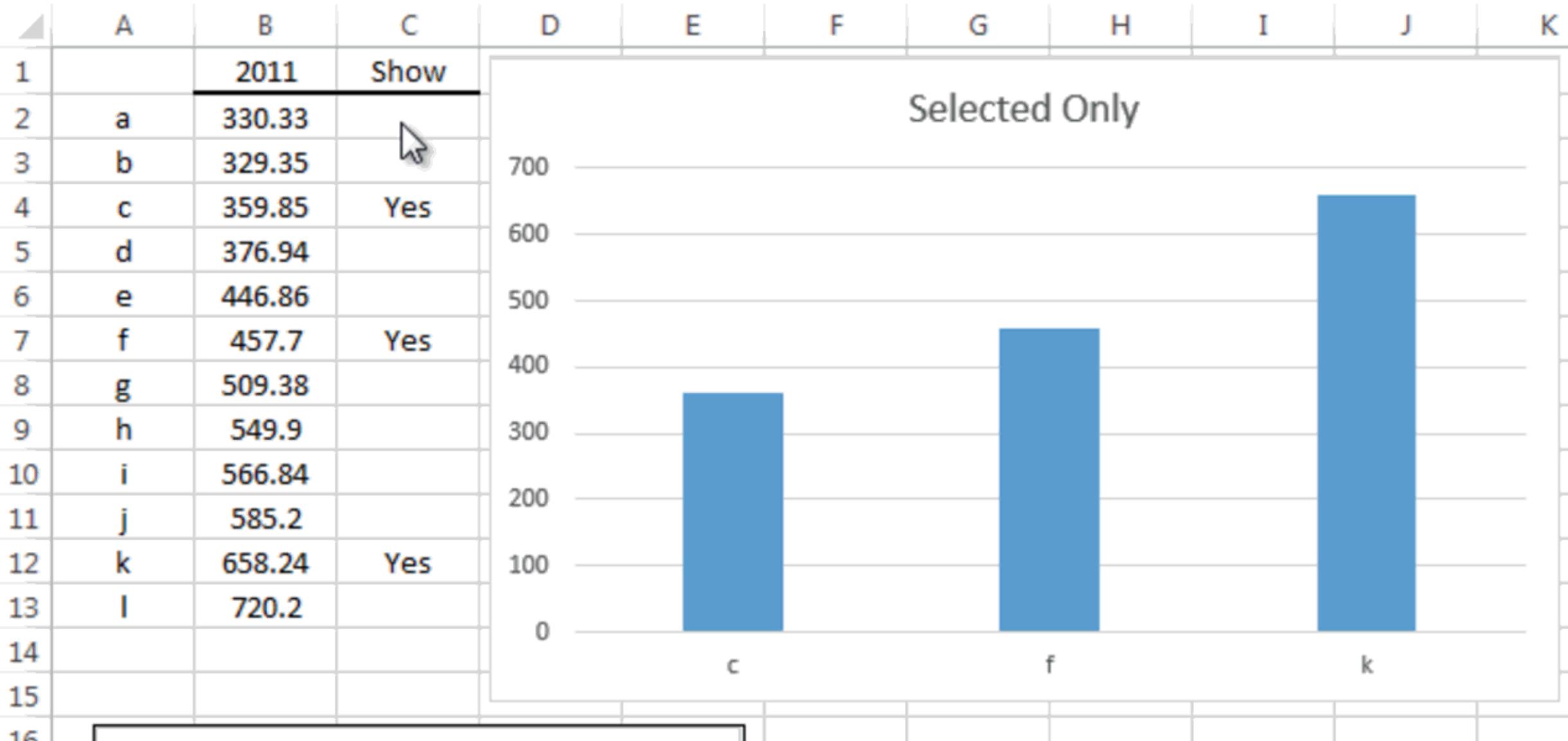
Number

T18

X

✓

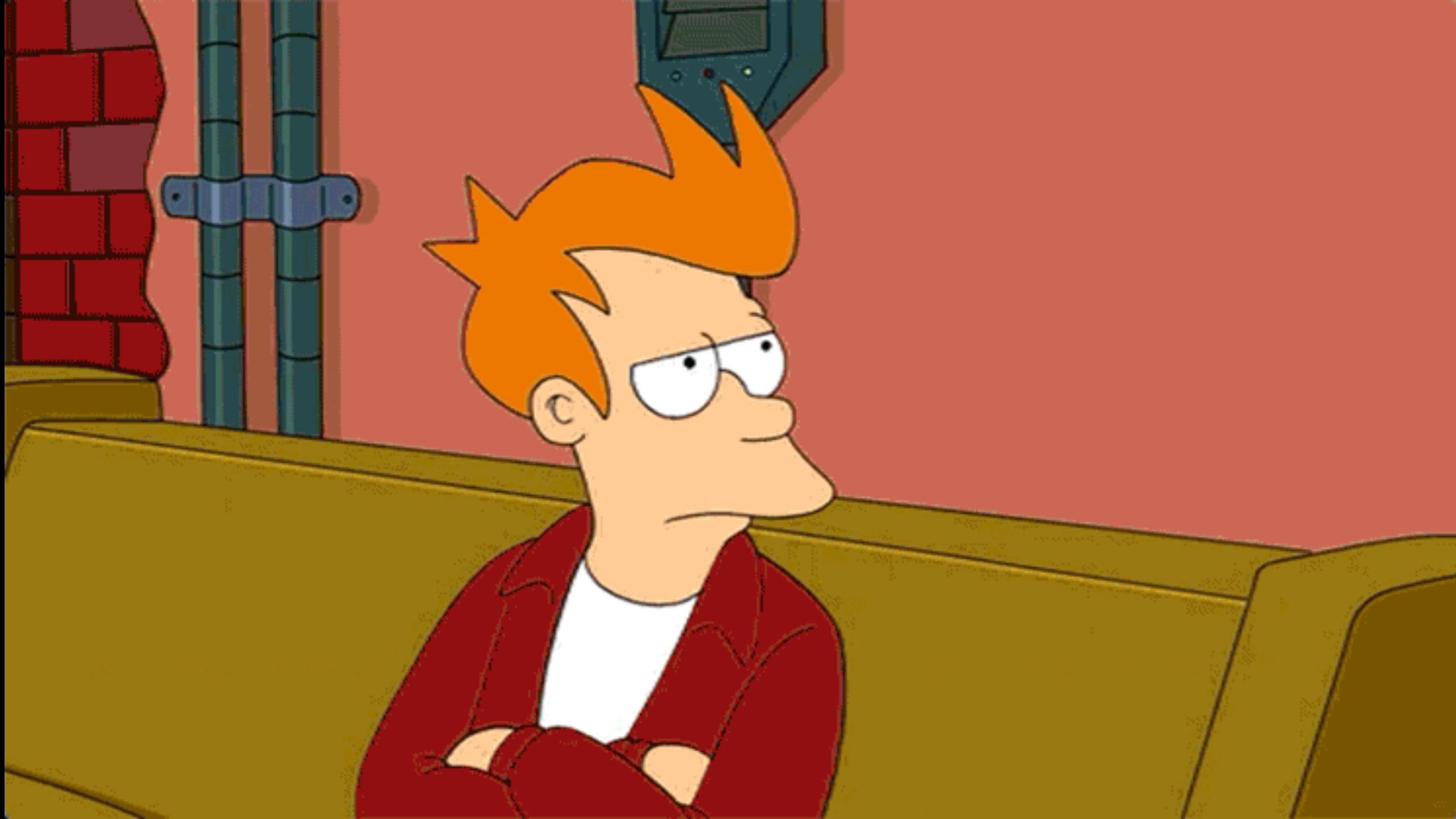
fx

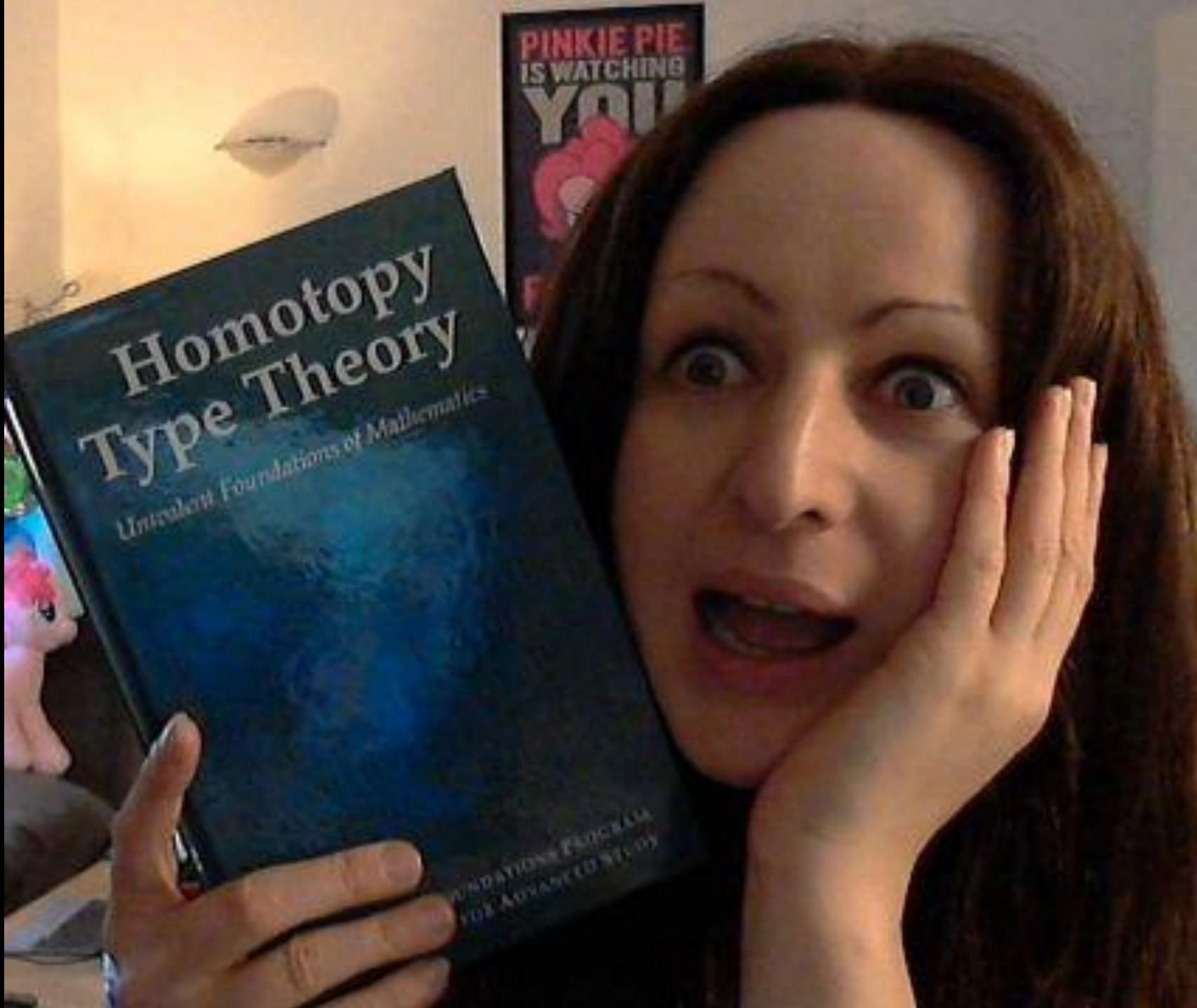


Wanna really know what Reactive Programming Is?

Real Time Programming: Special Purpose or General Purpose Languages
Gerard Berry

<http://bit.ly/reactive-paper>





ComputerWissenschaftAkademischesPapierPhobie

Functional Reactive Programming (FRP) is...

A concept consisting of

- Continuous Time**
- Behaviors: Values over time**
- Events: Discrete phenomena with a value and a time**
- Denotational Semantics**

type Behavior a

$\mu = \text{Behavior } a \rightarrow (\mathbb{R} \rightarrow a)$

Call Us RP, CEP, Compositional Event Processing



Yeah, well, that's just, like,
your opinion, man.

Rx Grammar Police

onNext  *

Zero or more values

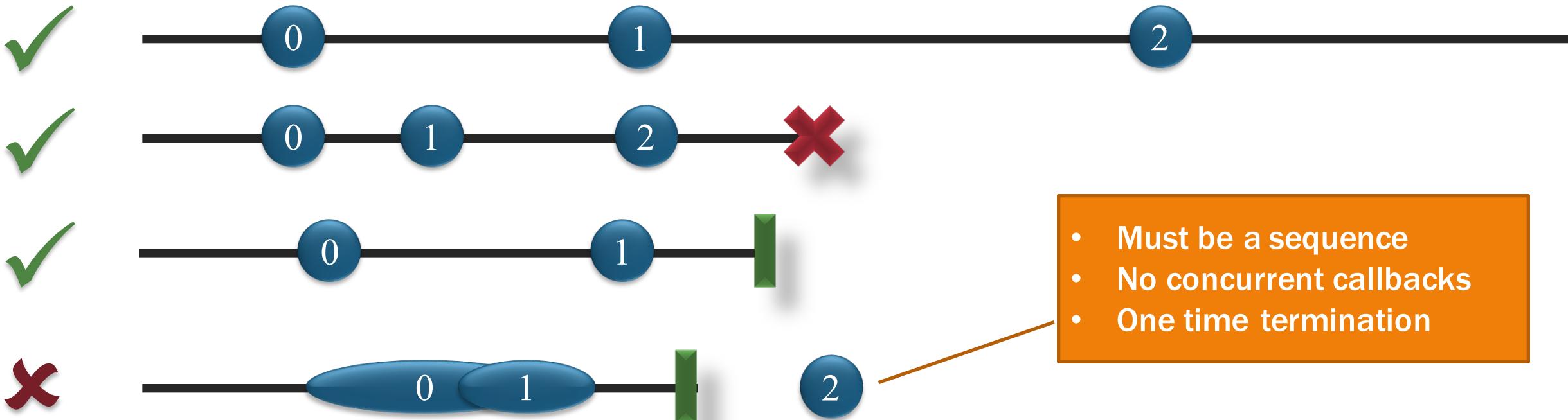
E.g. events are ∞ sequences

(onError 

Calls can fail

onCompleted ) ?

Resource management
Sequencing



Type Ahead Search...

The image shows a mobile application interface. At the top left, there is a red button labeled "MOVIES & TV". To its right is a search bar with a placeholder text "pl" and a magnifying glass icon at the end. Below the search bar, three movie titles are listed: "Prison Break", "Peep Show", and "Pirates of the Caribbean: The Curse of the Black Pearl".

pl

MOVIES & TV

Prison Break

Peep Show

Pirates of the Caribbean: The Curse
of the Black Pearl



Type Ahead Search with Observables

```
let data = dom.keyup(input)
    .map(() => input.value)
    .debounce(500)
    .distinctUntilChanged()
    .flatMapLatest(
        term => search(term)
    );
```

```
let subscription = data.forEach((data) => {
    // Bind data to the UI
});
```

What exactly is Rx?

Language neutral model with 3 concepts:

- 1. Observer/Observable**
- 2. Query operations (map/filter/reduce)**
- 3. How/Where/When**
 - **Schedulers: a set of types to parameterize concurrency**



What exactly is Rx?

Language neutral model with 3 concepts:

1. Observer/Observable
2. Query operations (map/filter/reduce)
3. How/Where/When
 - Schedulers: a set of types to parameterize concurrency



Your Netflix Video Lists

Netflix Row Update Polling

The screenshot shows a Netflix mobile application interface. At the top, there's a red header bar with the word "NETFLIX" on the left and "2 / 10" on the right. Below the header, there's a row of six movie thumbnails: "Band Baaja Baaraat", "The Mystery of the Sphinx", "HEROD'S LOST TOMB", "ARABIA", and "IRONCLAD". To the right of these thumbnails is a movie detail card for "Band Baaja Baaraat". The card includes the title, release year (2010), rating (NR), runtime (2h 19m), and a four-star rating icon. Below the title, the plot summary reads: "Shruti and Bittoo decide to start a wedding planning company together after they graduate from university, but romance gets in the way of business." Further down, the cast (Ranveer Singh, Anushka Sharma) and genre (Comedies, Foreign Movies) are listed, along with the director (Maneesh Sharma). At the bottom of the screen, there are three sections: "Top 10 for tester_jhusain_control" (with thumbnails for "There Will Be Blood", "Band Baaja Baaraat", "BROKEN ENGLISH", "THE HUNTED", and "RAÑÍ"), "Popular on Netflix" (with thumbnails for "The Great Indian Adventure", "The Great Indian Adventure", "The Great Indian Adventure", "The Great Indian Adventure", "The Walking Dead", and "The Great Indian Adventure"), and a small "Discover" section.

NETFLIX

2 / 10

Band Baaja Baaraat

2010 NR 2h 19m

★★★★★

Shruti and Bittoo decide to start a wedding planning company together after they graduate from university, but romance gets in the way of business.

Ranveer Singh, Anushka Sharma

Comedies, Foreign Movies

Director: Maneesh Sharma

Top 10 for tester_jhusain_control

BAND BAAJA BAARAAT

DANIEL DAY-LEWIS There Will Be Blood

Band Baaja Baaraat

BROKEN ENGLISH

THE HUNTED

RAÑÍ

Popular on Netflix

THE WALKING DEAD

Client: Polling for Row Updates

```
function getRowUpdates(row) {  
  let scrolls = Rx.Observable.fromEvent(document, 'scroll');  
  let rowVisibilities =  
    scrolls.debounce(50)  
      .map(scrollEvent => row.isVisible(scrollEvent.offset))  
      .distinctUntilChanged()  
      .share();  
  let [rowShows, rowHides] = rowVisibilities.partition(v => v);  
  
  return rowShows  
    .flatMap(Rx.Observable.interval(10))  
    .flatMap(() => row.getRowData().takeUntil(rowHides))  
    .toArray();  
}
```

Netflix Player



Player Callback Hell

```
function play(movieId, cancelButton, callback) {  
    var movieTicket,  
        playError,  
        tryFinish = function() {  
            if (playError) {  
                callback(null, playError);  
            }  
            else if (movieTicket && player.initialized) {  
                callback(null, ticket);  
            }  
        };  
    cancelButton.addEventListener("click", function() { playError = "cancel"; });  
    if (!player.initialized) {  
        player.init(function(error) {  
            playError = error;  
            tryFinish();  
        })  
    }  
    authorizeMovie(movieId, function(error, ticket) {  
        playError = error;  
        movieTicket = ticket;  
        tryFinish();  
    });  
});
```



Player With Observables

```
let authorizations =  
    player.init().flatMap(() =>  
        playAttempts.flatMap( movieId =>  
            player.authorize(movieId)  
                .retry(3)  
                .takeUntil(cancels))  
    )  
);  
let subscription = authorizations.forEach(  
    license => player.play(license),  
    error => showDialog('Sorry, can't play right now.'));
```





Implementing Spell Check in Slack

```
let userStoppedTyping = inputEvent  
.concatMap(() => Rx.Observable.just(true).concat(Rx.Observable.never()))  
.takeUntil(inputEvent.debounce(750))  
.repeat()  
.startWith(true);
```

```
let currentKeyboardLanguage = userStoppedTyping  
.map(() => this.getCurrentKeyboardLanguage())  
.distinctUntilChanged()  
.merge(this.overrideKeyboardLanguage)  
.distinctUntilChanged();
```

```
let subscription = currentKeyboardLanguage.subscribe( lang => {
```





A close-up photograph of a person's face, showing their eyes and nose, with a slight smile. The background is blurred, suggesting an indoor setting.

```
try {
  let resource = null;
  for (let i = 0; i < 10; i++) {
    process.nextTick(() => {
      if (resource) {
        resource.close();
        resource = null;
      }
    });
  }
} catch (e) {
  throw e;
} finally {
  resource?.close();
}
```

It's the final countdown

Observables and Backpressure

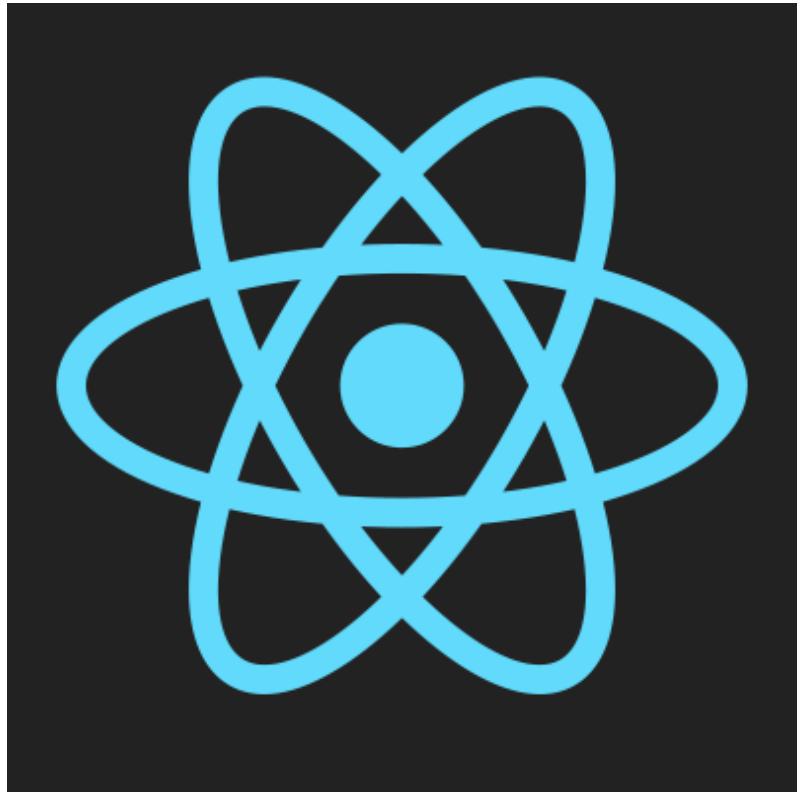
Yes, Observables can have backpressure

- Can be lossy (pausable, sample, throttle)
- Can be lossless (buffer, pausableBuffered, controlled)

```
let pausable = chattyObservable.pausableBuffered();
pausable.pause();
pausable.resume();
```

```
let controlled = chattyObservable.controlled();
controlled.request(10);
```

Winning Friends And Influences Others





angular / angular

Watch ▾ 585

Star 3,617

Fork 909

feat(http): add basic http service

[Browse files](#)

This implementation only works in JavaScript, while the Observable transpilation story gets worked out. Right now, the service just makes a simple request, and returns an Observable of Response.

Additional functionality will be captured in separate issues.

Fixes [#2028](#)

master (#4)

jeffbcross authored on Apr 29

1 parent 363b9ba commit 21568106b114943b59ce37832d82c8fb9a63285b

Showing 35 changed files with 1,054 additions and 2 deletions.

[Unified](#) [Split](#)

<https://github.com/angular/angular/commit/21568106b114943b59ce37832d82c8fb9a63285b>

Angular 2 and RxJS, perfect together?

Data Libraries

Select any of the following data-related libraries that you will likely utilize on a majority of your Angular 2 projects: (choose multiple)

RxJS 38.7% 522

Immutable.js 34.7% 469

Firebase 33.3% 449

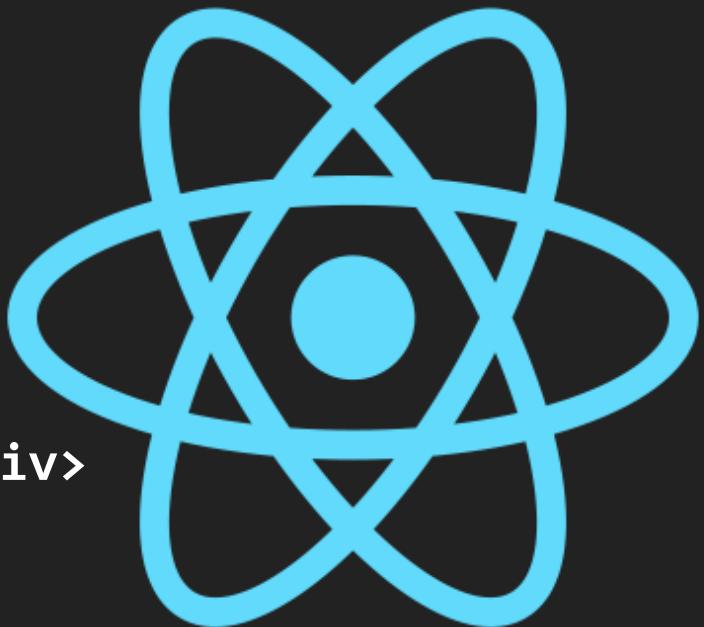
angular-meteor 23.8% 321

Falcor 14.5% 196

Other 9.1% 123

BaconJS 6.6% 89

```
const Timer = React.createClass({
  getInitialState: () => {
    return {secondsElapsed: 0};
  },
  componentDidMount: () => {
    this.subscription = Rx.Observable.interval(1000)
      .subscribe(x => this.setState({secondsElapsed: x}));
  },
  componentWillUnmount: () => {
    this.subscription.dispose();
  },
  render: () => {
    return (
      <div>Seconds Elapsed: {this.state.secondsElapsed}</div>
    );
  }
});
```



What is Rx?

Language neutral model with 3 concepts:

- 1. Observer/Observable**
- 2. Query operations (map/filter/reduce)**
- 3. How/Where/When**
 - Schedulers: a set of types to parameterize concurrency





Schedulers...

When?

Where?

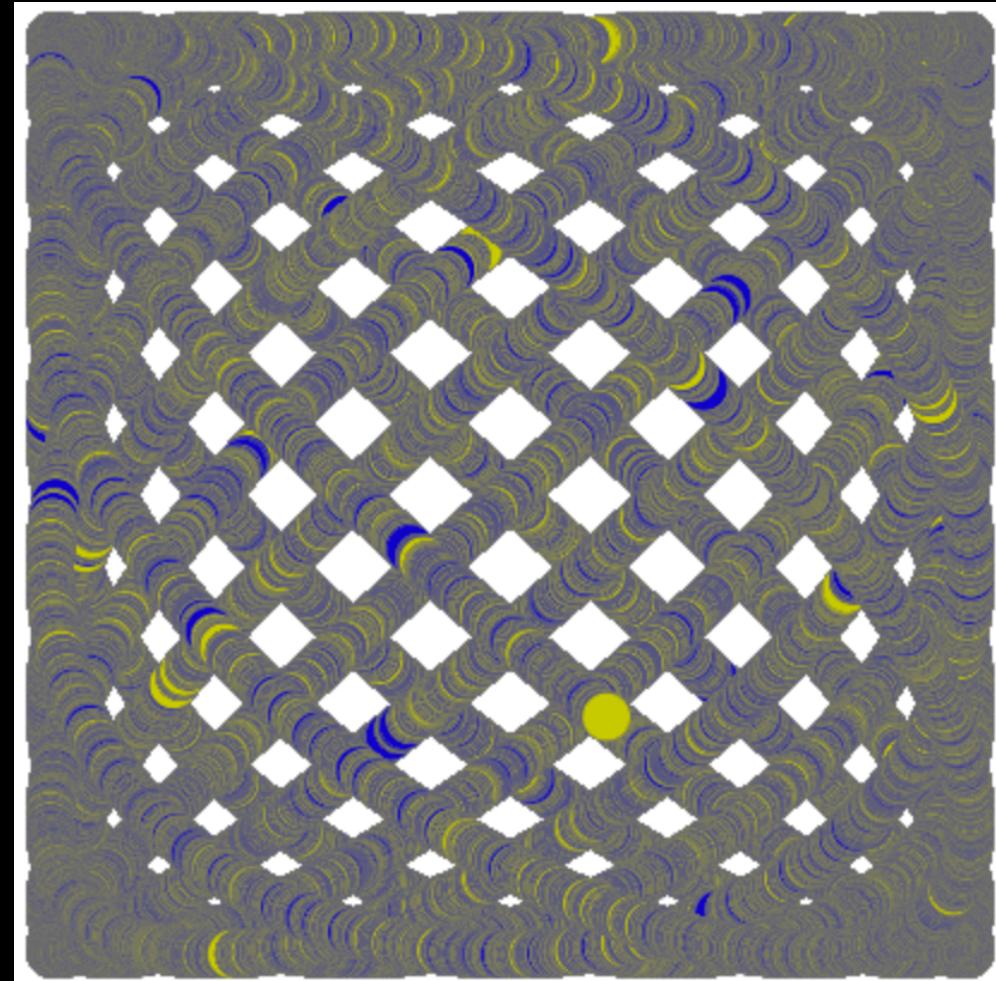
How?

```
let d = scheduler.schedule(state, () => {  
    // Do work async  
}, 1000);  
  
d.dispose();
```

Schedulers Matter!

```
const scheduler = Scheduler.default;
```

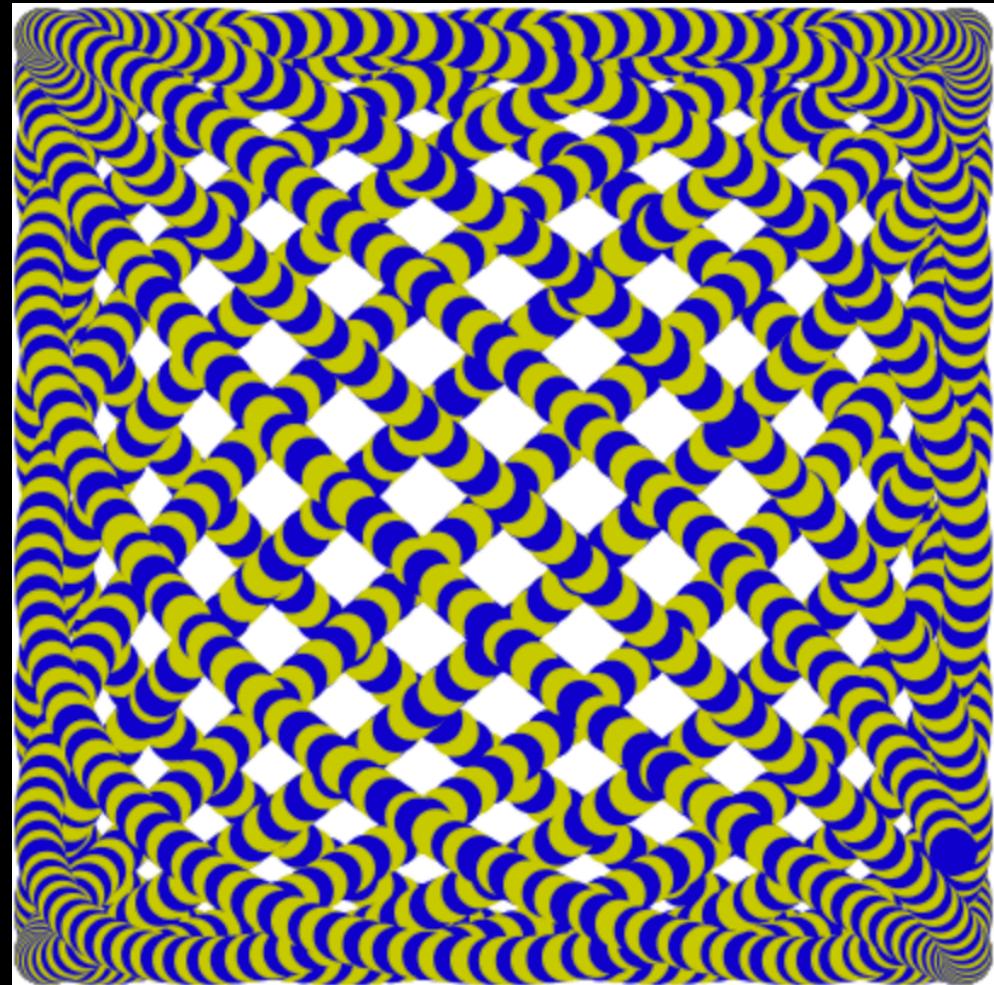
```
const subscription = Observable
  .generate(
    0,
    x => true,
    x => x + 1,
    x => x,
    scheduler
  )
  .timestamp()
  .subscribe(draw);
```



No, Really, They Do!

```
const scheduler = Scheduler.requestAnimationFrame;
```

```
const subscription = Observable
  .generate(
    0,
    x => true,
    x => x + 1,
    x => x,
    scheduler
  )
  .timestamp()
  .subscribe(draw);
```



Ur Kitteh of Death

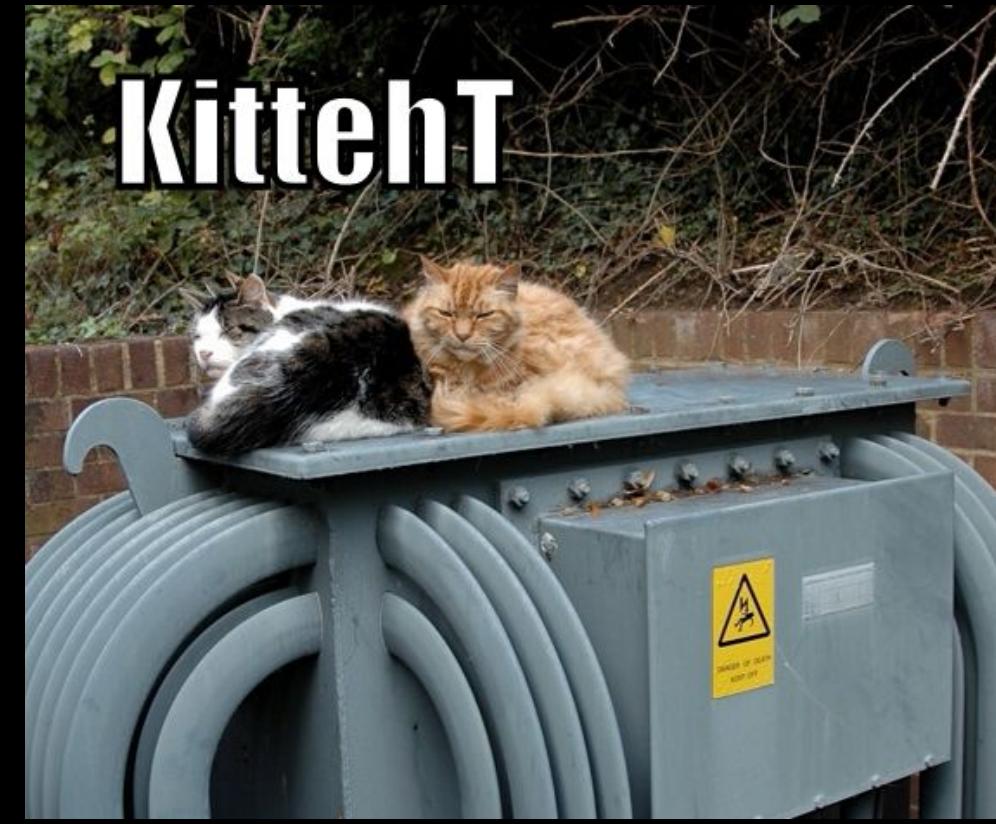
Awaits

ROFLBOT

ncawait

Async/Await with Observables and Generators...

```
Rx.Observable.spawn(function* () {  
  const result = yield get('http://jsconf.asia')  
  .retry(3)  
  .catch(cachedVersion)  
  .finally(doCleanup);  
  
  return result;  
}).subscribe(::console.log);
```



Observables Coming to ES2016?!?!

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.
- *Integrated with ES6*: Data is sent to consumers using the ES6 generator interface.

The **Observable** concept comes from *reactive programming*. See <http://reactivex.io/> for more information.

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(sink => {
        // Create an event handler which sends data to the sink
        let handler = event => sink.next(event);

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

<https://github.com/zenparsing/es-observable>

Observables in ES2016

```
function commandKeys(element) {
  const keyCommands = { "38": "up", "40": "down" };

  return listen(element, "keydown")
    .filter(event => event.keyCode in keyCommands)
    .map(event => keyCommands[event.keyCode])
}

const subscription = commandKeys(inputElement).subscribe({
  next(val) { console.log("Received key command: " + val) },
  error(err) { console.log("Received an error: " + err) },
  complete() { console.log("Stream complete") },
});

subscription.unsubscribe();
```



<http://github.com/Reactive-Extensions/RxJS>

<http://github.com/ReactiveX/RxJS>

The Future of RxJS...

