

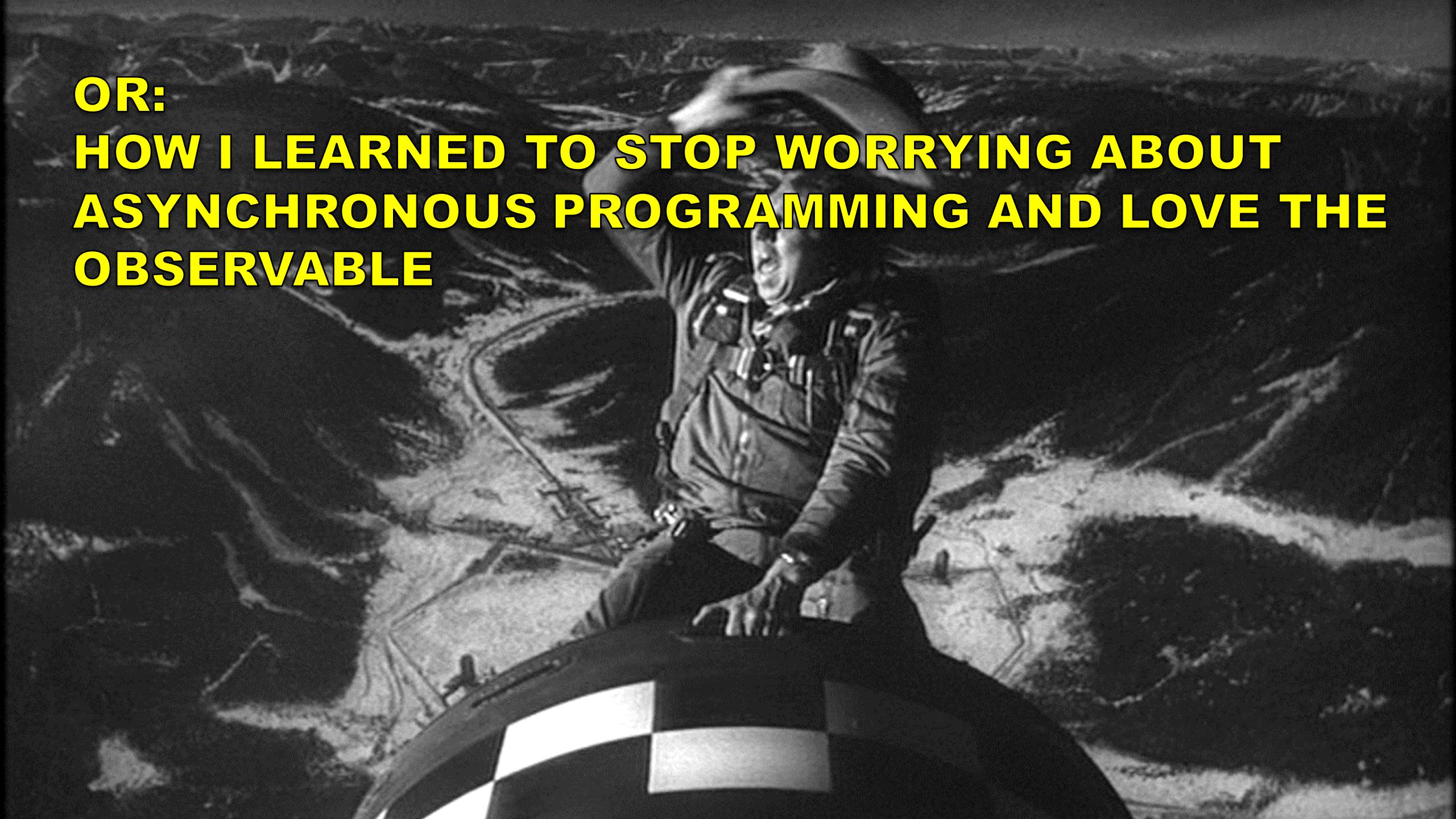
PROGRAMMING

You're Doing It Completely Wrong.

Async and Streaming JS Are We doing it wrong?

Matthew Podwysocki @mattpodwysocki

github.com/mattpodwysocki/ndc-oslo-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 callbacks and events!".
have Now problems. two I**



**Principal Open Source Engineer
Open Sourcerer
@mattpodwysocki
github.com/mattpodwysocki**



The Microsoft logo consists of the word "MICROSOFT" in a bold, black, sans-serif font. The letters are slanted slightly to the right. Above the letter "I", there are two small black dots. Above the letter "O", there are three black dots arranged horizontally.



Reactive Extensions (Rx)

@ReactiveX
<http://reactivex.io>

Real-Time is Everywhere...



SPIKE & SPARKY

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





culturepub.fr

next
ad ➔

Events and the Enemy of the State

```
var isDown = false, state;

function mousedown (e) {
  isDown = true;
  state = { startX: e.offsetX,
            startY: e.offsetY };
}

function mousemove (e) {
  if (!isDown) { return; }
  var delta = { endX: e.clientX - state.startX,
                endY: e.clientY - state.startY };
  // Now do something with it
}

function mouseup (e) {
  isDown = false;
  state = null;
}
```

```
function dispose() {
  elem.removeEventListener('mousedown', mousedown, false);
  elem.removeEventListener('mouseup', mouseup, false);
  doc.removeEventListener('mousemove', mousemove, false);
}

elem.addEventListener('mousedown', mousedown, false);
elem.addEventListener('mouseup', mouseup, false);
doc.addEventListener('mousemove', mousemove, false);
```





First Class Async with Promises

then

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

Breaking the Promise...

then

```
var 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



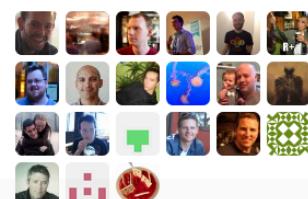
jakearchibald

Notifications

[Subscribe](#)

You're not receiving notifications from this thread.

21 participants



The Final Countdown...

then

```
var resource;
try {
  var resource = getResource();
} catch (e) {
  throw e;
} finally {
  resource && resource.dispose();
}

getresource()
  .success(
    function (resource) {
      })
  .catch(
    function (err) {
      });
}

// How do I clean up my resource?
```

then



MAKE GIFS AT GIFOUP.COM



UNSAFE AT ANY SPEED

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



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





WHATWG Streams

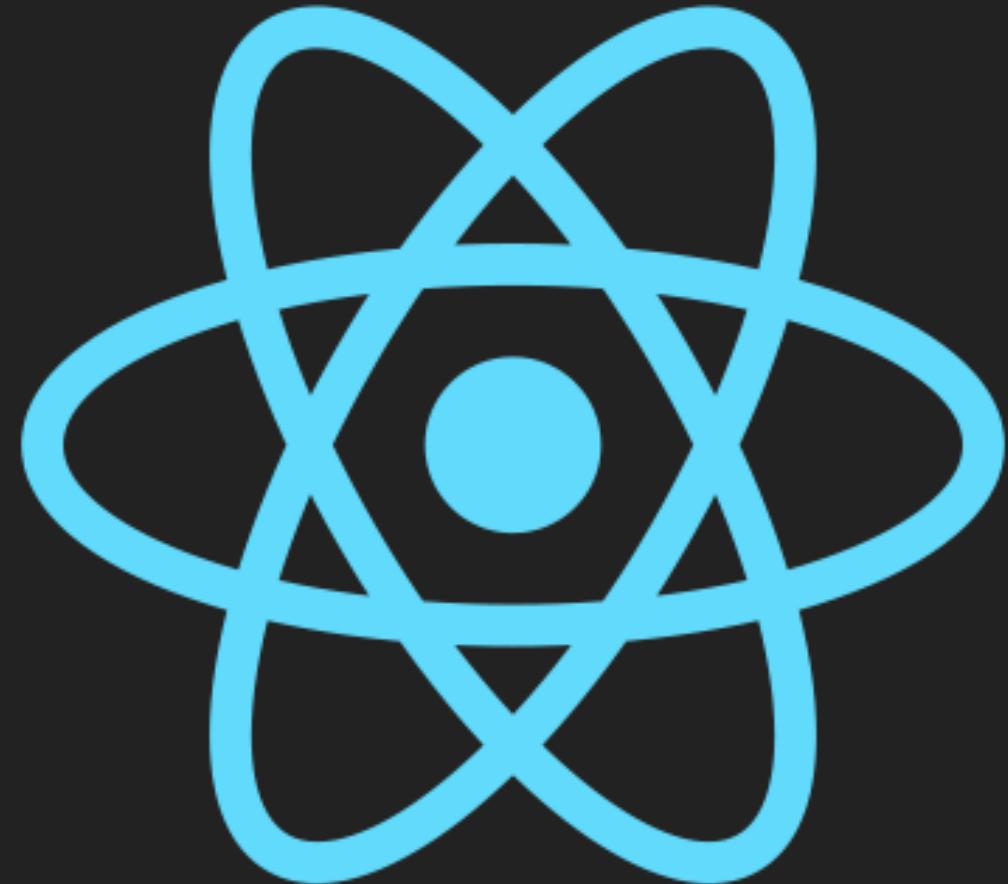
Specification for creating, composing and consuming streams of data

- Focused on low-level I/O, not on object mode
- Think Node.js Streams + Promises

```
readableStream.pipeTo(writableStream)
  .then(() => console.log("All data successfully written!"))
  .catch(e => console.error("Something went wrong!", e));
```

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

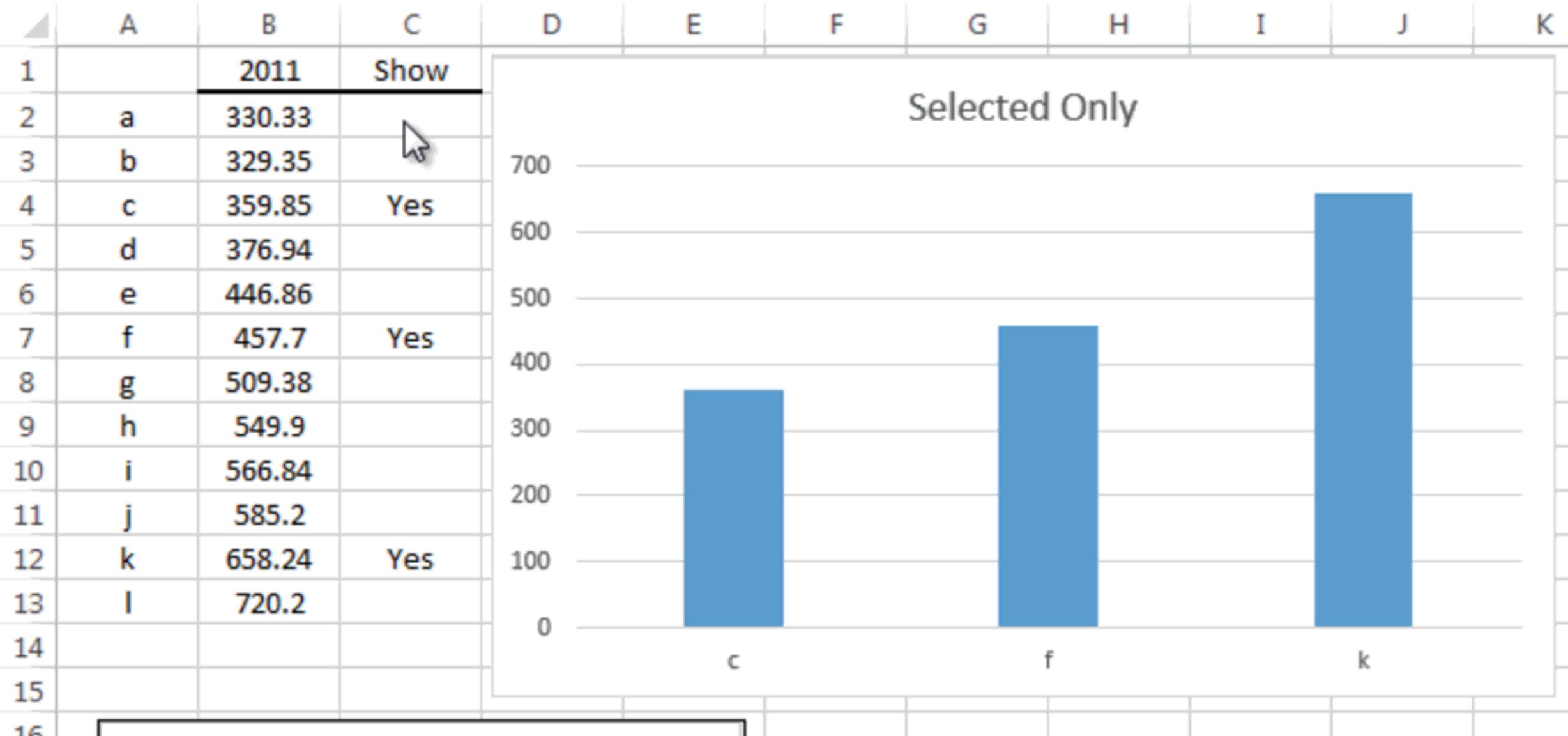
Number

T18

:



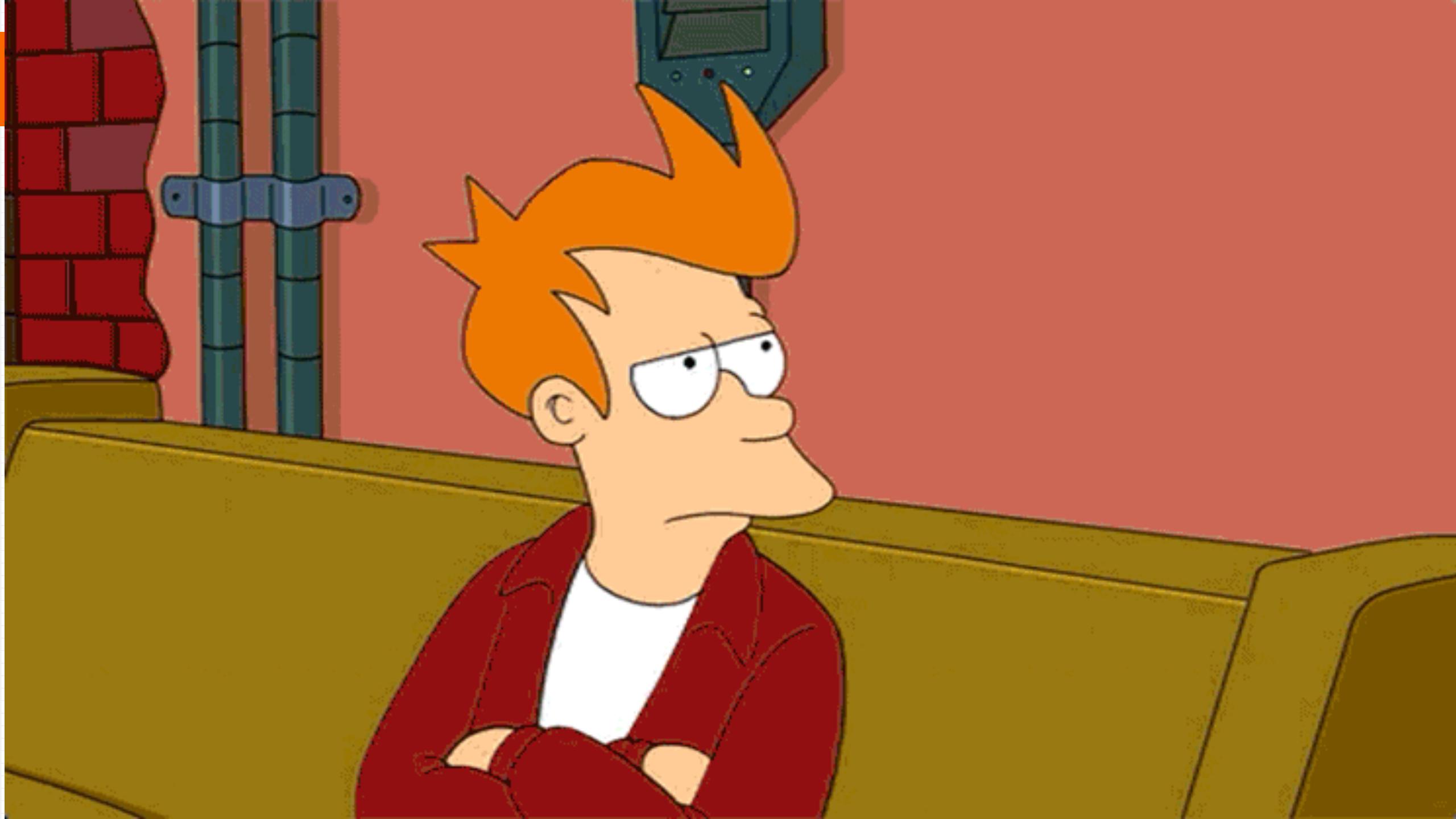
fx

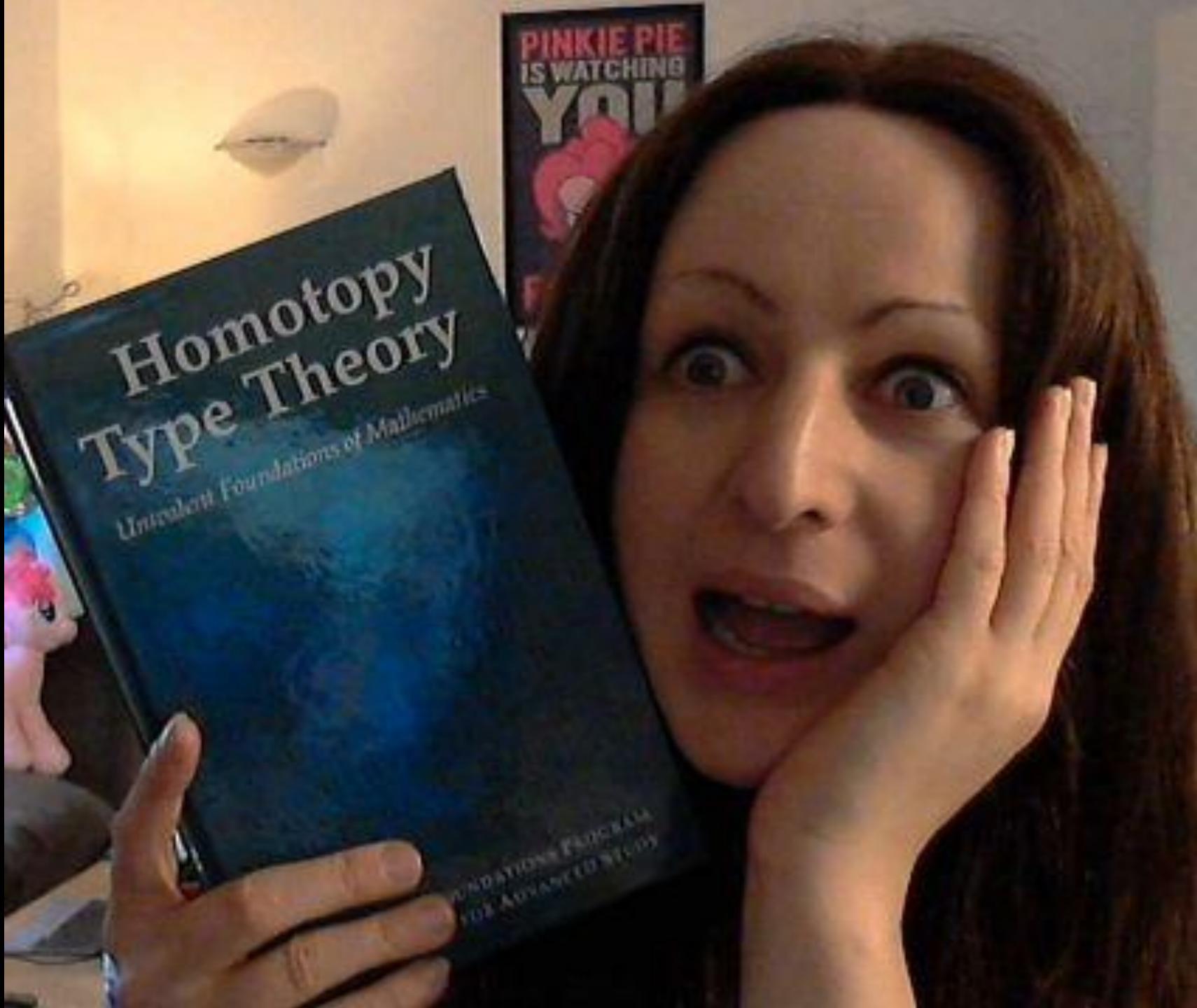


Wanna really know what Reactive Programming Is?

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

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





ComputerWissenschaftAkademischesPapierPhobie

Ok, maybe this is better...

 **staltz / introrx.md**
Last active 37 minutes ago

★ Star 5,397 fork 513 !

The introduction to Reactive Programming you've been missing

introrx.md Raw

The introduction to Reactive Programming you've been missing

(by [@andrestaltz](#))

So you're curious in learning this new thing called Reactive Programming, particularly its variant comprising of Rx, Bacon.js, RAC, and others.

Learning it is hard, even harder by the lack of good material. When I started, I tried looking for tutorials. I found only a handful of practical guides, but they just scratched the surface and never tackled the challenge of building the whole architecture around it. Library documentations often don't help when you're trying to understand some function. I mean, honestly, look at this:

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

Projects each element of an observable sequence into a new sequence of observable sequences

Code

- Revisions 257
- Stars 5397
- Forks 513

Embed URL
`<script src="https://gist.github.com/staltz/868e7e9bc2a7b8c1f754">`

HTTPS clone URL
`https://gist.github.com/staltz/868e7e9bc2a7b8c1f754`

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop Download ZIP

<http://https://gist.github.com/staltz/868e7e9bc2a7b8c1f754>

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

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

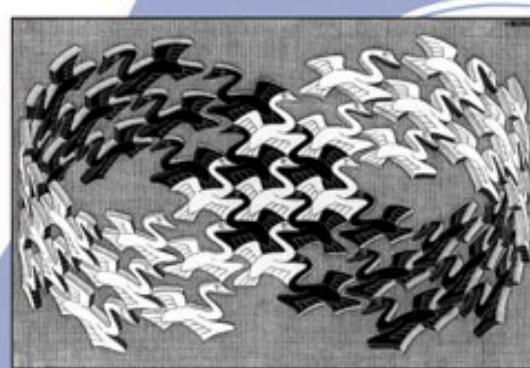
1994



Design Patterns

Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

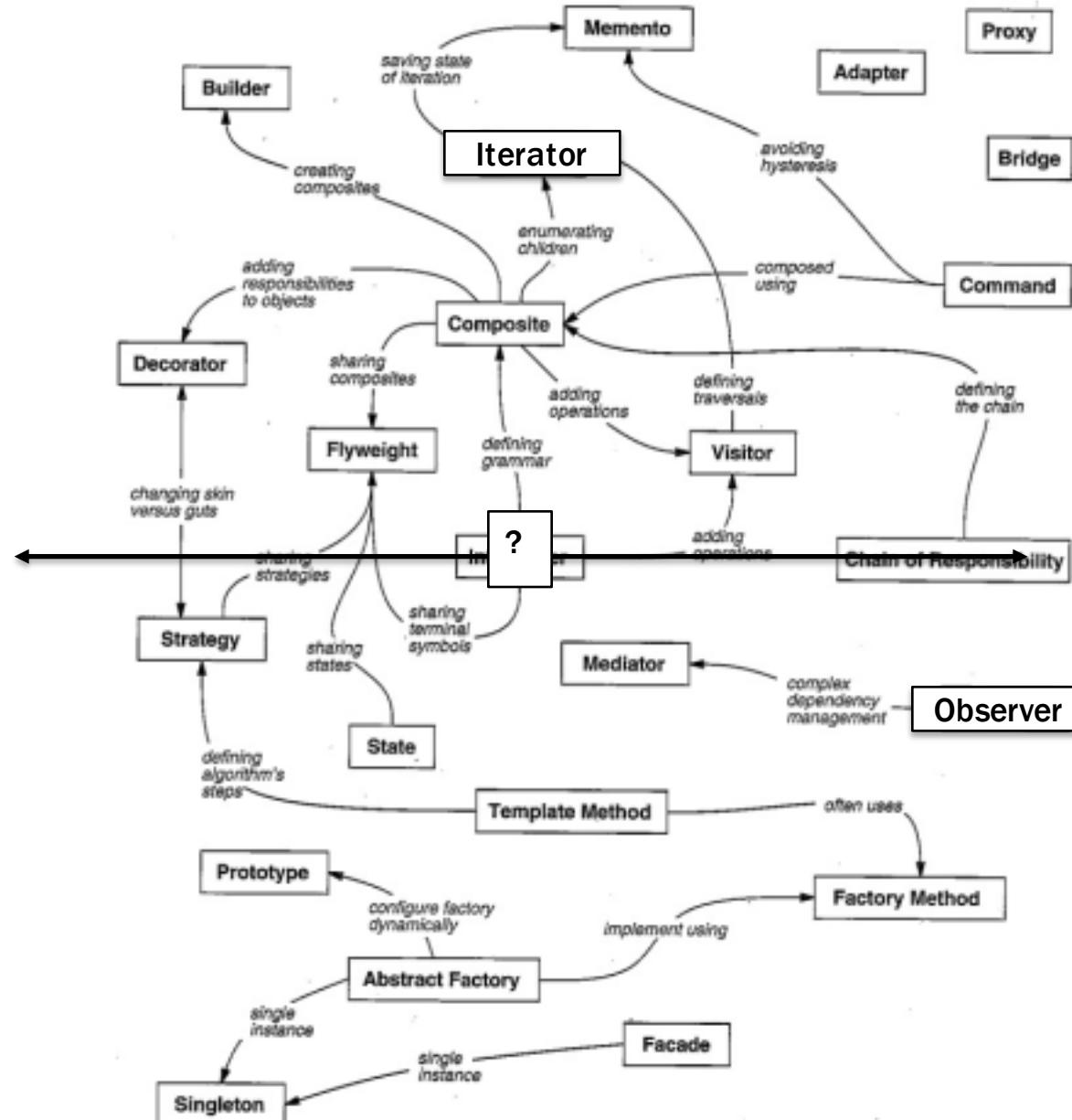


Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

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 }
```



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

Foreword by Grady Booch



Fundamental Abstractions

Gang of Four Subject/Observer Pattern

- Notifies Observers on State Change
- Not Very Compositional

```
function handler (e) {  
    // Handle the event  
}
```

```
eventTarget.addEventListener(event, handler, false);
```

```
eventTarget.removeEventListener(event, handler, false);
```



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

Foreword by Grady Booch



Fundamental Abstractions

Adapting the Subject/Observer Pattern

- Ensuring duality with the enumerator pattern
- More Compositional Approach

```
interface Observable<T> {  
    subscribe(observer : Observer) : Disposable  
}
```

```
interface Observer<T> {  
    onNext(value : T) : void  
    onError(error : Error) : void  
    onCompleted() : void  
}
```

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

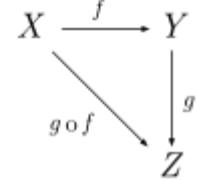
[{x: 23, y: 44}, {x:27, y:55}, {x:27, y:55}]



... and an Event?

**Events and Arrays are *both*
collections.**

Category Theory to the Rescue!



Category theory
www.Wikipedia.org

Adapting the Subject/Observer Pattern

- Observable/Observer (push) is dual to Iterable/Iterator (pull)
- Cross-influence from both domains

```
interface Observable<T> {  
    subscribe(observer : Observer) : Disposable  
}
```

```
interface Observer<T> {  
    onNext(value : T) : void  
    onError(error : Error) : void  
    onCompleted() : void  
}
```

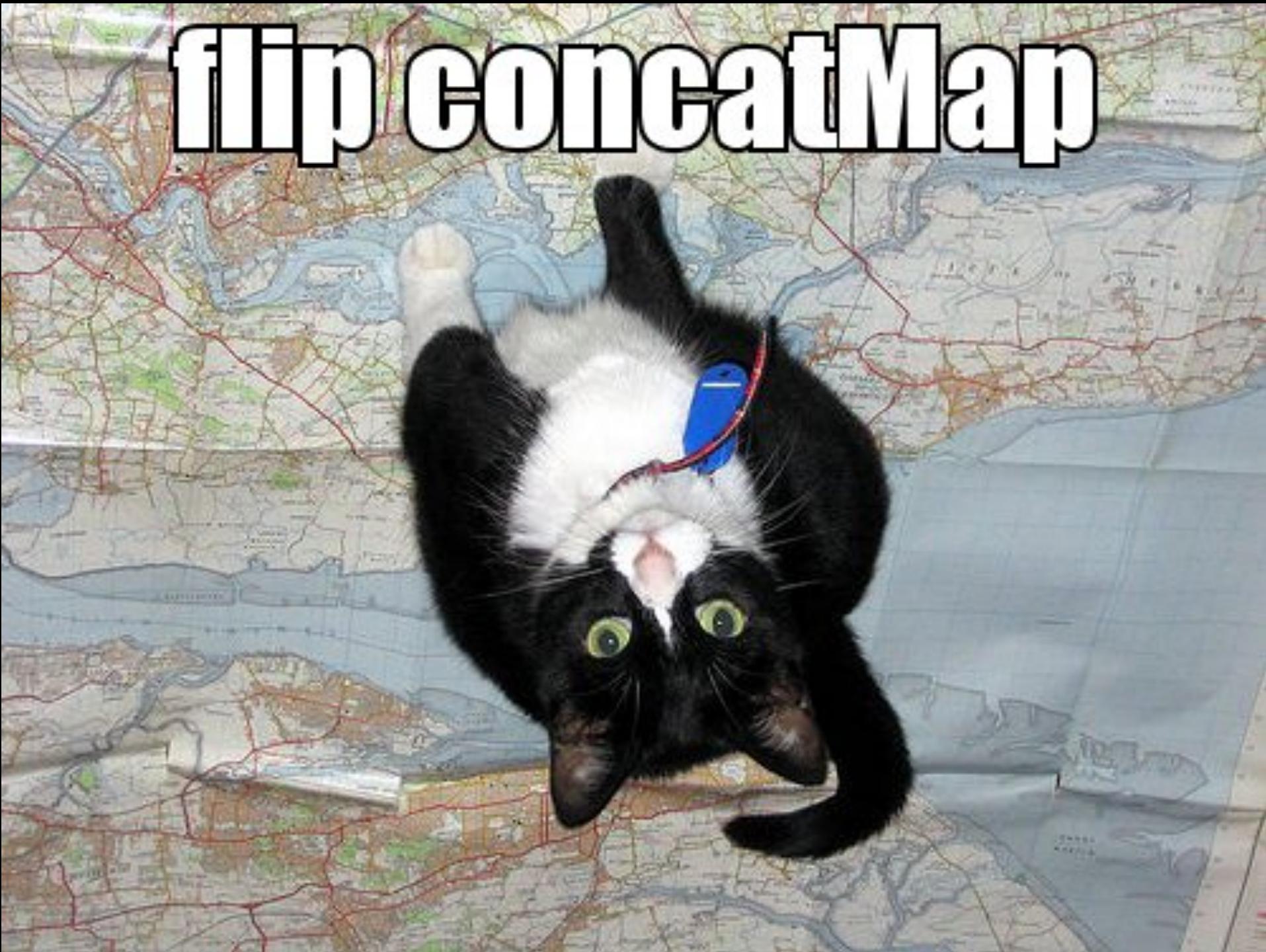
```
interface Iterable<T> {  
    [Symbol.iterator](): Iterator<T>  
}
```

```
interface Iterator<T> {  
    next() : IteratorResult<T> // throws  
}
```

```
interface IterationResult<T> {  
    done: boolean  
    value: T  
}
```

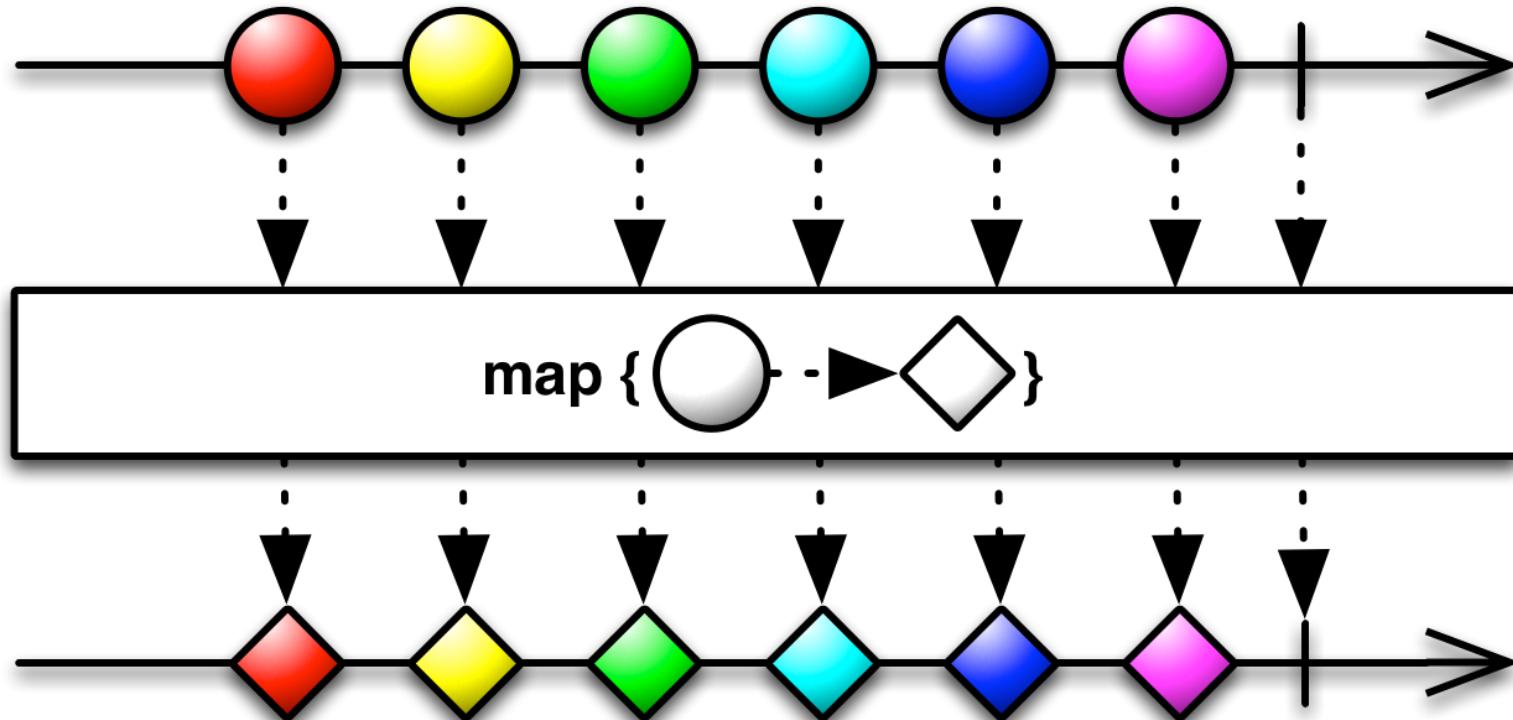
The majority of your asynchronous
code is written with just a few
flexible functions.

flip concatMap



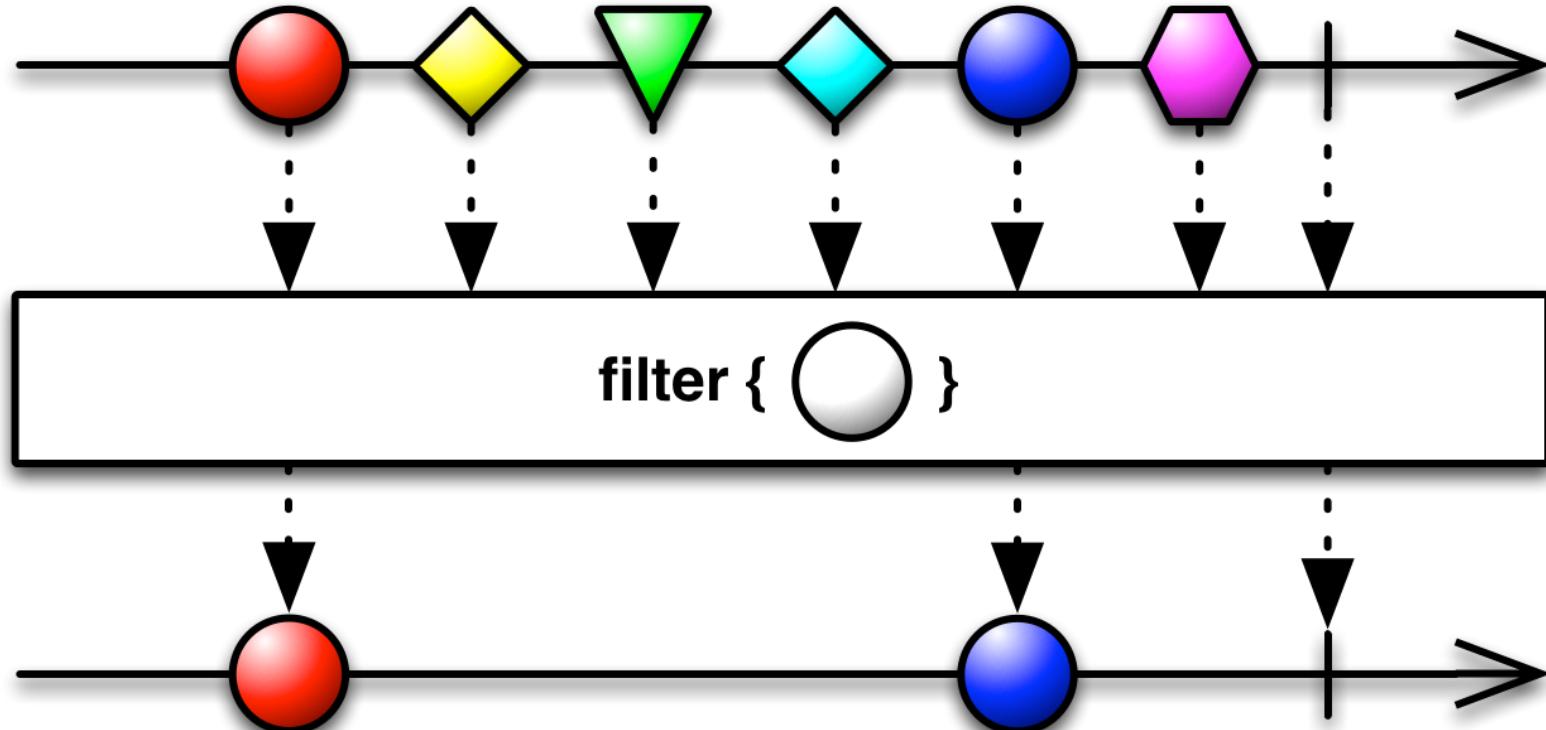
map()

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



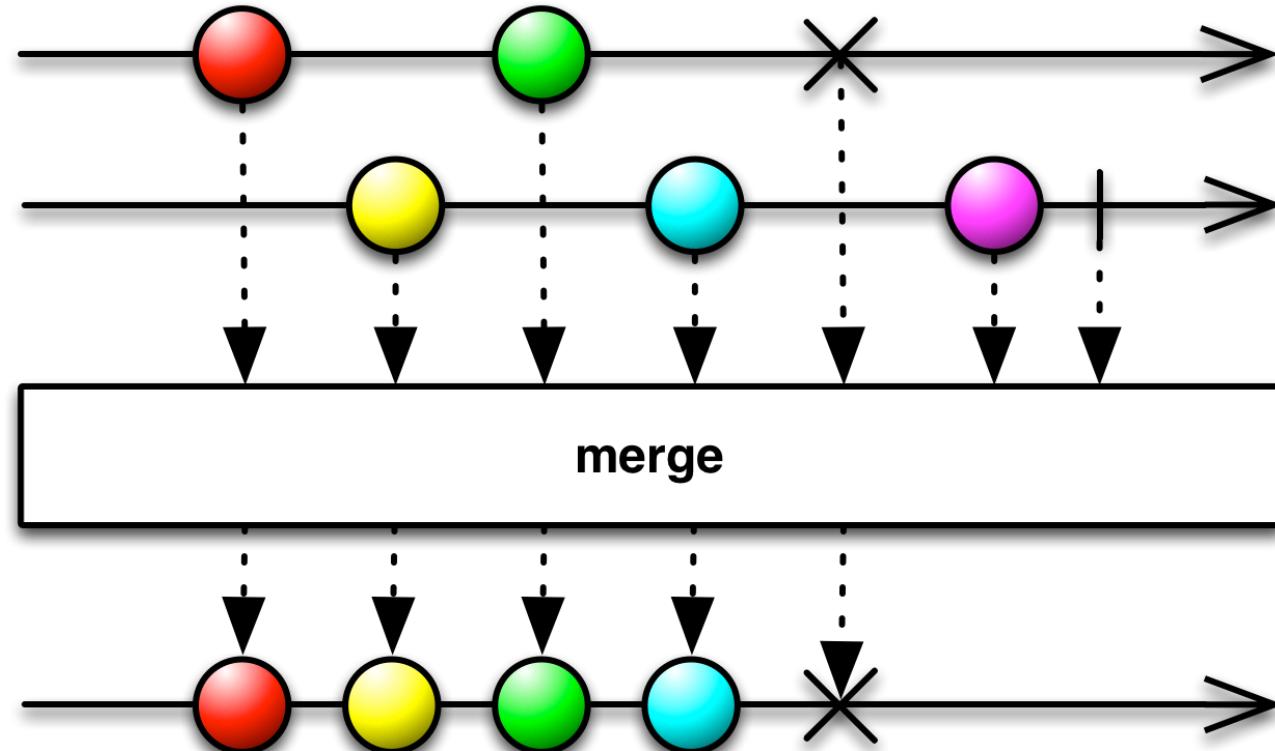
filter()

Filter items emitted by a Collection



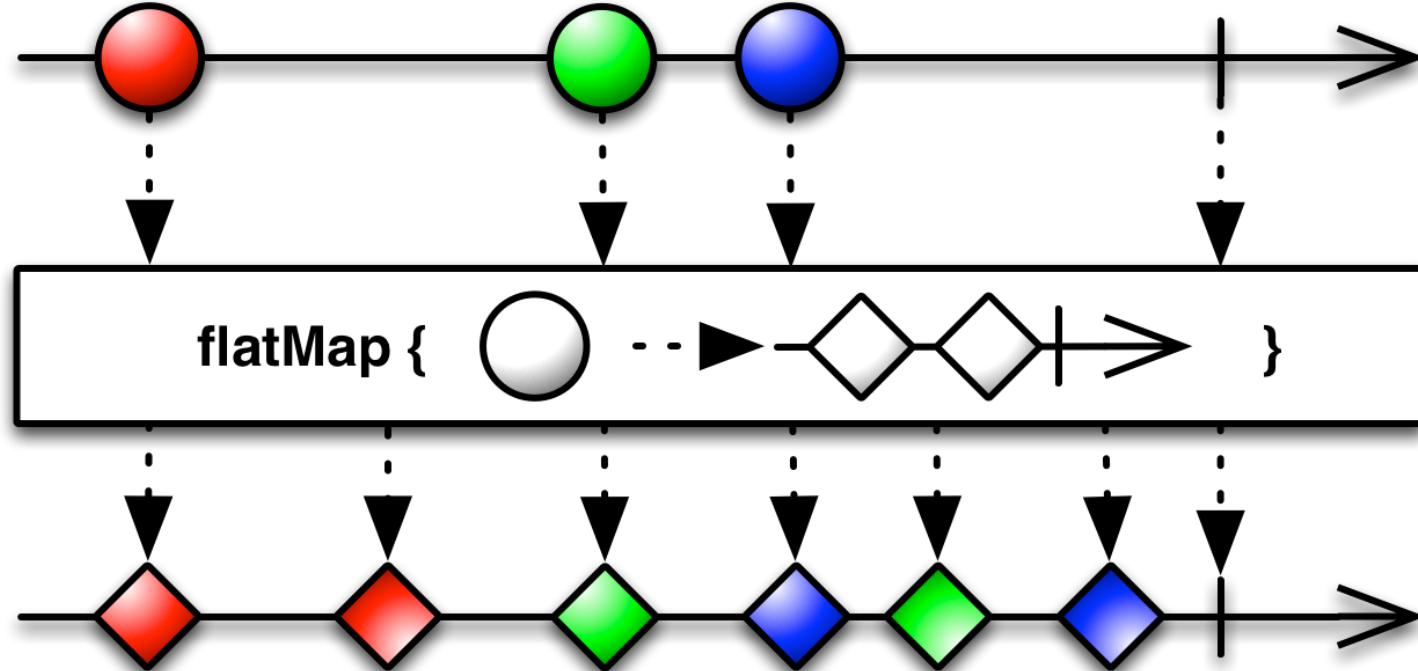
mergeAll()

combine multiple Collections into one by merging their emissions



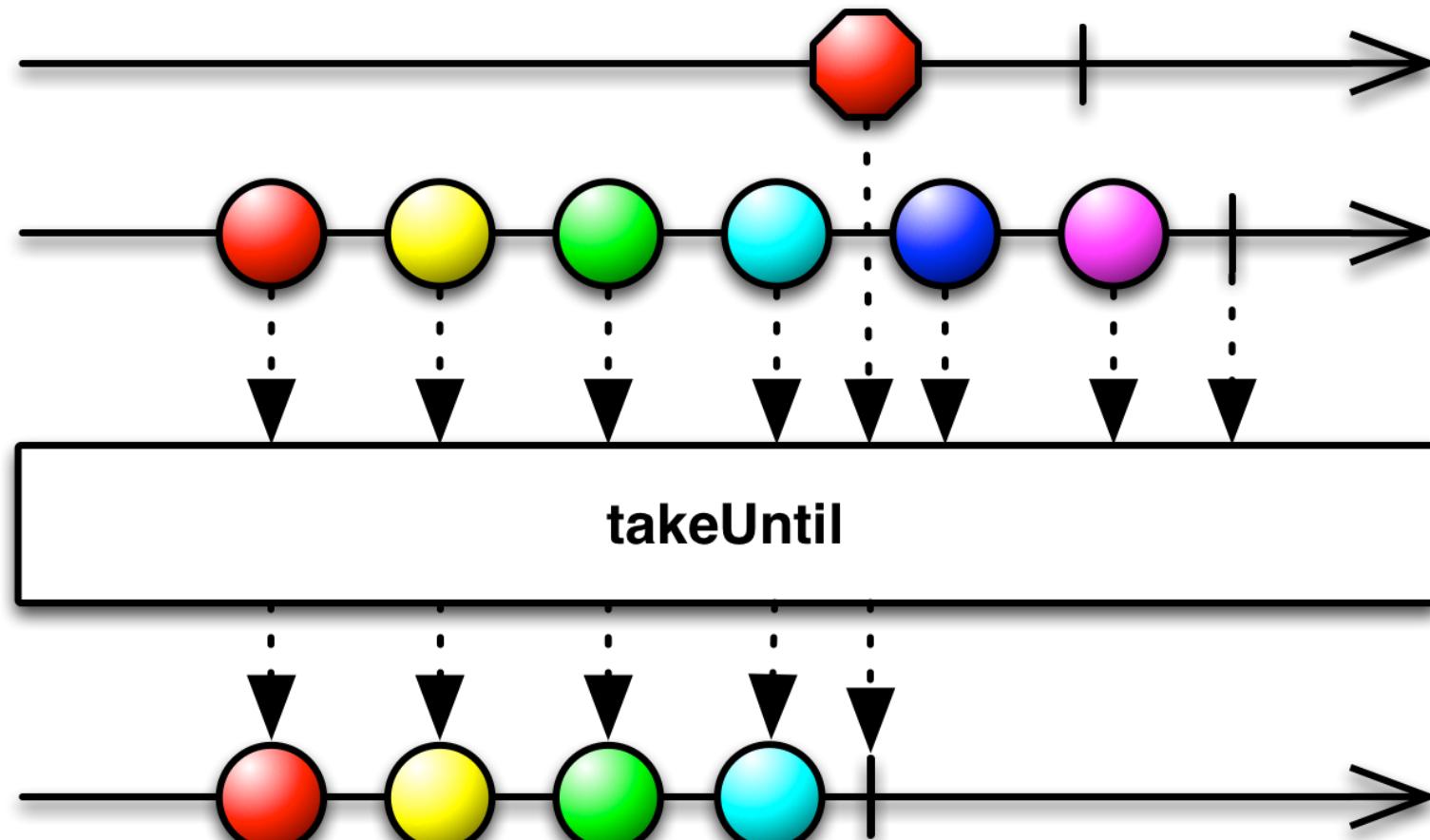
flatMap()

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



`takeUntil()`

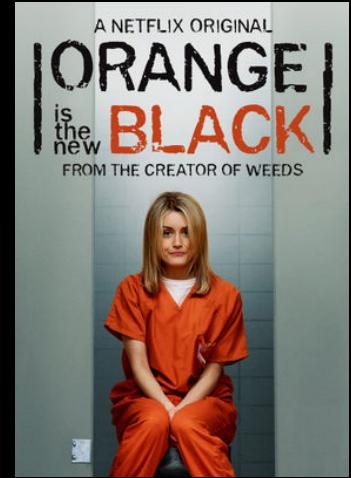
Discard any items emitted by a Collection after a second Collection emits an item or terminates



Top-rated Movies Collection

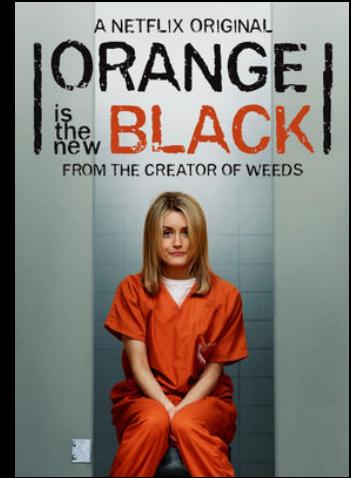
```
const getTopRatedFilms = (user) => {
  user.videoLists
    .map((videoList) =>
      videoList.videos
        .filter((v) => v.rating === 5)
    ).mergeAll();
};
```

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

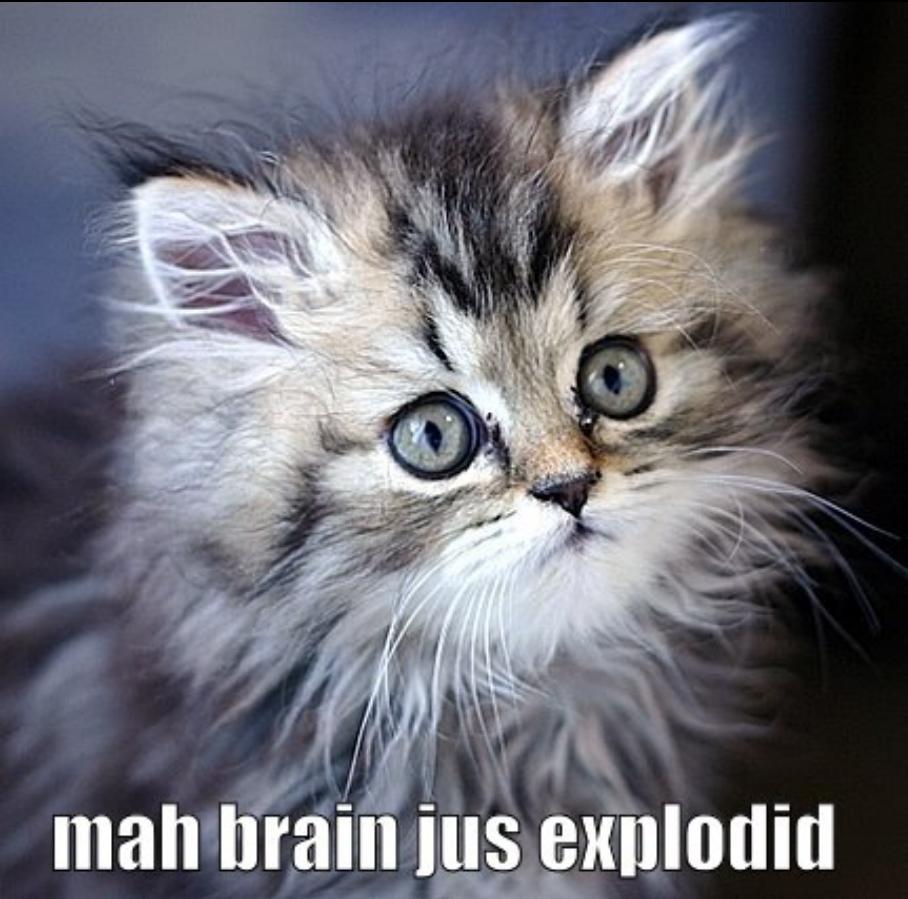


Top-rated Movies Collection

```
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

```
let getElementDrags = (elmt) =>
  dom.mousedown(elmt)
    .map((md) =>
      dommousemove(document)
        .filter .takeUntil(dommouseup(elmt)))
    ).mergeAll()
};

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



Mouse Drags Collection

```
let getElementDrags = (elmt) =>  
  dom.mousedown(elmt)  
    .flatMap((md) =>  
      dommousemove(document)  
        .filter .takeUntil(dommouseup(elmt))  
    )  
};  
  
getElementDrags(image)  
  .forEach(moveImage)
```





Everything is a stream

First-Class Async and Events

Objects to the rescue

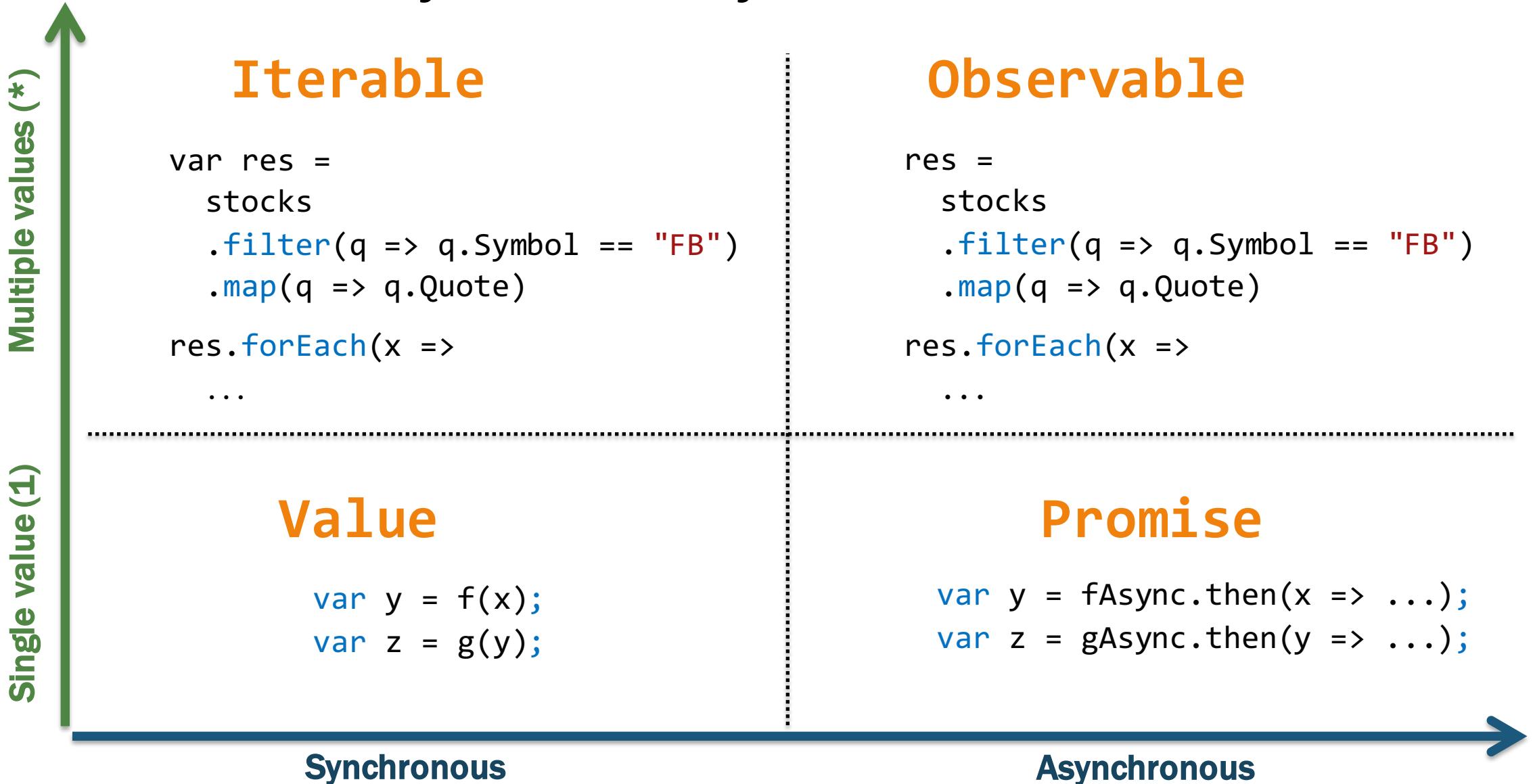
An object is **first-class** when it:^{[4][5]}

- 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)

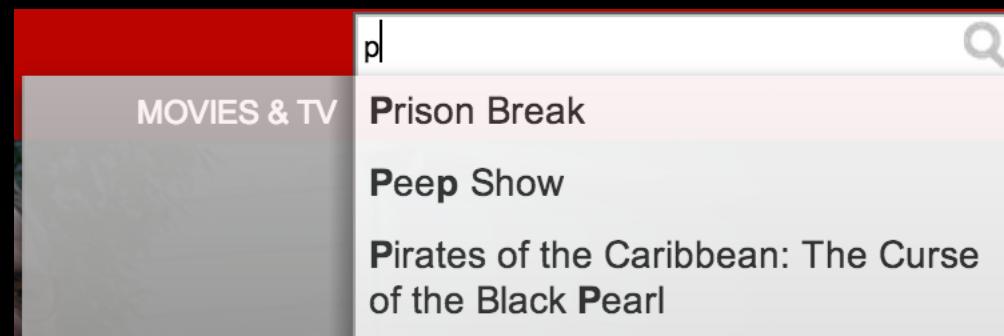


WIKIPEDIA
The Free Encyclopedia

The General Theory of Reactivity



Type Ahead Search



Type Ahead Search with Observables

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

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

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

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 = rowVisibilities.filter((v) => v);  
  let rowHides = rowVisibilities.filter((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.subscribe(  
  (license) => player.play(license),  
  (error) => showDialog('Sorry, can't play right now.'));
```



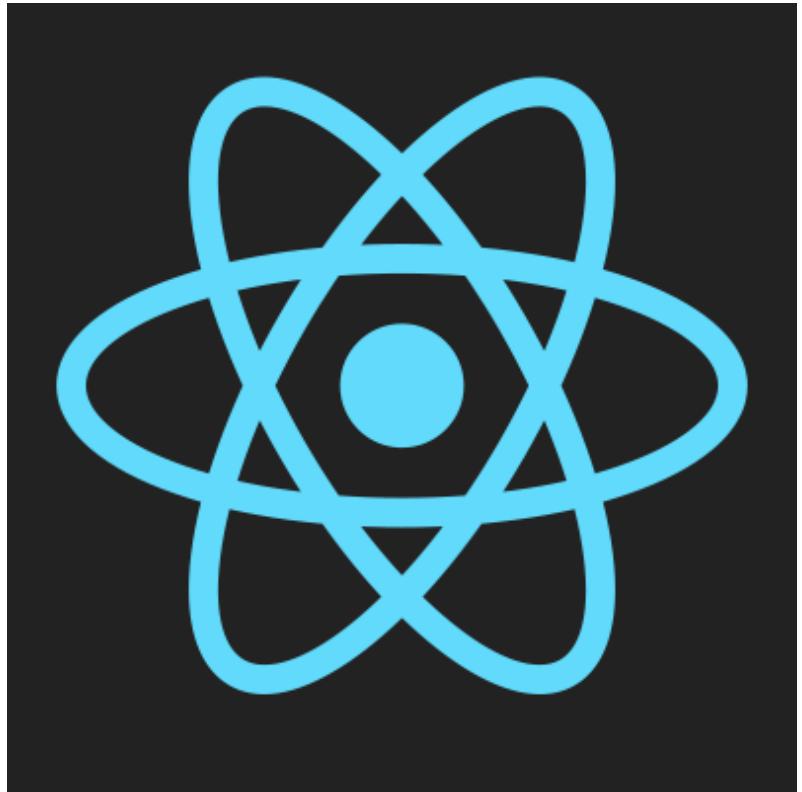


The Final Countdown...



```
try {  
  let resource = getData();  
  for (let data of resource) {  
    processItem(data);  
  }  
} catch (e) {  
  throw e;  
} finally {  
  resource && resource.dispose();  
}  
  
var d = getData();  
d.retry(3)  
.catch(defaultData)  
.finally(() => d.cleanup())  
.forEach(data =>  
  processItem(data));
```

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

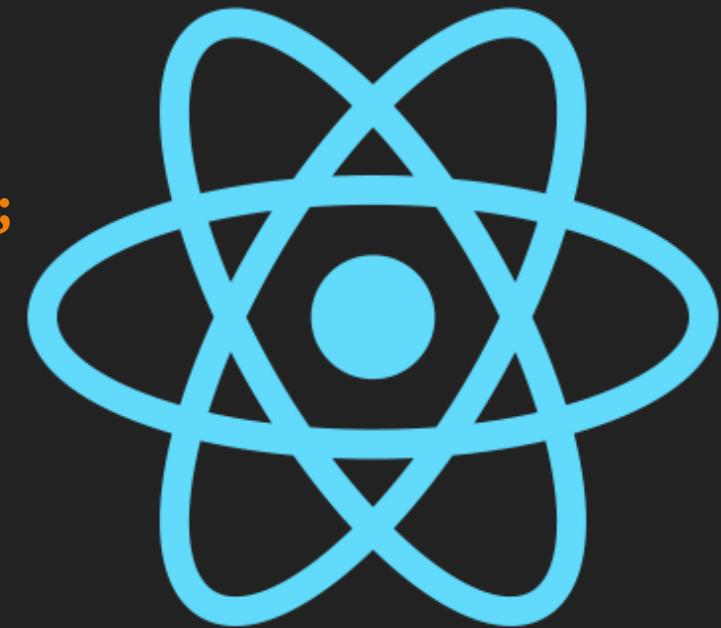
angular-meteor 23.8% 321

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>
    );
  }
});
```



React

The
Future
of

JS

Ur Kitteh of Death

Awaits

Async/Await

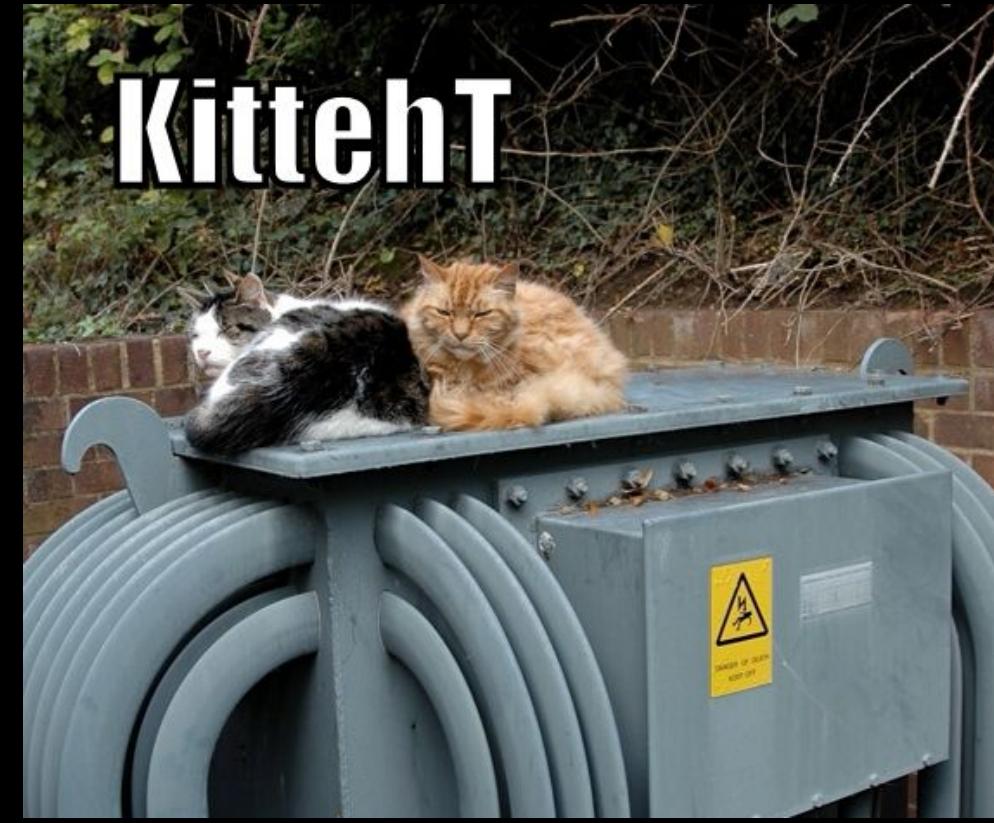
Coming to a JavaScript Engine Near You!

- Adds `async` and `await` keywords for Promises
- Accepted into Stage 1 of ECMAScript 7 in January 2014

```
async function chainAnimationsAsync(elem, animations) {  
    let ret = null;  
    try {  
        for (let anim of animations) {  
            ret = await anim(elem);  
        }  
    } catch (e) { /* ignore and keep going */ }  
    return ret;  
}
```

Async/Await with Observables and Generators...

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





Asynchronous Programming with Futures and Streams

- Futures are like JavaScript Promises
- Streams unify I/O and events
- Mirrors Rx by adding map/filter/reduce, etc

```
Stream<List<int>> stream = new File('quotes.txt').openRead();
stream.transform(UTF8.decoder).listen(print);

querySelector('#myButton').onClick.listen((_) => print('Click.'));
```

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 mouseDrags(element) {  
  
  let moveStreams = listen(element, "mousedown").map(e => {  
  
    e.preventDefault();  
  
    return takeUntil(  
      listen(element, "mousemove"),  
      listen(document, "mouseup"));  
  });  
  
  return switchLatest(moveStreams);  
}  
  
let subscription = mouseDrags(document.body).subscribe({  
  
  next(e) { console.log(`DRAG: <${ e.x }:${ e.y }>` ) },  
  throw(x) { console.log(`ERROR: ${ x }` ) },  
  return(x) { console.log(`COMPLETE: ${ x }` ) },  
});
```

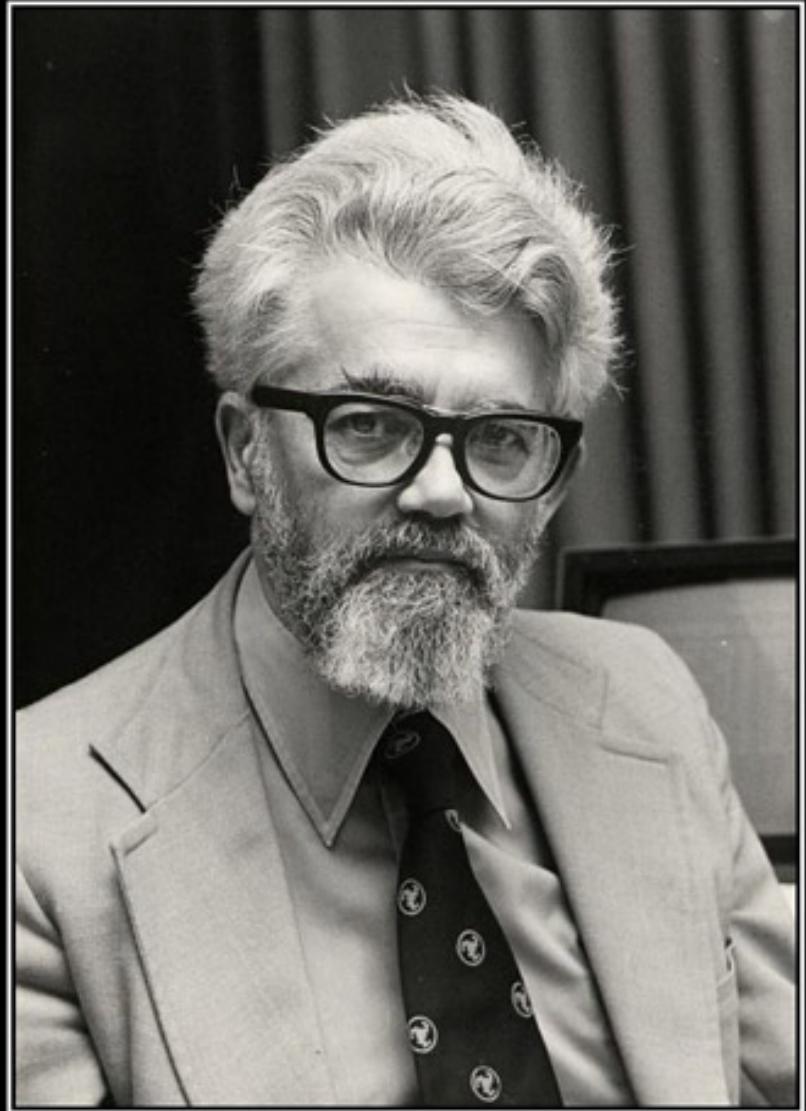
<https://github.com/zenparsing/es-observable/blob/master/demo/mouse-drags.js>



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

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

The Future of RxJS...



PROGRAMMING

YOU'RE DOING IT COMPLETELY WRONG.

Questions?

Matthew Podwysocki @mattpodwysocki

github.com/mattpodwysocki/nordic.js-2015