



Matthew Podwysocki

github.com/mattpodwysocki/fluentconf-2016

@mattpodwysocki

“FluentConf 2016”.sup();



λ Calrissian
@mattpodwysocki

OH: "take me down to concurrency city where
green pretty is grass the girls the and are"

RETWEETS

1,137

LIKES

624



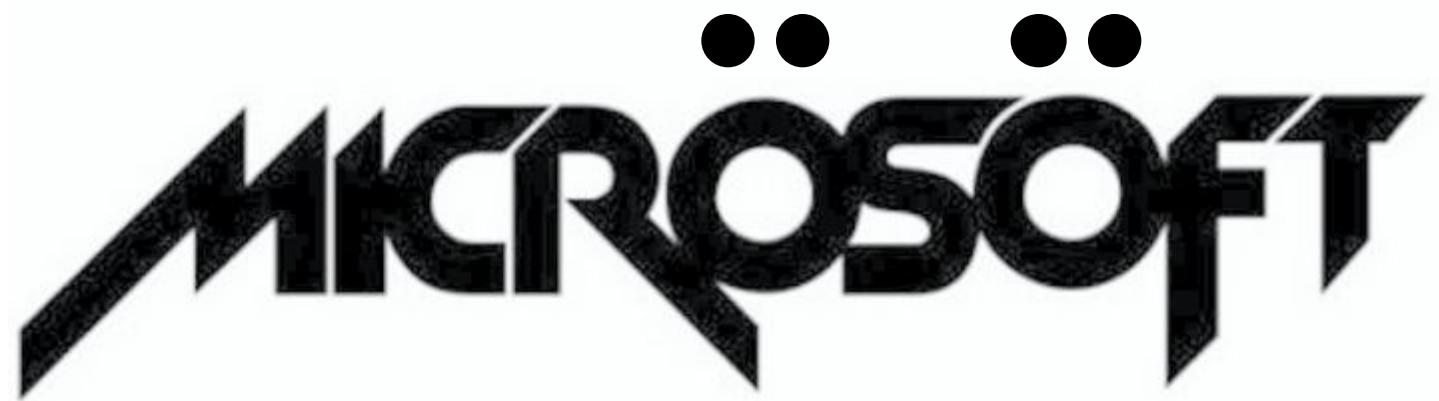
3:30 PM - 24 Oct 2013



...



**Principal SDE
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, and above the letter "O", there are three black dots, creating a stylized 'M' shape.



Reactive Extensions (Rx)

@ReactiveX
<http://reactivex.io>



Async is
Awful

Living in 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();  
  });  
}
```



Events and State, Oh My!

```
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);
```



Promises are ONLY PART of a solution

then

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

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





It's the final countdown

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



A black and white photograph of a man with short hair, wearing a light-colored shirt. He is seated at a desk, looking down and to his right with a contemplative expression. His hands are clasped on the desk in front of him. In the background, there are vertical blinds covering a window and some papers or books on the desk.

Can we do
better?

“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 Solved the Reactive Way!

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



Mouse Drags The Reactive Way

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

};
```

```
const subscription = getElementDrags(image)
  .subscribe(moveImage)
```



Type Ahead Search The Reactive Way

```
const data = dom.keyup(input)
    .map(e => e.target.value)
    .debounceTime(500)
    .distinctUntilChanged()
    .switchMap(term => search(term).retry(3));
```

```
const subscription = data.subscribe(
  data => bindData(data),
  err => handleError(err));
```



Oven Mitt

GRILL GLOVE

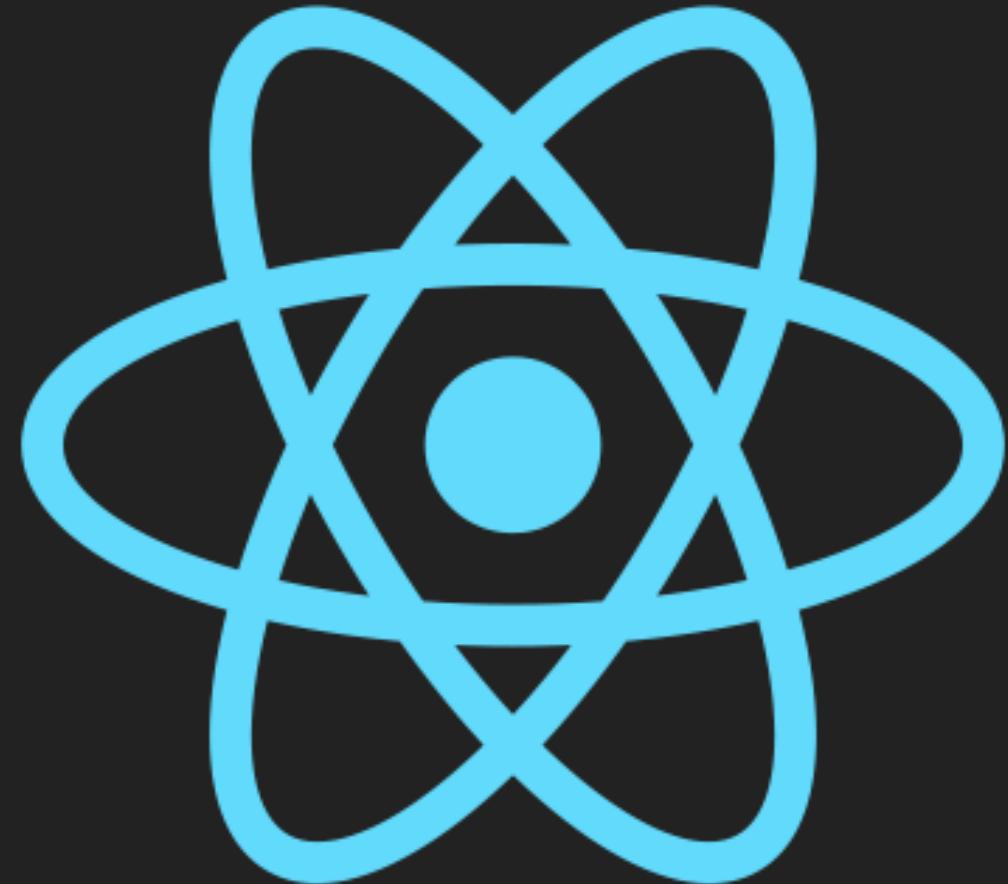
Much
better!

Professional Quality

Do Not Attempt

Reactive Programming





React

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>

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

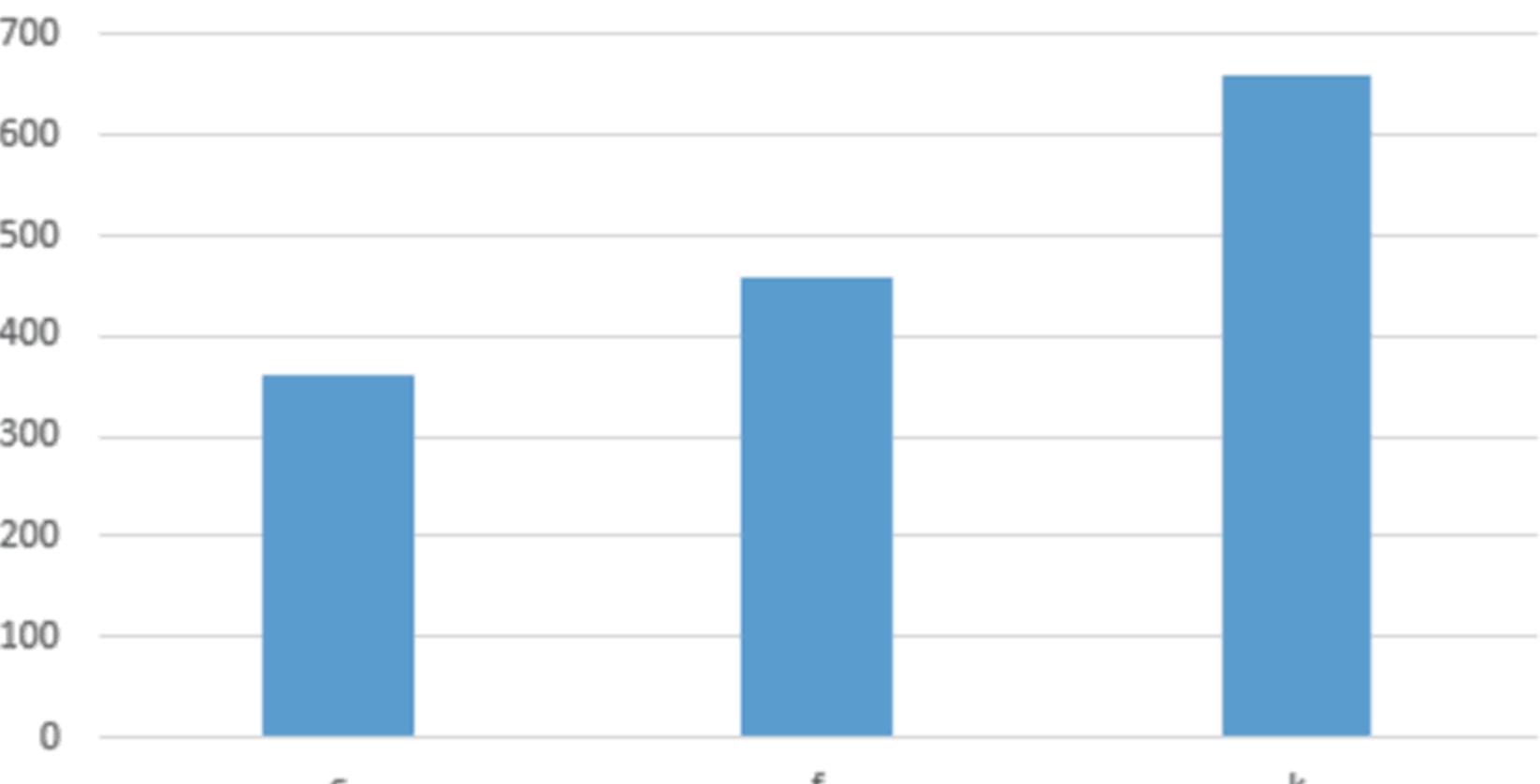
Number

T18

X ✓ fx

	A	B	C	D	E	F	G	H	I	J	K
1		2011	Show								
2	a	330.33									
3	b	329.35									
4	c	359.85	Yes								
5	d	376.94									
6	e	446.86									
7	f	457.7	Yes								
8	g	509.38									
9	h	549.9									
10	i	566.84									
11	j	585.2									
12	k	658.24	Yes								
13	l	720.2									
14											
15											
16											

Selected Only

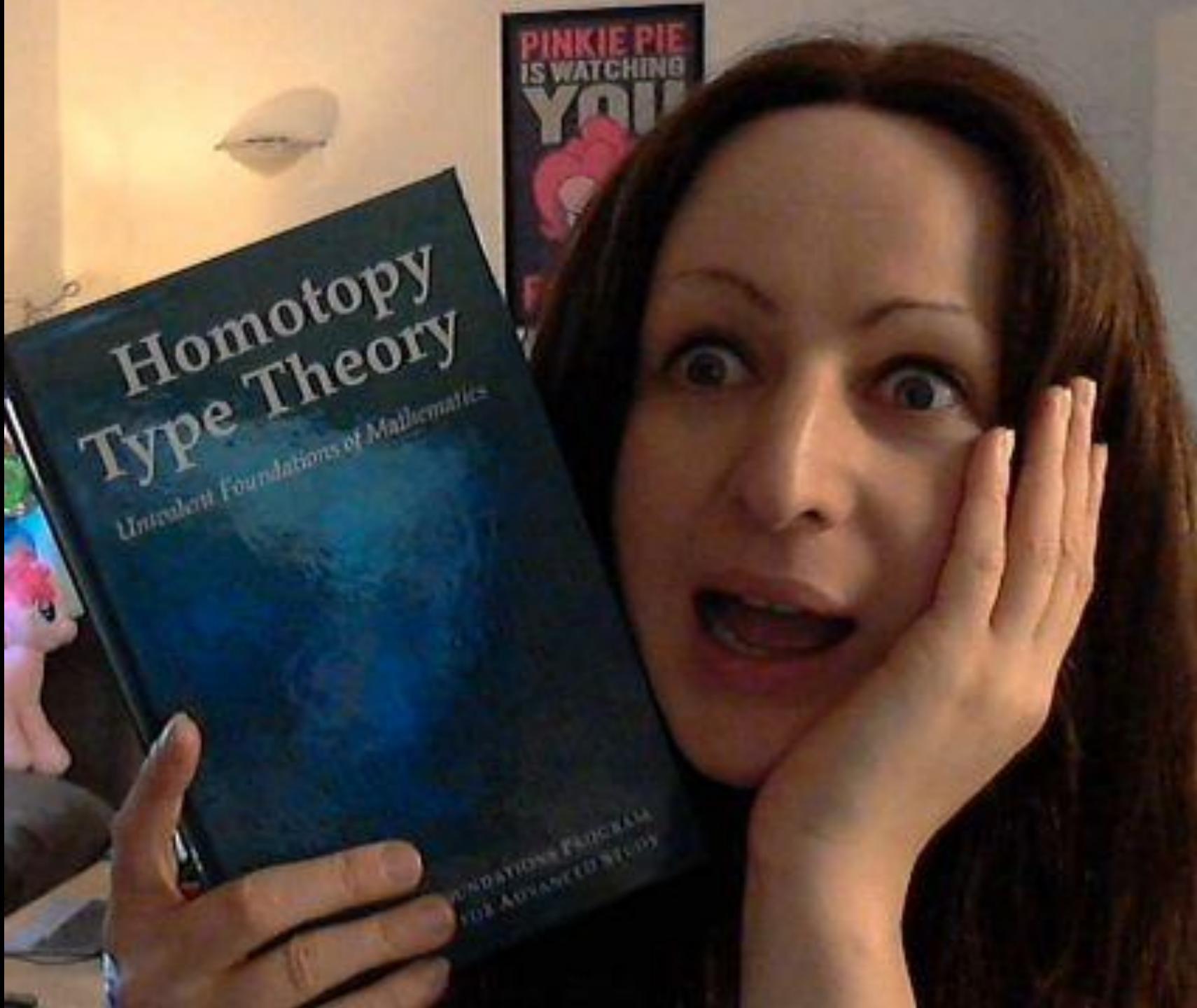


Wanna really know what Reactive Programming Is?

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

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



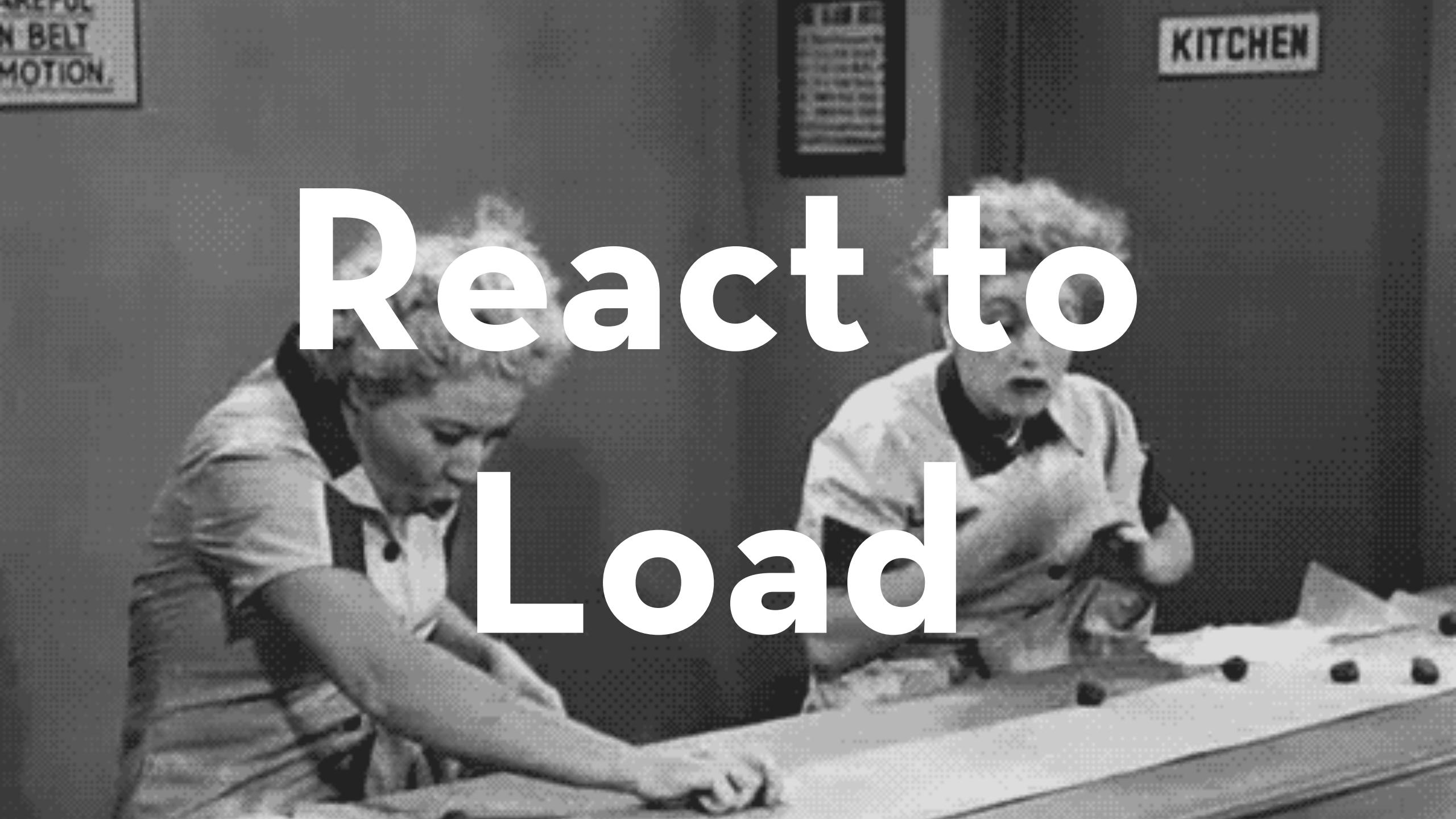


ComputerWissenschaftAkademischesPapierPhobie

AMERICAN
BELT
MOTION

KITCHEN

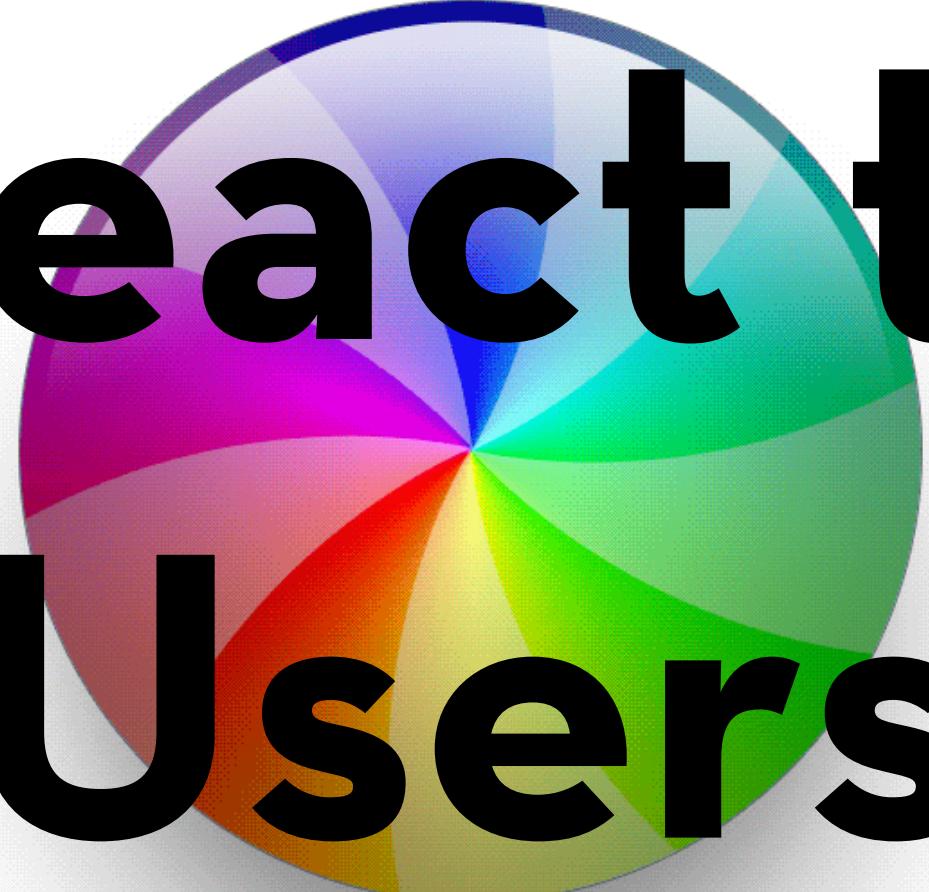
React to
Load





**React to
Failure**

next



**React to
Users**

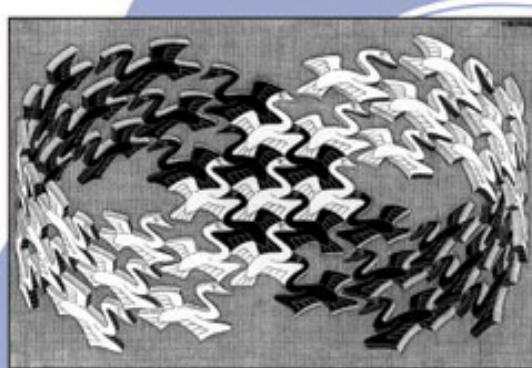
1994



Design Patterns

Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

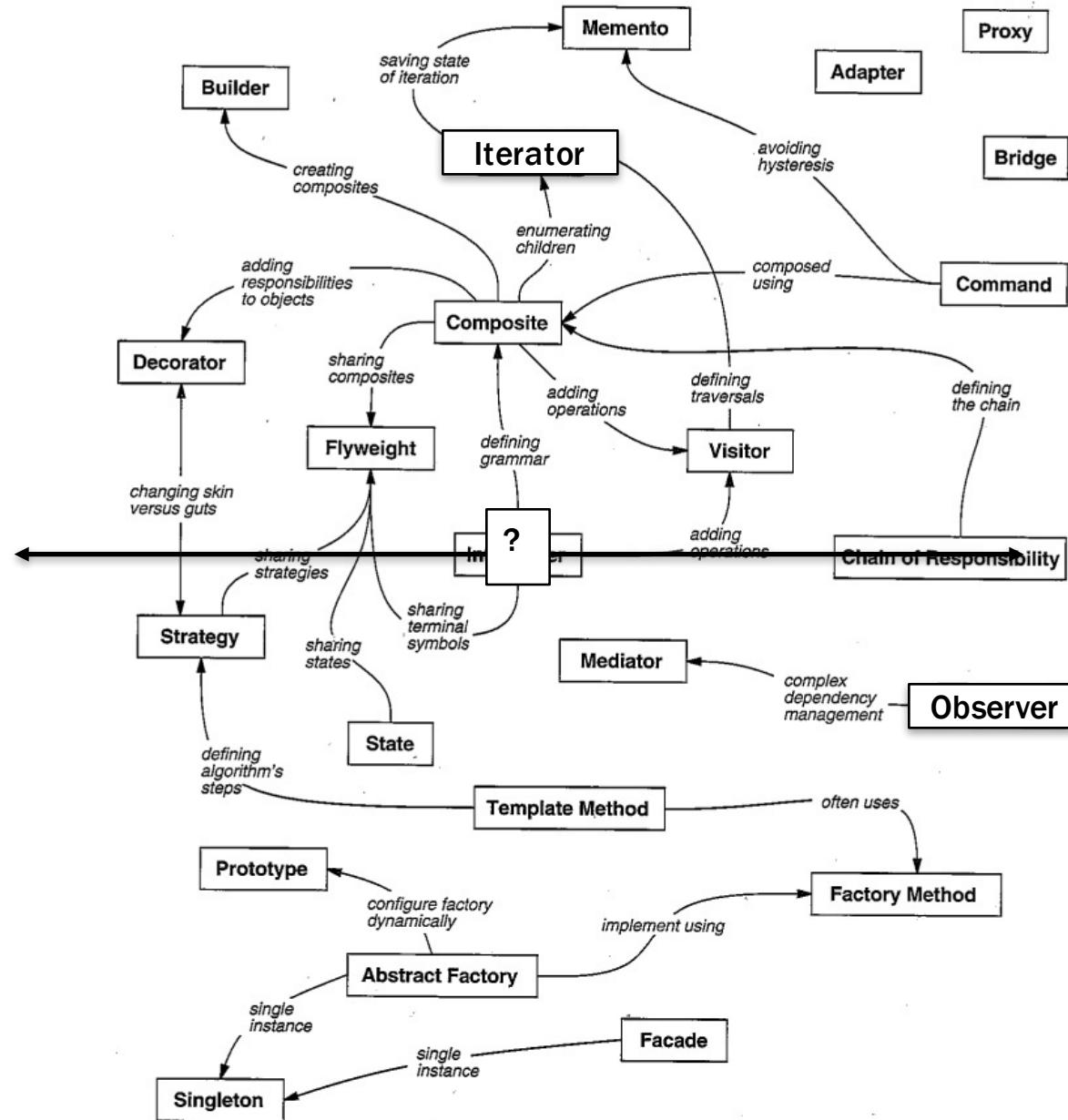


Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES





Design Pattern Relationships

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

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

... and an Event?



Events and Arrays are *both*
collections.

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 }  
> { clientX: 476, clientY: 760 }  
> { clientX: 476, clientY: 760 }
```

Observable is Subject/Observer Done Right

```
interface Observable<T>
    subscribe(o: Observer<T>) => Subscription<T>
}
```

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

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

Observable Grammar Police

next



Zero or more values

E.g. events are ∞ sequences

(error



Calls can fail

complete



Resource management

Sequencing



2

Must be a sequence
No concurrent callbacks
One time termination

First-Class Asynchronous Values

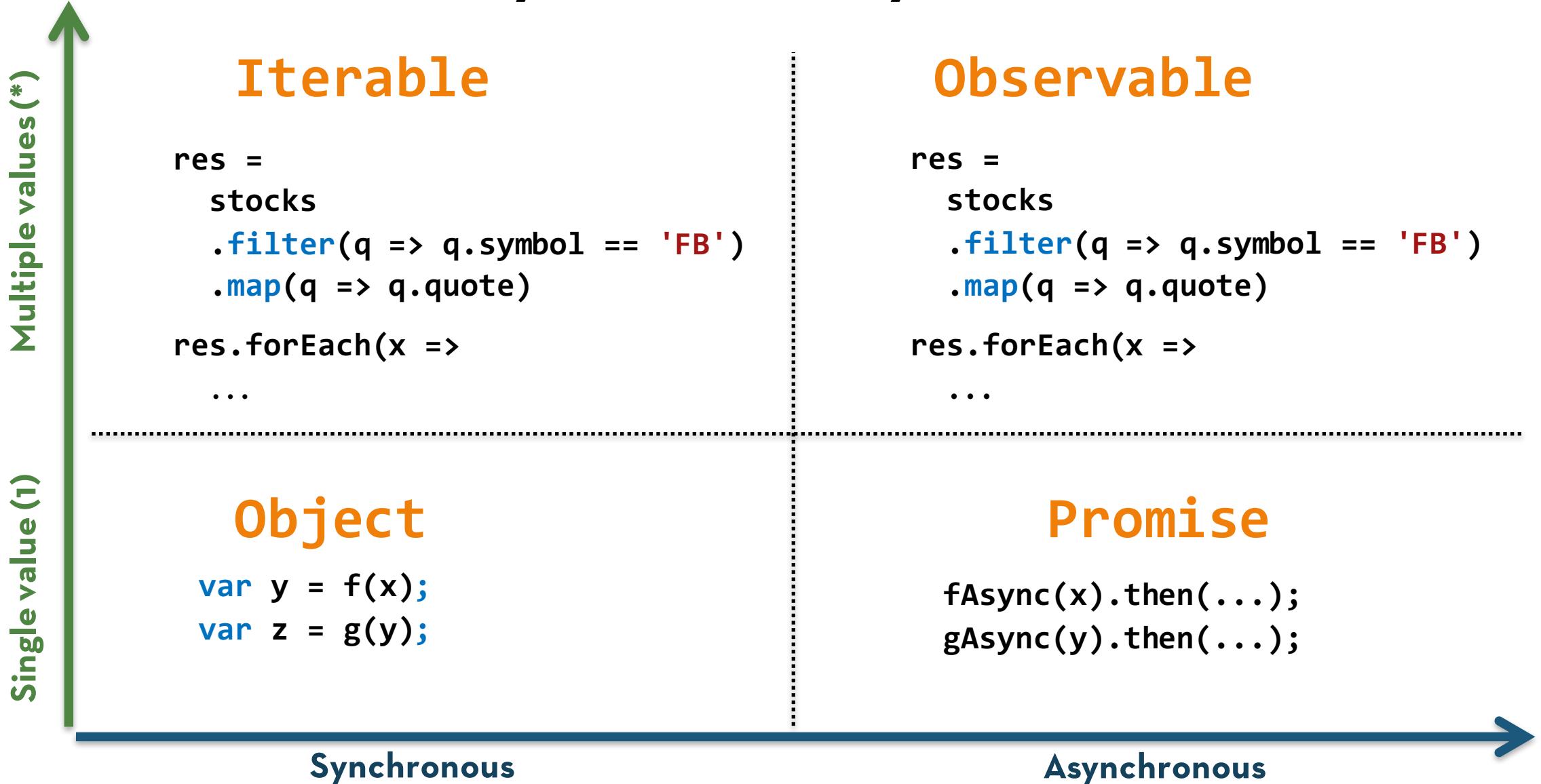
An object is **first-class** when it:^{[4][5]}

- 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

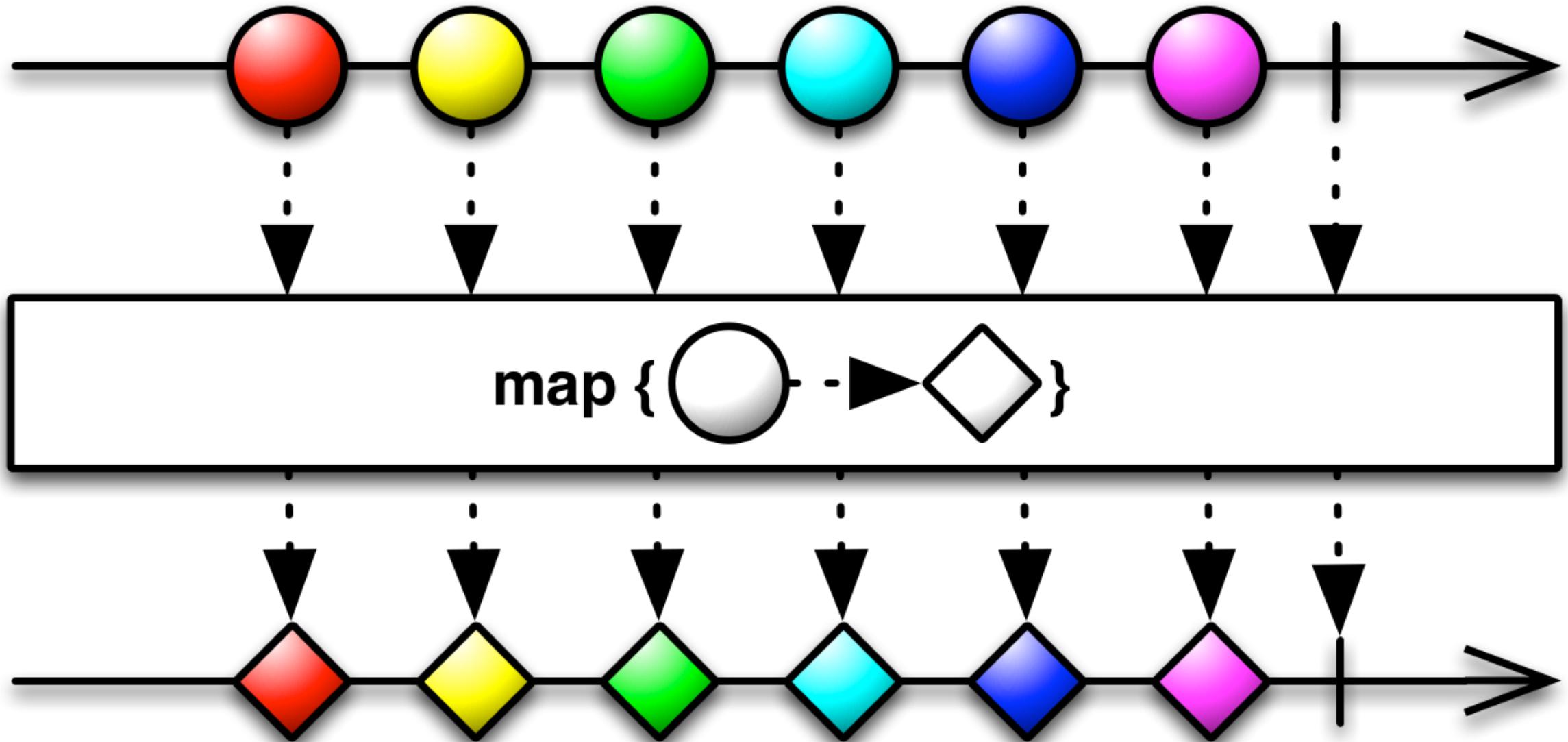
The General Theory of Reactivity

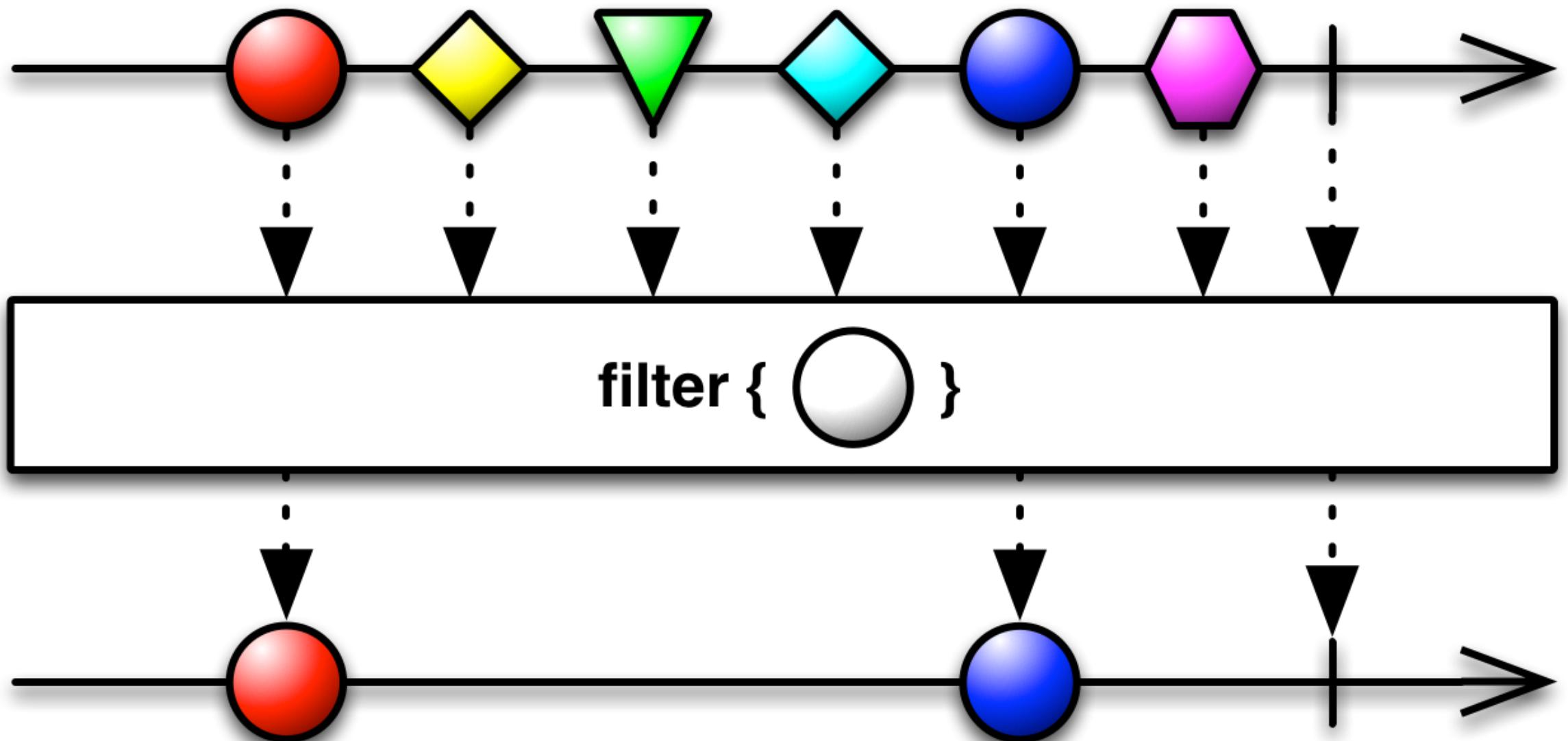


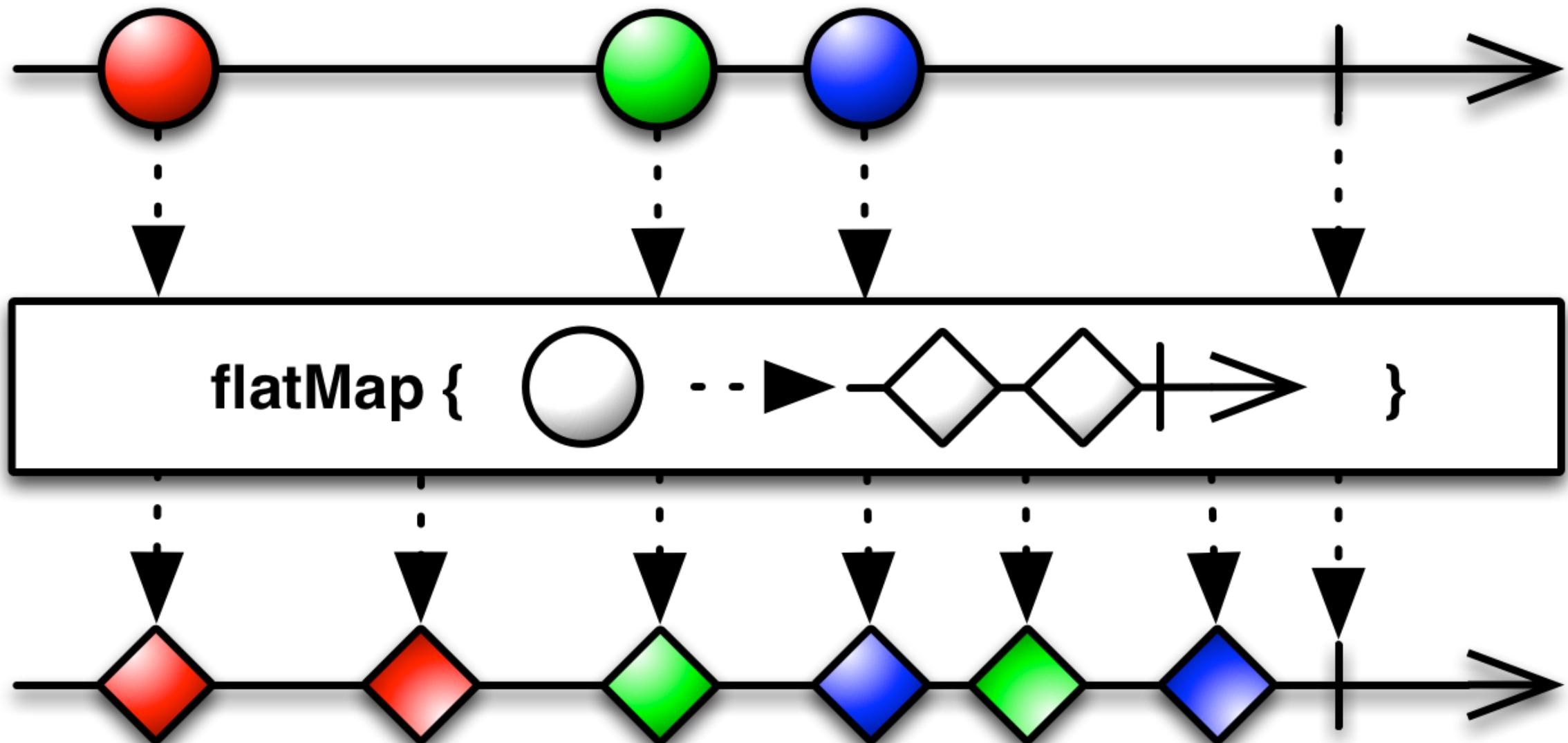


Everything is a stream

The majority of asynchronous
code using Observables can be
written with just a few *flexible*
functions.







MAYBE WE CAN
ITERATE OVER...

FLATMAP
THAT SHIT!



The Future
is Here...



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>

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

```
@Component({
  selector: 'my-app',
  template: `
    <div>
      <h2>Wikipedia Search</h2>
      <input type="text" [ngFormControl]="term"/>
      <ul>
        <li *ngFor="#item of items | async">{{item}}</li>
      </ul>
    </div>
  `
})
```

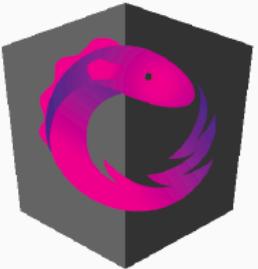
```
import {Injectable} from 'angular2/core';
import {URLSearchParams, Jsonp} from 'angular2/http';

@Injectable()
export class WikipediaService {
  constructor(private jsonp: Jsonp) {}

  search (term: string) {
  }
}
```

```
search (term: string) {  
  var url =  
    'http://en.wikipedia.org/w/api.php?callback=JSONP_CALLBACK';  
  var search = new URLSearchParams()  
  search.set('action', 'opensearch');  
  search.set('search', term);  
  search.set('format', 'json');  
  
  return this.jsonp  
    .get(url, { search })  
    .map((response) => response.json()[1]);  
}
```

```
export class App {  
  items: Observable<Array<string>>;  
  term = new Control();  
  constructor(private service: WikipediaService) {  
    this.items = this.term.valueChanges  
      .debounceTime(400)  
      .distinctUntilChanged()  
      .switchMap(term =>  
        this.service.search(term));  
  }  
}
```



ngrx

Reactive Extensions for angular2

Repositories

People 1

Filters ▾

Find a repository...

devtools

TypeScript 13 3

Developer Tools for ngrx

Updated 16 hours ago

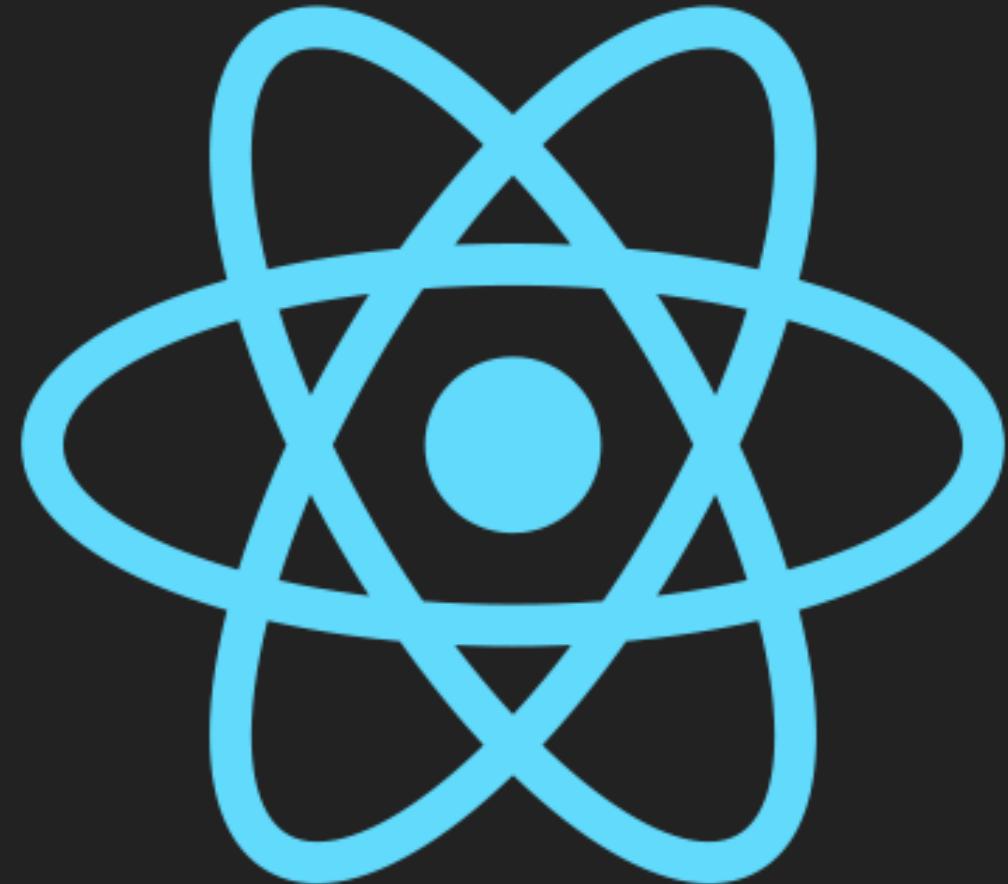
store

TypeScript 432 43

RxJS powered state management for Angular2 apps, inspired by Redux

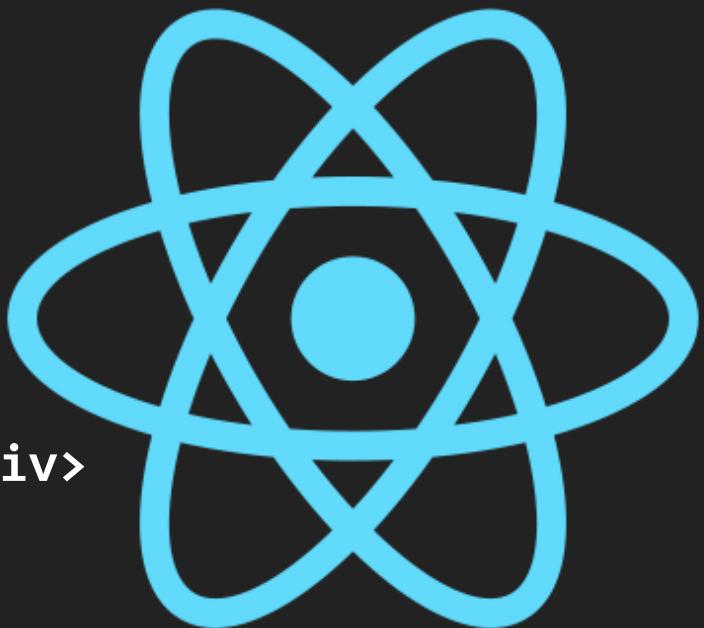
Updated 2 days ago

<https://github.com/ngrx>



React

```
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>
    );
  }
});
```





Fluorine

<https://github.com/phipl/fluorine>

```
// Create a dispatcher which is our event stream
const dispatcher = createDispatcher();

// Agendas, if fails, all is reverted
const addIfFetchSucceeds = Observable
  .of({ type: 'ADD' })
  .concat(Observable
    .of('/ping')
    .flatMap(fetch)
  );
  dispatcher.schedule(addIfFetchSucceeds);
```

The question is: do you really need Redux if you already use Rx? Maybe not. It's not hard to [re-implement Redux in Rx](#). Some say it's a two-liner using Rx `.scan()` method. It may very well be!

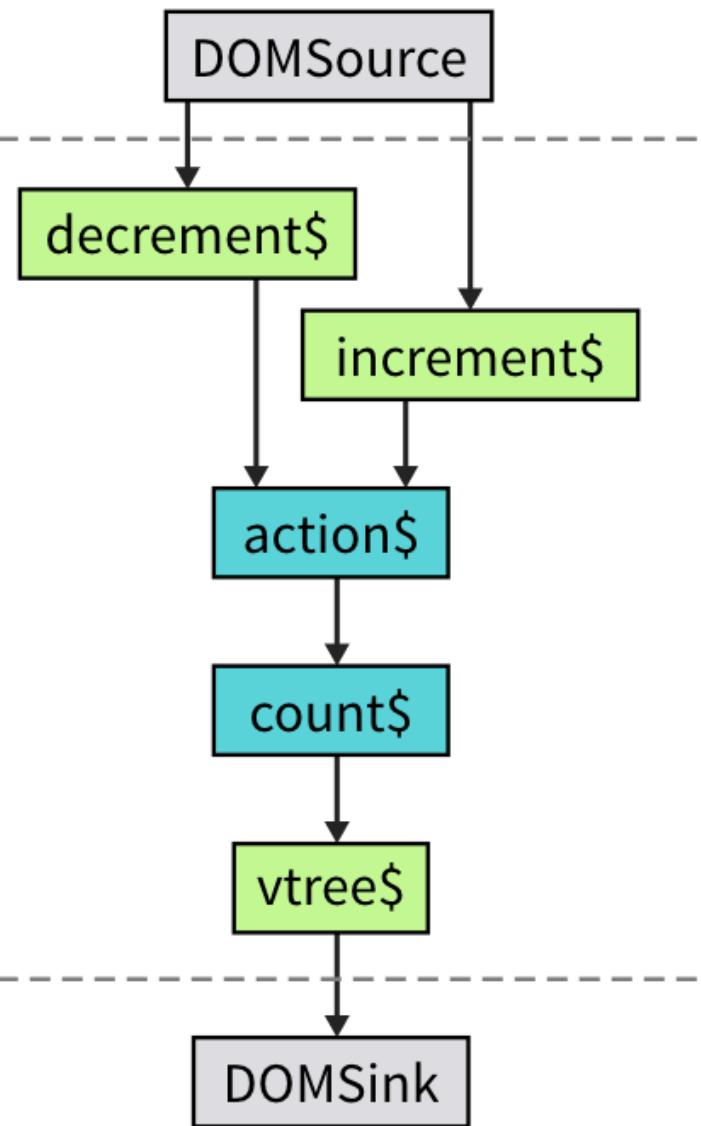
If you're in doubt, check out the Redux source code (there isn't much going on there), as well as its ecosystem (for example, [the developer tools](#)). If you don't care too much about it and want to go with the reactive data flow all the way, you might want to explore something like [Cycle](#) instead, or even combine it with Redux. Let us know how it goes!



Cycle.js

A fully reactive JavaScript framework for Human-Computer Interaction

[Get started](#)



```

function main(sources) {
  const decrement$ = sources.DOM
    .select('.decrement').events('click').map(ev => -1);

  const increment$ = sources.DOM
    .select('.increment').events('click').map(ev => +1);

  const action$ = Observable.merge(decrement$, increment$);
  const count$ = action$.startWith(0).scan((x,y) => x+y);

  const vtree$ = count$.map(count =>
    div([
      button('.decrement', 'Decrement'),
      button('.increment', 'Increment'),
      p('Counter: ' + count)
    ])
  );
  return { DOM: vtree$ };
}
  
```



[gitter](#) [join chat](#) [circleci](#) failing [npm package](#) 0.10.1

Rewrite in progress!!! While most of the API for Yolk will remain the same, it is currently being rewritten to accomodate various internal improvements that need to be made. [You can follow along here.](#) 😊

Anyway...

A library for building asynchronous user interfaces.

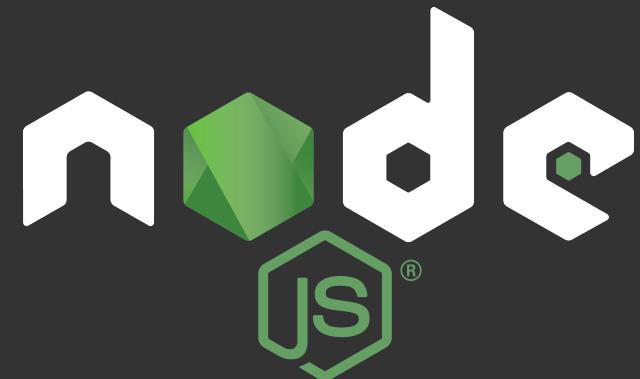
- **Familiar:** Yolk is a small library built on top of [Virtual DOM](#) and [RxJS](#). It exposes a very limited API so that you don't have to spend weeks getting up to speed. Yolk components are just plain functions that return JSX or hyperscript.
- **Everything is an observable:** Yolk components consume RxJS observable streams as if they were plain values. From a websocket connection to a generator function to an event handler. If it can be represented as an observable, then it can be rendered directly into your markup.
- **Stateless:** Being able to describe user interactions, control flow and plain values as observable streams means that application design becomes entirely declarative. There is no need to manually subscribe to observables in order to mutate or set component state.

```
const requests_ = new Rx.Subject();

function sendHello(e) {
  e.res.writeHead(200, { 'Content-Type': 'text/plain' });
  e.res.end('Hello World\n');
}

requests_
  .do(e => console.log('request to', e.req.url))
  .subscribe(sendHello)

http.createServer((req, res) => {
  requests_.next({ req: req, res: res });
})
```



Observables Coming to ES2017?!?!

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 ES2017

```
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();
```



Via the Washington Post. wapo.st/1OjJ02P

@ReactiveX
<http://reactivex.io>

Matthew Podwysocki
github.com/mattpodwysocki/fluentconf-2016

@mattpodwysocki