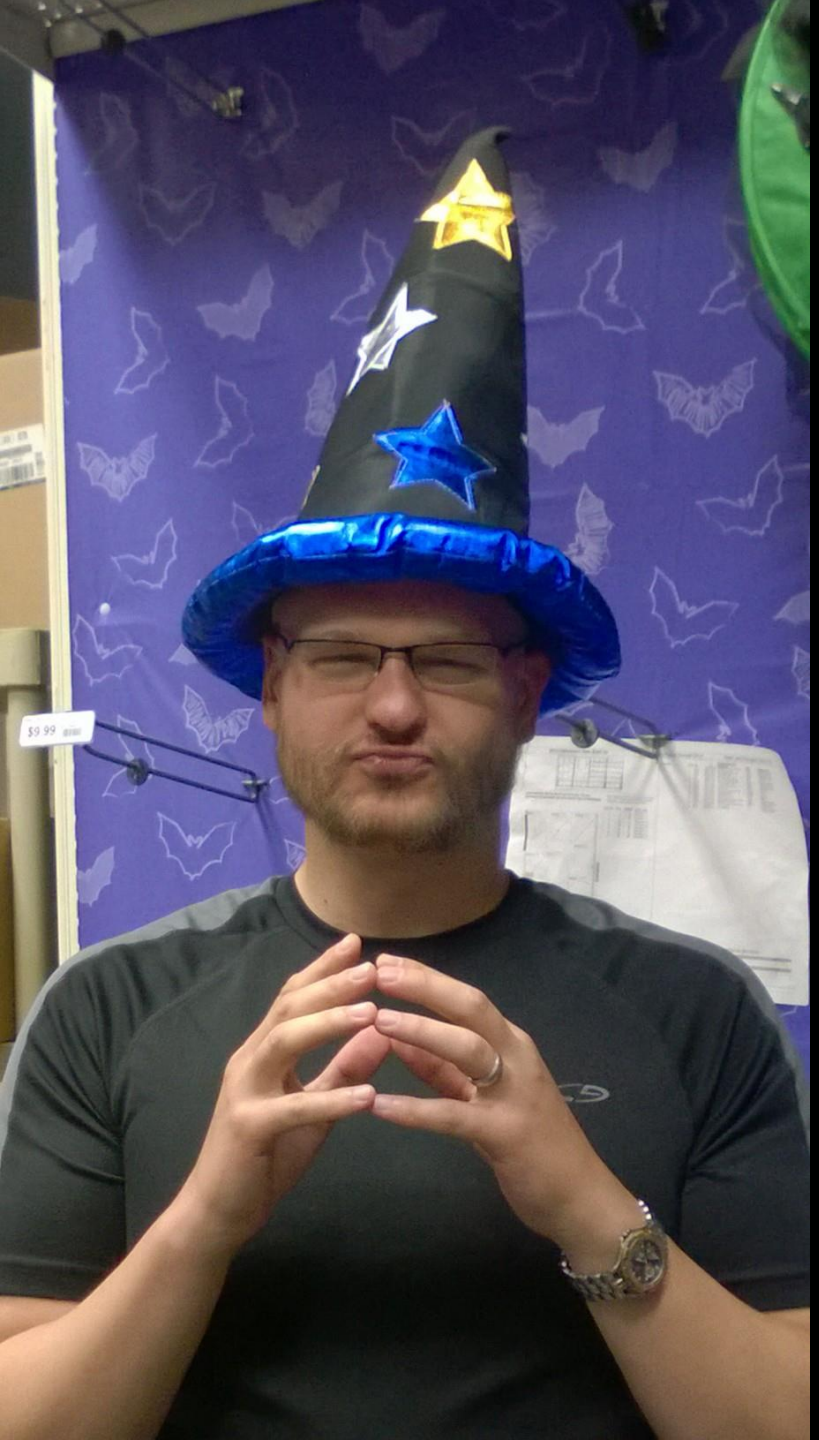
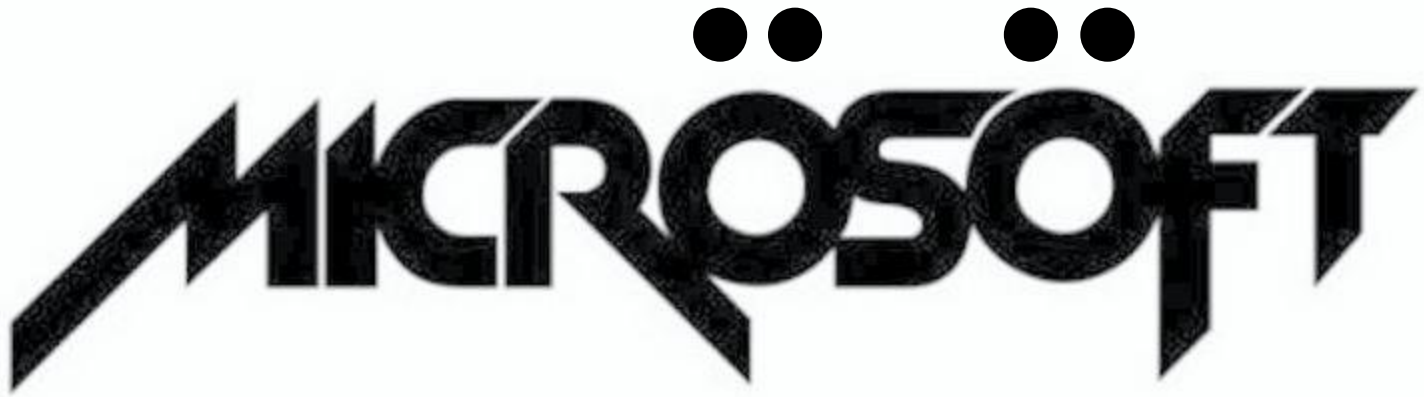
Reactive Extensions for JavaScript and the Future of Client-Side Development

Matthew Podwysocki @mattpodwysocki

github.com/mattpodwysocki/codeonthesea2015



Principal SDE
Open Sourcerer
@mattpodwysocki
github.com/mattpodwysocki


MICROSOFT

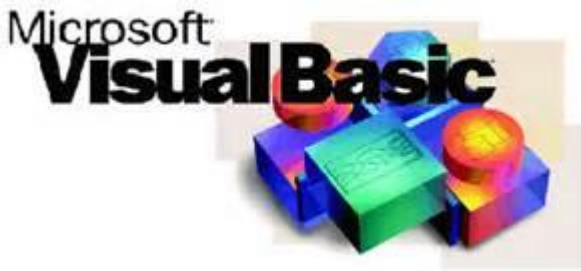
Reactive Extensions (Rx)

@ReactiveX

http://reactivex.io

NETSCAPE®

One Language to Rule Them All…




```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
 Function concatenate(first, last)
    Dim full
    full = first & last
    concatenate = full
  End Function

  Dim result
  result = concatenate("Matthew", "Podwysocki")
  MsgBox(result)
</script>
</body>
</html>
```


JS

```
<h1>
Hello
<img src="spacer.gif" width="200">
World
</h1>
<table>
  <tr>
    <td width="20%">Main Menu</td>
```

```
<a href="#" onclick="return hi();">
Click Me!
</a>
```

The Revolution Begins...

```
new ActiveXObject("Microsoft.XMLHTTP")
```

```
new ActiveXObject("Msxml2.XMLHTTP")
```

```
new ActiveXObject("Msxml2.XMLHTTP.6.0")
```

```
var xhr = new XMLHttpRequest();

xhr.open('get', 'getdata.php');

// Track the state changes of the request.
xhr.onreadystatechange = function () {
    if (xhr.readyState === 4) {
        if (xhr.status < 300) {
            alert(xhr.responseText);
        } else {
            alert('Error: ' + xhr.status);
        }
    }
};

// Send the request to getdata.php
xhr.send(null);
```

JavaScript Object Notation (JSON)

JSON-P

```
<script>
function apiStatus(data) {
  console.log(data.status);
}

var script = document.createElement('script');
script.src = 'https://status.github.com/api/status.json?callback=apiStatus';
document.body.appendChild(script);
</script>

apiStatus({
  "status": "good",
  "last_updated": "2012-12-07T18:11:55Z"
})
```


JS

Show Me the $$!


```
$('#text-input').keyup(function () {
  $.ajax({
    url: 'http://en.wikipedia.org/w/api.php',
    dataType: 'jsonp',
    data: {
      action: 'opensearch',
      format: 'json',
      search: $(this).val()
    },
    success: function (results) { ... }
    error: function (xhr, status, err) { ... }
  });
});
```

Add a number to another number in JavaScript

0

hallo

I have got a number in my JavaScript variable! Now how do I add another number to it? Please

javascript

3 Answers

oldest newest votes

22

You should definitely use jQuery. It's really great and does all things

link | edit | flag

answered **11 minutes ago**
I<3jQuery
1,234 2 13

I agree, jQuery is really the best, it solves all kinds of browser problems and is good, as well – ||sumc0da 8 mins ago

+1 jquery is best quality code ever, if you don't use your a idiot – Werry_Togan 4 mins ago

add comment

4

I think there's a jQuery plugin for that. Google for jQuery basic arithmetic plugin.

link | edit | flag

answered **5 minutes ago**
Timothy Goatse
4,321 1 12

yeah, jQuery is definately the way to go – fishnipples 5 mins ago

I used the jQuery diet plugin and lost 10kg in a week – jfatty 4 mins ago

add comment

-2

To add numbers together you should use the + operator, for example:

```
var a= 1;
var b= a+2;
alert(b);  // 3
```

link | edit | delete | flag

answered **50 seconds ago**
bobince
some

-1 not enough jQuery – ||sumc0da 30 secs ago

you suck – Timothy Goatse 3 secs ago

tagged
javascript × 18553

asked
a while ago

viewed
some times

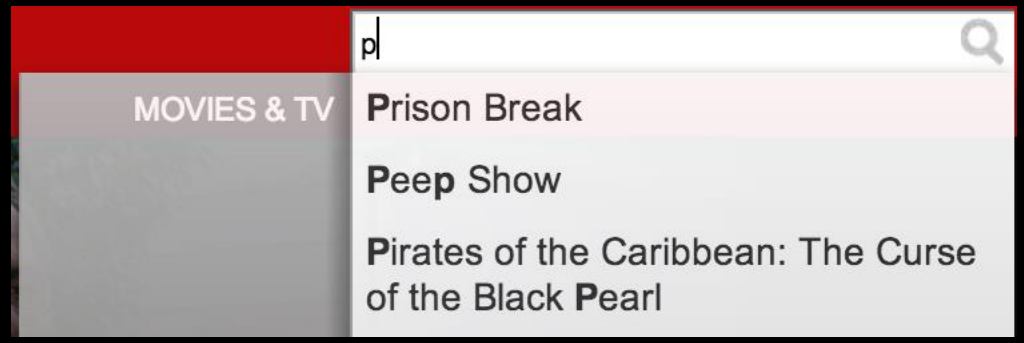
latest activity
just now

Related

- What is the best number?
- How can I use JavaScript to parse some HTML using regex?
- JavaScript: why is my text content getting mangled when I clone nodes? Obviously I must be doing something wrong as jQuery is perfect
- Stupid JavaScript floating point numbers are broken
- How can I extract number from HTML using a regex without ZALGO singing the song that ends the world?
- Is there a jQuery plugin for making an HTML page appear in the browser?
- Where are my legs?


JS

Autcomplete by Example



Let’s Face It, Asynchronous Programming is Awful!

Autocomplete without RxJS

Why RxJS?

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.clienyY - 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);
```


ELMER'S
Glue-All
MULTI-PURPOSE GLUE
For Household Repairs,
Craft Projects,
& more!
New
Stronger
Formula!
Dries Fast
Safe & Non-Toxic
4 FL OZ (118 mL)
1322 10001US

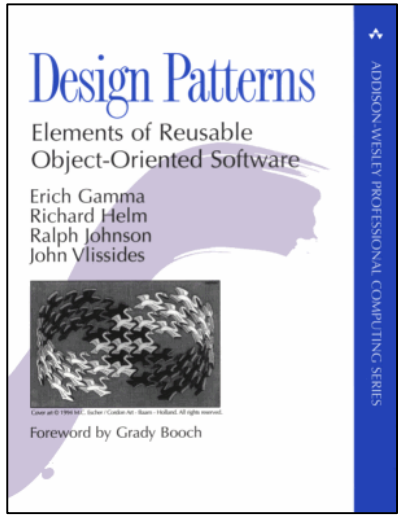

3M Tartan
3939

Fundamental Abstractions

Adapting the observer pattern

- Ensuring duality with the enumerator pattern
- More compositional approach

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

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



"Gang of four" book

Addison-Wesley

```
Notification grammar

OnNext* (OnError | OnCompleted)?
```

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

Highly Compositional

Function composition

www.Wikipedia.org

LINQ-style query operators

- Composition of 0-N input sequences

```
interface Observable<T> {
        map<R>(selector: (value: T) => R) : Observable<R>
        filter(predicate: (value: T) => boolean) : Observable<T>
}
```

- Composition of disposable subscriptions and scheduler resources

```
interface Scheduler {
        now(): Number
        schedule(action: () => void)
        ...
}
```

Highly Compositional

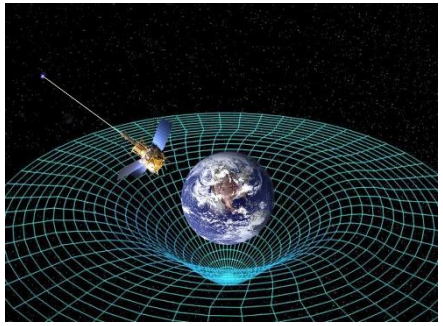
Function composition

www.Wikipedia.org

Building a binary Merge operator

```
class Observable<T> {
  merge(ys : Observable<T>) : Observable<T> {
    let xs = this;
    return Observable.create<T>(o =>
      new CompositeDisposable(
        xs.subscribe(x => o.onNext(x), err => o.onError(err), () => o.onCompleted()),
        ys.subscribe(x => o.onNext(x), err => o.onError(err), () => o.onCompleted())
      );
    )
  }
}
```

The Role of Schedulers



Space-time
www.Wikipedia.org

Pure architectural layering of the system

- Logical query operators (~ relational engine)
- Physical schedulers (~ runtime environment)

```
static just<T>(value: T, scheduler : Scheduler) {
  return Observable.create<T>(o => scheduler.schedule(() => {
    o.onNext(value); o.onCompleted(); })
  );
}
```

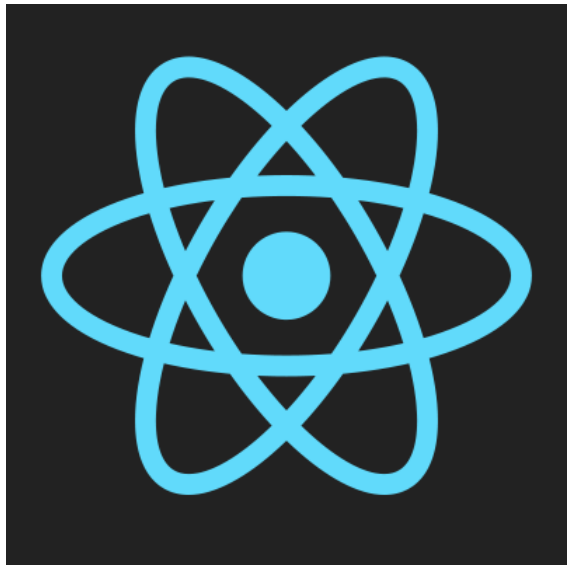
Abstract over sources of asynchrony and time

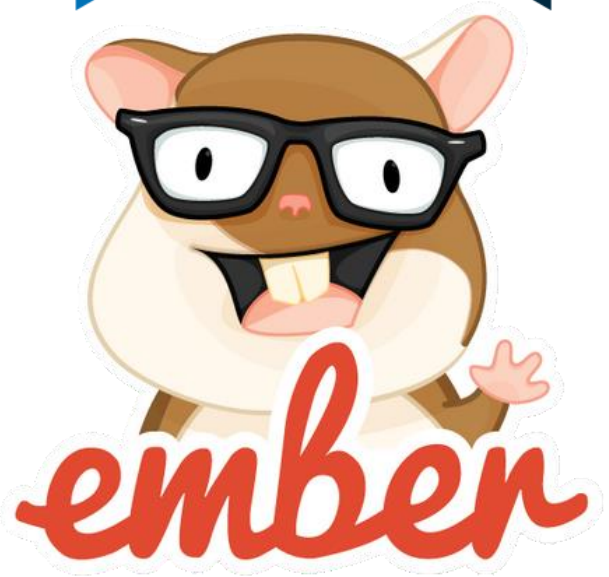
- setTimeout, setInterval, requestAnimationFrame, setImmediate
- Date.now(), timers

Enable virtual time testing


Everything is a stream

What About My Libraries?

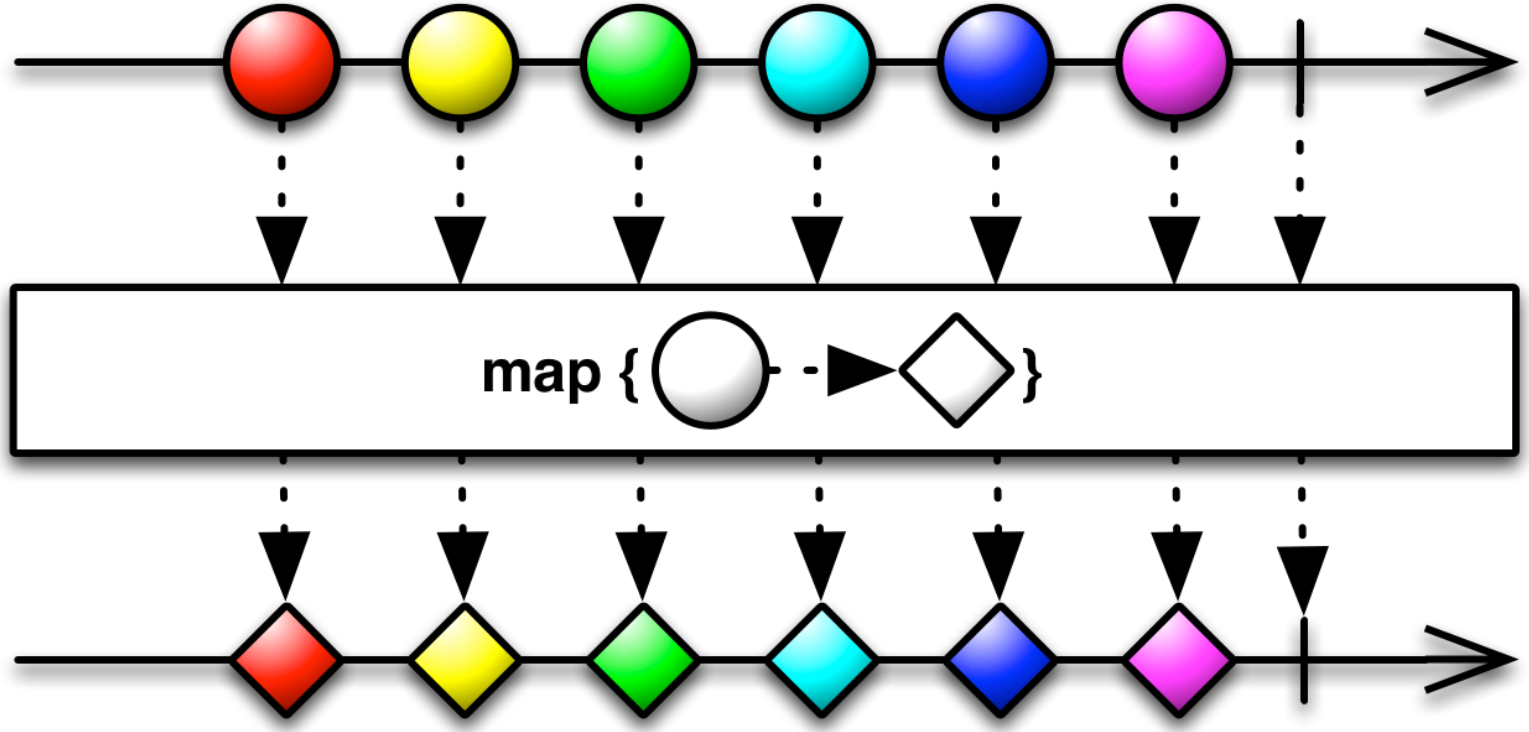




The majority of your asynchronous code can be written with just a few *flexible* functions.

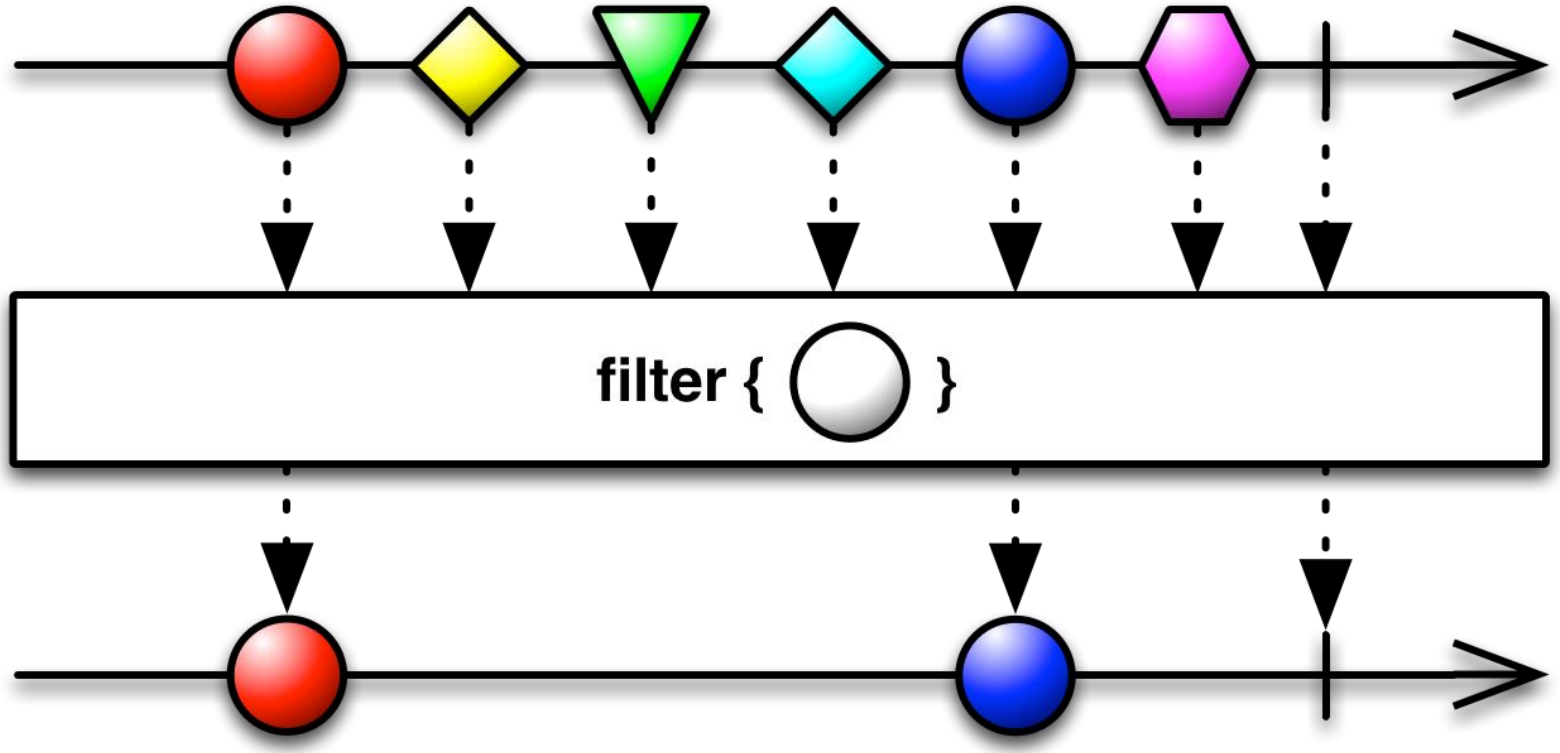
map()

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



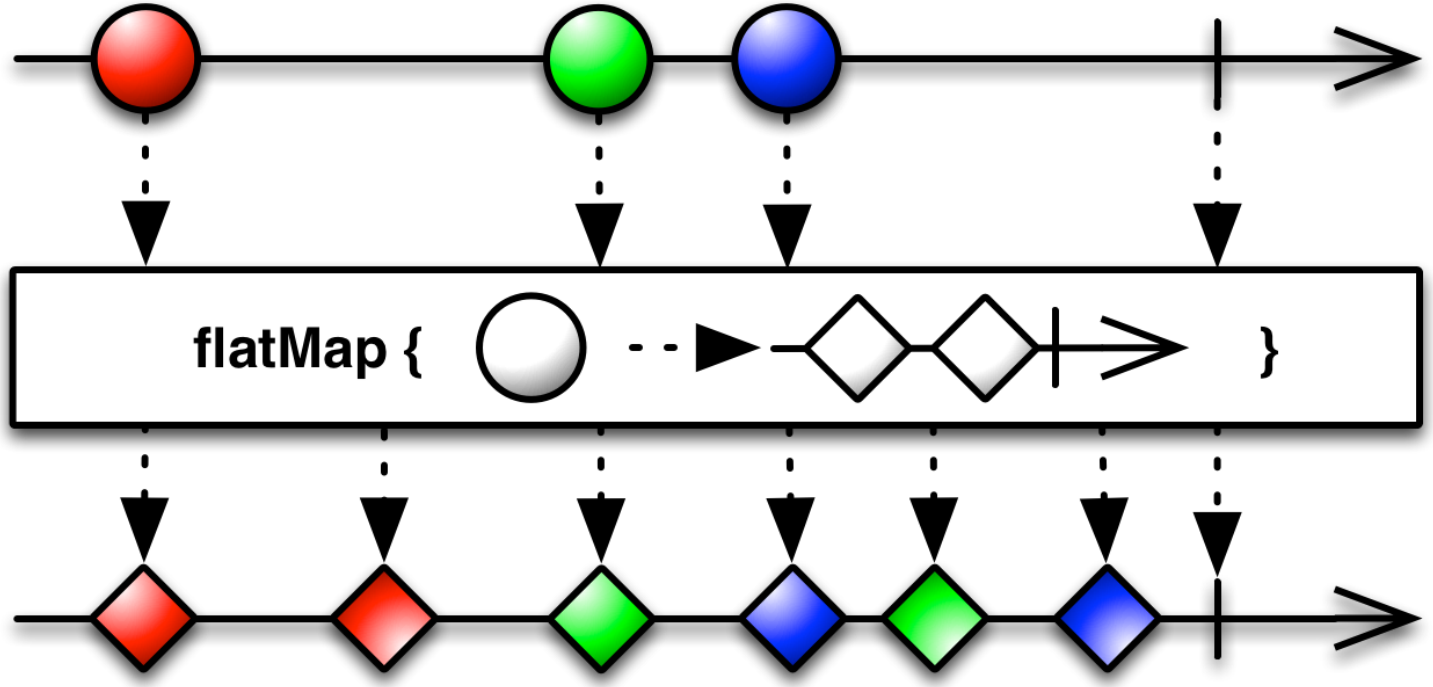
filter()

Filter items emitted by a Collection



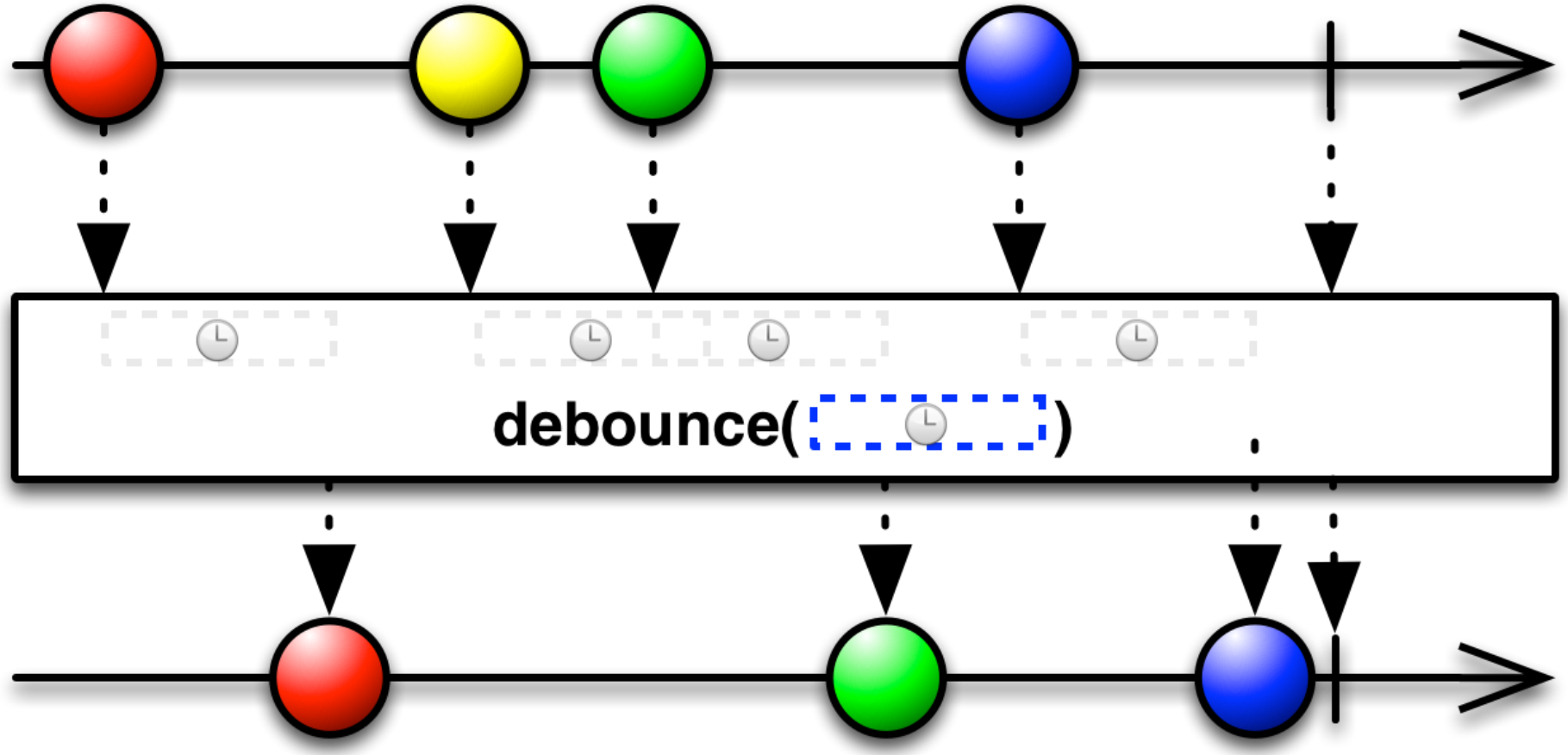
flatMap()

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



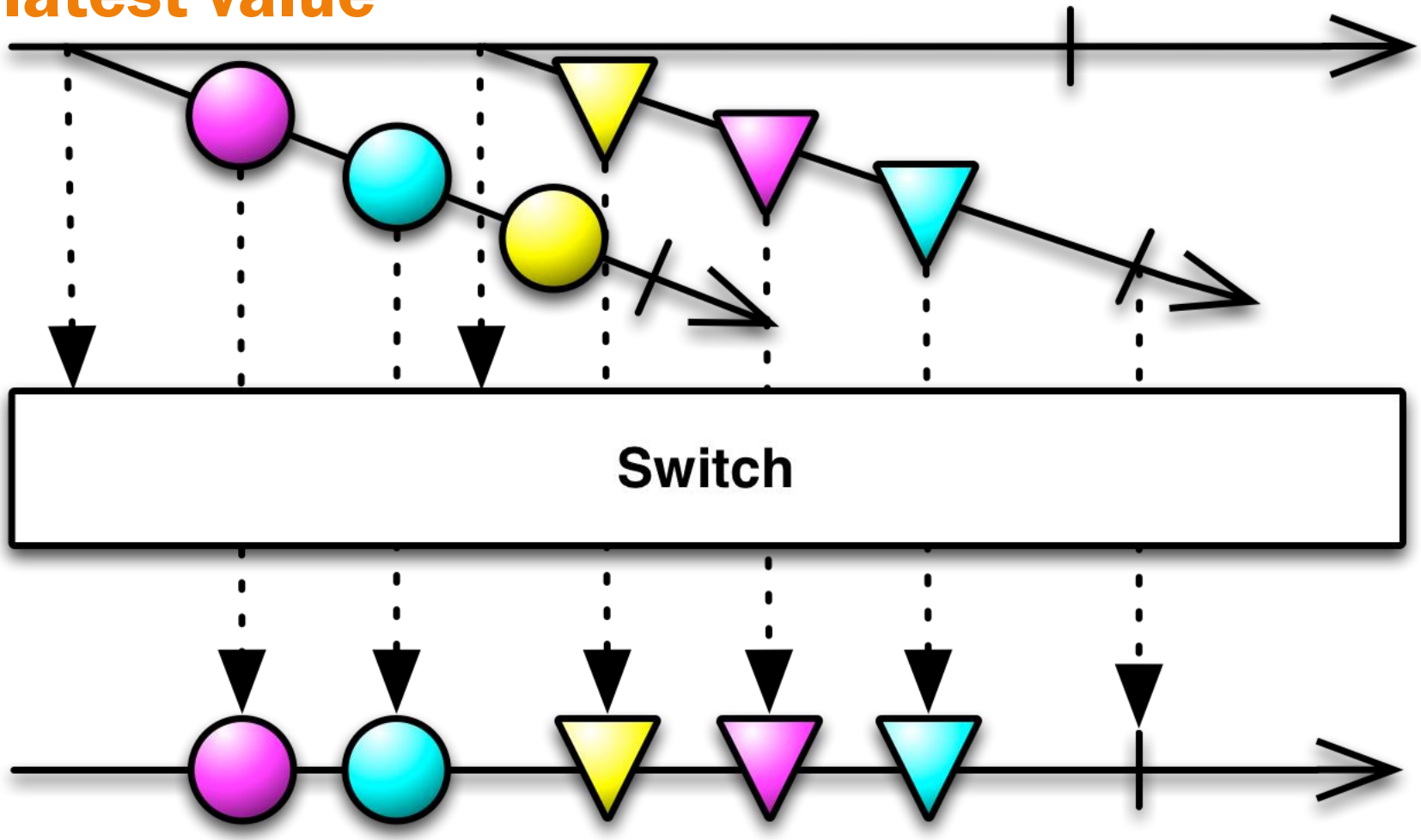
debounce()

Only emit an item from an Observable if a particular timespan has passed without it emitting another item



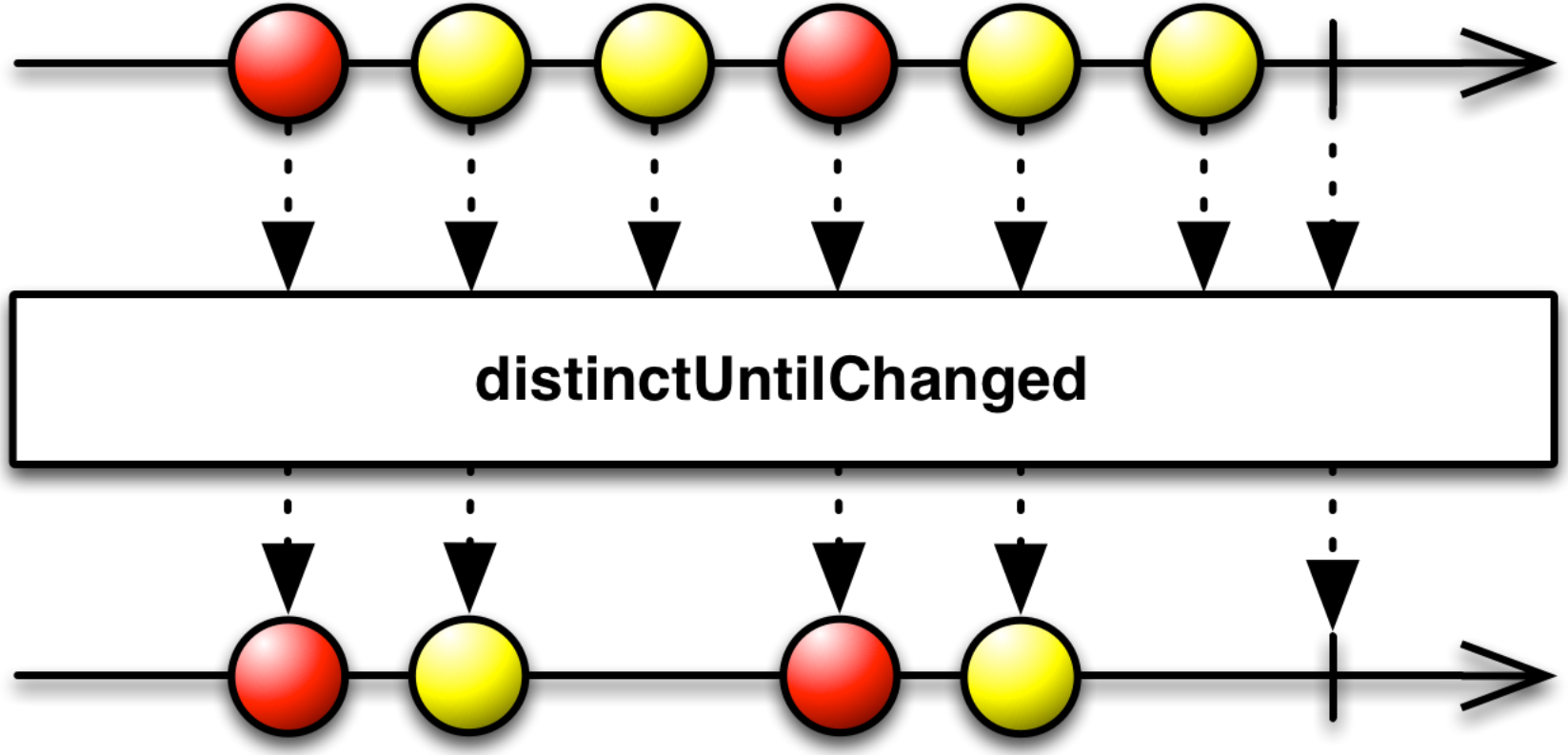
switch()

Convert a collection of collections into a single collection with the latest value



distinctUntilChanged()

Compares against previous value to only yield new distinct items



Netflix with Observables

```
var data = dom.keyup(input)
             .map(function() { return input.value; })
             .debounce(500)
             .distinctUntilChanged()
             .flatMapLatest(
                function(term) { return search(term); }
             );

data.subscribe(function(data) {
  // Bind data to the UI
});
```

DOM events as a sequence of strings

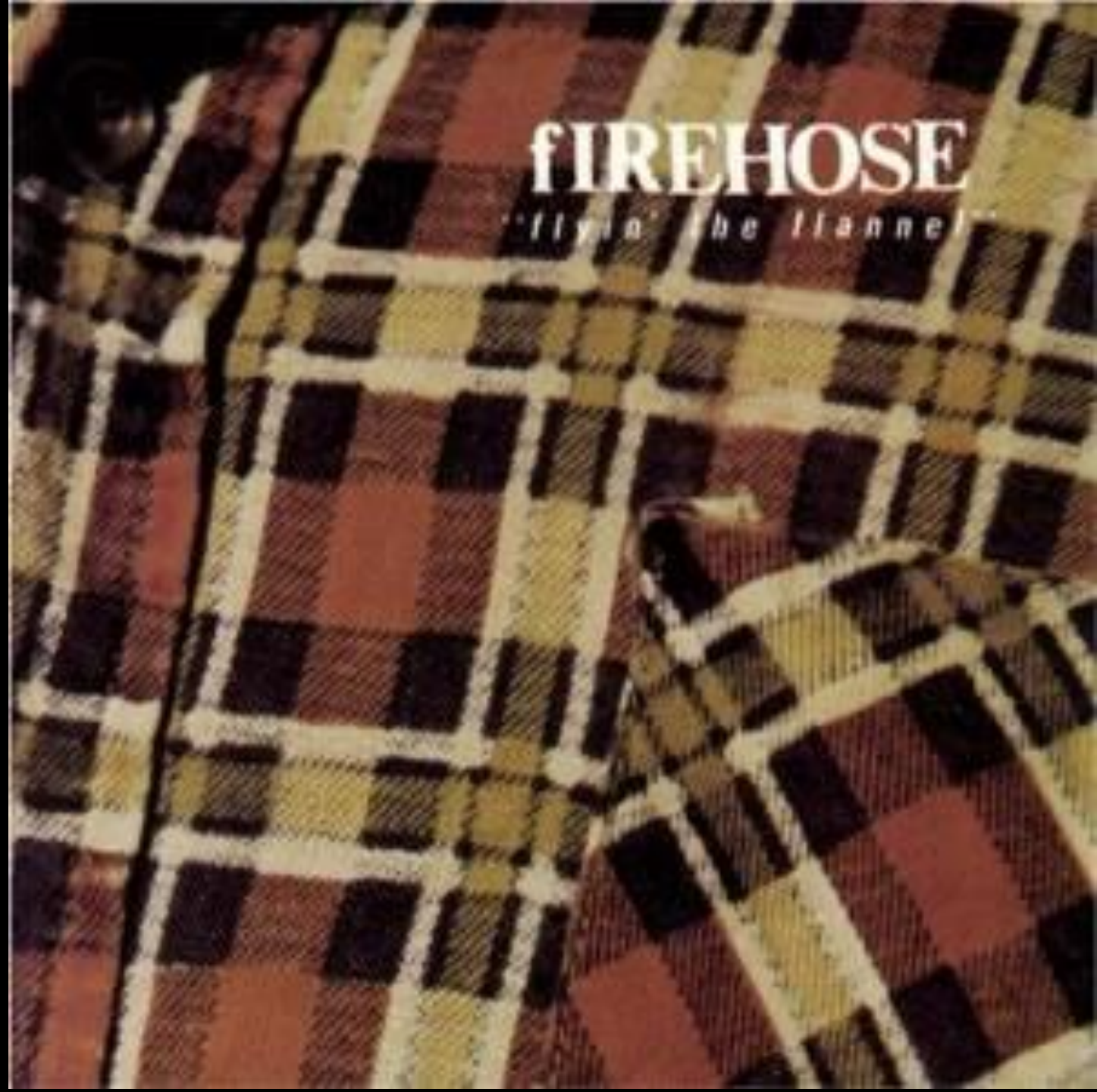
Reducing data traffic / volume

Latest response as movies

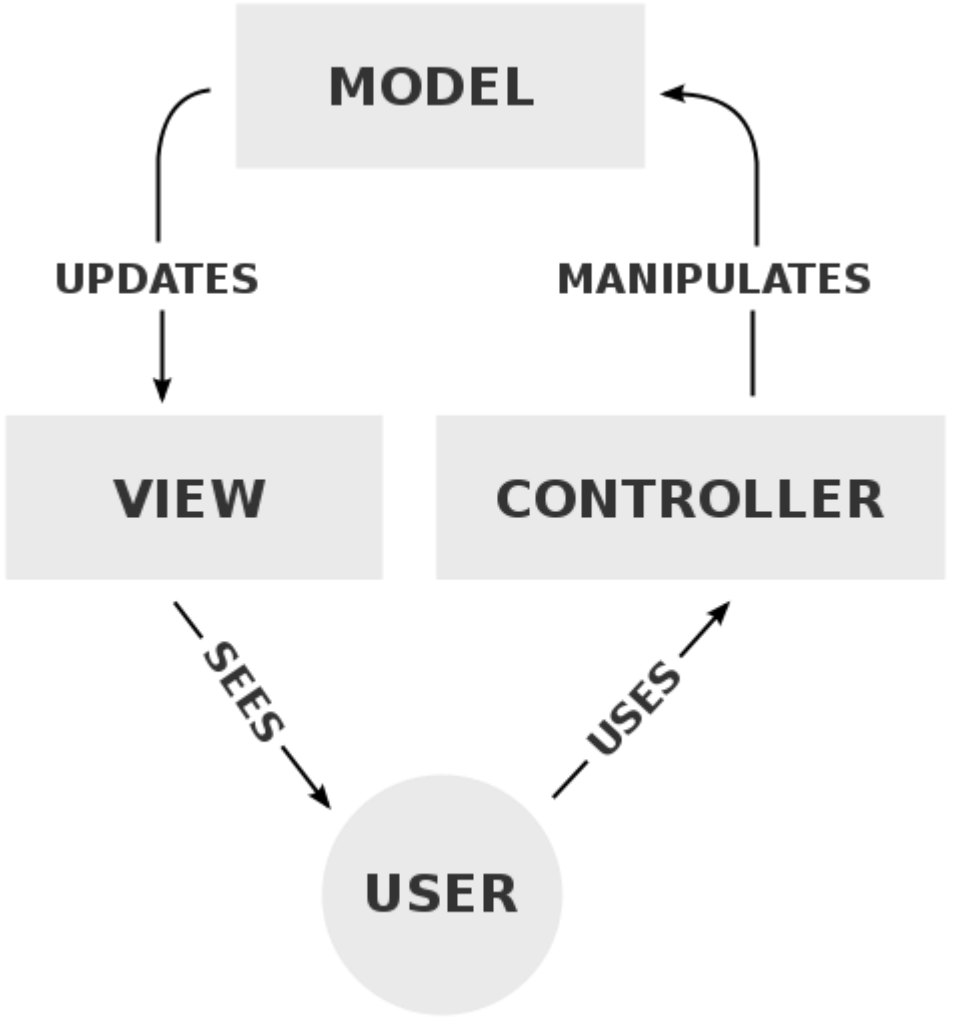
Web service call returns single value sequence

Binding results to the UI

Patterns Emerge...



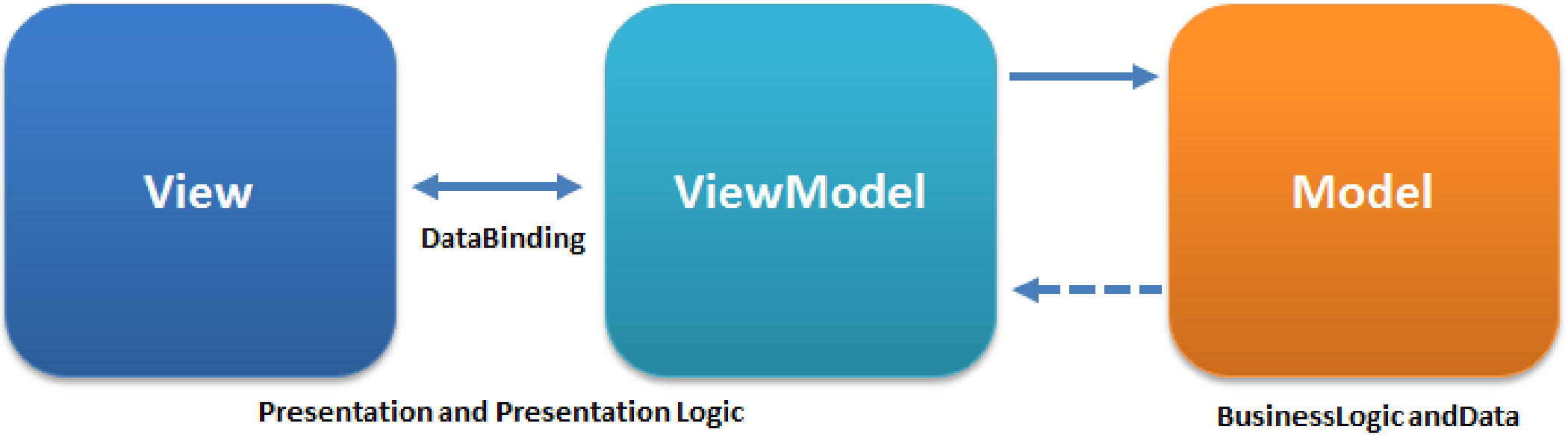
Model-View-Controller (MVC)



BACKBONE.JS

```
<script type="text/template" id="search_template">
  <label>Search</label>
  <input type="text" id="search_input" />
  <input type="button" id="search_button" value="Search" />
</script>

<div id="search_container"></div>

<script type="text/javascript">
    var SearchView = Backbone.View.extend({
        initialize: function(){
            this.render();
        },
        render: function(){
            var template = _.template( $("#search_template").html(), {} ); this.$el.html( template );
        }
    });

    var searchView = new SearchView({ el: $("#search_container") });
</script>
```

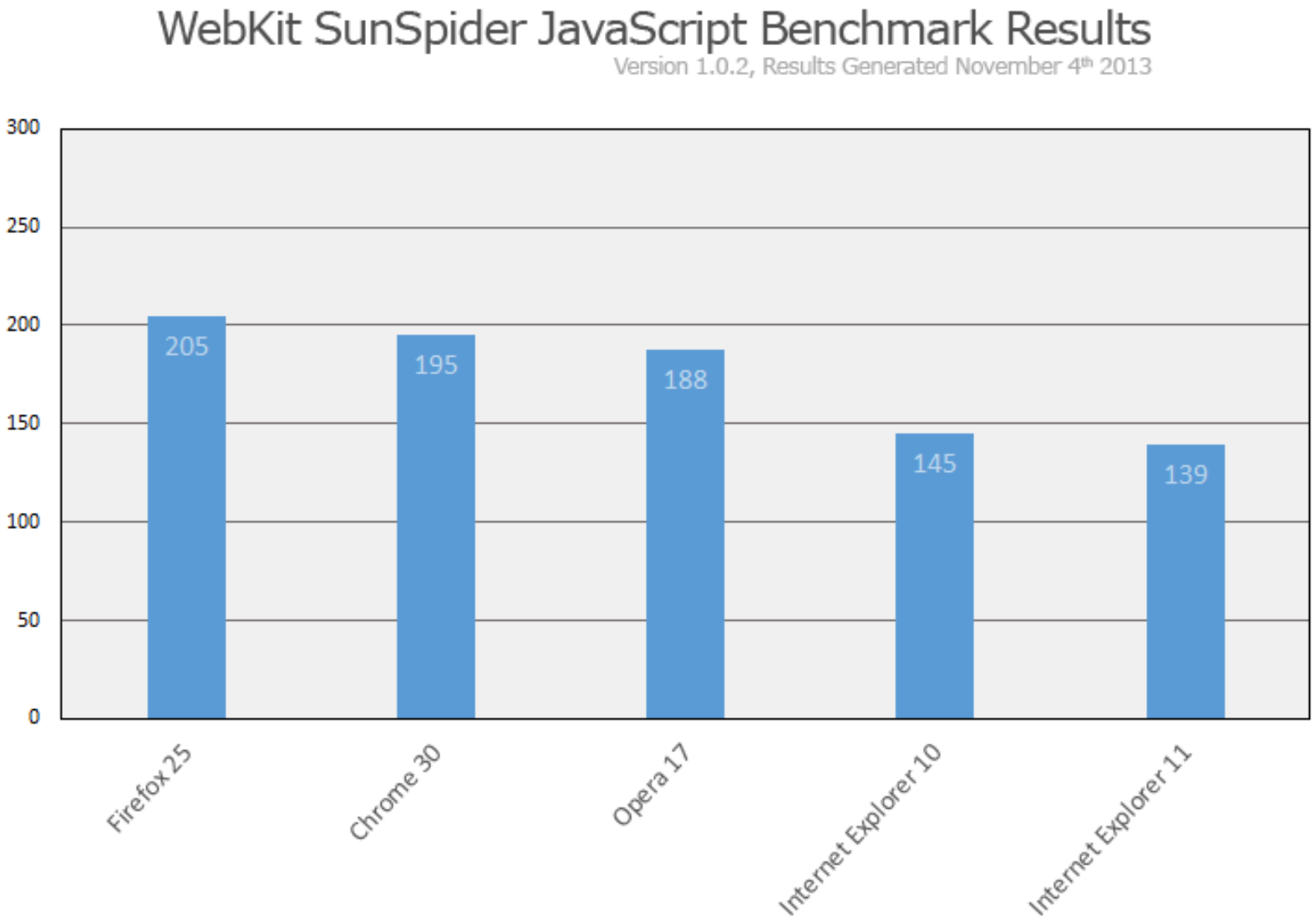
Model-View-ViewModel (MVVM)



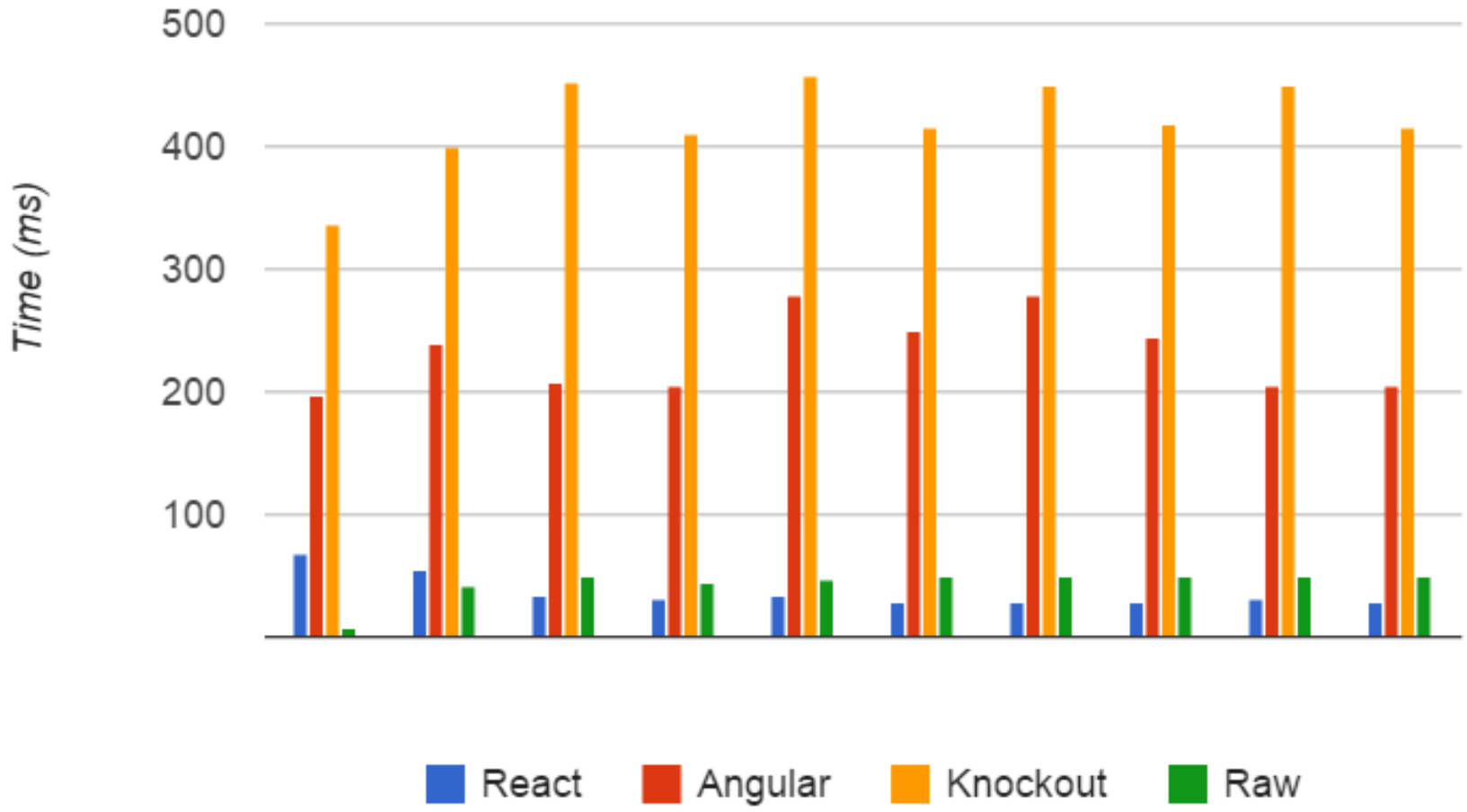

```
<p>First name: <input data-bind="value: firstName" /></p>
<p>Last name: <input data-bind="value: lastName" /></p>
<h2>Hello, <span data-bind="text: fullName" > </span>!</h2>


var ViewModel = function (first, last) {
    this.firstName = ko.observable(first);
    this.lastName = ko.observable(last);

    this.fullName = ko.pureComputed(function () {
        return this.firstName() + " " + this.lastName();
    }, this);
};

ko.applyBindings(new ViewModel("Planet", "Earth"));
```


```
<!doctype html>
<html data-ng-app="">
  <head>
    <script src="angular. js"></script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" data-ng-model="yourName" placeholder="Enter a name here">
      <hr>
      <h1>Hello {{yourName}}!</h1>
    </div>
  </body>
</html>
```

JavaScript Engines Got Faster...



But the DOM didn’t…

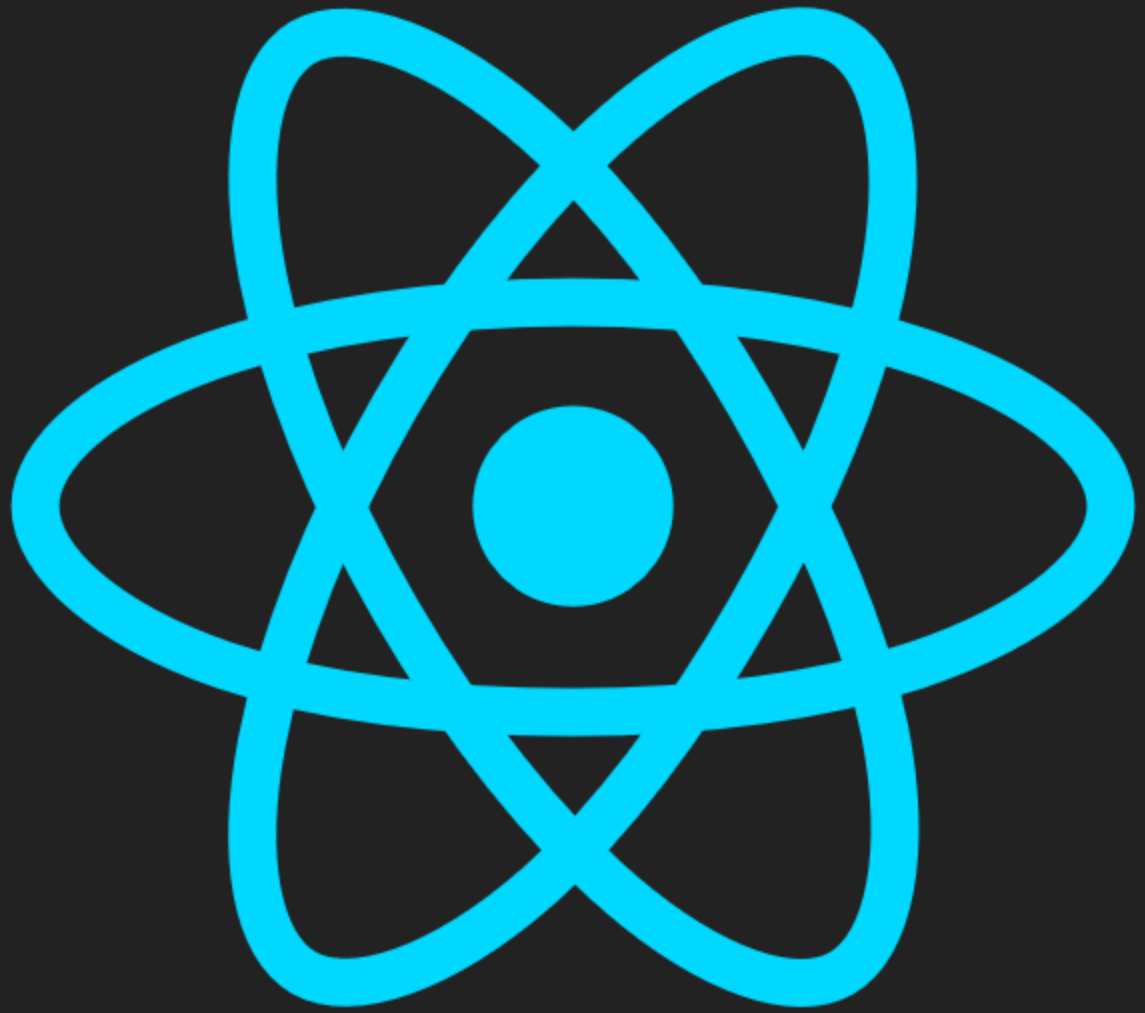


virtual-dom

A JavaScript DOM model supporting element creation, diff computation and patch operations for efficient re-rendering

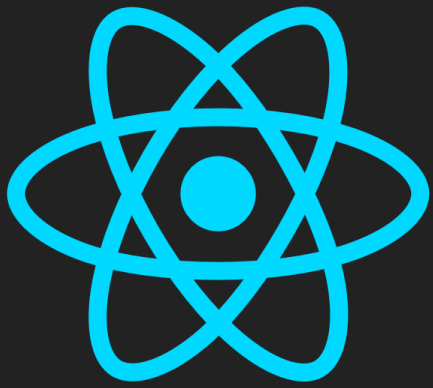
build passing | npm package 1.3.0 | coverage 97% | dependencies up to date | stability experimental

https://github.com/Matt-Esch/virtual-dom



http://facebook.github.io/react/

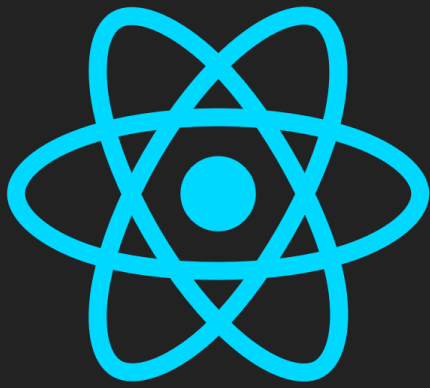
React



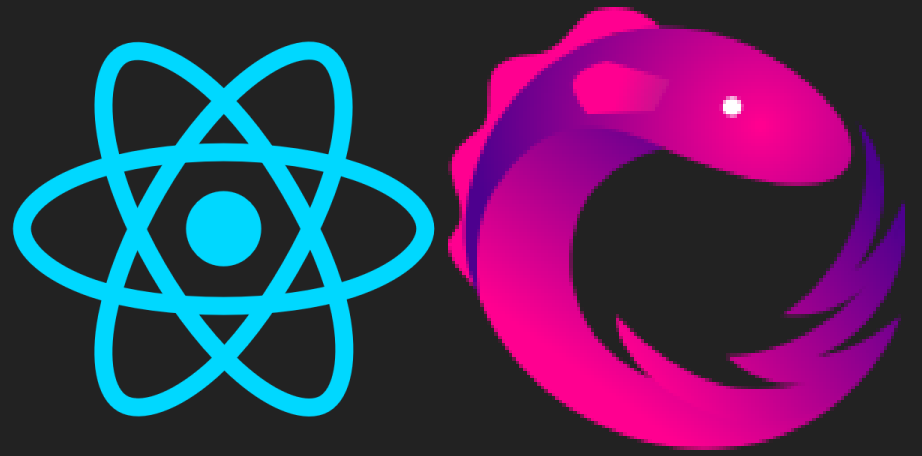
Key Concepts

- Just the V in MVC
- Templates written in JS or JSX Template Language
  - JSX can be rendered on the fly...
  - Precompiled into JavaScript
- Component Based
- Uses a Virtual-DOM tree as view representation
- Changes to the model are rendered automatically
- Browser DOM updates minimal instead of whole refresh
- Can be written on the client and the server

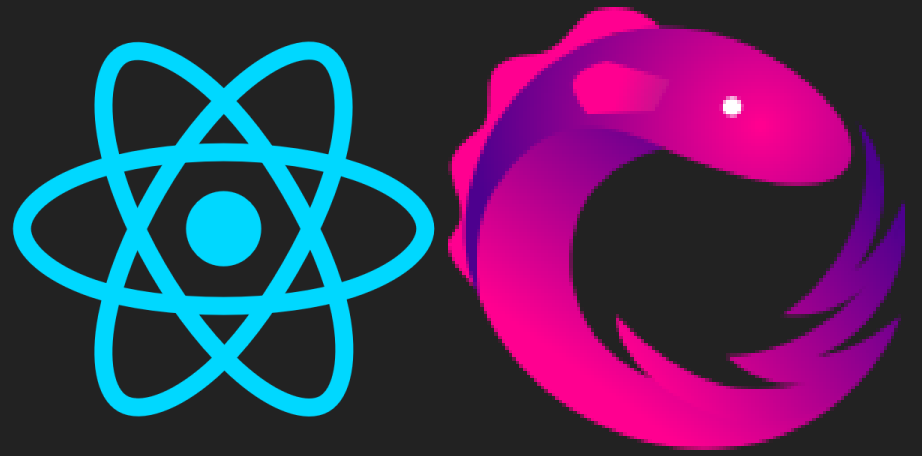
Hello JSX!



```
<!DOCTYPE html>
<html>
<head>
    <script src="http://fb.me/react.js"></script>
    <script src="http://fb.me/JSXTransformer.js"></script>
</head>
<body>
<div id="example"></div>
<script type="text/jsx">
      /** @jsx React.DOM */
      React.renderComponent(
        <h1>Hello, world!</h1>,
        document.getElementById('example')
      );
    </script>
</body>
</html>
```



```
var Timer = React.createClass({
  getInitialState: function() {
    return {secondsElapsed: 0};
  },
  tick: function() {
    this.setState({secondsElapsed: this.state.secondsElapsed + 1});
  },
  componentDidMount: function() {
    this.sub = Rx.Observable.interval(1000).subscribe(this.tick);
  },
  componentWillUnmount: function() {
    this.sub.dispose();
  },
  render: function() {
    return (
      <div>Seconds Elapsed: {this.state.secondsElapsed}</div>
    );
  }
});
```



Rx-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>
    );
  }
}
```

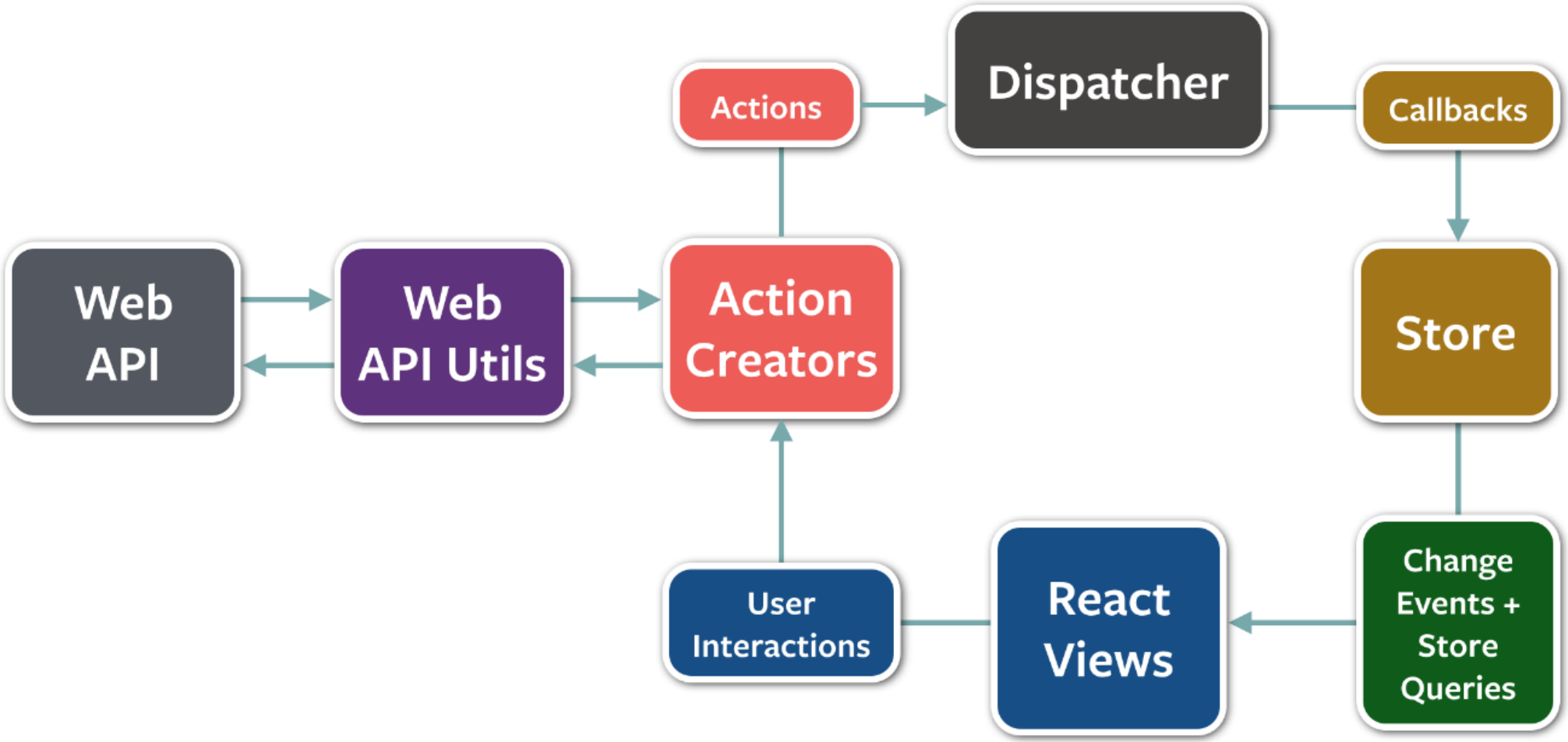
https://github.com/fdecampredon/rx-react

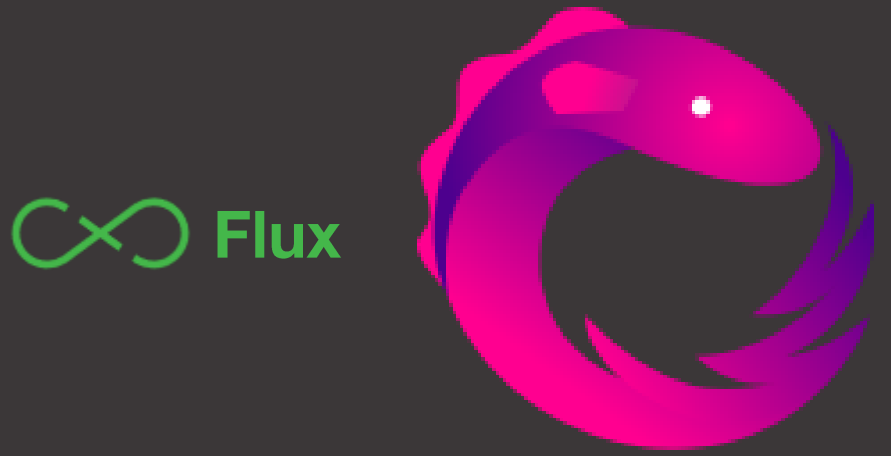
Event-Driven Pattern for Client-Side Applications

- Unidirectional data flow unlike MVC
- Three Major Parts
  - Dispatcher
  - Stores
  - React Components
- Many implementations
  - Facebook Flux
  - Yahoo Fluxible

http://facebook.github.io/flux/

Flux Architecture






```
var Store = require('rx-flux').Store;

class MyStore extends Store {
  constructor(value) {
    super();
    this.setValue(value);
  }
};

var myStore = new MyStore([]);

var operation1 = myStore.applyOperation(function (value) {
  return value.concat('foo');
});

operation1.cancel();
```

https://github.com/fdecampredon/rx-flux

RxJS + React + Flux

- Rx-React (https://github.com/fdecampredon/rx-react)
- Rx-Flux (https://github.com/fdecampredon/rx-flux)
- ReactiveFlux (https://github.com/codesuki/reactive-flux)
- RxReact (https://github.com/AlexMost/RxReact)

Cycle.js

Model-View-Intent architecture and Virtual-DOM Rendering

```
var Cycle = require('cyclejs');
var h = Cycle.h;

var Model = Cycle.createModel(Intent =>
  ({name$: Intent.get('changeName$').startWith('')})
});

var View = Cycle.createView(Model =>
  ({
    vtree$: Model.get('name$').map(name =>
      h('div', [
        h('label', 'Name:'),
        h('input.field', {attributes: {type: 'text'}}),
        h('h1.header', 'Hello ' + name)
      ])
    )
  })
);
```

Ractive.js

The diamond age
of web development

```
<script src='http://cdn.ractivejs.org/latest/ractive.js'></script>
```

```
var ractive = new Ractive({
  el: output,
  template: template,
  data: {
        greeting: 'Hello',
        name: 'world',
        color: 'purple',
        size: 4
  }
});
```

```
<p style='color: {{color}}; font-size: {{size}}em;;'>
  {{greeting}} {{name}}!
</p>
```

Ractive.js

The diamond age
of web development

```
<script src='http://cdn.ractivejs.org/latest/ractive.js'></script>
```

```
Rx.Observable.fromEvent(ractive, 'refresh')
  .startWith(null) // Simulate startup click
  .map(function () { return Math.floor(Math.random() * 500); })
  .map(function (offset) { return 'https://api.github.com/users?since=' + offset; })
  .flatMap($.getJSON)
  .subscribe(function (users) {
      ractive.set('users', users);
      for (var i = 0; i < maxSuggestions; i++)
        setRandomSuggestion(ractive, i);
  });
```

onNext('Questions?');
onCompleted();

Matthew Podwysocki @mattpodwysocki

github.com/mattpodwysocki/codeonthesea2015