RxJS for f2e

who

```
{
    "name": "ShihChi Huang",
    "work": "mobile engineer @ Netflix",
    "meta": "huang47 @ github/npm/twitter"
}
```

*.search.yahoo.com

Spotify®

iOS App

NETFLIX

NOT fb/React

async is hard

sync

```
function getValueSync(value) {
    // value is immediately available
    return value + 1;
}

var newValue = getValueSync(3);
// newValue: 4
```

async

```
function getValue(value) {
    setTimeout(function () {
        // how to return result?
        var result = value + 1;
    }, 100);
}

var newValue = getValue(3);
// newValue: undefined
```

callback

```
function getValue(value, callback) {
    setTimeout(function () {
        callback(value + 1);
    }, 100);
}

getValue(3, function callback(newValue) {
    // newValue: 4
});
```

callback seems simple

what's actually hard?

v1.1.6

 **isaacs** authored 23 days ago — latest commit `8345e51ee4`

lib	wait for all callbacks to complete	23 days ago
LICENSE	isc license	a year ago
README.md	Replace pdf by README.md.	a year ago
index.js	Just expose a main module, not a lib dir	4 years ago
package.json	v1.1.6	23 days ago

README.md

Controlling Flow: callbacks are easy

What's actually hard?

- Doing a bunch of things in a specific order.
- Knowing when stuff is done.
- Handling failures.
- Breaking up functionality into parts (avoid nested inline callbacks)

example

```
var searchbox = $('input[type="search"]');
// getInput
searchbox.addEventListener('input', function (e) {
    var query = e.target.value.trim();

    // getSearchResults
    xhr(encodeURIComponent(query), function (results) {

        // renderSearchResults
        results.forEach(renderSearchResult);
    });
});
```

it's bad to fire http request
for every single value

throttle

-> getInput
-> throttle input (250ms)
-> getSearchResults
-> renderSearchResults

throttle

```
var throttleId;

// getInput
searchbox.addEventListener('input', function (e) {
    var query = e.target.value.trim();

    // throttle
    if (throttleId) { clearTimeout(throttleId); }

    throttleId = setTimeout(function () {
        // getSearchResults
        xhr(encodeURIComponent(query), function (results) {
            results.forEach(renderSearchResult);
        });
    }, 250);
});
```

retry few times if request timeout

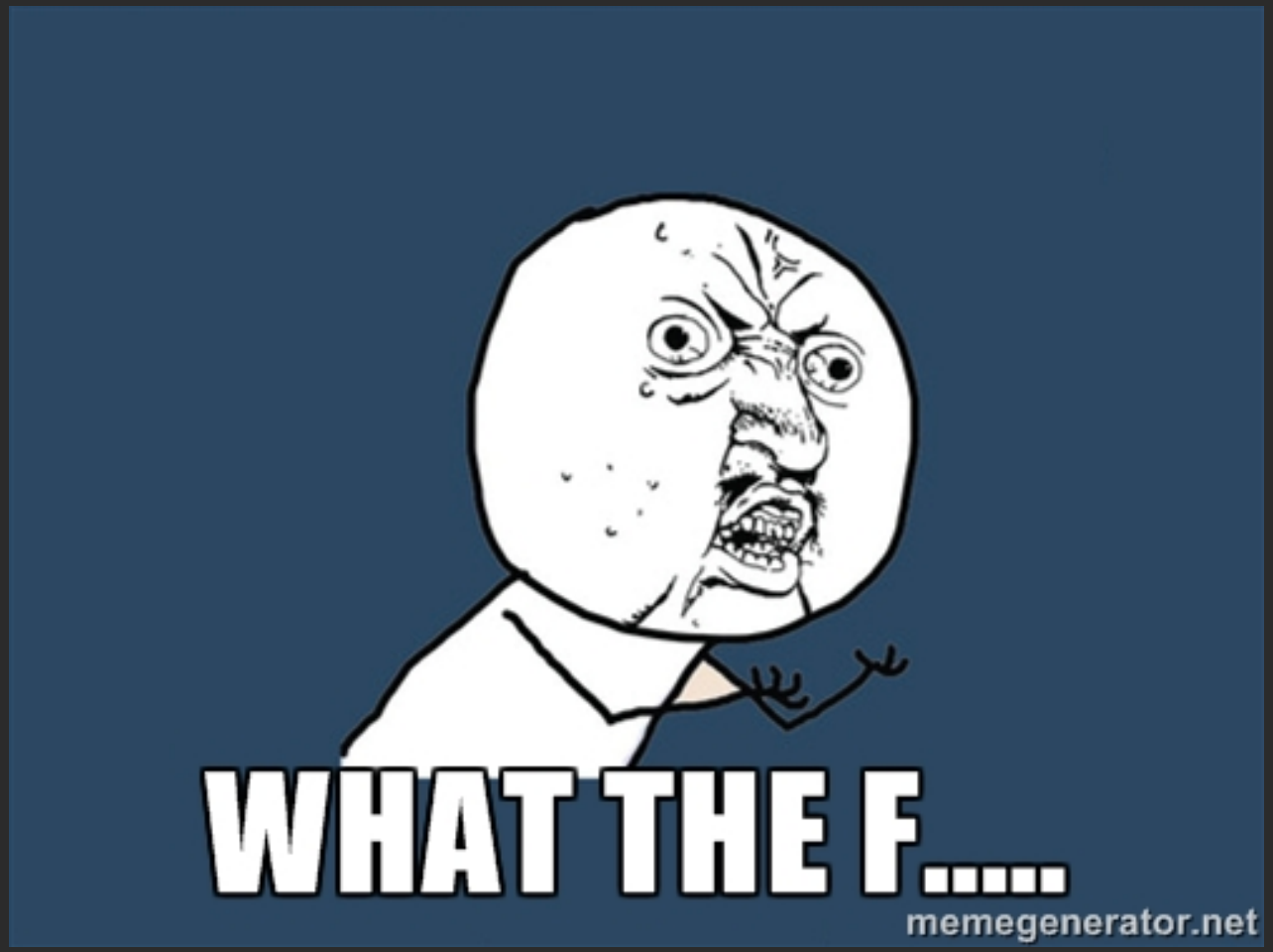
retry

-> getInput
-> throttle input (250ms)
-> getSearchResults
-> retry (2) times if failed
-> renderSearchResults (if latest)

retry

```
var retryCount = 0, throttleId, timeoutId;
searchbox.addEventListener('input', function (e) {
    var query = e.target.value.trim();
    function request(str) {
        xhr(encodeURIComponent(str), function (results) {
            retryCount = 0;
            clearTimeout(timeoutId);
            results.forEach(renderSearchResult);
        });
    }
    if (throttleId) { clearTimeout(throttleId); }
    throttleId = setTimeout(function () {
        timeoutId = setTimeout(function () {
            if (retryCount < 2) {
                retryCount += 1;
                request(query);
            } else {
                retryCount = 0; // attempt exhausted
            }
        }, 3000);
    }, 250);
});
```

can we....


WHAT THE F.....
memegenerator.net

promise?

is good for *single value*

however, in event-driven world

we need a better way to deal with collections

Jafar Husain
@jhusain

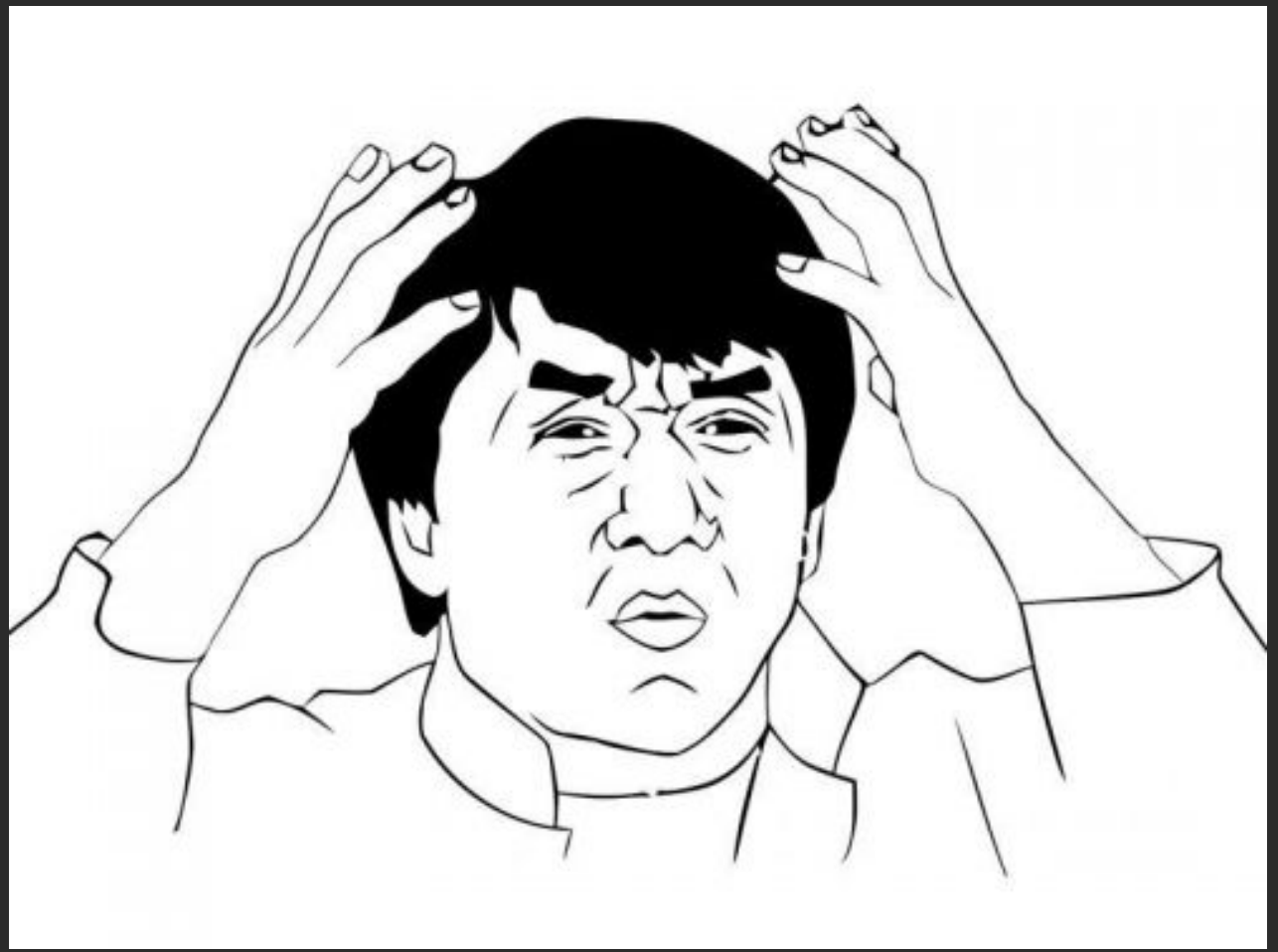
Follow

Using Promises for concurrency is like bringing a knife to a gunfight. Learn Rx. jhusain.github.io/learnrx

11:14 AM - 15 Nov 2013

3 FAVORITES

what's the difference between an Array and an Event ?



Collections !

```
// Array
[ { x: 10 }, { x: 20 }, { x: 30 } ]

// Event
{ pageX: 10, timeStamp: 238 },
{ pageX: 20, timeStamp: 375 },
{ pageX: 30, timeStamp: 577 },
...
```

before dive in

Array warm up

ES6

```
function (x) { return x + 1; }
```

```
x => x + 1
```

forEach

```
[1, 2, 3].forEach(v = > console.log(v))
```

```
> [1, 2, 3]
```

map

```
[1, 2, 3].map(v => v + 1);
```

> [2, 3, 4]

filter

```
[1, 2, 3].filter(v => v > 1);
```

> [2, 3]

reduce

```
[1, 2, 3].reduce(acc, curr =>
    return acc + curr;
, 0);
```

> acc: 0, curr: 1
> acc: 1, curr: 2
> acc: 3, curr: 3
> 6

concatAll

```
[
    [1],
    [2, 3],
    [],
    [4]
].concatAll();
```

> [1, 2, 3, 4]

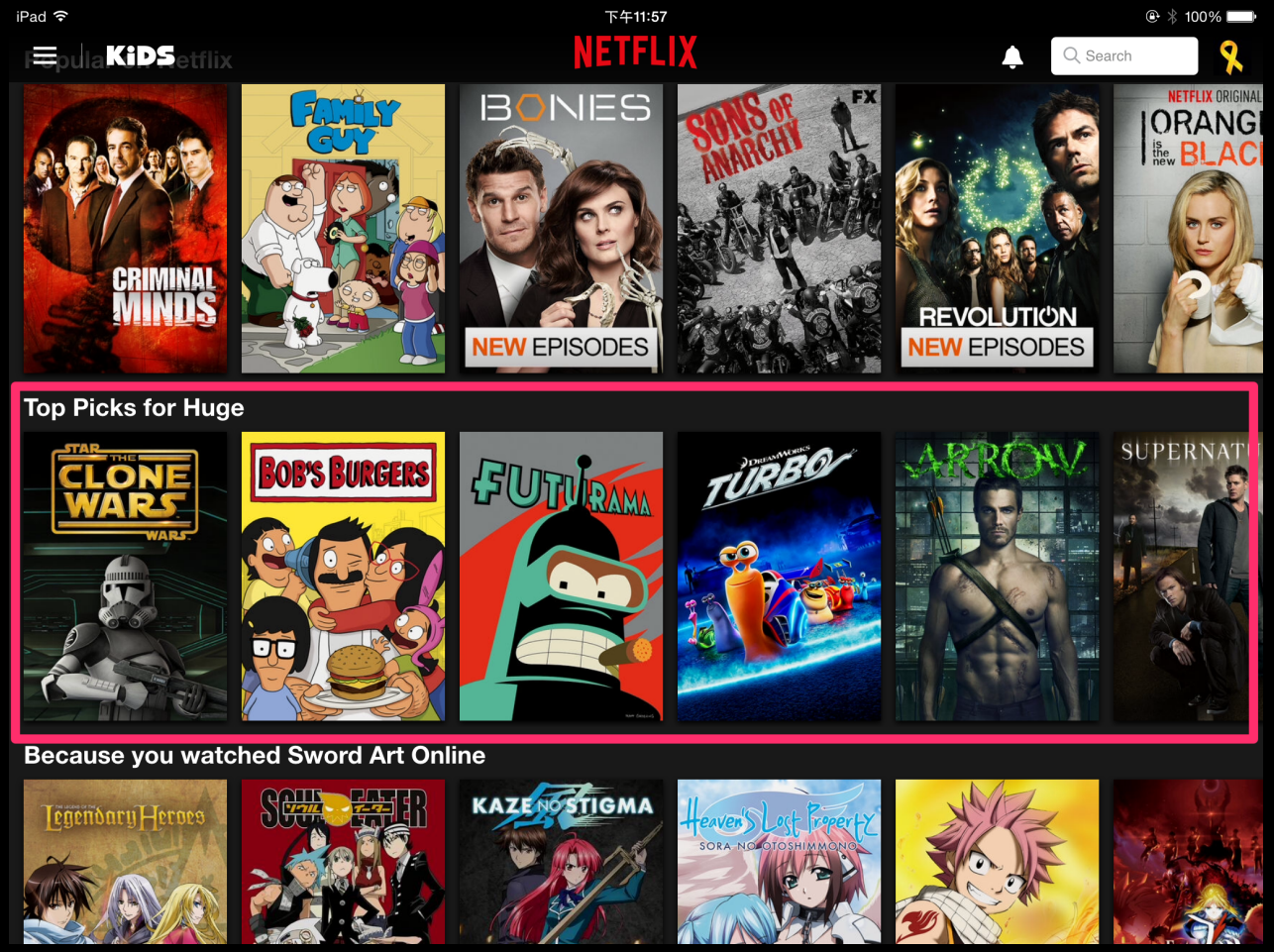
concatAll'

```
Array.prototype.concatAll = function concatAll() {
    return this.reduce(function (a, c) {
        return Array.prototype.concat.call(a, c);
    }, []);
};
```

recap - map/filter/concatAll

```
[1, 2, 3].forEach(v => console.log(v));
// [1, 2, 3]

[1, 2, 3].map(v => v + 1);
// [2, 3, 4]

[1, 2, 3].filter(v => v > 1);
// [2, 3]

[1, 2, 3].reduce(acc, curr => acc + curr, 0);
// 6

[[1], [2, 3], [], [4]].concatAll();
// [1, 2, 3, 4]
```

:0
deep breath


iPad
下午11:57
100%
KiDS
NETFLIX
Search
CRIMINAL MINDS
FAMILY GUY
BONES
NEW EPISODES
SONS OF ANARCHY
FX
REVOLUTION
NEW EPISODES
NETFLIX ORIGINAL
is the new
Top Picks for Huge
STAR WARS THE CLONE WARS
BOB'S BURGERS
FUTURAMA
DREAMWORKS TURBO
ARROW
Because you watched Sword Art Online
Legendary Heroes
KAZE NO STIGMA
Heaven's Lost Property
SORA NO OTOSHIMMONO

json

```
{ "videoLists": [
    // videoList
    { "category": "Recommend",
      "videos": [
        { "title": "Clone Wars", "rating": 5 },
        { "title": "How I Met Your Mother", "rating": 4 },
        { "title": "Turbo", "rating": 5 },
        { "title": "Arrow", "rating": 5 }
    ] },
    // videoList
    { "category": "Social Feed",
      "videos": [
        { "title": "Bob's Burgers", "rating": 5 },
        { "title": "World War Z", "rating": 4 },
        { "title": "Futurama", "rating": 5 },
        { "title": "Supernature", "rating": 5 }
    ] }

] }
```

getTopRatedVideos

```
var getTopRatedVideos = user =>
    user.videoLists.
        map(videoList =>
            videoList.videos.
                filter(video =>
                    5 === video.rating;
                );
        ).
        concatAll();
```

output


```
[
    { "title": "Clone Wars", "rating": 5 },
    { "title": "Bob's Burgers", "rating": 5 },
    { "title": "Futurama", "rating": 5 },
    { "title": "Turbo", "rating": 5 },
    { "title": "Arrow", "rating": 5 },
    { "title": "Supernature", "rating": 5 }
]
```

what if I told you
we can create a drag event
with nearly the same code?

getTopRatedVideos

```
var getTopRatedVideos = user =>
    user.videoLists.
        map(videoList =>
            videoList.videos.
                filter(video =>
                    5 === video.rating;
                );
        ).
        concatAll();
```

getMouseDrags

```
var getMouseDrags     = elem =>
    elem.mouseDowns.
        map(mouseDown =>
            elem.mouseMoves.
                takeUntil(
                    elem.mouseUps
                );
        ).
        concatAll();
```

like.. SQL

```
select video
from user.videos
where video.rating = 5
```

```
select *
from mousedown, mousemove, mouseup
where type != 'mouseup'
```

Observable

Collection + Time

Event

```
var mouseMoves = Rx.Observable.
    fromEvent(document.body, 'mousemove');

// addEventListener
var subscription = mouseMoves.
    forEach(function (mouseMove) {
        // handle mouse move
    });

// removeEventListener
subscription.dispose();
```

Event

```
mouseMoves.
    forEach(
        function onNext(mouseMove) {
            // handle mouse move
        },
        function onError(ex) {      // optional
            // handle error
        },
        function onCompleted() {    // optional
            // event completed
        }
    );
```

Observable can be

event
xhr, async I/O
promise
node-stream
iterable (Array)
generator

Observable literal

```
Time --------------->
{1......2............3}
```

```
{ ---> observable begin
} ---> observable end
... -> time
```

forEach

```
Time --------------->
{1.....2...........3}.forEach(v => console.log(v));

{1.....2...........3}
```

map

```
Time --------------->
{1......2............3}.map(x => x + 1);

{2......3............4}
```

filter

```
Time --------------->
{1......2............3}.filter(x => x > 1);

{ ......2............3}
```

distinctUntilChanged

```
Time ---------------->
{1......2...2......3}.distinctUntilChanged();

{1......2... ......3}
```

concatAll

```
Time --------------->
{
...{1},
.........{2.............3},
..............{},
................{4}
}.concatAll();

{...1.....2.............3...4}
```

mergeAll

```
Time --------------->
{
...{1},
.........{2.............3},
..............{},
................{4}
}.concatAll();

{...1.....2........4......3}
```

switchLatest

```
Time --------------->
{
...{1},
.........{2.............3},
..............{},
................{4}
}.concatAll();

{...1.....2........4}
```

takeUntil

```
Time --------------->
{...1...2.............3}.takeUntil(
{...............4})

{...1...2........}
```

search

```
var searchbox = $('input[type="search"]');

searchbox.addEventListener('input', function (e) {
    var query = e.target.value.trim();

    // fire request
    xhr(encodeURIComponent(query), function (results) {

        // render search results
        results.forEach(renderSearchResult);
    });
});
```

Rxify

```
var searchbox = $('input[type="search"]');
var inputs = Rx.Observable.fromEvent(searchbox, 'input');


inputs.
    map(function (e) {
        // get input string
        var query = e.target.value.trim();
        // construct url
        var url = '/search?' + encodeURIComponent(query);
        // fire request
        return Rx.DOM.Request.getJSON(url).
            takeUntil(inputs);
    }).
    concatAll();
```

response

```
Time --------------->
{
...{s(h)},
......{s(he)},
.........{s(hel)},
............{s(hell)}
..............{s(hello)}
}

// where s(x) is the output with given input `x`
```

too young too simple

real world

```
Time --------------->
{
...{s(h)},
......{s(he)},
.........{s(hel)},
.........................{s(hell)},
............{s(hello)}
}
```

Rxify

```
var searchbox = $('input[type="search"]');
var inputs = Rx.Observable.fromEvent(searchbox, 'input');


inputs.
    map(function (e) {
        // get input string
        var key = e.target.value.trim();
        // construct url
        var url = '/search?' + encodeURIComponent(key);
        // fire request
        return Rx.DOM.Request.getJSON(url).
            takeUntil(inputs);
    }).
    concatAll();
```

switchLatest

```
var searchbox = $('input[type="search"]');
var inputs = Rx.Observable.fromEvent(searchbox, 'input');


inputs.
    map(function (e) {
        // get input string
        var key = e.target.value.trim();
        // construct url
        var url = '/search?' + encodeURIComponent(key);
        // fire request
        return Rx.DOM.Request.getJSON(url);

    }).
    switchLatest();
```

final

```
var searchbox = $('input[type="search"]');
var inputs = Rx.Observable.fromEvent(searchbox, 'input');
inputs.
    throttle(250).
    map(function (e) {
        return e.target.value.trim();
    }).
    distinctUntilChanged().
    map(function (query) {
        var url = '/search?' + encodeURIComponent(key);
        // fire request
        return Rx.DOM.Request.getJSON(url).
            retry(3);
    }).
    switchLatest();
```

recap

everything is Observable

Observable is Collection + Time

forEach,map,filter,reduce,concatAll

references

End to End Reactive Programming at Netflix
http://jhusain.github.io/learnrx

{Q...U...E...S...T...I...O...N...S...?}