

RxJS for f2e

who

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

async is hard

sync

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

**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
```

code

```
var throttleId;

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

  if (throttleId) { clearTimeout(throttleId); }

  throttleId = setTimeout(function () {
    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)

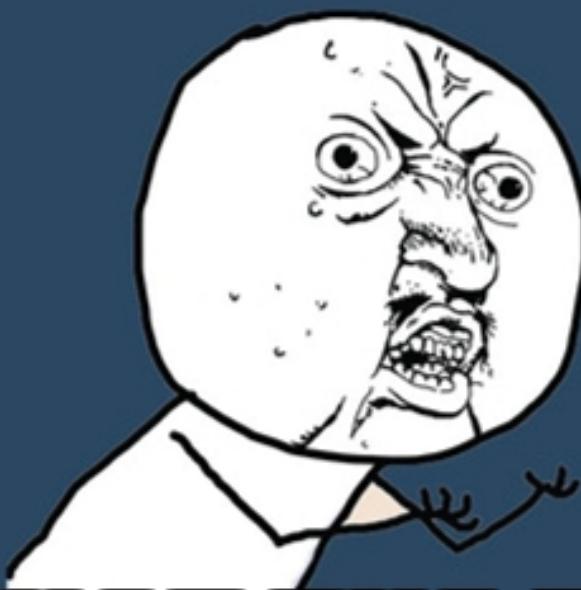
code

```
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....



memegenerator.net



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

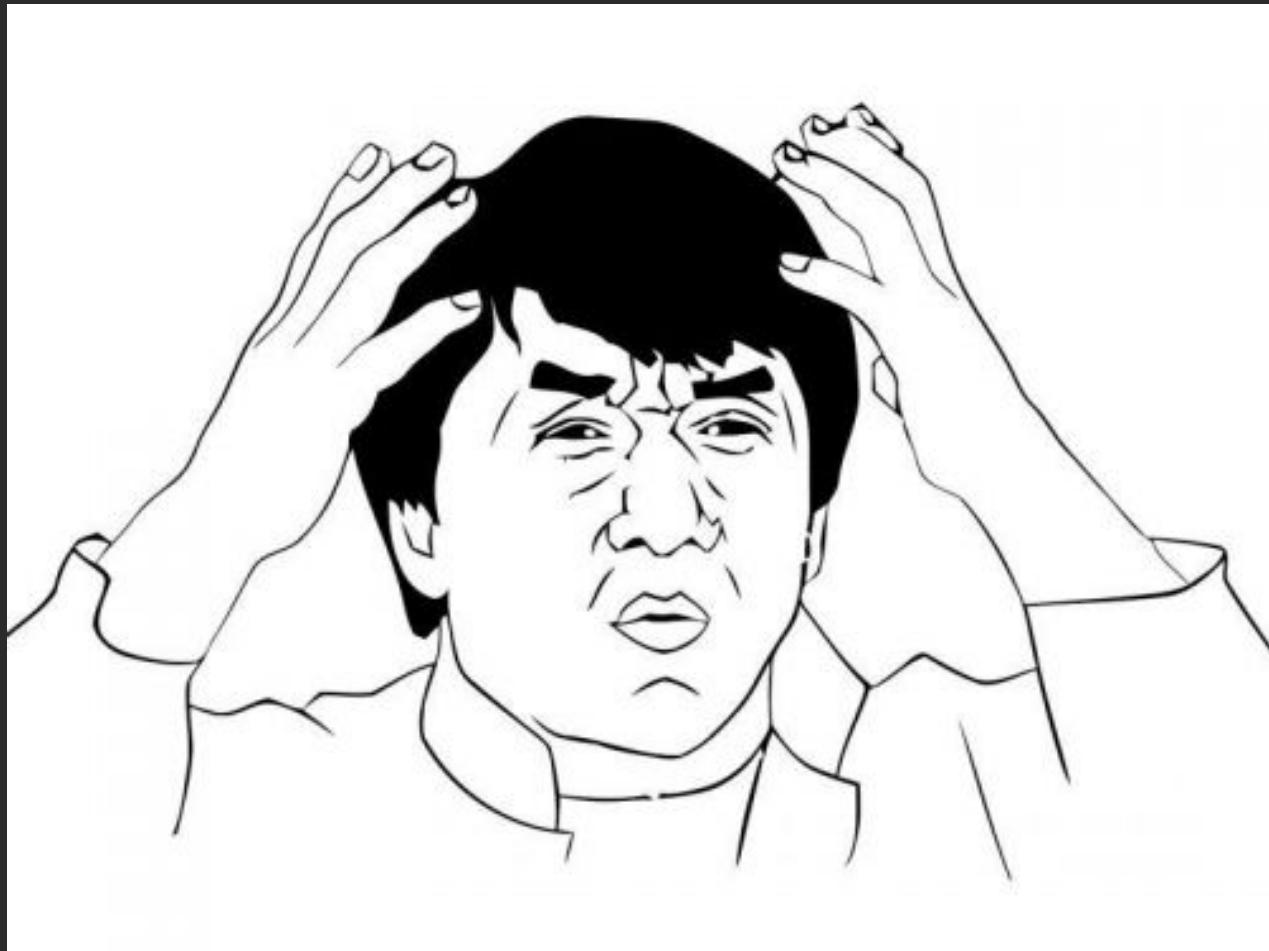
7:14 PM - 14 Nov 2013

3 FAVORITES

1



**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

forEach

```
[1, 2, 3].forEach(function (v) {  
  console.log(v);  
});
```

› [1, 2, 3]

map

```
[1, 2, 3].map(function (v) {  
    return v + 1;  
});
```

➤ [2, 3, 4]

filter

```
[1, 2, 3].filter(function (v) {  
    return v > 1;  
});
```

➤ [2, 3]

concatAll

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

➤ [1, 2, 3, 4]

concatAll

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

recap - map/filter/concatAll

```
[1, 2, 3].map(function (v) {  
    return v + 1;  
});  
// [2, 3, 4]  
  
[1, 2, 3].filter(function (v) {  
    return v > 1;  
});  
// [2, 3]  
  
[[1], [2, 3], [], [4]].concatAll();  
// [1, 2, 3, 4]
```

iPad

下午11:57

100%

epula Kids Netflix

NETFLIX



Search



Top Picks for Huge



Because you watched Sword Art Online



json

```
{ "videoLists": [
    { "category": "Recommend",
      "videos": [
          { "title": "Clone Wars", "rating": 5 },
          { "title": "How I Met Your Mother", "rating": 4 },
          { "title": "Turbo", "rating": 5 },
          { "title": "Arrow", "rating": 5 }
      ] },
    { "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

```
function getTopRatedVideos(user) {  
    return user.videoLists.  
        map(function (videoList) {  
            return videoList.videos.  
                filter(function (video) {  
                    return 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

```
function getTopRatedVideos(user) {  
    return user.videoLists.  
        map(function (videoList) {  
            return videoList.videos.  
                filter(function (video) {  
                    return 5 === video.rating;  
                });  
        }).  
        concatAll();  
}
```

getMouseDrags

```
function getMouseDrags(elem) {  
    return elem.mouseDowns.  
        map(function (mouseDown) {  
            return 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
- enumerable (Array)
- generator

Observable literal

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

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

forEach

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

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

map

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

➤ {2 ... 3 4}

filter

```
Time ----->
{1.....2.....3}.filter(function (x) {
    return 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}
```

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

mergeAll

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

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

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

switchLatest

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

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

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

takeUntil

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

```
> {...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)}
}
```

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();
```

even more

```
Rx.Observable.fromEvent(searchbox, 'input');
    throttle(250).
    map(function (e) {
        return e.target.value.trim());
    }).
    // drop if value doesn't change
    distinctUntilChanged().
    map(function (query) {
        var url = '/search?' + encodeURIComponent(key);
        // fire request
        return Rx.DOM.Request.getJSON(url).
            retry(3);
    }).
    switchLatest();
```

recap

everything is Observable
use rich composition

questions?

references

End to End Reactive Programming at Netflix