RXJS

A (practical) introduction to RxJS

Utah JS North – 2015-08-18

Seth House @whiteinge

UNIFIED API FOR ASYNC OPERATIONS

- DOM Events
- Ajax Events
- Server-sent Events
- Websockets
- setInterval
- Service Workers
- jQuery events, Node streams, etc.

OBSERVERS

Make an observer (if you're a library author):

```
var Rx = require('rx');

var myobsv = Rx.Observer.create(
    (x) => console.log('onNext', x),
    (err) => console.log('onError', err),
    () => console.log('onCompleted'));

myobsv.onNext(1);
myobsv.onNext(2);
myobsv.onCompleted();
```

OBSERVERS

Get an observer:

```
var myelem = document.querySelector('someel.myelem');
var keyups = Rx.Observable.fromEvent(myelem, 'keyup')
```

OBSERVERS

Get an observer from anything:

```
Rx.Observable.from(...)
Rx.Observable.fromCallback(...)
Rx.Observable.fromEvent(...)
Rx.Observable.fromEventPattern(...)
Rx.Observable.fromNodeCallback(...)
Rx.Observable.fromPromise(...)
Rx.Observable.generate(...)
Rx.Observable.generateWithAbsoluteTime(...)
Rx.Observable.generateWithRelativeTime(...)
```

OBSERVABLES

“Think of an Observable as an asynchronous immutable array.”

```
var mysubscription = myobsv.subscribe(
     (x) => console.log('onNext', x),
     (err) => console.log('onError', err),
     () => console.log('onCompleted'));
```

OBSERVABLES

Stop Listening:

```
mysubscription.dispose();
```

OBSERVABLES

Stop Listening Automatically

```
var myelem = document.querySelector('someel.myelem');
var keyups = Rx.Observable
    .fromEvent(myelem, 'keyup')
    .take(1);
```

OBSERVABLE METHODS

- Filtering (`filter`)
- Transforming (`map`, `reduce`)
- Collecting (`scan`)
- Buffering (`takeLast(10)`)
- Combining (`merge`, `concat`, `combineLatest`, `flatMap`)

DOM EVENTS

Close a modal:

```
var keysource = Rx.dom.keydown(window)
    .pluck('keycode')
    .filter(x => x === 27); // escape key

var clicksource = Rx.dom.click(window)
    .skip(1) // ignore first click that opens the modal
    .filter(ev =>
        document.querySelector(this).contains(ev.target));

keysource.merge(clicksource)
    .take(1)
    .subscribe(closeModal);
```

AJAX EVENTS

Fetching usernames from GitHub:

```
var github_users = Rx.DOM.ajax({
        method: 'GET',
        url: 'https://api.github.com/users'});
    .filter(x => x.status === 200)
    .map(JSON.parse)
    // Cache the deserialized response.
    .shareReplay(1)
    // Explode items in JSON response into stream items.
    .flatMap(x => Rx.Observable.from(x));

var usernames_list = github_users
    .pluck('login')
    .subscribe(x => console.log('GitHub user:', x));
```

MERGE AJAX EVENTS WITH DOM EVENTS

Combine users with click events:

```
var clicks = Rx.DOM.click(document.querySelector('#thelink'));

// Combine each click with a user.
clicks.zip(usernames_list, (click, user) => user)
    // When the list is exhausted the event handler is removed.
    .subscribe(x => console.log('GitHub user:', x));
```

MERGE TWO AJAX REQUESTS

How do you want to combine the results?

- Output each response as soon as it comes in?
- Output each response in the same order as the request?
- Wait for both to complete and combine them?

MERGE TWO AJAX REQUESTS

```
var combinedResponse = Rx
    .flatMap(function() {
        var request1 = Rx.dom.get('/url1');
        var request2 = Rx.dom.get('/url2');

        return request1.zip(request2, function(response1, respons
            // Manually combine responses here...
        });
    });
```

SIDE-EFFECTS

Useful for debugging:

```
myobsrv
     .filter(x => x.someAttr)
     .do(x => console.log('Passed the filter: ', x))
     .map(doSomethingWithSomeAttr)
     .do(x => console.log('Current value of x: ', x));
```

RESOURCES

- The big list of Rx methods
- The Rx Decision Tree
- Netflix's introduction to map/filter/reduce/etc