# ntro to RxJS

Osama AlghanmiFront End Developer @ *EffectiveUI*

itechdom@itechdom

## 1.Start with an example

- Let's see how we can display a list of tweets, filter it by hashtags.

```
```
var tweets = getTweets( );tweets.filter(t => t.username === me.username).filter(t
```
```

```
functional means: Non-mutating, Stateless, Composable
```

#2. Harsh Reality
- In the previous example, getTweets is asynchronous
- Easy! The solution is:

#3. Cute!!

#4.What about Promises?
- Promises serve almost the same purpose as RxJS Observables
- Problems with Promises:
  - Chaining Promises
  - Canceling Promises

#RxJS to the rescue! #1. Observables
- Building blocks of FRP in RxJS

- First class objects to represent Asynchronous Data
- Operators with fabulous Functional Grammar
- Interoperable with other sync/async data

```
 An Observable is an event stream which can emit zero or more events, and may or 
If it finishes, then it does so by either emitting an error or a special “complet
```

#1. Arrays Are iterable

~~~
```js
Array  .of(1, 2, 3, 4, 5)  .map(x => x * x)  .filter(x => x%2 === 0)  .reduce((x,
```
~~~

#2.Observables are iterables

``` Observable  .of(1, 2, 3, 4, 5)  .map(x => x * x)  .filter(x => x%2 === 0)  .reduce((x, acc) => x + acc) ```

```
 Remember: we return a new Observable every time we do any transformation
```

#3.Observables are lazy

~~~
```js
let users_ = getUsersObservable();
let notJohnConners_ =   users_    .map(user => user.fullName)    .map(name => nam
notJohnConners_.subscribe( (user) => console.log('New user arrived: ', user), (er
```
~~~

#4.You can make anything into an observable #1.Arrays

~~~
```js
var a = [1, 2, 3, 4, 5];var a$ = Observable.from(a);var b$ = Observable.of(1, 2,
```
~~~

## #2.Promises

~~~
```js
var a = iPromiseOfSomething();var a_ = Observable.fromPromise(a);
```
~~~

## #3.Events

~~~
```js
var result = document               .getElementById('result');var source = Rx.Obs
var observer = Rx.Observer.create(    function (x) {        console.log('Next: ' 
```
~~~

```
 Rx will truncate multiple arguments from events, you can use the selector functi
```

## #4.Callbacks

~~~
```js
writeFile = require('fs').writeFile;
wf_ = Observable      .fromNodeCallback(writeFile);
```
~~~

## #5.Custom Observables

- Similar to resolve and reject in Promises
- We can create custom observables
- We have three functions: OnNext, OnComplete, OnError

~~~
```js
var source = Rx.Observable.create(function (observer) {  observer.onNext(42);
  observer.onCompleted();
  return function () {
    console.log('disposed');  }});var subscription = source.subscribe(  function
subscription.dispose();
```
~~~

#5. You can combine multiple Observables

~~~
```js
var api1 = "https://en.wikipedia.org/w/api.php?action=query&prop=extracts&format=
var promise1 = $.get(api1);var promise2 = $.get(api2);
var source1 = Rx.Observable.fromPromise(promise1);
var source2= Rx.Observable.fromPromise(promise2);
var combined = Rx.Observable.concat(source1,source2);
```
~~~

#Reactive Programming

- RxJS is Based on the observer and iterator patterns
- Reactive can be explained as:
  - Module Foo and Module Bar
  - Module Bar is said to be reactive when it listens to modules Foo's changes
  - Foo doesn't know that Bar exists!

## 1. Imperative Programming

## 2. Reactive Programming

## Excercise

~~~
```

test('querying over events', function () {  var results = 0;
  var e = new EventEmitter();
  Observable.fromEvent(e, 'click')    .filter(function (click) { return click.x =
  e.emit('click', {x: 100, y: 50});  e.emit('click', {x: 75,  y: 75});  e.emit('c
  equal(results, 150);
// fill in the __ inside map

});
~~~

~~~
```
~~~

## How I use it

## 1.Node Example

## Questions?

## Resources

- The introduction to Reactive Programming you’ve been missing: a thorough introduction to RxJS by Cycle.js author Andre Staltz.
- Introduction to Rx: an online book focused on Rx.NET, but most concepts map directly to RxJS.
- ReactiveX.io: official cross-language documentation site for ReactiveX.
- Learn Rx: an interactive tutorial with arrays and Observables, by Jafar Husain.
- RxJS lessons at Egghead.io
- RxJS GitBook
- RxMarbles: interactive diagrams of RxJS operators, built with Cycle.js.
- Async JavaScript at Netflix: video of Jafar Husain introducing RxJS.

## Topics we didn't cover

## 1.Schedulers

Explanation:

- http://xgrommx.github.io/rx-book/content/getting_started_with_rxjs/scheduling_and_concurrency.html

API Docs:
- http://xgrommx.github.io/rx-book/content/schedulers/index.html

## 2.Subjects

Explanation:

- http://xgrommx.github.io/rx-book/content/getting_started_with_rxjs/subjects.html

API Docs:
- http://xgrommx.github.io/rx-book/content/subjects/index.html

## 3.Notification

Explanation:

- http://xgrommx.github.io/rx-book/content/notification/index.html

## 4.Hot vs Cold Observables

Docs:

- https://github.com/Reactive-Extensions/RxJS/blob/master/doc/gettingstarted/creating.md#cold-vs-hot-observables

## 5. Marble Diagrams

Visualize all Observable operators:

- http://rxmarbles.com/