



RXJS: De 0 a Programación Reactiva

Jorge Mario Lenis

CTO Kerberos Ingeniería S.A.S



@mlenislibreros



mariolenis

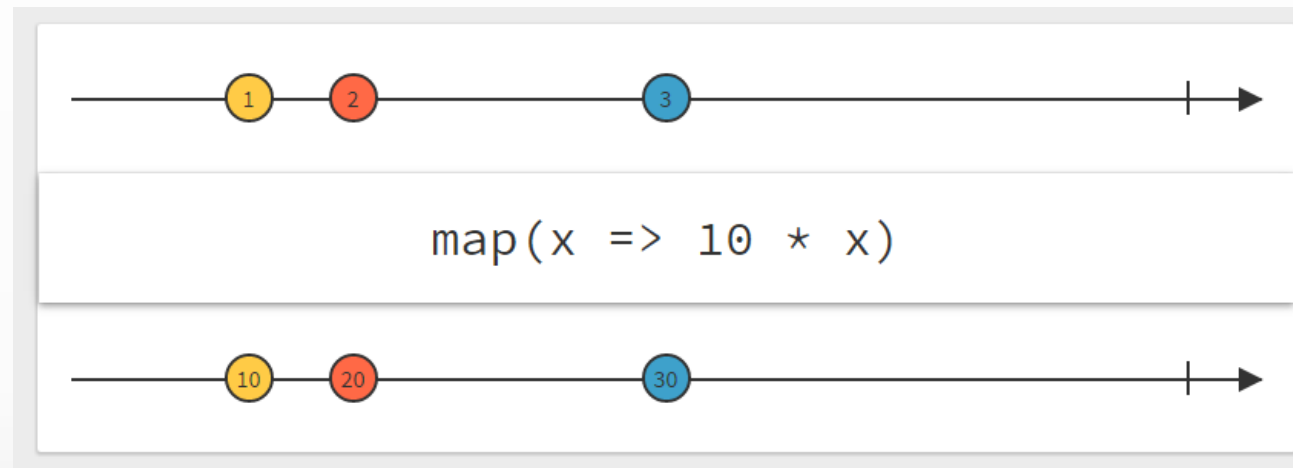
Agenda

1. Qué es Programación Reactiva / Reactive Programming
2. Que es RXJS
3. Observables VS Promises
4. Operadores y Observables
5. Code
6. Beneficios



Everything is a stream

Qué es Programación Reactiva



Marbles Diagram | <http://rxmarbles.com>

1. Paradigma de programación enfocado en la propagación de datos en el tiempo como flujos
2. Combina patrones de diseño como observable e iterable
3. Incorpora programación funcional

Programación Funcional

Paradigma que describe una relación entre la entrada y la salida como funciones de algebra e incorpora dos pilares fundamentales: Las funciones puras y la inmutabilidad de los datos

Ej: $f(x) = 2x$ [Función pura]

```
const x = {value: 2};  
function pureFn(x) {  
    const b = x.value * 2;  
    return b;  
}
```

```
const x = {value: 2};  
function impureFn(x) {  
    x.value = x.value * 2;  
    return x.value;  
}
```

RXJS – `npm install rxjs --save`

1. <https://github.com/ReactiveX/rxjs>
2. Versión 5 re-escrita por GDE's/Netflix Head Prog.
3. Librería que contiene **operadores, observables y schedulers**.



Observables VS Promises

1. Ambos proveen abstracción (resolución) para el manejo de llamados asincrónicos (***asynchronous***)
2. Observables pueden ser cancelables.
3. Observables pueden manejar múltiples datos.
4. Observables son funciones “perezosas”.

Operadores Y Observables

Existen alrededor de 454 **operadores**

1. Map()
2. Flatmap()
3. Switchmap()
4. Filter()
5. Last()
6. ...

455. **Build your own!**

<https://github.com/ReactiveX/rxjs/blob/master/doc/operator-creation.md>

Observables

1. of()
 2. timer()
 3. interval()
 4. fromEvent()
 5. fromPromise()
 6. merge()
- (n+1) ... **Build your own**

Subjects

1. **Subject**
2. **ReplySubject**
3. **BehaviorSubject**
4. **AsyncSubject**

Let's code!

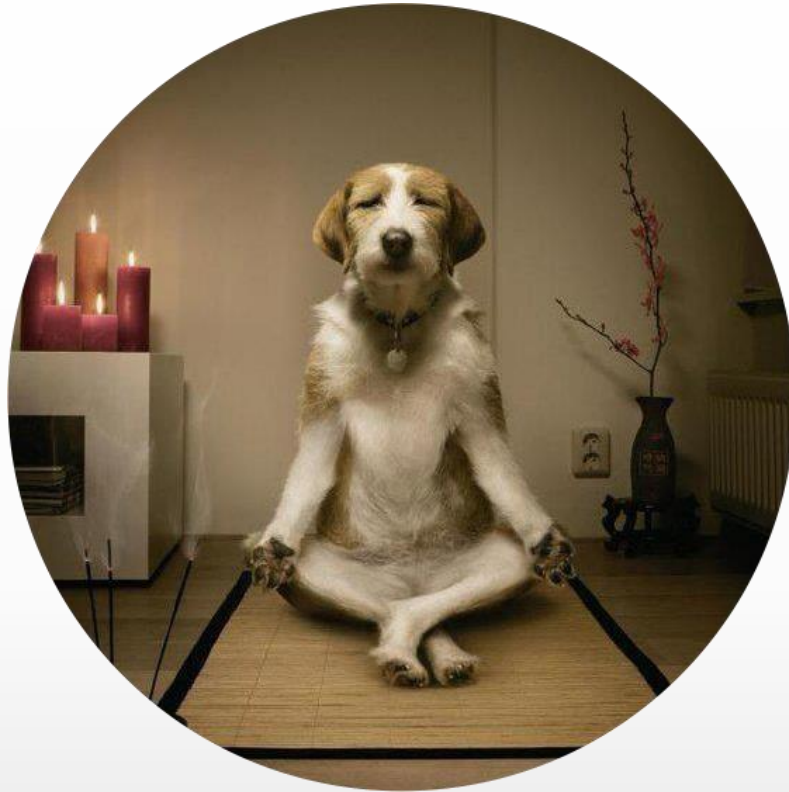
Observable desde 0 (from scratch) + rxjs/Observable + rxjs/*Subject

Beneficios

1. Tratar la información como flujo de datos con observables hace que tu código sea realmente re-utilizable.
2. Facilita las pruebas ya que los operadores y observables implementan funciones puras.
3. Al ser funciones cancelables mejoramos el UX drásticamente
4. Los Observables pueden manejar múltiples datos en un flujo continuo, no solo de lado del cliente sino del lado del servidor (websockets, socket.io)

Fuentes

1. <http://reactivex.io/documentation>
2. <https://egghead.io/courses/introduction-to-reactive-programming>
3. <http://rxmarbles.com>
4. <https://www.youtube.com/watch?v=uQ1zhJHclvs> You will learn RXJS (André Staltz @andrestaltz)



Gracias!