RxJS: De 0 a Programación Reactiva

Jorge Mario Lenis

CTO Kerberus Ingeniería S.A.S

@mlenislibreros

mariolenis

Agenda

1. Por qué Programación Reactiva
2. Qué es Programación Reactiva / Reactive Programming
3. Que es Rxjs
4. Rxjs |Observables, Subjects, Subscripciones y Operadores
5. Observables VS Promesas
6. Live Code
7. Beneficios de Rxjs

¿Por qué Programación Reactiva?

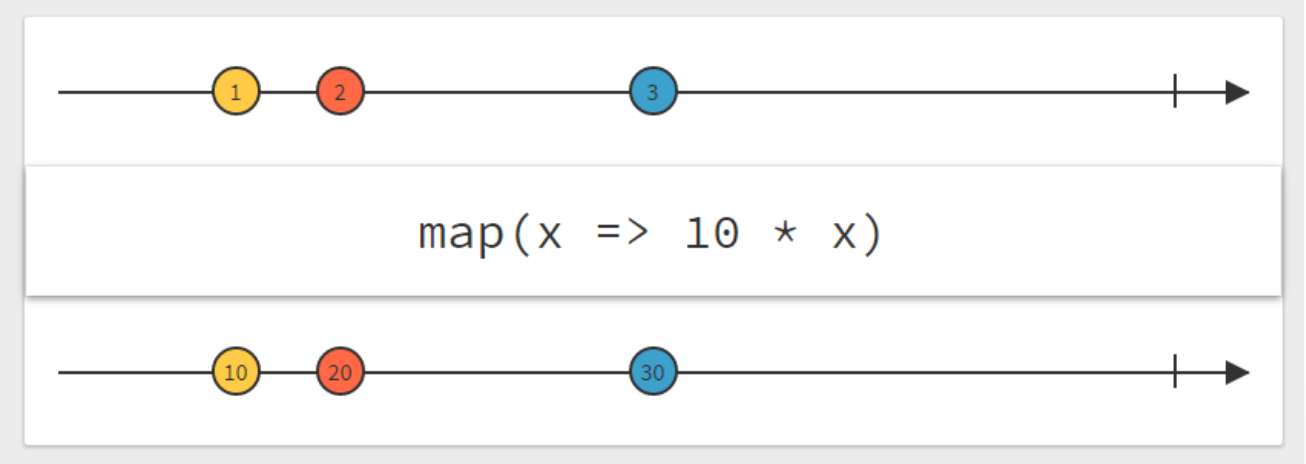
Porque vivimos en un mundo asincrónico!

¿Por qué Programación Reactiva?

Vivimos en un mundo asincrónico!


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.
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 inpureFn(x) {
        x.value = x.value * 2;
        return x.value;
}
```

Rxjs | Extensión Reactiva para JavaScript

1. Librería en versión 5 re-escrita por GDE's/Netflix Head Prog.
2. Contiene **observables, operadores, schedulers y subscripciones**.
3. Permite crear objetos observables, suscribirse a ellos y escuchar datos (similar a las promesas).


```
Observable.fromEvent(this.btn, 'click')
```

```
.subscribe(event => this.message = 'Power up!');
```

4. Permite encadenar operadores e integrar otros observables a un flujo de datos.

Rxjs | Observables, Subjects y Subscriptions

Observables

1. of()
2. from()
3. timer()
4. interval()
5. fromEvent()
6. fromPromise()
7. merge()

(n+1) … **Build your own**

Subjects

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

Subscripciones

1. Subscribe()
2. ForEarch()

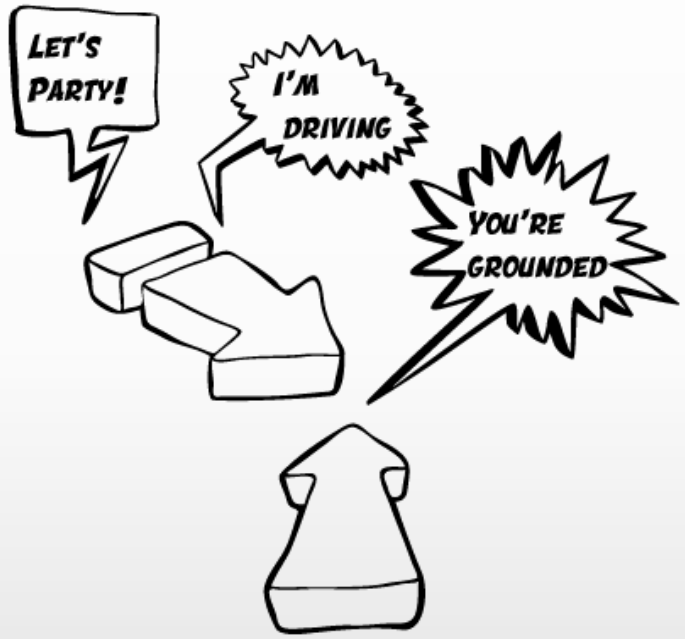
Rxjs | Operadores

Existen alrededor de 454 **operadores**

1. Map()
2. Flatmap()
3. SwitchMap()
4. Filter()
5. Last()
6. TakeUntil()
7. …

455. **Build your own!**

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




```
robin$
    .switchMap(
       batman$.takeUntil(alfred$))
```

Observables VS Promesas

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 objetos “perezosos” que devuelven una función.

Let’s code!

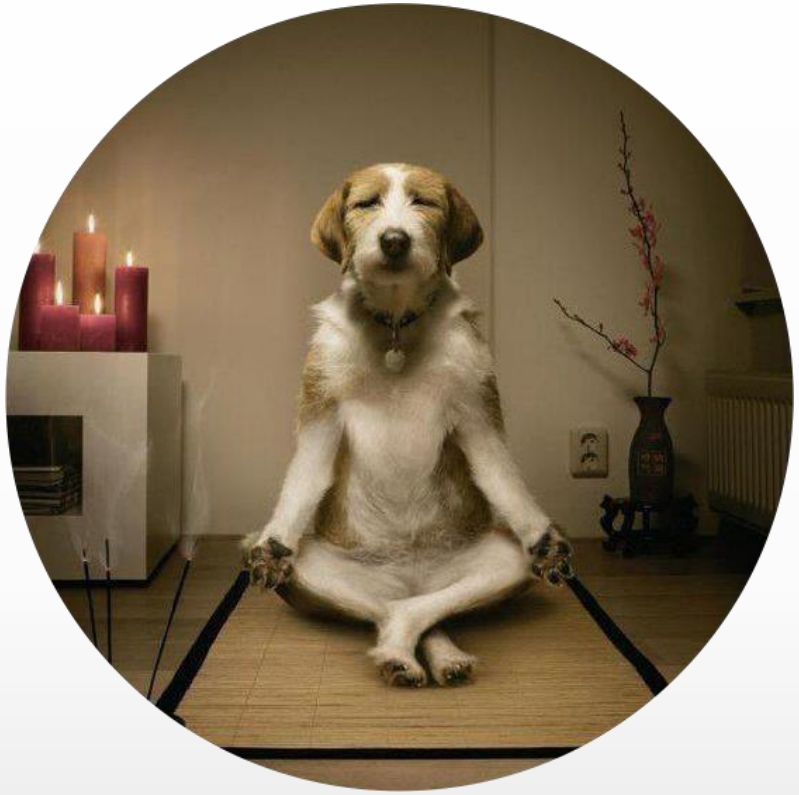
Observable desde 0 (from scratch) + rxjs/Observable + rxjs/*Subject

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. https://github.com/ReactiveX/rxjs
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!

@mlenislibreros
mariolenis