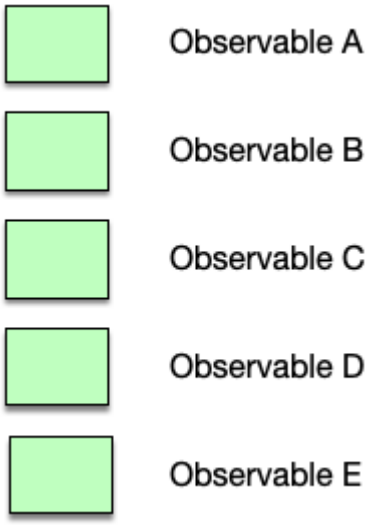
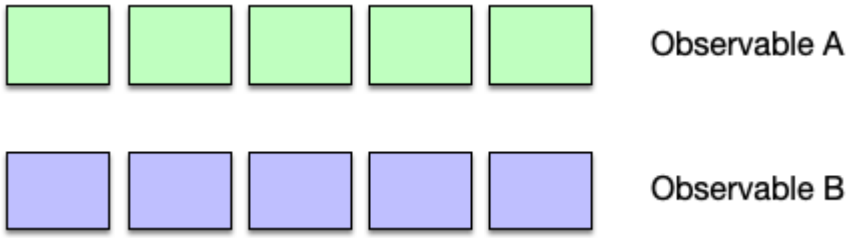
El concepto de Rx flatMap siempre es complejo de entender para todo el mundo . Sobre todo porque de entrada se trata de manejar un conjunto de observables y estos siempre son complejos de entender. ¿Para qué sirve la operación de flatMap? . En muchas ocasiones tenemos peticiones o solicitudes que nos generan observables que contienen un array de elementos.

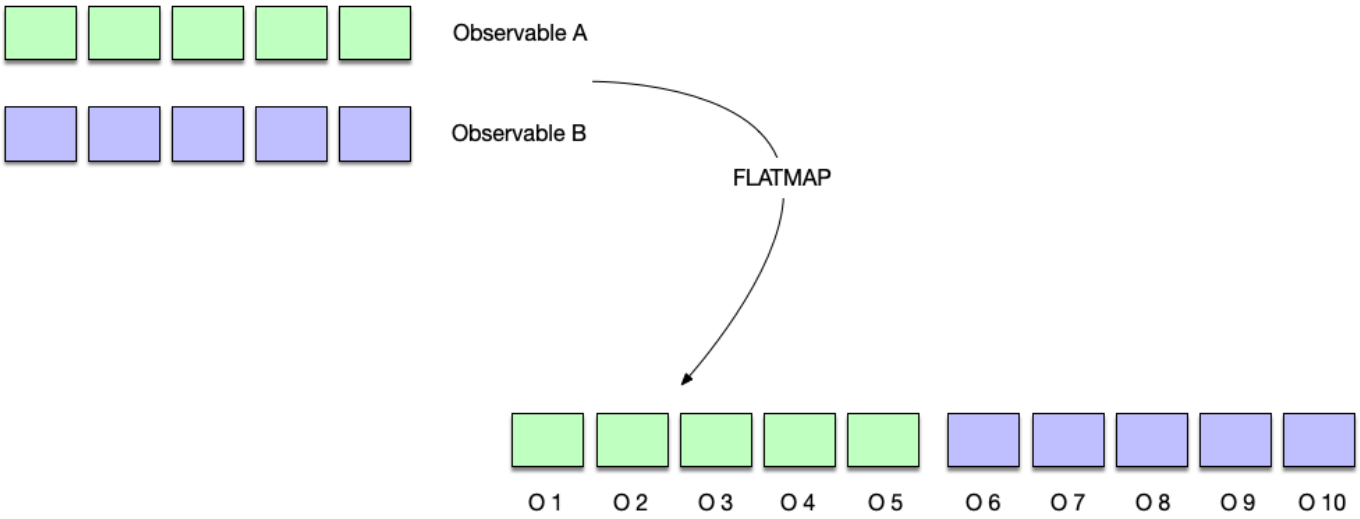
Muchas veces pensamos que una petición Ajax que nos llegue al cliente se convierte en un Array de Observables . Pero esto no es así sino que se trata de un Observable que contiene un Array de datos. Por lo tanto no es lo mismo un Observable que contiene un Array que un Array de Observables.



Esto muchas veces no es problemático ya que se trabaja de formas similares y se procesan . Ahora bien hay situaciones en las que nos encontramos con lo siguiente.



Este es un caso diferente ya que tenemos dos observables y cada uno de ellos contiene un array de elementos . Es muy posible que deseemos construir una nueva lista en la que cada elemento sea un observable en si y procesarlo como si fuera una lista plana (flat)

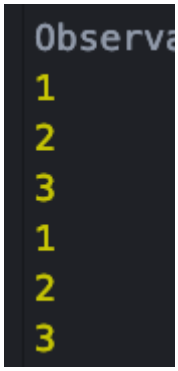


Vamos a ver un sencillo ejemplo en código usando Node.js:

```
import Rx from "rxjs";



let o1= Rx.from([1,2,3]);
let o2=  Rx.from([1,2,3]);
```

```
let combinado=new Rx.Observable.create(observer=> {
    observer.next(o1);
    observer.next(o2);

})

combinado.subscribe((datos)=> {

    datos.subscribe((otro)=> {
        console.log(otro);
    })

})
```

En este caso si nos fijamos en el código tenemos un observable que combina dos observables . Cada observable contiene un único dato que es un Array . Si queremos imprimir cada uno de los elementos del array deberemos primero subscribirnos al observable para después volvernos a subscribir a cada uno de ellos y obtener los observables que tienen dentro en el array. Este en principio no parece muy optimo pero funciona.

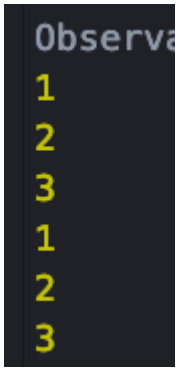


Rx FlatMap (MergeMap)

Para ello usamos Rx FlatMap (ojo en las últimas versiones de RxJs se ha renombrado como MergeMap) y extraemos de cada array los observables que tiene dentro para recorrerlos uno a uno . Esto hace todo mucho más legible.

```
import Rx from "rxjs";
import "rxjs/add/operator/mergeMap";


let o1= Rx.from([1,2,3]);
let o2=  Rx.from([1,2,3]);


let combinado=new Rx.Observable.create(observer=> {
    observer.next(o1);
    observer.next(o2);

})


combinado.mergeMap((datos)=>datos).subscribe((datos)=> {


        console.log(datos);


});
```

De esta manera el resultado será idéntico en la consola:



Pero la forma de programarlo se habrá simplificado sobremanera. Acabamos de usar el operador MergeMap (antiguamente FlatMap) . Para simplificar el manejo de observables anidados.

Otros artículos relacionados

1. ConnectableObservable RxJava (Hot Observables)
2. RxJS Ajax , fusionando peticiones asíncronas
3. Flux vs Mono ,Spring y la programación reactiva
4. Rxjs