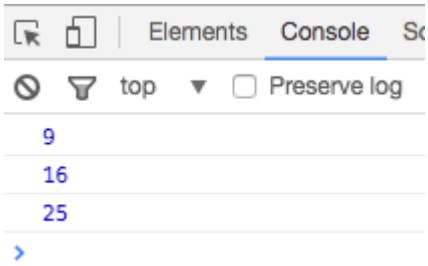
Observers vs Arrays una comparativa interesante. ¿Son lo mismo los Arrays que los Observers? . Ultimamente todas las plataformas nos permiten trabajar de forma reactiva de algún framework Rx. Cuando uno empieza a trabajar con ellos da la sensación de que son muy parecidos a los Arrays clásicos. Sin embargo su funcionamiento es muy diferente .Vamos a ver un ejemplo práctico.

Observers vs Arrays

Para poder entender mejor el concepto vamos a partir de una lista de 5 elementos en JavaScript. La vamos a filtrar y transformar en otra lista de elementos al cuadrado.

```
var lista = [1, 2, 3, 4, 5];

lista.filter(function(elemento) {

return elemento &gt; 2;

}).map(function(elemento) {

return elemento * elemento;
}).forEach(function(elemento) {

console.log(elemento);
})
```

El resultado por pantalla es :



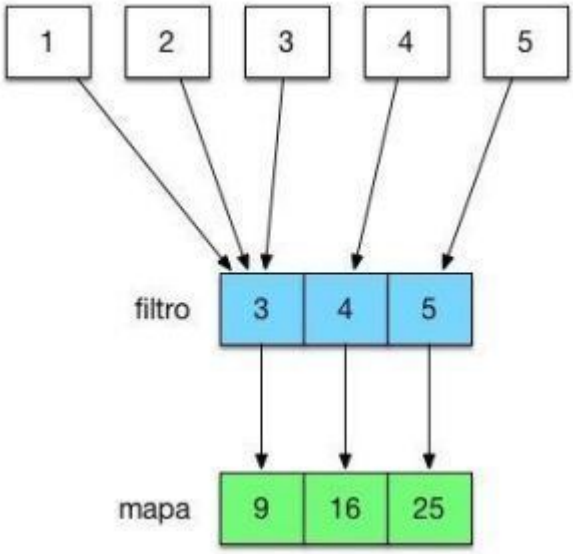
Ahora bien si modificamos un poco el código y añadimos un console.log con cada función

```
var lista=[1,2,3,4,5];

lista.filter (function(elemento) {
console.log("filtra:" +elemento);
return elemento>2;

}).map(function(elemento) {
console.log("mapea:" +elemento);
return elemento* elemento;
}).forEach(function(elemento) {

console.log(elemento);
})
```

El resultado obtenido será mucho más claro.

```
top ▼ Preserve log
filtra:1
filtra:2
filtra:3
filtra:4
filtra:5
mapea:3
mapea:4
mapea:5
9
16
25
```

JavaScript realiza primero la tarea de filtrar todos los elementos quedándose con el 3 , 4 y 5. Acto seguido aplica map y lo eleva al cuadrado.



Finalmente el bucle forEach recorre el array y lo imprime por pantalla.

Observers vs Arrays (Rx.js)

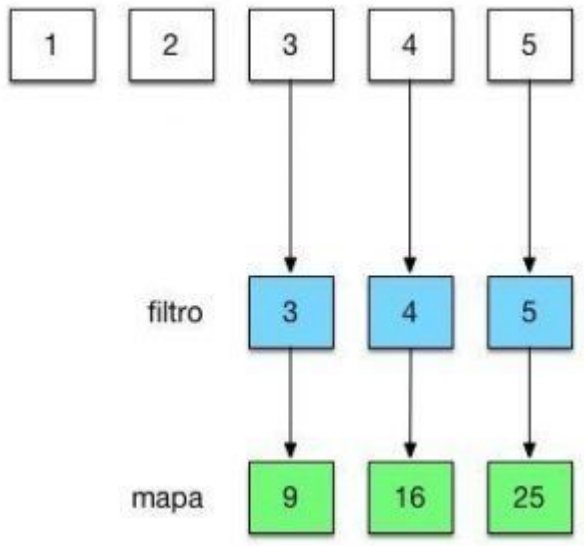
Podemos hacer la misma tarea usando Rx.js para ello las modificaciones a aplicar son puntuales

```
var observable = Rx.Observable.fromArray([1, 2, 3, 4, 5]);

observable.filter(function(elemento) {
console.log("filtra:" + elemento);
return elemento > 2;

}).map(function(elemento) {
console.log("mapea:" + elemento);
return elemento * elemento;
}).subscribe(function(elemento) {

console.log(elemento);
})
```

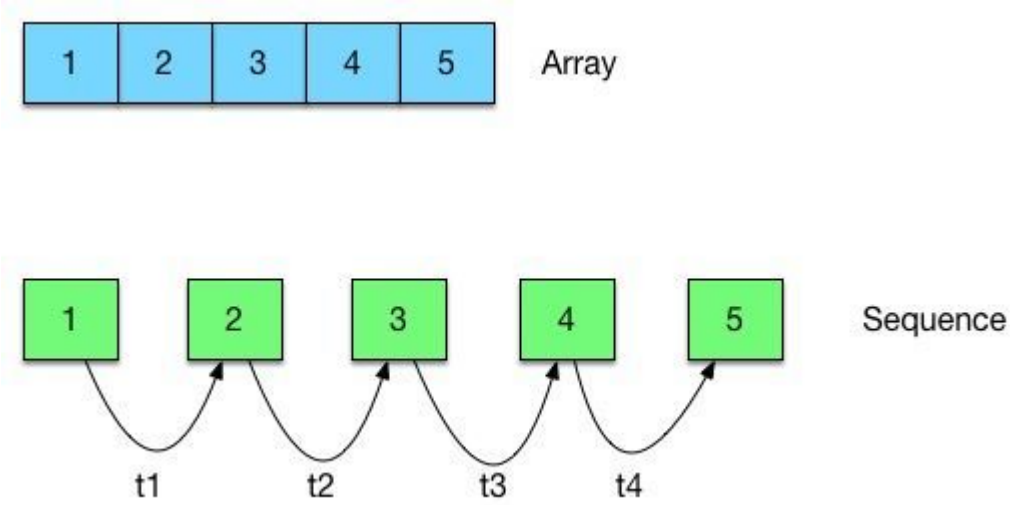
Acabamos de convertir nuestra lista en una lista de elementos observables. El código es muy muy similar pero si lo ejecutamos el resultado nos sorprenderá.

```
Elements  Console  Sources
top  Preserve log
filtra:1
filtra:2
filtra:3
mapea:3
9
filtra:4
mapea:4
16
filtra:5
mapea:5
25
>
```

No se opera de la misma forma sino que en este caso cada función se ejecuta de forma independiente en cada elemento. Estamos operando sobre un flujo de trabajo similar al que pueden tener los Streams de Java.



¿A qué se debe esto?. Recordemos las extensiones Rx se usan para gestionar peticiones asíncronas. Una secuencia de observables se parece mucho a una lista , pero no es una lista.



Una secuencia de observables es una lista cuyos elementos nos van llegando cada x tiempo . Por eso mientras que con un array todos los elementos se procesan de forma simultánea (los tenemos todos) con una secuencia de observables se realiza el flujo uno a uno . Nos van llegando poco a poco. Este concepto es importante para entender de forma más natural las extensiones Rx.

Otros artículos relacionados : Rx.js , Streams vs promises , Javascript Map