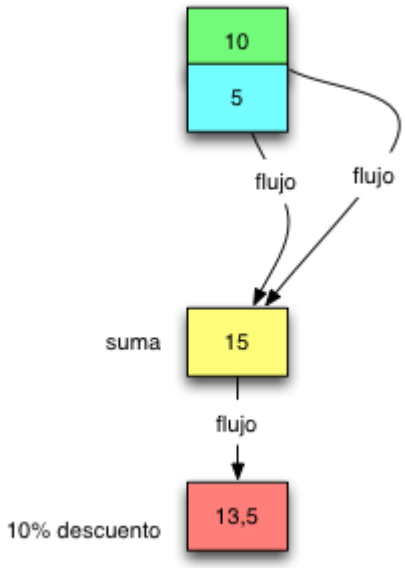
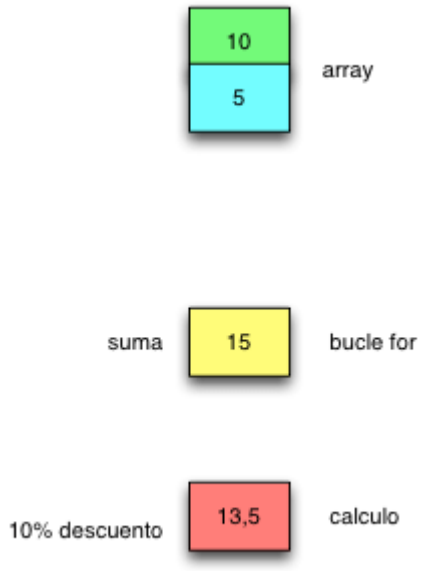
RxJS es una de las librerías JavaScript que cada día vamos a usar más y uno de los representantes de la programación reactiva .¿Qué es la programación reactiva?. La programación reactiva es un nuevo paradigma orientado a programar basado en flujos de datos que son los encargados de transmitir los cambios a nuestra aplicación . De entrada no suena fácil de entender. Vamos a intentar realizar una aproximación más sencilla basándonos en el concepto de Excel. Imaginemonos que disponemos del siguiente excel.



Son dos celdas iniciales que sumamos en otra celda para finalmente aplicar un descuento del 10% en otra. Hemos construido un flujo de datos y cualquier administrativo lo entiende. Es algo muy natural, cada vez que cambiemos un dato de las celdas actuales el resto se actualizará. Si enfocamos esto desde el mundo del desarrollo clásico nos encontraremos con un enfoque muy diferente:



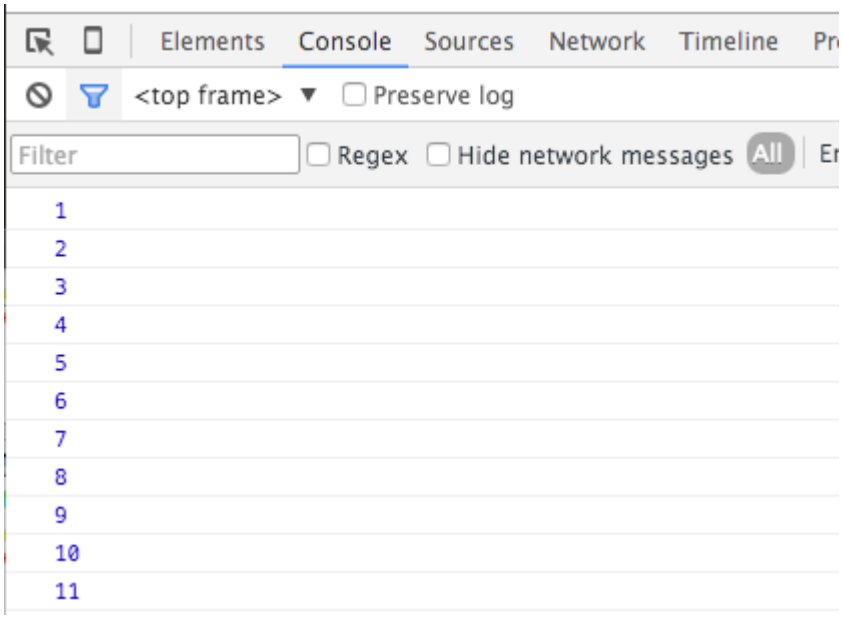
Dispondremos de un array , calcularemos la suma recorriendole con un bucle for y finalmente realizaremos el cálculo del descuento. No se parece en nada a lo anterior y es mucho más abstracto. Una vez explicadas estas diferencias vamos a abordar un ejemplo con programación reactiva y RxJS

Introducción a RxJS

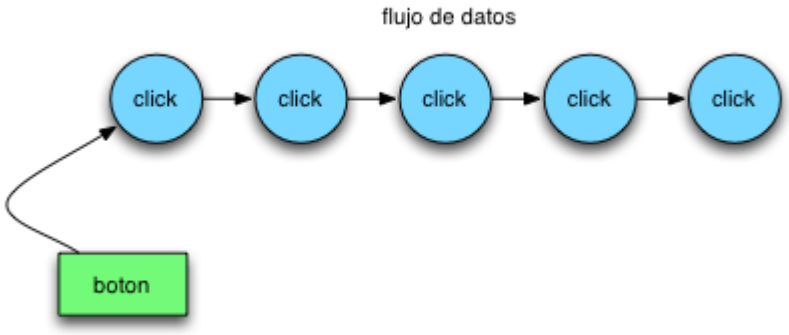
Para poder entender de forma sencilla RxJS debemos de empezar a trabajar con un ejemplo que no lo use. En este caso he elegido un simple botón que cada vez que le pulsamos nos imprime por consola las veces que hemos hecho click.

```
<!DOCTYPE>
<html>

<head>
 <title></title>
```

```
 <script src="rx.all.js" type="text/javascript">
 </script>

</head>

<body>
 <input type="button" id="boton" value="pulsar" />
</body>
 <script type="text/javascript">

 var total=0;

 var boton = document.getElementById("boton");
 boton.addEventListener("click", function(){

 console.log(++total);
 });

 </script>
</html>
```



Vamos a convertir ahora nuestro código a un enfoque más reactivo. En este caso vamos a usar RxJS para convertir los clicks en un flujo de datos.



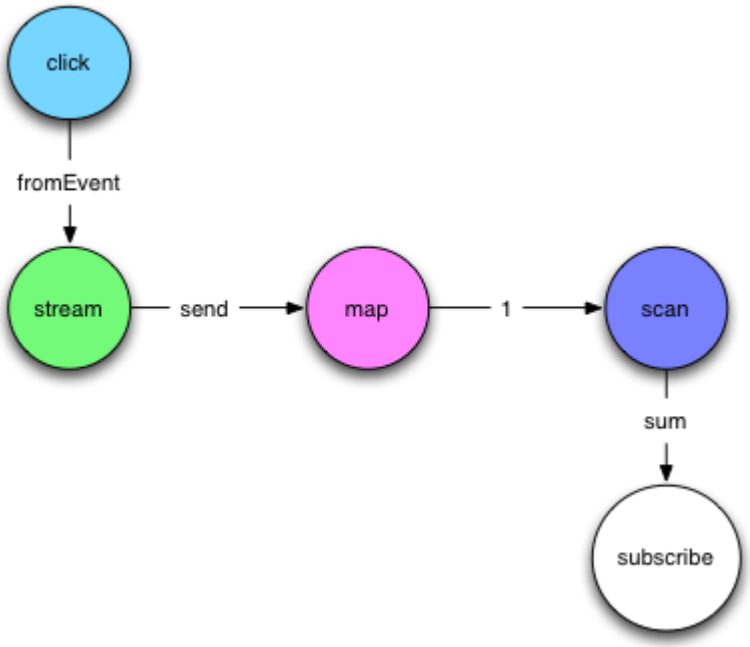
Vamos a ver la solución en código:

```
<!DOCTYPE>
<html>
```

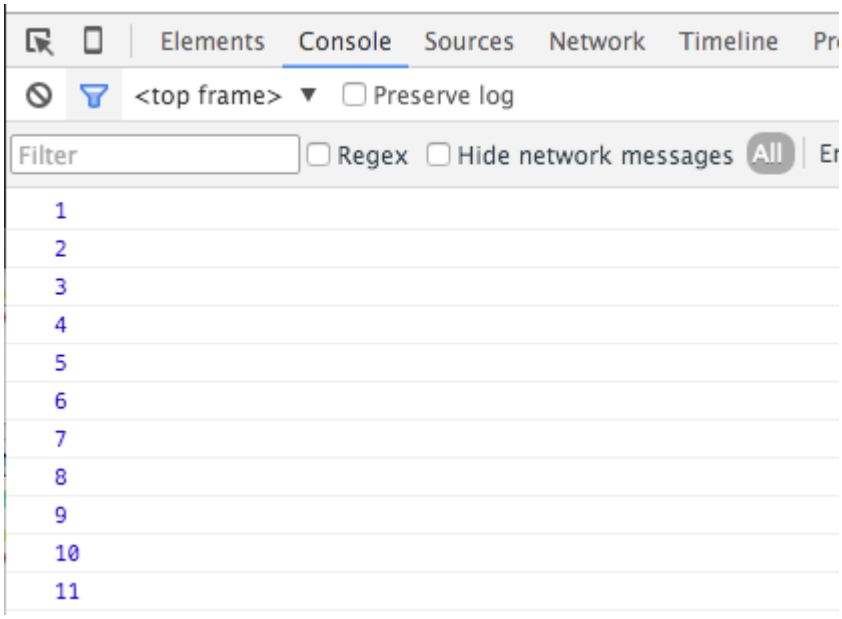
```
<head>
<title></title>
<script src="rx.all.js" type="text/javascript">
</script>
</head>

<body>
<input type="button" id="boton" value="pulsar" />
</body>
<script type="text/javascript">
var boton = document.getElementById("boton");
var botonStream = Rx.Observable.fromEvent(boton, 'click');
var suma=botonStream.map(function(x) {

return 1;

}).scan(function(x,y) {

return x+y;

}, 0).subscribe(function(resultado) {

console.log(resultado);
});
</script>

</html>
```

Como vemos usar RxJS , no es excesivamente complejo. El primer paso es generar un Stream (flujo de datos) a partir de un conjunto de eventos (en este caso el click). Una vez hecho esto usamos dos operaciones clásicas de programación funcional map y scan (scan es

similar a reduce) para convertir los eventos en números y finalmente sumarlos. El último paso es subscribirse al stream o flujo de datos e imprimir su resultado por consola , para ello usamos el método subscribe.



El código es mucho más reutilizable que el anterior y el resultado idéntico.



Poco a poco estos nuevos paradigmas se van haciendo un hueco.

Otros artículos relacionados: VertX , JavaScript Reduce ,JavaScript Pure Functions