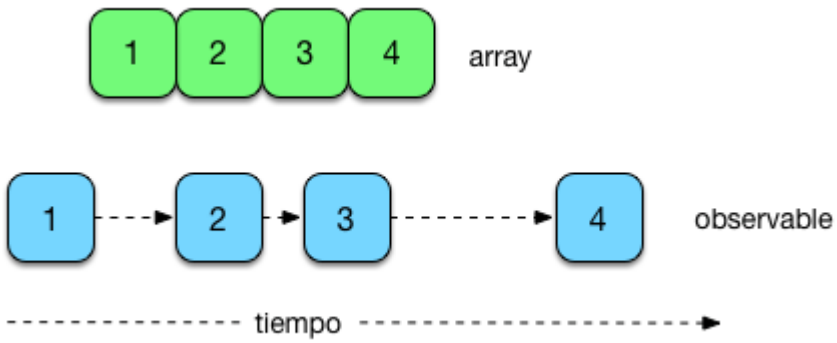
El uso de Rx Observables se va extendiendo poco a poco y la programación reactiva empieza a estar de moda ya que frameworks como Angular 2 se apoyan fuertemente en ella. Recordemos que a la programación reactiva se la suele comparar con el funcionamiento de una hoja Excel. En la hoja de calculo rellenamos dos valores en dos celdas y automáticamente otra celda calcula la suma.

Rx Observables y RxJS

Vamos a usar RxJS y los Rx Observables para realizar una operación de este estilo en una página HTML. Lo primero que tenemos que entender es el concepto de Rx Observable. Se trata de un concepto relativamente y hace referencia a una colección de elementos que se genera en un intervalo de tiempo.



Vamos a ver un ejemplo usando RxJS y generando dos observables en dos caja de texto para cada vez que pulsemos una tecla sobre ella.

Vamos a ver su código:

```
<html>

<head>

 <script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/4.1.0/rx.all.js">
 </script>

</head>

<body>
 <input type="text" name="dato1" id="dato1" />
 <input type="text" name="dato2" id="dato2" />
 <input type="text" name="resultado" id="resultado" />

</body>
<script type="text/javascript">
 var flujoCaja1 =
Rx.Observable.fromEvent(document.getElementById("dato1"), 'keyup');
 var flujoCaja2 =
Rx.Observable.fromEvent(document.getElementById("dato2"), 'keyup');

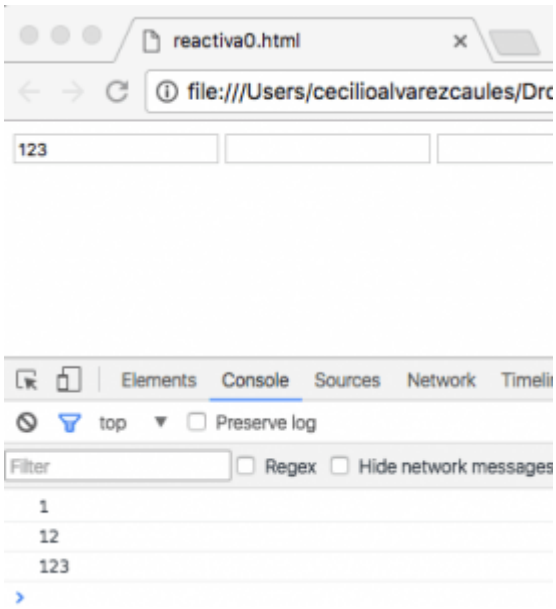
 flujoCaja1.subscribe(function(elemento) {

 console.log(elemento.srcElement.value);

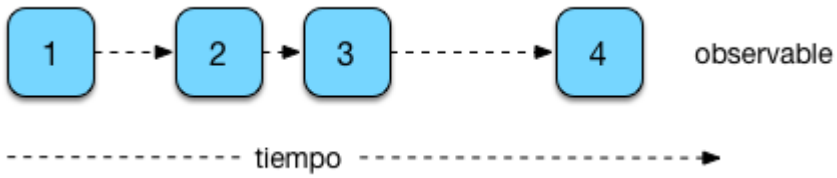
 });
```

```
</script>
</html>
```

Acabamos de generar dos observables uno sobre la caja “dato1” y otro sobre la caja “dato2” . Una vez hecho esto nos hemos subscrito al primer observable y hemos obtenido el value del elemento que lo contiene . En nuestro caso se trata del valor de la caja de texto.



Cada vez que pulsamos una tecla se produce un elemento con el contenido de la caja . Esto es un Rx Observable , una colección que se va generando con el tiempo.



Vamos a modificar el programa para disponer de dos Rx Observables y combinarlos :

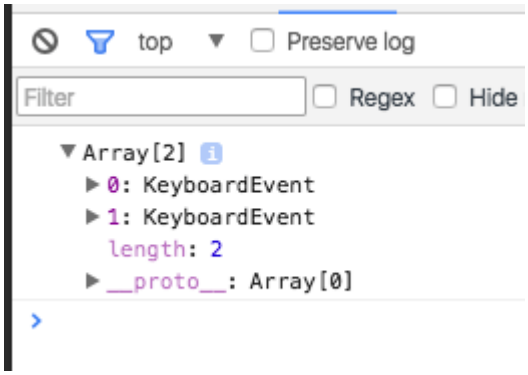
```
        var flujoCaja1 =
Rx.Observable.fromEvent(document.getElementById("dato1"), 'keyup');
        var flujoCaja2 =
Rx.Observable.fromEvent(document.getElementById("dato2"), 'keyup');

        var flujoCombinado = Rx.Observable
                .combineLatest(flujoCaja1,
flujoCaja2).subscribe(function (elemento) {

                                        console.log(elemento);

                });
```

Esta primera operación se encarga de unir el flujo de datos de ambas cajas.



Ahora cada vez que pulsamos en una de las dos cajas , nos genera un array de 2 elementos con sus valores como eventos. Es momento de aplicar algunas transformaciones más complejas para acceder a los valores de ambas cajas y sumarlos.

```
        var flujoCaja1 =
Rx.Observable.fromEvent(document.getElementById("dato1"), 'keyup');
```

```
        var flujoCaja2 =
Rx.Observable.fromEvent(document.getElementById("dato2"), 'keyup');

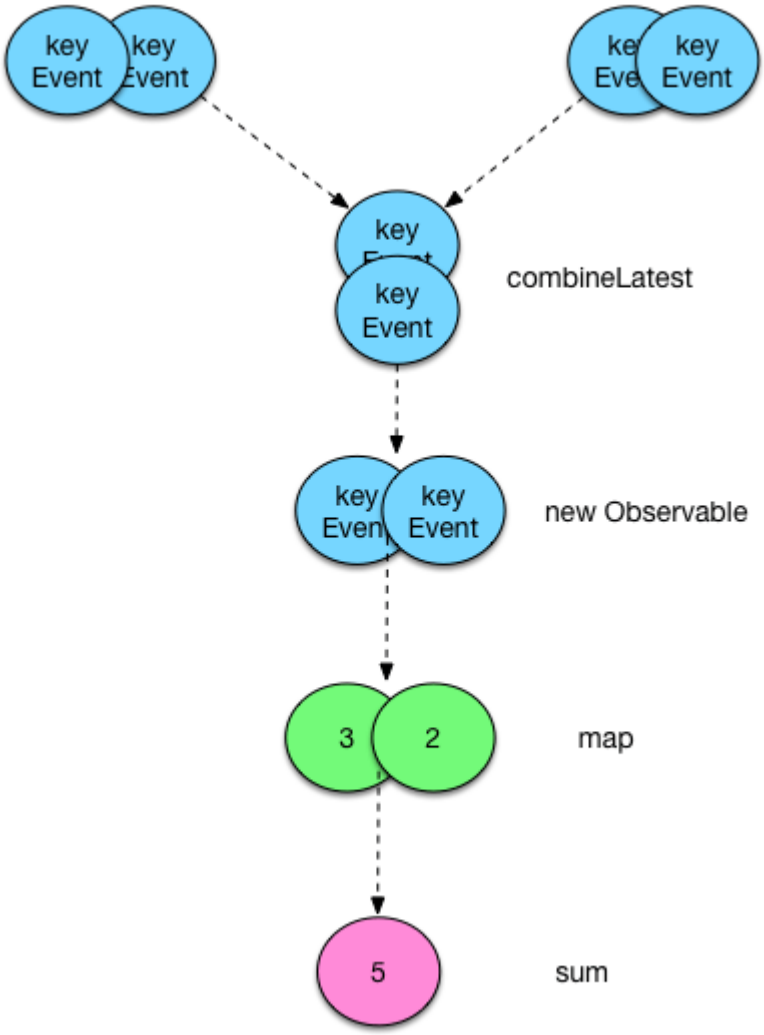
        var flujoCombinado = Rx.Observable
                .combineLatest(flujoCaja1, flujoCaja2)
                .map(function(array) {

                        return Rx.Observable.from(array);

                }).subscribe(function (observable) {

                                observable
                                .map(function(elemento) {
                                        return
elemento.srcElement.value == "" ? 0 :
parseInt(elemento.srcElement.value);
                                })
                                .sum()
                                .subscribe(function(total) {
document.getElementById("resultado").value=total;
                                });
                });
```

Hemos usado una operación de map para generar un nuevo Observable con dos elementos cada vez. Hecho esto hemos vuelto a usar la operación map dentro de nuestra primera subscripción para transformar la colección de elementos Observables de eventos de teclado a valores de las cajas. Por último hemos sumado esos valores.



El resultado lo mostramos en la tercera caja:



Ahora mismo nuestras cajas e comportan como un Excel, hemos usado programación reactiva con RxJS.

Otros artículos relacionados:RxJS y la programación reactiva. , Reactive MicroServices y Arquitectura,