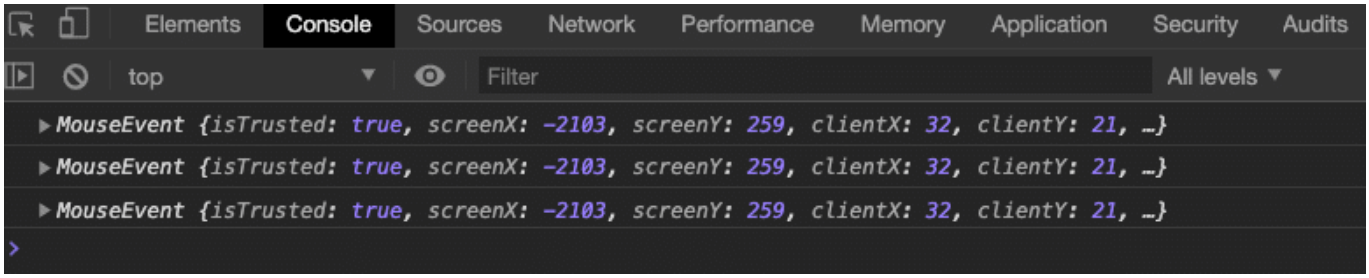
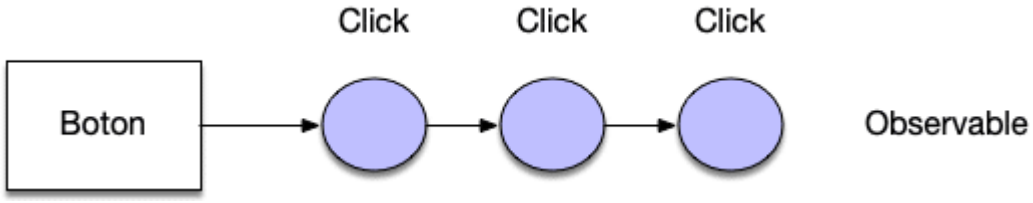
El concepto de Rx Back Pressure es uno de los conceptos importantes cuando trabajamos con programación reactiva. ¿Para que sirve este concepto? . Vamos a partir de un ejemplo sencillo de RxJs que nos ayude a entenderlo . Imaginemonos que tenemos un botón que cada vez que le pulsamos nos genera un evento y con RxJs generamos un observable que nos muestra en la consola que hemos pulsado el botón. El código podría ser algo similar a esto:

```
<html>
<head>
<script type="text/javascript"
src="https://unpkg.com/rxjs/bundles/rxjs.umd.min.js">

</script>
<script type="module">


var boton= document.getElementById('miboton');

rxjs.fromEvent(boton,'click').subscribe(function(datos) {

  console.log(datos);
})




</script>
</head>
<body>
<input type="button" id="miboton" value="aceptar">
</html>
```

Si pulsamos el botón automáticamente aparecerá en la consola el evento:



Todo funciona correctamente y las extensiones Rx controlan el evento a través de un observable.

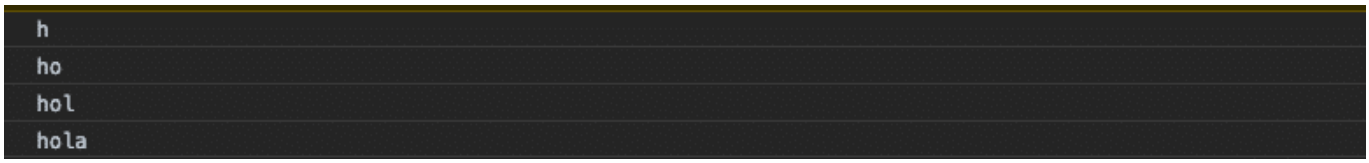


Esto nos puede parecer una situación óptima pero imaginémonos una situación similar pero algo diferente en la cual tenemos una caja de texto y cada vez que pulsamos una tecla y la rellenamos queremos que nos muestre por consola el contenido de la caja.

```
<html>
<head>
<script type="text/javascript"
src="https://unpkg.com/rxjs/bundles/rxjs.umd.min.js">

</script>
<script type="module">


var caja= document.getElementById('caja');

rxjs.fromEvent(caja,'keyup').subscribe(function(datos) {
```

```
  console.log(caja.value);
})



</script>
</head>
<body>
<input type="text" id="caja" >
</html>
```

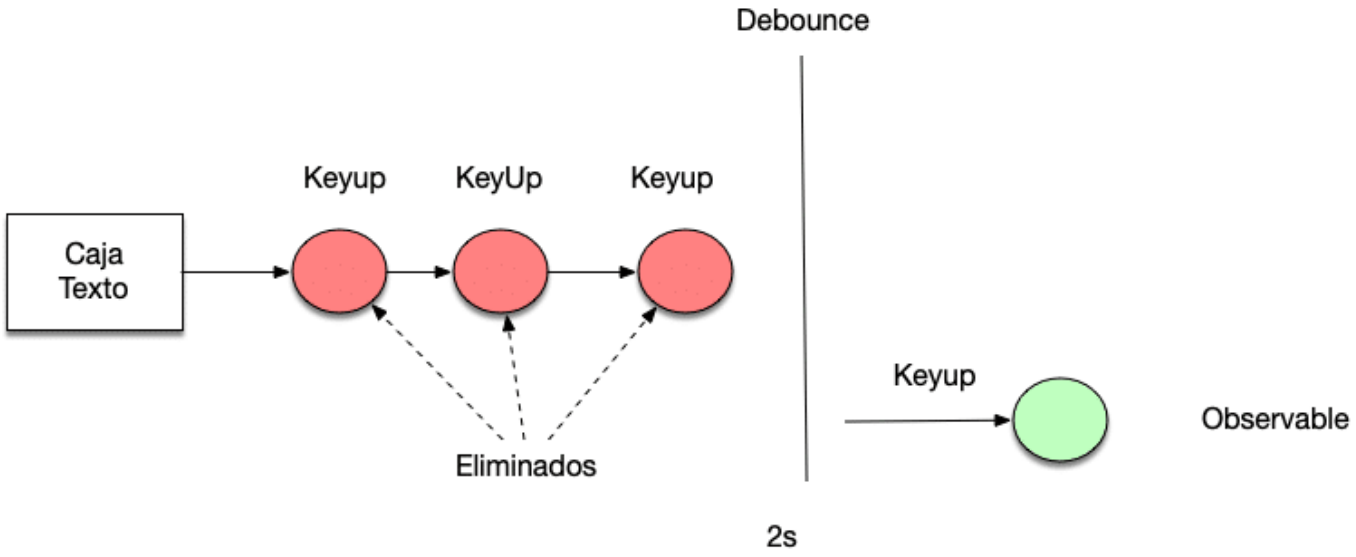
En este caso hemos utilizado el evento de keyup , por lo tanto cada vez que pulsamos una tecla nos mostrará el contenido de la caja por la consola



¿Es lo que queremos? . Nos puede parecer que sí , pero pensando un poco nos daremos cuenta de que si cada vez que escribimos una letra por ejemplo realizamos una petición Ajax al servidor para traernos un filtro para una consulta determinada. Es evidente que quizás lo podamos optimizar.

Rx Back Pressure

El concepto de Rx Back Pressure hace referencia a cuando tenemos un flujo de datos y nos damos cuenta de que no todo ese flujo de datos nos es interesante sino que podemos prescindir de una parte de él . Por ejemplo en nuestro caso sería interesante que cada 2 segundos nos muestre un mensaje en la consola con lo que en este se encuentre escrito . De tal forma que esa información pudiera usarse para realizar una llamada Ajax al servidor y obtener los datos.



De esta forma aliviaríamos al servidor de un conjunto de consultas intranscendente y nos centraríamos en los datos que el usuario ha introducido. Vamos a usar el operador debounce de RxJs que lo que hace es descartar automáticamente valores que se hayan generado antes de que se cumpla un intervalo de tiempo mínimo. De esta forma cuando vayamos escribiendo en la caja no se generarán eventos continuamente sino que el operador los descartará.

```
<html>
<head>
<script type="text/javascript"
src="https://unpkg.com/rxjs/bundles/rxjs.umd.min.js">

</script>
<script type="module">

var timer= rxjs.timer;
var debounce = rxjs.operators.debounce;

var caja= document.getElementById('caja');
```

```
rxjs.fromEvent(caja,'keyup')
.pipe(debounce(() => timer(2000))).subscribe(function(datos) {

  console.log(caja.value);
})



</script>
</head>
<body>
<input type="text" id="caja" >
</html>
```

En este caso hemos solicitado al operador debounce que descarte todos los items que ocurran antes de que pasen 2000 milisegundos (2 segundos) . De esta forma podremos escribir la palabra hola tranquilamente en la caja y al pasar 2 segundos el observable se emitirá por la consola

Otros artículos relacionados

1. Rx flatMap (mergeMap) y simplificación de observables
2. RxJS Ajax , fusionando peticiones asíncronas
3. Introducción a RxJava y sus observables
4. Web Rxjs