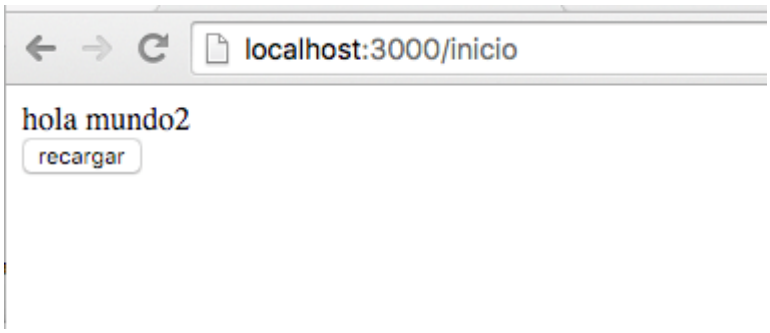
Streams vs Promises es una de las discusiones que se esta poniendo más de moda en el mundo de JavaScript. El uso de promises nunca ha sido sencillo y a mucha gente le cuesta introducir estos conceptos dentro de la programación asíncrona que realiza. Vamos a explicar la diferencia que existe entre ambos conceptos para ello vamos a partir de un código que utiliza jQuery y promesas para trabajar.

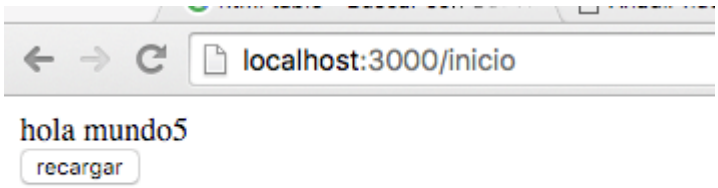
```
<html>
<head>

<script src="//code.jquery.com/jquery-1.12.0.min.js"></script>
<script type="text/javascript">

$(document).ready(function() {

$("#recargar").click(function() {
var promesa= $.get("/datos");

promesa.done(function(datos) {

$("#zona").html(datos);
})

});


});

</script>
</head>
```

```
<body>

<div id="zona">
</div>

<input type="button" id="recargar" value="recargar"/>
</body>
</html>
```

En este caso se trata de un botón que hace una petición Ajax al servidor utilizando una promesa .

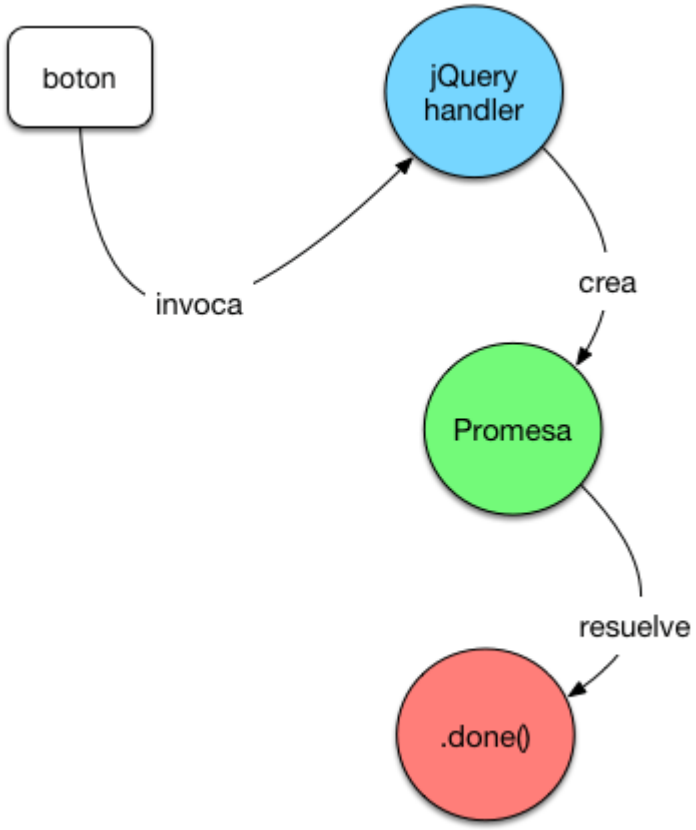


Al realizar la petición el servidor nos devuelve un mensaje con “holamundo” más la iteración por la que va. Así pues cada vez que pulsemos al botón el número se incrementará. Este incremento dependerá del tiempo que esperemos.



Hemos usado jQuery y su sistema de promesas , para ello hemos necesitado enlazar el

evento “click” y cada vez que pulsamos el botón se invoca la promesa.



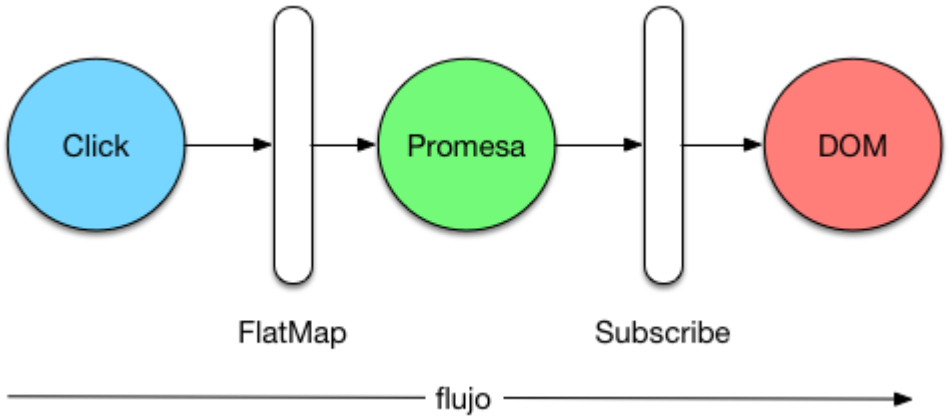
Streams vs Promises con RxJS

Todo funciona correctamente . Sin embargo algo que sucede con las promesas , es que el código no queda natural. Podemos construir el mismo ejemplo utilizando RxJS y programación reactiva . La solución es más natural.

```
<head>

<script src="http://code.jquery.com/jquery-1.12.0.min.js"></script>
```

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/4.1.0/rx.all.js"></sc
ript>
<script type="text/javascript">

$(document).ready(function() {

var stream=Rx.Observable.fromEvent($("#recargar"), 'click');

stream.flatMap(function() {
return $.get("datos");

}).subscribe(function(resultado) {

$("#zona").html(resultado);
});

});

</script>
</head>
<body>

<div id="zona">
</div>

<input type="button" id="recargar" value="recargar"/>
</body>
</html>

 
```

En este caso hemos construido un Stream o flujo con RxJS. El primer paso es crear el flujo a través del método fromEvent de RxJS. Una vez creado realizamos dos operaciones la primera es de transformación que transforma cada click en una promesa que solicita los datos y de la cual obtenemos su resultado. La operación flatMap convierte el retorno de una promesa en un dato puro. Obtenidos los datos usamos subscribe y trabajamos con jQuery contra el DOM para que se actualice.



Este flujo de trabajo es mucho más reutilizable y natural que el código anterior que desarrollamos con jQuery. RxJS se irá haciendo hueco en nuestro universo de JavaScript.

Otros artículos relacionados: Introducción RxJS , jQuery Promises , RxJava