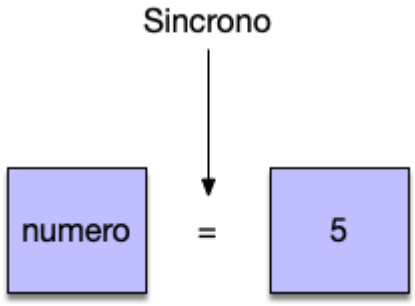
La diferencia entre Promise vs Observable es algo que muchas veces cuesta entender en el mundo de la programación asíncrona. Vamos a intentar explicarlo de una forma sencilla partiendo de un ejemplo muy elemental .Cuando nosotros trabajamos con JavaScript podemos definir una variable y asignarle un valor.

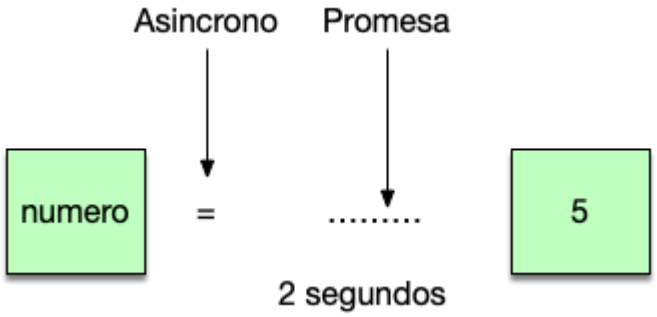
```
var numero=5;
console.log(numero);
```

Este código asigna un valor de 5 a la variable número . Parece que no tenemos nada que explicar ya que se trata de un ejemplo trivial. Sin embargo hay que remarcar una cosa se trata de un código que se ejecuta de forma síncrona es decir la asignación es instantánea.



JavaScript Promise

¿Qué diferencia existe entre este código y una promesa de JavaScript? . Muy sencillo una promesa de JavaScript hace lo mismo pero la asignación del valor es asíncrona es decir se asignará en un futuro.



Vamos a ver su código:

```
var promesa= new Promise (function(resolve,reject) {
  setTimeout(function() {
    resolve(5);
  },2000);

})
promesa.then(function(resultado) {
  console.log(resultado);
})
```

Este código nos imprimirá al cabo de 2 segundos un 5 en la consola:

```
5
>
```

JavaScript Promise vs Observable

Una vez tenemos claro para qué sirve una promesa y cómo asigna una variable de forma asíncrona la siguiente pregunta que es evidente es para que sirve un Observable. En este caso vamos a realizar la misma operación pero en este caso con una lista de elementos.

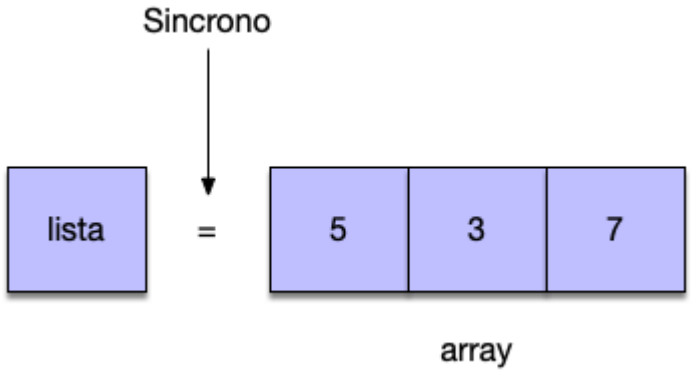
```
var lista=[1,2,3];

console.log(lista);
```

En este caso simplemente imprimimos la lista por la consola.

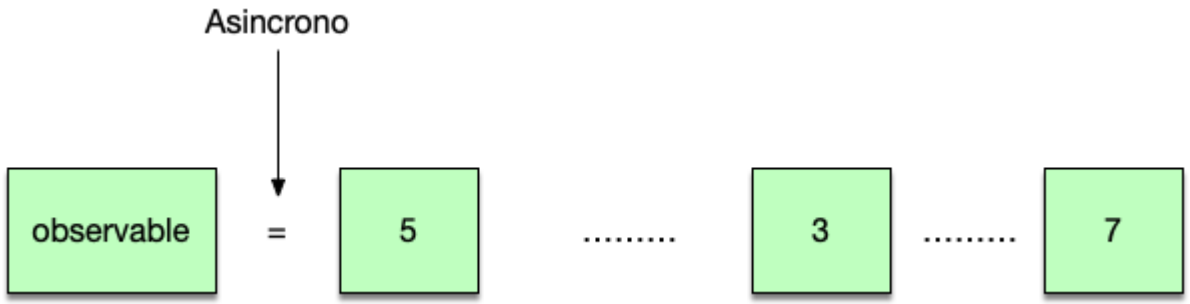
```
▼Array(3) i
   0: 1
   1: 2
   2: 3
   length: 3
  ▶__proto__: Array(0)
```

Volvemos a estar en una situación de código síncrono .

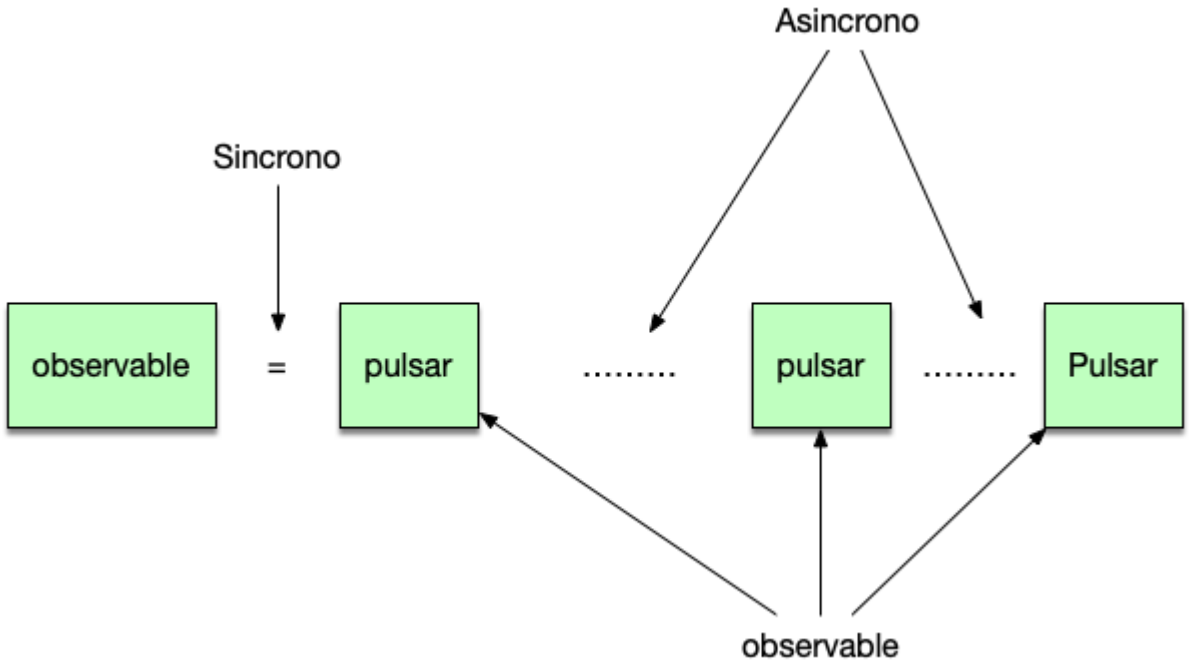


Promise vs Observable

¿Entonces qué es un Observable? . Un observable no es ni más ni menos que una colección de elementos asíncronos.



Esto en un principio nos puede parecer extraño ya que una colección siempre la hemos entendido como un grupo de elementos y punto. No parece muy razonable tener una colección de elementos asíncronos . Sin embargo a poco que pensemos se trata de algo muy común por ejemplo cada vez que pulsamos un botón en un interface de usuario se produce un evento de click. Se trata de una colección de elementos asíncronos.



Si volvemos a pulsar en el mismo botón saldrá un segundo evento , si volvemos a pulsar saldrá un tercero etc. Eso es lo que es un observable veamos su código:

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

```
<input type="button" value="pulsar" id="miboton"/>
</body>
</html>
<pre>
```

El código de JavaScript:

```
window.onload=function() {

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

  var pulsaciones = rxjs.fromEvent(boton, 'click');


    pulsaciones.subscribe(e => {
      console.log('has pulsado');

    });

}
```

Cada vez que pulsamos el botón RxJs nos mostrará un mensaje en la consola, acabamos de construir una lista de elementos asíncronos. Acabamos de ver la diferencia entre promise vs observable.

Otros artículos relacionados:

1. Angular async pipe y observables
2. Hot vs Cold Observable con Rx.js
3. Introducción a RxJava y sus observables
4. RxJS