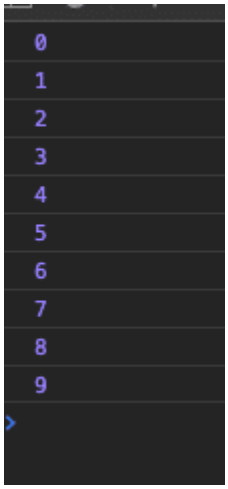
¿Javascript sincrono o asincrono?. Esta es una de las preguntas más habituales cuando uno empieza a tener cierta experiencia en programación con JavaScript. ¿Cómo funciona el lenguaje en sí? . ¿Es síncrono o es asíncrono?. Normalmente casi todo el mundo me suele responder que el lenguaje es asíncrono por sus peticiones Ajax etc . Sin embargo las cosas no son tan sencillas como nos parece en un primer momento. Vamos a echarle un vistazo a cómo JavaScript funciona realmente. Para ello es suficiente con construir un bucle for.

```
<htnl>
<script type="text/javascript">

    for (let i=0;i<10;i++) {

        console.log(i);
    }

</script>

<body>

</body>
</html>
```

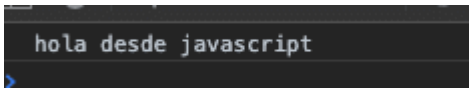
Vemos los números por la consola.



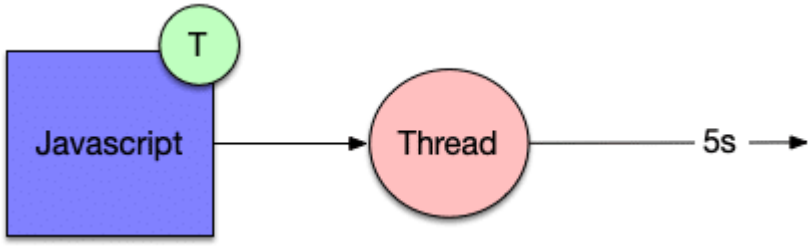

El resultado no nos sorprende para nada . Podemos ahora construir un programa que nos da la sensación de ser asíncrono en el cual pulsamos un botón y solicitamos que después de un intervalo de tiempo se muestre un mensaje por la consola.

```
<htnl>
<script type="text/javascript">

  function mensaje() {

    setTimeout(function() {

      console.log("hola desde javascript");

    },5000);
  }

</script>

<body>
<input type="button" value="asincrono" onclick="mensaje()"/>
</body>
</html>
```

Si lo ejecutamos el comportamiento será el esperado en este caso al cabo de 5 segundos se muestra el resultado por la consola.




¿Así pues hemos ejecutado un código “asíncrono” ? . Muchas personas piensan que cuando este código se ejecuta hay un Thread de JavaScript que se abre para realizar esta tarea al cabo de 5 segundos .




Algo similar a cuando realizamos este tipo de operaciones en el mundo Java.
Lamentablemente la realidad es diferente y a veces cuesta entenderla. Para ello vamos a añadir un sencillo botón que nos ejecute una función de JavaScript con un alert a nuestro código

```
<htnl>
<script type="text/javascript">

  function mensaje() {

    setTimeout(function() {

      console.log("hola desde javascript");

    },5000);
  }
```

```
  function alerta() {

    alert("hola");
  }

</script>

<body>
<input type="button" value="asincrono" onclick="mensaje()"/>
<input type="button" value="alerta" onclick="alerta()"/>
</body>
</html>
```

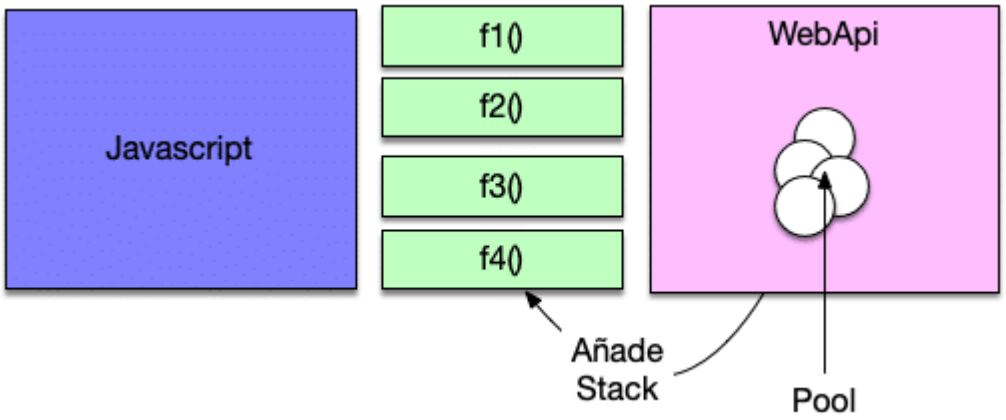
Si pulsamos el botón del setTimeout a los 5 segundos veremos el mensaje por la consola .


Ahora bien si pulsamos ese botón e inmediatamente después pulsamos el botón que nos muestra el alert la realidad es que NO veremos el mensaje de la consola pasados los 5 segundos. De hecho no lo veremos nunca hasta que aceptemos el mensaje del alert. ¿Qué esta ocurriendo? . ¿Es JavaScript sincrono o asíncrono? . La realidad es que el motor de Javascript es sincrono y solo dispone de un Thread de ejecución. Si nosotros bloqueamos el Thread con un alert a la espera de que confirmemos el motor de Javascript no será capaz de ejecutar nada más y se quedara esperando eternamente.

# ¿Javascript sincrono o asincrono?

¿Entonces cómo funciona una petición Ajax que todos sabemos que es asíncrona? . Ahora tenemos claro que el motor de JavaScript es sincrono. Las cosas no son tan complicadas como parece. El navegador tiene un pool de Thread a nivel de WebAPI fuera del runtime de JavaScript que es el que se encarga de realizar estas tareas asíncronas cómo puede ser una petición Ajax . Cuando esa petición Ajax termine el pool registra en la pila de llamadas de

JavaScript una nueva función con el resultado de la operación. Esta función será ejecutada por el motor de JavaScript en su event loop de forma sincrona. Es decir en cuando no tenga otra cosa que hacer.




Por lo tanto tengamos siempre en cuenta que el motor de JavaScript siempre es sincrono y todas las peticiones que consideremos asincronas se hacen a traves del pool de Thread de WebAPI que es algo externo. Este al terminar nos devolverá un resultado.

- Async y Await en Javascript
- JQuery Promise y Ajax
- Promise vs Observable