El concepto de JavaScript Web Workers nos permite gestionar de forma concurrente varias tareas en JavaScript. Muchas personas piensan que JavaScript es Asíncrono ya que existen operaciones como setTimeout y setInterval . Sin embargo las cosas son bastante diferentes a lo que uno cree. JavaScript es un lenguaje síncrono por naturaleza . Es decir solo dispone de un único hilo de ejecución que realiza todas las operaciones sobre el navegador . Esto a la mayor parte de la gente le resulta difícil de encajar pero es suficiente con ver un ejemplo sencillo para entender que evidentemente es síncrono .

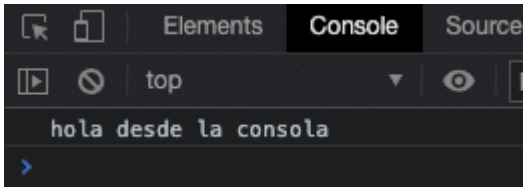
```
<html>
    <head>
        <script type="text/javascript" src="001.js">
        </script>

    </head>
    <body>
        <input type="button"  id="boton" value="pulsa"
onclick="alert('hola')"/>
    </body>
</html>
```

Esta Página nos presentará una página html con un botón de pulsar  pero a la vez carga un script 001.js que  se ejecuta nada más cargar la página . Este script  contiene el siguiente código que se ejecuta después de 5 segundos:

```
setTimeout(function() {
    console.log("hola desde la consola");
},5000)
```

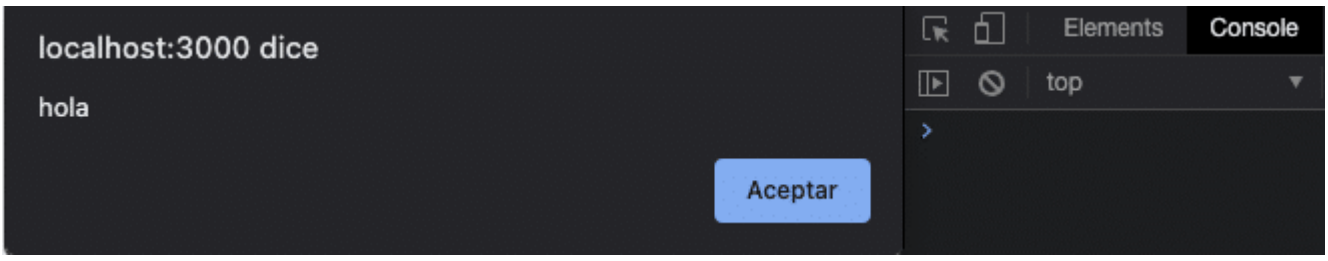
Por lo tanto lo que tiene que pasar es que la página se carga y a los 5 segundos presenta un mensaje por la consola:




Si probamos el ejemplo no hay problema todo se ejecuta correctamente . El problema aparecen cuando nosotros cargamos la aplicación y nada más cargarla pulsamos sobre el botón de “pulsa” que genera un mensaje de alerta modal en la ventana.

pulsa

Esto bloquea por completo como el navegador y el motor de JavaScript. por lo tantto l mensaje de “hola desde la consola” no se  imprimirá hasta que aceptemos la alerta. Si JavaScript fuera asíncrono y permitiera concurrencia el mensaje simplemente aparecería.




Sin embargo como se puede observar la consola no imprime nada . Esto se debe a que el motor de JavaScript es síncrono  y no asíncrono como mucha gente piensa . ¿Cómo podemos

solventar este problema y conseguir un funcionamiento asíncrono real?

# JavaScript  Web Workers

Para conseguir que él funcionamiento sea asíncrono podemos solicitar a JavaScript que genere un Worker. Este Worker es un proceso que se ejecuta en paralelo permitiendo un verdadero procesamiento asíncrono de nuestro código:

```
<html>
    <head>
        <script type="text/javascript" src="002.js">
        </script>

    </head>
    <body>
        <input type="button"  id="boton" value="pulsa"
onclick="alert('hola')"/>
    </body>
</html>
```

Acabamos de cambiar el código del 001.js por el 002.js . Este nuevo bloque de código generará un worker.

```
const worker = new Worker('worker.js');
```

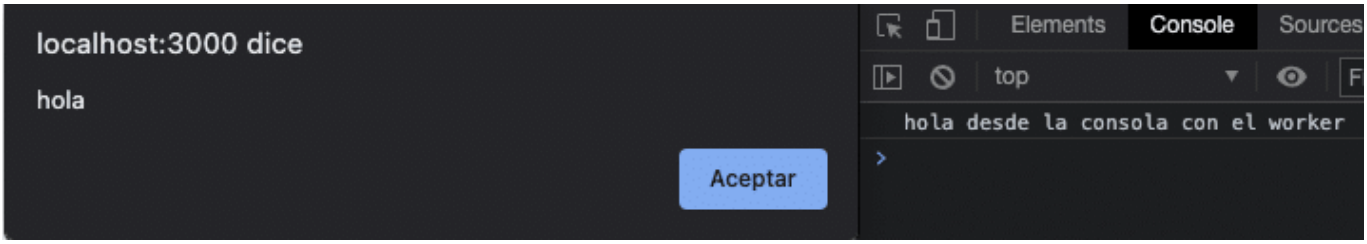
El worker puede ejecutar su propio código de forma concurrente a la ejecución del programa principal .




En nuestro caso el código es idéntico al anterior ejecutara un mensaje por la consola al cabo de 5 segundos.

```
setTimeout(function() {
    console.log("hola desde la consola con el worker");
},5000)
```

La ventaja es que ahora si existe una ejecución en paralelo y aunque activemos el mensaje de alerta el worker seguirá su ejecución aunque no la confirmemos:




JavaScript es un universo  que tiene cientos de peculiaridades y esta es una de ellas 

## Otros artículos relacionados

1. JavaScript for loop y sus opciones
2. JavaScript Hoisting y sus trucos
3. JavaScript Tagged Templates y flexibilidad