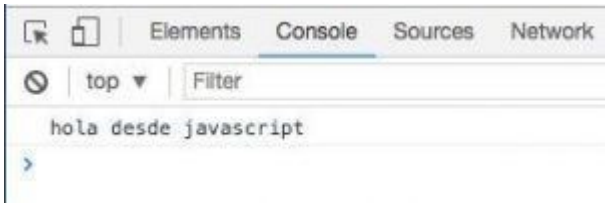
El uso de JavaScript setTimeout es muy habitual en la programación del día a día con JavaScript. Sin embargo muchas veces no se entiende bien como funciona . Vamos a explicar tanto el concepto de JavaScript setTimeout como el de setInterval que es el método complementario y ver su funcionamiento.

JavaScript setTimeout y curiosidades

La función de JavaScript setTimeout se encarga de ejecutar otra función después de un tiempo determinado . Se suelen usar para gestionar animaciones , transiciones y operaciones similares. Vamos a construir el ejemplo de “hola mundo”.

```
function mensaje() {

 console.log("hola desde javascript");

}

setTimeout(mensaje,5000);
```

Este mensaje se ejecutará pasados 5 segundos en la consola:

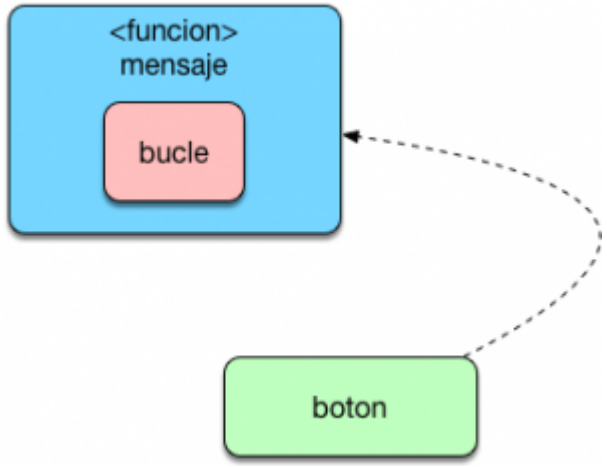


En estos momentos todo parece muy sencillo , simplemente retrasamos la ejecución 5 segundos. La realidad es un poco más compleja. Imaginemonos que añadimos el siguiente código a nuestra página:

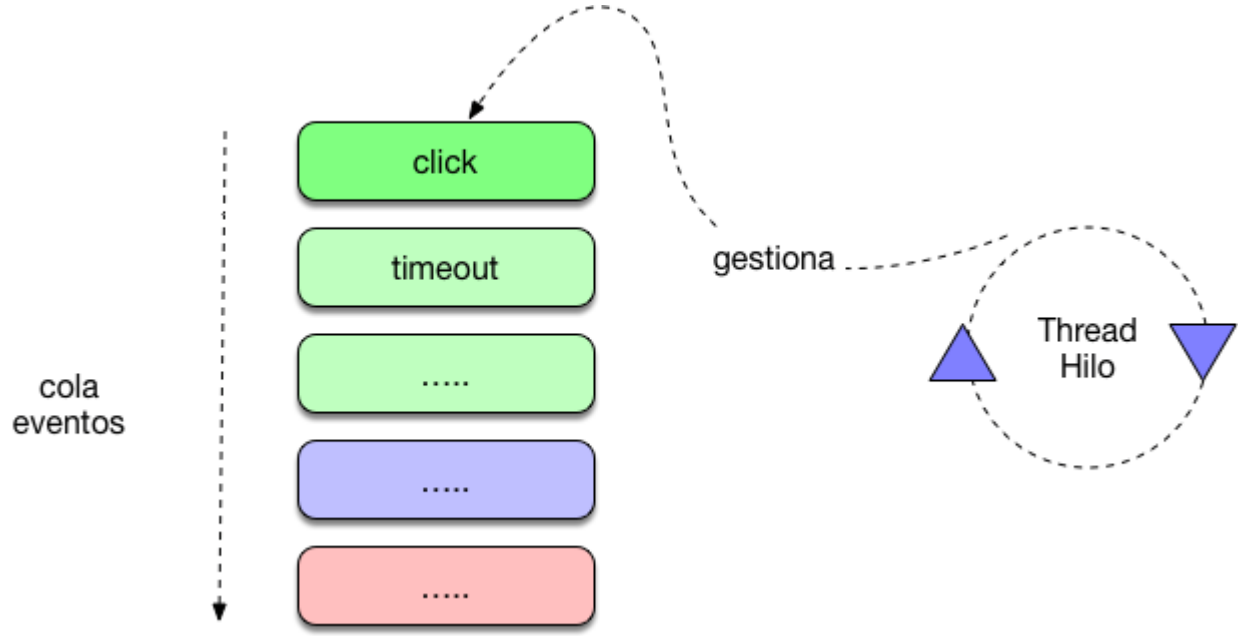
```
<html>
```

```
<head>
    <script type="text/javascript">
        function mensaje() {
            console.log("Hola desde JavaScript");
        }

        setTimeout(mensaje, 5000);

        function congelar() {
            var tiempo = new Date();
            console.dir(tiempo);

            tiempo.setSeconds(tiempo.getSeconds() + 10);

            // Bucle para simular el "congelamiento"
            while (new Date().getTime() < tiempo.getTime()) {
                // Este bloque mantendrá el navegador ocupado
            }
            console.log("Pulsaste el botón congelar");
        }
    </script>
</head>
<body>
    <input type="button" value="Congelar" onclick="congelar()" />
</body>
</html>
```

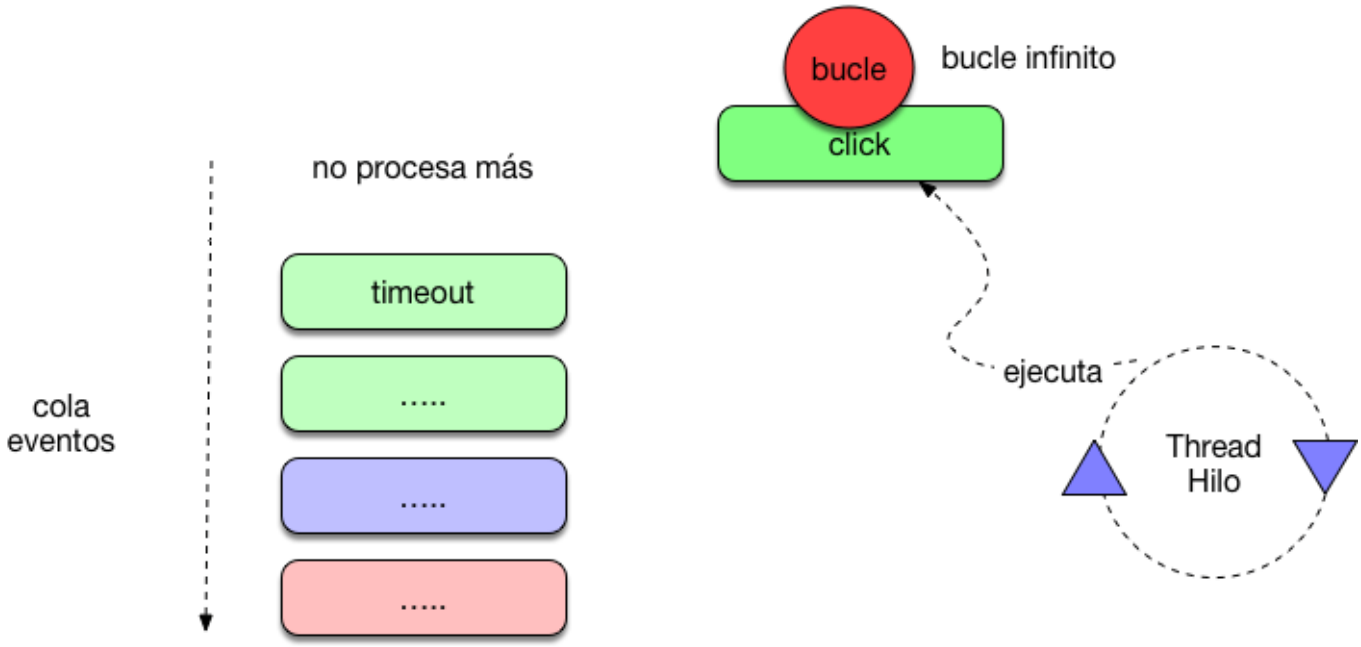
Acabamos de añadir un botón que al pulsar ejecuta la función de congelar. Esta función ejecuta un bucle infinito durante 10 segundos.



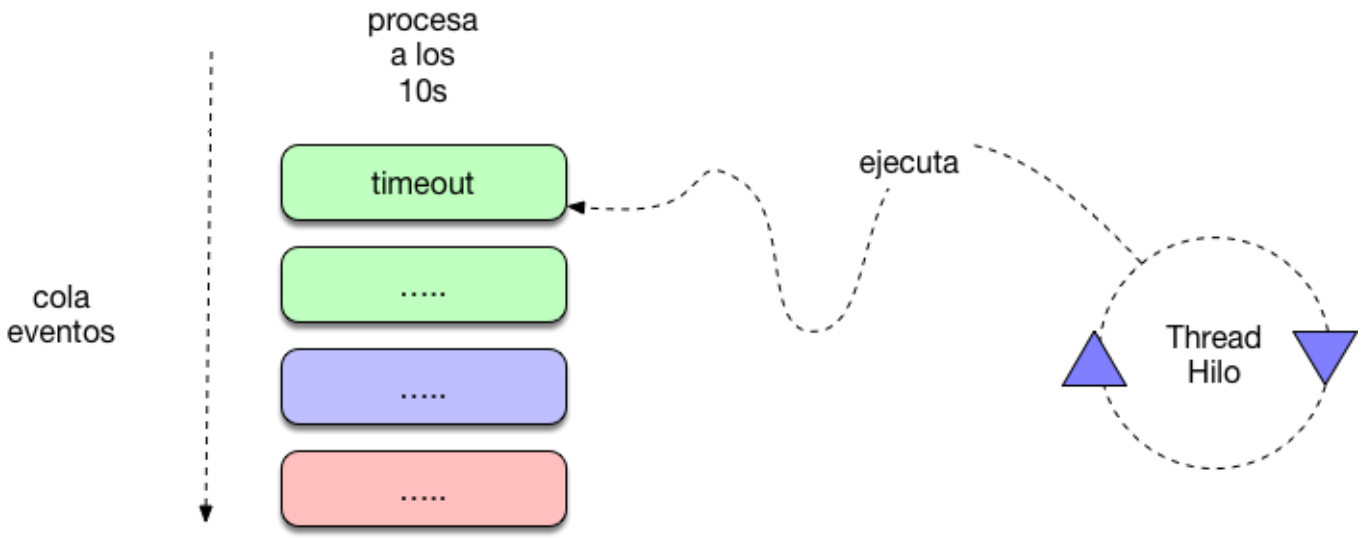
El resultado es que al ejecutar nuestro programa el setTimeout no se imprime a los 5 segundos sino que se imprime a los 10. ¿Qué es lo que ha pasado?. Debemos recordar que JavaScript solo dispone de un hilo de ejecución. Este hilo se encarga de ejecutar una cola de eventos que estén registrados.



En nuestro caso disponemos de dos eventos uno es el click y otro es el setTimeout que se ejecuta a los 5 segundos. Por lo tanto nuestra cola dispone de dos eventos que el hilo de JavaScript deberá ejecutar. Si pulsamos el botón nada mas cargar la página el primer evento que ejecutaremos será click. El problema es que genera un bucle infinito y mantendrá ocupado al thread de JavaScript durante 10 segundos.



Por lo tanto por mucho que hayamos definido el setTimeout a 5 segundos este jamas podrá ejecutarse hasta que el hilo se libere de ahí que tarde 10 segundos.



Tengamos siempre en cuenta como funciona JavaScript al lanzar operaciones de este tipo.

JavaScript setTimeout vs setInterval

La función setInterval es complementaria a setTimeout y lo que nos permite es ejecutar una cierta funcionalidad cada x tiempo . La peculiaridad es que el intervalo deberá tener una condición de finalización. Vamos a verlo:

```
<html>
<head>
    <script type="text/javascript">
        var tope = 0;
        var intervalo;

        function mensaje() {
            console.log("hola desde javascript");
            tope++;
            if (tope >= 10) {
                clearInterval(intervalo);
                console.log("Intervalo detenido después de 10
ejecuciones");
            }
        }

        function iniciarIntervalo() {
            intervalo = setInterval(mensaje, 1000);
        }
    </script>
</head>
<body>
    <input type="button" value="intervalo"
onclick="iniciarIntervalo()" />
</body>
```

```
</html>
```

Esto nos imprimirá cada segundo “hola desde Javascript” en la consola hasta que decidamos finalizar el intervalo utilizando el método clearInterval y apoyándonos en la variable tope.

Acabamos de ver las diferencias fundamentales entre JavaScript setTimeOut vs setInterval.

Otros artículos relacionados:

1. JavaScript Promise y la programación asíncrona
2. ¿Qué es un JavaScript Bundle?
3. Introducción a JavaScript ES6
4. Referencia JavaScript