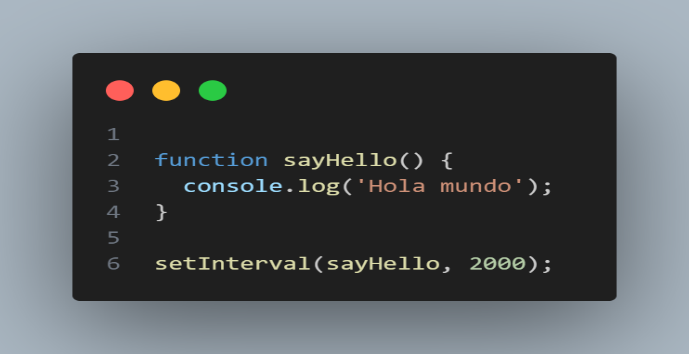
**setTimeout**

setTimeout es una función de JavaScript que le permite retrasar la ejecución de una función durante un número específico de milisegundos



**SetInterval**

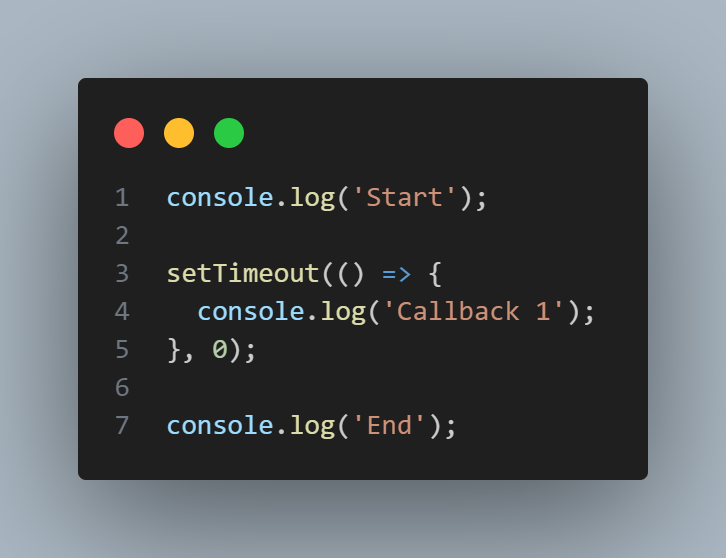
setIntervales una función de JavaScript que le permite ejecutar repetidamente una función en un intervalo específico



**Que es la asincronía y evento loop.**

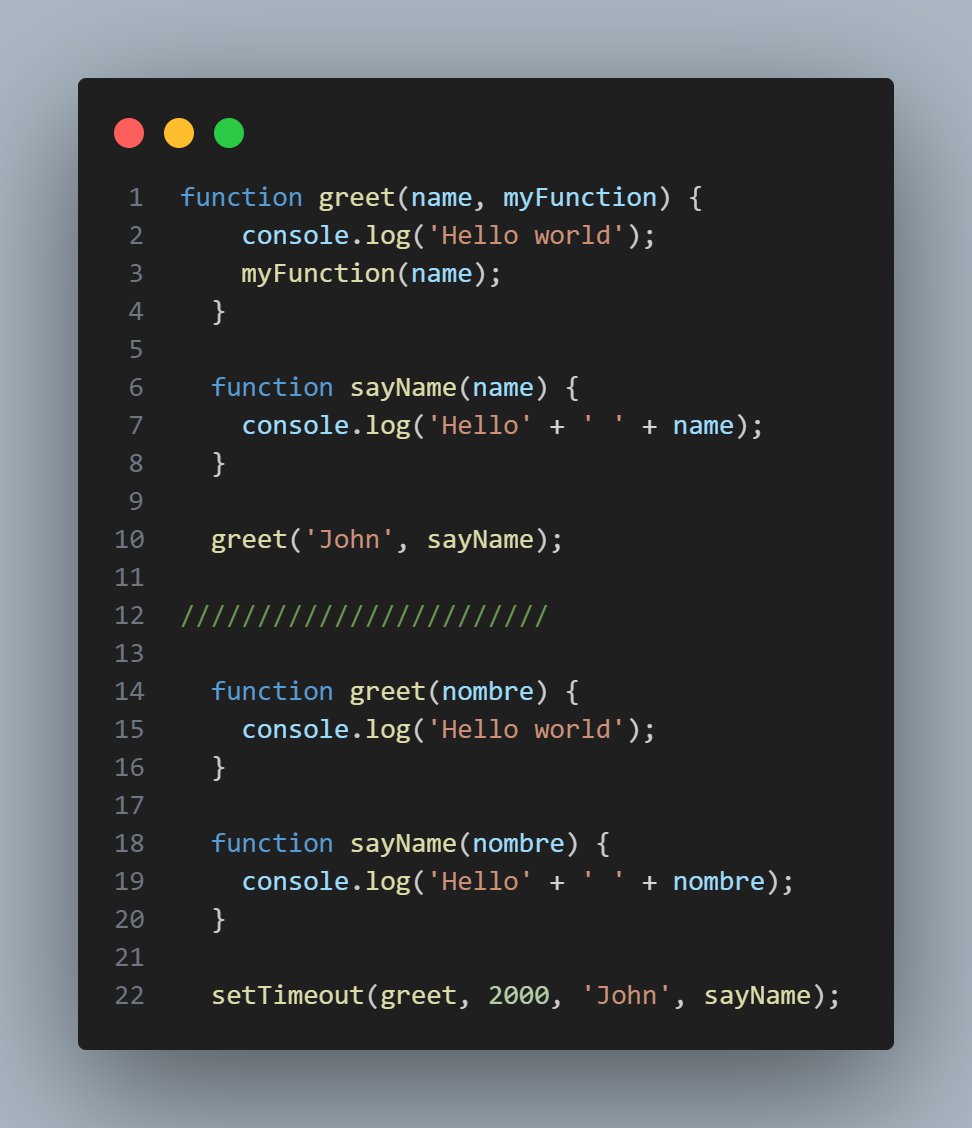
La asincronía en JavaScript se refiere a la capacidad de un programa para realizar tareas al mismo tiempo, en lugar de esperar a que se complete cada tarea antes de pasar a la siguiente. Esto es importante porque permite que JavaScript maneje tareas que pueden tardar mucho en completarse, como solicitudes de red u operaciones de E/S de archivos, sin bloquear el resto del programa.

El bucle de eventos es un componente clave del tiempo de ejecución de JavaScript que permite el comportamiento asincrónico. Es responsable de gestionar una cola de devoluciones de llamada que están programadas para ejecutarse en un momento posterior. Cuando se completa una tarea asincrónica, la devolución de llamada correspondiente se agrega a la cola. Luego, el bucle de eventos toma la devolución de llamada de la cola y la ejecuta.



**Qué son las funciones Call Back.**

Una función de devolución de llamada es una función que se pasa como argumento a otra función y se ejecuta después de que la función principal haya completado su ejecución. Las funciones de devolución de llamada se utilizan a menudo en JavaScript para manejar operaciones asincrónicas, como solicitudes de red u operaciones de E/S de archivos, que pueden tardar mucho en completarse.



**Qué son las promesas.**

Una Promesa es un objeto que representa la eventual finalización o falla de una operación asincrónica y su valor resultante. Las promesas se utilizan para manejar operaciones asincrónicas en JavaScript, como solicitudes de red u operaciones de E/S de archivos, que pueden tardar mucho en completarse.



**Qué son las funciones asíncronas.**

Las funciones asincrónicas, también conocidas como funciones asincrónicas, son una característica de JavaScript que le permite escribir código asincrónico de una manera más sincrónica. Se introducen con la async palabra clave y se utilizan para realizar tareas que tardan algún tiempo en completarse, como solicitudes de red u operaciones de E/S de archivos, sin bloquear el resto del código.

También puede usar la await palabra clave dentro de una función asíncrona para esperar a que se resuelva una promesa antes de continuar con la ejecución de la función.



**Procesamiento de un solo hilo, procesamiento de múltiples hilos.**

El procesamiento de un solo subproceso y el procesamiento de múltiples subprocesos son dos formas diferentes en que un programa puede ejecutar instrucciones.

El procesamiento de un solo subproceso, como su nombre indica, significa que un programa tiene un solo subproceso de ejecución. Esto significa que el programa sólo puede ejecutar una instrucción a la vez, de forma secuencial. JavaScript es un ejemplo de lenguaje de un solo subproceso, ya que solo tiene una pila de llamadas que se utiliza para ejecutar el programa.

Por otro lado, el procesamiento multiproceso significa que un programa tiene múltiples subprocesos de ejecución. Esto significa que el programa puede ejecutar varias instrucciones al mismo tiempo, de forma paralela. Los subprocesos múltiples se pueden implementar como subprocesos a nivel de usuario o subprocesos a nivel de kernel. Los subprocesos a nivel de usuario son implementados por los usuarios y el kernel no es consciente de la existencia de estos subprocesos, mientras que los subprocesos a nivel de kernel son manejados directamente por el sistema operativo.

El procesamiento de un solo subproceso es más sencillo de implementar, ya que no requiere manejar escenarios complejos que surgen en entornos de múltiples subprocesos, como los interbloqueos. Sin embargo, puede provocar un desperdicio de tiempo y recursos, ya que una tarea puede tardar mucho en completarse, lo que impide que se ejecuten otras tareas.

El procesamiento multiproceso, por otro lado, permite ejecutar múltiples tareas simultáneamente, mejorando el rendimiento del programa. Sin embargo, requiere manejar escenarios complejos, como la sincronización y el intercambio de recursos, lo que puede generar problemas como estancamientos y condiciones de carrera.

En resumen, el procesamiento de un solo subproceso es más sencillo de implementar, pero puede generar un desperdicio de tiempo y recursos, mientras que el procesamiento de múltiples subprocesos permite ejecutar múltiples tareas simultáneamente pero requiere manejar escenarios complejos.