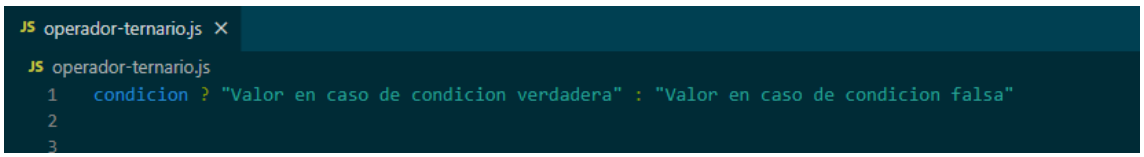
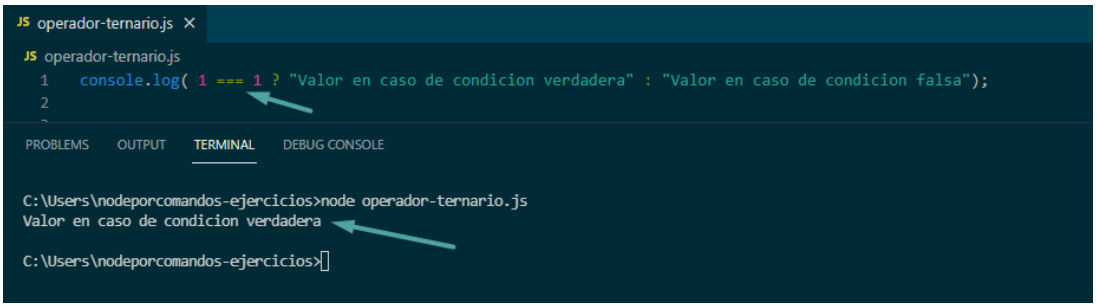
HINTS

EL OPERADOR TERNARIO

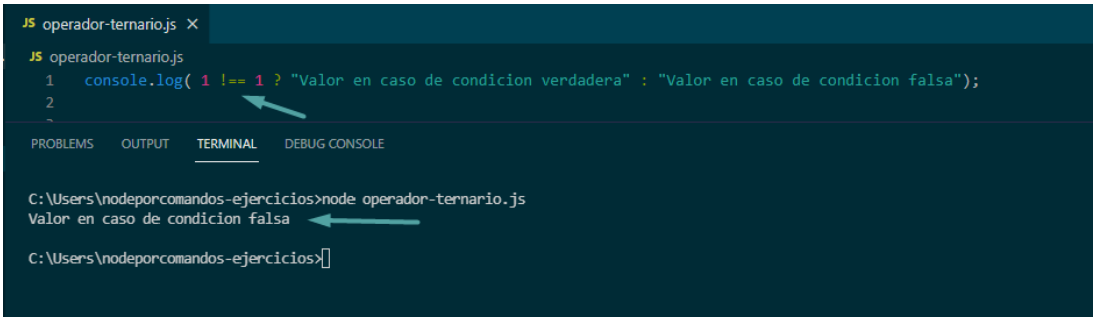
Es un bloque condicional que permite definir un resultado dependiendo de una condición, utilizando una sintaxis abreviada. Su orden es el siguiente:



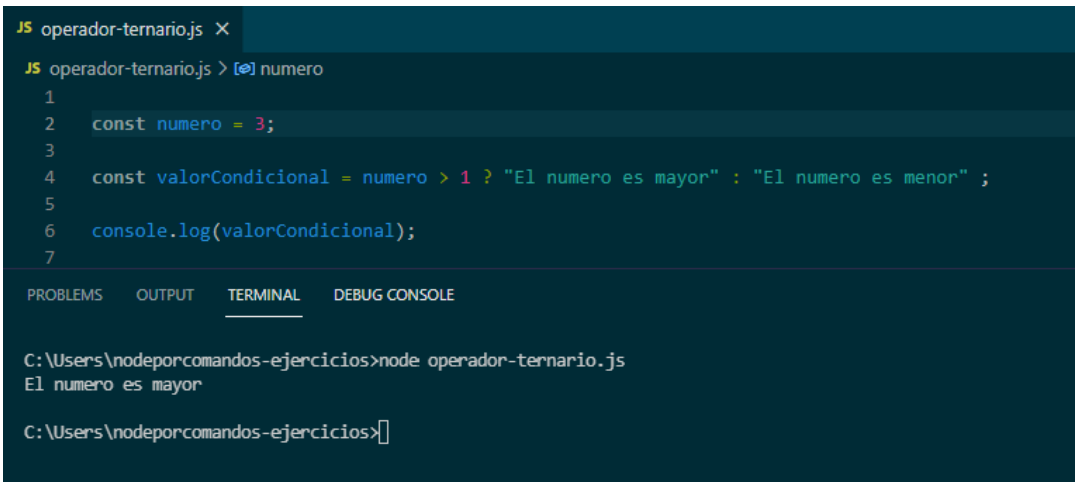
Por lo tanto, si reemplazamos condición por una condición verdadera, obtenemos el siguiente resultado:



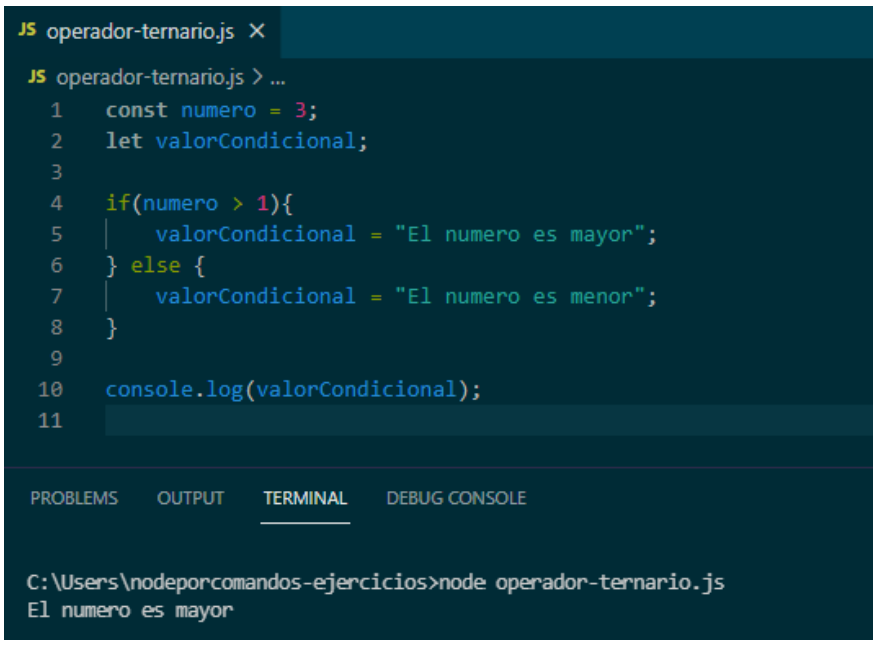
Y si utilizamos una condición falsa:



El operador ternario nos permite guardar valores condicionalmente dentro de variables:



Esto nos ahorra varias líneas de código. Si tuviéramos que definir un valor condicional utilizando un bloque `if`, nuestro código se vería así:



Por lo tanto, conocer el operador ternario abrirá una gran posibilidad para obtener un código mucho más fácil de leer, utilizando, además, menos líneas para conseguir el mismo resultado.

MÓDULO YARGS

Es una herramienta que permite construir programas interactivos con gran funcionalidad. Dentro de los ejercicios solo revisamos la superficie de todo lo que `yargs` puede hacer, pero es importante explorar todas las posibilidades que ofrece. Dentro de la página de `GitHub` de `yargs`, o incluso de la carpeta de instalación, se encuentran varios ejemplos de los distintos usos que se le puede dar. Considera que la documentación de `yargs` puede ser un poco desafiante, tal como lo hemos hablado antes, pero existen distintos estilos para escribir documentación, y algunos son más fáciles de seguir que otros. Por lo tanto, siempre puedes apoyarte con los ejemplos y videos.

Recuerda que las opciones se pueden pasar con su comando largo, utilizando doble guion `"--titulo"`, o bien, el alias con un solo guion `"-t"`.

IMPORTAR Y EXPORTAR FUNCIONES ASÍNCRONAS

Al importar y exportar funciones utilizando la palabra clave `async`, es importante mencionar que éstas **siempre devuelven una promesa.** Por lo tanto, cuando llegue el momento de importar y ejecutar una función asíncrona desde otro archivo .js, deberás utilizarla con la palabra clave `then()`, o con la sintaxis `async-await`.

CONTROL DE ERRORES

Recuerda que todas las operaciones con la palabra clave `await`, deben ser utilizadas dentro de un bloque `try-catch`. Si una promesa falla fuera de éste, tu programa dejará de correr, y levantará la excepción.

Existen métodos como `writeFile()` del módulo `fs`, que no retornan nada luego de su ejecución. Es por ello que su ejecución no va definida dentro de una variable. Para estos casos, dentro de un bloque `try-catch`, puedes escribir tranquilamente el código que necesites ejecutar, luego de la línea `await`; pues si el método falla, tu código saltará automáticamente a la sección `catch` del bloque.