MEDAC
Instituto Oficial de Formación Profesional

Preguntas generales

Antonio Jesús Pérez Ruiz

1- ¿Qué es una variable y cómo se diferencia de un identificador?

Una variable es un elemento que representa un valor y se utiliza para guardar distintos tipos de datos para poder operar con ellos posteriormente.

Un identificador es un nombre que sirve para denotar ciertos elementos de un programa. Estos elementos pueden ser variables, constantes y funciones.

La diferencia entre ambos es que un identificador es un "nombre dado a un elemento" en un programa, mientras que una variable es un "nombre dado a la ubicación de la memoria".

2- Menciona las cinco formas de declarar variables en JavaScript y sus características principales.

- **var**: Tienen un ámbito de función o global, lo que significa que están disponibles en toda la función y no tienen bloque de ámbito, por lo que no importa en qué bloque se declaren.
- **let**: Tienen un ámbito de bloque, lo que significa que están disponibles solo dentro del bloque en el que se declaran y pueden ser reasignadas después de su declaración.
- **const**: Tienen un ámbito de bloque, lo que significa que están disponibles solo dentro del bloque en el que se declaran y son útiles para declarar constantes y valores que no deben cambiar durante la ejecución del programa.
- **function**: No es una declaración de variable en sí misma, pero es importante mencionar que las funciones también pueden utilizarse para crear variables locales. Las variables declaradas dentro de una función están limitadas a ese ámbito y no son visibles fuera de la función.
- **class**: Las variables declaradas dentro de una clase pueden ser miembros de instancia o miembros de clase, y su ámbito depende de cómo se definen.

3- ¿Qué es el ámbito (scope) de una variable? Describe los tres tipos de ámbitos en JavaScript.

Se define como la región o contexto en el que una variable puede ser accedida y utilizada en un programa.

Hay tres tipos:

- **Ámbito Global (Global Scope):** Las variables declaradas en este ámbito son accesibles desde cualquier parte del código. Las variables globales se definen fuera de cualquier función o bloque y están disponibles siempre en el programa.
- **Ámbito de Función (Function Scope):** Las variables declaradas dentro de una función tienen un ámbito de función y solo son accesibles dentro de esa función. Las variables con ámbito de función no son visibles fuera de la función en la que se declaran.
- **Ámbito de Bloque (Block Scope):** Las variables declaradas con let y const tienen un ámbito de bloque y solo son accesibles dentro del bloque en el que se declaran. Esto ayuda a evitar problemas de colisión de nombres y permite un control más preciso sobre la visibilidad de las variables.

4- ¿Cuál es la diferencia entre una variable no inicializada y una no declarada en JavaScript?

-Una variable no declarada es aquella a la que intentas acceder sin haber sido previamente declarada en ningún lugar de tu código, ya sea mediante var, let, o const. Intentar acceder a una variable no declarada generará un error de referencia.

-Una variable no inicializada es una variable que ha sido declarada pero no se le ha asignado ningún valor en el momento de su declaración. Por defecto, estas variables tienen el valor especial “undefined”. Puedes acceder a una variable no inicializada sin generar un error, pero su valor será “undefined” hasta que se le asigne un valor.

5- Explica el concepto de "hoisting" en JavaScript y cómo afecta la ejecución del código.

El "hoisting" implica que las declaraciones de variables y funciones son movidas al principio de su ámbito durante la fase de compilación, antes de que se ejecute el código. Esto significa que, aunque puedes usar una variable o función antes de declararla en tu código, en realidad JavaScript la "mueve" al principio de su ámbito automáticamente durante la compilación.

El "hoisting" afecta la ejecución del código de la siguiente manera:

- **var**: Las declaraciones de variables son elevadas (hoisting) al principio de su ámbito, pudiendo usar una variable antes de declararla, pero su valor será “undefined” hasta que la variable se le asigne un valor.
- **let y const**: Las variables también son elevadas (hoisting), pero no se inicializan automáticamente con “undefined”. Intentar acceder a una variable antes de su declaración nos dará un error.
- **Funciones**: La declaración de funciones también se eleva al principio de su ámbito, por lo que puedes llamar a una función antes de declararla en el código.