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

Variable: Es un contenedor que almacena un valor que puede cambiar durante la ejecución del programa.

Identificador: Es el nombre con el que se identifica o se hace referencia a una variable. Los identificadores permiten acceder a las variables, pero no son el valor en sí.

1. ¿Qué es el ámbito (scope) de una variable?

El ámbito se refiere al contexto donde una variable está disponible para su uso.

Ámbito global: Las variables declaradas fuera de cualquier función o bloque están disponibles en cualquier parte del código, incluidas las funciones.

Ámbito de bloque (local): Las variables declaradas dentro de un bloque {} están disponibles solo dentro de ese bloque (p. ej., usando let o const).

1. Tipos primitivos de JavaScript

number: Representa tanto números enteros como decimales.

string: Representa secuencias de caracteres.

boolean: Representa dos valores: true o false.

undefined: Indica que una variable ha sido declarada pero no se le ha asignado valor.

null: Representa la ausencia intencional de cualquier valor.

symbol: Introducido en ES6, es un valor único y no modificable.

bigint: Introducido en ES2020, permite trabajar con enteros de precisión arbitraria.

1. Diferencia entre una variable no inicializada y una no declarada

Variable no inicializada: Es una variable que ha sido declarada pero no se le ha asignado ningún valor, su valor predeterminado es undefined.

Variable no declarada: Es una variable que nunca fue declarada en el código antes de su uso, lo que genera un error de referencia (ReferenceError) si intentas acceder a ella.

1. Concepto de "hoisting" en JavaScript

Hoisting es un comportamiento en JavaScript donde las declaraciones de variables y funciones son elevadas al inicio de su contexto antes de la ejecución del código. Esto implica que puedes usar una función o variable antes de declararla.

Funciones: Las declaraciones de funciones completas se elevan al principio de su ámbito y pueden ser llamadas antes de ser declaradas.

Variables (var): Solo la declaración se eleva, no la asignación. Si intentas usar una variable antes de asignarle un valor, será undefined.

let y const: Son elevadas, pero no están disponibles hasta que el código realmente alcanza su declaración (temporal dead zone).

1. Limitaciones de JavaScript y problemas que puede ocasionar

Modelo de concurrencia basado en un solo hilo (event loop):

JavaScript corre en un solo hilo, lo que significa que puede bloquear la ejecución si una operación es demasiado pesada. Esto puede causar un mal rendimiento en aplicaciones complejas.

Ámbito de variables en funciones (var):

El uso de var puede causar error