QUIZ - 43

(1.) What is JavaScript scope?

Ans. **It defines the accessibility of variables in a particular part of the code**
It defines the size of the code that can be executed at a time
It defines the layout of the code on the webpage
It defines the location of the code in the file system

(2.) Which keyword is used to declare a variable with global scope in JavaScript?

Ans. let
var
const
global

(3.) What is the difference between local and global scope in JavaScript?

Ans. Local scope variables can be accessed from anywhere in the code, whereas global scope variables can only be accessed from within a specific function.
Local scope variables can only be accessed from within a specific scope, whereas global scope variables can be accessed from anywhere in the code.
Local scope variables are declared using the var keyword, whereas global scope variables are declared using the let keyword.
There is no difference between local and global scope in JavaScript.

(4.) What is variable hoisting in JavaScript?

Ans. **It is the process of moving variable declarations to the top of their respective scopes.**
It is the process of initializing variables with default values.
It is the process of creating new variables.
It is the process of destroying variables.

(5.) Which of the following statements is true regarding block scope in JavaScript?

Ans. **Block scope variables can only be accessed from within the block they are defined in.**
Block scope variables can be accessed from anywhere in the code.
Block scope is not supported in JavaScript.
Block scope variables are declared using the var keyword.

(6.) What is the meaning of "single-threaded" in JavaScript?

Ans. **It means that JavaScript can only execute one statement at a time.**
It means that JavaScript can execute multiple statements at the same time.
It means that JavaScript can only access one webpage at a time.
It means that JavaScript can only run on a single processor core.

(7.) Why is JavaScript single-threaded?

Ans. It is a design choice made by the creators of JavaScript.
It is a limitation of the JavaScript language.
It makes JavaScript more secure.
It simplifies the programming model for developers.

(8.) What is the event loop in JavaScript?

Ans. It is a JavaScript function that executes in the background.
It is a JavaScript function that runs when a specific event occurs.
It is a mechanism that allows JavaScript to handle asynchronous code.
It is a mechanism that allows JavaScript to handle synchronous code.

(9.) Can a single-threaded JavaScript application perform multiple tasks at the same time?

Ans. Yes, by using multithreading.
Yes, by using web workers.
No, JavaScript can only perform one task at a time.
No, JavaScript can only perform multiple tasks sequentially.

(10.) What is the main advantage of a single-threaded JavaScript application?

Ans. **It is easier to program and debug.**
It can perform multiple tasks at the same time.
It is faster than a multithreaded application.
It is more secure than a multithreaded application.

(11.) What is an execution context in JavaScript?

Ans. It is a mechanism that manages the lifecycle of a JavaScript program.
It is a data structure that holds information about the current state of a JavaScript program.
It is a set of rules that govern how JavaScript code is executed.
It is a way to manage variables and functions in JavaScript.

(12.) How many types of execution contexts are there in JavaScript?

Ans. One
Two
Three
Four

(13.) What is the difference between global and local execution contexts in JavaScript?

Ans. **Global execution context is created for the entire program, whereas local execution context is created for a function.**
Local execution context is created for the entire program, whereas global execution context is created for a function.
Global execution context is created when a fun
There is no difference between global and local execution contexts.

(14.) What is the call stack in JavaScript?

Ans. **A data structure that stores function calls.**
A way to handle errors in JavaScript.
A method for debugging JavaScript code.
A technique for optimizing JavaScript performance.

(15.) Which function is at the bottom of the call stack in JavaScript?

Ans. The first function was called.
The most recently called function.
The function that is currently executing.
The main function of the program.

(16.) What happens when a function is called in JavaScript?

Ans. **The function is added to the call stack.**
The function is removed from the call stack.
The function is executed immediately.
The function is paused until all other functions have finished executing.

(17.) What is hoisting in JavaScript?

Ans. The process of declaring variables before they are used.
The process of initializing variables before they are declared.
The process of moving function declarations to the top of their scope.
The process of executing code before it is parsed.

(18.) Which of the following can be hoisted in JavaScript?

Ans. Function declarations
Function expressions
Variable declarations
All of the above

(19.) Which of the following is an example of function hoisting?

Ans. var x = 5;
function foo() { return 10; }
var y = function() { return 20; }
console.log(z);

(20.) What is the best practice to avoid unexpected behavior with hoisting in JavaScript?

Ans. Always use let or const instead of var.
Always declare variables at the top of their scope.
Avoid using function declarations and instead use function expressions.
Always use strict mode.

(21.) What is the global space in JavaScript?

Ans. The space where all the variables and functions are declared and defined.
The space where variables and functions can be accessed from any part of the program.
The space where the built-in JavaScript objects and functions are defined.
The space where the browser window or Node.js environment is located.

(22.) What is the value of a variable declared in the global scope?

Ans. **It is undefined until it is assigned a value.**
It is assigned a default value of null.
It is assigned a default value of NaN.
It is assigned a default value of 0.

(23.) What is the scope of a variable declared in the global space?

Ans. It is accessible only inside the function where it is declared.
It is accessible only inside the block where it is declared.
It is accessible from any part of the program.
It is not accessible at all.

(24.) Which of the following statements is true regarding global variables?

Ans. **Global variables should always be avoided because they can cause naming conflicts and security issues.**

Global variables should be used whenever possible to simplify the code and improve performance.

Global variables are always declared using the "var" keyword.

Global variables are always declared inside a function.

(25.) What happens if you try to redeclare a variable using the "let" keyword in the same scope?

Ans. **An error is thrown.**

The variable is reassigned with the new value.

The original variable is removed from memory and a new variable is created.

The original variable is assigned a value of undefined.

(26.) What happens if you try to redeclare a variable using the "var" keyword in the same scope?

Ans. An error is thrown.

The variable is reassigned with the new value.

The original variable is removed from memory and a new variable is created.

The original variable is assigned a value of undefined.

(27.) Which keyword should be used to declare a variable that will not be reassigned?

Ans. let

var

const

none of the above

(28.) Which of the following statements is true about "let" and "const" variables?

Ans. They are both hoisted to the top of their scope.

They can both be used before they are declared.

They both have block-level scope.

They both allow variables to be reassigned.

(29.) What is the Temporal Dead Zone in JavaScript?

Ans. **The period of time when a variable is declared but not yet assigned a value.**

The period of time between the creation of a function and its execution.

The period of time between the execution of a function and its completion.

The period of time when a variable is assigned a value but not yet declared.

(30.) What happens if you try to access a variable before it is declared in the current scope?

Ans. **An error is thrown.**

The variable is automatically initialized with a value of undefined.

The variable is automatically initialized with a value of null.

The variable is automatically initialized with a value of 0.