----------JAVA SCRIPT DAY WISE QUESTION----------

Day-1 Question topic:

Q1. Printing Statements in JavaScript //To print statements in JavaScript :Use…

Answer:

Document. write (): Document. write ('Hello, World!'); //Writes directly to the HTML document

. (not recommended for modern web development).

console. Log (): Console .log ('Hello, World!'); // Outputs to the console.

Alert (): Alert ('Hello, World!'); //Displays a pop-up alert box (mainly for debugging purposes).

Q2. Difference Between var and const

ANSWER:

var:

Scope: Function scope;
if declared outside
a function, it has global scope.

Hoisting: Variables declared with var are hoisted to the top of their scope and initialized with `undefined`.

Reassignable: Can be reassigned to a new value.

Redeclaration: Allowed within the same scope.

const:

Scope: Block scope (within {}).

Hoisting: Variables declared with **const** are . . . . . .
. hoisted but not initialized.
They are in a **"temporal dead zone"** from the start of the block until they are declared.

Reassignable: Cannot be reassigned. The variable must be initialized at the time of declaration.

Redeclaration: Not allowed within the same scope.

Q3. Rules to Declare Variable Names/Identifiers

ANSWER:

1.<u>Start with a Letter, Underscore, or Dollar Sign</u>:

Variable names must start with a **letter (a-z, A-Z),** an **underscore (_),** or a **dollar sign ($).**

FOR EXAMPLE: var _name = 'John';

var $age = 30;

2.<u>Followed by Letters, Numbers, Underscores, or Dollar Signs</u>:

After the initial character, variable names can include **letters, numbers (0-9), underscores (_),** and **dollar signs ($).**

FOR EXAMPLE**: var name1 = 'John';**

var age_2024 = 30;

3. C<u>ase Sensitive: Variable names are case sensitive.</u>

For eg ., : **var Name = 'John';** // different from var name = 'John';

4. <u>No Reserved Keywords:</u>

Cannot use JavaScript reserved keywords **FOR eg.,** var function = 'test'; // Invalid

like :

1.var,

2.let,

3.const,

4.if etc.

4. Scope in JavaScript

Answer:

<u>Scope:</u> refers to the visibility or accessibility of variables in different parts of your code.

<u>Global Scope:</u> Variables declared outside any function or block are globally accessible. **For eg .,: var global Var = 'I am global';**

<u>Function Scope:</u> Variables declared within a function are only accessible within that function.

function example() { var local Var = 'I am local'; }

<u>Block Scope</u>: Variables declared within a block **(using let or const)** are only accessible within that block.

if (true) { let block Var = 'I am block scoped'; } // block Var is not accessible here

Q5. Hoisting in JavaScript

Answer:

Hoisting: refers to the behavior in which variable declarations and function declarations are moved to the top of their containing scope during compilation.

Variables: Declarations are hoisted, but initializations are not.

```
console.log(x);  // undefined
var x = 5;
```

Functions: Entire function declarations are hoisted.

```
greet(); // Works because the function is hoisted
function greet() {
 console.log('Hello');
 }
```

6. Temporal Dead Zone (TDZ)

Answer:

Temporal Dead Zone : refers to the time from the start of a block until the variable is declared and initialized. During this time, accessing the variable will throw a ***Reference Error*****.**

Example:

```
console.log(x);          // ReferenceError: x is not defined
 let x = 10;
```

Q7. Difference Between Declaration, Initialization, and Redeclaration

Answer:

Declaration: Creating a variable by specifying its name. **var x;** // Declaration

Initialization: Assigning a value to a declared variable. **x = 5;** // Initialization

Redeclaration: Declaring a variable again in the same scope. **var x;** // First declaration

var x; // Redeclaration (allowed with var, not with let or const)

Q8. Difference Between Syntax Error, Reference Error, and Type Error

Answer:

<u>Syntax Error:</u> Occurs when the code has incorrect syntax.

Eval ('var x = ;'); // Syntax Error: Unexpected token ';'

<u>Reference Error</u>: Occurs when trying to reference a variable that doesn't exist.

console.log(x); // Reference Error: x is not defined.

<u>Type Error:</u> Occurs when an operation is performed on a value of an incorrect type.

var num = 5;

num.to Upper Case (); // Type Error: num.to Upper Case is not a function

write he difference b/w named function and arrow fuctions

what are higher order functions

explain the difference b/w rest & spread parameters

explain the Use of Default parameters

explain what are callbacks

what is lexical scope

what is scope chain

what are cloures

explain what is the call stack , event loop & webapis