JS DAYs -1

Here are the answers to your JavaScript questions:

**1. Printing Statements in JavaScript**

Java script

Copy code

// Printing to the console

console. Log ("Hello, World!"); // General print statement

console. Error ("This is an error!"); // Prints an error message

console. Warn ("This is a warning!"); // Prints a warning message

// Printing to the HTML document

document. Write ("This is printed on the webpage."); // Prints directly to the HTML body

// Printing using an alert box

Alert ("This is an alert message!"); // Displays a pop-up alert

**2. Difference Between var, let, and const**

| **Feature** | **var** | **let** | **const** |
| --- | --- | --- | --- |
| Scope | Function-scoped | Block-scoped | Block-scoped |
| Redeclaration | Allowed | Not allowed | Not allowed |
| Reassignment | Allowed | Allowed | Not allowed |
| Hoisting | Yes, but initialized as undefined | Yes, but not initialized (TDZ) | Yes, but not initialized (TDZ) |
| Use Case | Legacy code, generally avoided now | When value needs to be changed | When value should remain constant |

**3. Rules for Declaring Variable Names / Identifiers**

* **Start with a letter**, underscore (\_), or dollar sign ($). It cannot start with a number.
* **Case-sensitive**: my Var and my var are different.
* **Cannot be reserved keywords** like for, if, let, class, etc.
* **Names can contain letters, digits, underscores, and dollar signs**, but no spaces or special characters like # or @.

**4. What is Scope? Different Types of Scope in JavaScript**

**Scope** determines the accessibility (visibility) of variables.

* **Global Scope**: Variables declared outside any function or block are globally scoped and accessible from anywhere in the code.
* **Function Scope**: Variables declared inside a function using var are only accessible within that function.
* **Block Scope**: Variables declared using let and const inside a block {} are accessible only within that block.
* **Module Scope**: Variables declared inside a JavaScript module are scoped to that module.

**5. What is Hoisting in JavaScript?**

**Hoisting** is a JavaScript mechanism where variable and function declarations are moved to the top of their containing scope during compilation. However, only the declarations are hoisted, not the initializations.

Example:

Java script

Copy code

console.log(a); // undefined

var a = 5;

Here, var a is hoisted to the top, but a = 5 is not.

**6. What is Temporal Dead Zone (TDZ)?**

The **Temporal Dead Zone (TDZ)** is the period between entering a scope and the point at which a variable is declared. During this time, the variable is in an uninitialized state, and accessing it results in a Reference Error.

Example:

Java script

Copy code

console.log(x); // Reference Error

let x = 10;

**7. Difference Between Declaration, Initialization, and Redeclaration**

* **Declaration**: Declaring a variable with keywords like var, let, or const.  
  Example: let x;
* **Initialization**: Assigning a value to the variable for the first time.  
  Example: x = 5;
* **Redeclaration**: Declaring the same variable again.  
  Example: var x; (Valid with var, not with let or const)

**8. Difference Between Syntax Error, Reference Error, and Type Error**

* **Syntax Error**: Occurs when there is a mistake in the code syntax.  
  Example: if {(missing parenthesis)}
* **Reference Error**: Occurs when trying to access a variable that is not defined or is in the TDZ.  
  Example: console.log(x); (when x is not defined)
* **Type Error**: Occurs when a value is not of the expected type.  
  Example: null. f(); (trying to call a method on null)