**\*\* JAVASCRIPT INTRODUCTION \*\***

**Question 1: What is JavaScript? Explain the role of JavaScript in web development.**

JavaScript is a high-level, interpreted programming language primarily used to create interactive and dynamic content on websites. It is one of the core technologies of web development, alongside HTML and CSS.

In web development, JavaScript runs in the browser, making it a crucial part of frontend development. It enables web pages to respond to user input and interact with servers asynchronously (AJAX calls) without reloading the entire page, making websites more dynamic and engaging.

**Question 2: How is JavaScript different from other programming languages like Python or Java?**

JavaScript is primarily used for web development. It is a client-side language that runs in the web browser and is essential for interactive web pages. JavaScript is often used for front-end development and can also be used on the server-side (Node.js).

Python is a general-purpose programming language used for a variety of tasks, including web development (via frameworks like Django and Flask), data analysis, machine learning, automation, and more. It is known for its simple syntax and readability.

**Question 3: Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?**

The <script> tag in HTML is used to embed or link JavaScript code within an HTML document. It allows the browser to understand and execute the JavaScript code, which can then manipulate the DOM, handle events, and provide interactivity.

INLINE SCRIPT :-

<script>

alert("Hello, World!");

</script>

**\*\* VARIABLES AND DATATYPES \*\***

**Question 1: What are variables in JavaScript? How do you declare a variable using var, let, and const?**

Variables in JavaScript are used to store data values that can be used and manipulated throughout the program. These data values can be of different types, such as numbers, strings, objects, etc. Variables serve as containers for data that can change during the execution of the program.

In JavaScript, you declare variables using the keywords var, let, and const. This is how you declare var, let and const :- ***var:*** var name = "John"; ***let*:** let age = 25; ***const:*** const pi = 3.14;

**Question 2: Explain the different data types in JavaScript. Provide examples for each.**

JavaScript supports a variety of data types that can be divided into two categories: Primitive types and Reference types.

1. Primitive Data Types:

These are simple data types that represent a single value and are immutable (unchangeable).

***Number***: Represents both integers and floating-point numbers.

let age = 25; // Integer

let price = 19.99; // Float

***String***: Represents a sequence of characters enclosed in single ('), double ("), or backtick (`) quotes.

let name = "Alice";

let greeting = 'Hello, world!';

let templateString = `Hello, ${name}`; // Template literal

***Boolean***: Represents one of two values: true or false.

let isActive = true;

let isCompleted = false;

***Undefined***: Represents a variable that has been declared but not assigned a value. By default, uninitialized variables are undefined.

let x;

console.log(x); // Output: undefined

***Null***: Represents the intentional absence of any object value. It is often used to signify that a variable should not have a value.

let person = null;

***Symbol*** (ES6): Represents a unique and immutable value, often used as object property keys.

let uniqueKey = Symbol('id');

***BigInt*** (ES11): Represents large integers that are beyond the limits of the Number type.

let bigNumber = 1234567890123456789012345678901234567890n;

2. Reference Data Types:

These represent objects, arrays, and functions that are stored by reference.

***Object***: Represents a collection of key-value pairs. An object is a non-primitive type and can store various data types as its values.

let person = {

name: 'John',

age: 30,

isStudent: false

};

***Array***: Represents a list-like collection of values. Arrays are also objects in JavaScript.

let numbers = [1, 2, 3, 4, 5];

let fruits = ['apple', 'banana', 'orange'];

***Function:*** Functions are first-class objects in JavaScript, meaning they can be assigned to variables and passed around.

function greet(name) {

return `Hello, ${name}!`;

}

**Question 3: What is the difference between undefined and null in JavaScript?**

Both undefined and null are primitive data types in JavaScript, but they represent different concepts:

***Undefined:*** It represents a variable that has been declared but has not been assigned any value. It also represents the return value of a function that doesn't explicitly return anything.

Use Case: Typically, a variable that has been declared but not initialized is automatically assigned the value undefined.

Example:

let x;

console.log(x); // Output: undefined, because x is declared but not assigned any value

***null:*** It represents the intentional absence of any object value. It is a placeholder for "no value" and is often used to indicate that a variable should have no value.

Use Case: null is explicitly assigned to a variable to represent "nothing" or "no value".

Example:

let person = null;

console.log(person); // Output: null, because the variable is explicitly assigned null

Key Differences:

Type: undefined is a primitive type and is automatically assigned by JavaScript to variables that are declared but not initialized. null is also a primitive type but must be explicitly assigned.

Purpose: undefined usually means a variable has been declared but not assigned a value.

null indicates that a variable has been intentionally assigned to represent the absence of any value.

**\*\* JAVASCRIPT OPERATORS \*\***

**Question 1: What are the different types of operators in JavaScript? Explain with examples.**

Operators in JavaScript are used to perform operations on variables and values. The main types of operators include:

***1. Arithmetic Operators***

Arithmetic operators are used to perform mathematical calculations.

Example:

let a = 10;

let b = 5;

console.log(a + b); // Output: 15

console.log(a % b); // Output: 0

***2. Assignment Operators***

Assignment operators are used to assign values to variables.

let x = 10;

x += 5; // x = x + 5

console.log(x); // Output: 15

***3. Comparison Operators***

Comparison operators are used to compare values and return a boolean (true or false).

console.log(5 == "5"); // Output: true

console.log(5 === "5"); // Output: false

console.log(10 > 5); // Output: true

***4. Logical Operators***

Logical operators are used to combine multiple conditions.

console.log(10 > 5 && 8 > 3); // Output: true

console.log(10 > 5 || 8 < 3); // Output: true

console.log(!(10 > 5)); // Output: false

**Question 2: What is the difference between == and === in JavaScript?**

== Checks only the value, not the type 5 == "5" true

=== Checks both value and type 5 === "5" false

Explanation:

== (Loose Equality Operator): Compares values but does not check the data type. If the types are different, JavaScript tries to convert one type to match the other.

=== (Strict Equality Operator): Compares both the value and the data type. If the types are different, it returns false.

console.log(5 == "5"); // true (because JavaScript converts "5" to a number)

console.log(5 === "5"); // false (different data types)

console.log(0 == false); // true (because `false` is converted to `0`)

console.log(0 === false); // false (different types)

**\*\* CONTROL FLOW (IF-ELSE, SWITCH) \*\***

**Question 1: What is control flow in JavaScript? Explain how if-else statements work with an example.**

Control flow refers to the order in which statements are executed in a JavaScript program. Normally, code runs from top to bottom, but control flow structures like conditional statements (if-else, switch) and loops (for, while) allow us to control execution based on conditions.

If-Else Statements

The if-else statement allows the program to make decisions based on conditions.

Syntax:

let age = 18;

if (age >= 18) {

console.log("You are eligible to vote.");

} else {

console.log("You are not eligible to vote.");

}

// Output: "You are eligible to vote."

**Question 2: Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?**

The switch statement is used when multiple conditions need to be checked against a single value. It is often used instead of long if-else chains.

Syntax:

let day = 3;

switch (day) {

case 1:

console.log("Monday");

break;

case 2:

console.log("Tuesday");

break;

case 3:

console.log("Wednesday");

break;

default:

console.log("Invalid day");

}

// Output: "Wednesday"

***When to Use switch Instead of if-else?***

When checking multiple possible values for a single variable. When code readability is improved by avoiding long if-else chains.

**\*\* LOOPS (FOR, WHILE, DO-WHILE) \*\***

**Question 1: Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.**

Loops are used to execute a block of code multiple times.

***1. For Loop***

Used when the number of iterations is known.

Syntax:

for (let i = 1; i <= 5; i++) {

console.log(i);

}

// Output: 1 2 3 4 5

***2. While Loop***

Used when the number of iterations is unknown but based on a condition.

Syntax:

let num = 1;

while (num <= 3) {

console.log(num);

num++;

}

// Output: 1 2 3

***3. Do-While Loop***

Similar to while, but runs at least once.

Syntax:

let x = 5;

do {

console.log(x);

x--;

} while (x > 3);

// Output: 5 4

**\*\* FUNCTION \*\***

**Question 1: What are functions in JavaScript? Explain the syntax for declaring and calling a function.**

A function is a reusable block of code that performs a specific task.

Syntax:

function greet(name) {

console.log("Hello, " + name);

}

greet("Alice");

// Output: "Hello, Alice"

**Question 2: Discuss the concept of parameters and return values in functions.**

Parameters: Variables passed to a function.

Return Values: The value a function gives back.

Example:

function add(a, b) {

return a + b;

}

let sum = add(5, 3);

console.log(sum); // Output: 8

**\*\* ARRAYS IN JAVASCRIPT \*\***

**Question 1: What is an array in JavaScript? How do you declare and initialize an array?**

An array in JavaScript is a special type of object used to store multiple values in a single variable. Arrays allow easy access and manipulation of data. They can contain elements of different data types, including numbers, strings, and even other arrays.

Declaring and Initializing an Array:

1. Using square brackets (preferred method): let fruits = ["Apple", "Banana", "Cherry"];

2. Using the `new Array()` constructor: let numbers = new Array(1, 2, 3, 4, 5);

**Question 2: Explain the methods push(), pop(), shift(), and unshift() used in arrays.**

***push()***: Adds a new element to the end of an array.

let fruits = ["Apple", "Banana"];

fruits.push("Cherry");

console.log(fruits); // Output: ["Apple", "Banana", "Cherry"]

***pop():*** Removes the last element from an array.

fruits.pop();

console.log(fruits); // Output: ["Apple", "Banana"]

***shift():*** Removes the first element from an array.

fruits.shift();

console.log(fruits); // Output: ["Banana"]

***unshift():*** Adds a new element to the beginning of an array.

fruits.unshift("Mango");

console.log(fruits); // Output: ["Mango", "Banana"]

**\*\* JAVASCRIPT OBJECTS \*\***

**Question 1: What is an object in JavaScript? How are objects different from arrays?**

An object in JavaScript is a collection of key-value pairs where each key is a string (or symbol) and its associated value can be any data type. Objects are used to store structured data and represent real-world entities.

Difference Between Objects and Arrays:

Arrays are ordered collections of values indexed by numbers.

Objects are collections of properties indexed by keys (strings or symbols).

**Question 2: Explain how to access and update object properties using dot notation and bracket notation.**

***Dot Notation:*** Used when the property name is a valid JavaScript identifier.

console.log(person.name); // Output: "John"

person.age = 31;

***Bracket Notation:*** Used when the property name has special characters or is stored as a variable.

console.log(person["city"]); // Output: "New York"

person["city"] = "Los Angeles";

**\*\* JAVASCRIPT EVENTS \*\***

**Question 1: What are JavaScript events? Explain the role of event listeners.**

Events in JavaScript are actions that occur in the browser, such as clicks, key presses, or form submissions. Event listeners help detect and respond to these actions.

Example of an event listener:

document.getElementById("myButton").addEventListener("click", function() {

alert("Button clicked!");

});

**Question 2: How does the addEventListener() method work in JavaScript? Provide an example.**

`addEventListener()` attaches an event handler to an element without overwriting existing event handlers.

Example:

let button = document.getElementById("clickMe");

button.addEventListener("click", function() {

console.log("Button was clicked!");

});

**\*\* DOM MANIPULATION \*\***

**Question 1: What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?**

The DOM (Document Object Model) is a representation of the HTML document as a tree structure where each node represents an element or attribute. JavaScript interacts with the DOM to dynamically update content, styles, and structure.

Example of modifying the DOM:

document.getElementById("demo").innerHTML = "Hello, JavaScript!";

**Question 2: Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM.**

***getElementById()***: Selects an element by its `id`.

let element = document.getElementById("myElement");

***getElementsByClassName()***: Selects multiple elements with a specific class.

let elements = document.getElementsByClassName("myClass");

***querySelector():*** Selects the first matching element using a CSS selector.

let element = document.querySelector(".myClass");

**\*\* JAVASCRIPT TIMING EVENTS (SET TIMEOUT, SET INTERVAL) \*\***

**Question 1: Explain the setTimeout() and setInterval() functions in JavaScript. How are they used for timing events?**

***setTimeout()***: Executes a function after a specified delay (in milliseconds). It runs only once.

setTimeout(function() {

console.log("Hello after 3 seconds");

}, 3000);

***setInterval()***: Executes a function repeatedly at a specified interval (in milliseconds).

setInterval(function() {

console.log("This runs every 2 seconds");

}, 2000);

**Question 2: Provide an example of how to use setTimeout() to delay an action by 2 seconds.**

Example:

setTimeout(function() {

console.log("This message appears after 2 seconds");

}, 2000);

**\*\* JAVASCRIPT ERROR HANDLING \*\***

**Question 1: What is error handling in JavaScript? Explain the try, catch, and finally blocks with an example.**

Error handling in JavaScript helps manage and respond to runtime errors, preventing unexpected crashes.

try: Defines a block of code to test for errors.

catch: Defines a block of code to handle the error if one occurs.

finally: (Optional) Defines a block of code that runs regardless of the error.

Syntax:

try {

let result = x / 10; // x is undefined, causing an error

} catch (error) {

console.log("An error occurred: " + error.message);

} finally {

console.log("This code runs regardless of an error");

}

**Question 2: Why is error handling important in JavaScript applications?**

Prevents crashes: Ensures the app keeps running even when errors occur.

Improves debugging: Helps find and fix issues efficiently.

Enhances user experience: Shows helpful error messages instead of breaking the app.