**JAVASCRIPT BASIC & DOM**

1. **What is JavaScript. How to use it?**

**Ans: -** JavaScript is a scripting or programming language that allows you to implement complex features on web pages.

JavaScript is the Programming Language for the Web.

JavaScript can update and change both HTML and CSS.

JavaScript can calculate, manipulate and validate data.

A very common use of JavaScript is to dynamically modify HTML and CSS to update a user interface, via the Document Object Model API. the code in your web documents is generally loaded and executed in the order it appears on the page. Errors may occur if JavaScript is loaded and run before the HTML and CSS that it is intended to modify. You will learn ways around this later in the article, in the Script loading strategies section.

1. **How many types of Variables in JavaScript?**

**Ans: -**

In JavaScript, there are three primary ways to declare variables, which can broadly be categorized into three types:

1. var: var was the original way to declare variables in JavaScript. Variables declared with var are function-scoped, meaning their scope is limited to the function in which they are defined or, if declared outside of any function, they have global scope. Variables declared with var can be re-declared and updated within their scope.

Example: -

var x = 10;

1. let: let was introduced in ECMAScript 6 (ES6) and provides block-scoping, meaning their scope is limited to the block ({}) in which they are defined. Variables declared with let can be reassigned but cannot be re-declared within the same scope.

Example: -

let y = 20;

1. const: const also appeared in ES6 and is used to declare constants. Variables declared with const must be assigned a value when they are declared and cannot be reassigned or re-declared. However, for objects and arrays, while the variable itself cannot be reassigned, the properties or elements of the object or array can still be modified.

Example: -

const z = 30;

These variable types provide flexibility and control over the scope and mutability of variables in JavaScript code. It's generally recommended to use let and const over var due to the advantages they offer in terms of scoping and predictability of behaviour.

**3. Define a Data Types in js?**

**Ans**: - JavaScript has several built-in data types that are used to represent different kinds of values. These data types can be categorized as follows:

1. Primitive Data Types:

Number: Represents numeric values, both integers and floating-point numbers.

String: Represents sequences of characters, enclosed within single (''), double ("") or backticks (``).

Boolean: Represents a logical value, either true or false.

Null: Represents the intentional absence of any value. It is a primitive value.

Undefined: Represents an uninitialized variable or missing property. It is also a primitive value.

2. Non-primitive Data Types (Reference Types):

Object: Represents a collection of key-value pairs where keys are strings and values can be any data type, including other objects, functions, and arrays.

Array: Represents a list-like object, where elements can be of any data type and are accessed by numeric indices.

Function: Functions in JavaScript are first-class objects, meaning they can be assigned to variables, passed as arguments, and returned from other functions.

Here's an example demonstrating these data types:

// Primitive Data Types

let num = 10; // Number

let str = 'Hello'; // String

let bool = true; // Boolean

let nul = null; // Null

let undf; // Undefined

// Non-primitive Data Types

let obj = {key: 'value’}; // Object

let arr = [1, 2, 3]; // Array

let func = function () {return 'Function';}; // Function

These data types provide the foundation for working with different kinds of values in JavaScript. Understanding them is crucial for effectively manipulating data and building JavaScript applications.

**4. What the deference between undefined and undeclare in JavaScript?**

**Ans: -**

In JavaScript, "undefined" and "undeclared" refer to different concepts:

1. Undefined:

In JavaScript, undefined is a primitive value that represents the lack of a value or the absence of a value where one was expected.

Variables that are declared but not assigned a value, or properties that do not exist within an object, have a value of undefined.

Example: -

let x;

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

let obj = {};

console.log (obj. property); // Output: undefined

2. Undeclared:

"Undeclared" refers to a variable that has been referenced in code but has not been declared using var, let, or const.

When you try to access a variable that has not been declared, JavaScript will throw a Reference Error.

Example: -

console.log(y); // ReferenceError: y is not defined

In summary, undefined is a value that represents the absence of a value, typically seen when a variable has been declared but not yet assigned a value or when accessing non-existent properties. "Undeclared" refers to variables that have not been declared at all in the code.

**5. Using console.log () print out the following statement: The quote 'There is no exercise better for the heart than reaching down and lifting people up.' by John Holmes teaches us to help one another. Using console.log () print out the following quote by Mother Teresa:**

**Ans: -** You can print out both statements using console.log () as follows:

console.log ("The quote 'There is no exercise better for the heart than reaching down and lifting people up.' by John Holmes teaches us to help one another.");

console.log ("The quote 'If you can't feed a hundred people, then feed just one.' by Mother Teresa.");

This code will output the provided statements to the console:

The quote 'There is no exercise better for the heart than reaching down and lifting people up.' by John Holmes teaches us to help one another.

The quote 'If you can't feed a hundred people, then feed just one.' by Mother Teresa.

**6. Check if typeof '10' is exactly equal to 10. If not make it exactly equal?**

**Ans: -**

In JavaScript, the typeof operator returns a string indicating the type of the operand.

If you check typeof '10', it will return 'string'. To check if it's exactly equal to 10, which is a number, you need to convert '10' to a number first.

You can use the Number () function to convert a string to a number. Then, you can compare the typeof result with the number 10 using the === operator to check if they are exactly equal.

if (typeof '10’! == typeof 10) {

// Convert '10' to a number

let num = Number ('10');

// Check if the typeof result is exactly equal to 10

if (typeof num === typeof 10) {

console.log ("'10' is now exactly equal to 10.");

} else {

console.log ("'10' cannot be made exactly equal to 10.");

}

} else {

console.log ("'10' is already exactly equal to 10.");

}

This code will output '10' is now exactly equal to 10. since it converts the string '10' to a number, making it exactly equal to the number 10.

**7. Write a JavaScript program to calculate days left until next Christmas?**

**Ans: -**

// Get the current date

let today = new Date ();

// Get the current year

let currentYear = today. getFullYear ();

// Get the next Christmas date for the current year

let christmas = new Date (currentYear, 11, 25); // Month is 0-indexed, so December is 11

// If Christmas has already passed this year, get the next year's Christmas date

if (today. getMonth () === 11 && today.getDate() > 25) {

christmas. setFullYear (currentYear + 1);

}

// Calculate the difference in milliseconds between today and Christmas

let difference = christmas. getTime () - today. getTime ();

// Convert the difference from milliseconds to days

let daysLeft = Math.ceil(difference / (1000 \* 3600 \* 24));

// Output the number of days left until Christmas

console.log ("There are " + daysLeft + " days left until Christmas!");

**8. What is Condition Statement?**

**Ans: -** A condition statement, also known as a conditional statement or a control structure, is a programming construct that allows you to execute different blocks of code based on whether a specified condition evaluates to true or false. Condition statements are fundamental to control the flow of a program and enable you to create dynamic, decision-based behaviour within your code.

In most programming languages, including JavaScript, there are several types of condition statements:

1. if statement: The if statement executes a block of code if a specified condition is true. It can be followed by an optional else statement, which executes a different block of code if the condition is false.

Example: -

if (condition) {

} else {

}

2. else if statement: The else if statement provides an additional condition to test if the first condition of the if statement is false. It can be followed by an optional else statement.

Example: -

if (condition1) {

} else if (condition2) {

} else {

}

3. switch statement: The switch statement evaluates an expression and executes a block of code depending on a matching case label. It provides a cleaner alternative to multiple if statements when you have multiple possible conditions to check.

Example: -

switch (expression) {

case value1:

break;

case value2:

break;

default

}

Condition statements are crucial for implementing logic and making decisions within your code. They allow your programs to adapt and respond dynamically to different situations based on the values of variables, user input, or other conditions.

**9. What is the result of the expression (5 > 3 && 2 < 4)?**

**Ans: -** The expression (5 > 3 && 2 < 4) in JavaScript utilizes the logical AND operator (&&).

The logical AND operator returns true if both of its operands are true; otherwise, it returns false.

Let's evaluate each part of the expression:

5 > 3 evaluates to true because 5 is indeed greater than 3.

2 < 4 evaluates to true because 2 is indeed less than 4.

Since both conditions evaluate to true, the logical AND operator returns true.

So, the result of the expression (5 > 3 && 2 < 4) in JavaScript is true.

To check the result of the expression (5 > 3 && 2 < 4) in JavaScript, you can simply use console.log () to output the result to the console. Here's the code:

Example: -

console.log (5 > 3 && 2 < 4);

When you run this code, it will output true to the console because both conditions (5 > 3 and 2 < 4) are true, and the logical AND operator && returns true when both conditions are met.

**10. What is a Loop and Switch Case in JavaScript define that?**

**Ans: -**

In JavaScript, loops and switch cases are control flow constructs that allow you to execute blocks of code repeatedly or conditionally based on certain criteria.

1.Loop: -

A loop is a programming construct that allows you to execute a block of code repeatedly until a specified condition is met.

There are several types of loops in JavaScript:

1.for loop: Executes a block of code a specified number of times.

2.while loop: Executes a block of code as long as a specified condition is true.

3.do-while loop: Similar to a while loop, but it always executes the block of code at least once before checking the condition.

4.for...in loop: Iterates over the enumerable properties of an object.

Example: -

for (let i = 0; i < 5; i++) {

console.log(i);

}

2. Switch Case:

The switch statement is a control flow statement that evaluates an expression and executes code based on matching cases.

It provides an alternative to long chains of if...else if...else statements when you have multiple possible conditions to check.

The switch statement evaluates the expression once and then executes the code associated with the matching case, or the default case if no match is found.

Example: -

let day = "Monday";

switch (day) {

case "Monday":

console.log ("Today is Monday.");

break;

case "Tuesday":

console.log ("Today is Tuesday.");

break;

case "Wednesday":

console.log ("Today is Wednesday.");

break;

default:

console.log ("Today is not Monday, Tuesday, or Wednesday.");

}

**11. What is the use of is Nan function?**

**Ans:** - In JavaScript, the isNaN () function is used to determine whether a value is NaN (Not-a-Number) or not. NaN is a special value that represents the result of an invalid or undefined mathematical operation, such as dividing zero by zero.

The isNaN() function takes one argument and returns true if the argument is NaN, and false otherwise. If the argument is not already of the number type, isNaN () attempts to convert it to a number before determining whether it's NaN.

**12. What is the difference between && and || in JavaScript?**

**Ans: -** In JavaScript, `&&` and `||` are logical operators used for combining conditions and evaluating expressions.

1. \*\*Logical AND (`&&`) \*\*:

- The `&&` operator returns `true` if both operands are `true`; otherwise, it returns `false`.

- It short-circuits evaluation: if the left operand is `false`, the right operand is not evaluated, because the overall result will always be `false` regardless of the right operand's value.

- Example: `true && true` evaluates to `true`, `true && false` evaluates to `false`, `false && true` evaluates to `false`, and `false && false` evaluates to `false`.

2. \*\*Logical OR (`||`) \*\*:

- The `||` operator returns `true` if at least one of the operands is `true`; otherwise, it returns `false`.

- It also short-circuits evaluation: if the left operand is `true`, the right operand is not evaluated, because the overall result will always be `true` regardless of the right operand's value.

- Example: `true || true` evaluates to `true`, `true || false` evaluates to `true`, `false || true` evaluates to `true`, and `false || false` evaluates to `false`.

**ARRAY AND OBJECT QUESTIONS**

**1. Write a JavaScript Program to display the current day and time in the following format.**

**Sample Output: Today is Friday. Current Time is 12 PM: 12: 22 2?**

**Ans: -**

let currentDate = new Date ();

let weekdays = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];

let currentDay = weekdays [currentDate.getDay()];

let currentHour = currentDate.getHours();

let ampm = currentHour >= 12? 'PM’: 'AM';

currentHour = currentHour % 12;

currentHour = currentHour? currentHour: 12;

let currentMinutes = currentDate.getMinutes();

let currentSeconds = currentDate.getSeconds();

console.log ("Today is " + currentDay + ". Current Time is " + currentHour + " " + ampm + ": " + currentMinutes + “: " + currentSeconds);

**2. Write a JavaScript program to get the current date?**

**Ans: -**

let currentDate = new Date ();

let year = currentDate.getFullYear();

let month = currentDate.getMonth() + 1; // Months are zero-indexed, so we add 1

let day = currentDate.getDate();

let formattedDate = year + '-' + (month < 10? '0’: '') + month + '-' + (day < 10? '0’: '') + day;

console.log ("Current date is: " + formattedDate);

**3. form validation in js**

**Ans: -**

<body>

<table cellspacing="20">

<tr>

<td>FIRST NAME</td>

<td>

<input type="text" id="zero">

</td>

<td id="izero"></td>

</tr>

<tr>

<td>NUMBER</td>

<td>

<input type="number" id="zero">

</td>

<td id="izero"></td>

</tr>

<tr>

<td>ENTER PASSWORD</td>

<td>

<input type="password" id="one">

</td>

<td id="ione"></td>

</tr>

<tr>

<td>CONFIRM PASSWORD</td>

<td>

<input type="password" id="two">

</td>

<td id="itwo"> </td>

</tr>

<tr>

<td>

<button onclick="abc()">submit</button>

</td>

<td id="demo">

</td>

</tr>

</table>

<script>

function abc () {

let a = document.getElementById ("one").value;

let b = document. getElementById("two").value;

let c = document. getElementById("zero").value;

if (a && b && c && a === b) {

document. getElementById("demo"). innerHTML = "password is correct"

document. getElementById("demo"). style. color = "green"

}

else {

if (! c) {

document. getElementById("izero"). innerHTML = "ENTER FIRST NAME"

document. getElementById("izero"). Style. color = "red"

}

if (! a) {

document. getElementById("ione"). innerHTML = "ENTER VALUE ONE"

document. getElementById("ione"). style. color = "red"

}

if (! b) {

document. getElementById("itwo"). innerHTML = "ENTER VALUE TWO"

document. getElementById("itwo"). style. color = "red"

}

if (! a ===! b) {

document. getElementById("demo"). innerHTML = "password is incorrect"

document. getElementById("demo"). style. color = "red"

}

}

}

</script>

</body>

**BASIC QUESTIONS**

**1. What is JavaScript?**

**Ans: -**

JavaScript is a high-level, interpreted programming language primarily used for adding interactivity and dynamic behaviour to web pages. Originally developed by Brendan Eich at Netscape Communications Corporation, JavaScript is now standardized as ECMAScript and is widely supported by web browsers.

**2. What is the use of isNaN function?**

**Ans: -**

The isNaN () function in JavaScript is used to determine whether a value is NaN (Not-a-Number) or not. NaN is a special value in JavaScript that represents the result of an invalid or undefined mathematical operation, such as dividing zero by zero.

The isNaN () function takes one argument and returns true if the argument is NaN, and false otherwise. If the argument is not already of the number type, isNaN () attempts to convert it to a number before determining whether it's NaN.

**3. Which company developed JavaScript?**

**Ans:** - JavaScript was developed by Netscape Communications Corporation, particularly by Brendan Eich, in 1995.

**4. What are undeclared and undefined variables?**

**Ans: -** Undeclared and undefined variables are terms used to describe different states of variables in JavaScript:

1. Undeclared Variables:

- Undeclared variables are those that have not been declared using the `var`, `let`, or `const` keywords before their first use in the code.

- When you try to access an undeclared variable, JavaScript will throw a `ReferenceError`.

- For example:

console.log(myVariable);

2. Undefined Variables:

- Undefined variables are those that have been declared, but have not been assigned a value, or have been explicitly set to `undefined`.

- When you access an undefined variable, JavaScript will return the value `undefined`.

- For example:

let myVariable;

console.log(myVariable);

```

In summary, undeclared variables refer to variables that have not been declared at all, while undefined variables are variables that have been declared but have not been assigned a value or have been explicitly set to `undefined`. Both cases can lead to issues in your code and should be handled appropriately. It's generally recommended to always declare your variables before using them to avoid unintended behavior.

**5. What is === operator?**

**Ans:** - The === operator is called the strict equality operator in JavaScript. It is used to compare two values for equality without performing type coercion. In other words, it checks whether the two values are not only equal in value but also of the same data type.

Here's how the === operator works:

If the operands are of the same data type and have the same value, === returns true.

If the operands are of different data types or have different values, === returns false.

**6. What are all the looping structures in JavaScript?**

**Ans: -** JavaScript provides several looping structures for executing a block of code repeatedly. The common looping structures in JavaScript are:

1. for Loop:

- The `for` loop is used when you know how many times you want to execute a block of code.

- It consists of an initialization, a condition, and an iteration expression.

- Example:

for (let i = 0; i < 5; i++) {

console.log(i);

}

2. while Loop:

- The `while` loop is used when you want to execute a block of code as long as a condition is true.

- It evaluates the condition before executing the block of code.

- Example:

let i = 0;

while (i < 5) {

console.log(i);

i++;

}

```

3. do...while Loop:

- The `do...while` loop is similar to the `while` loop, but it executes the block of code once before checking the condition.

- It ensures that the block of code is executed at least once, even if the condition is false.

- Example:

let i = 0;

do {

console.log(i);

i++;

} while (i < 5);

4. for...in Loop:

- The `for...in` loop iterates over the enumerable properties of an object.

- It is often used to loop through the keys of an object.

- Example:

const person = {name: 'John', age: 30};

for (let key in person) {

console.log (key + ': ' + person[key]);

}

**7. What is the function of the delete operator?**

**Ans: -** In JavaScript, the delete operator is used to remove a property from an object. It can also be used to remove an element from an array, although it's not recommended for this purpose because it leaves a gap in the array (the deleted element becomes undefined) and does not update the length of the array.

Example: -

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

delete fruits [1];

console.log(fruits);

console.log (fruits. Length);