**Module (JAVASCRIPT BASIC & DOM) – 4**

**(Basic logic Question)**

* Q1\_What is JavaScript. How to use it?
  + JavaScript is a object oriented programming language which can be used by developer to make web pages interactive.
  + How to use JavaScript:
    - Open Chrome on your computer.
    - Click. then Settings.
    - Click Privacy and Security.
    - Click Site settings.
    - Click JavaScript.
    - Select Sites can use Javascript.
* Q2\_How many types of Variables in JavaScript?
  + In JavaScript, there are two main types of variables:
  + Primitive Data Types:
    - Null: Represents the intentional absence of any object value.
    - Number: Represents numeric values (integers or floating-point numbers).
    - String: Represents sequences of characters enclosed in single or double quotes.
    - Symbol: Introduced in ECMAScript 6 (ES6), symbols are unique and immutable values often used as object keys.
    - Boolean: Represents either true or false.
    - Undefined: Represents a variable that has been declared but not assigned a value.
  + Reference Data Types:
    - Object: Represents a collection of key-value pairs. Objects can be created using curly braces {} or via the new Object() constructor.
    - Array: A special type of object used for storing and manipulating ordered lists of values.
    - Function: Functions are also objects in JavaScript, and they can be defined using the function keyword.
* Q3\_Define a Data Types in js?
  + In JavaScript, data types are classifications of values that determine the kind of operations that can be performed on them and the way they are stored in memory. JavaScript has two main categories of data types: Primitive Data Types and Reference Data Types.
  + Primitive Data Types:
    - Null: Represents the intentional absence of any object value.
    - Number: Represents numeric values (integers or floating-point numbers).
    - String: Represents sequences of characters enclosed in single or double quotes.
    - Symbol: Introduced in ECMAScript 6 (ES6), symbols are unique and immutable values often used as object keys.
    - Boolean: Represents either true or false.
    - Undefined: Represents a variable that has been declared but not assigned a value.
  + Reference Data Types:
    - Object: Represents a collection of key-value pairs. Objects can be created using curly braces {} or via the new Object() constructor.
    - Array: A special type of object used for storing and manipulating ordered lists of values.
    - Function: Functions are also objects in JavaScript, and they can be defined using the function keyword.
    - Date: Represents a specific moment in time.
    - RegExp: Represents a regular expression, used for pattern matching with strings.
* Q5\_What the deference between undefined and undeclare in JavaScript?
  + Undefined:
    - Meaning: A variable is "undefined" when it has been declared, but it has not been assigned a value.
    - Example:
    - let x;
    - console.log(x); // Output: undefined
    - In this example, x has been declared using the let keyword, but it has not been assigned a value yet. As a result, its value is undefined.
  + Undeclared:
    - Meaning: An undeclared variable is one that has not been declared using the var, let, or const keywords before being used.
    - Example:
    - console.log(y); // ReferenceError: y is not defined
    - In this example, y is undeclared because it has not been declared using var, let, or const before being used. Attempting to access an undeclared variable results in a ReferenceError.
* Q6\_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.'
  + console.log("The quote 'There is no exercise better for the heart than reaching down and lifting people up.'");
* Q10\_What is Condition Statement?
  + JavaScript Conditional statements allow you to execute specific blocks of code based on conditions. If the condition meets then a particular block of action will be executed otherwise it will execute another block of action that satisfies that particular condition.
  + There are several methods that can be used to perform Conditional Statements in JavaScript.
    - if statement: Executes a block of code if a specified condition is true.
    - else statement: Executes a block of code if the same condition of the preceding if statement is false.
    - else if statement: Adds more conditions to the if statement, allowing for multiple alternative conditions to be tested.
    - switch statement: Evaluates an expression, then executes the case statement that matches the expression’s value.
    - ternary operator (conditional operator): Provides a concise way to write if-else statements in a single line.
* Q15\_What is the result of the expression (5 > 3 && 2 < 4)?
  + The result of the expression (5 > 3 && 2 < 4) is True.
* Q16\_What is the result of the expression (true && 1 && "hello")?
  + The result of the expression (true && 1 && "hello") is hello.
* Q17\_What is the result of the expression true && false || false && true?
  + The result of the expression true && false || false && true is false.
* Q18\_What is a Loop and Switch Case in JavaScript define that?
  + In JavaScript, a loop is a programming construct that allows you to repeatedly execute a block of code until a specified condition is met. There are several types of loops in JavaScript, including the for loop, while loop, and do-while loop. These loops provide different ways to control the flow of execution in your code.
  + Here's a brief overview of each type of loop:
    - For Loop: The for loop is commonly used when you know in advance how many times you want to iterate. It consists of three parts: initialization, condition, and iteration.
    - for (let i = 0; i < 5; i++) {
    - // Code to be repeated
    - }
    - While Loop: The while loop repeats a block of code as long as a specified condition evaluates to true.
    - let i = 0;
    - while (i < 5) {
    - // Code to be repeated
    - i++;
    - }
    - Do-While Loop: Similar to the while loop, the do-while loop also repeats a block of code based on a condition. However, the code inside the block is guaranteed to execute at least once, as the condition is checked after the block execution.
    - let i = 0;
    - do {
    - // Code to be repeated
    - i++;
    - } while (i < 5);
  + The switch statement is used for multi-way branching in JavaScript. It provides an alternative to a series of if-else statements when you want to compare a single value against multiple possible values.
    - let day = "Monday";
    - switch (day) {
    - case "Monday":
    - console.log("It's the start of the week.");
    - break;
    - case "Tuesday":
    - case "Wednesday":
    - console.log("It's the middle of the week.");
    - break;
    - case "Thursday":
    - case "Friday":
    - console.log("It's almost the weekend.");
    - break;
    - default:
    - console.log("It's the weekend!");
    - }
  + In the example above, the switch statement is used to check the value of the day variable and execute the corresponding code block based on the matched case. The break statement is used to exit the switch block once a match is found.
  + These constructs (loops and switch cases) are essential for controlling the flow of your JavaScript code and handling different scenarios efficiently.
* Q19\_What is the use of is Nan function?
  + In JavaScript, the isNaN() function is used to determine whether a value is NaN (Not-a-Number). 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 a single argument and returns a boolean value. If the argument is NaN, the function returns true; otherwise, it returns false. It's important to note that isNaN() attempts to convert its argument to a number before checking if it's NaN. If the argument cannot be converted to a number, the function also returns true.
  + Here's an example of how you might use isNaN():
    - let value1 = 42;
    - let value2 = "Hello";
    - console.log(isNaN(value1)); // false, 42 is a number
    - console.log(isNaN(value2)); // true, "Hello" cannot be converted to a number
  + In the second case, isNaN() returns true because the string "Hello" cannot be converted to a valid number. It's a way to check if a value is a valid numeric representation or not. Keep in mind that isNaN() may not behave as expected for non-numeric values, so it's often better to use more specific checks if you're dealing with specific data types.
* Q20\_What is the difference between && and || in JavaScript?
  + In JavaScript, && (logical AND) and || (logical OR) are two different logical operators used for combining conditional statements. Here's a brief explanation of each:
    - Logical AND (&&):
    - The && operator returns true if both operands are true, and false otherwise.
    - It can be used to chain multiple conditions, and the entire expression is true only if all conditions are true.
    - Example:
    - let x = 5;
    - let y = 10;
    - if (x > 0 && y > 0) {
    - console.log("Both x and y are greater than 0");
    - }
    - Logical OR (||):
    - The || operator returns true if at least one of the operands is true, and false if both operands are false.
    - It is commonly used to provide a fallback or alternative value when one condition fails.
    - Example:
    - let hour = 12;
    - if (hour < 10 || hour < 18) {
    - console.log("The office is closed.");
    - }
  + In summary:
    - && (logical AND) returns true if all conditions are true.
    - || (logical OR) returns true if at least one condition is true.
  + These operators are commonly used in conditional statements, loops, and other control flow structures to create more complex logic based on multiple conditions.
* Q21\_What is the use of Void (0)?
  + In JavaScript, void(0) is a way to evaluate an expression and return undefined. The void operator in JavaScript is used to evaluate an expression and return undefined regardless of the result of the expression.
  + Here's why void(0) is used:
    - Preventing navigation: It is often used in href attributes of anchor (<a>) tags to prevent the browser from navigating to a new page when the link is clicked. For example:
    - <a href="javascript:void(0)">Click me</a>
    - This ensures that clicking the link does not cause any navigation.
    - Returning undefined: It's also used when you want to explicitly return undefined from an expression. For example:
    - function doSomething() {
    - // Perform some actions
    - return void(0);
    - }
    - In this function, void(0) is used to explicitly return undefined without relying on an implicit return.
    - Creating self-executing anonymous functions: Sometimes void is used to create self-executing anonymous functions. For example:
    - void function() {
    - console.log("This function will execute immediately.");
    - }();
    - Here, void is used to make sure that the function is treated as an expression, and then the function is immediately invoked.
    - Preventing expressions from being evaluated: It can also be used to prevent the evaluation of an expression. For example:
    - void(expression);
    - This can be useful if you want to prevent certain expressions from being executed in specific contexts.
  + Overall, void(0) is primarily used for its side effect of returning undefined and is often used in scenarios where you need to evaluate an expression but don't want any particular result or action to occur.
* Q22\_Check Number Is Positive or Negative in JavaScript?
* Q24\_Write to check whether a number is negative, positive or zero?
  + You can check whether a number is positive or negative in JavaScript using a simple conditional statement. Here's an example using an if statement:
    - let number = 42;
    - if (number > 0) {
    - console.log("The number is positive.");
    - } else if (number < 0) {
    - console.log("The number is negative.");
    - } else {
    - console.log("The number is zero.");
    - }
  + In this example:
  + If the number is greater than 0, it is considered positive.
  + If the number is less than 0, it is considered negative.
  + If the number is neither greater nor less than 0, it is considered zero.
  + You can replace the number variable with any numeric value you want to check.
  + If you want to encapsulate this logic into a reusable function, you can create a function like this:
    - function checkNumberSign(number) {
    - if (number > 0) {
    - return "positive";
    - } else if (number < 0) {
    - return "negative";
    - } else {
    - return "zero";
    - }
    - }
    - let result = checkNumberSign(42);
    - console.log("The number is " + result + ".");
  + This function takes a number as an argument and returns a string indicating whether the number is positive, negative, or zero.
* Q23\_Find the Character Is Vowel or Not ?
  + To check if a character is a vowel or not in JavaScript, you can create a function that takes a character as an argument and then checks if it is one of the vowels (either uppercase or lowercase). Here's an example:
    - function isVowel(char) {
    - // Convert the character to lowercase to handle both uppercase and lowercase vowels
    - char = char.toLowerCase();
    - // Check if the character is a vowel
    - return ['a', 'e', 'i', 'o', 'u'].includes(char);
    - }
    - // Example usage:
    - let character = 'a';
    - if (isVowel(character)) {
    - console.log("'" + character + "' is a vowel.");
    - } else {
    - console.log("'" + character + "' is not a vowel.");
    - }
* Q25\_Write to find number is even or odd using ternary operator in JS?
  + You can use the ternary operator to check if a number is even or odd in JavaScript. Here's an example:
    - let number = 42;
    - let result = number % 2 === 0 ? "even" : "odd";
    - console.log("The number is " + result + ".");
  + In this example:
  + The expression number % 2 === 0 checks if the remainder when dividing number by 2 is equal to 0. This is a common way to determine if a number is even.
  + The ternary operator (? :) is used to assign the value "even" to the variable result if the condition is true, and "odd" if the condition is false.
  + You can replace the number variable with any numeric value you want to check. The result variable will then contain either "even" or "odd" based on the ternary operator's evaluation.
* Q26\_Write find maximum number among 3 numbers using ternary operator in JS?
  + You can find the maximum number among three numbers using the ternary operator in JavaScript. Here's an example:
    - let num1 = 10;
    - let num2 = 25;
    - let num3 = 15;
    - let maxNumber = (num1 > num2) ? ((num1 > num3) ? num1 : num3) : ((num2 > num3) ? num2 : num3);
    - console.log("The maximum number is: " + maxNumber);
  + In this example:
  + The ternary operator is nested to compare the three numbers.
  + The outer ternary operator checks whether num1 is greater than num2. If true, it compares num1 with num3; if false, it compares num2 with num3.
  + The inner ternary operator determines the maximum among the two numbers being compared in the outer condition.
  + You can replace the values of num1, num2, and num3 with any numeric values you want to compare. The maxNumber variable will then contain the maximum among the three numbers.
* Q27\_Write to find minimum number among 3 numbers using ternary operator in JS?
  + You can find the minimum number among three numbers using the ternary operator in JavaScript. Here's an example:
    - let num1 = 10;
    - let num2 = 25;
    - let num3 = 15;
    - let minNumber = (num1 < num2) ? ((num1 < num3) ? num1 : num3) : ((num2 < num3) ? num2 : num3);
    - console.log("The minimum number is: " + minNumber);
  + In this example:
  + The ternary operator is nested to compare the three numbers.
  + The outer ternary operator checks whether num1 is less than num2. If true, it compares num1 with num3; if false, it compares num2 with num3.
  + The inner ternary operator determines the minimum among the two numbers being compared in the outer condition.
  + You can replace the values of num1, num2, and num3 with any numeric values you want to compare. The minNumber variable will then contain the minimum among the three numbers.
* Q28\_Write to find the largest of three numbers in JS?
  + You can find the largest of three numbers in JavaScript using a combination of comparisons. Here's an example:
    - let num1 = 10;
    - let num2 = 25;
    - let num3 = 15;
    - let largestNumber = Math.max(num1, num2, num3);
    - console.log("The largest number is: " + largestNumber);
  + In this example, the Math.max() function is used, which returns the largest of the provided numbers. This is a straightforward and concise way to find the maximum among three numbers.
  + You can replace the values of num1, num2, and num3 with any numeric values you want to compare. The largestNumber variable will then contain the largest among the three numbers.
* Q29\_Write to show
* (i). Monday to Sunday using switch case in JS?
  + You can use a switch case statement in JavaScript to display the days of the week from Monday to Sunday:
    - let day = 1; // Assuming 1 corresponds to Monday, 2 to Tuesday, and so on
    - switch (day) {
    - case 1:
    - console.log("Monday");
    - break;
    - case 2:
    - console.log("Tuesday");
    - break;
    - case 3:
    - console.log("Wednesday");
    - break;
    - case 4:
    - console.log("Thursday");
    - break;
    - case 5:
    - console.log("Friday");
    - break;
    - case 6:
    - console.log("Saturday");
    - break;
    - case 7:
    - console.log("Sunday");
    - break;
    - default:
    - console.log("Invalid day");
    - }
  + This code will output "Monday" since we've set day to 1. You can change the value of day variable to get the respective day of the week.
* Q29\_Write to show
* (ii). Vowel or Consonant using switch case in JS?
  + you can use a switch case statement in JavaScript to determine whether a given letter is a vowel or a consonant:
    - let letter = 'a'; // Change this to the letter you want to check
    - switch (letter.toLowerCase()) {
    - case 'a':
    - case 'e':
    - case 'i':
    - case 'o':
    - case 'u':
    - console.log(letter + " is a vowel");
    - break;
    - default:
    - console.log(letter + " is a consonant");
    - }
  + In this code, if the variable letter contains a vowel ('a', 'e', 'i', 'o', or 'u'), it will print that the letter is a vowel. Otherwise, it will print that the letter is a consonant. You can change the value of the letter variable to check different letters. The .toLowerCase() method is used to ensure that both uppercase and lowercase letters are handled correctly.