Module 6) JAVASCRIPT BASIC & DOM

(Basic logic Question)

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

Ans:

Javascript is server-site and client-site scripting language.

It is a high-level, versatile programming language primarily used for front-end web development. It allows developers to add interactive elements, manipulate the DOM (Document Object Model), and create dynamic content on websites.

JavaScript is often executed in web browsers and can interact with HTML and CSS to enhance the user experience.

1.Embedding JavaScript in HTML:

You can include JavaScript code directly within HTML documents using the <script> tag.

Example:

```
<!DOCTYPE html>
<html>
 <head>
  <title>JavaScript Example</title>
 </head>
 <body>
  <script>
   // JavaScript code goes here
   console.log("Hello, World!");--
  </script>
 </body>
</html>
```

2. External JavaScript File:

You can also save your JavaScript code in a separate file (e.g., script.js) and link it to your HTML document.

HTML:

```
<!DOCTYPE html>
<html>
 <head>
  <title>JavaScript Example</title>
  <script src="script.js"></script>
 </head>
 <body>
  <!-- Content of the HTML document -->
 </body>
</html>
```

3. Basic JavaScript Syntax:

Variables, data types, operators, control structures (if statements, loops), functions, and objects are fundamental elements of JavaScript.

Example:

```
// Variables and data types
let greeting = "Hello";
let number = 42;

// Functions
function sayHello(name) {
 console.log(`${greeting}, ${name}!`);
}

// Function invocation
sayHello("Jay");

// Objects
```

```
let person = {
  firstName: "Jay",
  lastName: "patel",
  age: 24,
};

console.log(person.firstName); // Outputs: John
```

Q.2 How many type of Variable in JavaScript?

Ans:

There are three type of variable in js:-

1) var
2) let
3) const

1. var:

- var was the original keyword for declaring variables in JavaScript.
- Variables declared with var are function-scoped, meaning they are only visible within the function where they are declared.
- If a variable is declared outside any function, it becomes a global variable.

2.let:

- Introduced in ECMAScript 6 (ES6), let allows you to declare block-scoped variables.
- Variables declared with let are only accessible within the block, statement, or expression where they are defined.

3.Const

- Also introduced in ECMAScript 6, const is used to declare constants.
- Variables declared with const cannot be reassigned after their initial assignment.
- Like let, const is block-scoped.

Q.4 Write a mul Function Which will Work Properly When invoked With Following Syntax.

In javascript, a function can be a return value from another function. mul is returning a function which in case returns an array. The first element in the array is x*y and the second element is a function.

```
function mul(...args) {
   return args.reduce((acc, val) => acc * val, 1);
}

let result = mul(2, 3, 4);
console.log(result);  // Output: 24
```

Q.5 What the difference between undefined and undeclare in JavaScript?

S. No.	undeclared	undefined
1.	These are the variables that do not exist in the memory heap.	These variables are the ones that do exist in memory but nothing is being assigned to them explicitly by the programmer.
2.	The variables are considered to be undeclared because of programmer does not write them with var, let, or const.	

The variables are considered to be undefined because it is assigned by javascript to them.

3. If we try to access them in the code execution phase then javascript will throw a Reference error If we try to access these variables we'll get the undefined as value.

Q.6 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:

```
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 best way to find yourself is to lose yourself in the service of others. - Mother Teresa");
```

Q.7 Check if typeof '10' is exactly equal to 10. If not make it exactly equal?

The typeof operator in JavaScript is used to determine the data type of a variable or an expression.

```
let stringNumber = '10';

// Check if typeof '10' is exactly equal to 10
if (typeof stringNumber === 'number') {
   console.log('The variable is already a number.');
} else {
   // Convert the string to a number
   stringNumber = +stringNumber;

   console.log('After conversion:', stringNumber);
```

```
}

// Now you can safely compare with the number 10
if (stringNumber === 10) {
   console.log('The variable is exactly equal to 10.');
} else {
   console.log('The variable is not equal to 10.');
}
```

Q.8 Write a JavaScript Program to find the area of a triangle?

```
// Function to calculate the area of a triangle
function Area(base, height) {
   return 0.5 * base * height;
}

// Example usage
let Base = 5;
let Height = 8;

let area = Area(Base, Height);
console.log("The area of the triangle is:", area);
```

Q.9 Write a JavaScript program to calculate days left until next Christmas?

```
// Function to calculate days left until next Christmas
function daysUntilChristmas() {
   // Get the current date
   let currentDate = new Date();

   // Get the current year
   let currentYear = currentDate.getFullYear();
```

```
    // Set the next Christmas date (assuming Christmas is always
on December 25th)
    let nextChristmas = new Date(currentYear, 11, 25);

    // If Christmas has already occurred this year, set it for next
year
    if (currentDate > nextChristmas) {
        nextChristmas.setFullYear(currentYear + 1);
    }

    // Calculate the difference in milliseconds between the
current date and next Christmas
    let timeDifference = nextChristmas - currentDate;

    // Calculate the number of days left
    let daysLeft = Math.ceil(timeDifference / (1000 * 60 * 60 *
24));

    return daysLeft;
}

// Example usage
let daysLeftUntilChristmas = daysUntilChristmas();
console.log("Days left until next Christmas:",
daysLeftUntilChristmas);
```

Q.10 What is Condition Statement?

A condition statement, in the context of programming, is a construct that allows you to make decisions in your code based on whether a specified condition evaluates to true or false. These statements enable you to control the flow of your program by executing different blocks of code depending on whether certain conditions are met.

Q.11 Find circumference of Rectangle formula : C = 4 * a ?

```
// Function to calculate the circumference of a rectangle
function RectangleCir(length, width) {
   return 2 * (length + width);
}

// Function to calculate the circumference of a rectangle when
all the side is equal
function rectangleCir(side) {
   return 4 * side;
}

// Example usage for a rectangle
let rectangleLen = 5;
let rectangleWid = 8;
let rectangleCir= RectangleCir(rectangleLen, rectangleWid);
console.log("Circumference of the rectangle:", rectangleCir);

// Example usage for a rectangle when all the side are equal
let rectangleSide = 4;
let rectangleCir= rectangleCir(rectangleSide);
console.log("Circumference of the rec:",
rectangleCircumference);
```

Q.12 WAP to convert years into days and days into years?

```
function yearsToDays(years) {
   var days = years * 365;
   return days;
}

function daysToYears(days) {
   var years = days / 365;
```

```
    return years;
}

function main() {
    var choice = parseInt(prompt("Enter 1 to convert years to
days or 2 to convert days to years: "));

    if (choice === 1) {
        var years = parseFloat(prompt("Enter the number of years:
"));
        var days = yearsToDays(years);
        console.log(`${years} years is equal to ${days} days.`);
    } else if (choice === 2) {
        var days = parseFloat(prompt("Enter the number of days:
"));
        var years = daysToYears(days);
        console.log(`${days} days is equal to ${years} years.`);
    } else {
        console.log("Invalid choice. Please enter 1 or 2.");
    }
}

main();
```

Q.13 Convert temperature Fahrenheit to Celsius? (Conditional logic Question)

```
function fahrenheitToCelsius(fahrenheit) {
    var celsius;

    // Formula to convert Fahrenheit to Celsius: (F - 32) * 5/9
    if (typeof fahrenheit === 'number') {
        celsius = (fahrenheit - 32) * 5 / 9;
        return celsius;
    } else {
```

```
        return "Please enter a valid numerical temperature.";
    }
}

var fahrenheitTemperature = parseFloat(prompt("Enter
temperature in Fahrenheit: "));
var celsiusTemperature =
fahrenheitToCelsius(fahrenheitTemperature);

if (typeof celsiusTemperature === 'number') {
    console.log(`${fahrenheitTemperature}°F is equal to
${celsiusTemperature.toFixed(2)}°C.`);
} else {
    console.log(celsiusTemperature);
}
```

Q.14 Write a JavaScript exercise to get the extension of a filename.?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Filename Extension Exercise</title>
    <script>
        function getFileExtension(filename) {
            // Split the filename into an array using dot as the
delimiter
            const parts = filename.split('.');

            // Check if there's more than one part (at least one dot
in the filename)
            if (parts.length > 1) {
```

```
            // Return the last part as the extension
            return parts[parts.length - 1];
        } else {
            // No extension found
            return "No extension found";
        }
    }

    // Example usage
    const filename1 = "example.txt";
    const filename2 = "document.pdf";
    const filename3 = "script_without_extension";

    console.log(getFileExtension(filename1)); // Output: txt
    console.log(getFileExtension(filename2)); // Output: pdf
    console.log(getFileExtension(filename3)); // Output: No
extension found
  </script>
</head>
<body>
  <!-- Exercise content goes here -->
</body>
</html>
```

Q.15 What is the result of the expression (5 > 3 && 2 < 4)?

The expression (5 > 3 && 2 < 4) involves the logical AND operator (&&).

In these result will be true because both the expression are true.

Q.16 What is the result of the expression (true && 1 && "hello")?

In most programming languages, the && (logical AND) operator evaluates expressions from left to right and returns the first false value it encounters, or the last value if all expressions are true.
In the expression (true && 1 && "hello"):

1. true is a boolean value and is considered truthy.
2. 1 is a truthy value in many programming languages.
3. "hello" is a non-empty string, which is also considered truthy.

The logical AND operator (&&) returns the first falsy value it encounters or the last value if all are truthy. Since all the expressions in this case are truthy, the result of the expression will be the last truthy value, which is "hello"

Q.17 What is the result of the expression true && false || false && true?

The expression `true && false || false && true` involves logical AND (`&&`) and logical OR (`||`) operators.

In most programming languages, logical AND has higher precedence than logical OR, so the expression is evaluated from left to right.

Let's break down the expression:

1. `true && false`: This evaluates to `false` because the logical AND operator returns `false` if any of its operands is `false`.

2. `false || false`: This evaluates to `false` because the logical OR operator returns `true` only if at least one of its operands is `true`.

3. The final result is `false` because the entire expression is `false` due to the logical OR operation.

Therefore, the result of the expression `true && false || false && true` is `false`.

Q.18 What is a Loop and Switch Case in JavaScript define that ?

Loop:-
In JavaScript, a loop is a control structure that allows you to repeatedly execute a block of code as long as a specified condition is true.

Switch Case:-
In JavaScript, the switch statement is used to perform different actions based on different conditions. It evaluates an expression, and depending on the value of the expression, it executes the corresponding block of code.

Q.19 What is the use of is Nan function?

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.

The isNaN function returns a Boolean value indicating whether the provided value is NaN. It can be used with various types of values, not just numbers.

Q.20 What is the difference between && and || in JavaScript?

The main difference between the && (logical AND) and || (logical OR) operators in JavaScript lies in their behavior and the conditions under which they evaluate to true or false.

&& (Logical AND):

- Requires both operands to be true.
- Short-circuits if the first operand is false.

|| (Logical OR):

- Requires at least one operand to be true.

Short-circuits if the first operand is true.

Q.21 What is the use of Void (0)?

The void keyword in JavaScript is used to evaluate an expression and then return undefined. The most common use of void is in combination with the value 0 to create a self-executing anonymous function or to prevent a browser from navigating to a new page when clicking on a link.
Here's a common use case:
javascriptCopy code:-

```
<a href="javascript:void(0);" onclick="myFunction()">Click me</a>
```

In this example, clicking the link triggers the myFunction() JavaScript function. The void(0) part is used in the href attribute to ensure that clicking the link doesn't perform any actual navigation. It evaluates to undefined and prevents the browser from reloading the page or navigating elsewhere.

You might also see void 0 used instead of void(0). Both void(0) and void 0 achieve the same result of producing undefined. The choice between them is largely a matter of coding style

Q.22 Check Number Is Positive or Negative in JavaScript?

```
function checkNumber(number) {
 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.");
  }
}

// Example usage:
checkNumber(5);    // Output: The number is positive.
checkNumber(-3);   // Output: The number is negative.
checkNumber(0);    // Output: The number is zero.
```

Q.23 Find the Character Is Vowel or Not ?

```
function isVowel(char) {
    return ['a', 'e', 'i', 'o', 'u'].indexOf(char.toLowerCase()) !== -1;
}

// Example usage:
var character = prompt("Enter a character: ");

if (character.length === 1 && character.match(/[a-zA-Z]/)) {
    if (isVowel(character)) {
        console.log(character + " is a vowel.");
    } else {
        console.log(character + " is not a vowel.");
    }
} else {
    console.log("Please enter a single alphabetical character.");
}
```

Q.24 Write to check whether a number is negative, positive or zero?

```
function checkNumber(num) {
    if (num > 0) {
```

```
        return "Positive";
    } else if (num < 0) {
        return "Negative";
    } else {
        return "Zero";
    }
}

// Example usage:
var userInput = prompt("Enter a number: ");
var number = parseFloat(userInput);

if (!isNaN(number)) {
    var result = checkNumber(number);
    console.log(`The number is ${result}.`);
} else {
    console.log("Invalid input. Please enter a valid number.");
}
```

Q.25 Write to find number is even or odd using ternary operator in JS?

```
var userInput = prompt("Enter a number: ");
var number = parseInt(userInput);

var result = (number % 2 === 0) ? "Even" : "Odd";

console.log(`The number is ${result}.`);
```

Q.26 Write find maximum number among 3 numbers using ternary operator in JS?

```
        var num1 = parseFloat(prompt("Enter the first number:
"));
```

```
//parseFloat is used to convert the input string into a floating-point number.
var num2 = parseFloat(prompt("Enter the second number: "));
var num3 = parseFloat(prompt("Enter the third number: "));

var maxNumber = (num1 >= num2 && num1 >= num3) ? num1 :
            (num2 >= num1 && num2 >= num3) ? num2 : num3;

console.log(`The maximum number is: ${maxNumber}`);
```

Q.27 Write to find minimum number among 3 numbers using ternary operator in JS?

```
var num1 = parseFloat(prompt("Enter the first number: "));
var num2 = parseFloat(prompt("Enter the second number: "));
var num3 = parseFloat(prompt("Enter the third number: "));

var minNumber = (num1 <= num2 && num1 <= num3) ? num1 :
            (num2 <= num1 && num2 <= num3) ? num2 : num3;

console.log(`The minumum number is: ${minNumber}`);
```

Q.28 Write to find the largest of three numbers in JS?

```
var num1 = parseFloat(prompt("Enter the first number: "));
var num2 = parseFloat(prompt("Enter the second number: "));
var num3 = parseFloat(prompt("Enter the third number: "));

if (!isNaN(num1) && !isNaN(num2) && !isNaN(num3)) {
   var largestNumber;

   if (num1 >= num2 && num1 >= num3) {
      largestNumber = num1;
   } else if (num2 >= num1 && num2 >= num3) {
      largestNumber = num2;
```

```
    } else {
        largestNumber = num3;
    }

    console.log(`The largest number is: ${largestNumber}`);
} else {
    console.log("Invalid input. Please enter valid numbers.");
}
```

Q.29 Write to show

i. Monday to Sunday using switch case in JS?

```
var dayNumber = parseInt(prompt("Enter a number (1-7)
representing a day of the week:"));

switch (dayNumber) {
    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 input. Please enter a number between
1 and 7.");
}
```

ii. Vowel or Consonant using switch case in JS?

```
var character = prompt("Enter a single alphabet character:");

switch (character.toLowerCase()) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
        console.log("Vowel");
        break;
    default:
        console.log("Consonant");
}
```

(Conditional looping logic Question)

Q.30 What are the looping structures in JavaScript? Any one Example?

In JavaScript, there are several looping structures that allow you to repeatedly execute a block of code. The most common ones are:

1. for loop
2. while loop
3. do while loop

4. for in loop
5. for of loop

for loop:-
printing 1 to 5 number:

```
for (let i = 0; i < 5; i++) {
   console.log(i);
}
```

Q.31 Write a print 972 to 897 using for loop in JS?

```
for (let i = 972; i >= 897; i--) {
   console.log(i);
}
```

Q.32 Write to print factorial of given number?

```
function factorial(n) {
   if (n === 0 || n === 1) {
      return 1;
   } else {
      let result = 1;
      for (let i = 2; i <= n; i++) {
         result *= i;
      }
      return result;
   }
}

// Example: Calculate factorial of 5
let number = 5;
let result = factorial(number);
console.log(`The factorial of ${number} is: ${result}`);
```

Q.33 Write to print Fibonacci series up to given numbers?

```
function fibonacciSeries(limit) {
    let fibArray = [0, 1];

    for (let i = 2; fibArray[i - 1] + fibArray[i - 2] <= limit; i++) {
        fibArray[i] = fibArray[i - 1] + fibArray[i - 2];
    }

    return fibArray;
}

// Example: Print Fibonacci series up to 50
let limit = 50;
let series = fibonacciSeries(limit);
console.log(`Fibonacci series up to ${limit}: ${series.join(', ')}`);
```

Q.34 Write to print number in reverse order e.g.: number = 64728 ---> reverse =82746 in JS?

```
function reverseNumber(number) {
    // Convert the number to a string
    let numberString = number.toString();

    // Split the string into an array of characters, reverse it, and join back into a string
    let reversedString = numberString.split('').reverse().join('');

    // Convert the reversed string back to a number
    let reversedNumber = parseInt(reversedString);
```

```
    return reversedNumber;
}

// Example: Reverse the number 64728
let originalNumber = 64728;
let reversedNumber = reverseNumber(originalNumber);
console.log(`Original number: ${originalNumber}`);
console.log(`Reversed number: ${reversedNumber}`);
```

Q.35 Write a program make a summation of given number (E.g., 1523
in JS?

Ans: - 11)

```
function calculateDigitSum(number) {
    // Convert the number to a string
    let numberString = number.toString();

    // Initialize the sum
    let sum = 0;

    // Iterate through each digit and add it to the sum
    for (let i = 0; i < numberString.length; i++) {
        sum += parseInt(numberString[i]);
    }

    return sum;
}

// Example: Calculate the digit sum of 1523
let givenNumber = 1523;
let digitSum = calculateDigitSum(givenNumber);
console.log(`Summation of ${givenNumber}: ${digitSum}`);
```

Q.36 Write a program you have to make a summation of first and last Digit. (E.g., 1234 Ans: - 5) in JS?

```
function calculateFirstAndLastDigitSum(number) {
   // Convert the number to a string
   let numberString = number.toString();

   // Extract the first and last digits
   let firstDigit = parseInt(numberString[0]);
   let lastDigit = parseInt(numberString[numberString.length - 1]);

   // Calculate the sum of the first and last digits
   let sum = firstDigit + lastDigit;

   return sum;
}

// Example: Calculate the sum of the first and last digits of 1234
let givenNumber = 1234;
let digitSum = calculateFirstAndLastDigitSum(givenNumber);
console.log(`Sum of the first and last digits of ${givenNumber}: ${digitSum}`);
```

Q.37 Use console.log() and escape characters to print the following pattern in JS?

```
 1 1 1 1 1
 2 1 2 4 8
3 1 3 9 27
4 1 4 16 64
 5 1 5 25 125

// Define the number of rows for the pattern
const numRows = 5;
```

```
// Outer loop for rows
for (let i = 1; i <= numRows; i++) {
   let rowOutput = '';

   // Inner loop for columns
   for (let j = 1; j <= 5; j++) {
      if (j === 1) {
         // Print the first column with the row number
         rowOutput += `${i} `;
      } else {
         // Print the subsequent columns with calculated values
         rowOutput += `${Math.pow(i, j)} `;
      }
   }
   // Print the entire row
   console.log(rowOutput);
}
```

Q.38 Use pattern in console.log in JS?

1)

```
1
1 0
1 0 1
1 0 1 0
1 0 1 0 1
```

```
for (let i = 1; i <= 5; i++) {
   for (let j = 1; j <= i; j++) {
      if (j % 2 === 0) {
         console.log('0');
      } else {
         console.log('1');
      }
```

```
    }
    console.log('\n');
}
```

2)

```
A
B C
D E F
G H I J
K L M N O
let currentChar = 65; // ASCII code for 'A'

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

    for (let j = 1; j <= i; j++) {
        row += String.fromCharCode(currentChar) + ' ';
        currentChar++;
    }

    console.log(row);
}
```

3)

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15

let counter = 1;
```

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

   for (let j = 1; j <= i; j++) {
      row += counter + ' ';
      counter++;
   }

   console.log(row);
}
```

4)

```
*
* *
* * *
* * * *
* * * * *
```

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

   for (let j = 1; j <= i; j++) {
      row += '* ';
   }

   console.log(row);
}
```

Q.39 Accept 3 numbers from user using while loop and check each numbers palindrome?

```
function isPalindrome(number) {
   const originalNumber = number;
   let reversedNumber = 0;
```

```
    while (number > 0) {
        const digit = number % 10;
        reversedNumber = reversedNumber * 10 + digit;
        number = Math.floor(number / 10);
    }

    return originalNumber === reversedNumber;
}

let count = 1;
while (count <= 3) {
    const userInput = parseInt(prompt(`Enter number
${count}:`));

    if (!isNaN(userInput)) {
        if (isPalindrome(userInput)) {
            console.log(`${userInput} is a palindrome.`);
        } else {
            console.log(`${userInput} is not a palindrome.`);
        }
        count++;
    } else {
        alert('Invalid input. Please enter a valid number.');
    }
}
```

(Array and object Question)

Q.40 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 ?

```
function getCurrentDayAndTime() {
    // Array of days
```

```
    const days = ["Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"];

    // Get current date and time
    const now = new Date();
    const day = days[now.getDay()]; // Get day of the week
    let hours = now.getHours(); // Get hours
    const ampm = hours >= 12 ? 'PM' : 'AM'; // Check if it's AM or
PM
    hours = hours % 12 || 12; // Convert hours to 12-hour format
    const minutes = now.getMinutes(); // Get minutes
    const seconds = now.getSeconds(); // Get seconds
    const milliseconds = now.getMilliseconds(); // Get
milliseconds

    // Format time with leading zeros if needed
    const formattedTime = `${hours < 10 ? '0' : ''}${hours} :
${minutes < 10 ? '0' : ''}${minutes} : ${seconds < 10 ? '0' :
''}${seconds} ${milliseconds} ${ampm}`;

    // Display the result
    console.log(`Today is ${day}. Current Time is
${formattedTime}`);
}

// Call the function to display the current day and time
getCurrentDayAndTime();
```

Q.41 Write a JavaScript program to get the current date?

```
function getCurrentDate() {
    // Create a new Date object
    const currentDate = new Date();

    // Get the year, month, and day from the Date object
```

```
    const year = currentDate.getFullYear();
    // JavaScript months are 0-based, so add 1 to get the correct
month
    const month = currentDate.getMonth() + 1;
    const day = currentDate.getDate();

    // Format the date as needed (adding leading zeros if
necessary)
    const formattedDate = `${year}-${month < 10 ? '0' :
''}${month}-${day < 10 ? '0' : ''}${day}`;

    // Return the formatted date
    return formattedDate;
}

// Call the function to get the current date and log it to the
console
console.log("Current Date:", getCurrentDate());
```

Q.42 Write a JavaScript program to compare two objects?

```
function compareObjects(obj1, obj2) {
    // Get the keys of both objects
    const keys1 = Object.keys(obj1);
    const keys2 = Object.keys(obj2);

    // Check if the number of keys is the same
    if (keys1.length !== keys2.length) {
        return false; // Objects have different number of keys
    }

    // Iterate through the keys of obj1
    for (let key of keys1) {
        // Check if the key exists in obj2
        if (!obj2.hasOwnProperty(key)) {
            return false; // Key doesn't exist in obj2
```

```
        }

        // Check if the values of the corresponding keys are equal
        if (obj1[key] !== obj2[key]) {
            return false; // Values are not equal
        }
    }

    // If all keys and values are equal, return true
    return true;
}

// Example usage:
const obj1 = {a: 1, b: 2, c: 3};
const obj2 = {a: 1, b: 2, c: 3};

console.log(compareObjects(obj1, obj2));
```

Q.43 Write a JavaScript program to convert an array of objects into CSV string?

```
function arrayofObjectsToCSV(data) {
    // Check if the array is empty
    if (data.length === 0) {
        return ''; // Return an empty string if the array is empty
    }

    // Get the headers from the first object's keys
    const headers = Object.keys(data[0]);

    // Create CSV header by joining headers with commas
    const csvHeader = headers.join(',');

    // Create CSV rows by iterating over each object
    const csvRows = data.map(obj => {
```

```
        // Extract values for each header from the object
        const values = headers.map(header => {
            // Handle cases where the value might need to be
quoted
            const value = obj[header];
            return typeof value === 'string' && value.includes(',') ?
`"${value}"` : value;
        });
        // Join values with commas
        return values.join(',');
    });

    // Join CSV header and rows with newline characters
    return `${csvHeader}\n${csvRows.join('\n')}`;
}

// Example usage:
const data = [
    { name: 'John', age: 30, city: 'New York' },
    { name: 'Alice', age: 25, city: 'Los Angeles' },
    { name: 'Bob', age: 35, city: 'Chicago' }
];

const csvString = arrayofObjectsToCSV(data);
console.log(csvString);
```

Q.44 Write a JavaScript program to capitalize first letter of a string?

```
function capitalizeFirstLetter(str) {
    // Check if the string is empty
    if (str === '') {
        return ''; // Return an empty string if the input is empty
    }
```

```
    // Capitalize the first letter and concatenate it with the rest
of the string
    return str.charAt(0).toUpperCase() + str.slice(1);
}

// Example usage:
const inputString = 'hello world';
const capitalizedString = capitalizeFirstLetter(inputString);
console.log(capitalizedString); // Output: "Hello world"
```

Q. 45 Write a JavaScript program to determine if a variable is array?

```
function isArray(variable) {
    // Check if the variable is an array using Array.isArray()
method
    return Array.isArray(variable);
}

// Example usage:
const arr = [1, 2, 3];
const notArr = 'Hello';

console.log(isArray(arr));    // Output: true
console.log(isArray(notArr)); // Output: false
```

Q.46 Write a JavaScript program to clone an array?

You can clone an array in JavaScript using various methods such as slice(), concat(), or the spread operator (...).

1.Using slice method

```
function cloneArray(array) {
    // Use the slice() method with no arguments to create a
shallow copy of the array
    return array.slice();
```

```
}

// Example usage:
const originalArray = [1, 2, 3];
const clonedArray = cloneArray(originalArray);

console.log(clonedArray); // Output: [1, 2, 3]
```

2.Using the concat operator (...):

```
function cloneArray(array) {
   // Use the concat() method with an empty array to create a
shallow copy of the array
   return [].concat(array);
}

// Example usage:
const originalArray = [1, 2, 3];
const clonedArray = cloneArray(originalArray);

console.log(clonedArray); // Output: [1, 2, 3]
```

3. Using the spread operator (...):

```
function cloneArray(array) {
   // Use the spread operator (...) to create a shallow copy of
the array
   return [...array];
}

// Example usage:
const originalArray = [1, 2, 3];
const clonedArray = cloneArray(originalArray);

console.log(clonedArray); // Output: [1, 2, 3]
```

Q.47 What is the drawback of declaring methods directly in JavaScript objects?

One drawback of declaring methods directly in JavaScript objects is that it can lead to inefficient memory usage when creating multiple instances of objects.

When you declare a method directly within an object literal or constructor function, that method is duplicated in every instance of the object. This means that each object instance holds its own copy of the method in memory. If the method is large or complex, or if you have a large number of instances of the object, this duplication can result in excessive memory usage.

Q.48 Print the length of the string on the browser console using console.log()?

```
//code
const str = "Hello, World!";
console.log(str.length);
```

Q.49 Change all the string characters to capital letters using toUpperCase() method?

```
const str = "hello, world!";
const capitalizedString = str.toUpperCase();
console.log(capitalizedString);

//output:-HELLO, WORLD!
```

Q.52 Use indexOf to determine the position of the first occurrence of a in 30 Days Of JavaScript?

```
//code
const str = "30 Days Of JavaScript";
```

```
const position = str.indexOf("a");

console.log("Position of the first occurrence of 'a':", position);
```

Q.53 Use lastIndexOf to determine the position of the last occurrence of a in 30 Days Of JavaScript?

```
//code
const str = "30 Days Of JavaScript";
const position = str.lastIndexOf("a");

console.log("Position of the last occurrence of 'a':", position);
```

Q.54 Form Validtion in JS?

Form validation in JavaScript is the process of ensuring that user input in HTML forms meets certain criteria before it's submitted to a server. This is typically done to prevent the submission of invalid data, improve user experience, and ensure data integrity.

Here's a basic outline of how form validation can be implemented in JavaScript:

1. HTML Form: Create an HTML form with input fields that users will fill out.
2. JavaScript Validation Function: Write a JavaScript function that will be triggered when the form is submitted. This function should perform validation checks on the form fields.
3. Validation Rules: Define validation rules for each form field. These rules can include checks for required fields, minimum and maximum lengths, specific formats (like email or phone number), and more.

4. Error Handling: Display error messages next to the form fields if validation fails. Error messages should inform users about what went wrong and how to correct it.
5. Prevent Default Submission: If validation fails, prevent the form from being submitted to the server. This allows users to correct their mistakes before submitting again.

Q.55 Form in Email, number, Password, Validation?

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Document</title>
</head>
<body>
<form id="myForm">
   <label for="email">Email:</label>
   <input type="email" id="email" name="email" required>
   <span id="emailError" style="color: red;"></span><br>

   <label for="phoneNumber">Phone Number:</label>
   <input type="tel" id="phoneNumber" name="phoneNumber"
required>
   <span id="phoneError" style="color: red;"></span><br>

   <label for="password">Password:</label>
   <input type="password" id="password" name="password"
required>
   <span id="passwordError" style="color: red;"></span><br>

   <button type="submit">Submit</button>
</form>
```

```
<script>
document.getElementById("myForm").addEventListener("subm
it", function(event) {
   // Prevent default form submission
   event.preventDefault();

   // Validate email
   const emailInput = document.getElementById("email");
   const emailError = document.getElementById("emailError");
   if (!emailInput.value || !isValidEmail(emailInput.value)) {
      emailError.textContent = "Please enter a valid email
address";
      return;
   } else {
      emailError.textContent = "";
   }

   // Validate phone number
   const phoneInput =
document.getElementById("phoneNumber");
   const phoneError = document.getElementById("phoneError");
   if (!phoneInput.value
|| !isValidPhoneNumber(phoneInput.value)) {
      phoneError.textContent = "Please enter a valid phone
number";
      return;
   } else {
      phoneError.textContent = "";
   }

   // Validate password
   const passwordInput =
document.getElementById("password");
   const passwordError =
document.getElementById("passwordError");
```

```
    if (!passwordInput.value || passwordInput.value.length < 8) {
        passwordError.textContent = "Password must be at least 8
characters long";
        return;
    } else {
        passwordError.textContent = "";
    }

    // If all validation passes, submit the form
    this.submit();
});

// Function to validate email address
function isValidEmail(email) {
    // Regular expression for validating email format
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    return emailRegex.test(email);
}

// Function to validate phone number
function isValidPhoneNumber(phoneNumber) {
    // Regular expression for validating phone number format
    const phoneRegex = /^\d{10}$/;
    return phoneRegex.test(phoneNumber);
}
</script>

</body>
</html>
```

Q.56 Dynamic Form Validation in JS?

```
//code
<!DOCTYPE html>
<html lang="en">
```

```
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Document</title>
</head>
<body>

<form id="myForm">
   <label for="email">Email:</label>
   <input type="email" id="email" name="email">
   <span id="emailError" style="color: red;"></span><br>

   <label for="password">Password:</label>
   <input type="password" id="password" name="password">
   <span id="passwordError" style="color: red;"></span><br>

   <button type="submit">Submit</button>
</form>
<script>
// Get form elements
const emailInput = document.getElementById("email");
const passwordInput = document.getElementById("password");
const emailError = document.getElementById("emailError");
const passwordError =
document.getElementById("passwordError");

// Function to validate email address
function validateEmail(email) {
   // Regular expression for validating email format
   const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
   return emailRegex.test(email);
}

// Function to validate password
```

```
function validatePassword(password) {
   return password.length >= 8;
}

// Event listener for email input
emailInput.addEventListener("input", function() {
   if (!validateEmail(emailInput.value)) {
      emailError.textContent = "Please enter a valid email
address";
   } else {
      emailError.textContent = "";
   }
});

// Event listener for password input
passwordInput.addEventListener("input", function() {
   if (!validatePassword(passwordInput.value)) {
      passwordError.textContent = "Password must be at least 8
characters long";
   } else {
      passwordError.textContent = "";
   }
});

// Event listener for form submission
document.getElementById("myForm").addEventListener("subm
it", function(event) {
   if (!validateEmail(emailInput.value)) {
      emailError.textContent = "Please enter a valid email
address";
      event.preventDefault(); // Prevent form submission
   }
   if (!validatePassword(passwordInput.value)) {
      passwordError.textContent = "Password must be at least 8
characters long";
```

```
        event.preventDefault(); // Prevent form submission
    }
});
</script>
</body>
</html>
```

Q.57 how many type of JS Event? How to use it?

JavaScript events can be categorized into several types based on their triggers or sources. Here are some common types of JavaScript events:

1. Mouse Events: These events are triggered by mouse actions such as clicks, movements, and hovering.
   - click: Triggered when a mouse button is clicked.
   - dblclick: Triggered when a mouse button is double-clicked.
   - mouseover: Triggered when the mouse pointer moves over an element.
   - mouseout: Triggered when the mouse pointer moves out of an element.
2. Keyboard Events: These events are triggered by keyboard actions such as key presses and releases.
   - keydown: Triggered when a key is pressed down.
   - keyup: Triggered when a key is released.
   - keypress: Triggered when a key is pressed down and then released.
3. Form Events: These events are triggered by form-related actions such as submitting, resetting, and changing input fields.
   - submit: Triggered when a form is submitted.
   - reset: Triggered when a form is reset.
   - change: Triggered when the value of an input element changes (e.g., text input, checkbox, select).
4. Document/Window Events: These events are related to the document or window itself.

- load: Triggered when the document or window has finished loading.
- resize: Triggered when the window is resized.
- scroll: Triggered when the window is scrolled.

5. Focus Events: These events are triggered when an element gains or loses focus.
- focus: Triggered when an element gains focus.
- blur: Triggered when an element loses focus.

6. Other Events: There are many other events available in JavaScript, such as DOMContentLoaded, error, DOMContentLoaded, contextmenu, drag, drop, etc.

To use events in JavaScript, you can attach event handlers to HTML elements using the addEventListener() method or by using event attributes directly in the HTML markup.

Q.59 What is Bom vs Dom in JS?

In JavaScript, both BOM (Browser Object Model) and DOM (Document Object Model) are important concepts, but they serve different purposes.

1. DOM (Document Object Model):
- The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the structure of the document as a hierarchical tree of nodes, where each node represents a part of the document (such as elements, attributes, and text).
- The DOM provides methods and properties for interacting with and manipulating the structure and content of web pages. You can use DOM methods to access, create, modify, and delete elements and attributes in an HTML document dynamically.
- Example DOM operations include selecting elements by their ID, class, or tag name, changing element styles, adding event listeners, creating new elements, and more.

2. BOM (Browser Object Model):

- The Browser Object Model (BOM) represents the browser itself as an object, providing access to browser-specific features and functionalities.
- Unlike the DOM, which deals with the document structure, the BOM deals with browser windows and their properties, such as size, location, history, and the browser's navigator object.
- Common BOM objects include window, document, location, history, navigator, screen, localStorage, sessionStorage, etc.
- The BOM is not standardized like the DOM, and its features may vary between different browsers. However, it provides essential functionalities for web development, such as managing browser history, controlling window behavior, and handling user interactions.

Q.60 Array vs object defences in JS?

In JavaScript, arrays and objects are both used to store collections of data, but they have different characteristics and use cases. Here's a comparison between arrays and objects in terms of their definitions and typical uses:

1. Array:

- An array is a special type of object in JavaScript that stores data as a list of elements, each identified by an index.
- Arrays are ordered collections, meaning the order of elements in an array is preserved.
- Arrays are best suited for storing lists of items where the order is important, such as a list of numbers, strings, or objects.
- Arrays provide built-in methods for manipulating and iterating over elements, such as push(), pop(), shift(), unshift(), forEach(), map(), filter(), etc.
- Arrays are typically used when you need to store and access a collection of similar or related items, and when you

need to perform operations on the entire collection or iterate through its elements.

2. Object:

- An object is a collection of key-value pairs, where each key is a unique string (or symbol) that identifies a property, and each value can be any JavaScript data type, including other objects, arrays, functions, etc.
- Objects are unordered collections, meaning there is no guaranteed order of properties in an object.
- Objects are best suited for representing complex data structures, modeling real-world entities, or organizing data with named properties.
- Objects provide a flexible and dynamic way to store and access data, as properties can be added, updated, or removed dynamically.
- Objects are typically used when you need to represent entities with multiple properties or attributes, such as a user object with properties like name, age, email, etc., or when you need to organize data in a hierarchical or structured way.

In summary, arrays are best suited for storing ordered collections of similar items, while objects are best suited for representing entities with multiple properties or for organizing data with named attributes. Both arrays and objects are fundamental data structures in JavaScript and are widely used in web development for various purposes. Choosing between them depends on the specific requirements and structure of your data.

Q.61 Split the string into an array using split() Method?

```
//code
const str = "Hello, World!";
const array = str.split(','); // Split the string at commas

console.log(array); // Output: ["Hello", " World!"]
```

Q.62 Check if the string contains a word Script using includes() method?

```
//code
const str = "JavaScript is a scripting language.";
const wordToFind = "Script";

if (str.includes(wordToFind)) {
   console.log(`The string contains the word "${wordToFind}".`);
} else {
   console.log(`The string does not contain the word "${wordToFind}".`);
}
```

Q.63 Change all the string characters to lowercase letters using toLowerCase() Method.

```
const str = "Hello, World!";
const lowercaseString = str.toLowerCase();

console.log(lowercaseString); // Output: "hello, world!"
```

Q.64 What is Character at index 15 in '30 Days of JavaScript' string? Use charAt() method.

```
//code

const str = '30 Days of JavaScript';
const characterAtIndex15 = str.charAt(15);

console.log("Character at index 15:", characterAtIndex15); // Output: "S"
```

Q.65 copy to one string to another string in JS?

In JavaScript, you can copy the content of one string to another string using simple assignment or by using methods like slice(), substring(), or concatenation.