1) What is JavaScript. How to use it?

Ans :

=> JavaScript is a high-level, interpreted programming language that is commonly used to make web pages interactive and dynamic.

Originally created for use in web browsers, JavaScript has evolved to become a versatile language used for both front-end and back-end development, as well as in various other domains.

Client-side scripting language: Primarily executed in a web browser, making web pages interactive and responsive to user actions.

2) How many type of Variable in JavaScript?

Ans :

=> There are primarily three ways to declare variables in JavaScript.

1. let (introduced in ES6)

Scope: Block-scoped (limited to the code block where it's declared, typically curly braces {}).

Re-declaration: Not allowed within the same block.

2. const (introduced in ES6)

Scope: Block-scoped.

Re-declaration: Not allowed.

3. var (traditional, but less preferred due to potential scoping issues)

Scope: Function-scoped

Re-declaration: Allowed

3) Define a Data Types in js?

Ans:

=> JavaScript has two main categories of data types:

1. Primitive Data Types:

* Number: Represents numeric values, including integers (whole numbers) and floating-point numbers (decimals). Examples: 10, -5.2, 3.14159.

* String: Represents a sequence of characters enclosed in single or double quotes. Examples: "Hello, world!", 'This is a string'.

* Boolean: Represents logical values: true or false. Examples: true (something is true), false (something is false).

* Null: Represents the intentional absence of a value. It signifies that there is no value assigned to the variable. Example: null.

* Undefined: Represents a variable that has been declared but not yet assigned a value. Example: let x; // x is undefined.

* Symbol (introduced in ES6): A unique and immutable (cannot be changed) primitive value that represents an identifier.

2. Non-Primitive Data Types:

* Object: A collection of key-value pairs, where keys are unique strings and values can be any data type, including other objects.

* Array: An ordered collection of items, where each item can be accessed using an index

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

Ans:

=>function mul(x) {

```
  return function(y) {
    return function(z) {
      // Return the product of all arguments
      return x * y * z;
    };
  };
}
```

5) What the deference between undefined and undeclare in JavaScript?

Ans:

=>

Undefined: A variable has been declared but not yet assigned a value.

Undeclared:The variable has not been declared at all. It doesn't exist in the current scope.

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:

Ans :

=>

```
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("Spread love everywhere you go. Let no one ever come to you without leaving happier. - Mother Teresa");
```

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

Ans:

=> This code checks if the type of '10' (which is a string) is exactly equal to the type of 10 (which is a number).

If not, it converts '10' to a number using the Number() function. After conversion, both will be of the same type, and the code will print 10.

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

Ans:

=> You can write a JavaScript program to find the area of a triangle using the formula:

Area = (base * height) / 2

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

Ans :

=>

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

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

  // Create a new Date object for Christmas of the current year
  let christmasDate = new Date(currentYear, 11, 25); // Month is 0-based index (11 represents December)

  // If Christmas has already passed this year, set it to next year
  if (currentDate.getMonth() === 11 && currentDate.getDate() > 25) {
    christmasDate.setFullYear(currentYear + 1);
```

```
  }

  // Calculate the difference in milliseconds between the
current date and Christmas
  let timeDiff = christmasDate.getTime() -
currentDate.getTime();

  // Calculate the number of days left until Christmas
  let daysLeft = Math.ceil(timeDiff / (1000 * 3600 * 24));

  return daysLeft;
}

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

10) What is Condition Statement?

Ans:

=> A condition statement, often referred to simply as a "conditional," is a programming construct that allows you to execute different blocks of code based on whether a specified condition evaluates to true or false.

1) if statement

2) else if statement

3) switch statement

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

Ans:

=> The formula you provided, C=4×a, calculates the perimeter of a rectangle, not the circumference. In a rectangle, the perimeter is the sum of all its sides.

However, if you are specifically looking for the circumference, that's a property of circles, not rectangles. The circumference of a circle is given by the formula C=2πr is the radius of the circle.

12) WAP to convert years into days and days into years?

Ans:

```
=> 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();
```

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

Ans:

```
=> 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);
```

```
}
```

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

Ans:

```
=> <!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>
```

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

Ans :

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

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

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

Ans :

=>

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"

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

Ans:

=> `true && false || false && true` involves logical AND (`&&`) and logical OR (`||`) operators.

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

Ans:

=> 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.

19) What is the use of is Nan function?

Ans:

=> The is NaN 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.

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

Ans:

=> 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.

|| (Logical OR):

Requires at least one operand to be true.

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

Ans :

=> 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.

22) Check Number Is Positive or Negative in JavaScript?

Ans :

```
=> 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.");
  }
}
checkNumber(5);    // Output: The number is positive.
checkNumber(-3);   // Output: The number is negative.
checkNumber(0);    // Output: The number is zero.
```

23) Find the Character Is Vowel or Not ?

Ans:

```
=> 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.");
}
```

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

Ans:

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

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

Ans:

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

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

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

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

Ans :

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

```
let 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}`);
```

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

Ans:

=>

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

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

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

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

Ans:

```
=> let num1 = parseFloat(prompt("Enter the first number: "));
let num2 = parseFloat(prompt("Enter the second number: "));
let 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.");
}
```

29) Write to show

i. Monday to Sunday using switch case in JS?

ii. Vowel or Consonant using switch case in JS?

Ans :

=> let 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?

```
let 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 *

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

Ans:

=> 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);
}
```

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

Ans :

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

32) Write to print factorial of given number?

Ans :

```
=> 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;
    }
}

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

33) Write to print Fibonacci series up to given numbers?

Ans:

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

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

Ans :

```
=> function reverseNumber(number) {
    let numberString = number.toString();
    let reversedString = numberString.split('').reverse().join('');
    let reversedNumber = parseInt(reversedString);
```

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

35) Write a program make a summation of given number (E.g., 1523 Ans: - 11) in JS?

Ans :

```
=> function calculateDigitSum(number) {
    let numberString = number.toString();
    let sum = 0;
    for (let i = 0; i < numberString.length; i++) {
        sum += parseInt(numberString[i]);
    }

    return sum;
}
let givenNumber = 1523;
let digitSum = calculateDigitSum(givenNumber);
console.log(`Summation of ${givenNumber}: ${digitSum}`);
```

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

Ans :

```
=> function calculateFirstAndLastDigitSum(number) {
    let numberString = number.toString();
    let firstDigit = parseInt(numberString[0]);
    let lastDigit = parseInt(numberString[numberString.length - 1]);
    let sum = firstDigit + lastDigit;
    return sum;
}
let givenNumber = 1234;
let digitSum = calculateFirstAndLastDigitSum(givenNumber);
console.log(`Sum of the first and last digits of ${givenNumber}: ${digitSum}`);
```

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

Ans :

=>

```
// Define the number of rows for the pattern
const numRows = 5;
for (let i = 1; i <= numRows; i++) {
    let rowOutput = '';
    for (let j = 1; j <= 5; j++) {
        if (j === 1) {
            rowOutput += `${i} `;
        } else {
            rowOutput += `${Math.pow(i, j)} `;
        }
    }
    console.log(rowOutput);
}
```

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);
}
```

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

Ans :

=>

```
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)

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 ?

Ans :

```
=> function getCurrentDayAndTime() {    const days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];
    const now = new Date();
    const day = days[now.getDay()];
    let hours = now.getHours();
    const ampm = hours >= 12 ? 'PM' : 'AM'; // Check if it's AM or PM
```

```
    hours = hours % 12 || 12;
    const minutes = now.getMinutes();
    const seconds = now.getSeconds();
    const milliseconds = now.getMilliseconds();

    // Format time with leading zeros if needed
    const formattedTime = `${hours < 10 ? '0' : ''}${hours} : ${minutes < 10 ? '0' : ''}${minutes} : ${seconds < 10 ? '0' : ''}${seconds} ${milliseconds} ${ampm}`;
    console.log(`Today is ${day}. Current Time is ${formattedTime}`);
}
getCurrentDayAndTime();
```

41) Write a JavaScript program to get the current date?

Ans :

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

console.log("Current Date:", getCurrentDate());
```

42) Write a JavaScript program to compare two objects?

Ans :

```
=> function compareObjects(obj1, obj2) {

    const keys1 = Object.keys(obj1);
    const keys2 = Object.keys(obj2);

    if (keys1.length !== keys2.length) {
        return false;
    }

    for (let key of keys1) {
        if (!obj2.hasOwnProperty(key)) {
            return false
        if (obj1[key] !== obj2[key]) {
            return false; // Values are not equal
        }
    }
    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));
```

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

Ans :

```
=> function arrayofObjectsToCSV(data) {
   if (data.length === 0) {
      return '' ”
   }
      const headers = Object.keys(data[0]);

   const csvHeader = headers.join(',');
      const csvRows = data.map(obj => {       const values = headers.map(header => {
         const value = obj[header];
         return typeof value === 'string' && value.includes(',') ? `"${value}"` : value;
      });
      return values.join(',');
   });
      return `${csvHeader}\n${csvRows.join('\n')}`;
}
```

```
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);
```

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

Ans :

```
=> function capitalizeFirstLetter(str) {
    if (str === '') {
      return '';
    }
    return str.charAt(0).toUpperCase() + str.slice(1);
}

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

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

Ans :

=> function isArray(variable) {

    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

46) Write a JavaScript program to clone an array?

Ans :

=> 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) {

    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) {
    return [].concat(array);
}

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

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

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

Ans :

=> 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

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

Ans :

=>const str = "Hello, World!";

console.log(str.length);

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

Ans :

=> const str = "hello, world!";

const capitalizedString = str.toUpperCase();

console.log(capitalizedString);

//output:-HELLO, WORLD

51) Write a JavaScript program to get the current date.
Expected Output : mm-dd-yyyy,mm/dd/yyyy or dd-mm-yyyy, dd/mm/yyyy?

Ans :

=>

```
function getCurrentDate() {
    const currentDate = new Date();
    const day = currentDate.getDate();
    const month = currentDate.getMonth() + 1;
    const year = currentDate.getFullYear();


    const paddedDay = (day < 10) ? '0' + day : day;
    const paddedMonth = (month < 10) ? '0' + month : month;
    const dashFormat = `${paddedMonth}-${paddedDay}-${year}`;
    const slashFormat = `${paddedMonth}/${paddedDay}/${year}`;
        console.log(`mm-dd-yyyy: ${dashFormat}`);
    console.log(`mm/dd/yyyy: ${slashFormat}`);
    console.log(`dd-mm-yyyy: ${paddedDay}-${paddedMonth}-${year}`);
    console.log(`dd/mm/yyyy: ${paddedDay}/${paddedMonth}/${year}`);
```

```
}

getCurrentDate();
```

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

Ans :

```
=> const str = "30 Days Of JavaScript";
const position = str.indexOf("a");

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

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

Ans:

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

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

54) Form Validtion in JS?

Ans:

=> 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.

55) Form in Email, number, Password, Validation?

Ans:

=>
```
<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("sub
mit", 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>
```

56) Dynamic Form Validation in JS?

Ans:

```
=> <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("submit", 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>
```

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

Ans:

=> 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.

59) What is Bom vs Dom in JS?

Ans:

=> 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.

60) Array vs object defences in JS?

Ans:

=> 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.

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

Ans:

=> const str = "Hello, World!";

const array = str.split(',');

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

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

Ans:

=>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}".`);

}

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

Ans:

=> const str = "Hello, World!";

const lowercaseString = str.toLowerCase();

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

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

Ans:

=> const str = '30 Days of JavaScript';

const characterAtIndex15 = str.charAt(15);

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

65) copy to one string to another string in JS?

Ans:

=> 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.

66) Find the length of a string without using libraryFunction?

Ans:

```
=> function findStringLength(str) {
    let count = 0;
    while (str[count] !== undefined) {
        count++;
    }
    return count;
}

const str = "Hello, World!";
const length = findStringLength(str);
console.log("Length of the string:", length); // Output: 13
```