SAURABH SINGH

JAVASCRIPT BASIC

1. **Write a JavaScript program to display the current day and time in the following format.**
   Sample Output : Today is : Friday. Current time is : 4 PM : 50 : 22

```
function getCurrentDateTime()
 { let today = new Date(); // Array of weekday names
 let daysOfWeek = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"];
 let day = daysOfWeek[today.getDay()]; // Get day of the week
 let hour = today.getHours();
let minute = today.getMinutes();
 let second = today.getSeconds();
 let period = (hour >= 12) ? "PM" : "AM"; // Determine AM/PM
 // Convert hour from 24-hour format to 12-hour format
 hour = (hour % 12 === 0) ? 12 : hour % 12; // Format minute and second with leading
zeros if necessary minute = formatTimeComponent(minute);
second = formatTimeComponent(second); // Construct the formatted string
let timeString = hour + " " + period + " : " + minute + " : " + second; // Display the result
console.log("Today is : " + day + ".");
 console.log("Current time is : " + timeString);
}
 function formatTimeComponent(timeComponent)
{
 // Function to add leading zero to single digit numbers
 return (timeComponent < 10) ? "0" + timeComponent : timeComponent;
 }
 // Call the function to display current day and time
 getCurrentDateTime();
```

2. **Write a JavaScript program to print the contents of the current window.**

```
function printCurrentWindow()
 {
window.print()
; }
 // Call the function to print the current window printCurrentWindow();
```

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

```
function getCurrentDate()
 {
 let today = new Date();
 let dd = today.getDate();
 let mm = today.getMonth() + 1;
 // January is 0, so we add 1
 let yyyy = today.getFullYear();
 // Ensure mm and dd are two digits (e.g., 03 for March and 07 for the 7th)
mm = formatTimeComponent(mm);
 dd = formatTimeComponent(dd);
 // Choose your desired date format here
 let dateFormat = "mm-dd-yyyy";
// Change this to mm/dd/yyyy, dd-mm-yyyy, dd/mm/yyyy as needed
// Construct the formatted date string based on the chosen format let formattedDate;
 switch (dateFormat)
{
 case "mm-dd-yyyy":
 formattedDate = mm + "-" + dd + "-" + yyyy;
break;
case "mm/dd/yyyy":
formattedDate = mm + "/" + dd + "/" + yyyy;
 break; case "dd-mm-yyyy":
 formattedDate = dd + "-" + mm + "-" + yyyy;
 break;
 case "dd/mm/yyyy":
formattedDate = dd + "/" + mm + "/" + yyyy;
 break;
default: formattedDate = "Invalid date format";
} return formattedDate;
 }
function formatTimeComponent(timeComponent)
 {
 // Function to add leading zero to single digit numbers
 return (timeComponent < 10) ? "0" + timeComponent : timeComponent;
 }
 // Call the function to get the current date in the desired format
 let currentDate = getCurrentDate();
 console.log("Current date in mm-dd-yyyy format:", currentDate);
```

4. **. Write a JavaScript program to find the area of a triangle where lengths of the three of its sides are 5, 6, 7.**

```
function calculateTriangleArea(a, b, c) {
```

```
// Calculate the semi-perimeter of the triangle
let s = (a + b + c) / 2; // Calculate the area using Heron's formula
 let area = Math.sqrt(s * (s - a) * (s - b) * (s - c));
 return area;
 } // Given sides of the triangle
let side1 = 5;
let side2 = 6;
let side3 = 7; // Calculate the area of the triangle
 let area = calculateTriangleArea(side1, side2, side3);
 console.log("The area of the triangle with sides", side1, ",", side2, ",", side3, "is:", area);
```

5. **Write a JavaScript program to rotate the string 'w3resource' in right direction by periodically removing one letter from the end of the string and attaching it to the front.**

```
function rotateStringRight(str) {
let result = str; // Rotate the string 10 times
 for (let i = 0; i < 10; i++) {
 // Remove the last character
 let lastChar = result.charAt(result.length - 1);
// Attach the last character to the front of the string
result = lastChar + result.substring(0, result.length - 1); // Display the rotated string
console.log(result); }
 } // String to rotate
 let originalString = 'w3resource'; // Rotate the string right
rotateStringRight(originalString);
```

6. **Write a JavaScript program to determine whether a given year is a leap year in the Gregorian calendar.**

```
function isLeapYear(year) {
 // Leap year is divisible by 4 // However, years divisible by 100 are not leap years,
unless they are also divisible by 400
 if ((year % 4 === 0 && year % 100 !== 0) || (year % 400 === 0)) {
return true;
 } else { return false;
}
 } // Test cases
let year1 = 2020;
 let year2 = 2021;
let year3 = 2000;
let year4 = 1900;
console.log(year1 + " is a leap year: " + isLeapYear(year1));
console.log(year2 + " is a leap year: " + isLeapYear(year2));
```

```
console.log(year3 + " is a leap year: " + isLeapYear(year3));
console.log(year4 + " is a leap year: " + isLeapYear(year4));
```

7. **Write a JavaScript program to find 1st January be a Sunday between 2014 and 2050.**

```
function findJanuary1stSunday() {
 let results = [];
 for (let year = 2014; year <= 2050; year++) {
 let date = new Date(year, 0, 1);
 // January is month 0 in JavaScript Date
 // Check if January 1st is a Sunday (getDay() returns 0 for Sunday)
if (date.getDay() === 0) {
 results.push(year);
 }
 }
 return results;
 }
 // Call the function to find years where January 1st is a Sunday
 let years = findJanuary1stSunday();
// Print the results
 if (years.length > 0) {
 console.log("Years between 2014 and 2050 where January 1st is a Sunday:");
console.log(years.join(", "));
 }
 else
 {
 console.log("No years between 2014 and 2050 where January 1st is a Sunday.");
}
```

8. **Write a JavaScript program where the program takes a random integer between 1 to 10, the user is then prompted to input a guess number. If the user input matches with guess number, the program will display a message "Good Work" otherwise display a message "Not matched".**

```
let randomNumber = Math.floor(Math.random() * 10) + 1;
// Prompts the user to guess the number
 let userGuess = parseInt(prompt("Guess a number between 1 and 10:"));
 // Checks if the user's guess matches the random number
if (userGuess === randomNumber) {
alert("Good Work");
 }
else {
 alert("Not matched");
```

```
}
```

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

```
function daysUntilNextChristmas() {
let today = new Date();
 let year = today.getFullYear();
 let nextChristmas = new Date(year, 11, 25);
// December 25 of the current year
 // If Christmas has already passed this year, set the next Christmas to next year
 if (today > nextChristmas)
{
 nextChristmas.setFullYear(year + 1);
 }
 let oneDay = 24 * 60 * 60 * 1000; // milliseconds in one day
 let daysLeft = Math.round((nextChristmas - today) / oneDay); return daysLeft;
}
 alert("Days until next Christmas: " + daysUntilNextChristmas());
```

10. **Write a JavaScript program to calculate multiplication and division of two numbers (input from user).**

```
let num1 = parseFloat(prompt("Enter the first number:"));
let num2 = parseFloat(prompt("Enter the second number:"));
// Calculates the multiplication and division of the two numbers
 let multiplication = num1 * num2; let division = num1 / num2;
 // Displays the results alert("Multiplication of " + num1 + " and " + num2 + " is: " +
multiplication);
 alert("Division of " + num1 + " by " + num2 + " is: " + division);
```

11. **Write a JavaScript program to convert temperatures to and from celsius, fahrenheit. [Formula : c/5 = (f-32)/9 [where c = temperature in celsius and f = temperature in fahrenheit]**

```
function celsiusToFahrenheit(celsius) {
 return (celsius * 9/5) + 32;
 }
 function fahrenheitToCelsius(fahrenheit) {
 return (fahrenheit - 32) * 5/9; }
// Prompts the user to choose the conversion type
let conversionType = prompt("Type 'C' to convert from Celsius to Fahrenheit, or 'F' to
convert from Fahrenheit to Celsius:").toUpperCase();
```

```
 if (conversionType === 'C') { // Prompts the user to input the temperature in Celsius let
celsius = parseFloat(prompt("Enter the temperature in Celsius:"));
 let fahrenheit = celsiusToFahrenheit(celsius);
alert(celsius + "°C is " + fahrenheit.toFixed(2) + "°F");
 }
 else if (conversionType === 'F') {
// Prompts the user to input the temperature in Fahrenheit
let fahrenheit = parseFloat(prompt("Enter the temperature in Fahrenheit:"));
let celsius = fahrenheitToCelsius(fahrenheit);
alert(fahrenheit + "°F is " + celsius.toFixed(2) + "°C"); }
 else { alert("Invalid input. Please refresh and try again.");
 }
```

12. Write a JavaScript program to get the website URL (loading page).

```
let currentURL = window.location.href;
 // Display the URL to the user
alert("The current website URL is: " + currentURL);
```

JAVASCRIPT FUNCTIONS

1. **Write a JavaScript function that reverse a number. Example x = 32243; Expected Output : 34223**

```
function reverseNumber(num) {
 // Convert the number to a string
let numStr = num.toString();
 // Split the string into an array of characters, reverse the array, and join it back into a
string
let reversedStr = numStr.split('').reverse().join(''); // Convert the reversed string back to a
number
let reversedNum = parseInt(reversedStr);
return reversedNum;
}
```

2. **Write a JavaScript function that checks whether a passed string is palindrome or not? A palindrome is word, phrase, or sequence that reads the same backward as forward, e.g., madam or nurses run.**

```
function isPalindrome(str) {
// Remove non-alphanumeric characters and convert to lowercase
let cleanedStr = str.replace(/[^A-Za-z0-9]/g, '').toLowerCase(); // Reverse the cleaned
string
```

```
let reversedStr = cleanedStr.split('').reverse().join('');
// Check if the cleaned string is equal to its reversed version
return cleanedStr === reversedStr; } // Example usage
console.log(isPalindrome("madam")); // true console.log(isPalindrome("nurses run")); //
true console.log(isPalindrome("hello"));
 // false console.log(isPalindrome("A man, a plan, a canal, Panama")); // true
```

3. **Write a JavaScript function that generates all combinations of a string. Example string : 'dog' Expected Output : d,do,dog,o,og,g**

```
function generateCombinations(str) {
 let result = []; // Helper function to generate combinations
 function combine(prefix, str) {
 for (let i = 0; i < str.length; i++) { let newPrefix = prefix + str[i];
result.push(newPrefix);
combine(newPrefix, str.slice(i + 1)); }
}
 combine('', str); return result;
 } // Example usage
 let inputString = 'dog';
let combinations = generateCombinations(inputString);
 console.log(combinations); // Output: [ 'd', 'do', 'dog', 'o', 'og', 'g' ]
```

4. **Write a JavaScript function that returns a passed string with letters in alphabetical order**

```
function sortStringAlphabetically(str) {
   // Convert the string to an array of characters
   let charArray = str.split('');

   // Sort the array alphabetically
   charArray.sort();

   // Join the sorted array back into a string
   let sortedStr = charArray.join('');

   return sortedStr;
}

// Example usage
let inputString = 'dog';
let sortedString = sortStringAlphabetically(inputString);
console.log(sortedString); // Output: 'dgo'
```

```
inputString = 'javascript';
sortedString = sortStringAlphabetically(inputString);
console.log(sortedString); // Output: 'aacijprstv'
```

5. **Write a JavaScript function that accepts a string as a parameter and converts the first letter of each word of the string in upper case.**

```
function capitalizeFirstLetterOfEachWord(str) {
    // Split the string into an array of words
    let words = str.split(' ');

    // Capitalize the first letter of each word
    let capitalizedWords = words.map(word => {
        return word.charAt(0).toUpperCase() + word.slice(1).toLowerCase();
    });

    // Join the capitalized words back into a single string
    let capitalizedStr = capitalizedWords.join(' ');

    return capitalizedStr;
}

// Example usage
let inputString = 'hello world this is javascript';
let capitalizedString = capitalizeFirstLetterOfEachWord(inputString);
console.log(capitalizedString); // Output: 'Hello World This Is Javascript'

inputString = 'javaScript is fun';
capitalizedString = capitalizeFirstLetterOfEachWord(inputString);
console.log(capitalizedString); // Output: 'Javascript Is Fun'
```

6. **. Write a JavaScript function that accepts a string as a parameter and find the longest word within the string. Example string : 'Web Development Tutorial' Expected Output : 'Development'**

```
function findLongestWord(str) {
    // Split the string into an array of words
    let words = str.split(' ');
```

```
    // Initialize a variable to keep track of the longest word
    let longestWord = '';

    // Iterate through the words to find the longest one
    for (let word of words) {
        if (word.length > longestWord.length) {
            longestWord = word;
        }
    }

    return longestWord;
}

// Example usage
let inputString = 'Web Development Tutorial';
let longestWord = findLongestWord(inputString);
console.log(longestWord); // Output: 'Development'
```

7. **Write a JavaScript function that accepts a string as a parameter and counts the number of vowels within the string. Note : As the letter 'y' can be regarded as both a vowel and a consonant, we do not count 'y' as vowel here. Example string : 'The quick brown fox' Expected Output : 5**

```
        function countVowels(str) {
    // Define a string of vowels (excluding 'y')
    const vowels = 'aeiouAEIOU';

    // Initialize a counter for the number of vowels
    let vowelCount = 0;

    // Iterate through each character in the string
    for (let char of str) {
        if (vowels.includes(char)) {
            vowelCount++;
        }
    }

    return vowelCount;
}
```

```
// Example usage
let inputString = 'The quick brown fox';
let numberOfVowels = countVowels(inputString);
console.log(numberOfVowels); // Output: 5
```

8. **Write a JavaScript function that accepts a number as a parameter and check the number is prime or not. Note : A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself.**

```
function isPrime(number) {
    // Check if number is less than or equal to 1
    if (number <= 1) {
        return false;
    }

    // Check for divisibility from 2 to the square root of the number
    for (let i = 2; i <= Math.sqrt(number); i++) {
        if (number % i === 0) {
            return false;
        }
    }

    // If no divisors other than 1 and itself, then it's prime
    return true;
}

// Example usage
let num = 29;
if (isPrime(num)) {
    console.log(num + " is a prime number.");
} else {
    console.log(num + " is not a prime number.");
}
```

9. **Write a JavaScript function which accepts an argument and returns the type. Note : There are six possible values that typeof returns: object, boolean, function, number, string, and undefined.**

```
function getType(value) {
    return typeof value;
}

// Example usage
console.log(getType("Hello"));    // Output: "string"
console.log(getType(42));         // Output: "number"
console.log(getType(true));       // Output: "boolean"
console.log(getType(function() {})); // Output: "function"
console.log(getType({}));         // Output: "object"
console.log(getType(undefined));  // Output: "undefined"
```

10. **Write a JavaScript function which returns the n rows by n columns identity matrix.**

```
function identityMatrix(n) {
    let matrix = [];

    // Iterate through rows
    for (let i = 0; i < n; i++) {
        // Create a new row array
        let row = [];

        // Iterate through columns
        for (let j = 0; j < n; j++) {
            // Set 1 for diagonal elements, 0 otherwise
            if (i === j) {
                row.push(1);
            } else {
                row.push(0);
            }
        }

        // Push the row to the matrix
        matrix.push(row);
    }

    return matrix;
```

```
}

// Example usage
let n = 4;
let resultMatrix = identityMatrix(n);
console.log(resultMatrix);
```

11. **Write a JavaScript function which will take an array of numbers stored and find the second lowest and second greatest numbers, respectively. Sample array : [1,2,3,4,5] Expected Output : 2,4**

```
function findSecondLowestAndGreatest(numbers) {
    // Sort the array in ascending order
    numbers.sort(function(a, b) {
        return a - b;
    });

    // Second lowest number is at index 1 (after sorting)
    let secondLowest = numbers[1];

    // Second greatest number is at the second last index (before sorting)
    let secondGreatest = numbers[numbers.length - 2];

    return [secondLowest, secondGreatest];
}

// Example usage
let sampleArray = [1, 2, 3, 4, 5];
let result = findSecondLowestAndGreatest(sampleArray);
console.log("Second lowest and second greatest numbers:", result.join(', ')); // Output:
"2, 4"
```

12. **Write a JavaScript function which says whether a number is perfect. According to Wikipedia : In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself (also known as its aliquot sum). Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself).**

```
function isPerfectNumber(number) {
    if (number <= 1) {
        return false; // Perfect numbers are positive integers greater than 1
    }

    let sum = 0;

    // Find all proper divisors (excluding the number itself)
    for (let i = 1; i <= Math.sqrt(number); i++) {
        if (number % i === 0) {
            sum += i; // Add divisor
            if (i !== 1 && i !== number / i) {
                sum += number / i; // Add corresponding divisor
            }
        }
    }

    // Check if the sum of divisors equals the number itself
    return sum === number;
}

// Example usage
let num = 28;
if (isPerfectNumber(num)) {
    console.log(num + " is a perfect number.");
} else {
    console.log(num + " is not a perfect number.");
}

num = 6;
if (isPerfectNumber(num)) {
    console.log(num + " is a perfect number.");
} else {
    console.log(num + " is not a perfect number.");
}
```

13. Write a JavaScript function to compute the factors of a positive integer.

```
function computeFactors(number) {
    if (number <= 0 || !Number.isInteger(number)) {
```

```
        return "Please enter a positive integer.";
    }

    let factors = new Set();

    // Iterate through numbers up to the square root of 'number'
    for (let i = 1; i <= Math.sqrt(number); i++) {
        if (number % i === 0) {
            factors.add(i); // Add divisor
            factors.add(number / i); // Add complement (result of division)
        }
    }

    // Convert set to array and sort it
    let sortedFactors = Array.from(factors).sort((a, b) => a - b);

    return sortedFactors;
}

// Example usage
let num = 28;
let factors = computeFactors(num);
console.log("Factors of", num + ":", factors); // Output: [1, 2, 4, 7, 14, 28]

num = 12;
factors = computeFactors(num);
console.log("Factors of", num + ":", factors); // Output: [1, 2, 3, 4, 6, 12]
```

14. . Write a JavaScript function to convert an amount to coins. Sample function : amountTocoins(46, [25, 10, 5, 2, 1]) Here 46 is the amount. and 25, 10, 5, 2, 1 are coins. Output : 25, 10, 10, 1

```
function amountToCoins(amount, coins) {
    let result = [];

    for (let i = 0; i < coins.length; i++) {
        while (amount >= coins[i]) {
            result.push(coins[i]);
            amount -= coins[i];
        }
```

```
    }

    return result;
}
```

15. **Write a JavaScript function to compute the value of bn where n is the exponent and b is the bases. Accept b and n from the user and display the result.**

```
function computePower(base, exponent) {
    return Math.pow(base, exponent);
}

// Example usage
let b = 2;
let n = 5;
let result = computePower(b, n);
console.log(`${b}^${n} = ${result}`); // Output: 2^5 = 32
```

16. **Write a JavaScript function to extract unique characters from a string. Example string : "thequickbrownfoxjumpsoverthelazydog" Expected Output : "thequickbrownfxjmpsvlazydg"**

```
function extractUniqueCharacters(str) {
    // Use a Set to store unique characters
    let uniqueChars = new Set();

    // Iterate through each character in the string
    for (let char of str) {
        uniqueChars.add(char); // Add character to the set (automatically handles
duplicates)
    }

    // Convert Set to Array and then to string (joining elements)
    let uniqueString = Array.from(uniqueChars).join('');

    return uniqueString;
}
```

```
// Example usage
let inputString = "thequickbrownfoxjumpsoverthelazydog";
let uniqueCharacters = extractUniqueCharacters(inputString);
console.log("Unique characters:", uniqueCharacters); // Output:
"thequickbrownfxjmpsvlazydg"
```

17.. Write a JavaScript function to get the number of occurrences of each letter in specified string.

```
function countOccurrences(str) {
    // Initialize an empty object to store counts
    let charCount = {};

    // Convert the string to lowercase to ignore case sensitivity
    str = str.toLowerCase();

    // Iterate through each character of the string
    for (let char of str) {
        // Check if the character is a letter (a-z)
        if (/^[a-z]$/.test(char)) {
            // Increment count for the character in the object
            if (charCount[char]) {
                charCount[char]++;
            } else {
                charCount[char] = 1;
            }
        }
    }

    return charCount;
}

// Example usage
let inputString = "The quick brown fox jumps over the lazy dog.";
let result = countOccurrences(inputString);
console.log("Occurrences of each letter:", result);
```

18. **Write a function for searching JavaScript arrays with a binary search. Note : A binary search searches by splitting an array into smaller and smaller chunks until it finds the desired value.**

```
function binarySearch(arr, target) {
    let left = 0;
    let right = arr.length - 1;

    while (left <= right) {
        let mid = Math.floor((left + right) / 2);

        // Check if target is present at mid
        if (arr[mid] === target) {
            return mid;
        }

        // If target greater, ignore left half
        if (arr[mid] < target) {
            left = mid + 1;
        }
        // If target is smaller, ignore right half
        else {
            right = mid - 1;
        }
    }

    // If element was not present
    return -1;
}

// Example usage:
let sortedArray = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91];
let target = 23;
let result = binarySearch(sortedArray, target);
if (result !== -1) {
    console.log(`Element ${target} found at index ${result}.`);
} else {
    console.log(`Element ${target} not found in the array.`);
}
```

19. **Write a JavaScript function that returns array elements larger than a number.**

```
function elementsLargerThan(arr, number) {
    // Filter elements in the array that are larger than 'number'
    let result = arr.filter(element => element > number);
    return result;
}

// Example usage:
let numbers = [10, 5, 25, 3, 15, 7];
let threshold = 10;
let largerNumbers = elementsLargerThan(numbers, threshold);
console.log(`Elements larger than ${threshold}:`, largerNumbers);
```

20. **Write a JavaScript function that generates a string id (specified length) of random characters. Sample character list : "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345 678 9"**

```
function elementsLargerThan(arr, number) {
    // Filter elements in the array that are larger than 'number'
    let result = arr.filter(element => element > number);
    return result;
}

// Example usage:
let numbers = [10, 5, 25, 3, 15, 7];
let threshold = 10;
let largerNumbers = elementsLargerThan(numbers, threshold);
console.log(`Elements larger than ${threshold}:`, largerNumbers);
```

21. **Write a JavaScript function to get all possible subset with a fixed length (for example 2) combinations in an array. Sample array : [1, 2, 3] and subset length is 2 Expected output : [[2, 1], [3, 1], [3, 2], [3, 2, 1]]**

```
function subsetsOfSize(arr, k) {
    let results = [];
```

```
    // Helper function to generate subsets recursively
    function generateSubsets(current, start) {
        if (current.length === k) {
            results.push(current.slice()); // Add a copy of current subset to results
            return;
        }

        for (let i = start; i < arr.length; i++) {
            current.push(arr[i]);
            generateSubsets(current, i + 1);
            current.pop();
        }
    }

    generateSubsets([], 0);
    return results;
}

// Example usage:
let array = [1, 2, 3];
let subsetLength = 2;
let result = subsetsOfSize(array, subsetLength);
console.log(`Subsets of length ${subsetLength}:`, result);
```

22. Write a JavaScript function that accepts two arguments, a string and a letter and the function will count the number of occurrences of the specified letter within the string. Sample arguments : 'w3resource.com', 'o' Expected output : 2

```
function countOccurrences(str, letter) {
    // Convert the string and letter to lowercase to handle case insensitivity
    str = str.toLowerCase();
    letter = letter.toLowerCase();

    // Initialize a counter for occurrences
    let count = 0;

    // Loop through each character in the string
    for (let i = 0; i < str.length; i++) {
        // If the character matches the specified letter, increment count
```

```
        if (str[i] === letter) {
            count++;
        }
    }

    return count;
}

// Example usage:
let inputString = 'w3resource.com';
let targetLetter = 'o';
let result = countOccurrences(inputString, targetLetter);
console.log(`Number of occurrences of '${targetLetter}' in '${inputString}':`, result); //
Output: 2
```

23. Write a JavaScript function to find the first not repeated character. Sample arguments : 'abacddbec' Expected output : 'e'

```
function firstNonRepeatedCharacter(str) {
    // Initialize an empty object to store character counts
    let charCount = {};

    // Iterate through the string to count occurrences of each character
    for (let char of str) {
        if (charCount[char]) {
            charCount[char]++;
        } else {
            charCount[char] = 1;
        }
    }

    // Find the first character with a count of 1 in the original string order
    for (let char of str) {
        if (charCount[char] === 1) {
            return char;
        }
    }

    // If no non-repeated character found, return null or an appropriate message
```

```
    return null; // or return undefined, '', or throw an error as per requirement
}
```

24. . Write a JavaScript function to apply Bubble Sort algorithm. Note : According to wikipedia "Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order". Sample array : [12, 345, 4, 546, 122, 84, 98, 64, 9, 1, 3223, 455, 23, 234, 213] Expected output : [3223, 546, 455, 345, 234, 213, 122, 98, 84, 64, 23, 12, 9, 4, 1]

```
function bubbleSort(arr) {
    let len = arr.length;
    let swapped;

    do {
        swapped = false;
        for (let i = 0; i < len - 1; i++) {
            if (arr[i] < arr[i + 1]) {
                // Swap elements arr[i] and arr[i + 1]
                let temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
                swapped = true;
            }
        }
        len--; // Optimization: reduce length of array to sort on each pass
    } while (swapped);

    return arr;
}

// Example usage:
let array = [12, 345, 4, 546, 122, 84, 98, 64, 9, 1, 3223, 455, 23, 234, 213];
let sortedArray = bubbleSort(array.slice()); // Create a copy of array using slice() for sorting
console.log("Sorted array using Bubble Sort:", sortedArray);
```

25. **Write a JavaScript function that accept a list of country names as input and returns the longest country name as output. Sample function : Longest_Country_Name(["Australia", "Germany", "United States of America"]) Expected output : "United States of America"**

```
function longestCountryName(countryNames) {
    let longestName = "";

    // Iterate through the array of country names
    for (let country of countryNames) {
        // Update longestName if current country name is longer
        if (country.length > longestName.length) {
            longestName = country;
        }
    }

    return longestName;
}

// Example usage:
let countries = ["Australia", "Germany", "United States of America"];
let longest = longestCountryName(countries);
console.log("Longest country name:", longest); // Output: "United States of America"
```

26. **Write a JavaScript function to find longest substring in a given a string without repeating characters.**

```
function longestSubstringWithoutRepeating(s) {
    let maxLength = 0; // Track maximum length of substring without repeating characters
    let start = 0; // Start index of current substring
    let charMap = {}; // Hash map to store characters and their indices

    for (let end = 0; end < s.length; end++) {
        let currentChar = s[end];

        // If currentChar is already in charMap and its index is >= start, update start
        if (charMap[currentChar] !== undefined && charMap[currentChar] >= start) {
            start = charMap[currentChar] + 1;
        }
```

```
        // Update charMap with currentChar index
        charMap[currentChar] = end;

        // Update maxLength if current substring length is greater
        maxLength = Math.max(maxLength, end - start + 1);
    }

    // Return the longest substring length without repeating characters
    return maxLength;
}

// Example usage:
let inputString = "abcabcbb";
let longestLength = longestSubstringWithoutRepeating(inputString);
console.log(`Length of longest substring without repeating characters in
'${inputString}':`, longestLength); // Output: 3 (for "abc")
```

27. **7. Write a JavaScript function that returns the longest palindrome in a given string.**

```
function longestPalindrome(s) {
    if (s === null || s.length === 0) {
        return "";
    }

    let start = 0;
    let end = 0;

    // Helper function to expand around center
    function expandAroundCenter(left, right) {
        while (left >= 0 && right < s.length && s[left] === s[right]) {
            left--;
            right++;
        }
        // Adjust left and right to get the correct substring indices
        left++;
        right--;
```

```
        if (right - left > end - start) {
            start = left;
            end = right;
        }
    }

    // Iterate through the string to find the longest palindrome
    for (let i = 0; i < s.length; i++) {
        // Odd length palindromes (single character center)
        expandAroundCenter(i, i);

        // Even length palindromes (two consecutive characters center)
        expandAroundCenter(i, i + 1);
    }

    return s.substring(start, end + 1);
}

// Example usage:
let inputString = "babad";
let longestPal = longestPalindrome(inputString);
console.log(`Longest palindrome in '${inputString}':`, longestPal); // Output: "bab" or
"aba"
```

28. Write a JavaScript program to pass a 'JavaScript function' as parameter

```
function higherOrderFunction(callback) {
    // Call the callback function provided as parameter
    callback();
}

// Define a function that will be passed as a parameter
function sayHello() {
    console.log("Hello, World!");
}

// Call the higher order function and pass sayHello as parameter
higherOrderFunction(sayHello);
```

29. **Write a JavaScript function to get the function name.**

```
function getFunctionName(func) {
    // Return the name property of the function
    return func.name;
}

// Example usage:
function greet() {
    console.log("Hello, World!");
}

let functionName = getFunctionName(greet);
console.log("Function name:", functionName); // Output: "greet"
```

JAVA SCRIPT RECURSION

1. **Write a JavaScript program to calculate the factorial of a number. In mathematics, the factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n. For example, 5! = 5 x 4 x 3 x 2 x 1 = 120**

```
function factorialRecursive(n) {
    // Base case: factorial of 0 is 1
    if (n === 0) {
        return 1;
    }
    // Recursive case: n! = n * (n-1)!
    else {
        return n * factorialRecursive(n - 1);
    }
}
```

2. **Write a JavaScript program to find the greatest common divisor (gcd) of two positive numbers.**

```
function gcd(a, b) {
    // Using Euclidean algorithm
```

```
    while (b !== 0) {
        let temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// Example usage:
let number1 = 24;
let number2 = 36;
let result = gcd(number1, number2);
console.log(`Greatest Common Divisor (GCD) of ${number1} and ${number2} is:`,
result); // Output: 12
```

3. **Write a JavaScript program to get the integers in range (x, y). Example : range(2, 9)**

```
function range(x, y) {
    // Initialize an empty array to store the range of integers
    let result = [];

    // Iterate from x+1 to y-1 (excluding x and y)
    for (let i = x + 1; i < y; i++) {
        result.push(i); // Push each integer into the result array
    }

    return result;
}

// Example usage:
let start = 2;
let end = 9;
let numbersInRange = range(start, end);
console.log(`Integers in range (${start}, ${end}):`, numbersInRange); // Output: [3, 4, 5,
6, 7, 8]
```

4. **Write a JavaScript program to compute the sum of an array of integers. Example : var array = [1, 2, 3, 4, 5, 6] Expected Output : 21**

```
function computeSum(array) {
    let sum = 0;

    // Iterate through each element of the array
    for (let i = 0; i < array.length; i++) {
        sum += array[i]; // Add each element to the sum
    }

    return sum;
}

// Example usage:
let array = [1, 2, 3, 4, 5, 6];
let result = computeSum(array);
console.log("Sum of array", array, "is:", result); // Output: 21
```

5. **. Write a JavaScript program to compute the exponent of a number. Note : The exponent of a number says how many times the base number is used as a factor. 82 = 8 x 8 = 64. Here 8 is the base and 2 is the exponent.**

```
function computeExponent(base, exponent) {
    return Math.pow(base, exponent);
}

// Example usage:
let base = 8;
let exponent = 2;
let result = computeExponent(base, exponent);
console.log(`${base}^${exponent} =`, result); // Output: 8^2 = 64
```

6. **Write a JavaScript program to get the first n Fibonacci numbers. Note : The Fibonacci Sequence is the series of numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, . . . Each subsequent number is the sum of the previous two.**

```
function fibonacciRecursive(n) {
```

```
    if (n === 1) {
        return [0]; // Base case: return [0] for the first Fibonacci number
    } else if (n === 2) {
        return [0, 1]; // Base case: return [0, 1] for the first two Fibonacci numbers
    } else {
        let fibArray = fibonacciRecursive(n - 1); // Recursively get Fibonacci numbers
        fibArray.push(fibArray[fibArray.length - 1] + fibArray[fibArray.length - 2]); // Add new
Fibonacci number
        return fibArray;
    }
}

// Example usage:
let count = 10; // Number of Fibonacci numbers to generate
let fibonacciNumbers = fibonacciRecursive(count);
console.log(`First ${count} Fibonacci numbers:`, fibonacciNumbers);
```

7.Write a JavaScript program to check whether a number is even or not

```
function isEven(number) {
    return number % 2 === 0;
}

// Example usage:
let num = 10;
if (isEven(num)) {
    console.log(`${num} is even.`);
} else {
    console.log(`${num} is not even.`);
}
```

8. Write a JavaScript program for binary search. Sample array : [0,1,2,3,4,5,6]
console.log(l.br_search(5)) will return '5'

```
Array.prototype.br_search = function(target) {
    let min = 0;
    let max = this.length - 1;
    let mid;

    while (min <= max) {
```

```
        mid = Math.floor((min + max) / 2);

        if (this[mid] === target) {
            return mid; // Return the index if the target is found
        } else if (this[mid] < target) {
            min = mid + 1; // If target is greater, ignore left half
        } else {
            max = mid - 1; // If target is smaller, ignore right half
        }
    }

    return -1; // Return -1 if target is not found
};

// Example usage:
let array = [0, 1, 2, 3, 4, 5, 6];
console.log(array.br_search(5)); // Output: 5 (index of number 5 in the array)
```

9. Write a merge sort program in JavaScript. Sample array : [34,7,23,32,5,62] Sample output : [5, 7, 23, 32, 34, 62]

```
function mergeSort(array) {
    // Base case: If the array has 0 or 1 elements, it is already sorted
    if (array.length <= 1) {
        return array;
    }

    // Splitting the array into two halves
    const middle = Math.floor(array.length / 2);
    const leftHalf = array.slice(0, middle);
    const rightHalf = array.slice(middle);

    // Recursively sort each half
    const sortedLeft = mergeSort(leftHalf);
    const sortedRight = mergeSort(rightHalf);

    // Merge the sorted halves
    return merge(sortedLeft, sortedRight);
}
```

```
// Merge function to combine two sorted arrays
function merge(left, right) {
    let result = [];
    let leftIndex = 0;
    let rightIndex = 0;

    // Compare elements from both arrays and add the smaller one to the result array
    while (leftIndex < left.length && rightIndex < right.length) {
        if (left[leftIndex] < right[rightIndex]) {
            result.push(left[leftIndex]);
            leftIndex++;
        } else {
            result.push(right[rightIndex]);
            rightIndex++;
        }
    }

    // Concatenate any remaining elements from left and right arrays
    return result.concat(left.slice(leftIndex)).concat(right.slice(rightIndex));
}

// Example usage:
let array = [34, 7, 23, 32, 5, 62];
let sortedArray = mergeSort(array);
console.log("Sorted array:", sortedArray); // Output: [5, 7, 23, 32, 34, 62]
```

JavaScript conditional statements and loops

1. **Write a JavaScript program that accept two integers and display the larger.**

```
function displayLargerNumber(a, b) {
    if (a > b) {
        console.log(`The larger number is: ${a}`);
    } else if (b > a) {
        console.log(`The larger number is: ${b}`);
    } else {
        console.log("Both numbers are equal.");
```

```
    }
}

// Example usage:
let num1 = 10;
let num2 = 20;
displayLargerNumber(num1, num2); // Output: The larger number is: 20

function displayLargerNumber(a, b) {
    if (a > b) {
        console.log(`The larger number is: ${a}`);
    } else if (b > a) {
        console.log(`The larger number is: ${b}`);
    } else {
        console.log("Both numbers are equal.");
    }
}

// Example usage:
let num1 = 10;
let num2 = 20;
displayLargerNumber(num1, num2); // Output: The larger number is: 20
```

2. **Write a JavaScript conditional statement to find the sign of product of three numbers. Display an alert box with the specified sign. Sample numbers : 3, -7, 2 Output : The sign is -**

```
function findSignOfProduct(a, b, c) {
    let product = a * b * c;
    if (product > 0) {
        alert("The sign is +");
    } else if (product < 0) {
        alert("The sign is -");
    } else {
        alert("The product is 0, which has no sign");
    }
}

// Example usage:
```

```
let num1 = 3;
let num2 = -7;
let num3 = 2;
findSignOfProduct(num1, num2, num3); // Output: The sign is -
```

3. **Write a JavaScript conditional statement to sort three numbers. Display an alert box to show the result. Sample numbers : 0, -1, 4 Output : 4, 0, -1**

```
    function sortThreeNumbersDescending(a, b, c) {
  let sortedArray = [];

  if (a >= b && a >= c) {
    if (b >= c) {
      sortedArray = [a, b, c];
    } else {
      sortedArray = [a, c, b];
    }
  } else if (b >= a && b >= c) {
    if (a >= c) {
      sortedArray = [b, a, c];
    } else {
      sortedArray = [b, c, a];
    }
  } else {
    if (a >= b) {
      sortedArray = [c, a, b];
    } else {
      sortedArray = [c, b, a];
    }
  }

  alert("Sorted numbers in descending order: " + sortedArray.join(", "));
}

// Example usage:
let num1 = 0;
let num2 = -1;
let num3 = 4;
sortThreeNumbersDescending(num1, num2, num3); // Output: 4, 0, -1
```

4. **Write a JavaScript conditional statement to find the largest of five numbers. Display an alert box to show the result.**

```
function findLargestOfFive(a, b, c, d, e) {
    let largest = a;

    if (b > largest) {
        largest = b;
    }
    if (c > largest) {
        largest = c;
    }
    if (d > largest) {
        largest = d;
    }
    if (e > largest) {
        largest = e;
    }

    alert("The largest number is: " + largest);
}

// Example usage:
let num1 = 3;
let num2 = 7;
let num3 = 2;
let num4 = 10;
let num5 = 5;
findLargestOfFive(num1, num2, num3, num4, num5); // Output: The largest number is:
10
```

5. **Write a JavaScript for loop that will iterate from 0 to 15. For each iteration, it will check if the current number is odd or even, and display a message to the screen. Sample Output : "0 is even" "1 is odd" "2 is even"**

```
for (let i = 0; i <= 15; i++) {
```

```
    if (i % 2 === 0) {
        console.log(i + " is even");
    } else {
        console.log(i + " is odd");
    }
}
```

6. **Write a JavaScript program which compute, the average marks of the following students Then, this average is used to determine the corresponding grade. Student Name Marks David 80 Vinoth 77 Divya 88 Ishitha 95 Thomas 68 The grades are computed as follows : Range Grade <60 F <70 D <80 C <90 B <100 A**

```
        const students = [
    { name: 'David', marks: 80 },
    { name: 'Vinoth', marks: 77 },
    { name: 'Divya', marks: 88 },
    { name: 'Ishitha', marks: 95 },
    { name: 'Thomas', marks: 68 }
];

// Step 2: Calculate the average marks
let totalMarks = 0;
for (let i = 0; i < students.length; i++) {
    totalMarks += students[i].marks;
}
const averageMarks = totalMarks / students.length;

// Step 3: Determine the grade based on the average marks
let grade;
if (averageMarks < 60) {
    grade = 'F';
} else if (averageMarks < 70) {
    grade = 'D';
} else if (averageMarks < 80) {
    grade = 'C';
} else if (averageMarks < 90) {
    grade = 'B';
} else {
```

```
    grade = 'A';
}

// Step 4: Display the results
console.log('Average Marks: ' + averageMarks.toFixed(2));
console.log('Grade: ' + grade);

// To display the results on the webpage, you can use document.write or append to the
DOM
document.write('Average Marks: ' + averageMarks.toFixed(2) + '<br>');
document.write('Grade: ' + grade);
```

7. **Write a JavaScript program which iterates the integers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".**

```
    for (let i = 1; i <= 100; i++) {
  if (i % 3 === 0 && i % 5 === 0) {
    console.log("FizzBuzz");
  } else if (i % 3 === 0) {
    console.log("Fizz");
  } else if (i % 5 === 0) {
    console.log("Buzz");
  } else {
    console.log(i);
  }
}
```

8. **. According to Wikipedia a happy number is defined by the following process : "Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers (or sad numbers)". Write a JavaScript program to find and print the first 5 happy numbers.**

```
    function sumOfSquares(num) {
```

```
    let sum = 0;
    while (num > 0) {
        let digit = num % 10;
        sum += digit * digit;
        num = Math.floor(num / 10);
    }
    return sum;
}

// Function to check if a number is a happy number
function isHappyNumber(num) {
    let slow = num;
    let fast = num;
    do {
        slow = sumOfSquares(slow); // Move slow by one step
        fast = sumOfSquares(sumOfSquares(fast)); // Move fast by two steps
    } while (slow !== fast);
    return slow === 1; // If both meet at 1, it is a happy number
}

// Function to find the first n happy numbers
function findHappyNumbers(n) {
    let count = 0;
    let number = 1;
    let happyNumbers = [];
    while (count < n) {
        if (isHappyNumber(number)) {
            happyNumbers.push(number);
            count++;
        }
        number++;
    }
    return happyNumbers;
}

// Find and print the first 5 happy numbers
let first5HappyNumbers = findHappyNumbers(5);
console.log("The first 5 happy numbers are: " + first5HappyNumbers.join(", "));
```

9. **Write a JavaScript program to find the armstrong numbers of 3 digits. Note : An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since 3**3 + 7**3 + 1**3 = 371.**

```
function isArmstrongNumber(num) {
    const digits = num.toString().split('').map(Number);
    const sumOfCubes = digits.reduce((sum, digit) => sum + Math.pow(digit, 3), 0);
    return sumOfCubes === num;
}

// Function to find all 3-digit Armstrong numbers
function findArmstrongNumbers() {
    const armstrongNumbers = [];
    for (let i = 100; i <= 999; i++) {
        if (isArmstrongNumber(i)) {
            armstrongNumbers.push(i);
        }
    }
    return armstrongNumbers;
}

// Find and print all 3-digit Armstrong numbers
const armstrongNumbers = findArmstrongNumbers();
console.log("3-digit Armstrong numbers are: " + armstrongNumbers.join(", "));
```

10. . **Write a JavaScript program to construct the following pattern, using a nested for loop.**

```
*
* *
* * *
* * * *
* * * * *
function constructPattern(rows) {
    for (let i = 1; i <= rows; i++) {
        let pattern = '';
        for (let j = 1; j <= i; j++) {
            pattern += '* ';
        }
```

```
        console.log(pattern);
    }
}

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

// Call the function to construct and display the pattern
constructPattern(numRows);
```

11. **Write a JavaScript program to compute the greatest common divisor (GCD) of two positive integers.**

```
        function gcd(a, b) {
    // Using iterative approach
    while (b !== 0) {
        let temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// Example usage:
const num1 = 24;
const num2 = 36;

// Calculate GCD of num1 and num2
const result = gcd(num1, num2);

console.log(`The GCD of ${num1} and ${num2} is: ${result}`);
```

12. **Write a JavaScript program to sum the multiples of 3 and 5 under 1000.**

```
        function sumMultiplesOf3And5(limit) {
    let sum = 0;
    for (let i = 1; i < limit; i++) {
        if (i % 3 === 0 || i % 5 === 0) {
            sum += i;
```

```
        }
    }
    return sum;
}

const limit = 1000;
const result = sumMultiplesOf3And5(limit);
console.log(`Sum of multiples of 3 and 5 under ${limit} is: ${result}`);
```

Javascript Array

1. **Write a JavaScript function to check whether an `input` is an array or not. Test Data : console.log(is_array('w3resource')); console.log(is_array([1, 2, 4, 0])); false true**

```
        function is_array(input) {
    return Array.isArray(input);
}

// Test Data
console.log(is_array('w3resource')); // false
console.log(is_array([1, 2, 4, 0])); // true
```

2. **2. Write a JavaScript function to clone an array. Test Data : console.log(array_Clone([1, 2, 4, 0])); console.log(array_Clone([1, 2, [4, 0]])); [1, 2, 4, 0] [1, 2, [4, 0]]**

```
        function array_Clone(arr) {
    // Check if input is an array
    if (!Array.isArray(arr)) {
        return null; // Return null if input is not an array
    }

    // Initialize empty array for the clone
    let clone = [];

    // Iterate through each element in the original array
    for (let i = 0; i < arr.length; i++) {
        // Check if the element is an array itself (for deep cloning)
```

```
        if (Array.isArray(arr[i])) {
            // Recursively clone nested arrays
            clone.push(array_Clone(arr[i]));
        } else {
            // Copy elements directly
            clone.push(arr[i]);
        }
    }

    return clone;
}

// Test Data
console.log(array_Clone([1, 2, 4, 0])); // [1, 2, 4, 0]
console.log(array_Clone([1, 2, [4, 0]])); // [1, 2, [4, 0]]
```

3. **. Write a JavaScript function to get the first element of an array. Passing a parameter 'n' will return the first 'n' elements of the array. Test Data : console.log(first([7, 9, 0, -2])); console.log(first([],3)); console.log(first([7, 9, 0, -2],3)); console.log(first([7, 9, 0, -2],6)); console.log(first([7, 9, 0, -2],-3)); Expected Output : 7 [] [7, 9, 0] [7, 9, 0, -2] []**

```
function first(arr, n = 1) {
    // Check if arr is not an array or n is not a valid number
    if (!Array.isArray(arr) || n <= 0) {
        return [];
    }

    // Return the first 'n' elements of the array using slice
    return arr.slice(0, n);
}

// Test Data
console.log(first([7, 9, 0, -2])); // Output: 7
console.log(first([], 3)); // Output: []
console.log(first([7, 9, 0, -2], 3)); // Output: [7, 9, 0]
console.log(first([7, 9, 0, -2], 6)); // Output: [7, 9, 0, -2]
console.log(first([7, 9, 0, -2], -3)); // Output: []
```

4. **Write a JavaScript function to get the last element of an array. Passing a parameter 'n' will return the last 'n' elements of the array. Test Data : console.log(last([7, 9, 0, -2])); console.log(last([7, 9, 0, -2],3)); console.log(last([7, 9, 0, -2],6)); Expected Output : -2 [9, 0, -2] [7, 9, 0, -2]**

```
function last(arr, n = 1) {
  // Check if arr is not an array or n is not a valid number
  if (!Array.isArray(arr) || n <= 0) {
    return [];
  }

  // Return the last 'n' elements of the array using slice
  return arr.slice(-n);
}

// Test Data
console.log(last([7, 9, 0, -2])); // Output: -2
console.log(last([7, 9, 0, -2], 3)); // Output: [9, 0, -2]
```

5. **Write a simple JavaScript program to join all elements of the following array into a string. Sample array : myColor = ["Red", "Green", "White", "Black"]; Expected Output : "Red,Green,White,Black" "Red,Green,White,Black" "Red+Green+White+Black"**

```
let myColor = ["Red", "Green", "White", "Black"];
let joinedWithComma = myColor.join(',');
console.log(joinedWithComma); // Output: "Red,Green,White,Black"
```

6. **Write a JavaScript program which accept a number as input and insert dashes (-) between each two even numbers. For example if you accept 025468 the output should be 0-254-6-8.**

```
function insertDashes(number) {
  // Convert number to string to iterate through each digit
  let str = number.toString();
  let result = [];

  // Iterate through each digit in the string
  for (let i = 0; i < str.length; i++) {
```

```
        // Check if the current digit and the next digit are both even
        if (parseInt(str[i]) % 2 === 0 && parseInt(str[i + 1]) % 2 === 0) {
            result.push(str[i], '-'); // Add current digit and dash to result array
        } else {
            result.push(str[i]); // Add current digit to result array
        }
    }

    // Join the result array into a string and return
    return result.join('');
}

// Test cases
console.log(insertDashes(025468)); // Output: "0-254-6-8"
console.log(insertDashes(13579)); // Output: "13579"
console.log(insertDashes(246802)); // Output: "2-4680-2"
```

7. **Write a JavaScript program to sort the items of an array. Sample array : var arr1 = [3, 8, 7, 6, 5, -4, 3, 2, 1]; Sample Output : -4,-3,1,2,3,5,6,7,8**

```
var arr1 = [3, 8, 7, 6, 5, -4, 3, 2, 1];

function sortArray(arr) {
    return arr.sort(function(a, b) {
        return a - b;
    });
}

var sortedArr = sortArray(arr1);
console.log(sortedArr); // Output: [-4, 1, 2, 3, 3, 5, 6, 7, 8]
```

8. **Write a JavaScript program to find the most frequent item of an array. Sample array : var arr1=[3, 'a', 'a', 'a', 2, 3, 'a', 3, 'a', 2, 4, 9, 3]; Sample Output : a (5 times)**

```
function findMostFrequent(arr) {
    var frequency = {}; // Object to store frequency of each element
    var maxFrequency = 0; // Variable to store the maximum frequency
    var mostFrequentItem; // Variable to store the most frequent item
```

```
    // Loop through the array and count the occurrences of each element
    for (var i = 0; i < arr.length; i++) {
        var item = arr[i];
        frequency[item] = (frequency[item] || 0) + 1;

        // Update the most frequent item if the current item's frequency is greater than
maxFrequency
        if (frequency[item] > maxFrequency) {
            maxFrequency = frequency[item];
            mostFrequentItem = item;
        }
    }

    // Return the most frequent item and its frequency
    return mostFrequentItem + " ( " + maxFrequency + " times )";
}

// Sample array
var arr1 = [3, 'a', 'a', 'a', 2, 3, 'a', 3, 'a', 2, 4, 9, 3];

// Find and print the most frequent item
console.log(findMostFrequent(arr1)); // Output: a ( 5 times )
```

9. **Write a JavaScript program which accept a string as input and swap the case of each character. For example if you input 'The Quick Brown Fox' the output should be 'tHE qUICK bROWN fOX'.**

```
        function swapCase(str) {
    var swappedStr = '';

    for (var i = 0; i < str.length; i++) {
        var char = str[i];
        if (char === char.toUpperCase()) {
            swappedStr += char.toLowerCase();
        } else {
            swappedStr += char.toUpperCase();
        }
    }
```

```
    return swappedStr;
}

// Test the function
var input = 'The Quick Brown Fox';
var output = swapCase(input);
console.log(output); // Output: tHE qUICK bROWN fOX
```

10. **Write a JavaScript program which prints the elements of the following array. Note : Use nested for loops. Sample array : var a = [[1, 2, 1, 24], [8, 11, 9, 4], [7, 0, 7, 27], [7, 4, 28, 14], [3, 10, 26, 7]]; Sample Output : "row 0" " 1" " 2" " 1" " 24" "row 1"**

```
        var a = [
    [1, 2, 1, 24],
    [8, 11, 9, 4],
    [7, 0, 7, 27],
    [7, 4, 28, 14],
    [3, 10, 26, 7]
];

function printArray(arr) {
    for (var i = 0; i < arr.length; i++) {
        console.log("row " + i);
        for (var j = 0; j < arr[i].length; j++) {
            console.log(" " + arr[i][j]);
        }
    }
}

// Test the function
printArray(a);
```

11. **Write a JavaScript program to find the sum of squares of a numeric vector.**

```
        function sumOfSquares(vector) {
    return vector.reduce((sum, num) => sum + num ** 2, 0);
}
```

```
// Test the function
const vector = [1, 2, 3, 4, 5];
const result = sumOfSquares(vector);
console.log(result); // Output: 55
```

12. **Write a JavaScript program to compute the sum and product of an array of integers.**

```
function sumAndProduct(arr) {
    return arr.reduce(
        (acc, num) => {
            acc.sum += num;
            acc.product *= num;
            return acc;
        },
        { sum: 0, product: 1 }
    );
}

// Test the function
const numbers = [1, 2, 3, 4, 5];
const result = sumAndProduct(numbers);
console.log(`Sum: ${result.sum}, Product: ${result.product}`); // Output: Sum: 15,
Product: 120
```

13. **Write a JavaScript program to add items in an blank array and display the items.**

```
<head>
    <title>Add Items to Array</title>
</head>
<body>
    <h1>JavaScript Add Items to Array</h1>
    <button onclick="addItem()">Add Item</button>
    <p id="arrayDisplay"></p>

    <script>
        // Initialize an empty array
        let itemsArray = [];
```

```
// Function to add an item to the array
function addItem() {
    let item = prompt("Enter an item to add to the array:");
    if (item !== null && item !== '') {
        itemsArray.push(item);
        displayArray();
    }
}

// Function to display the array
```

14. **Write a JavaScript program to remove duplicate items from an array (ignore case sensitivity).**

```
    function removeDuplicates(arr) {
  // Create a Set to store unique values
  let uniqueSet = new Set();

  // Convert all items to lowercase and add to the Set
  arr.forEach(item => {
    uniqueSet.add(item.toLowerCase());
  });

  // Convert the Set back to an array
  return Array.from(uniqueSet);
}

// Example usage
let array = ["Apple", "Banana", "apple", "Orange", "banana"];
let uniqueArray = removeDuplicates(array);

console.log(uniqueArray); // Output: [ 'apple', 'banana', 'orange' ]
```

15. **. We have the following arrays : color = ["Blue ", "Green", "Red", "Orange", "Violet", "Indigo", "Yellow "]; o = ["th","st","nd","rd"] Write a JavaScript program to display the colors in the following way : "1st choice is Blue ." "2nd choice is Green." "3rd choice is Red.**

```
        let colors = ["Blue ", "Green", "Red", "Orange", "Violet", "Indigo", "Yellow "];
let suffixes = ["th", "st", "nd", "rd"];

function getSuffix(index) {
  if (index > 3) return suffixes[0]; // for 4th, 5th, etc.
  return suffixes[index]; // for 1st, 2nd, 3rd
}

colors.forEach((color, index) => {
  let suffix = getSuffix(index + 1);
  console.log(`${index + 1}${suffix} choice is ${color.trim()}.`);
});
```

16.. Find the leap years in a given range of years.

```
        function findLeapYears(startYear, endYear) {
  let leapYears = [];

  for (let year = startYear; year <= endYear; year++) {
    if ((year % 4 === 0 && year % 100 !== 0) || (year % 400 === 0)) {
      leapYears.push(year);
    }
  }

  return leapYears;
}

// Example usage
let startYear = 2000;
let endYear = 2024;
let leapYearsInRange = findLeapYears(startYear, endYear);

console.log(`Leap years between ${startYear} and ${endYear} are:
${leapYearsInRange.join(', ')}`);
```

17. Write a JavaScript program to shuffle an array.

```
        function shuffleArray(array) {
  // Loop through the array starting from the last element
```

```
  for (let i = array.length - 1; i > 0; i--) {
    // Pick a random index from 0 to i
    const j = Math.floor(Math.random() * (i + 1));

    // Swap elements array[i] and array[j]
    [array[i], array[j]] = [array[j], array[i]];
  }
  return array;
}

// Example usage
let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9];
console.log("Original array:", arr);
let shuffledArray = shuffleArray(arr);
console.log("Shuffled array:", shuffledArray);
```

18. . Write a JavaScript program to perform a binary search. Note : A binary search or half-interval search algorithm finds the position of a specified input value within an array sorted by key value. Sample array : var items = [1, 2, 3, 4, 5, 7, 8, 9]; Expected Output : console.log(binary_Search(items, 1)); //0 console.log(binary_Search(items, 5)); //4

```
        function binarySearch(arr, target) {
  let left = 0;
  let right = arr.length - 1;

  while (left <= right) {
    let middle = Math.floor((left + right) / 2);

    if (arr[middle] === target) {
      return middle;
    } else if (arr[middle] < target) {
      left = middle + 1;
    } else {
      right = middle - 1;
    }
  }

  // Return -1 if the target is not found in the array
  return -1;
```

```
}

// Sample array
let items = [1, 2, 3, 4, 5, 7, 8, 9];

// Expected Output
console.log(binarySearch(items, 1)); // 0
console.log(binarySearch(items, 5)); // 4
```

19. **There are two arrays with individual values, write a JavaScript program to compute the sum of each individual index value from the given arrays. Sample array : array1 = [1,0,2,3,4]; array2 = [3,5,6,7,8,13]; Expected Output : [4, 5, 8, 10, 12, 13]**

```
function sumArrays(array1, array2) {
  let maxLength = Math.max(array1.length, array2.length);
  let resultArray = [];

  for (let i = 0; i < maxLength; i++) {
    let value1 = array1[i] !== undefined ? array1[i] : 0;
    let value2 = array2[i] !== undefined ? array2[i] : 0;
    resultArray.push(value1 + value2);
  }

  return resultArray;
}

// Sample arrays
let array1 = [1, 0, 2, 3, 4];
let array2 = [3, 5, 6, 7, 8, 13];

// Expected Output
console.log(sumArrays(array1, array2)); // [4, 5, 8, 10, 12, 13]
```

20. **Write a JavaScript program to find duplicate values in a JavaScript array.**

```
function findDuplicates(arr) {
  let seen = new Set();
```

```
  let duplicates = new Set();

  arr.forEach(item => {
    if (seen.has(item)) {
      duplicates.add(item);
    } else {
      seen.add(item);
    }
  });

  return Array.from(duplicates);
}

// Sample array
let array = [1, 2, 3, 4, 5, 2, 3, 6, 7, 8, 1];

// Expected Output
console.log(findDuplicates(array)); // [1, 2, 3]


        function findDuplicates(arr) {
  let seen = new Set();
  let duplicates = new Set();

  arr.forEach(item => {
    if (seen.has(item)) {
      duplicates.add(item);
    } else {
      seen.add(item);
    }
  });

  return Array.from(duplicates);
}

// Sample array
let array = [1, 2, 3, 4, 5, 2, 3, 6, 7, 8, 1];

// Expected Output
console.log(findDuplicates(array)); // [1, 2, 3]
```

21. **Write a JavaScript program to flatten a nested (any depth) array. If you pass shallow, the array will only be flattened a single level. Sample Data : console.log(flatten([1, [2], [3, [[4]]],[5,6]])); [1, 2, 3, 4, 5, 6] console.log(flatten([1, [2], [3, [[4]]],[5,6]], true)); [1, 2, 3, [[4]], 5, 6]**

```
function flatten(array, shallow = false) {
  if (shallow) {
    // Flatten only one level
    return array.reduce((acc, val) => acc.concat(val), []);
  } else {
    // Flatten all levels
    return array.reduce((acc, val) =>
      Array.isArray(val) ? acc.concat(flatten(val)) : acc.concat(val), []);
  }
}

// Sample Data
console.log(flatten([1, [2], [3, [[4]]], [5, 6]]));
// [1, 2, 3, 4, 5, 6]
console.log(flatten([1, [2], [3, [[4]]], [5, 6]], true));
// [1, 2, 3, [[4]], 5, 6]
```

22. **Write a JavaScript program to compute the union of two arrays. Sample Data : console.log(union([1, 2, 3], [100, 2, 1, 10])); [1, 2, 3, 10, 100]**

```
function union(arr1, arr2) {
  let set = new Set([...arr1, ...arr2]);
  return Array.from(set);
}

// Sample Data
console.log(union([1, 2, 3], [100, 2, 1, 10]));
// [1, 2, 3, 100, 10]
```

23. **Write a JavaScript function to find the difference of two arrays. Test Data : console.log(difference([1, 2, 3], [100, 2, 1, 10])); ["3", "10", "100"] console.log(difference([1, 2, 3, 4, 5], [1, [2], [3, [[4]]],[5,6]])); ["6"] console.log(difference([1, 2, 3], [100, 2, 1, 10])); ["3", "10", "100"]**

```
function difference(arr1, arr2) {
  // Flatten the arrays to handle nested arrays
  const flatten = arr => arr.reduce((acc, val) =>
    Array.isArray(val) ? acc.concat(flatten(val)) : acc.concat(val), []);

  // Flatten both arrays
  arr1 = flatten(arr1);
  arr2 = flatten(arr2);

  // Create a set for the second array
  const set2 = new Set(arr2);

  // Find elements in arr1 that are not in arr2
  const diff1 = arr1.filter(x => !set2.has(x));

  // Create a set for the first array
  const set1 = new Set(arr1);

  // Find elements in arr2 that are not in arr1
  const diff2 = arr2.filter(x => !set1.has(x));

  // Combine the differences and remove duplicates
  const differenceArray = Array.from(new Set([...diff1, ...diff2]));

  return differenceArray;
}

// Test Data
console.log(difference([1, 2, 3], [100, 2, 1, 10]));
// ["3", "10", "100"]

console.log(difference([1, 2, 3, 4, 5], [1, [2], [3, [[4]]], [5, 6]]));
// ["6"]

console.log(difference([1, 2, 3], [100, 2, 1, 10]));
```

```
// ["3", "10", "100"]


        function difference(arr1, arr2) {
  // Flatten the arrays to handle nested arrays
  const flatten = arr => arr.reduce((acc, val) =>
    Array.isArray(val) ? acc.concat(flatten(val)) : acc.concat(val), []);

  // Flatten both arrays
  arr1 = flatten(arr1);
  arr2 = flatten(arr2);

  // Create a set for the second array
  const set2 = new Set(arr2);

  // Find elements in arr1 that are not in arr2
  const diff1 = arr1.filter(x => !set2.has(x));

  // Create a set for the first array
  const set1 = new Set(arr1);

  // Find elements in arr2 that are not in arr1
  const diff2 = arr2.filter(x => !set1.has(x));

  // Combine the differences and remove duplicates
  const differenceArray = Array.from(new Set([...diff1, ...diff2]));

  return differenceArray;
}

// Test Data
console.log(difference([1, 2, 3], [100, 2, 1, 10]));
// ["3", "10", "100"]

console.log(difference([1, 2, 3, 4, 5], [1, [2], [3, [[4]]], [5, 6]]));
// ["6"]

console.log(difference([1, 2, 3], [100, 2, 1, 10]));
// ["3", "10", "100"]
```

24. **Write a JavaScript function to remove. 'null', '0', '""', 'false', 'undefined' and 'NaN' values from an array. Sample array : [NaN, 0, 15, false, -22, '',undefined, 47, null] Expected result : [15, -22, 47]**

```
function removeFalsyValues(arr) {
  return arr.filter(value =>
    value !== null &&
    value !== 0 &&
    value !== "" &&
    value !== false &&
    value !== undefined &&
    !Number.isNaN(value)
  );
}

// Sample array
let sampleArray = [NaN, 0, 15, false, -22, '', undefined, 47, null];

// Expected result
console.log(removeFalsyValues(sampleArray));
// [15, -22, 47]
```

25. **Write a JavaScript function to sort the following array of objects by title value. Sample object : var library = [{ author: 'Bill Gates', title: 'The Road Ahead', libraryID: 1254}, { author: 'Steve Jobs', title: 'Walter Isaacson', libraryID: 4264}, { author: 'Suzanne Collins', title: 'Mockingjay: The Final Book of The Hunger Games ', libraryID: 3245}]; Expected result : [[object Object] { author: "Suzanne Collins", libraryID: 3245, title: "Mockingjay: The Final Book of The Hunger Games"**

```
function sortByTitle(library) {
  return library.sort((a, b) => {
    // Compare the title values
    let titleA = a.title.toUpperCase(); // Ignore upper and lowercase
    let titleB = b.title.toUpperCase(); // Ignore upper and lowercase

    if (titleA < titleB) {
```

```
      return -1;
    }
    if (titleA > titleB) {
      return 1;
    }

    // titles are equal
    return 0;
  });
}

// Sample object
var library = [
  { author: 'Bill Gates', title: 'The Road Ahead', libraryID: 1254 },
  { author: 'Steve Jobs', title: 'Walter Isaacson', libraryID: 4264 },
  { author: 'Suzanne Collins', title: 'Mockingjay: The Final Book of The Hunger Games',
libraryID: 3245 }
];

// Expected result
console.log(sortByTitle(library));
/*
[
  {
    author: 'Suzanne Collins',
    title: 'Mockingjay: The Final Book of The Hunger Games',
    libraryID: 3245
  },
  {
    author: 'Bill Gates',
    title: 'The Road Ahead',
    libraryID: 1254
  },
  {
    author: 'Steve Jobs',
    title: 'Walter Isaacson',
    libraryID: 4264
  }
]
*/
```

26. Write a JavaScript program to find a pair of elements (indices of the two numbers) from an given array whose sum equals a specific target number. Input: numbers= [10,20,10,40,50,60,70], target=50 Output: 3, 4

```
        function findPairWithSum(numbers, target) {
  let indicesMap = new Map();

  for (let i = 0; i < numbers.length; i++) {
    let complement = target - numbers[i];

    if (indicesMap.has(complement)) {
      return [indicesMap.get(complement), i];
    }

    indicesMap.set(numbers[i], i);
  }

  return null; // Return null if no pair is found
}

// Input
let numbers = [10, 20, 10, 40, 50, 60, 70];
let target = 50;

// Output
let result = findPairWithSum(numbers, target);
if (result) {
  console.log(`Indices: ${result[0]}, ${result[1]}`);
} else {
  console.log("No pair found");
}
// Expected Output: Indices: 3, 4
```

27.. Write a JavaScript function to retrieve the value of a given property from all elements in an array. Sample array : [NaN, 0, 15, false, -22, '',undefined, 47, null] Expected result : [15, -22, 47]

```
function filterTruthyValues(arr) {
  return arr.filter(value => value);
}

// Sample array
let sampleArray = [NaN, 0, 15, false, -22, '', undefined, 47, null];

// Expected result
console.log(filterTruthyValues(sampleArray));
// [15, -22, 47]
```

28. . Write a JavaScript function to find the longest common starting substring in a set of strings. Sample array : console.log(longest_common_starting_substring(['go', 'google'])); Expected result : "go"

```
function longestCommonStartingSubstring(arr) {
  if (arr.length === 0) return '';
  if (arr.length === 1) return arr[0];

  let commonPrefix = arr[0];

  for (let i = 1; i < arr.length; i++) {
    while (arr[i].indexOf(commonPrefix) !== 0) {
      commonPrefix = commonPrefix.substring(0, commonPrefix.length - 1);
      if (commonPrefix === '') return '';
    }
  }

  return commonPrefix;
}
```

29. Write a JavaScript function to fill an array with values (numeric, string with one character) on supplied bounds. Test Data : console.log(num_string_range('a', "z", 2)); ["a", "c", "e", "g", "i", "k", "m", "o", "q", "s", "u", "w", "y"]

```
function numStringRange(start, end, step) {
  let result = [];
```

```
  let isNumber = !isNaN(start) && !isNaN(end);

  if (isNumber) {
    start = Number(start);
    end = Number(end);
    for (let i = start; i <= end; i += step) {
      result.push(i);
    }
  } else {
    start = start.charCodeAt(0);
    end = end.charCodeAt(0);
    for (let i = start; i <= end; i += step) {
      result.push(String.fromCharCode(i));
    }
  }

  return result;
}

// Test Data
console.log(numStringRange('a', 'z', 2));
// ["a", "c", "e", "g", "i", "k", "m", "o", "q", "s", "u", "w", "y"]

console.log(numStringRange(1, 10, 2));
// [1, 3, 5, 7, 9]

console.log(numStringRange('A', 'Z', 2));
// ["A", "C", "E", "G", "I", "K", "M", "O", "Q", "S", "U", "W", "Y"]
```

30. Write a JavaScript function to merge two arrays and removes all duplicates elements. Test data : var array1 = [1, 2, 3]; var array2 = [2, 30, 1]; console.log(merge_array(array1, array2)); [3, 2, 30, 1]

```
        function mergeArrays(array1, array2) {
  let mergedArray = [...array1, ...array2];

  // Use Set to remove duplicates and spread syntax to convert back to array
  let uniqueArray = [...new Set(mergedArray)];
```

```
  return uniqueArray;
}

// Test data
var array1 = [1, 2, 3];
var array2 = [2, 30, 1];
console.log(mergeArrays(array1, array2));
// Expected output: [3, 2, 30, 1]
```

31. . Write a JavaScript function to remove a specific element from an array. Test data : console.log(remove_array_element([2, 5, 9, 6], 5)); [2, 9, 6]

```
function removeArrayElement(arr, element) {
  return arr.filter(item => item !== element);
}

// Test data
console.log(removeArrayElement([2, 5, 9, 6], 5));
// Expected output: [2, 9, 6]
```

32. Write a JavaScript function to find an array contains a specific element. Test data : console.log(remove_array_element([2, 5, 9, 6], 5)); [2, 9, 6]

```
function arrayContainsElement(arr, element) {
  return arr.includes(element);
}

// Test data
console.log(arrayContainsElement([2, 5, 9, 6], 5)); // true
console.log(arrayContainsElement([2, 5, 9, 6], 3)); // false
```

33. Write a JavaScript script to empty an array keeping the original.

```
function emptyArray(arr) {
  arr.length = 0;
```

```
}

// Test data
let originalArray = [1, 2, 3, 4, 5];
console.log("Original array:", originalArray);

// Empty the array
emptyArray(originalArray);

// Check the result
console.log("Emptied array:", originalArray);

        function emptyArray(arr) {
  arr.length = 0;
}

// Test data
let originalArray = [1, 2, 3, 4, 5];
console.log("Original array:", originalArray);

// Empty the array
emptyArray(originalArray);

// Check the result
console.log("Emptied array:", originalArray);
```

34. **Write a JavaScript function to get nth largest element from an unsorted array. Test Data : console.log(nthlargest([43, 56, 23, 89, 88, 90, 99, 652], 4)); 89**

```
        function nthLargest(arr, n) {
  // Sort the array in descending order
  arr.sort((a, b) => b - a);

  // Return the nth largest element
  return arr[n - 1];
}

// Test data
console.log(nthLargest([43, 56, 23, 89, 88, 90, 99, 652], 4));
```

```
// Expected output: 89
```

35. **Write a JavaScript function to get a random item from an array.**

```
function getRandomItem(arr) {
  // Generate a random index
  const randomIndex = Math.floor(Math.random() * arr.length);

  // Return the random item
  return arr[randomIndex];
}

// Test data
const array = [1, 2, 3, 4, 5];

// Get a random item from the array
console.log(getRandomItem(array));
```

36. **Write a JavaScript function to create a specified number of elements with pre-filled numeric value array. Test Data : console.log(array_filled(6, 0)); [0, 0, 0, 0, 0, 0] console.log(array_filled(4, 11)); [11, 11, 11, 11]**

```
function arrayFilled(length, value) {
  // Create an array with 'length' elements, filled with 'value'
  return Array.from({ length: length }, () => value);
}

// Test data
console.log(arrayFilled(6, 0)); // [0, 0, 0, 0, 0, 0]
console.log(arrayFilled(4, 11)); // [11, 11, 11, 11]
```

37. **Write a JavaScript function to create a specified number of elements with pre-filled string value array. Test Data : console.log(array_filled(3, 'default value')); ["default value", "default value", "default value"] console.log(array_filled(4, 'password')); ["password", "password", "password", "password"]**

```
        function arrayFilled(length, value) {
  // Create an array with 'length' elements, filled with 'value'
  return Array.from({ length: length }, () => value);
}

// Test data
console.log(arrayFilled(3, 'default value'));
// Expected output: ["default value", "default value", "default value"]

console.log(arrayFilled(4, 'password'));
// Expected output: ["password", "password", "password", "password"]
```

38. **Write a JavaScript function to move an array element from one position to another. Test Data : console.log(move([10, 20, 30, 40, 50], 0, 2)); [20, 30, 10, 40, 50] console.log(move([10, 20, 30, 40, 50], -1, -2)); [10, 20, 30, 50, 40]**

```
        function move(array, fromIndex, toIndex) {
  // Check if any of the indices are out of bounds
  if (fromIndex < 0 || fromIndex >= array.length || toIndex < 0 || toIndex >= array.length) {
    console.log("Invalid indices provided.");
    return array; // Return the original array if indices are invalid
  }

  // Remove the element from 'fromIndex' position
  const element = array.splice(fromIndex, 1)[0];

  // Insert the element at 'toIndex' position
  array.splice(toIndex, 0, element);

  return array;
}

// Test data
console.log(move([10, 20, 30, 40, 50], 0, 2));
// Expected output: [20, 30, 10, 40, 50]

console.log(move([10, 20, 30, 40, 50], -1, -2));
// Expected output: [10, 20, 30, 50, 40]
```

39. **Write a JavaScript function to filter false, null, 0 and blank values from an array. Test Data : console.log(filter_array_values([58, '', 'abcd', true, null, false, 0])); [58, "abcd", true]**

```
function filterArrayValues(arr) {
  return arr.filter(item => {
    // Filter out false, null, 0, and empty strings
    return item !== false && item !== null && item !== 0 && item !== '';
  });
}

// Test data
console.log(filterArrayValues([58, '', 'abcd', true, null, false, 0]));
// Expected output: [58, "abcd", true]
```

40. **Write a JavaScript function to generate an array of specified length, filled with integer numbers, increase by one from starting position. Test Data : console.log(array_range(1, 4)); [1, 2, 3, 4] console.log(array_range(-6, 4)); [-6, -5, -4, -3]**

```
function arrayRange(start, length) {
  // Initialize an empty array to store the result
  let result = [];

  // Loop to fill the array with numbers starting from 'start'
  for (let i = 0; i < length; i++) {
    result.push(start + i);
  }

  return result;
}

// Test data
console.log(arrayRange(1, 4)); // [1, 2, 3, 4]
console.log(arrayRange(-6, 4)); // [-6, -5, -4, -3]
```

41. **Write a JavaScript function to generate an array between two integers of 1 step length. Test Data : console.log(rangeBetwee(4, 7));**

[4, 5, 6, 7] console.log(rangeBetwee(-4, 7)); [-4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7]

```
function rangeBetween(start, end) {
  // Determine the direction of the range (increment or decrement)
  const step = start <= end ? 1 : -1;

  // Initialize an empty array to store the result
  let result = [];

  // Loop to fill the array with numbers between 'start' and 'end'
  for (let i = start; i !== end + step; i += step) {
    result.push(i);
  }

  return result;
}

// Test data
console.log(rangeBetween(4, 7)); // [4, 5, 6, 7]
console.log(rangeBetween(-4, 7)); // [-4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
```

42. **Write a JavaScript function to find the unique elements from two arrays. Test Data : console.log(difference([1, 2, 3], [100, 2, 1, 10])); ["1", "2", "3", "10", "100"] console.log(difference([1, 2, 3, 4, 5], [1, [2], [3, [[4]]],[5,6]])); ["1", "2", "3", "4", "5", "6"] console.log(difference([1, 2, 3], [100, 2, 1, 10])); ["1", "2", "3", "10", "100"]**

```
function difference(arr1, arr2) {
  // Concatenate both arrays and flatten any nested arrays
  const combinedArray = [...new Set([...arr1, ...flatten(arr2)])];

  // Convert all elements to strings
  const uniqueStrings = combinedArray.map(item => String(item));

  return uniqueStrings;
}

// Function to flatten a nested array (helper function)
```

```
function flatten(arr) {
  return arr.reduce((acc, val) => Array.isArray(val) ? acc.concat(flatten(val)) :
acc.concat(val), []);
}

// Test data
console.log(difference([1, 2, 3], [100, 2, 1, 10]));
// Expected output: ["1", "2", "3", "10", "100"]
```

JAVASCRIPT DATE

1. **Write a JavaScript function to check whether an `input` is a date object or not. Test Data : console.log(is_date("October 13, 2014 11:13:00")); console.log(is_date(new Date(86400000))); console.log(is_date(new Date(99,5,24,11,33,30,0))); console.log(is_date([1, 2, 4, 0])); Output : false true true false**

```
function isDate(input) {
  // Check if input is an instance of Date
  return input instanceof Date && !isNaN(input);
}

// Test data
console.log(isDate("October 13, 2014 11:13:00")); // false
console.log(isDate(new Date(86400000))); // true
console.log(isDate(new Date(99, 5, 24, 11, 33, 30, 0))); // true
console.log(isDate([1, 2, 4, 0])); // false
```

2. **2. Write a JavaScript function to get the current date. Note : Pass a separator as an argument. Test Data : console.log(curday('/')); console.log(curday('-')); Output : "11/13/2014" "11-13-2014"**

```
function curday(separator) {
  // Create a new Date object
  let today = new Date();

  // Extract day, month, and year from the Date object
  let day = today.getDate();
```

```
  let month = today.getMonth() + 1; // January is 0, so we add 1
  let year = today.getFullYear();

  // Format the date based on the separator provided
  return `${month}${separator}${day}${separator}${year}`;
}

// Test data
console.log(curday('/')); // "11/13/2014"
console.log(curday('-')); // "11-13-2014"
```

```
        function curday(separator) {
  // Create a new Date object
  let today = new Date();

  // Extract day, month, and year from the Date object
  let day = today.getDate();
  let month = today.getMonth() + 1; // January is 0, so we add 1
  let year = today.getFullYear();

  // Format the date based on the separator provided
  return `${month}${separator}${day}${separator}${year}`;
}

// Test data
console.log(curday('/')); // "11/13/2014"
console.log(curday('-')); // "11-13-2014"
```

3. **. Write a JavaScript function to get the number of days in a month. Test Data : console.log(getDaysInMonth(1, 2012)); console.log(getDaysInMonth(2, 2012)); console.log(getDaysInMonth(9, 2012)); console.log(getDaysInMonth(12, 2012));**

```
        function getDaysInMonth(month, year) {
  // Adjust the month (JavaScript months are 0-11)
  // Create a new date for the first day of the next month and subtract one day
  return new Date(year, month, 0).getDate();
}
```

```
// Test data
console.log(getDaysInMonth(1, 2012)); // January 2012 -> 31 days
console.log(getDaysInMonth(2, 2012)); // February 2012 -> 29 days (leap year)
console.log(getDaysInMonth(9, 2012)); // September 2012 -> 30 days
console.log(getDaysInMonth(12, 2012)); // December 2012 -> 31 days
```

4. **. Write a JavaScript function to get the month name from a particular date. Test Data : console.log(month_name(new Date("10/11/2009"))); console.log(month_name(new Date("11/13/2014"))); Output : "October" "November**

```
function month_name(date) {
  // Create an array of month names
  const monthNames = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
  ];

  // Extract the month index from the provided date object
  const monthIndex = date.getMonth();

  // Return the month name based on the index
  return monthNames[monthIndex];
}

// Test data
console.log(month_name(new Date("10/11/2009"))); // "October"
console.log(month_name(new Date("11/13/2014"))); // "November"
```

5. **Write a JavaScript function to compare dates (i.e. greater than, less than or equal to). Test Data : console.log(compare_dates(new Date('11/14/2013 00:00'), new Date('11/14/2013 00:00'))); console.log(compare_dates(new Date('11/14/2013 00:01'), new Date('11/14/2013 00:00'))); console.log(compare_dates(new Date('11/14/2013 00:00'), new Date('11/14/2013 00:01'))); Output : "Date1 = Date2" "Date1 > Date2" "Date2 > Date1"**

```
function compare_dates(date1, date2) {
  // Compare the timestamps of the dates
  if (date1.getTime() === date2.getTime()) {
    return "Date1 = Date2";
  } else if (date1.getTime() > date2.getTime()) {
    return "Date1 > Date2";
  } else {
    return "Date2 > Date1";
  }
}

// Test data
console.log(compare_dates(new Date('11/14/2013 00:00'), new Date('11/14/2013
00:00')));
console.log(compare_dates(new Date('11/14/2013 00:01'), new Date('11/14/2013
00:00')));
console.log(compare_dates(new Date('11/14/2013 00:00'), new Date('11/14/2013
00:01')));
```

6. **Write a JavaScript function to add specified minutes to a Date object. Test Data : console.log(add_minutes(new Date(2014,10,2), 30).toString()); Output : "Sun Nov 02 2014 00:30:00 GMT+0530 (India Standard Time)"**

```
function add_minutes(date, minutes) {
  // Copy the original date to avoid modifying the original object
  let newDate = new Date(date);

  // Add specified minutes to the date
  newDate.setMinutes(newDate.getMinutes() + minutes);

  return newDate;
}

// Test data
console.log(add_minutes(new Date(2014, 10, 2), 30).toString());
```

7. **Write a JavaScript function to test whether a date is a weekend. Note : Use standard Saturday/Sunday definition of a weekend. Test Data :**

console.log(is_weekend('Nov 15, 2014')); console.log(is_weekend('Nov 16, 2014')); console.log(is_weekend('Nov 17, 2014')); Output : "weekend"

```
function is_weekend(dateString) {
  // Create a Date object from the dateString
  let date = new Date(dateString);

  // Get the day of the week (0 = Sunday, 6 = Saturday)
  let day = date.getDay();

  // Check if the day is Saturday (6) or Sunday (0)
  if (day === 6 || day === 0) {
    return "weekend";
  } else {
    return "weekday";
  }
}

// Test data
console.log(is_weekend('Nov 15, 2014')); // "weekend"
console.log(is_weekend('Nov 16, 2014')); // "weekend"
console.log(is_weekend('Nov 17, 2014')); // "weekday"
```

8. **Write a JavaScript function to get difference between two dates in days. Test Data : console.log(date_diff_indays('04/02/2014', '11/04/2014')); console.log(date_diff_indays('12/02/2014', '11/04/2014')); Output : 216 -28**

```
function date_diff_indays(date1, date2) {
  // Convert both dates to Date objects
  let dt1 = new Date(date1);
  let dt2 = new Date(date2);

  // Calculate the difference in milliseconds
  let diffMs = Math.abs(dt1 - dt2);

  // Convert milliseconds to days
  const oneDayMs = 1000 * 60 * 60 * 24;
  let diffDays = Math.floor(diffMs / oneDayMs);
```

```
  return diffDays;
}

// Test data
console.log(date_diff_indays('04/02/2014', '11/04/2014')); // 216 days
console.log(date_diff_indays('12/02/2014', '11/04/2014')); // -28 days
```

9. **Write a JavaScript function to get the last day of a month. Test Data : console.log(lastday(2014,0)); console.log(lastday(2014,1)); console.log(lastday(2014,11)); Output : 31 28 31**

```
function lastday(year, month) {
  // Create a new date for the next month's first day (month + 1) and set the date to 0
  let lastDay = new Date(year, month + 1, 0).getDate();

  return lastDay;
}

// Test data
console.log(lastday(2014, 0)); // January 2014 -> 31 days
console.log(lastday(2014, 1)); // February 2014 -> 28 days (non-leap year)
console.log(lastday(2014, 11)); // December 2014 -> 31 daysa
```

10. **Write a JavaScript function to calculate 'yesterday day'. Test Data : console.log(yesterday('Nov 15, 2014')); console.log(yesterday('Nov 16, 2015')); console.log(yesterday('Nov 17, 2016')); Output : "Fri Nov 14 2014 00:00:00 GMT+0530 (India Standard Time)" "Sun Nov 15 2015 00:00:00 GMT+0530 (India Standard Time)" "Wed Nov 16 2016 00:00:00 GMT+0530 (India Standard Time)"**

```
function yesterday(dateString) {
  // Create a Date object from the dateString
  let date = new Date(dateString);

  // Subtract one day in milliseconds (86400000 milliseconds = 1 day)
  let yesterdayDate = new Date(date.getTime() - 86400000);

  return yesterdayDate.toString();
}
```

```
// Test data
console.log(yesterday('Nov 15, 2014'));
console.log(yesterday('Nov 16, 2015'));
console.log(yesterday('Nov 17, 2016'));
```

11. **Write a JavaScript function to get the maximum date from an array of dates.**
    Test Data : console.log(max_date(['2015/02/01', '2015/02/02', '2015/01/03']));
    Output : "2015/02/02"

```
function max_date(dateArray) {
  // Convert each date string to a Date object and map them
  let dates = dateArray.map(dateString => new Date(dateString));

  // Find the maximum date using the spread operator and Math.max
  let maxDate = new Date(Math.max(...dates));

  // Format the maximum date to match the input format (YYYY/MM/DD)
  let formattedMaxDate = maxDate.toISOString().slice(0, 10);

  return formattedMaxDate;
}

// Test data
console.log(max_date(['2015/02/01', '2015/02/02', '2015/01/03'])); // "2015/02/02"
```

12. **Write a JavaScript function to get the minimum date from an array of dates.**
    Test Data : console.log(min_date(['2015/02/01', '2015/02/02', '2015/01/03']));
    Output : "2015/01/03"

```
function min_date(dateArray) {
  // Convert each date string to a Date object and map them
  let dates = dateArray.map(dateString => new Date(dateString));

  // Find the minimum date using the spread operator and Math.min
  let minDate = new Date(Math.min(...dates));

  // Format the minimum date to match the input format (YYYY/MM/DD)
  let formattedMinDate = minDate.toISOString().slice(0, 10);
```

```
  return formattedMinDate;
}

// Test data
console.log(min_date(['2015/02/01', '2015/02/02', '2015/01/03'])); // "2015/01/03"
```

13. **Write a JavaScript function that will return the number of minutes in hours and minutes. Test Data : console.log(timeConvert(200)); Output : "200 minutes = 3 hour(s) and 20 minute(s)."**

```
function timeConvert(minutes) {
  // Calculate hours and minutes
  let hours = Math.floor(minutes / 60);
  let mins = minutes % 60;

  // Construct the result string
  let result = `${minutes} minutes = ${hours} hour(s) and ${mins} minute(s).`;

  return result;
}

// Test data
console.log(timeConvert(200)); // "200 minutes = 3 hour(s) and 20 minute(s)."
```

14. **Write a JavaScript function to get the amount of days of a year. Test Data : console.log(days_of_a_year(2015)); 365 console.log(days_of_a_year(2016)); 366**

```
function days_of_a_year(year) {
  // Check if the year is a leap year
  if ((year % 4 === 0 && year % 100 !== 0) || year % 400 === 0) {
    return 366; // Leap year has 366 days
  } else {
    return 365; // Regular year has 365 days
  }
}

// Test data
console.log(days_of_a_year(2015)); // 365
console.log(days_of_a_year(2016)); // 366
```

15. **Write a JavaScript function to get the quarter (1 to 4) of the year. Test Data : console.log(quarter_of_the_year(new Date(2015, 1, 21))); 2 console.log(quarter_of_the_year(new Date(2015, 10, 18))); 4**

```
function quarter_of_the_year(date) {
  // Get the month from the date (0-11)
  let month = date.getMonth();

  // Calculate the quarter based on the month
  switch (Math.floor(month / 3)) {
    case 0:
      return 1; // January - March is Q1
    case 1:
      return 2; // April - June is Q2
    case 2:
      return 3; // July - September is Q3
    case 3:
      return 4; // October - December is Q4
    default:
      return -1; // Error case
  }
}

// Test data
console.log(quarter_of_the_year(new Date(2015, 1, 21))); // 2 (Q2)
console.log(quarter_of_the_year(new Date(2015, 10, 18))); // 4 (Q4)
```

16. . **Write a JavaScript function to count the number of days passed since beginning of the year. Test Data : console.log(days_passed(new Date(2015, 0, 15))); 15 console.log(days_passed(new Date(2015, 11, 14))); 348**

```
function days_passed(date) {
  // January 1st of the current year
  let startOfYear = new Date(date.getFullYear(), 0, 1);

  // Calculate the difference in milliseconds between the given date and start of the year
  let difference = date - startOfYear;
```

```
  // Convert milliseconds difference to days and round it
  let daysPassed = Math.ceil(difference / (1000 * 60 * 60 * 24));

  return daysPassed;
}

// Test data
console.log(days_passed(new Date(2015, 0, 15))); // 15 days passed since January 1, 2015
console.log(days_passed(new Date(2015, 11, 14))); // 348 days passed since January 1, 2015
```

17. **Write a JavaScript function to convert a Unix timestamp to time. Test Data :**
    console.log(days_passed(new Date(2015, 0, 15))); 15
    console.log(days_passed(new Date(2015, 11, 14))); 348

```
    function unix_timestamp_to_time(timestamp) {
  // Create a new Date object using the timestamp multiplied by 1000 (to convert seconds to milliseconds)
  let date = new Date(timestamp * 1000);

  // Get the components of the date
  let year = date.getFullYear();
  let month = ("0" + (date.getMonth() + 1)).slice(-2); // Months are zero based
  let day = ("0" + date.getDate()).slice(-2);
  let hours = ("0" + date.getHours()).slice(-2);
  let minutes = ("0" + date.getMinutes()).slice(-2);
  let seconds = ("0" + date.getSeconds()).slice(-2);

  // Format the date and time
  let formattedTime = `${year}-${month}-${day} ${hours}:${minutes}:${seconds}`;

  return formattedTime;
}

// Test data (Unix timestamps)
console.log(unix_timestamp_to_time(1421284533)); // "2015-01-15 14:35:33"
console.log(unix_timestamp_to_time(1450099200)); // "2015-12-14 00:00:00"
```

18. **Write a JavaScript program to calculate age. Test Data :**
    console.log(calculate_age(new Date(1982, 11, 4))); 32
    console.log(calculate_age(new Date(1962, 1, 1))); 53

```
function calculate_age(birthDate) {
  // Get the current date
  let currentDate = new Date();

  // Calculate the difference in years
  let age = currentDate.getFullYear() - birthDate.getFullYear();

  // Adjust age based on the month and day of birth
  if (currentDate.getMonth() < birthDate.getMonth() ||
      (currentDate.getMonth() === birthDate.getMonth() && currentDate.getDate() <
birthDate.getDate())) {
    age--;
  }

  return age;
}

// Test data (Date of birth)
console.log(calculate_age(new Date(1982, 11, 4))); // 32 (for someone born on
December 4, 1982)
console.log(calculate_age(new Date(1962, 1, 1))); // 53 (for someone born on February
1, 1962)
```

19. **Write a JavaScript function to get the day of the month, 2 digits with leading zeros. Test Data : d= new Date(2015, 10, 1);**
    console.log(day_of_the_month(d)); "01"

```
function day_of_the_month(date) {
  // Get the day of the month
  let day = date.getDate();

  // Ensure the day is two digits with leading zeros if necessary
  let formattedDay = ("0" + day).slice(-2);

  return formattedDay;
```

```
}

// Test data (Date object)
let d = new Date(2015, 10, 1);
console.log(day_of_the_month(d)); // "01"
```

20. **Write a JavaScript function to get a textual representation of a day (three letters, Mon through Sun). Test Data : dt = new Date(2015, 10, 1); console.log(short_Days(dt)); "Sun"**

```
function short_Days(date) {
  // Array of short day names
  const daysOfWeek = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'];

  // Get the day of the week (0 for Sunday, 1 for Monday, ..., 6 for Saturday)
  let dayOfWeek = date.getDay();

  // Return the three-letter abbreviation of the day
  return daysOfWeek[dayOfWeek];
}

// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(short_Days(dt)); // "Sun"
```

21. **Write a JavaScript function to get a full textual representation of the day of the week (Sunday through Saturday). Test Data : dt = new Date(2015, 10, 1); console.log(long_Days(dt)); "Sunday"**

```
function long_Days(date) {
  // Array of full day names
  const daysOfWeek = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday'];

  // Get the day of the week (0 for Sunday, 1 for Monday, ..., 6 for Saturday)
  let dayOfWeek = date.getDay();

  // Return the full name of the day
  return daysOfWeek[dayOfWeek];
}
```

```
// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(long_Days(dt)); // "Sunday"
```

22. **Write a JavaScript function to get ISO-8601 numeric representation of the day of the week (1 (for Monday) to 7 (for Sunday)). Test Data : dt = new Date(2015, 10, 1); console.log(ISO_numeric_date(dt));**

```
function ISO_numeric_date(date) {
  // Get the day of the week (0 for Sunday, 1 for Monday, ..., 6 for Saturday)
  let dayOfWeek = date.getDay();

  // Adjust the day of the week to ISO-8601 format (1 for Monday to 7 for Sunday)
  if (dayOfWeek === 0) {
    dayOfWeek = 7; // Convert Sunday (0) to 7
  }

  return dayOfWeek;
}

// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(ISO_numeric_date(dt)); // 7
```

23. . **Write a JavaScript function to get English ordinal suffix for the day of the month, 2 characters (st, nd, rd or th.). Test Data : dt = new Date(2015, 10, 1); console.log(english_ordinal_suffix(dt)); "1st"**

```
function english_ordinal_suffix(date) {
  // Get the day of the month
  let dayOfMonth = date.getDate();

  // Determine the suffix based on the day of the month
  if (dayOfMonth === 1 || dayOfMonth === 21 || dayOfMonth === 31) {
    return dayOfMonth + "st";
  } else if (dayOfMonth === 2 || dayOfMonth === 22) {
    return dayOfMonth + "nd";
```

```
  } else if (dayOfMonth === 3 || dayOfMonth === 23) {
    return dayOfMonth + "rd";
  } else {
    return dayOfMonth + "th";
  }
}

// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(english_ordinal_suffix(dt)); // "1st"
```

24. **Write a JavaScript function to get ISO-8601 week number of year, weeks starting on Monday. Example : 42 (the 42nd week in the year) Test Data : dt = new Date(2015, 10, 1); console.log(ISO8601_week_no(dt)); 44**

```
function ISO8601_week_no(date) {
  // Copy the date so we don't modify the original
  date = new Date(Date.UTC(date.getFullYear(), date.getMonth(), date.getDate()));

  // Set the target to Monday of the same week
  date.setUTCDate(date.getUTCDate() + 4 - (date.getUTCDay() || 7));

  // Get January 4th, which is always in week 1
  var yearStart = new Date(Date.UTC(date.getUTCFullYear(), 0, 4));

  // Calculate full weeks between January 4th and the target date
  var weekNo = Math.ceil((((date - yearStart) / 86400000) + 1) / 7);

  return weekNo;
}

// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(ISO8601_week_no(dt)); // 44
```

25. **Write a JavaScript function to get a full textual representation of a month, such as January or June. Test Data : dt = new Date(2015, 10, 1); console.log(full_month(dt)); "November**

```
function full_month(date) {
```

```
  // Array of month names
  const monthNames = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
  ];

  // Get the month index from the date object (0 for January, 11 for December)
  let monthIndex = date.getMonth();

  // Return the full month name from the array
  return monthNames[monthIndex];
}

// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(full_month(dt)); // "November"
```

26. Write a JavaScript function to get a numeric representation of a month, with leading zeros (01 through 12). Test Data : dt = new Date(2015, 10, 1); console.log(numeric_month(dt)); "11"

```
function numeric_month(date) {
  // Get the month from the date object (0 for January, 11 for December)
  let month = date.getMonth() + 1; // Adding 1 because getMonth() returns zero-based index

  // Convert month to a string and pad with leading zeros if necessary
  return month.toString().padStart(2, '0');
}

// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(numeric_month(dt)); // "11"
```

27.. Write a JavaScript function to get a short textual representation of a month, three letters (Jan through Dec). Test Data : dt = new Date(2015, 10, 1); console.log(short_months(dt)); "Nov

```
function short_months(date) {
  // Array of abbreviated month names
```

```
  const monthNamesShort = [
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
  ];

  // Get the month index from the date object (0 for January, 11 for December)
  let monthIndex = date.getMonth();

  // Return the short month name from the array
  return monthNamesShort[monthIndex];
}

// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(short_months(dt)); // "Nov"
```

28. **Write a JavaScript function to get a full numeric representation of a year (4 digits). Test Data : dt = new Date(2015, 10, 1); console.log(full_year(dt)); 2015**

```
function full_year(date) {
  // Get the full year (4 digits) from the date object
  let year = date.getFullYear();

  // Return the full year as a number
  return year;
}

// Test data (Date object)
let dt = new Date(2015, 10, 1);
console.log(full_year(dt)); // 2015
```

29. **Write a JavaScript function to get a two digit representation of a year. Examples : 79 or 04 Test Data : dt = new Date(1989, 10, 1); console.log(sort_year(dt)); "89"**

```
function two_digit_year(date) {
  // Get the full year from the date object
  let year = date.getFullYear();
```

```
  // Extract the last two digits of the year
  let twoDigitYear = year % 100;

  // Convert two-digit year to a string and pad with leading zeros if necessary
  return twoDigitYear.toString().padStart(2, '0');
}

// Test data (Date object)
let dt = new Date(1989, 10, 1);
console.log(two_digit_year(dt)); // "89"
```

30.. Write a JavaScript function to get lowercase Ante meridiem and Post meridiem.

```
    function getMeridiem() {
  // Get the current date object
  let now = new Date();

  // Get hours from 0 to 23
  let hours = now.getHours();

  // Determine if it's AM or PM based on the hours
  if (hours < 12) {
    return "am";
  } else {
    return "pm";
  }
}

// Test the function
console.log(getMeridiem()); // Outputs "am" or "pm" based on the current time
```

31.. Write a JavaScript function to get uppercase Ante meridiem and Post meridiem.

```
    function getMeridiem() {
  // Get the current date object
```

```
  let now = new Date();

  // Get hours from 0 to 23
  let hours = now.getHours();

  // Determine if it's AM or PM based on the hours
  if (hours < 12) {
    return "AM";
  } else {
    return "PM";
  }
}

// Test the function
console.log(getMeridiem()); // Outputs "AM" or "PM" based on the current time
```

32. **Write a JavaScript function to swatch Internet time (000 through 999). Test Data : dt = new Date(1989, 10, 1); console.log(internet_time(dt)); 812**

```
function internet_time(date) {
  // Get UTC time in milliseconds since Jan 1, 1970
  let utcMillis = date.getTime();

  // Convert UTC time to Swatch Internet Time (beat time)
  let beats = Math.floor((utcMillis / 86400000) * 1000) % 1000;

  // Return the beat time as a three-digit string
  return beats.toString().padStart(3, '0');
}

// Test data (Date object)
let dt = new Date(1989, 10, 1);
console.log(internet_time(dt)); // Outputs "812"
```

33. **Write a JavaScript function to get 12-hour format of an hour with leading zeros. Test Data : dt = new Date(1989, 10, 1); console.log(hours_with_zeroes(dt)); "12"**

```
        function hours_with_zeroes(date) {
  // Get the hour from the date object (0-23)
  let hours = date.getHours();

  // Convert to 12-hour format
  let hours12 = hours % 12;
  if (hours12 === 0) {
    hours12 = 12; // Convert 0 hours to 12 in 12-hour format
  }

  // Convert to string and pad with leading zeros if necessary
  return hours12.toString().padStart(2, '0');
}

// Test data (Date object)
let dt = new Date(1989, 10, 1);
console.log(hours_with_zeroes(dt)); // Outputs "12"
```

34. . Write a JavaScript function to get 24-hour format of an hour without leading zeros. Test Data : dt = new Date(1989, 10, 1); console.log(hours_without_zeroes(dt)); 0

```
        function hours_without_zeroes(date) {
  // Get the hour from the date object (0-23)
  let hours = date.getHours();

  return hours;
}

// Test data (Date object)
let dt = new Date(1989, 10, 1);
console.log(hours_without_zeroes(dt)); // Outputs 0
```

35. Write a JavaScript function to get minutes with leading zeros (00 to 59). Test Data : dt = new Date(1989, 10, 1); console.log(minutes_with_leading_zeros(dt)); "00"

```
        function minutes_with_leading_zeros(date) {
```

```
  // Get the minutes from the date object (0-59)
  let minutes = date.getMinutes();

  // Convert to string and pad with leading zeros if necessary
  return minutes.toString().padStart(2, '0');
}

// Test data (Date object)
let dt = new Date(1989, 10, 1);
console.log(minutes_with_leading_zeros(dt)); // Outputs "00"
```

36. **Write a JavaScript function to get seconds with leading zeros (00 through 59). Test Data : dt = new Date(1989, 10, 1); console.log(seconds_with_leading_zeros(dt)); "00**

```
function seconds_with_leading_zeros(date) {
  // Get the seconds from the date object (0-59)
  let seconds = date.getSeconds();

  // Convert to string and pad with leading zeros if necessary
  return seconds.toString().padStart(2, '0');
}

// Test data (Date object)
let dt = new Date(1989, 10, 1);
console.log(seconds_with_leading_zeros(dt)); // Outputs "00"
```

37. **Write a JavaScript function to get Timezone. Test Data : dt = new Date(); console.log(seconds_with_leading_zeros(dt)); "India Standard Time"**

```
function getTimezone() {
  // Get the current date object
  let dt = new Date();

  // Get the timezone offset in minutes from UTC
  let offset = dt.getTimezoneOffset();

  // Calculate the sign of the offset (e.g., + or -)
  let sign = offset > 0 ? '-' : '+';
```

```
  // Convert offset to positive number
  offset = Math.abs(offset);
```

38. **Write a JavaScript function to find whether or not the date is in daylights savings time. Test Data : dt = new Date(); console.log(daylights_savings(dt)); 1**

```
        function daylights_savings(date) {
  // Get the timezone offset for the given date
  let timezoneOffsetStandard = date.getTimezoneOffset();

  // Create a new date object shifted by 6 months to check the offset change
  let halfYearLater = new Date(date);
  halfYearLater.setMonth(date.getMonth() + 6);

  // Get the timezone offset for the date 6 months later
  let timezoneOffsetDST = halfYearLater.getTimezoneOffset();

  // Compare the offsets to determine if DST is observed
  return timezoneOffsetStandard !== timezoneOffsetDST ? 1 : 0;
}

// Test data
let dt = new Date();
console.log(daylights_savings(dt)); // Outputs 1 if currently in DST, otherwise 0
```

39. **Write a JavaScript function to get difference to Greenwich time (GMT) in hours. Test Data : dt = new Date(); console.log(diff_to_GMT(dt)); "+05.500"**

```
        function diff_to_GMT(date) {
  // Get the timezone offset in minutes
  let offsetMinutes = date.getTimezoneOffset();

  // Convert minutes to hours and minutes
  let hours = Math.abs(Math.floor(offsetMinutes / 60));
  let minutes = Math.abs(offsetMinutes % 60);

  // Determine the sign of the offset
```

```
  let sign = offsetMinutes > 0 ? '-' : '+';

  // Format the output as ±HH.MM
  return sign + hours.toString().padStart(2, '0') + '.' + minutes.toString().padStart(2, '0');
}

// Test data
let dt = new Date();
console.log(diff_to_GMT(dt)); // Outputs "+05.500" for example
```

40. **Write a JavaScript function to get timezone offset in seconds. Note : The offset for timezones west of UTC is always negative, and for those east of UTC is always positive. Test Data : dt = new Date(); console.log(timezone_offset_in_seconds(dt)); 19800**

```
        function timezone_offset_in_seconds(date) {
  // Get the timezone offset in minutes
  let offsetMinutes = date.getTimezoneOffset();

  // Convert minutes to seconds with the correct sign
  let offsetSeconds = offsetMinutes * 60;

  return offsetSeconds;
}

// Test data
let dt = new Date();
console.log(timezone_offset_in_seconds(dt)); // Outputs the timezone offset in seconds
```

41. **Write a JavaScript function to add specified years to a date. Test Data : dt = new Date(2014,10,2); console.log(add_years(dt, 10).toString()); Output : "Sat Nov 02 2024 00:00:00 GMT+0530 (India Standard Time)"**

```
        function add_years(date, years) {
  // Copy the date to avoid mutating the original
  let newDate = new Date(date);

  // Calculate the new year by adding the specified number of years
```

```
  let newYear = newDate.getFullYear() + years;

  // Set the new year
  newDate.setFullYear(newYear);

  return newDate;
}
```

42. **Write a JavaScript function to add specified weeks to a date. Test Data : dt = new Date(2014,10,2); console.log(add_weeks(dt, 10).toString()); Output : "Sun Jan 11 2015 00:00:00 GMT+0530 (India Standard Time)"**

```
        function add_weeks(date, weeks) {
  // Copy the date to avoid mutating the original
  let newDate = new Date(date);

  // Calculate the milliseconds to add based on weeks
  let millisecondsToAdd = weeks * 7 * 24 * 60 * 60 * 1000;

  // Add the milliseconds to the date
  newDate.setTime(newDate.getTime() + millisecondsToAdd);

  return newDate;
}

// Test data
let dt = new Date(2014, 10, 2);
console.log(add_weeks(dt, 10).toString()); // Outputs "Sun Jan 11 2015 00:00:00
GMT+0530 (India Standard Time)"
```

43. **Write a JavaScript function to add specified months to a date. Test Data : dt = new Date(2014,10,2); console.log(add_months(dt, 10).toString()); Output : "Wed Sep 02 2015 00:00:00 GMT+0530 (India Standard Time)"**

```
        function add_weeks(date, weeks) {
  // Copy the date to avoid mutating the original
  let newDate = new Date(date);

  // Calculate the milliseconds to add based on weeks
```

```
  let millisecondsToAdd = weeks * 7 * 24 * 60 * 60 * 1000;

  // Add the milliseconds to the date
  newDate.setTime(newDate.getTime() + millisecondsToAdd);

  return newDate;
}

// Test data
let dt = new Date(2014, 10, 2);
console.log(add_weeks(dt, 10).toString()); // Outputs "Sun Jan 11 2015 00:00:00
GMT+0530 (India Standard Time)"
```

44. Write a JavaScript function to get time differences in minutes between two dates. Test Data : dt1 = new Date("October 13, 2014 11:11:00"); dt2 = new Date("October 13, 2014 11:13:00"); console.log(diff_minutes(dt1, dt2)); 2 45.

```
function diff_minutes(dt1, dt2) {
  let diff = (dt2.getTime() - dt1.getTime()) / 1000; // difference in seconds
  diff /= 60; // difference in minutes
  return Math.abs(Math.round(diff)); // return the absolute value of the rounded
difference
}

// Test Data
let dt1 = new Date("October 13, 2014 11:11:00");
let dt2 = new Date("October 13, 2014 11:13:00");
console.log(diff_minutes(dt1, dt2)); // Output: 2
```

45. . Write a JavaScript function to get time differences in hours between two dates. Test Data : dt1 = new Date("October 13, 2014 08:11:00"); dt2 = new Date("October 13, 2014 11:13:00"); console.log(diff_hours(dt1, dt2)); 3

```
function diff_hours(dt1, dt2) {
  let diff = (dt2.getTime() - dt1.getTime()) / 1000; // difference in seconds
  diff /= 3600; // difference in hours
  return Math.abs(Math.round(diff)); // return the absolute value of the rounded
difference
}
```

```
// Test Data
let dt1 = new Date("October 13, 2014 08:11:00");
let dt2 = new Date("October 13, 2014 11:13:00");
console.log(diff_hours(dt1, dt2)); // Output: 3
```

46.. Write a JavaScript function to get time differences in days between two dates. Test Data : dt1 = new Date("October 13, 2014 08:11:00"); dt2 = new Date("October 19, 2014 11:13:00"); console.log(diff_days(dt1, dt2)); 6

```
function diff_days(dt1, dt2) {
  let diff = (dt2.getTime() - dt1.getTime()) / (1000 * 3600 * 24); // difference in days
  return Math.abs(Math.round(diff)); // return the absolute value of the rounded
difference
}

// Test Data
let dt1 = new Date("October 13, 2014 08:11:00");
let dt2 = new Date("October 19, 2014 11:13:00");
console.log(diff_days(dt1, dt2)); // Output: 6
```

47. Write a JavaScript function to get time differences in weeks between two dates. Test Data : dt1 = new Date("June 13, 2014 08:11:00"); dt2 = new Date("October 19, 2014 11:13:00"); console.log(diff_weeks(dt1, dt2)); 18

```
function diff_weeks(dt1, dt2) {
  let diff = (dt2.getTime() - dt1.getTime()) / (1000 * 3600 * 24 * 7); // difference in weeks
  return Math.abs(Math.round(diff)); // return the absolute value of the rounded
difference
}

// Test Data
let dt1 = new Date("June 13, 2014 08:11:00");
let dt2 = new Date("October 19, 2014 11:13:00");
console.log(diff_weeks(dt1, dt2)); // Output: 18
```

48. Write a JavaScript function to get time differences in months between two dates. Test Data : dt1 = new Date("June 13, 2014 08:11:00"); dt2 = new Date("October 19, 2014 11:13:00"); console.log(diff_months(dt1, dt2)); 5

```
        function diff_months(dt1, dt2) {
  let months;

  months = (dt2.getFullYear() - dt1.getFullYear()) * 12;
  months -= dt1.getMonth();
  months += dt2.getMonth();

  return months <= 0 ? 0 : months;
}

// Test Data
let dt1 = new Date("June 13, 2014 08:11:00");
let dt2 = new Date("October 19, 2014 11:13:00");
console.log(diff_months(dt1, dt2)); // Output: 5
```

49. Write a JavaScript function to get time differences in years between two dates. Test Data : dt1 = new Date("June 13, 2014 08:11:00"); dt2 = new Date("October 19, 2017 11:13:00"); console.log(diff_years(dt1, dt2)); 3

```
        function diff_years(dt1, dt2) {
  let diff = dt2.getFullYear() - dt1.getFullYear();

  // Adjust if dt2's month and day are before dt1's month and day
  if (dt2.getMonth() < dt1.getMonth() || (dt2.getMonth() === dt1.getMonth() &&
dt2.getDate() < dt1.getDate())) {
    diff--;
  }

  return diff;
}

// Test Data
let dt1 = new Date("June 13, 2014 08:11:00");
let dt2 = new Date("October 19, 2017 11:13:00");
console.log(diff_years(dt1, dt2)); // Output: 3
```

50. Write a JavaScript function to get the week start date.

```
        function getWeekStartDate() {
  let today = new Date();
  let day = today.getDay(); // Get current day of the week (0-6, where 0 is Sunday)
  let diff = today.getDate() - day; // Calculate the difference between today's date and the
day of the week

  // Create a new date for the start of the week (Sunday)
  let weekStart = new Date(today.setDate(diff));

  // Return the start date of the week
  return weekStart;
}

// Test
console.log(getWeekStartDate());
```

51. Write a JavaScript function to get the week end date.

```
        function getWeekEndDate() {
  let today = new Date();
  let day = today.getDay(); // Get current day of the week (0-6, where 0 is Sunday)
  let diff = 6 - day; // Calculate the difference to Saturday

  // Create a new date for the end of the week (Saturday)
  let weekEnd = new Date(today.setDate(today.getDate() + diff));

  // Return the end date of the week
  return weekEnd;
}

// Test
console.log(getWeekEndDate());
```

52. Write a JavaScript function to get the month start date.

```
        function getMonthStartDate() {
  let today = new Date();
  let monthStart = new Date(today.getFullYear(), today.getMonth(), 1);
  return monthStart;
```

```
}

// Test
console.log(getMonthStartDate());
```

53. Write a JavaScript function to get the month end date.

```
        function getMonthEndDate() {
  let today = new Date();
  let monthEnd = new Date(today.getFullYear(), today.getMonth() + 1, 0);
  return monthEnd;
}

// Test
console.log(getMonthEndDate());
```

JAVA SCRIPT STRING

1. **Write a JavaScript function to check whether an `input` is a string or not. Test Data : console.log(is_string('w3resource')); true console.log(is_string([1, 2, 4, 0])); false**

```
        function is_string(input) {
  return typeof input === 'string' || input instanceof String;
}

// Test Data
console.log(is_string('w3resource')); // true
console.log(is_string([1, 2, 4, 0])); // false
```

2. **Write a JavaScript function to check whether a string is blank or not. Test Data : console.log(is_Blank('')); console.log(is_Blank('abc')); true false**

```
        function is_Blank(input) {
  return input.trim().length === 0;
}

// Test Data
```

```
console.log(is_Blank(''));       // true
console.log(is_Blank('abc'));    // false
console.log(is_Blank('   '));    // true
console.log(is_Blank('\n\t'));   // true
```

3. **Write a JavaScript function to split a string and convert it into an array of words. Test Data : console.log(string_to_array("Robin Singh")); ["Robin", "Singh"]**

```
function string_to_array(input) {
  return input.split(' ');
}

// Test Data
console.log(string_to_array("Robin Singh")); // ["Robin", "Singh"]
```

4. **Write a JavaScript function to remove specified number of characters from a string. Test Data : console.log(truncate_string("Robin Singh",4)); "Robi"**

```
function truncate_string(str, num) {
  return str.substring(0, str.length - num);
}

// Test Data
console.log(truncate_string("Robin Singh", 4)); // "Robin Si"
```

5. **Write a JavaScript function to convert a string in abbreviated form. Test Data : console.log(abbrev_name("Robin Singh")); "Robin S."**

```
function abbrev_name(name) {
  let nameArr = name.split(" ");
  if (nameArr.length > 1) {
    return `${nameArr[0]} ${nameArr[1].charAt(0)}.`;
  }
  return name;
}

// Test Data
console.log(abbrev_name("Robin Singh")); // "Robin S."
```

6. **. Write a JavaScript function to hide email addresses to protect from unauthorized user. Test Data : console.log(protect_email("robin_singh@example.com")); "robin...@example.com"**

```
function protect_email(email) {
  let [localPart, domainPart] = email.split("@");
  let hiddenPart = localPart.substring(0, 5) + "...";
  return `${hiddenPart}@${domainPart}`;
}

// Test Data
console.log(protect_email("robin_singh@example.com")); // "robin...@example.com"
```

7. **7. Write a JavaScript function to parameterize a string. Test Data : console.log(string_parameterize("Robin Singh from USA.")); "robin-singh-from-usa"**

```
function string_parameterize(str) {
  // Convert to lowercase
  str = str.toLowerCase();
  // Replace spaces and special characters with hyphens
  str = str.replace(/[^a-z0-9\s]/g, '').replace(/\s+/g, '-');
  return str;
}

// Test Data
console.log(string_parameterize("Robin Singh from USA.")); // "robin-singh-from-usa"
```

8. **. Write a JavaScript function to capitalize the first letter of a string. Test Data : console.log(capitalize('js string exercises')); "Js string exercises"**

```
function capitalize(str) {
  if (str.length === 0) return str;
  return str.charAt(0).toUpperCase() + str.slice(1);
}

// Test Data
```

```
console.log(capitalize('js string exercises')); // "Js string exercises"
```

9. **Write a JavaScript function to capitalize the first letter of each word in a string. Test Data : console.log(capitalize_Words('js string exercises')); "Js String Exercises"**

```
function capitalize_Words(str) {
  return str.split(' ').map(word => word.charAt(0).toUpperCase() + word.slice(1)).join(' ');
}

// Test Data
console.log(capitalize_Words('js string exercises')); // "Js String Exercises"
```

10. **Write a JavaScript function that takes a string which has lower and upper case letters as a parameter and converts upper case letters to lower case, and lower case letters to upper case. Test Data : console.log(swapcase('AaBbc')); "aAbBC**

```
function swapcase(str) {
 var swapped = '';
 for (var i = 0; i < str.length; i++) {
 var char = str[i];
 if (char === char.toUpperCase()) {
      swapped += char.toLowerCase();
    } else {
      swapped += char.toUpperCase();
    }
  }
 return swapped;
}

// Test Data
console.log(swapcase('AaBbc')); // "aAbBC"
```

11. **Write a JavaScript function to convert a string into camel case. Test Data : console.log(camelize("JavaScript Exercises")); console.log(camelize("JavaScript exercises")); console.log(camelize("JavaScriptExercises")); "JavaScriptExercises" "JavaScriptExercises**

```
function camelize(str) {
 return str.replace(/\s(.)/g, function(match, group1) {
 return group1.toUpperCase();
  }).replace(/\s/g, '').replace(/^(.)/, function(match, group1) {
 return group1.toLowerCase();
  });
}

// Test Data
console.log(camelize("JavaScript Exercises")); // "javaScriptExercises"
console.log(camelize("JavaScript exercises")); // "javaScriptExercises"
console.log(camelize("JavaScriptExercises")); // "javaScriptExercises"
```

12. Write a JavaScript function to uncamelize a string. Test Data : console.log(uncamelize('helloWorld')); console.log(uncamelize('helloWorld','-')); console.log(uncamelize('helloWorld','_')); "hello world" "hello-world" "hello_world"

```
function uncamelize(str, separator = ' ') {
 // Insert a space before all uppercase letters
  return str.replace(/[A-Z]/g, function(match) {
 return separator + match.toLowerCase();
  });
}

// Test Data
console.log(uncamelize('helloWorld')); // "hello world"
console.log(uncamelize('helloWorld', '-')); // "hello-world"
console.log(uncamelize('helloWorld', '_')); // "hello_world"
```

13. . Write a JavaScript function to concatenates a given string n times (default is 1). Test Data : console.log(repeat('Ha!')); console.log(repeat('Ha!',2)); console.log(repeat('Ha!',3)); "Ha!" "Ha!Ha!" "Ha!Ha!Ha!"

```
function repeat(str, n = 1) {
 // Initialize an empty string to store the result
  let result = '';

 // Repeat the string 'n' times
```

```
  for (let i = 0; i < n; i++) {
    result += str;
  }

 return result;
}

// Test Data
console.log(repeat('Ha!')); // "Ha!"
console.log(repeat('Ha!', 2)); // "Ha!Ha!"
console.log(repeat('Ha!', 3)); // "Ha!Ha!Ha!"
```

14. **Write a JavaScript function to insert a string within a string at a particular position (default is 1). Test Data : console.log(insert('We are doing some exercises.')); console.log(insert('We are doing some exercises.','JavaScript ')); console.log(insert('We are doing some exercises.','JavaScript ',18)); "We are doing some exercises." "JavaScript We are doing some exercises." "We are doing some JavaScript exercises."**

```
function insert(main_string, ins_string, pos = 1) {
 if (typeof pos == "undefined") {
    pos = 0;
  }
 if(typeof ins_string == "undefined") {
    ins_string = '';
  }
 let result = main_string.slice(0, pos) + ins_string + main_string.slice(pos);
 return result;
}

// Test Data
console.log(insert('We are doing some exercises.')); // "We are doing some exercises."
console.log(insert('We are doing some exercises.','JavaScript ')); // "JavaScript We are doing some exercises."
console.log(insert('We are doing some exercises.','JavaScript ',18)); // "We are doing some JavaScript exercises."
```

15. **Write a JavaScript function to humanized number (Formats a number to a human-readable string.) with the correct suffix such as 1st, 2nd, 3rd or 4th.**

Test Data : console.log(humanize_format());
console.log(humanize_format(1)); console.log(humanize_format(8));
console.log(humanize_format(301)); console.log(humanize_format(402));
"1st" "8th" "301st"

```
function humanize_format(num) {
 if (typeof num === 'undefined') return '';

 if (num % 100 >= 11 && num % 100 <= 13) {
 return num + 'th';
  }

 switch (num % 10) {
 case 1: return num + 'st';
 case 2: return num + 'nd';
 case 3: return num + 'rd';
 default: return num + 'th';
  }
}

// Test Data
console.log(humanize_format()); // ""
console.log(humanize_format(1)); // "1st"
console.log(humanize_format(8)); // "8th"
console.log(humanize_format(301)); // "301st"
console.log(humanize_format(402)); // "402nd"
```

16. **Write a JavaScript function to truncates a string if it is longer than the specified number of characters. Truncated strings will end with a translatable ellipsis sequence ("…") (by default) or specified characters.**
Test Data : console.log(text_truncate('We are doing JS string exercises.'))
console.log(text_truncate('We are doing JS string exercises.',19))
console.log(text_truncate('We are doing JS string exercises.',15,'!!')) "We are doing JS string exercises." "We are doing JS ..." "We are doing !!"

```
function text_truncate(str, length, ending) {
 if (length == null) {
    length = 100;
  }
 if (ending == null) {
```

```
    ending = '…';
  }
 if (str.length > length) {
 return str.substring(0, length - ending.length) + ending;
  } else {
 return str;
  }
}

// Test Data
console.log(text_truncate('We are doing JS string exercises.'));
console.log(text_truncate('We are doing JS string exercises.', 19));
console.log(text_truncate('We are doing JS string exercises.', 15, '!!'));
```

17. **Write a JavaScript function to chop a string into chunks of a given length. Test Data : console.log(string_chop('w3resource')); console.log(string_chop('w3resource',2)); console.log(string_chop('w3resource',3)); ["w3resource"] ["w3", "re", "so", "ur", "ce"] ["w3r", "eso", "urc", "e"]**

```
function string_chop(str, size) {
 if (size == null) {
    size = 1;
  }
 var chunks = [];
 for (var i = 0; i < str.length; i += size) {
    chunks.push(str.slice(i, i + size));
  }
 return chunks;
}

// Test Data
console.log(string_chop('w3resource'));
console.log(string_chop('w3resource', 2));
console.log(string_chop('w3resource', 3));
```

18. **Write a JavaScript function to count the occurrence of a substring in a string. Test Data : console.log(count("The quick brown fox jumps over the lazy dog", 'the')); Output : 2 console.log(count("The quick brown fox jumps over the lazy dog", 'fox',false)); Output : 1**

```
function count(mainStr, subStr, caseSensitive = true) {
 // If caseSensitive is false, convert both strings to lowercase
  if (!caseSensitive) {
    mainStr = mainStr.toLowerCase();
    subStr = subStr.toLowerCase();
  }

 // Using regular expression to find all occurrences of the substring
  var regex = new RegExp(subStr, 'g');
 var matches = mainStr.match(regex);

 // Return the count of occurrences
  return matches ? matches.length : 0;
}

// Test Data
console.log(count("The quick brown fox jumps over the lazy dog", 'the')); //
Output: 2
console.log(count("The quick brown fox jumps over the lazy dog", 'fox', false)); //
Output: 1
```

19. Write a JavaScript function to escape a HTML string. Test Data : console.log(escape_HTML('')); Output : "<a href="javascript-string-exercise-17.php" target="_blank">"

```
function escape_HTML(htmlStr) {
 return htmlStr.replace(/[&<"']/g, function(match) {
 switch(match) {
 case '&':
 return '&amp;';
 case '<':
 return '&lt;';
 case '>':
 return '&gt;';
 case '"':
 return '&quot;';
 case "'":
 return '&#039;';
   }
```

```
  });
}

// Test Data
console.log(escape_HTML('<a href="javascript-string-exercise-17.php"
target="_blank">'));
```

20. **Write a JavaScript function that can pad (left, right) a string to get to a determined length. Test Data : console.log(formatted_string('0000',123,'l')); console.log(formatted_string('00000000',123,")); Output : "0123" "12300000"**

```
function formatted_string(pad, input, direction) {
 if (direction === 'l') {
 return (pad + input).slice(-pad.length);
  } else {
 return (input + pad).substring(0, pad.length);
  }
}

// Test Data
console.log(formatted_string('0000', 123, 'l')); // "0123"
console.log(formatted_string('00000000', 123, ")); // "12300000"
```

21. **Write a JavaScript function to repeat a string a specified times. Test Data : console.log(repeat_string('a', 4)); console.log(repeat_string('a')); Output : "aaaa" "Error in string or count."**

```
function repeat_string(str, count) {
 if (typeof str !== 'string' || typeof count !== 'number') {
 return "Error in string or count.";
  }

 if (count <= 0) {
 return "";
  }

 return str.repeat(count);
}
```

```
// Test Data
console.log(repeat_string('a', 4)); // "aaaa"
console.log(repeat_string('a')); // "Error in string or count."
```

22. **Write a JavaScript function to get a part of a string after a specified character. Test Data : console.log(subStrAfterChars('w3resource: JavaScript Exercises', ':','a')); console.log(subStrAfterChars('w3resource: JavaScript Exercises', 'E','b')); Output : "w3resource" "xercises"**

```
function subStrAfterChars(str, char, mode = 'a') {
 if (mode === 'a') {
 // Using indexOf and substring
    let index = str.indexOf(char);
 return index !== -1 ? str.substring(0, index) : str;
  } else if (mode === 'b') {
 // Using lastIndexOf and substring
    let index = str.lastIndexOf(char);
 return index !== -1 ? str.substring(index + 1) : str;
  } else {
 return "Invalid mode specified.";
  }
}

// Test Data
console.log(subStrAfterChars('w3resource: JavaScript Exercises', ':', 'a')); //
"w3resource"
console.log(subStrAfterChars('w3resource: JavaScript Exercises', 'E', 'b')); //
"xercises"

function subStrAfterChars(str, char, mode = 'a') {
 if (mode === 'a') {
 // Using indexOf and substring
    let index = str.indexOf(char);
 return index !== -1 ? str.substring(0, index) : str;
  } else if (mode === 'b') {
 // Using lastIndexOf and substring
    let index = str.lastIndexOf(char);
 return index !== -1 ? str.substring(index + 1) : str;
  } else {
 return "Invalid mode specified.";
  }
```

```
}

// Test Data
console.log(subStrAfterChars('w3resource: JavaScript Exercises', ':', 'a')); //
"w3resource"
console.log(subStrAfterChars('w3resource: JavaScript Exercises', 'E', 'b')); //
"xercises"
```

23. **Write a JavaScript function to strip leading and trailing spaces from a string. Test Data : console.log(strip('w3resource ')); console.log(strip(' w3resource')); console.log(strip(' w3resource ')); Output : "w3resource" "w3resource" "w3resource"**

```
function strip(str) {
 // Using trim() method to remove leading and trailing spaces
  return str.trim();
}

// Test Data
console.log(strip('w3resource ')); // Output: "w3resource"
console.log(strip(' w3resource')); // Output: "w3resource"
console.log(strip(' w3resource ')); // Output: "w3resource"
```

24. **Write a JavaScript function to truncate a string to a certain number of words. Test Data : console.log(truncate('The quick brown fox jumps over the lazy dog', 4)); Output : "The quick brown fox"**

```
function truncate(str, numWords) {
 // Split the string into an array of words
  let words = str.split(' ');

 // Slice the array to select only the specified number of words
  let truncatedWords = words.slice(0, numWords);

 // Join the sliced array back into a string
  let truncatedString = truncatedWords.join(' ');

 // Return the truncated string
  return truncatedString;
}
```

```
// Test Data
console.log(truncate('The quick brown fox jumps over the lazy dog', 4));
```

25. **Write a JavaScript function to alphabetize a given string. Alphabetize string : An individual string can be alphabetized. This rearranges the letters so they are sorted A to Z. Test Data : console.log(alphabetize_string('United States')); Output : "SUadeeinsttt"**

```
function alphabetize_string(str) {
 // Convert the string to an array of characters
  let chars = str.split('');

 // Sort the array of characters alphabetically
  let sortedChars = chars.sort();

 // Join the sorted array back into a string
  let sortedString = sortedChars.join('');

 // Return the alphabetized string
  return sortedString;
}

// Test Data
console.log(alphabetize_string('United States'));
```

26. **Write a JavaScript function to remove the first occurrence of a given 'search string' from a string. Test Data : console.log(remove_first_occurrence("The quick brown fox jumps over the lazy dog", 'the')); Output : "The quick brown fox jumps over lazy dog"**

```
function remove_first_occurrence(str, searchStr) {
 // Find the index of the first occurrence of searchStr
  let index = str.indexOf(searchStr);

 // If searchStr is found, remove it using substring and concatenate the parts
  if (index !== -1) {
 return str.substring(0, index) + str.substring(index + searchStr.length);
  }
```

```
 // If searchStr is not found, return the original string
  return str;
}

// Test Data
console.log(remove_first_occurrence("The quick brown fox jumps over the lazy
dog", 'the'));
```

27. Write a JavaScript function to convert ASCII to Hexadecimal format. Test Data : console.log(ascii_to_hexa('12')); console.log(ascii_to_hexa('100')); Output : "3132" "313030"

```
function ascii_to_hexa(str) {
 let hex = '';

 for (let i = 0; i < str.length; i++) {
 // Get ASCII value of character at index i
    let ascii = str.charCodeAt(i);

 // Convert ASCII value to hexadecimal and append to hex string
    hex += ascii.toString(16);
  }

 return hex;
}

// Test Data
console.log(ascii_to_hexa('12'));
console.log(ascii_to_hexa('100'));
```

28. Write a JavaScript function to convert Hexadecimal to ASCII format. Test Data : console.log(hex_to_ascii('3132')); console.log(hex_to_ascii('313030')); Output : "12" "100"

```
function hex_to_ascii(hexStr) {
 let asciiStr = '';

 // Ensure the hex string has even length
  if (hexStr.length % 2 !== 0) {
 throw new Error('Invalid hexadecimal string');
```

```
 }

 // Convert each pair of hex digits to ASCII character
  for (let i = 0; i < hexStr.length; i += 2) {
 let hexPair = hexStr.substr(i, 2); // Extract two characters
    let asciiChar = String.fromCharCode(parseInt(hexPair, 16)); // Convert hex to
ASCII
    asciiStr += asciiChar;
  }

 return asciiStr;
}

// Test Data
console.log(hex_to_ascii('3132'));
console.log(hex_to_ascii('313030'));
```

29. Write a JavaScript function to find a word within a string. Test Data : console.log(search_word('The quick brown fox', 'fox')); console.log(search_word('aa, bb, cc, dd, aa', 'aa')); Output : "'fox' was found 1 times." "'aa' was found 2 times."

```
function search_word(str, word) {
 let count = 0;
 let index = str.indexOf(word);

 while (index !== -1) {
    count++;
    index = str.indexOf(word, index + 1);
  }

 return `'${word}' was found ${count} times.`;
}

// Test Data
console.log(search_word('The quick brown fox', 'fox'));
console.log(search_word('aa, bb, cc, dd, aa', 'aa'));
```

30. **Write a JavaScript function check if a string ends with specified suffix. Test Data : console.log(string_endsWith('JS PHP PYTHON','PYTHON')); true console.log(string_endsWith('JS PHP PYTHON','')); false**

```
function string_endsWith(str, suffix) {
 return str.endsWith(suffix);
}

// Test Data
console.log(string_endsWith('JS PHP PYTHON', 'PYTHON')); // true
console.log(string_endsWith('JS PHP PYTHON', '')); // false
```

31. . **Write a JavaScript function to escapes special characters (&, <, >, ', ") for use in HTML. Test Data : console.log(escape_html('PHP & MySQL')); "PHP & MySQL" console.log(escape_html('3 > 2')); "3 > 2"**

```
function escape_html(str) {
 let escapedString = str.replace(/&/g, '&amp;')
                        .replace(/</g, '&lt;')
                        .replace(/>/g, '&gt;')
                        .replace(/'/g, '&apos;')
                        .replace(/"/g, '&quot;');
 return escapedString;
}

// Test Data
console.log(escape_html('PHP & MySQL')); // "PHP &amp;amp; MySQL"
console.log(escape_html('3 > 2')); // "3 &amp;gt; 2"
console.log(escape_html('<script>alert("Hello!");</script>')); //
"&amp;lt;script&amp;gt;alert(&amp;quot;Hello!&amp;quot;);&amp;lt;/script&amp;gt;"
```

32. **2. Write a JavaScript function to remove non-printable ASCII chars. Test Data : console.log(remove_non_ascii('äÄçÇéÉêPHP-MySQLöÖÐþúÚ')); "PHP-MySQL**

```
function remove_non_ascii(str) {
 return str.replace(/[^\x20-\x7E]/g, '');
}

// Test Data
```

```
console.log(remove_non_ascii('äÄçÇéÉêPHP-MySQLöÖÐþúÚ'));
```

33. **Write a JavaScript function to remove non-word characters. Test Data : console.log(remove_non_word('PHP ~!@#$%^&*()+`-={}[]|\\:";\'/?><., MySQL')); "PHP - MySQL"**

```
function remove_non_word(str) {
 return str.replace(/[^\w\s]/gi, '');
}

// Test Data
console.log(remove_non_word('PHP ~!@#$%^&*()+`-={}[]|\\:";\'/?><., MySQL'));
```

34. **Write a JavaScript function to convert a string to title case. Test Data : console.log(sentenceCase('PHP exercises. python exercises.')); "Php Exercises. Python Exercises.**

```
function sentenceCase(str) {
 return str.toLowerCase().replace(/(^|\s)\S/g, function (match) {
 return match.toUpperCase();
  });
}

// Test Data
console.log(sentenceCase('PHP exercises. python exercises.'));
```

35. **Write a JavaScript function to remove HTML/XML tags from string. Test Data : console.log(strip_html_tags('**

PHP Exercises

')); "PHP Exercises"

```
function strip_html_tags(str) {
 if ((str === null) || (str === '')) {
 return false;
  } else {
    str = str.toString();
  }
 return str.replace(/<[^>]*>/g, '');
}
```

```
// Test Data
console.log(strip_html_tags('<p><strong><em>PHP
Exercises</em></strong></p>'));
```

37. Write a JavaScript function to test case insensitive (except special Unicode characters) string comparison. Test Data : console.log(compare_strings('abcd', 'AbcD')); true console.log(compare_strings('ABCD', 'Abce')); false

```
function compare_strings(str1, str2) {
 // Convert both strings to lowercase before comparison
  return str1.toLowerCase() === str2.toLowerCase();
}

// Test Data
console.log(compare_strings('abcd', 'AbcD')); // true
console.log(compare_strings('ABCD', 'Abce')); // false
```

38. Write a JavaScript function to create a case-insensitive search. Test Data : console.log(case_insensitive_search('JavaScript Exercises', 'exercises')); "Matched" console.log(case_insensitive_search('JavaScript Exercises', 'Exercises')); "Matched" console.log(case_insensitive_search('JavaScript Exercises', 'Exercisess')); "Not Matched"

```
function case_insensitive_search(str, searchStr) {
 // Convert both strings to lowercase before comparing
  const lowerCaseStr = str.toLowerCase();
 const lowerCaseSearchStr = searchStr.toLowerCase();

 // Check if the lowercased searchStr exists in lowercased str
  if (lowerCaseStr.includes(lowerCaseSearchStr)) {
 return "Matched";
  } else {
 return "Not Matched";
  }
}

// Test Data
```

```
console.log(case_insensitive_search('JavaScript Exercises', 'exercises')); //
"Matched"
console.log(case_insensitive_search('JavaScript Exercises', 'Exercises')); //
"Matched"
console.log(case_insensitive_search('JavaScript Exercises', 'Exercisess')); //
"Not Matched"
```

JAVASCRIPT VALIDATION WITH REGULAR

1. **. Write a JavaScript program to test the first character of a string is uppercase or not.**

```
function isFirstCharUpperCase(str) {
 // Check if the first character is uppercase using regex
  return /^[A-Z]/.test(str);
}

// Test Data
console.log(isFirstCharUpperCase('Hello')); // true
console.log(isFirstCharUpperCase('world')); // false
console.log(isFirstCharUpperCase('JavaScript')); // true
console.log(isFirstCharUpperCase('123abc')); // false
console.log(isFirstCharUpperCase('')); // false
```

2. **. Write a JavaScript program to check a credit card number.**

```
function isValidCreditCardNumber(cardNumber) {
 // Remove any non-digit characters
  cardNumber = cardNumber.replace(/\D/g, '');

 // Check if the card number is within the correct length
  if (cardNumber.length < 13 || cardNumber.length > 19) {
 return false;
  }

 // Check using Luhn algorithm
  let sum = 0;
 let doubleUp = false;
 for (let i = cardNumber.length - 1; i >= 0; i--) {
```

```
 let digit = parseInt(cardNumber.charAt(i), 10);
 if (doubleUp) {
     digit *= 2;
 if (digit > 9) digit -= 9;
   }
   sum += digit;
   doubleUp = !doubleUp;
 }
 return (sum % 10) === 0;
}

// Test Data
console.log(isValidCreditCardNumber('1234567890123456')); // true
console.log(isValidCreditCardNumber('1234-5678-9012-3456')); // true
console.log(isValidCreditCardNumber('1234 5678 9012 3452')); // false
console.log(isValidCreditCardNumber('12345678901234567 89')); // false
console.log(isValidCreditCardNumber('')); // false
```

4. . Write a JavaScript program to search a date within a string.

```
function searchDateInString(inputString) {
 // Regular expression to match dates in formats like dd/mm/yyyy,
dd-mm-yyyy, dd.mm.yyyy
  // Adjust the regex pattern based on your specific date format needs
  const dateRegex = /(\d{1,2})[-./](\d{1,2})[-./](\d{4})/g;

 // Array to store found dates
  let foundDates = [];

 // Executing regex search
  let match;
 while ((match = dateRegex.exec(inputString)) !== null) {
 let day = match[1];
 let month = match[2];
 let year = match[3];
 // Creating a Date object to validate the date
   let dateObj = new Date(`${year}-${month}-${day}`);
 // Checking if the date is valid
   if (!isNaN(dateObj.getTime())) {
     foundDates.push(dateObj.toDateString());
   }
```

```
  }

 return foundDates;
}

// Test Data
let inputString = "This string contains dates like 15/07/2024, 20-12-2025,
and 10.01.2023.";
console.log(searchDateInString(inputString));
```

5. . Write a JavaScript program that work as a trim function (string) using regular expression.

```
function customTrim(inputString) {
 // Use a regular expression to replace leading and trailing whitespace
  return inputString.replace(/^\s+|\s+$/g, '');
}

// Test Data
let testString = "   Hello, World!   ";
console.log("Original String:", testString);
console.log("Trimmed String:", customTrim(testString));
```

6. Write a JavaScript program to count number of words in string. Note : - Remove white-space from start and end position. - Convert 2 or more spaces to 1. - Exclude newline with a start spacing.

```
function countWords(inputString) {
 // Step 1: Trim leading and trailing whitespace
  let trimmedString = inputString.trim();

 // Step 2: Replace multiple spaces (including tabs and newlines) with a
single space
  let normalizedString = trimmedString.replace(/\s+/g, ' ');

 // Step 3: Split the string into words by space
  let wordsArray = normalizedString.split(' ');

 // Step 4: Count number of words
  let wordCount = wordsArray.length;
```

```
 return wordCount;

}

// Test Data
let testString1 = "   Hello, World!   ";
let testString2 = "This   is   a   sentence   with   multiple   spaces.";
let testString3 = "  Leading spaces should be trimmed.";
let testString4 = "New\nline should not break word count.";

console.log("Word count:", countWords(testString1)); // Output: 3
console.log("Word count:", countWords(testString2)); // Output: 8
console.log("Word count:", countWords(testString3)); // Output: 5
console.log("Word count:", countWords(testString4)); // Output: 7
```

8. Write a JavaScript function to count the number of vowels in a given string. Test Data : console.log(alphabetize_string('United States')); Output : "SUadeeinsttt"

```
function countVowels(inputString) {
 // Step 1: Define vowels (case insensitive)
  const vowels = 'aeiouAEIOU';

 // Step 2: Initialize a counter for vowels
  let vowelCount = 0;

 // Step 3: Loop through each character in the inputString
  for (let i = 0; i < inputString.length; i++) {
 // Step 4: Check if the character is a vowel
    if (vowels.includes(inputString[i])) {
      vowelCount++;
    }
  }

 // Step 5: Return the total count of vowels found
  return vowelCount;
}

// Test Data
```

```
console.log(countVowels('United States')); // Output: 5
console.log(countVowels('Hello World')); // Output: 3
console.log(countVowels('JavaScript')); // Output: 3
console.log(countVowels('gym')); // Output: 0
```

9. Write a JavaScript function to check whether a given value is an valid url or not.

```
function isValidUrl(url) {
 // Regular expression pattern for a valid URL
    const urlPattern = /^(https?:\/\/)?([\w-]+\.)*[\w-]+(:\d+)?(\/\S*)?$/;

 // Test the provided URL against the pattern
    return urlPattern.test(url);
}

// Test cases
console.log(isValidUrl('https://www.example.com')); // true
console.log(isValidUrl('http://example.com')); // true
console.log(isValidUrl('ftp://ftp.example.com/file.txt'));// true
console.log(isValidUrl('invalid-url')); // false
console.log(isValidUrl('www.example.com')); // false
```

10. Write a JavaScript function to check whether a given value is alpha numeric or not.

```
function isAlphaNumeric(value) {
 // Regular expression pattern for alphanumeric characters
    const alphaNumericPattern = /^[a-zA-Z0-9]+$/;

 // Test the provided value against the pattern
    return alphaNumericPattern.test(value);
}

// Test cases
console.log(isAlphaNumeric('abc123')); // true
console.log(isAlphaNumeric('12345')); // true
console.log(isAlphaNumeric('abcXYZ')); // true
console.log(isAlphaNumeric('abc$123')); // false (contains $)
console.log(isAlphaNumeric('   ')); // false (contains space)
```

```
console.log(isAlphaNumeric('')); // false (empty string)
```

12. Write a JavaScript function to check whether a given value is US zip code or not.

```
function isValidUSZipCode(value) {
 // Regular expression pattern for US zip codes
    const usZipCodePattern = /^\d{5}(?:-\d{4})?$/;

 // Test the provided value against the pattern
    return usZipCodePattern.test(value);
}

// Test cases
console.log(isValidUSZipCode('12345')); // true
console.log(isValidUSZipCode('12345-6789')); // true
console.log(isValidUSZipCode('1234')); // false (less than 5 digits)
console.log(isValidUSZipCode('123456')); // false (more than 5 digits)
console.log(isValidUSZipCode('abcde')); // false (contains non-digit
characters)
console.log(isValidUSZipCode('12345-67890')); // false (more than 5 digits
after hyphen)
console.log(isValidUSZipCode('12345-67')); // false (less than 4 digits after
hyphen)
console.log(isValidUSZipCode('')); // false (empty string)
console.log(isValidUSZipCode('12345-')); // false (hyphen without following
digits)
```

13. . Write a JavaScript function to check whether a given value is UK Post Code or not.

```
function isValidUSZipCode(value) {
 // Regular expression pattern for US zip codes
    const usZipCodePattern = /^\d{5}(?:-\d{4})?$/;

 // Test the provided value against the pattern
    return usZipCodePattern.test(value);
}
```

```
// Test cases
console.log(isValidUSZipCode('12345')); // true

console.log(isValidUSZipCode('12345-6789')); // true
console.log(isValidUSZipCode('1234')); // false (less than 5 digits)
console.log(isValidUSZipCode('123456')); // false (more than 5 digits)
console.log(isValidUSZipCode('abcde')); // false (contains non-digit
characters)
console.log(isValidUSZipCode('12345-67890')); // false (more than 5 digits
after hyphen)
console.log(isValidUSZipCode('12345-67')); // false (less than 4 digits after
hyphen)
console.log(isValidUSZipCode('')); // false (empty string)
console.log(isValidUSZipCode('12345-')); // false (hyphen without following
digits)
```

15. Write a JavaScript function to check whether a given value is a social security number or not.

```
function isValidSSN(value) {
 // Regular expression pattern for US Social Security Number (SSN)
    const ssnPattern = /^(?!000|666|9\d{2})\d{3}-(?!00)\d{2}-(?!0000)\d{4}$/;

 // Test the provided value against the pattern
    return ssnPattern.test(value);
}

// Test cases
console.log(isValidSSN('123-45-6789')); // true
console.log(isValidSSN('001-01-0001')); // false (000 as area number)
console.log(isValidSSN('666-01-0001')); // false (666 as area number)
console.log(isValidSSN('900-01-0001')); // false (900 as area number)
console.log(isValidSSN('123456789')); // false (missing dashes)
console.log(isValidSSN('1234567890')); // false (no dashes and incorrect
length)
console.log(isValidSSN('123-45-678')); // false (too short)
console.log(isValidSSN('123-45-67890')); // false (too long)
console.log(isValidSSN('')); // false (empty string)
console.log(isValidSSN('1234567abc')); // false (contains non-digit
characters)
```

16. Write a JavaScript function to check whether a given value is hexadecimal value or not.

```
function isHexadecimal(value) {
 // Regular expression pattern for hexadecimal value
    const hexPattern = /^(0x)?[0-9a-fA-F]+$/;

 // Test the provided value against the pattern
    return hexPattern.test(value);
}

// Test cases
console.log(isHexadecimal('abcdef')); // true
console.log(isHexadecimal('ABCDEF')); // true
console.log(isHexadecimal('0123456789')); // true
console.log(isHexadecimal('0x123abc')); // true
console.log(isHexadecimal('0XFF')); // true
console.log(isHexadecimal('#abcdef')); // false (not part of the regex
pattern)
console.log(isHexadecimal('123xyz')); // false (contains 'xyz')
console.log(isHexadecimal('hello')); // false (non-hex characters)
console.log(isHexadecimal('')); // false (empty string)
console.log(isHexadecimal('0x')); // false (needs at least one hex digit after
'0x')
```

17. Write a JavaScript function to check whether a given value is hexcolor value or not.

```
function isHexColor(value) { // Regular expression pattern for hexadecimal
color value const hexColorPattern = /^#?([0-9a-fA-F]{3}|[0-9a-fA-F]{6})$/; //
Test the provided value against the pattern return
hexColorPattern.test(value); } // Test cases
console.log(isHexColor('#abcdef')); // true
console.log(isHexColor('#ABCDEF')); // true
console.log(isHexColor('#123')); // true (short form)
console.log(isHexColor('#fff')); // true (short form)
console.log(isHexColor('abcdef')); // true (without #)
console.log(isHexColor('123456')); // true (without #)
```

```
console.log(isHexColor('#12345')); // false (invalid length)
console.log(isHexColor('#gggggg')); // false (invalid characters)
console.log(isHexColor('#1234567')); // false (invalid length)
console.log(isHexColor('')); // false (empty string)
console.log(isHexColor('red')); // false (non-hex characters)
```

19. Write a JavaScript function to check whether a given value is html or not.

```
function isHTML(value) {
 // Regular expression pattern for HTML tags
    const htmlPattern = /<[a-z][\s\S]*>/i;

 // Test the provided value against the pattern
    return htmlPattern.test(value);
}

// Test cases
console.log(isHTML('<p>Hello, world!</p>')); // true
console.log(isHTML('<div class="container"></div>')); // true
console.log(isHTML('<a href="https://example.com">')); // true
console.log(isHTML('This is just text.')); // false
console.log(isHTML('<h1>Header</h1>')); // true
console.log(isHTML('')); // false (empty string)
console.log(isHTML('<script>alert("Hello!")</script>'));// true
```

20. Write a JavaScript function to check a given value contains alpha, dash and underscore.

```
function containsAlphaDashUnderscore(value) {
 // Regular expression pattern for alphabetic characters, dash, and
underscore
    const pattern = /[a-zA-Z_-]/;

 // Test the provided value against the pattern
    return pattern.test(value);
}

// Test cases
console.log(containsAlphaDashUnderscore('hello-world')); // true
```

```
console.log(containsAlphaDashUnderscore('hello_world')); // true
console.log(containsAlphaDashUnderscore('hello world')); // true (contains
'hello' which has alpha characters)
console.log(containsAlphaDashUnderscore('-_-')); // true
console.log(containsAlphaDashUnderscore('12345')); // false (does not
contain any alphabetic characters)
console.log(containsAlphaDashUnderscore('@#$%')); // false (does not
contain any alphabetic characters)
console.log(containsAlphaDashUnderscore('')); // false (empty string)
```

21. Write a JavaScript function to print an integer with commas as thousands separators. Test Data :
console.log(thousands_separators(1000)); "1,000"
console.log(thousands_separators(10000.23)); "10,000.23"
console.log(thousands_separators(100000)); "100,000"

```
function thousands_separators(num) {
 // Convert number to string
    let strNum = num.toString();

 // Split the number into integer and decimal parts (if any)
    let parts = strNum.split('.');

 // Add commas to integer part
    parts[0] = parts[0].replace(/\B(?=(\d{3})+(?!\d))/g, ',');

 // Join the parts back together
    return parts.join('.');
}

// Test cases
console.log(thousands_separators(1000)); // "1,000"
console.log(thousands_separators(10000.23)); // "10,000.23"
console.log(thousands_separators(100000)); // "100,000"
```

JAVASCRIPT DOM

1. <!DOCTYPE html>

```
<html>
<head>
<meta charset="utf-8" />
<title>JS DOM paragraph style</title>
<script>
function js_style() {
 // Get the paragraph element by its id
    var text = document.getElementById('text');

// Modify the style properties
    text.style.fontSize = '20px';
    text.style.fontFamily = 'Arial, sans-serif';
    text.style.color = 'blue';
}
</script>
</head>
<body>
<p id="text">JavaScript Exercises - w3resource</p>
<div>
<button id="jsstyle" onclick="js_style()">Style</button>
</div>
</body>
</html>
```

2. Write a JavaScript function to get the values of First and Last name of the following form.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Return first and last name from a form - w3resource</title>
</head>
<body>
<form id="form1" onsubmit="getFormvalue()">
   First name: <input type="text" name="fname" value="David"><br>
   Last name: <input type="text" name="lname" value="Beckham"><br>
   <input type="submit" value="Submit">
```

```
</form>

<script>
function getFormvalue() {
 // Prevent the form from submitting
    event.preventDefault();

 // Get the values of first name and last name
    var firstName =
document.getElementById('form1').elements.namedItem('fname').value;
 var lastName =
document.getElementById('form1').elements.namedItem('lname').value;

 // Display the values or do something with them
    console.log('First Name:', firstName);
 console.log('Last Name:', lastName);
}
</script>

</body>
</html>
```

3. . Write a JavaScript program to set the background color of a paragraph.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Set Paragraph Background Color</title>
  <style>
    /* Optional: Some basic styling for better visualization */
    body {
 font-family: Arial, sans-serif;
 text-align: center;
 margin-top: 50px;
    }
 p {
 padding: 20px;
 border: 1px solid #ccc;
```

```
 display: inline-block;
   }
 </style>
</head>
<body>
  <p id="paragraph">This is a paragraph.</p>
  <br>
  <button onclick="changeBackgroundColor()">Change Background
Color</button>

  <script>
    function changeBackgroundColor() {
 var paragraph = document.getElementById('paragraph');
      paragraph.style.backgroundColor = 'lightblue';
    }
 </script>
</body>
```

5. Write a JavaScript function to add rows to a table

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Insert Row in a Table</title>
</head>
<body>
  <table id="sampleTable" border="1">
    <tr>
      <td>Row1 cell1</td>
      <td>Row1 cell2</td>
    </tr>
    <tr>
      <td>Row2 cell1</td>
      <td>Row2 cell2</td>
    </tr>
  </table>
  <br>
  <input type="button" onclick="insert_Row()" value="Insert Row">
```

```
 <script>
   function insert_Row() {
 var table = document.getElementById('sampleTable');
 var newRow = table.insertRow();

 var cell1 = newRow.insertCell(0);
 var cell2 = newRow.insertCell(1);

     cell1.innerHTML = 'New Row cell1';
     cell2.innerHTML = 'New Row cell2';
   }
 </script>
</body>
</html>
```

6.. Write a JavaScript function that accept row, column, (to identify a particular cell) and a string to update the content of that cell.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Change the content of a cell</title>
</head>
<body>
  <table id="myTable" border="1">
    <tr><td>Row1 cell1</td><td>Row1 cell2</td></tr>
    <tr><td>Row2 cell1</td><td>Row2 cell2</td></tr>
    <tr><td>Row3 cell1</td><td>Row3 cell2</td></tr>
  </table>
  <form>
    <label>Row:</label>
    <input type="number" id="rowIndex" value="1">
    <label>Column:</label>
    <input type="number" id="colIndex" value="1">
    <input type="text" id="newContent" placeholder="New content">
    <input type="button" onclick="changeContent()" value="Change content">
  </form>
```

```
  <script>
    function changeContent() {
 var rowIndex = parseInt(document.getElementById('rowIndex').value) - 1; //
Adjusting for zero-based index
      var colIndex = parseInt(document.getElementById('colIndex').value) - 1; //
Adjusting for zero-based index
      var newContent = document.getElementById('newContent').value;

 var table = document.getElementById('myTable');
 var row = table.rows[rowIndex];
 var cell = row.cells[colIndex];

      cell.innerHTML = newContent;
    }
 </script>
</body>
</html>
```

7. . Write a JavaScript function that creates a table, accept row, column numbers from the user, and input row-column number as content (e.g. Row-0 Column-0) of a cell.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Create Table Dynamically</title>
<style>
  body { margin: 30px; }
 table { border-collapse: collapse; }
 table, th, td { border: 1px solid black; padding: 8px; }
</style>
</head>
<body>
  <table id="myTable"></table>
  <form>
    <label for="rows">Rows:</label>
    <input type="number" id="rows" min="1" value="3">
```

```
  <label for="cols">Columns:</label>
  <input type="number" id="cols" min="1" value="3">
  <input type="button" onclick="createTable()" value="Create Table">
 </form>

 <script>
  function createTable() {
var rows = document.getElementById('rows').value;
var cols = document.getElementById('cols').value;
var table = document.getElementById('myTable');

// Clear existing table if any
    table.innerHTML = '';

for (var i = 0; i < rows; i++) {
var row = table.insertRow();
for (var j = 0; j < cols; j++) {
var cell = row.insertCell();
        cell.textContent = `Row-${i} Column-${j}`;
      }
    }
  }
 </script>
</body>
</html>
```

9. Write a JavaScript program to count and display the items of a dropdown list, in an alert window. Sample HTML file :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Count and Display Items of a Dropdown List</title>
<style>
  body { margin: 30px; }
</style>
</head>
<body>
```

```
 <form>
   <label>Select your favorite Color:</label>
   <select id="mySelect">
     <option>Red</option>
     <option>Green</option>
     <option>Blue</option>
     <option>White</option>
   </select>
   <input type="button" onclick="getOptions()" value="Count and Output all
items">
 </form>

 <script>
   function getOptions() {
 var select = document.getElementById('mySelect');
 var options = select.options;
 var items = [];

 for (var i = 0; i < options.length; i++) {
       items.push(options[i].textContent);
     }

 var message = "Number of items: " + options.length + "\n\n";
     message += "Items:\n";
     message += items.join("\n");

 alert(message);
   }
 </script>
</body>
</html>
```

11. Write a JavaScript program to display a random image (clicking on a button) from the following list. Sample Image information : "http://farm4.staticflickr.com/3691/11268502654_f28f05966c_m.jpg", width: "240", height: "160"
"http://farm1.staticflickr.com/33/45336904_1aef569b30_n.jpg", width: "320", height: "195"

"http://farm6.staticflickr.com/5211/5384592886_80a512e2c9.jpg", width: "500", height: "343"

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Random Image Display</title>
<style>
  body { text-align: center; margin-top: 50px; }
 img { margin-top: 20px; }
</style>
</head>
<body>
  <h2>Random Image Display</h2>
  <button onclick="displayRandomImage()">Show Random Image</button>
  <div id="imageContainer"></div>

  <script>
    // Array of image information objects
    var images = [
      { src: "http://farm4.staticflickr.com/3691/11268502654_f28f05966c_m.jpg",
width: "240", height: "160" },
      { src: "http://farm1.staticflickr.com/33/45336904_1aef569b30_n.jpg", width:
"320", height: "195" },
      { src: "http://farm6.staticflickr.com/5211/5384592886_80a512e2c9.jpg", width:
"500", height: "343" }
    ];

 // Function to display a random image
    function displayRandomImage() {
 var randomIndex = Math.floor(Math.random() * images.length); // Get random
index
      var randomImage = images[randomIndex]; // Get random image object

      var imageElement = document.createElement('img'); // Create <img> element
      imageElement.src = randomImage.src; // Set image src
      imageElement.width = randomImage.width; // Set image width
      imageElement.height = randomImage.height; // Set image height

      var imageContainer = document.getElementById('imageContainer');
```

```
    imageContainer.innerHTML = ''; // Clear previous image (if any)
    imageContainer.appendChild(imageElement); // Append new image
   }
 </script>
</body>
</html>
```

12. Write a JavaScript program to highlight the bold words of the following paragraph, on mouse over a certain link. Sample link and text : [On mouse over here bold words of the following paragraph will be highlighted] We have just started this section for the users (beginner to intermediate) who want to work with various JavaScript problems and write scripts online to test their JavaScript skill.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Highlight Bold Words on Mouse Over</title>
<style>
  body { font-family: Arial, sans-serif; }
 #highlightLink { color: blue; text-decoration: underline; cursor: pointer; }
 .highlighted { background-color: yellow; }
</style>
</head>
<body>
  <p id="paragraph">
    We have just started this section for the users (beginner to intermediate)
    who want to work with various JavaScript problems and write scripts online
    to test their JavaScript skill.
 </p>
  <div id="highlightDiv">
    <a id="highlightLink" onmouseover="highlightWords()">On mouse over here
bold words of the following paragraph will be highlighted</a>
  </div>

  <script>
    function highlightWords() {
 var paragraph = document.getElementById('paragraph');
 var words = paragraph.getElementsByTagName('strong');
```

```
 for (var i = 0; i < words.length; i++) {
       words[i].classList.add('highlighted');
     }
   }
 </script>
</body>
</html>
```

13. Write a JavaScript program to get the width and height of the window (any time the window is resized)

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Window Resize Dimensions</title>
<style>
  body { font-family: Arial, sans-serif; }
 #dimensions { margin-top: 20px; }
</style>
</head>
<body>
  <h2>Window Resize Dimensions</h2>
  <div id="dimensions">
    Width: <span id="width"></span> px<br>
    Height: <span id="height"></span> px
 </div>

  <script>
    // Function to update width and height
    function updateDimensions() {
 var widthSpan = document.getElementById('width');
 var heightSpan = document.getElementById('height');

 // Set width and height values
      widthSpan.textContent = window.innerWidth;
      heightSpan.textContent = window.innerHeight;
    }
```

```
// Initial call to set dimensions on page load
   updateDimensions();

// Add event listener for window resize
   window.addEventListener('resize', updateDimensions);
</script>
</body>
</html>
```

JAVASCRIPT DRAWING

1. **Write a JavaScript program to draw the following rectangular shape.**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Draw Rectangular Shape</title>
<style>
  body { font-family: Arial, sans-serif; }
 canvas { border: 1px solid #000; }
</style>
</head>
<body>
  <h2>Rectangular Shape</h2>
  <canvas id="myCanvas" width="400" height="200"></canvas>

  <script>
    // Get the canvas element
    var canvas = document.getElementById('myCanvas');
 // Get the drawing context for the canvas
    var ctx = canvas.getContext('2d');

 // Set rectangle properties
    var x = 50; // x-coordinate of top-left corner
    var y = 50; // y-coordinate of top-left corner
    var width = 300; // Width of the rectangle
    var height = 100; // Height of the rectangle
```

```
    // Draw the rectangle
    ctx.fillStyle = '#FF0000'; // Fill color
    ctx.fillRect(x, y, width, height);
  </script>
</body>
</html>
```

2. **Write a JavaScript program to draw a circle.**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Draw Red Circle</title>
<style>
  body { font-family: Arial, sans-serif; }
 canvas { border: 1px solid #000; }
</style>
</head>
<body>
  <h2>Red Circle</h2>
  <canvas id="myCanvas" width="400" height="400"></canvas>

  <script>
    // Get the canvas element
    var canvas = document.getElementById('myCanvas');
 // Get the drawing context for the canvas
    var ctx = canvas.getContext('2d');

 // Set circle properties
    var centerX = canvas.width / 2; // X-coordinate of circle center
    var centerY = canvas.height / 2; // Y-coordinate of circle center
    var radius = 100; // Radius of the circle
    var startAngle = 0; // Start angle (radians)
    var endAngle = Math.PI * 2; // End angle (full circle in radians)
    var anticlockwise = false; // Drawing direction (clockwise)

    // Draw the circle
    ctx.beginPath();
```

```
    ctx.arc(centerX, centerY, radius, startAngle, endAngle, anticlockwise);
    ctx.fillStyle = '#FF0000'; // Fill color red
    ctx.fill();
    ctx.closePath();
 </script>
</body>
</html>
```

3. **. Write a JavaScript program to draw two intersecting rectangles, one of which has alpha transparency.**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Draw Intersecting Rectangles</title>
<style>
  body { font-family: Arial, sans-serif; }
 canvas { border: 1px solid #000; }
</style>
</head>
<body>
  <h2>Intersecting Rectangles</h2>
  <canvas id="myCanvas" width="400" height="400"></canvas>

  <script>
    // Get the canvas element
    var canvas = document.getElementById('myCanvas');
 // Get the drawing context for the canvas
    var ctx = canvas.getContext('2d');

 // Set rectangle properties
    var rect1 = {
 x: 50,
 y: 50,
 width: 200,
 height: 150,
 color: '#FF0000'
    };
```

```
var rect2 = {
x: 150,
y: 100,
width: 200,
height: 150,
color: 'rgba(0, 0, 255, 0.5)' // Transparent blue (alpha = 0.5)
  };

// Function to draw a rectangle
  function drawRect(rect) {
    ctx.fillStyle = rect.color;
    ctx.fillRect(rect.x, rect.y, rect.width, rect.height);
  }

// Draw the rectangles
  drawRect(rect1);
drawRect(rect2);
</script>
</body>
</html>
```

4.. Write a JavaScript program to draw the following right-angled triangle.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Draw Right-Angled Triangle</title>
<style>
  body { font-family: Arial, sans-serif; }
 canvas { border: 1px solid #000; }
</style>
</head>
<body>
  <h2>Right-Angled Triangle</h2>
  <canvas id="myCanvas" width="400" height="400"></canvas>

  <script>
    // Get the canvas element
```

```
   var canvas = document.getElementById('myCanvas');
 // Get the drawing context for the canvas
   var ctx = canvas.getContext('2d');

 // Set triangle properties
   var triangle = {
 x: 50, // Starting x-coordinate of the triangle
     y: 50, // Starting y-coordinate of the triangle
     size: 150, // Size of the triangle (height of the perpendicular side)
     color: '#000'  // Color of the triangle (black)
   };

 // Function to draw a right-angled triangle
   function drawTriangle(triangle) {
     ctx.beginPath();
     ctx.moveTo(triangle.x, triangle.y); // Starting point (bottom-left corner)
     ctx.lineTo(triangle.x, triangle.y - triangle.size); // Draw the vertical side
     ctx.lineTo(triangle.x + triangle.size, triangle.y); // Draw the horizontal
side
     ctx.closePath();

     ctx.fillStyle = triangle.color; // Fill color of the triangle
     ctx.fill(); // Fill the triangle with the color
   }

 // Draw the triangle
   drawTriangle(triangle);
 </script>
</body>
</html>
```

6. Write a JavaScript program to draw the following diagram [diagonal, white to black circles].

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Draw Diagonal Circles (White to Black)</title>
<style>
```

```
 body { font-family: Arial, sans-serif; }
canvas { border: 1px solid #000; }
</style>
</head>
<body>
 <h2>Diagonal Circles (White to Black)</h2>
 <canvas id="myCanvas" width="400" height="400"></canvas>

 <script>
  // Get the canvas element
  var canvas = document.getElementById('myCanvas');
// Get the drawing context for the canvas
  var ctx = canvas.getContext('2d');

// Define circle properties
  var circle = {
x: 50, // Starting x-coordinate of the first circle
    y: 50, // Starting y-coordinate of the first circle
    radius: 20, // Radius of each circle
    gap: 10, // Gap between circles
    rows: 10, // Number of rows (adjust for the desired number)
    cols: 10, // Number of columns (adjust for the desired number)
    startColor: '#ffffff', // Starting color (white)
    endColor: '#000000'     // Ending color (black)
  };

// Function to draw circles with gradient from white to black
  function drawCircles(circle) {
var deltaRed = (parseInt(circle.endColor.substring(1, 3), 16) -
parseInt(circle.startColor.substring(1, 3), 16)) / (circle.rows + circle.cols -
1);
var deltaGreen = (parseInt(circle.endColor.substring(3, 5), 16) -
parseInt(circle.startColor.substring(3, 5), 16)) / (circle.rows + circle.cols -
1);
var deltaBlue = (parseInt(circle.endColor.substring(5, 7), 16) -
parseInt(circle.startColor.substring(5, 7), 16)) / (circle.rows + circle.cols -
1);

for (var row = 0; row < circle.rows; row++) {
for (var col = 0; col < circle.cols; col++) {
```

```
 var currentColor = '#' +
 Math.round(parseInt(circle.startColor.substring(1, 3), 16) + row *
deltaRed).toString(16) +
 Math.round(parseInt(circle.startColor.substring(3, 5), 16) + row *
deltaGreen).toString(16) +
 Math.round(parseInt(circle.startColor.substring(5, 7), 16) + row *
deltaBlue).toString(16);

        ctx.beginPath();
        ctx.arc(circle.x + col * (circle.radius * 2 + circle.gap), circle.y + row *
(circle.radius * 2 + circle.gap), circle.radius, 0, 2 * Math.PI);
        ctx.fillStyle = currentColor;
        ctx.fill();
      }
    }
  }

 // Draw the circles
   drawCircles(circle);
 </script>
</body>
</html>
```

JAVASCRIPT OBJECT

1. **Write a JavaScript program to list the properties of a JavaScript object. Sample object : var student = { name : "David Rayy", sclass : "VI", rollno : 12 }; Sample Output : name,sclass,rollno**

```
function listProperties(obj) {
 return Object.keys(obj).join(',');
}

// Sample object
var student = {
 name: "David Rayy",
 sclass: "VI",
 rollno: 12
};
```

```
// Output the properties
console.log(listProperties(student)); // Output: name,sclass,rollno
```

2. **Write a JavaScript program to delete the rollno property from the following object. Also print the object before or after deleting the property. Sample object : var student = { name : "David Rayy", sclass : "VI", rollno : 12 };**

```
var student = {
 name: "David Rayy",
 sclass: "VI",
 rollno: 12
};

// Print object before deletion
console.log("Object before deletion:");
console.log(student);

// Delete rollno property
delete student.rollno;

// Print object after deletion
console.log("\nObject after deletion:");
console.log(student);
```

3. **Write a JavaScript program to get the length of an JavaScript object. Sample object : var student = { name : "David Rayy", sclass : "VI", rollno : 12 };**

```
var student = {
 name: "David Rayy",
 sclass: "VI",
 rollno: 12
};

// Get the number of properties
var objectLength = Object.keys(student).length;
```

```
// Print the length
console.log("Length of the object:", objectLength);
```

5. Write a JavaScript program to get the volume of a Cylinder with four decimal places using object classes. Volume of a cylinder : V = πr2h where r is the radius and h is the height of the cylinder.

```
class Cylinder {
 constructor(radius, height) {
 this.radius = radius;
 this.height = height;
   }

 // Method to calculate the volume of the cylinder
   volume() {
 const pi = Math.PI;
 return parseFloat((pi * Math.pow(this.radius, 2) *
this.height).toFixed(4));
   }
}

// Example usage:
let cylinder = new Cylinder(5, 10); // Radius = 5, Height = 10
console.log("Volume of the cylinder:", cylinder.volume()); // Output:
Volume of the cylinder: 785.3982
```

7. Write a JavaScript program which returns a subset of a string. Sample Data : dog Expected Output : ["d", "do", "dog", "o", "og", "g"]

```
function substrings(str) {
 let result = [];
 // Loop to extract substrings
   for (let i = 0; i < str.length; i++) {
 for (let j = i + 1; j <= str.length; j++) {
          result.push(str.slice(i, j));
      }
   }
 return result;
}
```

```
// Test the function
let str = "dog";
console.log(substrings(str));
```

8. Write a JavaScript program to create a Clock. Note : The output will come every second. Expected Console Output : "14:37:42" "14:37:43" "14:37:44" "14:37:45" "14:37:46" "14:37:47"

```
function clock() {
 // Function to get current time
    function getCurrentTime() {
 let now = new Date();
 let hours = now.getHours().toString().padStart(2, '0');
 let minutes = now.getMinutes().toString().padStart(2, '0');
 let seconds = now.getSeconds().toString().padStart(2, '0');
 return `${hours}:${minutes}:${seconds}`;
    }

 // Update the clock every second
    setInterval(function() {
 console.log(getCurrentTime());
    }, 1000);
}

// Start the clock
clock();
```

9. Write a JavaScript program to calculate the area and perimeter of a circle. Note : Create two methods to calculate the area and perimeter. The radius of the circle will be supplied by the user.

```
function calculateArea(radius) {
 let area = Math.PI * radius * radius;
 return area.toFixed(2); // Round to 2 decimal places
}

// Function to calculate perimeter (circumference) of a circle
```

```
function calculatePerimeter(radius) {
 let perimeter = 2 * Math.PI * radius;
 return perimeter.toFixed(2); // Round to 2 decimal places
}

// Example usage:
let radius = parseFloat(prompt("Enter the radius of the circle:"));

// Check if the input is valid (a positive number)
if (isNaN(radius) || radius <= 0) {
 console.log("Invalid input. Please enter a valid positive number for radius.");
} else {
 // Calculate and display the area and perimeter
    let area = calculateArea(radius);
 let perimeter = calculatePerimeter(radius);

 console.log(`Radius of the circle: ${radius}`);
 console.log(`Area of the circle: ${area}`);
 console.log(`Perimeter (Circumference) of the circle: ${perimeter}`);
}
```

11. Write a JavaScript function to print all the methods in an JavaScript object. Test Data : console.log(all_properties(Array)); ["length", "name", "arguments", "caller", "prototype", "isArray", "observe", "unobserve"]

```
function all_properties(obj) {
 // Array to store method names
    let methods = [];

 // Iterate through the prototype chain
    for (let prop in obj.prototype) {
 // Check if the property is a function (method)
        if (typeof obj.prototype[prop] === 'function') {
            methods.push(prop);
        }
    }

 return methods;
}
```

```
// Test with Array object
console.log(all_properties(Array));
```

JAVASCRIPT VALIDATION

1. **Write a JavaScript function to validate whether a given value type is boolean or not.**

   ```
   function isBoolean(value) {
    return typeof value === 'boolean';
   }

   // Test cases
   console.log(isBoolean(true)); // true
   console.log(isBoolean(false)); // true
   console.log(isBoolean('true')); // false
   console.log(isBoolean(1)); // false
   console.log(isBoolean(null)); // false
   ```

2. **. Write a JavaScript function to validate whether a given value type is error or not.**

   ```
   function isError(value) {
    return value instanceof Error;
   }

   // Test cases
   console.log(isError(new Error())); // true
   console.log(isError(new TypeError())); // true (TypeError is a subclass of Error)
   console.log(isError(new SyntaxError())); // true (SyntaxError is a subclass of Error)
   console.log(isError('Error message')); // false
   console.log(isError(404)); // false
   console.log(isError(null)); // false
   ```

3. **Write a JavaScript function to validate whether a given value type is NaN or not.**

```
function isStrictlyNaN(value) {
 return typeof value === 'number' && isNaN(value);
}

// Test cases
console.log(isStrictlyNaN(NaN)); // true
console.log(isStrictlyNaN(123)); // false
console.log(isStrictlyNaN('abc')); // false
console.log(isStrictlyNaN(undefined)); // false
console.log(isStrictlyNaN(null)); // false
console.log(isStrictlyNaN({})); // false
```

4. **Write a JavaScript function to validate whether a given value type is null or not**

```
function isNull(value) {
 return value === null;
}

// Test cases
console.log(isNull(null)); // true
console.log(isNull(undefined)); // false
console.log(isNull(0)); // false
console.log(isNull('')); // false
console.log(isNull({})); // false
console.log(isNull([])); // false
```

5. **Write a JavaScript function to validate whether a given value is number or not.**

```
function isNumeric(value) {
 return typeof value === 'number' && !Number.isNaN(value);
}

// Test cases
console.log(isNumeric(123)); // true
console.log(isNumeric('123')); // false (string)
console.log(isNumeric('abc')); // false (string)
```

```
console.log(isNumeric(true)); // false (boolean)
console.log(isNumeric(null)); // false (null)
console.log(isNumeric(undefined)); // false (undefined)
console.log(isNumeric({})); // false (object)
console.log(isNumeric([])); // false (array)
console.log(isNumeric(function(){}));// false (function)
```

6. **. Write a JavaScript function to validate whether a given value is object or not.**

```
function isObject(value) {
 return value !== null && typeof value === 'object';
}

// Test cases
console.log(isObject({})); // true
console.log(isObject([])); // true (arrays are objects in JavaScript)
console.log(isObject(null)); // false (null is not an object)
console.log(isObject(123)); // false (numbers are not objects)
console.log(isObject('string')); // false (strings are not objects)
console.log(isObject(true)); // false (booleans are not objects)
console.log(isObject(undefined)); // false (undefined is not an object)
console.log(isObject(function(){})); // false (functions are not objects in this
context)
```

7. **Write a JavaScript function to validate whether a given value type is pure json object or not.**

```
function isPureJSONObject(value) {
 // Check if value is not null and is of type 'object'
   if (value !== null && typeof value === 'object') {
 // Check if it is not an array
       if (!Array.isArray(value)) {
 // Check if its prototype is Object.prototype
           if (Object.getPrototypeOf(value) === Object.prototype) {
 return true;
           }
       }
   }
```

```
 return false;
}

// Test cases
console.log(isPureJSONObject({})); // true
console.log(isPureJSONObject({ key: 'value' })); // true
console.log(isPureJSONObject([])); // false (arrays are not pure JSON
objects)
console.log(isPureJSONObject(null)); // false (null is not an object)
console.log(isPureJSONObject(123)); // false (numbers are not objects)
console.log(isPureJSONObject('string')); // false (strings are not objects)
console.log(isPureJSONObject(true)); // false (booleans are not objects)
console.log(isPureJSONObject(undefined)); // false (undefined is not an
object)
console.log(isPureJSONObject(function(){})); // false (functions are not
objects in this context)
```

8. **Write a JavaScript function to validate whether a given value is RegExp or not.**

```
function isRegExp(value) {
 return value instanceof RegExp;
}

// Test cases
console.log(isRegExp(/test/)); // true
console.log(isRegExp(new RegExp())); // true
console.log(isRegExp('not a RegExp')); // false
console.log(isRegExp(123)); // false
console.log(isRegExp(null)); // false
console.log(isRegExp(undefined)); // false
```

9. **Write a JavaScript function to validate whether a given value type is char or not.**

```
function isChar(value) {
 return typeof value === 'string' && value.length === 1;
}

// Test cases
```

```
console.log(isChar('a')); // true
console.log(isChar('')); // false (empty string)
console.log(isChar('ab')); // false (more than one character)
console.log(isChar(123)); // false (not a string)
console.log(isChar(true)); // false (not a string)
console.log(isChar(null)); // false (not a string)
console.log(isChar(undefined)); // false (not a string)
```

10. . Write a JavaScript function to check whether given value types are same or not.

```
function areSameType(value1, value2) {
 return typeof value1 === typeof value2;
}

// Test cases
console.log(areSameType(5, 10)); // true (both numbers)
console.log(areSameType('hello', 'world')); // true (both strings)
console.log(areSameType(true, false)); // true (both booleans)
console.log(areSameType(null, undefined)); // false (different types)
console.log(areSameType(5, '5')); // false (number vs string)
console.log(areSameType({}, [])); // false (object vs array)
```

IMPORTANT QUESTIONS

MOVIE DATABASE

```
let favoriteMovie = {
 title: "Puff the Magic Dragon",
 duration: 30, // duration in minutes
    stars: ["Puff", "Jackie", "Living Sneezes"]
};

// Define the function to print movie information
function printMovieInfo(movie) {
 let starsString = movie.stars.join(", ");
 console.log(`${movie.title} lasts for ${movie.duration} minutes. Stars:
${starsString}.`);
}
```

```
// Call the function to print the information of the favorite movie
printMovieInfo(favoriteMovie);
```

2. Keep track of which books you read and which books you want to read! Create an array of objects, where each object describes a book and has properties for the title (a string), author (a string), and alreadyRead (a boolean indicating if you read it yet). Iterate through the array of books. For each book, log the book title and book author like so: "The Hobbit by J.R.R. Tolkien". Now use an if/else statement to change the output depending on whether you read it yet or not. If you read it, log a string like 'You already read "The Hobbit" by J.R.R. Tolkien', and if not, log a string like 'You still need to read "The Lord of the Rings" by J.R.R. Tolkien.'

```
let books = [
    { title: "The Hobbit", author: "J.R.R. Tolkien",
alreadyRead: true },
    { title: "The Lord of the Rings", author: "J.R.R.
Tolkien", alreadyRead: false },
    { title: "1984", author: "George Orwell",
alreadyRead: true },
    { title: "To Kill a Mockingbird", author: "Harper
Lee", alreadyRead: false },
    { title: "The Great Gatsby", author: "F. Scott
Fitzgerald", alreadyRead: true }
];

// Iterate through the array of books
books.forEach(book => {
    // Log the book title and author
    console.log(`${book.title} by ${book.author}`);

    // Use an if/else statement to change the output
based on alreadyRead property
    if (book.alreadyRead) {
```

```
        console.log(`You already read "${book.title}"
by ${book.author}.`);
    } else {
        console.log(`You still need to read
"${book.title}" by ${book.author}.`);
    }
});
```

3. **The Recipe Card Never forget another recipe! Create an object to hold information on your favorite recipe. It should have properties for title (a string), servings (a number), and ingredients (an array of strings). On separate lines (one console.log statement for each), log the recipe information so it looks like: Mole Serves: 2 Ingredients: cinnamon cumin cocoa**

```
let favoriteRecipe = {
 title: "Mole",
 servings: 2,
 ingredients: ["cinnamon", "cumin", "cocoa"]
};

// Log the recipe information
console.log(favoriteRecipe.title);
console.log(`Serves: ${favoriteRecipe.servings}`);
console.log("Ingredients:");
favoriteRecipe.ingredients.forEach(ingredient => {
 console.log(`- ${ingredient}`);
});
```

4. The Fortune Teller Why pay a fortune teller when you can just program your fortune yourself? Write a function named tellFortune that: takes 4 arguments: number of children, partner's name, geographic location, job title. outputs your fortune to the screen like so: "You will be a X in Y, and married to Z with N kids." Call that function 3 times with 3 different values for the arguments.

```
// Function to tell fortune
function tellFortune(numChildren, partnerName, geoLocation, jobTitle) {
```

```
 console.log(`You will be a ${jobTitle} in ${geoLocation}, and married to
${partnerName} with ${numChildren} kids.`);
}

// Call the function three times with different values
tellFortune(2, 'Alice', 'New York', 'Software Developer');
tellFortune(3, 'Bob', 'London', 'Graphic Designer');
tellFortune(1, 'Charlie', 'Sydney', 'Architect');
```

5. The Age Calculator Forgot how old you are? Calculate it! Write a function named calculateAge that: takes 2 arguments: birth year, current year. calculates the 2 possible ages based on those years. outputs the result to the screen like so: "You are either NN or NN" Call the function three times with different sets of values. Bonus: Figure out how to get the current year in JavaScript instead of passing it in.

```
function calculateAge(birthYear, currentYear) {
 const age1 = currentYear - birthYear;
 const age2 = age1 - 1;
 console.log(`You are either ${age2} or ${age1}`);
}

// Call the function three times with different values
calculateAge(1990, 2024);
calculateAge(1985, 2024);
calculateAge(2000, 2024);

// Bonus: Automatically get the current year
function calculateAgeWithCurrentYear(birthYear) {
 const currentYear = new Date().getFullYear();
 const age1 = currentYear - birthYear;
 const age2 = age1 - 1;
 console.log(`You are either ${age2} or ${age1}`);
}

// Call the function three times with different birth years
calculateAgeWithCurrentYear(1990);
calculateAgeWithCurrentYear(1985);
calculateAgeWithCurrentYear(2000);
```

7. The Geometrizer Create 2 functions that calculate properties of a circle, using the definitions here. Create a function called calcCircumfrence : Pass the radius to the function. Calculate the circumference based on the radius, and output "The circumference is NN". Create a function called calcArea : Pass the radius to the function. Calculate the area based on the radius, and output "The area is NN"

```
function calcCircumference(radius) {
 const circumference = 2 * Math.PI * radius;
 console.log(`The circumference is ${circumference.toFixed(2)}`);
}

// Function to calculate the area of a circle
function calcArea(radius) {
 const area = Math.PI * Math.pow(radius, 2);
 console.log(`The area is ${area.toFixed(2)}`);
}

// Test the functions with a sample radius
const radius = 5;
calcCircumference(radius);
calcArea(radius);

// You can call these functions with different radii as needed
calcCircumference(10);
calcArea(10);
```

8. The Temperature Converter It's hot out! Let's make a converter based on the steps here. Create a function called celsiusToFahrenheit : Store a celsius temperature into a variable. Convert it to fahrenheit and output "NN°C is NN°F". Create a function called fahrenheitToCelsius : Now store a fahrenheit temperature into a variable. Convert it to celsius and output "NN°F is NN°C.

```
function celsiusToFahrenheit(celsius) {
 const fahrenheit = (celsius * 9/5) + 32;
 console.log(`${celsius}°C is ${fahrenheit.toFixed(2)}°F`);
}
```

```
// Function to convert Fahrenheit to Celsius
function fahrenheitToCelsius(fahrenheit) {
 const celsius = (fahrenheit - 32) * 5/9;
 console.log(`${fahrenheit}°F is ${celsius.toFixed(2)}°C`);
}

// Test the functions with sample temperatures
const celsiusTemp = 25;
const fahrenheitTemp = 77;

celsiusToFahrenheit(celsiusTemp);
fahrenheitToCelsius(fahrenheitTemp);

// Additional test cases
celsiusToFahrenheit(0); // Freezing point of water
fahrenheitToCelsius(32); // Freezing point of water in Fahrenheit
```

OTHER QUESTIONS

1. **Create an HTML page. Add JavaScript that includes 2 prompts for integer values from the user. Add JavaScript to add these 2 values together and display the result in an alert box**

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>Sum of Two Integers</title>
   <script>
      function sumTwoIntegers() {
 // Prompt the user for the first integer value
         var firstValue = prompt("Enter the first integer value:");
 // Convert the input to an integer
         var firstInteger = parseInt(firstValue, 10);

 // Prompt the user for the second integer value
         var secondValue = prompt("Enter the second integer value:");
 // Convert the input to an integer
         var secondInteger = parseInt(secondValue, 10);
```

```
// Check if the inputs are valid integers
        if (isNaN(firstInteger) || isNaN(secondInteger)) {
alert("Please enter valid integer values.");
        } else {
// Add the two integer values
            var sum = firstInteger + secondInteger;
// Display the result in an alert box
            alert("The sum of the two integers is: " + sum);
        }
    }

// Call the function when the page loads
    window.onload = sumTwoIntegers;
</script>
</head>
<body>
</body>
</html>
```

2. **Write a JavaScript function that returns a passed string with letters in alphabetical order.**

```
function alphabetizeString(str) {
 // Convert the string to an array of characters
   var charArray = str.split('');

 // Sort the array of characters
   var sortedArray = charArray.sort();

 // Join the sorted array back into a string
   var sortedString = sortedArray.join('');

 // Return the sorted string
   return sortedString;
}

// Test the function
console.log(alphabetizeString('javascript')); // "aacijprstv"
console.log(alphabetizeString('hello')); // "ehllo"
```

```
console.log(alphabetizeString('alphabet')); // "aabehlpt"
```

3. **Write a JavaScript function to extract unique characters from a string “thequickbrownfoxjumpsoverthelazydog”.**

```
function extractUniqueCharacters(str) {
 // Create a Set to store unique characters
   let uniqueChars = new Set();

 // Loop through each character in the string
   for (let char of str) {
      uniqueChars.add(char);
   }

 // Convert the Set back to a string
   return Array.from(uniqueChars).join('');
}

// Test the function
console.log(extractUniqueCharacters('thequickbrownfoxjumpsoverthelazy
dog'));
```

4. **Create an HTML page. Add a JavaScript function to convert local time and date to UTC and display the result. Add a button to your page that calls this function. Add JavaScript to create a timer that displays the current date and time after a specified amount of time delay. Use a button (or other event) to call the timer function**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Time Conversion and Timer</title>
</head>
<body>
   <h1>Time Conversion and Timer</h1>
   <p id="localTime"></p>
   <p id="utcTime"></p>
```

```
    <button onclick="convertToUTC()">Convert to UTC</button>
    <button onclick="startTimer()">Start Timer</button>

    <script>
        // Function to convert local time to UTC
        function convertToUTC() {
 const now = new Date();
 const utcTimeString = now.toISOString();
 document.getElementById('utcTime').innerText = `UTC Time:
${utcTimeString}`;
        }
```

5. **Create a basic page in html that consists of one image. Using the onClick event handler, when the user clicks on the image, change it to a unique image**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Image Click Change</title>
    <style>
        body {
 text-align: center;
 margin-top: 50px;
        }
 img {
 cursor: pointer;
 max-width: 100%;
 transition: transform 0.3s ease-in-out;
        }
 img:hover {
 transform: scale(1.1);
        }
 </style>
</head>
<body>
    <h1>Click to Change Image</h1>
```

```
    <p>Click on the image to see the magic!</p>

    <img id="myImage" src="image1.jpg" alt="Image 1"
onclick="changeImage()">

    <script>
        // Function to change the image source
        function changeImage() {
 const img = document.getElementById('myImage');
 if (img.src.match(/image1\.jpg$/)) {
                img.src = 'image2.jpg'; // Replace with your second image path
            } else {
                img.src = 'image1.jpg'; // Replace with your first image path
            }
        }
 </script>
</body>
</html>
```

6. **Reconfigure the previous question to execute using a function. Specifically, when the user clicks on the button, the onClick event handler calls a function. The function then executes the statement that changes the background color to blue.**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Change Background Color</title>
    <style>
        body {
 text-align: center;
 margin-top: 50px;
        }
 button {
 padding: 10px 20px;
 font-size: 16px;
```

```
 cursor: pointer;
      }
 #content {
 margin-top: 20px;
 font-size: 18px;
      }
 .blue-bg {
 background-color: blue;
 color: white;
      }
 </style>
</head>
<body>
   <h1>Change Background Color Example</h1>
   <button onclick="changeBackgroundColor()">Change Background
Color to Blue</button>
   <div id="content">
      <p>Click the button above to change the background color!</p>
   </div>

   <script>
      function changeBackgroundColor() {
 document.body.classList.toggle('blue-bg');
      }
 </script>
</body>
</html>
```

7. **Create an HTML page that includes 2 paragraphs of text. Add JavaScript to insert a new paragraph in between the other 2. Add JavaScript to use an event to change the style of the new paragraph using either the style property or the className property. Add JavaScript to use an event to change the style of the new paragraph back to the original.**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Dynamic Paragraph Styling</title>
    <style>
        body {
 font-family: Arial, sans-serif;
 text-align: center;
 margin-top: 50px;
        }
 #container {
 width: 60%;
 margin: 0 auto;
 border: 1px solid #ccc;
 padding: 20px;
 background-color: #f9f9f9;
        }
 .original {
 color: black;
        }
 .highlighted {
 color: red;
 font-weight: bold;
        }
 </style>
</head>
<body>
    <div id="container">
        <p class="original">First paragraph of text.</p>
        <p class="original">Second paragraph of text.</p>
    </div>

    <script>
        // Function to create and insert a new paragraph
        function insertNewParagraph() {
 var container = document.getElementById('container');
 var newParagraph = document.createElement('p');
            newParagraph.textContent = 'New paragraph inserted between the
existing ones.';
            container.insertBefore(newParagraph, container.children[1]); //
Insert between the 2nd and 3rd child
```

```
            // Add event listeners for mouseover and mouseout events
            newParagraph.addEventListener('mouseover', function() {
                newParagraph.classList.add('highlighted');
            });

            newParagraph.addEventListener('mouseout', function() {
                newParagraph.classList.remove('highlighted');
            });
        }
    </script>

      <button onclick="insertNewParagraph()">Insert New
    Paragraph</button>
    </body>
    </html>
```

8.) Create a slide show of 5 images.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Slideshow with JavaScript</title>
    <style>
        body {
 font-family: Arial, sans-serif;
 text-align: center;
 margin-top: 50px;
        }
 .slideshow-container {
 position: relative;
 max-width: 600px;
 margin: 0 auto;
        }
 .mySlides {
 display: none;
 width: 100%;
        }
```

```
 </style>
</head>
<body>
   <h2>Simple Slideshow with JavaScript</h2>

   <div class="slideshow-container">
      <img class="mySlides" src="image1.jpg">
      <img class="mySlides" src="image2.jpg">
      <img class="mySlides" src="image3.jpg">
      <img class="mySlides" src="image4.jpg">
      <img class="mySlides" src="image5.jpg">
   </div>

   <script>
      var slideIndex = 0;
 showSlides();

 function showSlides() {
 var slides = document.getElementsByClassName("mySlides");

 // Hide all slides
          for (var i = 0; i < slides.length; i++) {
             slides[i].style.display = "none";
          }

 // Increment slide index and reset if necessary
          slideIndex++;
 if (slideIndex > slides.length) {
             slideIndex = 1;
          }

 // Display current slide
          slides[slideIndex - 1].style.display = "block";

 // Change slide every 2 seconds (2000 milliseconds)
          setTimeout(showSlides, 2000);
       }
 </script>

</body>
```

```
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Slideshow with JavaScript</title>
    <style>
        body {
 font-family: Arial, sans-serif;
 text-align: center;
 margin-top: 50px;
        }
 .slideshow-container {
 position: relative;
 max-width: 600px;
 margin: 0 auto;
        }
 .mySlides {
 display: none;
 width: 100%;
        }
 </style>
</head>
<body>
    <h2>Simple Slideshow with JavaScript</h2>

    <div class="slideshow-container">
        <img class="mySlides" src="image1.jpg">
        <img class="mySlides" src="image2.jpg">
        <img class="mySlides" src="image3.jpg">
        <img class="mySlides" src="image4.jpg">
        <img class="mySlides" src="image5.jpg">
    </div>

    <script>
        var slideIndex = 0;
 showSlides();
```

```
function showSlides() {
var slides = document.getElementsByClassName("mySlides");

// Hide all slides
        for (var i = 0; i < slides.length; i++) {
           slides[i].style.display = "none";
        }

// Increment slide index and reset if necessary
        slideIndex++;
if (slideIndex > slides.length) {
           slideIndex = 1;
        }

// Display current slide
        slides[slideIndex - 1].style.display = "block";

// Change slide every 2 seconds (2000 milliseconds)
        setTimeout(showSlides, 2000);
      }
</script>

</body>
</html>
```

9. **) Use previous question images and add two image button (back and forward). Back button allows the user to move backward through the slide show one slide at a time. When the user reaches the end (or beginning when clicking on the back button) of the slide show, the slide show should not wrap around to the beginning (or end) and forward button should show next image. When at the beginning of the slide show, only the forward image button should be visible. Conversely, when at the end, only the back image button should be visible.**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
```

```
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Slideshow with Navigation Buttons</title>
   <style>
      body {
font-family: Arial, sans-serif;
text-align: center;
margin-top: 50px;
      }
.slideshow-container {
position: relative;
max-width: 600px;
margin: 0 auto;
      }
.mySlides {
display: none;
width: 100%;
      }
.prev, .next {
cursor: pointer;
position: absolute;
top: 50%;
width: auto;
padding: 16px;
margin-top: -22px;
color: white;
font-weight: bold;
font-size: 20px;
border-radius: 0 3px 3px 0;
background-color: rgba(0, 0, 0, 0.5);
      }
.prev {
left: 0;
border-radius: 3px 0 0 3px;
      }
.next {
right: 0;
border-radius: 3px 0 0 3px;
      }
</style>
```

```
</head>
<body>
  <h2>Slideshow with Navigation Buttons</h2>

  <div class="slideshow-container">
    <img class="mySlides" src="image1.jpg">
    <img class="mySlides" src="image2.jpg">
    <img class="mySlides" src="image3.jpg">
    <img class="mySlides" src="image4.jpg">
    <img class="mySlides" src="image5.jpg">

    <a class="prev" onclick="plusSlides(-1)">&#10094; Previous</a>
    <a class="next" onclick="plusSlides(1)">Next &#10095;</a>
  </div>

  <script>
    var slideIndex = 0;
showSlides(slideIndex);

function plusSlides(n) {
showSlides(slideIndex += n);
    }

function showSlides(n) {
var slides = document.getElementsByClassName("mySlides");
var prevButton = document.querySelector(".prev");
var nextButton = document.querySelector(".next");

if (n >= slides.length - 1) {
          slideIndex = slides.length - 1;
          nextButton.style.display = "none";
        } else {
          nextButton.style.display = "block";
        }

if (n <= 0) {
          slideIndex = 0;
          prevButton.style.display = "none";
        } else {
          prevButton.style.display = "block";
```

```
        }

    // Hide all slides
        for (var i = 0; i < slides.length; i++) {
            slides[i].style.display = "none";
        }

    // Display current slide
        slides[slideIndex].style.display = "block";
    }
</script>

</body>
</html>
```

10. **Create a basic page. Using the setTimeout() method, create an animation on the page. Allow the user to stop the animation by placing the cursor on any marble. Allow the user to restart the animation by removing the cursor from that marble.**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Marble Animation</title>
<style>
   body {
font-family: Arial, sans-serif;
text-align: center;
margin-top: 50px;
  }
.marble {
width: 50px;
height: 50px;
background-color: #3498db;
border-radius: 50%;
position: absolute;
transition: transform 0.3s ease-in-out;
```

```
    }
</style>
</head>
<body>
    <div id="container">
        <div class="marble" id="marble1"></div>
        <div class="marble" id="marble2"></div>
        <div class="marble" id="marble3"></div>
    </div>

    <script>
        // Animation function
        function animateMarbles() {
 const container = document.getElementById('container');
 const marbles = container.getElementsByClassName('marble');

 let angle = 0;
 const radius = 100;
 const speed = 0.05;

 // Calculate marble positions in a circle
            Array.from(marbles).forEach((marble, index) => {
 const x = radius * Math.cos(angle) + container.offsetWidth / 2 -
marble.offsetWidth / 2;
 const y = radius * Math.sin(angle) + container.offsetHeight / 2 -
marble.offsetHeight / 2;
                marble.style.transform = `translate(${x}px, ${y}px)`;
                angle += 2 * Math.PI / marbles.length;
            });

 // Update angle for next frame
            angle += speed;

 // Continue animation
            setTimeout(animateMarbles, 10);
        }

 // Start animation
        animateMarbles();
```

```
// Pause animation on mouseover
      const marbles = document.getElementsByClassName('marble');
Array.from(marbles).forEach(marble => {
        marble.addEventListener('mouseover', () => {
clearTimeout(animationTimer);
        });
        marble.addEventListener('mouseout', () => {
animateMarbles();
        });
      });
</script>
</body>
</html>
```

11. **Create a basic page in html. Then, display the following items in the page using only one or more document.write() statements: 1. Information about the web browser that the user is viewing this page with. 2. The height and width of the user's monitor, i.e. the resolution 3. The date that the page was created or last modified.**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Display Browser and System Information</title>
</head>
<body>
<script>
   // Function to get browser information
   function getBrowserInfo() {
const userAgent = navigator.userAgent;
const browserInfo = `Browser: ${userAgent}`;
return browserInfo;
   }

// Function to get screen resolution
   function getScreenResolution() {
const screenWidth = window.screen.width;
const screenHeight = window.screen.height;
```

```
 const resolutionInfo = `Screen Resolution: ${screenWidth}px x
${screenHeight}px`;
 return resolutionInfo;
   }

 // Function to get page creation or modification date
   function getPageModifiedDate() {
 const lastModified = document.lastModified;
 const dateInfo = `Page Last Modified: ${lastModified}`;
 return dateInfo;
   }

 // Output information using document.write()
   document.write(`<p>${getBrowserInfo()}</p>`);
 document.write(`<p>${getScreenResolution()}</p>`);
 document.write(`<p>${getPageModifiedDate()}</p>`);
</script>
</body>
</html>
```

12. **Create a basic page in html that displays 2 images. When the user places a cursor over any image, replace the image with a different image. When the user removes the cursor from the image and return it to its original state**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Image Swap on Hover</title>
<style>
   /* Optional CSS for image sizing */
   .image-container {
 display: flex;
 justify-content: center;
 align-items: center;
 margin: 20px;
   }
```

```
 img {
 max-width: 300px;
 max-height: 200px;
 transition: transform 0.3s ease-in-out;
   }
 img:hover {
 transform: scale(1.1); /* Optional: Enlarge image on hover */
   }
</style>
</head>
<body>

<div class="image-container">
   <!-- First Image -->
   <img src="original_image1.jpg" onmouseover="changeImage(this,
'new_image1.jpg')" onmouseout="restoreImage(this)">
</div>

<div class="image-container">
   <!-- Second Image -->
   <img src="original_image2.jpg" onmouseover="changeImage(this,
'new_image2.jpg')" onmouseout="restoreImage(this)">
</div>

<script>
   // Function to change image on mouseover
   function changeImage(element, newSrc) {
      element.src = newSrc;
   }

 // Function to restore original image on mouseout
   function restoreImage(element) {
 if (element.dataset.originalSrc) {
         element.src = element.dataset.originalSrc;
 delete element.dataset.originalSrc;
      }
   }
</script>

</body>
```

```
</html>
```

14. Create a basic page in html that displays text and an image. Track how many times a user has visited your page by storing this information in a cookie. Display this information to the user, e.g. You have visited x number of times! The current visit should be included in this number. Display the date and time of the user's last visit, e.g. "You lasted visted on..." If this is the user's first visit, display e.g. "You have never visited before" instead.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Visit Counter and Last Visit Time</title>
<style>
   body {
 font-family: Arial, sans-serif;
 text-align: center;
 margin-top: 50px;
   }
 img {
 max-width: 100%;
 height: auto;
 margin-bottom: 20px;
   }
</style>
</head>
<body>

<div>
   <h1>Welcome to My Website!</h1>
   <p id="visit-info">Loading visit information...</p>
   <img src="placeholder_image.jpg" alt="Placeholder Image">
</div>

<script>
   // Function to set a cookie with name, value, and expiration date
   function setCookie(name, value, days) {
```

```
var expires = "";
if (days) {
var date = new Date();
        date.setTime(date.getTime() + (days * 24 * 60 * 60 * 1000));
        expires = "; expires=" + date.toUTCString();
    }
document.cookie = name + "=" + value + expires + "; path=/";
  }

// Function to get cookie value by name
  function getCookie(name) {
var nameEQ = name + "=";
var cookies = document.cookie.split(';');
for (var i = 0; i < cookies.length; i++) {
var cookie = cookies[i];
while (cookie.charAt(0) === ' ') {
            cookie = cookie.substring(1, cookie.length);
        }
if (cookie.indexOf(nameEQ) === 0) {
return cookie.substring(nameEQ.length, cookie.length);
        }
    }
return null;
  }

// Function to display visit information
  function displayVisitInfo() {
var visitCount = getCookie('visitCount');
var lastVisit = getCookie('lastVisit');

if (visitCount) {
        visitCount = parseInt(visitCount) + 1;
    } else {
        visitCount = 1;
    }

var now = new Date();
var formattedLastVisit = lastVisit ? new Date(lastVisit).toLocaleString() :
"never";
```

```
setCookie('visitCount', visitCount, 365); // Set or update visit count cookie
    setCookie('lastVisit', now, 365); // Set current visit as last visit

    var visitInfoText = `You have visited ${visitCount} time(s)!<br>`;
    visitInfoText += `You last visited on: ${formattedLastVisit}`;
document.getElementById('visit-info').innerHTML = visitInfoText;
  }
```

15. Create a "Mad-libs" game using JavaScript.
Create a blank page.
Using a prompt box, prompt the user to supply his or her name. Then, using 5 additional prompt boxes, prompt the user to supply 5 words. Save each word in a separate variable. Then, using document.write() statements, use the information stored in variables to display a "Mad-libs" type of story, i.e. create a few paragraphs of information in story format. Also, prompt the user for a color (i.e. one of the 16 named colors or a hex value). Store this color in a variable. In the "mad libs" story, highlight, using the color supplied by the user, each of the words that you previously collected. Do this by surrounding the words with
tags and using an inline style.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>Mad Libs Game</title>
<style>
    body {
        font-family: Arial, sans-serif;
        text-align: center;
        margin-top: 50px;
    }
    .highlight {
```

```
        color: red; /* Default color if user doesn't input
a valid color */
    }
</style>
</head>
<body>

<script>
    // Prompt the user for their name
    var userName = prompt("Please enter your name:");
    if (!userName) {
        userName = "Guest"; // Default name if user cancels
or doesn't input anything
    }

    // Prompt the user for 5 words
    var word1 = prompt("Enter a noun (singular):");
    var word2 = prompt("Enter an adjective:");
    var word3 = prompt("Enter another noun (plural):");
    var word4 = prompt("Enter a verb (present tense):");
    var word5 = prompt("Enter a place:");

    // Prompt the user for a color
    var userColor = prompt("Choose a color (e.g., red,
#00ff00):");
    var highlightColor = validateColor(userColor);

    // Function to validate color input
    function validateColor(color) {
        // Check if color is a valid named color or hex
value
        var validColors = ["red", "blue", "green",
"yellow", "purple", "orange", "black", "white", "#ff0000",
"#00ff00", "#0000ff", "#ffff00", "#ff00ff", "#ffa500",
"#000000", "#ffffff"];
```

```
        if (validColors.includes(color.toLowerCase())) {
            return color; // Return the validated color
        } else {
            return "red"; // Default color if input is
invalid
        }
    }

    // Output the mad-libs story with highlighted words
    document.write(`<h2>Welcome, ${userName}!</h2>`);
    document.write(`<p>Once upon a time, in a
${highlightWord(word2)} ${highlightWord(word3)} far, far
away, there lived a ${highlightWord(word1)} who loved to
${highlightWord(word4)} in ${highlightWord(word5)}.</p>`);

    // Function to highlight words using the chosen color
    function highlightWord(word) {
        return `<span class="highlight" style="color:
${highlightColor};">${word}</span>`;
    }
</script>

</body>
</html>
```