83. Random Color Generator

To generate a random color in JavaScript, you can create a function that generates random values for the red, green, and blue components of the color.

```
function generateRandomColor() {
    // Generate random values for red, green, and blue components
    const red = Math.floor(Math.random() * 256);
    const green = Math.floor(Math.random() * 256);
    const blue = Math.floor(Math.random() * 256);

    // Create the RGB color string
    const color = `rgb(${red}, ${green}, ${blue})`;

    return color;
}

// Example usage:
const randomColor = generateRandomColor();

console.log("Random Color:", randomColor);
```

84. Check if a String is Empty

To check if a string is empty in JavaScript, you can use the `length` property of the string. If the length is zero, it means the string is empty.

```
function isEmptyString(str) {
    return str.length === 0;
}

// Example usage:
const emptyString = "";
const nonEmptyString = "Hello, world!";

console.log("Is emptyString empty?", isEmptyString(emptyString)); // Outputs:
true
console.log("Is nonEmptyString empty?", isEmptyString(nonEmptyString)); //
Outputs: false
```

85. Capitalize the First Letter of a String

To capitalize the first letter of a string in JavaScript, you can use a combination of the `charAt()`, `toUpperCase()`, and `slice()` methods.

```javascript
function capitalizeFirstLetter(str) {
    // Check if the string is not empty
    if (str.length === 0) {
        return "Empty string";
    }

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

// Example usage:
const originalString = "hello, world!";
const capitalizedString = capitalizeFirstLetter(originalString);

console.log("Original String:", originalString);
console.log("Capitalized String:", capitalizedString);
```

86. Find the Maximum Element in an Array

To find the maximum element in an array in JavaScript, you can use the `Math.max()` function along with the spread operator (`...`) to pass the array elements as individual arguments.

```javascript
function findMaxElement(arr) {
    // Check if the array is not empty
    if (arr.length === 0) {
        return "Empty array";
    }

    // Use Math.max() with the spread operator to find the maximum element
    const maxElement = Math.max(...arr);

    return maxElement;
}

// Example usage:
const numbers = [5, 2, 9, 1, 7];
const maxNumber = findMaxElement(numbers);

console.log("Array:", numbers);
console.log("Maximum Element:", maxNumber);
```

87. Reverse an Array

To reverse an array in JavaScript, you can use the `reverse()` method, which is available for arrays. This method reverses the elements of an array in place.

```javascript
function reverseArray(arr) {
    // Use the reverse() method to reverse the array in place
    return arr.reverse();
}

// Example usage:
const originalArray = [1, 2, 3, 4, 5];
const reversedArray = reverseArray(originalArray);

console.log("Original Array:", originalArray);
console.log("Reversed Array:", reversedArray);
```

88. Calculate the Power of a Number

To calculate the power of a number in JavaScript, you can use the `Math.pow()` method or the exponentiation operator (`**`).

```
// Using Math.pow()
function calculatePowerWithMathPow(base, exponent) {
    return Math.pow(base, exponent);
}

// Using the exponentiation operator (**)
function calculatePowerWithExponentiationOperator(base, exponent) {
    return base ** exponent;
}

// Example usage:
const baseNumber = 2;
const exponentNumber = 3;

const resultWithMathPow = calculatePowerWithMathPow(baseNumber,
exponentNumber);
const resultWithExponentiationOperator =
calculatePowerWithExponentiationOperator(baseNumber, exponentNumber);

console.log(`${baseNumber} to the power of ${exponentNumber} using Math.pow():
${resultWithMathPow}`);
console.log(`${baseNumber} to the power of ${exponentNumber} using the
exponentiation operator (**): ${resultWithExponentiationOperator}`);
```

89. Find the Minimum Element in an Array

To find the minimum element in an array in JavaScript, you can use the `Math.min()` function along with the spread operator (`...`) to pass the array elements as individual arguments.

```
function findMinElement(arr) {
    // Check if the array is not empty
    if (arr.length === 0) {
        return "Empty array";
    }

    // Use Math.min() with the spread operator to find the minimum element
    const minElement = Math.min(...arr);

    return minElement;
}

// Example usage:
const numbers = [5, 2, 9, 1, 7];
const minNumber = findMinElement(numbers);

console.log("Array:", numbers);
console.log("Minimum Element:", minNumber);
```

90. Convert Minutes to Hours and Minutes

To convert a total number of minutes to hours and remaining minutes in JavaScript, you can use simple mathematical operations.

```javascript
function convertMinutesToHoursAndMinutes(totalMinutes) {
    // Check if the input is a valid positive number
    if (isNaN(totalMinutes) || totalMinutes < 0) {
        return "Invalid input. Please provide a non-negative number of
minutes.";
    }

    // Calculate hours and remaining minutes
    const hours = Math.floor(totalMinutes / 60);
    const minutes = totalMinutes % 60;

    // Construct the result string
    const result = `${hours} hours and ${minutes} minutes`;

    return result;
}

// Example usage:
const totalMinutes = 135;
const convertedTime = convertMinutesToHoursAndMinutes(totalMinutes);

console.log(`${totalMinutes} minutes is equivalent to: ${convertedTime}`);
```

91. Find the Sum of Digits in a Number

To find the sum of digits in a number using JavaScript, you can use a loop to iterate through each digit and add them together.

```
function sumOfDigits(number) {
    // Check if the input is a valid number
    if (isNaN(number) || !Number.isInteger(number) || number < 0) {
        return "Invalid input. Please provide a non-negative integer.";
    }

    // Convert the number to a string to iterate through its digits
    const digitsArray = String(number).split('').map(Number);

    // Calculate the sum of digits
    const sum = digitsArray.reduce((acc, digit) => acc + digit, 0);

    return sum;
}

// Example usage:
const inputNumber = 12345;
const result = sumOfDigits(inputNumber);

console.log(`The sum of digits in ${inputNumber} is: ${result}`);
```

92. Check if a String is a Palindromic Phrase

To check if a string is a palindromic phrase in JavaScript, you can create a function that removes non-alphanumeric characters and compares the string with its reversed version.

```javascript
function isPalindromicPhrase(str) {
    // Check if the input is a valid string
    if (typeof str !== 'string') {
        return "Invalid input. Please provide a string.";
    }

    // Remove non-alphanumeric characters and convert to lowercase
    const cleanedStr = str.replace(/[^a-zA-Z0-9]/g, '').toLowerCase();

    // Compare the cleaned string with its reversed version
    return cleanedStr === cleanedStr.split('').reverse().join('');
}

// Example usage:
const phrase1 = "A man, a plan, a canal, Panama!";
const phrase2 = "Hello, world!";

console.log(`Is "${phrase1}" a palindromic phrase?
${isPalindromicPhrase(phrase1)}`);
console.log(`Is "${phrase2}" a palindromic phrase?
${isPalindromicPhrase(phrase2)}`);
```

93. Generate a Random Password

To generate a random password in JavaScript, you can create a function that combines random characters from various character sets.

```
function generateRandomPassword(length) {
    // Define character sets
    const uppercaseChars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    const lowercaseChars = 'abcdefghijklmnopqrstuvwxyz';
    const numericChars = '0123456789';
    const specialChars = '!@#$%^&*()-_+=';

    // Combine character sets
    const allChars = uppercaseChars + lowercaseChars + numericChars +
specialChars;

    // Check if the input length is a valid positive number
    if (!Number.isInteger(length) || length <= 0) {
        return "Invalid input. Please provide a positive integer for the
password length.";
    }

    // Generate the random password
    let password = '';
    for (let i = 0; i < length; i++) {
        const randomIndex = Math.floor(Math.random() * allChars.length);
        password += allChars.charAt(randomIndex);
    }

    return password;
}

// Example usage:
const passwordLength = 12;
const randomPassword = generateRandomPassword(passwordLength);

console.log(`Generated Password: ${randomPassword}`);
```

94. Calculate Simple Interest

Program To calculate simple interest in JavaScript:

```
function calculateSimpleInterest(principal, rate, time) {
    // Check if the inputs are valid positive numbers
    if (isNaN(principal) || isNaN(rate) || isNaN(time) || principal <= 0 ||
rate <= 0 || time <= 0) {
        return "Invalid inputs. Please provide valid positive numbers.";
    }

    // Calculate simple interest
    const simpleInterest = (principal * rate * time) / 100;

    return simpleInterest;
}

// Example usage:
const principalAmount = 1000;
const interestRate = 5; // 5%
const investmentTime = 2; // 2 years

const interestAmount = calculateSimpleInterest(principalAmount, interestRate,
investmentTime);

console.log(`Principal Amount: $${principalAmount}`);
console.log(`Interest Rate: ${interestRate}%`);
console.log(`Investment Time: ${investmentTime} years`);
console.log(`Simple Interest: $${interestAmount}`);
```

95. Implement a Basic Stopwatch

To implement a basic stopwatch in JavaScript, you can use the `Date` object to measure the elapsed time.

```
let startTime;
let stopwatchInterval;

function startStopwatch() {
  startTime = new Date().getTime();

  stopwatchInterval = setInterval(updateDisplay, 1000);
}

function stopStopwatch() {
  clearInterval(stopwatchInterval);
}

function resetStopwatch() {
  stopStopwatch();
  updateDisplay(0);
}

function updateDisplay() {
  const currentTime = new Date().getTime();
  const elapsedTime = Math.floor((currentTime - startTime) / 1000);

  const minutes = Math.floor(elapsedTime / 60);
  const seconds = elapsedTime % 60;

  const formattedTime = `${minutes}:${seconds < 10 ? "0" : ""}${seconds}`;

  document.getElementById("display").textContent = formattedTime;
}
```

96. Check if a Number is a Perfect Number

A perfect number is a positive integer that is equal to the sum of its proper divisors (excluding itself).

```javascript
function isPerfectNumber(number) {
    // Check if the input is a positive integer
    if (!Number.isInteger(number) || number <= 0) {
        return "Invalid input. Please provide a positive integer.";
    }

    // Find divisors and calculate sum
    let sum = 0;
    for (let i = 1; i <= Math.floor(number / 2); i++) {
        if (number % i === 0) {
            sum += i;
        }
    }

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

// Example usage:
const testNumber = 28;
const result = isPerfectNumber(testNumber);

console.log(`Is ${testNumber} a perfect number? ${result}`);
```

97. This program calculates the Volume of a Cylinder

This program calculates the Volume of a Cylinder:

```
function calculateCylinderVolume(radius, height) {
    // Check if the inputs are valid positive numbers
    if (isNaN(radius) || isNaN(height) || radius <= 0 || height <= 0) {
        return "Invalid inputs. Please provide valid positive numbers.";
    }

    // Calculate the volume of the cylinder
    const volume = Math.PI * Math.pow(radius, 2) * height;

    return volume;
}

// Example usage:
const cylinderRadius = 5;
const cylinderHeight = 10;

const cylinderVolume = calculateCylinderVolume(cylinderRadius, cylinderHeight);

console.log(`Cylinder Volume: ${cylinderVolume.toFixed(2)} cubic units`);
```

98. Generate a Random Quote

To generate a random quote in JavaScript, you can create an array of quotes and use a function to pick a random quote from that array.

```
function generateRandomQuote() {
    const quotes = [
        "The only way to do great work is to love what you do. - Steve Jobs",
        "In three words I can sum up everything I've learned about life: it
goes on. - Robert Frost",
        "The greatest glory in living lies not in never falling, but in rising
every time we fall. - Nelson Mandela",
        "Life is what happens when you're busy making other plans. - John
Lennon",
        "Get busy living or get busy dying. - Stephen King"
        // Add more quotes as needed
    ];

    // Generate a random index to pick a quote from the array
    const randomIndex = Math.floor(Math.random() * quotes.length);

    // Return the randomly selected quote
    return quotes[randomIndex];
}

// Example usage:
const randomQuote = generateRandomQuote();
console.log("Random Quote:", randomQuote);
```

99. Find the Intersection of Two Arrays

To find the intersection of two arrays in JavaScript, you can create a function that iterates through both arrays and identifies the common elements.

```
function findIntersection(arr1, arr2) {
    // Check if the inputs are valid arrays
    if (!Array.isArray(arr1) || !Array.isArray(arr2)) {
        return "Invalid inputs. Please provide valid arrays.";
    }

    // Use a Set to store unique elements of the first array
    const set = new Set(arr1);

    // Filter the second array to include only elements present in the set
    const intersection = arr2.filter(element => set.has(element));

    return intersection;
}

// Example usage:
const array1 = [1, 2, 3, 4, 5];
const array2 = [3, 4, 5, 6, 7];

const result = findIntersection(array1, array2);

console.log("Intersection of Arrays:", result);
```

100. Convert Feet to Meters

To convert feet to meters in JavaScript, you can use the following conversion formula:

$\text{Meters} = \text{Feet} \times 0.3048$$\text{Meters} = \text{Feet} \times 0.3048$

Here's a simple function that performs the conversion:

```javascript
function feetToMeters(feet) {
    // Check if the input is a valid number
    if (isNaN(feet)) {
        return "Invalid input. Please provide a valid number of feet.";
    }

    // Perform the conversion
    const meters = feet * 0.3048;

    return meters;
}

// Example usage:
const feetValue = 10;
const metersValue = feetToMeters(feetValue);

console.log(`${feetValue} feet is equal to ${metersValue.toFixed(2)} meters`);
```

101. Convert Days to Years, Months, and Days

To convert a given number of days into years, months, and remaining days in JavaScript, you can use the following function:

```
function convertDaysToYearsMonthsDays(days) {
    // Check if the input is a valid positive number
    if (isNaN(days) || days <= 0) {
        return "Invalid input. Please provide a valid positive number of
days.";
    }

    // Calculate years
    const years = Math.floor(days / 365);

    // Calculate remaining days after removing years
    const remainingDaysAfterYears = days % 365;

    // Calculate months
    const months = Math.floor(remainingDaysAfterYears / 30);

    // Calculate remaining days after removing months
    const remainingDaysAfterMonths = remainingDaysAfterYears % 30;

    return {
        years,
        months,
        days: remainingDaysAfterMonths
    };
}

// Example usage:
const totalDays = 1000;
const result = convertDaysToYearsMonthsDays(totalDays);

console.log(`${totalDays} days is approximately ${result.years} years,
${result.months} months, and ${result.days} days.`);
```

102. Find the Median of an Array

To find the median of an array in JavaScript, you can create a function that sorts the array and then determines the median based on its length.

```
function findMedian(arr) {
    // Check if the input is a valid array
    if (!Array.isArray(arr) || arr.length === 0) {
        return "Invalid input. Please provide a non-empty array.";
    }

    // Sort the array
    const sortedArray = arr.slice().sort((a, b) => a - b);

    // Calculate the median
    const middleIndex = Math.floor(sortedArray.length / 2);

    if (sortedArray.length % 2 === 0) {
        // If the array has an even number of elements, return the average of
the middle two
        const median = (sortedArray[middleIndex - 1] +
sortedArray[middleIndex]) / 2;
        return median;
    } else {
        // If the array has an odd number of elements, return the middle
element
        return sortedArray[middleIndex];
    }
}

// Example usage:
const numbers = [5, 2, 8, 1, 7, 3];
const result = findMedian(numbers);

console.log("Median:", result);
```

103. Calculate the Distance Between Two Points

To calculate the distance between two points (x1, y1) and (x2, y2) in a two-dimensional plane:

```
function calculateDistance(x1, y1, x2, y2) {
    // Check if the inputs are valid numbers
    if (isNaN(x1) || isNaN(y1) || isNaN(x2) || isNaN(y2)) {
        return "Invalid inputs. Please provide valid numerical coordinates.";
    }

    // Calculate the distance using the distance formula
    const distance = Math.sqrt(Math.pow((x2 - x1), 2) + Math.pow((y2 - y1),
2));

    return distance;
}

// Example usage:
const x1 = 1;
const y1 = 2;
const x2 = 4;
const y2 = 6;

const result = calculateDistance(x1, y1, x2, y2);

console.log(`The distance between (${x1}, ${y1}) and (${x2}, ${y2}) is
${result.toFixed(2)}`);
```