86. Find the Difference between Two Arrays

Find the elements that are present in the first array but not in the second array.

```
const difference = (arr1, arr2) => arr1.filter(val
=> !arr2.includes(val));

console.log(difference([1, 2, 3], [2, 3, 4]));
// Output: [1]
```

87. Check if a Number is a Fibonacci Number

Check if a given number is a Fibonacci number.

```
const isFibonacci = (num) => isPerfectSquare(5 * num
* num + 4) || isPerfectSquare(5 * num * num - 4);

console.log(isFibonacci(5));
// Output: true

console.log(isFibonacci(6));
// Output: false
```

88. Convert Hours to Minutes

Convert a given number of hours to minutes.

```
const hoursToMinutes = (hours) => hours * 60;

console.log(hoursToMinutes(2));
// Output: 120
```

89. Get the First N Elements of an Array

Get the first N elements from the beginning of an array.

```
const firstNElements = (arr, n) => arr.slice(0, n);

console.log(firstNElements([1, 2, 3, 4, 5], 3));
// Output: [1, 2, 3]
```

90. Check if a Number is Odd

Get the first N elements from the beginning of an array.

```
const isOdd = (num) => num % 2 !== 0;

console.log(isOdd(5));
// Output: true

console.log(isOdd(4));
// Output: false
```

91. Calculate the Standard Deviation of an Array of Numbers

Calculate the standard deviation of an array of numbers.

```javascript
const standardDeviation = (arr) => {
  const avg = mean(arr);
  const squaredDiffs = arr.map(num => Math.pow(num - 
avg, 2));
  const variance = mean(squaredDiffs);
  return Math.sqrt(variance);
};

console.log(standardDeviation([1, 2, 3, 4, 5]));
// Output: 1.4142135623730951
```

92. Check if a String ends with a specific Substring

Check if a string ends with a specific substring.

```
const endsWithSubstring = (str, subStr) =>
str.endsWith(subStr);

console.log(endsWithSubstring("Hello, world!",
"world!"));
// Output: true

console.log(endsWithSubstring("Hello, world!",
"Hello"));
// Output: false
```

93. Calculate the Sum of Squares of an Array

Calculate the sum of squares of an array of numbers.

```
const sumOfSquares = (arr) => arr.reduce((acc, val)
=> acc + val ** 2, 0);

console.log(sumOfSquares([1, 2, 3, 4, 5]));
// Output: 55
```

94. Check if a String is a Palindrome (case-sensitive)

Check if a string is a palindrome, considering case sensitivity.

```
const isPalindromeCaseSensitive = (str) => str ===
str.split('').reverse().join('');

console.log(isPalindromeCaseSensitive("level"));
// Output: true

console.log(isPalindromeCaseSensitive("Hello"));
// Output: false
```

95. Generate an Array of Random Numbers

Generate an array of random numbers.

```
const randomArray = (length) => Array.from({ length
}, () => Math.floor(Math.random() * 100));

console.log(randomArray(5));
// Output: Array with 5 random numbers, e.g., [23,
45, 67, 11, 88]
```

96. Calculate the Greatest Common Divisor (GCD) of Two Numbers

Calculate the Greatest Common Divisor (GCD) of two numbers.

```
const gcd = (num1, num2) => {
  while (num2 !== 0) {
    let temp = num2;
    num2 = num1 % num2;
    num1 = temp;
  }
  return num1;
};

console.log(gcd(48, 18)); // Output: 6
```

97. Convert Seconds to Hours, Minutes, and Seconds

Convert seconds to hours, minutes, and seconds.

```
const secsToHoursMinsSecs = (seconds) => {
  const hours = Math.floor(seconds / 3600);
  const remainingSeconds = seconds % 3600;
  const minutes = Math.floor(remainingSeconds / 60);
  const remainingSecs = remainingSeconds % 60;
  return `${hours} hours, ${minutes} minutes, and
${remainingSecs} seconds`;
};

console.log(secsToHoursMinsSecs(7320));
// Output: "2 hours, 2 minutes, and 0 seconds"
```

98. Calculate the LCM of Two Numbers

Calculate the Least Common Multiple (LCM) of two numbers.

```
const lcm = (num1, num2) => (num1 * num2) /
gcd(num1, num2);

console.log(lcm(6, 8)); // Output: 24
```

99. Find the Longest Word in a String

Find the longest word in a string.

```
const findLongestWord = (str) => str.split('
').reduce((longest, word) => word.length >
longest.length ? word : longest, '');

console.log(findLongestWord("Hello, how are you
doing?"));
// Output: "doing?"
```

100. Count the Occurrences of a Character in a String

Count the occurrences of a character in a string.

```
const countOccurrences = (str, char) =>
str.split(char).length - 1;

console.log(countOccurrences("hello world", "l"));
// Output: 3
```

101. Find the Median of an Array of Numbers

Find the median of an array of numbers.

```
const median = (arr) => {
  const sorted = arr.sort((a, b) => a - b);
  const mid = Math.floor(sorted.length / 2);
  return sorted.length % 2 === 0 ? (sorted[mid - 1]
+ sorted[mid]) / 2 : sorted[mid];
};

console.log(median([1, 3, 5, 7, 9]));
// Output: 5
```

102. Remove Duplicates from a String

Remove duplicate characters from a string.

```
const removeDuplicatesFromString = (str) => [...new
Set(str.split(''))].join('');

console.log(removeDuplicatesFromString("hello"));
// Output: "helo"
```

103. Find the Mode of an Array of Numbers

Calculate the mode, the most frequently occurring number(s), from an array of numbers. It identifies the number(s) with the highest frequency and returns them in an array.

```
const mode = (arr) => {
  const frequency = {};
  arr.forEach(num => frequency[num] =
(frequency[num] || 0) + 1);
  const maxFrequency =
Math.max(...Object.values(frequency));
  return Object.keys(frequency).filter(num =>
frequency[num] === maxFrequency).map(Number);
};

console.log(mode([1, 2, 2, 3, 3, 3, 4, 4, 4, 4]));
// Output: [4]
```

104. Check if a Number is a Harshad Number (Niven Number)

A Harshad number, also known as a Niven number, is an integer divisible by the sum of its digits. The isHarshadNumber function determines whether a given number meets this criterion. It calculates the sum of the digits of the number, and then checks if the number itself is divisible by this sum.

```
const isHarshadNumber = (num) => num %
[...String(num)].reduce((sum, digit) => sum +
Number(digit), 0) === 0;

console.log(isHarshadNumber(18));
// Output: true

console.log(isHarshadNumber(21));
// Output: false
```

105. Convert Binary Number to Decimal (without parseInt)

This function performs the conversion of a binary number to its equivalent decimal representation, all without utilizing the parseInt function. The process involves splitting the binary number's digits, reversing them, and using a reduce operation to calculate the decimal value by considering each digit's position and value.

```
const binaryToDecimalWithoutParseInt = (binary) =>
binary.split('').reverse().reduce((dec, bit, index)
=> dec + bit * (2 ** index), 0);

console.log(binaryToDecimalWithoutParseInt("1101"));
// Output: 13
```

106. Check if an Array is Sorted in Descending Order

This function determines if an array is sorted in descending order. It iterates through the array and verifies that each element is either greater than or equal to the preceding element, ensuring a descending order.

```
const isSortedDescending = (arr) => arr.every((el,
i) => i === 0 || el <= arr[i - 1]);

console.log(isSortedDescending([5, 4, 3, 2, 1]));
// Output: true

console.log(isSortedDescending([1, 5, 3, 8, 2]));
// Output: false
```