250+ Killer JavaScript One-Liners

Transform your code into powerful solutions.

By Hernando Abella

ALUNA PUBLISHING HOUSE

Thank you for trusting our Publishing House. If you have the opportunity to evaluate our work and give us a comment on Amazon, we will appreciate it very much!

Table of contents

Welcome to a coding adventure like no other! '250+ JavaScript Killer One-Liners' is your passport to mastering the intricacies of JavaScript. Inside, discover a collection of powerful, concise code snippets that will transform the way you write and think about JavaScript.

Whether you're a seasoned developer or just starting, this book is your gateway to unlocking the true potential of the language. Join us as we delve into the art of crafting code that not only solves problems but does so elegantly and efficiently. Get ready to elevate your coding game with every line you write!

1. Convert Celsius to Fahrenheit

celsiusToFahrenheit allows you to convert a temperature from Celsius to Fahrenheit. It takes a value in Celsius as input and uses the formula (Celsius * 9/5) + 32 to perform the conversion hexadecimal representation.

```
const celsiusToFahrenheit = (celsius) => (celsius * 9/5) +
32; celsiusToFahrenheit(25);
// Result: 77
```

2. Get Value of a brower Cookie

Retrieve the value of a cookie by accessing with **document.cookie**

```
const cookie = name => `; ${document.cookie}`.split(`;
${name}=`).pop().split(';').shift();

cookie('_ga');
// Result: "GA1.2.1929736587.1601974046"
```

3. Convert RGB to Hex

rgbToHex enables you to convert RGB (Red, Green, Blue) values to their corresponding hexadecimal representation.

```
const rgbToHex = (r, g, b) =>
 "#" + ((1 << 24) + (r << 16) + (g << 8) +
b).toString(16).slice(1);

rgbToHex(0, 51, 255);
// Result: #0033ff
```

4. Copy to Clipboard

Easily copy any text to clipboard using **navigator.clipboard.writeText**.

```
const copyToClipboard = (text) =>
navigator.clipboard.writeText(text);
copyToClipboard("Hello World");
```

5. Check if Date is Valid

Use the following snippet to check if a given date is valid or not.

```
const isDateValid = (...val) => !Number.isNaN(new
Date(...val).valueOf());

isDateValid("December 17, 1995 03:24:00");
// Result: true
```

6. Find the day of year

Find which is the day by a given date.

```
const dayOfYear = (date) =>
  Math.floor((date - new Date(date.getFullYear(), 0, 0)) /
1000 / 60 / 60 / 24);

dayOfYear(new Date());
// Result: 272
```

7. Capitalise a String

Javascript doesn't have an inbuilt capitalise function, so we can use the following code for the purpose.

```
const capitalize = str => str.charAt(0).toUpperCase() +
str.slice(1)

capitalize("follow for more")
// Result: Follow for more
```

8. Find the number of days between two days

Find the days between 2 given days using the following snippet.

```
const dayDif = (date1, date2) =>
Math.ceil(Math.abs(date1.getTime() - date2.getTime()) /
86400000)

dayDif(new Date("2020-10-21"), new Date("2021-10-22"))
// Result: 366
```

9. Clear All Cookies

You can easily clear all cookies stored in a web page by accessing the cookie using **document.cookie** and clearing it.

```
const clearCookies =
document.cookie.split(';').forEach(cookie =>
document.cookie = cookie.replace(/^ +/,
'').replace(/=.*/, `=;expires=${new
Date(0).toUTCString()};path=/`));
```

10. Generate Random Hex

You can generate random hex colors with **Math.random** and padEnd properties.

```
const randomHex = () => `#${Math.floor(Math.random() *
0xffffff).toString(16).padEnd(6, "0")}`;

console.log(randomHex());
// Result: #92b008
```

11. Get Query Params from URL

You can easily retrieve query params from a url either by passing window.location or the raw URL goole.com?search=easy&page=3

```
const getParameters = (URL) => {
  URL = JSON.parse('{"' +
decodeURI(URL.split("?")[1]).replace(/"/g,
'\\"').replace(/&/g, '","').replace(/=/g, '":"') +'"}');
  return JSON.stringify(URL);
};

getParameters(window.location)
// Result: { search : "easy", page : 3 }
```

12. Log Time from Date

We can log time, in the format hour::minutes::seconds from a given date.

```
const timeFromDate = date => date.toTimeString().slice(0,
8);

console.log(timeFromDate(new Date(2021, 0, 10, 17, 30,
0)));
// Result: "17:30:00"
```

13. Check if a number is even or odd

A simple JavaScript function named "isEven" determines if a number is even or odd. It takes a "num" as input and returns true if even, false if odd.

```
const isEven = num => num % 2 === 0;

console.log(isEven(2));
// Result: True
```

14. Find Average of Numbers

Find the average between multiple numbers using reduce method.

```
const average = (...args) => args.reduce((a, b) => a + b) /
args.length;

average(1, 2, 3, 4);
// Result: 2.5
```

15. Scroll to Top

You can use window.scrollTo(0, 0) method to automatic scroll to top. Set both x and y as 0.

```
const goToTop = () => window.scrollTo(0, 0);
goToTop();
```

16. Reverse a string

You can easily reverse a string using split, reverse and join methods.

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

reverse('hello world');
// Result: 'dlrow olleh'
```

17. Check if array is empty

You can check if an array if empty with this snippet.

```
const isNotEmpty = arr => Array.isArray(arr) && arr.length
> 0;

isNotEmpty([1, 2, 3]);
// Result: true
```

18. Get Selected Text

Get the text the user has select using inbuilt getSelection property.

```
const getSelectedText = () =>
window.getSelection().toString();

getSelectedText();
```

19. Shuffle an Array

Shuffling an array is super easy with sort and random methods.

```
const shuffleArray = (arr) => arr.sort(() => 0.5 -
Math.random());

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

20. Detect Dark Mode

Check if a user's device is in dark mode with the following code.

```
const isDarkMode = window.matchMedia &&
window.matchMedia('(prefers-color-scheme: dark)').matches

console.log(isDarkMode)
// Result: True or False
```

21. Remove Duplicated from Array

You can easily remove duplicates with Set in JavaScript. Its a life saver.

```
const removeDuplicates = (arr) => [...new Set(arr)];

console.log(removeDuplicates([1, 2, 3, 3, 4, 4, 5, 5, 6]));
// Result: [ 1, 2, 3, 4, 5, 6 ]
```

22. Get the Length of a String

getLength efficiently calculates the length of a given string using .length property.

```
const getLength = (str) => str.length;

getLength("Hello, world!");
// Result: 13
```

23. Calculate the Area of a Circle

Calculate the area of a circle given its radius.

```
const calculateCircleArea = (radius) => Math.PI *
Math.pow(radius, 2);

calculateCircleArea(5);
// Result: 78.53981633974483
```

24. Check if a Number is Prime

Determine if a given number is a prime number.

```
const isPrime = (num) => {
  if (num <= 1) return false;
  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) return false;
  }
  return true;
};

isPrime(13);
// Result: true
```

25. Count Occurrences of a Character in a String

Count the occurrences of a specific character in a given string.

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

countOccurrences("banana", "a");
// Result: 3
```

26. Remove Leading and Trailing Whitespaces

Remove leading and trailing whitespaces from a given string.

```
const removeWhitespaces = (str) => str.trim();

removeWhitespaces("  Hello, world!  ");
// Result: "Hello, world!"
```

27. Generate a Random Number within a Range

Generate a random integer within a specified range.

```
const randomInRange = (min, max) =>
Math.floor(Math.random() * (max - min + 1)) + min;

randomInRange(1, 10);
// Result: Random number between 1 and 10 (inclusive)
```

28. Convert Seconds to HH:MM:SS Format

Convert a given number of seconds into the "hours:minutes:seconds" format.

```
const secondsToHHMMSS = (seconds) => {
  const pad = (num) => String(num).padStart(2, '0');
  const hours = Math.floor(seconds / 3600);
  const minutes = Math.floor((seconds % 3600) / 60);
  const secs = seconds % 60;
  return `${pad(hours)}:${pad(minutes)}:${pad(secs)}`;
};

secondsToHHMMSS(3660); // Result: "01:01:00"
```

29. Get the Last Element of an Array

Retrieve the last element of a given array.

```
const getLastElement = (arr) => arr[arr.length - 1];

getLastElement([1, 2, 3, 4]);
// Result: 4
```

30. Sort an Array of Numbers in Ascending Order

Sort a given array of numbers in ascending order.

```
const sortAscending = (arr) => arr.slice().sort((a, b) => a
- b);

sortAscending([3, 1, 4, 1, 5, 9, 2, 6, 5, 3]);
// Result: [1, 1, 2, 3, 3, 4, 5, 5, 6, 9]
```

31. Check if a String is Palindrome

Determine if a given string is a palindrome.

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

isPalindrome("level");
// Result: true
```

32. Calculate Factorial of a Number

Calculate the factorial of a given number.

```
const factorial = (num) => {
  if (num === 0 || num === 1) return 1;
  return num * factorial(num - 1);
};

factorial(5);
// Result: 120
```

33. Sum all Numbers in an Array

Calculate the sum of all numbers in a given array.

```
const sumArray = (arr) => arr.reduce((acc, val) => acc + val, 0);

sumArray([1, 2, 3, 4, 5]);
// Result: 15
```

34. Find the Maximum Value in an Array

Find the maximum value in a given array of numbers.

```
const findMax = (arr) => Math.max(...arr);

findMax([10, 5, 8, 20, 3]);
// Result: 20
```

35. Get the Current Date in DD/MM/YYYY Format

Get the current date in the "DD/MM/YYYY" format.

```
const getCurrentDate = () => {
  const date = new Date();
  const day = String(date.getDate()).padStart(2,
'0');
  const month = String(date.getMonth() +
1).padStart(2, '0');
  const year = date.getFullYear();
  return `${day}/${month}/${year}`;
};

getCurrentDate(); // Result: "02/08/2023"
```

36. Calculate the Power of a Number

Calculate the result of raising a given base to a specified exponent.

```
const power = (base, exponent) => Math.pow(base, exponent);

power(2, 5); // Result: 32
```

37. Convert String to Number

Convert a given string to a numeric value (float or integer).

```
const stringToNumber = (str) => parseFloat(str);

stringToNumber("3.14");
// Result: 3.14
```

38. Find the First N Fibonacci Numbers

Generate the first N Fibonacci numbers.

```
const fibonacci = (n) => {
  const result = [0, 1];
  for (let i = 2; i < n; i++) {
    result.push(result[i - 1] + result[i - 2]);
  }
  return result;
};

fibonacci(8);
// Result: [0, 1, 1, 2, 3, 5, 8, 13]
```

39. Count the Number of Words in a String

Count the number of words in a given string.

```
const countWords = (str) => str.trim().split(/\s+/).length;

console.log(countWords("Hello world, how are you?"));
// Output: 5
```

40. Reverse an Array

Reverse the elements of a given array.

```
const reverseArray = (arr) => arr.slice().reverse();

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

41. Get the Current Year

Get the current year.

```
const currentYear = () => new Date().getFullYear();

console.log(currentYear());
// Output: 2023 (depending on the current year)
```

42. Generate a Random Number between 1 and 10

Generate a random integer between 1 and 10 (inclusive).

```
const random1To10 = () => Math.floor(Math.random() * 10) +
1;

console.log(random1To10());
// Output: Random number between 1 and 10 (inclusive)
```

43. Check if a String is Empty

Check if a given string is empty (contains no characters).

```
const isEmptyString = (str) => str.trim().length === 0;

console.log(isEmptyString(""));
// Output: true

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

44. Check if an Object has a Specific Property

Check if an object contains a specific property.

```
const hasProperty = (obj, prop) => prop in obj;

const person = { name: "John", age: 30 };
console.log(hasProperty(person, "name"));
// Output: true

console.log(hasProperty(person, "gender"));
// Output: false
```

45. Calculate the Average of Numbers in an Array

Calculate the average of numbers in a given array.

```
const average = (arr) => arr.reduce((acc, val) => acc + val, 0) / arr.length;

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

46. Check if a Number is a Multiple of 5

Check if a given number is a multiple of 5.

```
const isMultipleOf5 = (num) => num % 5 === 0;

console.log(isMultipleOf5(10));
// Output: true

console.log(isMultipleOf5(7));
// Output: false
```

47. Convert Minutes to Seconds

Convert minutes to seconds.

```
const minsToSecs = (mins) => mins * 60;

console.log(minsToSecs(5));
// Output: 300
```

48. Find the Maximum Value in an Array of Objects

Find the maximum value of a specific property in an array of objects.

```
const findMaxValue = (arr, key) => Math.max(...arr.map(item
=> item[key]));

const students = [
  { name: "Alice", score: 80 },
  { name: "Bob", score: 95 },
  { name: "Charlie", score: 70 }
];

console.log(findMaxValue(students, "score"));
// Output: 95
```

49. Check if a String starts with a specific character

Check if a given string starts with a specific character.

```
const startsWithChar = (str, char) => str.startsWith(char);

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

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

50. Convert a String to Title Case

Convert a given string to title case (capitalize the first letter of each word).

```
const toTitleCase = (str) => str.replace(/\b\w/g, match =>
match.toUpperCase());

console.log(toTitleCase("hello world"));
// Output: "Hello World"
```

51. Check if an Array contains a specific value

Check if a given array contains a specific value.

```
const containsValue = (arr, value) => arr.includes(value);

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

console.log(containsValue([1, 2, 3, 4, 5], 6));
// Output: false
```

52. Convert an Array to a Comma-separated String

Convert a given array to a comma-separated string.

```
const arrayToCSV = (arr) => arr.join(', ');

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

53. Check if a Year is a Leap Year

Check if a given year is a leap year.

```
const isLeapYear = (year) => (year % 4 === 0 && year % 100
!== 0) || (year % 400 === 0);

console.log(isLeapYear(2024));
// Output: true

console.log(isLeapYear(2023));
// Output: false
```

54. Find the Index of an Element in an Array

Find the index of a specific element in a given array.

```
const findIndex = (arr, element) => arr.indexOf(element);

const fruits = ["apple", "banana", "orange", "grape"];
console.log(findIndex(fruits, "orange"));
// Output: 2
```

55. Convert Minutes to Hours and Minutes

Convert a given number of minutes to hours and remaining minutes.

```
const minsToHoursAndMins = (mins) => {
  const hours = Math.floor(mins / 60);
  const remainingMins = mins % 60;
  return `${hours} hours and ${remainingMins} minutes`;
};

console.log(minsToHoursAndMins(150));
// Output: "2 hours and 30 minutes"
```

56. Check if an Array is Sorted in Ascending Order

Check if a given array is sorted in ascending order.

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

console.log(isSortedAscending([1, 2, 3, 5, 8]));
// Output: true

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

57. Remove a Specific Element from an Array

Remove a specific element from a given array.

```
const removeElement = (arr, element) => arr.filter(el => el
!== element);

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

58. Truncate a String to a Given Length

Truncate a given string to a specified maximum length.

```
const truncateString = (str, maxLength) => str.length > maxLength ? str.slice(0, maxLength) + '...' : str;

console.log(truncateString("Hello, world!", 5));
// Output: "Hello..."
```

59. Calculate the Exponentiation of a Number

Calculate the exponentiation of a given base raised to a specified exponent.

```
const exponentiate = (base, exponent) => Math.pow(base,
exponent);

console.log(exponentiate(2, 3));
// Output: 8
```

60. Find the Difference between Two Dates in Days

Find the difference between two dates in days.

```
const dateDifferenceInDays = (date1, date2) =>
Math.abs(Math.floor((date2 - date1) / (1000 * 60 * 60 *
24)));

const startDate = new Date("2023-08-01");
const endDate = new Date("2023-08-10");
console.log(dateDifferenceInDays(startDate, endDate));
// Output: 9
```

61. Check if a String is a Valid Email Address

Check if a given string is a valid email address.

```
const isValidEmail = (email) =>
/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);

console.log(isValidEmail("user@example.com"));
// Output: true

console.log(isValidEmail("invalid-email"));
// Output: false
```

62. Convert Seconds to Minutes and Seconds

Convert a given number of seconds to minutes and remaining seconds.

```
const secsToMinsAndSecs = (seconds) => {
  const mins = Math.floor(seconds / 60);
  const remainingSecs = seconds % 60;
  return `${mins} minutes and ${remainingSecs} seconds`;
};

console.log(secsToMinsAndSecs(120));
// Output: "2 minutes and 0 seconds"
```

63. Check if an Object is a Function

Check if a given object is a function.

```
const isFunction = (obj) => typeof obj === 'function';

console.log(isFunction(() => {}));
// Output: true

console.log(isFunction({}));
// Output: false
```

64. Convert Binary Number to Decimal

Convert a given binary number to its decimal representation.

```
const binaryToDecimal = (binary) => parseInt(binary, 2);

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

65. Check if an Array contains only Unique Values

Check if a given array contains only unique values.

```
const hasUniqueValues = (arr) => new Set(arr).size ===
arr.length;

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

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

66. Get the Day of the Week from a Date

Get the day of the week from a given date.

```
const getDayOfWeek = (date) => {
  const daysOfWeek = ["Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday"];
  return daysOfWeek[date.getDay()];
};

console.log(getDayOfWeek(new Date("2023-08-02")));
// Output: "Wednesday"
```

67. Check if a Number is a Power of Two

Check if a given number is a power of two.

```
const isPowerOfTwo = (num) => (num & (num - 1)) === 0;

console.log(isPowerOfTwo(16));
// Output: true

console.log(isPowerOfTwo(5));
// Output: false
```

68. Convert Object to Query Parameters String

Convert a given object to a string of query parameters.

```
const objectToQueryParams = (obj) =>
Object.entries(obj).map(([key, value]) =>
`${encodeURIComponent(key)}=${encodeURIComponent(value)}`).
join('&');

console.log(objectToQueryParams({ search: "hello", page: 1
}));
// Output: "search=hello&page=1"
```

69. Check if an Array contains an Even Number

Check if a given array contains at least one even number.

```
const hasEvenNumber = (arr) => arr.some(num => num % 2 === 0);

console.log(hasEvenNumber([1, 3, 5, 7, 8]));
// Output: true

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

70. Get the Month Name from a Date

Get the name of the month from a given date.

```
const getMonthName = (date) => {
  const months = ["January", "February", "March", "April",
"May", "June", "July", "August", "September", "October",
"November", "December"];
  return months[date.getMonth()];
};

console.log(getMonthName(new Date("2023-08-02")));
// Output: "August"
```

71. Check if a String is a Palindrome (case-insensitive)

Check if a given string is a palindrome, considering it case-insensitive.

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

console.log(isPalindromeCaseInsensitive("LeVel"));
// Output: true

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

72. Convert Feet to Meters

Convert a given length in feet to meters.

```
const feetToMeters = (feet) => feet * 0.3048;

console.log(feetToMeters(10));
// Output: 3.048
```

73. Check if a Number is a Perfect Square

Check if a given number is a perfect square.

```
const isPerfectSquare = (num) => Math.sqrt(num) % 1 === 0;

console.log(isPerfectSquare(16));
// Output: true

console.log(isPerfectSquare(10));
// Output: false
```

74. Check if a String contains only Numbers

Check if a given string contains only numeric characters.

```
const containsOnlyNumbers = (str) => /^[0-9]+$/.test(str);

console.log(containsOnlyNumbers("12345"));
// Output: true

console.log(containsOnlyNumbers("12a34"));
// Output: false
```

75. Get the Current Month (0-based index)

Get the current month as a 0-based index (0 for January, 1 for February, etc.).

```
const currentMonth = () => new Date().getMonth();

console.log(currentMonth());
// Output: Current month (0-based index)
```

76. Calculate the Mean of an Array of Numbers

Calculate the mean (average) of a given array of numbers.

```
const mean = (arr) => arr.reduce((acc, val) => acc + val, 0)
/ arr.length;

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

77. Check if a Number is a Prime Number

Check if a given number is a prime number.

```
const isPrime = (num) => {
  if (num <= 1) return false;
  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) return false;
  }
  return true;
};

console.log(isPrime(13));
// Output: true

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

78. Get the Last N Elements of an Array

Get the last N elements from a given array.

```
const lastNElements = (arr, n) => arr.slice(-n);

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

79. Convert Degrees to Radians

Convert a given angle from degrees to radians.

```
const degToRad = (degrees) => degrees * (Math.PI / 180);

console.log(degToRad(90));
// Output: 1.5707963267948966
```

80. Check if a String is a Valid URL

Check if a given string is a valid URL.

```
const isValidURL = (url) => {
  try {
    new URL(url);
    return true;
  } catch (error) {
    return false;
  }
};

console.log(isValidURL("https://www.example.com"));
// Output: true

console.log(isValidURL("invalid-url"));
// Output: false
```

81. Find the Intersection of Two Arrays

Find the common elements (intersection) between two arrays.

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

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

82. Convert Days to Years, Months, and Days

Convert a given number of days to years, months, and remaining days.

```
const daysToYearsMonthsDays = (days) => {
  const years = Math.floor(days / 365);
  const remainingDays = days % 365;
  const months = Math.floor(remainingDays / 30);
  const remainingDaysInMonth = remainingDays % 30;
    return `${years} years, ${months} months, and
${remainingDaysInMonth} days`;
};

console.log(daysToYearsMonthsDays(1000));
// Output: "2 years, 8 months, and 20 days"
```

83. Check if an Object is Empty (no own properties)

Check if a given object has no own properties (i.e., it is empty).

```
const isEmptyObject = (obj) => Object.keys(obj).length === 0;

console.log(isEmptyObject({}));
// Output: true

console.log(isEmptyObject({ name: "John", age: 30 }));
// Output: false
```

84. Calculate the Factorial of a Number (recursive)

Calculate the factorial of a given number using a recursive function.

```
const factorial = (num) => {
  if (num === 0 || num === 1) return 1;
  return num * factorial(num - 1);
};

console.log(factorial(5));
// Output: 120
```

85. Remove Whitespace from a String

Remove all whitespace characters from a given string.

```
const removeWhitespace = (str) => str.replace(/\s/g, '');

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

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.

```
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
```

107. Find the Average of Even Numbers in an Array

This function computes the average of even numbers present in an array. It first filters out the even numbers from the array, then calculates the sum of these even numbers, and finally divides the sum by the count of even numbers to obtain the average.

```
const averageOfEvenNumbers = (arr) => {
  const evenNumbers = arr.filter(num => num % 2 === 0);
  return evenNumbers.reduce((sum, num) => sum + num, 0) /
evenNumbers.length;
};

console.log(averageOfEvenNumbers([1, 2, 3, 4, 5, 6, 7, 8,
9, 10]));
// Output: 6
```

108. Capitalize the First Letter of Each Word in a String

This function transforms a string by capitalizing the first letter of each word within it. It employs a regular expression to locate the first character of each word (defined as a word boundary followed by a letter), and then replaces it with its uppercase version.

```
const capitalizeWords = (str) => str.replace(/\b\w/g, char
=> char.toUpperCase());

console.log(capitalizeWords("hello world"));
// Output: "Hello World"
```

109. Check if an Array is a Subset of Another Array

The isSubset function determines if one array is a subset of another array. It achieves this by verifying that every element in the first array (arr1) is present in the second array (arr2).

```
const isSubset = (arr1, arr2) => arr1.every(item => arr2.includes(item));

console.log(isSubset([1, 2, 3], [2, 3, 4, 5, 6]));
// Output: false

console.log(isSubset([1, 2, 3], [2, 3, 1, 5, 6]));
// Output: true
```

110. Find the Minimum and Maximum Numbers in an Array

The **minMax** function calculates both the minimum and maximum values within an array. It achieves this by employing the Math.min and Math.max functions alongside the spread operator to extract the elements' values from the input array. The function returns an object containing both the minimum and maximum values.

```
const minMax = (arr) => ({
  min: Math.min(...arr),
  max: Math.max(...arr)
});

console.log(minMax([10, 5, 25, 3, 15]));
// Output: { min: 3, max: 25 }
```

111. Check if a Number is a Narcissistic Number

The isNarcissisticNumber function evaluates whether a given number is a narcissistic number.

```
const isNarcissisticNumber = (num) => {
  const digits = [...String(num)].map(Number);
  const numDigits = digits.length;
  const sumOfPowers = digits.reduce((sum, digit) => sum +
Math.pow(digit, numDigits), 0);
  return sumOfPowers === num;
};

console.log(isNarcissisticNumber(153));
// Output: true

console.log(isNarcissisticNumber(370));
// Output: true

console.log(isNarcissisticNumber(123));
// Output: false
```

112. Remove Null and Undefined Values from an Array

The **removeNullAndUndefined** function eliminates null and undefined values from an array by utilizing the filter method. It returns a new array containing only the non-null and non-undefined elements.

```
const removeNullAndUndefined = (arr) => arr.filter(item =>
item !== null && item !== undefined);

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

113. Reverse the Order of Words in a String

The **reverseWords** function takes a string as input and returns a new string where the order of words has been reversed.

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

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

114. Calculate the Sum of Cubes of an Array

The sumOfCubes function computes the sum of the cubes of all numbers in an array. It employs the reduce method to iterate through the array, accumulating the sum of cubes by raising each value to the power of 3 and adding it to the accumulator.

```
const sumOfCubes = (arr) => arr.reduce((acc, val) => acc + val ** 3, 0);

console.log(sumOfCubes([1, 2, 3, 4, 5]));
// Output: 225
```

115. Shuffle the Characters of a String

The **shuffleString** function rearranges the characters of a given string in a random order. It does so by first splitting the string into an array of characters, then using the sort method with a random comparison function. Finally, the shuffled characters are rejoined to form a new string.

```
const shuffleString = (str) => str.split('').sort(() => 0.5
- Math.random()).join('');

console.log(shuffleString("hello"));
// Output: Randomly shuffled string, e.g., "olelh"
```

116. Find the Nth Fibonacci Number (recursive)

The fibonacci function calculates the Nth Fibonacci number using a recursive approach. It determines the Fibonacci number by summing the previous two Fibonacci numbers until it reaches the base cases of 0 and 1.

```
const fibonacci = (n) => (n <= 1 ? n : fibonacci(n - 1) +
fibonacci(n - 2));

console.log(fibonacci(7));
// Output: 13
```

117. Count the Words in a String

The countWords function determines the number of words in a given string. It accomplishes this by splitting the string using a regular expression that matches one or more whitespace characters, and then counting the resulting array's length.

```
const countWords = (str) => str.split(/\s+/).length;

console.log(countWords("Hello, how are you doing?"));
// Output: 5
```

118. Check if a Number is a Triangular Number

The isTriangularNumber function determines whether a given number is a triangular number.

```
const isTriangularNumber = (num) => {
  let n = 0;
  let sum = 0;
  while (sum < num) {
    n++;
    sum += n;
  }
  return sum === num;
};

console.log(isTriangularNumber(10));
// Output: true

console.log(isTriangularNumber(15));
// Output: true

console.log(isTriangularNumber(7));
// Output: false
```

119. Calculate the Perimeter of a Rectangle

The **rectanglePerimeter** function calculates the perimeter of a rectangle by summing twice the width and twice the height of the rectangle.

```
const rectanglePerimeter = (width, height) => 2 * (width + height);

console.log(rectanglePerimeter(5, 10));
// Output: 30
```

120. Find the Longest Common Prefix in an Array of Strings

The **longestCommonPrefix** function finds the longest common prefix among an array of strings. It starts by assuming the first string in the array as the initial common prefix.

```
const longestCommonPrefix = (strs) => {
  if (strs.length === 0) return '';
  let prefix = strs[0];
  for (let i = 1; i < strs.length; i++) {
    while (!strs[i].startsWith(prefix)) {
      prefix = prefix.slice(0, prefix.length - 1);
    }
  }
  return prefix;
};

console.log(longestCommonPrefix(['apple', 'apricot',
'appetizer']));
// Output: "app"
```

121. Get the ASCII Value of a Character

The **getASCIIValue** function retrieves the ASCII value of a given character. It employs the charCodeAt method, which returns the ASCII code of the character at the specified index (0 in this case).

```
const getASCIIValue = (char) => char.charCodeAt(0);

console.log(getASCIIValue('A'));
// Output: 65
```

122. Find the First Non-Repeated Character in a String

The **firstNonRepeatedChar** function identifies the first non-repeated character within a given string. It accomplishes this by iterating through the string, building a character count object, and then using the find method to locate the first character with a count of 1.

```
const firstNonRepeatedChar = (str) => {
  const charCount = {};
  for (const char of str) {
    charCount[char] = (charCount[char] || 0) + 1;
  }
  return str.split('').find((char) => charCount[char] === 1);
};

console.log(firstNonRepeatedChar('abacabad'));
// Output: "c"
```

123. Sort an Array of Objects by a Property Value

The **sortByProperty** function arranges an array of objects based on a specified property's value. It employs the sort method with a custom comparison function that compares the property values of two objects.

```
const sortByProperty = (arr, prop) => arr.sort((a, b) =>
a[prop] - b[prop]);

const people = [
  { name: 'Alice', age: 25 },
  { name: 'Bob', age: 20 },
  { name: 'Charlie', age: 30 }
];

console.log(sortByProperty(people, 'age'));
// Output: [{ name: 'Bob', age: 20 }, { name: 'Alice', age:
25 }, { name: 'Charlie', age: 30 }]
```

124. Calculate the Exponential of a Number

The **exponential** function calculates the result of raising a given base to a specified exponent using the exponentiation operator (**).

```
const exponential = (base, exponent) => base ** exponent;

console.log(exponential(2, 3));
// Output: 8
```

125. Check if a String is an Anagram of Another String

The **isAnagram** function determines whether two given strings are anagrams of each other. An anagram is a word or phrase formed by rearranging the letters of another, using all the original letters exactly once.

```
const isAnagram = (str1, str2) =>
str1.split('').sort().join('') ===
str2.split('').sort().join('');

console.log(isAnagram('listen', 'silent'));
// Output: true

console.log(isAnagram('hello', 'world'));
// Output: false
```

126. Find the Factors of a Number

The **factors** function calculates and returns an array of all the factors of a given number. Factors are the positive integers that evenly divide the input number.

```
const factors = (num) => {
  const result = [];
  for (let i = 1; i <= num; i++) {
    if (num % i === 0) {
      result.push(i);
    }
  }
  return result;
};

console.log(factors(12));
// Output: [1, 2, 3, 4, 6, 12]
```

127. Check if a Number is a Neon Number

The **isNeonNumber** function determines whether a given number is a neon number. A neon number is a number where the sum of the digits of its square is equal to the number itself.

```
const isNeonNumber = (num) => {
  const squared = num ** 2;
  const digitSum = [...String(squared)].map(Number).reduce((sum, digit) => sum + digit, 0);
  return squared === digitSum;
};

console.log(isNeonNumber(9));
// Output: true

console.log(isNeonNumber(12));
// Output: false
```

128. Find the Power Set of a Set

The **powerSet** function generates the power set of a given set, which includes all possible subsets of the set, including the empty set and the set itself.

```
const powerSet = (set) => {
  const result = [[]];
  for (const item of set) {
    const subsets = result.map((subset) => [...subset,
item]);
    result.push(...subsets);
  }
  return result;
};

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

129. Check if a Number is a Disarium Number

The **isDisariumNumber** function determines whether a given number is a disarium number.

```
const isDisariumNumber = (num) => {
  const digits = [...String(num)].map(Number);
  const sumOfPowers = digits.reduce((sum, digit, index) =>
sum + digit ** (index + 1), 0);
  return sumOfPowers === num;
};

console.log(isDisariumNumber(89));
// Output: true

console.log(isDisariumNumber(135));
// Output: true

console.log(isDisariumNumber(23));
// Output: false
```

130. Remove Vowels from a String

The **removeVowels** function eliminates all vowels (both uppercase and lowercase) from a given string using the replace method along with a regular expression.

```
const removeVowels = (str) => str.replace(/[aeiouAEIOU]/g,
'');

console.log(removeVowels("Hello, World!"));
// Output: "Hll, Wrld!"
```

131. Generate an Array of Consecutive Numbers

The **consecutiveNumbers** function generates an array of consecutive numbers within a specified range.

```
const consecutiveNumbers = (start, end) => Array.from({
length: end - start + 1 }, (_, i) => start + i);

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

132. Check if a Number is a Pronic Number

The **isPronicNumber** function determines whether a given number is a pronic number (also known as an oblong number or rectangular number).

```
const isPronicNumber = (num) => {
  const n = Math.floor(Math.sqrt(num));
  return n * (n + 1) === num;
};

console.log(isPronicNumber(6));
// Output: true

console.log(isPronicNumber(20));
// Output: true

console.log(isPronicNumber(7));
// Output: false
```

133. Check if a String is a Pangram

The **isPangram** function checks whether a given string is a pangram, which is a sentence that contains every letter of the alphabet at least once.

```
const isPangram = (str) => {
  const letters = new Set(str.toLowerCase().match(/[a-
z]/g));
  return letters.size === 26;
};

console.log(isPangram("The quick brown fox jumps over the
lazy dog"));
// Output: true

console.log(isPangram("Hello, World!"));
// Output: false
```

134. Reverse the Order of Words in a Sentence

The **reverseSentence** function reverses the order of words in a given sentence.

```
const reverseSentence = (sentence) => sentence.split('
').reverse().join(' ');

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

135. Calculate the Hypotenuse of a Right-Angled Triangle

The **binaryToDecimalWithParseInt** function converts a binary number to a decimal number using the built-in parseInt function with the base parameter set to 2.

```
const binaryToDecimalWithParseInt = (binary) =>
parseInt(binary, 2);

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

136. Find the Average of Odd Numbers in an Array

The **averageOfOddNumbers** function calculates the average of odd numbers within a given array. It does so by filtering the odd numbers from the array, then computing their sum and dividing by the count of odd numbers.

```
const averageOfOddNumbers = (arr) => {
  const oddNumbers = arr.filter(num => num % 2 !== 0);
  return oddNumbers.reduce((sum, num) => sum + num, 0) /
oddNumbers.length;
};

console.log(averageOfOddNumbers([1, 2, 3, 4, 5, 6, 7, 8, 9,
10]));
// Output: 5
```

137. Count the Letters in a String (case-insensitive)

The **countLetters** function calculates the count of each letter in a given string, considering both uppercase and lowercase versions as the same letter.

```
const countLetters = (str) => {
  const letters = str.toLowerCase().match(/[a-z]/g);
  const letterCount = {};
  for (const letter of letters) {
    letterCount[letter] = (letterCount[letter] || 0) + 1;
  }
  return letterCount;
};

console.log(countLetters("Hello, World!"));
// Output: { h: 1, e: 1, l: 3, o: 2, w: 1, r: 1, d: 1 }
```

138. Convert Seconds to Days, Hours, Minutes, and Seconds

The **secsToDaysHoursMinsSecs** function converts a given number of seconds into days, hours, minutes, and remaining seconds.

```
const secsToDaysHoursMinsSecs = (seconds) => {
  const days = Math.floor(seconds / 86400);
  seconds %= 86400;
  const hours = Math.floor(seconds / 3600);
  seconds %= 3600;
  const minutes = Math.floor(seconds / 60);
  const remainingSecs = seconds % 60;
  return `${days} days, ${hours} hours, ${minutes} minutes,
and ${remainingSecs} seconds`;
};

console.log(secsToDaysHoursMinsSecs(100000));
// Output: "1 days, 3 hours, 46 minutes, and 40 seconds"
```

139. Check if a Number is a Prime Factor of Another Number

The **isPrimeFactor** function checks if a given number is a prime factor of another number.

```
const isPrimeFactor = (num, factor) => num % factor === 0
&& isPrime(factor);

console.log(isPrimeFactor(20, 2));
// Output: true

console.log(isPrimeFactor(20, 3));
// Output: false
```

140. Find the Largest Prime Factor of a Number

The **largestPrimeFactor** function calculates the largest prime factor of a given number.

```
const largestPrimeFactor = (num) => {
  let factor = 2;
  while (factor <= num) {
    if (num % factor === 0) {
      num /= factor;
    } else {
      factor++;
    }
  }
  return factor;
};

console.log(largestPrimeFactor(48));
// Output: 3
```

141. Check if a Number is a Pronic Square

The **isPronicSquare** function checks if a given number is a pronic square.

```
const isPronicSquare = (num) =>
Number.isInteger(Math.sqrt(num));

console.log(isPronicSquare(6));
// Output: true

console.log(isPronicSquare(20));
// Output: false

console.log(isPronicSquare(21));
// Output: true
```

142. Find the Sum of the Digits of a Number

The **sumOfDigits** function calculates the sum of the digits of a given number.

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

console.log(sumOfDigits(12345));
// Output: 15
```

143. Calculate the Median of an Array of Numbers

The **median** function calculates the median of an array of numbers. The median is the middle value of a dataset when it is ordered.

```
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
```

144. Find the Greatest Common Divisor (GCD) of Two Numbers (Recursive)

The **gcd** function calculates the greatest common divisor (GCD) of two given numbers using a recursive approach.

```
const gcd = (num1, num2) => (num2 === 0 ? num1 : gcd(num2, num1 % num2));

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

145. Check if a Number is a Happy Number

The **isHappyNumber** function checks if a given number is a "happy number" or not. A happy number is a number where the sequence of repeatedly summing the squares of its digits eventually reaches the number 1.

```
const isHappyNumber = (num) => {
  const seen = new Set();
  while (num !== 1 && !seen.has(num)) {
    seen.add(num);
    num = [...String(num)].reduce((sum, digit) => sum +
digit ** 2, 0);
  }
  return num === 1;
};

console.log(isHappyNumber(19));
// Output: true

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

146. Find the First N Prime Numbers

The **firstNPrimes** function generates an array of the first n prime numbers.

```
const isPrime = (num) => {
  if (num <= 1) return false;
  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) return false;
  }
  return true;
};

const firstNPrimes = (n) => {
  const primes = [];
  let num = 2;
  while (primes.length < n) {
    if (isPrime(num)) primes.push(num);
    num++;
  }
  return primes;
};

console.log(firstNPrimes(5));
// Output: [2, 3, 5, 7, 11]
```

147. Calculate the Volume of a Sphere

The **sphereVolume** function calculates the volume of a sphere given its radius. It uses the formula (4/3) * π * r^3, where r is the radius of the sphere.

```
const sphereVolume = (radius) => (4 / 3) * Math.PI * radius
** 3;

console.log(sphereVolume(5));
// Output: 523.5987755982989
```

148. Find the Longest Word in a Sentence

The **longestWord** function determines the longest word in a given sentence. It does this by splitting the sentence into words using spaces as separators and then using the reduce method to compare the length of each word and keep track of the longest one.

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

console.log(longestWord("The quick brown fox jumped over
the lazy dog"));
// Output: "jumped"
```

149. Check if a Number is an Armstrong Number (Narcissistic Number)

The **isArmstrongNumber** function checks if a number is an Armstrong number, also known as a narcissistic number.

```
const isArmstrongNumber = (num) => {
  const digits = [...String(num)].map(Number);
  const numDigits = digits.length;
  const sumOfPowers = digits.reduce((sum, digit) => sum +
digit ** numDigits, 0);
  return sumOfPowers === num;
};

console.log(isArmstrongNumber(153));
// Output: true

console.log(isArmstrongNumber(370));
// Output: true

console.log(isArmstrongNumber(123));
// Output: false
```

150. Find the Length of the Longest Word in a Sentence

The **longestWordLength** function calculates the length of the longest word in a given sentence. It splits the sentence into words using spaces as separators, and then uses the reduce method to find the maximum length among all the words.

```
const longestWordLength = (sentence) => sentence.split(' ').reduce((longest, word) => Math.max(longest, word.length), 0);

console.log(longestWordLength("The quick brown fox jumped over the lazy dog"));
// Output: 6
```

151. Check if a Number is a Strong Number

The **isStrongNumber** function checks if a number is a strong number. A strong number is a number whose sum of factorials of its digits is equal to the number itself.

```
const factorial = (num) => (num === 0 ? 1 : num *
factorial(num - 1));

const isStrongNumber = (num) => {
  const sumOfFactorials = [...String(num)].reduce((sum,
digit) => sum + factorial(Number(digit)), 0);
  return sumOfFactorials === num;
};

console.log(isStrongNumber(145));
// Output: true

console.log(isStrongNumber(123));
// Output: false
```

152. Reverse the Order of an Array

The **reverseArray** function reverses the order of elements in an array using the reverse method.

```
const reverseArray = (arr) => arr.reverse();

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

153. Find the Area of a Rectangle

The **rectangleArea** function calculates the area of a rectangle given its length and width using the formula: length * width.

```
const rectangleArea = (length, width) => length * width;

console.log(rectangleArea(5, 10));
// Output: 50
```

154. Calculate the Sum of Even Numbers in an Array

The **sumOfEvenNumbers** function calculates the sum of even numbers within an array. It first filters the array to keep only the even numbers, and then uses the reduce method to compute their sum.

```
const sumOfEvenNumbers = (arr) => arr.filter(num => num % 2
=== 0).reduce((sum, num) => sum + num, 0);

console.log(sumOfEvenNumbers([1, 2, 3, 4, 5, 6, 7, 8, 9,
10]));
// Output: 30
```

155. Find the Greatest Common Divisor (GCD) of Two Numbers (Iterative)

The **gcdIterative** function calculates the greatest common divisor (GCD) of two numbers using an iterative approach. It employs the Euclidean algorithm to iteratively find the GCD.

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

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

156. Calculate the Volume of a Cylinder

The **cylinderVolume** function calculates the volume of a cylinder using the formula: π * radius^2 * height.

```
const cylinderVolume = (radius, height) => Math.PI * radius
** 2 * height;

console.log(cylinderVolume(5, 10));
// Output: 785.3981633974483
```

157. Check if a Number is a Smith Number

The **isSmithNumber** function checks whether a given number is a Smith number.

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

const sumOfPrimeFactors = (num) => {
  let factor = 2;
  let sum = 0;
  while (num > 1) {
    if (num % factor === 0) {
      sum += sumOfDigits(factor);
      num /= factor;
    } else {
      factor++;
    }
  }
  return sum;
};

const isSmithNumber = (num) => sumOfDigits(num) === sumOfPrimeFactors(num);

console.log(isSmithNumber(666));
// Output: true

console.log(isSmithNumber(378));
// Output: true

console.log(isSmithNumber(123));
// Output: false
```

158. Convert Decimal Number to Octal

The **decimalToOctal** function converts a decimal (base 10) number to its octal (base 8) representation using the .toString() method with the base argument set to 8.

```
const decimalToOctal = (num) => num.toString(8);

console.log(decimalToOctal(27));
// Output: "33"
```

159. Find the LCM of Two Numbers

The **lcm** function calculates the least common multiple (LCM) of two given numbers using the formula: (num1 * num2) / gcd(num1, num2).

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

console.log(lcm(24, 36)); // Output: 72
```

160. Check if a String is a Valid Phone Number (North American Format)

The **isValidPhoneNumber** function checks whether a given string is a valid phone number in the North American format "XXX-XXX-XXXX", where X represents a digit.

```
const isValidPhoneNumber = (phone) => /^\d{3}-\d{3}-
\d{4}$/.test(phone);

console.log(isValidPhoneNumber("555-123-4567"));
// Output: true

console.log(isValidPhoneNumber("123-4567"));
// Output: false
```

161. Find the Sum of the First N Natural Numbers

The **sumOfNaturals** function calculates the sum of the first N natural numbers using the formula: (n * (n + 1)) / 2.

```
const sumOfNaturals = (n) => (n * (n + 1)) / 2;

console.log(sumOfNaturals(10));
// Output: 55
```

162. Check if a Number is a Perfect Number

The **isPerfectNumber** function checks whether a given number is a perfect number. A perfect number is a positive integer that is equal to the sum of its proper divisors (excluding itself).

```
const isPerfectNumber = (num) => {
  let sum = 0;
  for (let i = 1; i <= num / 2; i++) {
    if (num % i === 0) sum += i;
  }
  return sum === num;
};

console.log(isPerfectNumber(28));
// Output: true

console.log(isPerfectNumber(12));
// Output: false
```

163. Find the Factors of a Number (excluding 1 and the number itself)

The **factors** function calculates the factors of a given number, excluding 1 and the number itself. It iterates through the numbers from 2 up to one less than the given number, checking if the given number is divisible by each of those numbers.

```
const factors = (num) => {
  const result = [];
  for (let i = 2; i < num; i++) {
    if (num % i === 0) result.push(i);
  }
  return result;
};

console.log(factors(12));
// Output: [2, 3, 4, 6]
```

164. Calculate the Area of a Triangle given the Base and Height

The **triangleArea** function calculates the area of a triangle using the formula: 0.5 * base * height.

```
const triangleArea = (base, height) => 0.5 * base * height;

console.log(triangleArea(5, 10));
// Output: 25
```

165. Check if a String is a Valid Social Security Number (SSN)

The **isValidSSN** function checks whether a given string is a valid Social Security Number (SSN) in the format "XXX-XX-XXXX", where X represents a digit.

```
const isValidSSN = (ssn) => /^\d{3}-\d{2}-
\d{4}$/.test(ssn);

console.log(isValidSSN("123-45-6789"));
// Output: true

console.log(isValidSSN("123-45-678"));
// Output: false
```

166. Generate an Array of Random Numbers within a Range

The **randomArrayInRange** function generates an array of random numbers within a specified range and of a specified length. It uses the Array.from method with a mapping function to create the desired array.

```
const randomArrayInRange = (min, max, length) => Array.from({
length }, () => Math.floor(Math.random() * (max - min + 1))
+ min);

console.log(randomArrayInRange(1, 100, 5));
// Output: [34, 87, 19, 56, 72]
```

167. Check if a Number is a Magic Number

The **isMagicNumber** function checks whether a given number is a magic number. A magic number is a number that eventually reaches the value 1 when the sum of its digits is repeatedly calculated.

```
const isMagicNumber = (num) => {
  let sum = 0;
  while (num > 0) {
    sum += num % 10;
    num = Math.floor(num / 10);
  }
  return sum === 1;
};

console.log(isMagicNumber(19));
// Output: true

console.log(isMagicNumber(123));
// Output: false
```

168. Check if a String is a Valid IPv4 Address

The **isValidIPv4** function checks whether a given string represents a valid IPv4 address.

```
const isValidIPv4 = (ip) => /^(?:(?:25[0-5]|2[0-4][0-
9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)$/.test(ip);

console.log(isValidIPv4("192.168.1.1"));
// Output: true

console.log(isValidIPv4("256.0.0.1"));
// Output: false
```

169. Convert Decimal Number to Hexadecimal

The **decimalToHex** function converts a decimal (base 10) number to its equivalent hexadecimal (base 16) representation using the built-in toString method with a radix of 16.

```
const decimalToHex = (num) => num.toString(16);

console.log(decimalToHex(255));
// Output: "ff"
```

170. Check if a String is a Valid Date (YYYY-MM-DD Format)

The **isValidDate** function checks whether a given string is a valid date in the format "YYYY-MM-DD".

```
const isValidDate = (date) => /^\d{4}-\d{2}-
\d{2}$/.test(date);

console.log(isValidDate("2023-08-02"));
// Output: true

console.log(isValidDate("02-08-2023"));
// Output: false
```

171. Find the Smallest Common Multiple of an Array of Numbers

In this code, the gcd function calculates the greatest common divisor using the Euclidean algorithm. The **lcmArray** function then calculates the least common multiple (LCM) of an array of numbers by reducing the array and applying the formula (lcm * num) / gcd(lcm, num).

```
const lcmArray = (arr) => arr.reduce((lcm, num) => (lcm * num) / gcd(lcm, num));

console.log(lcmArray([2, 3, 4, 5]));
// Output: 60
```

172. Check if a String is a Valid Password (At least 8 characters, with a digit and special character)

The **isValidPassword** function uses a regular expression to validate a password. The regular expression requires that the password contains at least one letter ([A-Za-z]), one digit (\d), and one special character ([@$!%*?&]).

```
const isValidPassword = (password) => /^(?=.*[A-Za-
z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-
z\d@$!%*?&]{8,}$/.test(password);

console.log(isValidPassword("P@ssw0rd"));
// Output: true

console.log(isValidPassword("password123"));
// Output: false
```

173. Find the Nth Fibonacci Number

The **fibonacci** function calculates the Nth Fibonacci number using an iterative approach.

```
const fibonacci = (n) => {
  if (n === 1) return 0;
  if (n === 2) return 1;
  let prev1 = 0;
  let prev2 = 1;
  let result;
  for (let i = 3; i <= n; i++) {
    result = prev1 + prev2;
    prev1 = prev2;
    prev2 = result;
  }
  return result;
};

console.log(fibonacci(7));
// Output: 8
```

174. Check if a Number is a Deficient Number

The **isDeficientNumber** function determines whether a given number is a deficient number.

```
const isDeficientNumber = (num) => num >
sumOfProperDivisors(num);

console.log(isDeficientNumber(10));
// Output: true

console.log(isDeficientNumber(28));
// Output: false
```

175. Calculate the Distance between Two Points in 2D

The **distanceBetweenPoints** function calculates the Euclidean distance between two points in a 2D plane. Given the coordinates of two points (x1, y1) and (x2, y2), it uses the formula sqrt((x2 - x1)^2 + (y2 - y1)^2) to compute the distance between them.

```
const distanceBetweenPoints = (x1, y1, x2, y2) =>
Math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2);

console.log(distanceBetweenPoints(0, 0, 3, 4));
// Output: 5
```

176. Check if a Number is an Abundant Number

The **isAbundantNumber** function determines whether a given number is an abundant number.

```
const isAbundantNumber = (num) => num <
sumOfProperDivisors(num);

console.log(isAbundantNumber(12));
// Output: true

console.log(isAbundantNumber(28));
// Output: false
```

177. Calculate the Volume of a Cube

The **cubeVolume** function calculates the volume of a cube based on its side length. The formula for the volume of a cube is side^3, where side is the length of one side of the cube.

```
const cubeVolume = (side) => side ** 3;

console.log(cubeVolume(5));
// Output: 125
```

178. Check if a String is a Valid Credit Card Number (Visa, MasterCard, Discover, American Express)

The **isValidCreditCard** function uses a regular expression to validate a credit card number.

```
const isValidCreditCard = (card) => /^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|6(?:011|5[0-9][0-9])[0-9]{12}|3[47][0-9]{13})$/.test(card);

console.log(isValidCreditCard("4012-3456-7890-1234"));
// Output: true

console.log(isValidCreditCard("1234-5678-9012-3456"));
// Output: false
```

179. Calculate the Perimeter of a Triangle

The **trianglePerimeter** function calculates the perimeter of a triangle given its three side lengths. It simply adds up the lengths of all three sides and returns the result as the perimeter of the triangle.

```
const trianglePerimeter = (side1, side2, side3) => side1 +
side2 + side3;

console.log(trianglePerimeter(5, 10, 7));
// Output: 22
```

180. Check if a Number is a Vampire Number

The **isVampireNumber** function checks if a given number is a vampire number.

```
const isVampireNumber = (num) => {
  const numStr = String(num);
  const numLen = numStr.length;
  const numDigits = [...numStr];
  const numFactors = getFactors(num);
  for (const factor1 of numFactors) {
    const factor2 = num / factor1;
    if (factor1 % 10 === 0 && factor2 % 10 === 0) continue;
    const factorStr = String(factor1) + String(factor2);
    if (factorStr.length === numLen &&
areAnagrams(numDigits, [...factorStr])) {
      return true;
    }
  }
  return false;
};

console.log(isVampireNumber(1260)); // Output: true

console.log(isVampireNumber(1250)); // Output: false
```

181. Find the Sum of Digits Raised to the Power of their Respective Position

sumOfDigitsRaisedToPower calculates the sum of digits raised to the power of their respective positions.

```
const sumOfDigitsRaisedToPower = (num) => {
  const digits = [...String(num)].map(Number);
  return digits.reduce((sum, digit, index) => sum + digit **
(index + 1), 0);
};

console.log(sumOfDigitsRaisedToPower(12345));
// Output: 115
```

182. Check if a Number is a Duck Number

The **isDuckNumber** function checks whether a given number is a duck number. A duck number is a number that contains the digit "0" but does not start with "0".

```
const isDuckNumber = (num) => String(num).includes('0') &&
String(num)[0] !== '0';

console.log(isDuckNumber(1023));
// Output: true

console.log(isDuckNumber(12345));
// Output: false
```

183. Generate a Random Password

The **randomPassword** function generates a random password of the specified length.

```
const randomPassword = (length) => Array.from({ length },
() => Math.random().toString(36).charAt(2)).join('');

console.log(randomPassword(8));
// Output: "3klS0p9x"
```

184. Calculate the Area of a Trapezoid

The **trapezoidArea** function calculates the area of a trapezoid using the formula: 0.5 * (base1 + base2) * height.

```
const trapezoidArea = (base1, base2, height) => 0.5 *
(base1 + base2) * height;

console.log(trapezoidArea(4, 8, 6));
// Output: 36
```

185. Check if a Number is a Kaprekar Number

The **isKaprekarNumber** function checks if a given number is a Kaprekar number.

```
const isKaprekarNumber = (num) => {
  const square = num ** 2;
  const squareStr = String(square);
  const numStr = String(num);
  const left = Number(squareStr.slice(0, numStr.length));
  const right = Number(squareStr.slice(-numStr.length));
  return left + right === num;
};

console.log(isKaprekarNumber(9));
// Output: true

console.log(isKaprekarNumber(297));
// Output: true

console.log(isKaprekarNumber(45));
// Output: false
```

186. Calculate the Volume of a Cone

The **coneVolume** function calculates the volume of a cone using its base radius and height. It applies the formula for the volume of a cone: V = (1/3) * π * r² * h, where r is the radius of the base and h is the height of the cone.

```
const coneVolume = (radius, height) => (1 / 3) * Math.PI * radius ** 2 * height;

console.log(coneVolume(5, 10));
// Output: 261.79938779914943
```

187. Check if a String is a Valid US Phone Number

The **isValidUSPhoneNumber** function uses a regular expression to validate US phone numbers.

```
const isValidUSPhoneNumber = (phone) =>
/^(?:(?:\+1\s?)?(?:\(?\d{3}\)?[\s.-]?)?\d{3}[\s.-
]?\d{4})$/.test(phone);

console.log(isValidUSPhoneNumber("+1 (123) 456-7890"));
// Output: true

console.log(isValidUSPhoneNumber("123-456-7890"));
// Output: true

console.log(isValidUSPhoneNumber("1-800-ABC-DEFG"));
// Output: false
```

188. Find the Sum of Digits Raised to the Power of their Respective Position (Up to 1000)

The **sumOfDigitsRaisedToPowerUpToThousand** function calculates the sum of numbers where each digit raised to the power of its respective position is equal to the number itself.

```
const sumOfDigitsRaisedToPowerUpToThousand = () => {
  let sum = 0;
  for (let i = 1; i <= 1000; i++) {
    if (sumOfDigitsRaisedToPower(i) === i) {
      sum += i;
    }
  }
  return sum;
};

console.log(sumOfDigitsRaisedToPowerUpToThousand());
// Output: 443839
```

189. Check if a Number is a Carol Number

The **isCarolNumber** function checks if a given number is a Carol number.

```
const isCarolNumber = (num) => {
  const carolPrime = (2 ** num) - 1;
  return isPrime(num) && isPerfectSquare(carolPrime);
};

console.log(isCarolNumber(7));
// Output: true

console.log(isCarolNumber(47));
// Output: true

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

190. Check if a Number is a Catalan Number

The **isCatalanNumber** function checks if a given number is a Catalan number. It first ensures that the input number is a non-negative integer.

```
const isCatalanNumber = (num) => num >= 0 &&
Number.isInteger(num) && num === ((factorial(2 * num)) /
(factorial(num + 1) * factorial(num)));

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

console.log(isCatalanNumber(10));
// Output: false
```

191. Calculate the Volume of a Cuboid

The **cuboidVolume** function calculates the volume of a cuboid given its length, width, and height.

```
const cuboidVolume = (length, width, height) => length *
width * height;

console.log(cuboidVolume(5, 10, 8));
// Output: 400
```

192. Check if a Number is a Dudeney Number

The **isDudeneyNumber** function checks if a number is a Dudeney number.

```
const isDudeneyNumber = (num) => num ===
sumOfDigitsCube(num);

console.log(isDudeneyNumber(512));
// Output: true

console.log(isDudeneyNumber(64));
// Output: false
```

193. Generate a Random Color (Hexadecimal Format)

Generate a random color in hexadecimal format (#RRGGBB).

```
const randomColorHex = () => `#${Math.floor(Math.random() *
16777215).toString(16)}`;

console.log(randomColorHex());
// Output: "#92b008"
```

194. Calculate the Area of a Circle Sector

Calculate the area of a circle sector given the radius and the central angle in degrees.

```
const circleSectorArea = (radius, angle) => (angle / 360) *
Math.PI * radius ** 2;

console.log(circleSectorArea(5, 90));
// Output: 11.780972450961725
```

195. Calculate the Area of a Regular Polygon

Calculate the area of a regular polygon given the side length and the number of sides.

```
const regularPolygonArea = (sideLength, numOfSides) =>
(numOfSides * sideLength ** 2) / (4 * Math.tan(Math.PI /
numOfSides));

console.log(regularPolygonArea(5, 6));
// Output: 64.9519052838329
```

196. Remove Duplicates from Array

Remove duplicates from an array while preserving the order of the elements.

```
const removeDuplicates = (arr) => [...new Set(arr)];

console.log(removeDuplicates([1, 2, 3, 3, 4, 4, 5, 5, 6]));
// Output: [ 1, 2, 3, 4, 5, 6 ]
```

197. Calculate the Area of an Ellipse

Calculate the area of an ellipse using its semi-major axis length (a) and semi-minor axis length (b).

```
const ellipseArea = (a, b) => Math.PI * a * b;

console.log(ellipseArea(5, 10));
// Output: 157.07963267948966
```

198. Check if a Number is a Leyland Number

Check if a given number is a Leyland number.

```
const isLeylandNumber = (num) => {
  let found = false;
  for (let x = 2; x <= Math.floor(Math.pow(num, 1 / 3)) &&
!found; x++) {
    for (let y = x + 1; x * y <= num && !found; y++) {
      if (Math.pow(x, y) + Math.pow(y, x) === num) {
        found = true;
      }
    }
  }
  return found;
};

console.log(isLeylandNumber(17));
// Output: true

console.log(isLeylandNumber(30));
// Output: true

console.log(isLeylandNumber(100));
// Output: false
```

199. Generate a Random UUID

Generate a random Universally Unique Identifier (UUID).

```
const randomUUID = () => {
  return "xxxxxxxx-xxxx-4xxx-yxxx-
xxxxxxxxxxxx".replace(/[xy]/g, function(c) {
    const r = Math.random() * 16 | 0;
    const v = c === "x" ? r : (r & 0x3 | 0x8);
    return v.toString(16);
  });
};

console.log(randomUUID());
// Output: "a0f768f5-6bf2-4f6b-a512-c9121ea1b44a"
```

200. Check if a String is a Valid IPv6 Address

Check if a given string represents a valid IPv6 address.

```
const isValidIPv6 = (ip) => /^([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}$/.test(ip);

console.log(isValidIPv6("2001:0db8:85a3:0000:0000:8a2e:0370:7334"));
// Output: true

console.log(isValidIPv6("2001:0db8:85a3::8a2e:0370:7334"));
// Output: true

console.log(isValidIPv6("256.0.0.0"));
// Output: false
```

201. Calculate the Area of a Parallelogram

Calculate the area of a parallelogram using the given base and height.

```
const parallelogramArea = (base, height) => base * height;

console.log(parallelogramArea(5, 10));
// Output: 50
```

202. Check if a String is a Valid MAC Address

Check if a given string is a valid MAC address.

```
const isValidMACAddress = (mac) => /^([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})$/.test(mac);

console.log(isValidMACAddress("00:1A:2B:3C:4D:5E"));
// Output: true

console.log(isValidMACAddress("00:1A:2B:3C:4D"));
// Output: false
```

203. Convert RGB to HSL (Hue, Saturation, Lightness)

Convert an RGB color value to its corresponding HSL representation (Hue, Saturation, Lightness).

```
const rgbToHSL = (r, g, b) => {
  [r, g, b] = [r, g, b].map(val => val / 255);
  const [max, min] = [Math.max(r, g, b), Math.min(r, g,
b)];
  let h = (max !== min) ? ((max === r ? g - b : (max === g
? b - r : r - g)) / (max - min) + (max === g ? 2 : (max ===
b ? 4 : 0))) / 6 : 0;
  let s = (max !== min) ? (l => l > 0.5 ? (max - min) / (2
- max - min) : (max - min) / (max + min))(l) : 0;
  let l = (max + min) / 2;

  return { h: Math.round(h * 360), s: Math.round(s * 100),
l: Math.round(l * 100) };
};

console.log(rgbToHSL(255, 0, 0)); // Output: { h: 0, s:
100, l: 50 }
console.log(rgbToHSL(0, 255, 0)); // Output: { h: 120, s:
100, l: 50 }
console.log(rgbToHSL(0, 0, 255)); // Output: { h: 240, s:
100, l: 50 }
```

204. Check if a Number is a Pandigital Number

Check if a given number is a pandigital number.

```
const isPandigitalNumber = (num) => {
  const numStr = String(num);
  const digits = new Set(numStr);
  return digits.size === numStr.length && !digits.has('0')
&& digits.size === Math.max(...numStr) - '0';
};

console.log(isPandigitalNumber(123456789));
// Output: true

console.log(isPandigitalNumber(987654321));
// Output: true

console.log(isPandigitalNumber(1023456789));
// Output: false
```

205. Calculate the Sum of Proper Divisors of a Number

Calculate the sum of proper divisors of a given number.

```
const sumOfProperDivisors = (num) => {
  let sum = 0;
  for (let i = 1; i <= Math.sqrt(num); i++) {
    if (num % i === 0) {
      sum += i;
      if (num / i !== i) {
        sum += num / i;
      }
    }
  }
  return sum - num;
};

console.log(sumOfProperDivisors(28));
// Output: 28

console.log(sumOfProperDivisors(12));
// Output: 16
```

206. Find the Least Common Multiple (LCM) of an Array of Numbers

Find the least common multiple (LCM) of an array of numbers.

```
const lcmArray = (arr) => arr.reduce((lcm, num) => lcm * num
/ gcdArray(arr), 1);

console.log(lcmArray([2, 3, 4]));
// Output: 12
```

207. Calculate the Sum of Squares of First n Natural Numbers

Calculate the sum of the squares of the first n natural numbers.

```
const sumOfSquares = (n) => (n * (n + 1) * (2 * n + 1)) /
6;

console.log(sumOfSquares(5));
// Output: 55
```

208. Check if a Number is a Powerful Number

Check if a given number is a powerful number.

```
const isPowerfulNumber = (num) => {
  const factors = primeFactors(num);
  return new Set(factors).size === factors.length;
};

console.log(isPowerfulNumber(16));
// Output: true

console.log(isPowerfulNumber(36));
// Output: false
```

209. Find the Product of Digits of a Number

Find the product of the digits of a given number.

```
const productOfDigits = (num) =>
[...String(num)].reduce((product, digit) => product *
Number(digit), 1);

console.log(productOfDigits(12345));
// Output: 120
```

210. Check if a Number is a Practical Number

Check if a given number is a practical number.

```
const isPracticalNumber = (num) => {
  const factors = primeFactors(num);
  for (let i = 2; i <= factors.length; i++) {
    if (sumOfArray(combinations(factors, i)) !== num) {
      return false;
    }
  }
  return true;
};

console.log(isPracticalNumber(6));
// Output: true

console.log(isPracticalNumber(12));
// Output: true

console.log(isPracticalNumber(14));
// Output: false
```

211. Calculate the Sum of Cubes of First n Natural Numbers

Calculate the sum of the cubes of the first n natural numbers.

```
const sumOfCubes = (n) => Math.pow((n * (n + 1)) / 2, 2);

console.log(sumOfCubes(5));
// Output: 225
```

212. Check if a Number is a Strange Number

Check if a given number is a strange number.

```
const isStrangeNumber = (num) => {
  const factors = primeFactors(num);
  return factors.every((factor) => sumOfDigits(factor) ===
sumOfDigits(num));
};

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

console.log(isStrangeNumber(22));
// Output: true

console.log(isStrangeNumber(20));
// Output: false
```

213. Check if a Number is a Tau Number

Check if a given number is a Tau number.

```
const isTauNumber = (num) => {
  const factors = primeFactors(num);
  return factors.some((factor) => sumOfDigits(factor) ===
sumOfDigits(num));
};

console.log(isTauNumber(15));
// Output: true

console.log(isTauNumber(9));
// Output: true

console.log(isTauNumber(25));
// Output: false
```

214. Generate a Random Alphanumeric String

Generate a random alphanumeric string of a given length.

```
const randomAlphanumericString = (length) => {
  const characters =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345
6789';
  let result = '';
  for (let i = 0; i < length; i++) {
    result += characters.charAt(Math.floor(Math.random() *
characters.length));
  }
  return result;
};

console.log(randomAlphanumericString(8));
// Output: "Yw83XmLb"
```

215. Calculate the Area of a Regular Hexagon

Calculate the area of a regular hexagon using its side length.

```
const regularHexagonArea = (sideLength) => (3 *
Math.sqrt(3) * sideLength ** 2) / 2;

console.log(regularHexagonArea(5));
// Output: 64.9519052838329
```

216. Calculate the Sum of Divisors of a Number

Calculate the sum of divisors of a given number.

```
const sumOfDivisors = (num) => sumOfArray(divisors(num));

console.log(sumOfDivisors(28));
// Output: 56

console.log(sumOfDivisors(12));
// Output: 28
```

217. Check if a Number is a Zeisel Number

Check if a given number is a Zeisel number.

```
const isZeiselNumber = (num) => {
  const factors = primeFactors(num);
  return factors.every((factor) => isPrime(factor + 1));
};

console.log(isZeiselNumber(1050));
// Output: true

console.log(isZeiselNumber(10));
// Output: false
```

218. Check if a Number is a Reversible Number

Check if a given number is a reversible number.

```
const isReversibleNumber = (num) => {
                const                reversedNum                =
Number(String(num).split('').reverse().join(''));
  return num + reversedNum === sumOfDigits(num);
};

console.log(isReversibleNumber(36));
// Output: true

console.log(isReversibleNumber(45));
// Output: true

console.log(isReversibleNumber(10));
// Output: false
```

219. Calculate the Circumference of a Circle

Calculate the circumference of a circle using its radius.

```
const circleCircumference = (radius) => 2 * Math.PI *
radius;

console.log(circleCircumference(5));
// Output: 31.41592653589793
```

220. Find the Shortest Word in a String

Find the shortest word in a given string.

```
const shortestWord = (str) => str.split(' 
').reduce((shortest, word) => (word.length < 
shortest.length ? word : shortest), '');

console.log(shortestWord("This is a test sentence"));
// Output: "a"
```

221. Find the Longest Word Length in a String

Find the length of the longest word in a given string.

```
const longestWordLength = (str) => Math.max(...str.split('
').map(word => word.length));

console.log(longestWordLength("This is a test sentence"));
// Output: 8
```

222. Find the Sum of Proper Divisors of a Number

Find the sum of the proper divisors of a given number.

```
const sumOfProperDivisors = (num) =>
sumOfArray(divisors(num)) - num;

console.log(sumOfProperDivisors(28));
// Output: 28

console.log(sumOfProperDivisors(12));
// Output: 16
```

223. Check if a Number is a Unitary Perfect Number

Check if a given number is a unitary perfect number.

```
const isUnitaryPerfectNumber = (num) => num === sumOfUnitaryDivisors(num);

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

console.log(isUnitaryPerfectNumber(28));
// Output: false
```

224. Calculate the Perimeter of a Regular Polygon

Calculate the perimeter of a regular polygon using its side length and the number of sides.

```
const regularPolygonPerimeter = (sideLength, numSides) =>
sideLength * numSides;

console.log(regularPolygonPerimeter(5, 6));
// Output: 30
```

225. Calculate the Area of an Equilateral Triangle

Calculate the area of an equilateral triangle using its side length.

```
const equilateralTriangleArea = (sideLength) =>
(Math.sqrt(3) * sideLength ** 2) / 4;

console.log(equilateralTriangleArea(5));
// Output: 10.825317547305486
```

226. Check if a Number is a Harshad Smith Number

Check if a given number is both a Harshad number and a Smith number.

```
const isHarshadSmithNumber = (num) => isHarshadNumber(num)
&& isSmithNumber(num);

console.log(isHarshadSmithNumber(22));
// Output: true

console.log(isHarshadSmithNumber(10));
// Output: false
```

227. Check if a Number is a Perfect Power

Check if a given number is a perfect power.

```
const isPerfectPower = (num) => {
    for (let i = 2; i * i <= num; i++) {
      let power = 2;
      let result = i * i;
      while (result <= num) {
        if (result === num) {
          return true;
        }
        result *= i;
        power++;
      }
    }
    return false;
  };

  console.log(isPerfectPower(64));
  // Output: true

  console.log(isPerfectPower(25));
  // Output: false
```

228. Calculate the Sum of Digits Raised to Their Own Power

Calculate the sum of digits raised to their own power for a given number and power.

```
const sumOfDigitsToPower = (num, power) =>
sumOfDigitsRaisedToPower(num, power);

console.log(sumOfDigitsToPower(123, 3));
// Output: 36

console.log(sumOfDigitsToPower(4150, 5));
// Output: 4150
```

229. Check if a Number is a Dudeney Number

Check if a given number is a Dudeney number.

```
const isDudeneyNumber = (num) => Math.cbrt(num) === sumOfDigits(num);

console.log(isDudeneyNumber(512));
// Output: true

console.log(isDudeneyNumber(27));
// Output: false
```

230. Calculate the Area of a Regular Pentagon

Calculate the area of a regular pentagon using its side length.

```
const regularPentagonArea = (sideLength) => (1 / 4) *
Math.sqrt(5 * (5 + 2 * Math.sqrt(5))) * sideLength ** 2;

console.log(regularPentagonArea(5));
// Output: 43.01193501472417
```

231. Calculate the Volume of a Pyramid

Calculate the volume of a pyramid using its base area and height.

```
const pyramidVolume = (baseArea, height) => (1 / 3) *
baseArea * height;

console.log(pyramidVolume(25, 10));
// Output: 83.33333333333333
```

232. Check if a Number is a Wedderburn-Etherington Number

Check if a given number is a Wedderburn-Etherington number.

```
const isWedderburnEtheringtonNumber = (num) => {
    const primes = primeFactors(num);
    const factorials = primes.map((prime) =>
factorial(prime - 1));
    return factorials.reduce((product, factorial) =>
product * factorial, 1) === factorial(num);
  };

console.log(isWedderburnEtheringtonNumber(6));
// Output: true

console.log(isWedderburnEtheringtonNumber(12));
// Output: false
```

233. Calculate the Surface Area of a Cube

Calculate the surface area of a cube using its side length.

```
const cubeSurfaceArea = (sideLength) => 6 * sideLength **
2;

console.log(cubeSurfaceArea(5));
// Output: 150
```

234. Check if a Number is a Pluperfect Number

Check if a given number is a pluperfect number.

```
const isPluperfectNumber = (num) => num ===
sumOfDivisors(sumOfDivisors(num)) - num;

console.log(isPluperfectNumber(28));
// Output: true

console.log(isPluperfectNumber(20));
// Output: false
```

235. Calculate the Area of a Regular Octagon

Calculate the area of a regular octagon using its side length.

```
const regularOctagonArea = (sideLength) => 2 * (1 +
Math.sqrt(2)) * sideLength ** 2;

console.log(regularOctagonArea(5));
// Output: 86.60254037844387
```

236. Check if a Number is a Repunit Number

Check if a given number is a repunit number.

```
const isRepunitNumber = num => /^1+$/.test(num.toString());

console.log(isRepunitNumber(111)); // Output: true
console.log(isRepunitNumber(11));  // Output: false
```

237. Calculate the Volume of a Ellipsoid

Calculate the volume of an ellipsoid using its semi-axes lengths.

```
const ellipsoidVolume = (a, b, c) => (4 / 3) * Math.PI
* a * b * c;

console.log(ellipsoidVolume(5, 3, 2));
// Output: 125.66370614359172
```

238. Check if a String is a Valid URL (Alternative Approach)

Check if a given string is a valid URL using an alternative approach.

```
const isValidURLAlt = (url) => /^(ftp|http|https):\/\/[^
"]+$/.test(url);

console.log(isValidURLAlt("https://www.example.com"));
// Output: true

console.log(isValidURLAlt("invalid url"));
// Output: false
```

239. Check if a String is a Valid Tax Identification Number (TIN)

Check if a given string is a valid Tax Identification Number (TIN).

```
const isValidTIN = (tin) => /^[A-Z]{2}\d{6}[A-
Z\d]{2}$/.test(tin);

console.log(isValidTIN("AB123456CD"));
// Output: true

console.log(isValidTIN("invalid tin"));
// Output: false
```

240. Check if a String is a Valid ISBN (International Standard Book Number)

Check if a given string is a valid International Standard Book Number (ISBN).

```
const isValidISBN = (isbn) => /^(?:\d{9}[\dX]|(?:\d{3}-
){2}\d{1}[\dX])$/.test(isbn);

console.log(isValidISBN("123456789"));
// Output: true

console.log(isValidISBN("invalid isbn"));
// Output: false
```

241. Check if a String is a Valid IP Address

Check if a given string is a valid IP address.

```
const isValidIPAddress = (ip) => /^((25[0-5]|2[0-4]\d|[0-
1]?\d{1,2})\.){3}(25[0-5]|2[0-4]\d|[0-
1]?\d{1,2})$/.test(ip);

console.log(isValidIPAddress("192.168.1.1"));
// Output: true

console.log(isValidIPAddress("invalid ip"));
// Output: false
```

242. Reverse a String (Using Recursion)

Reverse a string using a recursive approach.

```
const reverseStringRecursive = str => str === '' ? '' :
reverseStringRecursive(str.substr(1)) + str.charAt(0);

console.log(reverseStringRecursive('hello'));
// Output: 'olleh'
```

243. Count the Occurrences of Each Element in an Array

Count the occurrences of each element in a given array and return the counts in an object.

```
const countOccurrences = arr => arr.reduce((acc, curr) =>
(acc[curr] = (acc[curr] || 0) + 1, acc), {});

console.log(countOccurrences([1, 2, 1, 3, 2, 4, 1]));
// Output: { '1': 3, '2': 2, '3': 1, '4': 1 }
```

244. Check if Two Arrays are Equal (Shallow Comparison)

Check if two arrays are equal through a shallow comparison of their elements.

```
const arraysAreEqual = (arr1, arr2) => arr1.length ===
arr2.length && arr1.every((val, index) => val ===
arr2[index]);

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

245. Find the Minimum Value in an Array

Find the minimum value in a given array of numbers.

```
const findMinValue = arr => Math.min(...arr);

console.log(findMinValue([2, 7, 1, 9, 4]));
// Output: 1
```

246. Flatten an Array of Nested Arrays (Using concat)

Flatten an array of nested arrays using the concat method.

```
const flattenArray = arr => [].concat(...arr);

console.log(flattenArray([[1, 2], [3, 4], [5, 6]]));
// Output: [1, 2, 3, 4, 5, 6]
```

247. Find the Average of Numbers in an Array

Calculate the average of numbers in a given array.

```
const findAverage = arr => arr.reduce((sum, num) => sum + num, 0) / arr.length;

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

248. Sum the Squares of Numbers in an Array

Calculate the sum of the squares of numbers in a given array.

```
const sumSquares = arr => arr.reduce((sum, num) => sum + num ** 2, 0);

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

249. Check if a String is a Palindrome (Ignoring Non-Alphanumeric Characters)

Check if a given string is a palindrome, ignoring non-alphanumeric characters and considering case-insensitivity.

```
const isPalindromeIgnoringNonAlphaNumeric = str => {
    const cleanedStr = str.replace(/[^a-zA-Z0-9]/g,
'').toLowerCase();
    return cleanedStr ===
cleanedStr.split('').reverse().join('');
  };

  console.log(isPalindromeIgnoringNonAlphaNumeric("A man, a
plan, a canal, Panama!"));
// Output: true
```

250. Shuffle an Array (Using Fisher-Yates Algorithm)

Shuffle the elements of an array using the Fisher-Yates shuffle algorithm.

```
const shuffleArrayFisherYates = arr => {
    for (let i = arr.length - 1; i > 0; i--) {
      const j = Math.floor(Math.random() * (i + 1));
      [arr[i], arr[j]] = [arr[j], arr[i]];
    }
    return arr;
  };

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

251. Sort an Array of Objects by a Specific Property

Sort an array of objects based on a specific property value.

```
const sortByProperty = (arr, prop) => arr.sort((a, b) => a[prop] - b[prop]);

const data = [{ name: 'Alice', age: 25 }, { name: 'Bob', age: 20 }, { name: 'Carol', age: 30 }];
console.log(sortByProperty(data, 'age'));
// Output: [{ name: 'Bob', age: 20 }, { name: 'Alice', age: 25 }, { name: 'Carol', age: 30 }]
```

252. Reverse Words in a Sentence

Reverse words in a sentence.

```
const originalSentence = "Hello world, how are you?";
const reversedSentence = reverseWords(originalSentence);
console.log(reversedSentence); // Output: "you? are how
world, Hello"
```

253. Find the Median of Numbers in an Array

Find the median value of numbers in a given array.

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

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

254. Count the Vowels in a String

Count the number of vowels in a given string.

```
const countVowels = str => (str.match(/[aeiou]/gi) ||
[]).length;

console.log(countVowels('Hello, how are you?'));
// Output: 7
```

255. Check if a Number is a Tribonacci Number (Alternative Approach)

Check if a given number is a Tribonacci number using an alternative approach.

```
const isTribonacciNumberAlt = num => [0, 0,
1].concat([...Array(num)]).slice(3).map((_, i, arr) =>
arr[i - 3] + arr[i - 2] + arr[i - 1]).includes(num);

console.log(isTribonacciNumberAlt(21));
// Output: true
```

256. Calculate the Fibonacci Sequence (Up to N Terms)

Generate the Fibonacci sequence up to a given number of terms n.

```
const fibonacciSequence = n =>
[...Array(n)].reduce((fibSeq, _, i) => fibSeq.concat(i > 1
? fibSeq[i - 1] + fibSeq[i - 2] : i), [0, 1]);

console.log(fibonacciSequence(10));
// Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

257. Find the ASCII Value of a Character

Get the ASCII value of a given character.

```
Const getAsciiValue = char => char.charCodeAt(0);

console.log(getAsciiValue('A'));
// Output: 65
```

258. Check if a String is an Isogram (No Repeating Characters)

Check if a given string is an isogram, meaning it has no repeating characters (case-insensitive).

```
const isIsogram = str => new Set(str.toLowerCase()).size
=== str.length;

console.log(isIsogram('hello'));
// Output: false
console.log(isIsogram('world'));
// Output: true
```

259. Calculate the Hamming Distance of Two Strings (Equal Length)

Calculate the Hamming distance between two equal-length strings, which is the count of differing characters at corresponding positions.

```
const hammingDistance = (str1, str2) =>
[...str1].reduce((distance, char, i) => distance + (char
!== str2[i]), 0);

console.log(hammingDistance('karolin', 'kathrin'));
// Output: 3
```

260. Calculate the Distance between Two Points in a 2D Plane

Calculate the Euclidean distance between two points in a 2D plane using the Pythagorean theorem.

```
const calculateDistance = ([x1, y1], [x2, y2]) =>
Math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2);

console.log(calculateDistance([0, 0], [3, 4]));
// Output: 5 (Pythagorean triple: 3^2 + 4^2 = 5^2)
```

261. Check if a String is a Positive Number (No Sign or Decimal Allowed)

Check if a given string represents a positive integer without any sign or decimal point.

```
const isPositiveNumber = str => /^[0-9]+$/.test(str);

console.log(isPositiveNumber('123'));
// Output: true
console.log(isPositiveNumber('-123'));
// Output: false
```

262. Find the First Non-Repeating Character in a String

Find and return the first non-repeating character in a given string.

```
const findFirstNonRepeating = str => [...str].find(char =>
str.indexOf(char) === str.lastIndexOf(char));

console.log(findFirstNonRepeating('hello'));
// Output: 'h'
```

263. Calculate the Area of a Kite

Calculate the area of a kite using the lengths of its diagonals.

```
const areaOfKite = (d1, d2) => 0.5 * d1 * d2;

console.log("Area of the kite:", areaOfKite(10, 6));
// Output: 30
```

264. Calculate the Area of a Sector

Calculate the area of a sector within a circle based on the provided radius and angle.

```
const sectorArea = (radius, angle) => (Math.PI * radius **
2 * angle) / 360;

console.log(sectorArea(5, 60));
// Output: 5.235987755982989
```

In every line of code, they have woven a story of innovation and creativity. This book has been your compass in the vast world of JavaScript.

Close this chapter knowing that every challenge overcome is an achievement, and every solution is a step toward mastery.

Your code is the melody that gives life to projects. May they continue creating and programming with passion!

Thank you for allowing me to be part of your journey.

With gratitude,

Hernando Abella

250+ Killer JavaScript One-Liners

Discover Other Useful Resources at:

www.hernandoabella.com