### 1. ****Reverse a String****

function reverseString(str) {

return str.split('').reverse().join('');

}

console.log(reverseString("JavaScript")); *// "tpircSavaJ"*

### 2. ****Check if a Number is Even or Odd****

function isEven(num) {

return num % 2 === 0 ? "Even" : "Odd";

}

console.log(isEven(0)); *// "Even"*

console.log(isEven(-3)); *// "Odd"*

### 3. ****Find the Largest Number in an Array****

function findMax(arr) {

return Math.max(...arr);

}

console.log(findMax([10, -5, 100, 23])); *// 100*

### 4. ****Remove Duplicates from an Array****

function removeDuplicates(arr) {

return [...new Set(arr)];

}

console.log(removeDuplicates([1, 1, "a", "a", 2])); *// [1, "a", 2]*

### 5. ****Factorial of a Number****

function factorial(n) {

return n === 0 ? 1 : n \* factorial(n - 1);

}

console.log(factorial(3)); *// 6 (3 \* 2 \* 1)*

console.log(factorial(0)); *// 1*

### 6. ****Check if a String is a Palindrome****

function isPalindrome(str) {

const reversed = str.split('').reverse().join('');

return str === reversed;

}

console.log(isPalindrome("radar")); *// true*

console.log(isPalindrome("code")); *// false*

### 7. ****Sum of Array Elements****

function arraySum(arr) {

return arr.reduce((sum, num) => sum + num, 0);

}

console.log(arraySum([5, -2, 10])); *// 13*

console.log(arraySum([])); *// 0*

### 8. ****Fibonacci Sequence****

Another recursive function. Fibonacci numbers are the sum of the two preceding ones.

function fibonacci(n) {

if (n <= 1) return n;

return fibonacci(n - 1) + fibonacci(n - 2);

}

console.log(fibonacci(3)); *// 2 (0, 1, 1, 2)*

console.log(fibonacci(7)); *// 13 (0, 1, 1, 2, 3, 5, 8, 13)*

### 9. ****Debounce Function****

(Debouncing ensures a function isn’t called too frequently (e.g., during rapid user input)

function debounce(func, delay) {

let timeout;

return function (...args) {

clearTimeout(timeout);

timeout = setTimeout(() => func(...args), delay);

};

}

const logHi = debounce(() => console.log("Hi"), 1000);

logHi(); *// After 1 second: "Hi"*

logHi(); *// Resets timer, still only one "Hi" after 1 second*

### 10. ****Flatten an Array****

function flattenArray(arr) {

return arr.flat(Infinity);

}

console.log(flattenArray([1, [2, [3, 4], 5]])); *// [1, 2, 3, 4, 5]*

console.log(flattenArray([[1, 2], 3])); *// [1, 2, 3]*

#### 11. **Count Vowels in a String**

**Question:** Write a function to count the number of vowels (a, e, i, o, u) in a string.

**Answer:**

javascript

function countVowels(str) {

const matches = str.toLowerCase().match(/[aeiou]/g);

return matches ? matches.length : 0;

}

**Explanation:**

* toLowerCase() ensures case insensitivity.
* match(/[aeiou]/g) returns an array of vowels (or null if none found).
* g flag makes it match globally (all occurrences). **Edge Cases & Examples:**

javascript

console.log(countVowels("HELLO")); *// 2*

console.log(countVowels("xyz")); *// 0*

console.log(countVowels("")); *// 0*

#### 12. **FizzBuzz**

**Question:** Write a function that prints numbers from 1 to n, replacing multiples of 3 with "Fizz", multiples of 5 with "Buzz", and multiples of both with "FizzBuzz". **Answer:**

javascript

function fizzBuzz(n) {

for (let i = 1; i <= n; i++) {

if (i % 3 === 0 && i % 5 === 0) console.log("FizzBuzz");

else if (i % 3 === 0) console.log("Fizz");

else if (i % 5 === 0) console.log("Buzz");

else console.log(i);

}

}

**Explanation:**

* Simple conditionals check divisibility using %. **Additional Example:**

javascript

fizzBuzz(5); *// 1, 2, "Fizz", 4, "Buzz"*

#### 13. **Swap Two Variables Without a Temp Variable**

**Question:** Swap two variables without using a third variable. **Answer:**

javascript

function swap(a, b) {

[a, b] = [b, a];

return [a, b];

}

**Alternative (Using Arithmetic):**

javascript

function swap(a, b) {

a = a + b;

b = a - b;

a = a - b;

return [a, b];

}

**Explanation:**

* Destructuring assignment ([a, b] = [b, a]) is clean and modern.
* Arithmetic method works but risks overflow with large numbers. **Additional Example:**

javascript

console.log(swap(5, 10)); *// [10, 5]*

#### 14. **Find the First Non-Repeating Character**

**Question:** Write a function to find the first character in a string that doesn’t repeat. **Answer:**

javascript

function firstNonRepeatingChar(str) {

for (let char of str) {

if (str.indexOf(char) === str.lastIndexOf(char)) return char;

}

return null;

}

**Explanation:**

* indexOf finds the first occurrence, lastIndexOf finds the last. If they’re equal, the character appears only once. **Edge Cases & Examples:**

javascript

console.log(firstNonRepeatingChar("leetcode")); *// "l"*

console.log(firstNonRepeatingChar("aabb")); *// null*

### Edge Case Handling Tips

* **Input Validation:** Always check types (e.g., typeof, Array.isArray) or edge values (e.g., null, undefined).
* **Performance:** Recursive solutions (like factorial) can stack overflow with large inputs—mention iterative alternatives.
* **Browser Compatibility:** Some methods (e.g., flat()) are ES2019+, so know polyfills or alternatives.

Let me know if you want deeper dives into any of these, more questions, or help with specific scenarios!

### 15. ****Check if Two Strings are Anagrams****

**Question:** Write a function to check if two strings are anagrams (contain the same characters with the same frequency, ignoring order).

**Answer:**

javascript

function areAnagrams(str1, str2) {

const sortStr = str => str.toLowerCase().split('').sort().join('');

return sortStr(str1) === sortStr(str2);

}

**Explanation:**

* Convert both strings to lowercase to ignore case.
* split(''), sort(), and join('') rearrange characters alphabetically.
* Compare the sorted strings for equality. **Examples:**

javascript

console.log(areAnagrams("listen", "silent")); *// true*

console.log(areAnagrams("hello", "world")); *// false*

console.log(areAnagrams("", "")); *// true*

### 16. ****Find the Missing Number in an Array****

**Question:** Given an array of numbers from 1 to n with one number missing, find the missing number.

**Answer:**

javascript

function findMissingNumber(arr, n) {

const expectedSum = (n \* (n + 1)) / 2;

const actualSum = arr.reduce((sum, num) => sum + num, 0);

return expectedSum - actualSum;

}

**Explanation:**

* Use the formula for the sum of numbers from 1 to n: (n \* (n + 1)) / 2.
* Subtract the actual sum of the array to find the missing number. **Examples:**

javascript

console.log(findMissingNumber([1, 2, 4, 5], 5)); *// 3*

console.log(findMissingNumber([1, 2, 3], 4)); *// 4*

### 17. ****Rotate an Array to the Right****

**Question:** Write a function to rotate an array to the right by k steps.

**Answer:**

javascript

function rotateArray(arr, k) {

k = k % arr.length; *// Handle k > array length*

return [...arr.slice(-k), ...arr.slice(0, -k)];

}

**Explanation:**

* k % arr.length ensures k doesn’t exceed array length.
* slice(-k) takes the last k elements, slice(0, -k) takes the rest, and spread combines them. **Examples:**

javascript

console.log(rotateArray([1, 2, 3, 4, 5], 2)); *// [4, 5, 1, 2, 3]*

console.log(rotateArray([1, 2], 3)); *// [2, 1]*

### 18. ****Capitalize First Letter of Each Word****

**Question:** Write a function to capitalize the first letter of each word in a string.

**Answer:**

javascript

function capitalizeWords(str) {

return str

.split(' ')

.map(word => word.charAt(0).toUpperCase() + word.slice(1))

.join(' ');

}

**Explanation:**

* split(' ') breaks the string into words.
* map() transforms each word: charAt(0).toUpperCase() capitalizes the first letter, slice(1) keeps the rest.
* join(' ') reassembles the string. **Examples:**

javascript

console.log(capitalizeWords("hello world")); *// "Hello World"*

console.log(capitalizeWords("javascript is fun")); *// "Javascript Is Fun"*

### 19. ****Merge Two Sorted Arrays****

**Question:** Write a function to merge two sorted arrays into one sorted array.

**Answer:**

javascript

function mergeSortedArrays(arr1, arr2) {

return [...arr1, ...arr2].sort((a, b) => a - b);

}

**Explanation:**

* Spread operator combines both arrays.
* sort((a, b) => a - b) sorts numerically (default sort is lexicographical). **Alternative (Manual Merge):**

javascript

function mergeSortedArrays(arr1, arr2) {

const result = [];

let i = 0, j = 0;

while (i < arr1.length && j < arr2.length) {

result.push(arr1[i] < arr2[j] ? arr1[i++] : arr2[j++]);

}

return result.concat(arr1.slice(i), arr2.slice(j));

}

**Examples:**

javascript

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

console.log(mergeSortedArrays([1], [2, 3])); *// [1, 2, 3]*

### 20. ****Check if a Number is Prime****

**Question:** Write a function to check if a number is prime.

**Answer:**

javascript

function isPrime(num) {

if (num < 2) return false;

for (let i = 2; i <= Math.sqrt(num); i++) {

if (num % i === 0) return false;

}

return true;

}

**Explanation:**

* Numbers less than 2 aren’t prime.
* Check divisibility up to the square root (optimization).
* If no divisors are found, it’s prime. **Examples:**

javascript

console.log(isPrime(7)); *// true*

console.log(isPrime(12)); *// false*

console.log(isPrime(1)); *// false*

### 21. ****Reverse Words in a String****

**Question:** Write a function to reverse the order of words in a string.

**Answer:**

javascript

function reverseWords(str) {

return str.split(' ').reverse().join(' ');

}

**Explanation:**

* split(' ') creates an array of words.
* reverse() reverses the array.
* join(' ') recombines with spaces. **Examples:**

javascript

console.log(reverseWords("Hello World")); *// "World Hello"*

console.log(reverseWords("JavaScript is cool")); *// "cool is JavaScript"*

### 22. ****Find the Second Largest Number in an Array****

**Question:** Write a function to find the second largest number in an array.

**Answer:**

javascript

function secondLargest(arr) {

const unique = [...new Set(arr)].sort((a, b) => b - a);

return unique.length < 2 ? null : unique[1];

}

**Explanation:**

* Remove duplicates with Set, sort descending with sort((a, b) => b - a).
* Return the second element (index 1) if it exists. **Examples:**

javascript

console.log(secondLargest([5, 1, 9, 6, 3])); *// 6*

console.log(secondLargest([1])); *// null*

### 23. ****Throttle Function****

**Question:** Write a throttle function to limit how often a function can be called.

**Answer:**

javascript

function throttle(func, limit) {

let inThrottle;

return function (...args) {

if (!inThrottle) {

func(...args);

inThrottle = true;

setTimeout(() => (inThrottle = false), limit);

}

};

}

**Explanation:**

* inThrottle tracks if the function is in a cooldown period.
* Only calls func if not throttled, then sets a timeout to reset. **Example:**

javascript

const throttledLog = throttle(() => console.log("Throttled!"), 1000);

throttledLog(); *// "Throttled!"*

throttledLog(); *// Ignored if called within 1 second*

### 24. ****Deep Clone an Object****

**Question:** Write a function to create a deep clone of an object.

**Answer:**

javascript

function deepClone(obj) {

return JSON.parse(JSON.stringify(obj));

}

**Explanation:**

* JSON.stringify converts the object to a string, JSON.parse converts it back, breaking references.
* Note: This doesn’t handle functions or undefined values. **Examples:**

javascript

const obj = { a: 1, b: { c: 2 } };

const clone = deepClone(obj);

clone.b.c = 3;

console.log(obj.b.c); *// 2 (original unchanged)*

console.log(clone.b.c); *// 3*

These questions test a mix of problem-solving, algorithmic thinking, and JavaScript-specific knowledge. Let me know if you want more examples, alternative solutions, or explanations for any of these!

### 25. ****Find the Longest Word in a String****

**Question:** Write a function to find the longest word in a string.

**Answer:**

javascript

function longestWord(str) {

return str.split(' ').reduce((longest, current) =>

current.length > longest.length ? current : longest, '');

}

**Explanation:**

* split(' ') breaks the string into an array of words.
* reduce() compares each word’s length, keeping the longest. **Examples:**

javascript

console.log(longestWord("I love JavaScript")); *// "JavaScript"*

console.log(longestWord("Hello world")); *// "Hello"*

console.log(longestWord("")); *// ""*

### 26. ****Count Character Occurrences****

**Question:** Write a function to count the occurrences of each character in a string.

**Answer:**

javascript

function charCount(str) {

return str.split('').reduce((count, char) => {

count[char] = (count[char] || 0) + 1;

return count;

}, {});

}

**Explanation:**

* split('') converts the string to an array of characters.
* reduce() builds an object where keys are characters and values are their counts. **Examples:**

javascript

console.log(charCount("hello")); *// { h: 1, e: 1, l: 2, o: 1 }*

console.log(charCount("aaa")); *// { a: 3 }*

### 27. ****Generate a Range of Numbers****

**Question:** Write a function to generate an array of numbers from start to end (inclusive).

**Answer:**

javascript

function range(start, end) {

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

}

**Explanation:**

* Array.from creates an array of specified length.
* The mapping function (\_, i) => start + i fills it with numbers from start to end. **Examples:**

javascript

console.log(range(1, 5)); *// [1, 2, 3, 4, 5]*

console.log(range(0, 2)); *// [0, 1, 2]*

### 28. ****Check if an Object is Empty****

**Question:** Write a function to check if an object has no properties.

**Answer:**

javascript

function isEmpty(obj) {

return Object.keys(obj).length === 0;

}

**Explanation:**

* Object.keys(obj) returns an array of the object’s enumerable properties.
* If the length is 0, the object is empty. **Examples:**

javascript

console.log(isEmpty({})); *// true*

console.log(isEmpty({ a: 1 })); *// false*

console.log(isEmpty({ key: null })); *// false*

### 29. ****Implement a Simple Promise****

**Question:** Write a function that returns a promise resolving after a delay.

**Answer:**

javascript

function delay(ms) {

return new Promise(resolve => setTimeout(() => resolve("Done"), ms));

}

**Explanation:**

* new Promise creates a promise that resolves with "Done" after ms milliseconds using setTimeout. **Examples:**

javascript

delay(1000).then(result => console.log(result)); *// "Done" after 1 second*

delay(500).then(result => console.log(result)); *// "Done" after 0.5 seconds*

### 30. ****Find Intersection of Two Arrays****

**Question:** Write a function to find common elements between two arrays.

**Answer:**

javascript

function intersection(arr1, arr2) {

const set = new Set(arr1);

return arr2.filter(item => set.has(item));

}

**Explanation:**

* Convert arr1 to a Set for O(1) lookup.
* filter keeps only elements from arr2 that exist in the set. **Examples:**

javascript

console.log(intersection([1, 2, 3], [2, 3, 4])); *// [2, 3]*

console.log(intersection([1, 2], [3, 4])); *// []*

### 31. ****Remove Falsy Values from an Array****

**Question:** Write a function to remove all falsy values (false, null, undefined, 0, "", NaN) from an array.

**Answer:**

javascript

function removeFalsy(arr) {

return arr.filter(Boolean);

}

**Explanation:**

* Boolean converts each value to true/false; falsy values are filtered out. **Examples:**

javascript

console.log(removeFalsy([0, 1, false, 2, "", 3])); *// [1, 2, 3]*

console.log(removeFalsy([null, undefined, NaN])); *// []*

### 32. ****Sum of Digits****

**Question:** Write a function to calculate the sum of digits in a number.

**Answer:**

javascript

function sumOfDigits(num) {

return String(num).split('').reduce((sum, digit) => sum + Number(digit), 0);

}

**Explanation:**

* Convert the number to a string, split into digits, and sum them using reduce. **Examples:**

javascript

console.log(sumOfDigits(123)); *// 6 (1 + 2 + 3)*

console.log(sumOfDigits(50)); *// 5 (5 + 0)*

### 33. ****Create a Counter Using Closure****

**Question:** Write a function that returns a counter that increments each time it’s called.

**Answer:**

javascript

function createCounter() {

let count = 0;

return function () {

return ++count;

};

}

**Explanation:**

* Closure keeps count in scope, and the returned function increments it. **Examples:**

javascript

const counter = createCounter();

console.log(counter()); *// 1*

console.log(counter()); *// 2*

console.log(counter()); *// 3*

### 34. ****Truncate a String****

**Question:** Write a function to truncate a string to a specified length and add "..." if it’s longer.

**Answer:**

javascript

function truncate(str, max) {

return str.length > max ? str.slice(0, max) + "..." : str;

}

**Explanation:**

* If the string exceeds max, slice it and append "..."; otherwise, return it as is. **Examples:**

javascript

console.log(truncate("Hello World", 5)); *// "Hello..."*

console.log(truncate("Hi", 5)); *// "Hi"*

### 35. ****Find Pairs with a Given Sum****

**Question:** Write a function to find all pairs in an array that sum to a given target.

**Answer:**

javascript

function findPairs(arr, target) {

const seen = new Set();

const pairs = [];

for (let num of arr) {

const complement = target - num;

if (seen.has(complement)) pairs.push([complement, num]);

seen.add(num);

}

return pairs;

}

**Explanation:**

* Use a Set to track seen numbers.
* For each number, check if its complement (target - num) exists; if so, it’s a pair. **Examples:**

javascript

console.log(findPairs([1, 2, 3, 4], 5)); *// [[2, 3], [1, 4]]*

console.log(findPairs([1, 1, 2], 3)); *// [[1, 2]]*

These questions test a range of skills: string/array manipulation, object handling, closures, and basic algorithms. They’re concise yet impactful in interviews. Want more examples, alternatives, or deeper explanations for any of these?

### 36. ****Flatten an Object****

**Question:** Write a function to flatten a nested object into a single-level object with dot notation keys.

**Answer:**

javascript

function flattenObject(obj, prefix = '') {

return Object.keys(obj).reduce((acc, key) => {

const newKey = prefix ? `${prefix}.${key}` : key;

return typeof obj[key] === 'object' && obj[key] !== null

? { ...acc, ...flattenObject(obj[key], newKey) }

: { ...acc, [newKey]: obj[key] };

}, {});

}

**Explanation:**

* Recursively traverse the object.
* Build keys with dot notation (e.g., a.b.c) for nested properties.
* Spread operator merges results into a flat object. **Examples:**

javascript

console.log(flattenObject({ a: 1, b: { c: 2, d: 3 } })); *// { a: 1, 'b.c': 2, 'b.d': 3 }*

console.log(flattenObject({ x: { y: { z: 1 } } })); *// { 'x.y.z': 1 }*

### 37. ****Check if a String Contains Only Digits****

**Question:** Write a function to check if a string contains only digits.

**Answer:**

javascript

function isDigitsOnly(str) {

return /^\d+$/.test(str);

}

**Explanation:**

* Regular expression /^\d+$/:
  + ^ asserts start, \d+ matches one or more digits, $ asserts end.
* test() returns true if the string matches. **Examples:**

javascript

console.log(isDigitsOnly("12345")); *// true*

console.log(isDigitsOnly("12a34")); *// false*

console.log(isDigitsOnly("")); *// false*

### 38. ****Find the Most Frequent Element in an Array****

**Question:** Write a function to find the element that appears most often in an array.

**Answer:**

javascript

function mostFrequent(arr) {

const freq = arr.reduce((acc, val) => {

acc[val] = (acc[val] || 0) + 1;

return acc;

}, {});

return Object.keys(freq).reduce((a, b) => freq[a] > freq[b] ? a : b);

}

**Explanation:**

* First reduce builds a frequency map (e.g., { '1': 2, '2': 3 }).
* Second reduce finds the key with the highest value. **Examples:**

javascript

console.log(mostFrequent([1, 2, 2, 3, 2])); *// "2"*

console.log(mostFrequent(["a", "b", "a"])); *// "a"*

### 39. ****Convert String to Camel Case****

**Question:** Write a function to convert a string with spaces or hyphens to camelCase.

**Answer:**

javascript

function toCamelCase(str) {

return str

.split(/[\s-]+/)

.map((word, i) => i === 0 ? word.toLowerCase() : word.charAt(0).toUpperCase() + word.slice(1).toLowerCase())

.join('');

}

**Explanation:**

* split(/[\s-]+/) splits on spaces or hyphens.
* map keeps the first word lowercase, capitalizes others’ first letters. **Examples:**

javascript

console.log(toCamelCase("hello world")); *// "helloWorld"*

console.log(toCamelCase("user-id-name")); *// "userIdName"*

### 40. ****Binary Search Implementation****

**Question:** Write a function to perform binary search on a sorted array.

**Answer:**

javascript

function binarySearch(arr, target) {

let left = 0, right = arr.length - 1;

while (left <= right) {

const mid = Math.floor((left + right) / 2);

if (arr[mid] === target) return mid;

arr[mid] > target ? right = mid - 1 : left = mid + 1;

}

return -1;

}

**Explanation:**

* Works on sorted arrays, repeatedly dividing the search range in half.
* Returns the index of the target or -1 if not found. **Examples:**

javascript

console.log(binarySearch([1, 2, 3, 4, 5], 3)); *// 2*

console.log(binarySearch([1, 2, 4], 3)); *// -1*

### 41. ****Remove Specific Element from Array****

**Question:** Write a function to remove all occurrences of a specific value from an array.

**Answer:**

javascript

function removeElement(arr, val) {

return arr.filter(item => item !== val);

}

**Explanation:**

* filter creates a new array excluding the specified value. **Examples:**

javascript

console.log(removeElement([1, 2, 3, 2, 4], 2)); *// [1, 3, 4]*

console.log(removeElement([1, 1, 1], 1)); *// []*

### 42. ****Generate Random String****

**Question:** Write a function to generate a random string of specified length.

**Answer:**

javascript

function randomString(length) {

const chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';

return Array.from({ length }, () => chars.charAt(Math.floor(Math.random() \* chars.length))).join('');

}

**Explanation:**

* Array.from creates an array of length, mapping each to a random character from chars. **Examples:**

javascript

console.log(randomString(5)); *// e.g., "Kj9pM"*

console.log(randomString(3)); *// e.g., "x7q"*

### 43. ****Check if Array is Sorted****

**Question:** Write a function to check if an array is sorted in ascending order.

**Answer:**

javascript

function isSorted(arr) {

return arr.every((val, i) => i === 0 || val >= arr[i - 1]);

}

**Explanation:**

* every checks if each element is greater than or equal to the previous one. **Examples:**

javascript

console.log(isSorted([1, 2, 3, 4])); *// true*

console.log(isSorted([1, 3, 2])); *// false*

### 44. ****Memoize a Function****

**Question:** Write a function to memoize another function’s results.

**Answer:**

javascript

function memoize(fn) {

const cache = {};

return function (...args) {

const key = JSON.stringify(args);

return cache[key] ||= fn(...args);

};

}

**Explanation:**

* Uses a cache to store results based on arguments.
* ||= assigns and returns the result if not already cached. **Examples:**

javascript

const add = memoize((a, b) => a + b);

console.log(add(2, 3)); *// 5 (computed)*

console.log(add(2, 3)); *// 5 (cached)*

### 45. ****Sum All Numbers in a Mixed Array****

**Question:** Write a function to sum all numbers in an array, ignoring non-numeric values.

**Answer:**

javascript

function sumNumbers(arr) {

return arr.reduce((sum, val) => typeof val === 'number' && !isNaN(val) ? sum + val : sum, 0);

}

**Explanation:**

* reduce adds only numeric values, skipping NaN and non-numbers. **Examples:**

javascript

console.log(sumNumbers([1, "a", 2, null, 3])); *// 6*

console.log(sumNumbers([NaN, 1, "2"])); *// 1*

These questions push into more advanced territory—recursion, regex, optimization (e.g., memoization), and algorithmic thinking (e.g., binary search). They’re great for mid-to-senior-level interviews. Let me know if you’d like more examples, edge cases, or further clarification!

### 46. ****Power Function****

**Question:** Write a function to calculate x raised to the power n (xⁿ).

**Answer:**

javascript

function power(x, n) {

if (n === 0) return 1;

return x \* power(x, n - 1);

}

**Explanation:**

* Recursive approach: xⁿ = x \* xⁿ⁻¹.
* Base case: x⁰ = 1. **Optimized Alternative (Exponentiation by Squaring):**

javascript

function power(x, n) {

if (n === 0) return 1;

const half = power(x, Math.floor(n / 2));

return n % 2 === 0 ? half \* half : x \* half \* half;

}

**Examples:**

javascript

console.log(power(2, 3)); *// 8 (2 \* 2 \* 2)*

console.log(power(3, 2)); *// 9 (3 \* 3)*

### 47. ****Group Anagrams****

**Question:** Write a function to group an array of strings into anagrams.

**Answer:**

javascript

function groupAnagrams(arr) {

const map = {};

for (let str of arr) {

const key = str.split('').sort().join('');

map[key] = map[key] || [];

map[key].push(str);

}

return Object.values(map);

}

**Explanation:**

* Sort each string’s characters to create a key (anagrams sort to the same key).
* Group strings by key in a map, then return the values. **Examples:**

javascript

console.log(groupAnagrams(["eat", "tea", "tan", "ate", "nat", "bat"]));

*// [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]*

console.log(groupAnagrams([""])); *// [[""]]*

### 48. ****Reverse a Number****

**Question:** Write a function to reverse a number (e.g., 123 → 321).

**Answer:**

javascript

function reverseNumber(num) {

return Number(String(num).split('').reverse().join(''));

}

**Explanation:**

* Convert to string, reverse the digits, and convert back to a number. **Examples:**

javascript

console.log(reverseNumber(123)); *// 321*

console.log(reverseNumber(1000)); *// 1 (leading zeros dropped)*

### 49. ****Implement Array.map****

**Question:** Write a custom implementation of the map array method.

**Answer:**

javascript

function customMap(arr, callback) {

const result = [];

for (let i = 0; i < arr.length; i++) {

result.push(callback(arr[i], i, arr));

}

return result;

}

**Explanation:**

* Iterate over the array, apply the callback to each element (with index and array as args), and collect results. **Examples:**

javascript

console.log(customMap([1, 2, 3], x => x \* 2)); *// [2, 4, 6]*

console.log(customMap(["a", "b"], (x, i) => x + i)); *// ["a0", "b1"]*

### 50. ****Longest Common Prefix****

**Question:** Write a function to find the longest common prefix among an array of strings.

**Answer:**

javascript

function longestCommonPrefix(strs) {

if (!strs.length) return "";

return strs.reduce((prefix, str) => {

while (str.indexOf(prefix) !== 0) prefix = prefix.slice(0, -1);

return prefix;

}, strs[0]);

}

**Explanation:**

* Start with the first string as the prefix.
* Reduce it by comparing with each string, trimming until it matches the start. **Examples:**

javascript

console.log(longestCommonPrefix(["flower", "flow", "flight"])); *// "fl"*

console.log(longestCommonPrefix(["dog", "cat"])); *// ""*

### 51. ****Check Balanced Parentheses****

**Question:** Write a function to check if a string has balanced parentheses.

**Answer:**

javascript

function isBalanced(str) {

let stack = [];

for (let char of str) {

if (char === '(') stack.push(char);

else if (char === ')') {

if (!stack.pop()) return false;

}

}

return stack.length === 0;

}

**Explanation:**

* Use a stack: push opening ( and pop on closing ).
* Unmatched ) or leftover ( means unbalanced. **Examples:**

javascript

console.log(isBalanced("()")); *// true*

console.log(isBalanced("(())")); *// true*

console.log(isBalanced("(()")); *// false*

### 52. ****Get Unique Values from Two Arrays****

**Question:** Write a function to get unique values present in either of two arrays (union).

**Answer:**

javascript

function union(arr1, arr2) {

return [...new Set([...arr1, ...arr2])];

}

**Explanation:**

* Combine arrays with spread, then use Set to remove duplicates. **Examples:**

javascript

console.log(union([1, 2, 3], [2, 3, 4])); *// [1, 2, 3, 4]*

console.log(union([1, 1], [2, 2])); *// [1, 2]*

### 53. ****Convert Object to Query String****

**Question:** Write a function to convert an object into a URL query string.

**Answer:**

javascript

function toQueryString(obj) {

return Object.entries(obj)

.map(([key, val]) => `${key}=${val}`)

.join('&');

}

**Explanation:**

* Object.entries gets key-value pairs, map formats them as key=value, join combines with &. **Examples:**

javascript

console.log(toQueryString({ name: "John", age: 30 })); *// "name=John&age=30"*

console.log(toQueryString({ id: 1 })); *// "id=1"*

### 54. ****Find Minimum Steps to One****

**Question:** Write a function to find the minimum steps to reduce a number to 1 by dividing by 2 (if even) or subtracting 1.

**Answer:**

javascript

function minSteps(n) {

if (n === 1) return 0;

if (n % 2 === 0) return 1 + minSteps(n / 2);

return 1 + Math.min(minSteps(n - 1), minSteps(n + 1));

}

**Explanation:**

* Recursive dynamic problem: even numbers divide by 2, odd numbers choose min of n-1 or n+1. **Examples:**

javascript

console.log(minSteps(4)); *// 2 (4 → 2 → 1)*

console.log(minSteps(7)); *// 4 (7 → 8 → 4 → 2 → 1)*

### 55. ****Simple Event Emitter****

**Question:** Write a simple event emitter with on and emit methods.

**Answer:**

javascript

function EventEmitter() {

this.events = {};

this.on = (event, fn) => (this.events[event] = this.events[event] || []).push(fn);

this.emit = (event, ...args) => this.events[event]?.forEach(fn => fn(...args));

}

**Explanation:**

* on adds listeners to an event array.
* emit calls all listeners for an event with arguments. **Examples:**

javascript

const emitter = new EventEmitter();

emitter.on('greet', name => console.log(`Hello, ${name}!`));

emitter.emit('greet', 'John'); *// "Hello, John!"*

These questions span recursion, dynamic programming, custom implementations, and design patterns—great for testing advanced JavaScript skills. Let me know if you want more examples, edge cases, or deeper dives into any of these!

### 56. ****Find the Shortest Word in a String****

**Question:** Write a function to find the shortest word in a string.

**Answer:**

javascript

function shortestWord(str) {

return str.split(' ').reduce((shortest, current) =>

current.length < shortest.length ? current : shortest, str.split(' ')[0]);

}

**Explanation:**

* Split the string into words and use reduce to compare lengths, starting with the first word. **Examples:**

javascript

console.log(shortestWord("The quick brown fox")); *// "The"*

console.log(shortestWord("I am here")); *// "I"*

### 57. ****Check if a String is a Subsequence****

**Question:** Write a function to check if one string is a subsequence of another (characters appear in order, not necessarily contiguous).

**Answer:**

javascript

function isSubsequence(sub, str) {

let i = 0;

for (let char of str) {

if (char === sub[i]) i++;

if (i === sub.length) return true;

}

return i === sub.length;

}

**Explanation:**

* Iterate through str, incrementing i when characters match sub.
* If i reaches sub.length, all characters were found in order. **Examples:**

javascript

console.log(isSubsequence("abc", "ahbgdc")); *// true*

console.log(isSubsequence("axc", "ahbgdc")); *// false*

### 58. ****Move Zeros to End****

**Question:** Write a function to move all zeros to the end of an array while maintaining the order of non-zero elements.

**Answer:**

javascript

function moveZeros(arr) {

const nonZeros = arr.filter(x => x !== 0);

return [...nonZeros, ...Array(arr.length - nonZeros.length).fill(0)];

}

**Explanation:**

* Filter out zeros, then append the correct number of zeros to match the original length. **Examples:**

javascript

console.log(moveZeros([0, 1, 0, 3, 12])); *// [1, 3, 12, 0, 0]*

console.log(moveZeros([1, 2, 3])); *// [1, 2, 3]*

### 59. ****Convert Roman Numerals to Integer****

**Question:** Write a function to convert a Roman numeral string to an integer.

**Answer:**

javascript

function romanToInt(s) {

const values = { I: 1, V: 5, X: 10, L: 50, C: 100, D: 500, M: 1000 };

return s.split('').reduce((sum, char, i) =>

values[char] < values[s[i + 1]] ? sum - values[char] : sum + values[char], 0);

}

**Explanation:**

* Map Roman numerals to values.
* Subtract if a smaller value precedes a larger one (e.g., IV = 5 - 1); otherwise, add. **Examples:**

javascript

console.log(romanToInt("III")); *// 3*

console.log(romanToInt("IV")); *// 4*

console.log(romanToInt("MCMXCIV")); *// 1994*

### 60. ****Generate All Subsets of an Array****

**Question:** Write a function to generate all possible subsets (power set) of an array.

**Answer:**

javascript

function subsets(arr) {

return arr.reduce((subsets, val) =>

subsets.concat(subsets.map(set => [...set, val])), [[]]);

}

**Explanation:**

* Start with an empty subset [[]].
* For each element, duplicate existing subsets and add the element to the duplicates. **Examples:**

javascript

console.log(subsets([1, 2])); *// [[], [2], [1], [1, 2]]*

console.log(subsets([1])); *// [[], [1]]*

### 61. ****Check if Two Objects are Equal****

**Question:** Write a function to check if two objects are deeply equal.

**Answer:**

javascript

function areEqual(obj1, obj2) {

if (obj1 === obj2) return true;

if (typeof obj1 !== 'object' || typeof obj2 !== 'object' || !obj1 || !obj2) return false;

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

return keys1.length === keys2.length && keys1.every(key => areEqual(obj1[key], obj2[key]));

}

**Explanation:**

* Recursively compare properties; handle edge cases like null or non-objects. **Examples:**

javascript

console.log(areEqual({ a: 1, b: { c: 2 } }, { a: 1, b: { c: 2 } })); *// true*

console.log(areEqual({ a: 1 }, { a: 2 })); *// false*

### 62. ****Implement Array.filter****

**Question:** Write a custom implementation of the filter array method.

**Answer:**

javascript

function customFilter(arr, callback) {

const result = [];

for (let i = 0; i < arr.length; i++) {

if (callback(arr[i], i, arr)) result.push(arr[i]);

}

return result;

}

**Explanation:**

* Iterate and push elements to a new array if the callback returns true. **Examples:**

javascript

console.log(customFilter([1, 2, 3, 4], x => x > 2)); *// [3, 4]*

console.log(customFilter(["a", "ab"], x => x.length === 1)); *// ["a"]*

### 63. ****Pascal’s Triangle Row****

**Question:** Write a function to generate the nth row of Pascal’s Triangle (0-indexed).

**Answer:**

javascript

function pascalRow(n) {

let row = [1];

for (let i = 0; i < n; i++) {

row = [1, ...row.slice(0, -1).map((val, j) => val + row[j + 1]), 1];

}

return row;

}

**Explanation:**

* Start with [1], then build each row by summing adjacent elements from the previous row. **Examples:**

javascript

console.log(pascalRow(0)); *// [1]*

console.log(pascalRow(2)); *// [1, 2, 1]*

console.log(pascalRow(3)); *// [1, 3, 3, 1]*

### 64. ****Shuffle an Array (Fisher-Yates)****

**Question:** Write a function to shuffle an array randomly.

**Answer:**

javascript

function shuffle(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;

}

**Explanation:**

* Fisher-Yates algorithm: swap each element with a random earlier element. **Examples:**

javascript

console.log(shuffle([1, 2, 3])); *// e.g., [3, 1, 2]*

console.log(shuffle([1])); *// [1]*

### 65. ****Find Longest Consecutive Sequence****

**Question:** Write a function to find the length of the longest consecutive sequence in an unsorted array.

**Answer:**

javascript

function longestConsecutive(nums) {

const set = new Set(nums);

let maxLen = 0;

for (let num of nums) {

if (!set.has(num - 1)) {

let currLen = 1;

while (set.has(num + currLen)) currLen++;

maxLen = Math.max(maxLen, currLen);

}

}

return maxLen;

}

**Explanation:**

* Use a Set for O(1) lookup.
* Only start counting sequences from the smallest number in each run. **Examples:**

javascript

console.log(longestConsecutive([100, 4, 200, 1, 3, 2])); *// 4 (1, 2, 3, 4)*

console.log(longestConsecutive([1, 3, 5])); *// 1*

These questions test algorithmic efficiency, recursion, custom methods, and practical utilities—perfect for advanced interviews. Let me know if you want more examples, optimizations, or explanations!

### 66. ****Find First Duplicate in an Array****

**Question:** Write a function to find the first duplicate value in an array.

**Answer:**

javascript

function firstDuplicate(arr) {

const seen = new Set();

for (let num of arr) {

if (seen.has(num)) return num;

seen.add(num);

}

return -1;

}

**Explanation:**

* Use a Set to track seen values; return the first one that repeats.
* Return -1 if no duplicates exist. **Examples:**

javascript

console.log(firstDuplicate([2, 1, 3, 5, 2, 3])); *// 2*

console.log(firstDuplicate([1, 2, 3, 4])); *// -1*

### 67. ****Replace All Vowels with a Character****

**Question:** Write a function to replace all vowels in a string with a specified character.

**Answer:**

javascript

function replaceVowels(str, char) {

return str.replace(/[aeiou]/gi, char);

}

**Explanation:**

* Regular expression /[aeiou]/gi matches all vowels (case-insensitive with i, global with g).
* replace() swaps them with the given character. **Examples:**

javascript

console.log(replaceVowels("hello", "\*")); *// "h\*ll\*"*

console.log(replaceVowels("APPLE", "#")); *// "#PPL#"*

### 68. ****Find the Middle Element of an Array****

**Question:** Write a function to return the middle element of an array (or average of two middle elements if even length).

**Answer:**

javascript

function middleElement(arr) {

const mid = Math.floor(arr.length / 2);

return arr.length % 2 === 0 ? (arr[mid - 1] + arr[mid]) / 2 : arr[mid];

}

**Explanation:**

* For odd length, return the middle element.
* For even length, average the two middle elements. **Examples:**

javascript

console.log(middleElement([1, 2, 3, 4, 5])); *// 3*

console.log(middleElement([1, 2, 3, 4])); *// 2.5*

### 69. ****Count Words in a String****

**Question:** Write a function to count the number of words in a string (separated by spaces).

**Answer:**

javascript

function wordCount(str) {

return str.trim().split(/\s+/).length;

}

**Explanation:**

* trim() removes leading/trailing spaces.
* split(/\s+/) splits on one or more whitespace characters. **Examples:**

javascript

console.log(wordCount("Hello world")); *// 2*

console.log(wordCount(" One Two ")); *// 2*

console.log(wordCount("")); *// 0*

### 70. ****Find Maximum Difference in an Array****

**Question:** Write a function to find the maximum difference between any two elements in an array (larger after smaller).

**Answer:**

javascript

function maxDifference(arr) {

let min = arr[0], maxDiff = 0;

for (let i = 1; i < arr.length; i++) {

maxDiff = Math.max(maxDiff, arr[i] - min);

min = Math.min(min, arr[i]);

}

return maxDiff;

}

**Explanation:**

* Track the minimum element seen so far and the maximum difference.
* Update maxDiff if a larger difference is found. **Examples:**

javascript

console.log(maxDifference([7, 1, 5, 4])); *// 4 (5 - 1)*

console.log(maxDifference([9, 8, 7])); *// 0 (no increase)*

### 71. ****Implement Array.reduce****

**Question:** Write a custom implementation of the reduce array method.

**Answer:**

javascript

function customReduce(arr, callback, initialValue) {

let acc = initialValue !== undefined ? initialValue : arr[0];

const start = initialValue !== undefined ? 0 : 1;

for (let i = start; i < arr.length; i++) {

acc = callback(acc, arr[i], i, arr);

}

return acc;

}

**Explanation:**

* Handle optional initialValue: if provided, start from index 0; otherwise, use arr[0] and start from 1.
* Accumulate the result using the callback. **Examples:**

javascript

console.log(customReduce([1, 2, 3], (a, b) => a + b, 0)); *// 6*

console.log(customReduce([1, 2, 3], (a, b) => a \* b)); *// 6*

### 72. ****Check if a Number is a Power of Two****

**Question:** Write a function to check if a number is a power of 2.

**Answer:**

javascript

function isPowerOfTwo(n) {

return n > 0 && (n & (n - 1)) === 0;

}

**Explanation:**

* A power of 2 has exactly one 1-bit in its binary form (e.g., 8 = 1000).
* n & (n - 1) clears the rightmost 1; if n is a power of 2, the result is 0. **Examples:**

javascript

console.log(isPowerOfTwo(8)); *// true (2³)*

console.log(isPowerOfTwo(10)); *// false*

### 73. ****Chunk an Array****

**Question:** Write a function to split an array into chunks of a specified size.

**Answer:**

javascript

function chunkArray(arr, size) {

if (size <= 0) return [];

const result = [];

for (let i = 0; i < arr.length; i += size) {

result.push(arr.slice(i, i + size));

}

return result;

}

**Explanation:**

* Iterate with step size, slicing chunks into a new array. **Examples:**

javascript

console.log(chunkArray([1, 2, 3, 4, 5], 2)); *// [[1, 2], [3, 4], [5]]*

console.log(chunkArray([1, 2, 3], 1)); *// [[1], [2], [3]]*

### 74. ****Convert Number to Binary****

**Question:** Write a function to convert a positive integer to its binary representation as a string.

**Answer:**

javascript

function toBinary(n) {

return n.toString(2);

}

**Alternative (Manual):**

javascript

function toBinary(n) {

if (n === 0) return "0";

let binary = "";

while (n > 0) {

binary = (n % 2) + binary;

n = Math.floor(n / 2);

}

return binary;

}

**Explanation:**

* toString(2) is the built-in way.
* Manual: Build the string by repeatedly dividing by 2 and collecting remainders. **Examples:**

javascript

console.log(toBinary(5)); *// "101"*

console.log(toBinary(10)); *// "1010"*

### 75. ****Find the Closest Number to Zero****

**Question:** Write a function to find the number closest to zero in an array (if tied, return the positive one).

**Answer:**

javascript

function closestToZero(arr) {

return arr.reduce((closest, curr) =>

Math.abs(curr) < Math.abs(closest) || (Math.abs(curr) === Math.abs(closest) && curr > closest)

? curr : closest, arr[0]);

}

**Explanation:**

* Compare absolute values; if equal, prefer the positive number. **Examples:**

javascript

console.log(closestToZero([-4, -2, 1, 4, 8])); *// 1*

console.log(closestToZero([-2, 2])); *// 2*

These questions test bitwise operations, array manipulation, string handling, and custom method implementation—key skills for advanced JavaScript interviews. Let me know if you’d like more examples, edge cases, or deeper dives!

Here are more JavaScript 2-10 line interview questions with answers, explanations, and examples. These explore additional algorithmic challenges, functional programming, and practical utilities—perfect for rounding out your interview preparation.

### 76. ****Find the Smallest Positive Missing Number****

**Question:** Write a function to find the smallest positive integer missing from an array.

**Answer:**

javascript

function smallestMissingPositive(arr) {

const n = arr.length;

for (let i = 0; i < n; i++) {

while (arr[i] > 0 && arr[i] <= n && arr[arr[i] - 1] !== arr[i]) {

[arr[arr[i] - 1], arr[i]] = [arr[i], arr[arr[i] - 1]];

}

}

for (let i = 0; i < n; i++) if (arr[i] !== i + 1) return i + 1;

return n + 1;

}

**Explanation:**

* Place each positive number x in its correct index (x - 1) if it’s within range.
* Then find the first index where the value doesn’t match index + 1. **Examples:**

javascript

console.log(smallestMissingPositive([3, 4, -1, 1])); *// 2*

console.log(smallestMissingPositive([1, 2, 0])); *// 3*

### 77. ****Check if String is Valid Email****

**Question:** Write a function to check if a string is a valid email address (basic validation).

**Answer:**

javascript

function isValidEmail(str) {

return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(str);

}

**Explanation:**

* Regex: [^\s@]+ (one or more non-space, non-@ characters), @, then domain and TLD.
* Basic check for format, not full RFC 5322 compliance. **Examples:**

javascript

console.log(isValidEmail("test@example.com")); *// true*

console.log(isValidEmail("test@.com")); *// false*

### 78. ****Sum of Nested Array****

**Question:** Write a function to sum all numbers in a nested array.

**Answer:**

javascript

function sumNested(arr) {

return arr.flat(Infinity).reduce((sum, num) => sum + (typeof num === 'number' ? num : 0), 0);

}

**Explanation:**

* flat(Infinity) flattens all levels of nesting.
* reduce sums only numeric values, ignoring non-numbers. **Examples:**

javascript

console.log(sumNested([1, [2, 3], [4, [5]]])); *// 15*

console.log(sumNested([1, "a", [2]])); *// 3*

### 79. ****Implement Array.every****

**Question:** Write a custom implementation of the every array method.

**Answer:**

javascript

function customEvery(arr, callback) {

for (let i = 0; i < arr.length; i++) {

if (!callback(arr[i], i, arr)) return false;

}

return true;

}

**Explanation:**

* Iterate and return false if any element fails the callback; otherwise, return true. **Examples:**

javascript

console.log(customEvery([1, 2, 3], x => x > 0)); *// true*

console.log(customEvery([1, -2, 3], x => x > 0)); *// false*

### 80. ****Find All Palindromes in a String****

**Question:** Write a function to find all palindromic substrings in a string.

**Answer:**

javascript

function findPalindromes(str) {

const palindromes = new Set();

for (let i = 0; i < str.length; i++) {

for (let j = i + 1; j <= str.length; j++) {

const sub = str.slice(i, j);

if (sub === sub.split('').reverse().join('')) palindromes.add(sub);

}

}

return [...palindromes];

}

**Explanation:**

* Check all substrings; if one equals its reverse, it’s a palindrome.
* Use a Set to avoid duplicates. **Examples:**

javascript

console.log(findPalindromes("aba")); *// ["a", "b", "aba"]*

console.log(findPalindromes("hello")); *// ["h", "e", "l", "ll", "o"]*

### 81. ****Rotate String****

**Question:** Write a function to rotate a string left by n positions.

**Answer:**

javascript

function rotateString(str, n) {

n = n % str.length;

return str.slice(n) + str.slice(0, n);

}

**Explanation:**

* n % str.length handles cases where n exceeds string length.
* Concatenate the part after n with the part before n. **Examples:**

javascript

console.log(rotateString("hello", 2)); *// "llohe"*

console.log(rotateString("abc", 1)); *// "bca"*

### 82. ****Find Median of an Array****

**Question:** Write a function to find the median of an array of numbers.

**Answer:**

javascript

function findMedian(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];

}

**Explanation:**

* Sort the array, then handle odd (middle element) or even (average of two middle elements) cases. **Examples:**

javascript

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

console.log(findMedian([1, 2, 3, 4])); *// 2.5*

### 83. ****Implement a Queue Using Arrays****

**Question:** Write a simple queue implementation with enqueue and dequeue methods.

**Answer:**

javascript

function Queue() {

this.items = [];

this.enqueue = item => this.items.push(item);

this.dequeue = () => this.items.shift();

}

**Explanation:**

* push adds to the end (enqueue), shift removes from the front (dequeue). **Examples:**

javascript

const q = new Queue();

q.enqueue(1); q.enqueue(2);

console.log(q.dequeue()); *// 1*

console.log(q.dequeue()); *// 2*

### 84. ****Find the Next Greater Element****

**Question:** Write a function to find the next greater element for each element in an array (or -1 if none).

**Answer:**

javascript

function nextGreater(arr) {

const result = new Array(arr.length).fill(-1);

for (let i = 0; i < arr.length; i++) {

for (let j = i + 1; j < arr.length; j++) {

if (arr[j] > arr[i]) {

result[i] = arr[j];

break;

}

}

}

return result;

}

**Explanation:**

* For each element, look ahead for the first greater value; if none, keep -1. **Examples:**

javascript

console.log(nextGreater([4, 5, 2, 25])); *// [5, 25, 25, -1]*

console.log(nextGreater([1, 2, 3])); *// [2, 3, -1]*

### 85. ****Convert Array to Object****

**Question:** Write a function to convert an array of key-value pairs into an object.

**Answer:**

javascript

function arrayToObject(arr) {

return arr.reduce((obj, [key, value]) => ({ ...obj, [key]: value }), {});

}

**Explanation:**

* Use reduce to build an object from pairs, spreading the accumulator each time. **Examples:**

javascript

console.log(arrayToObject([["a", 1], ["b", 2]])); *// { a: 1, b: 2 }*

console.log(arrayToObject([["x", "y"]])); *// { x: "y" }*

These questions test regex, array manipulation, custom data structures, and algorithmic thinking—excellent for advanced JavaScript interviews. Let me know if you want more examples, edge cases, or deeper explanations!

### 86. ****Find the Longest Substring Without Repeating Characters****

**Question:** Write a function to find the length of the longest substring without repeating characters.

**Answer:**

javascript

function longestUniqueSubstring(str) {

let maxLen = 0, start = 0, seen = {};

for (let i = 0; i < str.length; i++) {

if (seen[str[i]] >= start) start = seen[str[i]] + 1;

seen[str[i]] = i;

maxLen = Math.max(maxLen, i - start + 1);

}

return maxLen;

}

**Explanation:**

* Use a sliding window: start moves when a repeat is found, tracked by seen.
* Update maxLen with the current window size. **Examples:**

javascript

console.log(longestUniqueSubstring("abcabcbb")); *// 3 ("abc")*

console.log(longestUniqueSubstring("bbbbb")); *// 1 ("b")*

### 87. ****Check if a String is a Valid Number****

**Question:** Write a function to check if a string represents a valid number (integer or decimal).

**Answer:**

javascript

function isValidNumber(str) {

return !isNaN(str) && str.trim() !== "" && /^-?\d\*\.?\d+$/.test(str);

}

**Explanation:**

* !isNaN checks if it’s numeric, trim() ensures no empty strings, and regex allows optional minus, decimals. **Examples:**

javascript

console.log(isValidNumber("123")); *// true*

console.log(isValidNumber("-12.34")); *// true*

console.log(isValidNumber("12a")); *// false*

### 88. ****Find the Majority Element****

**Question:** Write a function to find the majority element in an array (appears more than n/2 times).

**Answer:**

javascript

function majorityElement(arr) {

let candidate = arr[0], count = 1;

for (let i = 1; i < arr.length; i++) {

count += arr[i] === candidate ? 1 : -1;

if (count === 0) candidate = arr[i], count = 1;

}

return candidate;

}

**Explanation:**

* Boyer-Moore Voting Algorithm: Cancel out non-majority elements; the remaining candidate is the majority. **Examples:**

javascript

console.log(majorityElement([3, 2, 3])); *// 3*

console.log(majorityElement([2, 2, 1, 1, 2])); *// 2*

### 89. ****Implement Array.some****

**Question:** Write a custom implementation of the some array method.

**Answer:**

javascript

function customSome(arr, callback) {

for (let i = 0; i < arr.length; i++) {

if (callback(arr[i], i, arr)) return true;

}

return false;

}

**Explanation:**

* Return true if any element passes the callback; otherwise, false. **Examples:**

javascript

console.log(customSome([1, 2, 3], x => x > 2)); *// true*

console.log(customSome([1, -2], x => x < 0)); *// true*

### 90. ****Multiply Without Using \* Operator****

**Question:** Write a function to multiply two numbers without using the \* operator.

**Answer:**

javascript

function multiply(a, b) {

let result = 0;

for (let i = 0; i < Math.abs(b); i++) {

result += a;

}

return b < 0 ? -result : result;

}

**Explanation:**

* Add a to itself b times; adjust sign based on b. **Examples:**

javascript

console.log(multiply(5, 3)); *// 15*

console.log(multiply(4, -2)); *// -8*

### 91. ****Find the Odd One Out****

**Question:** Write a function to find the number that appears an odd number of times in an array (all others appear even times).

**Answer:**

javascript

function findOddOne(arr) {

return arr.reduce((acc, num) => acc ^ num, 0);

}

**Explanation:**

* XOR (^) cancels pairs (even occurrences become 0), leaving the odd one. **Examples:**

javascript

console.log(findOddOne([7, 3, 7, 3, 5])); *// 5*

console.log(findOddOne([1, 1, 2, 2, 3])); *// 3*

### 92. ****Reverse Linked List (Array Simulation)****

**Question:** Write a function to reverse a singly linked list represented as an array.

**Answer:**

javascript

function reverseList(arr) {

return arr.reverse();

}

**Manual Alternative:**

javascript

function reverseList(arr) {

let result = [];

for (let i = arr.length - 1; i >= 0; i--) {

result.push(arr[i]);

}

return result;

}

**Explanation:**

* reverse() is simplest; manual approach builds a new array backward. **Examples:**

javascript

console.log(reverseList([1, 2, 3, 4])); *// [4, 3, 2, 1]*

console.log(reverseList([1])); *// [1]*

### 93. ****Find the GCD (Greatest Common Divisor)****

**Question:** Write a function to find the GCD of two numbers using Euclid’s algorithm.

**Answer:**

javascript

function gcd(a, b) {

return b === 0 ? a : gcd(b, a % b);

}

**Explanation:**

* Recursive: GCD(a, b) = GCD(b, a % b); base case is when b = 0. **Examples:**

javascript

console.log(gcd(48, 18)); *// 6*

console.log(gcd(7, 13)); *// 1*

### 94. ****Partition Array into Even and Odd****

**Question:** Write a function to partition an array into even and odd numbers (maintain relative order).

**Answer:**

javascript

function partitionEvenOdd(arr) {

return [arr.filter(x => x % 2 === 0), arr.filter(x => x % 2 !== 0)];

}

**Explanation:**

* Use filter to separate evens and odds into two arrays. **Examples:**

javascript

console.log(partitionEvenOdd([1, 2, 3, 4, 5])); *// [[2, 4], [1, 3, 5]]*

console.log(partitionEvenOdd([1, 3, 5])); *// [[], [1, 3, 5]]*

### 95. ****Generate Prime Numbers Up to n****

**Question:** Write a function to generate all prime numbers up to n (Sieve of Eratosthenes).

**Answer:**

javascript

function primesUpToN(n) {

const sieve = Array(n + 1).fill(true);

sieve[0] = sieve[1] = false;

for (let i = 2; i \* i <= n; i++) {

if (sieve[i]) for (let j = i \* i; j <= n; j += i) sieve[j] = false;

}

return sieve.reduce((primes, isPrime, i) => isPrime ? [...primes, i] : primes, []);

}

**Explanation:**

* Mark multiples of each prime as non-prime; collect remaining true indices. **Examples:**

javascript

console.log(primesUpToN(10)); *// [2, 3, 5, 7]*

console.log(primesUpToN(5)); *// [2, 3, 5]*

These questions test bitwise tricks, mathematical algorithms, custom array methods, and practical string validation—excellent for showcasing advanced JavaScript mastery. Let me know if you’d like more examples, optimizations, or deeper explanations!

### 96. ****Count Set Bits in a Number****

**Question:** Write a function to count the number of 1s (set bits) in the binary representation of a number.

**Answer:**

javascript

function countSetBits(n) {

let count = 0;

while (n) {

count += n & 1;

n >>= 1;

}

return count;

}

**Explanation:**

* n & 1 checks the least significant bit; right shift (>>=) moves to the next bit.
* Continue until n becomes 0. **Examples:**

javascript

console.log(countSetBits(7)); *// 3 (111 in binary)*

console.log(countSetBits(4)); *// 1 (100 in binary)*

### 97. ****Check if String is Rotated Version of Another****

**Question:** Write a function to check if one string is a rotated version of another.

**Answer:**

javascript

function isRotation(str1, str2) {

return str1.length === str2.length && (str1 + str1).includes(str2);

}

**Explanation:**

* Concatenate str1 with itself; if str2 is a rotation, it will be a substring.
* Length check ensures they’re comparable. **Examples:**

javascript

console.log(isRotation("waterbottle", "erbottlewat")); *// true*

console.log(isRotation("hello", "lohel")); *// true*

console.log(isRotation("abc", "cba")); *// false*

### 98. ****Find the Single Number****

**Question:** Write a function to find the number that appears once in an array where every other number appears twice.

**Answer:**

javascript

function singleNumber(arr) {

return arr.reduce((acc, num) => acc ^ num, 0);

}

**Explanation:**

* XOR (^) of a number with itself is 0; XOR with 0 is the number itself.
* Pairs cancel out, leaving the single number. **Examples:**

javascript

console.log(singleNumber([2, 2, 1])); *// 1*

console.log(singleNumber([4, 1, 2, 1, 2])); *// 4*

### 99. ****Implement Array.flat****

**Question:** Write a custom implementation of the flat array method (one level deep).

**Answer:**

javascript

function customFlat(arr) {

return arr.reduce((flat, item) => flat.concat(Array.isArray(item) ? item : [item]), []);

}

**Explanation:**

* Use reduce with concat to flatten one level; wrap non-arrays in an array. **Examples:**

javascript

console.log(customFlat([1, [2, 3], 4])); *// [1, 2, 3, 4]*

console.log(customFlat([[1, 2], [3, 4]])); *// [1, 2, 3, 4]*

### 100. ****Find the kth Largest Element****

**Question:** Write a function to find the kth largest element in an array.

**Answer:**

javascript

function findKthLargest(arr, k) {

return arr.sort((a, b) => b - a)[k - 1];

}

**Explanation:**

* Sort in descending order and return the element at index k - 1. **Examples:**

javascript

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

console.log(findKthLargest([1, 2, 3], 1)); *// 3*

### 101. ****Check if String is Valid Parentheses with Wildcards****

**Question:** Write a function to check if a string of parentheses and wildcards (\*) is valid (\* can be (, ), or empty).

**Answer:**

javascript

function checkValidString(s) {

let low = 0, high = 0;

for (let char of s) {

low += char === '(' ? 1 : char === ')' ? -1 : 0;

high += char === ')' ? -1 : 1;

if (high < 0) return false;

low = Math.max(low, 0);

}

return low === 0;

}

**Explanation:**

* low tracks min open parens (treat \* as )), high tracks max (treat \* as ().
* Valid if low reaches 0 and high never goes negative. **Examples:**

javascript

console.log(checkValidString("()")); *// true*

console.log(checkValidString("(\*)")); *// true*

console.log(checkValidString("(\*))")); *// true*

console.log(checkValidString("(()")); *// false*

### 102. ****Cumulative Sum Array****

**Question:** Write a function to create a cumulative sum array from an input array.

**Answer:**

javascript

function cumulativeSum(arr) {

return arr.reduce((sums, num, i) => sums.concat(i === 0 ? num : sums[i - 1] + num), []);

}

**Explanation:**

* Build an array where each element is the sum of all previous elements plus the current one. **Examples:**

javascript

console.log(cumulativeSum([1, 2, 3, 4])); *// [1, 3, 6, 10]*

console.log(cumulativeSum([1, -1, 2])); *// [1, 0, 2]*

### 103. ****Find the LCM (Least Common Multiple)****

**Question:** Write a function to find the LCM of two numbers.

**Answer:**

javascript

function lcm(a, b) {

const gcd = (x, y) => y === 0 ? x : gcd(y, x % y);

return Math.abs(a \* b) / gcd(a, b);

}

**Explanation:**

* LCM = |a \* b| / GCD; use Euclid’s algorithm for GCD. **Examples:**

javascript

console.log(lcm(12, 18)); *// 36*

console.log(lcm(5, 7)); *// 35*

### 104. ****Sort Array by Parity****

**Question:** Write a function to sort an array with even numbers first, then odd numbers.

**Answer:**

javascript

function sortByParity(arr) {

return arr.sort((a, b) => (a % 2) - (b % 2));

}

**Explanation:**

* sort with (a % 2) - (b % 2): evens (0) come before odds (1). **Examples:**

javascript

console.log(sortByParity([3, 1, 2, 4])); *// [2, 4, 3, 1]*

console.log(sortByParity([1, 3, 5])); *// [1, 3, 5]*

### 105. ****Find All Permutations of a String****

**Question:** Write a function to find all permutations of a string.

**Answer:**

javascript

function permute(str) {

if (str.length <= 1) return [str];

const result = [];

for (let i = 0; i < str.length; i++) {

const char = str[i];

const subPerms = permute(str.slice(0, i) + str.slice(i + 1));

subPerms.forEach(sub => result.push(char + sub));

}

return result;

}

**Explanation:**

* Recursively generate permutations by fixing one character and permuting the rest. **Examples:**

javascript

console.log(permute("ab")); *// ["ab", "ba"]*

console.log(permute("abc")); *// ["abc", "acb", "bac", "bca", "cab", "cba"]*

These questions test bit manipulation, advanced string algorithms, mathematical relationships, and recursive thinking—great for showcasing deep JavaScript expertise. Let me know if you want more examples, edge cases, or further clarification!