Basic Coding Programs for Interview Preparation

---

1. Prime Number Check

Check if a number is prime.

```
function isPrime(num) {
    if (num <= 1) return false; // 0 and 1 are not prime numbers
    for (let i = 2; i < num; i++) {
        if (num % i === 0) return false; // If divisible by any number
other than 1 and itself
    }
    return true;
}

console.log(isPrime(7)); // true
console.log(isPrime(10)); // false
```

---

2. Fibonacci Sequence (First N Terms)

Print the first `n` Fibonacci numbers.

```
function fibonacci(n) {
    let fib = [0, 1];
    for (let i = 2; i < n; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
    return fib;
}

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

---

3. Factorial of a Number

Find the factorial of a number.

```
function factorial(n) {
    if (n === 0 || n === 1) return 1;
    return n * factorial(n - 1);
}

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

---

4. Reverse a String

Reverse a given string.

```
function reverseString(str) {
    return str.split('').reverse().join('');
}

console.log(reverseString("hello")); // "olleh"
```

---

5. Palindrome Check

Check if a string is a palindrome (reads the same forward and backward).

```
function isPalindrome(str) {
    let reversed = str.split('').reverse().join('');
    return str === reversed;
}

console.log(isPalindrome("madam")); // true
console.log(isPalindrome("hello")); // false
```

---

6. Sum of Digits of a Number

Find the sum of digits of a number.

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

console.log(sumOfDigits(123)); // 6
```

---

7. Find Largest Number in an Array

Find the largest number in an array.

```
function findLargest(arr) {
    return Math.max(...arr);
}

console.log(findLargest([1, 4, 3, 2, 7, 5])); // 7
```

---

8. Check for Armstrong Number

Check if a number is an Armstrong number (a number that is equal to the sum of its own digits each raised to the power of the number of digits).

```
function isArmstrong(num) {
    let digits = num.toString().split('');
    let sum = 0;
    let n = digits.length;
    for (let i = 0; i < n; i++) {
        sum += Math.pow(Number(digits[i]), n);
    }
    return sum === num;
}

console.log(isArmstrong(153)); // true (1^3 + 5^3 + 3^3 = 153)
console.log(isArmstrong(123)); // false
```

---

9. Find the Second Largest Element in an Array

Find the second largest number in an array.

```
function secondLargest(arr) {
    let largest = -Infinity;
    let second = -Infinity;

    for (let i = 0; i < arr.length; i++) {
        if (arr[i] > largest) {
            second = largest;
            largest = arr[i];
        } else if (arr[i] > second && arr[i] !== largest) {
            second = arr[i];
        }
    }
    return second;
}

console.log(secondLargest([1, 4, 3, 2, 7, 5])); // 6
```

---

10. Remove Duplicate Elements from Array

Remove duplicates from an array.

```
function removeDuplicates(arr) {
    return [...new Set(arr)];
}

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

---

11. Merge Two Sorted Arrays

Merge two sorted arrays into one sorted array.

```
function mergeArrays(arr1, arr2) {
    return arr1.concat(arr2).sort((a, b) => a - b);
}

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

---

12. Find Common Elements in Two Arrays

Find common elements in two arrays.

```
function findCommon(arr1, arr2) {
    return arr1.filter(value => arr2.includes(value));
}

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

---

13. Count Vowels in a String

Count the number of vowels in a given string.

```
function countVowels(str) {
    let count = 0;
    let vowels = "aeiou";
    for (let char of str.toLowerCase()) {
        if (vowels.includes(char)) {
            count++;
        }
    }
    return count;
}

console.log(countVowels("hello")); // 2
```

---

14. Find Missing Number in Array

Find the missing number in an array that contains numbers from 1 to n with one number missing.

```
function findMissingNumber(arr, n) {
    let totalSum = (n * (n + 1)) / 2;
    let arraySum = arr.reduce((a, b) => a + b, 0);
    return totalSum - arraySum;
}

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

---

15. Sorting an Array (Bubble Sort)

Sort an array using the Bubble Sort algorithm.

```
function bubbleSort(arr) {
    for (let i = 0; i < arr.length; i++) {
        for (let j = 0; j < arr.length - 1 - i; j++) {
            if (arr[j] > arr[j + 1]) {
                [arr[j], arr[j + 1]] = [arr[j + 1], arr[j]]; // Swap
elements
            }
        }
    }
    return arr;
}

console.log(bubbleSort([5, 2, 9, 1, 5, 6])); // [1, 2, 5, 5, 6, 9]
```

Basic JavaScript coding questions

Basic JavaScript questions cover concepts like data types, variables and scoping, array, string manipulation, OOP (Object Oriented Programming), control flow, error handling, DOM manipulation, and asynchronous programming. The basic JavaScript coding interview questions are:

1. Write a JavaScript function to calculate the sum of two numbers.

When managers ask this question, they are looking for the candidate's basic understanding of JavaScript. They assess their understanding of basic syntax along with problem-solving skills. This also helps evaluate the candidate's coding style and attention to detail.

I would take two parameters and the following function can be used to calculate the sum of any 2 numbers that are passed as arguments.

```
function sumOfTwoNumbers(a, b) {

  return a + b;

}
```

2. Write a JavaScript program to find the maximum number in an array.

A hiring manager asks this question to analyze the candidate's ability to write clear and efficient code. It's crucial for candidates to explain the code step-by-step while demonstrating bug-free code.

```
function findMaxNumber(arr) {

  return Math.max(...arr);

}
```

3. Write a JavaScript function to check if a given string is a palindrome (reads the same forwards and backwards).

The interviewer is looking for the candidate's familiarity with loop constructs, JavaScript string methods, and other basic JavaScript syntax. They will evaluate the candidate's skills based on the approach used to solve the palindrome problem.

```
function isPalindrome(str) {

  return str === str.split(”).reverse().join(”);

}
```

4. Write a JavaScript program to reverse a given string.

Hiring managers are expecting an accurate solution that demonstrates the interviewee's proficiency in JavaScript programming.

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

5. Write a JavaScript function that takes an array of numbers and returns a new array with only the even numbers.

Interviewers are looking for candidates who can not only clearly explain the solution along with the code, but also show the ability to think logically and articulate their thought processes.

By using the filter method on the array, I can check if each element is even or not by using the modulus operator (%) with 2. The element is even if the result is 0. This can be included in the new array.

```
function filterEvenNumbers(numbers) {

  return numbers.filter(num => num % 2 === 0);
}
```

6. Write a JavaScript program to calculate the factorial of a given number.

By asking this question, managers aim to assess the candidate's algorithmic thinking and understanding of JavaScript programming. The interviewer expects the candidate to demonstrate their knowledge of the factorial concept.

A factorial number is the product of all positive integers, which are equal to or less than the given number.

```
function factorial(number) {

  if (number === 0 || number === 1) {

    return 1;

  } else {

    return number * factorial(number – 1);

  }

}
```

7. Write a JavaScript function to check if a given number is prime.

Interviewers can analyze the candidate's knowledge of JavaScript algorithms and mathematical concepts. They expect the candidate to translate a mathematical concept into functional code.

To check if a given number is prime, loop from 2 to the square root of the number. If any integer evenly divides it, the number is not prime.

```
function isPrime(num) {

  if (num <= 1) return false;

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

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

  }

  return true;

}
```

8. Write a JavaScript program to find the largest element in a nested array.

When asking this question, interviewers are looking for the candidate's ability to handle nested data structures and apply their knowledge of conditional statements, arrays, and loops. Candidates must apply their knowledge to real-world scenarios.

```
function findLargestElement(nestedArray) {

  let largest = nestedArray[0][0];

  for (let arr of nestedArray) {

    for (let num of arr) {

      if (num > largest) {

        largest = num;

      }

    }

  }
  return largest;

}
```

9. Write a JavaScript function that returns the Fibonacci sequence up to a given number of terms.

This question helps hiring managers assess the interviewee’s understanding of fundamental algorithms in JavaScript. They expect the candidate to consider edge cases and handle errors.

```
function fibonacciSequence(numTerms) {

  if (numTerms <= 0) return [];

  if (numTerms === 1) return [0];



  let sequence = [0, 1];

  while (sequence.length < numTerms) {

       let nextNumber = sequence[sequence.length – 1] + sequence[sequence.length – 2];

    sequence.push(nextNumber);

  }

  return sequence;

}
```

10. Write a JavaScript program to convert a string to title case (capitalize the first letter of each word).

Interviewers analyze the candidate's ability to break down a problem into manageable steps and demonstrate knowledge of string manipulation, looping, and basic JavaScript functions.

```
function toTitleCase(str) {

  return str.replace(/\b\w/g, l => l.toUpperCase());

}
```

Advanced JavaScript coding interview questions

Advanced JavaScript coding includes various complex concepts and techniques. Such key concepts are often tested in JavaScript interviews. Some of the concepts are – closure and scope, prototypal inheritance, **functional programming**, design patterns, memory management, ES6+ features, and many more.

1. Implement a debounce function in JavaScript that limits the frequency of a function's execution when it's called repeatedly within a specified time frame.

Interviewers expect the candidate to showcase their ability to clearly explain the purpose of the debounce function and its usage in scenarios where function calls need to be controlled. They are looking for the person's ability to articulate technical concepts clearly.

By delaying the execution of the debounce function until the specified time frame has passed, the frequency can be limited.

```
function debounce(func, delay) {

  let timer;

  return function() {

    clearTimeout(timer);
    timer = setTimeout(func, delay);

  };

}
```

2. Write a function that takes an array of objects and a key, and returns a new array sorted based on the values of that key in ascending order.

By asking this question, hiring managers analyze how well the candidate can discuss the sorting algorithm and its time complexity. It's also crucial for candidates to demonstrate their code's robustness.

The following function takes an array of objects and a key to sort the array based on the values in ascending order.

```
function sortByKey(arr, key) {

  return arr.sort((a, b) => a[key] – b[key]);

}
```

3. Implement a deep clone function in JavaScript that creates a copy of a nested object or array without any reference to the original.

Hiring managers want to assess the interviewee's skill to handle complex coding tasks and understand the concept of avoiding reference issues while cloning.

By using two methods together and creating a deep clone, I can serialize the object to a JSON string. I would then parse it back into a new object, thereby removing any reference to the original object.

```
function deepClone(obj) {

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

4. Write a recursive function to calculate the factorial of a given number.

Interviewers expect the candidate to write a concise recursive function that handles edge cases. Candidates must show their understanding of how recursion works to avoid infinite loops or stack overflow errors.

```
function factorial(num) {

  if (num <= 1) return 1;

  return num * factorial(num – 1);

}
```

5. Implement a function that takes two sorted arrays and merges them into a single sorted array without using any built-in sorting functions.

When interviewers ask this question, they seek to assess the knowledge of algorithms and efficiency in handling sorted data. They also look for the ability to think of and execute a correct solution.

I can implement a function that can efficiently merge two sorted arrays.

```
function mergeSortedArrays(arr1, arr2) {

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

}
```

6. Write a function that checks if a given string is a palindrome, considering only alphanumeric characters and ignoring case.

Interviewers analyze the interviewee's approach to execute code and demonstrate familiarity with handling case-sensitive and alphanumeric checks, regular expressions, and JavaScript string methods.

```
function isPalindrome(str) {

  const cleanStr = str.replace(/[^a-zA-Z0-9]/g, ”).toLowerCase();

  const reversedStr = cleanStr.split(”).reverse().join(”);

  return cleanStr === reversedStr;

}
```

7. Create a JavaScript class for a linked list with methods to insert a node at the beginning, end, or at a specific position, and to delete a node from a given position.

By asking this question, interviewers can evaluate how well a candidate can design and implement a class for a linked list while also presenting their problem-solving skills.

I would implement a linked list with methods to insert a node at the beginning, end, and at specific positions. Then, I would delete a node from a given position.

8. Implement a function that flattens a nested array in JavaScript, converting it into a single-level array.

Managers can gauge the candidate's logical thinking skills and capability to handle complex data structures. Interviewees should demonstrate their knowledge of loops, recursion, and arrays.

```
const flattenArray = (nestedArray) => {

  return nestedArray.flat(Infinity);
};
```

9. Write a function that determines if two strings are anagrams of each other

When interviewers present this question, they aim to measure how well the candidate can use appropriate string-related methods and identify anagrams accurately.

```
function areAnagrams(str1, str2) {

  return str1.split(“”).sort().join(“”) === str2.split(“”).sort().join(“”);

}
```

10. Create a JavaScript function that returns the Fibonacci sequence up to a given number, utilizing memoization for optimized performance.

Interviewees are expected to show their proficiency in OOP and familiarity with recursion and memoization. They can also determine the candidate’s attention to detail in class design and organizing code.

By creating a function that uses an array to store the computed values, a Fibonacci sequence can be generated.

```
function fibonacciWithMemoization(n) {

  let memo = [0, 1];

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

    memo[i] = memo[i – 1] + memo[i – 2];

  }

  return memo;
}
```

Common JavaScript coding interview questions

Some of the common JavaScript coding interview questions typically cover these topics: checking for palindrome, finding missing/largest numbers, object manipulation, removing duplicates, merging, etc.

1. Write a function to check if a given string is a palindrome.

Hiring managers review how well a candidate can handle edge cases while handling case sensitivity, punctuation, and whitespace.

This function takes a string as input to convert it into lowercase and then compares it with its reverse. The string can be deemed a palindrome if the two match.

```
function isPalindrome(str) {

  return str.toLowerCase() === str.toLowerCase().split(”).reverse().join(”);

}
```

2. Implement a function to reverse a string without using the built-in reverse() method.

Hiring managers aim to analyze the candidate’s knowledge of string manipulation in JavaScript while also measuring their ability to think of alternative solutions.

I would use a for lopp to iterate through the characters from the end to the beginning. By appending the character to a new string, it results in the reversed output.

```
function reverseString(str) {

  let reversed = ”;

  for (let i = str.length – 1; i >= 0; i–) {

    reversed += str[i];

  }

  return reversed;

}
```

3. Given an array of numbers, write a function to find the largest and smallest numbers in the array.

By presenting the candidates with this question, managers can gauge how well a candidate is familiar with basic JavaScript functions and array manipulation.

I would use the following functions to find the smallest and largest numbers in the array.

```
function findMinMax(arr) {

  let min = Math.min(...arr);

  let max = Math.max(...arr);

  return [min, max];

}
```

4. Write a function that takes an array of integers as input and returns a new array with only the unique elements.

Hiring managers can evaluate the candidate's knowledge of JavaScript functions, array handling capabilities, and communicating technical concepts.

```
function getUniqueElements(arr) {

  return Array.from(new Set(arr));

}
```

5. Implement a function to find the factorial of a given number.

Interviewers can determine the candidate's capability to execute functional code and ability to handle input validation and edge cases. Interviewers also assess the ability to use concise and effective code and provide efficient code implementation.

```
function factorial(number) {

  if (number === 0 || number === 1) return 1;

  return number * factorial(number – 1);

}
```

6. Write a function that determines if a given number is prime or not.

By asking this question, interviewers can understand how good the candidate is proficient in math operations and JavaScript logic. The interviewee should excute a clean and optimized solution that is efficient.

```
function isPrime(num) {

  if (num <= 1) return false;

  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) return false;

  }

  return true;

}
```

7. Implement a function to find the sum of all the numbers in an array.

Such a question helps understand if the interviewee can manipulate arrays and handle numeric values. This also helps managers assess problem-solving capabilities and ability to pay attention to code efficiency.

I would use the reduce method to implement the following function:

```
function findSum(arr) {

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

}
```

8. Given a string, write a function to count the occurrences of each character in the string.

Hiring managers expect the candidate to be familiar with string manipulation and loop constructs. When they ask this question, they can evaluate whether the candidate knows data structures.

```
function countCharacterOccurrences(str) {

  const charCount = {};

  for (let char of str) {

    charCount[char] = (charCount[char] || 0) + 1;
  }

  return charCount;

}
```

9. Implement a function to remove duplicates from an array.

When interviewers present the candidate with this question, they can gauge the level of understanding a candidate has regarding array methods and different approaches to solve the problem.

The following function duplicates from an array by converting it into a Set. This automatically removes duplicates. Next, the function converts the Set back into an array.

```
function removeDuplicates(arr) {

  return Array.from(new Set(arr));

}
```

10. Write a function that sorts an array of numbers in ascending order.

Interviewees must show their knowledge of bubble sort, merge sort, sorting algorithms, and other approaches. The HR manager aims to measure the capability to execute strong algorithms and handle edge cases.

I can solve this by using JavaScript's built-in sort method.

```
function ascendingSort(numbers) {

  return numbers.sort((a, b) => a – b);

}
```

Tricky JavaScript coding questions

By asking tricky JavaScript coding questions, managers can assess problem—solving skills, JavaScript concepts, and **critical thinking**. These go beyond syntax knowledge and require the candidate to think creatively and logically to solve problems.

1. Write a function that reverses the order of words in a sentence without using the built-in reverse() method.

This question not only assesses the creativity of the candidates but also helps hiring managers understand how well a candidate can come up with a clean and understandable solution.

```
function reverseSentence(sentence) {

  const words = sentence.split(‘ ‘);

  const reversedWords = words.reverse();

  return reversedWords.join(‘ ‘);

}
```

2. Implement a function that checks if a given string is a palindrome (reads the same forwards and backwards) while ignoring whitespace and punctuation.

Interviewers can gauge the interviewee’s capability to handle whitespace and punctuation gracefully while also maintaining the palindrome-checking logic. Candidates must express their knowledge of regular expressions or any other efficient approach.

```
function isPalindrome(str) {

  const cleanedStr = str.replace(/[^\w]/g, ”).toLowerCase();

  const reversedStr = cleanedStr.split(”).reverse().join(”);
  return cleanedStr === reversedStr;

}
```

3. Write a function that takes an array of integers and returns the largest difference between any two numbers in the array.

Candidates should demonstrate their approach to finding the maximum difference between the array elements to handle edge cases and invalid inputs.

```
function largestDifference(arr) {

  let min = arr[0];

  let maxDiff = 0;

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

    if (arr[i] < min) {

      min = arr[i];

    }

else {

      const diff = arr[i] – min;

      if (diff > maxDiff) {

        maxDiff = diff;

      }

    }

  }
  return maxDiff;

}
```

4. Implement a function that removes duplicates from an array, keeping only the unique elements.

Interviewers can analyze how well a candidate can effectively communicate code explanations and their familiarity with algorithmic efficiency.

```
function removeDuplicates(arr) {

  return arr.filter((item, index) => arr.indexOf(item) === index);

}
```

5. Write a function that accepts a number and returns its factorial (e.g., factorial of 5 is 5 x 4 x 3 x 2 x 1).

By presenting this question in the interview, hiring managers can assess the capability of the candidate to handle numeric calculations. They can also determine how well the interviewee can pay attention to handling edge cases, if applicable.

```
function factorial(num) {

  if (num === 0 || num === 1) {

    return 1;

  } else {

    return num * factorial(num – 1);

  }

}
```

6. Implement a function that flattens a nested array into a single-dimensional array.

Interviewers expect the candidates to demonstrate their ability to work with complex data structures and use appropriate techniques to accomplish tasks.

```
function flattenArray(arr) {

  return arr.flat();

}
```

7. Write a function that checks if a given string is an anagram of another string (contains the same characters in a different order).

Candidates should showcase how well they can handle complex algorithms and logic. Interviewers are specifically looking for knowledge in string methods, data structures, and loop constructs.

```
 function isAnagram(str1, str2) {

  const sortedStr1 = str1.split(”).sort().join(”);

  const sortedStr2 = str2.split(”).sort().join(”);

  return sortedStr1 === sortedStr2;

}
```

8. Implement a function that finds the second smallest element in an array of integers.

Interviewers can measure the candidate’s problem-solving skills and their understanding of conditional statements, loops, and arrays.

```
function secondSmallest(arr) {

  const sortedArr = arr.sort((a, b) => a – b);

  return sortedArr[1];

}
```

9. Write a function that generates a random alphanumeric string of a given length.

By asking this question, interviewers can understand how well a candidate can ensure the function produces a reliable and consistent random output.

```
function generateRandomString(length) {

  const characters = ‘ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789’;

  let result = ”;

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

    const randomIndex = Math.floor(Math.random() * characters.length);

    result += characters.charAt(randomIndex);

  }

  return result;

}
```

10. Implement a function that converts a number to its Roman numeral representation.

Hiring managers can gauge a candidate's capability to implement coding solutions and create an efficient algorithm.

```
function toRomanNumeral(number) {

  // Implement your code here

}
```

JavaScript array coding questions

JavaScript array coding interview questions are technical questions asked to gauge candidates' ability to work with arrays along with their familiarity with fundamental data structures.

1. Write a function that returns the sum of all numbers in an array.

By asking such a question, **hiring managers** can evaluate whether the candidate would be able to perform common tasks and solve basic coding challenges.

```
function sumArray(arr) {

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

}
```

2. Implement a function that finds the maximum number in an array.

Depending on the candidate's answer, the manager can determine how effectively the interviewee can work with arrays. The managers can also understand the capability to communicate technical solutions.

```
function findMaxNumber(arr) {

  let max = arr[0];

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

    if (arr[i] > max) {

      max = arr[i];

    }

  }

  return max;

}
```

3. Write a function that returns a new array containing only the unique elements from an input array.

The hiring manager is specifically looking for candidates who can demonstrate an understanding in data manipulation and JavaScript arrays. Additionally, interviewers evaluate how well the candidate strives for an optimized solution without duplicate elements.

```
function getUniqueElements(inputArray) {

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

}
```

4. Implement a function that returns the average value of numbers in an array.

By asking this question, hiring managers can assess the candidate's knowledge of arithmetic operations, array manipulation, and looping.

```
function calculateAverage(numbers) {

  let sum = 0;

  for (let number of numbers) {

    sum += number;

  }

  return sum / numbers.length;

}
```

5. Write a function that sorts an array of strings in alphabetical order.

When interviewers present this question in the interview, they expect the candidate to be familiar with sorting algorithms and JavaScript array manipulation.

```
function sortStrings(arr) {

  return arr.slice().sort();

}
```

6. Implement a function that finds the index of a specific element in an array. If the element is not found, the function should return -1.

Interviewers aim to gauge the candidate's proficiency in use of array methods, handling edge cases, and in JavaScript syntax. Candidates must implement the function proper error handling.

```
function findElementIndex(arr, element) {

  const index = arr.indexOf(element);

  return index !== -1 ? index : -1;

}
```

7. Write a function that removes all falsy values (false, null, 0, "", undefined, and NaN) from an array.

Candidates must showcase communication skills and explain their solutions logically. Interviewers analyze the interviewee's ability to write a function that filters false values from an array.

```
function removeFalsyValues(arr) {

  return arr.filter(Boolean);

}
```

8. Implement a function that merges two arrays into a single array, alternating elements from each array.

Hiring managers determine a candidate's ability to craft efficient algorithms and knowledge of array manipulation.

```
function mergeArrays(array1, array2) {

  const mergedArray = [];

  const maxLength = Math.max(array1.length, array2.length);
  for (let i = 0; i < maxLength; i++) {

    if (i < array1.length) mergedArray.push(array1[i]);

    if (i < array2.length) mergedArray.push(array2[i]);

  }

  return mergedArray;

}
```

9. Write a function that finds the second largest number in an array.

Such a question reveals to the interviewers how well the candidate can use loops and array methods, work with them, and utilize logic to find solutions.

```
function findSecondLargest(arr) {

  arr.sort((a, b) => b – a);

  return arr[1];

}
```

10. Implement a function that groups elements in an array based on a given condition. For example, grouping even and odd numbers into separate arrays.

When interviews ask this question, they aim to evaluate a candidate's understanding of concepts like array methods, conditional statements, and other technical concepts. Candidate should demonstrate good coding style.

```
function groupByCondition(arr, condition) {

  return [
    arr.filter(element => condition(element)),

    arr.filter(element => !condition(element))

  ];

}
```