------------Javascript Interview -----------

1. What is JavaScript? Done
2. Explain the differences between JavaScript and Java. Done
3. How do you declare variables in JavaScript?
4. What are the data types in JavaScript? done
5. How do you check the type of a variable in JavaScript? Done
6. Explain the difference between == and === in JavaScript. Done
7. What is hoisting in JavaScript? Done
8. How do you handle asynchronous operations in JavaScript? Done
9. What are callbacks and how do you use them? Done
10. What are arrow functions in JavaScript? How are they different from regular functions? Done
11. How do you create objects in JavaScript? Provide examples of different methods. Done
12. What is the prototype chain in JavaScript?
13. How do you handle errors and exceptions in JavaScript?
14. Explain the concept of closures in JavaScript. Done
15. What are the different ways to loop over arrays in JavaScript? Done
16. How do you add, remove, and modify elements in an array?
17. What is the difference between null and undefined in JavaScript?
18. How do you work with JSON data in JavaScript?
19. Explain the concept of event bubbling and how to prevent it. Done
20. What are the different ways to include JavaScript in an HTML page? Done

----------------------

-Advanced Javascript Question

1.Explain the concept of "hoisting" in JavaScript and how it affects variable declarations and function declarations

2.What are the different ways to handle asynchronous programming in JavaScript? Compare and contrast callbacks, promises, and async/await.

3.How does the JavaScript event loop work? Describe the difference between the call stack and the task queue

4.What is the scope chain in JavaScript? How does lexical scoping contribute to this behavior

5.How do you handle memory management and garbage collection in JavaScript

6.What are closures in JavaScript, and how are they useful in practical scenarios?

7.Discuss the differences between classical inheritance and prototypal inheritance in JavaScript.

8.Explain the "this" keyword in JavaScript. How is its value determined in different contexts (e.g., global scope, object methods, event handlers).

9.What are IIFE (Immediately Invoked Function Expressions)? Provide an example of their usage.

10.How can you implement inheritance in JavaScript using both ES5 and ES6 syntax?

11.Discuss the differences between the ES5 and ES6 modules in JavaScript.

12.What are the new features introduced in ES6 (ECMAScript 2015), and how do they improve JavaScript development?

13.How do you handle and avoid callback hell in asynchronous JavaScript code?

14.What are generators in JavaScript, and how can they be used to simplify asynchronous code?

15.Explain the concept of memoization in JavaScript and how it can improve the performance of functions.

16.Discuss different design patterns commonly used in JavaScript, such as Singleton, Factory, and Observer.

17.What are the different methods for handling and manipulating arrays in JavaScript (e.g., map, reduce, filter, etc.)?

18.How can you achieve deep cloning of objects in JavaScript to avoid reference-related issues?

19.Explain the concepts of event delegation and event bubbling in JavaScript.

20.What are Web Workers in JavaScript, and how can they be used to achieve multi-threading in web applications?

-----------------------------

--------------Js Implmentation Questions------------

1. Write a function to find the second smallest number in an array of integers.
2. Implement a function that removes all duplicates from an array.
3. Create a function that takes an array of objects with a "name" property and returns an object with names as keys and the corresponding objects as values.

const people = [

{ name: 'John', age: 30 },

{ name: 'Jane', age: 25 },

{ name: 'Bob', age: 35 }

];

Output:

{

'John': { name: 'John', age: 30 },

'Jane': { name: 'Jane', age: 25 },

'Bob': { name: 'Bob', age: 35 }

}

1. Implement a function that groups elements of an array based on a given key.

const students = [

{ name: 'Alice', course: 'Math' },

{ name: 'Bob', course: 'Science' },

{ name: 'Carol', course: 'Math' },

{ name: 'David', course: 'History' }

];

Example Input:

{

'Math': [{ name: 'Alice', course: 'Math' }, { name: 'Carol', course: 'Math' }],

'Science': [{ name: 'Bob', course: 'Science' }],

'History': [{ name: 'David', course: 'History' }]

}

1. Create a function that recursively flattens an array of nested arrays.

Example Input: const nestedArray = [1, [2, [3, 4], 5], 6];

output

[1, 2, 3, 4, 5, 6]

1. Implement a function that finds the intersection of two arrays (elements that are present in both arrays).
2. Write a function that checks if an object contains all the properties of another object.

const obj1 = { a: 1, b: 2, c: 3 };

const obj2 = { a: 1, b: 2 };

false

1. Create a function that deep clones an object.
2. Implement a function that finds the most common element in an array.

const fruits = ['apple', 'banana', 'orange', 'apple', 'banana', 'apple'];

'apple'

1. Write a function that sorts an array of objects based on a numeric property in ascending order.

const products = [

{ name: 'Laptop', price: 1000 },

{ name: 'Phone', price: 800 },

{ name: 'Tablet', price: 500 }

];

[

{ name: 'Tablet', price: 500 },

{ name: 'Phone', price: 800 },

{ name: 'Laptop', price: 1000 }

]

=========================

1. Reverse an Array: Implement a function that reverses the elements of an array in place.

2. Merge Two Arrays: Write a function that merges two sorted arrays into a single sorted array.

3. Array Manipulation: Create a function that takes an array of numbers and returns a new array with each element multiplied by 2.

4. Object Key-Value Swap: Write a function that takes an object and returns a new object with keys and values swapped.

5. Array Unique Values: Implement a function that returns an array with unique values from a given array.

6. Object Property Check: Create a function that checks if a specific property exists in an object.

7. Array Intersection: Write a function that finds the intersection of two arrays (elements that are present in both arrays).

8. Object Deep Merge: Implement a function that deeply merges two objects, combining their properties and nested properties.

9. Array Chunking: Create a function that splits an array into smaller arrays of a specified size.

10. Object Key Mapping: Write a function that maps keys of an object to a new set of keys specified in a mapping object.

11. Array Flattening: Implement a function that recursively flattens an array of nested arrays.

12. Object Filtering: Create a function that filters an object based on a given condition.

13. Array Rotation: Write a function that rotates an array to the right by a given number of steps.

14. Object Sorting: Implement a function that sorts an array of objects based on a specific property.

15. Array Sum and Product: Create a function that calculates the sum and product of an array of numbers.

16. Object Nested Property Access: Write a function that retrieves a value from a deeply nested object based on a provided path.

17. Array Filtering and Mapping: Implement a function that filters an array and then maps the remaining elements to a new array.

18. Object to Array Conversion: Create a function that converts an object into an array of key-value pairs.

19. Array Partitioning: Write a function that divides an array into two parts based on a condition.

20. Object Grouping: Implement a function that groups objects in an array based on a specific property.