JavaScript Interview Question and answers Important

1. What is the difference between var, let, and const?

1. **var**: Function-scoped or globally scoped, can be re-declared and updated.
2. **let**: Block-scoped, can be updated but not re-declared in the same scope.
3. **const**: Block-scoped, cannot be updated or re-declared; must be initialized at declaration.

2. What is a closure in JavaScript?

A closure is a function that retains access to its lexical scope, even when the function is executed outside that scope. It allows the function to remember variables from its outer scope.

3. What is the difference between == and === in JavaScript?

1. ==: Compares values for equality after performing type coercion.
2. ===: Compares both value and type for strict equality, without type coercion.

4. What is the purpose of the 'this' keyword in Java

The 'this' keyword refers to the context in which a function is executed. It can refer to the global object, an object instance, or be undefined in strict mode, depending on how the function is called.

5. What is event delegation in JavaScript?

Event delegation is a technique where a single event listener is added to a parent element to manage events for multiple child elements. This improves performance and simplifies event handling by reducing the number of event listeners needed.

6. What is the difference between synchronous and asynchronous JavaScript?

1. **Synchronous**: Code execution happens in a sequential manner, blocking the thread until the current operation completes.
2. **Asynchronous**: Code execution can continue while waiting for operations (like network requests) to complete, allowing other code to run in the meantime.

7. What is a promise in JavaScript?

A promise is an object representing the eventual completion (or failure) of an asynchronous operation. It allows you to handle asynchronous results using `.then()` for success and `.catch()` for errors.

```
const promise = new Promise((resolve, reject) => { let success = true; success ?
resolve("Done") : reject("Error"); }); promise .then((res) => console.log(res))
.catch((err) => console.log(err));
```

8. What is the purpose of the 'async' and 'await keywords in JavaScript?

The 'async' keyword is used to declare a function as asynchronous, allowing it to use the 'await' keyword. The 'await' keyword pauses the execution of the async function until the promise is resolved, making asynchronous code easier to read and write.

9. What is the difference between null and undefined in JavaScript?

1. **null**: An intentional absence of any object value, explicitly assigned.
2. **undefined**: A variable that has been declared but not assigned a value, or a function that does not return a value.

10. What is the purpose of the 'bind', 'call', and 'apply' methods in JavaScript?

1. **bind**: Creates a new function with a specified 'this' context and optional arguments.
2. **call**: Invokes a function with a specified 'this' context and arguments passed individually.
3. **apply**: Invokes a function with a specified 'this' context and arguments passed as an array.

11. What is the purpose of the 'setTimeout' and ' setInterval' functions in JavaScript?

1. **setTimeout**: Executes a function after a specified delay (in milliseconds).
2. **setInterval**: Repeatedly executes a function at specified intervals (in milliseconds) until cleared.

12. What is the purpose of the 'forEach' method in JavaScript?

The 'forEach' method is used to iterate over elements in an array, executing a provided function once for each element. It does not return a new array and cannot be used with 'break' or 'continue'.

13. What is the purpose of the 'map' method in JavaScript?

The 'map' method creates a new array populated with the results of calling a provided function on every element in the original array. It is used for transforming data without modifying the original array.

14. What is the purpose of the 'filter' method in JavaScript?

The 'filter' method creates a new array with all elements that pass the test implemented by the provided function. It is used to remove unwanted elements from an array based on a condition.

15. What is the purpose of the 'reduce' method in Java Script?

The 'reduce' method executes a reducer function on each element of the array, resulting in a single output value. It is commonly used for aggregating or accumulating values from an array.

16. What is the purpose of the 'slice' method in Java Script?

The 'slice' method returns a shallow copy of a portion of an array into a new array object, without modifying the original array. It can take two arguments: the start index and the end index (exclusive).

17. What is the purpose of the 'splice' method in Java Script?

The 'splice' method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place. It can take multiple arguments: the start index, the number of elements to remove, and any elements to add.

18. What is the purpose of the 'find' method in Java Script?

The 'find' method returns the first element in an array that satisfies a provided testing function. It is used to search for a specific element based on a condition.

19. What is the purpose of the 'findIndex' method in JavaScript?

The 'findIndex' method returns the index of the first element in an array that satisfies a provided testing function. If no elements satisfy the condition, it returns -1. It is useful for locating the position of an element based on a condition.

20. What is the purpose of the 'includes' method in JavaScript?

The 'includes' method determines whether an array includes a certain value among its entries, returning true or false. It is useful for checking the presence of an element in an array.

21. What is the purpose of the 'indexOf' method in JavaScript?

The 'indexOf' method returns the first index at which a given element can be found in an array, or -1 if it is not present. It is useful for finding the position of an element in an array.

22. What is the purpose of the 'join' method in JavaScript?

The 'join' method joins all elements of an array into a string, separated by a specified separator (default is a comma). It is useful for converting an array to a string representation.

23. What is the purpose of the 'toString' method in JavaScript?

The 'toString' method converts an array to a string representation, with elements separated by commas. It is called automatically when an array is used in a string context.

24. What is the purpose of the 'sort' method in JavaScript?

The 'sort' method sorts the elements of an array in place and returns the sorted array. By default, it sorts elements as strings, but a custom comparison function can be provided for more complex sorting.

25. What is the purpose of the 'reverse' method in JavaScript?

The 'reverse' method reverses the elements of an array in place, changing the order of the elements. It does not return a new array, but modifies the original array.

26. What is the purpose of the 'concat' method in JavaScript?

The 'concat' method is used to merge two or more arrays into a new array. It does not modify the original arrays but returns a new array containing the combined elements.

27. What is the purpose of the 'flat' method in JavaScript?

The 'flat' method creates a new array with all sub-array elements concatenated into it recursively up to the specified depth. It is useful for flattening nested arrays into a single-level array.

28. What is the purpose of the 'flatMap' method in JavaScript?

The 'flatMap' method first maps each element using a mapping function, then flattens the result into a new array. It is useful for applying a transformation to each element and flattening the result in one step.

29. What is the purpose of the 'every' method in JavaScript?

The 'every' method tests whether all elements in an array pass a provided testing function. It returns true if all elements satisfy the condition, otherwise false. It is useful for validating conditions across all elements in an array.

30. What is the purpose of the 'some' method in JavaScript?

The 'some' method tests whether at least one element in an array passes a provided testing function. It returns true if any element satisfies the condition, otherwise false. It is useful for checking if any elements meet a specific criterion.

31. What is the purpose of the 'findLast' method in JavaScript?

The 'findLast' method returns the last element in an array that satisfies a provided testing function. It is useful for searching for an element based on a condition, starting from the end of the array.

32. What is the purpose of the 'findLastIndex' method in JavaScript?

The 'findLastIndex' method returns the index of the last element in an array that satisfies a provided testing function. If no elements satisfy the condition, it returns -1. It is useful for locating the position of an element based on a condition, starting from the end of the array.

33. What is the purpose of the 'at' method in JavaScript?

The 'at' method returns the element at a specified index in an array, allowing for negative indices to access elements from the end of the array. It is useful for retrieving elements without modifying the original array.

34. What is the purpose of the 'fill' method in JavaScript?

The 'fill' method fills all elements of an array with a static value from a start index to an end index (default is the entire array). It modifies the original array and is useful for initializing or resetting array values.

35. What is the purpose of the 'copyWithin' method in JavaScript?

The 'copyWithin' method shallow copies part of an array to another location in the same array, without modifying its length. It can be used to rearrange elements within the array.

36. What is the purpose of the 'entries' method in JavaScript?

The 'entries' method returns a new Array Iterator object that contains key/value pairs for each index in the array. It is useful for iterating over both the index and value of each element in an array.

37. What is the purpose of the 'keys' method in JavaScript?

The 'keys' method returns a new Array Iterator object that contains the keys (indices) for each index in the array. It is useful for iterating over the indices of an array without accessing the values.

38. What is the purpose of the 'values' method in JavaScript?

The 'values' method returns a new Array Iterator object that contains the values for each index in the array. It is useful for iterating over the values of an array without accessing the indices.

39. What is the purpose of the 'toLocaleString' method in JavaScript?

The 'toLocaleString' method returns a string representing the array in a locale-sensitive manner, using the default locale and options. It is useful for formatting array elements based on locale-specific conventions.

40. What is the purpose of the 'toReversed' method in JavaScript?

The 'toReversed' method returns a new array with the elements in reverse order, without modifying the original array. It is useful for creating a reversed copy of an array.

41. What is the purpose of the 'toSorted' method in JavaScript?

The 'toSorted' method returns a new array with the elements sorted according to a specified compare function, without modifying the original array. It is useful for creating a sorted copy of an array.

42. What is the purpose of the 'toSpliced' method in JavaScript?

The 'toSpliced' method returns a new array with elements removed or replaced, and/or new elements added, without modifying the original array. It is useful for creating a modified copy of an array based on specified indices and values.

43. What is the purpose of the 'toString' method in JavaScript?

The 'toString' method returns a string representation of the array, with elements separated by commas. It is called automatically when an array is used in a string context, providing a simple way to convert an array to a string.

44. What is the purpose of the 'toLocaleString' method in JavaScript?

The 'toLocaleString' method returns a string representing the array in a locale-sensitive manner, using the default locale and options. It is useful for formatting array elements based on locale-specific conventions.

45. What is the purpose of the 'toReversed' method in JavaScript?

The 'toReversed' method returns a new array with the elements in reverse order, without modifying the original array. It is useful for creating a reversed copy of an array.

Programs

1. Write a program to check if a number is prime or not.

```
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; } console.log(isPrime(7)); // true
```

2. Write a program to 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
```

3. Write a program to reverse a string.

```
function reverseString(str) { return str.split('').reverse().join(''); } console.log(reverseString('hello')); // 'olleh'
```

4. Write a program to check if a string is a palindrome.

```
function isPalindrome(str) { const reversed = str.split('').reverse().join(''); return str === reversed; } console.log(isPalindrome('racecar')); // true
```

5. Write a program to find the largest number in an array.

```
function findLargest(arr) { return Math.max(...arr); } console.log(findLargest([1, 2, 3, 4, 5])); // 5
```

6. Write a program to sort an array of numbers.

```
function sortArray(arr) { return arr.sort((a, b) => a - b); } console.log(sortArray([5, 3, 8, 1, 2])); // [1, 2, 3, 5, 8]
```

7. Write a program to remove duplicates from an array.

```
function removeDuplicates(arr) { return [...new Set(arr)]; }
console.log(removeDuplicates([1, 2, 2, 3, 4, 4])); // [1, 2, 3, 4]
```

8. Write a program to find the sum of all numbers in an array.

```
function sumArray(arr) { return arr.reduce((acc, curr) => acc + curr, 0); }
console.log(sumArray([1, 2, 3, 4, 5])); // 15
```

9. Write a program to find the second largest number in an array.

```
function findSecondLargest(arr) { const uniqueArr = [...new Set(arr)]; uniqueArr.sort((a,
b) => b - a); return uniqueArr[1]; } console.log(findSecondLargest([5, 3, 8, 1, 2])); //
5
```

10. Write a program to check if two strings are anagrams.

```
function areAnagrams(str1, str2) { const sortedStr1 = str1.split('').sort().join('');
const sortedStr2 = str2.split('').sort().join(''); return sortedStr1 === sortedStr2; }
console.log(areAnagrams('listen', 'silent')); // true
```

11. Write a program to find the intersection of two arrays.

```
function intersection(arr1, arr2) { return arr1.filter(value => arr2.includes(value)); }
console.log(intersection([1, 2, 3], [2, 3, 4])); // [2, 3]
```

12. Write a program to find the union of two arrays.

```
function union(arr1, arr2) { return [...new Set([...arr1, ...arr2])]; }
console.log(union([1, 2, 3], [2, 3, 4])); // [1, 2, 3, 4]
```

13. Write a program to find the difference between two arrays.

```
function difference(arr1, arr2) { return arr1.filter(value => !arr2.includes(value)); }
console.log(difference([1, 2, 3], [2, 3, 4])); // [1]
```

14. Write a program to find the maximum and minimum numbers in an array.

```
function findMaxMin(arr) { return { max: Math.max(...arr), min: Math.min(...arr) }; }
console.log(findMaxMin([1, 2, 3, 4, 5])); // { max: 5, min: 1 }
```

15. Write a program to check if a number is even or odd.

```
function isEvenOrOdd(num) { return num % 2 === 0 ? 'Even' : 'Odd'; }
console.log(isEvenOrOdd(4)); // 'Even' console.log(isEvenOrOdd(5)); // 'Odd'
```

16. Write a program to find the Fibonacci sequence up to a given number.

```
function fibonacci(n) { const fib = [0, 1]; for (let i = 2; i < n; i++) { fib[i] = fib[i
- 1] + fib[i - 2]; } return fib.slice(0, n); } console.log(fibonacci(10)); // [0, 1, 1,
2, 3, 5, 8, 13, 21, 34]
```

17. Write a program to find the GCD (Greatest Common Divisor) of two numbers.

```
function gcd(a, b) { while (b !== 0) { const temp = b; b = a % b; a = temp; } return a; }
console.log(gcd(48, 18)); // 6
```

18. Write a program to find the LCM (Least Common Multiple) of two numbers.

```
function lcm(a, b) { return (a * b) / gcd(a, b); } console.log(lcm(4, 6)); // 12
```

19. Write a program to check if a number is a perfect square.

```
function isPerfectSquare(num) { const sqrt = Math.sqrt(num); return sqrt * sqrt === num;
} console.log(isPerfectSquare(16)); // true console.log(isPerfectSquare(20)); // false
```

20. Write a program to find the sum of digits of a number.

```
function sumOfDigits(num) { return num.toString().split('').reduce((acc, digit) => acc +
Number(digit), 0); } console.log(sumOfDigits(1234)); // 10
```

21. Write a program to check if a number is Armstrong number.

```
function isArmstrong(num) { const digits = num.toString().split('').map(Number); const
sum = digits.reduce((acc, digit) => acc + Math.pow(digit, digits.length), 0); return sum
=== num; } console.log(isArmstrong(153)); // true console.log(isArmstrong(123)); // false
```

22. Write a program to find the length of a string without using the length property.

```
function stringLength(str) { let count = 0; for (let char of str) { count++; } return
count; } console.log(stringLength('hello')); // 5
```

23. Write a program to convert a string to title case.

```
function toTitleCase(str) { return str .toLowerCase() .split(' ') .map(word =>
word.charAt(0).toUpperCase() + word.slice(1)) .join(' '); }
console.log(toTitleCase('hello world')); // 'Hello World'
```

24. Write a program to find the longest word in a sentence.

```
function findLongestWord(sentence) { const words = sentence.split(' '); return
words.reduce((longest, current) => current.length > longest.length ? current : longest,
''); } console.log(findLongestWord('The quick brown fox jumps over the lazy dog')); //
'jumps'
```

25. Duplicate the Array in JS

```
function duplicateArray(arr) { return arr.map(item => item); }
console.log(duplicateArray([1, 2, 3])); // [1, 2, 3]
```

26. Write a program to find the common elements in two arrays.

```
function commonElements(arr1, arr2) { return arr1.filter(value => arr2.includes(value));
} console.log(commonElements([1, 2, 3], [2, 3, 4])); // [2, 3]
```

27. Write a program to find the unique elements in an array.

```
function uniqueElements(arr) { return [...new Set(arr)]; } console.log(uniqueElements([1,
2, 2, 3, 4, 4])); // [1, 2, 3, 4]
```

28. Display Data Using map Function

```
const fruits = ["apple", "banana", "cherry"]; fruits.map((fruit, index) => {
console.log(`${index + 1}. ${fruit}`); });
```

29. Find the Error in async/await

```
async function fetchData() { let response = await fetch('https://api.example.com/data');
let data = await response.json(); console.log(data); } fetchData();
```

30. Counter Using useReducer

```
function counterReducer(state, action) { switch (action.type) { case 'increment': return
{ count: state.count + 1 }; case 'decrement': return { count: state.count - 1 }; default:
throw new Error(); } } const [state, dispatch] = useReducer(counterReducer, { count: 0
}); console.log(state.count); // Initial count
```

31. Form Handling Using useReducer

```
function formReducer(state, action) { switch (action.type) { case 'update': return {
...state, [action.field]: action.value }; default: throw new Error(); } } const
[formState, dispatch] = useReducer(formReducer, { name: '', email: '' });
console.log(formState); // Initial form state
```

32. What is the difference between var, let, and const?

```
function testVar() { var x = 1; if (true) { var x = 2; console.log(x); // 2 }
console.log(x); // 2 } function testLet() { let y = 1; if (true) { let y = 2;
console.log(y); // 2 } console.log(y); // 1 }
```

33. What is hoisting in JavaScript

```
console.log(a); // undefined (not error) var a = 10; sayHi(); // Works because function
declarations are hoisted function sayHi() { console.log("Hello"); }
```

34. What is the difference between synchronous and asynchronous JavaScript?

Synchronous: Code runs line by line.
Asynchronous: Code runs in the background (e.g., setTimeout, fetch).

```
console.log("Start"); setTimeout(() => { console.log("Async Code"); }, 1000);
console.log("End"); // Output: Start → End → Async Code
```

35. What is event bubbling?

When an event starts from the innermost element and bubbles up to its ancestors.

Click me

36. What is event delegation?

Event delegation is a technique where a single event listener is added to a parent element to manage events for multiple child elements, improving performance and reducing memory usage.

```
document.getElementById('parent').addEventListener('click', function(event) { if
(event.target.matches('.child')) { console.log('Child clicked:', event.target); } });
```

37. What is the purpose of the 'this' keyword in JavaScript?

The 'this' keyword refers to the context in which a function is called. It can refer to the global object, an object instance, or be undefined in strict mode. Its value depends on how a function is invoked.

```
const obj =
{ name: 'JavaScript', greet: function() { console.log('Hello from', this.name); } };
obj.greet(); // Hello from JavaScript
```

38. What is the purpose of the 'new' keyword in JavaScript?

The 'new' keyword is used to create an instance of a constructor function. It initializes a new object, sets the prototype of the new object to the constructor's prototype, and returns the new object.

```
function
Person(name) { this.name = name; } const person1 = new Person('Alice');
console.log(person1.name); // Alice
```