Javascript Theoretical Interview Questions with Answers

1. What is JavaScript?

Answer:

JavaScript is a lightweight, interpreted programming language used for web development. It allows you to add interactivity, dynamic content, and behavior to websites.

---

2. What are the data types in JavaScript?

Answer:

JavaScript has 7 primitive data types:

- String
- Number
- Boolean
- Undefined
- Null
- BigInt
- Symbol

And one non-primitive data type:

- Object
- Arrays
- Functions

---

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

Answer:

- var: Function-scoped, can be redeclared and updated.
- let: Block-scoped, cannot be redeclared but can be updated.
- const: Block-scoped, cannot be redeclared or updated.

---

4. What is hoisting in JavaScript?

Answer:

Hoisting is a JavaScript mechanism where variable and function declarations are moved to the top of their scope before code execution. Only declarations are hoisted, not initializations.

---

5. What is the difference between == and ===?

Answer:

- == checks for equality after type coercion.
- === checks for strict equality without type coercion.

---

6. What is closure in JavaScript?

Answer:

A closure is a function that has access to its outer function's scope even after the outer function has returned. It is created every time a function is created.

---

7. What is an IIFE?

Answer:

An IIFE (Immediately Invoked Function Expression) is a function that is executed immediately after it is defined.

Example:

```
 (function() {
   console.log("IIFE");
})();
```

---

8. What is the this keyword in JavaScript?

Answer:

The this keyword refers to the object that the function is a property of. Its value depends on how the function is called.

---

9. What is the difference between null and undefined?

Answer:

- undefined means a variable has been declared but not assigned a value.
- null is an assignment value that represents no value or no object.

---

10. What is the difference between synchronous and asynchronous code?

Answer:

- Synchronous code executes line by line, blocking further execution until the current operation is completed.
- Asynchronous code allows other operations to run while waiting for the current operation to complete.

---

11. What are promises in JavaScript?

Answer:

A promise is an object that represents the eventual completion (or failure) of an asynchronous operation and its resulting value. It has three states: pending, fulfilled, and rejected.

---

12. What is async/await?

Answer:

async/await is syntactic sugar for working with promises. It allows you to write asynchronous code that looks like synchronous code.

---

13. What is the difference between call, apply, and bind?

Answer:

- call: Invokes a function with a specific this value and arguments provided individually.
- apply: Similar to call, but arguments are provided as an array.
- bind: Returns a new function with a specific this value and arguments.

---

14. What is event bubbling?

Answer:

Event bubbling is a process where an event propagates from the target element up to the root of the DOM tree.

---

15. What is event delegation?

Answer:

Event delegation is a technique where you add a single event listener to a parent element to handle events for all its child elements.

---

16. What is the DOM?

Answer:

The DOM (Document Object Model) is a programming interface for HTML and XML documents. It represents the structure of a document as a tree of objects.

---

17. What is JSON?

Answer:

JSON (JavaScript Object Notation) is a lightweight data interchange format. It is easy for humans to read and write and for machines to parse and generate.

---

18. What is the difference between slice and splice?

Answer:

- slice: Returns a shallow copy of a portion of an array without modifying the original array.
- splice: Changes the contents of an array by removing or replacing existing elements and/or adding new elements.

---

19. What is the purpose of use strict?

Answer:

use strict enforces stricter parsing and error handling in your code. It helps you write cleaner and more secure JavaScript.

---

20. What is the difference between forEach and map?

Answer:

• forEach: Executes a provided function once for each array element. It does not return a new array.

• map: Creates a new array by applying a function to each element of the original array.

---

21. What is a callback function?

Answer:

A callback function is a function passed as an argument to another function and is executed after some operation is completed.

---

22. What is the difference between localStorage and sessionStorage?

Answer:

• localStorage: Stores data with no expiration date.

• sessionStorage: Stores data for one session (data is lost when the tab is closed).

---

23. What is the purpose of the fetch API?

Answer:

The fetch API is used to make network requests (e.g., to fetch resources from a server). It returns a promise that resolves to the response of the request.

---

24. What is the difference between undefined and not defined?

Answer:

- undefined: A variable is declared but not assigned a value.
- not defined: A variable is not declared at all.

---

25. What is the purpose of the typeof operator?

Answer:

The typeof operator returns a string indicating the type of the operand.

---

26. What is the difference between null and undefined?

Answer:

- null: Represents an intentional absence of any object value.
- undefined: Represents a variable that has been declared but not assigned a value.

---

27. What is the purpose of the Array.reduce() method?

Answer:

The reduce() method executes a reducer function on each element of the array, resulting in a single output value.

---

28. What is the difference between let and const?

Answer:

- let: Allows reassignment of the variable.
- const: Does not allow reassignment of the variable.

---

29. What is the purpose of the Array.filter() method?

Answer:

The filter() method creates a new array with all elements that pass the test implemented by the provided function.

---

30. What is the difference between == and ===?

Answer:

- ==: Compares values after type coercion.
- ===: Compares values without type coercion (strict equality).

---

list of 30 commonly asked Data Structures and Algorithms (DSA) questions in JavaScript with answers

Array-Based Questions

1. **Reverse an Array**
2. **Find Maximum and Minimum Element in an Array**
3. **Find the Second Largest Element**
4. **Check if an Array is Sorted**
5. **Remove Duplicates from an Array**
6. **Rotate an Array to the Right by k Steps**
7. **Move All Zeros to the End**
8. **Find Missing Number in an Array (1 to n)**
9. **Find the Intersection of Two Arrays**
10. **Find the Union of Two Arrays**

String-Based Questions

11. **Reverse a String**
12. **Check if a String is a Palindrome**
13. **Count the Occurrences of Characters**
14. **Check if Two Strings are Anagrams**

15. **Find the First Non-Repeating Character**

Searching and Sorting

16. **Linear Search**
17. **Binary Search (on a sorted array)**
18. **Bubble Sort**
19. **Selection Sort**
20. **Insertion Sort**

Recursion

21. **Factorial of a Number**
22. **Fibonacci Sequence**
23. **Power of a Number (x^n)**
24. **Sum of Digits of a Number**
25. **Generate All Subsequences of a String**

Stack and Queue

26. **Implement a Stack using an Array**
27. **Implement a Queue using an Array**

Miscellaneous

28. **Check if a Number is Prime**
29. **Find GCD of Two Numbers (Euclidean Algorithm)**
30. **Merge Two Sorted Arrays**

______________________SOLUTIONS_______________________

// 1. Reverse an Array

```
function reverseArray(arr) {

  let reversed = [];

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

    reversed[reversed.length] = arr[i];

  }

  return reversed;

}
```

// 2. Find Max and Min

```
function findMinMax(arr) {
   let min = arr[0], max = arr[0];
   for (let i = 1; i < arr.length; i++) {
      if (arr[i] < min) min = arr[i];
      if (arr[i] > max) max = arr[i];
   }
   return { min, max };
}
```

// 3. Find Second Largest

```
function secondLargest(arr) {
   let max = -Infinity, secondMax = -Infinity;
   for (let i = 0; i < arr.length; i++) {
      if (arr[i] > max) {
         secondMax = max;
         max = arr[i];
      } else if (arr[i] > secondMax && arr[i] !== max) {
         secondMax = arr[i];
      }
   }
   return secondMax === -Infinity ? -1 : secondMax;
}
```

// 4. Check if Array is Sorted

```
function isSorted(arr) {
  for (let i = 1; i < arr.length; i++) {
    if (arr[i] < arr[i - 1]) return false;
  }
  return true;
}
```

// 5. Remove Duplicates

```
function removeDuplicates(arr) {
  let unique = [];
  for (let i = 0; i < arr.length; i++) {
    let found = false;
    for (let j = 0; j < unique.length; j++) {
      if (arr[i] === unique[j]) {
        found = true;
        break;
      }
    }
    if (!found) unique[unique.length] = arr[i];
  }
  return unique;
}
```

// 6. Rotate Array Right by k Steps

```
function rotateArray(arr, k) {
  k = k % arr.length;
```

```
    let n = arr.length;
    let rotated = [];
    for (let i = 0; i < n; i++) {
        rotated[(i + k) % n] = arr[i];
    }
    return rotated;
}
```

// 7. Move Zeros to End

```
function moveZerosToEnd(arr) {
    let nonZeroIndex = 0;
    for (let i = 0; i < arr.length; i++) {
        if (arr[i] !== 0) {
            arr[nonZeroIndex++] = arr[i];
        }
    }
    while (nonZeroIndex < arr.length) {
        arr[nonZeroIndex++] = 0;
    }
    return arr;
}
```

// 8. Missing Number (1 to n)

```
function findMissingNumber(arr, n) {
    let total = n * (n + 1) / 2;
    let sum = 0;
    for (let i = 0; i < arr.length; i++) {
        sum += arr[i];
    }
    return total - sum;
}
```

// 9. Intersection of Two Arrays

```
function arrayIntersection(arr1, arr2) {
    let result = [];
    for (let i = 0; i < arr1.length; i++) {
        for (let j = 0; j < arr2.length; j++) {
            if (arr1[i] === arr2[j]) {
                result[result.length] = arr1[i];
                arr2[j] = undefined;
                break;
            }
        }
    }
    return result;
}
```

// 10. Union of Two Arrays

```
function arrayUnion(arr1, arr2) {
    let result = arr1.slice();
    for (let i = 0; i < arr2.length; i++) {
        let found = false;
        for (let j = 0; j < arr1.length; j++) {
            if (arr2[i] === arr1[j]) {
                found = true;
                break;
            }
        }
        if (!found) result[result.length] = arr2[i];
    }
    return result;
}
```

// 11. Reverse a String

```
function reverseString(str) {
    let reversed = '';
    for (let i = str.length - 1; i >= 0; i--) {
        reversed += str[i];
    }
    return reversed;
}
```

// 12. Palindrome Check

```
function isPalindrome(str) {
    let start = 0, end = str.length - 1;
    while (start < end) {
        if (str[start++] !== str[end--]) return false;
    }
    return true;
}
```

// 13. Count Character Occurrences

```
function charFrequency(str) {
    let freq = {};
    for (let i = 0; i < str.length; i++) {
        if (!freq[str[i]]) freq[str[i]] = 1;
        else freq[str[i]]++;
    }
    return freq;
}
```

// 14. Check for Anagram

```
function isAnagram(str1, str2) {
    if (str1.length !== str2.length) return false;
    let freq = {};
    for (let i = 0; i < str1.length; i++) {
        freq[str1[i]] = (freq[str1[i]] || 0) + 1;
        freq[str2[i]] = (freq[str2[i]] || 0) - 1;
    }
    for (let key in freq) {
```

```
    if (freq[key] !== 0) return false;
  }
  return true;
}
```

// 15. First Non-Repeating Character

```
function firstNonRepeatingChar(str) {
  for (let i = 0; i < str.length; i++) {
    let isUnique = true;
    for (let j = 0; j < str.length; j++) {
      if (i !== j && str[i] === str[j]) {
        isUnique = false;
        break;
      }
    }
    if (isUnique) return str[i];
  }
  return null;
}
```

// 16. Linear Search

```
function linearSearch(arr, target) {
  for (let i = 0; i < arr.length; i++) {
    if (arr[i] === target) return i;
  }
  return -1;
}
```

// 17. Binary Search (Sorted Array)

```
function binarySearch(arr, target) {
    let left = 0, right = arr.length - 1;
    while (left <= right) {
        let mid = Math.floor((left + right) / 2);
        if (arr[mid] === target) return mid;
        else if (arr[mid] < target) left = mid + 1;
        else right = mid - 1;
    }
    return -1;
}
```

// 18. Bubble Sort

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

// 19. Selection Sort

```
function selectionSort(arr) {
    let n = arr.length;
    for (let i = 0; i < n - 1; i++) {
        let minIndex = i;
        for (let j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) minIndex = j;
        }
        let temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
    return arr;
}
```

// 20. Insertion Sort

```
function insertionSort(arr) {
    for (let i = 1; i < arr.length; i++) {
        let key = arr[i];
        let j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
    return arr;
}
```

// 21. Factorial

```
function factorial(n) {
    return n === 0 ? 1 : n * factorial(n - 1);
}
```

// 22. Fibonacci Sequence

```
function fibonacci(n) {
    if (n <= 1) return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

// 23. Power (x^n)

```
function power(x, n) {
    if (n === 0) return 1;
    return x * power(x, n - 1);
}
```

// 24. Sum of Digits

```
function sumOfDigits(n) {
  if (n === 0) return 0;
  return n % 10 + sumOfDigits(Math.floor(n / 10));
}
```

// 25. Subsequences of String

```
function subsequences(str, index = 0, current = '') {
  if (index === str.length) {
    console.log(current);
    return;
  }
  subsequences(str, index + 1, current + str[index]);
  subsequences(str, index + 1, current);
}
```

// 26. Stack Implementation

```
class Stack {
  constructor() {
    this.stack = [];
  }
  push(val) {
    this.stack[this.stack.length] = val;
  }
  pop() {
    if (this.stack.length === 0) return null;
    let val = this.stack[this.stack.length - 1];
    this.stack.length--;
    return val;
  }
}
```

// 27. Queue Implementation

```
class Queue {
  constructor() {
    this.queue = [];
  }
  enqueue(val) {
    this.queue[this.queue.length] = val;
  }
  dequeue() {
    if (this.queue.length === 0) return null;
    let val = this.queue[0];
    for (let i = 0; i < this.queue.length - 1; i++) {
      this.queue[i] = this.queue[i + 1];
    }
    this.queue.length--;
    return val;
  }
}
```

// 28. Prime Check

```
function isPrime(n) {
  if (n < 2) return false;
  for (let i = 2; i <= Math.sqrt(n); i++) {
    if (n % i === 0) return false;
  }
  return true;
}
```

// 29. GCD (Euclidean Algorithm)

```
function gcd(a, b) {
  while (b !== 0) {
    let temp = b;
    b = a % b;
    a = temp;
  }
  return a;
}
```

// 30. Merge Two Sorted Arrays

```
function mergeSortedArrays(arr1, arr2) {
  let i = 0, j = 0, result = [];
  while (i < arr1.length && j < arr2.length) {
    if (arr1[i] < arr2[j]) result[result.length] = arr1[i++];
    else result[result.length] = arr2[j++];
  }
  while (i < arr1.length) result[result.length] = arr1[i++];
  while (j < arr2.length) result[result.length] = arr2[j++];
  return result;
}
```