**Front-End Interview Questions**

1. **Order of the console.log**

function main(){

console.log('A');

setTimeout(

function print(){ console.log('B'); }

,0);

console.log('C');

}

main();

**Answer**: A,C and B

1. **Output of the below program**

function foo() {

return

{

message: "Hello World"

};

}

console.log(foo());

**Answer**: Undefined

1. **Output of the below program**

var a = 5;

(function() {

console.log(a);

var a = 10;

})();

**Output**: undefined

Explanation: This code creates a new function and immediately calls it. Within the function, a is declared later in the code using var, which means it is hoisted to the top of the function scope. Therefore, the console.log() statement refers to the hoisted variable, which has not yet been assigned a value, resulting in undefined as the output.

1. **Convert below object to array**

**let** obj = { id: "1", name: "user22", age: "26", work: "programmer" };

**Answer:**

//Method 1: Convert the keys to Array using - Object.keys()

console.log(Object.keys(obj));

["id", "name", "age", "work"]

// Method 2 Converts the Values to Array using - Object.values()

console.log(Object.values(obj));

["1", "user22r", "26", "programmer"]

// Method 3 Converts both keys and values using - Object.entries()

console.log(Object.entries(obj));

[["id", "1"],["name", "user22"],["age", "26"],["work", “programmer"]]

1. **Write a Higher-Order Component (HOC) called**withAuthentication**that wraps a component and provides an**isAuthenticated**prop indicating whether the user is authenticated or not. Assume there’s an**authenticate**function that returns a boolean value.**

**Answer**:

import React from 'react';

const withAuthentication = (WrappedComponent) => {

return function WithAuthentication(props) {

const isAuthenticated = authenticate();

return <WrappedComponent {...props} isAuthenticated={isAuthenticated} />;

};

};

export default withAuthentication;

1. **Take below two variables as input and write a javascript program to produce mentioned output.**

**var array1 = [["Hyd", 1], ["Hyd", 2], ["Blr", 4], ["Hyd", 4]]**

**var array2 = [["Hyd", 1], ["Hyd", 2], ["Blr", 6], ["Hyd", 6], ["del", 8]]**

**Output: {Hyd: 3, Blr: 5, del: 8}**

**Answer:**

**var final = {};**

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

**var avg = 0;**

**var count = 0;**

**for (let j = 0; j < array.length; j++) {**

**if (array[i][0] === array[j][0]) {**

**avg += array[j][1];**

**count = count + 1;**

**}**

**}**

**console.log(array[i][0], avg, count)**

**final = {...final, [array[i][0]]: Math.round(avg/count)}**

**}**

1. **Execute below program, currently this program will throw an error like name.call is not a function. Please resolve this error.**

**function Name (name) {**

**this.name = name**

**}**

**Name.call = function () {**

**console.log(this.name + ' say Hello')**

**}**

**let name = new Name('Sridhar')**

**name.call()**

**Answer: Name.prototype.call, In JavaScript everything is object, object and methods inherits from prototype.**

1. **Remove duplicates from given array**

**var dup = [1, 1, 2, -2, -2, 4, 5, 4, 7, 8, 7, 7, 71, 3, 6, 1, 8, 7, 3, 19, 19, 19];**

**Answer:**

**const findDup = dup.filter((val, key, arr) => arr.indexOf(val) === key)**

**console.log(findDup)**

1. **Write program to print word counts from given array**

**let input = ['How is your health', 'what is your native', 'How is your family', 'where your friend', 'is is']**

**Output: is: 5, How: 2**

**Answer:**

**var zz = [];**

**var yy = {};**

**var key = '';**

**var max = 0;**

**for (let val of repWords) {**

**zz = [...zz, ...(val.split(' '))]**

**}**

**for (let val of zz) {**

**if (!yy[val]) {**

**yy = {...yy, [val]: 1};**

**} else {**

**yy = {...yy, [val]: yy[val] + 1};**

**if (yy[val] > max) {**

**max = yy[val];**

**key = val;**

**}**

**}**

**}**

**console.log(key+': '+ max);**

1. **Write JavaScript program it should print 15 from given input**

**multiply (1)(2)**

**Output:**

**function multiply(a) {**

**return (b) => {**

**return a \* b**

**}**

**}**

1. **Write a program to print common letters from given array**

**let arr = ["Avenger","AvSall","Avtar"]**

**Output: Av**

**Answer:**

**let a = '';**

**let b = arr[0].split('');**

**for(vals of b) {**

**let count = 0;**

**for(val of arr) {**

**if (val.indexOf(vals) === b.indexOf(vals)) {**

**count++;**

**}**

**}**

**if (count === arr.length && !a.includes(vals)) a = a + vals;**

**}**

**console.log(a);**

1. **Write a program to print IBM from given string.**

**var string = 'Internation Business Machines'**

**Answer: string.split(' ').map(x => x.charAt(0)).join('')**

1. **https://playcode.io/**

**Javascript Playground (Sandbox, Repl)**

**Javascript Playground (Sandbox, Repl)**

**Scenario Based - Async Data Fetch and Rendering**

**Create a React component that fetches data from a given API endpoint and renders it as a list.**

**The API endpoint returns an array of objects, each containing an name and a email .**

**Implement the React component to fetch the data from the API and display it as a list.**

**API Endpoint: https://jsonplaceholder.typicode.com/users**

**Scenario Based - Managing Focus with useRef**

**Create a React component that includes an input field and a button. When the button is clicked, it should focus on the input field.**

**Implement this functionality using the useRef hook.**

1. **Identify the performance bottleneck in this code snippet?**

import React from 'react';  
const MyComponent = () => {  
  const data = fetchData(); // Assume this function fetches data from an API

  return (  
<div>  
      {data.map((item) => (  
<p key={item.id}>{item.name}</p>  
      ))}  
</div>  
  );  
};

export default MyComponent;

1. What will be the value of ‘**typeof val**’ in both these questions,

let val = [];  
typeof val;

let val = {}  
typeof val;

1. Write the program to list the active use whose age < 50 (Do NOT use foreach, rather use higher order function – Filter, map, filter etc.)

const users = [  
    {  
        id:1,  
        name:"Jack",  
        isActive:true,  
      age:50  
    },  
    {  
        id:2,  
        name:"John",  
        isActive:true,  
      age:11  
    },  
    {  
        id:3,  
        name:"Mike",  
        isActive:false  
    }  
    ]

**Node Js:**

**1. How do you decide whether to use Node.js Express or Node.js Nest based on the project size?**

**2. Explain 0auth2.0 and its benefits**

**3. Compare service-oriented architecture (SOA) and microservice architecture (MSA)**

**4. What does “low in coupling and high in cohesion” mean in backend development?**

**5. How do you ensure the security of backend systems?**

**6. What’s the difference between using PostgreSQL and MongoDB for Node.js server projects?**

**7. How do you decide when to implement caching in your backend systems?**

**8. How does the architecture of a Nest.js application differ from other Node.js frameworks like Express.js?**

**9. How can you test Nest.js applications?**

**10. What is an interceptor in Nest.js?**

**11. Can you describe the role of modules in a Nest.js project?**

**12. How would you describe a good approach to deploying a backend project?**

**13. Can you give an example of a project that required WebSocket communication?**

**14. List pros and cons of using GraphQL vs REST API approaches**

**15. How do you ensure that your Nest.js applications are scalable and maintainable? What techniques or best practices do you follow?**

**Java Script:**

**1. Can you explain how JavaScript handles asynchronous operations?**

**2. Can you explain closures in JavaScript?**

**3. How does JavaScript's event loop work?**

**4. What is the difference among "let", "const", and "var" in JavaScript**

**5. Can you explain how prototypal inheritance works in JavaScript?**

**6. What is the purpose of JavaScript's "this" keyword?**

**7. Can you explain how hoisting works in JavaScript?**

**8. Can you explain the concept of promises in JavaScript?**

**9. Explain event delegation in JavaScript**

**10. Describe the difference between an arrow function and a regular function in JavaScript**

**11. Explain memoization in JavaScript**

**React Js**

**1. How do you create refs in React?**

**2. Name three advantages of using React Hooks.**

**3. Outline the different stages of the React.js lifecycle.**

**4. Explain what a higher-order component.**

**5. What is the difference between class and functional components?**

**6. Name an architectural difference between React and Angular.**

**7. Explain what a controlled component and uncontrolled component.**

**8. What are package managers in React.js?**

**9. What is prop drilling.**

**10. Explain Strict Mode.**

**11. Explain what the virtual DOM.**

**12. What is pure function.**

**13. what is JSX.**

**14. What is Flux architecture.**

**15. What are bundlers in React.js?**

**16. Explain what is Redux?**

**17. Explain Context API?**

**18. What is middleware?**

**19. Explain what is Webpack?**

**20. What is use of generator function in REDUX saga?**

**Client Interview questions:**

**1. let sampleArray = [1,4,5,6,7];**

**const result = sampleArray.some((ele)=>ele>5);**

**console.log(result);**

**What is the output?**

**Answer:**

**True(some will return Boolean)**

**==============================================**

**2. Identify and fix any performance bottlenecks in the following React component:**

**import React from 'react';**

**const MyComponent = () => {**

**const data = fetchData(); // Assume this function fetches data from an API**

**return (**

**<div>**

**{data.map((item) => (**

**<p key={item.id}>{item.name}</p>**

**))}**

**</div>**

**);**

**};**

**export default MyComponent;**

**API end point**

**https://api.github.com/users/gaearon/repos**

**==============================================**

**Refactored code:**

**import React, { useState, useEffect } from 'react';**

**const MyComponent = () => {**

**const [data, setData] = useState([]); // State to hold the data**

**const [loading, setLoading] = useState(true); // State to handle loading status**

**const [error, setError] = useState(null); // State to handle errors**

**useEffect(() => {**

**const fetchData = async () => {**

**try {**

**const response = await fetch('https://api.github.com/users/gaearon/repos');**

**if (!response.ok) {**

**throw new Error('Network response was not ok');**

**}**

**const result = await response.json();**

**setData(result);**

**} catch (error) {**

**setError(error);**

**} finally {**

**setLoading(false);**

**}**

**};**

**fetchData();**

**}, []); // Empty dependency array means this effect runs once when the component mounts**

**if (loading) return <p>Loading...</p>;**

**if (error) return <p>Error: {error.message}</p>;**

**return (**

**<div>**

**{data.map((item) => (**

**<p key={item.id}>{item.name}</p>**

**))}**

**</div>**

**);**

**};**

**export default MyComponent;**

**1. What is the output?**

**let sampleOb = [{**

**name:'AA',**

**passed: 'true'**

**},**

**{name:'BB',**

**passed:'false'}**

**];**

**const res = sampleOb.every((ele)=>ele.passed === 'true');**

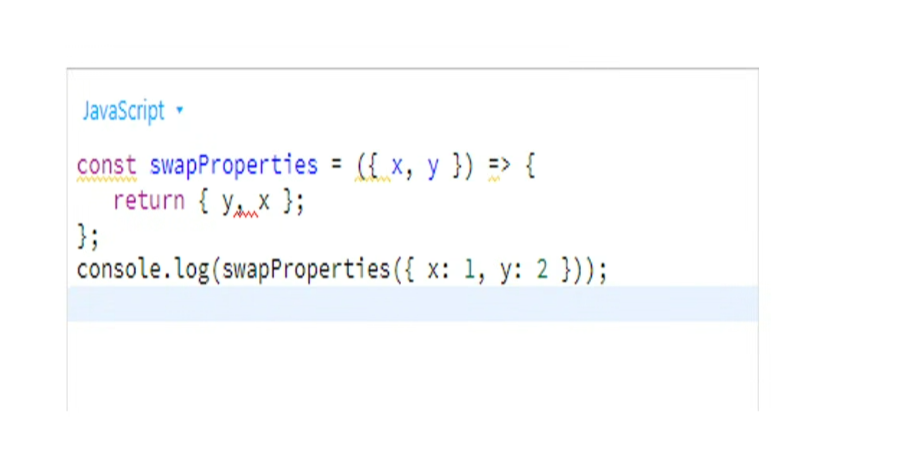
**console.log("every", res);**

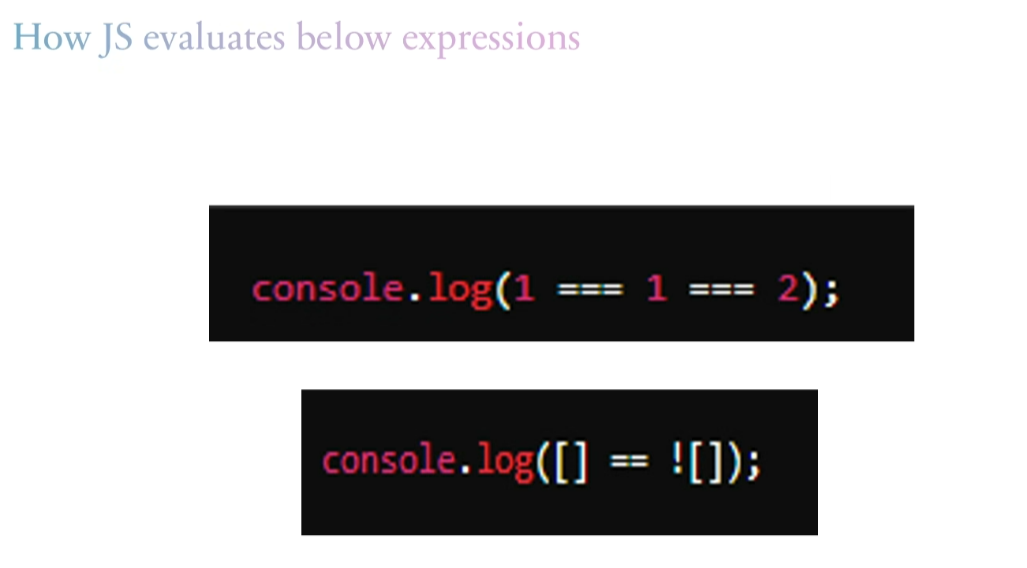
**has context menu**

**2. What is the minimum number and occurrence of the minimum number**

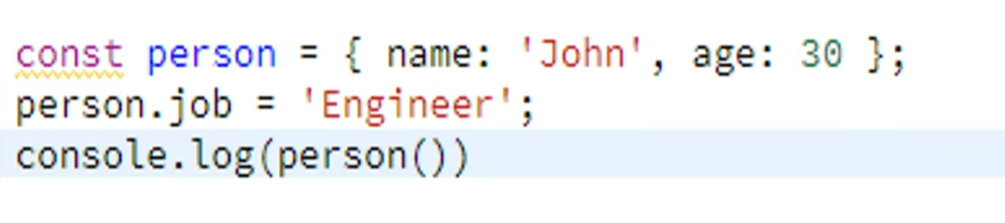
**const arr = [2,5,6,7,3,2,1,3,4,5,6,1,3,1,3,1]**

Question # 1: What would print on console: Is there any error in any of the statement



Question # 2: What would below statement print?

Question # 3: What would print on console:



Question # 4: What will be the output ?

A group of black and white text

Description automatically generated

Question # 5: How would the execution control flow?A computer code with text

Description automatically generated