🔥 Ultimate JavaScript Mock Interview Questions

JavaScript Basics

1. What are the different data types in JavaScript?
2. What is the difference between var, let, and const?
3. What is hoisting in JavaScript?
4. What is the difference between == and ===?
5. What is the typeof operator, and how does it work?
6. What is the difference between null and undefined?
7. What is the difference between pass by value and pass by reference in JavaScript?
8. What is NaN, and how do you check if a value is NaN?
9. What is the difference between primitive and reference types in JavaScript?
10. What happens when you use console.log() before declaring a variable?

Functions & Scope

11. What are function expressions vs. function declarations?
12. What is a higher-order function? Provide an example.
13. What is closure in JavaScript?
14. What is the lexical scope?
15. What is an Immediately Invoked Function Expression (IIFE)?
16. How does JavaScript handle asynchronous execution inside a loop?
17. What is the difference between call(), apply(), and bind()?
18. What happens when you return a function inside another function?
19. What is function currying?
20. What is a pure function?

By- Sanskar Srivastava (LinkedIn)

Objects & Prototypes

21. What is prototypal inheritance in JavaScript?
22. How do you create an object without a prototype?
23. What is the difference between Object.create() and Object.assign()?
24. What is the difference between shallow copy and deep copy?
25. What are getters and setters in JavaScript objects?
26. What is the __proto__ property in JavaScript?
27. How do you check if an object is empty in JavaScript?
28. How do you freeze an object in JavaScript?
29. What is the difference between Object.freeze() and Object.seal()?
30. What is the difference between for...in and for...of loops?

Arrays & Iteration

31. What is the difference between map(), forEach(), filter(), and reduce()?
32. How do you remove duplicates from an array?
33. What is the best way to loop over an array in JavaScript?
34. How does the .reduce() function work?
35. How do you shuffle an array randomly?
36. What is the difference between slice() and splice()?
37. What does the .sort() function do in JavaScript?
38. How does JavaScript handle sorting numbers in an array?
39. What is the difference between find() and findIndex()?
40. What is the best way to flatten an array in JavaScript?

Asynchronous JavaScript

41. What is the event loop in JavaScript?
42. What are Promises in JavaScript?

By- Sanskar Srivastava (LinkedIn)

43. What is the difference between async and defer attributes in script tags?
44. What is async/await, and how does it work?
45. What is the difference between Promise.all(), Promise.allSettled(), Promise.race(), and Promise.any()?
46. How do you handle errors in Promises?
47. What is a microtask queue in JavaScript?
48. How does setTimeout() work inside an async function?
49. What is the difference between synchronous and asynchronous code?
50. What are Web Workers in JavaScript?

Advanced JavaScript

51. What is the difference between new Object() and {}?
52. How does garbage collection work in JavaScript?
53. What are WeakMap and WeakSet in JavaScript?
54. What are modules in JavaScript?
55. What is the difference between import and require()?
56. What are JavaScript Symbols?
57. What are tagged template literals?
58. What are generator functions in JavaScript?
59. What is tail call optimization?
60. What is the difference between shallow copy and deep copy in JavaScript?

DOM & Browser APIs

61. How do you select elements in the DOM?
62. What is event delegation?
63. How does addEventListener() work?
64. What is localStorage, sessionStorage, and cookies?

By- Sanskar Srivastava (LinkedIn)

65. How do you make an HTTP request in JavaScript?
66. What is the difference between innerHTML, textContent, and innerText?
67. What is the difference between bubbling and capturing in JavaScript events?
68. What is the MutationObserver API?
69. What is the difference between setTimeout() and requestAnimationFrame()?
70. How does debounce and throttle work?

Performance & Optimization

71. What are the different ways to optimize JavaScript performance?
72. What is memoization in JavaScript?
73. How does JavaScript handle memory leaks, and how can you prevent them?
74. What are different types of caching strategies?
75. What are Service Workers and how do they improve performance?
76. How do you optimize JavaScript for faster page loads?
77. What are WebSockets, and how do they work?
78. What is the difference between Long Polling, WebSockets, and Server-Sent Events?
79. What is the difference between deep cloning and shallow cloning an object?
80. How do you optimize loops in JavaScript?

🔥 React.js Mock Interview Questions (Extensive List)

React Basics

1. What is React, and why is it used?
2. What are the main features of React?
3. What is JSX, and why is it used in React?
4. How is React different from other front-end frameworks like Angular and Vue?
5. What are props in React? How do they work?
6. What is state in React? How is it different from props?

7. Can you change the state directly in React? Why or why not?
8. What is the difference between function components and class components?
9. How does React handle one-way data flow?
10. What are default props in React? How do you define them?
11. What is the significance of key in React lists?
12. How does React optimize updates in the Virtual DOM?
13. Can React work without JSX? If yes, how?
14. What are fragments in React, and why are they useful?
15. What is the purpose of React Portals?
16. What is the difference between controlled and uncontrolled components?
17. How do you handle forms in React?
18. How do you implement conditional rendering in React?
19. What is the role of map() function in React?
20. How do you pass functions as props in React?

React Rendering and Performance

21. What is the Virtual DOM, and how does it improve performance?
22. How does React's reconciliation process work?
23. What is React Fiber, and how does it improve performance?
24. What are synthetic events in React, and how do they work?
25. How does React handle re-renders?
26. What are the common causes of unnecessary re-renders in React?
27. How can you prevent unnecessary re-renders in React?
28. What is the significance of React.memo()?
29. What is the difference between React.memo() and useMemo()?
30. What are the different ways to optimize performance in a React application?
31. What are event handlers in React, and how do they work?

By- Sanskar Srivastava (LinkedIn)

32. How does React handle async operations in event handlers?
33. Why is setState asynchronous in React?
34. What are the effects of calling setState multiple times in a single function?
35. How does batching work in React, and why is it useful?
36. What is the difference between controlled and uncontrolled inputs in React?
37. How does React handle list rendering efficiently?
38. What are the disadvantages of using indexes as keys in React lists?

React Component Lifecycle (Class Components)

39. What are the different phases of the React component lifecycle?
40. What lifecycle methods are available in class components?
41. What is componentDidMount(), and when is it used?
42. What is componentDidUpdate(), and how does it work?
43. What is componentWillUnmount(), and why is it important?
44. What is shouldComponentUpdate(), and how does it work?
45. How do lifecycle methods compare to hooks in functional components?
46. What is the role of getDerivedStateFromProps()?
47. How does componentDidCatch() help in error handling?
48. What are the best practices when using lifecycle methods in class components?

React Hooks

49. What are React Hooks, and why were they introduced?
50. What are the rules of Hooks in React?
51. What is useState(), and how does it work?
52. How does useEffect() work in React?
53. What are the common use cases of useEffect()?
54. What is the difference between useEffect() and lifecycle methods?

By- Sanskar Srivastava (LinkedIn)

55. How do you handle cleanup inside useEffect()?
56. What is useRef(), and when should it be used?
57. What is useCallback(), and how does it optimize performance?
58. What is useMemo(), and when should you use it?
59. What is useReducer(), and how is it different from useState()?
60. What is useContext(), and how does it simplify state management?
61. What are custom hooks, and why are they useful?
62. What are some best practices when working with React Hooks?
63. How does useLayoutEffect() differ from useEffect()?
64. What is the difference between useEffect() and useInsertionEffect()?

State Management in React

65. How does state management work in React without external libraries?
66. What is the Context API, and how does it help in state management?
67. What are the limitations of using the Context API for global state?
68. What is Redux, and how does it work?
69. How does Redux differ from Context API?
70. What are Redux actions, reducers, and stores?
71. What are middlewares in Redux, and why are they used?
72. How does Redux Toolkit simplify Redux implementation?
73. What is Recoil, and how does it compare to Redux?
74. What is Zustand, and how does it simplify state management?
75. What are the advantages and disadvantages of using external state management libraries?

Advanced React Topics

76. What are Higher-Order Components (HOCs) in React?

By- Sanskar Srivastava (LinkedIn)

77. How does React handle lazy loading and code splitting?
78. What is React Suspense, and how does it work?
79. What is Server-Side Rendering (SSR) in React?
80. What are React portals, and when should you use them?
81. What is Concurrent Mode in React?
82. How does React handle hydration in SSR applications?
83. What is the difference between Next.js and Create React App?
84. How does React handle accessibility (A11Y)?
85. What are error boundaries in React, and how do they improve stability?

Testing in React

86. What are the different ways to test a React application?
87. What is Jest, and how is it used in React testing?
88. How do you test React components using React Testing Library?
89. What is the difference between unit testing and integration testing in React?
90. What is snapshot testing, and how does it work?
91. How do you test user interactions in React?
92. What are mocks, spies, and stubs in Jest testing?
93. What is End-to-End (E2E) testing in React?
94. What is Cypress, and how is it used in React testing?

React Interview Scenarios & Coding Challenges

95. How do you optimize a slow React application?
96. How would you handle large lists in React?
97. How would you debug performance issues in React?
98. How do you prevent memory leaks in React applications?
99. Implement a custom hook for fetching data in React.

By- Sanskar Srivastava (LinkedIn)

100. Implement infinite scrolling in a React application.
101. Implement a debounced search input using React hooks.
102. Implement drag-and-drop functionality in React.
103. Build a dynamic form using React hooks.
104. How would you implement dark mode in a React app?

🔥 Node.js Interview Questions

Core Node.js Concepts

1. What is Node.js, and why is it popular for backend development?
2. How does the Node.js event loop work? Explain its phases in detail.
3. What are the differences between setImmediate, process.nextTick, and setTimeout?
4. What is the difference between event-driven and multi-threaded architectures?
5. Explain the concept of non-blocking I/O in Node.js.
6. What are worker threads in Node.js, and when should you use them?
7. How does Node.js handle memory management and garbage collection?
8. Explain the difference between synchronous and asynchronous programming.
9. How does the cluster module in Node.js work?
10. What are Streams in Node.js? What are the different types of streams?
11. What is the difference between readable, writable, duplex, and transform streams?
12. What is piping in Node.js, and how does it work with streams?
13. What is buffering, and how does it affect performance in Node.js?

Node.js APIs & Modules

14. How do you create a custom module in Node.js?
15. What is the difference between require() and import in Node.js?
16. Explain the difference between CommonJS and ES Modules.
17. How do you handle file operations in Node.js? (Using the fs module)
18. How do you read a large file in Node.js without blocking the event loop?

19. What are the built-in modules in Node.js? Name a few important ones.
20. How does path module work in Node.js?
21. How do you watch for file changes using the fs module?

Error Handling & Debugging

22. What are the different types of errors in Node.js?
23. How do you handle uncaught exceptions in Node.js?
24. What is the difference between process.on('uncaughtException') and process.on('unhandledRejection')?
25. How do you use try/catch with async/await in Node.js?
26. How do you debug a Node.js application?

Security & Performance

27. What are some common security vulnerabilities in Node.js applications?
28. How do you prevent SQL injection and NoSQL injection in Node.js?
29. What is the Helmet.js module, and how does it improve security?
30. What is rate limiting, and how do you implement it in Node.js?
31. How do you optimize the performance of a Node.js server?
32. What is PM2, and why is it used in production?
33. How do you implement caching in a Node.js application?

Package Management

34. What is the difference between npm and yarn?
35. How do you manage package dependencies in Node.js?
36. What are peerDependencies, devDependencies, and dependencies in package.json?
37. How do you create a Node.js package and publish it to npm?

🔥 Express.js Interview Questions

Core Express Concepts

38. What is Express.js, and how is it different from Node.js?

By- Sanskar Srivastava (LinkedIn)

39. How does Express.js handle incoming HTTP requests?
40. What is middleware in Express.js? How does it work?
41. What are the different types of middleware in Express?
42. How do you write custom middleware in Express?
43. What is the difference between app.use(), app.get(), and app.post()?
44. How do you serve static files in an Express app?
45. What is express.json() and express.urlencoded() middleware used for?
46. What is express.static() used for?
47. How do you handle file uploads in Express.js?

Routing & API Development

48. How do you define routes in Express.js?
49. How do you create a RESTful API using Express?
50. What is route parameterization in Express?
51. How do you use req.params, req.query, and req.body?
52. What is the difference between req.params and req.query?
53. How do you implement pagination in an Express API?

Authentication & Security

54. How do you implement JWT-based authentication in Express?
55. How do you handle session-based authentication in Express?
56. What is Passport.js, and how do you use it for authentication?
57. How do you implement OAuth authentication in an Express app?
58. How do you hash passwords before storing them in the database?
59. What is CORS, and how do you enable it in Express?
60. How do you prevent brute-force attacks in Express applications?

Error Handling & Logging

61. How do you implement global error handling in Express?
62. What is the next() function in Express, and how is it used in error handling?

By- Sanskar Srivastava (LinkedIn)

63. How do you log requests and errors in an Express.js app?
64. What are some popular logging libraries for Express.js?

🔥 MongoDB Interview Questions

MongoDB Basics

65. What is MongoDB, and how is it different from relational databases?
66. What is the structure of a document in MongoDB?
67. How do you create a database and collection in MongoDB?
68. How does MongoDB handle data consistency and transactions?

CRUD Operations

69. How do you insert a document into a MongoDB collection?
70. What is the difference between insertOne() and insertMany()?
71. How do you update documents in MongoDB?
72. How do you delete a document from MongoDB?
73. What is the difference between find(), findOne(), and aggregate()?
74. How do you implement search queries in MongoDB?
75. How do you sort and filter data in MongoDB?

Indexes & Performance Optimization

76. What is an index in MongoDB, and why is it important?
77. How do you create an index in MongoDB?
78. What are compound indexes, and how do they work?
79. How do you analyze query performance in MongoDB?
80. What is a text index, and how do you use it for full-text search?

Data Modeling

81. What is the difference between embedded documents and referenced documents?
82. When should you normalize data vs. denormalize it in MongoDB?

By- Sanskar Srivastava (LinkedIn)

83. How does MongoDB handle relationships between collections?
84. What are the advantages and disadvantages of schema-less databases?

Aggregation & Advanced Queries

85. What is the aggregation framework in MongoDB?
86. How do you use $match, $group, and $project in aggregation pipelines?
87. How do you perform real-time analytics using MongoDB aggregation?

🔥 **Bonus: Full-Stack Integration**

88. How do you connect a MongoDB database to an Express.js server?
89. What is Mongoose, and how does it simplify MongoDB operations?
90. How do you handle database migrations in MongoDB?

🔹 **Basic MERN Stack Questions**

1. Can you briefly explain the MERN stack and why it's used?
2. How does React, Node.js, Express, and MongoDB interact in a MERN stack application?
3. What are the advantages of using the MERN stack over traditional LAMP stack?
4. Have you built any projects using the MERN stack? If yes, explain one in detail.

🔹 **JavaScript & React.js Questions**

5. Explain the difference between == and === in JavaScript.
6. How does **event delegation** work in JavaScript?
7. What is the **Virtual DOM**, and how does React use it?
8. What are **React Hooks**? Can you name a few and explain their use cases?
9. What is the difference between useState, useEffect, and useContext?
10. What is **prop drilling**, and how can you avoid it?

By- Sanskar Srivastava (LinkedIn)

11. What is React.memo(), and how does it optimize performance?
12. How do you handle **authentication in a React app**?
13. How would you implement **infinite scrolling** in a React application?

- **Node.js & Express.js Questions**

14. How does **Node.js handle asynchronous operations**?
15. What is **middleware** in Express.js? Can you give an example?
16. How do you handle **file uploads** in Node.js?
17. How does **JWT authentication** work in a MERN app?
18. Explain the difference between **blocking and non-blocking I/O** in Node.js.
19. What is the difference between process.nextTick() and setImmediate()?
20. What are **CORS issues**, and how do you solve them in an Express backend?
21. What is multer, and how is it used in Express?
22. How would you handle **rate limiting** in an Express API?

- **MongoDB & Database Design**

23. What is the difference between **SQL and NoSQL databases**?
24. How does MongoDB store data?
25. What are **indexes** in MongoDB, and how do they improve performance?
26. What is **aggregation in MongoDB**, and when would you use it?
27. Explain the difference between **$match, $group, and $project** in MongoDB aggregation.
28. How do you structure relationships in MongoDB? **(Embedding vs. Referencing)**
29. What are the different ways to **optimize MongoDB queries**?
30. How would you store **user sessions** in MongoDB?

- **Full Stack Architecture & Deployment**

By- Sanskar Srivastava (LinkedIn)

31. How does a **MERN app handle state management**?
32. What is the **best way to handle API errors** in a MERN stack project?
33. How do you manage **authentication & authorization** in a full-stack app?
34. What are **WebSockets**, and how would you implement real-time features in MERN?
35. How would you deploy a MERN stack application? **(Frontend + Backend + Database)**
36. What are **environment variables**, and why are they important in production?
37. How do you scale a Node.js backend for **high-traffic applications**?

🔹 **Coding Challenges**

📌 **Challenge 1: Reverse Words in a Sentence**

Question: Write a JavaScript function to **reverse words in a given sentence**.

```
js

CopyEdit

function reverseWords(sentence) {

  // Your code here

}


console.log(reverseWords("MERN stack is powerful"));

// Output: "powerful is stack MERN"
```

📌 **Challenge 2: Find the First Non-Repeating Character**

Question: Given a string, find the first non-repeating character and return it.

```
js

CopyEdit

function firstUniqueChar(str) {

  // Your code here

}
```

By- Sanskar Srivastava (LinkedIn)

```
console.log(firstUniqueChar("aabbccddeeef")); // Output: "f"
```

📌 Challenge 3: Implement an Express Middleware

Question: Write an Express middleware to **log request details** (method, URL, timestamp).

```
const express = require("express");

const app = express();


// Middleware function

function requestLogger(req, res, next) {

  // Your code here

}


app.use(requestLogger);


app.get("/", (req, res) => {

  res.send("Hello, MERN!");

});

app.listen(3000, () => console.log("Server running on port 3000"));
```

📌 Challenge 4: Design a REST API for a TODO App

- **GET /todos** → Get all todos
- **POST /todos** → Add a new todo
- **PUT /todos/:id** → Update a todo
- **DELETE /todos/:id** → Delete a todo

By- Sanskar Srivastava (LinkedIn)

🚀 Interview Wrap-up Questions

1. What are your strengths as a MERN stack developer?
2. Where do you see yourself in the next **1-2 years** in web development?
3. What are some **challenges you faced** while working with MERN stack projects?
4. Do you have any **questions for us**?

By- Sanskar Srivastava (LinkedIn)