JD Questions:

`React.js and Frontend Development (React Focus)`

1. **What are the core principles of React?**
   - **Expected Answer**: React is based on components, uses a virtual DOM for better performance, and focuses on unidirectional data flow. It makes use of hooks (like `useState`, `useEffect`) to manage state and side effects in functional components.
2. **What is the difference between functional and class components in React?**
   - **Expected Answer**: Functional components are stateless and simpler, but can now hold state using hooks like `useState` and `useEffect`. Class components use lifecycle methods like `componentDidMount`, `componentDidUpdate`, etc., for managing state and side effects.
3. **Can you explain the concept of hooks in React?**
   - **Expected Answer**: Hooks are functions that allow you to use state and other React features in functional components. Examples include `useState` for state management and `useEffect` for side effects (e.g., fetching data or subscribing to events).
4. **What is the Virtual DOM and why is it important in React?**
   - **Expected Answer**: The Virtual DOM is a lightweight copy of the real DOM. React uses it to optimize updates by comparing the current state with the previous one and updating only the necessary parts of the real DOM, improving performance.
5. **What are controlled and uncontrolled components in React?**
   - **Expected Answer**: Controlled components are those where form elements (like `<input>`) are controlled by React state. Uncontrolled components are those where form elements handle their own state (e.g., using `ref` to access DOM values).
6. **Explain the concept of "lifting state up" in React.**

- **Expected Answer**: Lifting state up refers to the process of moving the state to a common parent component when two or more components need access to that state. The parent holds the state and passes it down via props.

7. **What is Redux, and why is it used in React applications?**
   - **Expected Answer**: Redux is a state management library for JavaScript applications. It helps in managing global state in complex React applications by using a centralized store. Redux follows the unidirectional data flow principle.
8. **How do you handle side effects in React?**
   - **Expected Answer**: Side effects are handled using the `useEffect` hook in functional components. It allows you to perform operations like fetching data, subscribing to events, or manually changing the DOM after the component renders.
9. **What are React Router and its uses?**
   - **Expected Answer**: React Router is a routing library for React that allows navigation between different components based on URL paths. It helps in creating Single Page Applications (SPAs) by dynamically rendering components based on route changes.
10. **Explain how `useMemo` and `useCallback` work in React.**
    - **Expected Answer**: `useMemo` is used to memoize expensive calculations, preventing unnecessary re-calculations on re-renders. `useCallback` is used to memoize functions, preventing their re-creation on each render.

JavaScript (Core Knowledge)

1. **What is the difference between `let`, `const`, and `var` in JavaScript?**
   - **Expected Answer**: `let` and `const` are block-scoped variables, while `var` is function-scoped. `let` can be reassigned, `const` cannot be reassigned, and `var` can have issues with hoisting and scoping.
2. **What is a closure in JavaScript?**

- **Expected Answer**: A closure is a function that retains access to its lexical scope, even when the function is executed outside that scope. This is key to creating private variables and functions.

3. **What is the `this` keyword in JavaScript?**
   - **Expected Answer**: `this` refers to the context in which a function is called. In regular functions, it refers to the global object, but in class methods and arrow functions, it refers to the instance or scope where the function is defined.
4. **What are Promises and how do they work in JavaScript?**
   - **Expected Answer**: A `Promise` is an object representing the eventual completion (or failure) of an asynchronous operation. It has three states: `pending`, `fulfilled`, and `rejected`. Methods like `.then()` and `.catch()` allow handling success and failure.
5. **Explain event delegation in JavaScript.**
   - **Expected Answer**: Event delegation is the technique of attaching a single event listener to a parent element rather than multiple listeners to child elements. It takes advantage of event bubbling to handle events on dynamically added elements.
6. **What is the difference between synchronous and asynchronous code?**
   - **Expected Answer**: Synchronous code is executed line by line, while asynchronous code allows non-blocking execution. Asynchronous code uses mechanisms like callbacks, Promises, or `async/await` to handle operations without blocking the main thread.

Backend Development (Node.js & Express)

1. **What is Node.js, and how does it differ from traditional backend technologies like PHP or Java?**
   - **Expected Answer**: Node.js is a runtime environment that allows JavaScript to be run on the server side. Unlike traditional backend technologies, Node.js is event-driven, non-blocking, and uses a single-threaded event loop, which makes it highly scalable.

2. **What is Express.js, and how does it relate to Node.js?**
    - **Expected Answer**: Express.js is a lightweight web framework built on top of Node.js. It simplifies routing, middleware handling, and server-side logic for building web applications or APIs.
3. **What is the role of middleware in Express.js?**
    - **Expected Answer**: Middleware is a function that is executed during the request-response cycle in Express.js. It is used for logging, authentication, error handling, modifying request or response objects, etc.
4. **How do you handle asynchronous operations in Node.js?**
    - **Expected Answer**: In Node.js, asynchronous operations are handled using callbacks, Promises, or `async/await`. This allows Node.js to perform I/O operations without blocking the event loop.

MongoDB (Database)

1. **What is MongoDB, and how does it differ from SQL databases?**
    - **Expected Answer**: MongoDB is a NoSQL, document-oriented database. Unlike SQL databases, which store data in tables, MongoDB stores data in flexible JSON-like documents, which makes it scalable and easy to manage unstructured data.
2. **What are collections and documents in MongoDB?**
    - **Expected Answer**: In MongoDB, a collection is a group of MongoDB documents (similar to a table in SQL). Each document is a set of key-value pairs, and the structure can vary between documents.
3. **How would you perform a CRUD operation in MongoDB?**
    - **Expected Answer**:
        - **Create**: `db.collection.insertOne()` or `insertMany()`
        - **Read**: `db.collection.find()` for querying documents
        - **Update**: `db.collection.updateOne()` or `updateMany()`
        - **Delete**: `db.collection.deleteOne()` or `deleteMany()`

4. **What are indexes in MongoDB, and why are they important?**
    - **Expected Answer**: Indexes in MongoDB improve query performance by allowing faster searching. Without indexes, MongoDB must scan each document in a collection, which is inefficient for large datasets.

Version Control & Deployment

1. **What is Git, and how do you manage branches in Git?**
    - **Expected Answer**: Git is a version control system that tracks changes in code. To manage branches, you use commands like `git branch` to create a branch, `git checkout` to switch, and `git merge` to merge changes from one branch into another.
2. **Explain how you would deploy a React application.**
    - **Expected Answer**: React applications can be deployed using services like Netlify, Vercel, or AWS S3. First, you need to build the application using `npm run build`, then upload the build folder to your deployment service.

Soft Skills & Other Expectations

1. **Can you describe a challenging project you worked on, and how did you handle it?**
    - **Expected Answer**: Discuss a past project (whether academic, personal, or internship) where you faced challenges such as time constraints, technical issues, or team collaboration. Explain how you identified the problem, the steps you took to resolve it, and the final outcome.
2. **How do you keep up with emerging web technologies and frameworks?**
    - **Expected Answer**: I regularly follow blogs, attend meetups, and participate in online courses (e.g., on platforms like Udemy, Coursera). I also follow leading developers and communities on GitHub, Twitter, or Stack Overflow to stay updated.
3. **How do you manage your time and prioritize tasks in a project?**
    - **Expected Answer**: I use tools like Trello or Jira for project management, where I break tasks down into smaller, manageable steps. I prioritize tasks

based on deadlines, dependencies, and the complexity of the task.

Additional Topics (If Time Permits)

- **AWS and Cloud Services**: Basic knowledge of AWS (EC2, S3, Lambda) could be useful.
- **Agile/Scrum Methodologies**: Understanding of Agile practices and Scrum roles (Scrum Master, Product Owner) and ceremonies (Sprint, Retrospective, Standup).
- **JSON Web Tokens (JWT)**: How JWT is used for authentication and authorization in web applications.