MERN Stack Basics

1. What is the MERN Stack?

Answer:
MERN stands for **MongoDB, Express.js, React.js, and Node.js**.
It is a JavaScript tech stack for building full-stack web applications:

- **MongoDB**: NoSQL database for storing data.
- **Express.js**: Web framework for Node.js.
- **React.js**: Front-end library for building user interfaces.
- **Node.js**: JavaScript runtime for server-side programming.

---

2. What are the advantages of using the MERN stack?

Answer:

- Full JavaScript stack (client & server).
- React enables fast, responsive UI.
- Non-blocking Node.js improves performance.
- Scalable MongoDB with flexible schemas.
- Large community and open-source tools.

---

3. How do the components of the MERN stack interact?

Answer:

- React sends HTTP requests (typically via Axios/Fetch) to the backend API.
- Express handles the routes and logic.
- Node.js executes the server-side JavaScript.
- MongoDB stores and retrieves data, often accessed via Mongoose.

MongoDB

4. What is MongoDB and why is it used in the MERN stack?

Answer:
MongoDB is a NoSQL, document-based database. It's used in MERN for:

- Flexibility with JSON-like BSON documents.
- Scalability and high performance.
- Easy integration with Node.js via Mongoose.

---

5. What is Mongoose?

Answer:
Mongoose is an ODM (Object Data Modeling) library for MongoDB and Node.js.
It allows:

- Schema-based data modeling.
- Middleware support.
- Built-in validation and query helpers.

---

6. How do you define a schema in Mongoose?

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  name: String,
  email: { type: String, required: true, unique: true },
  age: Number,
});

const User = mongoose.model('User', UserSchema);
```

---

7. Explain the difference between `findOne()` and `find()` in Mongoose.

Answer:

- `findOne(query)` returns the **first matching document**.
- `find(query)` returns an **array of all matching documents**.

---

8. How do you connect to MongoDB in Node.js?

```
const mongoose = require('mongoose');
```

```
mongoose.connect('mongodb://localhost:27017/myDB')
  .then(() => console.log("Connected"))
  .catch((err) => console.error(err));
```

Express.js

9. What is Express.js?

Answer:
Express.js is a minimal and flexible Node.js web application framework that provides:

- Routing
- Middleware support
- API handling

---

10. How do you define routes in Express.js?

```
javascript
CopyEdit
app.get('/users', (req, res) => {
  res.send('User list');
});
```

---

11. What is middleware in Express?

Answer:
Middleware functions have access to `req`, `res`, and `next()`. They can:

- Modify request/response objects.
- Execute code.
- End request-response cycle.
- Call `next()` to continue.

---

12. How do you handle errors in Express?

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).send('Something broke!');
});
```

13. What is the purpose of body-parser in Express?

Answer:
It parses incoming request bodies before handlers. For JSON:

```
app.use(express.json());
```

---

14. What are route parameters in Express?

```
app.get('/user/:id', (req, res) => {
  res.send(`User ID is ${req.params.id}`);
});
```

Node.js

15. What is Node.js and why use it?

Answer:
Node.js is a JavaScript runtime that executes JS code outside the browser.
Used in MERN for:

- Event-driven, non-blocking I/O.
- Single-threaded but handles many connections concurrently.
- Seamless use of JS on server.

---

16. What is the event loop in Node.js?

Answer:
It's a mechanism that handles asynchronous operations. Phases:

1. Timers
2. Pending callbacks
3. I/O callbacks
4. Poll
5. Check
6. Close callbacks

---

17. What is the difference between `require` and `import`?

Answer:

- `require()` is CommonJS syntax (Node.js default).
- `import` is ES6 module syntax.
  Use `"type": "module"` in package.json for `import`.

---

18. How do you create a REST API in Node.js?

Answer:
Using Express:

```
app.get('/api/data', (req, res) => {
  res.json({ message: 'Data fetched' });
});
```

React.js

19. What is React and its key features?

Answer:

- Component-based
- Virtual DOM
- Declarative UI
- One-way data binding

---

20. What is the Virtual DOM?

Answer:
A lightweight copy of the real DOM. React updates it and syncs with the real DOM efficiently using diffing.

---

21. What are functional components?

```
function Greeting() {
```

```
  return <h1>Hello!</h1>;
}
```

---

22. What are React hooks?

Answer:
Functions like `useState`, `useEffect`, etc., to use state and lifecycle methods in functional components.

---

23. How does `useEffect` work?

```
useEffect(() => {
  // runs after every render
  return () => {
    // cleanup on unmount or before next run
  };
}, []);
```

---

24. What is JSX?

Answer:
JavaScript XML – a syntax extension that lets you write HTML in React:

```
const element = <h1>Hello</h1>;
```

---

25. How do you manage form inputs in React?

```
const [name, setName] = useState('');
<input value={name} onChange={e => setName(e.target.value)} />
```

---

26. What is the difference between props and state?

- **Props**: Passed from parent, read-only.
- **State**: Managed within component, mutable.

---

27. What is React Router?

Answer:
Library to enable routing in React SPA:

```
<Route path="/about" element={<About />} />
```

Integration (Full MERN)

28. How does data flow from React to MongoDB?

Answer:

1. React form submits data to Express API.
2. Express processes request and calls Mongoose.
3. Mongoose interacts with MongoDB to store data.

---

29. How do you fetch data from Express API in React?

```
useEffect(() => {
  fetch('/api/users')
    .then(res => res.json())
    .then(data => setUsers(data));
}, []);
```

---

30. What is CORS and how do you handle it in MERN?

Answer:
CORS (Cross-Origin Resource Sharing) allows restricted resources on a web page to be requested from another domain.

```
const cors = require('cors');
app.use(cors());
```

---

31. How do you implement authentication in MERN?

Answer:

- Use JWT (JSON Web Token).
- React stores token (e.g., localStorage).
- Express middleware verifies JWT.
- MongoDB stores user info.

32. Example of JWT authentication setup

```
const jwt = require('jsonwebtoken');
const token = jwt.sign({ id: user._id }, 'secret', { expiresIn: '1h' });
```

Testing & Deployment

33. How do you test a React component?

Answer:
Using Jest + React Testing Library:

```
render(<MyComponent />);
expect(screen.getByText('Hello')).toBeInTheDocument();
```

---

34. How do you test Express routes?

Answer:
Use **Supertest**:

```
request(app).get('/api/users').expect(200);
```

---

35. How do you deploy a MERN app?

Answer:

- Host backend on **Render/Heroku/EC2**.
- Host frontend on **Vercel/Netlify**.
- Use **MongoDB Atlas** for DB.

---

36. How to serve React from Express in production?

```
app.use(express.static('client/build'));
app.get('*', (req, res) => {
  res.sendFile(path.join(__dirname, 'client/build/index.html'));
});
```

Advanced Questions

37. What is throttling and debouncing in React?

Answer:
Techniques to optimize performance:

- **Debounce**: Delay function until user stops typing.
- **Throttle**: Ensure function runs at fixed intervals.

---

38. Explain Context API in React.

Answer:
A way to pass data through component tree without props:

```
const ThemeContext = React.createContext();
```

---

39. How do you use `async/await` with Express routes?

```
app.get('/users', async (req, res) => {
  const users = await User.find();
  res.json(users);
});
```

---

40. How to handle 404 and errors globally in Express?

```
app.use((req, res) => res.status(404).send('Not Found'));
app.use((err, req, res) => res.status(500).send('Server Error'));
```

Version Control & CI/CD

41. What is Git and why use it in MERN projects?

Answer:
Version control system to track code changes, collaborate, and manage deployment.

---

42. How do you create a CI/CD pipeline for a MERN app?

Answer:
Use tools like:

- GitHub Actions or GitLab CI
- Linting, testing, build steps
- Auto-deploy to Heroku/Vercel/Netlify

Security

43. How do you hash passwords in Node.js?

```
const bcrypt = require('bcrypt');
const hash = await bcrypt.hash(password, 10);
```

---

44. How to prevent NoSQL injection in MongoDB?

- Use Mongoose validation.
- Sanitize input.
- Avoid `$where` queries.

---

45. What is helmet in Express?

Answer:
A middleware to secure Express apps by setting HTTP headers:

```
const helmet = require('helmet');
app.use(helmet());
```

State Management & Optimization

46. How do you manage global state in React?

- Context API
- Redux
- Zustand / Jotai / Recoil (newer options)

---

47. Difference between Redux and Context API?

- **Context API**: Best for light state sharing.
- **Redux**: Powerful, scalable, middleware support.

---

48. What are React Suspense and Lazy?

Answer:

- `React.lazy()` dynamically loads components.
- `Suspense` shows fallback while loading:

```
<Suspense fallback={<div>Loading...</div>}>
  <LazyComponent />
</Suspense>
```

---

49. What is the use of `useMemo` and `useCallback`?

- `useMemo`: Memoize values.
- `useCallback`: Memoize functions.
  Used to optimize performance and avoid unnecessary renders.

---

50. What is server-side rendering (SSR) in React?

Answer:
Rendering React on the server for faster load and SEO. Frameworks: **Next.js**, **Remix**.