1. Core Node.js Concepts

- **Event Loop**: Understanding how Node.js handles asynchronous operations.
- **Modules**: Using `require` / `import` and `module.exports` / `export` for modular development.
- **File System**: Reading and writing files using the `fs` module.
- **Streams and Buffers**: Understanding data streams for I/O operations.
- **Global Objects**: Knowledge of `process`, `__dirname`, `__filename`, etc.
- **Asynchronous Programming**: Using callbacks, Promises, and async/await.

---

2. Building Servers

- **HTTP Module**: Creating basic HTTP servers without a framework.
- **Express.js**: (Part of MERN):
  - Middleware: Built-in, third-party, and custom middleware.
  - Routing: Handling GET, POST, PUT, DELETE requests.
  - Error Handling: Managing errors centrally in Express.

3. Working with Databases

- Connecting Node.js with **MongoDB** using:
  - **Mongoose**: A popular ODM (Object Data Modeling) library.
  - Native MongoDB Driver: For direct interaction with MongoDB.
- CRUD Operations: Creating, Reading, Updating, and Deleting data.
- Data Validation and Schemas: Ensuring data integrity with Mongoose.

4. APIs and Backend Development

- **REST APIs**:
  - Structuring endpoints and handling routes.
  - Using HTTP methods effectively (GET, POST, PUT, DELETE).
- **Authentication**:
  - JSON Web Tokens (JWT) for securing APIs.
  - Middleware for authentication and authorization.
- **API Testing**: Tools like Postman or Insomnia.

5. Middleware and Utilities

- **Body Parsers**: Parsing JSON and URL-encoded data.
- **CORS**: Cross-Origin Resource Sharing for frontend-backend communication.
- **Environment Variables**: Using `dotenv` for managing configuration.

---

6. Real-Time Features

- **WebSockets**: Using `socket.io` for real-time communication if required.

7. Package Management

- Using `npm` or `yarn` for installing and managing dependencies.
- Understanding `package.json` and `package-lock.json`.

---

8. Build Tools and Deployment

- Setting up scripts for starting and deploying apps.
- Preparing for production (e.g., using `pm2` or Docker).

9. Integrating React with Node.js

- Serving React frontend via a Node.js/Express backend.
- Proxying requests in development (`http-proxy-middleware`).

Optional Advanced Topics:

- **Error Logging**: Using tools like `winston` or `morgan`.
- **Testing**: Writing tests with `Jest` or `Mocha`.
- **TypeScript with Node.js**: For type safety.
- **Performance Optimization**: Caching with Redis or optimizing database queries.