React.js

1. **What is React.js?**
   React.js is a JavaScript library for building user interfaces, especially single-page applications. It enables developers to create reusable UI components.
2. **What is frontend and what is the need?**
   The frontend is the user interface part of a web application. It's needed to allow users to interact with the backend/data visually. It includes HTML, CSS, and JavaScript.
3. **What are hooks, states, lifecycle methods?**
   Hooks are functions that let you use state and other React features in functional components. State is data managed within a component. Lifecycle methods are used in class components to run code at particular times (e.g., componentDidMount).
4. **How to run a React application?**
   Use npx create-react-app app-name, navigate to the folder, and run npm start.
5. **What is a React Router?**
   React Router is a library for routing in React applications. It lets you handle navigation and rendering of components based on URL.
6. **What is JSX?**
   JSX stands for JavaScript XML. It allows writing HTML in React.
7. **What is a functional component?**
   A functional component is a JavaScript function that returns JSX and behaves like a component.
8. **What is state and props?**
   State is local and mutable, managed in a component. Props are read-only data passed from parent to child.
9. **Difference between hooks and state:**
   Hooks like useState help manage state in functional components. State is the actual data being stored.
10. **Difference between state and props:**
    State is internal; props are external and passed to components.
11. **Difference between client-side and server-side rendering:**
    Client-side rendering loads content via JavaScript in the browser. Server-side rendering sends a fully rendered page from the server.

12. **Explain useState and useEffect hooks:**
    useState manages local state. useEffect performs side effects like API calls after render.

JavaScript

1. **What do you know about JavaScript?**
   JavaScript is a scripting language used to create dynamic web content.
2. **What is a promise?**
   A promise is an object representing the eventual completion or failure of an asynchronous operation.
3. **Async/Await:**
   Syntax for writing asynchronous code that looks synchronous, using promises.
4. **Callback hell:**
   Nested callbacks that make code hard to read and maintain.
5. **Event loop:**
   Mechanism in JS that handles asynchronous callbacks.
6. **JavaScript hoisting:**
   Variables and functions are moved to the top of their scope before code execution.
7. **Modules:**
   Reusable JS files. Export using export, import using import.
8. **How to use a module from another JS file?**
   Export the variable/function in one file and import in another using import.
9. **How to send photos from frontend to server?**
   Use FormData in frontend and multipart/form-data in HTTP request.
10. **How to decrypt a token?**
    Use libraries like jsonwebtoken in Node.js to verify/decode JWT.
11. **API Methods:**
    Common methods: GET, POST, PUT, PATCH, DELETE.
12. **Prime number program:**

```
function isPrime(n) {
  if (n < 2) return false;
  for (let i = 2; i <= Math.sqrt(n); i++) {
    if (n % i === 0) return false;
```

```
  }
  return true;
}
```

13. **Equilateral Triangle Pattern:**

```
let n = 5;
for(let i=1; i<=n; i++) {
  console.log(" ".repeat(n-i) + "* ".repeat(i));
}
```

14. **Factorial:**

```
function factorial(n) {
  return n <= 1 ? 1 : n * factorial(n - 1);
}
```

HTML

1. **How to connect CSS:**

```
<link rel="stylesheet" href="style.css" />
```

2. **How to connect JS:**

```
<script src="script.js"></script>
```

CSS

1. **Center a div without flex:**
   Use margin: 0 auto; and set a width.
2. **CSS Positions:**

- static
- relative
- absolute
- fixed
- sticky

3. **Width vs Max-width:**

- width sets exact width

- max-width sets upper limit

4. **Types of CSS:**

- Inline
- Internal
- External

Node.js & Express

1. **What is Node.js?**
   JavaScript runtime built on Chrome's V8 engine.
2. **What is Express.js?**
   Minimal web framework for Node.js.
3. **What is MVC?**
   Model-View-Controller design pattern.
4. **What is ODM?**
   Object Data Modeling (e.g., Mongoose for MongoDB).
5. **Flow of backend structure:**
   Routes → Controllers → Services/Models → Database
6. **How to start a server?**

```
app.listen(3000, () => console.log('Server started'));
```

MongoDB

1. **What is MongoDB?**
   NoSQL document-oriented database.
2. **How to insert an object:**

```
db.collection.insertOne({ name: "John" });
```

3. **Get all values:**
   find() method
4. **Get one value:**
   findOne() method
5. **Create a schema:**

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

const UserSchema = new mongoose.Schema({ name: String });
```

6. **Use of mongoose:**
   Helps manage MongoDB using schema and models.
7. **Difference between GET and PATCH:**
   GET retrieves data; PATCH updates partial data.

SQL

1. **What is schema?**
   Structure of a database (tables, fields, relationships).
2. **What is a foreign key?**
   A key used to link two tables.

Deployment

1. **How to deploy a project?**
   Use platforms like Vercel, Netlify (frontend) or Render, Railway, Heroku (backend).

Git

1. **What is Git?**
   Version control system to track code changes.

Next.js

1. **Can you work on Next.js?**
   Yes. Next.js is a React framework with server-side rendering and file-based routing.

MERN Stack

1. **Can you work on Full MERN Stack?**
   Yes. MERN: MongoDB, Express.js, React.js, Node.js.