MERN Stack Viva Questions and Answers

(For beginner and intermediate levels)

1. What is the MERN stack?

- **Answer**:

MERN is a stack of technologies used to develop full-stack web applications. It consists of:

- **MongoDB**: A NoSQL database for storing data as JSON-like documents.
- **Express.js**: A backend web application framework for Node.js.
- **React**: A front-end JavaScript library for building user interfaces.
- **Node.js**: A runtime environment that allows JavaScript to run on the server side.

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

- **Answer**:

- Single language throughout (JavaScript) for front-end, back-end, and database logic.
- Easier to manage data flow using React's unidirectional data binding.
- High flexibility and scalability with MongoDB and Node.js.
- Open-source technologies with strong community support.

3. How does React differ from traditional HTML?

- **Answer**:

React uses **JSX (JavaScript XML)**, which allows developers to write HTML inside JavaScript. JSX makes it easier to manipulate the DOM with a component-based structure. Unlike static HTML, React components are dynamic and re-render when data changes.

4. What is Express.js, and why is it used?

- **Answer**:

Express.js is a lightweight framework built on top of Node.js. It simplifies server-side coding by providing tools to manage routes, handle requests/responses, and integrate middleware. It is commonly used to create RESTful APIs.

5. How does MongoDB store data?

- **Answer**:

MongoDB stores data in **BSON (Binary JSON) documents** within collections. Each document can have nested fields, arrays, and varying structures, making it highly flexible.

6. What is the role of Node.js in the MERN stack?

- **Answer**:

Node.js enables JavaScript to run on the server side. It provides non-blocking, asynchronous I/O operations, which are ideal for building scalable back-end services and APIs.

7. What is the difference between relational and NoSQL databases like MongoDB?

- **Answer**:

- **Relational databases** use tables with fixed schemas (e.g., MySQL).

- **NoSQL databases** like MongoDB use collections and documents, providing flexibility in schema and supporting hierarchical data structures.

8. What is middleware in Express.js?

- **Answer**:

Middleware functions in Express.js are used to process incoming requests before they reach the route handler. Middleware can perform tasks like logging, authentication, or parsing JSON bodies.

9. How does React ensure fast rendering?

- **Answer**:

React uses a **Virtual DOM**. When a component state changes, React compares the new state with the previous one using a process called **reconciliation** and updates only the changed elements in the real DOM, ensuring fast rendering.

10. How do React and Express communicate in a MERN application?

- **Answer**:

React makes HTTP requests (usually via **Axios** or **Fetch API**) to Express.js, which serves as the backend API. Express processes the request, interacts with the MongoDB database, and sends the response back to React.

JavaScript Viva Questions and Answers

(For beginner and intermediate levels)

1. What is JavaScript?

- **Answer**:

JavaScript is a **high-level, interpreted programming language** used primarily for web development. It adds interactivity to web pages by manipulating the DOM, handling events, and making asynchronous requests.

2. What is the difference between `var`, `let`, and `const`?

- **Answer**:

- **var**: Function-scoped and can be redeclared.

- **let**: Block-scoped and cannot be redeclared within the same scope.

- **const**: Block-scoped, used for constants, and must be initialized during declaration.

3. What are Promises in JavaScript?

- **Answer**:

A **Promise** is an object representing the eventual completion (or failure) of an asynchronous operation. It allows chaining with `.then()` and handling errors with `.catch()`.

---

4. What is the difference between `==` and `===` in JavaScript?

- **Answer**:

- **`==`**: Performs **type coercion** before comparison.

- **`===`**: Checks both **value and type** for strict equality.

---

5. What are Arrow Functions?

- **Answer**:

Arrow functions are a concise way to write functions in JavaScript. They use the `=>` syntax and do not have their own `this` binding.

Example:

```javascript

const sum = (a, b) => a + b;

```

6. What is the difference between synchronous and asynchronous JavaScript?

- **Answer**:

- **Synchronous**: Code is executed line by line, blocking the next line until the current one finishes.

- **Asynchronous**: Code is non-blocking, allowing multiple operations to run without waiting for each other (e.g., using callbacks or Promises).

---

7. What is the DOM in JavaScript?

- **Answer**:

The **Document Object Model (DOM)** represents the HTML structure of a web page as a tree of nodes. JavaScript can manipulate the DOM to change the content, style, and structure of a web page dynamically.

8. What is an Event Listener?

- **Answer**:

An **Event Listener** is a function that waits for a specific event to occur (like a button click) and executes a callback function when the event is detected.

Example:

```javascript
document.getElementById('btn').addEventListener('click', () => {
  alert('Button clicked!');
});
```

9. What is the difference between `call()`, `apply()`, and `bind()`?

- **Answer**:

- **`call()`**: Invokes a function with a specified `this` value and arguments.

- **`apply()`**: Same as `call()`, but the arguments are passed as an array.

- **`bind()`**: Returns a new function with the specified `this` value, but does not invoke it immediately.

---

10. What is Hoisting in JavaScript?

- **Answer**:

Hoisting is JavaScript's behavior of moving declarations to the top of their scope. **Variables declared with `var`** and **function declarations** are hoisted, but `let` and `const` are not initialized during hoisting.

11. What are JavaScript Closures?

- **Answer**:

A **closure** is a function that remembers the variables from its outer scope even after the outer function has finished executing.

Example:

```javascript
function outerFunction(outerVar) {
  return function innerFunction(innerVar) {
    console.log(`Outer: ${outerVar}, Inner: ${innerVar}`);
  };
}
```

```
const closure = outerFunction('outside');

closure('inside');  // Output: Outer: outside, Inner: inside
```

12. What is the Fetch API in JavaScript?

- **Answer**:

The **Fetch API** is used to make network requests similar to `XMLHttpRequest` but with a more modern syntax. It returns a Promise that resolves to the response.

Example:

```javascript
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

These questions cover fundamental and intermediate concepts of both the **MERN stack** and **JavaScript**, helping you prepare well for viva exams at multiple levels.

**Viva Questions and Answers: React, Express, MongoDB, and Node.js**

(Excluding previously mentioned questions)

React Viva Questions and Answers

1. What are React hooks? Give some examples.

- **Answer**:

Hooks are special functions introduced in React to allow the use of state and other React features in functional components.

- **Examples**:

  - `useState`: For managing local component state.

  - `useEffect`: For performing side effects (like data fetching).

  - `useContext`: To access context values across components.

2. What is the purpose of React Router?

- **Answer**:

React Router is a library used to manage routing in React applications. It enables navigation between different views or pages without reloading the page, maintaining the single-page application (SPA) experience.

3. Explain the difference between controlled and uncontrolled components in React.

- **Answer**:

- **Controlled components**: The component's state controls the form input, with data managed using React’s state (`useState`).

- **Uncontrolled components**: The DOM handles the form data, using refs to access input values directly.

4. What is the purpose of `useEffect`?

- **Answer**:

`useEffect` is a hook used to perform side effects in functional components, such as fetching data, adding event listeners, or updating the DOM. It runs after the component renders.

5. What are React portals?

- **Answer**:

A **portal** allows rendering a child component into a different part of the DOM, outside the parent hierarchy. Useful for rendering modals or tooltips.

Example:

```javascript
ReactDOM.createPortal(<Child />, document.getElementById('portal-root'));
```

Express.js Viva Questions and Answers

1. How does Express handle routing?

- **Answer**:

Express uses **router methods** like `app.get()`, `app.post()`, etc., to define routes. Each route maps a specific URL path to a callback function that handles the request and sends a response.

---

2. What are query parameters and route parameters in Express?

- **Answer**:

- **Query parameters**: Passed as part of the URL after a `?` (e.g., `/users?name=John`).

- **Route parameters**: Embedded in the route path (e.g., `/users/:id`).

3. How can you handle errors in Express?

- **Answer**:

Error handling middleware is used in Express. It has four parameters: `err`, `req`, `res`, and `next`.

Example:

```javascript
app.use((err, req, res, next) => {
  res.status(500).send('Something went wrong!');
});
```

4. How do you serve static files in Express?

- **Answer**:

You can serve static files (like HTML, CSS, and JS) using the built-in `express.static()` middleware.

Example:

```javascript
app.use(express.static('public'));
```

---

5. What is the role of CORS in Express?

- **Answer**:

CORS (Cross-Origin Resource Sharing) is a security feature that allows or restricts resources to be requested from a different domain. In Express, the **`cors`** middleware is used to enable CORS.

MongoDB Viva Questions and Answers

---

1. What is a collection in MongoDB?

- **Answer**:

A **collection** in MongoDB is a group of documents, similar to a table in relational databases. However, collections in MongoDB can store documents with different structures.

---

2. How do you insert data into a MongoDB collection?

- **Answer**:

You can insert data using the `insertOne()` or `insertMany()` methods.

Example:

```javascript
db.users.insertOne({ name: "John", age: 25 });
```

3. What is an aggregation pipeline in MongoDB?

- **Answer**:

The **aggregation pipeline** is a framework for processing data in stages, like filtering, grouping, and sorting. Each stage transforms the data and passes it to the next stage.

---

4. What are indexes in MongoDB? Why are they important?

- **Answer**:

Indexes improve the speed of queries by allowing quick lookups of documents. Without indexes, MongoDB must scan the entire collection for matching documents.

5. How does MongoDB handle relationships between data?

- **Answer**:

MongoDB supports two ways to represent relationships:

  - **Embedding**: Storing related data in the same document.

  - **Referencing**: Using ObjectId references to relate documents across collections.

Node.js Viva Questions and Answers

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

- **Answer**:

The **event loop** in Node.js is responsible for handling asynchronous operations. It allows Node.js to perform non-blocking I/O operations by

offloading tasks to the system and processing callbacks when tasks are completed.

2. What is the difference between `require()` and `import` in Node.js?

- **Answer**:

- **`require()`**: Used in CommonJS modules (Node.js before ES6).

- **`import`**: Used in ES6 modules, offering better support for tree-shaking and static analysis.

3. What is the purpose of `npm` in Node.js?

- **Answer**:

npm (Node Package Manager) is a tool used to manage libraries and packages in Node.js projects. It allows developers to install, update, and manage dependencies.

4. What is a middleware function in Node.js?

- **Answer**:

Middleware functions are functions that process requests in a web application. They run between the incoming request and the route handler.

5. How does Node.js handle file operations?

- **Answer**:

Node.js provides the `fs` (File System) module to handle file operations, such as reading, writing, and deleting files.

Example:

```javascript

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

```
fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

6. What is clustering in Node.js?

- **Answer**:

Clustering in Node.js allows an application to utilize multiple CPU cores by creating child processes that share the same server port, improving performance.

7. How do you handle environment variables in Node.js?

- **Answer**:

Environment variables can be stored in a `.env` file and accessed using the `dotenv` package or `process.env`.

These questions cover essential concepts of **React, Express.js, MongoDB, and Node.js**, helping you prepare for viva exams at both beginner and intermediate levels.