**JAVASCRIPT Assignment**

**MODULE: 18 (REACT – Lists, Hooks, API,)**

* **Lists**

1. How do you render list of items is React? Why is it important to use keys while rendering lists?

Ans. In React, you can render a list of items using the **map()** method, which iterates over an array and returns a new array of JSX elements.

Keys help React identify which items in a list have changed, been added, or removed.

React uses keys to match each list item to its corresponding DOM element during reconciliation, minimizing unnecessary re-renders.

Without keys, React might mix up item indices, leading to unexpected behavior (e.g., incorrect updates or loss of input state).

1. What are keys in React and what happens if you do not provide a unique key?

Ans. Keys are essential for helping React efficiently identify and update items in a list.

If we do not provide a unique key, React will struggle to optimize rendering, causing performance issues, incorrect updates, and potential loss of state.

* **Hooks**

1. What are React hooks? How do useState() and useEffect() hooks work in functional components?

Ans. React Hooks are functions that let you use state and other React features in functional components.

1. **useState() Hook**

* The useState() hook allows you to add state management to a functional component.
* syntax: const [state, setState] = useState(initialState);
* state: The current state value.
* setState: A function that updates the state.
* initialState: The initial value of the state.

1. useEffect() Hook

* The useEffect() hook allows you to perform side effects in functional components. This is similar to lifecycle methods in class components, like componentDidMount, componentDidUpdate, and componentWillUnmount.
* syntax: useEffect(() => {

// Effect code here

return () => {

// Cleanup code (optional)

};

}, [dependencies]);

* useEffect() fetches data from an API when the component mounts ([] as the dependency array).
* When the data is fetched, it updates the data state, and the loading state is set to false.

1. What problems did hooks solve in React development? Why are hooks considered an important addition to React?

Ans. React hooks are functions that allow you to use state and other React features in functional components. They solve several issues by enabling state management, handling side effects, and sharing logic between components without the need for class components. Hooks like useState() and useEffect() simplify code, making it more declarative and easier to maintain. They improve reusability by allowing logic to be encapsulated in custom hooks, avoiding complex component hierarchies and reducing code duplication. Overall, hooks provide a simpler, more modular approach to React development, aligning with functional programming principles and improving performance.

* **API (Fetching Data)**

1. How do you fetch data from an API in React? Explain the role of fetch() or axios() in making API requests.

Ans. To fetch data from an API in React, you can use either the fetch() function or the axios library. fetch() is a built-in JavaScript function that makes network requests to a specified URL and returns a promise. After making the request, you typically use .then() to handle the response and convert it to JSON format. The fetch() function is simple but requires manually handling response parsing and error handling.

On the other hand, axios is a third-party library that simplifies the process of making HTTP requests. It automatically handles JSON parsing, error handling, and provides additional features like interceptors and request cancellation. Both fetch() and axios are used in React to interact with APIs, but axios is often preferred for its cleaner syntax and extra features.

1. Discuss the importance of handling errors and loading states when working with APIs in React.

Ans. Handling errors and loading states when working with APIs in React is crucial for creating a seamless user experience and ensuring that the application behaves predictably, especially during asynchronous operations like data fetching.

**Loading States** are important because they provide feedback to the user that data is being fetched and the application is processing. Without a loading indicator, users might think the app is not responsive or has crashed, leading to frustration. Showing a loading spinner or message informs users that the request is in progress and assures them that the application is working.

**Error Handling** is equally important because network requests can fail for various reasons (e.g., server issues, network failures, invalid URLs). If errors are not handled properly, the application might crash or display incorrect information, causing confusion or a poor user experience. By catching errors (e.g., using .catch() with fetch() or try-catch with axios), developers can gracefully handle issues like API downtime or incorrect data, and display meaningful error messages to the user.

Together, handling loading and error states ensures that the app remains functional, provides clear feedback to the user, and maintains a high level of reliability, even in case of failures.