Module 9) React

Lists , Hooks , Localstorage , Api Project

Hooks (useState, useEffect, useReducer, useMemo, useRef, useCallback)

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

Answer: React Hooks let us use state and lifecycle features in functional components.

- **`useState()`** is used to create and update values (state).
- **`useEffect()`** runs side effects like fetching data or changing the DOM after render.

• Question 2: What problems did hooks solve in React development? Why are hooks considered an important addition to React?

Answer: Hooks solved the problem of using state and lifecycle in functional components without writing class components.
They made code simpler, reusable, and easier to manage.

• Question 3: What is useReducer ? How we use in react app?

Answer: `useReducer()` is a hook used to manage complex state logic,
Eg:like updating a todo list.
It works like Redux and takes a reducer function and initial state.

```
const [state, dispatch] = useReducer(reducerFunction, initialState);
```

• Question 4: What is the purpose of useCallback & useMemo Hooks?

Answer: **`useCallback`** saves a function so it doesn't get recreated on every render.

`useMemo` saves a calculated value so it doesn't get recalculated every time.
Both improve performance.

• Question 5: What's the Difference between the useCallback & useMemo Hooks?

Answer: **`useCallback`** → returns a memoized function

`useMemo` → returns a memoized value
Use **`useCallback`** for functions, and **`useMemo`** for values.

• Question 6 : What is useRef ? How to work in react app?

Answer: **`useRef()`** stores a value that doesn't change across renders.
It's mostly used to access DOM elements directly or to store mutable values.

Eg:

```
const inputRef = useRef(null);
<input ref={inputRef} />
```