**✅ Question 1: What is Redux, and why is it used in React applications? Explain the core concepts of actions, reducers, and the store.**

**Redux** is a **predictable state management library** used mainly with React to manage complex application states. It allows you to maintain the entire app state in a single store, making it easier to debug, test, and maintain.

**Why Redux is used:**

* To manage **global state** across multiple components.
* Ensures a **unidirectional data flow**, which makes state predictable.
* Simplifies debugging using tools like Redux DevTools.
* Avoids **prop drilling** (passing props through multiple levels).

**Core Concepts:**

1. **Actions:**
   * Plain JavaScript objects that describe **what happened**.
   * Must have a type property and can include a payload.
   * Example:
   * const incrementAction = {
   * type: "INCREMENT",
   * payload: 1
   * };
2. **Reducers:**
   * Pure functions that take the **current state** and an **action**, and return a **new state**.
   * They determine how the state changes in response to actions.
   * Example:
   * const counterReducer = (state = 0, action) => {
   * switch (action.type) {
   * case "INCREMENT":
   * return state + action.payload;
   * default:
   * return state;
   * }
   * };
3. **Store:**
   * The **centralized container** that holds the state of the application.
   * Created using createStore() and allows access to getState(), dispatch(), and subscribe().
   * Example:
   * import { createStore } from 'redux';
   * const store = createStore(counterReducer);

**✅ Question 2: How does Recoil simplify state management in React compared to Redux?**

**Recoil** is a modern state management library developed by Facebook, designed to work seamlessly with React. It simplifies state handling by being more **React-friendly** and avoiding the boilerplate code required by Redux.

**Key Differences and Simplifications:**

1. **Less Boilerplate:**
   * No need for actions, reducers, or a central store.
   * State can be created and used with just a few lines using atoms and selectors.
2. **Atoms (State Units):**
   * Small pieces of state that can be shared across components.
   * Each atom behaves like a local state but is globally accessible.
3. **Selectors (Derived State):**
   * Functions that compute derived data from atoms or other selectors.
   * Useful for computed or filtered data.
4. **Better React Integration:**
   * Works directly with React hooks like useRecoilState() or useRecoilValue().
   * No need for context providers or connect functions.

**Example (Recoil Atom):**

import { atom, useRecoilState } from 'recoil';

const counterState = atom({

key: 'counterState',

default: 0,

});

function Counter() {

const [count, setCount] = useRecoilState(counterState);

return (

<button onClick={() => setCount(count + 1)}>

Count: {count}

</button>

);

}

**In summary:**

Recoil simplifies state management by:

* Reducing boilerplate
* Encouraging modular state
* Offering better integration with React
* Eliminating the need for reducers and actions