**Chapter 12 – Let’s Build Our Store**:

**1️⃣ What is Redux? How is it related with React? Why do we use Redux?**

* **What**: Redux is a state management library for JavaScript applications.
* **Why**: It provides a **single source of truth** (a global store) for the state of your app, making it easier to manage and debug complex applications.
* **Where**: Often used with React, but it can work with any JS framework.
* **Relation with React**: React itself manages state locally with useState or useReducer. But in large apps, state needs to be shared across many components. Redux integrates with React via react-redux to provide a global store accessible from anywhere.

✅ **Interview Answer**:  
Redux is a predictable state management library. In React, it’s used to maintain global state outside of components, preventing problems like prop drilling. We use Redux when the application state becomes large and complex, and needs to be consistently shared across multiple components.

**2️⃣ useContext vs Redux**

* **useContext**: Good for simple apps, avoids prop drilling by sharing data across components. But it doesn’t handle complex logic well.
* **Redux**: Better for larger apps because it has middleware, dev tools, predictable state updates, and can handle async operations.

✅ **Interview Answer**:  
useContext is best for small-scale state sharing, while Redux provides a more scalable, structured solution with middlewares, debugging, and async capabilities for large apps.

**3️⃣ Advantages of using Redux Toolkit over Redux**

* Redux Toolkit (RTK) is the official, recommended way to use Redux now.
* **Problems with old Redux**: lots of boilerplate code, manual setup, verbose actions & reducers.
* **RTK fixes**:
  + Simplified syntax (createSlice, createAsyncThunk)
  + Built-in immutability with Immer
  + Less boilerplate, faster development
  + Best practices built-in

✅ **Interview Answer**:  
Redux Toolkit reduces boilerplate, enforces best practices, and provides APIs like createSlice and createAsyncThunk to simplify reducers and async logic, making Redux much easier to use compared to traditional Redux.

**4️⃣ Explain Dispatcher**

* **What**: Dispatcher is how you send an action to Redux.
* **Why**: Tells Redux what happened (e.g., user clicked a button).
* **Where**: You call dispatch({ type: "increment" }) from your component.

✅ Example:

dispatch({ type: "cart/addItem", payload: item });

**5️⃣ Explain Reducer**

* **What**: A pure function that decides how state should change based on action.
* **Why**: Reducers update the store without mutating it.
* **Where**: Written inside createSlice or manually.

✅ Example:

A screen shot of a computer code

AI-generated content may be incorrect.

**6️⃣ Explain Slice**

* **What**: A “slice” is a piece of Redux state + its reducer + its actions, bundled together.
* **Why**: Helps modularize the store (e.g., cartSlice, userSlice).
* **Where**: Created with createSlice.

✅ Example:

A computer screen with colorful text

AI-generated content may be incorrect.

**7️⃣ Explain Selector**

* **What**: A function to read specific parts of the store.
* **Why**: Keeps components clean and prevents re-rendering with unrelated state changes.
* **Where**: Used with useSelector in React.

✅ Example:

A black screen with white text

AI-generated content may be incorrect.

**8️⃣ Explain createSlice and the configuration it takes**

* **What**: Helper function from Redux Toolkit to create a slice.
* **Config Options**:
  + name: Name of the slice
  + initialState: The starting state
  + reducers: Functions that update state

✅ Example:

A screen shot of a computer code

AI-generated content may be incorrect.

🡪

A diagram of a process

AI-generated content may be incorrect.

✅ **Interview-friendly line:**  
“When a user clicks Add to Cart, React dispatches an action. That action is processed by a reducer, which updates the cart slice in the Redux store. Any component subscribed via selectors automatically re-renders with the updated state, keeping the UI in sync with the store.”