







Chapter 12 - Let's Build our Store

Theory:

- useContext vs Redux.

Feature	useContext	Redux Toolkit (Redux)
Purpose	For small-scale state sharing	For large/global state management
Setup	Simple, no extra library	Needs setup: actions, reducers, store
Scope	Works within a subtree	Works globally across the entire app
Use case	Light apps or local state	Big apps with shared/global state
Tools	React core (createContext, useContext)	Redux Toolkit, store, actions, dispatch

- Advantage of using Redux Toolkit over Redux.
Redux Toolkit (RTK) makes Redux **easier and faster**:
 -  **Less boilerplate** – fewer lines of code to set up actions and reducers.
 -  **Built-in DevTools** – easy debugging.
 -  **Simple store setup** – just use `configureStore()`.
 -  **Best practices by default** – safe immutability via Immer.
 -  **Easy async logic** – with `createAsyncThunk()`.
 -  **Compatible with older Redux** – works with existing code.
- Explain Dispatcher.
 - A **dispatcher** is the `dispatch()` function.
 - You use it to **send actions** to the Redux store.
 - Example:

```
dispatch({ type: 'INCREMENT' });
```
 - It's how you **trigger changes** in your Redux state.
- Explain Reducer.
 - A **reducer** is a function that takes the **current state** and an **action** and returns the **new state**.
 - In Redux Toolkit, reducers are inside `createSlice()`.
 - Example:

```
reducers: {  
  increment: (state) => { state.value += 1; }  
}
```

- Explain slice.

A **slice** is a small part of the Redux store with:

- Its own state
- Reducer functions
- Action creators

□ Think of a slice like a “module” or “section” of your app’s state.

```
const counterSlice = createSlice({
  name: 'counter',
  initialState: { value: 0 },
  reducers: {
    increment: (state) => { state.value++ },
  },
});
```

- Explain selector.

A **selector** is a function to **get specific data** from the Redux store.

Helps in **cleanly accessing** state in components using `useSelector`.

```
const selectCount = (state) => state.counter.value;
const count = useSelector(selectCount);
```

- Explain `createSlice` and the configuration it takes.

`createSlice()` is a function from Redux Toolkit to **create slices easily**.

It takes:

Option	Description
<code>name</code>	Name of the slice
<code>initialState</code>	Starting data
<code>reducers</code>	Object with functions to handle actions
<code>extraReducers</code> (Optional)	To handle actions from other slices

Returns:

- actions: auto-generated action creators
- reducer: reducer function to add to the store

```
const cartSlice = createSlice({
  name: 'cart',
  initialState: { items: [] },
  reducers: {
    addItem: (state, action) => {
      state.items.push(action.payload);
    }
  }
});
```

Coding:

- Practice making a store, slices and do read and write operations using Redux Store
- Build Cart Flow using Redux Store

