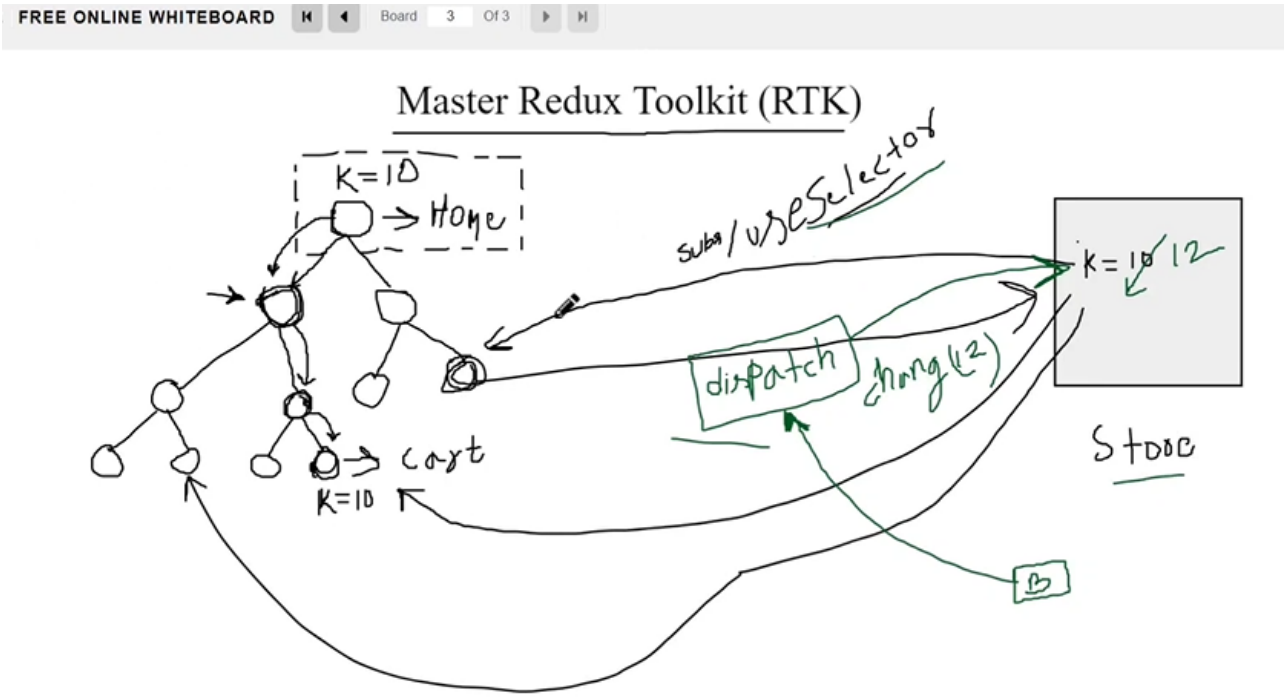
Redux Toolkit & State Management Notes

1. What is Redux?

Redux is a **state management library** for JavaScript apps.
It provides a **single source of truth (store)** where all your application's state lives.

- Without Redux: Each component manages its own state (hard to share data between components).
- With Redux: All components can **access and update state** from one central store.



---

2. Redux Toolkit

Redux Toolkit (RTK) is the **official, recommended way** to write Redux code.
It reduces boilerplate and provides utilities like `createSlice`, `configureStore`.

Key Features:

- Simplifies store setup
- Avoids writing manual action creators & reducers
- Built-in good practices

---

3. Redux Store

The **store** is a JavaScript object that holds the global state.

- Created using `configureStore()` in Redux Toolkit.
- Example (`store.js`):

```
import { configureStore } from "@reduxjs/toolkit";
import authSlice from "./authSlice";

const store = configureStore({
  reducer: {
    auth: authSlice
  }
});

export default store;
```

Step by Step:

1. `configureStore` → function to create store (like a container for global state).
2. `reducer: { auth: authSlice }` → we tell Redux:
    - Our global state has one piece called `auth`
    - That piece will be managed by the logic inside `authSlice`.
3. `export default store` → so we can give this store to the whole React app.

---

4. What is a Slice?

A **slice** is a piece of the global state + the logic to update it.
Created using `createSlice()`.

Example (`authSlice.js`):

```
import { createSlice } from "@reduxjs/toolkit";

const authSlice = createSlice({
  name: "auth",
  initialState: { loading: false },
  reducers: {
    setLoading: (state, action) => {
      state.loading = action.payload;
    }
  }
});

export const { setLoading } = authSlice.actions;
export default authSlice.reducer;
```

Step by Step:

1. `name: "auth"` → identifies this slice.
2. `initialState: { loading: false }` → default state of this slice.
3. `reducers: { setLoading }` → functions that **change state**.
    - `setLoading` takes current state + action.payload (new value) and updates `loading`.

4. `authSlice.actions` → auto-generated action creators (like `setLoading()`).
5. `authSlice.reducer` → the reducer function for this slice, used in the store.

---

5. Adding Store to Project

To use Redux in your React project, wrap your app with `Provider` in `main.jsx`:

```
import { Provider } from "react-redux";
import store from "./redux/store";

ReactDOM.createRoot(document.getElementById("root")).render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

Now the **store is available everywhere** in your app.

---

6. Using Redux in Components

a) Reading Data → `useSelector`

```
const { loading } = useSelector((store) => store.auth);
```

- Access `loading` state from `auth` slice.

b) Updating Data → `useDispatch`

```
const dispatch = useDispatch();
dispatch(setLoading(true));
```

- Calls the `setLoading` reducer to update state.

---

7. Example: Signup Page with Redux

In `Signup.jsx`, when API request starts:

```
dispatch(setLoading(true));
```

When API finishes (success/error):

```
dispatch(setLoading(false));
```

Then:

```
const { loading } = useSelector((store) => store.auth);
```

- If `loading` is `true`, show spinner
- Else, show signup button

---

8. Why Use Redux?

- Global state (e.g., user login info, theme, API loading states)
- Easier to debug and maintain
- Centralized logic (predictable behavior)
- Works well for **medium to large apps**

---

9. Important Interview Questions & Answers

Q1. What is Redux?

Answer: Redux is a predictable state container for JavaScript apps. It helps manage global state in a centralized store so components can easily share data.

Q2. Difference between Redux and Redux Toolkit?

Answer:

- **Redux** → requires manual setup, more boilerplate, manual actions/reducers.
- **Redux Toolkit** → provides `createSlice`, `configureStore`, reduces boilerplate, recommended way.

Q3. What is a Slice in Redux Toolkit?

Answer: A slice is a piece of the Redux state with its own initial state, reducers (functions to update state), and actions. It is created with `createSlice()`.

Q4. What is `useSelector` and `useDispatch`?

Answer:

- `useSelector` → read values from Redux store.
- `useDispatch` → send actions to Redux store to update state.

Q5. What is the difference between Action and Reducer?

Answer:

- **Action** = an object that describes "what happened" (e.g., `{ type: "auth/setLoading", payload: true }`).
- **Reducer** = a function that takes state + action, and returns new state.

Q6. Why wrap App in `<Provider>`?

Answer: The `Provider` component makes the Redux store available to all React components in the app, so they can use `useSelector` and `useDispatch`.

Q7. When should we use Redux?

Answer: Use Redux when:

- State is shared across many components
- App is medium or large scale
- You need predictable state and debugging tools

For small apps, `useState` and `useContext` might be enough.

☑ With this setup, **any component in your app can read/write global state without prop drilling.**