The Main Concept of Redux is:

- Store.
- Action.
- Reducers

****

Redux: It is Global State Management Library That Can Be Used for Managing State.

****

Store: it is global state that can be accessible from any component.

We Can Set Any Data in It.

****

The Store Contain Set of **Slices**.

Each **Slice** Serve a Domain in The Application.

For Example, If We Have a Counter-Component Then We Have Slice for It to Save its State For the use.

****

```
// Store
interface CounterState {
  value: number;
}

interface UserState {
  isSignedIn: boolean;
}
```

Here We Represent Each Interface in Store as Slice.

****

Then It Comes the **Actions**: What Must Happened for Data in Store When Handling Changes of Application.

The Action Hold Two Important Things the **Type** and **Payload**.

The **Payload** is Optional.

Payload Is Any Data That We Want to Send to Redux.

Depending On the Sign That Comes to Redux It Will Fire the Corresponding Action (For Example: Increment **OR** Decrement).

****

Then It Comes **Reducers**: Depending on The Action That Fired by The Type it will make the changes to the Store.

****

The Store Had Concept of Mutability Where We Can't Update the State Without the Reducers.

****

That Means We Must Make Copy of Our State, Then Make Our Changes, Then Override the State of Our Data.

****

The First Step to Create Store in ReactJs is:

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

The Next Step is to export the store that we create:

```
export const store = configureStore({
    reducer: {}
});
```

Here We Don't Set Any Reducer Because We Don't Define Any Slices OR Actions For The State That We Want To Configure.

The Next Step is to export the type of store that we create:

```
export type RootState = ReturnType<typeof store.getState>;
```

using this line, we can access to all root state in our project.

Then We export the dispatcher of the store, that is very useful when we want to make asynchronous jobs using useDispatch-react-hook.

```
export type RootDispatch = typeof store.dispatch;
```

****

Then We Import Provider From react-redux:

npm i @reduxjs/toolkit

npm i react-redux

****

```
import { Provider } from 'react-redux';
import { store } from './state/store.ts';

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>,
)
```

****

Then We Create the First Slice of Counter Application:

First Create The interface of our state:

```
interface CounterState {
    value: number;
}
```

Then We Create the Initial State of Our Counter:

```
const initialState: CounterState = {
    value : 0,
}
```

Then We Create the Slice:

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


const counterSlice = createSlice({
    name: "counter",
    initialState,
    reducers: {}
});

export default counterSlice.reducer;
```

By Creating The Slice, We Can Access the Reducer From it and Export it as default.

****

Note (From Me): We Must Export the Interface That Define the Shape of Initial State of Counter, else we get error of undefined Type OR Name Error.

****

To Define Action And Reducer We Can Define Them Inside The Reducers Object:

```
const counterSlice = createSlice({
    name: "counter",
    initialState,
    reducers: {}
});
```

It Also, take another parameter that is action:

```
const counterSlice = createSlice({
    name: "counter",
    initialState,
    reducers: {
        increment: (state) => {
            state.value += 1;
        },

        decrement: (state) => {
            state.value -= 1;
        }
    }
});
**********************************************************
```

After Make the Actions Inside the Reducers we can export them:

```
export const { increment, decrement} = counterSlice.actions;
**********************************************************
```

Then We Make Our State Inside The Component That We Want To Use:

```
import { useSelector } from "react-redux"
import { RootState } from "../state/store";


const state = useSelector((state: RootState) => state.counter);
**********************************************************
```

Then We Connect the App to Dispatch State Using useDispatch from `react-redux`:

```
import { useDispatch, useSelector } from "react-redux"


const state = useSelector((state: RootState) => state.counter.value);

const dispatch = useDispatch();
```

To Define The Action To Specific Action, We Can Set the PayloadAction<typeHere>:

```
incrementByValue: (state, action: PayloadAction<number>) => {
            state.value += action.payload;
},

decrementByValue: (state, action: PayloadAction<number>) => {
            state.value -= action.payload;
}
```

Note: We Can Set Interface **OR** Object Instead of Number, But the Shape of Payload Then Will Change.

****

To Use The Actions With Payload We export them, then use dispatch with right values:

```
import { increment, decrement, incrementByValue, decrementByValue } from
"../slices/CounterSlice";



<div>
        <button onClick={() => dispatch(incrementByValue(100))}>Increment By
100</button>
        <button onClick={() => dispatch(decrementByValue(100))}>Decrement By
100</button>
      </div>
**********************************************************
```

To Use Async with Redux (Like We Fetch Data From Server):

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



const incrementAsync = createAsyncThunk<number, number, { }>(
    "counter/incrementAsync",
    async (amount: number) => {
        await new Promise((resolve) => setTimeout(resolve, 2000));

        return amount;
    }
);
**********************************************************
```

To Add The Async Function To Reducers Of Store (Redux-Store), We Use extraReducers-property Of createSlice-function:

```
extraReducers: (builder) => {
        builder.addCase(incrementAsync.fulfilled, (state, action:
PayloadAction<number>) =>{
            state.value += action.payload;
        }).addCase(incrementAsync.pending, (state) =>{
            state.value = 0;
        });
    }
**************************************************************

const counterSlice = createSlice({
    name: "counter",
    initialState,
    reducers: {
        ......
    },

    extraReducers: (builder) => {
        builder.addCase(incrementAsync.fulfilled, (state, action:
PayloadAction<number>) =>{
            state.value += action.payload;
        }).addCase(incrementAsync.pending, (state) =>{
            state.value = 0;
        });
    }

});
**************************************************************
```