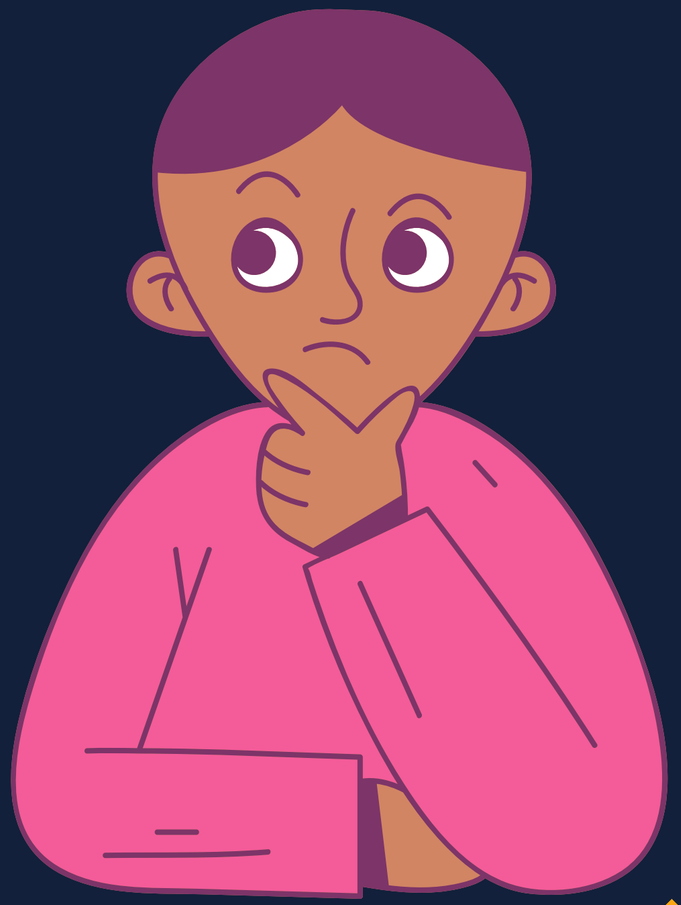
REDUX TOOLKIT

What is Redux Toolkit?



What is Redux Toolkit?

- Redux Toolkit is a complete rewrite of the standard Redux library.
- It is designed to make it easier to write Redux applications by providing a set of helper functions.
- The Redux Toolkit library is divided into 2 parts: The core library and the React bindings.

Importance of RTK

Importance of RTK

- To make Redux development easier
- Easy Store Configurations
- To reduce the size of Redux bundles

Redux Toolkit Common APIs

createAction → Action Creator + Action types

createReducer → It's the easiest way of creating Redux reducer functions

creatSlice → CreateAction + CreateReducer to generate actions and reducer

createAsyncThunk → Handle Async Actions (redux-thunk)

configureStore → Easiest way to create redux store

CreateAction

CreateAction

```
1
2  const increment = createAction("INCREMENT");
```

CreateAction

- It combines action type constants and the action together to create action creator
- The action creator can be called with or without a payload
- By default it accepts one parameter(action type) but can customized
- Action type is a required as a parameter

Customising CreateAction

```
const incrementBy = createAction("INCREMENT_BY", (amount, user) => {
  return {
    payload: {
      amount: amount,
      id: nanoid(),
      user: user,
    },
  };
});
```

createReduer

createReduer

```
1  const counterSlice = createReducer(initialState2, (builder) => {
2    builder
3      .addCase(increment, (state) => {
4        state.counter += 1;
5      })
6      .addCase(incrementBy, (state, action) => {
7        state.counter += action.payload.amount;
8      });
9  });
10
```

createReduer

- It's the easiest way of creating Redux reducer functions
- We can directly mutate the data because it uses immer internally
- It doesn't use switch or case statement
- There are two types of creating reducers (builder callback or map object notation)

createSlice

createSlice

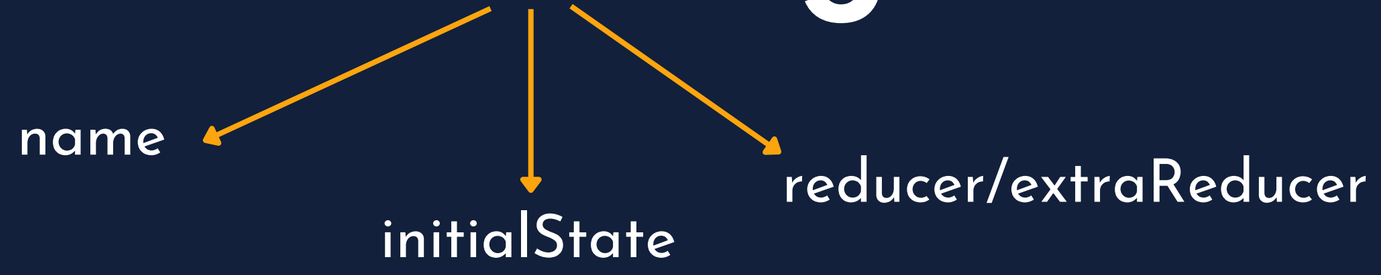
```
1  //create slice
2  const counterSlice = createSlice({
3    name: "counter",
4    initialState,
5    reducers: {
6      increment: (state) => {
7        state.counter++;
8      },
9      decrement: (state) => {
10       state.counter--;
11     },
12     incrementByAmount: (state, action) => {
13       state.counter += action.payload;
14     },
15   },
16 });
```

createSlice

- It simplifies the creation of action creators and reducers
- createSlice = createAction + createReducer
- It doesn't use switch or case statement
- Each slice reducer "owns" its state independently
- This API is the standard approach for writing Redux logic.

createSlice Arguments

createSlice Arguments



name is used in action types, and it must be unique, it represent a particular reducer in the state

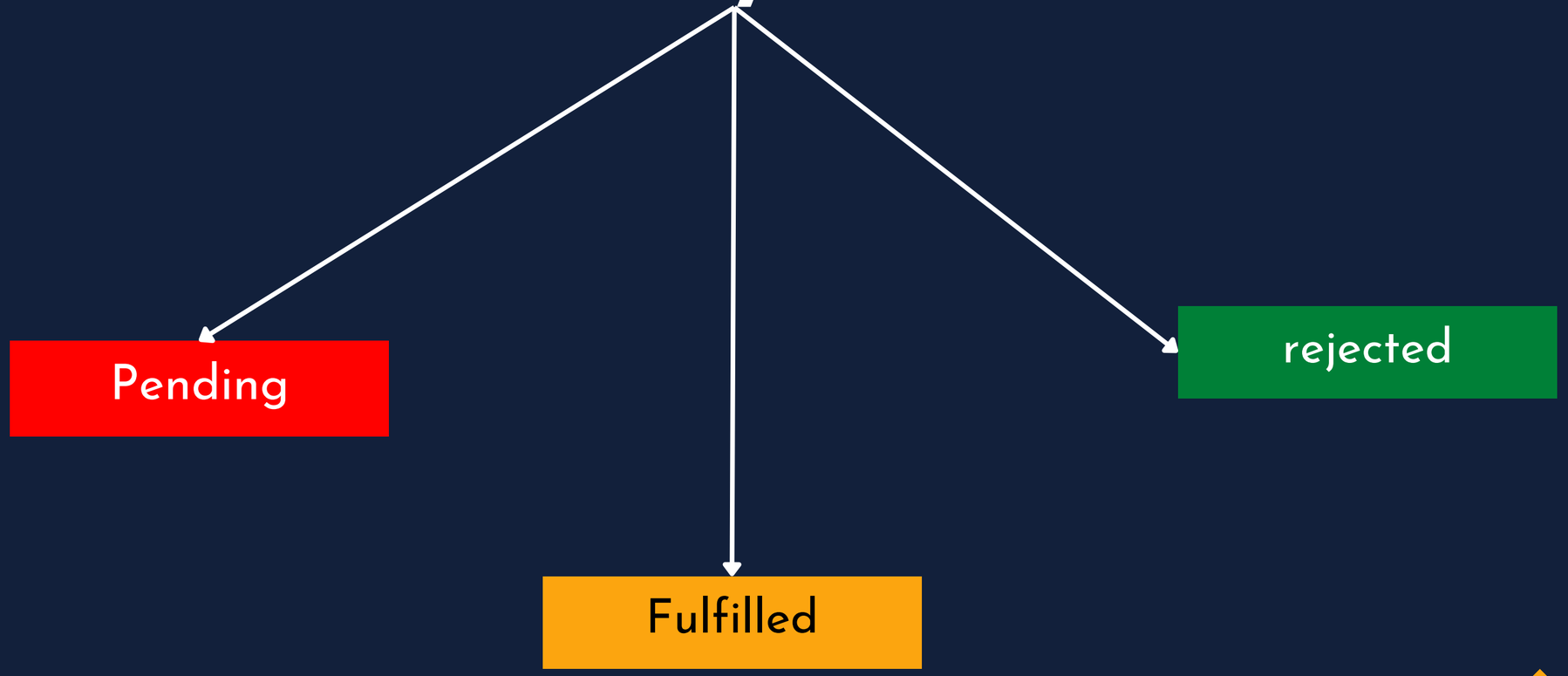
reducer It handle specific action type/ Implement business logic

CreateAsyncThunk

CreateAsyncThunk

- It's the recommended approach for handling async request lifecycles
- This API has eliminated the traditional of installing redux thunk for async actions
- It returns a promise

CreateAsyncThunk



CreateAsyncThunk

```
1  //Async thunk
2  const fetchPosts = createAsyncThunk("posts/fetchPosts", async () => {
3    const data = await axios.get("https://jsonplaceholder.typicode.com/posts");
4    return data;
5  });
```

REDUX TOOLKIT WITH

REDUX TOOLKIT MEGA PROJECT

I-Novotek Academy

I-Novotek Academy

www.inovotekacademy.com