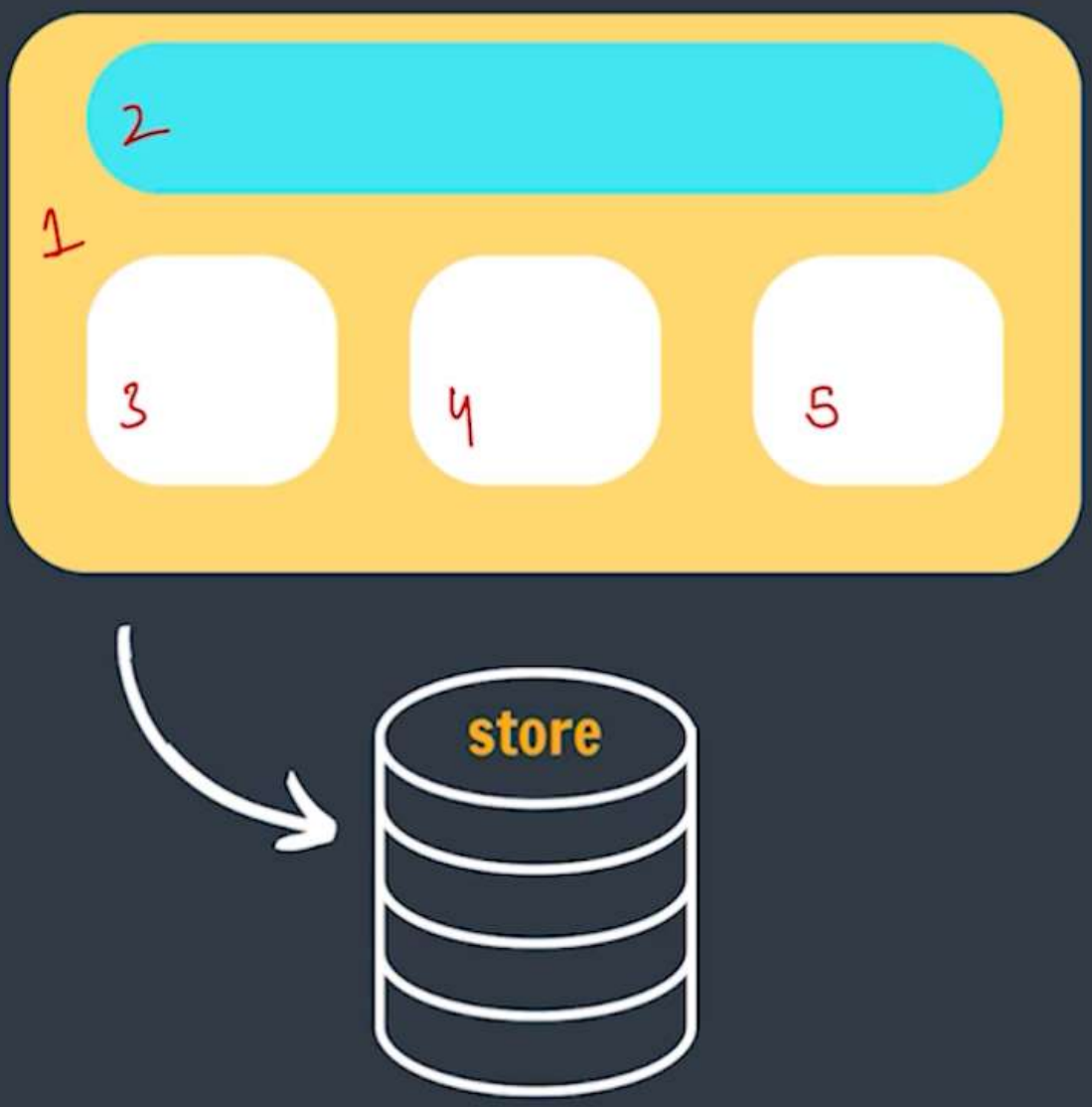
Redux

State Management Library for JS Apps

Redux is built for larger, more complex applications.

Redux Toolkit is the official recommendation of writing Redux code

Redux



store (obj)
reducers (fnx)

mutate

store (obj)
reducers (fnx) [state , action]

count

Understanding Terms

Store : A centralized store holds the whole state tree of your application..

Reducers : Functions that take the current state and an action as arguments, and return a new state result. In other words, (state, action) => newState.

Action : It is a plain JavaScript object that has a type field. (like events)

Slice : Collection of Redux reducer logic and actions for a singl together in a single file.

Understanding Terms

Store : A centralized store holds the whole state tree of your application..

Reducers : Functions that take the current state and an action as arguments, and return a new state result. In other words, (state, action) => newState.

Action : It is a plain JavaScript object that has a type field. (like events)

Slice : Collection of Redux reducer logic and actions for a singl

Setup Project

Store

Actions

Reducer(s)

Todo App

Designing the Store

todo -> id, task, isDone

Actions

Add a Todo, Mark as Done, Delete a Todo

```
{
    type : "ADD_TODO",
    payload : "write code",
}
```

Todo list App

state

todos [{ ✓ }, { ✓ }, { ✓ }, { }]

todo ⇒ { id, task, isDone }

Create a Redux Store

Create a file named `src/app/store.js`. Import the `configureStore` API from Redux Toolkit. We'll start by creating an empty Redux store, and exporting it:

TypeScript JavaScript

app/store.js

```
import { configureStore } from '@reduxjs/toolkit'

export const store = configureStore({
  reducer: {},
})

// Infer the `RootState` and `AppDispatch` types from the store itself
export type RootState = ReturnType<typeof store.getState>
// Inferred type: {posts: PostsState, comments: CommentsState, users: UsersState}
export type AppDispatch = typeof store.dispatch
```

Todo App

Designing the Store

todo -> id, task, isDone

Actions → object

Add a Todo, Mark as Done, Delete a Todo

```
{
    type : "ADD_TODO",
    payload : "write code",
}
```

Creating a reducer

Redux Toolkit automatically generates action creators (fnxs that create action objects)

```
(state, action) => { // update state }
```

*Redux Toolkit lets you write simpler immutable update logic using "mutating" syntax.

Create a Redux State Slice

Add a new file named `src/features/counter/counterSlice.js`. In that file, import the `createSlice` API from Redux Toolkit.

Creating a slice requires a string name to identify the slice, an initial state value, and one or more reducer functions to define how the state can be updated. Once a slice is created, we can export the generated Redux action creators and the reducer function for the whole slice.

Redux requires that we write all state updates immutably, by making copies of data and updating the copies. However, Redux Toolkit's `createSlice` and `createReducer` APIs use Immer inside to allow us to write "mutating" update logic that becomes correct immutable updates.

TypeScript JavaScript

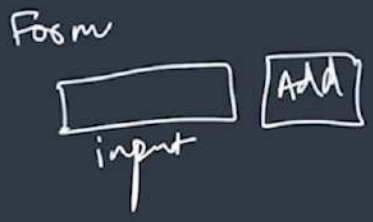
features/counter/counterSlice.js

```
import { createSlice } from '@reduxjs/toolkit'
import type { PayloadAction } from '@reduxjs/toolkit'

export interface CounterState {
  value: number
}

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

Provider Component

The <Provider> component makes the Redux store available to any nested components that need to access the Redux store.

Provider Component



The <Provider> component makes the Redux store available to any nested components that need to access the Redux store.

Provider Component

The <Provider> component makes the Redux store available to any nested components that need to access the Redux store.

use Selector

Dispatching Action

Triggering a State Change

The useDispatch hook allows you to send or dispatch an action to the redux store by giving the action as an argument to the dispatch variable.

The useSelector hooks allow you to extract data or the state from the Redux store using selector function. (returns the data)