**Redux – React**

**בסוף המסמך כתבתי הגדרות לכל מונח כדי להקל על ההבנה – מומלץ לעבור על ההגדות.**

לצורך הבנה בסיסית נפתח פרוייקט ריאקט חדש ריק וכדי לפשט את העניין ככל האפשר נשתמש בקומפוננטה 1 או 2 בלבד.

1. ראשית יש להתקין את Redux ע"י השורה הבאה:  
   npm I react-redux @reactjs/toolkit
2. נפתח תיקיה חדשה בתוך src ברמת components בשם store ובה נפתח קובץ בשם index.ts
3. בתוך index.ts נעשה import לconfigureStore ולcreateSlice
4. import { configureStore,createSlice } from "@reduxjs/toolkit";
5. כעת ניצור state ראשוני בשם initialState ונזין בו את המשתנים שנרצה לעבוד איתם. בגלל שאנחנו נשחק עם הסטייט של הcounter ב + - נזין סטייט בשם counter ובנוסף תהיה אפשרות להציג ולהסתיר את הcounter ולכן ניצור סטייט בשם showCounter ונגדיר true
6. const initialState = {
7. counter: 0,
8. showCounter: true
9. }
10. כעת, ניצור slice שיכלול בתוכו את שם הslice, הסטייט ההתחלתי, ואת כל הreducers שנצטרך לפעולות הבאות: פלוס 1 / מינוס 1 / פלוס התאמה אישית / הסתרה או הצגה של counter
11. const counterSlice = createSlice({
12. name: `counter`,
13. initialState,
14. reducers: {
15. increment(state) {},
16. decrement(state) {},
17. incrementCustom(state) {},
18. toggleCounter(state) {},
19. },
20. });
21. לכל reducer ניתן פרמטר בשם state ונמלא את הלוגיקה שהוא צריך לבצע בכל פעם שנקרא לו:

* שימו לב – אם נרצה להעביר מידע לעבוד איתו בreducer נוסיף פרמטר בשם action
* נוכל לגשת למידע שנכניס ע"י action.payload

מלאו את הלוגיקה לreducers בצורה הבאה:

const counterSlice = createSlice({

  name: `counter`,

  initialState,

  reducers: {

    increment(state) {

      state.counter++

    },

    decrement(state) {

      state.counter--

    },

    incrementCustom(state, action) {

      state.counter += action.payload

    },

    toggleCounter(state) {

      state.showCounter = !state.showCounter

    },

  },

});

1. כעת עלינו ליצור את store ע"י קריאה לפונקציה configureStore אשר מקבלת reducer.

את store נייצא (export) ונעביר אובייקט המכיל reducer אשר מצביע על הslice שיצרנו בתוכו הreducers

\*שימו לב – במידה ויש לנו מספר slices יש להשתמש בsyntax הבא (Comment) כדי לבחור את הרצוי.

export const store = configureStore({

  reducer: counterSlice.reducer,

  // if multiple slices:

  // reducer: {counter:counterSlice.reducer}

});

1. כעת כדי שנוכל להשתמש בreducers שיצרנו עלינו לייצא את action של כל אחד מהם ע"י יצירת משתנה ובו נכניס את האובייקט actions שבתוך הslice שבנינו:
2. export const counterActions = counterSlice.actions;

לבסוף הקובץ index.ts שלנו צריך להראות כך:

import { configureStore, createSlice } from "@reduxjs/toolkit";

const initialState = {

  counter: 0,

  showCounter: true,

};

// Slice Creation

const counterSlice = createSlice({

  name: `counter`,

  initialState,

  reducers: {

    increment(state) {

      state.counter++

    },

    decrement(state) {

      state.counter--

    },

    incrementCustom(state, action) {

      state.counter += action.payload

    },

    toggleCounter(state) {

      state.showCounter = !state.showCounter

    },

  },

});

export const store = configureStore({

  reducer: counterSlice.reducer,

  // if multiple reducers:

  // reducer: {counter:counterSlice.reducer}

});

export const counterActions = counterSlice.actions;

**נעבור לקובץ Root של הפרוייקט שלנו - index.tsx שבתוך src**

1. ונעטוף את הקומפוננטה שנרצה שכל הילדים שלה יקבלו גישה לstore שיצרנו.

נייבא את Provider מreact-redux ואת store ונגדיר את store כ store שבנינו:

import { Provider } from "react-redux";

import Container from "./components/Container/Container";

import { store } from "./store";

const root = ReactDOM.createRoot(

  document.getElementById("root") as HTMLElement

);

root.render(

  <Provider store={store}>

    <Container />

  </Provider>

);

1. כעת, לכל הילדים של Container יש גישה לstore שלנו.

**נעבור לקובץ Counter**

1. נייבא את useSelector, useDispatch מreact-redux ואת counterActions וניצור את dispatch כדי שנוכל לעדכן .state

בנוסף, ניצור interface לסטייט שנרצה לעדכן ובסוף ניצור selectors לstates שנרצה לעבוד איתם.

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

import { counterActions } from "../store";

Interface:

interface RootState {

  counter: number;

  showCounter:boolean;

}

Dispatch:

const dispatch = useDispatch();

selectors:

  const counter = useSelector((state) => state.counter);

  const show = useSelector((state) => state.showCounter);

1. כעת, עלינו ליצור פונקציות לקריאה לreducers ובהן נצביע על הreducer המתאים.

על מנת לעדכן את הסטייט נשתמש בפונקציה useDispatch שעשינו לה import קודם.

בתוך כל פונקציה נקרא לdispatch ובתוך הסוגריים שלה נצביע על הaction שנרצה לעבוד איתו.

שימו לב שיש לנו reducer שמקבל מספר כלשהו ובעזרת כפתור מעלה את הstate בקפיצות לפי המספר שהכנסנו – לצורך העניין 5. הסטייט יעלה ב5 כל פעם.

\*שימו לב לספרה 5 בתוך הסוגריים בפונקציה incrementCustom נוכל לשים כל מספר שנרצה שם.

  const incrementHandler = () => {

    dispatch(counterActions.increment())

    console.log(counter);

  }

  const decrementHandler = () => {

    dispatch(counterActions.decrement())

    console.log(counter);

  }

  const incrementCustom = () => {

    dispatch(counterActions.incrementCustom(5))

    console.log(counter);

  }

  const toggleCounterHandler = () => {

    dispatch(counterActions.toggleCounter())

    console.log(counter);

  }

1. כעת, הרוב מוכן וכל שנצטרך לעשות כדי שהאפליקציה תעבוד הוא לשים onClick על הכפתורים שבנינו ולכוון את האיוונט לפונקציה הרצויה.

בנוסף, על מנת שנוכל להחביא ולהציג את הcounter נרצה לעשות conditional rendering לפי show – הסלקטור שבנינו קודם לכן. השינויים הללו יבוצעו בjsx בלבד ויראו כך:

  return (

    <div className={styles.Counter}>

      <h1>Redux Counter</h1>

     {show && <div className={styles.display}>

        <h2>{counter}</h2>

      </div>}

      <div className={styles.buttons}>

        <button onClick={increment} className={styles.increment}> + </button>

        <button onClick={incrementCustom} className={styles.customIncrement}> custom </button>

        <button onClick={decrement} className={styles.decrement}> - </button>

      </div>

      <div className={styles.toggle}>

        <button onClick={toggleCounter} className={styles.counterToggle}>Toggle Counter</button>

      </div>

    </div>

  );

לבסוף, הקומפוננטה Counter שלכם צריכה להיראות כך:

import React, { FC } from "react";

import styles from "./Counter.module.scss";

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

import { counterActions } from "../../store";

interface RootState {

  counter: number;

  showCounter:boolean

}

const Counter: FC = () => {

const dispatch = useDispatch();

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

const show = useSelector((state: RootState) => state.showCounter);

const increment = () => {

  dispatch(counterActions.increment())

  console.log(counter);

}

const decrement = () => {

  dispatch(counterActions.decrement())

}

const incrementCustom = () => {

  dispatch(counterActions.incrementCustom(5))

}

const toggleCounter = () => {

  dispatch(counterActions.toggleCounter())

  console.log(show);

}

  return (

    <div className={styles.Counter}>

      <h1>Redux Counter</h1>

     {show && <div className={styles.display}>

        <h2>{counter}</h2>

      </div>}

      <div className={styles.buttons}>

        <button onClick={increment} className={styles.increment}> + </button>

        <button onClick={incrementCustom} className={styles.customIncrement}> custom </button>

        <button onClick={decrement} className={styles.decrement}> - </button>

      </div>

      <div className={styles.toggle}>

        <button onClick={toggleCounter} className={styles.counterToggle}>Toggle Counter</button>

      </div>

    </div>

  );

};

export default Counter;

זהו! עכשיו ניתן לייבא ולשנות סטייט דרך redux בכל הסקופ שבנינו כשעטפנו בProvider את הקומפוננטה Containerב – index.tsx!

**הגדרות:**

* **Slice** – מקבץ reducers המקבלים state מאותו הקשר. Slice נועד לעזור לנו לאגד את הreducers שאנו בונים במעטפת אחת כדי שיהיה לנו קל לעבוד ולשנות את הקוד ככל שנרצה. מעין group of reducers.
* **immutable** – כאשר נגיד שproperty בobject הוא immutable אנו בעצם אומרים שהוא בלתי ניתן לשינוי.
* **configureStore** - זוהי פונקציה בRedux Toolkit שעודכנה כהמשך לcreateStore. בעזרתה נוכל ליצור store ולכפות immutability על מנת למנוע שינוי לא רצוי של state מכל סיבה. היא עושה זאת ע"י יצירת שכפול הסטייט ובכל פעם שנעדכן את הסטייט היא תעדכן את הסטייט המשוכפל כך שהסטייט המקורי נשאר initial State.
* **Reducer** – זוהי פונקציה המקבלת סטייט ומיועדת לפעולה בלעדית ליצירת סטייט חדש ולהחליפו בסטייט שקיבלה. לעולם redux לא תשנה את הסטייט הנוכחי! לכן נוכל בקלות לחזור לסטייט הקודם.
* **Action** – זהו אובייקט שמטרתו לשנות סטייט שיקבל. הוא כולל בתוכו שני properties:

Type: `example`,

Payload:1,

כאשר type זה בעצם שם ה,Action וpayload זהו המידע שקיבל על מנת שיוכל לשנות את הסטייט לפי בקשתנו.