**Redux**

**Встановлення**

Додамо до проекту бібліотеку [Redux](https://redux.js.org/) - набір функцій для створення стора (сховища, store), роботи зі станом програми (state) та відправлення екшенів (дій, actions). Безпосередньо із **createStore combineReducers** , буквально вся логіка в цьому пакеті.

npm install redux

Для використання React та Redux разом, необхідно додати до проекту бібліотеку [React Redux](https://react-redux.js.org/). Це набір компонентів та хуків зв'язуючих компонентів React та Redux стор (store). Цей пакет нам потрібен для цього щоб ми могли звязувати наш React додаток з Redux сховищем, щоб коли в Redux сховищі дані змінювалися то в React додатку компонент перемальовувався.

npm install react-redux

Можна встановити їх однією командою :

**npm install redux react-redux**

**Store**

Об'єкт, який містить повний стан програми, методи доступу до стану та відправлення екшенів. У програмі може бути лише один стор. Для створення стора є функція createStore(), яка приймає кілька параметрів та повертає новий об'єкт стора.

**createStore(reducer, preloadedState, enhancer)**

* **reducer** - функція із логікою зміни стану Redux. Обов'язковий параметр.
* **preloadedState** - початковий стан програми. Це має бути об'єкт тієї ж форми, що й, як мінімум, частина стану. Необов'язковий параметр.
* **enhancer** - функція розширення можливостей стору. Необов'язковий параметр.

Шлях до створення : **src/redux/store.js**

import { combineReducers, createStore } from 'redux';

// rootReducer буде викликати combineReducers з Redux і сюди ми будемо передавати обєкт редюсерів. Дальше його передаємо в сховище.

const rootReducer = combineReducers({});

// store - саме сховище, в нього передається корневий редюсер (rootReducer)

export const store = createStore(rootReducer);

Дальше щоб наш додаток знав про існування нашого сховища, огортаємо компонент App в Index.js компонентом Provider імпортованого із React, і в нього передати пропс store із значенням store імпортоване із нашого сховища.

import { Provider } from 'react-redux';

import { store } from 'redux/store';

<Provider store={store}>

<App />

</Provider>

Дальше ми створюємо редюсер

Шлях до створення : **src/redux/template-reducer/template.reducer.js**

// Записуамо в початковий стан дані із локал стореджу або null

const initialState = {

tepm: JSON.parse(localStorage.getItem('templateExaple')) ?? null,

};

// Функція Редюсера приймає два параметри (state, action) і обовязково має повертати стан , змінений або ні. Кожемо що наш стан за замовчуванню = початковому стану.

export const templateReducer = (state = initialState, action) => {

return state;

};

Дальше ми вертаємось в Store і в combineReducers записуємо імпортований наш templateReducer і даємо йому назву

const rootReducer = combineReducers({

tempName: templateReducer,

});

Тепер щоб витягнути наші дані зі стану в компоненті ми використовуємо хук useSelector і передаємо колбек функцію із параметром state, дальше в ньому через крапку витягуємо наш tempName який знаходиться в store combineReducer і наш стейт

const tempData = useSelector(state => state.tempName.tepmState);

console.log('tempData: ', tempData);

Дальше в нашому редюсері пишемо обробники подій (додавання-віднімання елемента ), це робиться через switch/case

switch (action.type) {

case 'products/addProducts': {

return {

...state,

products: [...state.products, action.payload],

};

}

case 'products/deleteProducts': {

return {

...state,

products: state.products.filter(

product => product.id !== action.payload

),

};

}

default:

return state; }

І тепер переходимо до останього етапу, для того щоб надсилати екшини до сховища (store)

Спробуємо спочатку видалити продукт, для того переходимо в компонент ListComponents.

Тут створюємо ф-ю handleDeleteProduct яка приймає параметр id і в ній створюємо екшин функцію тіло якої є обєкт із типом таким як у редюсері і пейлоадом id, дальше діспачимо наш екшн да допомогою функції dispatch.

const handleDeleteProduct = productId => {

const deleteActionProduct = {

type: 'products/deleteProducts',

payload: productId,

};

dispatch(deleteActionProduct);

};

Додаємо продукт аналогічним методом, спочатку збираємо з форми дані , записуємо в обєкт, після чого до цього обєкта додаємо ще id за допомогою бібліотеки nanoid. Створюємо наш екшн із типом таким як у редюсері і в пейлоаді записуємо наш обєкт, після чого діспачимо.

const onHandleSubmit = e => {

e.preventDefault();

const name = e.currentTarget.elements.prodName.value;

const price = e.currentTarget.elements.prodPrice.value;

const discount = e.currentTarget.elements.prodDiscount.value;

const formData = { name, price: Number.parseFloat(price), discount };

const newProduct = {

...formData,

id: nanoid(),

};

const addProductAction = {

type: 'products/addProducts',

payload: newProduct,

};

dispatch(addProductAction);

};