Redux

Redux - відкрита JavaScript бібліотека призначена для управління станом програми (одна із реалізацій Flux). Найчастіше використовується разом з React або Angular для побудови інтерфейсів користувача.

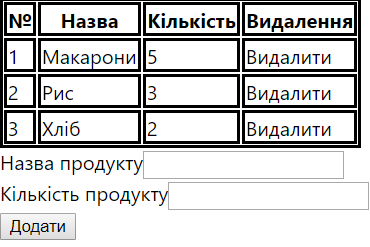
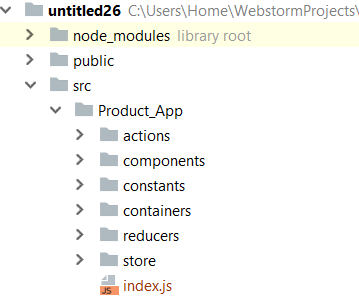
npm install --save redux

Основні принципи Redux:

* єдине джерело істини (Redux зберігає cтан всього додатку в одному сховищі (store) );
* стан призначений тільки для читання;
* єдиний спосіб змінити стан - це виокремити дію (об'єкт, що описує те, що сталося);
* зміни відбувається за допомогою редьюсерів (редьюсери - це лише чисті функції, які приймають попередній стан і дію, і повертають наступний стан)

(previousState, action) => newState .

Розглянемо процес розробки на прикладі розробки списку продуктів

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| Створення типів дій  (для зручності типи подій можна представити у вигляді властивостей об’єкта-константи)  {  властивість\_дія : назва\_дії,  . . . . . . . .  } | **------- ./src/constants/** ActionTypes.js **-------**  **const** ActionTypes={  **ADD\_PRODUCT**:**"ADD\_PRODUCT"**, //🡨 Дія – додавання нового продукту  **DELETE\_PRODUCT**:**"DELETE\_PRODUCT"**, //🡨 Дія – видалення продукту  **EDIT\_PRODUCT**:**"EDIT\_PRODUCT"**, //🡨 Дія – редагування продукту } **export default** ActionTypes; |
| Генератори дій  Дія – це об’єкт, у якому обов’язковим є поле типу type (вказує на тип події), а також можуть бути інші властивості, які несуть у собі необхідні дані (що необхідні для виконання даної дії).  Генератор дії – це функція, яка створює і повертає об’єкт дії.  {  type : тип події ,  payload: необхідні дані  }  Для спрощення використання подій прийнято створювати спеціальні функції-генератори. | **------- ./src/ actions /ActionGenerators**.js **-------**  **import** ActionTypes **from "../constants/ActionTypes" import** {v4} **from "uuid"  const** ActionProductGenerators={ //🡨 Об’єкт, що містить генератори дій  addProduct:**function** (productTitle, productCount) { //🡨 Генератор дії додавання  **return** {  **type**:ActionTypes.**ADD\_PRODUCT**, //🡨 Тип дії (створення нового продукту)  **payload**:{ //🡨 Необхідні дані (інформація про новий продукт)  **id**:v4(), //🡨Генерування унікального id (v4 – функція з модуля **uuid (**npm install uuid --save**)**)  **title**:productTitle,  **count**:productCount  }  }  },  deleteProduct:**function** (productID) { //🡨 Генератор дії видалення  **return** {  **type**:ActionTypes.**DELETE\_PRODUCT**, //🡨 Тип дії (видалення)  **payload**:{ //🡨 Необхідні дані (id прод., що треба видалити)  **id**:productID  }  }  },  editProduct:**function** (productID, newTitle,newCount) {  **return** {  **type**:ActionTypes.**EDIT\_PRODUCT**, //🡨 Тип дії (редагування)  **payload**:{ //🡨 Необхідні дані (нові дані полів)  **id**:productID,   **title**:newTitle,  **count**:newCount  }  }  }, } **export default** ActionProductGenerators; |
| Reducers (перетворювачі сховища) -- це функції, які на вхід приймають поточний стан і дію, та повертає новий стан)  (previousState, action) =>  newState  Для кожної події можна створювати окрему функцію-reducer або ж використовувати оператор вибору switch (аналізуючи тип події) | **------- ./src/reducers/index**.js **-------**  **import** ActionTypes **from "../constants/ActionTypes" import** ActionGenerators **from "../actions/ActionGenerators"  const** *productsReducers*=(state,action) => {  **switch** (action.**type**)  {  **case** ActionTypes.**ADD\_PRODUCT**: //🡨 Додаємо новий продукт  **return** {  **products**: { ...(state.**products**),  [action.**payload**.**id**]:action.**payload** }  }  **case** ActionTypes.**DELETE\_PRODUCT**: //🡨 Видаляємо продукт  **let** ob={...state.**products**};  **delete** ob[action.**payload**.**id**];  **return** {  **products**: ob  }  **case** ActionTypes.**EDIT\_PRODUCT**: //🡨 Змінюємо продукт  **return** {  **products**: {...state.**products**}[action.**payload**.**id**]={...action.**payload}** }  **default**:  **return** state;  } } **export default** *productsReducers*; |
| Компоненти | **-------------------------------------------------------------------------------------------**  **------- ./src/components/AddForm**.js (форма для додавання нового продукту) **-----------------**  **-------------------------------------------------------------------------------------------**  **import** React, { Component} **from 'react'**; **import** PropTypes **from 'prop-types' const** *AddForm*=({onAddClick})=>{  **let** \_title, \_count;  **return**(  <**form action=""**>  <**div**>  <**label** >Назва продукту  <**input type="text" ref=**{el=>\_title=el}/>  </**label**>  </**div**>  <**div**>  <**label**>Кількість продукту  <**input type="number" ref=**{el=>\_count=el}/>  </**label**>  </**div**>  <**button onClick=**{(e)=>onAddClick(e,\_title.**value**,\_count.**value**)}>Додати</**button**>  </**form**>  ) } *AddForm*.**propTypes** = {  **onAddClick**: PropTypes.func.**isRequired**, } **export default** *AddForm*;  **-------------------------------------------------------------------------------------------**  **----------- ./src/components/Product**.js (компонент для відображення одного продукту)**-------**  **-------------------------------------------------------------------------------------------**  **import** React, { Component} **from 'react'**; **import** PropTypes **from 'prop-types' const** *Product*=({id,title,count,onDelClick,num})=>{  **return**(  <**tr**>  <**td**>{num}</**td**>  <**td**>{title}</**td**>  <**td**>{count}</**td**>  <**td**><**a onClick=**{()=>onDelClick(id)}>Видалити</**a**></**td**>  </**tr**>  ) } *Product*.**propTypes** = {  **id**: PropTypes.any.**isRequired**,  **count**: PropTypes.number.**isRequired**,  **title**: PropTypes.string.**isRequired**,  **onDelClick**:PropTypes.func.**isRequired** } **export default** *Product*;  **-------------------------------------------------------------------------------------------**  **------------- ./src/components/Product**.js (компонент для відображення таблиці продуктів)**----------------------------------------------------------------------------------------------**  **import** React, { Component} **from 'react'**; **import** PropTypes **from 'prop-types' import** *Product* **from "./Product" const** *ProductTable*=({products,onDelClick})=>{  **return** (Object.keys(products).**length**>0) ?(  <**table**>  <**thead**>  <**th**>№</**th**>  <**th**>Назва</**th**>  <**th**>Кількість</**th**>  <**th**>Видалення</**th**>  </**thead**>  <**tbody**>  {Object.keys(products).map(  (id,num) => (<**Product key=**{id} {...products[id]}  **onDelClick=**{onDelClick} **num=**{num+1}/>)  )  }  </**tbody**>  </**table**>)  : <**div**>Cписок продуктів порожній</**div**> } *Product*.**propTypes** = {  **products**: PropTypes.object,  **onDelClick**:PropTypes.func.**isRequired** } **export default** *ProductTable*; |
|  | **-------------------------------------------------------------------------------------------**  **------ ./src/containers/ProductApp**.js (компонент, для відображення усього додатку)**---------**  **-------------------------------------------------------------------------------------------**  **import** React, { Component, PropTypes } **from 'react'**; **import** { createStore } **from 'redux' import** ActionProductGenerators **from "../actions/ActionGenerators" import** *ProductTable* **from "../components/ProductTable" import** *AddForm* **from "../components/AddForm"  class** ProductApp **extends** Component{  constructor(props){  **super**(props);  **this**.**store**=**this**.**props**.**store**;  }  onAddClick(e,title,count)  {  e.preventDefault()  **this**.**store**.dispatch(ActionProductGenerators.addProduct(title,parseInt(count)))  }  onDelClick(id)  {  **this**.**store**.dispatch(ActionProductGenerators.deleteProduct(id));  }  render()  {  **let** prod=**this**.**store**.getState().**products**;  **return**(  <**div**>  <**ProductTable products=**{prod} **onDelClick=**{**this**.onDelClick.bind(**this**)}/>  <**AddForm onAddClick=**{**this**.onAddClick.bind(**this**)}/>  </**div**>   )  } } ProductApp.**propTypes** = {  **store**: PropTypes.**object**.**isRequired**, } **export default** ProductApp; |
| Сховище (store) – місце, де зберігається стан додатку  Створення сховища  **const** store=createStore(*reducer*,  initialStore);  **Store Methods**   * [getState()](https://redux.js.org/api/store#getState) -повертає поточний стан * [dispatch(action)](https://redux.js.org/api/store#dispatch) – ініціює виконання дії * [subscribe(listener)](https://redux.js.org/api/store#subscribe) – підписує функцію на події сховища * [replaceReducer(nextReducer)](https://redux.js.org/api/store#replaceReducer)- зміна перетворювача сховища (reduser) | **-------------------------------------------------------------------------------------------**  **- ./src/store/index**.js (створюється сховище) **-**  **-------------------------------------------------------------------------------------------**  **import** { createStore } **from "redux"**; **import** *productsReducers* **from "../reducers"**;   **let** initialStore= **localStorage**.getItem(**"products-store"**)?  {**products**:***JSON***.parse(**localStorage**.getItem(**"products-store"**))}  : {**products**:{}} ;  **export default** createStore(*productsReducers*,initialStore); |
|  | **-------------------------------------------------------------------------------------------**  **- ./src/index\_pr**.js (основний файл, у якому використ. комп. додатку і створюється сховище)  **-------------------------------------------------------------------------------------------**  **import** React,{Component} **from 'react' import** ReactDOM **from 'react-dom' import** ProductApp **from './containers/ProductApp' import** { createStore } **from 'redux' import** *productsReducers* **from "./reducers" import** store **from './store' //🡨 Підключаємо сховище**  **//-------- Функція, яку будемо підписувати на події сховища ---------** **const** *renderProductsList* = () => {  **localStorage**.setItem(**"products-store"**, ***JSON***.stringify(store.getState().**products**));  ReactDOM.render(  <**ProductApp store=**{store}/>,  **document**.getElementById(**"product-container"**)  ); };  **//-------- Підписуємо функцію на події сховища**  store.subscribe(*renderProductsList*); **export default** *renderProductsList*; |
| Використання додатку | **-------------------------------------------------------------------------------------------**  **------- ./src/index\_pr**.js (використання компонента додатку)**----------**  **-------------------------------------------------------------------------------------------**  **import** React, { Component } **from 'react'**; **import** logo **from './logo.svg'**; **import './App.css'**; **import** *renderProductsList* **from "./Product\_App " class** App **extends** Component {  render() {  **return** (  <**div className="App"**>  **{*renderProductsList*()}**  </**div**>  );  } }  **export default** App; |