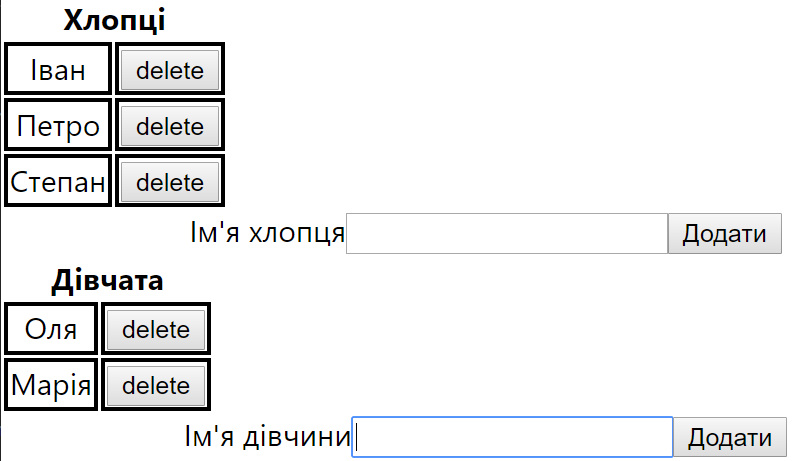
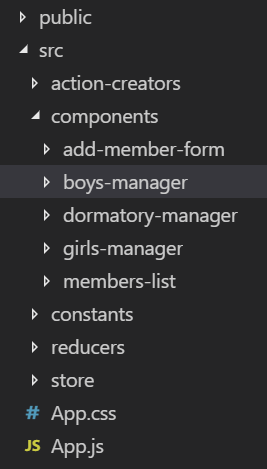
Приклад. Розробити програму “МенеджерГуртожитку”. Зберігаються імена дівчат і хлопців. Використати combineReducers

|  |  |
| --- | --- |
| Загальна форма |  |
| Створення типів дій  (для зручності типи подій можна представити у вигляді властивостей об’єкта-константи)  {  властивість\_дія : назва\_дії,  . . . . . . . .  } | ----------- типи дій для хлопців -----------  const BOYS\_ACTION\_TYPES={  ADD\_BOY:"ADD\_BOY",  DELETE\_BOY:"DELETE\_BOY"  }  export default BOYS\_ACTION\_TYPES  ----------- типи дій для дівчат -----------  const GIRLS\_ACTION\_TYPES={  ADD\_GIRL:"ADD\_GIRL",  DELETE\_GIRL:"DELETE\_GIRL"  }  export default GIRLS\_ACTION\_TYPES  ----------- index.js --------  import BOYS\_ACTION\_TYPES from './boys-actions-types'  import GIRLS\_ACTION\_TYPES from './girls-actions-types'  export {BOYS\_ACTION\_TYPES,GIRLS\_ACTION\_TYPES} |
| 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. | ----------- для дівчат --------  import {GIRLS\_ACTION\_TYPES} from '../constants/action-types'  const addGirl=(girlName)=>(  {  type:GIRLS\_ACTION\_TYPES.ADD\_GIRL,  payload:{  girlName  }  }  )  const deleteGirl=(girlId)=>(  {  type:GIRLS\_ACTION\_TYPES.DELETE\_GIRL,  payload:{  girlId  }  }  )  export {addGirl,deleteGirl}  ----------- для хлопців --------  import {BOYS\_ACTION\_TYPES} from '../constants/action-types'  const addBoy=(boyName)=>(  {  type:BOYS\_ACTION\_TYPES.ADD\_BOY,  payload:{  boyName  }  }  )  const deleteBoy=(boyId)=>(  {  type:BOYS\_ACTION\_TYPES.DELETE\_BOY,  payload:{  boyId  }  }  )  export {addBoy,deleteBoy}  ----------- index.js --------  import {addBoy,deleteBoy} from './boys-actions-creators'  import {addGirl,deleteGirl} from './girl-actions-creators'  export {addBoy,deleteBoy,addGirl,deleteGirl} |
| 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) | ----------- для дівчат --------  import {GIRLS\_ACTION\_TYPES} from '../constants/action-types'  const itialGirlsState={girls:[]}  const girlsReducer=(prevState=itialGirlsState,action)=>{  const {girls} =prevState  switch (action.type) {  case GIRLS\_ACTION\_TYPES.ADD\_GIRL:  const {girlName} =action.payload  return {  girls:[...girls,girlName]  }  case GIRLS\_ACTION\_TYPES.DELETE\_GIRL:  const {girlId}=action.payload  return {  girls:girls.filter((el,id)=>id!==girlId)  }  default: return prevState  }  }  export default girlsReducer  ----------- для хлопців --------  import {BOYS\_ACTION\_TYPES} from '../constants/action-types'  const itialBoysState={boys:[]}  const boysReducer=(prevState=itialBoysState,action)=>{  const {boys} =prevState  switch (action.type) {  case BOYS\_ACTION\_TYPES.ADD\_BOY:  const {boyName} =action.payload  return {  boys:[...boys,boyName]  }  case BOYS\_ACTION\_TYPES.DELETE\_BOY:  const {boyId}=action.payload  return {  boys:boys.filter((el,id)=>id!==boyId)  }  default: return prevState  }  }  export default boysReducer |
| Об’єднуємо редьюсери  rootReducer = **combineReducers**({  частина1 : Reducer1 ,  частина2 : Reducer2 ,  . . . . . . . . . . . . . . . . .  }) | ----------- index.js --------  import {combineReducers} from "redux"  import boysReducer from './boys-reducers'  import girlsReducer from './girls-reducers'  export default combineReducers({  boysGroup:boysReducer,  girlsGroup:girlsReducer  }) |
| Компоненти | ----------- універсальний елемент для виведення списку --------  import React from 'react';  import './member-list.css'  const MemberList = ({listTitle,membersList,onDeleteMember}) => {  return (  <table>  <thead>  <tr>  <th colSpan='2'>  {listTitle}  </th>  </tr>  </thead>  <tbody>  {membersList.map((memberName,i)=>(  <tr key={i}>  <td>{memberName}</td>  <td>  <button onClick={()=>onDeleteMember(i)}>  delete  </button>  </td>  </tr>  ))}  </tbody>  </table>  );  }    export default MemberList;  ----------- універсальний елемент-форма для додавання жильців --------  import React from 'react';  const AddMemberForm = ({label,onAddMember}) => {  let nameInp=null  return (  <>  <label>  {label}  <input ref={(node)=>nameInp=node}/>  </label>  <button onClick={()=>onAddMember(nameInp.value)}>  Додати  </button>  </>  );  }    export default AddMemberForm;  -(краще було звичайно описати ще один універсальний, що об’єднує два попередні) -  ----------- конкретизація для дівчат --------  import React from 'react';  import MemberList from '../members-list'  import AddMemberForm from '../add-member-form'  const GirlsManager = ({girlsList,onAddGirl,onDeleteGirl}) => {  return (  <>  <MemberList  listTitle='Дівчата'  membersList={girlsList}  onDeleteMember={onDeleteGirl}/>  <AddMemberForm  label={`Ім'я дівчини`}  onAddMember={onAddGirl}/>  </>  );  }    export default GirlsManager;  ----------- конкретизація для хлопців --------  import React from 'react';  import MemberList from '../members-list'  import AddMemberForm from '../add-member-form'  const BoysManager = ({boysList,onAddBoy,onDeleteBoy}) => {  return (  <>  <MemberList  listTitle='Хлопці'  membersList={boysList}  onDeleteMember={onDeleteBoy}/>  <AddMemberForm  label={`Ім'я хлопця`}  onAddMember={onAddBoy}/>  </>  );  }    export default BoysManager;  ----------- Гуртожиток --------  import React, { Component } from 'react';  import BoysManager from '../boys-manager'  import GirlsManager from '../girls-manager'  const DormatoryManager = (props) => {  return (  <>  <BoysManager {...props}/>  <GirlsManager {...props}/>  </>  );  }    export default DormatoryManager; |
|  |  |
| 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) | ----------- index.js --------  import { createStore } from "redux";  import rootReducer from '../reducers'  export default createStore(rootReducer) |
| Приєднання компонента до store    //Функція для зчитування даних зі state  const mapStateToProps = (state, ownProps) => ({  . . . повертаємо дані для компонентів . . .  })  //Функція для формування колбеків для компонентів  const mapDispatchToProps = dispatch => {  … повертаємо об’єкт з колбеками для компонентів …  }  // До компонента додаємо можливість зчитувати дані  // та вносити зміни у сховище через actions  export default connect(  mapStateToProps,  mapDispatchToProps  )(компонент)  Передача store через провайдер  export default () => (  <Provider store={store}>  < />  </Provider>  ) | ----------- index.js --------  import React from 'react'  import {Provider, connect} from 'react-redux'  import DormatoryManager from './dormatory-manager'  import store from "../../store"  import {addBoy,deleteBoy,addGirl,deleteGirl} from '../../action-creators'  //---------------------  const mapStateToProps = (state) => {  return {  boysList:state.boysGroup.boys,  girlsList:state.girlsGroup.girls  }  }  //-----------------------  const mapDispatchToProps = dispatch => {  return {  onAddBoy:(boyName)=>dispatch( addBoy(boyName) ),  onDeleteBoy:(boyId)=>dispatch( deleteBoy(boyId)),  onAddGirl:(girlName)=>dispatch( addGirl(girlName) ),  onDeleteGirl:(girlId)=>dispatch( deleteGirl(girlId)),  }  }  //---------------------  let ConnectedDormatory = connect(  mapStateToProps,  mapDispatchToProps  )(DormatoryManager);  //---------------------  export default () => (  <Provider store={store}>  <ConnectedDormatory/>  </Provider>  ) |
|  |  |
| Використання додатку | import React from 'react';  import logo from './logo.svg';  import './App.css';  import Dormatory from './components/dormatory-manager'  function App() {  return (  <div className="App">  <Dormatory/>  </div>  );  }  export default App; |