react-redux

разберем по частям

- **app\src\store\reducer\userReduser.js:**
  - **defaultState**: Это начальное состояние вашего хранилища (store) для пользователей.
  - **Типы действий (actions)**: Это константы, которые представляют допустимые действия, которые можно выполнять над вашим состоянием. В вашем случае есть действие на добавление пользователя (`ADD`) и на удаление пользователя (`REMOVE`).
  - **removeAction и addActtion**: Эти функции называются "создателями действий" (action creators). Они возвращают объекты действий, которые затем можно передать в `dispatch` для обработки редьюсером.
  - **userReduser**: Это ваш редьюсер. Редьюсер - это функция, которая принимает текущее состояние и действие, а затем возвращает новое состояние. В вашем редьюсере проверяется тип действия и, в зависимости от этого, возвращается новое состояние.

```
//app\src\store\reducer\userReduser.js:
const defaultState = [
    { id: 1, name: "John", lastname: "Doe", age: 25, gender: "male" },
    { id: 2, name: "Jane", lastname: "Smith", age: 30, gender: "female" },
    { id: 3, name: "Michael", lastname: "Johnson", age: 22, gender: "male" },
    { id: 4, name: "Emily", lastname: "Williams", age: 28, gender: "female" },
    { id: 5, name: "David", lastname: "Brown", age: 35, gender: "male" },
];

//Типы действий (actions):
const REMOVE = 'REMOVE'
const ADD = 'ADD'

//Функции-конструкторы действий (action creators): removeAction и addActtion.
// создаем (action creators)
export const removeAction = payload => ({type: REMOVE, payload})
export const addActtion = payload => ({type: ADD, payload})


//Редьюсер: userReduser.
export const userReduser = (state = defaultState, action) => {
    if(action.type === REMOVE){
        return state.filter(({id}) => id !== action.payload)
    } else if(action.type === ADD){
        return [...state, {id: Date.now(), ...action.payload}]
    }
```

```
    return state
}
```

- **app\src\store\index.js:**
  - **combineReducers**: Эта функция из `redux` позволяет объединить несколько редьюсеров в один, что упрощает управление хранилищем.
  - **rootReducer**: Это объединенный редьюсер. Пока в нем только один редьюсер (`userReduser`), но в будущем, когда у вас будет больше редьюсеров, это станет удобным.
  - **store**: Это ваше хранилище. Оно создается с использованием `rootReducer`.

```
// app\src\store\index.js :
import { combineReducers, createStore } from "redux";
import { userReduser } from "./reducer/userReduser";

const rootReducer = combineReducers({
    users: userReduser
})

export const store = createStore(rootReducer)
```

- **app\src\components\AddUser\index.jsx:**
  - **useDispatch**: Этот хук из `react-redux` предоставляет функцию `dispatch`, которая позволяет отправлять действия в ваш редьюсер.
  - **onSubmit**: Эта функция вызывается, когда пользователь отправляет форму. Она собирает данные из формы, создает объект пользователя и отправляет действие `addActtion` с этим пользователем в редьюсер через `dispatch`.
  - **return**: В этом разделе кода рендерится форма, которая позволяет пользователю добавить нового пользователя.

```
// app\src\components\AddUser\index.jsx :
import React from 'react'
import s from './style.module.css'
import { addActtion } from '../../store/reducer/userReduser';
import { useDispatch } from 'react-redux';

export default function AddUser() {

    const dispatch = useDispatch();

    const onSubmit = e => {
        e.preventDefault();
        const { name, lastname, age, gender } = e.target;
        const user = {
            name: name.value,
            lastname: lastname.value,
            age: age.value,
            gender: gender.value
        };
```

```
        dispatch(addActtion(user));
        e.target.reset()
    }
console.log(addActtion(user));

    return (
        <form
        onSubmit={onSubmit}
        className={s.form}>
            <input type="text" name='name' placeholder='name' />
            <input type="text" name='lastname' placeholder='lastname' />
            <input type="number" min={0} max={140} name='age' placeholder='age' />
            <select defaultValue={undefined} name='gender'>
                <option disabled value={undefined}>Выберите пол </option>
                <option value="male">Male</option>
                <option value="female">Female</option>
            </select>
            <button>Add</button>
        </form>
    )
}
```

Основной процесс работы следующий:

1. Пользователь заполняет форму и отправляет ее.
2. Функция `onSubmit` собирает данные из формы и создает объект `user`.
3. С помощью `dispatch` и `addActtion` объект `user` отправляется в `userReduser`.
4. `userReduser` обрабатывает действие `ADD` и добавляет нового пользователя в текущий список пользователей.

Это классический пример работы с Redux: компоненты отправляют действия, редьюсеры обрабатывают эти действия и возвращают новое состояние, которое затем рендерится компонентами.

---

1. **app\src\store\reducer\tasksReducer.js**:
    - **defaultState**: Начальное состояние для списка задач. Оно пустое.
    - **Типы действий (actions)**: Определен только один тип действия `ADD` для добавления задачи.
    - **addActtion**: Это создатель действий (action creator) для добавления задачи. Передается заголовок задачи в качестве параметра.
    - **tasksReducer**: Это ваш редьюсер для задач. Если тип действия соответствует `ADD`, то новая задача добавляется в состояние. Для каждой новой задачи генерируется уникальный `id` с помощью `Date.now()`.

```
// app\src\store\reducer\tasksReducer.js :
// tasksReducer
const defaultState = [];

//Типы действий (actions):
const ADD = '[TASKS] ADD'
```

```
//Функции-конструкторы действий (action creators): removeAction и addActtion.

export const addActtion = payload => ({type: ADD, payload})


//Редьюсер: userReduser.
export const tasksReducer = (state = defaultState, action) => {
    if(action.type === ADD){
        return [...state, {id: Date.now(), title: action.payload}]
    }
    return state
}
```

- **app\src\store\index.js**:

  Теперь у вас два редьюсера: `userReduser` и `tasksReducer`. Эти редьюсеры объединяются в `rootReducer` и создается общее хранилище (store).

```
//app\src\store\index.js :
import { combineReducers, createStore } from "redux";
import { userReduser } from "./reducer/userReduser";
import { tasksReducer } from "./reducer/tasksReducer";

const rootReducer = combineReducers({
    users: userReduser,
    tasks: tasksReducer
})

export const store = createStore(rootReducer)
```

- **app\src\components\AddTask\index.jsx**:
  - Это компонент для добавления новой задачи.
  - Когда форма отправляется, `onSubmit` считывает заголовок задачи и отправляет его в `tasksReducer` с помощью `dispatch` и `addActtion`.

```
// app\src\components\AddTask\index.jsx
import React from 'react'
import s from './style.module.css'
import { useDispatch } from 'react-redux';
import { addActtion } from '../../store/reducer/tasksReducer';

export default function AddTask() {

    const dispatch = useDispatch();

    const onSubmit = e => {
        e.preventDefault();
        const title = e.target.title.value;
        dispatch(addActtion(title));
        e.target.reset()
    }
    return (
        <form onSubmit={onSubmit} className={s.form} >
            <input type="text" name='title' placeholder='title' />
            <button>Add</button>
```

```
        </form>
    )
}
```

- **app\src\components\Task\index.jsx**:
    - Это простой компонент для отображения отдельной задачи.

```
// app\src\components\Task\index.jsx :
import React from 'react'

export default function Task({title}) {
  return (
    <div>
      <p>{title}</p>
    </div>
  )
}
```

- **app\src\components\TasksList\index.jsx**:
    - Здесь вы используете `useSelector` из `react-redux` для доступа к состоянию задач в вашем хранилище.
    - Вы затем отображаете каждую задачу с помощью компонента `Task`.

```
// app\src\components\TasksList\index.jsx :
import React from 'react'
import { useSelector } from 'react-redux'
import Task from '../Task';

export default function TasksList() {

    const tasks = useSelector(({tasks}) => tasks);

  return (
    <div>
        {
            tasks.map(task => <Task key={task.id} {...task} />)
        }
    </div>
  )
}
```

- **AddTask Component**:
    - Вы импортировали `useSelector` из `react-redux` для доступа к данным из Redux хранилища.
    - Вы используете `useSelector`, чтобы извлечь всех пользователей из хранилища и представить их в виде выпадающего списка.
    - В форме добавления задачи теперь есть дополнительный элемент `select`, который позволяет выбрать пользователя для задачи.

- При отправке формы вы извлекаете значение `userId` из элемента `select` и добавляете его вместе с заголовком задачи в Redux хранилище.

```
// app\src\components\AddTask\index.jsx :
import React from 'react'
import s from './style.module.css'
import { useDispatch, useSelector } from 'react-redux';
import { addActtion } from '../../store/reducer/tasksReducer';

export default function AddTask() {

    const dispatch = useDispatch();
    const users = useSelector(({ users }) => users);

    const onSubmit = e => {
        e.preventDefault();
        const title = e.target.title.value;
        const userId = e.target.userId.value;
        dispatch(addActtion({title, userId}));
        e.target.reset()
    }
    return (
        <form onSubmit={onSubmit} className={s.form} >
            <input type="text" name='title' placeholder='title' />
            <select name={'userId'}>
                {
                    users.map(({ id, name, lastname }) =>
                        <option key={id} value={id}>{`${name} ${lastname}`}</option>
                    )
                }
            </select>
            <button>Add</button>
        </form>
    )
}

// добавить select со всеми пользователями (имя фамилия через пробел)
// при выборе пользователя должен указываться его ID
// в вывод задачи добавить id пользователя
```

- **tasksReducer**:
    - Вы изменили структуру добавляемой задачи. Теперь, кроме `id`, в задачу также добавляются заголовок и ID пользователя.

```
// app\src\store\reducer\tasksReducer.js :
// tasksReducer
const defaultState = [];

//Типы действий (actions):
const ADD = '[TASKS] ADD'

//Функции-конструкторы действий (action creators): removeAction и addActtion.

export const addActtion = payload => ({type: ADD, payload})


//Редьюсер: userReduser.
export const tasksReducer = (state = defaultState, action) => {
```

```
    if(action.type === ADD){
        // return [...state, {id: Date.now(), title: action.payload}]
        return [...state, {id: Date.now(), ...action.payload}]
    }
    return state
}
```

- **Task Component**:
    - Вы импортировали `useSelector` для извлечения данных пользователя из Redux хранилища.
    - Вы используете `useSelector` для поиска пользователя на основе `userId`, который передается в компонент в качестве свойства.
    - Вы отображаете имя и фамилию пользователя вместе с заголовком задачи.

```
// app\src\components\Task\index.jsx :
import React from 'react'
import { useSelector } from 'react-redux'

export default function Task({ title, userId }) {

  // обратиться к стейту и по userId достать пользователя
  // его данные использоавть для вывода

  const { name, lastname } = useSelector(({ users }) => users.find(elem => elem.id === +userId))

  return (
    <div>
      <p>{`${title} (${name} ${lastname})`}</p>
    </div>
  )
}
```