Redux – final part

|  |  |  |  |
| --- | --- | --- | --- |
| № урока: | 5 | Курс: | React Advanced |
|  | | |  |
| Средства обучения: | | | Текстовый редактор или IDE, браузер, Node.js, терминал |

# Обзор, цель и назначение урока

Это заключительный практический урок по Redux, в этом уроке мы допишем с вами чат начатый еще в предыдущем уроке, будем использовать WebSocket-сервер, узнаем что такое Higher-Order Component и еще раз хорошенько закрепим все знания касательно Redux.

**Изучив материал данного занятия, учащийся сможет:**

* Реализовывать полноценные приложения используя Redux.
* Встраивать в свои приложение WebSocket.
* Переопределять состояние и поведение компонентов с помощью Higher-Order Component

# Содержание урока

1. Подключение в проект chat-websocket
2. WebSocket + Redux
3. Higher-Order component
4. Имплементация функционала чата

# Резюме

**WebSocket** - протокол, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

Он позволяет пересылать любые данные, на любой домен, безопасно и почти без лишнего сетевого трафика.

В этом уроке мы взяли готовую реализацию WebSocket-сервера - <https://github.com/feiyunruyue/chat-websocket>

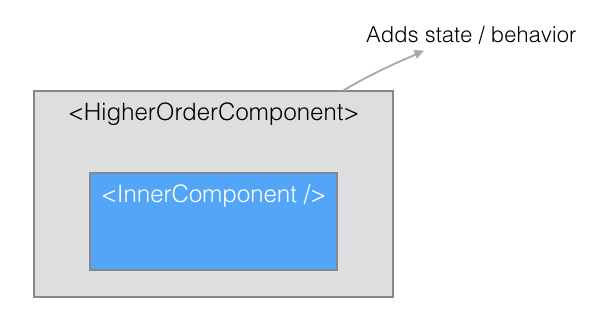
Эта реализация довольно простая и идеально подходит для нашего случая.

<https://www.npmjs.com/package/concurrently> - тул позволяющий запускать несколько команд одновременно.

<https://www.npmjs.com/package/nodemon> - лайврилоад в мире node, nodemon следит за файлами и папками и как только они подвергаются изменениям – тул автоматически перезагрузит ваш скрипт.

**Higher-Order Components –** прием для увеличения процента повторного использования ваших компонентов**,** позволяет изменить логику или состояние компонента не затрагивая при этом сам компонент.

HOC не является частью React API – это паттерн.



На деле же – это функция, которая возвращает React-компонент.

function myHoc(ChildComponent) {

return class myHocClass extends React.Component {

render() {

return (

<div style={{ background: 'red' }}>

<ChildComponent />

</div>

)

}

}

}

Самая близкая аналогия – это декораторы в других языках программирования.

Babel также позволяет использовать декораторы при помощи плагина - <https://babeljs.io/docs/plugins/transform-decorators/>

Но фича является экспериментальной и в продакшн-коде такое использовать крайне не рекомендуется.

Больше про HOC можно почитать на оф.документации по React - <https://facebook.github.io/react/docs/higher-order-components.html>

# Закрепление материала

* Зачем нужны Higher-Order components ?
* Почему в нашем случае WebSocket изначально не мог отправить сообщение сразу и как мы это решили?
* Почему мы использовали localStorage вместо sessionStorage?
* Назовите несколько изменений, которые мы сделали в app.js

# Самостоятельная деятельность учащегося

Задание 1

Реализуйте отправку сообщений в чат с помощью нижнего контрола (ChatControl) – textarea и кнопки “Send”.

# Рекомендуемые ресурсы

Репозиторий данного курса с примерами кода и материалами к урокам

<https://github.com/fnnzzz/react-advanced-itvdn>

Реализация WebSocket-сервера, которые мы использовали в этом уроке

<https://github.com/feiyunruyue/chat-websocket>

<https://www.npmjs.com/package/concurrently>

Утилита позволяющая запускать несколько команд одновременно.

<https://www.npmjs.com/package/nodemon>

Livereload для Node

<https://facebook.github.io/react/docs/higher-order-components.html>

Больше информации про Higher-Order Components