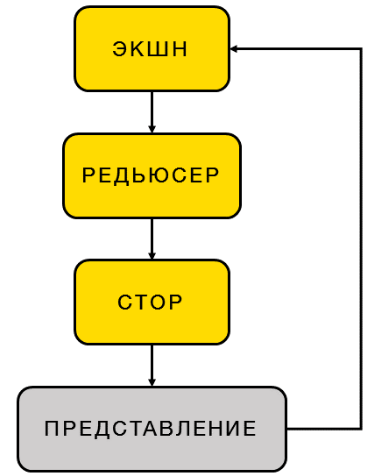
Практическая работа №2 на тему React-Redux

Цель работы: подготовка окружения для работы с Redux и его использование

Ход работы:

Redux – это библиотека, которая воплощает в себе идеи Flux архитектуры для веб-приложений. Flux - это термин, придуманный для обозначения однонаправленного потока данных с очень специфичными событиями и слушателями. Redux предлагает думать о приложении, как о начальном состоянии, модифицируемом последовательностью действий (actions). Схематично это можно представить следующим образом.



У нас имеется некоторое начальное состояние, которое хранится в стор (store), на его основе создается представление, пользователь, взаимодействуя с представлением, вызывает действие (action), на это действие срабатывает редьюсер (reducer). Этот редьюсер на основе предыдущего состояния создает новое, что приводит к изменениям в представлении.

В работе будут использоваться AJAX запросы с помощью fetch, который появился в стандарте ES6. Чтобы наш код работал в браузерах не поддерживающих fetch, нам потребуется установить новую зависимость с помощью команды: npm install whatwg-fetch --save



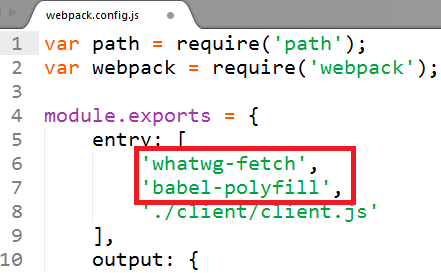
Так же для поддержки старых браузеров установим пакет babel-polyfill

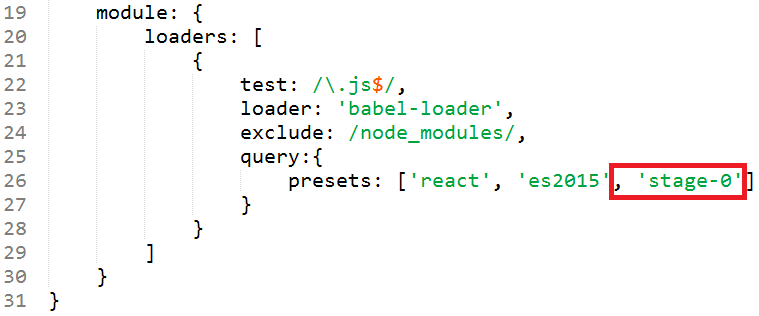


И пакет babel-preset-stage-0 для использования некоторых новшеств из ES7.



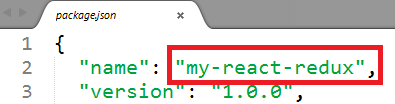
Внесите изменения в файл webpack.config.js, чтобы использовать при сборке установленные пакеты.





Сейчас при сборке bundle.js используемые в коде новшества из ES6 и ES7 будут преобразовываться в стандарт ES5.

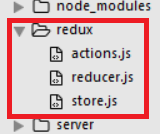
Перейдем к работе с redux, вначале необходимо установить необходимые пакеты (redux, react-redux). Так как имя нашего проекта совпадает с именем пакета react-redux, то npm не сможет его установить, поэтому откройте файл package.json и измените название проекта.



Далее установите зависимости.

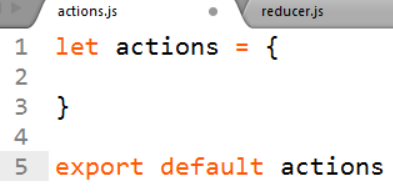


После этого создайте в папке проекта новый каталог redux и уже в нем создайте три файла: action.js, reducer.js, store.js

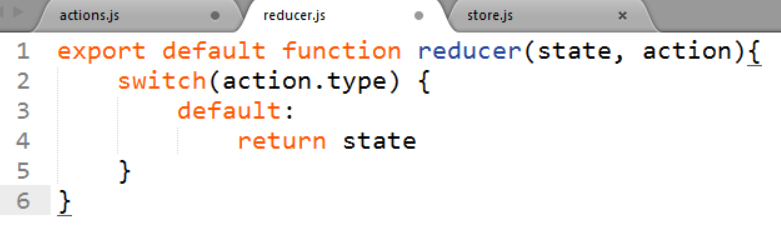


В файле action.js будут храниться действия, они представляют собой чистые функции (функции, которые не меняют состояние приложения). Эти функции будут срабатывать на различные действия пользователя, на сработанные действия будут реагировать reducer, который будет возвращать обновленное состояние приложения. В store будет создаваться начальное состояние приложения.

Откройте action.js и введите код по образцу для создания объекта actions, в котором в дальнейшем будут храниться действия.

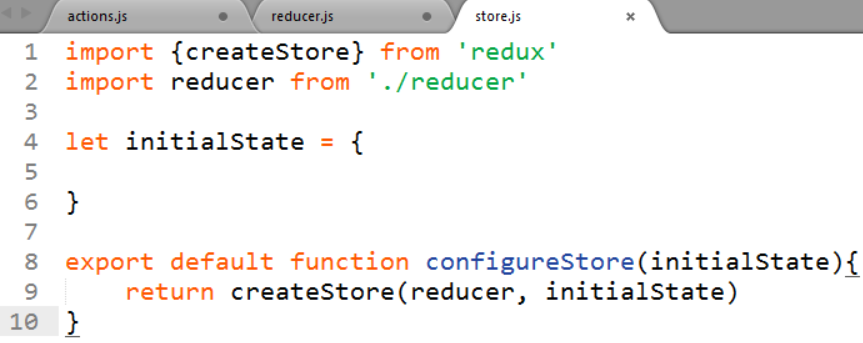


Сохраните изменения, откройте файл reducer.js и введите код по образцу.



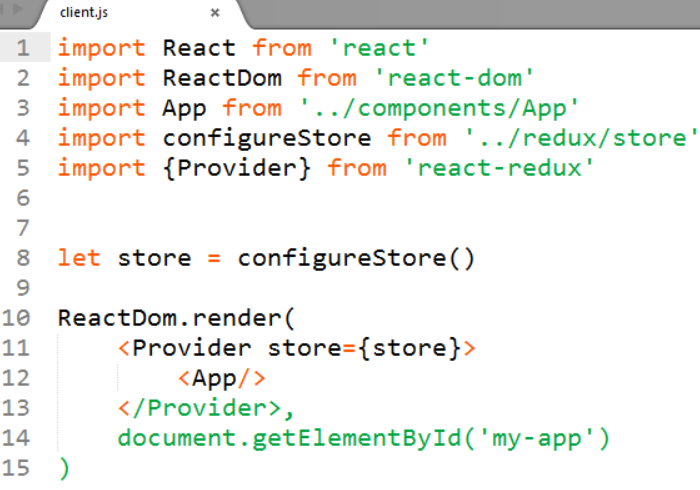
На данном этапе reducer не обрабатывает никаких действий и просто возвращает такой же state, который принимался на вход.

Сохраните изменения, откройте файл store.js и введите код по образцу.



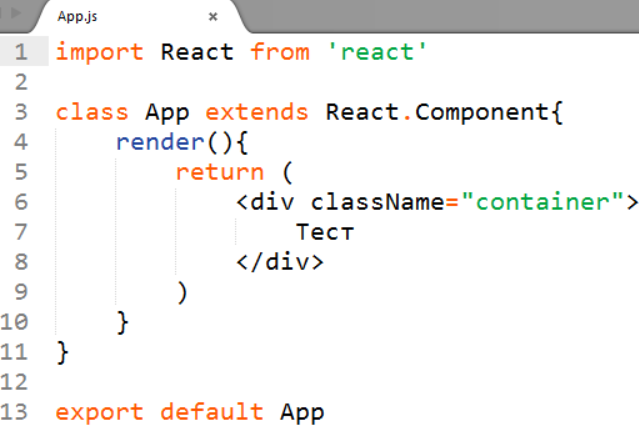
Здесь мы создаем переменную initialState с начальным состоянием приложения (пока оно пустое) и функцию configureStore, в которой вызываем функцию createStore из библиотеки redux, она принимает reducer и начальное состояние, и возвращает рабочее состояние приложения.

Сохраните изменения, откройте файл client.js и измените код по образцу.

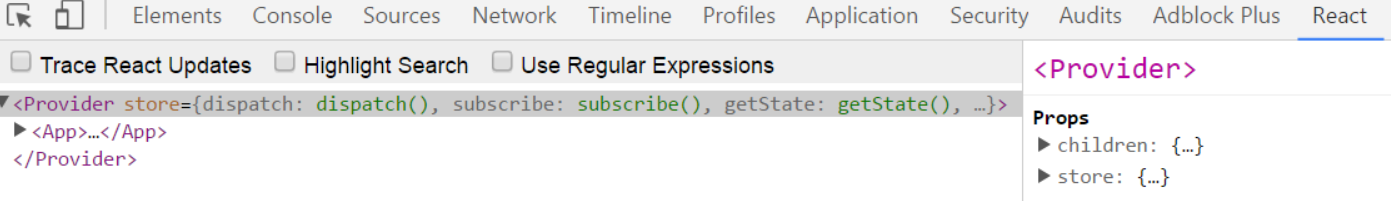


Здесь мы импортируем функцию configureStore и создаем с помощью неё состояние приложения. Для того чтобы пробросить это состояние в компонент App, оборачиваем его в компонент Provider из библиотеки react-redux.

Сохраните изменения, откройте файл App.js и уберите все лишнее по образцу.



Сохраните изменения, соберите bundle.js, запустите веб-сервер, откройте страницу в браузере, нажмите правой кнопкой мышки, выберите просмотреть код и откройте вкладку React.



Далее посмотрим, как пробрасывать состояние в компоненты приложения. Откройте файл store.js и добавьте в initialState поле text по образцу.



Сохраните изменения, откройте файл App.js и добавьте код по образцу.