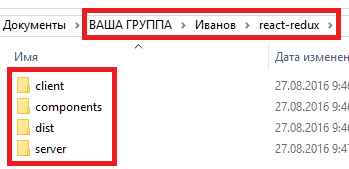
Практическая работа №1 на тему React-Redux

Цель работы: подготовка окружения для работы с React

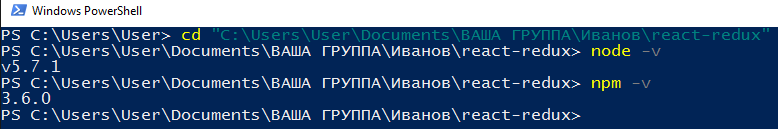
Ход работы:

Создайте в своей папке новую директорию «react-redux» со следующей структурой подкаталогов.



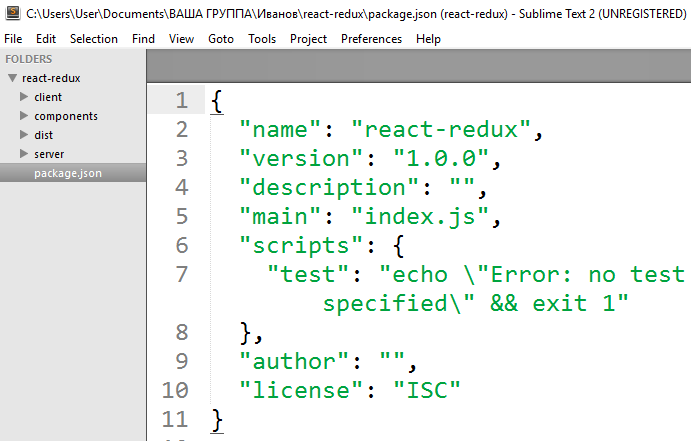
За серверный код будет отвечать Node.js – это платформа, основанная на JavaScript движке V8, и предлагающая асинхронное API для работы с сетью и диском. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода, другими словам это своего рода серверный JavaScript. Для подготовки среды приложений на Node.js используется пакетный менеджер npm, позволяющий скачивать и устанавливать различные библиотеки.

Запустите командную строку (например, PowerShell), перейдите из неё в каталог проекта и убедитесь, что на компьютере установлен Node.js и npm.



Инициализируйте каталог с помощью команды npm init, в процессе инициализации согласитесь со всеми настройками по умолчанию.

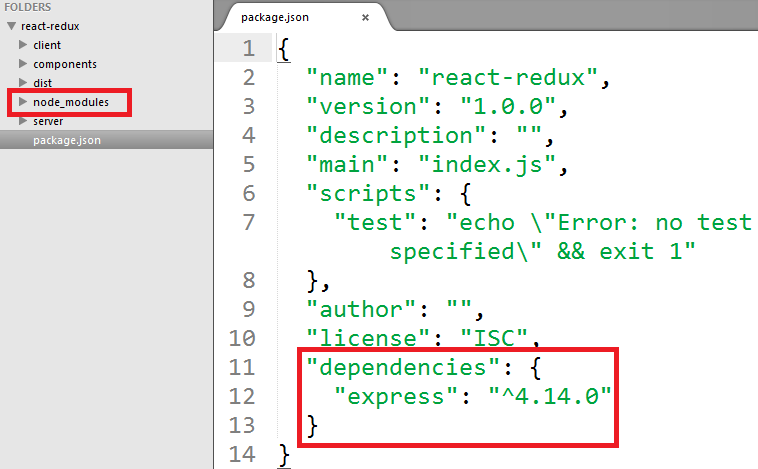
Откройте папку проекта в Sublime Text. После инициализации в каталоге будет создан файл package.json с описанием нашего проекта.



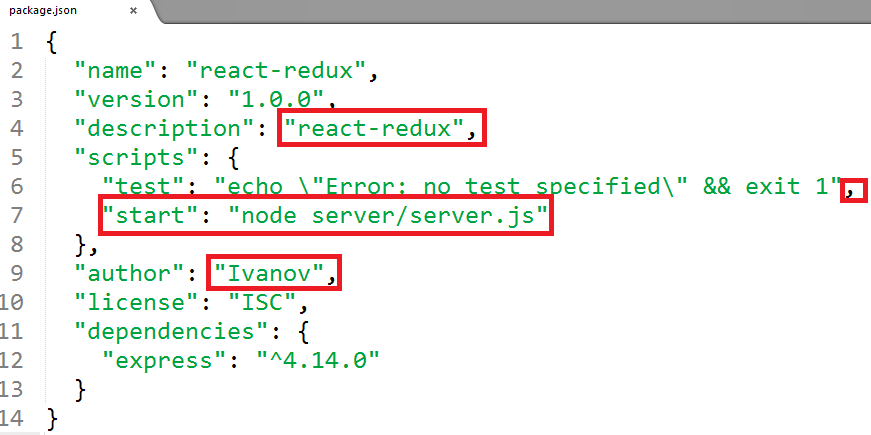
Настроим локальный веб-сервер, для этого скачаем фреймворк Express с помощью команды «npm install express --save» (ключ save добавит в файле package.json версию установленного express).



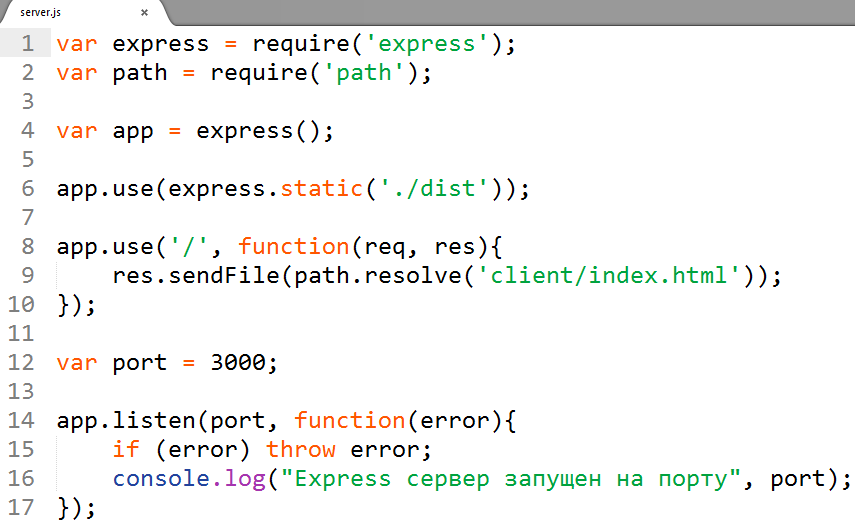
В файле package.json добавится новая зависимость и в каталоге проекта создастся папка node\_modules со скаченными и установленными модулями.



Уберите из файла package.json параметр main, добавьте какое-нибудь описание, новый скрипт start для запуска веб-сервера, укажите себя в качестве автора.

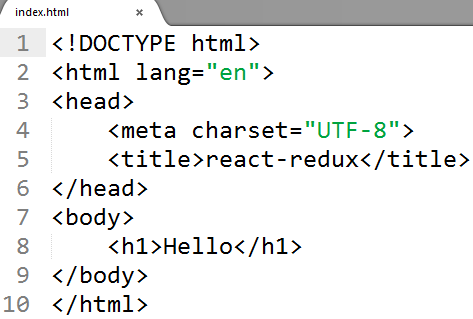


Сохраните изменения и создайте в папке server новый файл server.js со следующим содержимым.

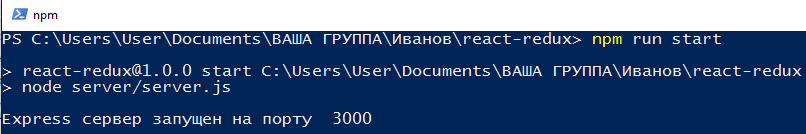


Здесь мы импортируем установленный модуль express и стандартный модуль path для работы с путями файлов и каталогов. Создаем объект экземпляр сервера с помощью вызова express(). Указываем серверу каталог со статическими файлами, в нашем случае это dist, в нем будут храниться клиентские js-файлы и css-стили. На 8 строке указываем серверу, чтобы на запрос «/» выполнялась функция с двумя параметрами (req – request/запрос, res – response/ответ), которая в ответ будет отправлять файл index.html. В конце указываем, на каком порту будет работать сервер.

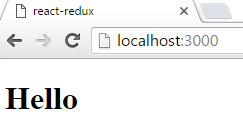
Сохраните изменения и создайте в папке client новый файл index.html со следующим содержимым.



Сохраните изменения и запустите сервер с помощью команды npm run start (где start название скрипта из файла package.json).



Перейдите в браузере по адресу localhost:3000.



Для того чтобы остановить веб-сервер нажмите сочетание клавиш Ctrl + C.

С помощью npm установите react и react-dom.



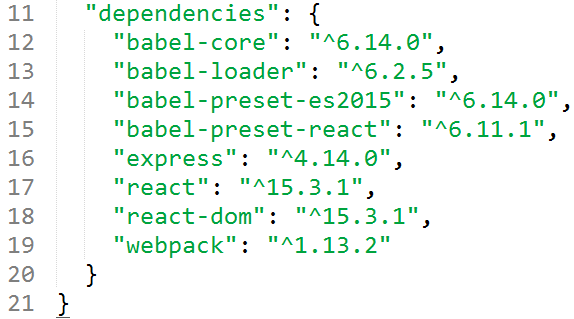
Для сборки клиентских скриптов будем использовать Webpack, установите его с помощью следующей команды.



Дополнительно из пакетов нам понадобится Babel, это специальный компилятор, который будет преобразовывать написанный нами JSX код из React и ECMAScript 6 в ECMAScript 5, для лучшей поддержки старых браузеров. Если перейти на официальный сайт Babel на страницу Setup, то можно найти необходимые пакеты. Для работы нам потребуются установить следующие пакеты: babel-core, babel-loader, babel-preset-es2015, babel-preset-react.

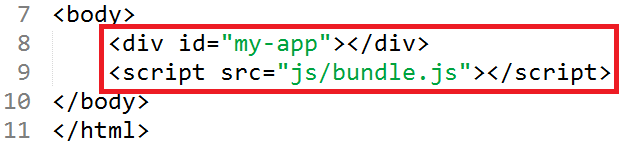


В итоге секция зависимостей в файле package.json будет выглядеть следующим образом.

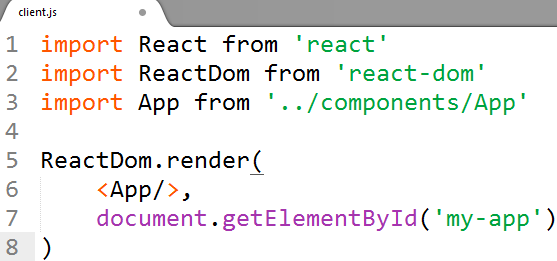


К моменту начала работы версии пакетов могут отличаться.

Добавьте в фале index.html код по образцу.

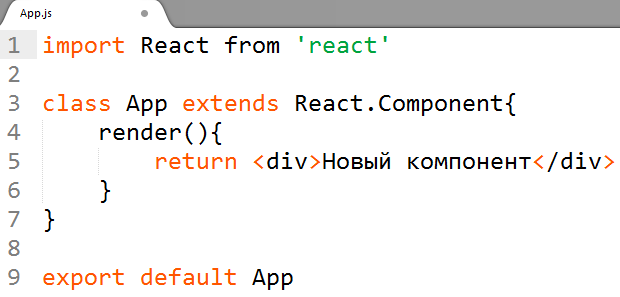


Сохраните и создайте в папке client новый файл client.js со следующим содержимым.



Здесь мы подключаем объекты React и ReactDom, используя синтаксис ES 6, и еще подключаем компонент App, файл который создадим чуть позже. Далее с помощью метода render у ReactDom вызываем отрисовку импортируемого компонента App и вторым параметром указываем элемент на странице, в котором будет производиться отрисовка.

Сохраните изменения и создайте в папке components новый файл App.js со следующим содержимым.



Здесь мы используем новый синтаксис ES 6 для создания класса App, который наследуем от Component из React. У компонента должен быть метод render, который возвращает разметку для отрисовки компонента (здесь используется не HTML код, а похожий по синтаксису JSX). Так как мы импортируем компонент App в другом файле (client.js), то вызываем для него export default.

Сохраните изменения и добавьте в папку проекта новый файл webpack.config.js со следующим содержимым.

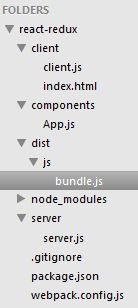


Здесь настраиваем конфигурацию для webpack, в которой указываем начальный файл client.js, параметры для выходного файла. В секции module указываем загрузчики для преобразования кода в ES 5. В параметре test вводим регулярное выражение для поиска js файлов, указываем имя загрузчика и путь к его файлам. В query указываем библиотеки, которые будут участвовать в преобразованиях.

Сохраните изменения и выполните команду webpack --config webpack.config.js для сборки файла bundle.js.



После этого в папке dist создаться каталог с файлом bundle.js, который подключен в index.html. Суть в следующем, у нас есть начальный файл client.js, в котором используются сущности из других файлов (App, React, ReactDom). Webpack анализирует все используемые зависимости и соединяет их в одном файле, дополнительно с помощью загрузчика Babel, производится преобразования JSX кода React и синтаксиса ES 6 в ES 5 для поддержки старых браузеров.

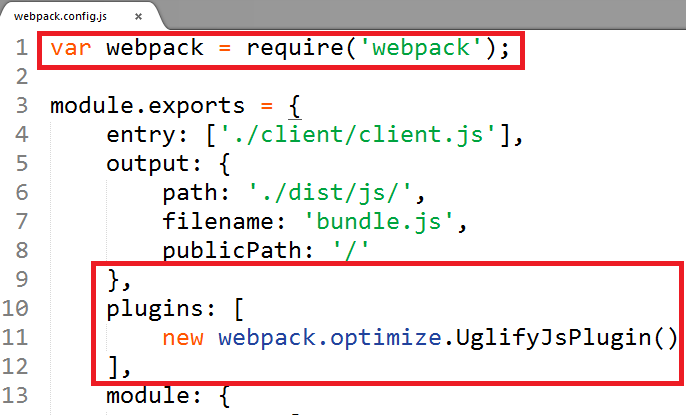
  
Запустите веб-сервер (npm run start) и проверьте в браузере отрисовку компонента.



Сейчас наш собранный файл bundle.js имеет довольно большой размер.

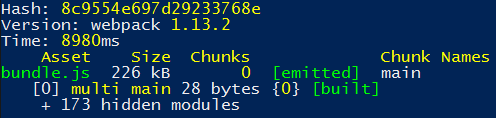


Мы можем сжать его с помощью webpack, используя плагин UglifyJsPlugin, который удалит пробелы и прочие не нужные символы. Введите код по образцу в файл webpack.config.js.



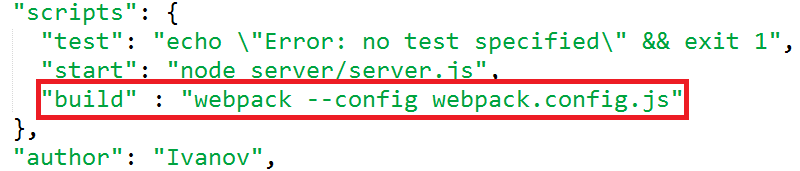
Сохраните изменения и повторно запустите сборку файла.





Сейчас собранный файл имеет размер в несколько раз меньше, что ускорит его загрузку. Так как процесс сжатия занимает определенное время, то на этапе разработки приложения его можно отключить.

Добавьте в файл package.json новый скрипт для упрощенного запуска webpack.



Сейчас сборку файла можно будет запускать командой npm run build.

Для начала реализуем только базовый функционал работы с файлами без использования регистрации и авторизации. Создайте в папке components новый файл AddDoc.js со следующим содержимым.

Сохраните изменения

React - это JavaScript библиотека для работы со слоем представления и разработанная в Facebook.

React часто упоминают в одном ряду с другими javascript фреймворками, но споры «React vs Angular» не имеют смысла, потому что это не сопоставимые вещи. Angular — это полноценный фреймворк (включающий и уровень представления). React — нет. Вот почему React вызывает столько непонимания в развивающемся мире полноценных фреймворков — это только представление.

React дает вам язык шаблонов и некоторые callback-функции для отрисовки HTML. Весь результат работы React — это HTML. Ваши связки HTML/JavaScript, называемые компонентами, занимаются тем, что хранят свое внутреннее состояние в памяти (например: какая закладка выбрана), но в итоге вам просто генерируется HTML.

Вместо сжатия реализуем функцию автоматической сборки bundle.js при внесении изменений в файлы из папки components, чтобы не запускать постоянно вручную команду webpack. Для этого установите библиотеку babel-preset-react-hmre.



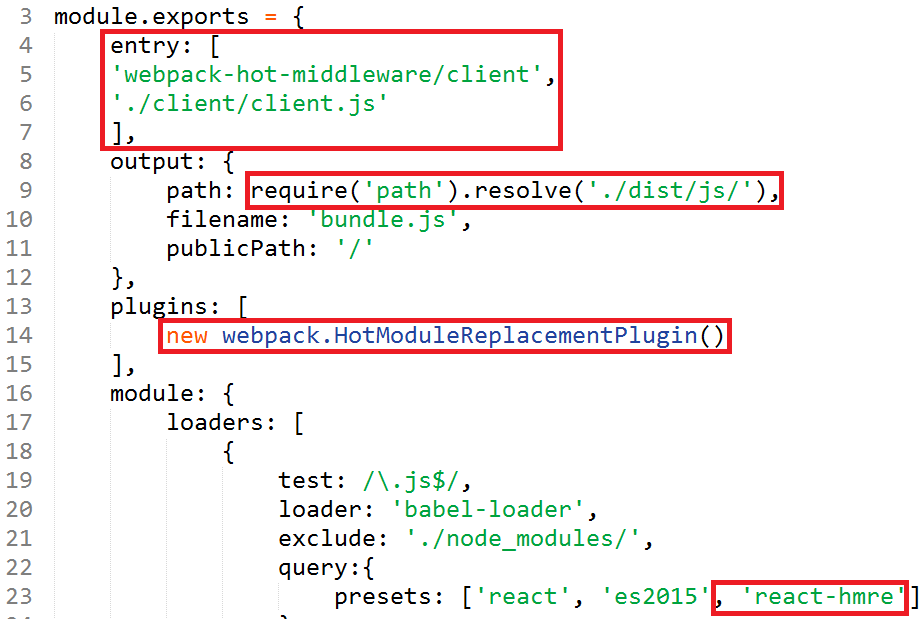
Установите с помощью npm пакет nodemon, глобально (ключ –g). Пакеты, установленные глобально, можно использовать в любом проекте без повторной установки. Мы будем использовать nodemon для запуска веб-сервера, вместо node. Nodemon отслеживает изменения в файлах и автоматически перезапускает веб-сервер.



Установите с помощью npm два пакета: webpack-hot-middleware, webpack-dev-middleware. Эти пакеты будут автоматически запускать webpack для сборки файлов и отображения изменений в браузере.

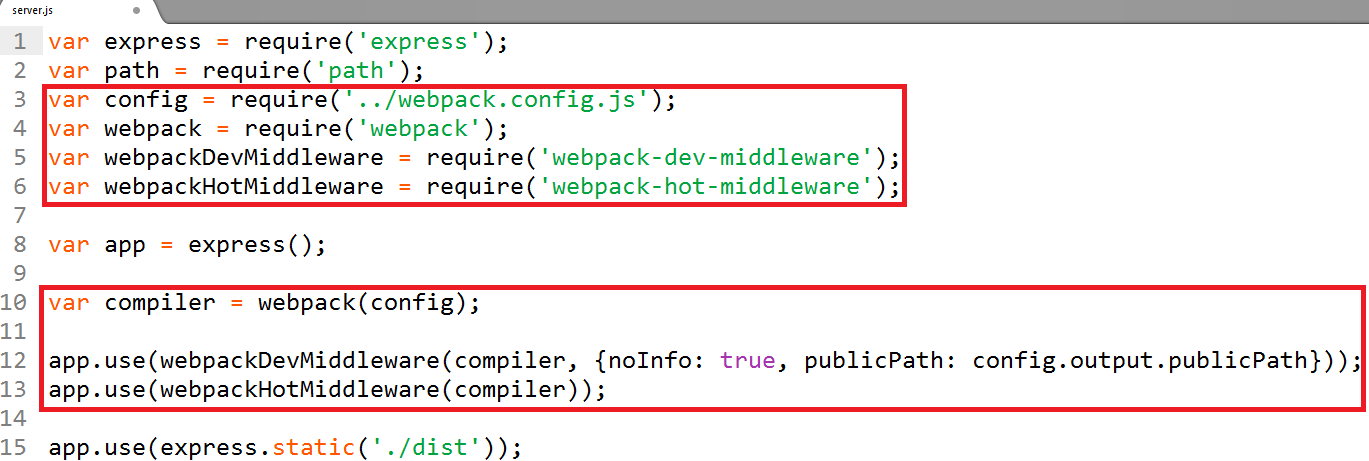


Внесите изменения в файле webpack.config.js по образцу.

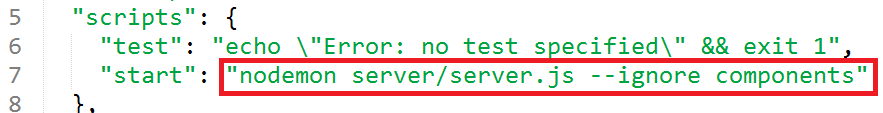


Здесь в параметре path требуется указать полный путь, а не относительный. Плагин HotModuleReplacementPlugin включает «горячую» замену модулей, что позволит обновлять React компоненты в браузере на страницы без её перезагрузки. Сохраните изменения.

Внесите изменения в файл server.js по образцу.



Сохраните изменения и внесите изменения в скрипт start из файла package.json.



Здесь мы говорим, чтобы сервер запускался с помощью nodemon, но чтобы изменения в файлах из папки components игнорировались, так как за ними будет следить webpack.

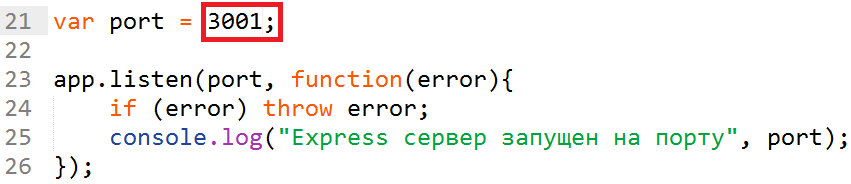
Сохраните изменения, запустите webpack вручную.

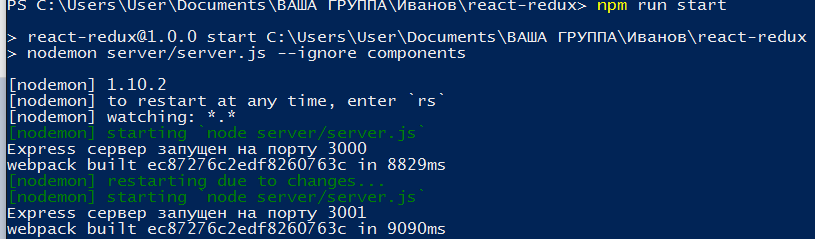


Запустите сервер.

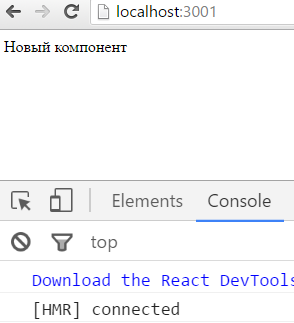


Сейчас если мы внесем изменения в какой-либо файл, кроме папки components, то nodemon перезапустит сервер.





Если вы откроете консоль браузера, то увидите информацию о работе модуля HMR, который сейчас отслеживает изменения компонентов React.



Если вы внесете изменения в файл App.js и сохраните, то webpack автоматически изменит файл bundle.js и изменения отобразятся в браузере.