

Курс React JS

На данном курсе рассмотрим

1. фрейворк (библиотеку) React JS
2. поймем технологию виртуального DOM
3. поймем state
4. props
5. что такое React компонент
6. сделаем обзор IDE для редактирования кода javascript
7. обзор инструментов для отладки
8. рассмотри построение компонентов на JSX
9. коснемся webpack
10. babel
11. рассмотрим стандарт ES6 языка javascript
12. рассмотри библиотеку Redux по управлению state – ми компонентов, React-а
13. сделаем обзор Flux pattern-а, сравним его с Redux
14. рассмотри promise – ы в fetch
15. рассмотри конструкцию async в fetch
16. рассмотри инструмент git
17. рассмотрим инструмент для работы с хранилищем git
18. рассмотри React hooks
19. рассмотрим, сделаем обзор CSS фреймворков для React (material-ui)
20. разберем некоторые примеры
21. разработаем формочки по аутентификации пользователя
22. рассмотрим (сделаем обзор) инструменты для прототипирования макетов мобильных приложений

Для начала, перед тем как начнем установим

1. VS code ( <https://code.visualstudio.com/> )
   1. Установим расширения для удобной работы для javascript
   2. Пропишем в path ( C:\Program Files\Microsoft VS Code\bin )
2. Установим NodeJS ( <https://nodejs.org/en/> ), пропишем в path ( C:\Program Files\nodejs\ )
3. Установим Chocolatey ( <https://chocolatey.org/> )
4. Установим yarn ( <https://yarnpkg.com/lang/en/> ), пропишем в path ( C:\Program Files (x86)\Yarn\bin\ )
5. Установим git ( https://git-scm.com/downloads )
6. Установим Sourcetree ( <https://www.sourcetreeapp.com/> )
7. Установим ADOBE XD CC ( <https://www.adobe.com/ru/products/xd.html> )

======================================================================

1.

Начнем ( <https://facebook.github.io/create-react-app/> , <https://reactjs.org/> )

**Quick Start**

npx create-react-app my-app1

cd my-app

yarn start

2.

Коснемся webpack, babel ( <https://habr.com/ru/company/ruvds/blog/436886/> ), для понятия что такое webpack, babel

Рассмотрим, как запустить тоже самое, но с использованием webpack, babel, react

|  |
| --- |
| Webpack Webpack используется для компиляции JavaScript-модулей. Этот инструмент часто называют «бандлером» (от bundler) или «сборщиком модулей». После его [установки](https://webpack.js.org/guides/installation) работать с ним можно, используя [интерфейс командной строки](https://webpack.js.org/api/cli) или его [API](https://webpack.js.org/api/node). Если вы не знакомы с Webpack — рекомендуется почитать об [основных принципах](https://webpack.js.org/concepts) его работы и посмотреть его [сравнение](https://webpack.js.org/comparison) с другими сборщиками модулей. Вот как, на высоком уровне, выглядит то, что делает Webpack.  https://habrastorage.org/getpro/habr/post_images/dc0/e21/470/dc0e21470d18e7abc53a1c7a087b2807.png  *Работа Webpack*  Webpack берёт всё, от чего зависит проект, и преобразует это в статические ресурсы, которые могут быть переданы клиенту. Упаковка приложений — это очень важно, так как большинство браузеров ограничивает возможности по одновременной загрузке ресурсов. Кроме того, это позволяет экономить трафик, отправляя клиенту лишь то, что ему нужно. В частности, Webpack использует внутренний кэш, благодаря чему модули загружаются на клиент лишь один раз, что, в итоге, приводит к ускорению загрузки сайтов. |

|  |
| --- |
| Babel Babel — это транспилятор, который, в основном, используется для преобразования конструкций, принятых в свежих версиях стандарта ECMAScript, в вид, понятный как современным, так и не самым новым браузерам и другим средам, в которых может выполняться JavaScript. Babel, кроме того, умеет преобразовывать в JavaScript и JSX-код, используя [@babel/preset-react](https://babeljs.io/docs/en/babel-preset-react).  https://habrastorage.org/getpro/habr/post_images/385/f39/809/385f398094efe878d5b031f9177e1084.jpg  *Babel*  Именно благодаря Babel мы, при разработке React-приложений, можем пользоваться JSX. Например, вот код, в котором используется JSX:  import React from "react";  function App(){  return(   <div>      <b>Hello world!</b>   </div>  )  }  export default App;  Выглядит такой код аккуратно, он понятен, его легко читать и редактировать. Глядя на него, сразу можно понять, что он описывает компонент, возвращающий элемент <div>, в котором содержится текст Hello world!, выделенный жирным шрифтом. А вот пример кода, делающего то же самое, в котором JSX не используется:  "use strict";  Object.defineProperty(exports, "\_\_esModule", {   value: true  });  var \_react = require("react");  var \_react2 = \_interopRequireDefault(\_react);  function \_interopRequireDefault(obj) { return obj && obj.\_\_esModule ? obj : { default: obj }; }  function App(props) {   return \_react2.default.createElement(     "div",     null,     \_react2.default.createElement(       "b",       null,       "Hello world!"     )   );  }  exports.default = App;  Преимущества первого примера перед вторым очевидны. |

|  |
| --- |
| Предварительные требования Для того чтобы настроить проект React-приложения, нам понадобятся следующие npm-модули.   * react — библиотека React. * react-dom — библиотека, которая поможет нам использовать возможности React в браузере. * [babel](https://habr.com/ru/users/babel/)/core — транспиляция JSX в JS. * [babel](https://habr.com/ru/users/babel/)/preset-env — создание кода, подходящего для старых браузеров. * [babel](https://habr.com/ru/users/babel/)/preset-react — настройка транспилятора для работы с React-кодом. * babel-loader — настройка Webpack для работы с Babel. * css-loader — настройка Webpack для работы с CSS. * webpack — сборка модулей. * webpack-cli — работа с Webpack из командной строки. * style-loader — загрузка всего используемого CSS-кода в заголовке HTML-файла. * webpack-dev-server — настройка сервера разработки. |

Приступим

1. mkdir my-app2
2. cd my-app2
3. yarn init
4. yarn add @babel/core @babel/preset-env @babel/preset-react babel-loader css-loader webpack webpack-cli style-loader webpack-dev-server --dev

тем самым в папке my-app2 будет создан файл package.json

далее

1. yarn add --dev @babel/core @babel/preset-env @babel/preset-react babel-loader react-hot-loader webpack webpack-cli webpack-dev-server
2. yarn add react react-dom

сам проект лежит: <https://github.com/uzerp2/CourseReactJS> (https://github.com/uzerp2/CourseReactJS.git)

<https://facebook.github.io/create-react-app/>

<http://facebook.github.io/react-native/>

<https://material-ui.com/>

<https://mlab.com/>

<https://www.mockable.io/>

<https://code.visualstudio.com/>