# Основы React

## Запуск первого React App

Делаем установку с [https://github.com/facebook/create-react-app через *node.js*](https://github.com/facebook/create-react-app%20через%20node.js) *command prompt.*

Переходим с помощью вышеупомянутой консольной утилиты в папку с будущим проектом и прописываем следующие вещи:

npx create-react-app my-app

cd my-app

npm start

* npm install – загрузит папку node\_modules. Грузит модули на основании package.json
* \*npx в отличии от npm не только установит модули, но и подготовит их.
* \*вместо my-app можно свое название проекта
* package.json сообщает node.js, что папка является проектом в терминологии node.js

В файле App.js в функции App() должен быть только один корневой элемент

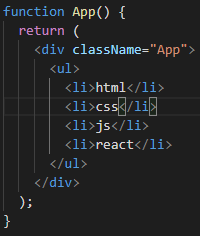
Ctrl+F+K – автоформатирование

Ctrl+Shift+F – поиск

Ctrl+K+C – комментировать в JSX

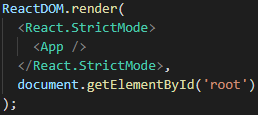
## Компонента

Компонента – это функция, возвращающая разметку JSX. Пример:

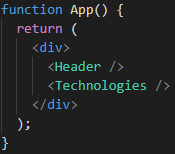
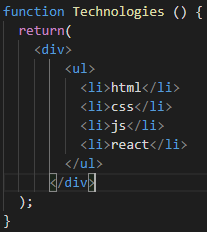
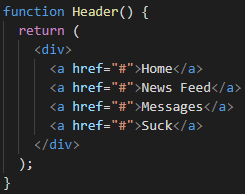


Соответственно появится тег <App/>. То есть также можно сказать, что компонента - это еще и тег.

Этот тег мы вставляет в файле *index.js*:



Одни такой тег можно вставить в другой тег, вот пример в App.js:

## Импорт/Экспорт

В реальном проекте каждая компонента (Technologies, Header), будет лежать в отдельном файле.

* ./ - относительно текущего местоположения
* ./.. – на уровень выше относительно текущего местоположения
* Импорт CSS отличается от импорта JS модулей: import './App.css';
* Модули из папки *node\_modules* импортируются по-другому:

import React from 'react'; - не нужно ставить никаких точек, нужная папка сама найдется

* Чтобы использовать синтаксис JSX нужно сделать импорт:

import React from 'react';

* Импорт картинки: import logo from './logo.png'; Вставка:

<img src={logo} alt='d' className="logo" />

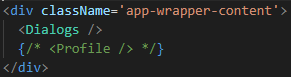
## Организация CSS

Для каждой компоненты нужно создать свой css файл. И подключить уже внутри каждой компоненты свой css файл.

* Чтобы css был изолирован в своих компонентах нужно назвать их так: name.module.css.
* Импорт такого css: import classes from './name.module.css'. classes – это объект, ключами которого являются названия стилей, а значения – сгенерированные для них автоматические названия:

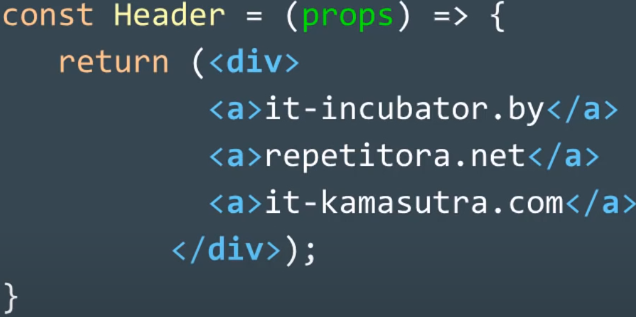


* Вот так применяются классы: <div className={classes.sideBar}>
* Два класса можно применить с помощью форматной строки: {`${s.class1} ${s.class2}`}
* Компонент и css группируем в одну папку для удобства
* Если у SPA приложения есть странички, которые меняются, но свойства контейнера у них одинаковые, их можно положить в тег div и вынести для него общие css свойства:



## Props

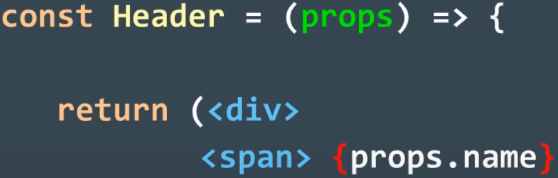
Пропсы – это параметр функции-компонента, которые содержит некоторые данные для нее:

 В качестве пропса всегда приходит какой-то объект. Когда вызываем так: <Header /> - то в пропсе будет пустой объект.

Чтобы передать объект с параметрами используются атрибуты компоненты:

  Каждый атрибут со значением становится парой ключ-значение объекта-параметра.

Использование:



**Все атрибуты тега-компоненты превращаются в Пробсы!**

## route, browser-router, маршрутизация

Система роутинга отслеживает адрес в адресной строке и будет реагировать на изменения в ней.

* Чтобы настроить эту систему нужно установить соответствующий модуль: npm install --save react-router-dom

save – внесет запись в package.json информацию об установленном нами модуле и после этого при *npm install* этот модуль будет автоматически загружаться в папку *node\_modules.*

### <Route/> - компонента

* Не забыть импортировать модули: import { BrowserRouter, Route } from 'react-router-dom';
* Все приложение (например внутри App.js) нужно обернуть в тег-компоненту <BrowserRouter >
* Пути настраиваются с помощью тега-компоненты <Route /> Пример:

<Route path='/dialogs' component={Dialogs} />

<Route path='/profile' component={Profile} />

### <NavLink /> - компонента

* import { NavLink } from 'react-router-dom';
* <Navlink><Navlink /> используется вместо тега <a></a> чтобы менять URL и не перезагружать страницу снова, что по умолчанию делает <a>. Пример: <NavLink to='/profile'>Profile</NavLink>
* Атрибут activeClassName={obj.class} указывает какой класс применить для активной ссылки. По умолчанию навлинку ставится активный класс class=’active’