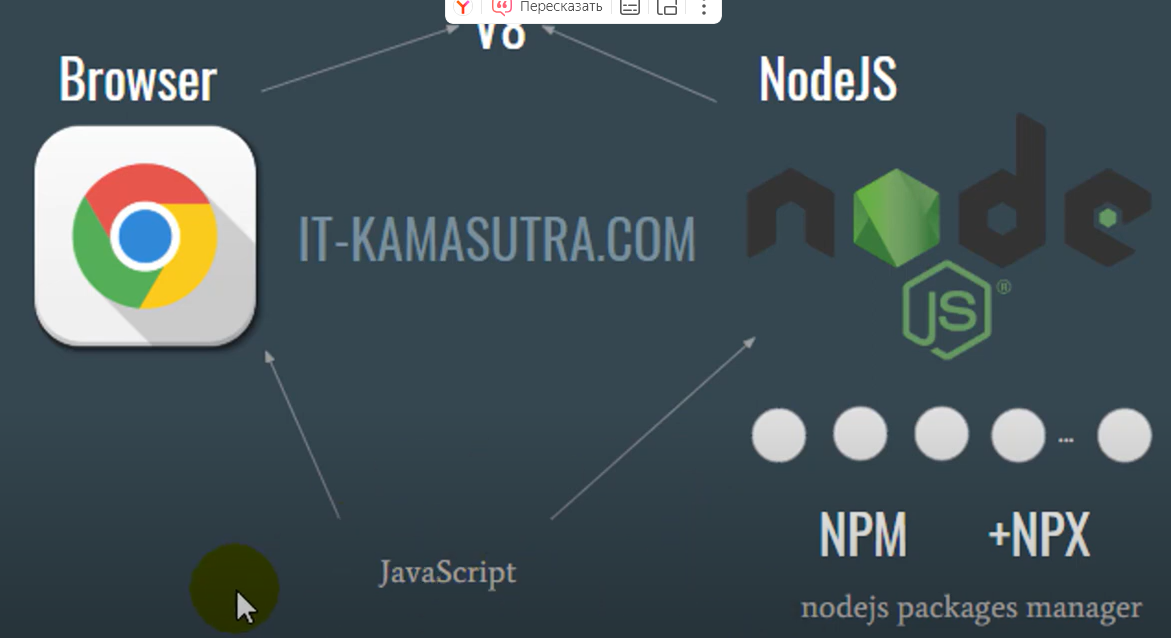
<https://www.youtube.com/watch?v=TPYgQvY9VVQ&list=PLcvhF2Wqh7DNVy1OCUpG3i5lyxyBWhGZ8&index=5>



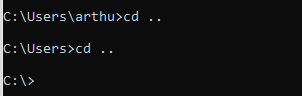
Браузер и node.js являются одинаковыми программами для обработки js кода. Браузер более красивый (визуальная часть), а node.js это консоль (серверная часть, базы данных).

https://github.com/facebook/create-react-app

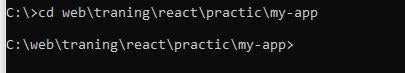
Первым делом необходимо установить Node.js.

1. Необходимо установить node.js
2. Заходим в консоль Node.js через поиск, заходим папку в которой хотим разместить проект.
3. Вводим команды

Для начала необходимо создать папку и перейти в нее в консоли node.js. Спуститься на уровень ниже:



Заходим в папку:



После перехода в созданную папку, вводим команды

npx create-react-app my-app

cd my-app

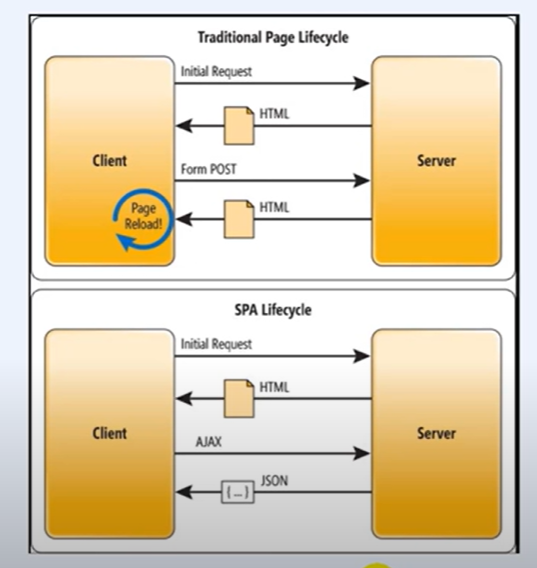
npm start

Если выходят ошибки, значит необходимо реакт установить глобально

При помощи команды:

npm install -g react-scripts

Single page application (SPA)



SPA – это небольшая HTML страница с большим кол-м JS при помощи которого эта страница динамически наполняется контентом.

JSX – специальный язык, который позволяет писать html код внутри JS

# Компонента

Компонента – это функция, возвращающая разметку JSX. Это тег. Название компонента обязательно должно быть с большой буквы.



import './App.css';

Это значит, что App.css импортируется в наш файл на нашем уровне.

const App = () => {

  return (

    <div className="App">

      <div>hello</div>

      <ul>

        <li>html</li>

        <li>css</li>

      </ul>

    </div>

  );

}

В return должен быть общий div или любой другой элемент, но он должен быть единственным, а все остальное внутри данного элемента.

App – это компонента. Данная функция (компонента) вызывается в файле index.js при помощи тега <App />. В переменную root попадает элемент из index.html, затем в этот элемент помещается содержимое <App /> и сервер отправляет пользователю index.html внутри которого много js кода.

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

**<App />**

  </React.StrictMode>

);

Также можно вставлять одну компоненту в другую

const App = () => {

  return (

    <div>

      <Header />

      <Technologies />

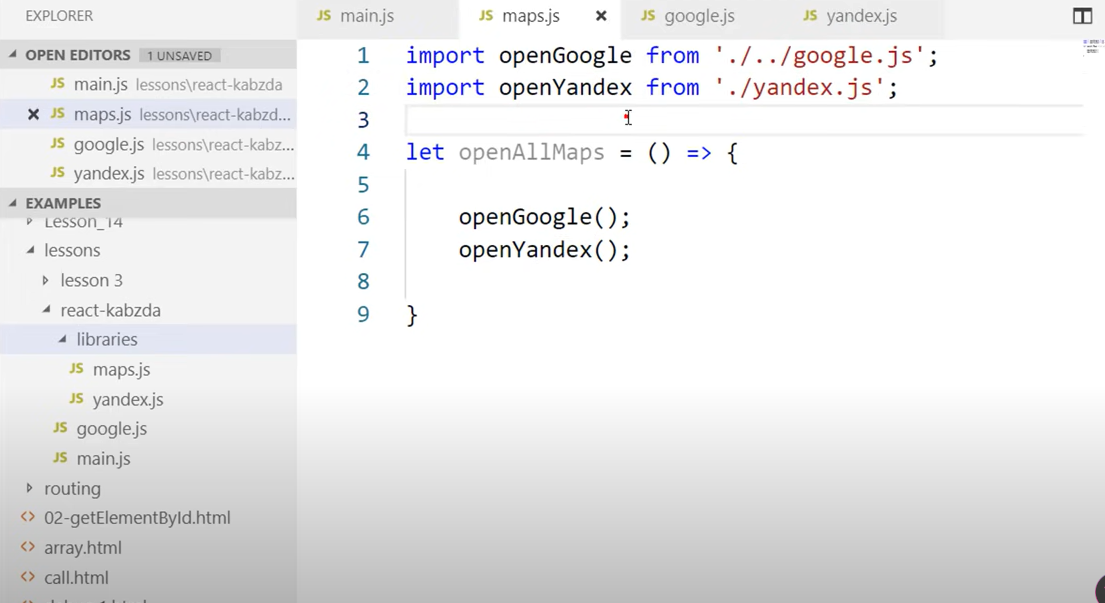
    </div>

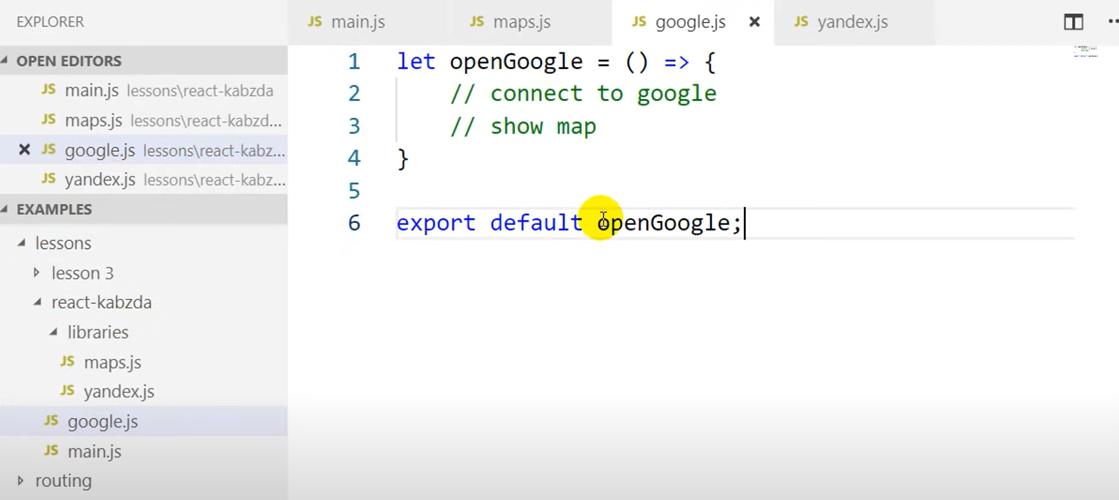
  );

}

# import\export – теория

# Вызываем функцию openGoogle из файла google.js. Для этого данную функцию нужно проимпортировать. Путь прописываем относительно файла в котором вызывается функция.

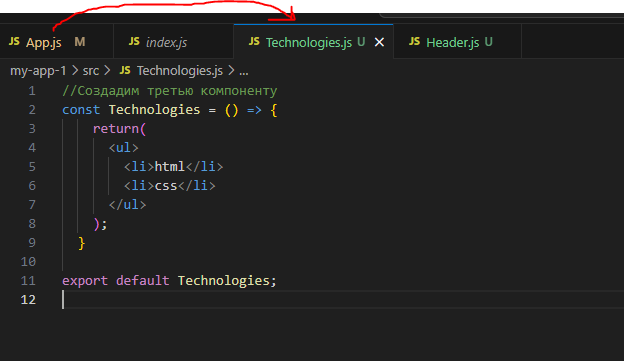




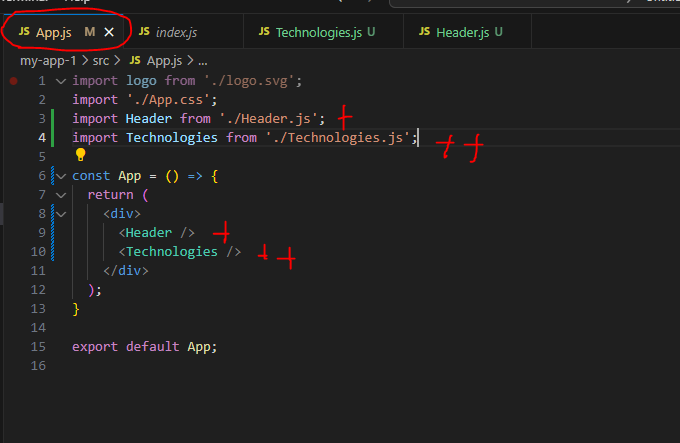
Но для того чтобы что-то импортировать нужно что-то экспортировать.

Для автоматического проставления import нужно установить плагин auto import!!!!!

ПЕРЕНОС КОМПОНЕНТЫ В ОТДЕЛЬНЫЙ JS ФАЙЛ!!!!



Импортируем его в App.js



В react приложении компоненты в отдельных файлах недоступны если не сделать import и export!!!

**Проблемы с default**

Может быть такая ситуация.

import logo from './logo.svg';

import './App.css';

//import Header from './Header.js';

import Technologies from './Header.js';

const App = () => {

  return (

    <div>

      <Technologies />

    </div>

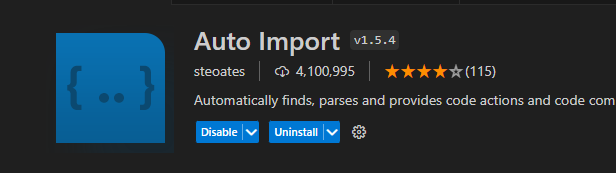
  );

}

export default App;

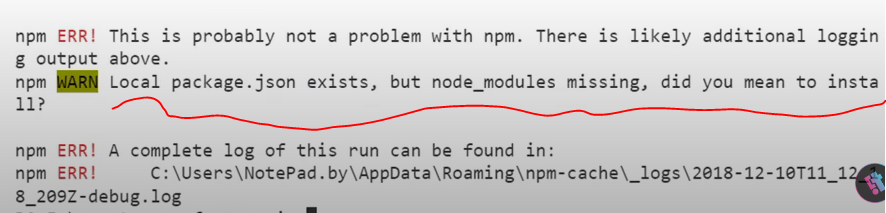
Т.е мы импортируем компоненту Technologies из файла Header.js и вызываем. Но данной компоненты в файле Header.js НЕТ. В результате будет выводится компонента из файла Header.js. ЭТО ПРОИСХОДИТ ИЗ-ЗА export. DEFAULT – это значит, что мы импортируем что-то. На месте Technologies можно написать что угодно

Установим расширение auto\_import

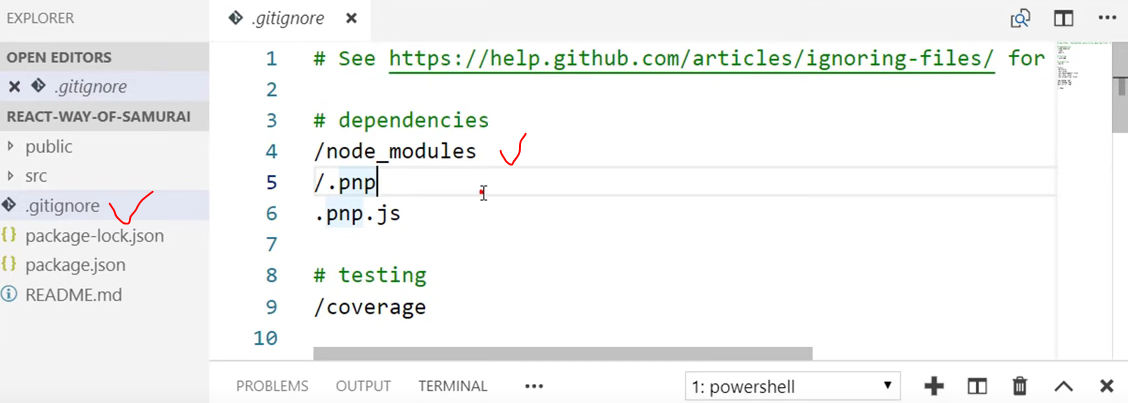


Почитать про module AMD require

Папку node\_modules не заливают на GitHub (никогда). Поэтому, когда клонируем проект и запускаем его при помощи команды git clone может выскочить ошибка.



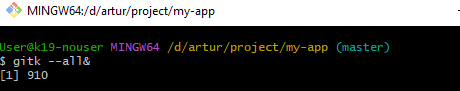
При заливке на GitHub название этой папки прописывается в .gitignore, чтобы она не попала в облачное хранилище.



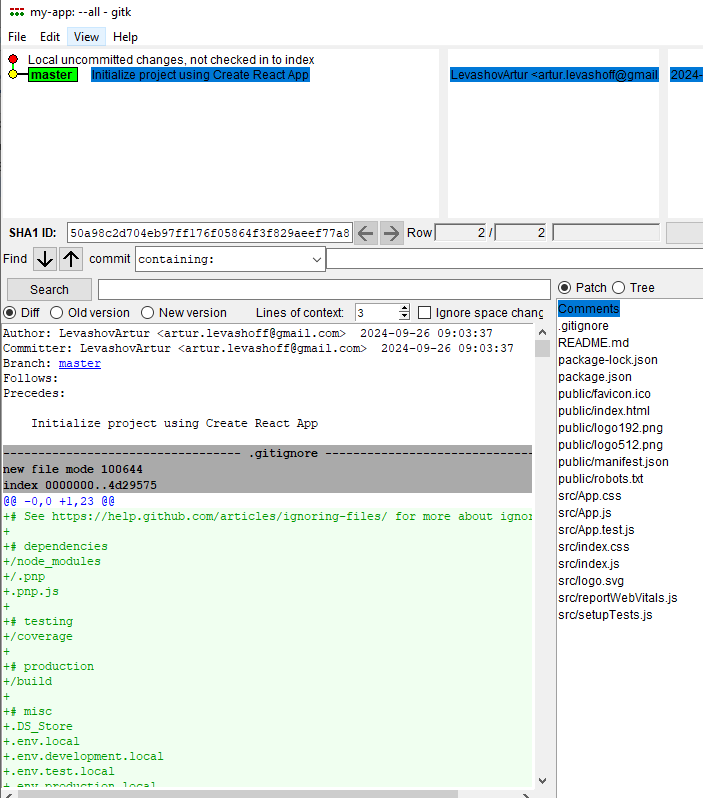
npm install – подгружаем папку node\_modules в проект

Файл package.json – самый основной, в нем прописываются все зависимости

Команда для просмотра веток коммита



Результат



Нужно скопировать ключ SHA1 ID при помощи команды ctrl+insert и вставить SHIFT+INSERT в командную строку git checkout (SHA1 ID)

**Css module**

В стандартном шаблоне react за стили отвечает файл APP.CSS и он подключен в APP.JS. Но что если мы хотим создать отдельные файлы css для каждого шаблона.

CSS модуль — это CSS файл, в котором все имена классов и анимаций имеют локальную область видимости по умолчанию.

Что это значит?

* Первым делом нужно наш css файл переименовать. Header.css -> Header.module.css
* Затем нужно подключить этот css к нашему файлу js

import styles from './Navbar.module.css'

После того как это было сделано, наши стили видоизменились. В файле они прописаны так:

.nav {

*grid-area*: n;

*background-color*: burlywood;

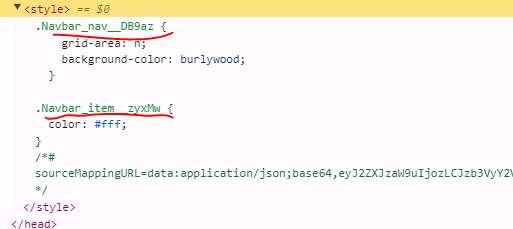
  }

.item {

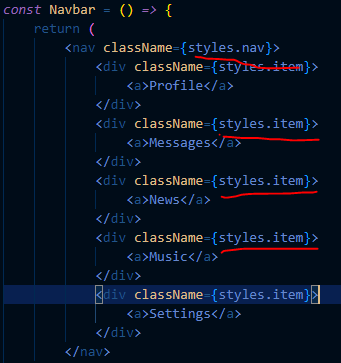
*color*: #fff;

}

Преобразовались в такой вид:



Соответственно подключать нужно в файл jsx таким образом



Оператор *import* в JavaScript файле загружает CSS файл и конвертирует его в объект. Каждое имя класса из CSS файла является свойством объекта (ключ). А значение этого ключа это уникальное имя класса, и уникальность обеспечивает то, что стили не «протекают» в другие компоненты. Вот пример хешированного имени класса: *\_header\_\_1OUvt*.

Т.е мы получаем объект

let styles = {

nav: “Navbar\_nav\_\_DB9az”,

item: “Navbar\_item\_\_zyxMw”

}

Желательно не использовать дефис в названии стилей в модулях поскольку необходимо тогда обращаться

let styles = {

nav-item: “Navbar\_nav\_\_DB9az”,

item-s: “Navbar\_item\_\_zyxMw”

}

{style[“nav-item”]}

* Если элементу принадлежат сразу 2 класса

<div class=”main strong”>

Css (если у элемента есть класс strong)

.main.strong {

font-weight: bold

}

На react это будет выглядеть так

<div className={`${styles.main} ${styles.strong}`}>

Или второй способ

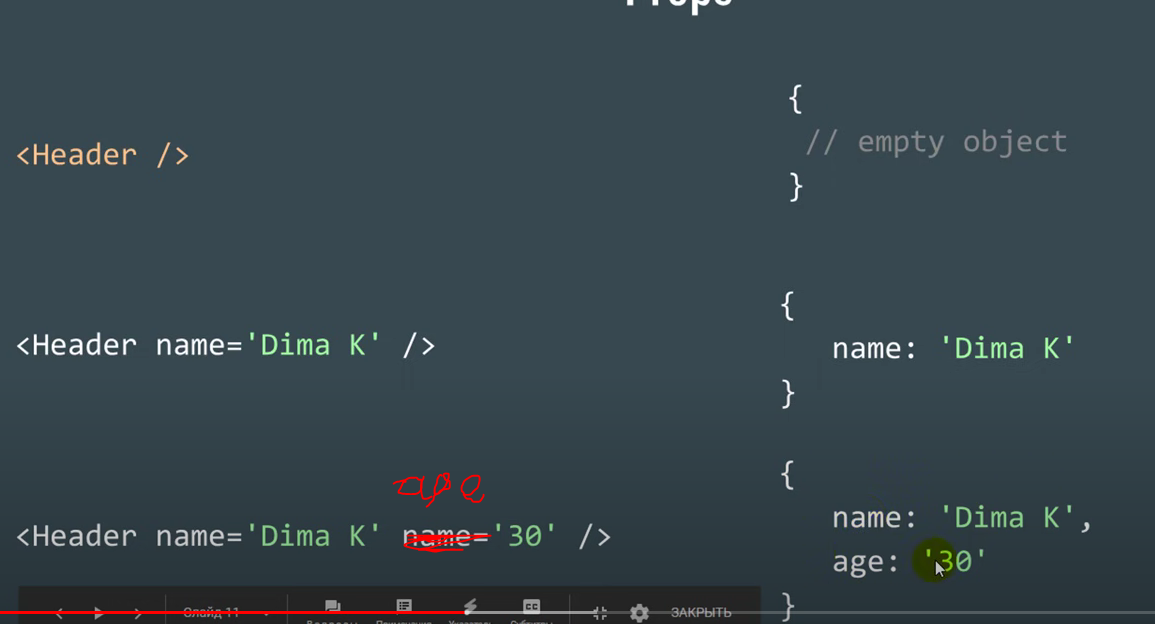
<div className={styles.dialog + ' ' + styles.active}>

# PROPS

По сути props является параметром для компоненты.



Компонента вызывается при помощи тега. Если у тега нет параметров, значит в качестве props будет пустой объект { }, если укажем параметры, то они станут свойствами объекта. Параметры можно указывать любые.



Пример



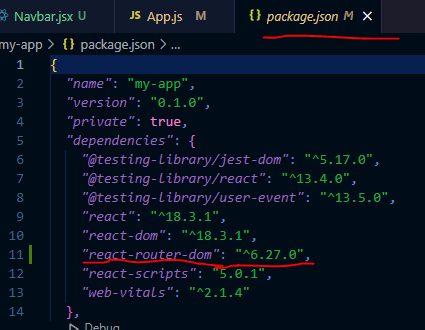
# Route, browser-router, маршрутизация

Routing – это маршрутизация, router – маршрутизатор

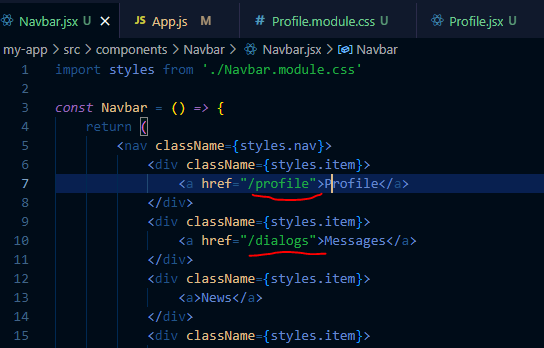
Разработчики CRA – create react app не включили в сборку routing, поэтому этот пакет нужно подгрузить в наш проект в папку node\_modules.

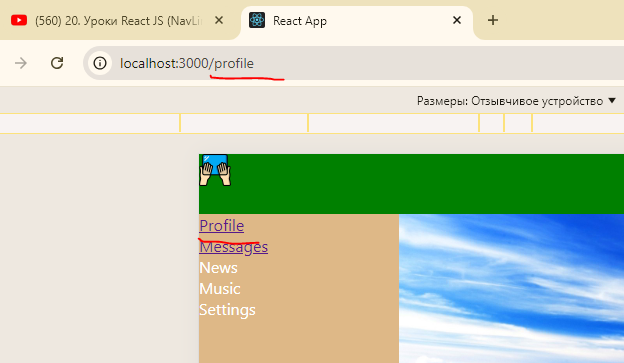


Сам пакет устанавливает в папку node\_modules, но при загрузке проекта в репозиторий папка node\_modules игнорируется, а следовательно и наш пакет тоже. При загрузке с репозитория route работать не будет и придется вручную подкачивать этот файл. Для того, чтобы решить эту проблему необходимо прописать -save. Эта команда занесет изменения в файл package.json. Это позволит при помощи одной команды npm i подгрузить все пакеты необходимые для работы программы.



Прописываем маршруты по которым мы сможем обратиться к нашим страницам





Теперь необходимо научить реакт понимать какую из перечисленных страниц нужно выбирать при клике по ссылке.



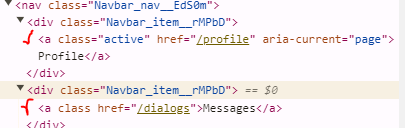
Обрамляем всю конструкцию обязательными тегами <BrowserRouter>, <Routes>. Они проимпортируются автоматически или прописать как написаны в import. Далее необходимо прописать path=”” и сам element=””. Данная конструкция отлавливает адрес в адресной строке, который мы прописали в теге <a>

НО ПРИ ПЕРЕХОДЕ НА ДРУГУЮ СТРАНИЦУ САЙТ ПЕРЕЗАГРУЖАЕТСЯ. ДЛЯ ТОГО ЧТОБЫ ЭТОГО ИЗБЕЖАТЬ НУЖНО ВМЕСТО <a> ПРОПИСАТЬ <NAVLINK>

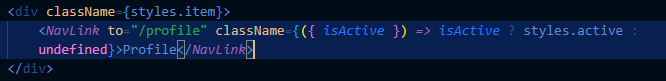
Для использования NavLink необходим import.



Но в режиме разработчика <navlink> отображается как <a>



Если хотим добавить активное состояние ссылке



Css

.item a.active {

*color*: gold;

}