

JS Level 3



при поддержке



REACT

React

[React](#) - это библиотека, предназначенная для создания UI (User Interface - пользовательский интерфейс).

Что такое библиотека? Это готовый код, который мы можем подключить к своему проекту и использовать. Это позволяет нам не с нуля писать весь проект, что в свою очередь значительно ускоряет время создания проекта.

React

Основная идея React строится вокруг компонентов: мы разбиваем наш интерфейс на блоки, которые обладают определённой функциональностью. Каждый блок - это компонент.

Когда у нас есть готовые компоненты, мы можем уже собирать весь интерфейс проекта из этих компонентов (как из кубиков или из элементов лего).

Компоненты

Когда мы используем HTML, то на самом деле, мы так же собираем весь интерфейс из компонентов (в HTML они называются элементы):

```
<body>
  <main>
    <article>
      <h1>Video</h1>
      <video src="video.mp3" controls>
      <footer>
        <button>Like</button>
        <button>Share</button>
      </footer>
    </article>
  </main>
</body>
```

При этом компоненты представляют из себя дерево (т.е. они вложены друг в друга и у каждого компонента есть только один родитель, а у каждого родителя может быть от 0 детей).

У некоторых компонентов ([video](#)) есть свойства (атрибуты): [src](#) и [controls](#), которые мы можем назначать, чтобы изменить поведение. Например, [src](#) отвечает за то, какое видео будет проигрываться, а [controls](#) - будут ли показываться элементы управления ([play](#), [pause](#) и т.д.)

HTML Elements

Все элементы, которые понимает браузер, описаны в специальном документе, который называется [спецификация](#):

§ 4.8.9 The **video** element

Content attributes:

[Global attributes](#)

src — Address of the resource

crossorigin — How the element handles crossorigin requests

poster — Poster frame to show prior to video playback

preload — Hints how much buffering the [media resource](#) will likely need

autoplay — Hint that the [media resource](#) can be started automatically when the page is loaded

playsinline — Encourage the user agent to display video content within the element's playback area

loop — Whether to loop the [media resource](#)

muted — Whether to mute the [media resource](#) by default

controls — Show user agent controls

width — Horizontal dimension

height — Vertical dimension

HTML Elements

Конечно же, хотелось бы, чтобы мы могли создавать свои подобные элементы и использовать их.

HTML Elements

Например, если мы посмотрим, на страницу Яндекса:



Фильмы



Нало 4: Идущий
к рассвету



Без тормозов



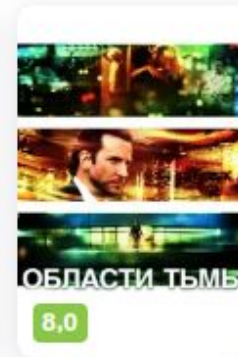
Волк с Уолл-
стрит



Балканский
рубеж



Отель Мумбаи:
Противостояние



Области тьмы



Охранник

Было бы здорово, если бы каждый элемент можно было задавать как-нибудь вот так:

```
<film src="poster.jpg" rating="6.8" name="Нало 4: Идущий к рассвету">
```


Web Components

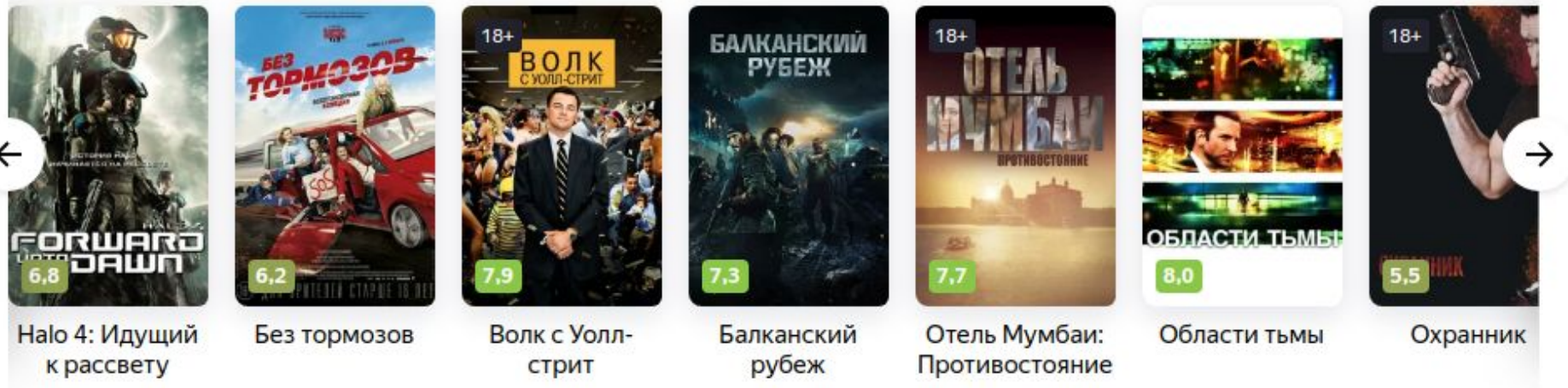
В своё время, для реализации подобной функциональности была придумана технология Web Components.

Несмотря на то, что она действительно работает, реализация Web Component'ов - не такая уж простая задача, поэтому появились решения, которые позволяют добиться подобного же поведения более простыми средствами.

Итого

Основная задача React - дать нам создавать собственные компоненты, чтобы можно было делать вот так:

Фильмы



`<film src="poster.jpg" rating="6.8" name="Halo 4: Идущий к рассвету">`

APP

App

В React уже есть один компонент, он называется `App` и располагается в файле `App.js`:

```
JS App.js  X
src > JS App.js > ...
1  import React from 'react';
2  import logo from './logo.svg';
3  import './App.css';
4
5  function App() {
6    return (
7      <div className="App">
8        <header className="App-header">
9          <img src={logo} className="App-logo" alt="logo" />
10         <p>
11           Edit <code>src/App.js</code> and save to reload.
12         </p>
13         <a
14           className="App-link"
15           href="https://reactjs.org"
16           target="_blank"
17           rel="noopener noreferrer"
18         >
19           Learn React
20         </a>
21       </header>
22     </div>
23   );
24 }
25
26 export default App;
```

Это основной компонент, который загружается React при старте приложения (именно его мы и видели при старте - возвращающаяся иконка React).

import/export

В React'е используется система модулей ESM. Что это для нас значит? Это значит, что каждый файл с расширением `.js` представляет себя изолированный код, который может делать две вещи:

1. Что-то импортировать (инструкция `import`), чтобы использовать код, объявленный вне этого файла
2. Что-то экспортировать (инструкция `export`), чтобы другие модули могли использовать код из этого файла.

import/export

В простейшем случае: мы импортируем React и компоненты, а экспортируем тот компонент, который объявлен в файле (инструкция `export default <имя компонента>`).

Давайте приведём наш компонент к следующему виду (удалим всё ненужное):

```
JS App.js  ×
src > JS App.js > ...
1  import React from 'react';
2
3  function App() {
4    return (
5      <div className="App">
6        </div>
7    );
8  }
9
10 export default App;
```

Компонент

Получается, что в React'е компонент - это просто функция, которая возвращает что-то очень сильно похожее на HTML (но это не HTML):

```
JS App.js  ×  [redacted]  
src > JS App.js > ...  
1  import React from 'react';  
2  
3  function App() {  
4    return (  
5      <div className="App">  
6        </div>  
7    );  
8  }  
9  
10 export default App;
```

Если вы не знаете, что такое функция, вам нужно ознакомиться с материалами JS Level 1.

Компонент

Запустим наш проект. Для этого в консоли VS Code (**Ctrl + `**) введите команду **npm start**. Теперь вместо вращающегося логотипа React мы получим чистую пустую страницу.

Но если мы нажмём **Ctrl + Shift + I** (либо **F12**), то на вкладке Elements мы увидим то, что возвращали из компонента **App**:



```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      ... <div class="App"></div> == $0
    </div>
```


Компонент

Обратите внимание на 2 вещи:

1. В компоненте мы писали `className`, а в результате получили `class`
2. В полученном результате нет ничего от нашего компонента кроме полученной разметки

Дело в том, что React берёт на себя всю "грязную" работу по превращению наших компонентов в элементы DOM. И в DOM никаких React компонентов просто не существует.

Компонент

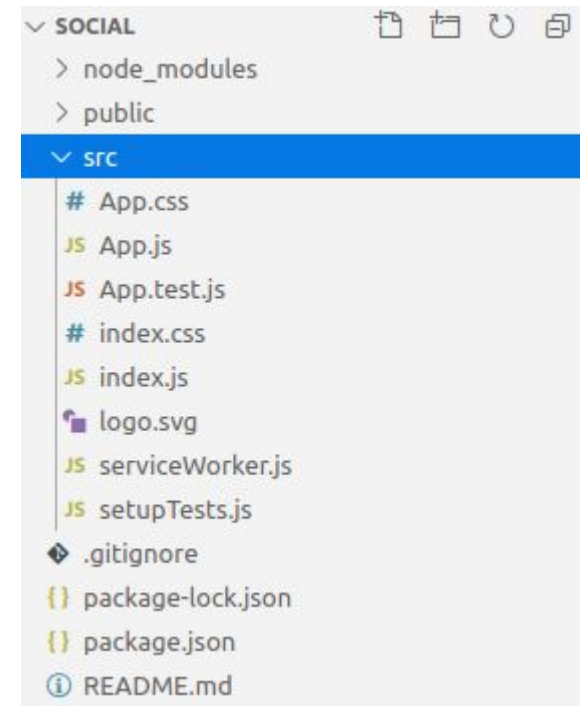
А почему `className`, а не `class`? Дело в том, что в рамках DOM свойство, через которое устанавливается имя класса (CSS-класса) элемента называется `className`, а не `class`. Но в разметке оно отображается как `class`. Поэтому именно для этого атрибута приходится писать `className` (будут ещё исключения).

ПЕРВЫЙ КОМПОНЕНТ

Первый компонент

Итак, давайте попробуем создать наш первый собственный компонент. Для этого вам нужно создать проект через create-react-app (далее - CRA).

CRA сразу создаёт за вас структуру файлов и каталогов. Мы с вами ещё будем её детально разбирать, пока же важно следующее: вы заходите в каталог `src` и все изменения делаете там:



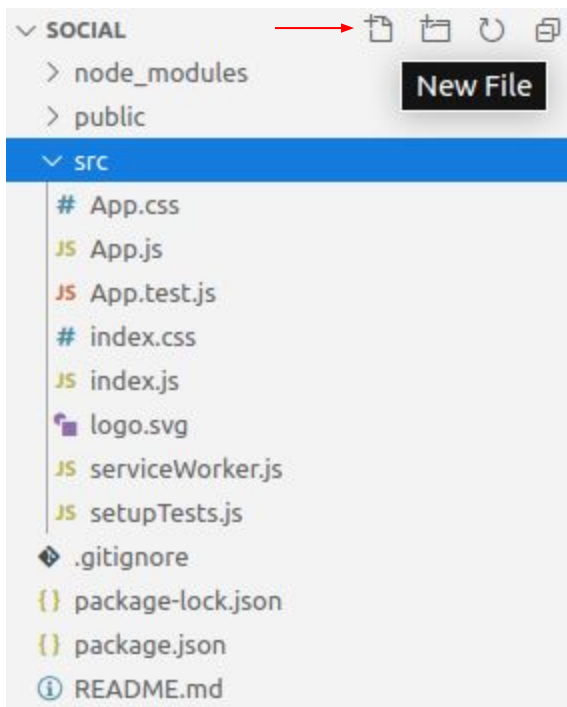
Социальная сеть

Мы будем делать социальную сеть
наподобие Vk, поэтому первый и
ключевой компонент, который нам нужен
- это пост. Мы хотим карточку поста
представлять в виде React Component'a:



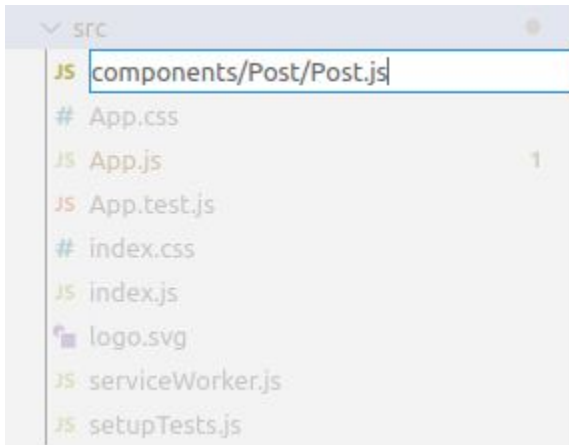
Первый компонент

Принято, что каждый компонент располагается в отдельном файле. Поэтому переходим в панельку проекта и нажимаем на кнопку +:



Первый компонент

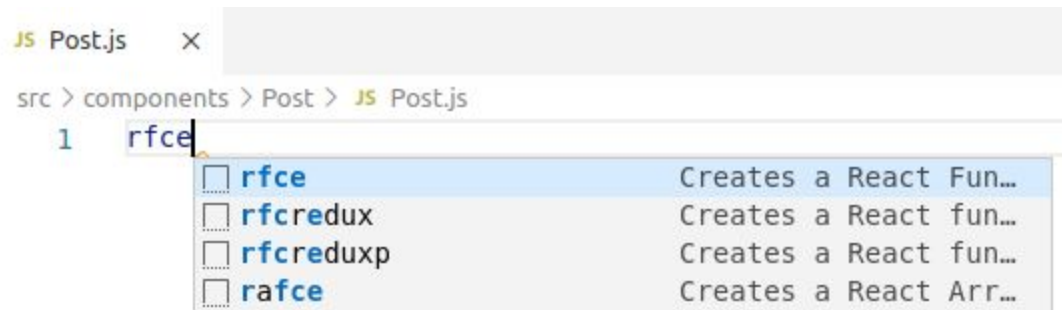
Далее, вы сразу вводите имя вместе с именем каталога:



Важно: именуруйте файлы именно так, как мы показываем. Компоненты в каталоге `components`, для каждого компонента свой каталог (с большой буквы) и каждый компонент хранится в своём файле (тоже с большой буквы).

Первый компонент

Благодаря расширению, которое мы поставили на прошлой лекции, мы можем удобно генерировать код компонента через snippet **rfce**:



Сниппет - это просто некое сокращение, которое вы набираете, затем дожидаетесь, когда будет показана подсказка и нажимаете **Tab**. Если подсказка не появилась, то жмите **Ctrl + пробел**.

Первый компонент

В итоге то, что мы написали (+ **Tab**) приведёт к вот такому коду:

```
JS Post.js x
src > components > Post > JS Post.js > Post
1  import React from 'react'
2  ⚡
3  function Post() {
4    return (
5      <div>
6        |
7      </div>
8    )
9  }
10
11 export default Post
```

Нажмите **Enter**, чтобы завершить редактирование.

Обратите внимание: это расширение достаточно умное, оно сразу называет компонент так же, как ваш файл (вот почему файлы нужно называть правильно).

В React компоненты должны называться именно с большой буквы! Иначе ничего работать не будет (именно так React будет отличать ваши компоненты от встроенных, которые описывают элементы HTML).

Первый компонент

Отредактируйте файл, чтобы он выглядел следующим образом:

```
JS Post.js  X  [ ]
src > components > Post > JS Post.js > ...
1  import React from 'react'
2
3  function Post() {
4    return (
5      <div>
6        Наш Пост
7      </div>
8    )
9  }
10
11  export default Post
```

Компонент

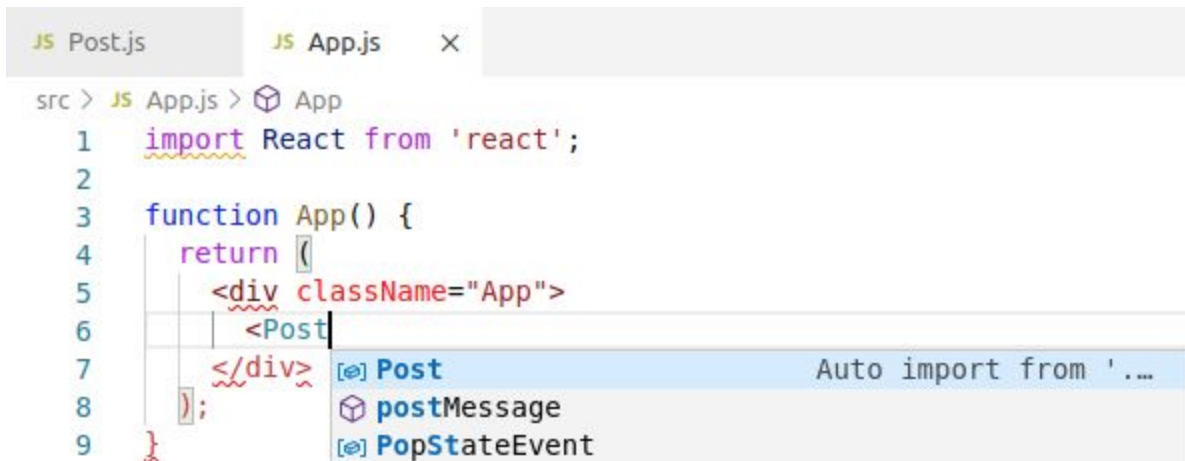
Чтобы наш компонент отображался на странице (а сейчас он там не отображается), нужно его разместить в компоненте `App` (или любом другом компоненте, который уже подключен в `App`).

Поскольку никаких других компонентов, кроме компонента `App` у нас нет, то мы идём в компонент `App` и размещаем там свой компонент.

Компонент

Важный момент: когда мы написали компонент `Post`, мы можем встроить его в разметку React с помощью тега: `<Post />` (именно такого, а не `<Post>`).

При этом когда вы будете набирать, не надо писать всё целиком, VS Code вам будет давать возможность использовать автодополнение:



```
JS Post.js JS App.js ×
src > JS App.js > App
1  import React from 'react';
2
3  function App() {
4    return (
5      <div className="App">
6        <Post
7      </div>
8    );
9  }
```

Autocomplete dropdown for `<Post`:

- `Post` (Auto import from '...')
- `postMessage`
- `PopStateEvent`

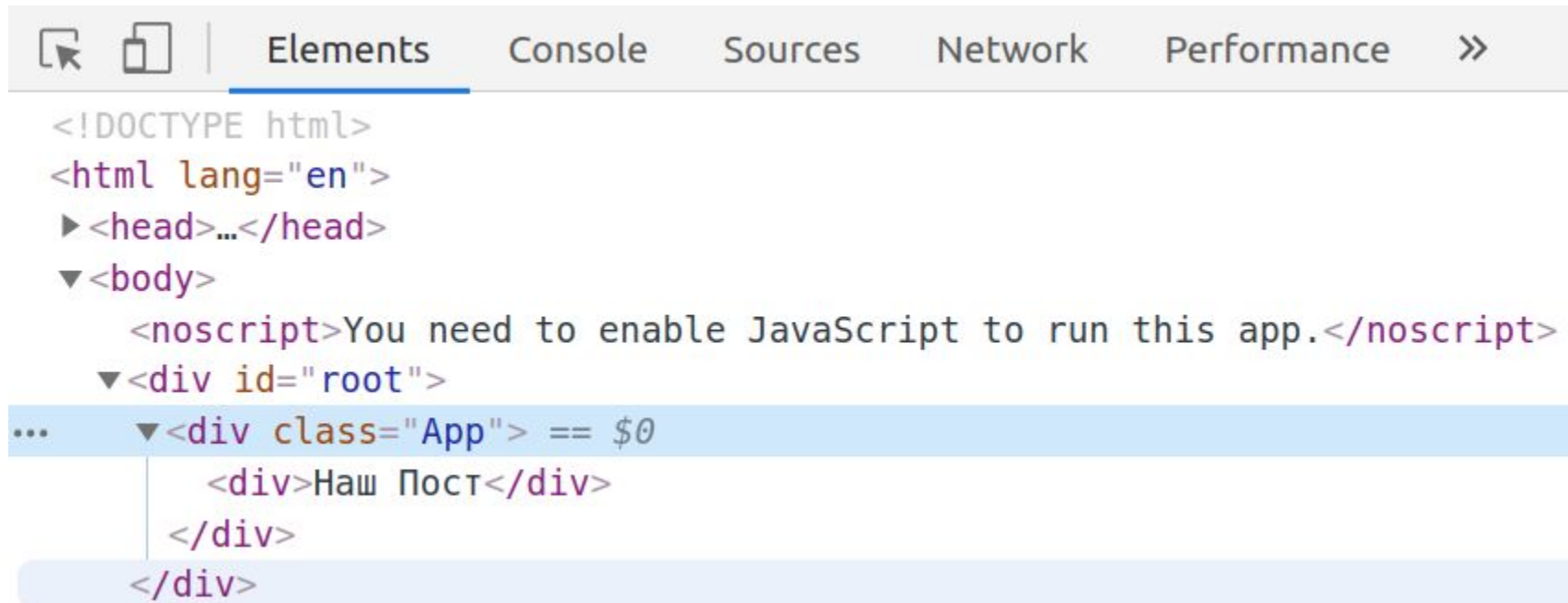
Компонент

Если сейчас нажать **Tab** (нажимайте **Ctrl + пробел**, если подсказка пропала), то VS Code допишет за вас тег и сам добавит инструкцию **import** (**/>** придётся дописать самим):

```
src > JS App.js > ...
1  import React from 'react';
2  import Post from './components/Post/Post';
3
4  function App() {
5    return (
6      <div className="App">
7        <Post />
8      </div>
9    );
10 }
11
12 export default App;
```

Результаты

Смотрим на полученный результат: действительно, пост появился:



```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      <div class="App"> == $0
        <div>Наш Пост</div>
      </div>
    </div>
```

Props

Конечно, это не очень интересно. Хотелось бы, так же как с элементом `video` или примером с фильмами, что-то передавать в `Post`. Как это сделать? Да очень просто: так же как с `video`, мы просто пишем `<Post content="Наш пост" />`:

```
src > JS App.js > ...
1  import React from 'react';
2  import Post from './components/Post/Post';
3
4  function App() {
5    return (
6      <div className="App">
7        <Post content="Наш пост" />
8      </div>
9    );
10 }
11
12 export default App;
```

Обратите внимание, название атрибута (`content`) мы придумали сами.

Props

Передать-то мы передали, но теперь, как получить переданные значения и отобразить их? Здесь тоже всё достаточно просто: всё, что мы передаём через тег, автоматически собирается в объект и кладётся в первый аргумент функции (его принято называть `props`):

```
src > components > Post > JS Post.js > ...  
1  import React from 'react'  
2  
3  function Post(props) {  
4    return (  
5      <div>  
6        Наш Пост  
7      </div>  
8    )  
9  }  
10  
11  export default Post
```


Props

А чтобы отобразить это значение в разметке, используется специальный синтаксис с фигурными скобками:

```
src > components > Post > JS Post.js > ...  
1  import React from 'react'  
2  
3  function Post(props) {  
4    return (  
5      <div>  
6        {props.content}  
7      </div>  
8    )  
9  }  
10  
11  export default Post
```

Важно: мы назвали свойство `content`, оно попало под именем `content` в `props`, поэтому мы и обращаемся к нему как `props.content`.

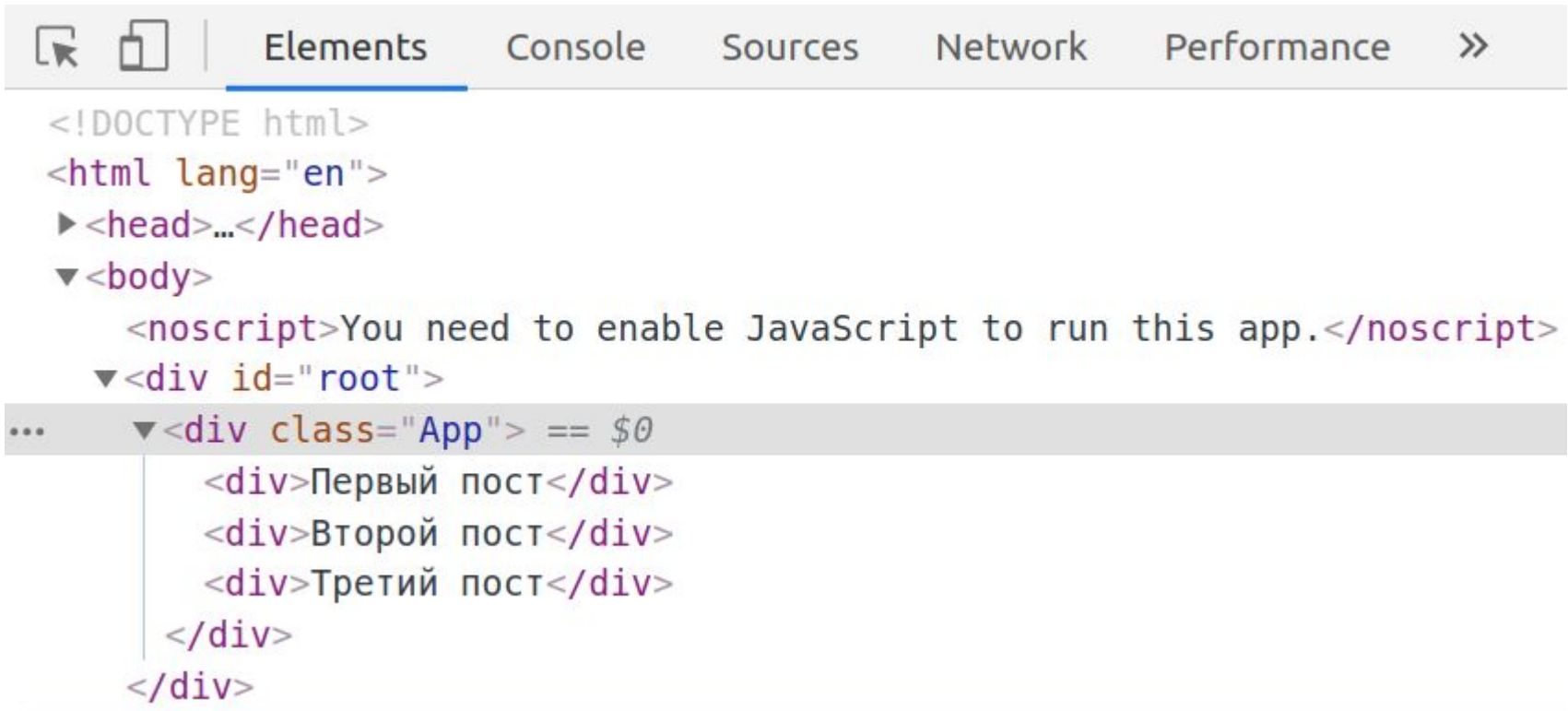
Components

Что это нам дало? Теперь мы можем отображать сразу несколько постов, просто **меняя свойство `content`**:

```
src > JS App.js > ...
1  import React from 'react';
2  import Post from './components/Post/Post';
3
4  function App() {
5    return (
6      <div className="App">
7        <Post content="Первый пост" />
8        <Post content="Второй пост" />
9        <Post content="Третий пост" />
10     </div>
11   );
12 }
13
14 export default App;
```

Т.е. мы как раз создали тот самый переиспользуемый компонент.

Результаты



```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      ...
      <div class="App"> == $0
        <div>Первый пост</div>
        <div>Второй пост</div>
        <div>Третий пост</div>
      </div>
    </div>
```

ИТОГИ

ИТОГИ

В этой лекции мы обсудили достаточно много важных моментов и создали свой первый React компонент.

Дальше дело за малым - изучить все возможности компонентов, в том числе по взаимодействию их друг с другом.

ДОМАШНЕЕ ЗАДАНИЕ

Д3: Messages

Помимо постов, в нашей социальной сети будут ещё личные сообщения (как и в Vk).

Создайте компонент `Message`, который отображает переданное ему свойство `text`.

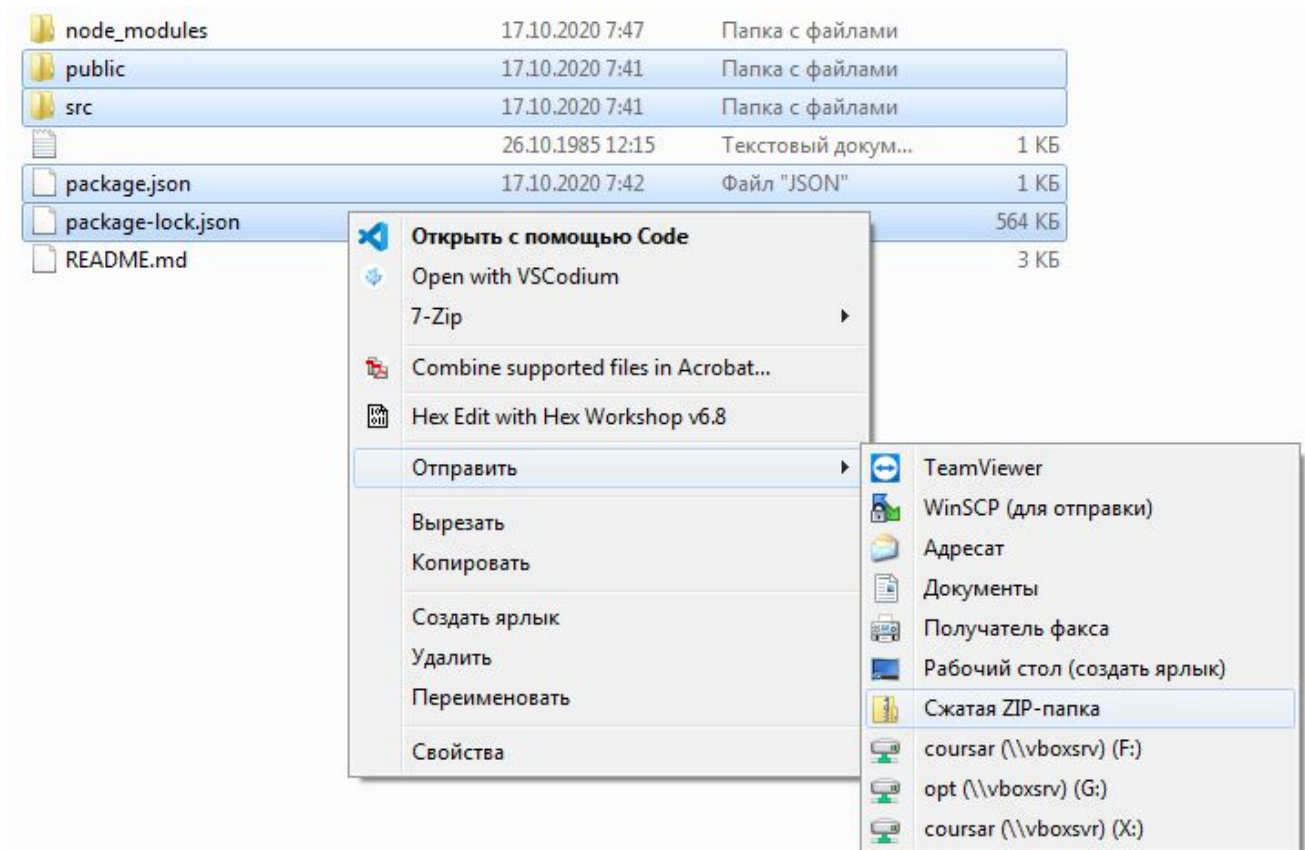
В компоненте `App` разместите три сообщения со следующим текстом:

- Первое сообщение
- Второе сообщение
- Третье сообщение

```
function App() {  
  return (  
    <div className="App">  
      <Post content="Первый пост" />  
      <Post content="Второй пост" />  
      <Post content="Третий пост" />  
      <Message text="Первое сообщение" />  
      <Message text="Второе сообщение" />  
      <Message text="Третье сообщение" />  
    </div>  
  );  
}
```

Как сдавать ДЗ

Вам нужно запаковать в zip-архив ваш проект те файлы и каталоги, которые указаны на скриншоте ниже. Для этого выберите их, нажмите правую кнопку мыши и выберите Отправить → Сжатая ZIP-папка:



Как сдавать ДЗ

Полученный архив загружаете в личном кабинете пользователя.

Важно: учитывается только последняя отправленная попытка.

Спасибо за внимание

alif academy совместно с aims
2020г.