Условный рендер

Цель работы

Познакомить и получить навыки в реализации условного рендера.

Задания для выполнения

1. На сегодняшнем занятии по условному рендерингу мы поговорим об использовании логического оператора && (И). Экспериментировать будем со стандартным проектом, созданным средствами create-react-app, в файле App.js которого находится следующий код:

```
import React, {Component} from "react"

class App extends Component {
    constructor() {
        super()
        this.state = {
            unreadMessages: [
                "Call your mom!",
                "New spam email available. All links are definitely safe to
click."
            ]
        }
    }

    render() {
        return (
            <div>

<h2>You have {this.state.unreadMessages.length} unread messages!</h2>
            </div>
        )
    }
}

export default App
```

Возможно, вы уже пользовались оператором && в конструкциях наподобие true && false (что даёт false). Для того, чтобы в результате вычисления подобного выражения было бы возвращено true, оно должно выглядеть как true && true. При обработке таких выражений JavaScript определяет, является ли их левая часть истинной, и, если это так, просто возвращает то, что находится в их правой части. Если обрабатывается выражение вида false && false, то сразу будет возвращено false, без проверки правой части выражения. В результате оператор && можно использовать в условном рендеринге. С его помощью можно либо вернуть что-то, что будет выведено на экран, либо не возвращать ничего.

Проанализируем код учебного приложения.

В состоянии компонента App хранится массив строк unreadMessages. Каждая строка в этом массиве представляет собой непрочитанное сообщение. На страницу выводится количество непрочитанных сообщений, определяемое на основе длины массива. Если же этот массив будет пустым, то есть в нём не будет ни одного элемента, то приложение выведет на страницу 0.

Для того чтобы добиться такого эффекта, достаточно привести массив к такому виду: unreadMessages: [].

Если непрочитанных сообщений нет, то вполне можно не выводить вообще никакого сообщения. Если воспользоваться для реализации такого поведения приложения тернарным оператором, о котором мы говорили в прошлый раз, метод render() компонента App можно переписать так:

```
render() {
    return (
        <div>
            {
                this.state.unreadMessages.length > 0 ?

<h2>You have {this.state.unreadMessages.length} unread messages!</h2> :
                null
            }
        </div>
    )
}
```

Теперь в том случае, если массив unreadMessages пуст, на страницу не будет выводиться ничего. Но представленный здесь код можно упростить благодаря использованию оператора &&. Вот как это будет выглядеть:

```
render() {
    return (
        <div>
            {
                this.state.unreadMessages.length > 0 &&

<h2>You have {this.state.unreadMessages.length} unread messages!</h2>
            }
        </div>
    )
}
```

Техническое задание

Добавьте на страницу, которую формирует компонент, кнопку, которая позволяет пользователю входить в систему и выходить из неё.

Сделайте так, чтобы, если пользователь не вошёл в систему, на кнопке выводилась бы надпись LOG IN, а если вошёл — надпись LOG OUT.

Выведите на странице, формируемой компонентом, надпись Logged in в том случае, если пользователь вошёл в систему, и Logged out в том случае, если не вошёл.

Добавьте остальные компоненты формы.

2. Загрузить созданную страницу на GitHub в репозиторий tera, используя формат в названии Фамилия (латинскими буквами)_14.

Методические указания

Чтобы выводить на страницу текст, зависящий от того, вошёл пользователь в систему или нет, можно воспользоваться кодом:

```
render() {
    let buttonText = this.state.isLoggedIn ? "LOG OUT" : "LOG
    IN"
    let displayText = this.state.isLoggedIn ? "Logged in" :
    "Logged out"
    return (
        <div>
            <button onClick={this.handleClick}>{buttonText}
            </button>
            <h1>{displayText}</h1>
        </div>
```

Дополнительные задания:

1. Добавьте другие компоненты React на страницу.

Полезные ссылки:

https://scrimba.com/scrim/c893vh2?pl=p7P5Hd

https://habr.com/ru/company/ruvds/blog/443210/