Компоненты, основанные на классах

Цель работы

Получить навыки работы с классами.

Задания для выполнения

1. Перепишите код любого (кроме первого) компонента на сайте https://reactjs.org/ через функцию:

2. Перепишите код, используя класс

```
function Comment(props) {
  return (
    <div className="Comment">
      <div className="UserInfo">
        <img className="Avatar"
          src={props.author.avatarUrl}
          alt={props.author.name}
        />
        <div className="UserInfo-name">
          {props.author.name}
        </div>
      </div>
      <div className="Comment-text">
        {props.text}
      </div>
      <div className="Comment-date">
        {formatDate(props.date)}
      </div>
    </div>
  );
}
```

3. Загрузить созданную страницу на GitHub в репозиторий tera, используя формат в названии Фамилия (латинскими буквами)_15.

Методические указания

Как вы уже знаете, в React мы можем задавать компоненты как функции или как классы.

Сейчас актуально направление, в соответствии с которым везде, где это возможно, используют функциональные компоненты, а компоненты, основанные на классах — лишь там, где они действительно необходимы. При этом надо отметить, что всё это — лишь рекомендации. Каждый разработчик сам решает как именно он будет конструировать свои приложения.

Классы в JavaScript представляют собой надстройку над традиционной моделью прототипного наследования. Сущность конструкции class App extends React.Component сводится к тому, что мы объявляем новый класс и указываем на то, что его прототипом должен быть React.Component. Наличие у нашего компонента этого прототипа позволяет пользоваться в этом компоненте всеми теми полезными возможностями, которые имеются в React.Component.

У компонента, основанного на классах, должен быть, по меньшей мере, один метод. Это — метод render(). Данный метод должен возвращать то же самое, что мы обычно возвращаем из функциональных компонентов.

Сравните этот код:

```
function App() {
    return (
        <div>
            <h1>Code goes here</h1>
        </div>
    )
}
```

И этот:

```
class App extends React.Component {
    render() {
        return (
            <div>
                <h1>Code goes here</h1>
            </div>
        )
    }
}
```

Если, перед формированием элементов, возвращаемых этим методом, нужно выполнить некие вычисления, их выполняют именно в этом методе, перед командой return. То есть, если у вас есть некий код, определяющий порядок формирования визуального представления компонента, этот код нужно поместить в метод render.

Если некоему компоненту нужно работать с состоянием, то это должен быть компонент, основанный на классе. Если попытаться, в компоненте, основанном на классе, использовать конструкцию наподобие this.props.name = "NoName" — мы столкнёмся с сообщением об ошибке.

```
class App extends React.Component {
    constructor() {
        super()
        this.state = {
            answer: "Yes"
        }
    }

    render() {
        return (
            <div>
                <h1>Is state important to know? {this.state.answer}</h1>
            </div>
        )
    }
}
```

Компонент выводит на экран вопрос Is state important to know? В состоянии хранится ответ на этот вопрос. Для того чтобы добавить этот ответ после вопроса, нужно поступить так же, как мы обычно поступаем, добавляя

JavaScript-конструкции в JSX-код. А именно, надо добавить в конец строки конструкцию {this.state.answer}.

Вопросы для закрепления:

1. Чем класс отличается от функции?

2. Может ли класс не содержать ни одного из методов?

3. Можно ли в компоненте, основанном на классе, использовать конструкцию наподобие this.props.name = "NoName"?

Дополнительные задания:

1. Перепишите все компоненты через функцию.

Полезные ссылки:

https://habr.com/ru/company/ruvds/blog/438986/