Жизненный цикл, управление ресурсами и составные компоненты

Цель работы

Познакомить с этапами жизненного цикла компонента, научить создавать составные компоненты и оптимизировать ресурсы приложения.

Задания для выполнения

1. Реализуйте события жизненного цикла (**constructor(props), componentWillMount(), render(),** а также после рендеринга **shouldComponentUpdate(nextProps, nextState), componentWillUpdate(nextProps, nextState), render(), componentDidUpdate(prevProps, prevState)**) для кнопки с помощью функций в React.
2. Создайте часы, которые выводят текущее время на страницу и для обновления времени используют таймер с помощью функции **componentDidMount()**. А для освобождения ресурсов применяется функция **componentWillUnmount().**
3. Реализуйте поиск по списку с помощью составного компонента.
4. Загрузить созданную страницу на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_7.

Методические указания

1. Создание событий внутри `class ClickButton extends React.Component {`:

```
constructor(props) {
    super(props);
    this.state = {class: "off", label: "Нажми"};

    this.press = this.press.bind(this);

    console.log("constructor");
}
componentWillReceiveProps(nextProps) {
    console.log("componentWillReceiveProps()");
}
```

3. Список группы

```
const propsValues = {
    title: "Список группы",
    items: [
        "Студент 1",
        "Студент 2",
        "Студент 3",
        "Студент 4",
        "Студент 5",
        "Студент 6"
```

```
    ]
};

class Item extends React.Component {
    render() {
        return <li>{this.props.name}</li>;
    }
}

class SearchPlugin extends React.Component{

    constructor(props){
        super(props);
        this.onTextChanged = this.onTextChanged.bind(this);
    }

    onTextChanged(e){
        var text = e.target.value.trim();   // удаляем пробелы
        this.props.filter(text); // передаем введенный текст в
родительский компонент
    }

    render() {
        return <input placeholder="Поиск"
onChange={this.onTextChanged} />;
    }
}

class ItemsList extends React.Component {
    constructor(props){
        super(props);
        this.state = { items: this.props.data.items};

        this.filterList = this.filterList.bind(this);
    }

    filterList(text){
        var filteredList =
this.props.data.items.filter(function(item){
            return item.toLowerCase().search(text.toLowerCase())!==
-1;
        });
        this.setState({items: filteredList});
    }

    render() {
        return(
            <div>
                <h2>{this.props.data.title}</h2>
                <SearchPlugin filter={this.filterList} />
                <ul>
                    {
                        this.state.items.map(function(item){
                            return <Item key={item} name={item} />
                        })
                    }
                </ul>
```

```
                </div>);
        }
    }

    ReactDOM.render(
        <ItemsList data={propsValues} />,
        document.getElementById("app")
    )
```

Контрольные вопросы

1. Что делает функция **componentWillUnmount()**?
2. Вызовется ли **componentDidUpdate(prevProps, prevState)** если shouldComponentUpdate возвращает false?

Дополнительные задания

1. Оформить документы с помощью css.