Урок 5.3. Рендер коллекций. Стейт компонента. Хуки жизненного цикла

Рендер коллекций

В React очень удобно отрисовывать несколько одинаковых элементов с разными данными (например, из массива). Для этого используется обычный метод `map` массива, использованный, как показано в примере, внутри jsx-кода (вёрстки компонента)

```
import React from 'react';

export default class List extends React.Component {
    render() {
        const list = [
            {
                id: 1,
                title: 'Элемент 1'
            },
            {
                id: 2,
                title: 'Элемент 2'
            }
        ]
        return (
            <div>
                {
                    // Используем map и возвращаем
                    // из функции обработчика вёрстку,
                    // использующую данные из конкретного
                    // элемента массива
                    list.map((element) => {
                        return (
                            // Для корневого элемента
                            // при рендере элемента коллекции
                            // обязательно использовать уникальный ключ
                            // в атрибуте key
                            <div key={element.id}>
                                { element.title }
                            </div>
                        )
                    })
```

```
                }
            </div>
        )
    }
}
```

State классового компонента

Состояние компонента необходимо для хранения каких-либо изменяющихся динамически внутри компонента данных. Например, `input` должен хранить своё значение, и оно будет изменяться при вводе текста.

Далее рассмотрим пример управления состоянием компонента внутри классового компонента.

```
import React from 'react';

export default class List extends React.Component {
    constructor(props) {
        super(props);
        // Задаём state в constructor
        // как объект.
        this.state = {
            counter: 0,
            title: 'lorem ipsum'
        }
    }
    onButtonClick() {
       // Допустим, эта функция вызывается
       // при нажатии на какую-то абстрактную кнопку
       // и нам нужно обновить счётчик counter в state
       // Передаём в setState новый объект state (или часть этого объекта только
       // с теми значениями, которые необходимо обновить
        this.setState({ counter: this.state.counter + 1 })
    }    render() {
        return (
            <div></div>
        )
    }
}
```

Хуки жизненного цикла

Хуки жизненного цикла классового компонента – это функции внутри класса компонента, которые вызываются автоматически (благодаря наследованию от `React.Component`) в определённым момент существования компонента. Существующие хуки рассмотрены в примере ниже:

```
import React from 'react';

export default class List extends React.Component {
    componentDidMount() {
        // Вызывается после того, как компонент смонтирован
        // (преобразован в настоящий DOM, который мы видим в браузере)
        // Здесь можно получать данные, выставлять обработчики на элементы (они уже готов
ы)
    }
    componentWillUnmount() {
        // Вызывается перед удалением элемента, например при уходе
        // со страницы, на которой расположен данный элемент
        // Обычно в этой функции удаляются глобальные обработчики
        // типа document.addEventListener('click'), которые были
        // использованы данным компонентом и после его удаления не
         // должны больше существовать
    }
    componentDidUpdate(prevProps, prevState, snapshot) {
        // Компонент обновился. То есть были изменены либо props, либо state компонента.
        // Соответственно, prevProps - это значения props, которое было до обновления комп
онента
        // prevState - предыдущее значение state компонента
    }
    componentDidCatch(error, errorInfo) {
        // Вызывается после возникновения ошибки в одном из дочерних компонентов.
        // Можно использовать, чтобы сообщить об ошибке в лог
    }
    shouldComponentUpdate(nextProps, nextState, nextContext) {
        // Существует для того, чтобы определить, нужно ли перерендеривать компонент
        // при изменении свойств.
        // Можно сравнить текущее состояние и пропсы с теми, что будут следующими (они
        // уже пришли в компонент, но сам стейт и пропсы ещё не обновились)
        // Метод должен вернуть true или false
        // В случае true пропсы и/или стейт будут обновлены
        // В случае false обновление будет отклонено
    }    render() {
        return (
            <div></div>
        )
    }
}
```