Урок 5.5. Хуки useState, useEffect, события.

`useState`

Данный хук используется для того, чтобы использовать состояние в функциональных компонентах.

```
import {useState} from "react";

function Component() {
    // Здесь counter - само состояние. В данном случае счётчик типа number
    // setCounter - функция, с помощью которой можно задать новое значение для counter
    // 0 в useState - начальное значение для counter
    const [counter, setCounter] = useState(0);

    function onSomeEvent() {
        // Здесь мы увеличиваем счётчик на 1,
        // когда происходит какое-то абстрактное событие (про события - дальше)
        setCounter(counter + 1);
    }

    return (
        // Можно использовать counter, где удобно
        // Но изменять только с помощью функции
        <div>{counter}</div>
    )
}
```

`useEffect`

`useEffect` вызывается каждый раз, когда компонент ре-рендерится, изменяется. Ре-рендер происходит при изменении внутреннего состояния компонента (state) и при изменении пропсов.

```
import {useEffect, useState} from "react";

function Component() {
    const [counter, setCounter] = useState(0);

    useEffect(() => {
        // Каждый раз, когда состояние (counter) меняется
```

```
        // заголовок будет обновляться.
        document.title = `Вы нажали ${counter} раз`;
    });

    return (
        <div></div>
    )
}
```

Если из `useEffect` вернуть функцию, то она сработает, когда компонент будет размонтироваться (при удалении). Это полезно, когда нужно снять какие-то глобальные события.

```
import {useEffect} from "react";

function Component() {
    useEffect(() => {
        function clickHandler() {
            console.log('Вы кликнули!');
        }
         // Глобальное событие, которое не удалится само,
         // даже если компонент был удалён со страницы (при
        // переходе на другую страницу, например)
        document.addEventListener('click', clickHandler);

        return function () {
            // Функция выполнится при размонтировании компонента (удалении)
            // Мы снимаем глобальный обработчик клика.
            document.removeEventListener('click', clickHandler);
        }
    });

    return (
        <div></div>
    )
}
```

События

Функциональные компоненты

```
function Component() {
    // Обработчик-функция
    // Может быть задана как с помощью
```

```
     // Function Expression, так и с помощью Function Declaration
    onButtonClick = () => {
        console.log('Вы кликнули');
    }

    function onSecondButtonClick() {
        console.log('Вы кликнули на вторую кнопку');
    }

    return (
        <div>
            {/* События пишутся в camelCase прямо в теге */}
            <button onClick={onButtonClick}>Click me!</button>
            <button onClick={onSecondButtonClick}>Click me 2!</button>
        </div>
    )
}
```