**Лекція 21. Робота з компонентами у React**

1. Компоненти.
2. Пропси.
3. Компоненти в циклі.
4. **Компоненти**

Погляньмо на будь-який сайт. Він складається із набору незалежних блоків: хедер, сайдбари, футер, контент. Можна сказати, що ці блоки є компонентами в тому сенсі, в якому мається на увазі в React.

Якщо подивитися на той же хедер, що в ньому можна виділити блок із логотипом, блок контактів, блок з меню тощо. Тобто, компоненти можуть складатися з інших підкомпонентів і так далі.

Аналогічно справа в React - сайт будується з набору компонентів, які в свою чергу можуть містити інші компонети і так далі.

Кожен компонент є класом. Зазвичай розробка починається з головного компонента **App**, який містить у собі інші.

Кожен компонент може мати свої стейти і кожен компонент має метод **render** , який повертає результат роботи цього компонента.

Давайте для прикладу зробимо компонент **User**, в стейті якого записані ім'я та прізвище користувача, а результатом рендерингу є абзац із цими даними:

class User extends React.Component {

constructor() {

super();

this.state **=** {name: 'Микола', surname: 'Шевченко'};

}

render() {

return **<**p**>** ім’я: {this.state.name}, прізвище: {this.state.surname} **</**p**>**;

}

}

Тепер підключимо наш компонент **User** всередині компонента **App** . Це робиться так: у те місце, де має з'явитися результат роботи нашого компонента, слід вставити тег **<User />** , де **User** - ім'я нашого класу:

class App extends React.Component {

render() {

return **<**div**>**

**<**User **/>**

**</**div**>**;

}

}

Зберемо все разом і запустимо:

class User extends React.Component {

constructor() {

super();

this.state **=** {name: 'Микола', surname: 'Шевченко'};

}

render() {

return **<**p**>** ім’я: {this.state.name}, прізвище: {this.state.surname} **</**p**>**;

}

}

class App extends React.Component {

render() {

return **<**div**>**

**<**User **/>**

**</**div**>**;

}

}

Результатом роботи цього коду буде наступне:

<div>

<p>

ім’я: Микола, прізвище: Шевченко

</p>

</div>

1. **Пропси**

Давайте тепер зробимо так, щоб ім'я і прізвище нашого користувача зберігалися не у самому компоненті, а отримувались ним ззовні.

Для цього при підключенні компонента **<User />** необхідно додати атрибути **name** і **surname** , так:

class App extends React.Component {

render() {

return **<**div**>**

**<**User name**=**"Микола" surname**=**"Шевченко" **/>**

**</**div**>**;

}

}

Значення доданих атрибутів стають доступними всередині нашого класу так: за допомогою **this.props.name** ми можемо отримати ім'я, а за допомогою **this.props.surname** - прізвище:

class User extends React.Component {

constructor() {

super();

}

render() {

return **<**p**>**

ім’я: {this.props.name}, прізвище: {this.props.surname}

**</**p**>**;

}

}

Таким чином, за допомогою атрибутів можна передавати дані з батьківського компонента у дочірній.

Конструкцію **this.props** в побуті називають **пропсом** (українською мовою перекладається як *властивість* ).

Виходить, що будь-який компонент має стейти і пропси. **Стейти** – це дані самого компонента, а **пропси** – дані, передані з батьківського компонента.

Давайте зберемо все разом і запустимо:

class App extends React.Component {

render() {

return **<**div**>**

**<**User name**=**"Микола" surname**=**"Шевченко" **/>**

**</**div**>**;

}

}

class User extends React.Component {

constructor(props) {

super(props);

}

render() {

return **<**p**>**

ім’я: {this.props.name}, прізвище: {this.props.surname}

**</**p**>**;

}

}

Результатом роботи цього коду буде наступне:

<div>

<p> ім’я: Микола, прізвище: Шевченко </p>

</div>

**Переваги компонентів**

Переваги компонентів виявляються, коли ми хочемо вивести на екран не одного, а кількох користувачів.

У цьому випадку один компонент для відображення багатьох користувачів зручніший: якщо нам потрібно буде підправити те, як ми хочемо відображати нашого користувача, - це зручніше буде зробити в окремому компоненті, *не чіпаючи* те, звідки беруться дані для нашого користувача.

Тобто вийде поділ: відображення даних окремо від цих даних. Кожен компонент просто отримує якісь дані зовні і відображає їх у потрібному нам вигляді. А ось зовні компонента дані є, але ми їх ніяк не відображаємо - за це відповідає компонент.

Давайте реалізуємо це: за допомогою нашого компонента **User** виведемо на екран кілька користувачів:

class User extends React.Component {

constructor() {

super();

}

render() {

return **<**div**>**

ім’я: {this.props.name}, прізвище: {this.props.surname}

**</**div**>**;

}

}

class App extends React.Component {

constructor() {

super();

}

//Виведемо кількох користувачів:

render() {

return **<**div**>**

**<**User name**=**"Микола" surname**=**"Шевченко" **/>**

**<**User name**=**"Василь" surname**=**"Королюк" **/>**

**<**User name**=**"Петро" surname**=**"Кармалюк" **/>**

**</**div**>**;

}

}

Результатом роботи цього коду буде наступне:

<div>

<p> ім’я: Микола, прізвище: Шевченко </p>

<p> ім’я: Василь, прізвище: Королюк </p>

<p> ім’я: Петро, прізвище: Кармалюк </p>

</div>

Нехай тепер ми вирішили редагувати спосіб відображення користувача і зробити так, щоб ім'я та прізвище були в окремому абзаці. У цьому випадку редагування доведеться робити тільки в одному місці - в коді компонента **User** .

Давайте зробимо це:

class User extends React.Component {

constructor() {

super();

}

//змінимо відображення окремого користувача:

render() {

return **<**div**>**

**<**p**>**ім’я: {this.props.name}**</**p**>**

**<**p**>**прізвище: {this.props.surname}**</**p**>**

**</**div**>**;

}

}

class App extends React.Component {

constructor() {

super();

}

render() {

return **<**div**>**

**<**User name**=**"Микола" surname**=**"Шевченко" **/>**

**<**User name**=**"Василь" surname**=**"Королюк" **/>**

**<**User name**=**"Петро" surname**=**"Кармалюк" **/>**

**</**div**>**;

}

}

Запустіть цей код і ви побачите, що відображення користувачів змінилося - тепер і ім'я, і ​​прізвище розташовані в окремих абзацах.

**Додамо стейти**

Нехай тепер дані користувачів зберігаються у стейті компонента **App**. Поправимо наш код відповідно до цього:

class User extends React.Component {

constructor() {

super();

}

render() {

return **<**div**>**

**<**p**>**ім’я: {this.props.name}**</**p**>**

**<**p**>**прізвище: {this.props.surname}**</**p**>**

**</**div**>**;

}

}

class App extends React.Component {

constructor() {

super();

//збережемо дані в стейті:

this.state **=** {

user1: {name: 'Микола', surname: 'Шевченко'},

user2: {name: 'Василь', surname: 'Королюк'},

user3: {name: 'Петро', surname: 'Кармалюк'},

};

}

render() {

return **<**div**>**

**<**User name**=**{this.state.user1.name} surname**=**{this.state.user1.surname} **/>**

**<**User name**=**{this.state.user2.name} surname**=**{this.state.user3.surname} **/>**

**<**User name**=**{this.state.user3.name} surname**=**{this.state.user3.surname} **/>**

**</**div**>**;

}

}

Якщо ви запустите цей код, то візуально нічого не зміниться, але дані будуть братися вже зі стейту компонента **App** .

1. **Компоненти в циклі**

У попередньому прикладі ми використали кілька екземплярів одного компонента, ось так:

class App extends React.Component {

...

render() {

return **<**div**>**

**<**User name**=**{this.state.user1.name} surname**=**{this.state.user1.surname} **/>**

**<**User name**=**{this.state.user2.name} surname**=**{this.state.user3.surname} **/>**

**<**User name**=**{this.state.user3.name} surname**=**{this.state.user3.surname} **/>**

**</**div**>**;

}

}

Давайте тепер зробимо те саме, але так, щоб за нас це зробив цикл.

Трохи змінимо стейт із масивом користувачів, ось так:

class App extends React.Component {

constructor() {

super();

this.state **=** {

users: [

{name: 'Микола', surname: 'Шевченко'},

{name: 'Василь', surname: 'Королюк'},

{name: 'Петро', surname: 'Кармалюк'},

]

};

}

}

Давайте тепер зробимо цикл, який перебере наш масив з користувачами з **this.state.users** і зробить кілька екземплярів компонента **User** :

class App extends React.Component {

...

render() {

const users **=** this.state.users.map((item, index) **=>** {

return **<**User key**=**{index} name**=**{item.name} surname**=**{item.surname}**/>**;

});

return **<**div**>**

{users}

**</**div**>**;

}

}

Зверніть увагу на атрибут **key** - тут він також є обов'язковим для правильної внутрішньої роботи React.

Давайте зберемо все разом і запустимо - отримаємо виведення компонентів у циклі:

class User extends React.Component {

constructor() {

super();

}

render() {

return **<**div**>**

**<**p**>**ім’я: {this.props.name}**</**p**>**

**<**p**>**прізвище: {this.props.surname}**</**p**>**

**</**div**>**;

}

}

class App extends React.Component {

constructor() {

super();

this.state **=** {

users: [

{name: 'Микола', surname: 'Шевченко'},

{name: 'Василь', surname: 'Королюк'},

{name: 'Петро', surname: 'Кармалюк'},

]

};

}

render() {

const users **=** this.state.users.map((item, index) **=>** {

return **<**User key**=**{index} name**=**{item.name} surname**=**{item.surname} **/>**;

});

return **<**div**>**

{users}

**</**div**>**;

}

}

Якщо ви запустите цей код, то візуально нічого не зміниться, але компоненти формуватимуться за допомогою циклу.