**Лекція 22. Просунута робота з компонентами в React**

1. Передача методу
2. Передача параметра з компонента
3. Робота з декількома компонентами
4. Видалення елементів

**1.Передача методу**

Нехай у нас сьогодні є компонент **User** , в який передається ім'я та прізвище користувача. Використовуємо наш компонент у класі **App** :

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 **=** {

name: 'Микола',

surname: 'Шевченко',

};

}

render() {

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

**<**User

name**=**{this.state.name}

surname**=**{this.state.surname}

**/>**

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

}

}

Нехай тепер ми маємо метод **showMessage** , який виводить алерт з '!!!'. Давайте передамо цей метод параметром у наш компонент за допомогою атрибута:

class App extends React.Component {

...

//Метод, який ми хочемо передати в компонент:

showMessage() {

alert('!!!');

}

render() {

//Метод this.showMessage, який передається в атрибут showMessage:

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

**<**User

name**=**{this.state.name}

surname**=**{this.state.surname}

showMessage**=**{this.showMessage}

**/>**

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

}

}

Вийде, що метод **showMessage** класу **App** буде доступний усередині компонента **User** як пропс цього компонента, тобто як **this.props.showMessage** .

Давайте зробимо кнопку всередині компонента **User** , на кліку на яку буде спрацьовувати метод **this.props.showMessage** :

class User extends React.Component {

constructor() {

super();

}

render() {

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

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

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

**<**button onClick**=**{this.props.showMessage}**>**

натисни на мене

**</**button**>**

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

}

}

Давайте зберемо все разом і запустимо:

class User extends React.Component {

constructor() {

super();

}

render() {

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

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

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

**<**button onClick**=**{this.props.showMessage}**>**

натисни на мене

**</**button**>**

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

}

}

class App extends React.Component {

constructor() {

super();

this.state **=** {

name: 'Микола',

surname: 'Шевченко',

};

}

showMessage() {

alert('!!!');

}

render() {

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

**<**User

name**=**{this.state.name}

surname**=**{this.state.surname}

showMessage**=**{this.showMessage}

**/>**

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

}

}

Запустіть цей код і натисніть кнопку - ви побачите алерт із текстом '!!!'.

А ось якщо ми спробуємо вивести всередині **showMessage** щось, пов'язане з цим, то отримаємо помилку. Наприклад, спроба звернеться до **this.state.name** видасть помилку:

showMessage() {

alert(this.state.name);

}

Зрозуміло, чому виникає помилка - треба було байндити **this** . Зробимо це:

render() {

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

**<**User

name**=**{this.state.name}

surname**=**{this.state.surname}

showMessage**=**{this.showMessage.bind(this)}

**/>**

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

}

Зберемо все разом і запустимо - тепер все буде ок:

class User extends React.Component {

constructor() {

super();

}

render() {

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

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

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

**<**button onClick**=**{this.props.showMessage}**>**

натисни на мене

**</**button**>**

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

}

}

class App extends React.Component {

constructor() {

super();

this.state **=** {

name: 'Микола',

surname: 'Шевченко',

};

}

showMessage() {

alert(this.state.name);

}

render() {

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

**<**User

name**=**{this.state.name}

surname**=**{this.state.surname}

showMessage**=**{this.showMessage.bind(this)}

**/>**

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

}

}

Запустіть цей код і натисніть кнопку - ви побачите алерт з текстом 'Микола'.

**2.Передача параметру з компонента**

Давайте тепер зробимо так, щоб у метод **showMessage** можна було передати параметр, так:

showMessage(num) {

alert(this.state.name **+** ' ' **+** num);

}

Цей параметр ми передаватимемо зсередини компонента **User** . Навіщо нам це знадобиться: ми зможемо зробити кілька екземплярів компонента **User** і кожен з них передаватиме свій параметр у **showMessage** . Тобто вийде, що компонент передаватиме дані нагору (нагадую, що для передачі даних вниз, тобто від батька до нащадка, у нас є пропси).

Спробуймо вирішити описане завдання. Розглянемо клас компонента **User** :

class User extends React.Component {

constructor() {

super();

}

render() {

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

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

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

**<**button onClick**=**{this.props.showMessage}**>**

натисни на мене

**</**button**>**

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

}

}

Давайте спробуємо просто передати параметр методу **this.props.showMessage** :

**<**button onClick**=**{this.props.showMessage(3)}**>**

натисни на мене

**</**button**>**

Однак це не буде працювати - ми не можемо просто передати параметр, адже тоді це буде не код методу, який ми хочемо прив'язати, а результат його роботи.

Для передачі параметра ми можемо використовувати хитрий спосіб **bind** , ось за таким принципом: **showMessage.bind (контекст, параметр1, параметр2 ...)** .

Але що писати замість контексту? Вже точно не це - адже він якщо ми його напишемо, то всередині **showMessage це буде** посилатися на об'єкт класу **User** , а там якраз треба, щоб він посилався на об'єкт класу **App** . Більше того, ми вже зробили це **до** нашого **showMessage** ось тут:

render() {

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

**<**User

name**=**{this.state.name}

surname**=**{this.state.surname}

showMessage**=**{this.showMessage.bind(this)}

**/>**

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

}

Для вирішення проблеми як контекст треба передати **null** . У цьому випадку прив'язаний контекст раніше залишиться, але параметри передадуть. Ось так по-хитрому. Зробимо це:

**<**button onClick**=**{this.props.showMessage(null, 3)}**>**

натисни на мене

**</**button**>**

Зберемо все разом і запустимо - тепер усе працюватиме:

class User extends React.Component {

constructor() {

super();

}

render() {

//Передаємо по кліку на кнопку 3:

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

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

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

**<**button onClick**=**{this.props.showMessage.bind(null, 3)}**>**

натисни на мене

**</**button**>**

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

}

}

class App extends React.Component {

constructor() {

super();

this.state **=** {

name: 'Микола',

surname: 'Шевченко',

};

}

//Метод принимает параметром число:

showMessage(num) {

alert(this.state.name **+** ' ' **+** num);

}

render() {

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

**<**User

name**=**{this.state.name}

surname**=**{this.state.surname}

showMessage**=**{this.showMessage.bind(this)}

**/>**

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

}

}

Запустіть цей код і натисніть на кнопку - ви побачите алерт з повідомленням 'Микола 3' - тобто параметр передасть з нашого компонента **User** вгору в **App** .

**3.Робота з декількома компонентами**

Давайте тепер зробимо кілька екземплярів компонента **User** – і нехай кожен із них передає свій номер нагору до батька.

Нехай зараз у стейті зберігається кілька користувачів:

class App extends React.Component {

constructor() {

super();

this.state **=** {

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

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

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

};

}

}

Поправимо метод **showMessage** так, щоб він виводив лише номер:

showMessage(num) {

alert(num);

}

Давайте тепер зробимо кілька екземплярів компонента **User** . У кожен із них передамо своє ім'я, прізвище, номер (проставимо його вручну - першому екземпляру 1, другому 2 і так далі) і метод **showMessage** :

class App extends React.Component {

...

render() {

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

**<**User

name**=**{this.state.user1.name}

surname**=**{this.state.user1.surname}

num**=**"1"

showMessage**=**{this.showMessage.bind(this)}

**/>**

**<**User

name**=**{this.state.user2.name}

surname**=**{this.state.user2.surname}

num**=**"2"

showMessage**=**{this.showMessage.bind(this)}

**/>**

**<**User

name**=**{this.state.user3.name}

surname**=**{this.state.user3.surname}

num**=**"3"

showMessage**=**{this.showMessage.bind(this)}

**/>**

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

}

}

Зберемо все разом і запустимо - вийде, що після натискання на кнопку кожен компонент буде виводити свій номер на екран:

class User extends React.Component {

constructor() {

super();

}

render() {

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

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

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

**<**button onClick**=**{this.props.showMessage.bind(null, this.props.num)}**>**

натисни на мене

**</**button**>**

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

}

}

class App extends React.Component {

constructor() {

super();

this.state **=** {

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

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

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

};

}

//Виводимо алертом число, що передається:

showMessage(num) {

alert(num);

}

render() {

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

**<**User

name**=**{this.state.user1.name}

surname**=**{this.state.user1.surname}

num**=**"1"

showMessage**=**{this.showMessage.bind(this)}

**/>**

**<**User

name**=**{this.state.user2.name}

surname**=**{this.state.user2.surname}

num**=**"2"

showMessage**=**{this.showMessage.bind(this)}

**/>**

**<**User

name**=**{this.state.user3.name}

surname**=**{this.state.user3.surname}

num**=**"3"

showMessage**=**{this.showMessage.bind(this)}

**/>**

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

}

}

Запустіть цей код і натисніть на кнопки - ви бачитимете алерти з різними числами, що відповідають компонентам.

**Цикл**

Давайте переробимо попередній приклад так, щоб компоненти виводилися в циклі:

class User extends React.Component {

constructor() {

super();

}

render() {

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

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

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

**<**button onClick**=**{this.props.showMessage.bind(null, this.props.num)}**>**

натисни на мене

**</**button**>**

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

}

}

class App extends React.Component {

constructor() {

super();

//Переробляємо стейт з юзерами на масив:

this.state **=** {

users: [

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

{name: 'Василь', surname: 'Ковальчук'},

{name: 'Петро', surname: 'Петренко'},

]

};

}

// Виводимо алертом число, що передається:

showMessage(num) {

alert(num);

}

render() {

//Формуємо компоненти в циклі:

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

return **<**User

key**=**{index}

name**=**{item.name}

surname**=**{item.surname}

num**=**{index **+** 1}

showMessage**=**{this.showMessage.bind(this)}

**/>**;

});

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

{users}

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

}

}

Запустіть цей код – візуально нічого не зміниться, але тепер компоненти будуть створюватися у циклі.

**4.Видалення елементів**

Давайте тепер зробимо так, щоб усередині кожного компонента була кнопочка, натискання на яку цей компонент буде видалятися.

Для цього всередині класу **App** реалізуємо метод **deleteUser** , який параметром прийматиме номер компонента та видалятиме дані цього компонента зі стейту. Оскільки компоненти створюються у циклі, то зміна стейту призведе до того, що компонент вилучиться з циклу.

Отже, ось реалізація **deleteUser** :

class App extends React.Component {

...

//Видаляємо заданий елемент:

deleteUser(index) {

this.state.users.splice(index, 1);

this.setState({users: this.state.users});

}

render() {

...

}

}

Оскільки видалення компонента проводиться у разі його номеру в циклі, цей номер (назвемо його **index** ) необхідно передати компонент. Зробимо це:

class App extends React.Component {

...

render() {

//Передаємо index и this.deleteUser() в компонент:

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

return **<**User

key**=**{index}

name**=**{item.name}

surname**=**{item.surname}

index**=**{index}

deleteUser**=**{this.deleteUser.bind(this)}

**/>**;

});

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

{users}

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

}

}

Метод **this.deleteUser** буде доступний у компоненті як **this.props.deleteUser** . Давайте прив'яжемо його до кнопки, передавши параметр компонент компонент **index** , який всередині нього буде доступний як **this.props.index** :

class User extends React.Component {

render() {

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

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

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

**<**button onClick**=**{this.props.deleteUser.bind(null, this.props.index)}**>**

видалити

**</**button**>**

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

}

}

Зберемо всі разом і запустимо - тепер за натисканням на кнопку компонент видалятиме сам себе:

class User extends React.Component {

constructor() {

super();

}

render() {

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

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

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

**<**button onClick**=**{this.props.deleteUser.bind(null, this.props.index)}**>**

видалити

**</**button**>**

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

}

}

class App extends React.Component {

constructor() {

super();

this.state **=** {

users: [

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

{name: 'Василь', surname: 'Ковальчук'},

{name: 'Петро', surname: 'Петренко'},

]

};

}

//Видаляємо заданий елемент:

deleteUser(index) {

this.state.users.splice(index, 1);

this.setState({users: this.state.users});

}

render() {

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

return **<**User

key**=**{index}

index**=**{index}

name**=**{item.name}

surname**=**{item.surname}

deleteUser**=**{this.deleteUser.bind(this)}

**/>**;

});

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

{users}

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

}

}

Запустіть цей код і натискайте на кнопочки - по кожному натисканню буде видалятися відповідний компонент.