Сейчас мы с вами разберем более продвинутую работу с компонентами. Мы научимся передавать методы в компоненты, а также передавать данные из компонента наверх - к родителю.

Передача метода

Пусть у нас сейчас есть компонент **User**, в который передается имя и фамилия юзера. Используем наш компонент в классе **App**:

```
class User extends React.Component {
	constructor() {
		super();
	}

	render() {
		return <div>
			<p>имя: {this.props.name}</p>
			<p>фамилия: {this.props.surname}</p>
		</div>;
	}
}

class App extends React.Component {
	constructor() {
		super();

		this.state = {
			name: 'Коля',
			surname: 'Иванов',
		};
	}

	render() {
		return <div>
			<User
				name={this.state.name}
				surname={this.state.surname}
			/>
		</div>;
	}
}
```

Пусть теперь у нас есть метод **showMessage**, который выводит алерт с '!!!'. Давайте передадим этот метод параметром в наш компонент с помощью атрибута:

```
class App extends React.Component {
	...

	//Метод, который хотим передать в компонент:
```

```
	showMessage() {
		alert('!!!!');
	}

	render() {

		//Передаем метод this.showMessage в атрибут showMessage:
		return <div>
			<User
				name={this.state.name}
				surname={this.state.surname}
				showMessage={this.showMessage}
			/>
		</div>;
	}
}
```

Получится, что метод **showMessage** класса **App** будет доступен внутри компонента **User** как пропс этого компонента, то есть как **this.props.showMessage**.

Давайте сделаем кнопку внутри компонента **User**, по клику на которую будет срабатывать метод **this.props.showMessage**:

```
class User extends React.Component {
	constructor() {
		super();
	}

	render() {
		return <div>
			<p>имя: {this.props.name}</p>
			<p>фамилия: {this.props.surname}</p>

			<button onClick={this.props.showMessage}>
				нажми на меня
			</button>
		</div>;
	}
}
```

Давайте соберем все вместе и запустим:

```
class User extends React.Component {
	constructor() {
		super();
	}

	render() {
		return <div>
			<p>имя: {this.props.name}</p>
```

```
                <p>фамилия: {this.props.surname}</p>

                <button onClick={this.props.showMessage}>
                        нажми на меня
                </button>
            </div>;
        }
}

class App extends React.Component {
        constructor() {
            super();

            this.state = {
                name: 'Коля',
                surname: 'Иванов',
            };
        }

        showMessage() {
            alert('!!!!');
        }

        render() {
            return <div>
                <User
                    name={this.state.name}
                    surname={this.state.surname}
                    showMessage={this.showMessage}
                />
            </div>;
        }
}
```

Запустите этот код и нажмите на кнопку - вы увидите алерт с текстом '!!!'.

А вот если мы попробуем вывести внутри **showMessage** что-то, связанное с this, то получим ошибку. Например, попытка обратится к **this.state.name** выдаст ошибку:

```
showMessage() {
        alert(this.state.name);
}
```

Понятно, почему возникает ошибка - надо было байндить **this**. Сделаем это:

```
render() {
        return <div>
            <User
                name={this.state.name}
                surname={this.state.surname}
```

```
                showMessage={this.showMessage.bind(this)}
            />
        </div>;
}
```

Соберем все вместе и запустим - теперь все будет ок:

```
class User extends React.Component {
        constructor() {
                super();
        }

        render() {
                return <div>
                        <p>имя: {this.props.name}</p>
                        <p>фамилия: {this.props.surname}</p>

                        <button onClick={this.props.showMessage}>
                                нажми на меня
                        </button>
                </div>;
        }
}

class App extends React.Component {
        constructor() {
                super();

                this.state = {
                        name: 'Коля',
                        surname: 'Иванов',
                };
        }

        showMessage() {
                alert(this.state.name);
        }

        render() {
                return <div>
                        <User
                                name={this.state.name}
                                surname={this.state.surname}
                                showMessage={this.showMessage.bind(this)}
                        />
                </div>;
        }
}
```

Запустите этот код и нажмите на кнопку - вы увидите алерт с текстом 'Коля'.

Передача параметра из компонента наверх

Давайте теперь сделаем так, чтобы в метод **showMessage** можно было передать параметр, вот так:

```
showMessage(num) {
	alert(this.state.name + ' ' + num);
}
```

Этот параметр мы будем передавать изнутри компонента **User**. Зачем нам это может понадобится: мы сможем сделать несколько экземпляров компонента **User** и каждый из них будет передавать свой параметр в **showMessage**. То есть получится, что компонент будет передавать данные наверх (напоминаю, что для передачи данных вниз, то есть от родителя к потомку, у нас есть пропсы).

Давайте попробуем решить описанную задачу. Рассмотрим класс компонента **User**:

```
class User extends React.Component {
	constructor() {
		super();
	}

	render() {
		return <div>
			<p>имя: {this.props.name}</p>
			<p>фамилия: {this.props.surname}</p>

			<button onClick={this.props.showMessage}>
				нажми на меня
			</button>
		</div>;
	}
}
```

Давайте попробуем просто передать параметр в метод **this.props.showMessage**:

```
<button onClick={this.props.showMessage(3)}>
	нажми на меня
</button>
```

Однако, это не будет работать - мы не можем просто передать параметр, ведь тогда это будет не код метода, который мы хотим привязать, а результат его работы.

Для передачи параметра мы можем использовать хитрый способ с **bind**, вот по такому принципу: **showMessage.bind(контекст, параметр1, параметр2...)**.

Но что писать вместо контекста? Уж точно не **this** - ведь он если мы его напишем, то внутри **showMessage** это **this** будет ссылаться на объект класса **User**, а там как раз-таки надо, чтобы он ссылался на объект класса **App**. Более того, мы уже прибайндили **this** к нашему **showMessage** вот тут:

```
render() {
	return <div>
		<User
			name={this.state.name}
			surname={this.state.surname}
			showMessage={this.showMessage.bind(this)}
		/>
	</div>;
}
```

Для решения проблемы в качестве контекста надо передать **null**. В этом случае привязанный ранее контекст останется, но параметры передадутся. Вот так по-хитрому. Сделаем это:

```
<button onClick={this.props.showMessage(null, 3)}>
	нажми на меня
</button>
```

Соберем все вместе и запустим - теперь все будет работать:

```
class User extends React.Component {
	constructor() {
		super();
	}

	render() {

		//Передаем по клику на кнопку число 3:
		return <div>
			<p>имя: {this.props.name}</p>
			<p>фамилия: {this.props.surname}</p>

			<button onClick={this.props.showMessage.bind(null, 3)}>
				нажми на меня
```

```
                </button>
            </div>;
        }
}

class App extends React.Component {
        constructor() {
            super();

            this.state = {
                name: 'Коля',
                surname: 'Иванов',
            };
        }

        //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>;
        }
}
```

Запустите этот код и нажмите на кнопку - вы увидите алерт с сообщением 'Коля 3' - то есть параметр передастся из нашего компонента **User** наверх в **App**.

Несколько компонентов

Давайте теперь сделаем несколько экземпляров компонента **User** - и пусть каждый из них передает свой номер наверх к родителю.

Пусть сейчас в стейте хранится несколько юзеров:

```
class App extends React.Component {
        constructor() {
            super();

            this.state = {
                user1: {name: 'Коля', surname: 'Иванов'},
                user2: {name: 'Вася', surname: 'Петров'},
                user3: {name: 'Петя', surname: 'Сидоров'},
            };
```

```
	}
}
```

Поправим метод **showMessage** так, чтобы он выводил только номер:

```
showMessage(num) {
	alert(num);
}
```

Давайте теперь сделаем несколько экземпляров компонента **User**. В каждый из них передадим свое имя, фамилию, номер (проставим его вручную - первому экземпляру 1, второму 2 и так далее) и 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>;
	}
}
```

Соберем все вместе и запустим - получится, что по нажатию на кнопку, каждый компонент будет выводить свой номер на экран:

```
class User extends React.Component {
	constructor() {
		super();
	}
```

```
	render() {
		return <div>
			<p>имя: {this.props.name}</p>
			<p>фамилия: {this.props.surname}</p>

			<button onClick={this.props.showMessage.bind(null,
this.props.num)}>
				нажми на меня
			</button>
		</div>;
	}
}

class App extends React.Component {
	constructor() {
		super();

		this.state = {
			user1: {name: 'Коля', surname: 'Иванов'},
			user2: {name: 'Вася', surname: 'Петров'},
			user3: {name: 'Петя', surname: 'Сидоров'},
		};
	}

	//Выводим алертом переданное число:
	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>;
```

```
	}
}
```

Запустите этот код и понажимайте на кнопки - вы будете видеть алерты с разными числами, соответствующими компонентам.

Цикл

Давайте переделаем предыдущий пример так, чтобы компоненты выводились в цикле:

```
class User extends React.Component {
	constructor() {
		super();
	}

	render() {
		return <div>
			<p>имя: {this.props.name}</p>
			<p>фамилия: {this.props.surname}</p>

			<button onClick={this.props.showMessage.bind(null,
this.props.num)}>
				нажми на меня
			</button>
		</div>;
	}
}

class App extends React.Component {
	constructor() {
		super();

		//Переделаем стейт с юзерами на массив:
		this.state = {
			users: [
				{name: 'Коля', surname: 'Иванов'},
				{name: 'Вася', surname: 'Петров'},
				{name: 'Петя', surname: 'Сидоров'},
			]
		};
	}

	//Выводим алертом переданное число:
	showMessage(num) {
		alert(num);
	}

	render() {
```

```
            //Сформируем компоненты в цикле:
            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>;
      }
}
```

Запустите этот код - визуально ничего не поменяется, но теперь компоненты будут создаваться в цикле.

Удаление элементов

Давайте теперь сделаем так, чтобы внутри каждого компонента была кнопочка, по нажатию на которую этот компонент будет удаляться.

Для этого внутри класса **App** реализуем метод **deleteUser**, который параметром будет принимать номер компонента и удалять данные этого компонента из стейта. Так как компоненты создаются в цикле, то изменение стейта приведет к тому, что компонент удалится из цикла.

Итак, вот реализация **deleteUser**:

```
class App extends React.Component {

      ...

      //Удаляем заданный элемент:
      deleteUser(index) {
            this.state.users.splice(index, 1);
            this.setState({users: this.state.users});
      }

      render() {
            ...
      }
}
```

Так как удаление компонента производится по его номеру в цикле, то этот номер (назовем его **index**) необходимо передать в компонент. Сделаем это:

```
class App extends React.Component {

	...

	render() {

		//Передаем 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** будет доступен в компоненте как **this.props.deleteUser**. Давайте привяжем его к кнопке, передав параметром номер компонента **index**, который внутри него будет доступен как **this.props.index**:

```
class User extends React.Component {
	render() {
		return <div>
			<p>имя: {this.props.name}</p>
			<p>фамилия: {this.props.surname}</p>

			<button onClick={this.props.deleteUser.bind(null,
this.props.index)}>
				удалить
			</button>
		</div>;
	}
}
```

Соберем все вместе и запустим - теперь по нажатию на кнопку компонент будет удалять сам себя:

```
class User extends React.Component {
```

```
	constructor() {
		super();
	}

	render() {
		return <div>
			<p>имя: {this.props.name}</p>
			<p>фамилия: {this.props.surname}</p>

			<button onClick={this.props.deleteUser.bind(null,
this.props.index)}>
				удалить
			</button>
		</div>;
	}
}

class App extends React.Component {
	constructor() {
		super();

		this.state = {
			users: [
				{name: 'Коля', surname: 'Иванов'},
				{name: 'Вася', surname: 'Петров'},
				{name: 'Петя', surname: 'Сидоров'},
			]
		};
	}

	//Удаляем заданный элемент:
	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>;
	}
}
```

Запустите этот код и по нажимайте на кнопочки - по каждому нажатию будет удаляться соответствующий компонент.