Сейчас мы с вами освоим работу с событиями в фреймворке React, а также подробнее поговорим о свойствах и методах объектов-компонентов.

State

Основным понятием React является **state** (состояние, стейт).

State - это свойство класса-компонента, которое устанавливается в конструкторе:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {объект со значениями}; //задаем состояние
	}
}
```

В state хранится *текущее состояние компонента* в виде объекта {} с ключами и значениями. Что входит в понятие текущее состояние?

Во-первых, это какие-либо данные. Например, компонент показывает на экране список имен юзеров. Эти имена изначально должны где-то хранится, обычно в каком-то массиве. Вот этот массив и должен хранится в **state**.

Во-вторых, это действительно текущее состояние компонента. К примеру: некоторый блок может находиться в развернутом или свернутом положении. Вот **state** и хранит состояние этого блока: к примеру, **true** - блок развернут, а **false** - свернут.

Думаю, вам не до конца понятно, что имеется ввиду, но не переживайте - сейчас мы начнем разбираться на практических примерах - и вы все поймете.

Работа со state

Пусть в **this.state** хранится объект с юзером - его имя и фамилия:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван', surname: 'Иванов'};
	}
}
```

Давайте выведем данные из стейта при рендеринге, то есть в методе **render()**:

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

		//Запишем данные в стейт:
		this.state = {name: 'Иван', surname: 'Иванов'};
	}

	//Выведем данные из стейта:
	render() {
		return <div>
			имя: {this.state.name}, фамилия: {this.state.surname}
		</div>;
	}
}
```

Результатом работы этого кода будет следующее:

```
<div>
	имя: Иван, фамилия: Иванов
</div>
```

Таким образом, в конструкторе нашего класса мы можем записывать какие-то данные в стейт, а затем выводить их в других методах (чаще всего в методе render, но не обязательно), обращаясь к ним через **this.state.имяСостояния**.

К примеру, имя выведем так: **this.state.name**, а фамилию - вот так: **this.state.surname**.

Работа с событиями

Давайте теперь изучим работу с событиями на React. К примеру, сделаем так, чтобы *по клику на блок выводился алерт с некоторым текстом*.

Пусть у нас есть метод **showMessage**, который выводит алерт с сообщением, а в методе **render** у нас есть **div**, по клику на который нам и хотелось бы видеть этот алерт:

```
class App extends React.Component {

	//Метод, который должен сработать по клику:
	showMessage() {
		alert('Привет!');
	}

	render() {
		//Див, на который будем кликать:
```

```
            return <div>
                  нажми на меня
            </div>;
      }
}
```

Давайте теперь привяжем к нашему диву событие **onclick** так, чтобы по клику на этот див срабатывал метод **showMessage**.

Для этого мы напишем следующее: ** ** - то есть добавим атрибут **onClick** (именно в camelCase, то есть onClick, а не onclick), а в нем укажем метод, который выполнится по этому событию (без кавычек и круглых скобок, через **this**).

Теперь по клику на див будет выводиться сообщение:

```
class App extends React.Component {
      showMessage() {
            alert('Привет!');
      }

      render() {
            return <div onClick={this.showMessage}>
                  нажми на меня
            </div>;
      }
}
```

Запустите этот код и нажмите на div - вы увидите алерт с сообщением.

Таким образом и происходит работа с событиями: *добавляется атрибут (к примеру, onClick или onFocus), значением атрибута указывается метод, который будет вызван по этому событию*.

Работа с this

Пусть теперь в конструкторе у нас задан **this.state**, в котором хранится имя и фамилия пользователя:

```
class App extends React.Component {
      constructor() {
            super();
            this.state = {name: 'Иван', surname: 'Иванов'};
      }
}
```

Давайте сделаем так, чтобы метод **showMessage** выводил имя нашего пользователя, хранящегося в стейте (то есть 'Иван'). Кажется, что в методе **showMessage** следует поменять **alert('Привет!')** на **alert(this.state.name)**, но это не будет работать:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван', surname: 'Иванов'};
	}

	//Это не будет работать:
	showMessage() {
		alert(this.state.name);
	}

	render() {
		return <div onClick={this.showMessage}>
			нажми на меня
		</div>;
	}
}
```

Если вы запустите этот код и нажмете на div - в консоли вы увидите ошибку реакта. До нажатия на див никакой ошибки не будет. Почему так - читайте ниже.

Почему это не работает: имеет место потеря контекста - **this** внутри метода, вызванного снаружи (то есть тут: **onClick={this.showMessage}**), не будет указывать на наш объект (проблема именно с внешнем вызовом, вызов метода внутри класса работает нормально).

Чтобы поправить проблему, нужно привязать контекст к нашему методу, например, с помощью bind, вот так: **onClick={this.showMessage.bind(this)}**.

Теперь все будет работать, как надо:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван', surname: 'Иванов'};
	}

	//Все работает, как надо:
	showMessage() {
		alert(this.state.name);
	}
```

```
	render() {
		return <div onClick={this.showMessage.bind(this)}>
			нажми на меня
		</div>;
	}
}
```

Запустите этот код и нажмите на div - вы увидите алерт с сообщением (там будет **this.state.name**, то есть 'Иван').

Подробнее про **bind** читайте в уроке по продвинутой работе с контекстом и this на JavaScript.

Можно сделать и по-другому: привяжем контекст к нашему методу **showMessage** прямо в конструкторе - и потом сможем спокойно использовать наш метод, не заботясь о контексте:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван', surname: 'Иванов'};

		//Привяжем к showMessage контекст навсегда:
		this.showMessage = this.showMessage.bind(this);
	}

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

	render() {
		//Тут больше не надо байндить:
		return <div onClick={this.showMessage}>
			нажми на меня
		</div>;
	}
}
```

Если вы запустите этот код и нажмете на div, то все будет работать так, как и работало: вы увидите алерт с сообщением 'Иван'.

Вы можете пользоваться тем способом, который вам кажется более удобным.

Изменение стейта

Пусть в **this.state** хранится имя пользователя:

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

```
	constructor() {
		super();
		this.state = {name: 'Иван'};
	}
}
```

Давайте в методе **render** выведем его на экран:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван'};
	}

	render() {
		//Выводим имя из стейта:
		return <div>
			<p>имя: {this.state.name}</p>
		</div>;
	}
}
```

Давайте теперь сделаем кнопку, по нажатию на которую *будет меняться имя пользователя*. Пусть по клику на эту кнопку вызывается метод **changeName**:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван'};
	}

	changeName() {
		//тут код для изменения имени;
	}

	render() {
		return <div>
			<p>имя: {this.state.name}</p>
			<button onClick={this.changeName.bind(this)}>сменить
имя</button>
		</div>;
	}
}
```

Обратите также внимание на **.bind(this)** в строчке onClick={this.changeName.bind(this)}. Тут мы привязываем this так, как это было описано выше.

Итак, давайте напишем реализацию метода **changeName**. В нем мы хотим поменять имя пользователя. Однако, менять значение **this.state.name** напрямую, вот так: **this.state.name = 'новое значение'** - нельзя.

Для таких дел следует использовать специальный встроенный метод **this.setState**. Этот метод не нужно создавать в нашем классе - он наследуется от класса **React.Component** и доступен в любом компоненте по умолчанию.

С помощью этого метода мы сможем поменять значение **this.state.name**, вот так: **this.setState({name: 'Коля'})**.

Почему именно так, а не напрямую: потому что в этом случае все места вставки вида **{this.state.name}** сменят свое значение.

То есть: после вызова **this.setState({name: 'Коля'})** в коде ** **имя: {this.state.name}** ** вместо this.state.name станет не Вася, а Коля. Причем изменения произойдут мгновенно во всех таких вставках. В нашем случае - сразу после нажатия на кнопку 'сменить имя'.

В этом фишка фреймворков типа React - любые изменения стейта мгновенно отображаются на странице. Конечно, если эти изменения сделаны через **this.setState**, а не напрямую (напрямую вообще их делать не стоит).

Такое поведение называется *реактивностью* (отсюда видимо и произошло название фреймворка - реакт, реактивность).

Давайте соберем все вместе:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван'};
	}

	//Меняем имя на 'Коля':
	changeName() {
		this.setState({name: 'Коля'});
	}

	render() {
		//Выводим имя и кнопку для изменения:
		return <div>
			<p>имя: {this.state.name}</p>
			<button onClick={this.changeName.bind(this)}>сменить
имя</button>
		</div>;
```

```
	}
}
```

Запустите этот код и нажмите на кнопку - изменение имени произойдет мгновенно по нажатию на кнопку.

Примечание

Если в стейте несколько значений, то в **this.setState** мы передаем только те, которые хотим поменять. Пример: пусть в стейте кроме имени хранится еще и фамилия, но мы по прежнему хотим менять только имя:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван', surname: 'Иванов'};
	}

	changeName() {
		//Меняем только имя:
		this.setState({name: 'Коля'});
	}

	render() {
		//Выводим имя, фамилию и кнопку для изменения:
		return <div>
			<p>имя: {this.state.name}</p>
			<p>фамилия: {this.state.surname}</p>
			<button onClick={this.changeName.bind(this)}>сменить
имя</button>
		</div>;
	}
}
```

Если вы запустите этот код и нажмете на div, то все будет работать так, как и работало: 'Иван' мгновенно поменяется на 'Коля'.

Но никто не мешает нам поменять несколько значений:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {name: 'Иван', surname: 'Иванов'};
	}

	changeName() {
		//Меняем и имя, и фамилию:
		this.setState({name: 'Коля', surname: 'Петров'});
```

```
	}

	render() {
		//Выводим имя, фамилию и кнопку для изменения:
		return <div>
			<p>имя: {this.state.name}</p>
			<p>фамилия: {this.state.surname}</p>
			<button onClick={this.changeName.bind(this)}>сменить
имя</button>
		</div>;
	}
}
```