Сейчас мы с вами будем разбирать добавление и удаление элементов в массив. Это добавление или удаление будет приводить к тому, что вывод элементов массива на экран будет автоматически изменятся. Если, к примеру, этот массив выведем в виде списка - то в этом списке будут появляется новые элементы и удалятся старые. Давайте попробуем на практике.

Начало

Пусть у нас в стейте хранится массив. Давайте выведем его элементы в виде списка **ul**. Такое вы уже умеете делать, пока это предварительный код, на основе которого мы будем изучать новый материал.

Итак, вот код:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {items: [1, 2, 3, 4, 5]};
	}

	render() {
		const list = this.state.items.map((item, index) => {
			return <li key={index}>{item}</li>;
		});

		return (
			<div>
				<ul>
					{list}
				</ul>
			</div>
		);
	}
}
```

Результатом работы этого кода будет следующее:

```
<ul>
	<li>1</li>
	<li>2</li>
	<li>3</li>
	<li>4</li>
	<li>5</li>
</ul>
```

Добавление элемента

Продолжаем. Следующий нюанс, который вы должны уловить до конца: *изменение любого элемента стейта через **setState** приводит к перерендеренгу соответствующих элементов страницы*.

Это работает, даже если там хранится массив. То есть, если мы в **this.state.items** добавим новый элемент - он сразу же появится на странице в конце списка **ul**. Главное, чтобы изменения были сделаны не напрямую, а через **setState**.

Давайте в нашем классе сделаем метод **addItem**, который будет добавлять новый элемент с текстом 'пункт' в **this.state.items**. Реализуем это, к примеру, через обычный метод push для работы с массивами:

```
addItem() {
	this.state.items.push('пункт'); //тут изменяем стейт напрямую
	this.setState({items: this.state.items}); //заставляем изменения
применитьcя
}
```

Добавим также кнопку, по нажатию на которую будет вызываться **addItem**:

```
return (
	<div>
		<ul>
			{list}
		</ul>
		<button onClick={this.addItem.bind(this)}>добавить</button>
	</div>
);
```

Соберем все вместе и запустим наш код - каждое нажатие на кнопку будет приводить к добавлению **li** с текста 'пункт' в конец нашей **ul**:

```
class App extends React.Component {
	constructor() {
		super();
		this.state = {items: [1, 2, 3, 4, 5]};
	}

	//Добавляем в конец списка новый пункт:
	addItem() {
		this.state.items.push('пункт');
		this.setState({items: this.state.items});
	}

	render() {

		//Формируем набор из тегов li:
```

```
            const list = this.state.items.map((item, index) => {
                  return <li key={index}>{item}</li>;
            });

            //Привязываем к кнопке метод addItem:
            return (
                  <div>
                        <ul>
                              {list}
                        </ul>
                        <button
onClick={this.addItem.bind(this)}>добавить</button>
                  </div>
            );
      }
}
```

Запустите этот код и понажимайте на кнопку - каждое нажатие будет приводить к добавлению нового элемента в конец списка.

Удаление первого элемента

Давайте теперь по клику на кнопку будем каждый раз удалять первый элемент. Для этого к **this.state.items** применим обычный метод slice для работы с массивами:

```
class App extends React.Component {
      constructor() {
            super();
            this.state = {items: [1, 2, 3, 4, 5]};
      }

      //Удаляем первый элемент:
      deleteItem() {
            this.setState({items: this.state.items.slice(1)});
      }

      render() {

            //Формируем набор из тегов li:
            const list = this.state.items.map((item, index) => {
                  return <li key={index}>{item}</li>;
            });

            //Привязываем к кнопке метод deleteItem:
            return (
                  <div>
                        <ul>
                              {list}
                        </ul>
```

```
                        <button
onClick={this.deleteItem.bind(this)}>удалить</button>
                    </div>
            );
      }
}
```

Запустите этот код и понажимайте на кнопку - каждое нажатие будет приводить к удалению первого элемента из списка.

Удаление заданного элемента

Давайте теперь по клику на кнопку удалим **li** с номером **3** (нумерация с нуля, то есть это будет 4-тая li сверху).

Для этого к **this.state.items** применим метод splice для работы с массивами:

```
deleteItem() {
      this.state.items.splice(3, 1); //тут 3 - номер элемента для удаления
      this.setState({items: this.state.items});
}
```

Не будем это проверять, думаю идея понятна. Давайте лучше сделаем следующее: пусть номер **li** для удаления передается параметром в метод **deleteItem**, вот так:

```
//Параметр num - номер li для удаления:
deleteItem(num) {
      this.state.items.splice(num, 1); //num - номер элемента для удаления
      this.setState({items: this.state.items});
}
```

Этот номер для удаления мы можем передать вторым параметром в метод **bind** (см. уроки по работе с контекстом, если вы не понимаете, почему мы так можем):

```
<button onClick={this.deleteItem.bind(this, 3)}>удалить</button>
```

Соберем все вместе. Если вы запустите следующий код и нажмете на кнопку - удалится **li** с номером **3**:

```
class App extends React.Component {
      constructor() {
            super();
            this.state = {items: [1, 2, 3, 4, 5]};
      }
```

```
	//Удаляем заданный элемент:
	deleteItem(num) {
		this.state.items.splice(num, 1);
		this.setState({items: this.state.items});
	}

	render() {

		//Формируем набор из тегов li:
		const list = this.state.items.map((item, index) => {
			return <li key={index}>{item}</li>;
		});

		//По нажатию на кнопку удаляем третий элемент:
		return (
			<div>
				<ul>
					{list}
				</ul>
				<button onClick={this.deleteItem.bind(this,
3)}>удалить</button>
			</div>
		);
	}
}
```

Запустите этот код и нажмите на кнопку - удалится элемент с номером 3.

Удаление любого элемента

Давайте теперь сделаем кнопочку 'удалить' внутри каждой **li**. По нажатию на эту кнопочку будет удалятся та **li**, в которой эта кнопочка находится.

Для этого внутри **li** кроме текста выведем еще и кнопочку, к которой привяжем метод **deleteItem**:

```
let list = this.state.items.map((item, index) => {
	return <li key={index}>
		{item}
		<button onClick={this.deleteItem.bind(this, index)}>
			удалить
		</button>
	</li>;
});
```

Обратите внимание на то, что вторым параметром в методе **bind** передается **index** - номер элемента в массиве:

```
<button onClick={this.deleteItem.bind(this, index)}>
      удалить
</button>
```

Этот номер будет передаваться параметром в метод **deleteItem**:

```
deleteItem(index) {
      this.state.items.splice(index, 1);
      this.setState({items: this.state.items});
}
```

Таким образом каждая кнопочка при нажатии будет передавать номер своей **li** и именно **li** с таким номером и будет удаляться методом **deleteItem**.

Давайте напишем этот код:

```
class App extends React.Component {
      constructor() {
            super();
            this.state = {items: [1, 2, 3, 4, 5]};
      }

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

      render() {

            //Формируем набор из тегов li:
            const list = this.state.items.map((item, index) => {
                  return <li key={index}>
                        {item}
                        <button onClick={this.deleteItem.bind(this, index)}>
                              удалить
                        </button>
                  </li>;
            });

            return (
                  <div>
                        <ul>
                              {list}
                        </ul>
                  </div>
```

```
                );
        }
}
```

Запустите этот код и понажимайте на кнопки - каждая кнопка будет удалять свою **li**.