**Документация**

ОСНОВНЫЕ ПОНЯТИЯ

* [1. Привет, мир](https://ru.reactjs.org/docs/hello-world.html)
* [2. Знакомство с JSX](https://ru.reactjs.org/docs/introducing-jsx.html)
* [3. Рендеринг элементов](https://ru.reactjs.org/docs/rendering-elements.html)
* [4. Компоненты и пропсы](https://ru.reactjs.org/docs/components-and-props.html)
* [5. Состояние и жизненный цикл](https://ru.reactjs.org/docs/state-and-lifecycle.html)
* [6. Обработка событий](https://ru.reactjs.org/docs/handling-events.html)
* [7. Условный рендеринг](https://ru.reactjs.org/docs/conditional-rendering.html)
* [8. Списки и ключи](https://ru.reactjs.org/docs/lists-and-keys.html)
* [9. Формы](https://ru.reactjs.org/docs/forms.html)
* [10. Подъём состояния](https://ru.reactjs.org/docs/lifting-state-up.html)
* [11. Композиция против наследования](https://ru.reactjs.org/docs/composition-vs-inheritance.html)
* [12. Философия React](https://ru.reactjs.org/docs/thinking-in-react.html)

Почитать отдельно:

Посмотрите руководство о SyntheticEvent - <https://ru.reactjs.org/docs/events.html>

Почему индексы-ключи приводят к проблемам - <https://medium.com/@robinpokorny/index-as-a-key-is-an-anti-pattern-e0349aece318>

Индексы массива в качестве ключей плохо - <https://ru.reactjs.org/docs/reconciliation.html#recursing-on-children>

альтернативная техника реализации ввода данных в форму - <https://ru.reactjs.org/docs/uncontrolled-components.html>

присмотритесь к Formik - <https://jaredpalmer.com/formik/> Эта библиотека построена на принципах управляемых компонентов и управления состоянием, так что не пренебрегайте их изучением.

**Урок 6 - Обработка событий**

Например, в HTML:

<button onclick="activateLasers()">

Активировать лазеры

</button>

В React немного иначе:

<button onClick={activateLasers}>

Активировать лазеры

</button>

**Передача аргументов в обработчики событий**

Внутри цикла часто нужно передать дополнительный аргумент в обработчик события. Например, если id — это идентификатор строки, можно использовать следующие варианты:

<button onClick={(e) => this.deleteRow(id, e)}>Удалить строку</button>

<button onClick={this.deleteRow.bind(this, id)}>Удалить строку</button>

Две строки выше — эквивалентны, и используют [стрелочные функции](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Functions/Arrow_functions) и [Function.prototype.bind](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Function/bind) соответственно.

В обоих случаях аргумент e, представляющий событие React, будет передан как второй аргумент после идентификатора. Используя стрелочную функцию, необходимо передавать аргумент явно, но с bind любые последующие аргументы передаются автоматически.

**Урок 7**

**Условный рендеринг**

сыссы

**Урок 8**

**Списки и ключи**

Почему индексы-ключи приводят к проблемам - <https://medium.com/@robinpokorny/index-as-a-key-is-an-anti-pattern-e0349aece318>

Индексы массива в качестве ключей плохо - <https://ru.reactjs.org/docs/reconciliation.html#recursing-on-children>

**Урок - 9**

**Формы**

форму удобнее обрабатывать с помощью JavaScript-функции, у которой есть доступ к введённым данным. Стандартный способ реализации такого поведения называется «управляемые компоненты»

import React from 'react';

class FlavorForm extends React.Component {

    constructor(props) {

*super*(props);

*this*.state = {

            value: 'coconut'

        };

*this*.handleChange = *this*.handleChange.bind(*this*);

*this*.handleSubmit = *this*.handleSubmit.bind(*this*);

    }

    handleChange(event) {

*this*.setState({

            value: event.target.value

        });

    }

    handleSubmit(event) {

        alert('Ваш любимый вкус: ' + *this*.state.value);

        event.preventDefault();

    }

    // event.target.children - не работает

    render() {

        return (

            <div style={{

                border: '1px solid #ccc',

                padding: '20px',}}

            >

                <h2>Выбери любимый вкус</h2>

                <form onSubmit={*this*.handleSubmit}>

                    <label>

                        casc:

                        <select value={*this*.state.value} onChange={*this*.handleChange}>

                            <option value="grapefruit">Грейпфрут</option>

                            <option value="lime">Лайм</option>

                            <option value="coconut">Кокос</option>

                            <option value="mango">Манго</option>

                        </select>

                    </label>

                    <input type="submit" value='send'/>

                </form>

            </div>

        );

    }

}

export default FlavorForm;

import React from 'react';

class EssayForm extends React.Component {

    constructor(props) {

*super*(props);

*this*.state = ({

            value: 'Напишите ваш текст сюда'

        });

*this*.handleChange = *this*.handleChange.bind(*this*);

*this*.handleSubmit = *this*.handleSubmit.bind(*this*);

    }

    handleChange(event) {

*this*.setState({

            value: event.target.value

        });

    }

    handleSubmit(event) {

        console.log(*this*.state.value);

        event.preventDefault();

    }

    render() {

        return (

            <div style={{

                border: '1px solid #ccc',

                padding: '20px',}}

            >

                <form onSubmit={*this*.handleSubmit}>

                    <h2>Поле для текста</h2>

                    <label>

                        Текст:

                        <textarea value={*this*.state.value} onChange={*this*.handleChange}></textarea>

                    </label>

                    <input type="submit" value='Push'/>

                </form>

            </div>

        );

    }

}

export default EssayForm;

import React from 'react';

class Reservation extends React.Component {

    constructor(props) {

*super*(props);

*this*.state = {

            isGoing: true,

            numberOfGuests: 2

        }

*this*.handleInputChange = *this*.handleInputChange.bind(*this*);

    }

    handleInputChange(e) {

        const target = e.target;

        const value = target.type === 'checkbox' ? target.checked : target.value;

        const name = target.name;

        // внутри setState() передаем вычисляемое св-во объекта name

*this*.setState({

            [name]: value

        });

        ((n = *this*.state.numberOfGuests) => console.log('Кол-во гостей: ', n))();

    }

    render() {

        return(

            <div style={{

                border: '1px solid #ccc',

                padding: '20px',}}

            >

                <form>

                    <label>

                        Пойду:

                        <input

                            name='isGoing'

                            type="checkbox"

                            checked={*this*.state.isGoing}

                            onChange={*this*.handleInputChange}/>

                    </label>

                    <br/>

                    <label>

                        Кол-во гостей:

                        <input

                            name='numberOfGuests'

                            type="number"

                            value={*this*.state.numberOfGuests}

                            onChange={*this*.handleInputChange}/>

                    </label>

                </form>

            </div>

        );

    }

}

export default Reservation;

**Урок**

**10 Подъём состояния**

import React from 'react';

import { template } from '@babel/core';

const scaleNames = {

    c: 'Цельсия',

    f: 'Фаренгейта'

};

class TemperatureInput extends React.Component {

    constructor(props) {

*super*(props);

        // this.state = {

        //     temperature: ''

        // };

*this*.handleChange = *this*.handleChange.bind(*this*);

    }

    handleChange(e) {

        // this.setState({

        //     temperature: e.target.value

        // });

*this*.props.onTemperatureChange(e.target.value);

    }

    render() {

        const temperature = *this*.props.temperature;

        const scale = *this*.props.scale;

        return (

            <fieldset>

                <legend>Введите температуру в градусах {scaleNames[scale]}: </legend>

                <input

                    value={temperature}

                    onChange={*this*.handleChange}/>

            </fieldset>

        );

    }

}

class Calculater extends React.Component {

    constructor(props) {

*super*(props);

*this*.state = {

            scale: 'c',

            temperature: ''

        };

*this*.handleCelsiusChange = *this*.handleCelsiusChange.bind(*this*);

*this*.handleFahrenheitChange = *this*.handleFahrenheitChange.bind(*this*);

    }

    handleCelsiusChange(temperature) {

*this*.setState({

            scale: 'c',

            temperature

        });

    }

    handleFahrenheitChange(temperature) {

*this*.setState({

            scale: 'f',

            temperature

        });

    }

    render() {

        const scale = *this*.state.scale;

        const temperature = *this*.state.temperature;

        const celsius = scale === 'f' ? tryConvert(temperature, toCelsius) : temperature;

        const fahrenheit = scale === 'c' ? tryConvert(temperature, toFahrenheit) : temperature;

        return(

            <div>

                <TemperatureInput

                    scale='c'

                    temperature={celsius}

                    onTemperatureChange={*this*.handleCelsiusChange}/>

                <TemperatureInput

                    scale='f'

                    temperature={fahrenheit}

                    onTemperatureChange={*this*.handleFahrenheitChange}/>

                <BoilingVerdict

                    celsius={parseFloat(celsius)} />

            </div>

        );

    }

}

function BoilingVerdict(props) {

    if(props.celsius >= 100) {

        return <p>Вода Кипит ;)</p>

    }

    return <p>Вода еще не закипела :(</p>;

}

function toCelsius(fahrenheit) {

    return (fahrenheit - 32) \* 9 / 5;

}

function toFahrenheit(celsius) {

    return (celsius \* 9 / 5) + 32;

}

function tryConvert(temperature, convert) {

    const input = parseFloat(temperature);

    if(Number.isNaN(input)) {

        return '';

    }

    const output = convert(input);

    const rounded = Math.round(output \* 1000) / 1000;

    return rounded.toString();

}

export default Calculater;