**Создание проекта**

**Create React App**

**Команда** npx create-react-app my-app

Плагин JSHint надо отключать

Заходим на cdnjs сервер <https://cdnjs.com/>

# twitter-bootstrap

bootstrap.min.css копируем

вставляем перед

вставляем в index.html перед <title></title>

затем

# font-awesome

копируем

all.min.css

также вставляем перед

вставляем в index.html перед <title></title>

**91. Свойства компонентов**

function WhoAmi(props) {  //props - это объект

  return (

    <div>

      <h1>my names {props.name}, surname - {props.surname}</h1>

      <a href={props.link}>My profile</a>

    </div>

  )

}

function App() {

  return (

    <div className="App">

        <WhoAmi name='Ivan' surname='Petrov' link='facebook.com'/>

        <WhoAmi name='Alex' surname='Igolkin' link='vk.com'/>

    </div>

  );

}

Современный код

function WhoAmi({name, surname, link}) {  //Деструктурируем

  return (

    <div>

      <h1>my names {name}, surname - {surname}</h1>

      <a href={link}>My profile</a>

    </div>

  )

}

function App() {

  return (

    <div className="App">

        <WhoAmi name='Ivan' surname='Petrov' link='facebook.com'/> // это props они не измяняемы

        <WhoAmi name='Alex' surname='Igolkin' link='vk.com'/>

    </div>

  );

}

Передаем объект в пропсы

function WhoAmi({name, surname, link}) {  //Деструктурируем

  return (

    <div>

      <h1>my names {name.firstname}, surname - {surname}</h1>

      <a href={link}>My profile</a>

    </div>

  )

}

function App() {

  return (

    <div className="App">

         {/\* //ПЕРЕДАЕМ объект {firstname: 'Ivan'} \*/}

        <WhoAmi name={{firstname: 'Ivan'}} surname='Petrov' link='facebook.com'/>

        <WhoAmi name='Alex' surname='Igolkin' link='vk.com'/>

    </div>

  );

}

Передаем функцию в пропс

function WhoAmi({name, surname, link}) {  //Деструктурируем

  return (

    <div>

       {/\* должны вызвать функцию name() \*/}

      <h1>my names {name()}, surname - {surname}</h1>

      <a href={link}>My profile</a>

    </div>

  )

}

function App() {

  return (

    <div className="App">

         {/\* ПЕРЕДАЕМ ФУНКЦИЮ () => {return 'Ivan'} \*/}

        <WhoAmi name={() => {return 'Ivan'}} surname='Petrov' link='facebook.com'/>

        <WhoAmi name={() => {return 'Ivan'}} surname='Igolkin' link='vk.com'/>

    </div>

  );

}

**Практика**

const EmployeesList = ({data}) => {

    const element = data.map(item => {

        return (

            <EmployeesListItem {...item}/>

        )

    })

    return (

        <ul className="app-list list-group">

            {element}

        </ul>

    )

}

**129. Работа со списками и алгоритм согласования**

**Работа со списками key**

**ДЛЯ ЧЕГО НУЖЕН – ДЛЯ ОПТИМИЗАЦИИ РАБОТЫ ПРИЛОЖЕНИЯ (СКОРОСТЬ)**

ПРИМЕР

const EmployeesList = ({data}) => {

    const element = data.map(item => {

        const {id, ...itemProps} = item;

        return (

            <EmployeesListItem key={id} {...itemProps}/>

        )

    })

**94. Состояния компонентов**

import { Component } from 'react';

import logo from './logo.svg';

import './App.css';

class WhoAmi extends Component {  //Деструктурируем

  constructor(props) {

*super*(props);

*this*.state = {

      years: 37,

      experience: 10,

      text: '+++++'

    }

 }

    nextYear = () => {

*this*.setState(state => ({

            years: state.years + 1

          }))

      }

  render() {

    const {name, surname, link} = *this*.props;

    return (

      <div>

         {/\* должны вызвать функцию name() \*/}

         <button onClick={*this*.nextYear} >{*this*.state.text}</button>

        <h1>my names {name}, surname - {surname}, years - {*this*.state.years}, experience - {*this*.state.experience} </h1>

        <a href={link}>My profile</a>

      </div>

    )

  }

}

function App() {

  return (

    <div className="App">

         {/\* ПЕРЕДАЕМ ФУНКЦИЮ () => {return 'Ivan'} \*/}

        <WhoAmi name='Ivan' surname='Petrov' link='facebook.com'/>

        <WhoAmi name='Max' surname='Igolkin' link='vk.com'/>

    </div>

  );

}

export default App;

**132. События в React и вспоминаем this**

**События commitInputChanges**

commitInputChanges = (e) => {

*this*.setState({

      position: e.target.value

    })

  }

  render() {

    const {name, surname, link} = *this*.props;

    const {position, years} = *this*.state;

    return (

      <div>

        <button onClick={*this*.nextYear}>{*this*.state.text}</button>

        <h1>  My name is {name},

              surname - {surname},

              age - {years},

              position - {position}</h1>

        <a href={link}>My profile</a>

        <form>

          <span>Введите должность</span>

          <input type="text" onChange={*this*.commitInputChanges} />

        </form>

      </div>

    )

  }

}

**Аргументы**

 commitInputChanges = (e, color) => { //аргументы 'some color'

    console.log(color);

*this*.setState({

      position: e.target.value

    })

  }

  render() {

    const {name, surname, link} = *this*.props;

    const {position, years} = *this*.state;

    return (

      <div>

        <button onClick={*this*.nextYear}>{*this*.state.text}</button>

        <h1>  My name is {name},

              surname - {surname},

              age - {years},

              position - {position}</h1>

        <a href={link}>My profile</a>

        <form>

          <span>Введите должность</span>

          <input type="text"

           onChange={(e) => *this*.commitInputChanges(e, 'some color')}/> аргументы 'some color'

        </form>

      </div>

    )

  }

}

**134. Работа с формами, управляемые и неуправляемые**

render() {

        const {name, salary} = *this*.state

        return (

            <div className="app-add-form">

                <h3>Добавьте нового сотрудника</h3>

                <form

                    className="add-form d-flex">

                    <input type="text"

                        className="form-control new-post-label"

                        placeholder="Как его зовут?"

                        name="name"

                        value={name} //управляемая форма, двойная привязка

                        onChange={*this*.onValueChange}/>