Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 8**

### Тема: Доклад на тему «Библиотека JavaScript React.js»

Студент: Гордов М.О.

Группа: 4ИСИП-619

Преподаватель: Сибирев И.В.

Дата: 28.02.2023

React.js

React.js (также известная как ReactJS или React) - это интерфейсная библиотека JavaScript с открытым исходным кодом. Он был создан в 2013 году Джорданом Уолке, который работает в Facebook инженером-программистом.

Теперь он имеет лицензию MIT, но изначально был выпущен под лицензией Apache 2.0. React был разработан, чтобы сделать создание интерактивного пользовательского интерфейса безболезненным.

Просто создайте простое представление для отдельных состояний в вашем приложении. Затем он будет эффективно отображать и обновлять нужный компонент при изменении данных.

Библиотека React

Библиотека React

Особенности / преимущества:

Код React содержит компоненты или сущности, которые нуждаются в рендеринге для определенного элемента в DOM с помощью библиотеки React DOM.

Он использует виртуальный DOM, создавая кэш в памяти в структуре данных, вычисляя разницу и эффективно обновляя отображаемый DOM в браузере.

Благодаря такому выборочному рендерингу повышается производительность приложения, при этом экономятся усилия разработчиков по пересчету макета страницы, стилей CSS и рендеринга всей страницы.

Он использует методы жизненного цикла, такие как render и componentDidMount, чтобы разрешить выполнение кода в определенные моменты в течение срока службы объекта.

Он поддерживает JavaScript XML (JSX), который сочетает в себе как JS, так и HTML. Это помогает в рендеринге компонентов с вложенными элементами, атрибутами, выражениями JS и условными операторами.

Примеры использования:

Служит основой при разработке мобильных или одностраничных приложений.

Отображение состояния в DOM и управление им.

Создание эффективных пользовательских интерфейсов при разработке веб-приложений и интерактивных сайтов.

Упрощение отладки и тестирования.

Бонусный момент: Facebook, Instagram и Whatsapp используют React.

Примеры кода:

Компонент с использованием внешних плагинов

React позволяет работать с другими библиотеками и фреймворками. В этом примере используется внешняя библиотека remarkable, чтобы в режиме реального времени преобразовать Markdown-синтаксис, введённый в элемент <textarea>.

class MarkdownEditor extends React.Component {

constructor(props) {

super(props);

this.md = new Remarkable();

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

this.state = { value: 'Привет, \*\*мир\*\*!' };

}

handleChange(e) {

this.setState({ value: e.target.value });

}

getRawMarkup() {

return { \_\_html: this.md.render(this.state.value) };

}

render() {

return (

<div className="MarkdownEditor">

<h3>Редактор</h3>

<label htmlFor="markdown-content">

Введите что-нибудь

</label>

<textarea

id="markdown-content"

onChange={this.handleChange}

defaultValue={this.state.value}

/>

<h3>Вывод</h3>

<div

className="content"

dangerouslySetInnerHTML={this.getRawMarkup()}

/>

</div>

);

}

}

root.render(<MarkdownEditor />);

Простой компонент

React-компоненты реализуют метод render(), который принимает входные данные и возвращает что-то для вывода. В этом примере используется XML-подобный синтаксис под названием JSX. Входные данные, передаваемые в компонент, доступны в render() через this.props.

JSX необязателен для работы с React. Попробуйте [Babel REPL](https://babeljs.io/repl/" \l "?presets=react&code_lz=MYewdgzgLgBApgGzgWzmWBeGAeAFgRgD4AJRBEAGhgHcQAnBAEwEJsB6AwgbgChRJY_KAEMAlmDh0YWRiGABXVOgB0AczhQAokiVQAQgE8AkowAUAcjogQUcwEpeAJTjDgUACIB5ALLK6aRklTRBQ0KCohMQk6Bx4gA" \t "_blank), чтобы увидеть JavaScript-код, полученный на этапе компиляции JSX.

class HelloMessage extends React.Component {

render() {

return (

<div>

Привет, {this.props.name}

</div>

);

}

}

root.render(<HelloMessage name="Саша" />);

Приложение

Используя props и state, можно создать небольшое приложение списка дел. В этом примере используется state для отслеживания текущего списка элементов, а также текста, введённого пользователем. Хотя обработчики событий встроены в разметку, они собираются и реализуются с помощью делегирования событий.

class TodoApp extends React.Component {

constructor(props) {

super(props);

this.state = { items: [], text: '' };

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

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

}

render() {

return (

<div>

<h3>Список дел</h3>

<TodoList items={this.state.items} />

<form onSubmit={this.handleSubmit}>

<label htmlFor="new-todo">

Что нужно сделать?

</label>

<input

id="new-todo"

onChange={this.handleChange}

value={this.state.text}

/>

<button>

Добавить #{this.state.items.length + 1}

</button>

</form>

</div>

);

}

handleChange(e) {

this.setState({ text: e.target.value });

}

handleSubmit(e) {

e.preventDefault();

if (this.state.text.length === 0) {

return;

}

const newItem = {

text: this.state.text,

id: Date.now()

};

this.setState(state => ({

items: state.items.concat(newItem),

text: ''

}));

}

}

class TodoList extends React.Component {

render() {

return (

<ul>

{this.props.items.map(item => (

<li key={item.id}>{item.text}</li>

))}

</ul>

);

}

}

root.render(<TodoApp />);