федеральное государственное автономное образовательное учреждение

высшего образования

«Российский университет транспорта (МИИТ)»

Кафедра «Управление и защита информации»

**ОТЧЕТ**

по учебной практике

|  |  |
| --- | --- |
| Выполнила | Куликов Г.А. |
| Группа | ТУУ–311 |
| Проверил | Сафронов А.И. |

Москва 2024

**Содержание**

[Цель работы 3](#_Toc177364864)

[Задачи 3](#_Toc177364865)

[Сеть Петри 4](#_Toc177364866)

[Листинг (код) программного обеспечения 4](#_Toc177364867)

[Листинг (код) внешнего вида калькулятора 9](#_Toc177364868)

[Графический пользовательский интерфейс 13](#_Toc177364869)

[Вывод 14](#_Toc177364870)

# Цель работы

Освоить навыки создания веб приложения в виде простейшего калькулятора на основе библиотек React.

# Задачи

Построить внешний вид нашего калькулятора с заданными простейшими функциями.

Сделать рабочими все кнопки и логические операции с помощью VS Code.

Сделать из рабочего калькулятора монолитное приложение, чтобы всегда был доступ к нему без использования интернета.

# Сеть Петри

Изображение выглядит как диаграмма, зарисовка, линия

Автоматически созданное описание

Рисунок 1. –– Сеть Петри для калькулятора (общий вид).

Как видно из схемы, калькулятор имеет возможность в любой момент времени очистить поле ввода, либо его частью. Одного «маркера» достаточно, чтобы построенная сеть работала без перебоев, предусмотрена возможность ввода неограниченного ряда чисел.

# Листинг (код) программного обеспечения

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="utf-8" />**

**<link rel="icon" href="favicon.ico" />**

**<meta name="viewport" content="width=device-width, initial-scale=1" />**

**<meta name="theme-color" content="#000000" />**

**<meta**

**name="description"**

**content="Создано на основе React.js"**

**/>**

**<script src="react.development.js"></script>**

**<script src="react-dom.development.js"></script>**

**<script src="babel.min.js"></script>**

**<link href="index.css" rel="stylesheet">**

**<title>Калькулятор React</title>**

**</head>**

**<body>**

**<noscript>Необходимо активировать JavaScript для запуска калькулятора</noscript>**

**<div id="root"></div>**

**<script type="text/babel">**

**class Calculator extends React.Component**

**{**

**constructor(props)**

**{**

**super(props);**

**this.state = {**

**calc: '',**

**result: '',**

**digits: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '.', '='],**

**ops: ['/', '\*', '+', '-', 'DEL', 'C'],**

**flgLoaded: false**

**};**

**}**

**updateCalc(e)**

**{**

**let item = e.target.innerText;**

**let val = '';**

**let res = 0;**

**switch (item)**

**{**

**case 'C':**

**this.setState({calc: '', result: ''});**

**break;**

**case 'DEL':**

**if (this.state.calc == '')**

**{**

**return;**

**}**

**val = this.state.calc.slice(0, -1);**

**this.setState({calc: val});**

**if (val.length > 0)**

**{**

**if (!this.state.ops.includes(val.slice(-1)))**

**{**

**res = eval(val);**

**this.setState({result: res.toString()});**

**}**

**}**

**else**

**{**

**this.setState({result: ''});**

**}**

**break;**

**case '=':**

**val = this.state.result;**

**this.setState({calc: val, result: ''});**

**break;**

**default:**

**// Заперетили использовать операторы**

**// больше одного раза и вводить ранше чисел**

**if (this.state.ops.includes(item) && this.state.calc === '' ||**

**this.state.ops.includes(item) &&**

**this.state.ops.includes(this.state.calc.slice(-1)))**

**{**

**return;**

**}**

**val = this.state.calc + item;**

**this.setState({calc: val});**

**if (!this.state.ops.includes(item))**

**{**

**res = eval(val);**

**this.setState({result: res.toString()});**

**}**

**break;**

**}**

**}**

**eachOp(item, i)**

**{**

**return (**

**<button key={i} index={i} onClick={(e) => this.updateCalc(e)}>**

**{item}**

**</button>**

**)**

**}**

**eachDig(item, i)**

**{**

**return (**

**<button key={i} index={i} onClick={(e) => this.updateCalc(e)}>**

**{item}**

**</button>**

**);**

**}**

**render()**

**{**

**return (**

**<div className="calculator">**

**<div className="display">**

**{ this.state.result ? <span>({this.state.result})</span> : '' } { this.state.calc || "0" }**

**</div>**

**<div className="operators">**

**{ this.state.ops.map((item, i) => this.eachOp(item, i)) }**

**</div>**

**<div className="digits">**

**{ this.state.digits.map((item, i) => this.eachDig(item, i)) }**

**</div>**

**</div>**

**)**

**}**

**}**

**const root = ReactDOM.createRoot(document.getElementById('root'));**

**root.render(**

**<div className="App">**

**<Calculator />**

**</div>);**

**</script>**

**</body>**

**</html>**

# Листинг (код) внешнего вида калькулятора

/\* Задали используемые в будущем цвета \*/

:root {

--primary: #d81e5b;

--secondary: #8c38ff;

--dark: #131a26;

--light: #eee;

}

/\* Задали размеры для кнопок \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Fira Sans', sans-serif;

}

/\* Задали задний фон \*/

body {

background-color: var(--light);

}

/\* Переместили наши кнопки в ценрт экрана \*/

.App {

display: flex;

min-height: 100vh;

align-items: center;

justify-content: center;

padding: 16px;

}

/\* Задали размеры и цыет нашего калькулятора \*/

.calculator {

width: 100%;

max-width: 400px;

background-color: #FFF;

border-radius: 16px;

overflow: hidden;

box-shadow: 0px 2px 64px rgba(0, 0, 0, 0.2);

}

/\* Задали размеры и цвета дисплея \*/

.display {

//display: flex;

//position: relative;

padding: 16px;

text-align: right;

background-color: var(--dark);

color: var(--light);

font-size: 24px;

font-weight: 300;

}

/\* Задали размер и цвет числа в скобках \*/

.display span {

font-size: 14px;

color: #888;

}

.display button {

flex: 0 1 20%;

font-size: 24px;

background-color: var(--primary);

font-weight: 700;

}

/\* Задали размер и цвет кнопок, добавили курсор \*/

button {

appearance: none;

border: none;

outline: none;

color: #FFF;

font-size: 20px;

padding: 16px;

cursor: pointer;

transition: 0.4s;

}

/\* Сделали кнопки немного светлее при наведении на них\*/

button:hover {

opacity: 0.9;

}

/\* Можем растягивать теперь кнопки операций \*/

.operators {

display: flex;

}

/\* Подогнали размер под калькулятор, изменили цвет и жирность \*/

.operators button {

flex: 1 1 0%;

background-color: var(--primary);

font-weight: 700;

}

/\* Можем растягивать кнопки цифр \*/

.digits {

display: flex;

flex-wrap: wrap;

}

/\* Подогнали размер под калькулятор, изменили цвет и жирность \*/

.digits button {

flex: 1 1 25%;

max-width: 25%;

background-color: var(--dark);

}

# Графический пользовательский интерфейс

При запуске программы появляется окно, изображенное на рисунке 2, это и есть пустой калькулятор, готовый к работе.

Изображение выглядит как снимок экрана, текст, гаджет, Электронное устройство

Автоматически созданное описание

Рисунок 2. –– Калькулятор.

Калькулятор способен складывать, вычитать, умножать и делить, а так же есть возможность полного или частичного удаления введённых данных

# Вывод

В ходе учебной практики мною были освоены навыки подключения внешних модулей React (библиотек классов) и их использования в разрабатываемых программах на языке Java Script и HTML, а так же настройку внешнего вида программы с использованием CSS с помощью интегрированной среды разработки Visual Studio Code и блокноте Notepad++.