МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ОТЧЁТ**

**по практике № 1**

**«***Получение практических навыков работы с JS и выполнение простейших примитивов в рамках знакомства с языком»*

**по дисциплине: «Технологии и методы программирования»**

*4 вариант*

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБ-121», «АВТФ»  *Втюрин А.Р.*  «20» Мая 2023 г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  Ассистент кафедры ЗИ:  *Медведев Михаил*  *Александрович*  «\_\_\_» \_\_\_\_\_\_ 2023 г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2023г.

**Цель:** рассмотреть базовые конструкции на языке программирования js и получить первоначальные навыки работы с ним.

**Задача:** выполнить указанные ниже задания согласно варианту. Вариант выбирается согласно номеру в списке группы.

**Задание 1:** Функция находит в строке пары инвертированных фрагментов (например "123apr" и "rpa321") и возвращает номер строки на последний. С помощью функции найти все пары.

**Ход работы:**

1. Создание функции, отфильтровывающей лишние символы из строки.
2. Создание функции разбиения отфильтрованной строки на отдельные слова и запись их в массив.
3. Создание функции перебора и сравнения обычной и инвертированной строки для поиска инвертированных фрагментов.

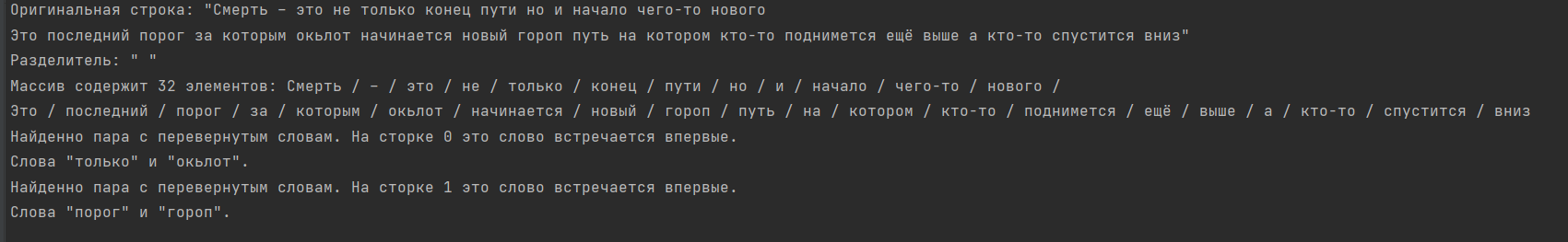


Рисунок 1 – Результат работы программы

**Задание 2:** Напишите мини-игру связав с событием на клик или отправки из формы текста. По необходимости можете за разъяснениями адресовать вопрос преподавателю. Игра «Гребешок»

**Ход работы:**

1. Создание набора переменных, хранящих состояние игры (см. приложение 2)
2. Создание функции выбора рандомных букв для составления слов (см. приложение 2)
3. Создание функции проверки существования слова по средствам API Yandex Dictionary API(см. приложение 2)
4. Создание функции проверки на повторяющиеся слова(см приложение 2)
5. Создание функции ведения счёта, добавления походящих под требования слов и функции проигрыша в игре (см. приложение 2)
6. Создание тела игры по средствам html и css(см. рис. )
7. Подключение скриптов к телу сайта игры

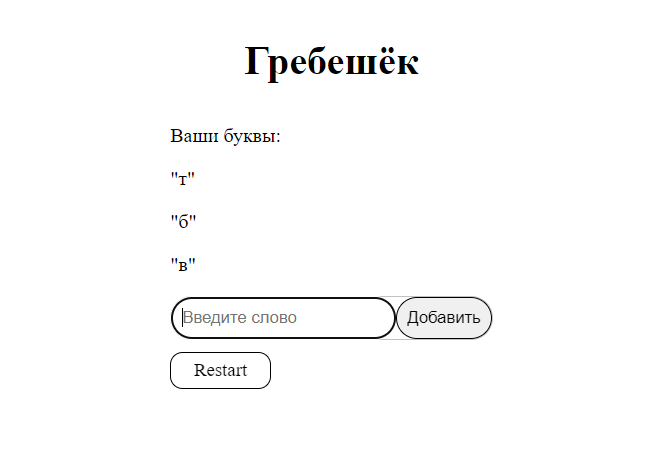


Рисунок 2 – Общий вид игры

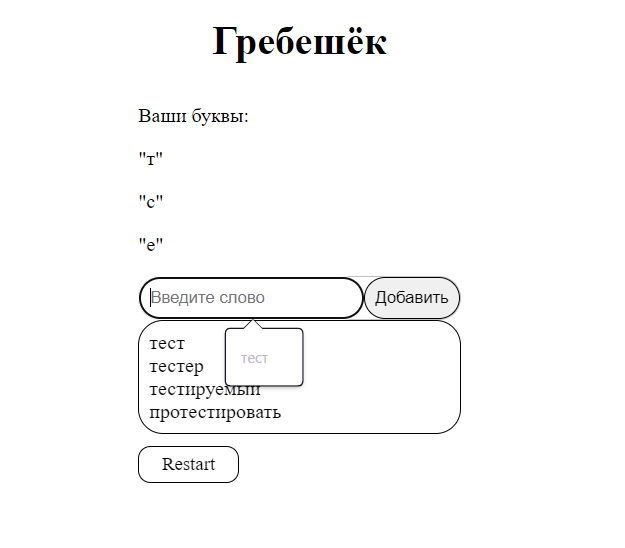


Рисунок 3 – Процесс игры

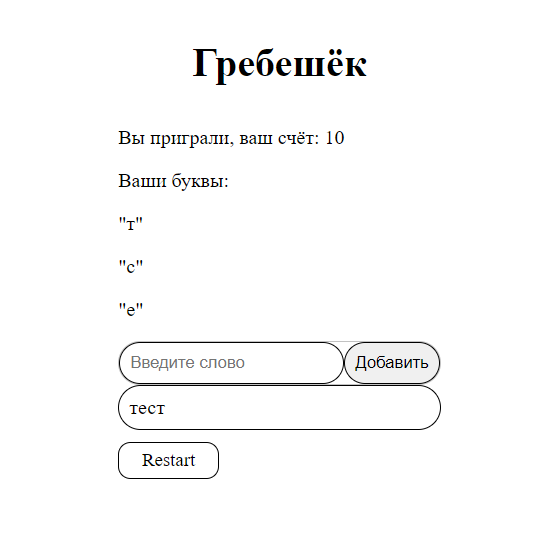


Рисунок 4 – Экран проигрыша

**Вывод**

ПРИЛОЖЕНИЕ 1

Листинг программы

index.js

'use strict';  
  
var ***originalStrings*** = `Смерть – это не только конец пути, но и начало чего-то нового.   
Это последний порог, за которым окьлот начинается новый гороп путь, на котором: кто-то поднимется ещё выше, а кто-то спустится вниз.`;// Оригинальная строка внутра которой выполняется поиск  
var ***lineWithoutPunctuation*** = ***originalStrings***.replace(/,/g, "").replace(/\./g, "")  
 .replace(/:/g, "").replace(/;/g, "").replace(/"/g, "")  
 .replace(/'/g, "").replace(/\(/g, "").replace(/\)/g, "");//Удаление лишних знаков из строк  
var ***arrayOfStrings*** = splitString(***lineWithoutPunctuation***, ' ')// Вызов функ разбиения и передача ей строки оригинала и символа разбиения  
  
var ***line*** = 0;  
function splitString(stringToSplit, separator) {// Разбиение строки на слова и запись их в массив  
 var arrayOfStrings = stringToSplit.split(separator);  
 ***console***.log('Оригинальная строка: "' + stringToSplit + '"');  
 ***console***.log('Разделитель: "' + separator + '"');  
 ***console***.log('Массив содержит ' + arrayOfStrings.length + ' элементов: ' + arrayOfStrings.join(' / '));// вывод преобразованных строк  
 return arrayOfStrings;// Возврат преобразованной строки  
}  
  
for (var ***i*** = 0; ***i*** < ***arrayOfStrings***.length - 1; ***i***++){// Цикл поиска зеркальных слов по средствам сравнения инвертированных слов в строке с обычной строкой  
  
 var ***word*** = ***arrayOfStrings***[***i***];  
 for (var ***p*** = 0; ***p*** < ***word***.length; ***p***++) if (***word***[***p***] == "\n") {// подсчёт строк  
 ***line***++;  
 }  
  
 for (var ***j*** = ***i*** + 1; ***j*** < ***arrayOfStrings***.length; ***j***++){// сравнение инверсированных и обычных элементов строки  
 var ***revWord*** = ***arrayOfStrings***[***j***].split('').reverse().join('');  
 if(***arrayOfStrings***[***i***] === ***revWord***){  
 ***console***.log('Найденно пара с перевернутым словам. На сторке ' + ***line*** + ' это слово встречается впервые.' + '\nСлова "' + ***arrayOfStrings***[***i***] + '" и "' + ***arrayOfStrings***[***j***] + '".');  
 }  
 }  
}

ПРИЛОЖЕНИЕ 2

Листинг программы

index.js

import React from 'react';  
import ReactDOM from 'react-dom';  
import App from './App';  
  
  
ReactDOM.***render***(  
<App/>,  
 ***document***.getElementById('root')  
);

App.js

import React from 'react'  
import Words from "./Words";  
function App() {  
  
 return (  
 <Words/>  
);  
}  
  
export default App;

Words.jsx

import React, {useState} from 'react';  
import './styles/app.css'  
  
const Words = () => {  
 const [words, setWords]=useState([])  
 const alphabet = ['а','б','в','г','д','е','ё','ж','з','и','й','к','л','м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ы','э','ю','я']  
 const [letters, setLetters] = useState(['т', 'с', 'е'])  
 const[word,setWord]=useState('')  
 const [isWordExist, setIsWordExist] = useState(false)  
 const [isLoading, setIsLoading] = useState(false)  
 const [game, setGame] = useState(true)  
 const [wordRepeats, setWordRepeats] = useState(false)  
  
 const [checkWord, setCheckWord] = useState(false)  
 const [score, setScore] = useState(0)  
  
 const clearWords=()=>{  
 setWords([])  
 setWord('')  
 setLetters([])  
 }  
  
 function getRandomInt(max){  
 return ***Math***.floor(***Math***.random() \* max);  
 }  
  
 const checkLetters = () =>{  
 const wordLetters = word.toLowerCase().split('')  
 const areLettersExist = letters.every((letter) => wordLetters.includes(letter))  
 checkRepeat()  
 setCheckWord(areLettersExist)  
 }  
  
 const checkRepeat = () =>{  
 const isExist = words.includes(word.toLowerCase());  
 setWordRepeats(isExist);  
 }  
  
 const addList = () =>{  
 if (word !== ''){  
 setWords([...words, word.toLowerCase()])  
 setWord('')  
 setScore(score + 10)  
 }  
 }  
  
 const formSubmit = async (event) => {  
 event.preventDefault()  
  
 if(checkWord && game && !wordRepeats){  
 setIsLoading(true)  
  
 try {  
 const response = await fetch(`https://dictionary.yandex.net/api/v1/dicservice.json/lookup?key=dict.1.1.20230516T152258Z.7ca86a7925d50b31.103d10cc5c380c6417a91e7dce6016267c31ea11&lang=ru-ru&text=${word}`)  
  
 const data = await response.json()  
 setIsWordExist(data.def.length > 0)  
 } catch (error) {  
 ***console***.log(error)  
 }  
  
 setIsLoading(false)  
 if(isWordExist) {  
 addList()  
 setScore(score + 10)  
 setIsWordExist(false)  
 }  
  
 }  
 else if(!checkWord || wordRepeats || (!isWordExist && !isLoading)){  
 setGame(false)  
 setWord('')  
  
 }  
 }  
  
 const restart = () => {  
 clearWords()  
 const letter = []  
 for(let i = 0; i < 3; i++)  
 {  
 letter[i] = alphabet[getRandomInt(31)]  
 }  
 setLetters([ letter[0], letter[1], letter[2]])  
 setGame(true)  
 setScore(0)  
 }  
  
  
 return (  
 <div className="App">  
 <h1 className="Name">Гребешёк</h1>  
 {!game && <p>Вы приграли, ваш счёт: {score}</p>}  
  
  
 {  
 letters.length ? (  
 <div>  
 Ваши буквы:  
 {  
 letters.map((el, id) => <p key={id}>"{el}"  
  
 </p>)  
 }  
  
 </div>  
 ) : ''  
  
 }  
  
  
  
 <form onSubmit={formSubmit} className="Inputbox">  
 <input value ={word} className="input" onChange={e=> setWord(e.target.value)} type="text" id="txt" placeholder="Введите слово" required/>  
 <button onClick={checkLetters} type="submit" className="submit">Добавить</button>  
 </form>  
  
  
  
 {  
 words.length ? (  
 <div>  
 <div className="Words">  
 {  
 words.map((el, id) => <div key={id}>{el}  
  
 </div>)  
 }  
 </div>  
  
 </div>  
 ) : ''  
  
 }  
 <div onClick={restart} className="clearList" id="circ">Restart</div>  
  
  
  
 </div>  
 )  
}  
  
export default Words;

app.css

тзь  
\* {  
 margin: 3px;  
 padding: 2px;  
 box-sizing: border-box;  
 background-color: #F5F5F5;  
}  
  
.Name{  
 padding:10px;  
 display: flex;  
 justify-content: space-around;  
 text-align: center;  
}  
  
#root{  
 display: flex;  
 justify-content: center;  
}  
  
.Inputbox{  
 display: flex;  
 width: auto;  
 border-radius: 20px;  
 border: 1px solid #c2c2c2;  
}  
.input, .Words{  
 border-radius: 20px;  
 border: 1px solid black;  
 padding: 8px;  
  
}  
.submit {  
 border-radius: 20px;  
 border: 1px solid black;  
 padding: 8px;  
}  
.clearList {  
 margin-top: 10px;  
 font-size: 15px;  
 text-align: center;  
 border-radius: 10px;  
 border: 1px solid black;  
 padding: 5px;  
 width: 69px;  
}