Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №2 по курсу «Архитектура ЭВМ»

Тема Работа с JSON и знакомство с Express

Студент Романов А.В.

Группа ИУ7-53Б

Преподаватели Попов А.Ю.

Москва — 2020 г.

Отчет по разделу #3

Цель работы

Ознакомиться с особенностями работы с форматом **JSON**.

Задание 1

Условие

С клавиатуры считывается число N. Далее считывается N строк. Необходимо создать массив и сохранять в него строки только с четной длинной. Получившийся массив необходимо преобразовать в строку JSON и сохранить в файл.

Код программы

Язык: **Javascript**

ex_03_01.js

```
function task1() {
    console.log("===== TASK1 =====");

    const fname = path.join(__dirname, "task1.txt")
    const n = readline.questionInt("Enter N: ");

    let storage = Array.from(Array(n).keys()).reduce((acc, i) => {
        const str = readline.question(`Enter string №${i + 1}: `);
        return str.length % 2 == 0 ? acc.concat([str]) : acc;
    }, []);

    fs.writeFileSync(fname, JSON.stringify(storage, null, 4));
}
```

Результаты тестирования

Все тесты были пройдены успешно.

Задание 2

Условие

Необходимо считать содержимое файла, в котором хранится массив строк в формате JSON. Нужно вывести только те строки на экран, в которых содержатся только гласные буквы.

Код программы

Язык: **Javascript**

ex_03_02.js

```
function task2() {
    console.log("===== TASK2 =====");

    const fname = path.join(__dirname, "task1.txt")
    const strings = JSON.parse(fs.readFileSync(fname)).filter(str =>
        !str.toUpperCase().split("").reduce((acc, char) =>
            VOWELS.includes(char) ? acc + false : acc + true, false)
    );

    strings.map(str => console.log(str))
}
```

Результаты тестирования

Все тесты были пройдены успешно.

Задание 3

Условие

С клавиатуры считывается строка - название расширения файлов. Далее считывается строка - адрес папки. Необходимо перебрать все файлы в папке и вывести содержимое файлов, у которых расширение совпадает с введенным расширением.

Код программы

Язык: **Javascript**

ex_03_03.ts

```
function task3() {
    console.log("===== TASK3 =====");

    const extension = readline.question("Enter files extension: ");
    const dir = readline.question("Enter folder: ");

    fs.readdirSync(dir).map(
        filename => path.join(dir, filename)).map(
            file => { if (file.endsWith(extension)) {
                console.log(`File: ${file}\nContents:\n${fs.readFileSync(file)}` );
            }
            }
        )
}
```

Результаты тестирования

Все тесты были пройдены успешно.

Задание 4

Условие

Дана вложенная структура файлов и папок. Все файлы имеют раширение "txt". Необходимо рекурсивно перебрать вложенную структуру и вывести имена файлов, у которых содержимое не превышает по длине 10 символов.

Код программы

Язык: **Javascript**

ex_03_04.ts

```
function walk_and_find(fname, correct) {
    if (fs.lstatSync(fname).isDirectory()) {
        fs.readdirSync(fname).map(f => walk_and_find(path.join(fname, f), correct));
    } else if (fs.readFileSync(fname).length <= 10) {
        correct.push(fname);
    }
}

function task4() {
    console.log("===== TASK4 =====");

    const dir = readline.question("Enter path to folder: ");
    const names = [];

    walk_and_find(dir, names);
    names.map(name => console.log(name));
}
```

Результаты тестирования

Все тесты были пройдены успешно.

Задание 5

Условие

С клавиатуры считывается число N. Далее считывается N строк - имена текстовых файлов. Необходимо склеить всё содержимое введенных файлов в одну большую строку и сохранить в новый файл.

Код программы

Язык: **Javascript**

ex_03_05.ts

```
function task5() {
    console.log("===== TASK5 =====");

    const n = readline.questionInt("Enter number of strings: ");
    const fname = path.join(__dirname, "task5.txt")
    const merged = Array.from(Array(n).keys()).reduce((acc, i) =>
        acc + fs.readFileSync(readline.question(`Enter string №${i + 1}: `)), "")

    fs.writeFileSync(fname, merged);
}
```

Результаты тестирования

Все тесты были пройдены успешно.

Задание 6

Условие

Написать код, который позволяет определить максимальный возможный уровень вложенности друг в друга полей в объекте, чтобы данный объект можно было преобразовать в строку формата JSON. Ответом является целое число.

Код программы

Язык: **Javascript**

ex_03_06.ts

```
function max_deep(obj) {
    obj.cnt++;
    obj.acc = { acc: obj.acc };

    try {
        JSON.stringify(obj.acc)
    } catch (_) {
        console.log(obj.cnt)
        return;
    }

    max_deep(obj);
}

function task6() {
    console.log("===== TASK6 =====");

    max_deep( { cnt: 0, acc: 1 } );
}
```

Результаты тестирования

Все тесты были пройдены успешно.

Задание 7

Условие

Из файла считывается строка в формате JSON. В этой строке информация об объекте, в котором находится большое количество вложенных друг в друга полей. Объект представляет из себя дерево. Необходимо рекурсивно обработать дерево и найти максимальную вложенность в дереве. Необходимо вывести на экран ветку с максимальной вложенностью.

Код программы

Язык: **Javascript**

ex_03_07.ts

```
function max_branch(obj, saver) {
    if (!("object" === typeof(obj))) {
        if (saver.current_depth > saver.max_depth) {
            saver.max_branch = saver.current_branch.filter(t => true);
            saver.current_depth = saver.max_depth;
        }
    } else {
        ++saver.current_depth;

        Object.keys(obj).map(branch => {
            saver.current_branch.push(branch);
            max_branch(obj[branch], saver);
            saver.current_branch.pop();
        })

        --saver.current_depth;
    }
}

function task7() {
    console.log("===== TASK7 =====");

    const fname = readline.question("Enter filename: ");
    const obj = JSON.parse(fs.readFileSync(fname));
    const saver = {
        current_depth: 0,
        current_branch: ["/"],
        max_depth: 0,
        max_branch: []
    }

    max_branch(obj, saver);

    console.log("Max branch:", saver.max_branch);
}
```

Результаты тестирования

Все тесты были пройдены успешно.

Вывод

В результате работы были изучены особенности работы с форматом **JSON**, изучены основные методы и функции работы с данным форматом.

Отчет по разделу #4

Цель работы

Ознакомиться с фреймворком серверной разработки **Express.**

Задания 1-4

- Запустить сервер. Реализовать на сервере функцию для сравнения трёх чисел и выдачи наибольшего из них. Реализовать страницу с формой ввода для отправки запроса на сервер.
- Запустить сервер. На стороне сервера должен храниться файл, внутри которого находится JSON строка. В этой JSON строке хранится информация о массиве объектов. Реализовать на сервере функцию, которая принимает индекс и выдает содержимое ячейки массива по данному индексу. Реализовать страницу с формой ввода для отправки запроса на сервер.
- Написать программу, которая на вход получает массив названий полей и адрес запроса (куда отправлять). Программа должна генерировать HTML разметку страницы, в которую встроена форма для отправки запроса.
- Запустить сервер. Реализовать на сервере функцию, которая принимает на вход числа A, B и C. Функция должна выдавать массив целых чисел на отрезке от A до B, которые делятся на C нацело.

Код программы

Язык: **Javascript**

src/app.js

```
"use strict";

const express = require("express");
const path = require("path");
const body_parser = require("body-parser");

const post = require("./controllers").post;
const router = require("./router");

const port = 5015;
const index_page = path.join(__dirname, "..", "public", "index.html")

const app = express();

app.use(body_parser.urlencoded({ extended: false }));
app.use((req, _, next) => {
    console.log(req.url);

    if (req.method === "POST") {
        console.log(req.body);
    }
    else if (req.method === "GET") {
        console.log(req.query);
    }

    next();
})

app.use(express.static(path.join(__dirname, "..", "public")));
app.use("/tasks", router);

app.use((req, res, next) => {
    if (req.method === "POST") {
        // обработка всех подряд POST-запросов
        res.send(`Address: ${req.url}, \nBody of POST: ${JSON.stringify(req.body)}`);
    } else {
        next();
    }
});

app.use((req, res) => {
    res.sendFile(index_page);
});

app.listen(port);

console.log(`Server running. Port: ${port}`);
```

src/controllers.js

```
const path = require("path");
```

```
const path = require("path");
const fs = require("fs");

module.exports.max3 = (req, res) => {
    let fst = parseInt(req.body.fst.toString());
    let snd = parseInt(req.body.snd.toString());
    let thd = parseInt(req.body.thd.toString());

    if (!fst || !snd || !thd) {
        res.status(400).send("Bad numbers, try again.");
    }

    res.status(200).send(`Result: ${Math.max(fst, snd, thd)}`);
}

module.exports.get_by_index = (req, res) => {
    let index = parseInt(req.body.index.toString());

    if (!index || index < 0 || index > array.length) {
        res.status(400).send("Invalid index.");
    }

    const array = JSON.parse(fs.readFileSync(path.join(__dirname, "..", "data", "array.json")));

    res.status(200).send(JSON.stringify(array[index]));
}

module.exports.find_mod = (req, res) => {
    let a = parseInt(req.body.a.toString());
    let b = parseInt(req.body.b.toString());
    let c = parseInt(req.body.c.toString());

    if (!a || !b || !c) {
        res.status(400).send("Invalid input.");
    }

    if (b <= a) {
        res.status(400).send("B must be greate than A.");
    }

    if (c <= 0) {
        res.status(400).send("C must be greater than 0.");
    }

    const numbers = Array.from(Array(b + 1).keys()).filter(x => x >= a && x % c === 0);

    res.status(200).send(JSON.stringify(numbers));
}

function gen_header(address) {
    return `<form method="POST" action="${address}">`;
}

function gen_form(fields) {
    let content = fields.reduce((acc, field) => acc + `  <label for="${field}">Enter ${field}:</label>
        <input type="text" name="${field}">
        <br>`, "")

    content += `<input type="submit"></form>`;

    return content;
}

function gen_page(fields, address) {
    content = `<!DOCTYPE html>
    <html lang="en">
```

```
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>NodeLab04</title>
    </head>
    <body>` + gen_header(address) + gen_form(fields) + `</body></html>`

    return content;
}

module.exports.generate = (req, res) => {
    let fields = req.body.fields;
    let address = req.body.address;

    if (!fields || !address) {
        res.status(400).send("Invalid input.");
    }

    if (address.charAt(0) !== '/') {
        address = '/' + address;
    }

    res.send(gen_page(fields.split(' '), address));
}
```

src/router.js

```
const router = require("express").Router();
const controllers = require("./controllers");

router.post("/max3", controllers.max3)
router.post("/index", controllers.get_by_index)
router.post("/find_mod", controllers.find_mod)
router.post("/generate", controllers.generate)

module.exports = router;
```

public/index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>NodeLab04</title>
    </head>
    <style>

        form,h1{background-color:white;
                color:#42A4D3;
                padding:15px}
        form{margin-bottom:100px}

    </style>
    <!--<body style="background-image:url(bg.jpg)">-->
    <body>
    <div style="display: flex">
        <div>
            <img src="bg.jpg" alt="bg">
            <h1>TASK1: Максимум из трёх.</h1>
            <form method="POST" action="/tasks/max3">
                <label for="a">Введите первое число:</label>
                <input type="text" name="fst">
                <br>
                <label for="b">Введите второе число:</label>
```

```
                <input type="text" name="snd">
                <br>
                <label for="c">Введите третье число:</label>
                <input type="text" name="thd">
                <br>
                <input type="submit", value="B3JIOMATb", style="font-weight: bold; background-color:black; color:white; padding: 10p
            </form>
        </div>
        <img src="bg.jpg" alt="bg">
    </div>


        <h1>TASK2: Достать элемент из массива по индексу.</h1>
        <form method="POST" action="/tasks/index">
            <label for="index">Введите индекс массива:</label>
            <input type="text" name="index">
            <br>
            <input type="submit", value="B3JIOMATb", style="font-weight: bold; background-color:black; color:white; padding: 10px">
        </form>


        <h1>TASK3: Генерация форм.</h1>
        <form method="POST" action="/tasks/generate">
            <label for="fields">Введите поля:</label>
            <input type="text" name="fields">
            <br>
            <label for="address">Введите URL:</label>
            <input type="text" name="address">
            <br>
            <input type="submit", value="B3JIOMATb", style="font-weight: bold; background-color:black; color:white; padding: 10px">
        </form>

        <h1>TASK 4: Все числа от A до B, которые делятся на C.</h1>
        <form method="POST" action="/tasks/find_mod">
            <label for="A">Введите A:</label>
            <input type="text" name="a">
            <br>
            <label for="B">Введите B:</label>
            <input type="text" name="b">
            <br>
            <label for="C">Введите C:</label>
            <input type="text" name="c">
            <br>
            <input type="submit", value="B3JIOMATb", style="font-weight: bold; background-color:black; color:white; padding: 10px">
        </form>
    </body>
</html>
```

Результаты тестирования

Все тесты были пройдены успешно.

Вывод

В результате работы был разработан сервер на основе **Express**, изучены основные достоинства и недостатки данного фреймворка.