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**

tasks.js

```
function t1() {
    const filename = readline.question("Enter name of file to write data: ");
    const N = readline.questionInt("Enter number of strings: ");

    const strings = [];

    for (let i = 0; i < N; i++) {
        const str = readline.question(`Enter ${i + 1} string: `);
        if (str.length % 2 === 0)
            strings.push(str);
    }

    fs.writeFileSync(filename, JSON.stringify(strings, null, 4));
}
```

Результаты тестирования

Все тесты 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**

tasks.js

```
function only_vowels(string) {
    for (const char of string)
        if (!VOWELS.includes(char.toLowerCase()))
            return false;

    return true;
}

function t2() {
    const filename = readline.question("Enter name of file with data: ");
    const file_content = fs.readFileSync(filename);
    const strings = JSON.parse(file_content);

    for (const string of strings)
        if (only_vowels(string))
            console.log(string);
}
```

Результаты тестирования

Все тесты 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**

tasks.ts

```
function printFile(filename) {
    console.log(`File: ${filename}\nContent:\n${fs.readFileSync(filename)}`);
}

function t3() {
    let extension = readline.question("Enter extension of files: ");
    if (!extension.startsWith(".")) {
        extension = "." + extension;
    }

    const folderpath = readline.question("Enter path to folder: ");
    let filenames = fs.readdirSync(folderpath);
    //filenames = filenames.map(filename => path.join(folderpath, filename));

    for (const filename of filenames)
        if (filename.endsWith(extension))
            printFile(path.join(folderpath, filename));
}
```

Результаты тестирования

Все тесты 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**

tasks.ts

```
function is_good_file(file) {
    return file.endsWith(T4_EXTENSION) && fs.readFileSync(file).length <= 10;
}

function recursive_walk_file(file, names) {
    if (is_good_file(file))
        names.push(file);
    else if (fs.lstatSync(file).isDirectory()) {
        const files = fs.readdirSync(file);
        for (const f of files)
            recursive_walk_file(path.join(file, f), names);
    }
}

function t4() {
    const file = readline.question("Enter path to folder/file: ");
    const names = [];

    recursive_walk_file(file, names);

    for (const name of names)
        console.log(name);
}
```

Результаты тестирования

Все тесты 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**

tasks.ts

```
function t5() {
    const N = readline.questionInt("Enter number of files: ");
    const filename = readline.question("Enter name of file to write data: ");

    let content = "";
    for (let i = 0; i < N; i++) {
        const tmp_filename = readline.question(`Enter ${i + 1} file: `);
        content += fs.readFileSync(tmp_filename);
    }

    fs.writeFileSync(filename, content);
}
```

Результаты тестирования

Все тесты 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**

tasks.ts

```
function t6() {
    let a = 1;
    let cnt = 0;
    try {
        while (JSON.stringify(a)) {
            cnt++;
            a = { a };
        }
    } catch(_) {
        console.log(cnt);
    }
}
```

Результаты тестирования

Все тесты 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**

tasks.ts

```
function get_max_branch(object, metadata) {
    if (!check_iterable(object)) {
        if (metadata.depth > metadata.max_depth) {
            metadata.max_depth = metadata.depth;
            metadata.max_branch = [...metadata.cur_branch];
        }
    } else {
        metadata.depth++;

        for (const field in object) {
            metadata.cur_branch.push(field);
            get_max_branch(object[field], metadata);
            metadata.cur_branch.pop();
        }

        metadata.depth--;
    }
}

function t7() {
    const filename = readline.question("Enter name of file with complicated object: ");
    const content = fs.readFileSync(filename);
    const object = JSON.parse(content);

    const metadata = {
        depth: 0,
        max_depth: 0,
        cur_branch: ['root'],
        max_branch: []
    };

    get_max_branch(object, metadata);
    console.log("Object:\n", JSON.stringify(object, null, 4), "\nMax branch:", metadata.max_branch);
}
```

Результаты тестирования

Все тесты 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

```
const express = require("express");
const path = require("path");
const body_parser = require("body-parser");
const default_post = require("./controllers").default_post;

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

const port = process.env.PORT | 3000;
const PAGES_FOLDER = "public";
const MAIN_PAGE = "index.html";
const PATH_TO_MAIN_PAGE = path.join(__dirname, '..', PAGES_FOLDER, MAIN_PAGE);

const app = express();

app.use(express.static(path.join(__dirname, "..", PAGES_FOLDER)));
app.use(body_parser.urlencoded({ extended: false }));
app.use("/task", router);

app.use("/", (req, res) => {
    if (req.method === "GET")
        res.sendFile(PATH_TO_MAIN_PAGE);
    else
        default_post(req, res);
});

app.listen(port);
console.log(`Running on port ${port}`);
```

src/controllers.js

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

const FILE_WITH_ARRAY = path.join(__dirname, "..", "data", "array.json");

module.exports.find_min = (req, res) => {
    // for get request we use req.query.<param_name>!
    let a = req.body.a;
    let b = req.body.b;
    let c = req.body.c;

    if (isNaN(a) || isNaN(b) || isNaN(c))
        res.status(400).send("Bad input!");

    a = parseFloat(a);
    b = parseFloat(b);
    c = parseFloat(c);

    res.status(200).send(Math.max(a, b, c).toString());
}

module.exports.get_array_elem = (req, res) => {
    let index;
    try {
        index = parseInt(req.body.index.toString().trim());
        if (index.toString().length !== req.body.index.trim().length)
            throw Error("");
        index -= 1;
    } catch(_) {
        res.status(400).send("Bad input! Can't convert index to number!");
        return;
    }
```

```
    const array = JSON.parse(fs.readFileSync(FILE_WITH_ARRAY));

    if (index < 0 || index >= array.length)
        res.status(400).send("Bad request, out of range");
    else
        res.status(200).send(JSON.stringify(array[index]));
};

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

    try {
        a = parseInt(a);
        b = parseInt(b);
        c = parseInt(c);
        if (c <= 0)
            throw Error("");
    } catch(_) {
        res.status(400).send("Bad input! Can't convert args to number!");
        return;
    }

    const all_numbers = [];
    const start = a % c === 0 ? a : a + (c - a % c);
    for (let num = start; num <= b; num += c)
        all_numbers.push(num);

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

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

function generate_form(fields) {
    let content = "";

    for (const field of fields)
        content += `  <label for="${field}">Enter ${field}:</label>
<input type="text" name="${field}">
<br>`;
    content += `<input type="submit">`;
    content += "</form>";

    return content;
}

function generate_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>Lab 04</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>`;
    content += generate_form_header(address);
    content += generate_form(fields);
    content += "</body></html>";
    return content;
}
```

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

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

    const html_code = generate_page(fields.split(' '), address);
    res.send(html_code);
};

module.exports.default_post = (req, res) => {
    const body = JSON.stringify(req.body);
    res.status(200).send(`Address: ${req.url};\nBody: ${body}`);
}
```

src/router.js

```
 const Router = require("express").Router;

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

const router = Router();

router.post("/calc", controllers.find_min);
router.post("/array", controllers.get_array_elem);
router.post("/findall", controllers.find_all);
router.post("/genform", controllers.form_generator);

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>Lab 04</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Task 1. Find max from 3 values.</h1>
    <form method="POST" action="/task/calc">
        <label for="a">Enter a:</label>
        <input type="text" name="a">
        <br>
        <label for="b">Enter b:</label>
        <input type="text" name="b">
        <br>
        <label for="c">Enter c:</label>
        <input type="text" name="c">
        <br>
        <input type="submit">
    </form>

    <h1>Task 2. Get element from array of objects.</h1>
    <form method="POST" action="/task/array">
        <label for="index">Enter index:</label>
        <input type="text" name="index">
        <br>
        <input type="submit">
    </form>

    <h1>Task 3. Generate form.</h1>
    <form method="POST" action="/task/genform">
        <label for="fields">Enter fields, separating by spaces:</label>
        <input type="text" name="fields">
        <br>
        <label for="address">Enter URL address:</label>
        <input type="text" name="address">
        <br>
        <input type="submit">
    </form>



    <h1>Task 4. Find all numbers from A to B whose mod(C) = 0.</h1>
    <form method="POST" action="/task/findall">
        <label for="A">Enter A:</label>
        <input type="text" name="a">
        <br>
        <label for="B">Enter B:</label>
        <input type="text" name="b">
        <br>
        <label for="C">Enter C:</label>
        <input type="text" name="c">
        <br>
        <input type="submit">
    </form>

</body>
</html>
```

```
html {
    margin: 0;
    padding: 0;
}

body {
    padding-top: 2em !important;
    margin: 0;
    padding: 0;
    background-color: #060818;
    color: #fa00ff !important;
}

label {
    font-size: 1.5em;
    font-weight: bold;
}

input {
    justify-self: center;
    background-color: #88ff88;
    color: #060818;
    padding: 0.3em;
    font-size: 1em;
    margin-bottom: 0.3em;
    border-radius: 0.2em;
}

input[type=submit] {
    cursor: pointer;
    margin-top: 1em;
    background: #00ff00;
    border-radius: 2em;
    padding: 0.5em 1.5em;
    margin-bottom: 3em;
}

h1 {
    margin-top: 2em;
    margin-bottom: 1em;
    width: 70%;
    margin: 0 auto;
    text-align: center;
}

form {
    text-align: center;
    margin-top: 1em;
}
```

Результаты тестирования

Все тесты пройдены успешно.

Вывод

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