|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

**Лабораторная работа № 2**

|  |  |
| --- | --- |
| **Студент**  Гадоев А. А.  **Группа** ИУ7-53Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель Попов А. Ю.** |  |

Москва.

2020 г.

**Цель работы:** получить навыки работы c файлом зависимостей, JSON, получением объекта из строки JSON. Научиться работать с файлами, считывать с клавиатуры. Работа с серверной частью при помощи express, генерация HTML страниц, формы.

Ссылка на github: https://github.com/mavennn/nodejs

**Task2**

### Задание 1

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**

1. "use strict";
3. **const** readLine = require('readline');
4. **const** fs = require('fs');
6. **const** DATA\_DIRECTORY = "/Users/gadoevalex/Desktop/evm/lab2/task3/data/";
8. **const** rl = readLine.createInterface({
9. input: process.stdin,
10. output: process.stdout
11. });
13. **const** readLinePromise = (text) => {
14. **return** **new** Promise((resolve, reject) => {
15. rl.question(text, (result) => {
16. **if** (!result) reject();
17. resolve(result);
18. })
19. });
20. }
22. **const** writeToFile = (fileName, data) => {
23. **return** **new** Promise((resolve, reject) => {
24. fs.writeFile(fileName, data, (err) => {
25. **if** (err) reject(err);
27. resolve();
28. })
29. })
30. }

33. (async **function** main() {
35. **var** rowsAmount = **null**;
36. **var** resultArray = [];
38. *// считываем число - количество строк*
39. while (!rowsAmount) {
40. **try** {
41. let result = await readLinePromise('Enter number of lines: ');
43. **if** (!isNaN(result)) {
44. rowsAmount = result;
45. }
46. } **catch** (e) {
48. }
49. }
51. **if** (rowsAmount && !isNaN(rowsAmount)) {
53. **for**(let i = 0; i < rowsAmount; i++) {
54. let data = await readLinePromise(`Enter ${i + 1}'s string: `);
56. if (typeof(data) == 'string' && data.length % 2 == 0) {
57. resultArray.push(data);
58. }
59. }
60. }
62. // записать в файл
63. if (Array.isArray(resultArray)) {
64. console.log(JSON.stringify(resultArray));
65. writeToFile(DATA\_DIRECTORY + '1\_result.txt', JSON.stringify(resultArray))
66. .then(() => {
67. process.exit(1);
68. })
69. .catch(err => console.log(err))
70. }
71. })();

**Задание 2**

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**

1. "use strict";
3. **const** fs = require('fs');
5. **const** DATA\_DIRECTORY = "/Users/gadoevalex/Desktop/evm/lab2/task3/data/"
7. **const** readFilePromise = (fileName) => {
8. **return** **new** Promise((resolve, reject) => {
9. fs.readFile(fileName, 'utf-8', (err, data) => {
10. **if** (err) reject(err);
12. resolve(data);
13. })
14. })
15. }
17. **const** writeToFile = (fileName, data) => {
18. **return** **new** Promise((resolve, reject) => {
19. fs.writeFile(fileName, data, (err) => {
20. **if** (err) reject(err);
22. resolve();
23. })
24. })
25. }
27. */\*\**
28. *\* возвращает true если символ - гласная*
29. *\* @param {\*} char*
30. *\*/*
31. **function** isVowelRegEx(char) {
32. **if** (char.length == 1) {
33. **return** */[aeiou]/*.test(char);
34. }
35. }
37. **const** firstObj = {
38. alex: "darkStralker"
39. };
41. **const** secondObj = {
42. aaa: "aaaaa"
43. };
45. **const** thirdObj = {
46. "sdfsdf": "1ffsdf"
47. };
49. **const** isAllVowels = (string) => {
50. **for**(let s of string) {
51. **if** (!isVowelRegEx(s)) **return** **false**;
52. }
53. **return** **true**;
54. }
56. (async **function** main() {
58. let str = JSON.stringify({ array: [firstObj, secondObj, thirdObj] })
60. *// записываем в файл*
61. **try** {
62. await writeToFile(DATA\_DIRECTORY + 'second.txt', str)
63. } **catch** (err) {
64. console.log(err);
65. }
67. **try** {
69. let dataString = await readFilePromise(DATA\_DIRECTORY + '2\_result.txt');
71. **if** (!dataString || dataString == ''){
72. **throw** **new** Error('file is empty');
73. }
75. let data = JSON.parse(dataString);

78. **if** (data.hasOwnProperty("array")) {
79. let array = data.array;
81. array.map(obj => {
82. **try** {
83. **for** (let prop **in** obj) {
84. **if** (isAllVowels(prop.toString()) && isAllVowels(obj[prop].toString())) {
85. console.log(JSON.stringify(obj));
86. }
87. }
88. } **catch** (e) {
89. console.log(e);
90. }
91. })
93. }
95. } **catch** (err) {
96. console.log(err);
97. }
98. })()

### Задание 3

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**

1. "use strict";
3. **const** fs = require('fs');
4. **const** readLine = require('readline');
6. **const** rl = readLine.createInterface({
7. input: process.stdin,
8. output: process.stdout
9. });
11. **const** readLinePromise = (text) => {
12. **return** **new** Promise((resolve, reject) => {
13. rl.question(text, (result) => {
14. **if** (!result) reject();
15. resolve(result);
16. });
17. });
18. }
20. **const** readFilePromise = (fileName) => {
21. **return** **new** Promise((resolve, reject) => {
22. fs.readFile(fileName, "utf-8", (err, data) => {
23. **if** (err) reject(err);
24. resolve(data);
25. });
26. });
27. }
29. **const** readdirPromise = (path) => {
30. **return** **new** Promise((resolve, reject) => {
31. fs.readdir(path, (err, data) => {
32. **if** (err) reject(err);
33. resolve(data);
34. });
35. });
36. }
38. (async **function** main() {
40. **try** {
41. **const** ext = await readLinePromise('Enter file extension: ');
43. **const** path = await readLinePromise('Enter file path: ');
45. let files = await readdirPromise(path);
47. **for** (let file of files) {
48. let parts = file.split('.');
49. **if** (parts[parts.length - 1] == ext) {
50. readFilePromise(path + "/" + file)
51. .then(fileText => console.log(fileText))
52. .**catch**(err => console.log(err))
53. }
54. }
56. } **catch** (e) {
57. console.log(e);
58. }

61. })()

### Задание 4

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**

1. "use strict";
3. **const** fs = require('fs');
4. **const** path = require('path');
6. **const** directory = "/Users/gadoevalex/Desktop/evm/lab2/";
8. **const** MAX\_LENGTH = 10;
10. **var** walk = **function**(dir, done) {
11. **var** results = [];
13. fs.readdir(dir, **function**(err, list) {
14. **if** (err) **return** done(err);
16. **var** pending = list.length;
18. **if** (!pending) **return** done(**null**, results);
20. list.forEach(**function**(file) {
21. file = path.resolve(dir, file);
22. fs.stat(file, **function**(err, stat) {
23. **if** (stat && stat.isDirectory()) {
24. walk(file, **function**(err, res) {
25. results = results.concat(res);
26. **if** (!--pending) done(**null**, results);
27. });
28. } **else** {
29. fs.readFile(file, "utf-8", (err, text) => {
30. **if** (text.length <= MAX\_LENGTH) {
31. console.log(text);
32. }
33. })
35. results.push(file);
36. **if** (!--pending) done(**null**, results);
37. }
38. });
39. });
40. });
41. };
43. (async **function** main() {
45. walk(directory, (err, results) => {
46. **if** (err) console.log(err);
47. })
49. })()

### Задание 5

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**

1. "use strict";
3. **const** readLine = require('readline');
4. **const** fs = require('fs');
6. **const** DATA\_DIRECTORY = "/Users/gadoevalex/Desktop/evm/lab2/task3/data/";
8. **const** rl = readLine.createInterface({
9. input: process.stdin,
10. output: process.stdout
11. });

14. **const** readLinePromise = (text) => {
15. **return** **new** Promise((resolve, reject) => {
16. rl.question(text, (result) => {
17. **if** (!result) reject();
18. resolve(result);
19. })
20. });
21. }
23. **const** writeToFile = (fileName, data) => {
24. **return** **new** Promise((resolve, reject) => {
25. fs.writeFile(fileName, data, (err) => {
26. **if** (err) reject(err);
28. resolve();
29. })
30. })
31. }
33. **const** readFilePromise = (fileName) => {
34. **return** **new** Promise((resolve, reject) => {
35. fs.readFile(fileName, "utf-8", (err, data) => {
36. **if** (err) reject(err);
37. resolve(data);
38. })
39. })
40. }
42. (async **function** main() {
44. **var** filesAmount = **null**;
45. **var** result = '';
47. *// считываем число - количество файлов*
48. while (!filesAmount) {
49. **try** {
50. let result = await readLinePromise('Enter number of files: ');
52. **if** (!isNaN(result)) {
53. filesAmount = result;
54. }
56. } **catch** (e) {
57. console.log(e);
58. }
59. }

62. */\**
63. */Users/gadoevalex/Desktop/study/nodejs/task3/result.txt*
64. */Users/gadoevalex/Desktop/study/nodejs/task3/five.js*
65. */Users/gadoevalex/Desktop/study/nodejs/task3/second.txt*
66. *\*/*
68. **if** (filesAmount&& !isNaN(filesAmount)) {
69. **for**(let i = 0; i < filesAmount; i++) {
70. let fileName = await readLinePromise(`Enter ${i + 1}'s file: `);
72. // считываем содержимое файла и конкатенируем
73. result += await readFilePromise(DATA\_DIRECTORY + fileName);
74. }
75. }
77. // записать в файл
78. writeToFile(DATA\_DIRECTORY + '5\_result.txt', result)
79. .then(() => {
80. process.exit(1);
81. })
82. .catch(err => console.log(err))

85. })();

### Задание 6

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**

1. "use strict"
3. **var** count = 1;
4. **var** obj = { prop: 'prop' };
5. **var** string = '';
7. while (**true**) {
8. **try** {
9. string = JSON.stringify(obj);
10. obj = { 'prop': { ...obj } };
11. count++;
12. } **catch** (e) {
13. console.log(e);
14. console.log(`Result is ${count}`);
15. **break**;
16. }
17. }

### Задание 7

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**

1. 'use strict';
3. **const** fs = require('fs');
5. **const** readFilePromise= async (file) => {
6. **return** **new** Promise((resolve, reject) => {
7. fs.readFile(file, "utf-8", (err, data) => {
8. **if** (err) reject(err);
10. resolve(data);
11. });
12. });
13. }
15. **const** recursive = (obj) => {
17. let maxHeight = 0;
18. let maxNode = 0;
20. **for**(**var** prop **in** obj) {
21. **if** (**typeof** obj[prop] === "object") {
22. **const** [h, n] = recursive(obj[prop]);
23. **if** (maxHeight < h) {
24. maxHeight = h;
25. maxNode = { [prop]: n };
26. }
27. }
29. **if** (Array.isArray(obj[prop])) {
30. obj[prop].forEach(node => {
31. **const** [h, n] = height(obj[prop]);
32. **if** (maxHeight < h) {
33. maxHeight = h;
34. maxNode = { [prop]: n };
35. }
36. });
37. }
39. **if** (maxHeight < 1) {
40. maxHeight = 1;
41. maxNode = { [prop]: obj[prop] };
42. }
44. }

47. **return** [maxHeight + 1, maxNode];
48. }
50. (async **function** main () {

53. */\**
54. *let obj = {*
55. *alex: {*
56. *age: 21,*
57. *university: "bmstu",*
58. *},*
59. *alina: {*
60. *age: 18,*
61. *university: "B&D",*
62. *look: {*
63. *hat: {*
64. *color: "gray",*
65. *size: "medium"*
66. *},*
67. *hasTie: false,*
68. *}*
69. *}*
70. *}*
71. *\*/*
73. **try** {
74. let objString = await readFilePromise("/Users/gadoevalex/Desktop/evm/lab2/task3/data/generated.txt");
75. let obj = JSON.parse(objString);
77. **const** result = JSON.stringify(recursive(obj), 0, 2);
78. console.log(result);
79. } **catch** (e) {
80. console.log(e);
81. }
82. })();

**Task4**

**Задание 1**

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.

### Задание 2

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.

### Задание 3

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.

### Задание 4

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.

### server.js

1. **const** express = require('express');
2. **const** bodyParser = require('body-parser');
3. **const** app = express();
4. **const** path = require('path');
5. **const** logic = require('./logic');
7. **const** PORT = 3000;
9. app.use(bodyParser.json());
10. app.use(express.**static**(\_\_dirname + "/static"));
12. app.**get**("/", (req, res) => {
13. res.sendFile(path.join(\_\_dirname, "/static", "largestOfThree.html"));
14. });
16. app.**get**("/larger", (req, res) => {
17. res.sendFile(path.join(\_\_dirname, "/static", "largestOfThree.html"));
18. });
20. app.**get**("/getByIndex", (req, res) => {
21. res.sendFile(path.join(\_\_dirname, "/static", "getByIndex.html"));
22. });
24. app.**get**("/generate", (req, res) => {
25. res.sendFile(path.join(\_\_dirname, "/static", "generate.html"));
26. });
28. app.**get**("/numbers", (req, res) => {
29. res.sendFile(path.join(\_\_dirname, "/static", "numbers.html"));
30. });
32. *// получить наибольшее число*
33. app.post('/largerOfThree', (req, res) => {
34. let { a, b, c } = req.body;
36. res.status(200).json({ result: Math.max(a, b, c) });
37. });
39. *// получить объет из массива по индексу*
40. app.post('/infoByIndex', async (req, res) => {
41. **const** index = Number(req.body.index);
43. **try** {
44. let data = await logic.readFromFileByIndex(index);
45. console.log(data);
46. res.status(200).json(data);
47. } **catch** (err) {
48. res.status(500).json(err);
49. }
51. });

54. *// сформировать html с формой отправки*
55. app.post('/generateHtml', (req, res) => {
57. **try** {
58. let fields = ["alpha", "beta", "gamma"];
59. let url = "/myurl";
60. let html = logic.generateHTML(fields, url);
61. console.log(html);
63. res.status(200).json({ html });
64. } **catch** (e) {
65. res.status(500).json(e);
66. }
68. });

71. *// получить числа, входящие в диапазон A до Bо*
72. *// и которые делятся нацело на С*
73. app.post('/numbers', (req, res) => {
75. **const** { a, b, c } = req.body;
77. **try** {
78. **const** result = logic.simpleNumbers(a, b, c);
80. res.status(200).json({ result });
81. } **catch**(e) {
82. res.status(500).json(e);
83. }
84. })

87. app.listen(PORT, (err) => {
88. **if** (err) console.log(err);
90. console.log(`server is listening on port ${PORT}`);
91. });

### logic.js

1. **const** fs = require('fs');
3. **const** INFO\_FILE\_PATH = '/Users/gadoevalex/Desktop/evm/lab2/task4/jsonstring.txt';
5. **const** readFilePromise = (filePath) => {
6. **return** **new** Promise((resolve, reject) => {
7. fs.readFile(filePath, "utf-8", (err, data) => {
8. **if** (err) reject(err);
10. resolve(data);
11. });
12. });
13. }

16. **function** simpleNumbers(a, b, c) {
17. **if** (isNaN(a) || isNaN(b) || isNaN(c))
18. **throw** **new** Error();
19. let result = [];
21. **for**(let i = a; i <= b; i++) {
22. **if** (i % c === 0) {
23. result.push(i);
24. }
25. }
27. **return** result;
28. }
30. async **function** readFromFileByIndex(index) {
31. **if** (isNaN(index))
32. **throw** **new** Error("index is NaN");
34. let allData = await readFilePromise(INFO\_FILE\_PATH)
36. **return** JSON.parse(allData).info[index];
38. }
40. **function** generateHTML(fields, url) {
41. **if** (!url || !fields)
42. **throw** **new** Error();
44. let firstHtml=
45. `
46. <!DOCTYPE html>
47. <html lang="en">
48. <head>
49. <meta charset="UTF-8">
50. <meta name="viewport" content="width=device-width, initial-scale=1.0">
51. <title>Form</title>
52. </head>
53. <body>
54. <form method="POST" action="${url}">\n
55. `
57. **for**(let field of fields) {
58. firstHtml += `\t<input id="${field}" placeholder="${field}"/>\n`
59. }
61. let secondHtml = `
62. </form>
63. </body>
64. </html>
65. `
67. **return** firstHtml += secondHtml;
68. }
70. module.exports = {
71. readFromFileByIndex,
72. simpleNumbers,
73. generateHTML
74. }

### api.js

1. **const** API\_URL = "http://localhost:3000";
3. **const** api = {
4. postParams: **function** (body) {
5. **return** {
6. method: "POST",
7. headers: {
8. "Content-Type": "application/json"
9. },
10. body: JSON.stringify(body)
11. }
12. },
13. largerOfThree: **function** (a, b, c) {
14. **return** fetch(API\_URL + "/largerOfThree", **this**.postParams({ a, b, c })).then(res => res.json());
15. },
16. getByIndex: **function**(index) {
17. **return** fetch(API\_URL + "/infoByIndex", **this**.postParams({ index })).then(res => res.json());
18. },
19. getNumbersInRange: **function**(a, b, c) {
20. **return** fetch(API\_URL + "/numbers", **this**.postParams({ a, b, c  }))
21. .then(res => res.json())
22. },
23. generateHtml: **function**() {
24. **return** fetch(API\_URL + "/generateHtml", **this**.postParams())
25. .then(res => res.json())
26. }
28. };

### Структура проекта

### 

### Результаты

### 

### 

### 

### 