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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

## **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 2**

**Название лабораторной:** Работа с файлами. Формат JSON. Основы Express

**Предмет:** Архитектура ЭВМ

Студент Павлов Н.А. Группа ИУ7-52Б Подпись \_\_\_\_\_

*фамилия, имя, отчество*

Преподаватель Попов А.Ю. Подпись \_\_\_\_\_

*фамилия, имя, отчество*

*Москва, 2020 г.*

## Цели:

Познакомиться с механизмами работы с файловой системой в **Node.js**, используя **fs**. Научиться обращаться с форматом **JSON**, изучить способы преобразования объектов в строки и обратно. Познакомиться с фреймворком **Express**, научиться поднимать сервер с его использованием. Изучить основы **HTML** и **CSS**, научиться генерировать **HTML**-страницы, познакомиться с принципом работы **HTTP**; узнать, что такое **GET** и **POST** запросы.

## Задание 1

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

## Решение

```
"use strict";

// Чтение массива чисел с клавиатуры, вывод в формате JSON в файл всех чётных по длине
function readArray(filename = "result.txt") {
  const readlineSync = require('readline-sync');
  const fs = require("fs");

  const n = parseInt(readlineSync.question(" Input N: "));
  if (isNaN(n) || n < 0) {
    console.log( " Number of files is wrong!");
    return false;
  }
  console.log();

  let string;
  let array = [];

  for (let i = 0; i < n; i++) {
    string = readlineSync.question(" Input string: ");
    if (!(string.length % 2)) {
      array.push(string);
    }
  }
  console.log();

  const jsonString = JSON.stringify(array);
  fs.writeFileSync(filename, jsonString);

  return true;
}

readArray();
```

## Тесты

### Входные данные:

```
Input N: 6
Input string: debug
Input string: puss
Input string:
Input string: nope
Input string: 1
Input string: lot
```

### Выходные данные

```
["puss","","nope"]
```

## 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.

## Решение

```
"use strict";

// Получение дескриптора файла
// Если честно, выносить толку мало
function getFileDescriptor(filename = "data.txt") {
    const fs = require("fs");
    return fs.existsSync(filename) ? fs.readFileSync(filename, "utf8") : null;
}

// Проверка нахождения гласной в строке
// Гласные - строчные гласные кириллицы
function containsVowel(string) {
    let vowels = 'еаоэяиюёуы';
    let result = false;

    for (let symbol of string) {
        for (let vowel of vowels) {
            if (symbol == vowel) {
                result = true;
                break;
            }
        }
    }

    return result;
}
```

```
// Вывод слов с гласными из файла
function outputWithVowels(filename = "data.txt") {
    let array = JSON.parse(getFileDescriptor(filename));
    if (!array) {
        return false;
    }
    for (let string of array) {
        if (containsVowel(string)) {
            console.log(string);
        }
    }
}

outputWithVowels();
```

## Тесты

### Входные данные:

```
[ "слово", "мхъ", "", "половцы", "0", "ьо", "л" ]
```

### Выходные данные

```
слово
половцы
ьо
```

## 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.

## Решение

```
// Для хранения содежимого используется Map
function getFiles() {
  const fs = require("fs");
  const path = require('path');
  const readlineSync = require('readline-sync');

  const extension = readlineSync.question(' Enter extension: ');
  let directory = readlineSync.question(' Enter directory: ');
  console.log();

  let array;

  if (fs.existsSync(directory)) {
    array = fs.readdirSync(directory);
  }
  else {
    console.error(" Directory is not found!");
    return false;
  }







  let contents = new Map();

  // Проверка доступности каждого файла
  for (let filename of array) {
    if (filename.endsWith(extension)) {
      const path = directory + '/' + filename;
      let content;
      if (fs.existsSync(path)) {
        content = fs.readFileSync(path, "utf8");
      } else {
        console.error(" File is unavalible!");
        return false;
      }
      contents[filename] = content;
    }
  }
  return contents;
}

console.log(getFiles());
```

# Тесты

## Рабочая директория src

 jcssp.2020.1195.1202.pdf	17.09.2020 23:28	Chrome HTML Do...	1 099 КБ
 Latency.docx	02.06.2020 16:27	Документ Microso...	14 КБ
 settings.txt	19.09.2020 7:42	Текстовый докум...	1 КБ
 smol.txt	19.09.2020 7:42	Текстовый докум...	1 КБ
 song.txt	17.09.2020 15:19	Текстовый докум...	2 КБ
 TEMP.LST	19.09.2020 12:50	MASM Listing	3 КБ

### Входные данные

```
Enter extension: .txt
Enter directory: src
```

```
Enter extension: 1
Enter directory:
```

```
Enter extension: 123
Enter directory: ./src
```

### Выходные данные

```
Map {
  'settings.txt': "Her som'!",
  'smol.txt': 'smol.tx',
  'song.txt': 'Кто здесь самый главный анархист?\r\n' +
    'Кто здесь самый хитрый шпиён?\r\n' +
    'Кто здесь самый лютый судья?\r\n' +
    'Кто здесь самый удалой господь?\r\n' +
    '\r\n' +
    'Неба синь да земли конура \r\n' +
    'Тебя бензин да меня дыра \r\n' +
    'Пока не поздно—пошёл с ума прочь\r\n' +
    'Пока не поздно—из крысы прямо в ангелы \r\n' +
    '\r\n' +
}
```

```
Directory is not found!
```

```
Map {}
```

## Задание 4

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

## Решение

```
// Получение всех файлов с заданным расширением в директории (и её директориях)
function getFiles() {
  const fs = require("fs");
  const path = require('path');
  const readlineSync = require('readline-sync');

  const len = 10
  const extension = '.txt'
  let directory = readlineSync.question(' Enter directory: ');
  console.log();

  let files = [];

  // Поиск подходящих файлов в папке
  function parseDirectory(directory) {
    let array;
    if (fs.existsSync(directory)) {
      array = fs.readdirSync(directory);
    }
    else {
      console.error(" Directory is not found!");
      return false;
    }

    for (let filename of array) {
      let path = directory + '/' + filename;
      if (fs.statSync(path).isDirectory()) {
        parseDirectory(path);
      }
      else if (path.endsWith(extension)) {
        let content
        if (fs.existsSync(path)) {
          content = fs.readFileSync(path, "utf8");
        }
        else {
          console.error(" File is unavalible!");
          return false;
        }
      }





      if (content.length <= len) {
        files.push(path);
      }
    }
  }

  parseDirectory(directory);
  return files;
}




outputPaths(getFiles());
```

## Тесты





### Директория src

 scripts	17.09.2020 15:20	Папка с файлами	
 settings.txt	19.09.2020 7:42	Текстовый докум...	1 КБ
 Новый точечный рисунок.bmp	17.09.2020 15:18	Файл "BMP"	0 КБ
 Отчёт по лабораторной.pdf	17.09.2020 10:56	Chrome HTML Do...	2 419 КБ

### Директория scripts

 js	19.09.2020 7:42	Папка с файлами	
 song.txt	17.09.2020 15:19	Текстовый докум...	2 КБ
 unnamed0.graphml	16.06.2020 20:50	Graph Markup Lan...	23 КБ

### Директория js

 smol.txt	19.09.2020 7:42	Текстовый докум...	1 КБ
 speech.txt	17.04.2020 15:22	Текстовый докум...	13 КБ
 Unity_lic.alf	30.03.2020 17:49	Файл "ALF"	1 КБ
 ДБ.png	27.08.2020 12:29	Файл "PNG"	43 КБ

### Входные данные

```
Enter directory: ./src
```

```
Enter directory: ./not_found
```

```
Enter directory: ./src/scripts
```

### Выходные данные

```
./src/scripts/js/smol.txt  
./src/settings.txt
```

```
Directory is not found!
```

```
./src/scripts/js/smol.txt
```



## 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.

## Решение

```
"use strict";

// Получение n и списка файлов
function getFiles(){
    const readlineSync = require('readline-sync');
    const fs = require("fs");

    const n = parseInt(readlineSync.question(" Input N: "));
    if (isNaN(n) or n < 0) {
        console.log( "Number of files is wrong!");
        return false;
    }
    console.log();

    let strings = [];

    for (let i = 0; i < n; i++) {
        const string = readlineSync.question(" Input filename: ");
        strings.push(string);
    }

    return strings;
}
```

```
// Соединение содержимого файлов и выгрузка в новый файл
function concatFiles(files) {
    const readlineSync = require('readline-sync');
    const fs = require("fs");

    let contents = "";
    const output_name = readlineSync.question(" Input output file name: ");
    let content;




    for (let file of files) {
        if (fs.existsSync(file)) {
            content = fs.readFileSync(file, "utf8");
        }
        else {
            console.error(" File is unavalible!");
            return false;
        }
        contents += content ;
    }

    fs.writeFileSync(output_name, contents + '\n');
    return true;
}



concatFiles(getFiles());
```

# Тесты

## Директория src

 contains	20.09.2020 16:06	Папка с файлами	
 1.txt	20.09.2020 16:05	Текстовый докум...	1 КБ
 2.txt	20.09.2020 16:05	Текстовый докум...	1 КБ

## Директория contains

 3.txt	20.09.2020 16:06	Текстовый докум...	1 КБ
 4.txt	20.09.2020 16:06	Текстовый докум...	1 КБ

### Файл 1.txt

|This is my rifle!

### Файл 2.txt

|This is my gun!

### Файл 3.txt

|This is for fighting!

### Файл 4.txt

|This is for fun!

## Входные данные:

```
Input N: -4
```

```
Input N: 4
```

```
Input filename: ./src/1.txt
Input filename: ./src/2.txt
Input filename: ./src/contains/2.txt
Input filename: ./src/contains/4.txt
Input output file name: cd
```

```
Input N: 4
```

```
Input filename: ./src/1.txt
Input filename: ./src/2.txt
Input filename: ./src/contains/3.txt
Input filename: ./src/contains/4.txt
Input output file name: file.txt
```

## Выходные данные:

```
Number of files is wrong!
```

```
File is unavalible!
```

```
This is my rifle!
This is my gun!
This is for fighting!
|This is for fun!
```

## 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.

```
// use strict";

// Добавление в объекта в качестве поля в другой объект
function wrapObject(object) {
    let new_object = {};
    new_object["value"] = object;
    return new_object;
}

// Поиск максимальной возможной вложенности JSON
// При превышении вложенности "бросается" ошибка переполнения кучи
// Дерево с максимально возможной вложенностью записывается в файл
function getMaxNesting(dumping_file = "nesting.txt") {
    const fs = require('fs');
    let object = {value: 'Я очень глубоко!'};
    let value = 0;
    let string;

    while (true) {
        try {
            string = JSON.stringify(object);
        } catch(error) {
            console.log(" Max nesting is " + (value - 1) + "!");
            console.log(" Resulting JSON is dumped into " + dumping_file + "!");
            fs.writeFileSync(dumping_file, string);
            break;
        }
        object = wrapObject(object);
        value++;
    }
}

getMaxNesting();
```

## Результат

```
Max nesting is 964!  
Resulting JSON is dumped into nesting.txt!
```

Файл `nesting.txt` (это только часть [весьма малая часть] )

[illegible]

## 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.

## Решение

```
"use strict";

// Получение случайного целого числа в диапазоне
function randInt(min, max) {
    let rand = min - 0.5 + Math.random() * (max - min + 1);
    return Math.round(rand);
}

// Генерации бинарного дерева и запись в формате JSON в файл
function generateTree(filename = 'tree.txt') {
    const fs = require('fs');
    const readlineSync = require('readline-sync');
    const symbols = "QWERTYUIOPASDFGHJKLZXCVBNM";

    // Рекурсивное создание ветки дерева
    function generateBranch() {
        let branches = randInt(0, 2);
        let tree = {"value": symbols[randInt(0, symbols.length - 1)]};

        if (branches >= 1) {
            tree["left"] = generateBranch();
        }
        if (branches == 2) {
            tree["right"] = generateBranch();
        }

        return tree;
    }

    const string = JSON.stringify(generateBranch(), null, '  ');
    fs.writeFileSync(filename, string);
}
```

```
// Получение бинарного дерева из файла
function parseTree() {
  const fs = require('fs');
  const readLineSync = require('readline-sync');

  const filename = readLineSync.question(" Enter filename: ");
  let content;
  let tree;

  if (fs.existsSync(filename)){
    content = fs.readFileSync(filename, "utf8");
  }
  else {
    console.error(" File is unavalible!");
    return false;
  }

  try {
    tree = JSON.parse(content);
  }
  catch (error) {
    console.error(" File doesn't contain JSON!");
    return false;
  }

  return tree;
}
```

```
// Получение максимального пути в бинарном дереве
function getMaxTrace(tree) {

  if (!tree)
    return "";

  if (!tree["left"] && !tree["right"])
    return tree["value"];

  let left = getMaxTrace(tree["left"]);
  let right = getMaxTrace(tree["right"]);
  return tree["value"] + ((left.length > right.length) ? left : right);
}

function main() {
  generateTree();
  let tree = parseTree();
  if (tree) {
    let max_trace = getMaxTrace(tree)
    console.log(" Tree: ");
    console.log(tree);
    console.log(" Max trace: " + max_trace);
    console.log(" Max depth: " + max_trace.length);
  }
}

main();
```

## Тесты

### Входные данные

```
{  
  "value": "I"  
}
```

### Выходные данные

```
Tree:  
{ value: 'I' }  
Max trace: I  
Max depth: 1
```

```
{  
  "value": "B",  
  "left": {  
    "value": "P",  
    "left": {  
      "value": "F",  
      "left": {  
        "value": "U",  
        "left": {  
          "value": "M",  
          "left": {  
            "value": "M"  
          },  
          "right": {  
            "value": "N"  
          }  
        },  
        "right": {  
          "value": "Q",  
          "left": {  
            "value": "D"  
          }  
        }  
      }  
    }  
  }  
}
```

```
Tree:  
{  
  value: 'B',  
  left: {  
    value: 'P',  
    left: { value: 'F', left: [Object] },  
    right: { value: 'P' }  
  },  
  right: { value: 'P', left: { value: 'E', left: [Object] } }  
}  
Max trace: BPFUQD  
Max depth: 6
```

```
{ }{} tree
```

```
Enter filename: false.txt  
File doesn't contain JSON!
```

## Задание 8

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.

## Решение

```
"use strict";

// Класс сервера express
class Server {
  // Объекты используемых библиотек задаются как переменные класса
  static fs = require("fs");
  static express = require("express");
  static pug = require("pug");

  // Создание экземпляра сервера с проверкой свободности порта
  // После запуска идёт привязка маршрутов
  constructor(port = 5015) {
    this.app = Server.express();
    this.port = port;
    try {
      this.app.listen(this.port);
      console.log(" Starting server on port " + this.port + "... ");
    } catch (error) {
      console.log(" Failure while starting server!");
      console.log(` Message: ${error.message}`);
      throw new Error('Server starting failure');
    }

    this.app.get("/page", this.getPage);
    this.app.get("/max_of_three", this.getMaxOfThree);
    console.log(" Server started succesfully!");
  }
}
```

```
// Запрос максимума из трёх
getMaxOfThree(request, response) {

  const a = parseInt(request.query.a);
  const b = parseInt(request.query.b);
  const c = parseInt(request.query.c);

  if (isNaN(a) || isNaN(b) || isNaN(c)) {
    const contentString = Server.fs.readFileSync("html/nan.html", "utf8");
    response.end(contentString);
  } else {
    const templateString = Server.pug.compileFile('html/result.pug');
    response.end(templateString({
      number: Math.max(a, b, c),
      array: [a, b, c]
    }));
  }
}
```

```
// Запрос на получение страницы
getPage(request, response) {
  const nameString = request.query.p;
  if (Server.fs.existsSync(nameString)) {
    const contentString = Server.fs.readFileSync(nameString, "utf8");
    response.end(contentString);
  } else {
    const contentString = Server.fs.readFileSync("html/page_not_found.html", "utf8");
    response.end(contentString);
  }
}

}

let server = new Server(5015);
```

## max\_of\_three.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Максимум из 3</title>
  <link rel="stylesheet" type="text/css" href="https://cdn.discordapp.com/attachments/454170646348562454/757885706025566248/style.css" />
</head>
<body>
  <h1>Какая же классная страница!</h1>
  <form method="GET" action="/max_of_three">
    <p>Введите A</p>
    <input name="a" spellcheck="false" autocomplete="off">
    <p>Введите B</p>
    <input name="b" spellcheck="false" autocomplete="off">
    <p>Введите C</p>
    <input name="c" spellcheck="false" autocomplete="off">
    <br>
    <br>
    <input type="submit" value="Отправить запрос">
  </form>
</body>
</html>
```

## nan.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Поиск максимума не получился</title>
  <link rel="stylesheet" type="text/css" href="https://cdn.discordapp.com/attachments/454170646348562454/757885706025566248/style.css" />
</head>
<body>
  <h1>Одно из чисел не является числом.</h1>
  <h3>Нам очень жаль</h3>
</body>
</html>
```

## page\_not\_found.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Ресурс отсутствует</title>
  <link rel="stylesheet" type="text/css" href="https://cdn.discordapp.com/attachments/454170646348562454/757885706025566248/style.css" />
</head>
<body>
  <h1>Ресурс отсутствует</h1>
  <h3>Какая досада</h3>
</body>
</html>
```

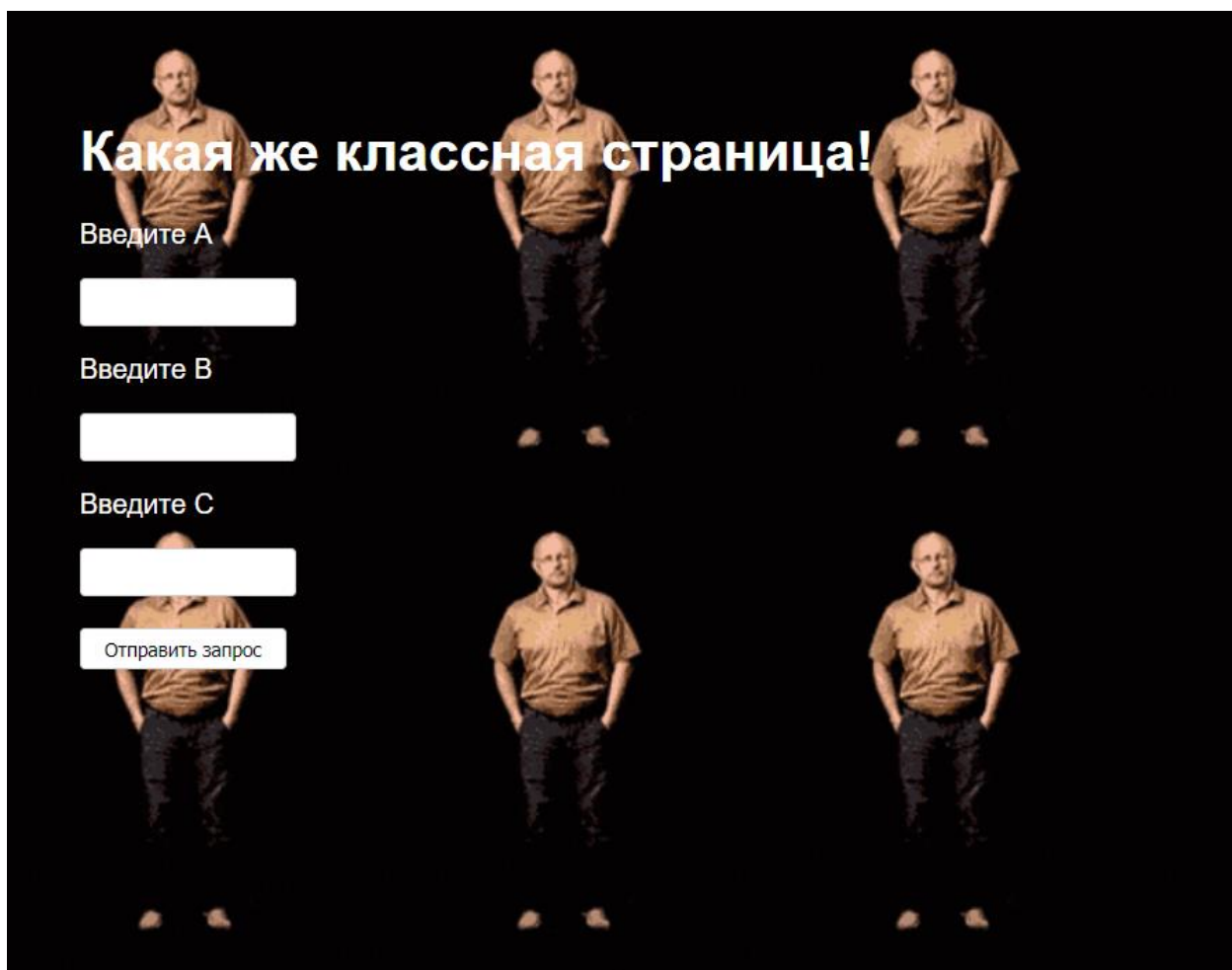
## result.pug

```
doctype html
head
  <meta charset="UTF-8">
  title Максимальное число
  style
    include style.css
body
  h1 #{number} - наибольшее число среди #{array}!
```



## Страницы

<http://localhost:5015/page/?p=html/max of three.html>



The screenshot shows a web application interface with a black background. On the left side, there are three input fields for numbers A, B, and C, each preceded by a label "Введите" (Enter). Below the input fields is a button labeled "Отправить запрос" (Send request). On the right side, there are three identical images of a man in a brown shirt and dark pants, standing with his hands in his pockets. The text "Какая же классная страница!" (What a great page!) is displayed in white at the top left.

Какая же классная страница!

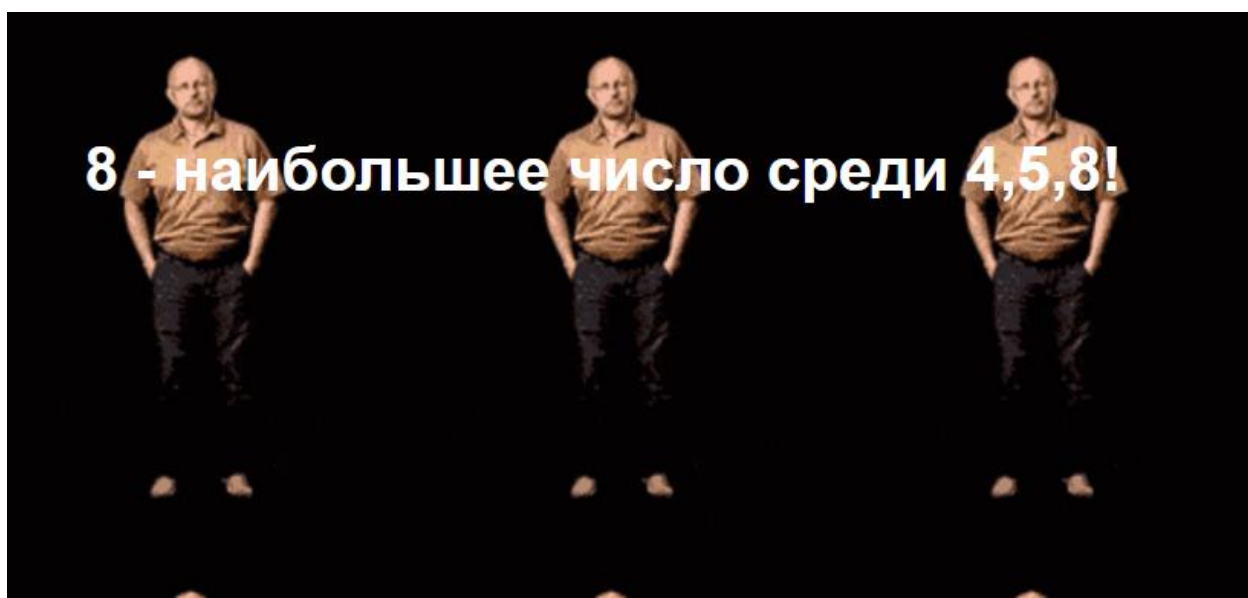
Введите А

Введите В

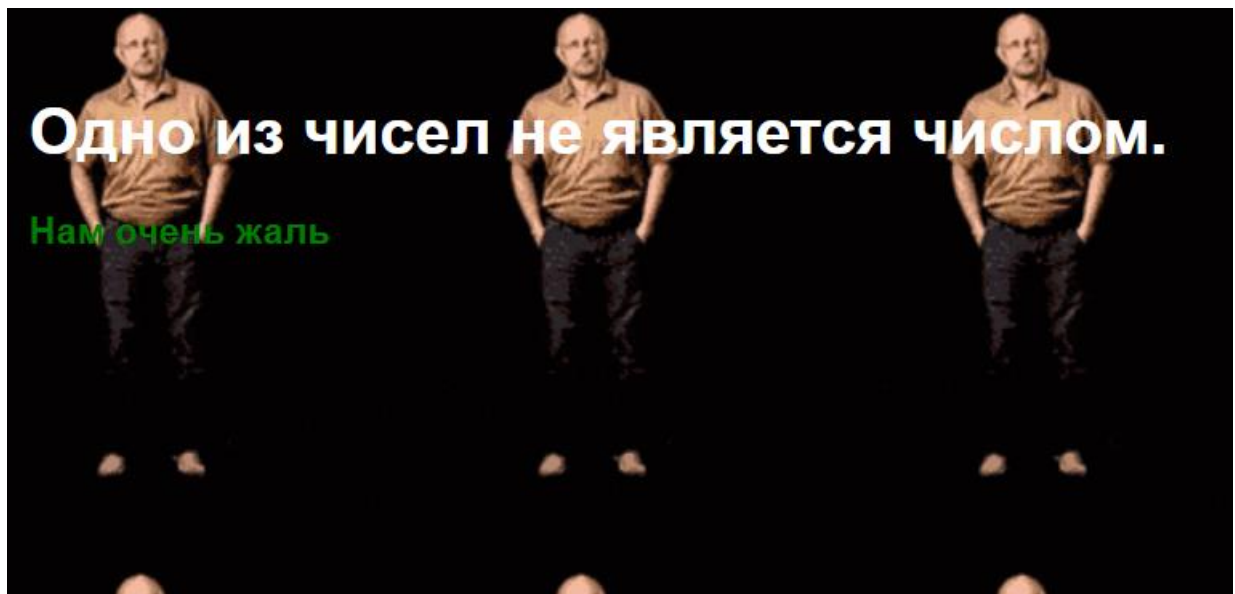
Введите С

Отправить запрос

<http://localhost:5015/max of three?a=4&b=5&c=8>



<http://localhost:5015/max of three?a=4&b=5&c=d>



<http://localhost:5015/page/?p=html/eee.html>



## Задание 9

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.

```
"use strict";

// Класс сервера express
class Server {
  // Объекты библиотек задаются как методы класса
  // Для создания шаблонных HTML-файлов используется библиотека pug
  static fs = require("fs");
  static express = require("express");
  static pug = require("pug");

  // При создании экземпляра происходит проверка доступности порта
  // Для использования изображений со стороны сервера используется express.static
  // После создания происходят бинды маршрутов
  constructor(port = 5015) {
    this.app = Server.express();
    this.port = port;
    try {
      this.app.listen(this.port);
      console.log(" Starting server on port " + this.port + "... ");
    } catch (error) {
      console.log(" Failure while starting server!");
      console.log(` Message: ${error.message}`);
      throw new Error('Server starting failure');
    }

    this.app.use(Server.express.static(__dirname + '/visuals'));
    this.app.get("/page", this.getPage);
    this.app.get("/get_element", this.getElement);
    console.log(" Server started succesfully!");
  }
}

// Метод получения элемента из JSON
getElement(request, response) {
  const index = parseInt(request.query.index);

  if (isNaN(index)) {
    const contentString = Server.fs.readFileSync("public/nan.html", "utf8");
    response.end(contentString);
  } else {
    const array = JSON.parse(Server.fs.readFileSync("src/array.json", "utf8"));
    if (index < 0 || index >= array.length) {
      const contentString = Server.fs.readFileSync("public/out_of_range.html", "utf8");
      response.end(contentString);
    } else {
      const templateString = Server.pug.compileFile('public/result.pug');
      response.end(templateString({
        value: array[index],
        index: index
      }));
    }
  }
}
```

```
// Метод получения страницы
getPage(request, response) {
    const nameString = request.query.p;
    if (Server.fs.existsSync(nameString)) {
        const contentString = Server.fs.readFileSync(nameString, "utf8");
        response.end(contentString);
    } else {
        const contentString = Server.fs.readFileSync("public/page_not_found.html", "utf8");
        response.end(contentString);
    }
}

}

let server = new Server(5015);
```

## array.json

```
[1, 1, 2, 3, 5, 8, 13, 21, 34]
```

## get\_element.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Значение по индексу</title>
    <link rel="stylesheet" type="text/css" href="/style.css" />
</head>
<body>

    <h1>Какая же классная страница!</h1>
    <form method="GET" action="/get_element">
        <p>Введите индекс</p>
        <input name="index" spellcheck="false" autocomplete="off">
        <br>
        <br>
        <input type="submit" value="Отправить запрос">
    </form>
</body>
</html>
```

## nan.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Индекс некорректен.</title>
    <link rel="stylesheet" type="text/css" href="/style.css" />
</head>
<body>
    <h1>Индекс некорректен.</h1>
    <h3>Заданное значение не индекса не является числом!</h3>
</body>
</html>
```

## out\_of\_range.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Индекс некорректен.</title>
    <link rel="stylesheet" type="text/css" href="/style.css" />
</head>
<body>
    <h1>Индекс некорректен.</h1>
    <h3>Заданное значение индекса выходит за границы массива!</h3>
</body>
</html>
```

[http://localhost:5015/page/?p=public/get\\_element.html](http://localhost:5015/page/?p=public/get_element.html)

**Какая же классная страница!**

Введите индекс

Отправить запрос

[http://localhost:5015/get\\_element?index=-6](http://localhost:5015/get_element?index=-6)

**Индекс некорректен.**

Заданное значение индекса выходит за границы массива!

[http://localhost:5015/get\\_element?index=3](http://localhost:5015/get_element?index=3)

**3 - значение по индексу 3!**



[http://localhost:5015/get\\_element?index=d](http://localhost:5015/get_element?index=d)

**Индекс некорректен.**

Заданное значение не индекса не является числом!

## Задание 10

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.

```
"use strict";

// Генератор HTML-файлов для набора полей ввода
function generateHTML(path = "input.html") {
  const fs = require("fs");
  const readlineSync = require("readline-sync");

  const query_address = readlineSync.question(" Enter query address: ");
  const title = readlineSync.question(" Enter title: ");
  const page_header = readlineSync.question(" Enter page header: ");
  const n = parseInt(readlineSync.question(" Enter number of inputs: "));
  if (isNaN(n) || n <= 0) {
    console.log(" Invalid number of inputs!");
    return false;
  }

  const array = [];
  for (let i = 0; i < n; i++) {
    array.push(readlineSync.question(" Enter field name: "));
  }

  // Метод генерации заголовка (head)
  function generateHead(title = "Шаблонная форма") {
    let stringHTML =
      `<!DOCTYPE html>\n<html>\n<head>\n<meta charset="UTF-8">\n<title>${title}</title>\n</head>\n`;
    return stringHTML;
  }

  // Метод генерации одного поля ввода (input)
  function generateInput(field_name) {
    let stringHTML =
      `
      <p>Введите ${field_name}</p>
      <input name="${field_name}" spellcheck="false" autocomplete="off">\n`;
    return stringHTML;
  }

  // Метод генерации тела (body)
  function generateBody(page_header, query_address, array) {
    let stringHTML =
      `<body>\n<h1>${page_header}</h1>\n<form method = "GET" action="/${query_address}">\n`;
    for (let element of array) {
      stringHTML += generateInput(element);
    }
    stringHTML +=
      `
      <br>\n<br>\n<input type = "submit" value = "Отправить запрос">\n</form>\n</body>\n</html>\n`;
    return stringHTML;
  }

  console.log(array);
  let stringHTML = generateHead(title) + generateBody(page_header, query_address, array);
  fs.writeFileSync(path, stringHTML);
}

generateHTML();
```

# Тесты

## Входные данные

```
Enter query address: set_data
Enter title: Some stuff
Enter page header: Some stuff
Enter number of inputs: 6
Enter field name: a
Enter field name: b
Enter field name: c
Enter field name: d
Enter field name: e
Enter field name: f
```

## Выходные данные

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Some stuff</title>
</head>
<body>
  <h1>Some stuff</h1>
  <form method = "GET" action="/set_data">
    <p>Введите a</p>
    <input name="a" spellcheck="false" autocomplete="off">
    <p>Введите b</p>
    <input name="b" spellcheck="false" autocomplete="off">
    <p>Введите c</p>
    <input name="c" spellcheck="false" autocomplete="off">
    <p>Введите d</p>
    <input name="d" spellcheck="false" autocomplete="off">
    <p>Введите e</p>
    <input name="e" spellcheck="false" autocomplete="off">
    <p>Введите f</p>
    <input name="f" spellcheck="false" autocomplete="off">
    <br>
    <br>
    <input type = "submit" value = "Отправить запрос">
  </form>
</body>
</html>
```

## Some stuff

Введите a

Введите b

Введите c

Введите d

Введите e

Введите f

Отправить запрос

## Задание 11

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.

```
"use strict";

// Класс сервера с использованием express
class Server {
  // Объекты библиотек задаются как методы класса
  // Для создания шаблонных HTML-файлов используется библиотека pug
  static fs = require("fs");
  static express = require("express");
  static pug = require("pug");

  // При создании экземпляра происходит проверка доступности порта
  // Для использования изображений со стороны сервера используется express.static
  // После создания происходят бинды маршрутов
  constructor(port = 5015) {
    this.app = Server.express();
    this.port = port;
    try {
      this.app.listen(this.port);
      console.log(" Starting server on port " + this.port + "... ");
    } catch (error) {
      console.log(" Failure while starting server!");
      console.log(` Message: ${error.message}`);
      throw new Error('Server starting failure');
    }

    this.app.use(Server.express.static(__dirname + '/visuals'));
    this.app.get("/page", this.getPage);
    this.app.get("/get_dividers", this.getDividers);
    console.log(" Server started succesfully!");
  }

  // Метод класса для получения делителей на отрезке
  static dividersOnRange(a, b, c) {
    const array = [];
    for (let number = a; number <= b; number++) {
      if (!(number % c)) {
        array.push(number);
      }
    }
  }
}
```



```
// Метод получения делителей при запросе
getDividers(request, response) {
  const a = parseInt(request.query.a);
  const b = parseInt(request.query.b);
  const c = parseInt(request.query.c);

  if (isNaN(a) || isNaN(b) || isNaN(c)) {
    const contentString = Server.fs.readFileSync("public/nan.html", "utf8");
    response.end(contentString);
  } else {
    if (a > b) {
      const contentString = Server.fs.readFileSync("public/invalid_range.html", "utf8");
      response.end(contentString);
    } else {
      const array = Server.dividersOnRange(a, b, c);
      console.log(array);
      if (!array.length) {
        const templateString = Server.pug.compileFile('public/not_found.pug');
        response.end(templateString({
          a: a,
          b: b,
          c: c
        }));
      } else {
        const templateString = Server.pug.compileFile('public/found.pug');
        response.end(templateString({
          a: a,
          b: b,
          c: c,
          array: array
        }));
      }
    }
  }
}
}
```

```
// Метод получения страницы
getPage(request, response) {
  const nameString = request.query.p;
  if (Server.fs.existsSync(nameString)) {
    const contentString = Server.fs.readFileSync(nameString, "utf8");
    response.end(contentString);
  } else {
    const contentString = Server.fs.readFileSync("public/page_not_found.html", "utf8");
    response.end(contentString);
  }
}

let server = new Server(5015);
```

## found.pug

```
doctype html
head
  <meta charset="UTF-8">
  title Делители на отрезке
  style
    include style.css
body
  h1 На отрезке [{a}, {b}] делители {c}:
  h2 #{array}
  
```

## get\_dividers.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="/style.css" />
  <title>Поиск делителей на отрезке</title>
</head>
<body>
  <h1>Поиск делителей на отрезке</h1>
  <form method = "GET" action="/get_dividers">
    <p>Введите a</p>
    <input name="a" spellcheck="false" autocomplete="off">
    <p>Введите b</p>
    <input name="b" spellcheck="false" autocomplete="off">
    <p>Введите c</p>
    <input name="c" spellcheck="false" autocomplete="off">
    <br>
    <br>
    <input type = "submit" value = "Отправить запрос">
  </form>
</body>
</html>
```

## invalid\_range.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Значения некорректны.</title>
  <link rel="stylesheet" type="text/css" href="/style.css" />
</head>
<body>
  <h1>Значения некорректны.</h1>
  <h3>Начало отрезка не может быть больше его конца!</h3>
  
</body>
</html>
```

## nan.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Значения некорректны.</title>
  <link rel="stylesheet" type="text/css" href="/style.css" />
</head>
<body>
  <h1>Значения некорректны.</h1>
  <h3>Одно из заданных значений не является числом!</h3>
  
</body>
</html>
```

## not\_found.pug

```
doctype html
head
  <meta charset="UTF-8">
  title Делители на отрезке
  style
    include style.css
body
  h1 На отрезке [{a}, {b}] делителей {c} не найдено!
  
```

## page\_not\_found.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Ресурс отсутствует</title>
  <link rel="stylesheet" type="text/css" href="/style.css" />
</head>
<body>
  <h1>Ресурс отсутствует</h1>
  <h3>Какая досада!</h3>
  
</body>
</html>
```

## Страницы

[http://localhost:5015/page/?p=public/get\\_dividers.html](http://localhost:5015/page/?p=public/get_dividers.html)

### Поиск делителей на отрезке

Введите a

Введите b

Введите c

Отправить запрос

[http://localhost:5015/get\\_dividers?a=5&b=3&c=2](http://localhost:5015/get_dividers?a=5&b=3&c=2)

### Значения некорректны.

Начало отрезка не может быть больше его конца!



[http://localhost:5015/get\\_dividers?a=4&b=100&c=5](http://localhost:5015/get_dividers?a=4&b=100&c=5)

**На отрезке [4, 100] делители 5:**

5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100



[http://localhost:5015/get\\_dividers?a=4&b=100&c=101](http://localhost:5015/get_dividers?a=4&b=100&c=101)

**На отрезке [4, 100] делителей 101 не найдено!**



**Цели:** За время выполнения лабораторной работы были изучены механизмы работы с файловой системой и JSON в Node.js. Были получены навыки работы с фреймворком Express, создания и описания HTML и CSS