**ОТЧЕТ**

по лабораторной работе №5

по курсу «Web-программирование»

на тему «Реализация ответа обработчиков и POST-запросы»

Выполнили:

студенты группы 16ВВ1

Малинин Э.В.

Кузнецов А.В.

Фадеев Е.Д.

Принял:

к.т.н., доцент Дубравин А.В.

к.т.н., доцент Карамышева Н.С.

Пенза 2020

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Название

Реализация ответа обработчиков и POST-запросы.

### Цель работы

Реализовать ответ обработчиков и POST-запросы.

### Лабораторное задание

a. На стороне клиента реализовать проверку: если в текстовом поле указано одно из трех животных в следующем формате: «antilopa» или «antilopa 7» (число от 0 до 9), выполнить формирование и отправку на сервер POST-запроса, в противном случае вывести сообщение о некорректно указанных данных.

b. На стороне сервера реализовать три обработчика, соответствующих каждому из указанных в варианте животных.

c. В обработчике, в случае, если в URL запросе отсутствует параметр с указанием номера запрашиваемого изображения животного, выполнить генерацию случайного числа в диапазоне от 0 до 9 и выполнить отправку клиенту соответствующего изображения. Если в URL запросе присутствует параметр с указанием номера запрашиваемого изображения животного – выполнить отправку клиенту соответствующего изображения.

|  |  |
| --- | --- |
| Вариант | Задание |
| 6 | Животные: еж,ехидна,енот |

### Порядок выполнения работы

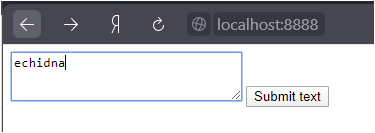


Рисунок 1 – ввод сообщения

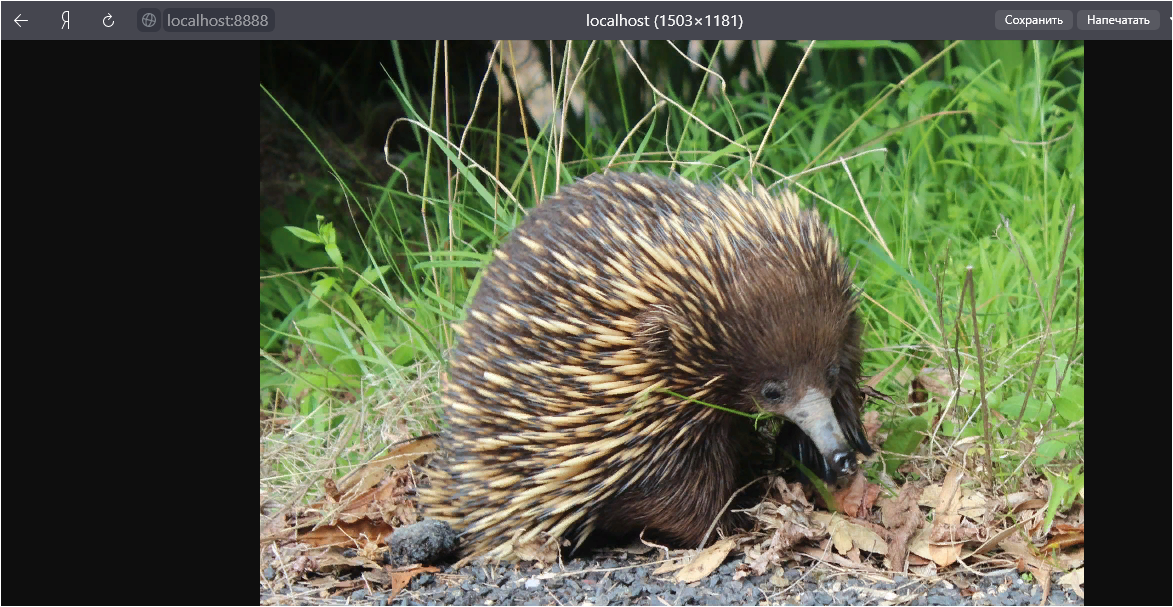


Рисунок 2 - вывод случайного изображения ехидны



Рисунок 3 - вывод в консоль

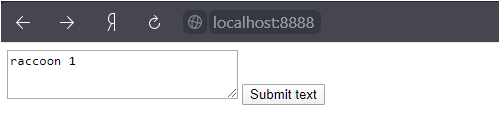


Рисунок 4 - ввод сообщения с числом

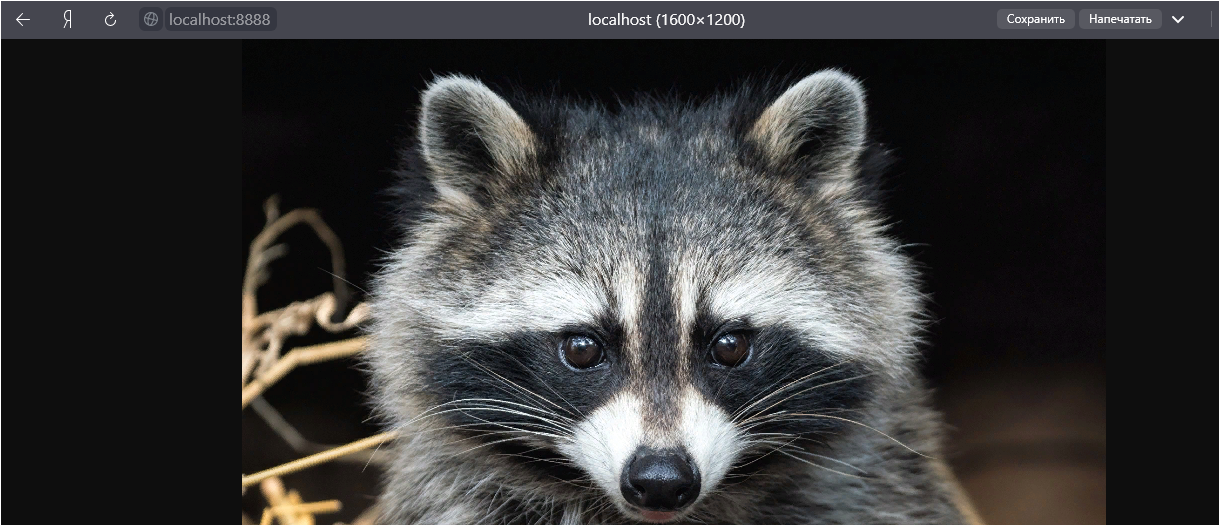


Рисунок 5 - вывод изображения



Рисунок 6 - вывод в консоль

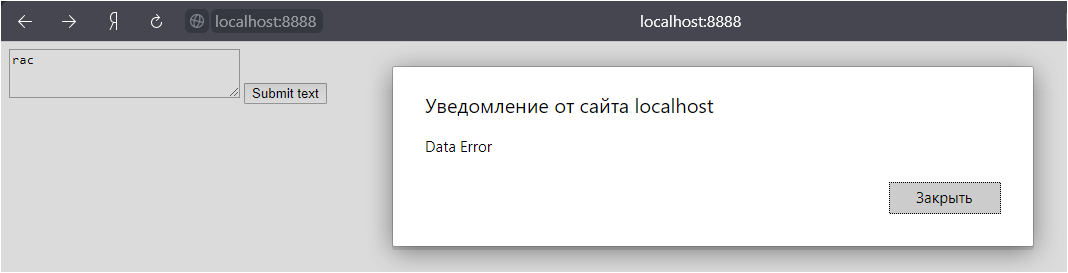


Рисунок 7 – проверка и вывод ошибки на стороне клиента

### Листинг

#### Lab5.html

<html>

    <head>

        <meta charset="utf-8" />

        <script>

            let array = ["hedgehog", "echidna", "raccoon"];

            function check(form) {

                for (var i = 0; i < array.length; i++) {

                    if (form.text.value == array[i]) {

                        return true;

                    }

                    for (var j = 0; j < 10; j++) {

                        if (form.text.value == array[i] + " " + j) {

                            return true;

                        }

                    }

                }

                alert("Data Error");

                return false;

            }

        </script>

    </head>

    <body>

        <form method="post" onsubmit="return check(this)">

            <textarea name="text" rows="3" cols="30"></textarea>

            <input type="submit" value="Submit text" />

        </form>

    </body>

</html>

#### Server.JS

var http = require("http");

var url = require("url");

var querystring = require("querystring");

function start(route, handle) {

  function onRequest(request, response) {

    var postData = "";

    var pathname = url.parse(request.url).pathname;

    console.log("Request for " + pathname + " received.");

    request.setEncoding("utf8");

    request.addListener("data", function(postDataChunk)

    {

      postData += postDataChunk;

      pathname ="/"+ querystring.parse(postData).text.split(' ')[0];

      console.log("Received POST data chunk '"+

      postDataChunk + "'.");

    });

    request.addListener("end", function() {

    route(handle, pathname, response, postData);

    });

  }

  http.createServer(onRequest).listen(8888);

  console.log("Server has started.");

}

exports.start = start;

#### Index.JS

var server = require("./server");

var router = require("./router");

var requestHandlers = require("./requestHandlers");

var handle = {}

handle["/"] = requestHandlers.start;

handle["/start"] = requestHandlers.start;

handle["/upload"] = requestHandlers.upload;

handle["/hedgehog"] = requestHandlers.hedgehog;

handle["/echidna"] = requestHandlers.echidna;

handle["/raccoon"] = requestHandlers.raccoon;

server.start(router.route, handle);

#### router.JS

function route(handle, pathname, response, postData)

{

    console.log("About to route a request for " + pathname);

    if (typeof handle[pathname] === 'function')

    {

        handle[pathname](response, postData);

    }

    else

    {

        console.log("No request handler found for " + pathname);

        response.writeHead(404, {"Content-Type": "text/plain"});

        response.write("404 Not found");

        response.end();

    }

}

exports.route = route;

#### requestHandlers.JS

var fs = require("fs");

var querystring = require("querystring");

function start(response, postData)

{

    console.log("Request handler 'start' was called.");

    var body = fs.readFileSync('lab5.html');

    response.writeHead(200, {"Content-Type": "text/html"});

    response.write(body);

    response.end();

}

function hedgehog(response, postData)

{

    console.log("Request handler 'hedgehog' was called.");

    printfunc(response, postData);

}

function echidna(response, postData)

{

    console.log("Request handler 'echidna' was called.");

    printfunc(response, postData);

}

function raccoon(response, postData)

{

    console.log("Request handler 'raccoon' was called.");

    printfunc(response, postData);

}

function printfunc(response, postData)

{

  var data = querystring.parse(postData).text.split(' ')[1];

  var data2 = querystring.parse(postData).text.split(' ')[0];

  response.writeHead(200, {'Content-Type': 'image/jpeg'});

  if (data != null) {

    var name = data2 + data;

    var imgname =('images/'+ name + '.jpg');

    var image = fs.readFileSync(imgname);

    response.write(image);

    response.end();

  } else {

    var name = data2 + Math.floor(Math.random() \* 10);

    var imgname =('images/'+ name + '.jpg');

    var image = fs.readFileSync(imgname);

    response.write(image);

    response.end();

  }

}

exports.start = start;

exports.hedgehog = hedgehog;

exports.echidna = echidna;

exports.raccoon = raccoon;

### Выводы

В ходе выполнения лабораторной работы реализованы ответы обработчиков и POST-запросы.