**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

по дисциплине «Современные технологии Front-end разработки»

на тему: «Использование объектов типа Promise в JS»

Выполнил: студент гр. ИТП-31

Солодков М.А

Принял: преподаватель-стажёр

Васюкова В.О.

Гомель 2020

**Цель работы:** научиться использовать объекты типа Promise для формирования цепи последовательных вызовов функций.

**Задание**

Подготовить JSON-файл, содержащий следующую информацию:

1. ФИО студента;
2. Факультет;
3. Специальность;
4. Список дисциплин, по которым есть лабораторные работы (каждая дисциплина – это объект, содержащий список лабораторных работ);

**Примечание:** объект «лабораторная работа» должен содержать следующие поля: наименование, дата сдачи, полученная отметка.

**Ход работы**

**Результат выполнения:**

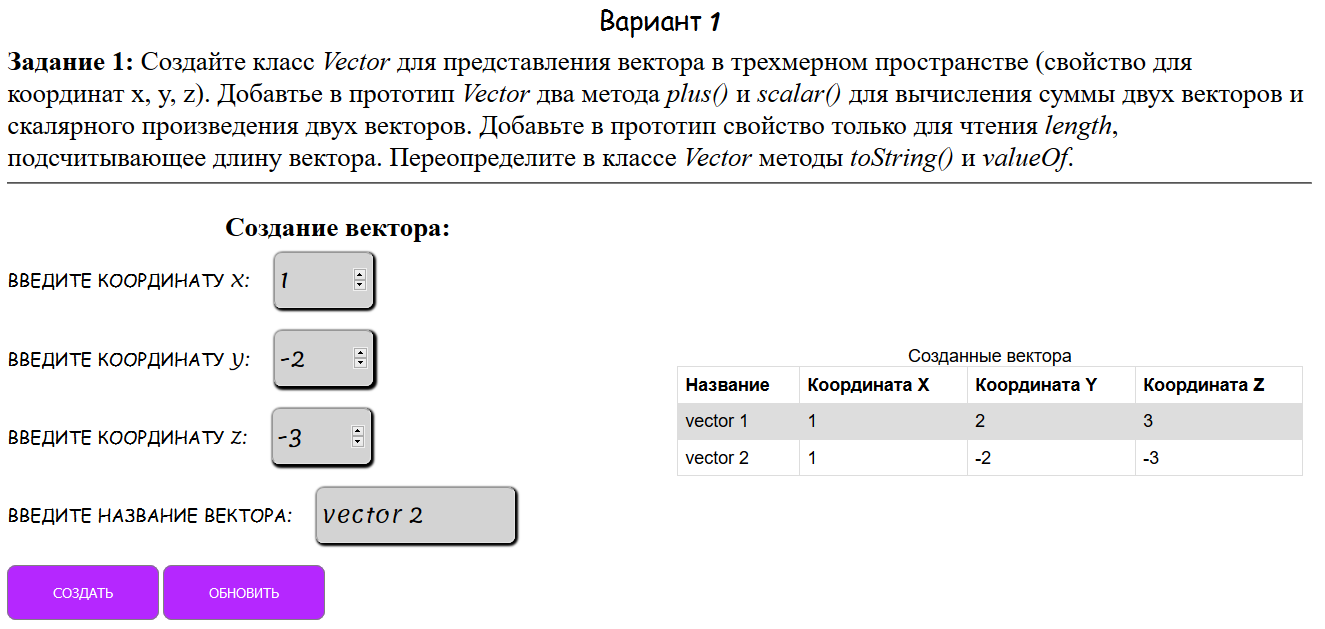


Рисунок 1 – Результат создания трехмерного вектора

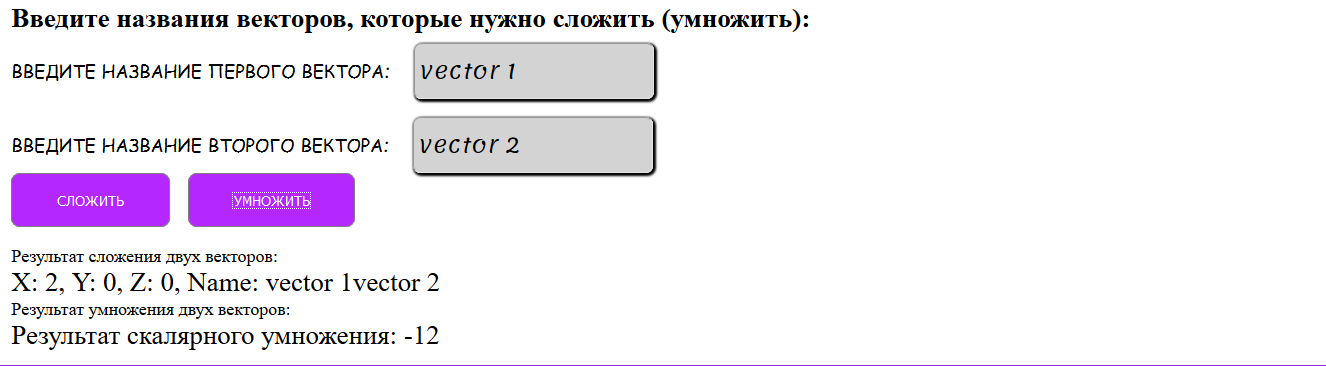


Рисунок 2 – Результат выполнения операций над указанными векторами

**Задание 2.** Реализовать класс Факультет, который хранит список студентов. Класс Студент содержит информацию о студенте. Создайте классы наследники для студентов отдельных спкциальностей. В программе предусмотреть возможность вывода всех студентов заданной специальности и поиска студента по фамилиии.

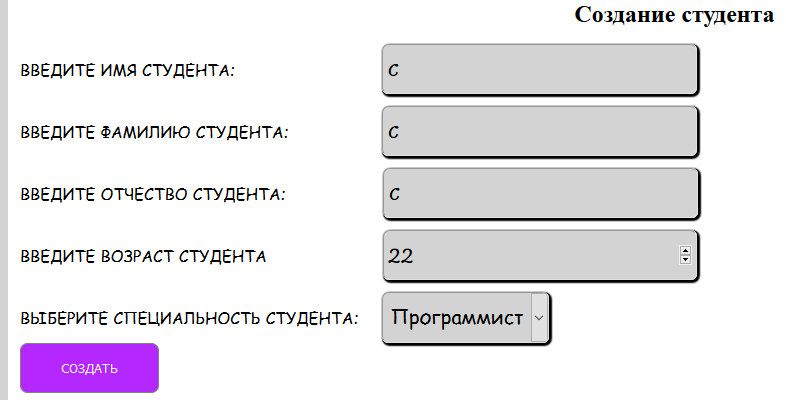


Рисунок 3 – Окно создания студента

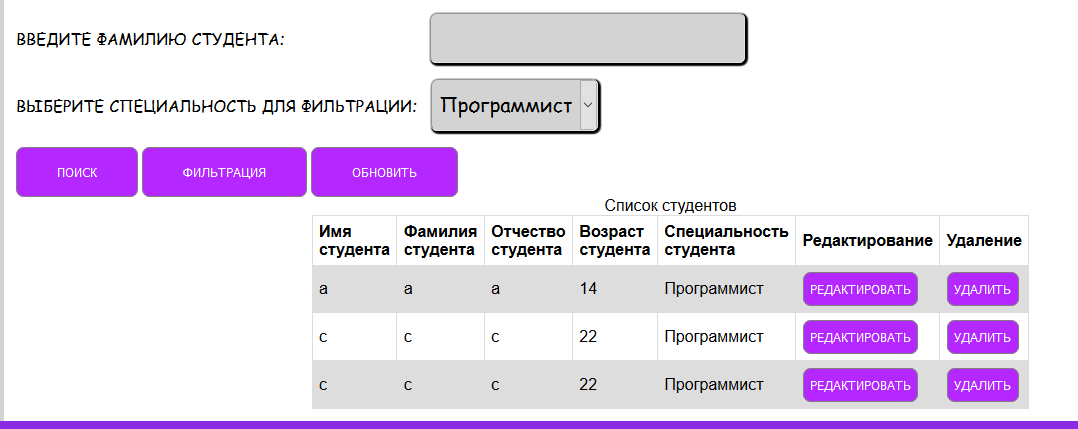


Рисунок 4 – Результат добавления студента

**Вывод:** получили навыки работы с классами изучить принципы ООП в языке программированием в JavaScript.

**Приложение А – Листинг программы**

**layout.hbs**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Лабораторная работа №2, Выполнил: Солодков М.А, Группа: ИТП-31</title>

<link rel="stylesheet" type="text/css" href="/resources/css/style.css">

<link rel="stylesheet" type="text/css" href="/resources/css/style\_lab3.css">

<link rel="stylesheet" type="text/css" href="/resources/css/style\_lab4.css">

<link href="https://fonts.googleapis.com/css?family=Sriracha&display=swap" rel="stylesheet">

<link href="https://fonts.googleapis.com/css?family=Lemonada:300&display=swap" rel="stylesheet">

</head>

<body>

{{> header}}

<div class="nav">

<div class="nav\_\_inner">

{{> nav}}

</div>

</div>

<div class="links">

<div class="links\_\_inner">

{{> links}}

</div>

</div>

<div class="content">

{{{body}}}

</div>

{{> footer}}

<script type="text/javascript" src="/resources/js/jsCalcCurrentDate.js"></script>

<script type="text/javascript" src="/resources/js/lab2Request.js"></script>

<script type="text/javascript" src="/resources/js/lab3Request.js"></script>

<script type="text/javascript" src="/resources/js/lab4Request.js"></script>

</body>

</html>

Файл **app.js**

const express = require("express");

const expressHbs = require("express-handlebars");

const hbs = require("hbs");

const app = express();

const labsRoute = express.Router();

const apiRouter = express.Router();

const jsonParser = express.json();

const lab2 = require('./src/lab2');

const lab3 = require('./src/lab3');

const lab4 = require('./src/lab4');

// устанавливаем настройки для файлов layout

app.engine("hbs", expressHbs(

{

layoutsDir: "views/layouts",

defaultLayout: "layout",

extname: "hbs"

}

));

app.set("view engine", "hbs");

app.use(express.static(\_\_dirname + '/public'));

apiRouter.post('/parser-text', jsonParser, (req, res) => {

if (!req.body.text){

res.status(400);

}

let text = req.body.text;

let parseText = lab2.parserText(text);

res.json({

'answer' : parseText

});

});

apiRouter.post('/calc-date', jsonParser, (req, res) => {

if (!req.body.firstDate || !req.body.secondDate || !req.body.firstTime || !req.body.secondTime){

res.status(400);

}

let firstDate = req.body.firstDate;

let secondDate = req.body.secondDate;

let firstTime = req.body.firstTime;

let secondTime = req.body.secondTime;

let calcDate = lab2.calcCountHourAction(firstDate, secondDate, firstTime, secondTime);

res.json({

'answer' : calcDate

});

});

apiRouter.post('/generate-string', jsonParser, (req, res) => {

if (!req.body.size){

res.status(400);

}

let strGenerate = lab3.strGenerate(req.body.size);

res.json({

'string' : strGenerate.outputStr,

'charArray' : strGenerate.charArray,

'sum' : strGenerate.sum

});

});

apiRouter.post('/create-function', jsonParser, (req, res) => {

if (!req.body.array) {

res.status(400);

}

let result = lab3.getArgsAvg(req.body.array);

res.json({

'avgSum' : result.avgSum,

'avgHarmonic' : result.avgHarmonic

});

});

apiRouter.post('/create-vector', jsonParser, (req, res) => {

if (!req.body.X || !req.body.Y || !req.body.Z || !req.body.Name){

res.status(400);

}

let result = lab4.createVector(req.body.X, req.body.Y, req.body.Z, req.body.Name);

res.json({

'arrayVectors' : result

})

});

apiRouter.post('/add-vectors', jsonParser, (req, res) => {

if (!req.body.firstNameVector || !req.body.secondNameVector){

res.status(400);

}

let result = lab4.plus(req.body.firstNameVector, req.body.secondNameVector);

res.json({

'plus' : result

});

});

apiRouter.post('/scalar-vectors', jsonParser, (req, res) => {

if (!req.body.firstNameVector || !req.body.secondNameVector){

res.status(400);

}

let result = lab4.scalar(req.body.firstNameVector, req.body.secondNameVector);

res.json({

'scalar' : result

});

});

apiRouter.get('/update-table', jsonParser, (req, res) => {

let result = lab4.update();

res.json({

'tableValue' : result

});

});

labsRoute.use('/:id', function (request, response) {

response.render('lab' + request.params['id'] + '.hbs');

});

app.use('/lab', labsRoute);

app.use('/api', apiRouter);

app.use('/', function (request, response) {

response.render('lab1.hbs');

});

app.listen(49749);

console.log('Server started successfully');

Файл **lab4.js**

class Point{

constructor(xCoordinate, yCoordinate) {

this.xCoordinate = xCoordinate;

this.yCoordinate = yCoordinate;

}

toString(){

return 'Координата X: ' +this.xCoordinate+ ', Координата Y: ' + this.yCoordinate;

}

valueOf(){

return 'Координата X: ' +this.xCoordinate+ ', Координата Y: ' + this.yCoordinate;

}

toJSON(){

return {

'XCoordinate' : this.xCoordinate,

'YCoordinate' : this.yCoordinate

};

}

}

class Vector extends Point{

constructor(name, xCoordinate, yCoordinate, zCoordinate) {

super(xCoordinate, yCoordinate);

this.nameVector = name;

this.zCoordinate = zCoordinate;

}

get length(){

return Math.sqrt(Math.pow(this.xCoordinate, 2) + Math.pow(this.yCoordinate, 2) + Math.pow(this.zCoordinate, 2));

}

static plus(nameVectorOne, nameVectorTwo) {

let firstVectorIndex = arrayVectors.findIndex(vector => vector.Name === nameVectorOne);

let secondVectorIndex = arrayVectors.findIndex(vector => vector.Name === nameVectorTwo);

let vectorOne, vectorTwo;

if (firstVectorIndex !== -1 && secondVectorIndex !== -1){

vectorOne = arrayVectors[firstVectorIndex];

vectorTwo = arrayVectors[secondVectorIndex];

}else{

return 0;

}

let newXCoordinate = parseInt(vectorOne.X) + parseInt(vectorTwo.X);

let newYCoordinate = parseInt(vectorOne.Y) + parseInt(vectorTwo.Y);

let newZCoordinate = parseInt(vectorOne.Z) + parseInt(vectorTwo.Z);

let newNameVector = vectorOne.Name + vectorTwo.Name;

return (new Vector(newNameVector, newXCoordinate, newYCoordinate, newZCoordinate)).toJSON();

}

static scalar(nameVectorOne, nameVectorTwo){

let firstVectorIndex = arrayVectors.findIndex(vector => vector.Name === nameVectorOne);

let secondVectorIndex = arrayVectors.findIndex(vector => vector.Name === nameVectorTwo);

let vectorOne, vectorTwo;

if (firstVectorIndex !== -1 && secondVectorIndex !== -1){

vectorOne = arrayVectors[firstVectorIndex];

vectorTwo = arrayVectors[secondVectorIndex];

}else{

return 0;

}

let multiplicationX = parseInt(vectorOne.X) \* parseInt(vectorTwo.X);

let multiplicationY = parseInt(vectorOne.Y) \* parseInt(vectorTwo.Y);

let multiplicationZ = parseInt(vectorOne.Z) \* parseInt(vectorTwo.Z);

return multiplicationX + multiplicationY + multiplicationZ;

}

toString(){

return 'Название: ' +this.nameVector;

}

valueOf(){

return 'Координата X: ' +this.xCoordinate+ ', Координата Y: ' +this.yCoordinate+ ', Координата Z: ' +this.zCoordinate;

}

toJSON() {

return {

'X' : this.xCoordinate,

'Y' : this.yCoordinate,

'Z' : this.zCoordinate,

'Name' : this.nameVector

};

}

}

let arrayVectors = [];

function createVector(xCoordinate, yCoordinate, zCoordinate, nameVector) {

let vector = new Vector(nameVector, xCoordinate, yCoordinate, zCoordinate);

let checkNameVector = arrayVectors.find(vector => vector.Name === nameVector);

if (checkNameVector !== undefined){

return arrayVectors;

}else{

arrayVectors.push(vector.toJSON());

return arrayVectors;

}

}

function getArrayVectors(){

return arrayVectors;

}

module.exports.createVector = createVector;

module.exports.plus = Vector.plus;

module.exports.scalar = Vector.scalar;

module.exports.update = getArrayVectors;

Файл **lab4Request.js**

'use strict';

/\*создание вектора\*/

function createVector() {

let dataJSON = JSON.stringify({

'X' : document.getElementById('enterX').value,

'Y' : document.getElementById('enterY').value,

'Z' : document.getElementById('enterZ').value,

'Name' : document.getElementById('enterName').value,

});

let xhr = new XMLHttpRequest();

xhr.open('POST', '/api/create-vector', true);

xhr.setRequestHeader('Content-type', 'application/json');

xhr.responseType = 'json';

xhr.send(dataJSON);

xhr.onload = () => {

if (xhr.status === 200){

let tableBody = document.getElementById("tableBody");

tableBody.innerHTML = "";

let tr = document.createElement('tr');

let thName = document.createElement('th');

let thX = document.createElement('th');

let thY = document.createElement('th');

let thZ = document.createElement('th');

thName.innerHTML = 'Название';

thX.innerHTML = 'Координата X';

thY.innerHTML = 'Координата Y';

thZ.innerHTML = 'Координата Z';

tr.appendChild(thName);

tr.appendChild(thX);

tr.appendChild(thY);

tr.appendChild(thZ);

tableBody.appendChild(tr);

for (let i = 0; i < xhr.response.arrayVectors.length; i++) {

let tr = document.createElement("tr");

for (let j = 0; j < 4; j++) {

if (j === 0){

let td = document.createElement("td");

td.innerHTML = xhr.response.arrayVectors[i].Name;

tr.appendChild(td);

}

else if (j === 1){

let td = document.createElement('td');

td.innerHTML = xhr.response.arrayVectors[i].X;

tr.appendChild(td);

}

else if (j === 2){

let td = document.createElement('td');

td.innerHTML = xhr.response.arrayVectors[i].Y;

tr.appendChild(td);

}

else{

let td = document.createElement('td');

td.innerHTML = xhr.response.arrayVectors[i].Z;

tr.appendChild(td);

}

}

tableBody.appendChild(tr);

}

}

else if (xhr.status === 404){

alert('Такого url не существует');

}

else{

alert('Серверу не удалось отправить ответ');

}

}

}

/\*сложение двух векторов\*/

function addVectors() {

let jsonData = JSON.stringify({

'firstNameVector' : document.getElementById('firstName').value,

'secondNameVector' : document.getElementById('secondName').value,

});

let xhr = new XMLHttpRequest();

xhr.open('POST', '/api/add-vectors', true);

xhr.setRequestHeader('Content-type', 'application/json');

xhr.responseType = 'json';

xhr.send(jsonData);

xhr.onload = () => {

if (xhr.status === 200){

if (xhr.response === 0){

document.getElementById('errorPrint').innerHTML = 'Некорректные значения имени вектора (векторов)';

}else{

document.getElementById('sumRes').innerHTML = 'X: ' +xhr.response.plus.X+ ', Y: ' +xhr.response.plus.Y+ ', Z: ' +xhr.response.plus.Z+ ', Name: ' +xhr.response.plus.Name;

}

}

else if (xhr.status === 404) {

alert('Такого url-адреса не существует');

}else{

alert('Серверу не удалось отправить ответ');

}

}

}

function scalarVectors() {

let jsonDate = JSON.stringify({

'firstNameVector' : document.getElementById('firstName').value,

'secondNameVector' : document.getElementById('secondName').value

});

let xhr = new XMLHttpRequest();

xhr.open('POST', '/api/scalar-vectors', true);

xhr.setRequestHeader('Content-type', 'application/json');

xhr.responseType = 'json';

xhr.send(jsonDate);

xhr.onload = () => {

if (xhr.status === 200){

if (xhr.response === 0){

document.getElementById('errorPrint').innerHTML = 'Некорректные значения имени вектора (векторов)';

}else{

document.getElementById('multiplyRes').innerHTML = 'Результат скалярного умножения: ' +xhr.response.scalar;

}

}

else if (xhr.status === 404) {

alert('Такого url-адреса не существует');

}else{

alert('Серверу не удалось отправить ответ');

}

}

}

function updateTable() {

let xhr = new XMLHttpRequest();

xhr.open('GET', '/api/update-table', true);

xhr.setRequestHeader('Content-type', 'application/json');

xhr.responseType = 'json';

xhr.send();

xhr.onload = () => {

if (xhr.status === 200){

let tableBody = document.getElementById("tableBody");

tableBody.innerHTML = "";

let tr = document.createElement('tr');

let thName = document.createElement('th');

let thX = document.createElement('th');

let thY = document.createElement('th');

let thZ = document.createElement('th');

thName.innerHTML = 'Название';

thX.innerHTML = 'Координата X';

thY.innerHTML = 'Координата Y';

thZ.innerHTML = 'Координата Z';

tr.appendChild(thName);

tr.appendChild(thX);

tr.appendChild(thY);

tr.appendChild(thZ);

tableBody.appendChild(tr);

for (let i = 0; i < xhr.response.tableValue.length; i++) {

let tr = document.createElement("tr");

for (let j = 0; j < 4; j++) {

if (j === 0){

let td = document.createElement("td");

td.innerHTML = xhr.response.tableValue[i].Name;

tr.appendChild(td);

}

else if (j === 1){

let td = document.createElement('td');

td.innerHTML = xhr.response.tableValue[i].X;

tr.appendChild(td);

}

else if (j === 2){

let td = document.createElement('td');

td.innerHTML = xhr.response.tableValue[i].Y;

tr.appendChild(td);

}

else{

let td = document.createElement('td');

td.innerHTML = xhr.response.tableValue[i].Z;

tr.appendChild(td);

}

}

tableBody.appendChild(tr);

document.getElementById('errorPrint').innerHTML = 'Таблица векторов успешно обновлена';

}

}else{

document.getElementById('errorPrint').innerHTML = 'Не удалось обновить таблицу векторов';

}

}

}