**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по дисциплине **«Разработка приложений для Интернет»**

на тему: **«NodeJs EventLoop»**

Выполнил: студент гр. ИП-31

Кузнецова Е. А.

Принял: ст. преподаватель

Фролова Е. В.

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2024

**Цель работы:** изучить основные понятия EventLoop, возможность по работе с событиями в NodeJs.

**Задание 1.**

Написать скрипт, который наглядно будет демонстрировать работу каждой из фаз EventLoop:

1.1 Создать три таймера SetTimeOut с разным временем срабатывания. При срабатывании таймера выводить время, прошедшее с момента запуска таймера.

1.2 Применить операции ввода-вывода для записи и чтения файла. После окончания операции ввода-вывода выводить время, прошедшее с момента начала операции.

1.3 Добавить вызов SetImmediate. Вывести время, прошедшее с момента вызова.

**Код программы:**

const fs = require('fs');

// Создаем example.txt, если его нет

if (!fs.existsSync('example.txt')) {

fs.writeFileSync('example.txt', 'Пример данных для записи');

console.log("Файл example.txt создан");

} else {

console.log("Файл example.txt уже существует");

}

console.log("Начало выполнения скрипта");

// Таймер 1

const startTime1 = Date.now();

setTimeout(() => {

const timeElapsed = Date.now() - startTime1;

console.log(`Таймер 1: Прошло ${timeElapsed} мс`);

}, 2000);

// Таймер 2

const startTime2 = Date.now();

setTimeout(() => {

const timeElapsed = Date.now() - startTime2;

console.log(`Таймер 2: Прошло ${timeElapsed} мс`);

}, 3000);

// Таймер 3

const startTime3 = Date.now();

setTimeout(() => {

const timeElapsed = Date.now() - startTime3;

console.log(`Таймер 3: Прошло ${timeElapsed} мс`);

}, 4000);

console.log("Начало операции записи файла");

const fileWriteStartTime = Date.now();

fs.writeFile('example.txt', 'Пример данных для записи', (err) => {

if (err) throw err;

const timeElapsed = Date.now() - fileWriteStartTime;

console.log(`Операция записи файла: Прошло ${timeElapsed} мс`);

});

console.log("Начало операции чтения файла");

const fileReadStartTime = Date.now();

fs.readFile('example.txt', 'utf8', (err, data) => {

if (err) throw err;

const timeElapsed = Date.now() - fileReadStartTime;

console.log(`Операция чтения файла: Прошло ${timeElapsed} мс`);

});

console.log("Вызов SetImmediate");

const immediateStartTime = Date.now();

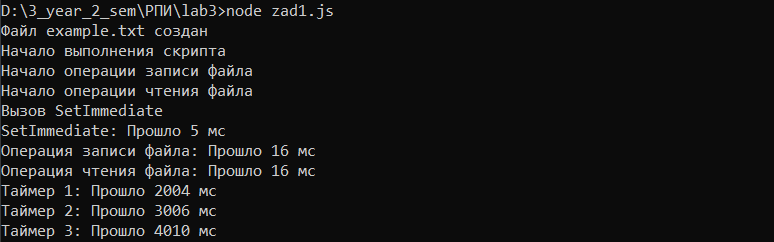
setImmediate(() => {

const timeElapsed = Date.now() - immediateStartTime;

console.log(`SetImmediate: Прошло ${timeElapsed} мс`);

});

**Результат выполения:**

****

**Задание 2.**

Зарегистрировать в системе пользовательское событие. Сгенерировать и обработать пользовательское событие с выводом информации о событии.

**Код программы:**

const { EventEmitter } = require('events');

const myEventEmitter = new EventEmitter();

myEventEmitter.on('myEvent', eventData => {

console.log('Получено событие:');

console.log(`Сообщение: ${eventData.message}`);

console.log(`Дата: ${eventData.date}`);

});

const eventData = {

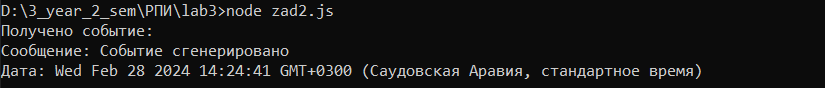
message: 'Cобытие сгенерировано',

date: new Date()

};

myEventEmitter.emit('myEvent', eventData);

**Результат выполнения:**



**Задание 3.**

Промоделировать ситуацию, в которой Вы являетесь продавцом товара (описание товара в лабораторной №2). На обновление списка товара подписать нескольких клиентов, которым будет приходить сообщение при добавлении нового товара или модификации старого. Клиенты должны логгировать информацию от продавца в своих журналах в виде файла с занесением времени.

**Код программы:**

const { EventEmitter } = require('events');

const fs = require('fs');

const path = require('path');

class GraphicsCard {

constructor(producer, name, frequency, memory) {

this.producer = producer;

this.name = name;

this.frequency = frequency;

this.memory = memory;

}

}

class Seller extends EventEmitter {

constructor() {

super();

this.products = [];

this.clientsSubscribed = false

this.logFilePath = path.join(\_\_dirname, 'client\_logs.txt');

this.init();

}

// подписываем клиентов на обновления списка товаров

init() {

if (!this.clientsSubscribed) {

this.on('productAdded', (product) => {

const message = `Добавлен новый товар: ${product.name}`;

this.logEvent(message);

});

this.on('productModified', ({ index, modifiedProduct }) => {

const message = `Товар с индексом ${index} был изменен на: ${modifiedProduct.name}`;

this.logEvent(message);

});

this.clientsSubscribed = true;

}

}

addProduct(product) {

this.products.push(product);

this.emit('productAdded', product);

}

modifyProduct(index, modifiedProduct) {

if (index >= 0 && index < this.products.length) {

this.products[index] = modifiedProduct;

this.emit('productModified', { index, modifiedProduct });

}

}

// Логгирование

logEvent(event) {

// Проверяем существование файла

if (!fs.existsSync(this.logFilePath)) {

// Если файл не существует, создаем его

fs.writeFileSync(this.logFilePath, '');

}

const currentTime = new Date().toISOString();

const logMessage = `${currentTime}: ${event}\n`;

console.log(logMessage);

fs.appendFileSync(this.logFilePath, logMessage);

}

}

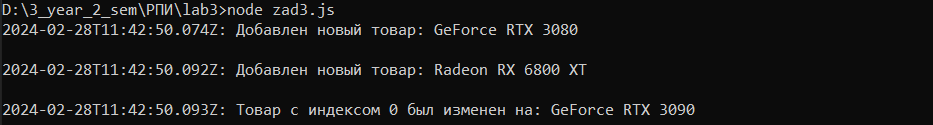
const seller = new Seller();

seller.addProduct(new GraphicsCard('NVIDIA', 'GeForce RTX 3080', '1.44 GHz', '10 GB'));

seller.addProduct(new GraphicsCard('AMD', 'Radeon RX 6800 XT', '2.25 GHz', '16 GB'));

seller.modifyProduct(0, new GraphicsCard('NVIDIA', 'GeForce RTX 3090', '1.70 GHz', '24 GB'));

**Результат выполнения:**



**Вывод:** в результате выполнения лабораторной работы мноюизучены основные понятия EventLoop, возможность по работе с событиями в NodeJs.