МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Полоцкий государственный университет имени Евфросинии Полоцкой»

Факультет информационных технологий

Кафедра технологий программирования

Лабораторная работа №11

по дисциплине: «Веб-технологии»

на тему: «Установка и настройка Node.js. Создание первого HTTP-сервера»

|  |  |
| --- | --- |
| Выполнил | Студент гр. 23-ИТ-2  Вечерский М.И. |
|  |  |
| Проверил | преподаватель-стажер кафедры ТП  Якубович Е.О. |

Полоцк 2025

**Цель работы**: Целью данной лабораторной работы является ознакомление с процессом установки и настройки среды выполнения Node.js, изучение основ модульной системы Node.js, а также получение практических навыков по созданию базового HTTP-сервера с использованием встроенного модуля http и понимание принципов обработки HTTP-запросов и формирования ответов.

**Вариант 4**

**Задание:**

Задание 1: Базовый HTTP-сервер "Hello World"

Создать новый Node.js проект.

Написать JavaScript-скрипт, который использует встроенный модуль http для создания HTTP-сервера.

Сервер должен слушать входящие соединения на порту 3000.

На любой входящий GET-запрос сервер должен отвечать строкой "Hello World!" и устанавливать HTTP-статус 200 OK.

Проверить работу сервера, открыв http://localhost:3000 в браузере или используя утилиту curl.

Задание 2: Простая маршрутизация

Расширить функциональность сервера, созданного в Задании 1.

Реализовать простую маршрутизацию на основе URL-пути (req.url):

* Для пути / сервер должен отвечать "Welcome to the homepage!".
* Для пути /about сервер должен отвечать "This is the about page.".
* Для любого другого URL-пути сервер должен отвечать "404 Not Found" и устанавливать HTTP-статус 404.

Проверить работу сервера, обращаясь к различным URL-путям.

Задание 3: Обработка различных HTTP-методов

Расширить сервер из Задания 2.

Добавить обработку POST-запросов:

* Сервер должен отвечать "Data received via POST!" на POST-запросы к маршруту /submit.
* Для других HTTP-методов (кроме GET и POST) на маршруте /submit сервер должен отвечать "Method Not Allowed" и устанавливать HTTP-статус 405.

**Ход работы**

В ходе выполнения данной работы была создан сервер со следующими эндпоинтами «/», «/about», «/submit». Реализации программы представлена в листинге 1.

Листинг 1 – Сервер

const server = http.createServer((req, res) => {

switch(req.url){

case '/':

res.statusCode = 200

res.setHeader("Content-Type", 'text/plain')

Продолжение листинга 1

res.end('Welcome to the homepage!\n')

break

case '/about':

res.statusCode = 200

res.setHeader("Content-Type", 'text/plain')

res.end('This is the about page.\n')

break

case '/submit':

if (req.method === 'POST') {

res.statusCode = 200

res.setHeader("Content-Type", 'text/plain')

res.end('Data received via POST!\n')

} else {

res.statusCode = 405

res.setHeader("Content-Type", 'text/plain')

res.end('Method Not Allowed.\n')

}

break

default:

res.statusCode = 404

res.setHeader("Content-Type", 'text/plain')

res.end('404 Not Found.\n')

break

}

});

Хочется отметить в качестве архитектурного решения для решиения определения маршрутов и отлова ошибок был выбрана оператор switch тем самым код стал более структурируемым и легким для расширения функционала.

Также для того чтобы сервер заработал ему необходимо назначить порт. (Листинг 2)

Листинг 2 – Установка порта для сервера

server.listen(port, hostname , () => {

console.log(`Server running at http://${hostname}:${port}/`);

});

**Результат работы**

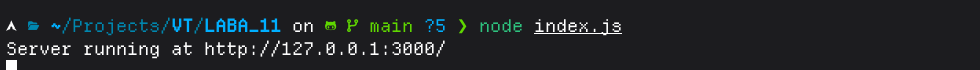
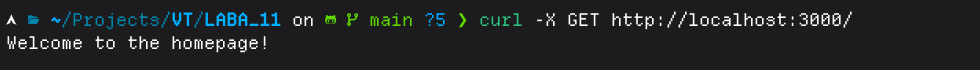
Для проверки работы сервера необходимо сначало его запустить. При запуске сервер сообщает в консоле о том что он работает и где.(Рисунок 1)

Рисунок 1 – Запуск сервера

После уже можно отправлять запросы. Поробуем обратиться к эндпоинту «/». (Рисунке 2)

Рисунок 2 – Изменение фона

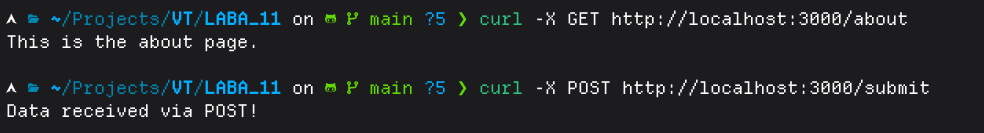
Теперь обратимся к эндпоинтам «/about», «submit». Результат представлен на рисунке 3.

Рисунок 3 – результат конвертации

Как видно программа все правильно обрабатывает.

Вывод: В ходе выполнения данной работы был настроен сервер на Node.js. При выполнении были сделаны выводы, что Node.js является очень удобным инструментом для быстрого создания простых серверов. Также получены некоторые понимание необходимости таких важных компонентов как коды возврата.

**Контрольные вопросы**

Контрольные вопросы

1. Что такое Node.js и для чего он используется в контексте веб-разработки?

Node.js представляет собой кроссплатформенную среду выполнения JavaScript, предназначенную для серверной разработки.

2. Объясните ключевые различия между CommonJS и ECMAScript Modules

(ESM) в Node.js.

Различие в синтаксите в CommonJS зависисмости подключаются через require() и module.exports. В свою очередь в ESM для этого используется import и export. Также CommonJS загружает зависимости синхронно, а ESM асинхронно.

3. Какова роль объектов req (http.IncomingMessage) и res (http.ServerResponse)

в обработке HTTP-запросов и формировании ответов на нативном Node.js сервере?

Объект req это запрос к серверу который содержит URL, метод для выполнения, заголовки и тело. А объект res это ответ сервера на запрос он в свою очередь содержит код ответа, заголовки, тело.

4. Как можно получить URL-путь и HTTP-метод входящего запроса в нативном Node.js сервере? Приведите примеры кода.

Листинг 3 – Код для получения метода и URL

const http = require(«http»);

http.createServer((req, res) => {

console.log(req.method);

Продолжение листинга 3

console.log(req.url);

res.end(«»);

});

5. Почему важно устанавливать правильный HTTP-статус код в ответе сервера (например, 200 OK, 404 Not Found)?

Код ответа необходим для логирования работы сервера и понимании что происходит с запросами и ответами. Поэтому это очень важный элемент работы сервера так как при помощи них можно понять ту ли информацию вернул сервер или правильно ли обработал её.