**IP Lab**

**Experiment - 10**

Name: Shashwat Tripathi

Class: D15A

Roll no: 64

**Aim:** Experiment to study basics of Node.js and Express.

**Theory:**

**Basics of Node.js:**

**1. Node.js:** It is an open-source, server-side runtime environment built on Chrome's V8 JavaScript engine. It allows you to run JavaScript code on the server, making it a popular choice for building web applications, APIs, and more.

**2. Modules in Node.js:** Node.js uses a module system that allows you to organize your code into reusable and independent modules. You can create your own modules or use built-in modules like 'fs' (File System) and 'http' for various tasks.

**3. REPL (Read-Eval-Print Loop):** Node.js comes with a built-in REPL that allows you to interactively run JavaScript code. It's useful for testing code snippets and debugging.

**Routing in Node.js:**

**1. HTTP Module:** Node.js provides an 'http' module for handling HTTP requests and responses. You can create an HTTP server and define how it should respond to different routes and HTTP methods.

**2. Routing:** Routing in Node.js involves mapping specific URLs (routes) to functions or handlers that will be executed when a client makes a request to that URL. You can achieve this by parsing the URL and choosing the appropriate action based on it.

**Basics of Express:**

**1. Express.js:** It is a popular web application framework for Node.js. Express simplifies the process of building web applications by providing a set of features and middleware to handle common tasks.

**2. Request and Response Objects in Express:**

- Request (req): The request object contains information about the incoming HTTP request, including headers, parameters, and data sent by the client.

- Response (res): The response object is used to send a response to the client. You can set response headers, send data, and control the response status.

**3. Routing in Express:** Express makes routing straightforward with its built-in routing capabilities. You can define routes using HTTP methods (GET, POST, PUT, DELETE) and specify the route path and handler function.

**Programs:  
NODEJS CODE**  
const http = require('http');

const fs = require('fs');

const url = require('url');

const stream = require('stream');

const buffer = require('buffer');

const path = require('path');

// Create an HTTP server

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

// Parse the request URL

const parsedUrl = url.parse(req.url, true);

const pathname = parsedUrl.pathname;

const query = parsedUrl.query;

// Web Module - Serving HTML content

if (pathname === '/') {

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

res.write('<h1>Hello, Node.js Web Module!</h1>');

res.end();

}

// Network Module - Simple server to return JSON data

if (pathname === '/api/data') {

res.writeHead(200, { 'Content-Type': 'application/json' });

const data = { message: 'This is a JSON response' };

res.end(JSON.stringify(data));

}

// File System - Reading a file and streaming its content

if (pathname === '/readfile') {

const filePath = 'sample.txt';

const readStream = fs.createReadStream(filePath);

readStream.on('open', () => {

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

readStream.pipe(res); // Stream the file content to the response

});

readStream.on('error', (err) => {

res.writeHead(500, { 'Content-Type': 'text/plain' });

res.end('Error reading the file');

});

}

// Buffer and Stream - Creating a buffer and streaming it

if (pathname === '/bufferstream') {

const bufferData = Buffer.from('This is a buffer and stream example');

const bufferStream = new stream.PassThrough();

bufferStream.end(bufferData);

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

bufferStream.pipe(res);

}

// Routing - Handling undefined routes

if (pathname === '/undefined') {

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

res.end('Page not found');

}

});

// Start the server

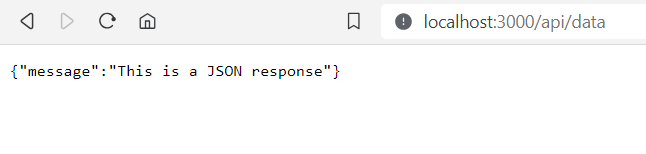
const port = 3000;

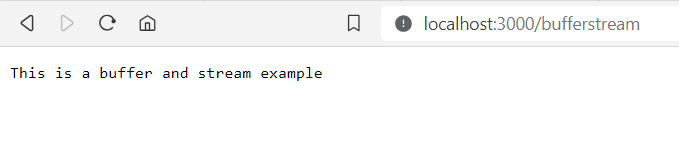
server.listen(port, () => {

console.log(`Server is running on http://localhost:${port}`);

});







**EXPRESS CODE:**

Program reading values from request and response object.

Express Routing.

const express = require("express");

const app = express();

const port = 4000;

// Middleware to log the request URL

app.use((req, res, next) => {

console.log(`Received a ${req.method} request at ${req.url}`);

next();

});

// Route to handle a GET request

app.get("/hello/:name", (req, res) => {

const name = req.params.name; // Reading a parameter from the URL

res.send(`Hello, ${name}!`);

});

// Route to handle a POST request

app.post("/submit", (req, res) => {

const data = req.body; // Reading data from the request body

res.json({ message: "Data received successfully", receivedData: data });

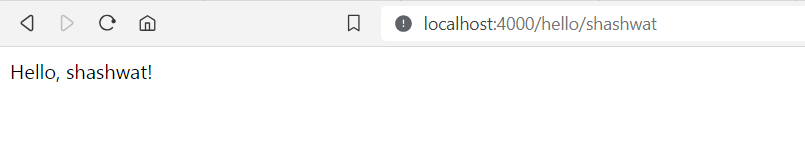
});

// Start the server

app.listen(port, () => {

console.log(`Server is listening on port ${port}`);

});



**Conclusion:** Thus, we have studied the basics of Nodejs and Express.