**Exercise 2: Using HTTP in Node.js**

Node.js provides the ability to create both a powerful HTTP server and client through the http module. Here is a step-by-step guide on how to use HTTP in Node.js.

# Create HTTP Server

## Step 1: Create a New Node.js Project

Create a new directory for the project and initialize the Node.js project:

mkdir my-http-server

cd my-http-server

npm init -y

## Step 2: Create an HTTP Server

Create a new file named server.js

In server.js, enter the following code to create a simple HTTP server:

const http = require('http');

const hostname = '127.0.0.1';

const port = 3000;

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

    res.statusCode = 200;

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

    res.end('Hello World\n');

});

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

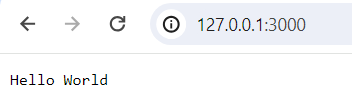
    console.log(`Server running at http://${hostname}:${port}/`);

});

## Step 3: Run Server

Open terminal and run server: node server.js

Open your browser and access http://127.0.0.1:3000, you will see the text "Hello World".



# Create HTTP Client

Node.js also allows you to make HTTP requests to other servers. This is useful when you want to consume APIs or access data from other websites.

## Step 1: Create File Client

Create a new file named client.js

## Step 2: Create HTTP GET Request

In client.js, enter the following code to make an HTTP GET request to the server you just created:

const http = require('http');

const options = {

    hostname: '127.0.0.1',

    port: 3000,

    path: '/',

    method: 'GET'

};

const req = http.request(options, res => {

    console.log(`statusCode: ${res.statusCode}`);

    res.on('data', d => {

        process.stdout.write(d);

    });

});

req.on('error', error => {

    console.error(error);

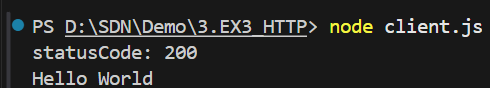
});

req.end();

## Step 3: Run Client

While the server is still running, open a new terminal and run: node client.js

You will see the response from the server printed in the terminal.



# Use Third Module to Work with HTTP

While Node.js's basic http module is powerful, it can be quite low-level and direct. For real applications, you may want to use third-party libraries like axios or node-fetch to make HTTP requests.

## Install axios:

npm install ax ios

## Use axios to make GET requests:

const axios = require('axios');

axios.get('http://127.0.0.1:3000')

    .then(response => {

        console.log(response.data);

    })

    .catch(error => {

        console.log(error);

    });

By using the http module along with third-party libraries, you can create powerful Node.js applications, from simple HTTP servers to complex API consuming clients.

