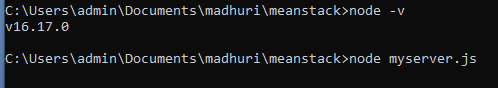
**AIM:Course Name: Node.js Module Name: How to use Node.js Verify how to execute different functions successfully in the Node.js platform.**

**Description:**

To check whether Node.js is installed or not in your machine, open the Node command prompt and check the Node.js version by

The flag -v will display the version of Node.js installed in the machine.

Node.js also provides a package manager called NPM(Node Package Manager) which is a collection of all open-source JavaScript libraries. It helps in installing any library or module into your machine.



**Step 1:** Create a folder NodeJS in D drive and create a new JavaScript file, **first.js**inside the folder.Type the below code inside the JavaScript file.

**Step 2:** Navigate to the created NodeJS folder in the NodeJS command prompt and execute the JavaScript file, first.js using the **node first.js** command.

**Step 3:** After the successful interpretation of the code, we can see the output in the Node.js command prompt.

**Program:**

function tester() {

var m=10;

var message;

if (m=10) {

message = "m value is 10";

} else {

message = "m value is not 10";

}

console.log(message);

}

tester();

**Output:**

****

**AIM:Course Name: Node.js Module Name: Create a web server in Node.js Write a program to show the workflow of JavaScript code executable by creating web server in Node.js.**

**Description:**

Using require() and createServer() method

Running a web server in myserver.js

**Step 1:** Create a new JavaScript file **httpserver.js** and include the HTTP module.

**Step 2:** Use the createServer() method of the HTTP module to create a web server.

**Step 3:** Save the file and start the server using the **node**command. When the file executes successfully, we can observe the following output in the console.

**Program:**

const http = require("http");

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

res.write("Hello World! I have created my first server!");

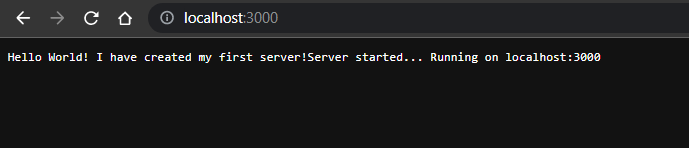
res.end('Server started... Running on localhost:3000');

});

server.listen(3000);

**Output:**





**AIM: Course Name: Node.js Module Name: Modular programming in Node.js Write a Node.js module to show the workflow of Modularization of Node application**

**Description:**

Modularization is a software design technique in which the functionality of a program is separated into independent modules, such that each module contains the desired functionality.

Advantages of modularization:

**Readability:**Modular code highly organizes the program based on its functionality. This allows the developers to understand what each piece of code does in the application.

**Easier to debug:** When debugging large programs, it is difficult to detect bugs. If a program is modular, then each module is discrete, so each module can be debugged easily by the programmer.

**Reusable Code:**Modular code allows programmers to easily reuse code to implement the same functionality in a different program. If the code is not organized modularly into discrete parts, then code reusability is not possible.

**Reliability:**Modular code will be easier to read. Hence it will be easier to debug and maintain the code which ensures smoother execution with minimum errors.

**Program:**

**Module.js**

exports.authenticateUser = (username, password) => {

if (username === "admin" && password === "admin") {

return "Valid User";

} else return "Invalid User";

};

**Exp6c.js**

const http = require("http");

var dbmodule = require("./module");

var server = http.createServer((request, response) => {

result = dbmodule.authenticateUser("admin", "admin");

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

response.end("<html><body><h1>" + result + "</h1></body></html>");

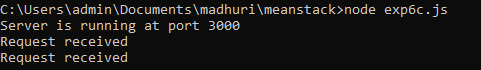
console.log("Request received");

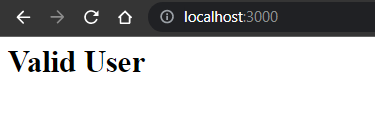
});

server.listen(3000);

console.log("Server is running at port 3000");

**Output:**

****

****