Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

A.Y.: 2022-23 **Class:** S.Y.B.Tech. **Sem:** IV **Sub:** Web Engineering

Experiment 5
(Server-Side Scripting)

Name: Nishant Golakiya
Sap: 60009220150
Batch: D2-2

Aim: Implement Server side scripting using Nodejs

Lab Assignments to complete in this session

1. **Installation and Configuration of Node.js server**
2. **Export functions and variables in module**
3. **Program based on inbuilt functions in Node.js e.g. os, path, fs**
4. **Create imports the "http" module and uses it to create a server**
5. **Create a text file named input.txt with the following content**

Theory:

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

Where to Use Node.js?

Following are the areas where Node.js is proving itself as a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications

- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Where Not to Use Node.js?

- It is not advisable to use Node.js for CPU intensive applications.

Installing Node

In order to use *Express* you will first have to install *Nodejs* and the Node Package Manager (NPM)

1. Install NodeJs:
   A. Go to https://nodejs.org/en/
   B. Select the button to download the LTS build that is "Recommended for most users".Install Node by double-clicking on the downloaded file and following the installation prompts.
2. Testing Nodejs and NPM installation

```
> node -v
v12.18.4
```

3. Creating function and displaying message on console

```
function intro(msg)
{
   console.log(msg)
}
intro("Good Morning")
```

4. Export functions and variables in module

This is **intro.js** file

```
function intro(msg)
{
   console.log(msg)
}
module.exports.intro = intro
```

This is **main.js** file

```
var p=require('./intro.js')
p.intro("Module called")
```

Run Program

```
>node main.js
  Module called
```

Node.js Built-in Modules

Node.js has a set of built-in modules which you can use without any further installation.

Here is a list of the built-in modules of Node.js version 6.10.3:

Module	Description
buffer	To handle binary data
events	To handle events
fs	To handle the file system
http	To make Node.js act as an HTTP server
https	To make Node.js act as an HTTPS server.
net	To create servers and clients
os	Provides information about the operation system
path	To handle file paths

File System Methods

Method	Description
access()	Checks if a user has access to this file or directory
accessSync()	Same as access(), but synchronous instead of asynchronous
appendFile()	Appends data to a file
chmod()	Changes the mode of a file
chown()	Changes the owner of a file
close()	Closes a file
open()	Opens a file
read()	Reads the content of a file
readdir()	Reads the content of a directory
readFile()	Reads the content of a file
realpath()	Returns the absolute pathname
rename()	Renames a file

rmdir()	Removes a directory
stat()	Returns the status of a file
truncate()	Truncates a file
unlink()	Removes a link
write()	Writes buffer to a file
write()	Writes data to a file

The File System module provides a way of working with the computer's file system.

The syntax for including the File System module in your application:
```
var fs = require('fs');
```

1. **Open a file, and output the content:**

```
var fs = require('fs');
  fs.readFile('demofile.txt', 'utf8', function(err, data) {
  if (err) throw err;
  console.log(data);
});
```

Node development environment up and running on computer that can be used for creating Express web applications. Also seen how NPM can be used to import Express into an application.

Lab Assignments to complete in this session

6. **Installation and Configuration of Node.js server**
7. **Export functions and variables in module**

Code:

```
const myModule = require('./myModule');
```

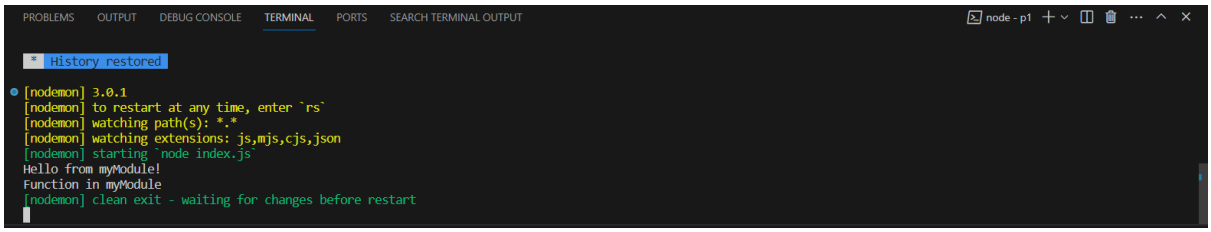
```
console.log(myModule.myVariable);
myModule.myFunction();
```

```
const myVariable = 'Hello from myModule!';

const myFunction = () => {
    console.log('Function in myModule');
};

module.exports = {
    myVariable,
    myFunction
};
```

Output:



8. Program based on inbuilt functions in Node.js e.g. os, path, fs

Code:

```
const fs = require('fs');

fs.readFile('input.txt', 'utf8', (err, data) => {
    if (err) {
        console.error(err);
        return;
    }
    console.log(data);
});
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH TERMINAL OUTPUT

PS D:\web\Node.js> cd p2
PS D:\web\Node.js\p2> nodemon
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Web Engineering Lab is giving self-learning content
to teach the world in simple and easy way!!!!!

[nodemon] clean exit - waiting for changes before restart
```

9. Create imports the "http" module and uses it to create a server

Code:

```
const http = require('http');

const server = http.createServer((req, res) => {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello, this is a basic HTTP server!');
});

server.listen(3000, '127.0.0.1', () => {
    console.log('Server is running at http://127.0.0.1:3000/');
});
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH TERMINAL OUTPUT

[nodemon] clean exit - waiting for changes before restart
PS D:\web\Node.js\p2> cd ..
PS D:\web\Node.js> cd p3
PS D:\web\Node.js\p3> nodemon
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server is running at http://127.0.0.1:3000/
```

10. Create a text file named input.txt with the following content

Web Engineering Lab is giving self-learning content

to teach the world in simple and easy way!!!!!

Create a js file to read this .txt file