DOP: / /2023 **DOS: / /2023**

Experiment No: 09

Title: Modules in Node.js (Networking, File system, Web module).

Theory:

🔷Networking Module:

Node.js Net

Node.js provides the ability to perform socket programming. We can create chat application or communicate client and server applications using socket programming in Node.js. The Node.js net module contains functions for creating both servers and clients.

Node.js Net Example

In this example, we are using two command prompts:

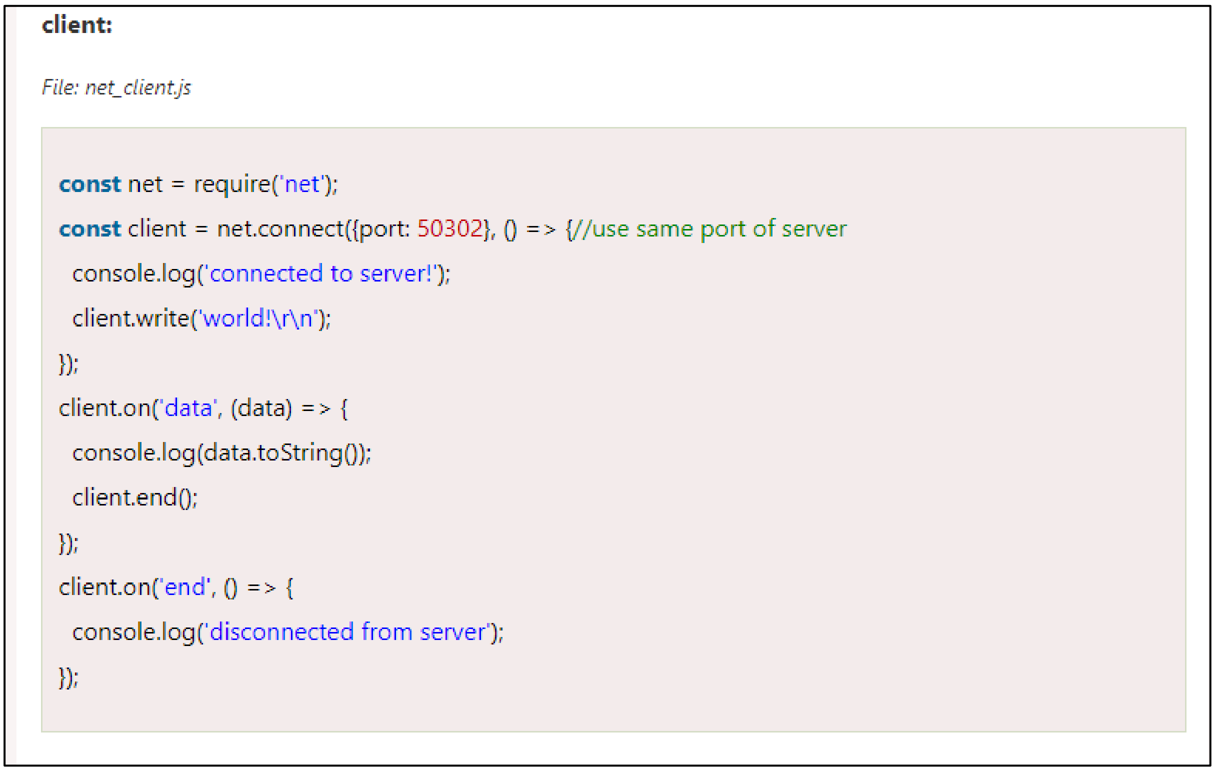
1. Node.js command prompt for server.
2. Window's default command prompt for client.

Syntax: var net = require('net');

server:

File: net_server.js

```
const net = require('net');
var server = net.createServer((socket) => {
  socket.end('goodbye\n');
}).on('error', (err) => {
  // handle errors here
  throw err;
});
// grab a random port.
server.listen(() => {
  address = server.address();
  console.log('opened server on %j', address);
});
```



client:

File: net_client.js

```
const net = require('net');
const client = net.connect({port: 50302}, () => {//use same port of server
  console.log('connected to server!');
  client.write('world!\r\n');
});
client.on('data', (data) => {
  console.log(data.toString());
  client.end();
});
client.on('end', () => {
  console.log('disconnected from server');
});
```

🔷Node.js File System (FS):

In Node.js, file I/O is provided by simple wrappers around standard POSIX functions. Node File System (fs) module can be imported using following syntax:

```
Syntax: var fs = require("fs")
```

Node.js FS Reading File

Every method in fs module has synchronous and asynchronous forms.

Asynchronous methods take a last parameter as completion function callback. Asynchronous method is preferred over synchronous method because it never blocks the program execution where as the synchronous method blocks.

Let's take an example:

Create a text file named "input.txt" having the following content.

File: input.txt

```
Hello NodeJS
```

Let's take an example to create a JavaScript file named "main.js" having the following code:

File: main.js

```
var fs = require("fs");
// Asynchronous read
fs.readFile('input.txt', function (err, data) {
  if (err) {
      return console.error(err);
  }
  console.log("Asynchronous read: " + data.toString());
});
// Synchronous read
var data = fs.readFileSync('input.txt');
console.log("Synchronous read: " + data.toString());
console.log("Program Ended");
```

Node.js Open a file

Syntax :fs.open(path, flags[, mode], callback)

Parameter explanation:

Following is the description of parameters used in the above syntax:

- **path**: This is a string having file name including path.
- **flags**: Flag specifies the behavior of the file to be opened. All possible values have been mentioned below.
- **mode**: This sets the file mode (permission and sticky bits), but only if the file was created. It defaults to 0666, readable and writeable.
- **callback**: This is the callback function which gets two arguments (err, fd).

◆**Node.js Web Module**:

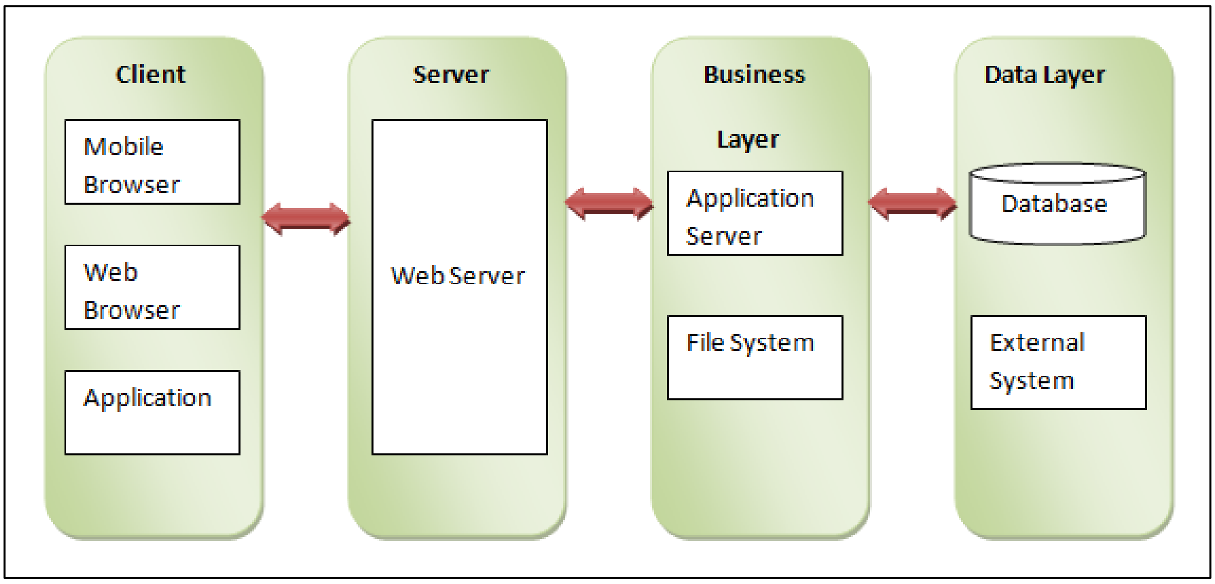
What is Web Server:

Web Server is a software program that handles HTTTP requests sent by HTTP clients like web browsers, and returns web pages in response to the clients. Web servers usually respond with html documents along with images, style sheets and scripts.

Web Application Architecture

A web application can be divided in 4 layers:

- **Client Layer**: The Client layer contains web browsers, mobile browsers or applications which can make HTTP request to the web server.
- **Server Layer:** The Server layer contains Web server which can intercepts the request made by clients and pass them the response.
- **Business Layer**: The business layer contains application server which is utilized by web server to do required processing. This layer interacts with data layer via data base or some external programs.
- **Data Layer**: The Data layer contains databases or any source of data.



Creating Web Server using Node.js syntax:

1. var http = require('http');
2. var fs = require('fs');
3. var url = require('url');
4. // Create a server
5. http.createServer(function (request, response) {
6. }

Conclusion: - We Successfully implement and understanding Modules in Node.js