Programme : Bachelor of Science(Third Year)

Subject : Computer Science

Semester : VI

Course Code : CSC109

Course Title: Full Stack Web Development

Unit II : Node.js

Module Name: Module, Module types: Core

Modules, Local Modules,

Module.Exports

Presenter : Nilesh R. Natekar

Associate Professor in Computer Science

Govt. College, Sanquelim- Goa

NOTES

Module

- ✓ Module in Node.js is a simple or complex functionality organized in single or multiple JavaScript files which can be reused throughout the Node.js application.
- ✓ Each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope. Also, each module can be placed in a separate .js file under a separate folder.

Module Types

Node.js includes three types of modules:

- ✓ Core Modules
- ✓ Local Modules
- ✓ Third Party Modules

Core Modules

- ✓ Node.js is a light weight framework.
- ✓ The core modules include bare minimum functionalities of Node.js.
- ✓ These core modules are compiled into its binary distribution and load automatically when Node.js process starts.

- ✓ However, you need to import the core module first in order to use it in your application.
- ✓ **Core Modules**

Core Module	Description
http	http module includes classes, methods and events to create Node.js http server.
url	url module includes methods for URL resolution and parsing.
querystring	querystring module includes methods to deal with query string.
path	path module includes methods to deal with file paths.
fs	fs module includes classes, methods, and events to work with file I/O.
util	util module includes utility functions useful for programmers.

Loading Core Modules

- ✓ In order to use Node.js core or NPM modules, you first need to import it using require() function as shown below.

  var module = require('http');

- ✓ As per above syntax, specify the module name in the require() function. The require() function will return an object, function, property or any other JavaScript type, depending on what the specified module returns.

Example Code

- ✓ The following example demonstrates how to use Node.js http module to create a web server.

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

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

        // write code here
```

```
});

server.listen(5000);
```

Local Modules

- ✓ Local modules are modules created locally in your Node.js application.
- ✓ These modules include different functionalities of your application in separate files and folders.
- ✓ You can also package it and distribute it via NPM, so that Node.js community can use it.

Example Code

- ✓ Let's write simple logging module which logs the information, warning or error to the console.
- ✓ This code should be kept in a separate file. Eg. Log.js

```
var log={
        info: function(info){
                console.log('Info: ' + info); }
        warning: function(warning){
                console.log('Warning: ' + warning); }
        error: function(error){
                console.log('Error: ' + error); }
        };
module.exports = log;
```

Loading Local Modules

- ✓ Create app.js file as follows:

```
var myLogModule = require('./Log.js');

myLogModule.info('Node.js Started');
```

- ✓ Run the above code using – node app.js and we will get the following output:

```
Info: Node.js Started
```

module.exports

- ✓ The module.exports is a special object which is included in every JavaScript file in the Node.js application by default.

- ✓ The module is a variable that represents the current module, and exports is an object that will be exposed as a module.
- ✓ So, whatever you assign to module.exports will be exposed as a module.

Export Literals

- ✓ As mentioned earlier, exports is an object. So it exposes whatever you assigned to it as a module. For example, if you assign a string literal then it will expose that string literal as a module.

Message.js

```
module.exports = "Hello World';
```

App.js

```
var msg = require('./Messages.js');

console.log(msg);
```

Export Object

- ✓ Message.js

```
exports.simpleMessage = 'Hello World';
```

OR

```
module.exports.simpleMessage='Hello World';
```

App.js

```
var msg = require('./Messages.js');

console.log(msg.simpleMessage);
```

- ✓ In the same way as above, you can expose an object with function.
- ✓ Log.js

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

App.js

```
var msg = require('./Messages.js');

msg.log('Hello World');
```