Advanced Node.JS Modules

After completing this reading, you will be able to describe the three types of Node.js modules. You will also be able to explain what several prominent core modules can be used for.

Libraries are the same thing as modules in regard to Node.js. Libraries contain code that has been developed that can be reused throughout an application. There are three types of modules: core, local, and third-party.

Core Node.js modules form a minimal library. They contain the minimal functionality needed to develop Node.js applications.

Local modules are the next type of Node.js module. Local modules are the modules written by you and the development team as part of creating your Node.js application.

Third-party modules are available online and have been created by the back-end Node.js community. These libraries are available to use as stated per their licenses. Many third-party modules are either in the public domain, which does not require a license, or they are open source. Open source resources are usually governed by the "copyleft" license, which allows the developer to use and modify the code but also requires the developer to share their work under that same license.

The most important of the core modules are http, path, fs, os, util, url, and querystring. Let's briefly discuss each of these and provide code examples.

The http module provides methods to transfer data over HTTP. To include the http module in your application, you should require it.

Here is sample code that creates an instance of a server using the http module. This code makes use of the http.createServer() method to create the server instance.

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

```
1. let http = require('http');
2. http.createServer(function (req, res) {
3.     res.write('hello from server');//write a response to the client
4.     res.end();//end of response from server
5. }).listen(6000);//the server instance listens for http requests on port 6000
```

Copied!

The fs module is used to interact with a file system. It does not need to be installed because it is part of the Node.js core and can simply be required. The following code sample reads a local file synchronously using the fs module and prints the file contents to the console.

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

```
1. // sample code to read a file synchronously using the fs.readFile() method
2. const fs = require('fs');
3. const data = fs.readFile('sample.txt', 'utf8');
4. // print contents of the file "sample.txt" to console
```

```
5. console.log(data);
```

Copied!

The fs module can also be used for input and output, known as I/O. The fs module methods can be used to retrieve information from or write data to an external file.

```
1. 1
2. 2
3. 3
```

```
1. const fs = require('fs');
2. const data = fs.readFileSync('/content.md'); // blocks here until file is read
3. console.log(data);//writes data in the content.md file to the console
```

Copied!

The OS module provides methods to retrieve information from the operating system that the application is running on and interact with it. This is sample code from the OS module that gets the computer's platform and architecture.

```
1. 1
2. 2
3. 3
```

```
1. let os = require('os');
2. console.log("Computer OS Platform Info : " + os.platform());
3. console.log("Computer OS Architecture Info: " + os.arch());
```

Copied!

The path module allows you to retrieve and manipulate directory and file paths.
The following code retrieves the last portion of a given file path and prints that value to the console:

```
1. 1
2. 2
3. 3
```

```
1. const path = require('path');
2. let result = path.basename('/content/index/home.html');
3. console.log(result); //outputs home.html to the console
```

Copied!

The Node.js util module is intended for internal use for accomplishing such tasks as debugging and deprecating functions. Say you want to debug a program to count the number of iterations in a loop. You could use util.format() method as follows:

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

```
1. let util = require('util');
2. let str = 'The loop has executed %d time(s).';
3. for (let i = 1; i <= 10; i++) {
4.     console.log(util.format(str, i)); //outputs 'The loop has executed i time(s)'
5. }
```

Copied!

The URL module is used to divide up a web address into readable parts. Here is a sample code that returns the value of the "firstName" query object from the given URL.

```
1
2
3
4
5
```

```
const url = require('url');
let webAddress = 'http://localhost:2000/index.html?lastName=Kent&firstName=Clark';
let qry = url.parse(webAddress, true);
let qrydata = qry.query; //returns an object: {lastName: 'Kent', firstName: 'Clark'}
console.log(qrydata.firstName); //outputs Clark
```

Copied!

The querystring module provides methods to parse through the query string of a URL.
For example,

```
1
2
3
```

```
let qry = require('querystring');
let qryParams = qry.parse('lastName=Kent&firstName=Clark');
console.log(qryParams.firstName); //returns Clark
```

Copied!

There are also a number of useful third-party packages for use with Node.js. Some of those include AsyncJS, Axios, and Express. These libraries will be discussed later in the course.

Changelog

Date	Version	Changed by	Change Description
31-10-2022	1.0	K Sundararajan	Initial version created in Markdown format
02-05-2023	1.1	K Sundararajan	Instructions updated for clarity based on learner reviews