Unit 2

Exercises

How to Create a Node.js Module?

A Node.js Module is a .js file with one or more functions.

The syntax to define a function in Node.js module is

```
exports.<function_name> = function (argument_1, argument_2, .. argument_N)
{
   /** function body */
};
```

exports – is a keyword which tells Node.js that the function is available outside the module.

function_name – is the function name using which we can access this function in other programs.

Create a module

```
// Returns addition of two numbers
exports.add = function (a, b) {
    return a+b;
};
 // Returns difference of two numbers
exports.subtract = function (a, b) {
    return a-b;
};
 // Returns product of two numbers
exports.multiply = function (a, b) {
    return a*b;
};
```

```
var calculator = require('./calculator');

var a=10, b=5;

console.log("Addition : "+calculator.add(a,b));
console.log("Subtraction : "+calculator.subtract(a,b));
console.log("Multiplication : "+calculator.multiply(a,b));
```

```
$ node moduleExample.js
Addition : 15
Subtraction : 5
Multiplication : 50
```

Extend or add functions to Node.js module

Include the module

- The first step to extend a module is to include the module itself using require function.

```
var newMod = require('<module_name>');
```

- We have retrieved the module to a variable.

Add function to the module variable

Using variable to the module,newMod , add a new function to it using following syntax.

```
newMod.<newFunctionName> = function(function_parameters) {
   // function body
 };
```

You may add as many number of new functions to the module as per your requirement.

Any modifications to the module variable does not affect the actual module in its original form.

Re-export the module

To re-export the module for the new added functionalities:

```
module.exports = newMod;
```

Now, use the variable to the module, newMod , for calling new functionalities added.

Example

```
// include the module that you like extend
var fs = require('fs');
 // add a new function, printMessage(), to the module
fs.printMessage = function(str){
    console.log("Message from newly added function to the module");
    console.log(str);
}
 // re-export the module for changes to take effect
module.exports = fs
 // you may use the newly added function
fs.printMessage("Success");
```

Steps to Override Function of a Module in Node.js

To override an existing function of a Node.js module, following is a step by step guide.

Step 1: Include the module.

The first step to override a function in a module is to include the module itself using require function.

```
var newMod = require('<module_name>');
```

We have retrieved the module to a variable.

Step 2: Delete function from the module variable.

Using variable to the module,newMod , delete the function from it using following syntax.

```
delete newMod['<function_name>'];
```

Please remember that the changes would be only to the module variable, newMod, but not to the original module itself.

Step 3: Add function, with same name, to the module variable.

Using variable to the module,newMod , add the function with the same name, that we deleted in the previous step, using following syntax.

```
newMod.<function_name> = function(function_parameters) {
        // function body
        };
```

Step 4: Re-export the module.

You have to re-export the module for the overriden functionalities to take effect.

```
        module.exports = newMod;
```

Now, you may use the variable to the module,newMod , for calling the function, and the overridden functionality would be executed.

Example

```
// include the module whose functions are to be overridden
var fs = require('fs');
 // delete the function you would like to override
delete fs['readFile'];
 // add new functional with the same name as deleted function
fs.readFile = function(str){
    console.log("The functionality has been overridden.");
    console.log(str);
}
 // re-export the module for changes to take effect
module.exports = fs
 // you may use the newly overriden function
fs.readFile("sample.txt");
```