Exercise (Instructions): Understanding Node Modules

Objectives and Outcomes

In this exercise, you will create a simple JavaScript application and explore how the Node modules work. First, you will implement a simple quadratic equation solver in JavaScript and see it working correctly using Node. This illustrates the JavaScript runtime capabilities of Node. Thereafter, you will divide the application into several modules in order to explore how Node modules are designed and how they can be leveraged to implement a node application. At the end of this exercise, you will be able to:

- Understand how to write and run JavaScript programs using Node
- Understand about Node modules
- Write JavaScript code as Node modules
- Export Node modules and include them in other JavaScript programs

Writing a Simple Quadratic Equation Solver

- In this example, we create a simple JavaScript program to solve the quadratic equation ax^2+bx+c=0 and find its roots. As we recall from our high school mathematics, in order to solve the equation, we first compute the discriminant = (b^2 - 4ac). The quadratic equation will have real roots if a is not equal to 0, and discriminant is not less than 0. The two roots of the equation are given as root1 = (-b - sqrt(discriminant))/2a and root2 = (-b + sqrt(discriminant))/2a.

- To implement the solver, create a new file named *simplequad.js* with the following JavaScript code:

```
var a = 1, b = 4, c = 1;


var discriminant = function (a, b, c) {
        return (b*b - 4*a*c);
}


var root1 = (-b - Math.sqrt(discriminant(a,b,c)))/(2*a);


var root2 = (-b + Math.sqrt(discriminant(a,b,c)))/(2*a);
```

```
console.log("Roots are "+root1 + "  " + root2);
```

- To execute this program using the Node runtime, type the following at the command prompt:

```
    node simplequad
```

- Upon completion of execution, the following should be displayed in the terminal screen:

```
Roots are -3.732050807568877  -0.2679491924311228
```

Writing Node Modules

- We will now restructure the code into multiple node modules as follows. Fisrt we create a node module for the discriminant. To do this, create a new file named *discriminant.js* and add the following code to this file:

```
module.exports = function(a,b,c) {
   return (b*b - 4*a*c);
}
```

Here by declaring that module.exports = function, we are specifying that this module exports the function that computes the discriminant.

- Next we implement the quadratic equation solver. To do this, we create a file named *quadratic.js* and add the following code to it:

```
var disc = require('./discriminant');

module.exports = function(a,b,c,next) {
    if (a == 0) {
        next(new Error("a should not be zero"));
    }
    else if ( disc(a,b,c) < 0) {
        next(new Error("discriminant is less than zero, no real roots"));
    }
```

```
    else
         return next(null, {
            root1:function() {
                return (-b - Math.sqrt(disc(a,b,c)))/(2*a);
            },
            root2:function() {
                return (-b + Math.sqrt(disc(a,b,c)))/(2*a);
            }
        });
}
```

Note that we use the require ('./discriminant') method in order to specify that this module uses the discriminant module that we created earlier. The .js extension to the file name is assumed by Node.

- This file implements the quadratic equation solver as a module. Hence the use of the module.exports = function.
- Next, we implement a JavaScript program that makes use of the quadratic module. To do this, create a file named *solve.js* and add the following code to it:

```
var quad = require('./quadratic');


quad(1,4,1, function(err,quadsolve) {
        if (err) {
        console.log('Error: ', err);
        }
        else {
                console.log("Roots are "+quadsolve.root1() + "  " + quadsolve.root2(
));
        }
});
```

- Now you can execute the solver by typing "*node solve*" at the command prompt.
- Modify the code to generate the results for a=0 and see that the solver generates the error saying that a is zero.

- Then, change the parameters to a=4, b=2, c=1, then you will note that the discriminant is less than zero. In this case, the error indicating that the discriminant is less than zero is displayed.

Using the "prompt" Node Module

- To check for the availability of the node module named "prompt", type the following at the command prompt:

```
npm search prompt
```

This will list a number of NPM modules with the prompt in the name or description.

- Next we learn to create the *package.json* file using NPM. To do this, type the following at the prompt:

```
npm init
```

Answer the questions that NPM asks to fill in some information to the package.json file.

- To install the "prompt" module, type the following at the command prompt:

```
npm install prompt -S
```

This will install the prompt module and also save this dependency in the package.json file.

- To use the prompt module in the quadratic solver to prompt the user for the coefficients, modify the contents of the *solve.js* file as follows:

```
var quad = require('./quadratic');

var prompt = require('prompt');

prompt.get(['a', 'b', 'c'], function (err, result) {
        if (err) { return onErr(err); }
        console.log('Command-line input received:');
```

```
        console.log('a: ' + result.a);
        console.log('b: ' + result.b);
        console.log('c: ' + result.c);

        quad(result.a,result.b,result.c, function(err,quadsolve) {
        if (err) {
                    console.log('Error: ', err);
                }
                else {
                    console.log("Roots are "+quadsolve.root1() + "  " + quadsolve.r
oot2());
                }
       });
});
```

- Now you can run the solver by typing "*node solve*" at the command prompt.

Conclusions

In this exercise, you learnt about node modules and how you can create your own node modules. You also learnt about fetching and using node modules from npmjs.com and use it within your program.