Chapter 3: Node.js

Exercise 3.1: Hello World

In this exercise, you will create the obligatory first project for any new programming endeavor. In this case, you will create a very simple Node.js application that will display the standard greeting message.

Use Visual Studio Code for this and all other node exercises.

Time: 15 minutes
Format: Programming exercise

1. Navigate to the `RESTfulServices` folder.

2. Create a file named `hello.js`.

3. Type the following inside `hello.js`:

   ```
   console.log("Node says Hello World");
   ```

4. Open a command window in `RESTfulServices`.

5. Type the following inside the command window:

   ```
   node hello.js
   ```

6. Verify that the message is displayed in the command window.

Exercise 3.2: Creating a Server

In this exercise, you will create a very simple server and run that server with Node.

Time: 20 minutes
Format: Programming exercise

1. Return to the `RESTfulServices` folder.

2. Create a new file named `server.js`.

3. Type the following inside `server.js`:

```
let http = require('http');
let server = http.createServer((req, res)
=> {
    res.end('Hello World from the Server');
});
server.listen(8081);
```

4. Open a command window in the `RESTfulServices` folder.

5. Enter the following on the command line:

```
node server.js
```

6. Browse to `http://localhost:8081` in any browser.
   a. Verify the message is displayed in the browser.

7. When you're done, use `Ctrl+C` in the command window to close the server

Exercise 3.3: Returning HTML

In this exercise, you will create a very simple server that returns HTML and run that server with Node.

Time: 20 minutes
Format: Programming exercise

1. Return to the `RESTfulServices` folder.

2. Create a new file named `serverHtml.js`.

3. Type the following inside `serverHtml.js`:

```
let http = require('http');
let server = http.createServer( (req, res)
=> {
    res.writeHead(200, { 'Content-Type':
'text/html'});
    res.write('<h1>Hello Node World</h1>');
    res.end();
});
server.listen(8081);
```

4. Use node to run `serverHtml.js`.

5. Using your browser, verify the server is operating as expected.

6. Use Chrome Developer Tools to verify existence of `<div>` object in the response from the server.

7. When you're done, use `Ctrl+C` in the command window to close the server

Exercise 3.4: Using a Core Module

In this exercise, you will write a Node.js application that loads a core module and uses some of its functionality.

Time: 20 minutes
Format: Programming exercise

1. Create a new file `content.html` in the `RESTfulServices` folder.

2. Add html content that will define a simple (but complete) web page.

3. Save the file.

4. Create a new file named `serverContent.js`.

5. Type the following inside `serverContent.js`:

```
let fs = require('fs');
let http = require('http');
let server = http.createServer( (req, res) => {
    fs.readFile('content.html', (err, fileData) => {
        res.writeHead(200, { 'Content-Type': 'text/html'});
        res.write(fileData);
        res.end();
    })
});
server.listen(8081);
```

6. Use node to run `serverContent.js`.

7. Then browse to `http://localhost:8081`.

8. Verify the html content defined in `content.html` is displayed in the browser.

9. When you're done, use `Ctrl+C` in the command window to close the server.

Exercise 3.5: Using npm

In this exercise, you will use npm to create and initialize the `package.json` file for a new Node application.

Time: 20 minutes
Format: Programming exercise

1. Open a command window in the `RESTfulServices` folder.

2. Run the following command: `npm init`

3. Enter the bold text below at the appropriate prompts, using the `Return/Enter` key to accept defaults for all other prompts:

   package name: (begin) **`hellonpm`**
   description: **`A simple Node server`**
   author: **`Your Name`**

4. Examine the file `package.json` that has been generated in the folder.
   Note: the *start* script calls the command *node server.js*.

5. Edit the `server.js` file.

6. Type the following inside `server.js`:

```
const express = require('express');
const app = express();
app.get('/', (req, res) => {
    res.send('Hello Express');
});
const server = app.listen(8081, () => {
    console.log('App listening on port 8081');
});
```

7. Enter **`npm start`** at the command line.

8. Then in your browser view `http://localhost:8081`.

9. When you're done, use `Ctrl+C` in the command window to close the server.