Creating and importing modules

Welcome to this lab activity

In this lab activity, you will explore the concept of Node modules. To start, the code will be almost identical to the previous lab but, to keep things organised, you will work in another project folder.

What is a module?

A module is essentially a set of functions that you want to include in your application. Think of it as a library: a set of functions or code written by you or someone else which you can import and export across your project.

There are certain modules which are built in inside Node. For example, do you recall the line require('http') in the previous lab? Well, http is a module which lives inside Node and is available for you to use.

Other modules, on the other hand, are not part of Node and you will need to include them inside your project if you want to use them. You will look at including external modules in the next lab when we introduce **Express**.

For now, let's see how you can create your own modules, export them and successively use them in your web server application.

Task 1: Create a new Node.js/NPM project

Let's create a new project in Visual Studio Code.

1. Create a new folder called 03_modules on your computer.
2. Open the folder in Visual Studio Code.
3. Open a Terminal pane in Visual Studio Code.
4. Run the following command in the Terminal pane:

```
npm init
```

Just hit enter to accept the default answers.

Task 2: Create the code to serve a web page

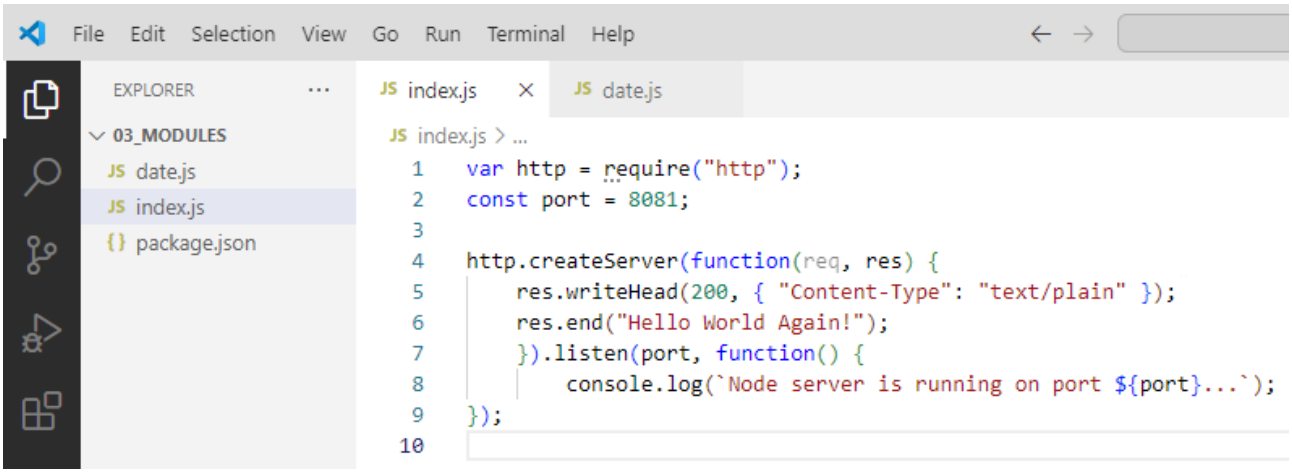
Now you will write the code to serve the web page.

5. Create a file called index.js in the project.
6. Add the following code to the index.js file and remember to save it:

```
var http = require("http");
const port = 8081;

http.createServer(function(req, res) {
   res.writeHead(200, { "Content-Type": "text/plain" });
   res.end("Hello World Again!");
   }).listen(port, function() {
      console.log(`Node server is running on port ${port}...`);
});
```

If you have correctly followed the steps above, your environment should be similar to the one below:



Task 3: Run the code

7. Run the index.js file with the following Terminal command:

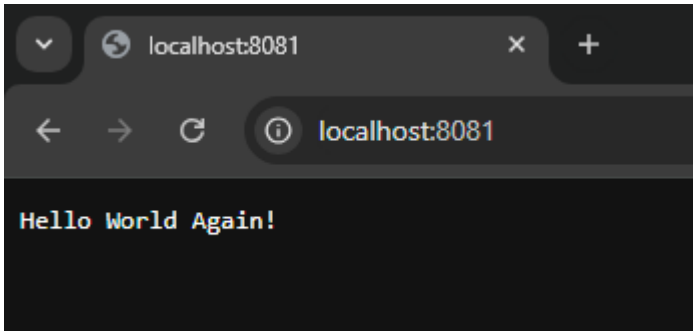
```
node index.js
```

The above command will start a web server running on port 8081.

Task 4: Access your server via HTTP

Now that your code is running and serving your application on port 8081, you can access your web page from a browser!

8. Start up a browser and navigate to http://localhost:8081. If you have correctly followed all the steps, your web browser should show the ‘Hello World Again!’ message.



Task 5: Add a new module to your code

It is now the time to create your own module and import it inside the index.js file.

Let's create a module that returns a date and time object so that you can import it and use it in the index.js file. The goal is to show a message with the current date and time just before the 'Hello Word Again!' message in the web page served by your server.

9. Create a new file called date.js in your project.

10. Add the following code to date.js file and remember to save it:

```
exports.myDateTime = function () {
    return Date();
};
```

The exports keyword makes properties and methods available outside of the module (in this case outside of the date.js script). In this case, you are declaring a new function called myDateTime() and making it available to other scripts in your project.

At this point, you have exported your module, and you only need to require it inside the index.js file to be able to use it.

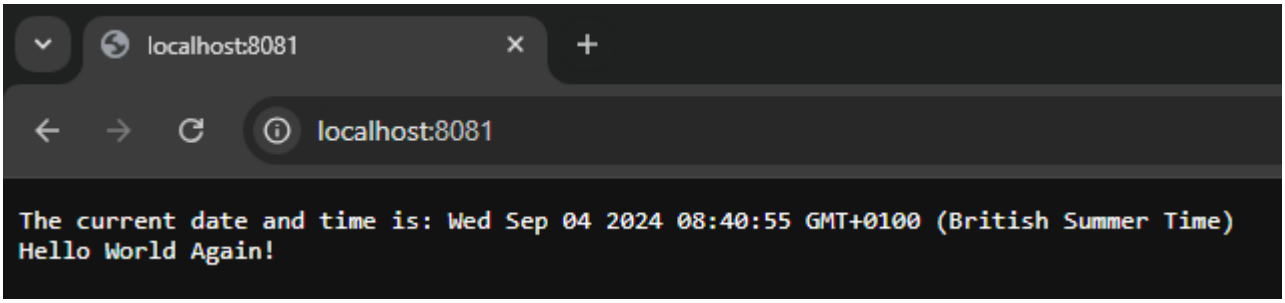
11. Add this line of code just below the 'const port' line in index.js:

```
var dt = require("./date.js");
```

12. In addition, add this line of code to your index.js file:

```
res.write("The current date and time is: "+ dt.myDateTime() + "\n");
```

Can you guess where you should add this piece of code? Remember we would like you to show this in the browser before the 'Hello World again!' message:



13. In order to visualise your latest changes, you will have to both restart your web server (index.js) and also refresh the 'Browser Preview' tab.

Task 6: Exercise

When tackling these lab activities, it's always good to stretch yourself by doing some research and attempting some changes on your own.

- Add another function to date.js. This function, called tomorrow() will retrieve the date and time 24 hours from the current date and time. Display the return from this function in your web application.
- Add another module, called temperature, to your application. This module should have functions that convert between Centigrade and Fahrenheit. Use these functions to display the conversion of 0 degrees Centigrade in Fahrenheit and 0 degrees Fahrenheit in Centigrade.

End of session

Well done on completing this session.

If you have correctly followed all the steps, you should be able to see your web page with both the date, time and the welcome message like in the above picture.