Create, modify and run Node files

Welcome to this lab activity

In the previous lab activity, you installed the Visual Studio Code IDE, Node and NPM. It is now the time for you to write some code through a guided exercise. The main focus is for you to become familiar with the explorer, terminal and editor sections of Visual Studio Code. By the end of this exercise, you should be able to create your own project folder structure, add and modify JavaScript files and run them using the Node.js runtime environment.

The exercise overview

The goal of this exercise is to create a very simple program that prints all the information regarding a particular object in the form of key/value pairs to the terminal. More specifically, you will initialise a `course` object with the following properties:

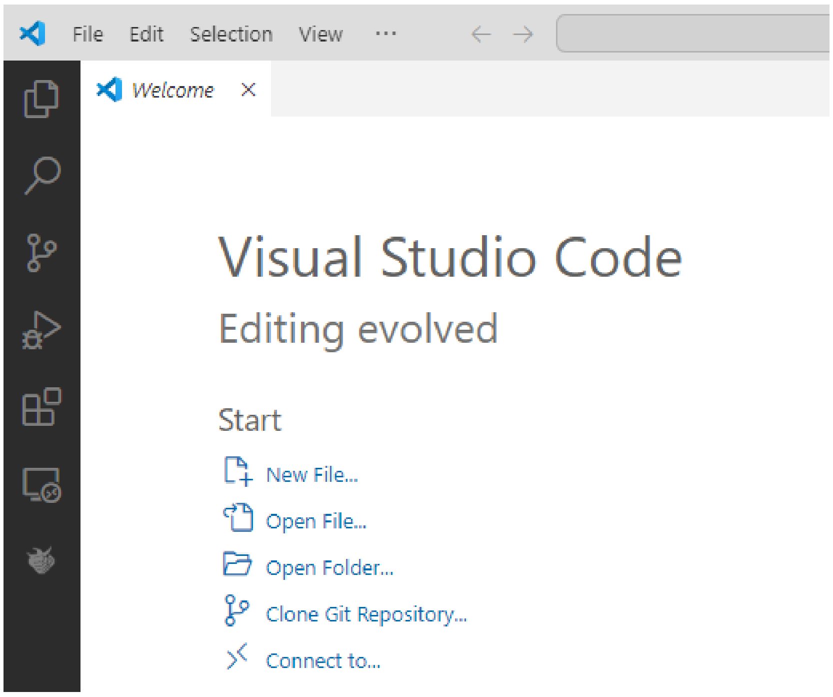
```
let course = {
  name: "Databases, Networks and the Web",
  platform: "Coursera",
  category: "Computer Science"
};
```

The aim is to print all the key/value pairs inside the `course` object to the terminal with the help of a call to a custom `printCourseInformation(course)` function. The function will accept the `course` object as a parameter and will print all the information to the terminal as shown below:

```
name: Databases, Networks and the Web
platform: Coursera
category: Computer Science
```

Start Visual Studio Code

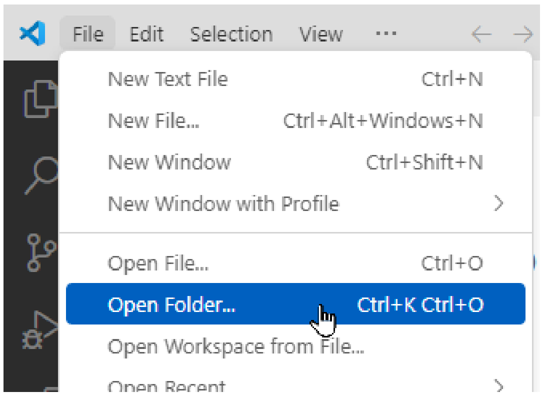
1. Start up the Visual Studio Code on your computer. You should see a welcome screen that looks like this:



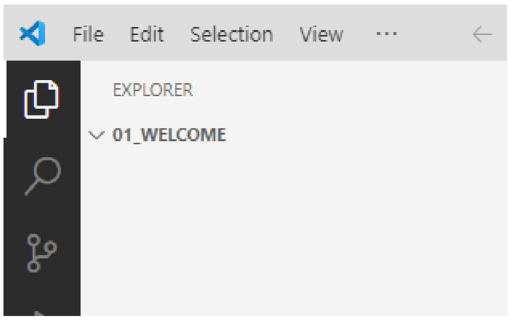
Create a project folder

When starting a new project, create a folder on your computer to hold all the files associated with that project. Use your operating system's file manager to do this, for example Finder on a Mac or Windows Explorer on Windows. Alternatively, you can do this from your computer's Terminal application.

2. Create a new folder called `01_welcome`.
3. In Visual Studio Code, open the folder by going to File > Open folder:



You should see the folder open in the explorer in Visual Studio Code:

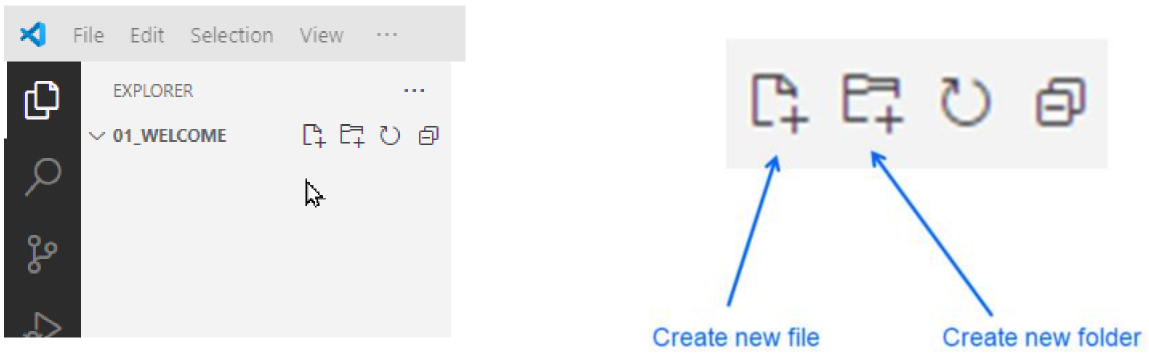


You can manage all the sub-folders and files of your project here.

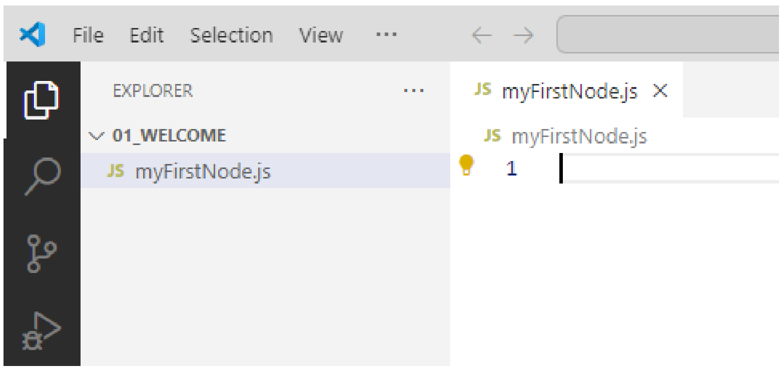
Create a JavaScript code file

Now that you have a project folder open in Visual Studio Code, you can create a JavaScript file that will contain your code.

If you move your mouse next to the project folder name, you will see icons that allow you to create new folders and files:



4. Create a new JavaScript file called `myFirstNode.js`:



Add JavaScript code to the file

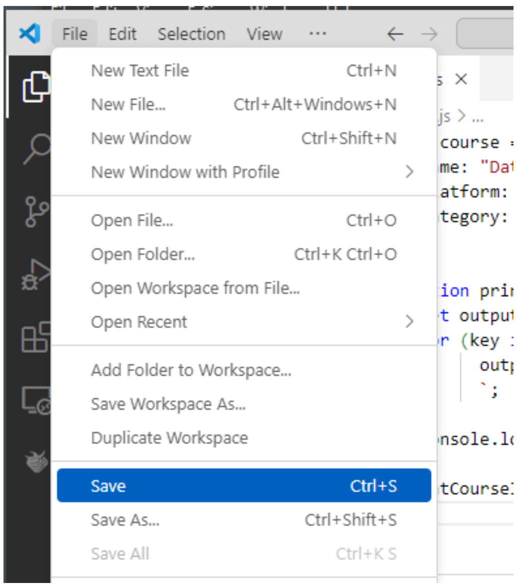
Adding content to a file using the built in Visual Studio Code GUI is extremely easy. Let's add some code to the `myFirstNode.js` file:

5. Open the `myFirstNode.js` file from the "Explorer" project section
6. Copy the following code:

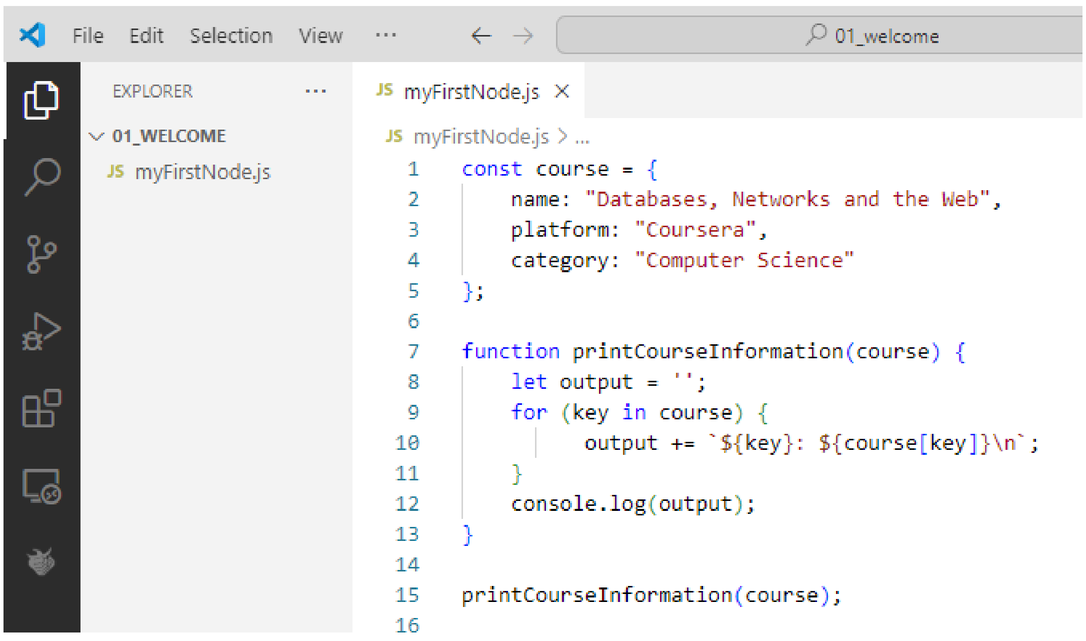
```
const course = {
    name: "Databases, Networks and the Web",
    platform: "Coursera",
    category: "Computer Science"
 };
```

```
function printCourseInformation(course) {
   let output = '';
   for (key in course) {
         output += `${key}: ${course[key]}\n`;
   }
   console.log(output);
 }
printCourseInformation(course);
```

7. Paste it inside the 'Editor' section of the `myFirstNode.js` file
8. Select the `myFirstNode.js` file, click the 'File' menu, then the 'Sav'" menu*:*



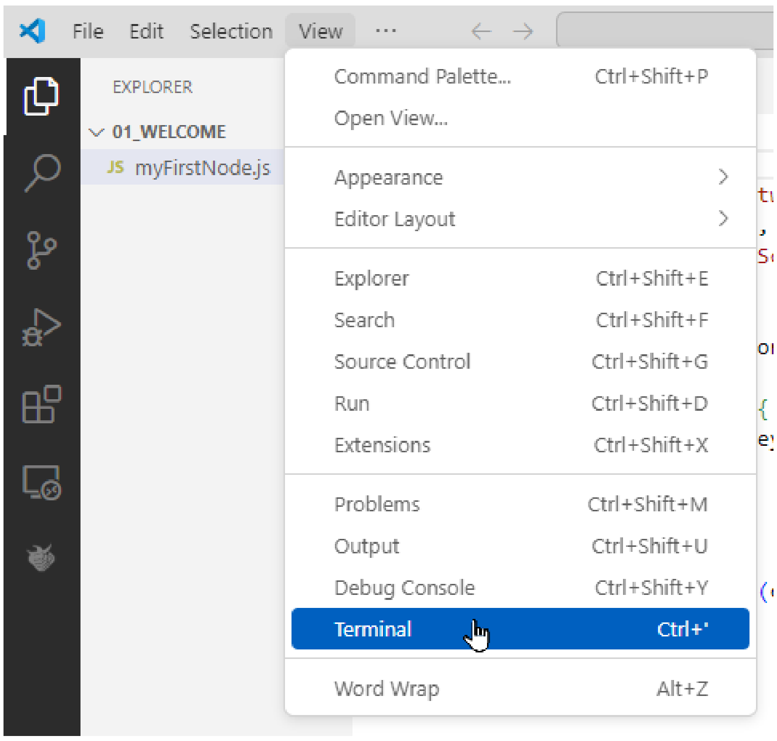
Once you have completed these steps, your project should look like the picture below:



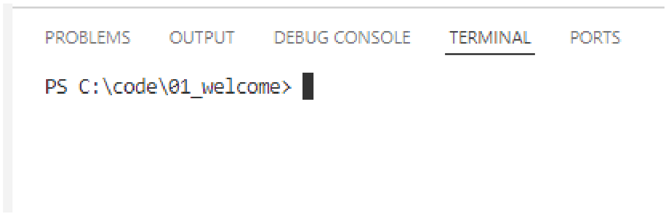
```
const course = {
    name: "Databases, Networks and the Web",
    platform: "Coursera",
    category: "Computer Science"
};

function printCourseInformation(course) {
    let output = '';
    for (key in course) {
        output += `${key}: ${course[key]}\n`;
    }
    console.log(output);
}

printCourseInformation(course);
```

Run the JavaScript file with Node.js

In order to execute a JavaScript file using Node.js, you need to type a command in a Terminal window. Conveniently, Visual Studio Code has a built-in Terminal panel. To start it up, select Terminal from the View menu:



A Terminal panel should open up at the bottom of the Visual Studio Code window:



```
PS C:\code\01_welcome>
```

Here you can type in commands.

9. Enter the following command in the Terminal panel:

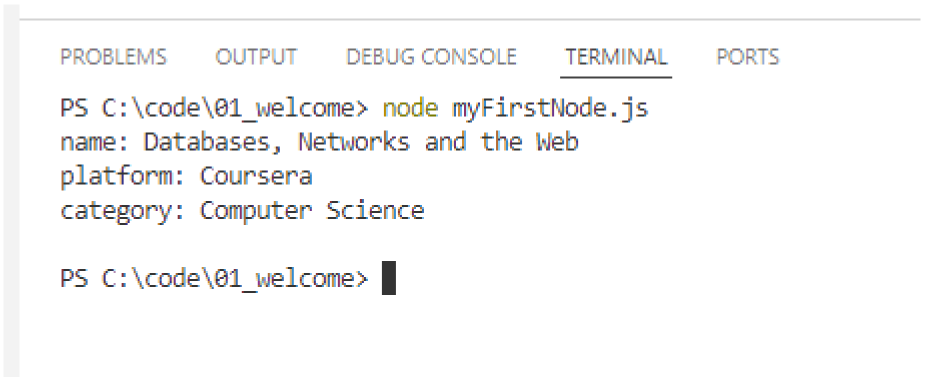
```
node myFirstNode.js
```

then press **Enter**.

The node command, followed by the file name, tells Node.js to execute the content of the file.

The node command only works with JavaScript files located in the same working directory as where the command was executed.

Running the above command will produce the following output:



```
PS C:\code\01_welcome> node myFirstNode.js
name: Databases, Networks and the Web
platform: Coursera
category: Computer Science

PS C:\code\01_welcome> 
```

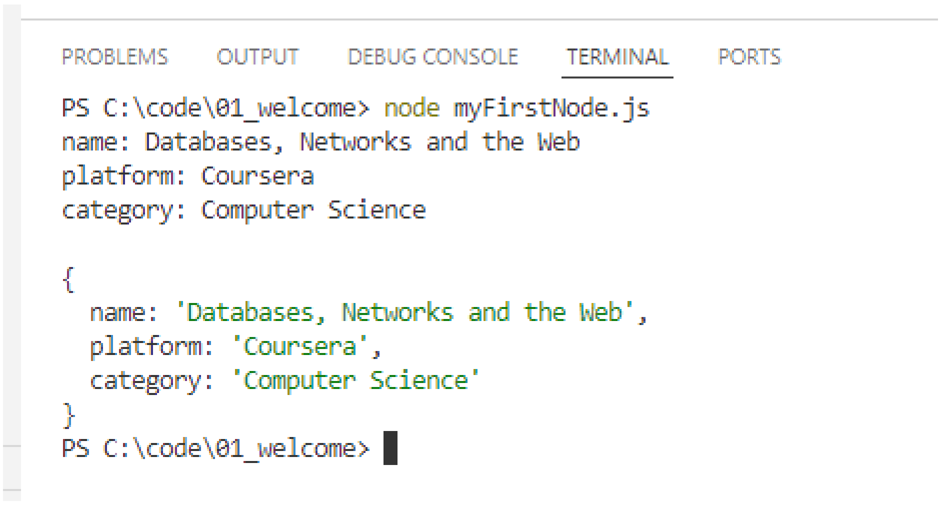
You can see that the Terminal panel printed out the information about the **course** object in the form of key/value pairs.

The `console.log` command allows you to quickly debug and output data about your running application.

10. Add the following line at the end of the `myFirstNode.js` script:

```
console.log(course);
```

Now run your application again. You should see the following output:



```
PS C:\code\01_welcome> node myFirstNode.js
name: Databases, Networks and the Web
platform: Coursera
category: Computer Science

{
  name: 'Databases, Networks and the Web',
  platform: 'Coursera',
  category: 'Computer Science'
}
PS C:\code\01_welcome> 
```

As you can see, the line `console.log(course);` printed out the raw content of the variable `course`.

Get in the habit of using `console.log` to quickly debug your lab applications and print out useful information.

Clear the Terminal panel logs

You can clear your Terminal panel logs by typing the following command in the Terminal:

```
clear
```

Type this command and press **Enter**.

How to ask for assistance

Use the discussion forums to ask your peers for help if you get stuck on a particular lab assignment.

End of section

If you are getting the same output, congratulations for completing this lab activity. You can practise further by modifying the `myFirstNode.js` file. Perhaps you can add more properties to the `course` object or rewrite your own version of the `printCourseInformation()` function. Make sure to save your changes before running the script again.

In the next lab activity, you will create your own virtual server. Isn't that exciting? There are no other requirements for this section, and you can carry on with the module.