Creating my first Node.js web server

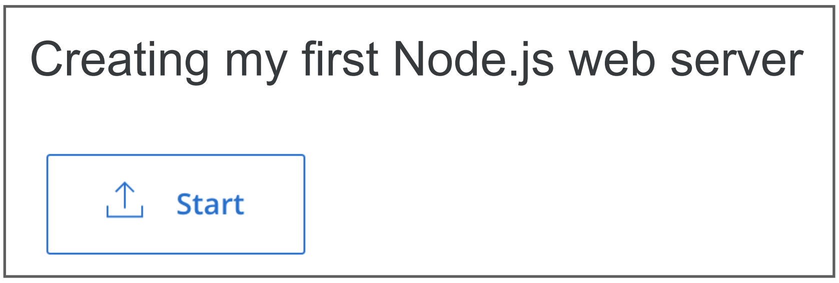
Welcome to this Lab activity

Let's build a Web Server with Node.js!
In this lab you will be creating a virtual server in order to serve web pages!
That's right; you will actually setup your very own server in the sky!

Start the Lab environment application

It is simple to launch a lab exercise. You only need to click on the button “Start” below the activity title to enter a lab environment.

Let’s explore this lab activity. Go ahead and click on the “Start” button!



Task 1: Serve a web page

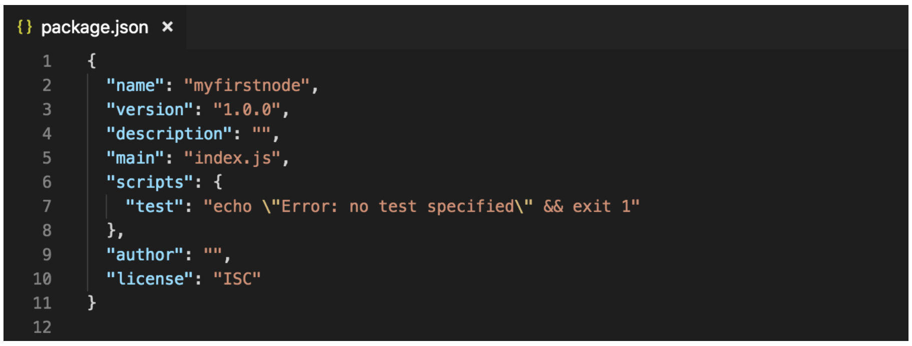
The folder structure has already been partially constructed for you and organised into different topics. For the purpose of this lab, you will be making changes inside the *topic2* folder structure. Let’s get started!

Please do not delete or move any existing folders/files inside the lab environment.

Use the Visual Studio Code Terminal and run the following commands:

- **cd topic2/myFirstNode**: type this command and press *Enter*. This command will change your working directory to be the *myFirstNode* folder.
- **npm init**: type this command and press *Enter*. This command will set up a new or existing npm package. You can skip all the npm initialisation questions by pressing *Enter* after each request. Once the npm package file has been created, you can view it inside the *myFirstNode* directory (*package.json*). The *package.json* file contains a set of information regarding your current node project.

Here is how your *package.json* file should look like:



package.json

```
{
  "name": "myfirstnode",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

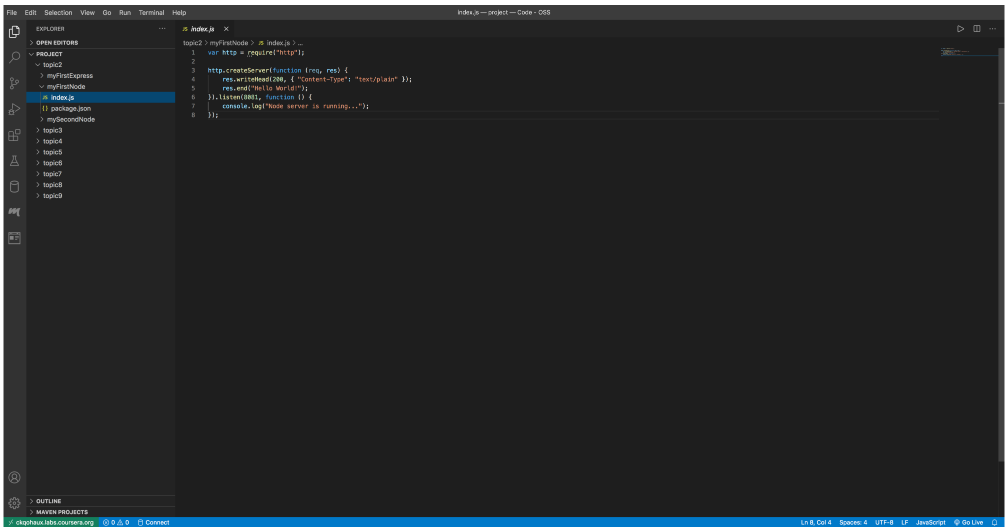
The very next step is to add some code to the *index.js* file located inside the *topic2/myFirstNode* folder in order to create your first web server.

Add the following code to the *index.js* file and remember to save it:

```
var http = require("http");

http.createServer(function(req, res) {
   res.writeHead(200, { "Content-Type": "text/plain" });
   res.end("Hello World!");
}).listen(8081, function() {
   console.log("Node server is running...");
});
```

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



```
var http = require("http");

http.createServer(function (req, res) {
    res.writeHead(200, { "Content-Type": "text/plain" });
    res.end("Hello World!");
}).listen(8081, function () {
    console.log("Node server is running...");
});
```

Understanding the code

Lets briefly understand the code that you have just copied inside your *index.js* file.

First of all we require the **http** module:

```
var http = require("http");
```

We will explore modules in the next session but they are essentially snippets of code which can be exported and imported across files.

Successively we create a server using the built in method of the **http** module called createServer:

```
http.createServer(function(req, res) {
   res.writeHead(200, { "Content-Type": "text/plain" });
   res.end("Hello World!");
})
```

This function responds with a 200 status code (request has succeeded), sets the response content type to be plain text and finally sends the “Hello World” text as the response.

Finally we start the web server using the built in function listen:

```
.listen(8081, function() {
   console.log("Node server is running...");
});
```

This function specifies the port (in our case 8081) and prints “Node server is running…” to the Terminal console:

Running the index.js file

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

- **node index.js**: type this command and press Enter. The node command, followed by the file name, tells Node.js to execute the content of the file.

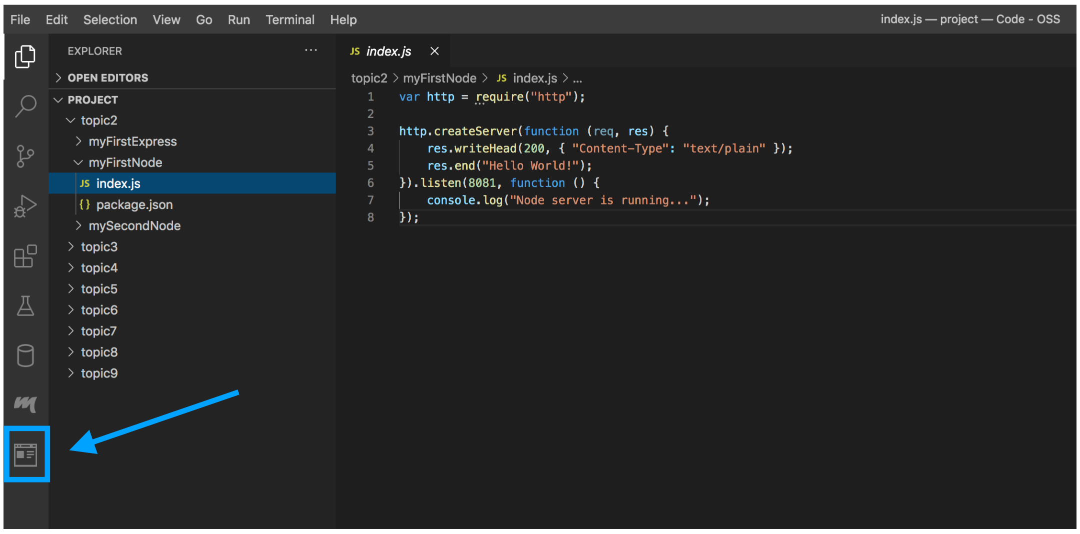
The above command will start a web server running on port 8081.

Task 2: Access your server via HTTP

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

The VsCode environment is equipped with a plugin called “Browser Preview” which allows you to visualise your web applications directly within the environment.

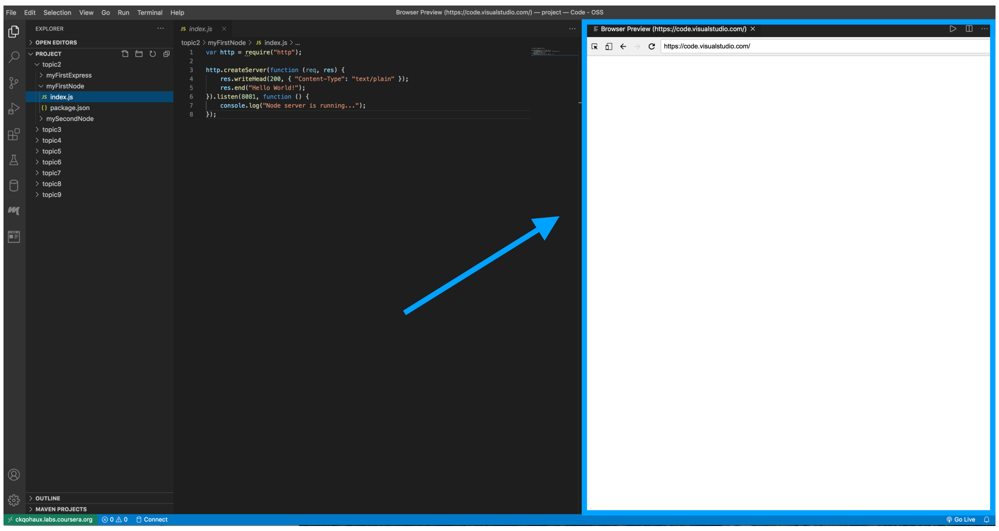
You can access the “Browser Preview” plugin by clicking on the corresponding activity bar icon as shown below:



```
var http = require("http");

http.createServer(function (req, res) {
    res.writeHead(200, { "Content-Type": "text/plain" });
    res.end("Hello World!");
}).listen(8081, function () {
    console.log("Node server is running...");
});
```

At this point you should see the “Browser Preview” appearing inside your environment:



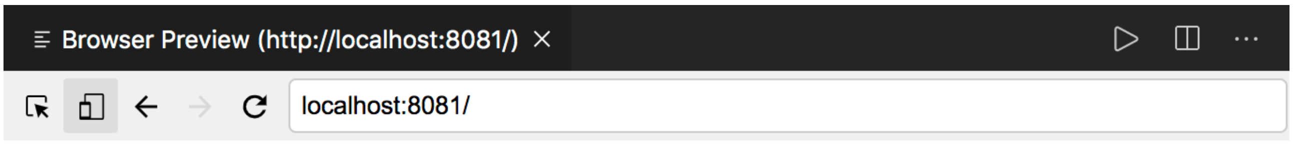
File Edit Selection View Go Run Terminal Help
Browser Preview (https://code.visualstudio.com/) — project — Code - OSS
EXPLORER
OPEN EDITORS
PROJECT
topic2
myFirstExpress
myFirstNode
index.js
package.json
mySecondNode
topic3
topic4
topic5
topic6
topic7
topic8
topic9
OUTLINE
MAVEN PROJECTS
index.js
topic2 > myFirstNode > index.js > ...
Browser Preview (https://code.visualstudio.com/)
https://code.visualstudio.com/
Go Live
Connect

```
var http = require("http");

http.createServer(function (req, res) {
    res.writeHead(200, { "Content-Type": "text/plain" });
    res.end("Hello World!");
}).listen(8081, function () {
    console.log("Node server is running...");
});
```

The very last step to visualise your application is to type the URL in the “Browser Preview” window.Your web application is running on localhost and port 8081.

Type *localhost:8081* on the “Browser Preview” tab and press *Enter* on your keyboard to visualise your web application:



If you have correctly followed all the steps, your web application should show the “Hello World!” message.

Shut down your web server

Remember that your web server is still running on port 8081. In order to shut down your server, you need to press the following keyboard keys combination while inside the Terminal:

- **Control + c**

This will stop the *index.js* script from running, shutting down the web server.

End of Section

Congratulations for completing this section. As long as you have saved your work, your files will remain when you close this lab activity so do not worry about losing your data. Next you will be making some changes to this lab activity and research more on the code you wrote.