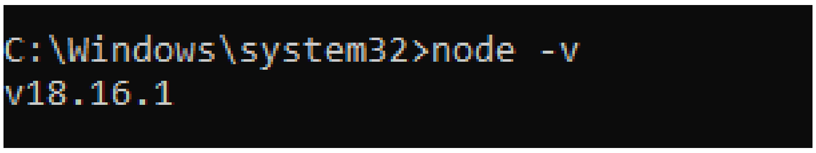
University of Colombo School of Computing

SCS 2208 - Rapid Application Development

Lab Sheet 07 - Node.js

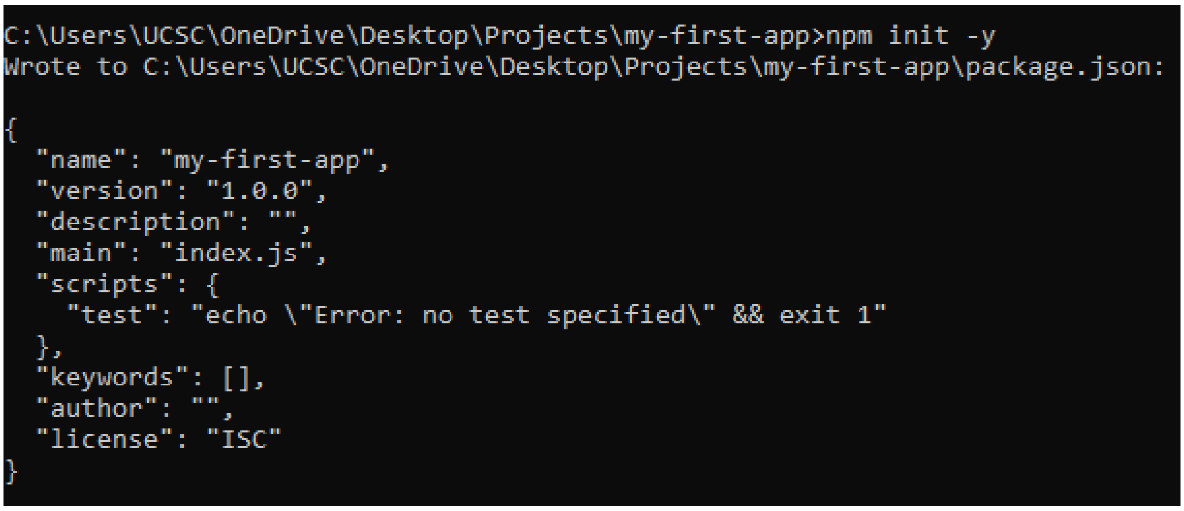
Starting with Node.js

1. Set Up the Development Environment.
   - Ensure you have Node.js installed on your computer. You can download it from the official Node.js website. (https://nodejs.org/)
   - Verify that Node.js is installed correctly by opening a terminal or command prompt and running the following command.



```
C:\Windows\system32>node -v
v18.16.1
```

2. Initialize a new Node.js Application.
   - Create a new folder for your Node.js application. You can name it whatever you like. For example, let's call it "my-first-app ".
   - Navigate into the project directory, and run the following command to initialize a new Node.js project using "**npm init -y**".



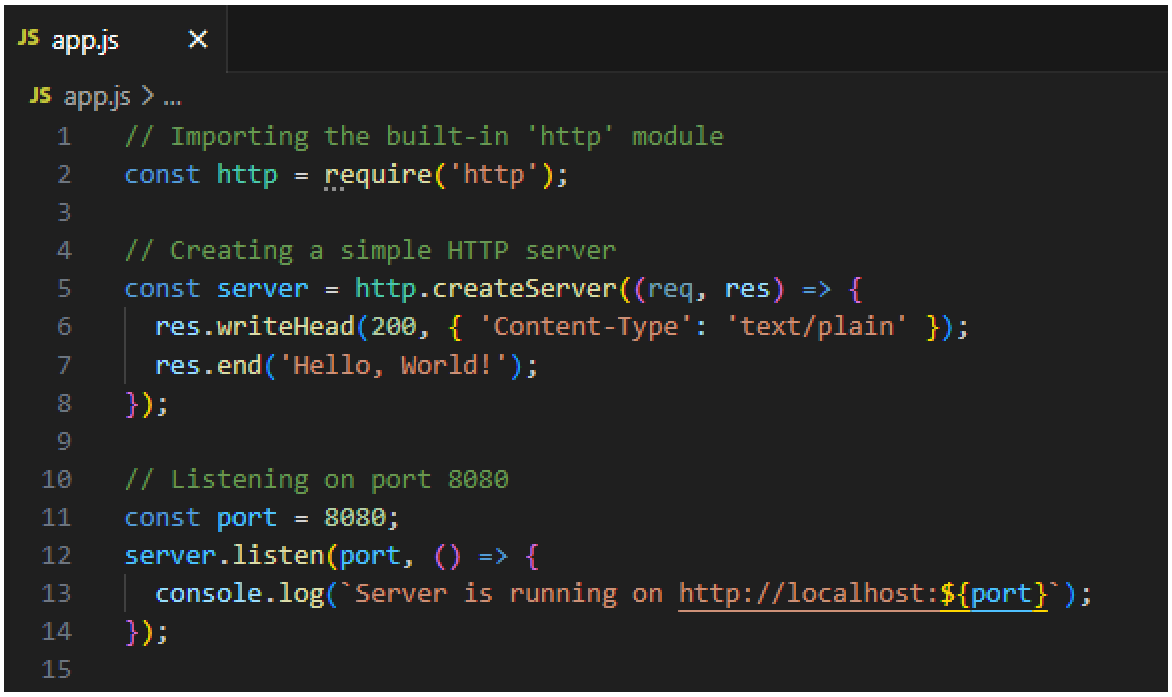
```
C:\Users\UCSC\OneDrive\Desktop\Projects\my-first-app>npm init -y
Wrote to C:\Users\UCSC\OneDrive\Desktop\Projects\my-first-app\package.json:

{
  "name": "my-first-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

This will create a new **package.json** file in your project folder.

3. Create the main application.
   - Open your code editor and navigate to the project directory (my-first-app).

- create a new file named “**app.js**” in your project folder. This will be the main entry point of our application.

4. Write the application code.
   - Open the app.js file and add the following code:



app.js ×

app.js > ...

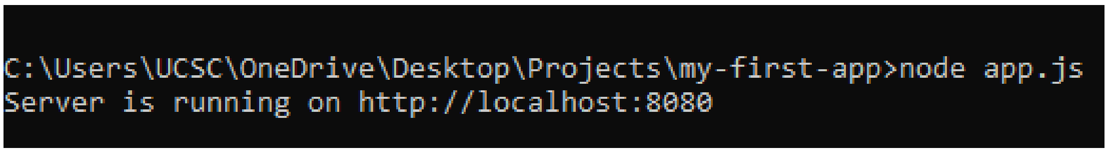
```javascript
// Importing the built-in 'http' module
const http = require('http');

// Creating a simple HTTP server
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello, World!');
});

// Listening on port 8080
const port = 8080;
server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

We begin by importing the built-in **http** module, which provides functionality to create an HTTP server and handle HTTP requests and responses. We do this by using the **require** function to load the **http module**.

Next, we create a simple HTTP server using the **createServer** method of the http module. This method takes a callback function as its argument, which will be executed whenever a new HTTP request is received by the server. The callback function has two parameters: **req** (short for request) and **res** (short for response). These parameters represent the incoming request and the server's response, respectively. Inside the callback function, we use the **res.writeHead** method to set the **HTTP status code to 200**, which means the request was successful. We also set the **Content-Type** header to **'text/plain'**, indicating that we are sending plain text as the response.

Finally, we need to specify on which port the server should listen for incoming requests. We choose to use **port 8080**, but you can change it to any other available port if needed. The **server.listen** method starts the server and tells it to listen on the specified port.

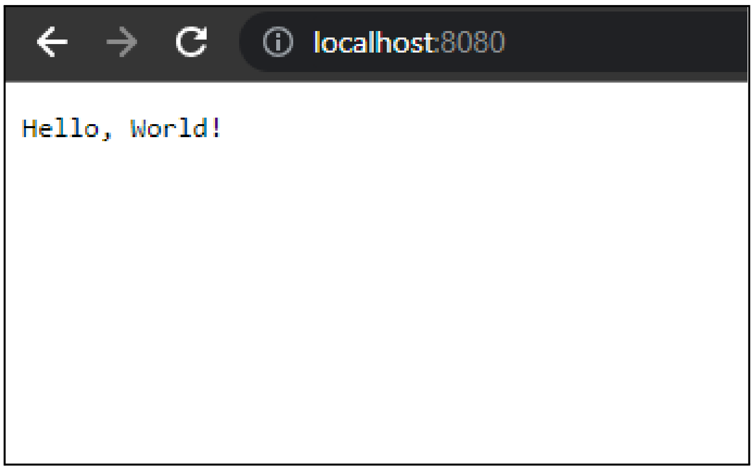
5. Run the application.
    - Save the **app.js** file, and now it's time to run your Node.js application using **“node app.js”**.



```
C:\Users\UCSC\OneDrive\Desktop\Projects\my-first-app>node app.js
Server is running on http://localhost:8080
```

You should see the message **"Server is running on http://localhost:8080"** in the console. This means your server is up and running.

6. Test the application.
    - Open your web browser and navigate to **http://localhost:8080**. You should see the message **"Hello, World!"** displayed on the page.



```
Hello, World!
```

Activities

1. Create a Node.js function that takes two numbers as input and returns their sum.

2. Create an array of numbers in Node.js and print the square of each number.

3. Create an array of numbers in Node.js and find the maximum number in the array.
   Hint: Use the Math.max() function to find the maximum number in an array.

4. Create a Node.js function named factorial that takes a positive integer as input and returns its factorial.
   Hint: Consider using recursion to calculate the factorial of a number.

5. Create a Node.js script that reads a JSON file named **data.json**, parses its content, and displays the values of the **"name"** and **"age"** properties.
   Hint: Use the fs module to read the JSON file, then parse its content using JSON.parse() to access and display the "name" and "age" properties.

6. Create a Node.js program that takes a string as input and prints its reverse to the console.
   Hint: Use the split(), reverse(), and join() methods to reverse the input string.

7. Write a Node.js program that reads a CSV file named **"data.csv"** and displays its content in a tabular format.
   Hint: You need to install the **csv-parser** package using npm (e.g. **npm install csv-parser**).

8. Write a Node.js program that fetches data from an API (e.g. https://jsonplaceholder.typicode.com/posts) and displays the response to the console.
   Hint: You need to install the **axios** package using npm (e.g. **npm install axios**).