

1. Send JSON data from node.js server: student info
 - Create a Javascript to let a node.js server send the following data to the client.

Student ID	Student Name	Score
11111	Bruce Lee	84
22222	Jackie Chen	93
33333	Jet Li	88

- If the client types the following URL, then student ID 11111 data, including Student ID, Student Name, and Score, will be sent from the node.js server to the client:
 - `http://<server>:8000/api/score?student_id=11111`
- If the client types the following URL, then student ID 22222 data, including Student ID, Student Name, and Score, will be sent from the node.js server to the client:
 - `http://<server>:8000/api/score?studnt_id=22222`

=====

Requirements:

- 1.install node.js
- 2.install npm (package manager for javascript)
- 3.install browser sync: <https://browsersync.io/>

JavaScript programming language is used for Node.js Application Development. The source files of Node.js applications have extension of “.js”. Any text editor is sufficient to write Node.js code and save it as .js file.

4. install visual studio code

=====

Create a simple Node.js web server and handle HTTP requests:

To access web pages of any web application, you need a [web server](#). The web server will handle all the http requests for the web application e.g., Apache is a web server for PHP or Java web applications.

Node.js provides capabilities to create your own web server which will handle HTTP requests asynchronously.

Part 1: Create Node.js Web Server

The following example is a simple Node.js web server contained in **student_server.js** file => which means, we will save our code as student_server.js file.

```
var http = require('http'); // 1 - Import Node.js core module

var server = http.createServer(function (req, res) { // 2 - creating server

    //handle incoming requests here..

});

server.listen(8000); //3 - listen for any incoming requests

console.log('Node.js web server is running on port 8000 ...')
```

First, we import the http module using require() function. The http module is a core module of Node.js.

The next step is to call createServer() method of http and specify callback function with request and response parameter.

Finally, call listen() method of server object which was returned from createServer() method with port number, to start listening to incoming requests on port 8000. You can specify any unused port here.

Run the above web server by writing `node student_server.js` command in command prompt or terminal window, ubuntu, etc and it will display message as shown below.

```
C:\> node student_server.js
Node.js web server is running on port 8000 ...
```

Part2: Handle HTTP Request

The `http.createServer()` method includes [request](#) and [response](#) parameters which is supplied by Node.js.

The request object can be used to get information about the current HTTP request e.g., url, request header, and data.

The response object can be used to send a response for a current HTTP request.

```
var http = require('http'); // Import Node.js core module

const server = http.createServer(function (req, res) => { //create web server

  if (req.url == "/api/score?student_id=11111") { //check the URL of the current request
    //set response header
    res.writeHead(200, { 'Content-Type': 'application/json' });
    //set response content
    res.write(JSON.stringify({id: "11111", name: "Bruce Lee", score: "84"}));
    res.end();
  }
  else if (req.url == "/api/score?student_id=22222") {
    res.writeHead(200, { 'Content-Type': 'application/json' });
    res.write(JSON.stringify({id: "22222", name: "Jackie Chen", score: "93"}));
    res.end();
  }
  else if (req.url == "/api/score?student_id=33333") {
    res.writeHead(200, { 'Content-Type': 'application/json' });
    res.write(JSON.stringify({id: "33333", name: "Jet Li", score: "88"}));
    res.end();
  }
  else {
    res.end('Invalid request');
  }
});

server.listen(8000); //listen for any incoming requests

console.log('Node.js server is running on port 8000 ...');
```

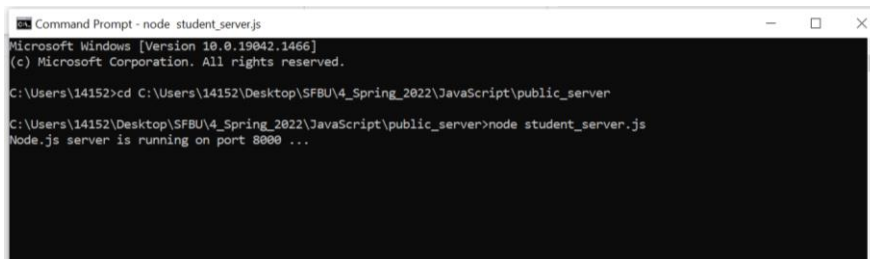
`req.url` is used to check the url of the current request and based on that it sends the response.

To send a response, first it sets the response header using `writeHead()` method

Then writes a string as a response body using `write()` method.

Finally, Node.js web server sends the response using `end()` method.

Now, run the above web server as shown below.



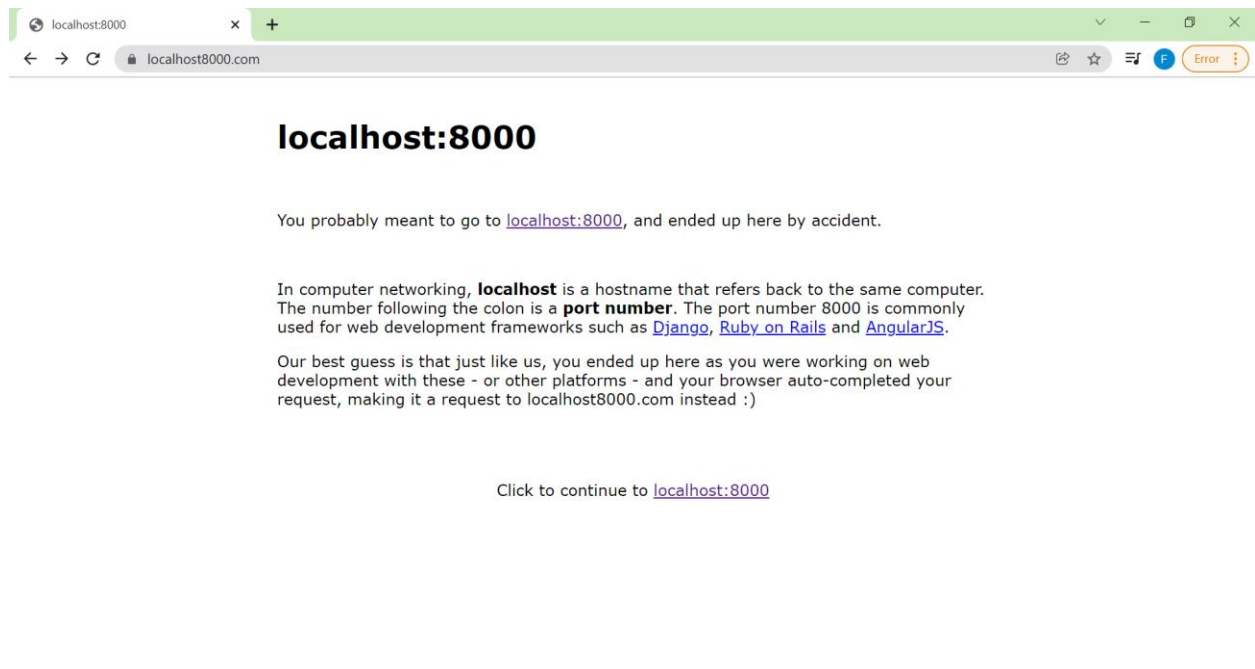
```
Command Prompt - node student_server.js
Microsoft Windows [Version 10.0.19042.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\14152>cd C:\Users\14152\Desktop\SFBU\4_Spring_2022\JavaScript\public_server
C:\Users\14152\Desktop\SFBU\4_Spring_2022\JavaScript\public_server>node student_server.js
Node.js server is running on port 8000 ...
```

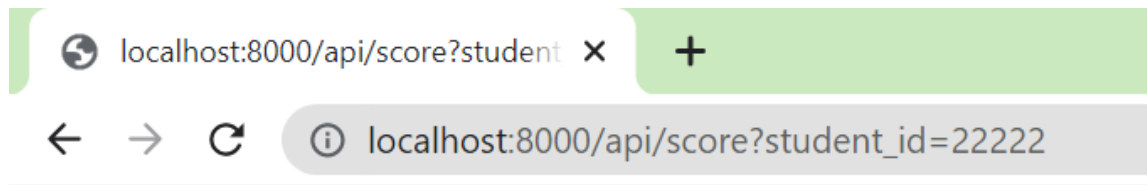
To test it, you can use the command-line program `curl`, which most Mac and Linux machines have pre-installed.

```
curl -i http://localhost:8000
```

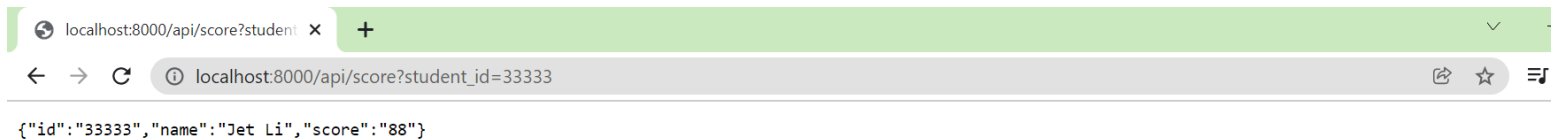
For Windows users, point your browser to **`http://localhost:8000`** and see the following result.



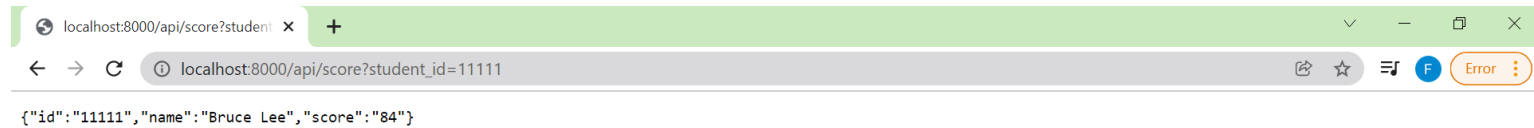
Click on go to **`localhost:8000`** and type the full url address:



```
{"id":"22222","name":"Jackie Chen","score":"93"}
```



```
{"id":"33333","name":"Jet Li","score":"88"}
```



```
{"id":"11111","name":"Bruce Lee","score":"84"}
```