

Introduction to Building a Node.js Server

First of all, you need to install Node.js to your PC and start the following developments.

Create a new directory and open it using VSCode or any other development tool that you prefer. Then run `npm init` command to initialize the project. You can change default data as you like. In the end, you will have the **package.json** file with the data that you entered.

```
package name: (auth-app)
version: (1.0.0)
description:
entry point: (index.js) app.js
test command:
git repository:
keywords:
author: dev
license: (ISC)
About to write to C:\Users\Devindi\Desktop\auth app\package.json:

{
  "name": "auth-app",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "dev",
  "license": "ISC"
}

Is this OK? (yes) y
```

Then you need to install **nodemon** globally. Although it is not necessary, it automatically restarts the application. To install nodemon run,

```
npm install -g nodemon.
```

Create **app.js** file inside your directory. To carry on future developments you need to install **express.js** framework.

```
npm install express --save
```

After the installation, import Express in your app.js file and initialize the express module in it.

```
const express = require('express');
```

```
const app = express();
```

Now you can access the express module through the app. Therefore create a server to handle server-side functionalities. Here you can choose a port number for the server to listen.

```
const port = 3000
app.listen(port, function(){
  console.log("server is running port " + port);
});
```

Express.js gives us routing capabilities too. So we can give any URL pattern for a simple **get** request. Here I give <http://localhost:3000/>.

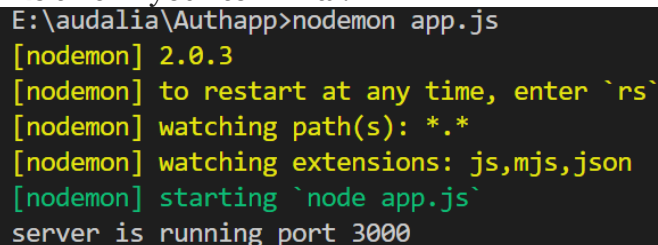
You can give it as *http://localhost:3000/user* or any other pattern you like.

A callback function passes through the get method with two objects **req** and **res**. req object has headers of the request while the res object sends the expected data/message also known as the response.

```
app.get("/",function (req, res) {  
  res.send("hello world");  
});
```

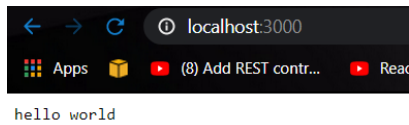
Finally, you will have following app.js file

Now you can run the server by the command `nodemon app.js` In this case, you will get like this in your terminal.

A terminal window with a black background and yellow and green text. The prompt is 'E:\audalia\Authapp>' followed by the command 'nodemon app.js'. The output shows 'nodemon' version 2.0.3, instructions to restart with 'rs', watched paths and extensions, and the message 'starting node app.js'. The final line says 'server is running port 3000' with a cursor on the next line.

```
E:\audalia\Authapp>nodemon app.js  
[nodemon] 2.0.3  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node app.js`  
server is running port 3000
```

You can see the output of your server here <http://localhost:3000/>. Check your endpoint using a browser or postman kind of application.



Done! You have successfully implemented the Node.js server.