4.5 Debug a Node.js Application



This section will guide you to:

* Debug a Node.js application

This guide has four subsections, namely:

4.5.1 Setting up a Node.js application

4.5.2 Setting up a debugging mode in Visual Studio Code

4.5.3 Performing a debug task for the existing Node.js application

4.5.4 Pushing the code to your GitHub repositories

**Step 4.5.1:** Setting up a Node.js application

**Note:** You will be able to proceed with this demonstration only if you have completed all the steps mentioned in the document **Debug the Node.js Application.**

* Create a folder **Debug Node.js application** and import the project in Visual Studio Code
* Open the terminal navigating to **Terminal** > **New Terminal**
* Run the following command to initialize the package.json file:

*npm init*

* Enter package name, version number, description, entry point, test command, git repository, keywords, author, and license. You can set it to default by pressing [enter] 9 times
* Check if the configuration you suggested is correct. Press [enter] again
* Create a file **app.js** and add the following source code in it:

var debug = require('debug')('http')

, http = require('http')

, name = 'My App';

// fake app

debug('booting %o', name);

http.createServer(function(req, res){

debug(req.method + ' ' + req.url);

res.end('hello\n');

}).listen(3000, function(){

debug('listening');

});

// fake worker of some kind

require('./worker');

* Create a file **worker.js** and add the following source code in it:

var a = require('debug')('worker:a')

, b = require('debug')('worker:b');

function work() {

a('doing lots of uninteresting work');

setTimeout(work, Math.random() \* 1000);

}

work();

function workb() {

b('doing some work');

setTimeout(workb, Math.random() \* 2000);

}

workb();

* To ensure all required packages are available, run the following command:

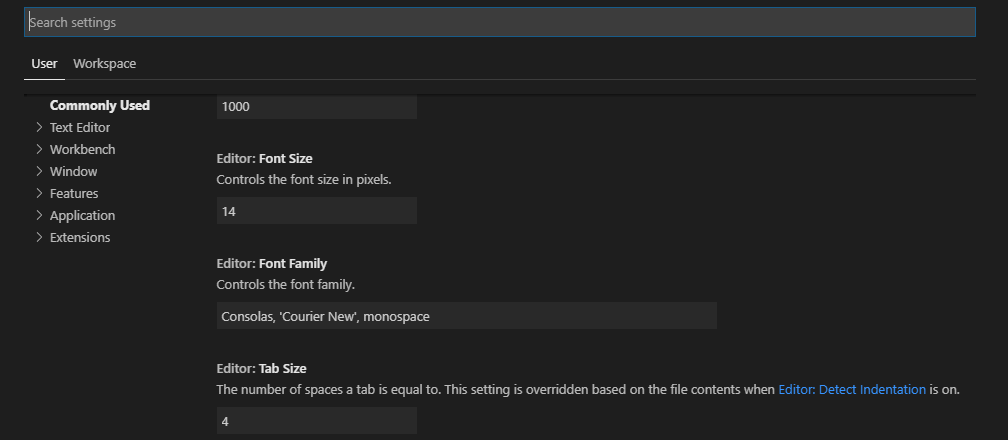
*npm install*

* Run the following command in the terminal to ensure that the application is running at the URL: <http://localhost:3000/>

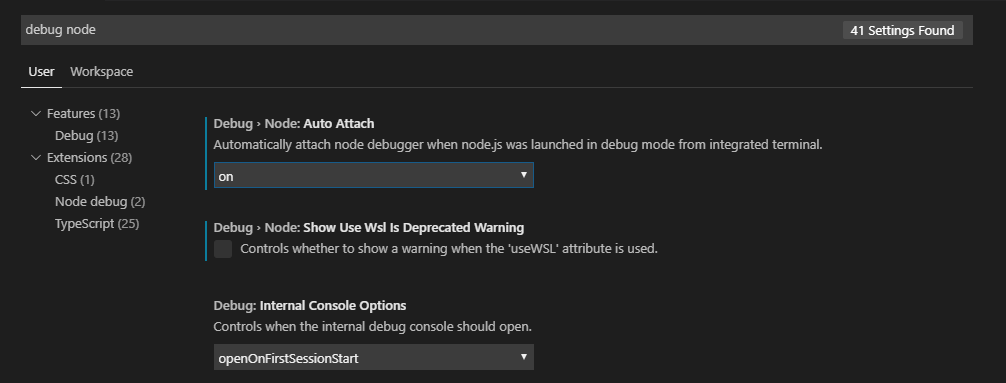
*node app.js*

**Step 4.5.2:** Setting up a debugging mode in Visual Studio Code

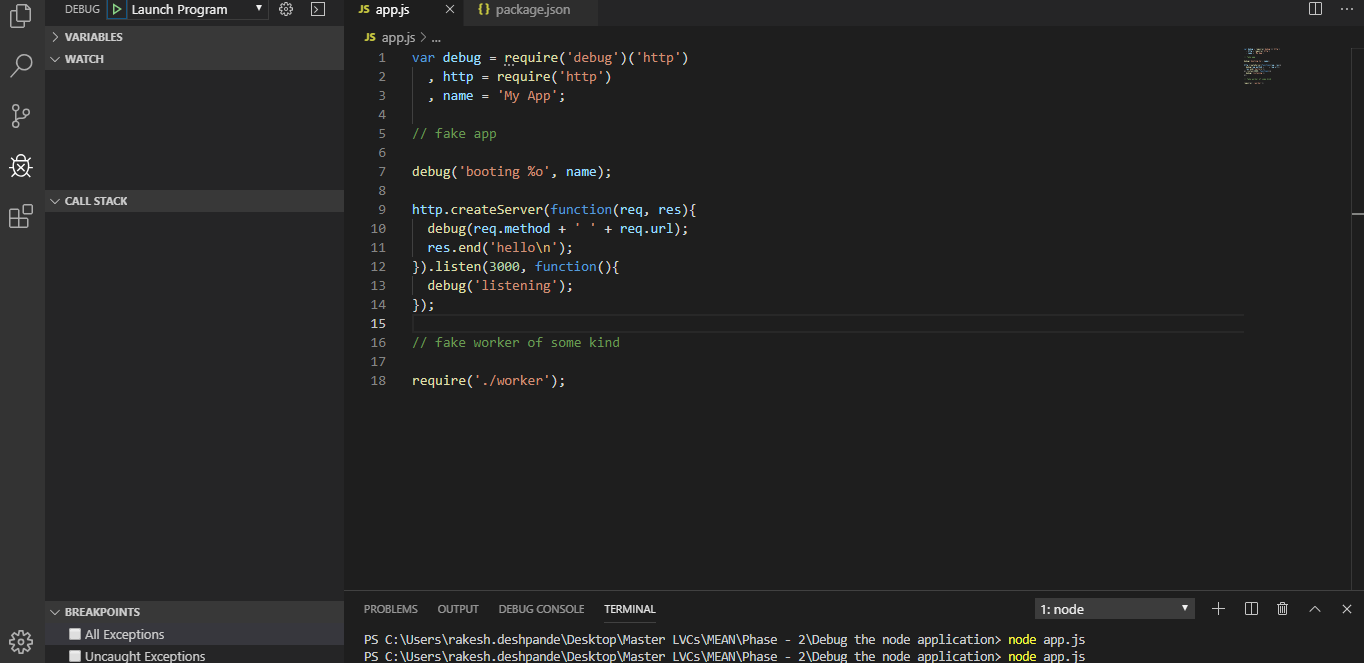
* Click on the **gear** symbol available at the bottom-left corner of your Visual Studio Code and select **Settings**. You can use the shortcut, [ctrl+,], to open an alert dialog box like the one shown in the screenshot:



* Search for **debug node**. You should be able to see **Debug>Node:Auto Attach** section which has the default option as **disabled.** Click on the drop down and select **On** as shown in the screenshot. Close the window.



* On the left panel of the Visual Studio Code, you should be able to see a **debug** icon. You can open the debugger by pressing [ctrl+shift+D]. It opens a whole new UI for debugging.



**Step 4.5.3:** Performing a debug task for the existing Node.js application

* You can choose the breakpoints from the bottom-left section of the Visual Studio Code. Click on the [Play] icon available at the top-left section beside **DEBUG.** You should be able to see that debug mode is now ready to use and a supporting pallet is opened to help you navigate as shown in the screenshot below. Feel free to explore the debugging mechanism in Visual Studio Code.



**Step 4.5.4:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add . 

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you initially created using the following command:

git push -u origin master