



NPM Node js

#Node JS Notes

Node Package Manager



What is NPM ?

- NPM Means Node Package Manager.
- The **NPM** program is installed on your computer when you install **Node.js**.
- **NPM** install packages in two modes local and global.



Version Check

- `npm -v`
- `npm -version`



What is use of NPM ?

- **npm** makes it easy for **JavaScript** developers to share and reuse code, and it makes it easy to update the code that you're sharing.
- Just like php has **composer** for downloading dependencies which are just code in files.
- To Find Various NPM Packages visit www.npmjs.com



Command

- How to Install/Upgrade NPM
 - `npm install -g npm`
 - `npm install -g npm@7.5.0`

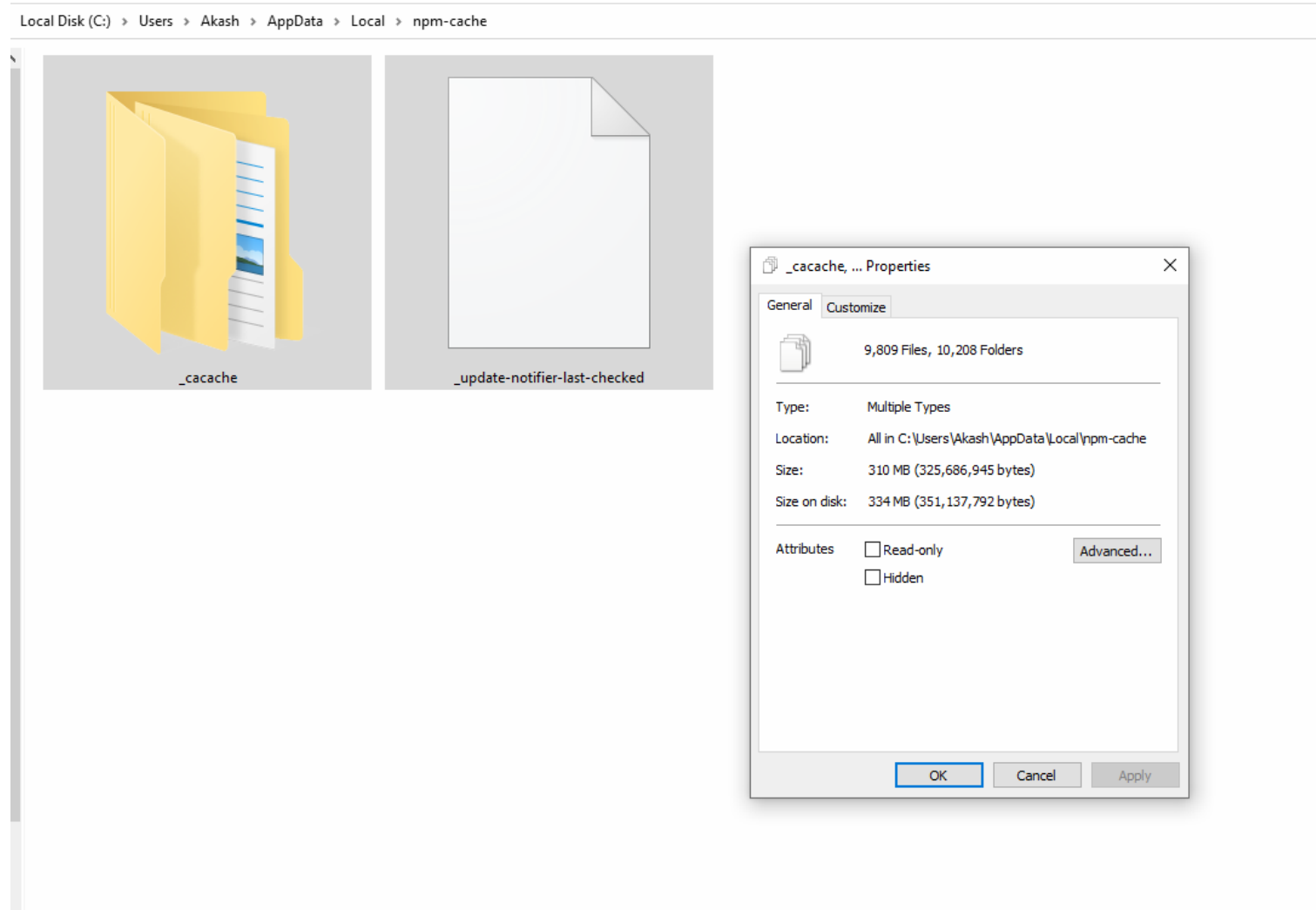


Local vs Global

- Local – folder based module
 - In the local mode, **NPM** installs packages in the **node_module** directory of the current working directory which location is owned by current user.
 - **D:\ProjectName\node_module**
- Global – System Based Modules
 - Global packages installed in the directory where the node is installed and the location is owned by the root user
 - **C:\Users\AkashPadhiyar\AppData\Roaming\npm**
 - Cache : **C:\Users\AkashPadhiyar\AppData\Local\npm-cache**



Example



What is Module ?

- Each js file is a module
- Variables outside the file cannot be visible except they are global or exported
- Use **module.exports** to write modules
- Use `require('fs')` to import fs module.



What is Package

- Package is a group of a modules
- Package metadata is defined in package.json
- package.json in the root of a package dir



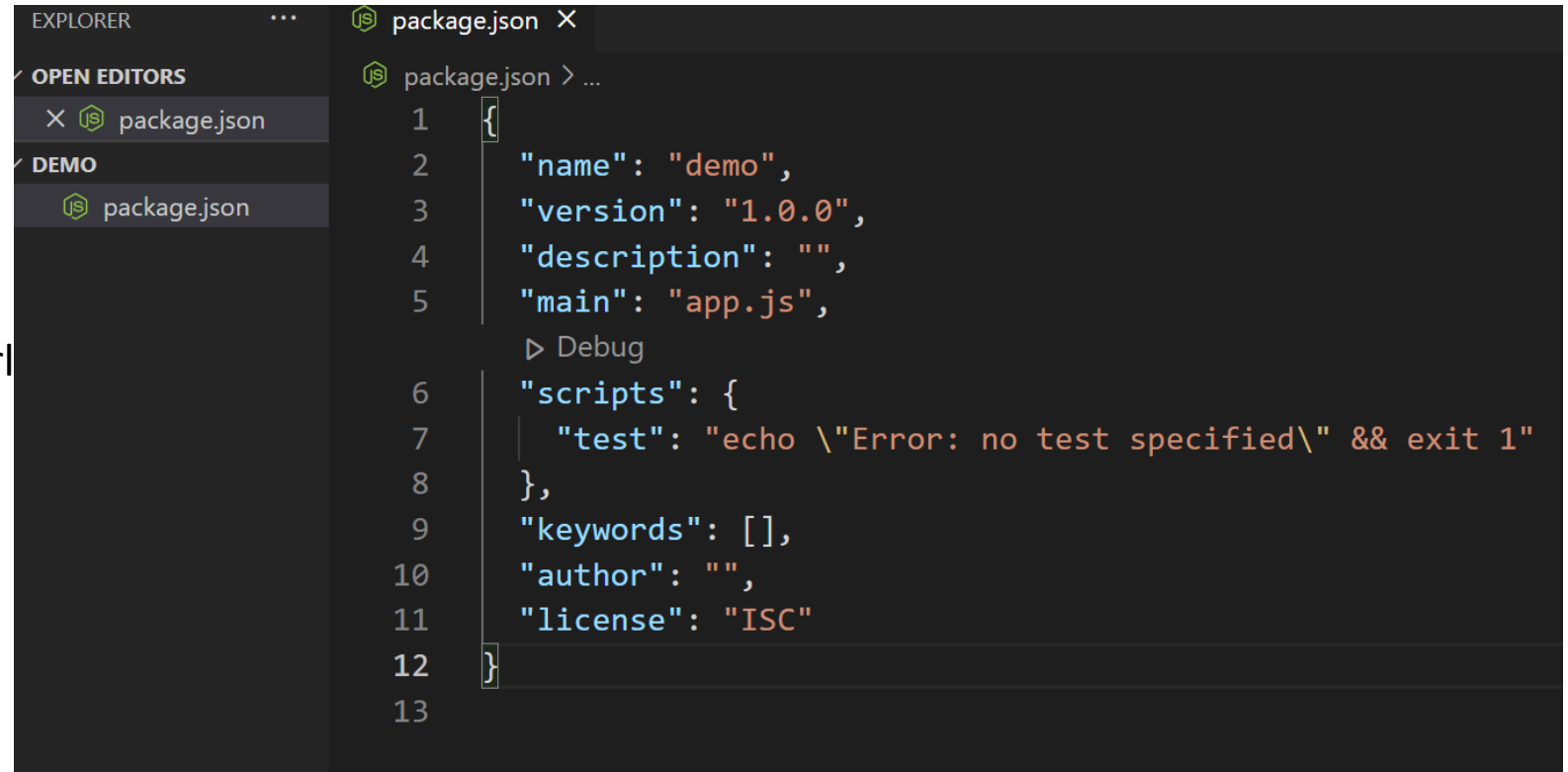
What is package.json?

- package.json is a plain JSON text file which manages all the packages which you installed in your node application.
- Every Node.js application should have this file at the root directory to describe the application metadata.
- A simple package.json file looks like below



Package.json

```
{  
  "name" : "akashpadhiyar",  
  "version" : "1.0.0"  
  "repository": {  
    "type" : "git",  
    "url" : "github_repository_url"  
  },  
  "dependencies": {  
    "async" : "0.8.0",  
    "express" : "4.2.x"  
  }  
}
```

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a file named 'package.json' under a 'DEMO' folder. The main editor window displays the content of 'package.json' with line numbers 1 through 13. The JSON content is: { "name": "demo", "version": "1.0.0", "description": "", "main": "app.js", "scripts": { "test": "echo \"Error: no test specified\" && exit 1" }, "keywords": [], "author": "", "license": "ISC" }.

```
1 {  
2   "name": "demo",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "app.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "keywords": [],  
10  "author": "",  
11  "license": "ISC"  
12 }  
13
```



Who uses package.json file?

- NPM (Node Package Manager) uses this package.json file information about Node JS Application information or Node JS Package details.
- package.json file contains a number of different directives or elements.
- It uses these directives to tell NPM “How to handle the module or package”.



How to Create package.json file?

- Create one folder.
- Navigate to Folder.
- Open Command prompt
- Type below command
 - `npm init`
- Provide required information



Create Folder

C:\Windows\system32\cmd.exe

```
D:\>mkdir akash
```

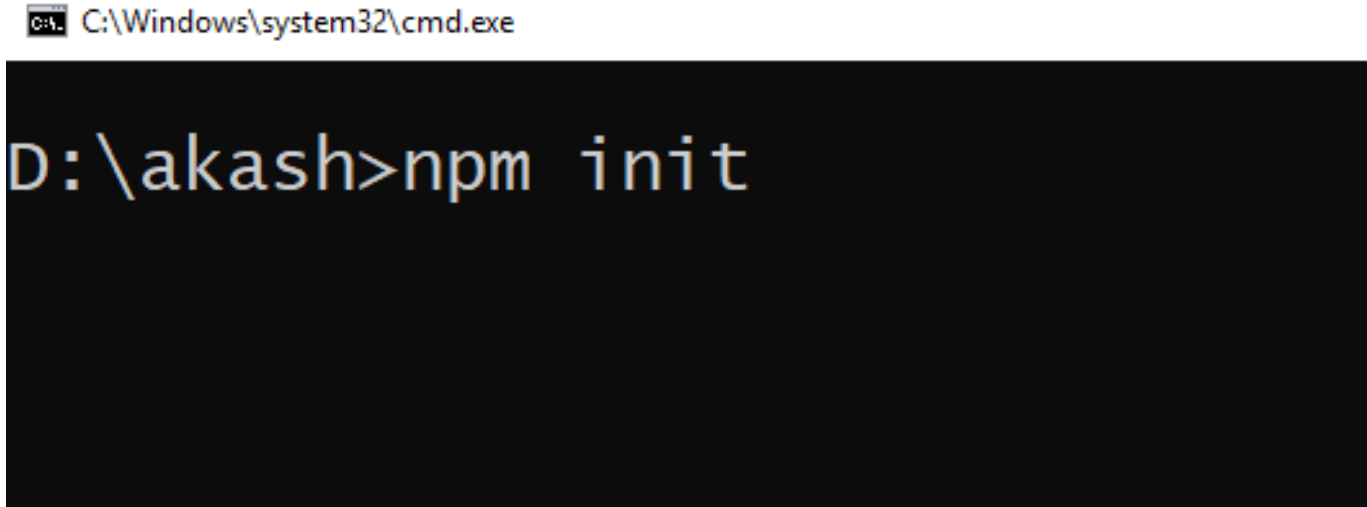
```
D:\>cd akash
```

```
D:\akash>
```



Npm init

Npm init command will create package.json file.



```
C:\Windows\system32\cmd.exe  
D:\akash>npm init
```



Provide Package Name

cmd npm

```
D:\akash>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (akash)
```



Provide Basic information

Ca npm

```
D:\akash>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (akash) helloworld
version: (1.0.0)
description: First Node js Application
entry point: (index.js) app.js
test command:
git repository:
keywords:
author:
license: (ISC)
```



Review your Details

```
license: (ISC)
About to write to D:\akash\package.json:

{
  "name": "helloworld",
  "version": "1.0.0",
  "description": "First Node js Application",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
```



Package json file demo

```
D:\akash\package.json - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
package.json
1 {
2   "name": "helloworld",
3   "version": "1.0.0",
4   "description": "First Node js Application",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "",
10  "license": "ISC"
11 }
12
```



Mandatory Directives of Package.json

- name:

- It is a unique Node JS Package (Module) name or our Node JS Project name.
- This name should be in lower case letters. It is mandatory directive. Without this directive, we cannot install our Node JS Package by using NPM.

- Version

- version is the Node JS Package version number.
- It is a Mandatory directive in package.json file. NPM uses this version number to install or uninstall or update the right package in our NODE JS Environment.

```
{  
  "name" : "helloworld",  
  "version" : "1.0.0"  
}
```



Optional Directives

- **description:**

- It is the description of the Node JS Project or module. We need to provide brief and concise description about “What this module does”.

- **dependencies:**

- Every Node JS Project or Module may dependent on other Node JS or Third Party Modules or our own Custom Modules. We should provide all those dependencies by using this “dependencies” directive.



Demo of Package.json file

```
{  
  "name" : "sampleapp",  
  "version" : "1.0.0"  
  "dependencies": {  
    "mypackage": "1.0.0",  
    "async": "0.8.0",  
    "express": "4.2.x"  
  }  
}
```



Short Explanation:

- **name:** The name of the application/project.
- **version:** The version of application. The version should follow semantic versioning rules.
- **description:** The description about the application, purpose of the application, technology used like React, MongoDB, etc.
- **main:** This is the entry/starting point of the app. It specifies the main file of the application that triggers when the application starts. Application can be started using **npm start**.
- **scripts:** The scripts which needs to be included in the application to run properly.



- **keywords:** It specifies the array of strings that characterizes the application.
- **author:** It consist of the information about the author like name, email and other author related information.
- **license:** The license to which the application confirms are mentioned in this key-value pair.
- **dependencies:** The third party package or modules installed using **npm** are specified in this segment.



Easy Way to Create package.json

- Npm init --yes / -y
- It will create package.json file with default values.



npm init -y

```
C:\Windows\system32\CMD.exe

D:\>npm init -y
Wrote to D:\package.json:

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

D:\>
```



#Global vs. Local Installation.

- By default, **NPM** installs any dependency in the local mode. Here local mode refers to the package installation in the **node_modules** directory inside our working project. Locally deployed packages are accessible via `require()` method.
- For example, when we installed an express module, it created a **node_modules** directory in the current directory where it installed the express module.



Local Example

```
$ npm install express
```

Now you can use this module in your js file as following

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



Global

- Globally installed packages are stored in the system directory. Such dependencies can be used in **CLI** (Command Line Interface) function of any **node.js** but cannot be imported using **require()** in Node application directly.
- Example:
- `npm install express -g`



NPM Commands



Commands

- **npm install <Module Name>**
- **Install Express :**
 - npm install express
- **Uninstalling a Module**
 - npm uninstall express
- **Updating a Module**
 - npm update express
- **Search a Module**
 - npm search express



Global Package

- **Install Package Globally:**

- `npm install -g express`

- **Npm Help**

- `npm help`

NPM installs global packages into `/<User>/local/lib/node_modules` folder.

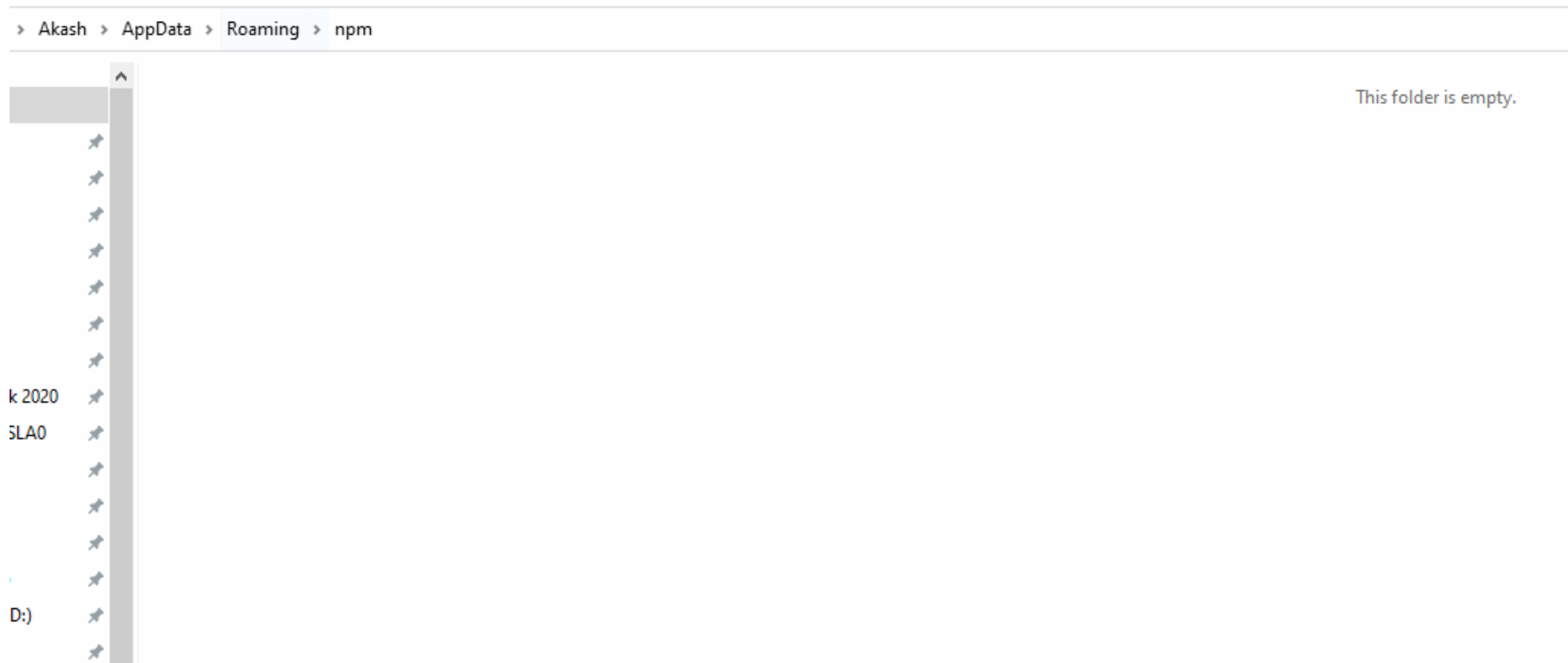
Global Package Location :

`C:\Users\Akash\AppData\Roaming\npm`



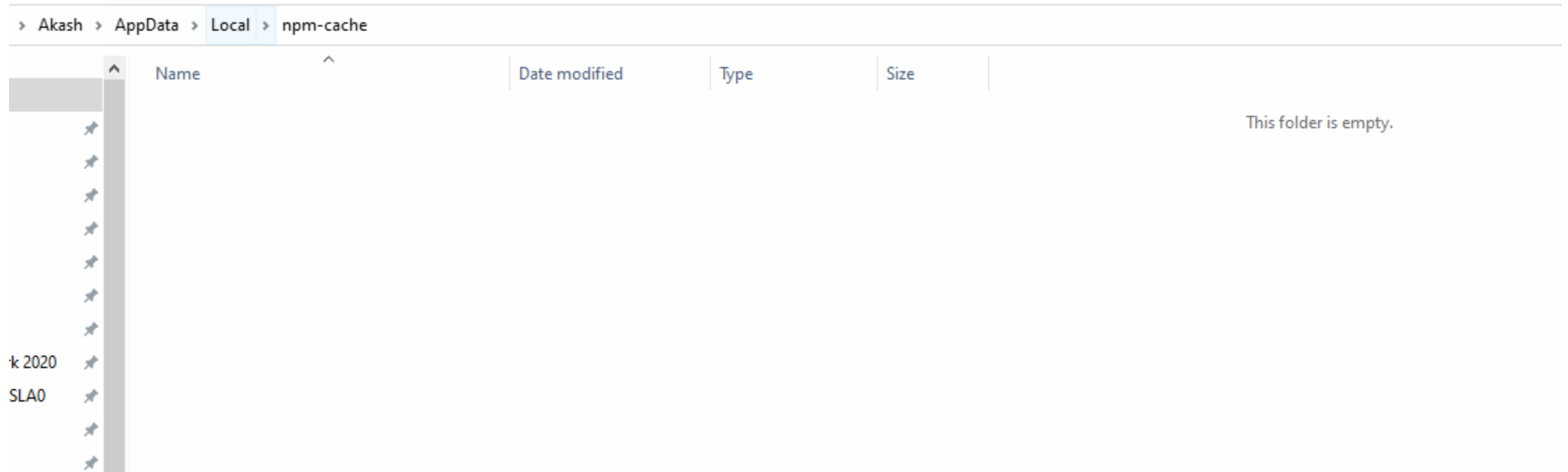
Global Folder Location

- C:\Users\Akash\AppData\Roaming\npm



Global File Location of Cache

- C:\Users\Akash\AppData\Local\npm-cache



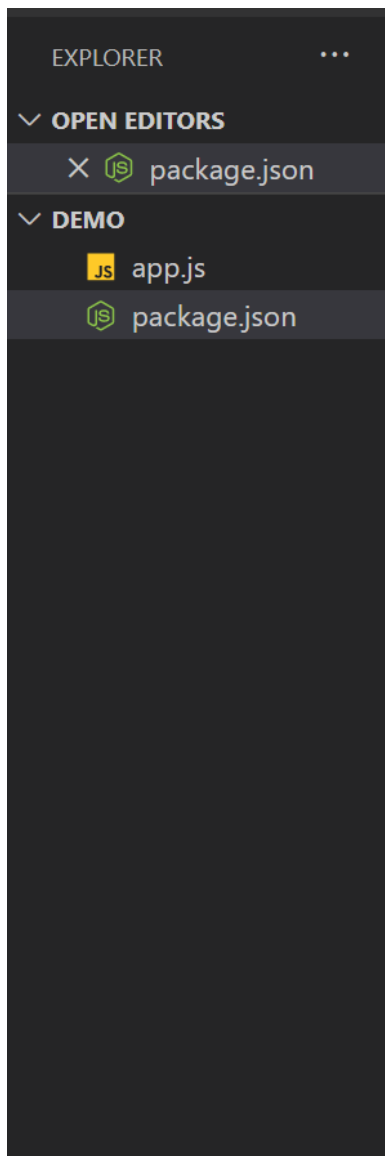


VS Code Explorer sidebar showing:

- EXPLORER
- OPEN EDITORS
 - package.json
- DEMO
 - package.json

```
1 {
2   "name": "demo",
3   "version": "1.0.0",
4   "description": "",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC"
12 }
13
```





EXPLORER

OPEN EDITORS

- package.json

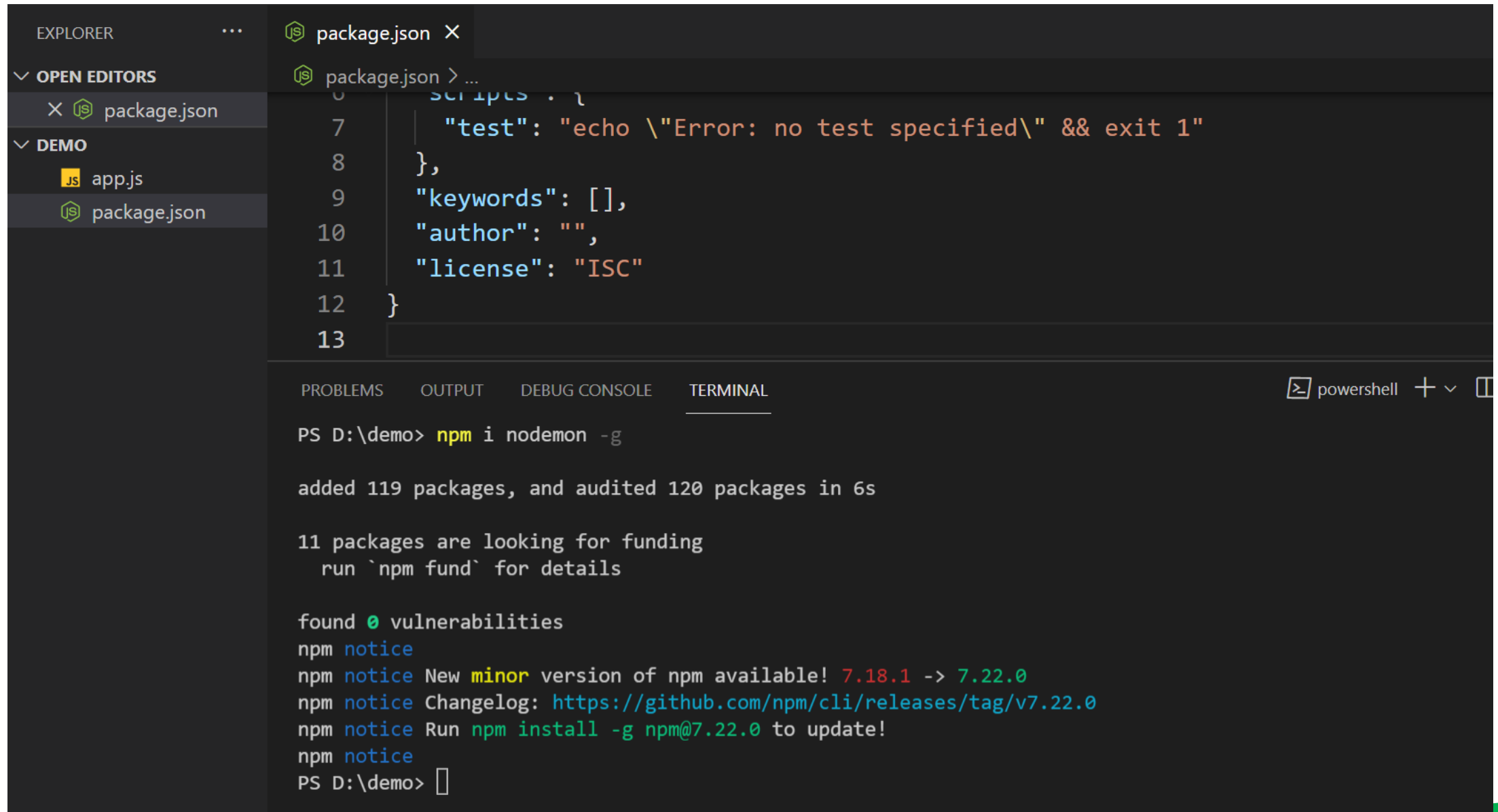
DEMO

- app.js
- package.json

```
package.json > ...
1  {
2    "name": "demo",
3    "version": "1.0.0",
4    "description": "",
5    "main": "app.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC"
12 }
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS D:\demo> npm i nodemon -g

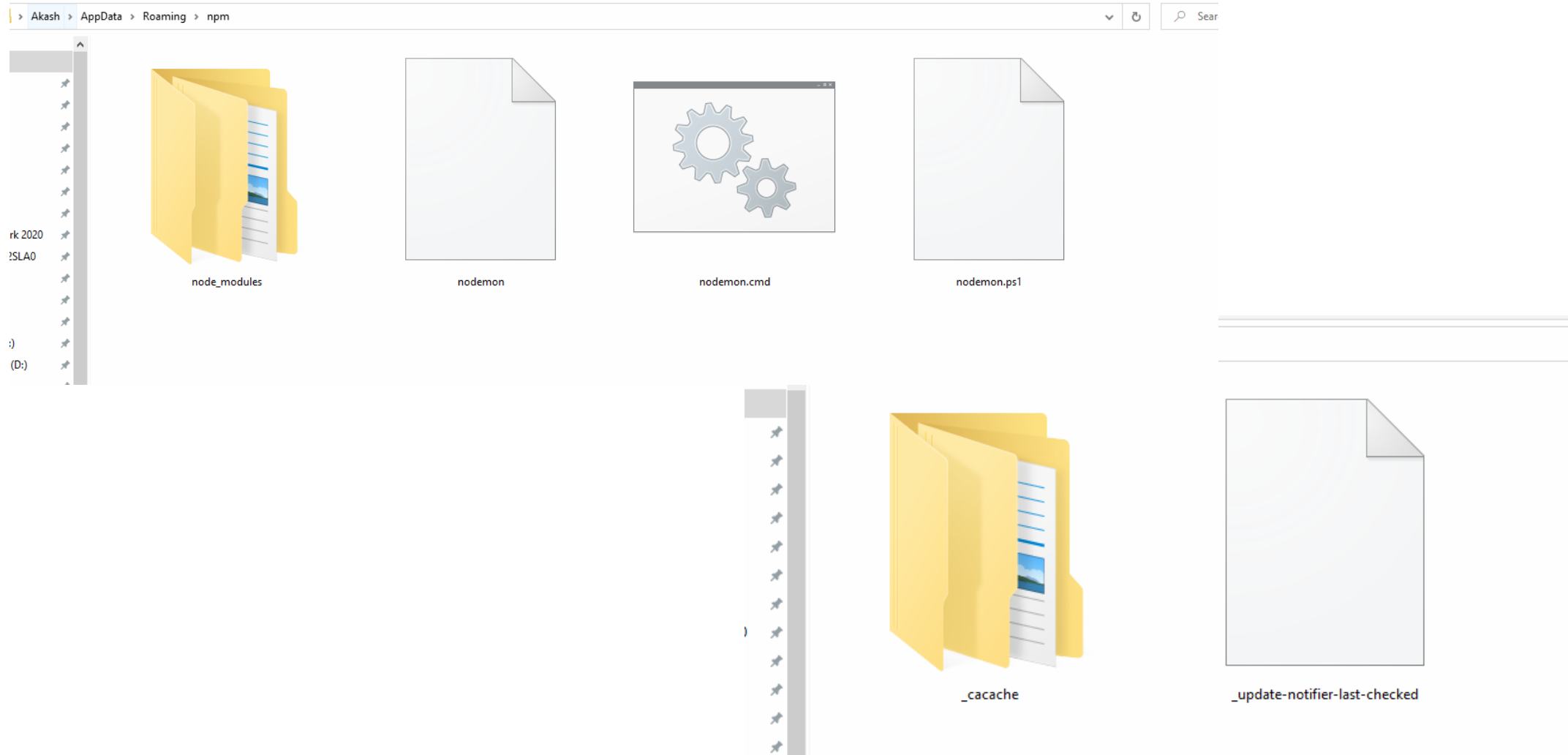


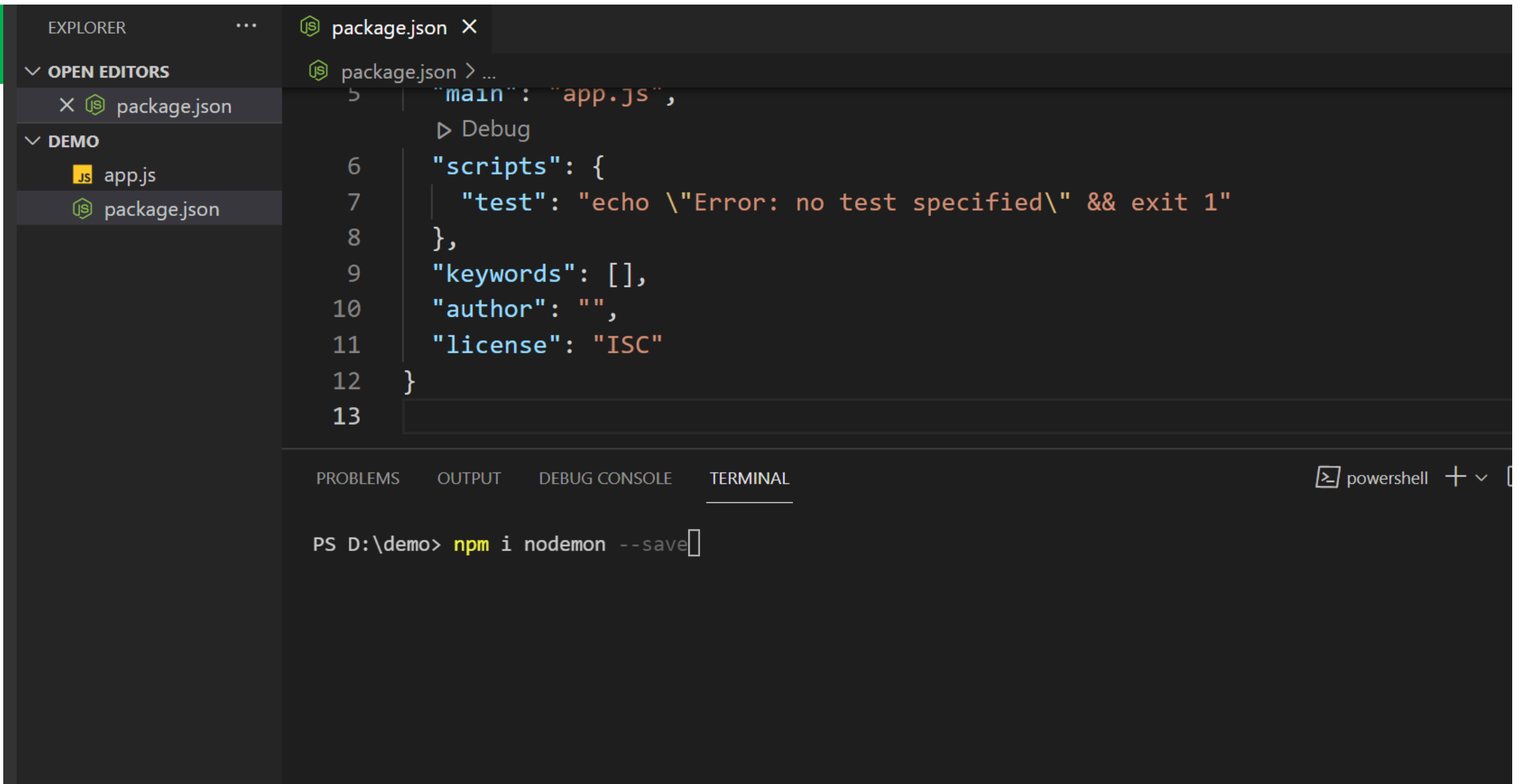
The screenshot shows a VS Code editor with a dark theme. On the left, the Explorer sidebar shows a project named 'DEMO' with files 'app.js' and 'package.json'. The main editor area displays the 'package.json' file with the following content:

```
1  {
2    "scripts": {
3      "test": "echo \"Error: no test specified\" && exit 1"
4    },
5    "keywords": [],
6    "author": "",
7    "license": "ISC"
8  }
```

Below the editor, the Terminal panel is active, showing the output of the command `npm i nodemon -g` executed in a PowerShell window. The output indicates that 119 packages were added and 120 packages were audited in 6 seconds. It also shows 11 packages looking for funding and 0 vulnerabilities found. A notice from npm states that a new minor version (7.22.0) is available, replacing the current version (7.18.1). The terminal ends with the prompt `PS D:\demo>`.

After Downloading Global Package

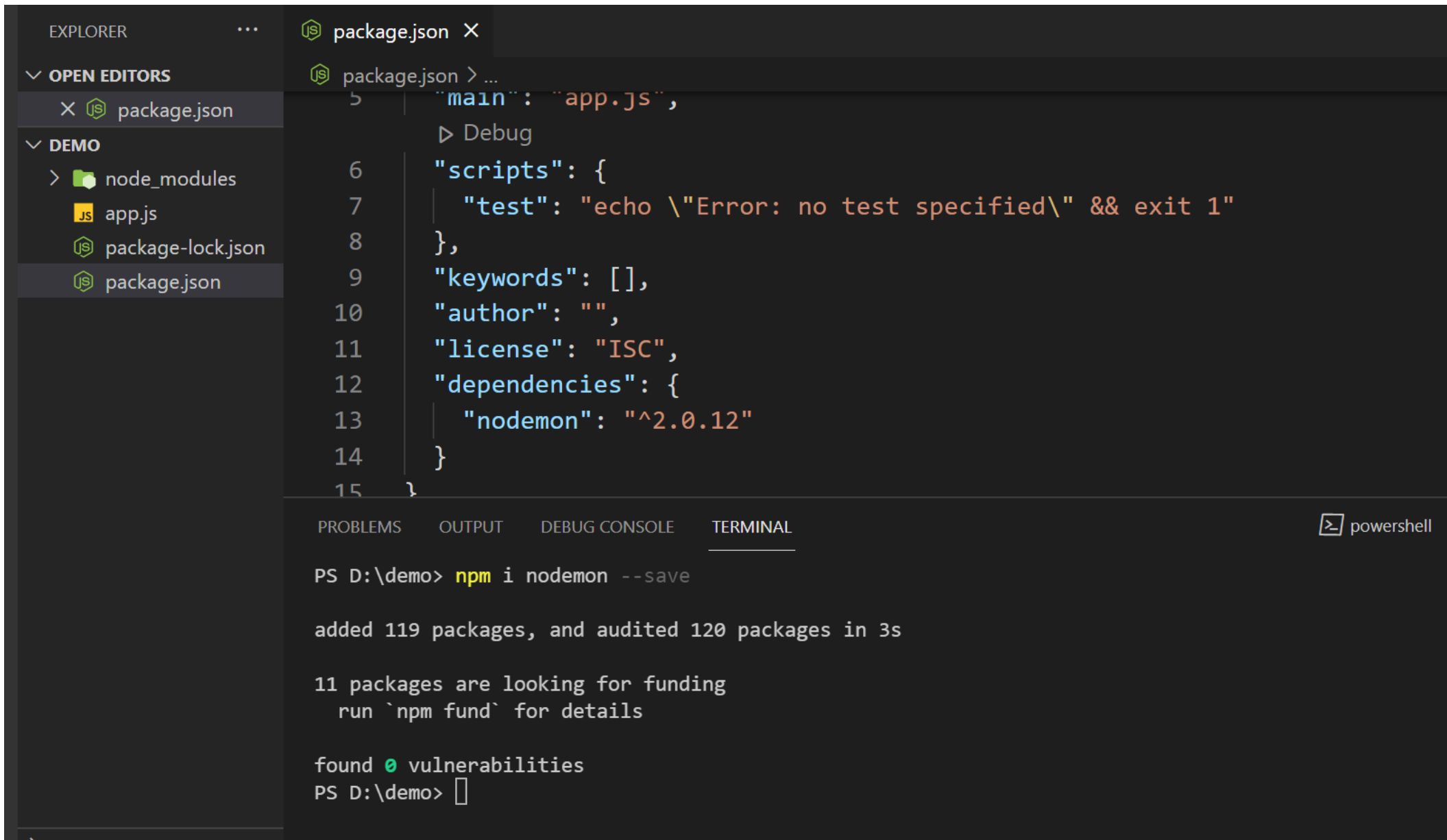




The image shows a Visual Studio Code editor window. On the left, the Explorer sidebar is open, showing a file named `package.json` under a `DEMO` folder. The main editor area displays the content of `package.json`, which includes a `main` field pointing to `app.js`, a `scripts` section with a `test` script, and fields for `keywords`, `author`, and `license`. Below the editor, the TERMINAL panel is active, showing a PowerShell prompt at `D:\demo` with the command `npm i nodemon --save` entered.

```
package.json X
package.json > ...
5  "main": "app.js",
   ▶ Debug
6  "scripts": {
7    "test": "echo \"Error: no test specified\" && exit 1"
8  },
9  "keywords": [],
10 "author": "",
11 "license": "ISC"
12 }
13

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS D:\demo> npm i nodemon --save
```

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'DEMO' with files 'app.js', 'package-lock.json', and 'package.json'. The 'package.json' file is open in the editor, showing the following content:

```
5  "main": "app.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "nodemon": "^2.0.12"
14   }
15 }
```

Below the editor, the TERMINAL panel is active, showing the output of the command `npm i nodemon --save` in a PowerShell shell:

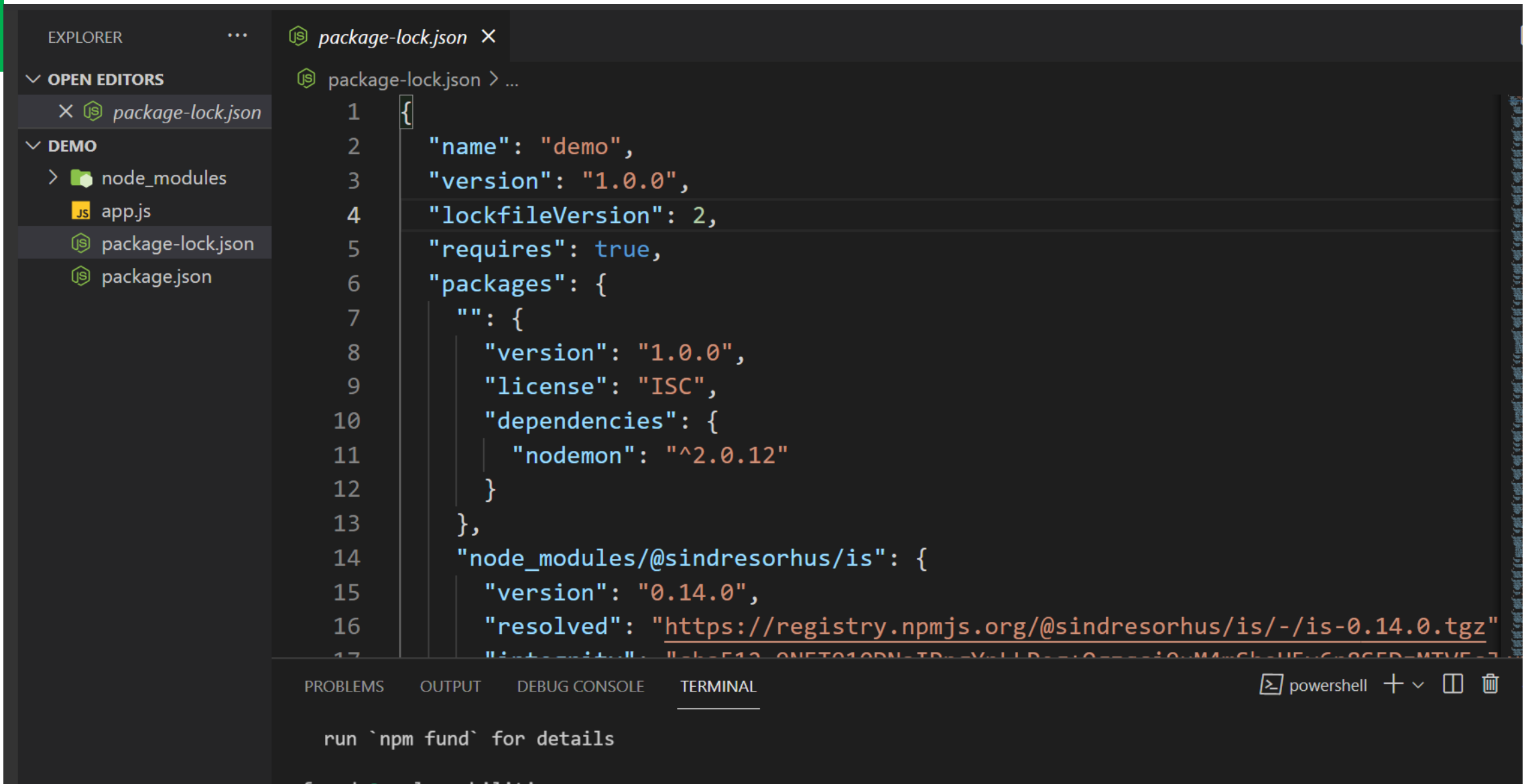
```
PS D:\demo> npm i nodemon --save

added 119 packages, and audited 120 packages in 3s

11 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\demo>
```

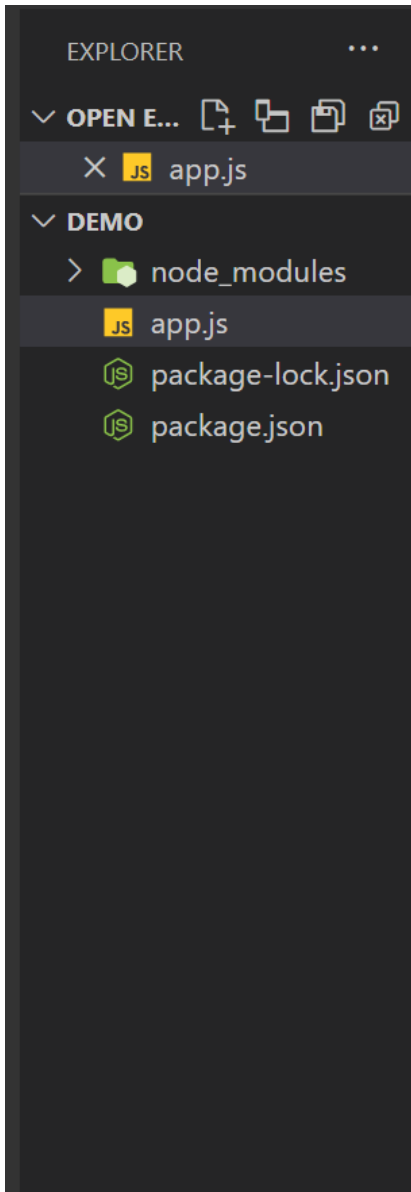




The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'DEMO' with files 'app.js', 'package-lock.json', and 'package.json'. The 'package-lock.json' file is open in the main editor. The code is a JSON object representing a lockfile. The 'requires' field is set to 'true'. The 'packages' field contains an entry for the 'is' module, which is a dependency of 'nodemon'. The entry for 'is' includes its version '0.14.0', its license 'ISC', and its resolved URL from the npm registry. The terminal at the bottom shows the command 'run `npm fund` for details'.

```
1 {
2   "name": "demo",
3   "version": "1.0.0",
4   "lockfileVersion": 2,
5   "requires": true,
6   "packages": {
7     "": {
8       "version": "1.0.0",
9       "license": "ISC",
10      "dependencies": {
11        "nodemon": "^2.0.12"
12      }
13    },
14    "node_modules/@sindresorhus/is": {
15      "version": "0.14.0",
16      "resolved": "https://registry.npmjs.org/@sindresorhus/is/-/is-0.14.0.tgz"
17    }
18  }
```

run `npm fund` for details



EXPLORER

OPEN E... [Icons]

- JS app.js
- DEMO
 - node_modules
 - JS app.js
 - package-lock.json
 - package.json


JS app.js

```
1 var http = require('http');
2
3 http.createServer(function(req,res){
4
5     res.end("Welcome to My Website 123");
6
7 }).listen(3000);
8
9 console.log("Server Started http://127.0.0.1:3000");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

run `npm fund` for details





File Edit Selection View Go Run Terminal Help app.js - demo - Visual Studio Code

EXPLORER

OPEN EDITORS

- app.js

DEMO

- node_modules
- app.js
- package-lock.json
- package.json

app.js

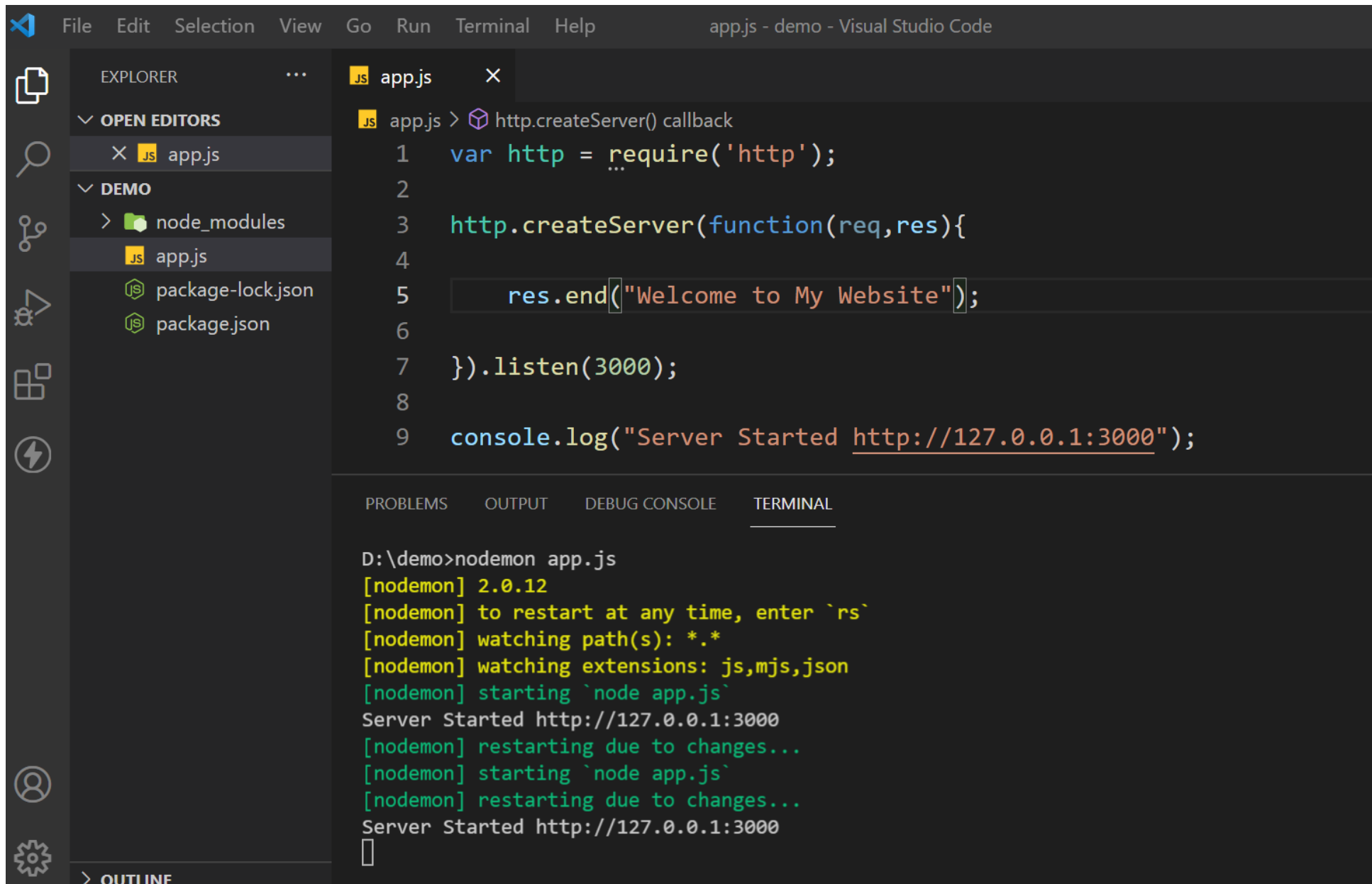
```
1 var http = require('http');
2
3 http.createServer(function(req,res){
4
5     res.end("Welcome to My Website 123");
6
7 }).listen(3000);
8
9 console.log("Server Started http://127.0.0.1:3000");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\demo>nodemon app.js
[nodemon] 2.0.12
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Server Started http://127.0.0.1:3000
█
```

OUTLINE





The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'DEMO' with a 'node_modules' directory and files 'app.js', 'package-lock.json', and 'package.json'. The code editor displays 'app.js' with the following JavaScript code:

```
app.js > http.createServer() callback
1 var http = require('http');
2
3 http.createServer(function(req,res){
4
5     res.end("Welcome to My Website");
6
7 }).listen(3000);
8
9 console.log("Server Started http://127.0.0.1:3000");
```

The terminal at the bottom shows the command 'D:\demo>nodemon app.js' and the output of the application, including the message 'Server Started http://127.0.0.1:3000' and the 'nodemon' status messages.





- `npm -v`

- `npm update`

- `npm help`

- `npm install -h`

- `npm help update`

- `npm init`

- `npm init --yes`

- `npm init --y`



Install Command

- `npm install express`
- `npm update express`
- `npm install express -g`
- `npm install express --save`
- `npm install express --save-dev`



Install multiple package

- Npm install express nodemon --save



Uninstall Command

- `npm uninstall express`
- `npm un express`
- `npm remove express`
- `npm rm express`
- `npm uninstall express -g`
- `npm uninstall express --save`
- `npm uninstall express --save-dev`



Search Command

- `npm list`
- `npm list --depth 1`
- `npm list --global true --depth 0`
- `npm list --depth 1`



Npm Package Version Check

- > npm view express version



Npm Package Version Listing

- To View Specific Package version history use below command

```
> npm view express versions
```



Example

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 10.0.17763.253]  
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Users\Akash>npm view express version  
4.16.4
```

```
C:\Users\Akash>npm view express versions
```

```
[ '0.14.0',  
  '0.14.1',  
  '1.0.0-beta',  
  '1.0.0-beta2',  
  '1.0.0-rc',  
  '1.0.0-rc2',  
  '1.0.0-rc3',  
  '1.0.0-rc4',  
  '1.0.0',  
  '1.0.1',  
  '1.0.2',  
  '1.0.3',  
  '1.0.4',  
  '1.0.5',  
  '1.0.6',  
  '1.0.7',  
  '1.0.8',  
  '2.0.0-beta',  
  '2.0.0-beta2',  
  '2.0.0-beta3',  
  '2.0.0-rc',  
  '2.0.0-rc2',
```

C:\Windows\system32\cmd.exe

```
'4.12.4',  
'4.13.0',  
'4.13.1',  
'4.13.2',  
'4.13.3',  
'4.13.4',  
'4.14.0',  
'4.14.1',  
'4.15.0',  
'4.15.1',  
'4.15.2',  
'4.15.3',  
'4.15.4',  
'4.15.5',  
'4.16.0',  
'4.16.1',  
'4.16.2',  
'4.16.3',  
'4.16.4',  
'5.0.0-alpha.1',  
'5.0.0-alpha.2',  
'5.0.0-alpha.3',  
'5.0.0-alpha.4',  
'5.0.0-alpha.5',  
'5.0.0-alpha.6',  
'5.0.0-alpha.7']
```

```
C:\Users\Akash>
```



Install Specific Version of Package

- To Install old version of Package Use Version number.
- `> npm install express@4.15.0`



Example

```
C:\Windows\system32\cmd.exe
'4.15.0',
'4.15.1',
'4.15.2',
'4.15.3',
'4.15.4',
'4.15.5',
'4.16.0',
'4.16.1',
'4.16.2',
'4.16.3',
'4.16.4',
'5.0.0-alpha.1',
'5.0.0-alpha.2',
'5.0.0-alpha.3',
'5.0.0-alpha.4',
'5.0.0-alpha.5',
'5.0.0-alpha.6',
'5.0.0-alpha.7' ]

C:\Users\Akash>npm view express version
4.16.4

C:\Users\Akash>npm install express@4.15.0
```



Npm Audit

C:\Windows\system32\cmd.exe

```
C:\Users\Akash>npm view express version  
4.16.4
```

```
C:\Users\Akash>npm install express@4.15.0
```

```
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\Akash\package.json'
```

```
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\Akash\package.json'
```

```
npm WARN Akash No description
```

```
npm WARN Akash No repository field.
```

```
npm WARN Akash No README data
```

```
npm WARN Akash No license field.
```

```
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
```

```
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
```

```
+ express@4.15.0
```

```
added 45 packages from 30 contributors and audited 41660 packages in 12.658s
```

```
found 8 vulnerabilities (3 low, 2 moderate, 3 high)
```

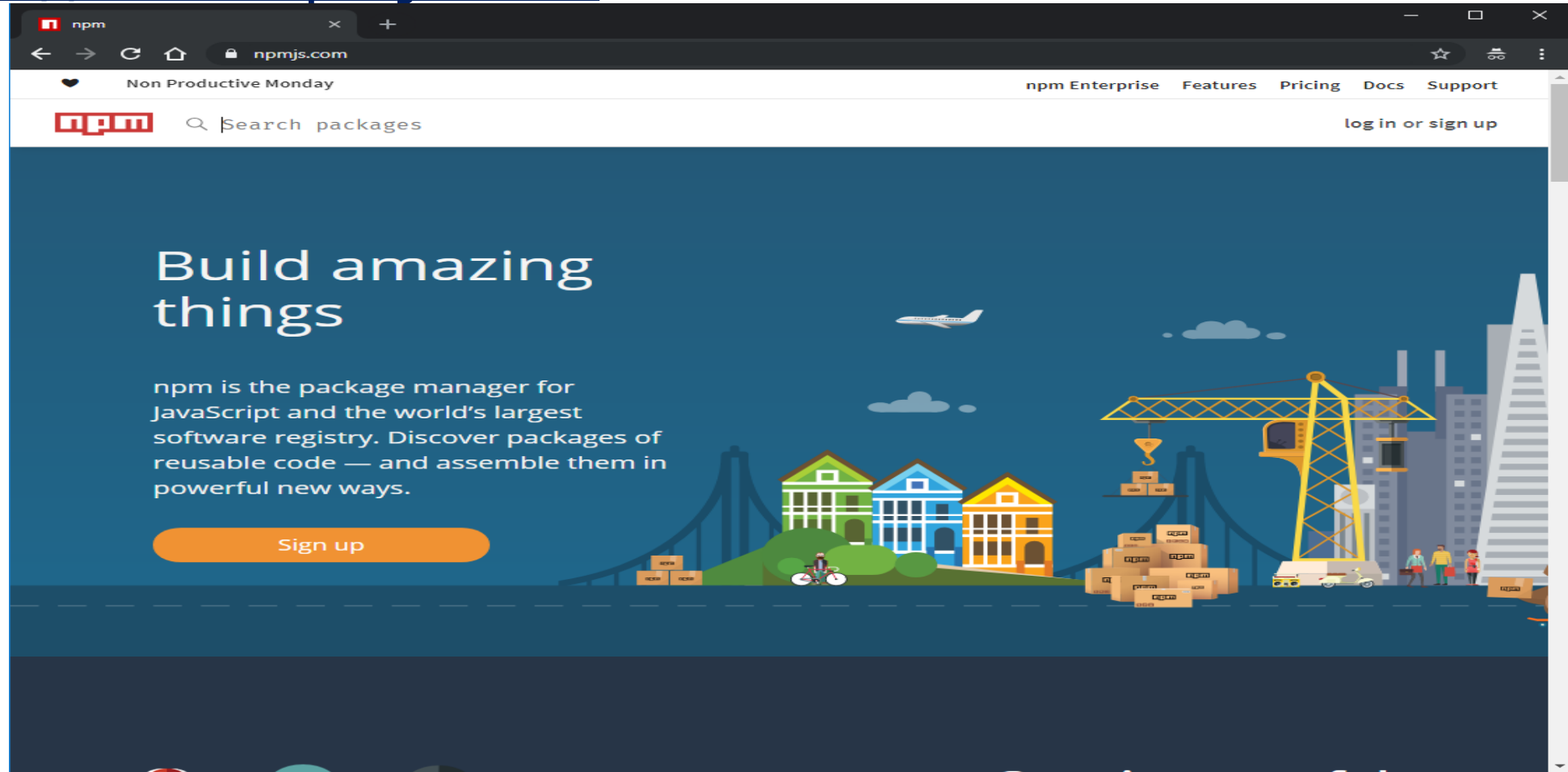
```
  run `npm audit fix` to fix them, or `npm audit` for details
```

```
C:\Users\Akash>
```



NPM Package Search Engine.

- <https://www.npmjs.com>



Nodemon Npm



Nodemon NPM

- “nodemon” module helps to automatically restart your Server once you made the changes in Source code.
- <https://nodemon.io/>
- Install Command
- `npm install -g nodemon`



How to use Nodemon:

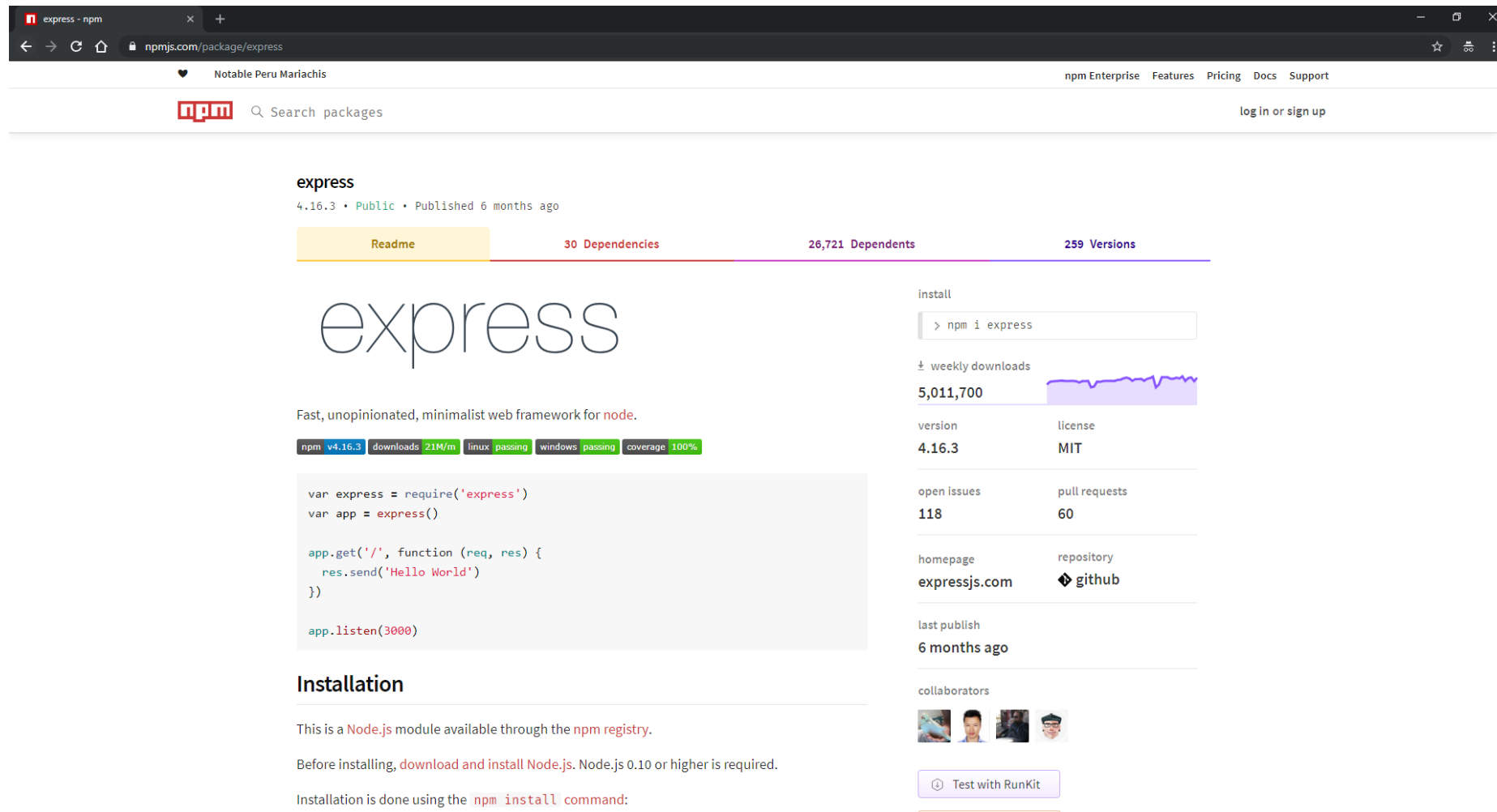
- To restart Node Server program we do this by using
 - `node Server_file.js`

but now you have to this using

- `nodemon Server_file.js`
- Nodemon will deal with starting it on Specific port and handling code updates. default the time is 1 second but if you want to change
 - `nodemon --delay 10 server.js`



Express NPM



The screenshot shows the Express.js package page on the NPM website. The page header includes the NPM logo, a search bar, and links for login or sign up. The package name 'express' is prominently displayed, along with its version '4.16.3', public status, and publication date '6 months ago'. Below this, there are tabs for 'Readme', 'Dependencies' (30), 'Dependents' (26,721), and 'Versions' (259). The 'Readme' tab is active, showing the 'express' logo and a description: 'Fast, unopinionated, minimalist web framework for node.' Below the description are badges for the current version (v4.16.3), download rate (21M/m), and platform support (Linux, Windows, and coverage). A code block shows a basic Express.js application snippet. The 'Installation' section explains that it's a Node.js module available through the NPM registry and provides the command to install it. On the right side, there's a sidebar with an 'install' button, a 'weekly downloads' chart showing 5,011,700 downloads, and a table of package metadata including version (4.16.3), license (MIT), open issues (118), pull requests (60), homepage (expressjs.com), repository (github), last publish date (6 months ago), and collaborators.

express

4.16.3 • Public • Published 6 months ago

Readme 30 Dependencies 26,721 Dependents 259 Versions

express

Fast, unopinionated, minimalist web framework for [node](#).

npm v4.16.3 downloads 21M/m linux passing windows passing coverage 100%

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Installation

This is a [Node.js](#) module available through the [npm registry](#).

Before installing, [download and install Node.js](#). Node.js 0.10 or higher is required.

Installation is done using the `npm install` command:

install

```
> npm i express
```

weekly downloads

5,011,700

| | |
|-------------------------------|------------------------|
| version | license |
| 4.16.3 | MIT |
| open issues | pull requests |
| 118 | 60 |
| homepage | repository |
| expressjs.com | github |
| last publish | |
| 6 months ago | |
| collaborators | |

Test with RunKit





Get Exclusive Video Tutorials



www.apptutorials.com

<https://www.youtube.com/user/Akashtips>





Get More Details

www.akashsir.com



If You Liked It !

Rating Us Now



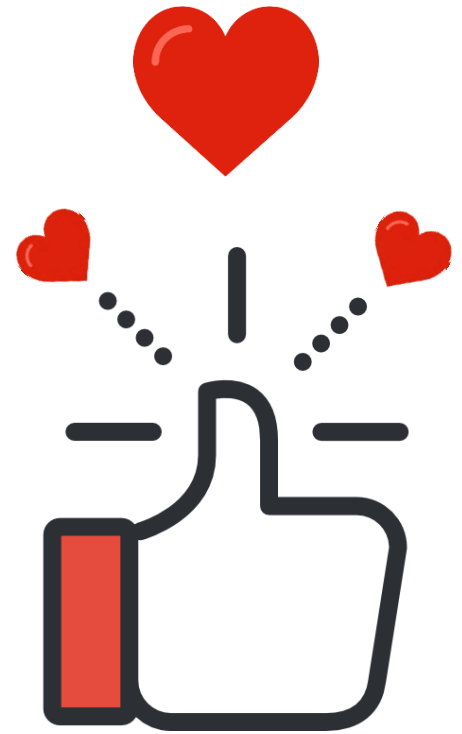
Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET



Sulekha

<https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad>



Connect With Me



Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechlabs.com
www.akashpadhiyar.com
www.apptutorials.com

Social Info



Akash.padhiyar



Akashpadhiyar



Akash_padhiyar



+91 99786-21654



#Akashpadhiyar
#apptutorials