

SETTING UP A NODE PROJECT

Intention:

The goal of this document is to walk you through how to setup a project with Node.js. Node.js relies on a project file called package.json. Every Node project has a package.json. This file is similar to a Manifest files used by Java or C#. It is possible Visual Studio has created a Manifest file for you in languages like C#, but in Node, we will create our own manually for now.

The package.json file is used by the Node Package Manager (NPM). NPM is one of Node's internal tools and is installed as a command line command alongside Node itself. NPM controls a Node project, the packages/libraries that project uses and linking of files.

Getting Started:

1. If you are on the lab machines, skip to step 2. If you are on your own machine, you will need to install node/npm. They are installed together.

<https://nodejs.org/en/> (get current, not LTS).

2. In the folder of your project, you will need to create the package.json. It must be in the top level directory of your Node project. That is the folder with all of the project code in it.

To do this, open a command/terminal window in that folder and run "npm init" (without quotes). The npm init command creates (initializes) a new node project. The init command asks basic questions about the project such as the name, description, main program file (entry point), git repo, author, license, etc. This commands creates a package.json for the application. Inside of the package.json file created by npm init, there is a section for scripts. The script section holds scripts used by node under certain commands.


On Windows, you can hold shift and right click inside of the folder to choose "Open Powershell Window Here".

On Mac/Linux, open a terminal window. Then type "cd" (without quotes), followed by a space and then drag in the folder you want to access. "cd" is change directory. That command will bring you into the folder you want to work in.

SETTING UP A NODE PROJECT

3. Once you have run the command "npm init" to initialize a new project, you will be presented with the following options. When there is a value in parenthesis, that is the default value that will be entered if you don't enter anything.

For example, in the images below, it says "version: (1.0.0)". I manually retyped 1.0.0, but I could have also typed nothing and just pressed enter to set the version number to 1.0.0.

 Windows PowerShell

```
PS /NewNode> 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 init` 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: (newnode) new-project-name
version: (1.0.0) 1.0.0
description: My example for setting up a node project
entry point: (index.js) ./path/to/your/main/js/file.js
test command:
git repository:
keywords:
author: Austin Willoughby
license: (ISC) UNLICENSED
```

About to write to C:\Users\Austin\Downloads\NewNode\package.json:

```
{
  "name": "new-project-name",
  "version": "1.0.0",
  "description": "My example for setting up a node project",
  "main": "./path/to/your/main/js/file.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Austin Willoughby",
  "license": "UNLICENSED"
}
```

```
Is this OK? (yes)
PS /NewNode>
```

SETTING UP A NODE PROJECT

"npm init" will have created a package.json file in the folder. The package.json can be created by hand, but it is less error prone to use "npm init". This is where all of the information about the project is stored.

Read more here:

<https://docs.npmjs.com/cli/v8/configuring-npm/package-json>

```
{
  "name": "new-project-name",
  "version": "1.0.0",
  "description": "My example for setting up a node project",
  "main": "./path/to/your/main/js/file.js",
  ▶ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Austin Willoughby",
  "license": "UNLICENSED"
}
```

4. Once the package.json exists, we will make some slight changes to it. We are going to use the *scripts* section. The *scripts* section contains command line scripts (operating system specific) that will allow node to run tasks automatically. This is very important for when working with other developers or with servers (such as Heroku or Amazon), who will rely on a "start" script

Inside of the scripts section, we want to change the test script to just print out "no tests" unless you actually have unit tests to run. You will need to use escape characters to include the quotes correctly. We are using the operating system agnostic command "echo" which will print to the command window regardless of your OS.

```
{
  "name": "new-project-name",
  "version": "1.0.0",
  "description": "My example for setting up a node project",
  "main": "./path/to/your/main/js/file.js",
  ▶ Debug
  "scripts": {
    "test": "echo \"No Tests\""
  },
  "author": "Austin Willoughby",
  "license": "UNLICENSED"
}
```

SETTING UP A NODE PROJECT

5. Once you have changed the test script, save the file. Copy the json inside of the file and test it as <http://jsonlint.com/> . Make sure it is valid JSON. If there are errors, correct them before moving on.
6. Once it is valid JSON and the file is saved, run the "npm test" command inside of the Powershell/terminal window (in the same directory as the package.json).

You should see it print out the new message you put into your package.json test script.

```
PS /NewNode> npm test  
  
> new-project-name@1.0.0 test C:\Users\Austin\Downloads\NewNode  
> echo "No Tests"  
  
"No Tests"  
PS /NewNode>
```

By running "npm test" you are telling the Node Package Manager (npm) to run this project's test script. It will look in the package.json for a script called "test" and then run the operating system commands you put in there. This can be quite powerful. In our case, there are no tests to run.

7. Now we will use the "start" script. Add a start script that points to the main file you specified in your package.json. This file may or may not have been created yet.

Add a "start" script to the *scripts* section. This script will run the node command and then the file you want to start. This is the command you would type on the command line to start your node server. We put this into a start script so that we can just run "npm start" and it will start our server. This means we could pass the package.json to other developers or to a server (such as Heroku), and they will just be able to run "npm start" without knowing custom information about your project. Most servers that run node projects will automatically run "npm start" to start your server. Otherwise, they would need to know the exact file paths and folder structure of your project. That would be inefficient and harder to make portable code.

This "start" script will save other developers time and allow your code to be used on servers.

SETTING UP A NODE PROJECT

```
{
  "name": "new-project-name",
  "version": "1.0.0",
  "description": "My example for setting up a node project",
  "main": "./path/to/your/main/js/file.js",
  ▶ Debug
  "scripts": {
    "start": "node ./path/to/your/main/js/file.js",
    "test": "echo \"No Tests\""
  },
  "author": "Austin Willoughby",
  "license": "UNLICENSED"
}
```

8. Save your file. Copy the json inside of the file and test it as <http://jsonlint.com/> . Make sure it is valid JSON. If there are errors, correct them before moving on.
9. IF your main file already exists, try to run "npm start" from the command line to see if the "start" script runs successfully. If it runs your code, then you did it correctly. If the file is missing or the path is wrong, you will get a fatal error.

SETTING UP A NODE PROJECT

10. Now we need to add the *engines* section to the package.json. The *engines* field is a special field that allows you to specify which version of Node and NPM to run. This is particularly useful on servers that have many versions of Node installed simultaneously (such as Heroku or Amazon).

Many server hosts will default to a certain Node version (usually the stable LTS version, not the current cutting-edge version). For that reason, we need to add an engines field to make our code run on server hosts.

Go ahead and add an engines field and put in the appropriate versions of node and npm. You can check your node & npm versions with the command line commands "node -v" and "npm -v"

Please make sure you put in the version numbers of node and npm that you are using for your project. The ones below are just examples at the time of writing this. They may NOT match the node version you wrote your code for. Failure to do this will result in your code failing to run online. !

```
{
  "name": "new-project-name",
  "version": "1.0.0",
  "engines": {
    "node": "14.15.4",
    "npm": "6.14.10"
  },
  "description": "My example for setting up a node project",
  "main": "./path/to/your/main/js/file.js",
  ▶ Debug
  "scripts": {
    "start": "node ./path/to/your/main/js/file.js",
    "test": "echo \"No Tests\""
  },
  "author": "Austin Willoughby",
  "license": "UNLICENSED"
}
```