***NPM & NODE-JS***

***Day-1:***

* **Introduction to NPM**

1. What is **npm**?

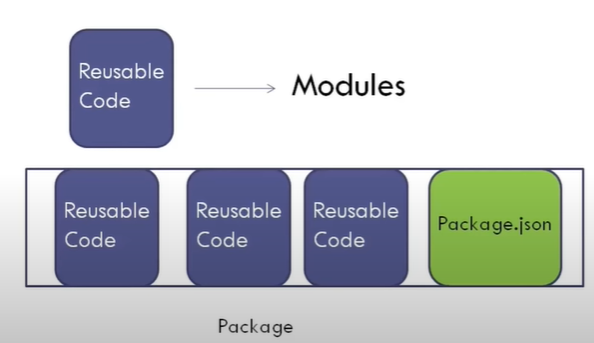
* **npm** (Node Package Manager) is the package manager for JavaScript.

1. Package Manager

* Problem 🡺 Code
* Same problem 🡺 Reuse the Code

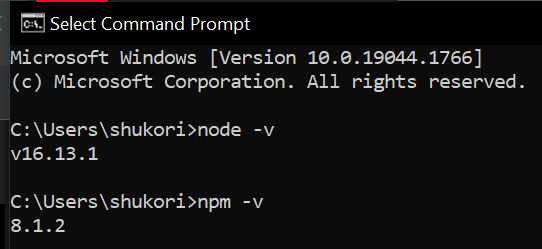
1. Modules and Package

* Individual files containing reusable code are called modules.
* Package is a directory with one or more modules and along with a special file called package.JSON
* package.JSON contains metadata about the package.



* **Getting Started with NPM**

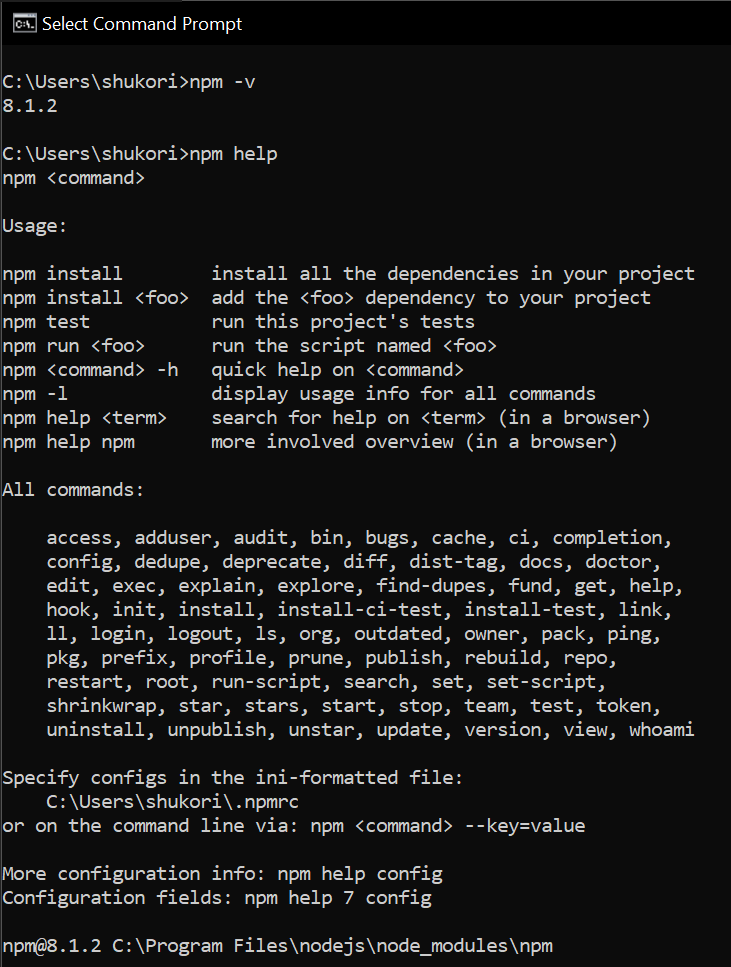
1. **NPM** is shipped with node.js so in order to have NPM on our machines we need to download and install node.js.
2. **node - v** gives the version of node installed on our system which in my system is **v16.13.1.**
3. Now that we have node installed, we can run **npm -v** and this gives the version of **npm** on our machines which in my system is **8.1.2.**



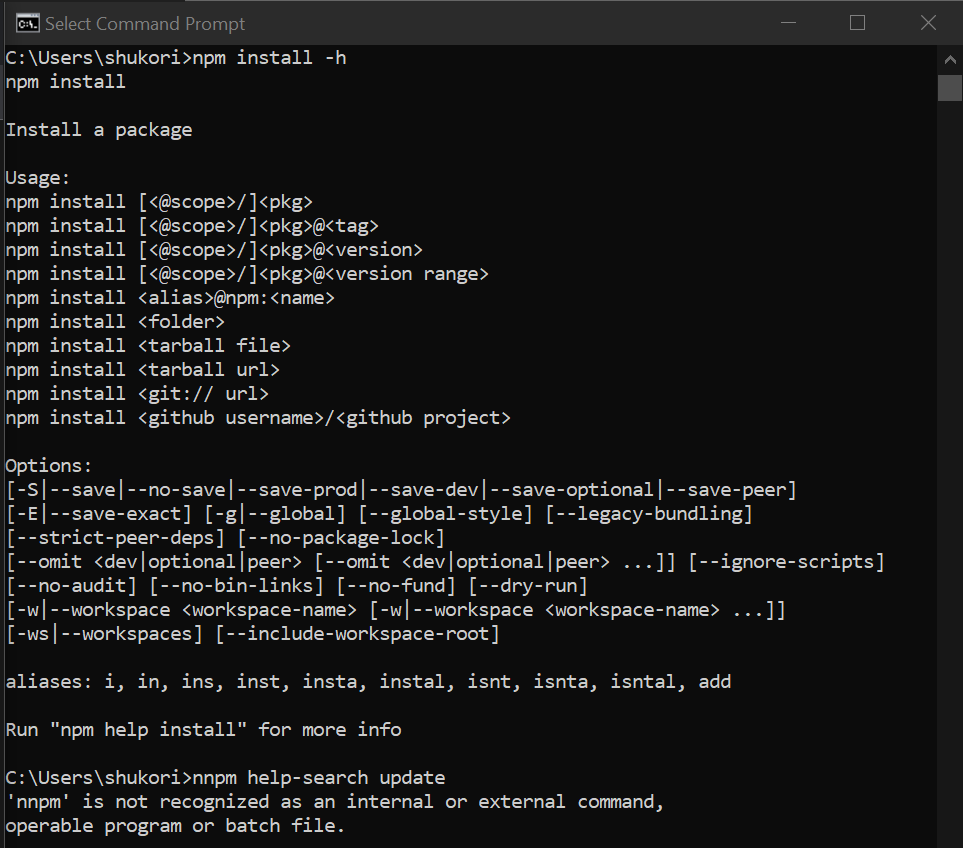
* **NPM Help**

1. **npm -help** shows us the typical usage of **npm**, i.e., it is **npm** followed by a command name

**npm <command>**

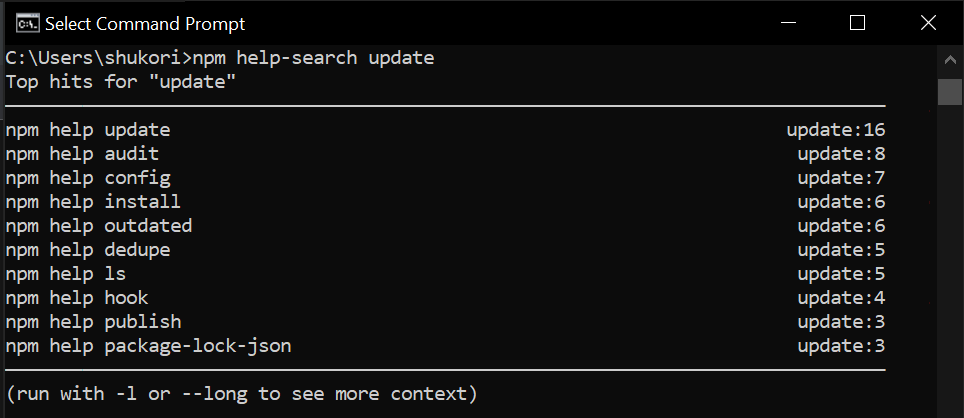
****

1. **npm install -h** gives us the different ways we can use **npm** **install**.



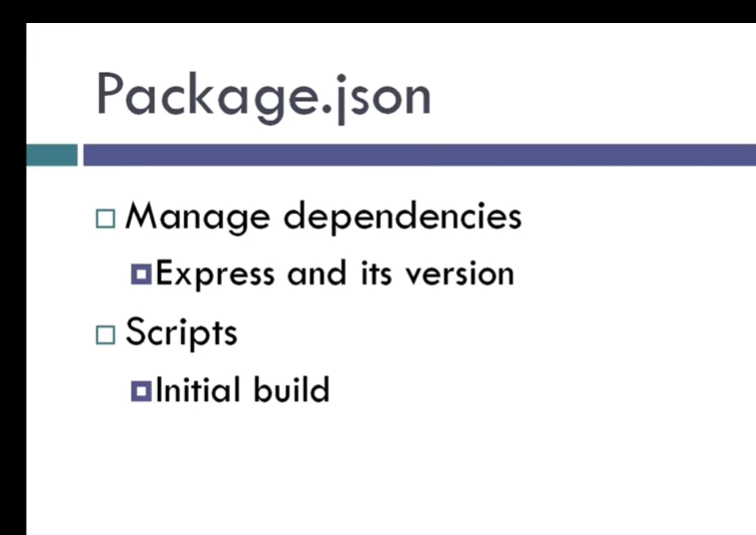
1. **npm help-search <text>:** it allow us to specify some text and it is going to search the **npm** documentation for that text.

Here is an example of npm **help-search update,** when we run this command, it gives us the **top hits** for the text update and their corresponding documentation in the browser.



* **Package JSON**

1. When we work on a project we are going to include a file called **package.json** in our project folder.
2. Using a **package.json** file has two main benefits:
   1. It lets us manage the dependencies of our project, for example, if I work on an old project that makes use of the Express framework then I can specify the Express package and the version of Express that my project depends on.
   2. It lets you add scripts that help with initial build of your project.



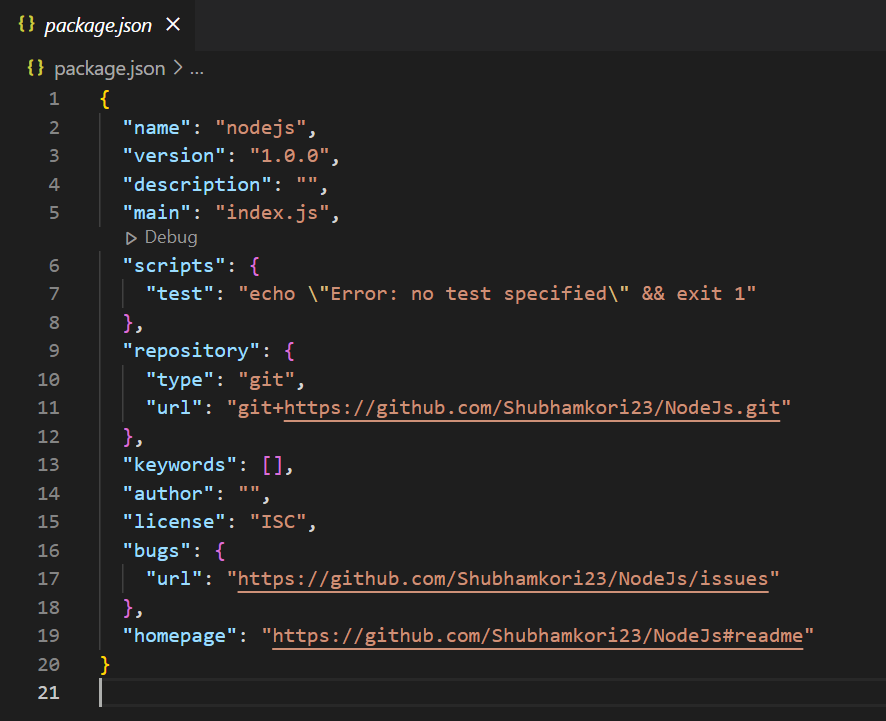
1. How to create a **package.json** file?

* We can use the “**npm init” or** “**npm init --yes”** command to create a **package.json** file.

1. “**npm init” vs** “**npm init --yes”:**

* “**npm init”** command asks us a series of questions but **npm** itself gives us the default answer to every question. Also, it gives the preview of how **package.json** file is going to look.
* “**npm init --yes”** command skip through the questions and just creates the **package.json** file.

1. Below is the **package.json** file which we created:

[](https://github.com/Shubhamkori23/NodeJs/blob/main/package.json)

* **Package JSON Default**

1. How to set a default in our **package.json** file?

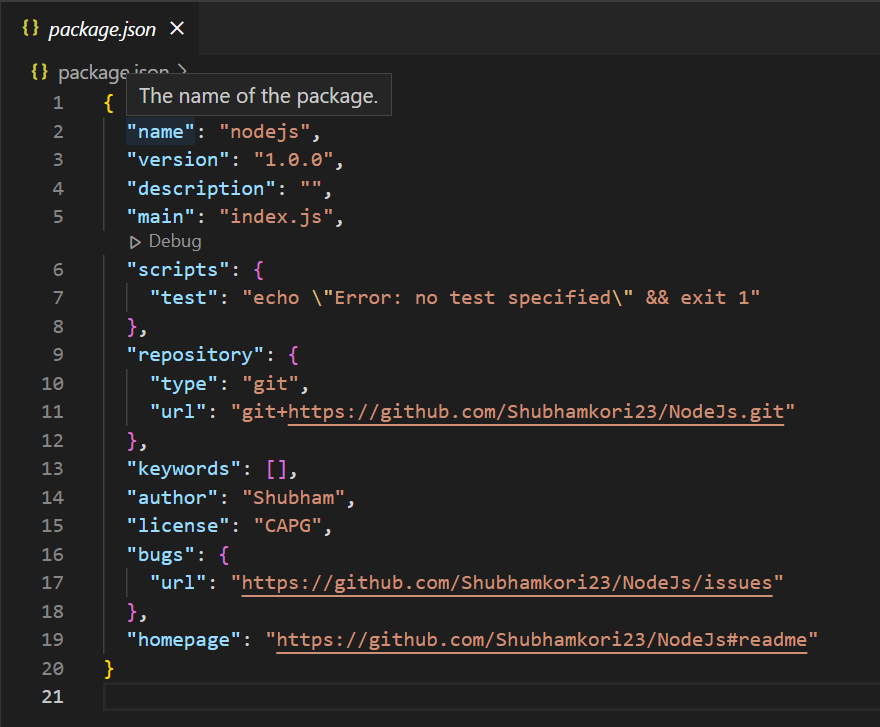
* For Example, for every project I want the **author** to be **“Shubham”** and **license** to be **“CAPG”**, so, to set a **default author name** and a **default license** for our **package.json** file, the following commands are used:

🡺 **npm config set init-author-name “Shubham”:** The default value of author has been set in the **package.json**.

🡺 **npm set init-license “CAPG”:** The default value of the license has been set in the **package.json.**

(***Note:*** *The “****config****” keyword can be emitted to set the default value.*)

1. The following snippet shows the **package.json** file after setting the ***default author*** as “**Shubham**” and ***default license*** as “**CAPG**”.

[](https://github.com/Shubhamkori23/NodeJs/blob/6f48fc5c862bc3b6559de74b389bf173ab7c60b8/package.json)

1. To check the default values that have been set, we can use the “***get***” keyword, so, the command is as follows:

**🡺npm config get init-author-name** gives me “**Shubham”**

**🡺npm get init-license** gives me “**CAPG**”

1. To delete the default values, the following command is used:

🡺**npm config delete init-author-name**

**🡺npm config delete init-license**

* **Installing Local Package**

1. How to install package locally?

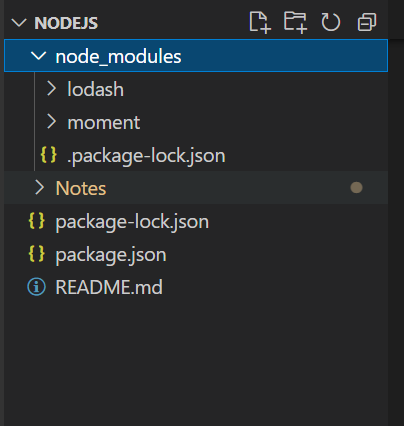
* Following is the command to install package:

🡺**npm install <package\_name>**

**🡺npm install <package\_name> --save-dev (***To install package as**development dependency package***)**

1. When we install a package “***node\_modules***” folder gets created in out project folder and within this folder, we can see the packages we installed.

As I have installed ***moment*** and ***lodash*** package, they are reflecting in the ***node\_modules*** folder in the project directory.

****

* **Uninstalling Local Package**

1. How to uninstall package?

* Following is the command to uninstall package:

🡺**npm uninstall <package\_name>**

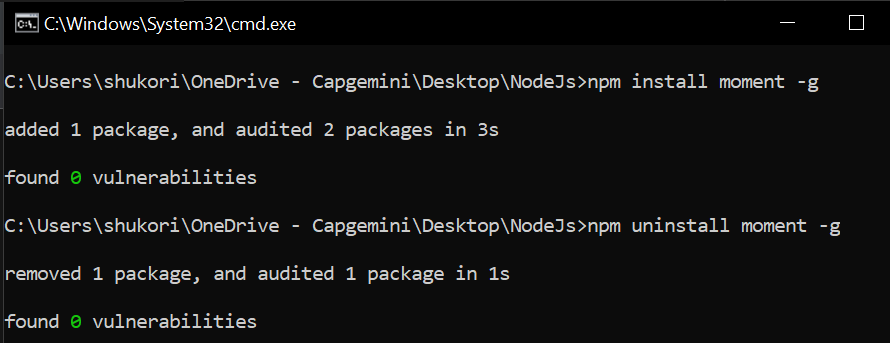
**🡺npm uninstall <package\_name> --save-dev (***To uninstall development dependency package***)**

* **Installing Global Package**

1. How to install package globally?

* Following is the command to install package:

🡺 **npm install <package\_name> -g**

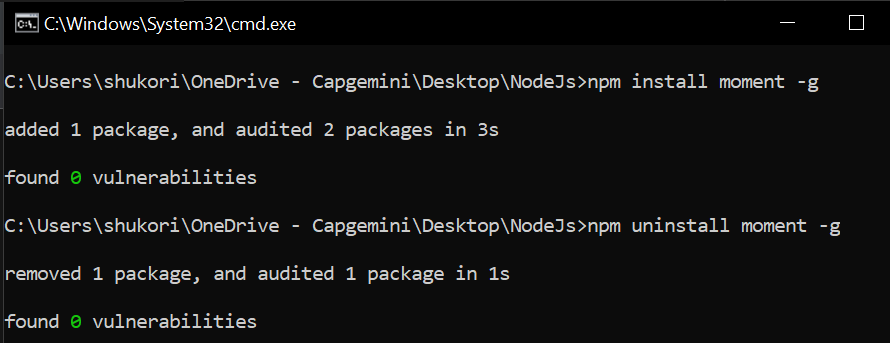
****

* **Uninstalling Global Package**

1. How to uninstall package globally?

* Following is the command to uninstall package:

🡺**npm uninstall <package\_name> -g**

****

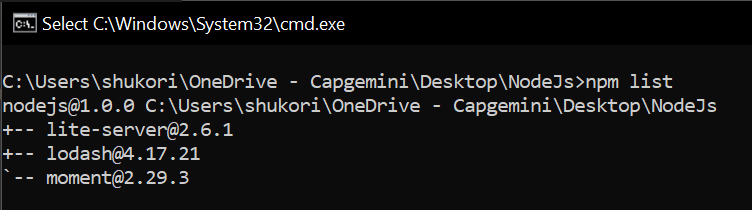
**Aliases for Uninstall Command:**

* **npm remove <package\_name>**
* **npm rm <package\_name>**
* **npm un <package\_name>**
* **Listing Packages**

1. How to list the locally and globally installed packages?

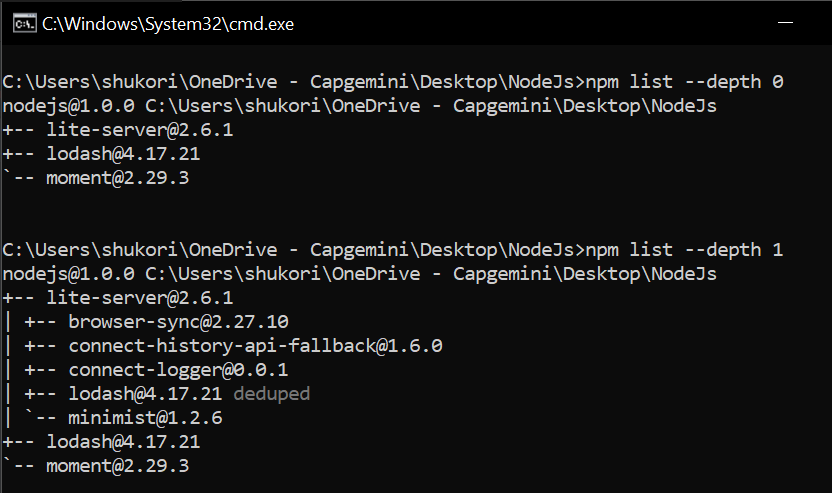
* Following commands are used to list the locally and globally installed packages:

🡺 **npm list:** It displays all packages that are installed locally.

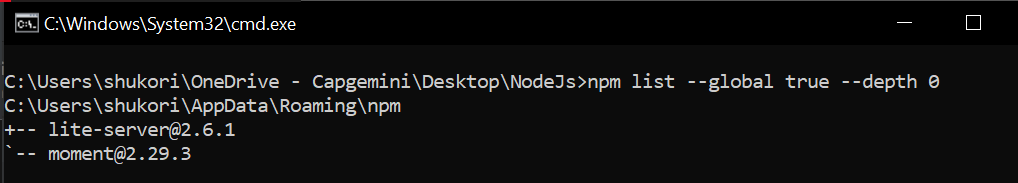


🡺**npm list --depth 0:** It displays packages at the top level of tree.

🡺 **npm list --depth 1:** It restrict the depth of the tree using this command.



🡺 **npm list --global true --depth 0:** It gives the list of packages that we have installed globally.



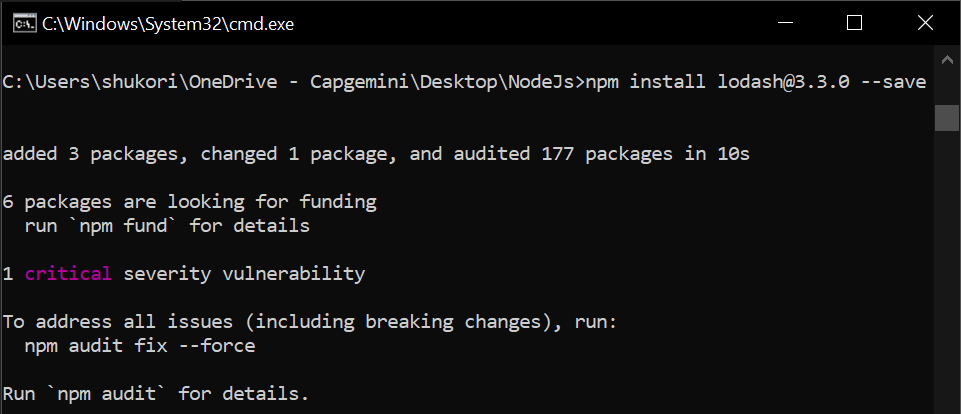
* **NPM Versioning**

1. What is **npm** versioning?

* It is used to change the version of packages.

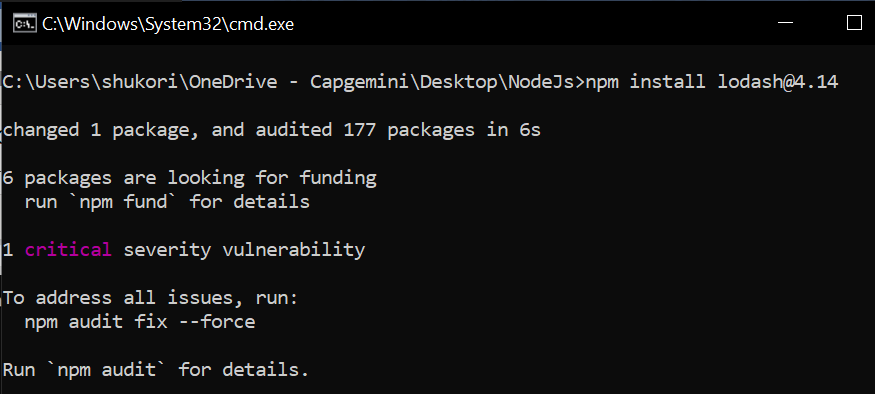
1. Following Commands are used for versioning:

* **npm install <package\_name>@version**
* **e.g., npm install lodash@3.3.0:** It install the specified version.



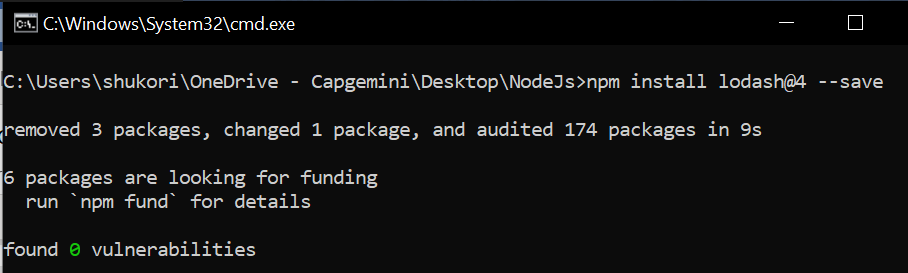
* **e.g., npm install lodash@4.14:** It install the latest patchversion of **lodash@4.14**

***i.e., lodash@4.14.2***.



* **e.g., npm install lodash@4:** It install the latest version of **lodash@4**

***i.e.,* *lodash@4.17.21*.**



* **Updating Packages**

1. To update the packages, following commands are used:

🡺 **npm update:** updates all packages locally.

🡺 **npm update <package\_name>:** update the specified package.

🡺 **npm update --dev –save-dev:** updates the dev dependencies package.

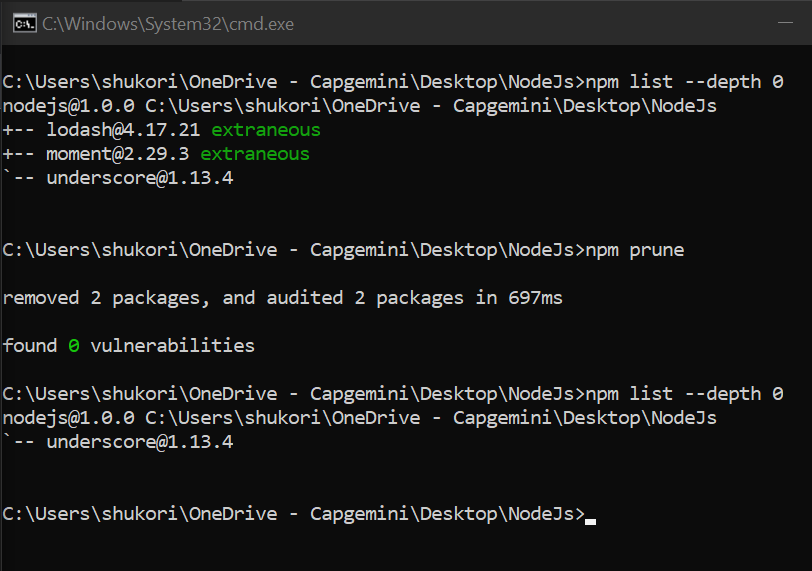
🡺 **npm update -g:** updates all packages globally.

🡺 **npm update -g <package\_name>:** updates the specified global package.

🡺 **npm install npm@latest -g:** updates the npm to its latest version.

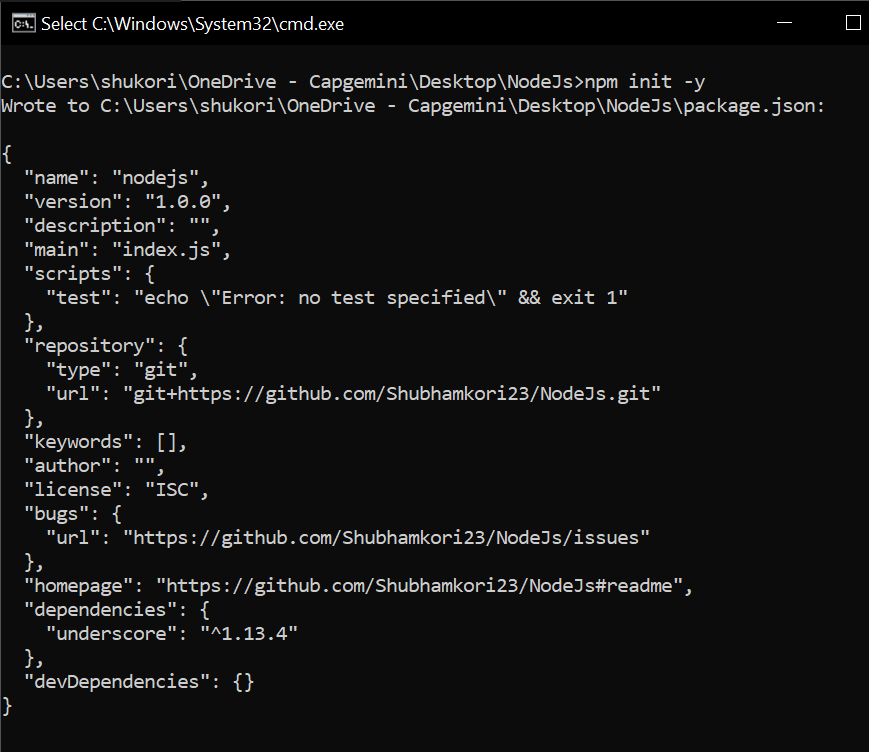
* **NPM Prune**

1. **npm prune** command is used to remove all the extraneous packages from our project.

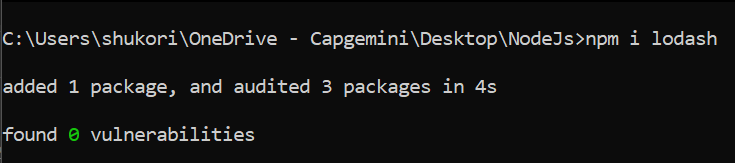


* **Shortcuts**

1. Creating a package.json file with default value



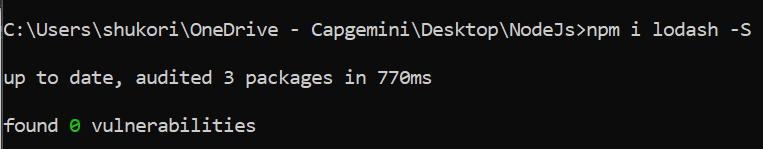
1. Install package locally



1. Save package in package.json

Before – npm install lodash –save

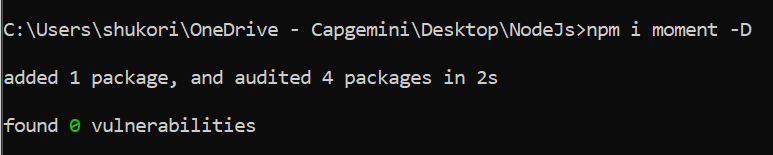
After- npm i lodash -S



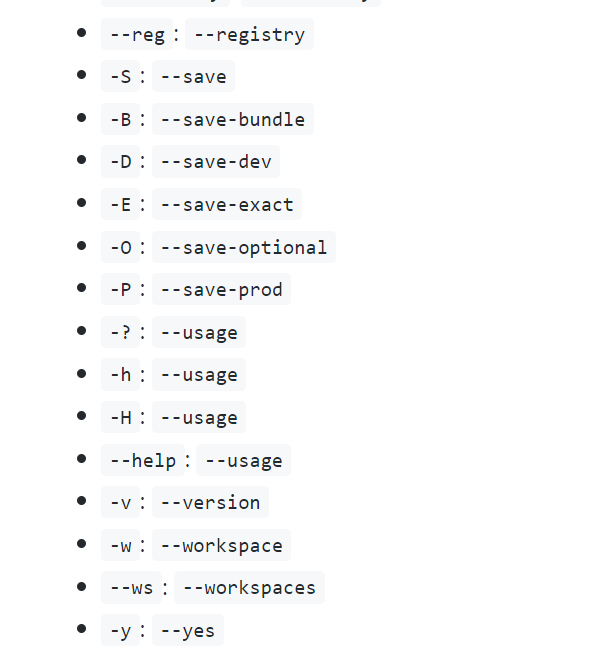
1. Save a package as a dev dependency

Before – npm install moment –save-dev

After – npm I moment -D

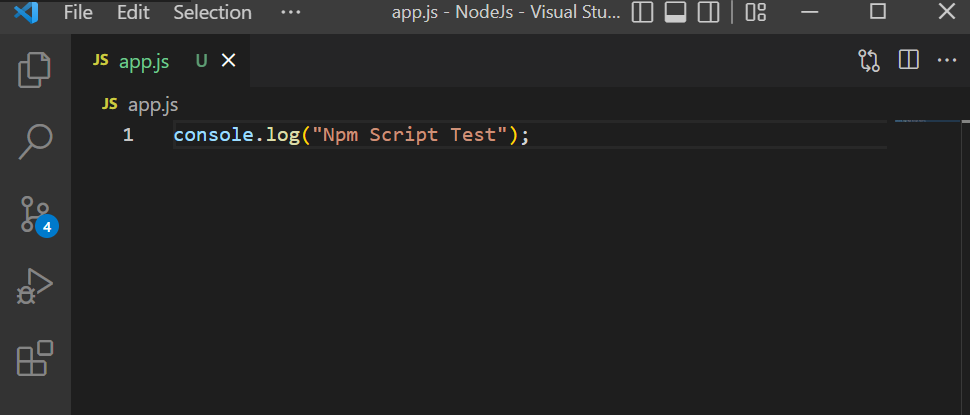


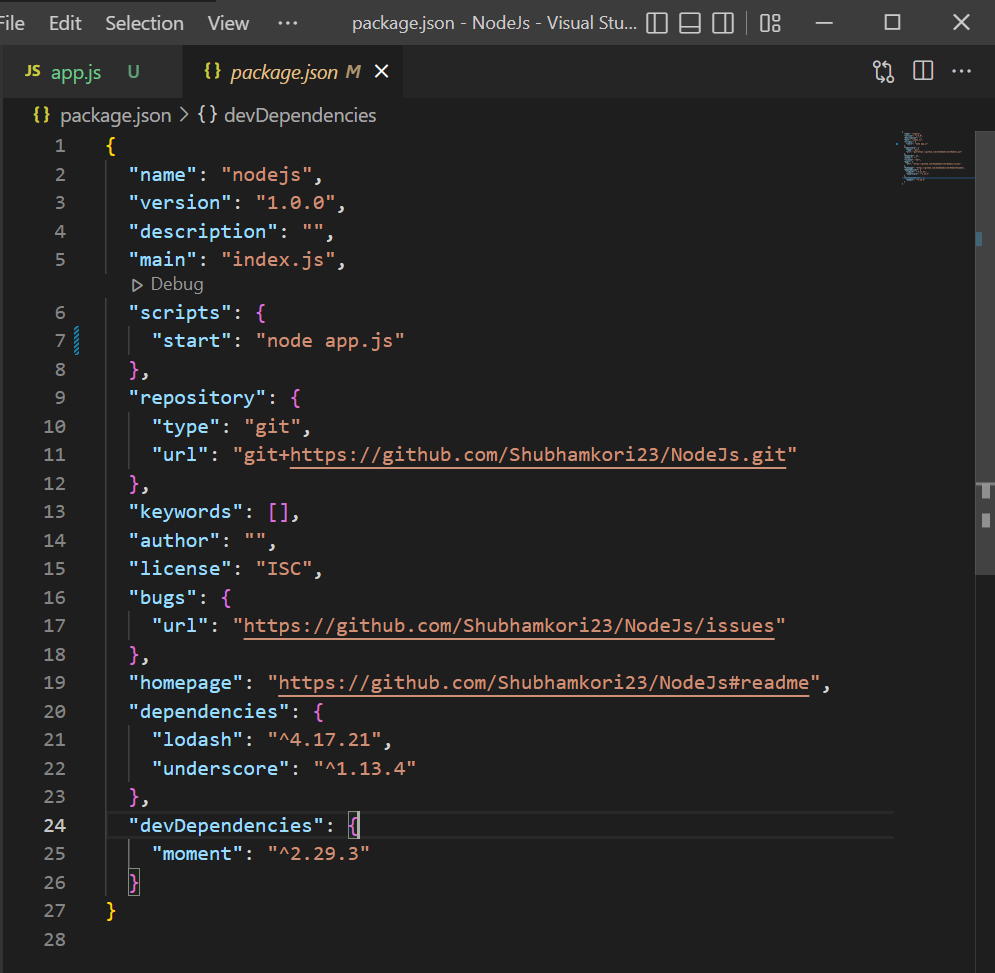
1. Some More Shorthand’s are:

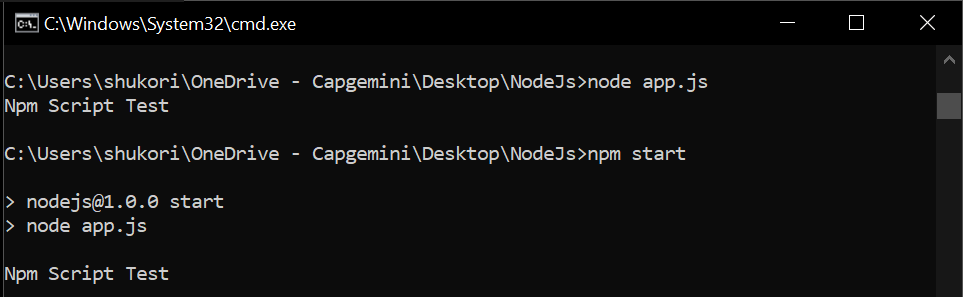


* **NPM Scripting**

1. First, we change the scripts like node app.js then create app.js file and enter some code inside this file and then go to command prompt and run it by using commands like: **npm start node app.js**.







---------------------------------------------------------------------------------------------------------------------------