

NPM TUTORIAL

Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well.

npm consists of three distinct components:

- the website
 - the Command Line Interface (CLI)
 - the registry
-
- Creating a new user account on the public registry:
Creating an account on the website:

1. Go to the [npm signup page](#)
2. In the user signup form, type in the fields.
3. Read the [End User License Agreement](#) and [Privacy Policy](#), and indicate that you agree to them.
4. Click **Create An Account**.

Note: After signing up for an npm account, you will receive an account verification email. You must verify your email address in order to publish packages to the registry

- **Testing your new account with npm login**

- On the command line, type the following command:
 - `npm login`
 - When prompted, enter your username, password, and email address.
 - If you have [two-factor authentication](#) enabled, when prompted, enter a one-time password.
 - To test that you have successfully logged in, type:
 - `npm whoami`
- Your npm username should be displayed.

Two-factor authentication on npm

Authorization and writes

By default, 2FA is enabled for authorization and writes. We will request a second form of authentication for certain authorized actions, as well as write actions.

Action	CLI command
Log in to npm	<code>npm login</code>
Change profile settings (including your password)	<code>npm profile set</code>
Change 2FA modes for your user account	<code>npm profile enable-2fa</code> <code>auth-and-writes</code>
Disable 2FA for your user account	<code>npm profile disable-2fa</code>
Create tokens	<code>npm token create</code>
Revoke tokens	<code>npm token revoke</code>
Publish packages	<code>npm publish</code>
Unpublish packages	<code>npm unpublish</code>
Deprecate packages	<code>npm deprecate</code>
Change package visibility	<code>npm access public/restricted</code>

Change user and team package access	<code>npm access grant/revoke</code>
Change package 2FA requirements	<code>npm access 2fa-required/2fa-not-required</code>

Authorization only

If you enable 2FA for authorization only. We will request a second form of authentication only for certain authorized actions.


Action	CLI command
Log in to npm	<code>npm login</code>
Change profile settings (including your password)	<code>npm profile set</code>
Change 2FA modes for your user account	<code>npm profile enable-2fa auth-only</code>
Disable 2FA for your user account	<code>npm profile disable-2fa</code>
Create tokens	<code>npm token create</code>
Revoke tokens	<code>npm token revoke</code>

Configuring two-factor authentication

Configuring 2FA from the website

Enabling 2FA

1. On the npm "[Sign In](#)" page, enter your account details and click **Sign In**.



Login

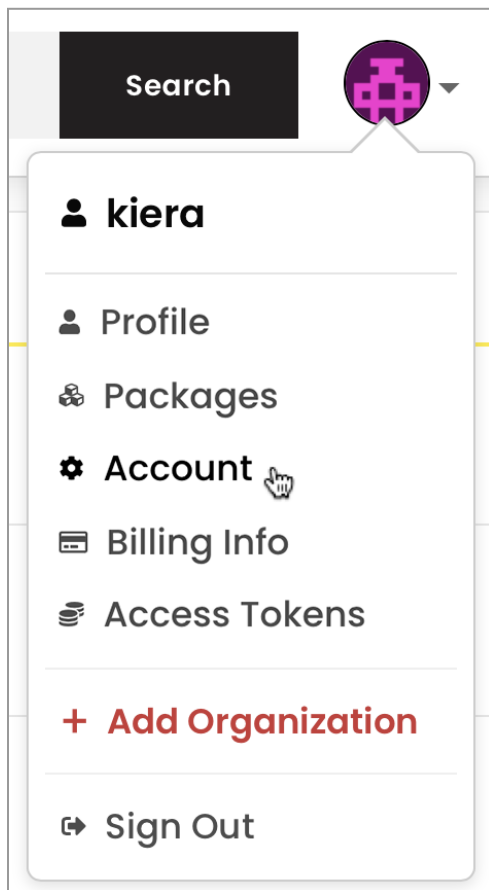
Username

Password [Forgot password?](#)

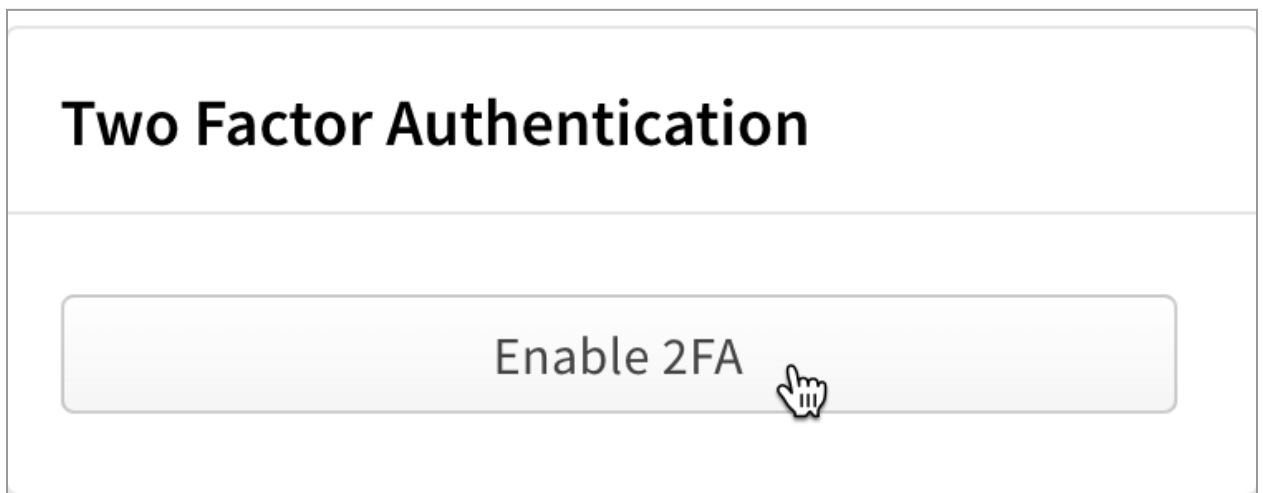
Login

[Create Account](#)

2. In the upper right corner of the page, click your profile picture, then click **Account**.



3. On the account settings page, under "Two-Factor Authentication", click **Enable 2FA**.



4. When prompted provide your current account password and then click **Confirm password to continue**.
5. On the 2FA method page, select the method you would like to enable and click **Continue**. For more information on supported 2FA methods, see "[About two-factor](#)"

authentication".


Two-Factor Authentication

LOGIN

2FA METHOD

CONFIGURE


How should we protect it?



Security key

Use a physical security key over USB or NFC, fingerprint reader, facial recognition, or password/PIN as a security key.

☒



Authenticator app

Use an application to get two-factor authentication codes when prompted.

☐

Continue

6. Configure the 2FA method of your choice:

- When using a **security-key**, provide a name for it and click **Add security key**. Follow the browser specific steps to add your security-key.

Two-Factor Authentication

LOGIN

2FA METHOD

CONFIGURE



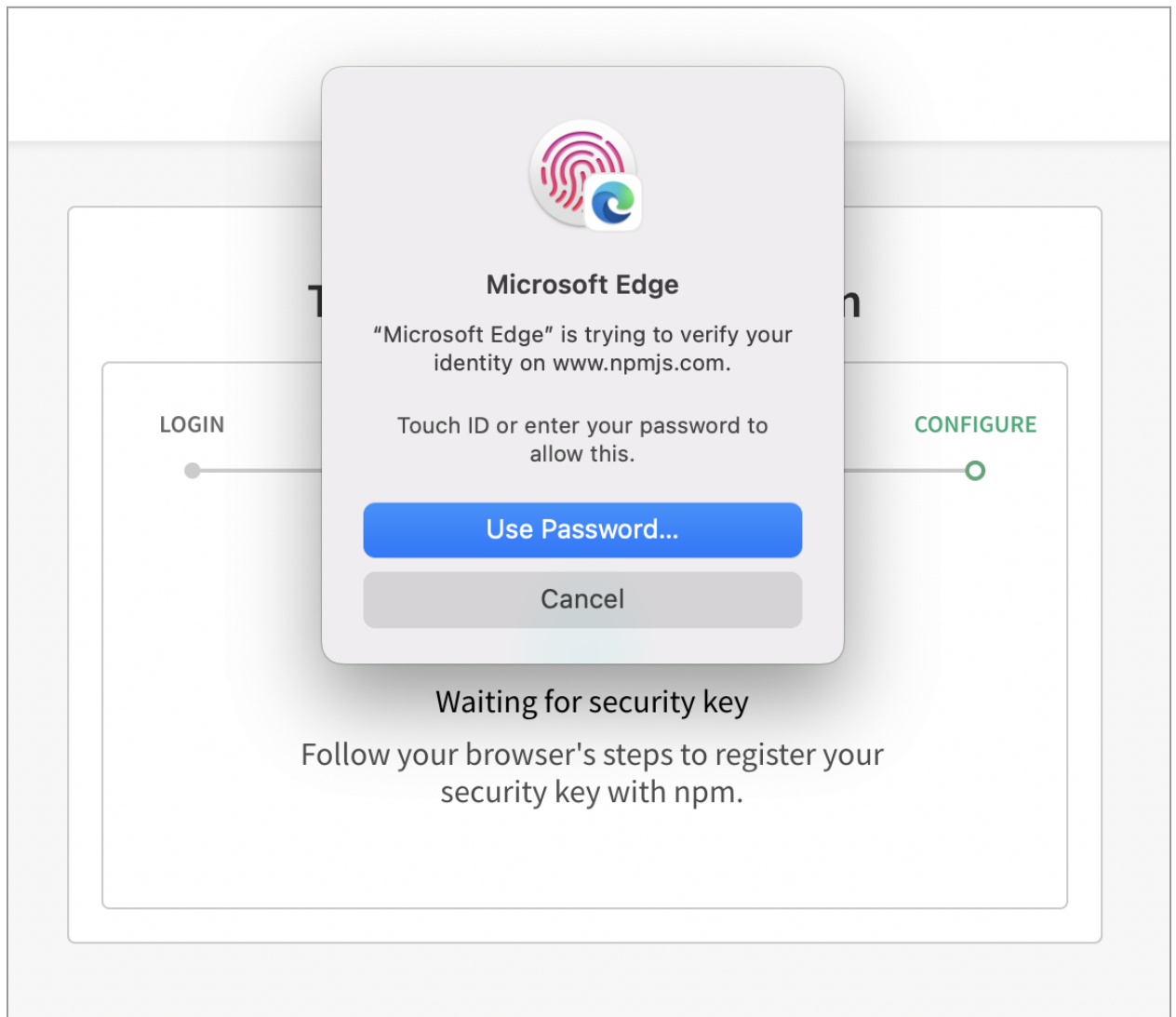
Add your security key

Give it a name so you can identify it in your list of devices.

Enter your security key name

Add security key

Below is an example of configuration from Microsoft Edge running on a MacOS



- When using an **authenticator application** on your phone, open it and scan the QR code on the two-step verification page. Enter the code generated by the app, then click **Verify**.


Enter Code

123456

Verify

7. On the recovery code page, copy the recovery codes to your computer or other safe location that is not your second factor device. We recommend using a password manager.

2FA Successfully Enabled



Recovery Codes

Please make sure you have saved your codes in a secure place before leaving this page.

```
bea8f8c6187c87b5ec275e99c085a21c0e4c6bd035605c3ba59ed3ed90730ec0
b3852c6b2bc8cc5dff8f69110def69e387f3e53d2a72e732b7dcc842aff0f16c
4250dd8761465f54603d7eeba161a4cbc7bf2019d28a5aa604d72f46e52f5e97
efdf744c79d3616f80a28c0222c51b4d4af5991c31676550d2e62f958ae30d8a
b7840aaca5ee6055550fe6962c81c79916b65e4374da3f0e037ff024bb1bfe12
```

Copy

Download

Print

☒ I confirm I have saved my codes

Go back to settings

Recovery codes are the only way to ensure you can recover your account if you lose access to your second factor device. Each code can be used only once. You can [view and regenerate your recovery code](#) from your 2FA settings page. For secondary account recovery options, see "[Configuring account recovery options](#)."

8. Click **Go back to settings** after confirming that you have saved your codes.

Disabling 2FA

If you have 2FA enabled, you can remove it from your account settings page.

Note: You cannot remove 2FA if you are a member of an organization that enforces 2FA. You can view the list of organizations memberships from your profile page under the "Organizations" tab.

1. On the npm "[Sign In](#)" page, enter your account details and click **Sign In**.



Login

Username

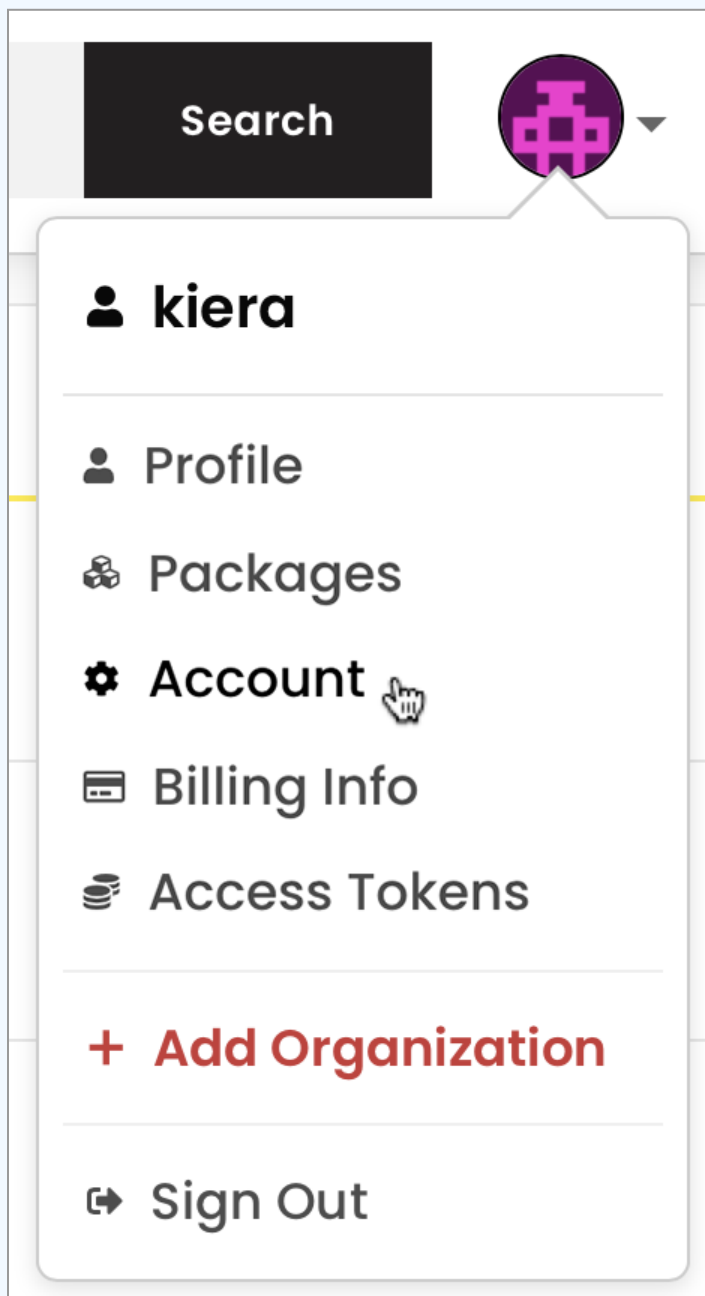
Password

[Forgot password?](#)

Login

[Create Account](#)

2. In the upper right corner of the page, click your profile picture, then click **Account**.



3. On the account settings page, under "Two-Factor Authentication", click **Modify 2FA**.

Two-Factor Authentication

✓ Enabled for authorization and publishing

– Authenticator app disabled

✓ 1 security key

Modify 2FA

4. Scroll to the bottom of the "Manage Two-Factor Authentication" page and click Disable 2FA.

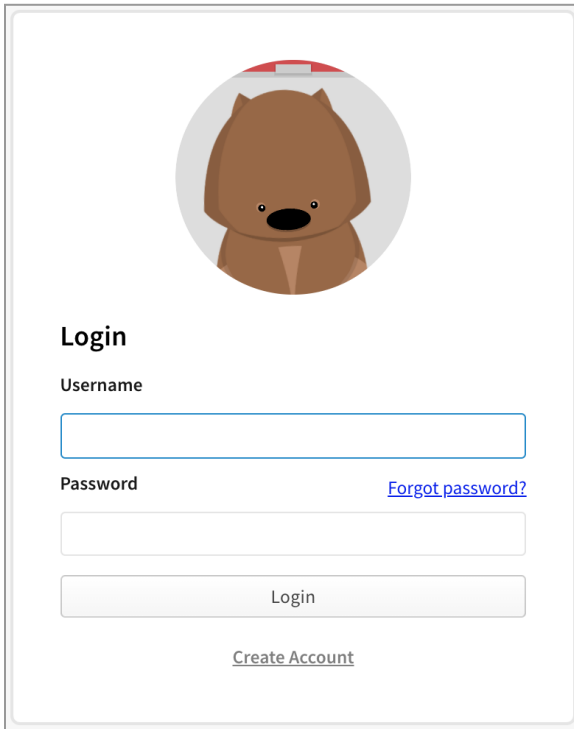
Disable Two-Factor Authentication

We strongly recommend using two-factor authentication to secure your account. If you need to disable 2FA, we recommend re-enabling it as soon as possible.

Disable 2FA

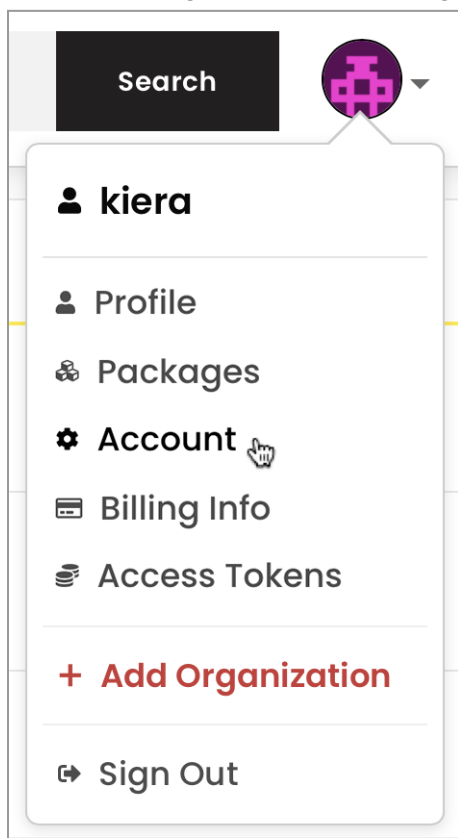
5. Agree to the prompt from the browser.

6. On the npm ["Sign In"](#) page, enter your account details and click **Sign In**.



The image shows the npm login page. At the top is a circular profile picture of a brown beaver. Below it is the heading "Login". There are two input fields: "Username" and "Password". To the right of the "Password" field is a link that says "Forgot password?". Below the input fields is a "Login" button. At the bottom is a link that says "Create Account".

7. In the upper right corner of the page, click your profile picture, then click **Account**.



8. On the account settings page, under "Two-Factor Authentication", click **Modify 2FA**.

Two-Factor Authentication

✓ Enabled for authorization and publishing

– Authenticator app disabled

✓ 1 security key

Modify 2FA

9. From the "Manage Two-Factor Authentication" navigate to "Additional Options" section
10. Clear the checkbox for "Require two-factor authentication for write actions" and click "Update Preferences"

Additional Options

☐ Require two-factor authentication for write actions

Update Preferences

Node is the javascript run time.

As a developer,use 3rd party liberty to make it easier to work.

Where does the javascript liberty or utility come from for 3rd party ?

This is **NPM**.

Npm is the registry ,the place where all the packages are listed down.You can upload your own package.

There is public and private liberty, some companies have their own private liberty.

NPM has the ability to upload and consume packages.

Providing us CLI,Click to download packages.

NPM website- npmjs.com

- **NPM vs Yarn:**

What is yarn?

Yarn, or Yet Another Resource Navigator, is a relatively new package manager developed by Facebook. It was developed to provide more advanced capabilities that NPM lacked at the time (such as version locking) while also making it safer, more reliable, and more efficient.

NPM has introduced several important features ever since Yarn was released.

Yarn is now more of an alternative to NPM than a replacement in its current version.

```
npm install yarn -g
```


	Yarn	NPM
1.	It uses the yarn add command to install dependencies.	It uses the npm install command to install dependencies.
2.	It installs dependencies in parallel.	It installs dependencies sequentially.
3.	The version lock file is known as yarn.lock.	The version lock file is known as <u>package-lock.json</u> .
4.	It supports the Plug'n'Play feature where it generates a .pnp.cjs file containing the map of dependencies for the project.	NPM doesn't support any such feature.

Yarn and NPM Commands

Let us see the different commands for NPM and Yarn in different scenarios:

Command	NPM	Yarn
Initialize project	npm init	yarn init
Run script	npm run	yarn run
Run tests	npm test	yarn test
Install dependencies	npm install	yarn

Install packages	npm install <package-name>	yarn add <package-name>
Uninstall packages	npm uninstall <package-name>	yarn remove <package-name>
Install packages globally	npm install -g <package-name>	yarn global add <package-name>
Uninstall packages globally	npm uninstall -g <package-name>	yarn global remove <package-name>
Update packages	npm update <package-name>	yarn upgrade <package-name>

Interactive dependency update	npm run upgrade-interactive	yarn upgrade-interactive
Check for outdated packages	npm outdated	yarn outdated
Manage local cache	npm cache clean	yarn cache clean
Login/Logout	npm login/logout	yarn login/logout
Publish package	npm publish	yarn publish

Update package manager	npm update	yarn upgrade
Run package remotely	Not Supported (but npx)	yarn dlx
Check licenses	Not Supported	yarn licenses ls

Yarn is security limitation ,speed limitation ,basically this is faster than npm.

Yarn website- <https://yarnpkg.com/>

- **Install nodejs and npm:**

If you install nodeJs ,npm install by default.

Website- <https://nodejs.org/en>

Version check of node - **\$node -v**

Version check of node - **\$npm -v**

Information of package-**\$ npm view <package name>**

\$npm view axios

```
windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Windows\system32> node -v
v19.6.1
PS C:\Windows\system32> npm -v
9.4.0
PS C:\Windows\system32> npm view axios

axios@1.3.4 | MIT | deps: 3 | versions: 77
Promise based HTTP client for the browser and node.js
https://axios-http.com

keywords: xhr, http, ajax, promise, node

dist
.tarball: https://registry.npmjs.org/axios/-/axios-1.3.4.tgz
.shasum: f5760cefd9c9fb51fd2481acf88c05f67c4523024
.integrity: sha512-toYm+Bsyl6VC5wSkfkbNB6ROv7KY93PEBBL6xyDczaIHasAiv4wPqQ/c4RjoQzipxRD2W5g21cOqQuLZ7rHwQ==
.unpackedSize: 1.7 MB

dependencies:
follow-redirects: ^1.15.0 form-data: ^4.0.0 proxy-from-env: ^1.1.0

maintainers:
- mzabriskie <mzabriskie@gmail.com>
- nickuraltsev <nick.uraltsev@gmail.com>
- emilyemorehouse <emilyemorehouse@gmail.com>
- jasonsaaayman <jasonsaaayman@gmail.com>

dist-tags:
latest: 1.3.4 next: 1.2.0-alpha.1

published a month ago by jasonsaaayman <jasonsaaayman@gmail.com>
PS C:\Windows\system32>
```

Tarball is a link where one file downloads information about a package.

What is next gen comes ,version is present.

\$npm - -help

- **Nodejs versioning:**

Suppose three different projects,all of them are js project.Suppose,1st and 2nd project version use node v1 and 3rd project is required version v2.That means project3 supports v1 but project1 and 2 does not support v2.

When two different projects are different versions,this time call it node versioning.

This is managing via node version manager(NVM)

Install link- <https://github.com/coreybutler/nvm-windows>

Cmd- \$nvm

Check current version- \$nvm current

Showing of node version- \$nvm list

To use in our system- \$nvm use <version>

To install node version- \$nvm install <version>

- Getting started with npm commands:

For help npm-> `$npm -help`

See the all commands-> `$npm -l`

To init the proj-> `$npm init`

```
CA: npm init
C:\npmteach>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 d

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: (npmteach)
version: (1.0.0) 0.0.1
description: This package is just for learning npm
entry point: (index.js)
test command:
git repository:
keywords:
author: Anideep
license: (ISC)
About to write to C:\npmteach\package.json:
{
  "name": "npmteach",
  "version": "0.0.1",
  "description": "This package is just for learning npm ",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Anideep",
  "license": "ISC"
}

Is this OK? (yes)
```

What is the package. Json do?

It manages metadata of the project.You can add cmd in the package. Json.


```

package.json X
package.json > {} scripts
1  {
2    "name": "npmteach",
3    "version": "0.0.1",
4    "description": "This package is just for learning npm ",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Anideep",
10   "license": "ISC"
11  }
12

```

Add cmd:

```

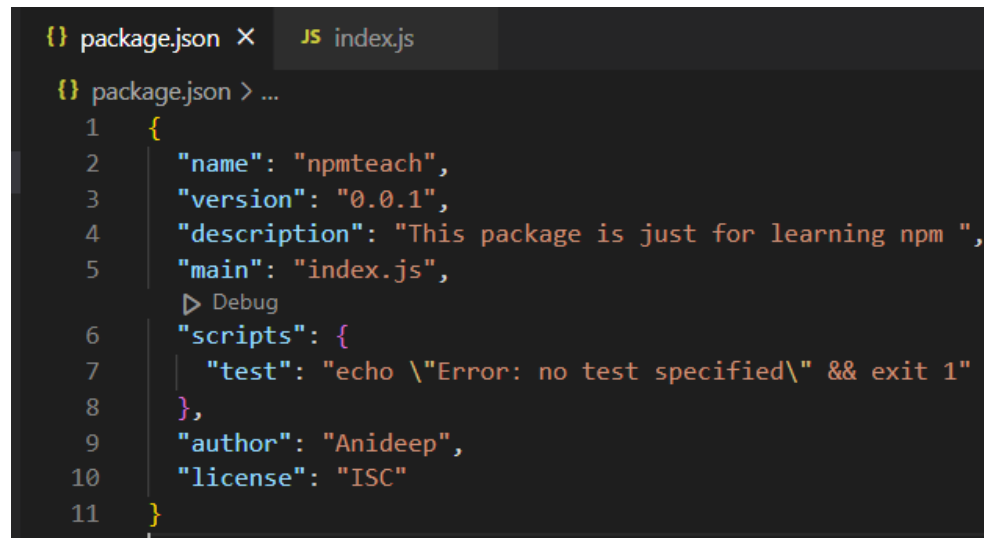
package.json X
package.json > {} scripts
1  {
2    "name": "npmteach",
3    "version": "0.0.1",
4    "description": "This package is just for learning npm ",
5    "main": "index.js",
6    "scripts": {
7      "start": "node index.js",
8      "preinstall": "",
9      "install": "",
10     "test": "echo \"Error: no test specified\" && exit 1"
11   },
12   "author": "Anideep",
13   "license": "ISC"
14  }
15

```

Whenever install dependencies ,where will be listed in the package. Json.Key will be added: dependencies as key value pair: , it is not manually via cmd line.

- **Exploring npm install & package.json file:**

1st create an entry file which is in the package. Json. suppose index.js file in package. Json then creates a file in vs code index.js.

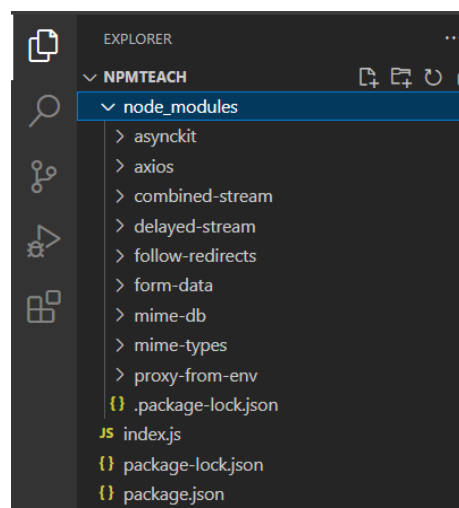


The screenshot shows the VS Code editor interface. At the top, there are two tabs: 'package.json' and 'JS index.js'. The 'package.json' tab is active, displaying the following JSON content:

```
{
  "name": "npmteach",
  "version": "0.0.1",
  "description": "This package is just for learning npm ",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Anideep",
  "license": "ISC"
}
```

For an example->In terminal: **\$npm i axios**

After install any dependencies:



Install package with node module.

Download tarball.(Copy and paste tarball)

For unzip the folder->`$tar -xvzf "<folder path of zip>"`

`$tar -xvzf "C:\Users\esolz\Downloads\axios-1.3.4 (1).tgz"`

After unzipping the folder you can see the package folder in the folder.

When downloading any package ,npm i <package> , install node modules it is to install various folders..so axios also depended on other dependencies with some utility. So this is called peer dependencies or dependencies.

You can see the package manually via tarball and also in npm i.

Note: DO NOT CHANGE ANY FILE/FOLDER IN NODE MODULES.

- Dependencies & devDependencies in package.json:

Two ways to add it ,two types of dependencies ,one is prod level dependency means that used in production and other dependencies non prod dependency is

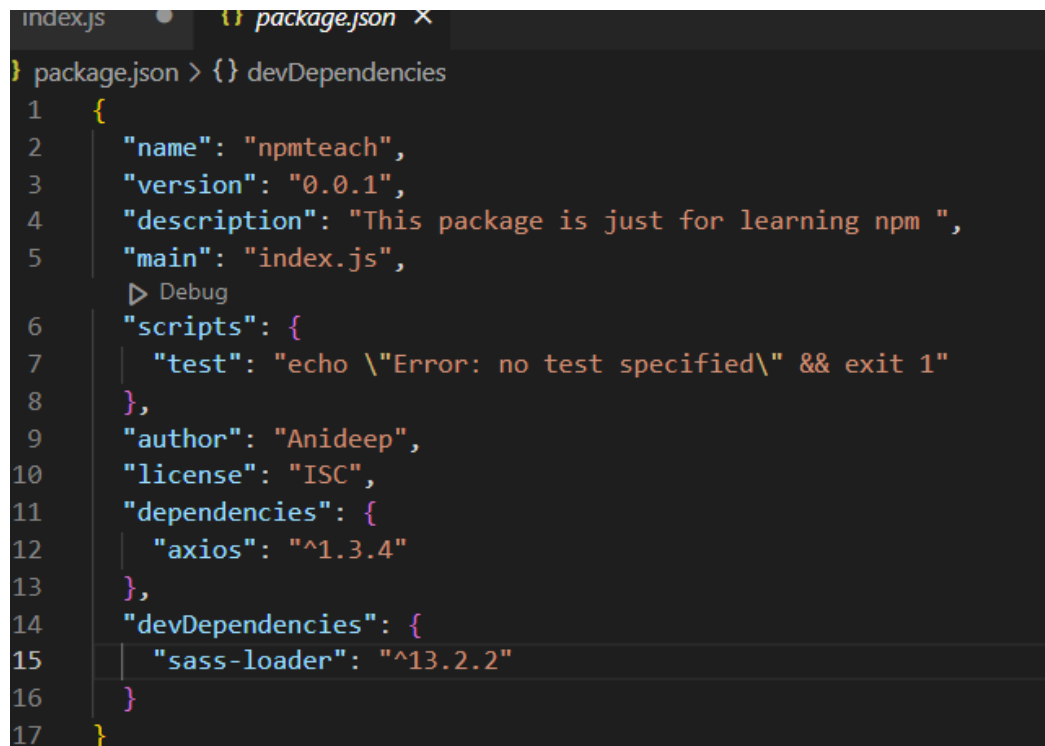
also known as dev dependency that is required at time in local development ,such as testing files.

When installing npm i , with many folders downloaded with node modules.

Non-prod dependency is dev dependency is only req. Local development,such as testing files that means test your code in local .Just locally or on CI Pipeline,it is not bundled with the production code.

Separated in package.json file in dev and prod level dependency.

Install non prod level dependency or dev ->`$ npm i <package> - - save-dev`



```
index.js  package.json x
} package.json > {} devDependencies
1  {
2    "name": "npmteach",
3    "version": "0.0.1",
4    "description": "This package is just for learning npm ",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Anideep",
10   "license": "ISC",
11   "dependencies": {
12     "axios": "^1.3.4"
13   },
14   "devDependencies": {
15     "sass-loader": "^13.2.2"
16   }
17 }
```

This dependency is local level.

Prod level dependency install-> `$npm i <packagename>`

```
{ } package.json > { } devDependencies
1  {
2    "name": "npmteach",
3    "version": "0.0.1",
4    "description": "This package is just for learning npm ",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Anideep",
10   "license": "ISC",
11   "dependencies": {
12     "axios": "^1.3.4"
13   },
14   "devDependencies": {
15     "sass-loader": "^13.2.2"
16   }
17 }
```

- Uninstalling dependencies:

Uninstall the the package->

`$npm uninstall <package name>`

It is uninstall from node modules and as well as package.json also.\

So you want to remove it from your local package without removing from package. Json .

`$npm uninstall - -no-save <package name>`

If you want install globally(around your machine) in the project->

```
$npm i -g <package name>
```

If you want uninstall globally(around your machine) in the project->

```
$npm uninstall -g <package name>
```

- Semantic Versioning:

To keep the JavaScript ecosystem healthy, reliable, and secure, every time you make significant updates to an npm package you own, we recommend publishing a new version of the package with an updated version number in the package.json file that follows the semantic versioning spec.

Updating any package with semantic behavior.


When you go to in package.json, There is a version. When you upgrading this package version

```
{ } package.json > { } devDependencies
1  {
2    "name": "npmteach",
3    "version": "0.0.1",
4    "description": "This package is just for learning npm ",
5    "main": "index.js",
   > Debug
6    "scripts": {
```

Change version(0.0.1).

Go to npm and take any package. Then you can see any package version


homepage

 github.com/webpack-contrib/sass-loader

♥Fund this package

⬆ Weekly Downloads

10,796,678



Version

13.2.2

License

MIT

Unpacked Size

59.3 kB

Total Files

9

Version->13.2.2

13->Major change.

2->Minor change.

2->Patch

Code status	Stage	Rule	Example version

First release	New product	Start with 1.0.0	1.0.0
Backward compatible bug fixes	Patch release	Increment the third digit	1.0.1
Backward compatible new features	Minor release	Increment the middle digit and reset last digit to zero	1.1.0
Changes that break backward compatibility	Major release	Increment the first digit and reset middle and last digits to zero	2.0.0

Link->

<https://docs.npmjs.com/about-semantic-versioning>

- Understanding package-lock.json:

NPM5 brought the concept of package-lock.json For maintaining semantic versioning. Still they can maintain semantic versioning because developers need to follow the semantic versioning, if they are not following, there is no use of this package-lock.json file and package.json because it can break your application when auto-upgrading the dependency.

When versions match with package.json and package-lock.json, then it is perfect.

Suppose I want any version->

Go to any package-> Go to versions-> Check the version

Now, go to package.json version change version which is available, then remove node modules and package.json from project, go to terminal `$ npm i <package name>`

NPM internally checks that what the current version is and adds it into the node modules folder.

Package.json tells different things, go to node modules and package name and go to package.json in node modules, see that Package.json in the main app and package.json in node modules versions are different.

This follows semantic versioning.

Package.lock.json and node modules package.json versions are the same. Whatever the change in package.json in the main folder.

If NPM follow the semantic versioning follow it but developer do es not follow it ,npm and semantic versioning are no breaking change but developer breaking change of version ,lod version package.json ,it is brings the latest version and add to node modules folder and it breaks your project .

That time you have to check package.json of node modules.

- Understanding '^' , '~' symbols:

How does the NPM decide to auto upgrade the package of the latest version?

There is 3 method of package of package.json->

1. **^10.2.3** ->> upgrade patch version and minor version.-->**4.X.X** ->> 4.2.0 TO 4.4.3
2. **~3.0.2** → upgrade patch version version —>**4.2.x** -> 4.3.2 to 4.3.6
3. 4.2.0 → no change -> 4.2.3

If you update the version ,the npm check the semantic version follows but it breaks the code.

^10.2.3 ->>

```
1  {
2    "name": "npmteach",
3    "version": "0.0.1",
4    "description": "This package is just for learning npm ",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Anideep",
10   "license": "ISC",
11   "dependencies": {
12     "axios": "^1.3.4",
13     "moment": "^2.29.3"
14   },
15   "devDependencies": {
16     "sass-loader": "^13.2.2"
17   }
```

~3.0.2 → upgrade patch version → **4.2.x** -> 4.3.2 to 4.3.6

```
{} package.json • {} package-lock.json
{} package.json > {} dependencies > moment
1  {
2    "name": "npmteach",
3    "version": "0.0.1",
4    "description": "This package is just for learning npm ",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Anideep",
10   "license": "ISC",
11   "dependencies": {
12     "axios": "^1.3.4",
13     "moment": "~2.29.3"
14   },
15   "devDependencies": {
16     "sass-loader": "^13.2.2"
```

- Understanding peer dependencies:

Link-<https://github.com/facebook/react/blob/main/packages/react-dom/package.json>

Dependency is packages ,modules .Peer dependencies are simply require when you are developing a liberty ,suppose developing two things app ,liberty both have package.json.When you are developing app ,do not need peer dependency.When you are developing liberty that is used by other app developer or other liberty developers that might need peer dependency.

Suppose,In package.json in main folder ,one package and install another package like react dom ,go to react dom package and go particular package package.json

```
21     "scheduler": "^0.23.0"
22   },
23   "peerDependencies": {
24     "react": "^18.2.0"
25   },
26   "files": [
```

If you mention peer dependency ,I want my liberty users ,whoever my liberty are ,to use the host end.

UPDATE: npm versions 1, 2, and 7 will automatically install peerDependencies if they are not explicitly depended upon higher in the dependency tree. For npm

versions 3 through 6, you will receive a warning that the peerDependency is not installed instead.

Link- <https://nodejs.org/en/blog/npm/peer-dependencies>

```
PS C:\npmteach> npm -v
9.4.0
PS C:\npmteach> nvm -v
1.1.10
PS C:\npmteach> nvm install 14.16.1
Downloading node.js version 14.16.1 (64-bit)...
Complete
Creating C:\Users\esolz\AppData\Roaming\nvm\temp
Downloading npm version 6.14.12... Complete
Installing npm v6.14.12...
```

- Understanding NPM Scripts :

Script is like a command line, Which is installed with npm.

The "scripts" property of your package.json file supports a number of built-in scripts and their preset life cycle events as well as arbitrary scripts.

```
6  "scripts": {
7    "test": "echo \"Error: no test specified\" && exit 1"
8  },
9  "author": "Anideep",
```

Script is an object with a key value pair.

```

6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "node index.js"
9   },
10  "author": "Anideep"

```

```

{} package.json  JS index.js  X
JS index.js
1  console.log("This this run by npm scripts");

```

This is run-> `$ npm <script name>`

`$npm start`

```

{} package.json X  JS index.js
{} package.json > {} scripts > start
1  {
2    "name": "npmteach",
3    "version": "0.0.1",
4    "description": "This package is just for learning npm ",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "start": "node index.js"
9    },

```

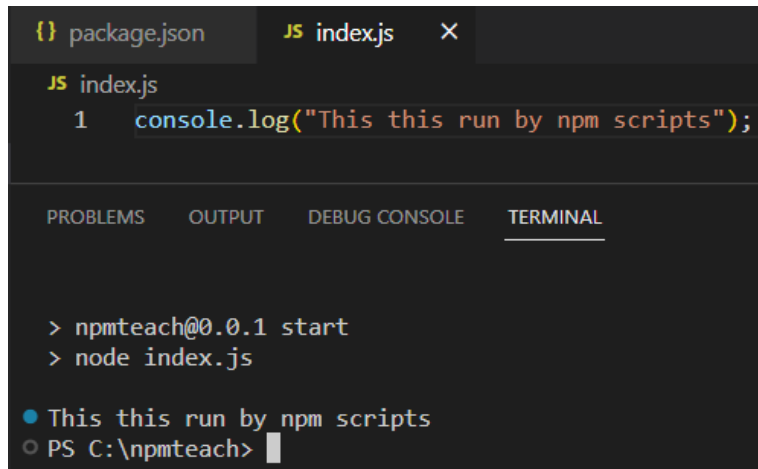
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

> npmteach@0.0.1 start
> node index.js

● This this run by npm scripts
○ PS C:\npmteach>

```



The screenshot shows a VS Code editor window with two tabs: 'package.json' and 'index.js'. The 'index.js' tab is active, displaying a single line of JavaScript code: `console.log("This this run by npm scripts");`. Below the editor, the 'TERMINAL' panel is open, showing the command prompt output. The prompt is `npmteach@0.0.1 start`, followed by the command `node index.js`. The output of the command is `This this run by npm scripts`, displayed in green text. The terminal prompt is `PS C:\npmteach>`.

“Start”: “eco ‘I am okay with npm scripts ’”

This eco is used for terminal/powershell.

When you add a custom script, it is not run with `$ npm <script name>` but it is run by `$ npm run <script name>`.

`$ npm <script name>`

```
{ } package.json X JS index.js
{ } package.json > { } scripts > myscript
2   "name": "npmteach",
3   "version": "0.0.1",
4   "description": "This package is just for learning npm ",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "echo 'index.js'",
9     "myscript": "node index.js"
10  }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
> npmteach@0.0.1 start
> node index.js
This this run by npm scripts
PS C:\npmteach> npm myscript
Unknown command: "myscript"
Did you mean this?
  npm run myscript # run the "myscript" package script
To see a list of supported npm commands, run:
  npm help
PS C:\npmteach>
```

\$ npm run <script name>

```
{ } package.json X JS index.js
{ } package.json > { } scripts > myscript
2   "name": "npmteach",
3   "version": "0.0.1",
4   "description": "This package is just for learning npm ",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "echo 'index.js'",
9     "myscript": "node index.js"
10  }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\npmteach> npm run myscript
> npmteach@0.0.1 myscript
> node index.js
This this run by npm scripts
PS C:\npmteach>
```

Script DOC- <https://docs.npmjs.com/cli/v6/using-npm/scripts>

- Understanding npx:

NPX is a NPM package runner. NPM is a node package runner, While NPM gives an executable package runner. NPX(Node Package eXecute) runs a package without installing its system.

What is the main goal of NPX?

Sometimes some packages and cmd are used often. For example create react app, it is not used everytime use NPX ,it just installs necessary packages and no other unnecessary packages installed with it. NPX brings the package from NPM and executes it and deletes it.

- Understanding npm cache and purging:

When called npm ,executable binary is big and not required that binary store into cache. NPM has its own chasing mechanism. It is run locally. Program runs faster.

```
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\esolz\AppData\Roaming\npm-cache>dir
Volume in drive C has no label.
Volume Serial Number is 1C48-EE54

Directory of C:\Users\esolz\AppData\Roaming\npm-cache

29-11-2022  12:00    <DIR>          .
29-11-2022  12:00    <DIR>          ..
29-11-2022  12:00             173 anonymous-cli-metrics.json
09-11-2022  14:25    <DIR>          node-sass
14-11-2022  10:10    <DIR>          _cacache
29-11-2022  12:00    <DIR>          _locks
28-11-2022  15:11    <DIR>          _logs
29-11-2022  09:43    <DIR>          _npx
                1 File(s)             173 bytes
                7 Dir(s)  82,399,141,888 bytes free

C:\Users\esolz\AppData\Roaming\npm-cache>
```

```
C:\Windows\System32\cmd.exe

C:\Users\esolz\AppData\Roaming\npm-cache\_cacache>dir
Volume in drive C has no label.
Volume Serial Number is 1C48-EE54

Directory of C:\Users\esolz\AppData\Roaming\npm-cache\_cacache

14-11-2022  10:10    <DIR>          .
14-11-2022  10:10    <DIR>          ..
14-11-2022  11:54    <DIR>          content-v2
14-11-2022  11:56    <DIR>          index-v5
29-11-2022  12:00    <DIR>          tmp
               0 File(s)                0 bytes
               5 Dir(s)  82,397,495,296 bytes free

C:\Users\esolz\AppData\Roaming\npm-cache\_cacache>
```

For cache folder location in pc- >> C:\Users\esolz\AppData\Roaming

This log files created when NPM failed->>

```
C:\Users\esolz\AppData\Roaming\npm-cache\_logs>dir
Volume in drive C has no label.
Volume Serial Number is 1C48-EE54

Directory of C:\Users\esolz\AppData\Roaming\npm-cache\_logs

28-11-2022  15:11    <DIR>          .
28-11-2022  15:11    <DIR>          ..
22-11-2022  11:09             986 2022-11-22T05_39_33_833Z-debug.log
22-11-2022  11:10             986 2022-11-22T05_40_42_714Z-debug.log
22-11-2022  11:11             986 2022-11-22T05_41_36_680Z-debug.log
22-11-2022  16:56            2,131 2022-11-22T11_26_58_898Z-debug.log
28-11-2022  10:07            3,203 2022-11-28T04_37_37_754Z-debug.log
28-11-2022  10:09            1,855 2022-11-28T04_39_13_945Z-debug.log
28-11-2022  10:37            1,803 2022-11-28T05_07_27_926Z-debug.log
28-11-2022  14:48             946 2022-11-28T09_18_21_758Z-debug.log
28-11-2022  15:10             946 2022-11-28T09_40_03_755Z-debug.log
               9 File(s)             13,842 bytes
               2 Dir(s)  82,398,838,784 bytes free

C:\Users\esolz\AppData\Roaming\npm-cache\_logs>
```

It does not call again and again, it makes less api calls again and again.

Sometimes cache is interrupted. You want to cache clear manually and add cache because there is no api call again and again.

Cache corruption will either trigger an error, or signal to `pacote` that the data must be re fetched, which **it will do automatically**.

When an error occurs with cache->>>>

\$npm cache clean - -force

NPM Cache DOC->> <https://docs.npmjs.com/cli/v9/commands/npm-cache>

- **npm Prune;**


Now Sometimes ,I might have packages that are not really required.If you see in package.json ,see one dependency but other dependencies are present in node modules.And see with commands->>>

\$npm list - -depth 0

```
C:\npmteach>npm list --depth 0
npmteach@0.0.1 C:\npmteach
+-- @jridgewell/gen-mapping@0.3.2 extraneous
+-- @jridgewell/resolve-uri@3.1.0 extraneous
+-- @jridgewell/set-array@1.1.2 extraneous
+-- @jridgewell/source-map@0.3.2 extraneous
+-- @jridgewell/sourcemap-codec@1.4.14 extraneous
+-- @jridgewell/trace-mapping@0.3.17 extraneous
```

There are extraneous packages that have packages in the project folder but not in the package.json. In order to remove packages, have command:

\$npm prune

 C:\Windows\System32\cmd.exe

```
C:\npmteach>npm prune

removed 77 packages, and audited 10 packages in 3s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\npmteach>npm list --depth 0
npmteach@0.0.1 C:\npmteach
|-- axios@1.3.4
```