npm Description npm is the package manager for the Node JavaScript platform. It puts modules in place so that node can find them, and manages dependency conflicts intelligently.

It is extremely configurable to support a variety of use cases. Most commonly, you use it to publish, discover, install, and develop node programs.

npm packages can be imported into your website directly or can be installed on a project as well which can be bundled if required.

Fun fact: npm is right now acquired by Github, which was acquired by Microsoft earlier. VSCode, Github, npm is now owned by Microsoft Installation if you already have node installed, you should have npm in your system. check by using the following code in your cli, npm -v if you need to know how to setup with node and nvm check the following link To download the latest version of npm, on the command line, run the following command: npm install -g npm Directories In particular, npm has two modes of operation: local mode: npm installs packages into the current project directory, which defaults to the current working directory. Packages install to ./node_modules, and bins to ./node_mod- ules/.bin.

global mode: npm installs packages into the install prefix at $npm_config_pre- fix/lib/node_modules and bins to $npm_config_prefix/bin.

Local mode is the default. Use -g or --global on any command to run in global mode instead. Start a new project to create a new project first create a new folder for the project and enter it and run the following npm init you can read the all questions being asked, for now, say yes to everything or you can use npm init -y You will now see a package.json file

package.json This document is all you need to know about what's required in your project. It must be actual JSON, not just a JavaScript object literal.

If you plan to publish your package, the most important things in your package.json are the name and version fields as they will be required. The name and version together form an identifier that is assumed to be completely unique. Changes to the package should come along with changes to the version. If you don't plan to publish your package, the name and version fields are optional.

The name must be less than or equal to 214 characters. This includes the scope for scoped packages.

The names of scoped packages can begin with a dot or an underscore. This is not permitted without a scope.

New packages must not have uppercase letters in the name.

The name ends up being part of a URL, an argument on the command line, and a folder name. Therefore, the name can't contain any non-URL-safe characters. Tips:

Don't use the same name as a core Node module.

Don't put "js" or "node" in the name. It's assumed that it's js, since you're writing a package.json file, and you can specify the engine using the "engines" field. (See below.)

The name will probably be passed as an argument to require(), so it should be something short, but also reasonably descriptive.

You may want to check the npm registry to see if there's something by that name already, before you get too attached to it. https://www.npmjs.com/

Docs

installing packages Docs

if you use npm install then you are asking npm to install all the packages mentioned in package.json into your local machine Install the dependencies in the local node_modules folder.

In global mode (ie, with -g or --global appended to the command), it installs the current package context (ie, the current working directory) as a global package.

By default, npm install will install all modules listed as dependencies in package.json.

This command installs a package and any packages that it depends on. If the package has a package-lock, or an npm shrinkwrap file, or a yarn lock file, the installation of dependencies will be driven by that.

A package is: a) a folder containing a program described by a package.json file b) a gzipped tarball containing (a) c) a url that resolves to (b) d) a @ that is published on the registry (see registry) with (c) e) a @ (see npm dist-tag) that points to (d) f) a that has a "latest" tag satisfying (e) g) a that resolves to (a)

npm install saves any specified packages into dependencies by default. Additionally, you can control where and how they get saved with some additional flags:

-D, --save-dev: Package will appear in your devDependencies. for example npm install node-tap --save-dev or npm install node-tap -D install your first package lets install a package called qrcode which will help you create qrcode npm install qrcode -g

Examples: qrcode "some text" Draw in terminal window qrcode -o out.png "some text" Save as png image qrcode -d F00 -o out.png "some text" Use red as foreground color Ensure you are in the root folder where package.json is already present for your project to install any new packages npm install package_name here package_name will be the package that you want to install once you install a package, you will see that there will be a node_modules folder This folder contains all the code necessary to run the project, (also known as dependencies) This folder will be heavy, and we dont want to copy the dependency code as well, rather this information is stored in the package.json, so if anyone else has to run the project, they can just run npm install in the root project and it will install the required files for you on the machine add a .gitignore file with the following content

dependencies

/node_modules if you dont know what .gitignore does, dont worry about it now, we will cover git soon, and by then it will make sense but essentially it will ensure that the project does not store the node_modules folder when you use your version control system cli let use another library called commander (there are alternatives like yargs etc.) this library will allow us to make your program easily working with cli run the following code from the project folder npm install commander change the index.js file to var { Command } = require('commander'); var program = new Command(); program.version('0.0.1'); program.argument("","user login details") .argument("

[password]","password for user, if needed", "default") .action(function(username, password){ console.log("username",username) console.log("password",password) }) now when you run the program with node index.js masai password it will print username masai password password you can use the following commands as well node index.js --help node index.js masai Here is another example of taking printing a name for n number of times program.argument("","name to print") .argument("[number]","number of times to print",1) .action(function(name,number){ number = Number(number) for(var i=0; i<number; i++){ console.log(name) } }) glossary Just keeping a list of commands and topics for reference

npm npm init npm init -y npm install npm install package_name npm install package_name -g package.json node_modules qrcode