NPM

- is a package manager for Node.js packages, or modules if you like.
- www.npmjs.com hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js
- npm consists of three distinct components:
  - The website.
    - This is the main website of node packages which will allow you to download and install various of packages.
  - The Command Line Interface (CLI).
    - Runs from a terminal, we use cli to interact with npm.
  - The registry.
    - Is a large public database of JavaScript software and the meta-information surrounding it.
  - Use npm to:
    - Download standalone tools you can use right away.
    - Run packages without downloading using npx.
    - Restrict code to specific developers.
    - Manage multiple versions of code and code dependencies.
    - Update applications easily when underlying code is updated.
    - Discover multiple ways to solve the same puzzle.

About npm

- npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well.

What is a Package

- A package in Node.js contains all the files you need for a module. Modules are JavaScript libraries you can include in your project.

- Download a package
  - Open the command line interface and tell NPM to download the package you want. I want to download a package called "upper-case":
    Download "upper-case":

  - `C:\Users\Your Name>npm install upper-case`
  - Now you have downloaded and installed your first package!

- NPM creates a folder named "node_modules", where the package will be placed. All packages you install in the future will be placed in this folder.My project now has a folder structure like this:
  - C:\Users\My Name\node_modules\upper-case

node_modules

- This directory contains dependencies and sub-dependencies of packages used by the current React app, as specified by package.json. If you take a look, you may be surprised by how many there are.

package.json

- Specifics of npm's package.json handling.as
- This document is all you need to know about what's required in your package.json file. It must be actual JSON, not just a JavaScript object literal.
- A lot of the behavior described in this document is affected by the config settings described in config.
- In Node.js, package.json is a versioning file used to install multiple packages in your project. As you initialize your node application, you will see three files installed in your app that is node_modules, package.json, and package.lock.json.
- a package.json file contains metadata about the project and also the functional dependencies that is required by the application.

Package-lock.json

- This file contains the exact dependency tree installed in node_modules/. This provides a way for teams working on private apps to ensure that they have the same version of dependencies and sub-dependencies. It also contains a history of changes to package.json, so you can quickly look back at dependency changes.
- package.lock.json is created for locking the dependency with the installed version.
- package.lock.json is created for locking the dependency with the installed version.

- Let's say if the current version of the package is 1.3.2 then it will save the version with (^) sign. Here carot(^) means, it will support any higher version with major version 1 for eg. 1.2.2.
- Without package.lock.json, there might be some differences in installed versions in different environments. To overcome this problem, package.lock.json is created to have the same results in every environment. It should be in source control with the package.json file because if any other user will clone the project and install dependencies then it will install the exact same dependencies as in package.lock.json to avoid differences.
-

package.json	package.lock.json
It contains basic information about the project.	It describes the exact tree that was generated to allow subsequent installs to have the identical tree.
It is mandatory for every project.	It is automatically generated for those operations where npm modifies either node_modules tree or package.json.
It records important metadata about the project.	It allows future devs to install the same dependencies in the project.
It contains information such as name, description, author, script, and dependencies.	It contains the name, dependencies, and locked version of the project.