# MISC

Node.js or Node is an open-source, cross-platform, **JavaScript runtime environment(JSRE)** that executes JavaScript code outside of a web browser.

npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js.

Nvm is a Nodejs version manager. It lets you easily install and switch between versions. It retains globally installed packages for each version.

Note when install Node.js from <https://nodejs.org/en/> it installs 2 things: Node.js and NPM

# Quick introduction to NPM

## What is npm?

npm is 2 things:

* Npm is a software repository for JavaScript
* npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js.

## install npm

people often install both nodejs and npm from the website of Nodejs

## How to use npm

npm manages downloads of dependencies of your project.

### package.json

All npm packages contain a file, usually in the project root, called package.json - this file holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies. It can also contain other metadata such as a project description, the version of the project in a particular distribution, license information, even configuration data - all of which can be vital to both npm and to the end users of the package.

### Installing all dependencies

If a project has a packages.json file, by running

npm install

it will install everything the project needs, in the node\_modules folder, creating it if it’s not existing already.

### Installing a single package

You can also install a specific package by running

npm install <package-name>

Often you’ll see more flags added to this command:

--save installs and adds the entry to the package.json file dependencies (default as of npm 5)

--save-dev installs and adds the entry to the package.json file devDependencies

The difference is mainly that devDependencies are usually development tools, like a testing library, while dependencies are bundled with the app in production.

### Versioning

In addition to plain downloads, npm also manages versioning, so you can specify any specific version of a package, or require a version higher or lower than what you need.

Many times you’ll find that a library is only compatible with a major release of another library.

Or a bug in the latest release of a lib, still unfixed, is causing an issue.

Specifying an explicit version of a library also helps to keep everyone on the same exact version of a package, so that the whole team runs the same version until the package.json file is updated.

In all those cases, versioning helps a lot, and npm follows the semantic versioning (semver) standard.

### Running Tasks

The package.json file supports a format for specifying command line tasks that can be run by using

npm run <task-name>

For example:

{

"scripts": {

"start-dev": "node lib/server-development",

"start": "node lib/server-production"

},

}

It’s very common to use this feature to run Webpack:

{

"scripts": {

"watch": "webpack --watch --progress --colors --config webpack.conf.js",

"dev": "webpack --progress --colors --config webpack.conf.js",

"prod": "NODE\_ENV=production webpack -p --config webpack.conf.js",

},

}

So instead of typing those long commands, which are easy to forget or mistype, you can run:

$ npm run watch

$ npm run dev

$ npm run prod

# Setting up the local environment and workspace