**03 Node Package Manager**

**1) NPM (Node Package Manager)**:

NPM (short for Node.js package manager) is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called NPM, and an online database of public and paid-for private packages, called the NPM registry. The registry is accessed via the client, and the available packages can be browsed and searched via the NPM website. The package manager and the registry are managed by NPM, Inc.

NPM is used to manage dependencies for packages. If we were to unpack a framework and use it outside NPM, we would have to do this every time we want to update the framework. NPM does this for us. We always know what version we are on, and we can limit a dependency to a specific major/minor/patch version.

**In this section we learn**:

1. Every Node application has a package.json file that includes metadata about the application. This includes the name of the application, its version, dependencies, etc.
2. We use NPM to download and install 3rd-party packages from NPM registry:
3. All the installed packages and their dependencies are stored under node modules folders. This folder should be excluded from the source control.
4. Node packages follow semantic versioning: major.minor.patch

**Useful NPM commands are**:

1. Install a package

npm i <packageName> (Specific for application)

npm i -g <packageName> (Install globally)

sudo npm i -g <packageName> (For Linux)

1. Check NPM version

npm -v

1. Install a specific version of a package

npm i <packageName>@<version> (Specific for application)

npm i -g <packageName>@<version> (Install globally)

sudo npm i -g <packageName>@<version> (For Linux)

1. Install a package as a development dependency

npm i <packageName> —save-dev

1. Uninstall a package

npm un <packageName>

1. List installed packages

npm list —depth=0

1. View outdated packages

npm outdated

1. View outdated packages

npm update

**2) 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. The package.json file is normally located at the root directory of a Node.js project.

**Create package.json**:

Go to the project directory and open CMD and run the following command

1. npm init
2. Press ^C at any time to quit.

package name: (practice) (place name or press enter)

1. version: (1.0.0) (Press Enter)
2. description: (Press Enter)
3. entry point: (index.js) (Press Enter)
4. test command: (Press Enter)
5. git repository: (Press Enter)
6. keywords: (Press Enter)
7. author: (Press Enter)
8. license: (ISC) (Press Enter)
9. Is this ok? (yes) (Press Enter)

Now "package.json" file is created in our directory.

**package.json**:

{

"name": "practice",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

There is a faster way to create "package.json" file. Go to cmd and run "npm init --yes"

**3) Installing a Node Package**:

Now we ad

**4) Using a Package**:

**5) Package Dependencies**:

**6) NPM Packages and Source Control**:

**7) Semantic Versioning**:

**8) Listing the Installed Packages**:

**9) Viewing Registry Info for a Package**:

**10) Installing a Specific Version of a Package**:

**11) Updating Local Packages**:

**12) Dev-Dependencies**:

**13) Uninstalling a Package:**

**14) Working with Global Packages**:

**15) Publishing a Package**:

**16) Updating a Published Package**:

Node Package Manager