NPM

What is NPM?

- Imagine, you are working on Node.js application and need to use a third party library, for example –

lodash (https://lodash.com)
JavaScript library providing various utility functions for arrays, objects, numbers, strings etc.

moment (https://momentjs.com)
JavaScript library to parse, validate, manipulate, and display date and time

- One way is to download the source file(s) of the third party library and include that in your application (using require).

What is NPM?

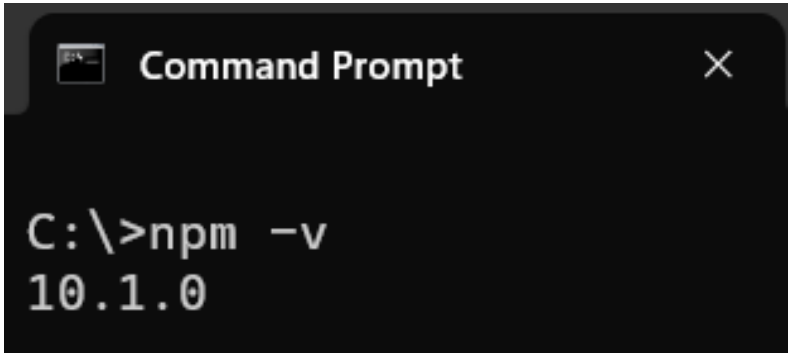
- What if you have too many such dependencies?
  → You will have to download each and every third-party library or dependency.

- What if you want to handover/distribute the Node.js application?
  → You will have to bundle the complete application code along with all the dependencies that you have downloaded.

- How do you ensure the correct versions of these dependencies are used?

What is NPM?

- JavaScript / Node.js ecosystem provides Package Manager
- It is a default package manager for Node.js.
- NPM is included with Node.js installation.
- NPM installation can be verified by below command in terminal or command prompt.



What is NPM?

- https://www.npmjs.com/about
- It was created as an open source project to help JavaScript developers to easily distribute/share packaged modules of code.
- Largest software registry in the world with more than 2 million packages.
- It is managed by npm, Inc. a company founded in 2014, and was acquired by GitHub in 2020.

What is NPM?

- NPM handles dependencies and package management for your project.
- You define all your project's dependencies inside your package.json file.
- Anytime you or a team member needs to get started with your project, all they have to do is run npm install. This will immediately install all the necessary dependencies for your project.
- In the package.json file, you can also specify which versions your project depends upon. This is useful to prevent updates from these packages from breaking your project.

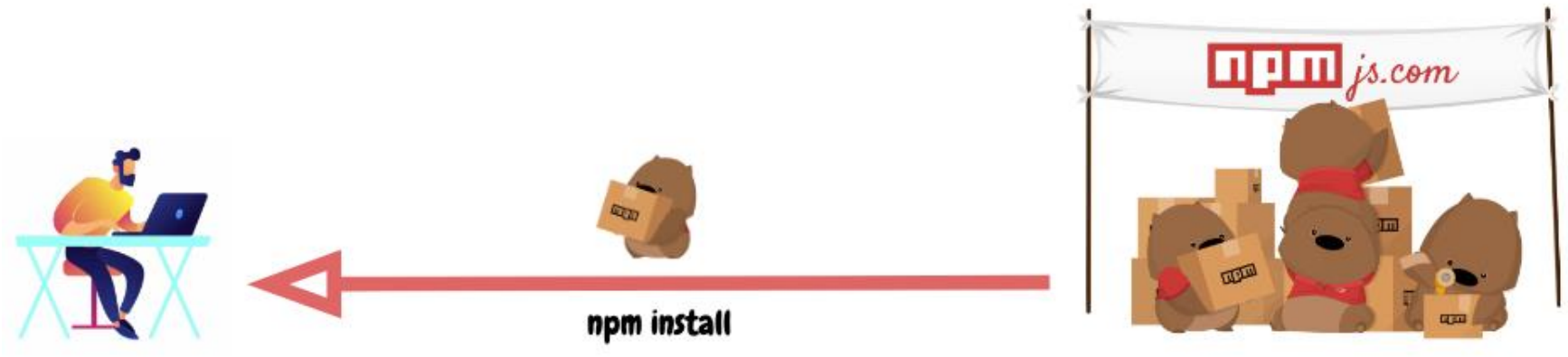
What is NPM?

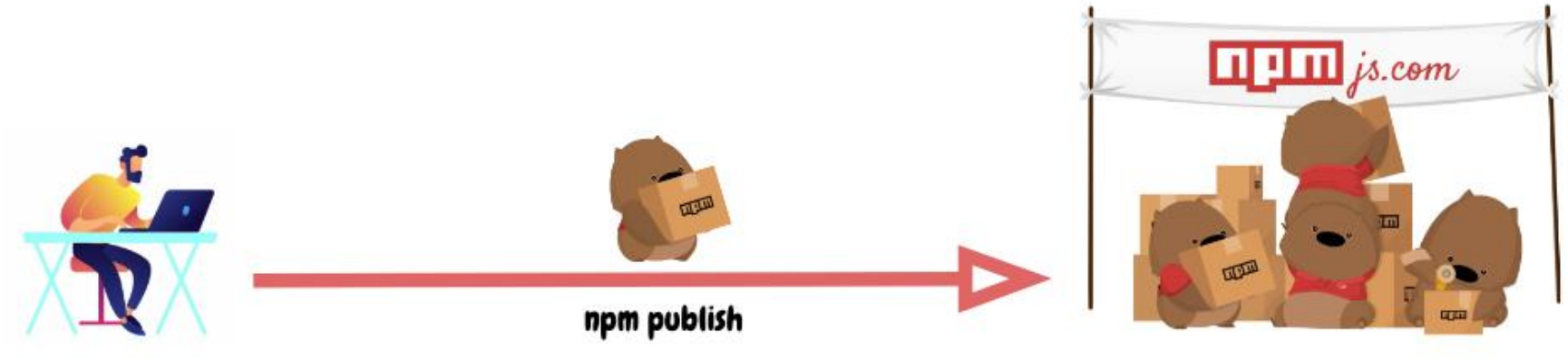
- To summarize, NPM is -
  - ✓ A package manager for Node.js packages (or modules).
  - ✓ An online repository (aka registry / library) of open-source Node.js packages.
  - ✓ A CLI (command line interface) that helps install, update, uninstall and publish the packages as well as manage package versions and dependencies.

Other package managers:

- Yarn (Yet Another Resource Negotiator)
- PNPM (Performant Node Package Manager)

What is NPM?


npm js.com
npm install
Consumer


npm js.com
npm publish
Author

Create new project using npm

- To create a new project using npm, execute the command

```
npm init
```

- Run the CLI questionnaire *(default value is shown in brackets)*

```
package name: <Add package name>
version: <Add version>
description: <Add description>
entry point: <Add entry point to Node application, typically index.js>
test command: <Enter test script or leave it blank>
git repository: <leave it blank, if not using git repo>
keywords: <leave it blank or add keywords for package search>
author: <Add your name>
license: <Keep default>
```

- The package.json file gets created. This is a configuration file which contains metadata of your Node.js project.

Create new project using npm

- Create JavaScript source file as an entry point to your application.
- Modify package.json and add this as a first line inside scripts section: `"start": "node index.js",`
- Now the command `npm start` will run index.js file.
- Let's use a third party library as a dependency and install using npm.
- Install a dependency using command `npm install moment`
- The *node_modules* folder gets created, dependency gets added in the package.json file.
- Include this sample code in index.js:

```
let moment = require('moment');
let dateTime = moment().format('MMMM Do YYYY, h:mm:ss a');
console.log(`Current date/time: ${dateTime}`);
```

Reference: https://momentjs.com

Create new project using npm

- When you bundle your node application, you just need to include the source code and package.json file. No need to include node_modules folder.

- **Npm install** commands:
  - Install a dependency: `npm install <dependency>`
  - Specific version: `npm install <dependency>@major.minor.patch`
  - In order to install multiple dependencies: `npm install`

    This command identifies all the needed dependencies, creates node_modules folder and downloads all the dependencies there.

Sample package.json

```
{
"name": "my-project",
"version": "1.5.0",
"description": "Express server project using compression",
"main": "src/index.js",
"scripts": {
"start": "node index.js",
"dev": "nodemon",
"lint": "eslint **/*.js"
},
"dependencies": {
"express": "^4.16.4",
"compression": "~1.7.4"
},
"devDependencies": {
"eslint": "^5.16.0",
"nodemon": "^1.18.11"
},
```

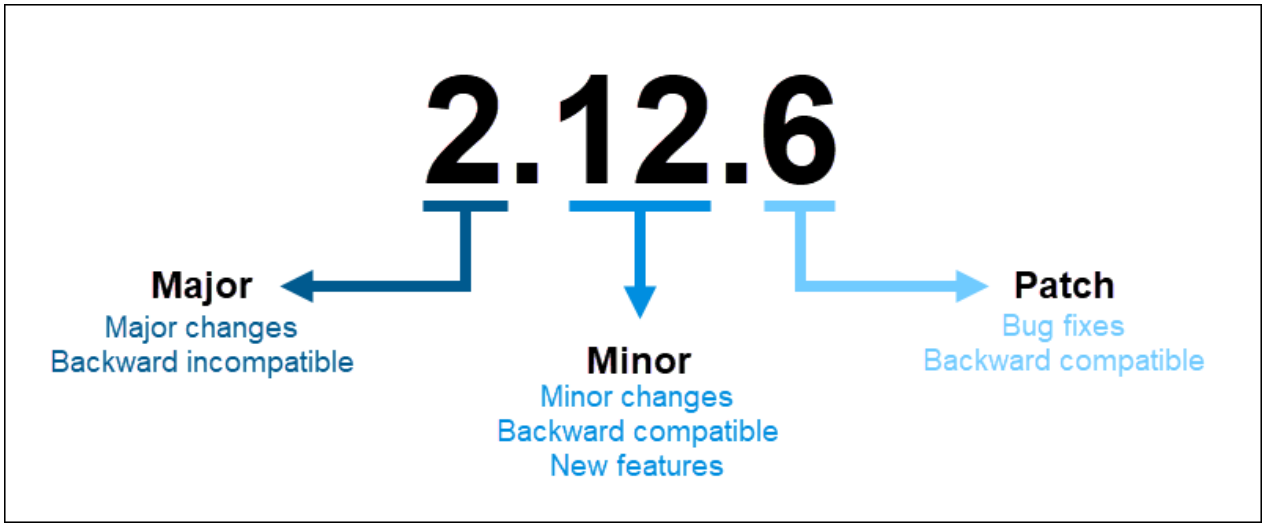
Dependencies

- Dependencies are the list of modules/packages that are required for your project to run.
- These are installed using npm install to add the package to the dependencies list.

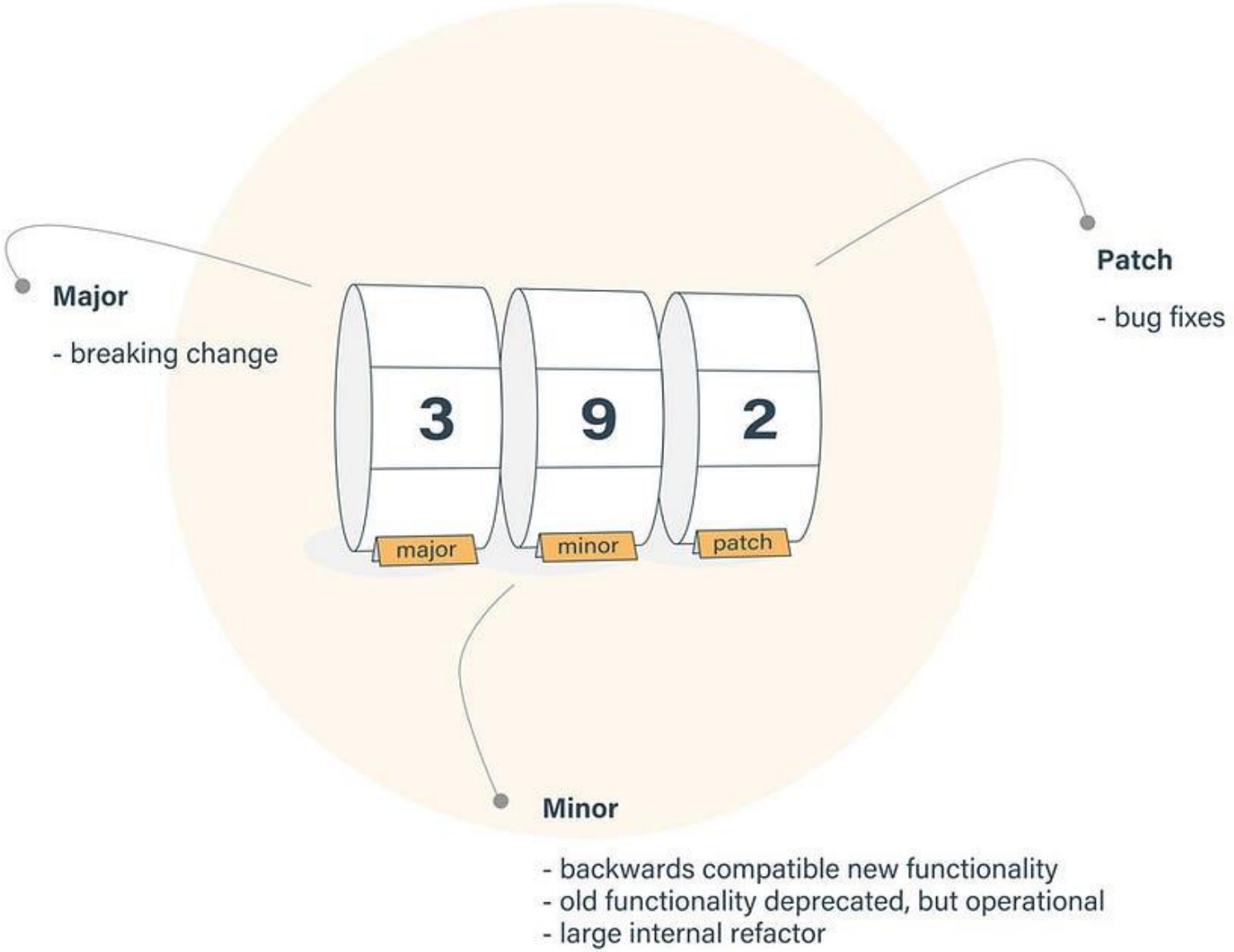
- devDependencies (short for development dependencies), are modules/packages that are NOT required for your project to run.
- These are often things that help the development process but aren't part of the project themselves.
- For example - eslint, nodemon, testing, etc.

Semantic Versioning (SemVer)

- The package versions are in the format `MAJOR.MINOR.PATCH`.
- A `MAJOR` version involves major code / functionality changes, which is often *backward incompatible*.
- A `MINOR` version change is *backwards compatible*, meaning it should work without breaking the existing things.
- A `PATCH` version change is *backwards compatible* bug fixes, or other small fixes.



Semantic Versioning (SemVer)



Dependencies -Tilde: Flexible Patch

```
"dependencies": {
    "express": "~4.17.0"
},
```

- Using a tilde (~) before the version number of the dependency means that it will accept only patch releases from the version specified but will not receive any major or minor release, if we were to install or update our dependency package.

- In the above example, if we try to update our npm package 'express', the latest version won't be installed but only further patch releases for **4.17**.X will be installed, if available.

Dependencies - Caret: Flexible Minor and Patch

```
"dependencies": {
    "loadash": "^3.9.3"
},
```

- Using a caret (^) before the version number of the dependency means that it can accept both patch and minor releases from the version specified but will not receive any major release if we were to install or update our dependency package.

- In the above example, if we try to update our npm package 'loadash', the latest major version won't be installed but only further minor & patch releases for **3**.X.X will be installed, if available.

Package.json vs package-lock.json

package.json	package-lock.json
It is a metadata file that describes the project's dependencies, scripts, configuration, and other details.	It is a lockfile that provides an exact, deterministic list of all the installed packages and their dependencies, including their exact version numbers.
It is typically created and modified manually by the developer to manage the project's dependencies and configuration.	It is automatically generated by npm and updated whenever you install or update packages.
It lists the required dependencies and their version ranges, but not the exact versions to be installed.	It is used to ensure that the same dependencies are installed consistently across different environments and prevent conflicts due to different versions being installed.
It can be easily shared and committed to version control systems.	It is not meant to be manually modified and should be committed to the version control system to ensure consistency across all team members.

Installing packages locally

- Use the following command to install any third party module in your local Node.js project folder.

```
npm install <package name>
```

- All the modules installed using NPM are installed under node_modules folder.

- For example - `npm install moment` command will create **moment** folder under node_modules folder in the root folder of your project and download moment.js (and other files) there.

Installing packages globally

- NPM can also install packages globally so that all the node.js applications on that computer can import and use the installed packages.

  ```
  npm install –g <package name>
  ```

- Apply -g in the install command to install package globally

- NPM installs global packages into `C:\Users\<user>\AppData\Roaming\npm\node_modules` folder.

- These packages are not listed in the dependencies section in the package.json.

Updating packages

- To update the version of the package installed locally in your Node.js project, navigate to the project folder using the command prompt or terminal window and write the following command

```
npm update <package name> --save
```

- For example - `npm update moment` command will update existing package '**moment**' to its latest version *(in package-lock.json file).*

- For example - `npm update moment --save` command will update the dependency version in package.json file.

Uninstalling packages

- Use the following command to remove a package from your project / Node.js application.

  ```
  npm uninstall <package name>
  ```

- For example - `npm uninstall moment` command will uninstall existing local package '**moment**' from the Node.js application.

- For example - `npm uninstall -g moment` command will uninstall global package '**moment**' from the system.

Publish packages

- To publish your package on the NPM registry, you need to have an account.

- After creating the account, open your terminal and run the following command in the root of your package: `npm login`

- You will get a prompt to enter your username and password. If login is successful, you should see a message like this:

```
Logged in as <your-username> on https://registry.npmjs.org/
```

Publish packages

- You can now run the following command to publish your package on the NPM registry: `npm publish`

- You can visit the NPM website and run a search for your package. You should see your package show up in the search result.

- When you publish a scoped package, you have the option to make it public or private. If it's private, you can choose who you want to share the package with.

Home Work

- What is NPM? What are the advantages of using NPM?
- What is package.json? How is it used to manage third party dependencies in your application?
- Explain the steps to create new project using npm

Home Work

- Explain with syntax –
  - → Installing a package locally & globally
  - → SemVer
  - → Caret (^) and Tilde (~) dependencies
  - → Updating packages
  - → Uninstalling packages (locally and globally installed)