Node package manager

Hilmy A. T.
[hilmi.tawakal@gmail.com]

Getting Started

What is npm?

NPM (node package manager) adalah package manager untuk javascript.

npm memudahkan pengembang Javascript untuk berbagi atau menggunakan kembali kode javascript, serta memudahkan untuk meng-update kode yang telah dibagikan.

Kode yang dibagikan atau digunakan kembali tersebut dinamakan ***package*** atau ***module***

Installation

Windows / Mac

Download file installer nodejs yang sesuai dengan sistem operasi yang digunakan

Linux

Untuk sistem operasi linux dapat juga dengan menggunakan binary dari masing-masing distro

Updating

- Untuk mengetahui versi nodejs yang terinstall, jalankan perintah:

```
node -v
```

- Untuk mengetahui versi npm yang terinstall, jalankan perintah:

```
npm -v
```

- Untuk menginstall atau update versi npm, jalankan sebagai administrator perintah:

```
npm install npm@latest -g
```

npm permission

- Saat mencoba untuk mengistall package secara global, akan muncul error EACCESS jika perintah tidak dijalankan sebagai super user

```
npm ERR! npm  v2.1.2
npm ERR! path /usr/local/lib/node_modules/jshint
npm ERR! code EACCES
npm ERR! errno 3

npm ERR! Error: EACCES, mkdir '/usr/local/lib/node_modules/jshint'
npm ERR!  { [Error: EACCES, mkdir '/usr/local/lib/node_modules/jshint']
npm ERR!   errno: 3,
npm ERR!   code: 'EACCES',
```

- Untuk memperbaikinya, ada 3 cara:
  - Ubah permission dari direktori default npm
  - Pindahkan direktori default npm ke direktori lain yang memiliki akses
  - Install node dengan package manager bawaan OS

Option 1. Change permission

1. Ketahui lokasi direktori default npm

   ```
   npm config get prefix
   ```

   Warning: jika lokasi default di ***/usr***, jangan lakukan cara ini, gunakan opsi lain.

2. Ubah permission direktori default tersebut

   ```
   sudo chown -R $(whoami) $(npm config get prefix)/{lib/node_modules,bin,share}
   ```

Option 2. Change npm default directory

1. Buat direktori untuk menyimpan package global

   ```
   mkdir ~/.npm-global
   ```

2. Ubah konfigurasi npm untuk menggunakan direktori default tersebut

   ```
   npm config set prefix '~/.npm-global'
   ```

3. Buka atau buat file ~/.profile dan tambahkan baris berikut

   ```
   export PATH=~/.npm-global/bin:$PATH
   ```

4. Update system variable

   ```
   source ~/.profile
   ```

Install npm package

- Terdapat dua cara dalam menginstall package atau modul npm, yaitu install scara global dan install secara lokal
- Saat menginstall secara global, package yang diinstall dapat diakses secara global, sedangkan jika diinstall secara lokal, package yang diinstall akan diletakkan pada direktori lokal modul atau aplikasi.
- Jika ingin menggunakan modul atau package untuk kepentingan pembuatan modul atau aplikasi, maka dapat dengan mengginstall secara default (install lokal)

```
> npm install lodash
> ls node_modules                 # use `dir` for Windows

#=> lodash
```

- Jika ingin menggunakan package yang berkaitan dengan CLI, maka lebih baik diinstall secara global denga menambahkan argumen -g .

```
npm install grunt-cli -g
```

Using **package.json**

- Cara terbaik untuk mengelola package secara lokal adalah dengan menggunakan package.json
- Beberapa keuntungan menggunakan package.json antara lain:
  - Menjadi dokumentasi yang menjelaskan package apa saja yang digunakan dalam project
  - Memungkinkan untuk memilih versi package yang sesuai untuk digunakan dalam project
  - Memudahkan untuk direproduksi kembali sehingga memudahkan ketika projectakan didistribusikan
- package.json minimal harus berisi dua hal berikut
  - “Name”
    - All lowercase
    - One word, no space
    - Dashes and underscore allowed
  - “version”
    - In the form of `x.x.x`

Creating a **package.json**

- package.json dapat dibuat dengan menggunakan perintah `npm init`
- Perintah tersebut akan memunculkan serangkaian pertanyaan untuk mendeskripsikan isi dari package.json
- Perintah diatas juga dapat ditambahkan attribute **`-y`** untuk men-generate package.json dengan nilai default:
  - **name**: the current directory name
  - **version**: always **1.0.0**
  - **description**: info from the readme, else an empty string **""**
  - **main**: always **index.js**
  - **scripts**: by default creates an empty **test** script
  - **keywords**: empty
  - **author**: empty
  - **license**: **ISC**
  - **bugs**: info from the current directory, if present
  - **homepage**: info from the current directory, if present

Specifying packages

- Untuk menentukan package apa saja yang dibutuhkan oleh project, nama package yang dibutuhkan harus dituliskan dalam package.json
- Terdapat 2 jenis dependensi:
  - “dependencies”: dependensi yang dibutuhkan oleh aplikasi ketika sudah dirilis
  - “devDependencies”: dependensi yang dibutuhkan ketika aplikasi masih dalam tahap pengembangan

```
{
  "name": "my_package",
  "version": "1.0.0",
  "dependencies": {
    "my_dep": "^1.0.0"
  },
  "devDependencies" : {
    "my_test_framework": "^3.1.0"
  }
}
```

Editing package.json

- Untuk merubah isi package.json, dapat dilakukan dengan merubah file package.json secara langsung secara manual
- Atau dapat pula dengan menambahkan argument `--save` ketika mengistall package
- Untuk menambahkan list package kedalam “dependencies” dapat dengan command berikut:

```
npm install <package_name> --save
```

- Untuk menambahkan list package kedalam “devDependencies” dapat dengan command berikut:

```
npm install <package_name> --save-dev
```

Updating local packages

- npm menggunakan semantic versioning dalam mengelola versi package
- Saat menjalankan perintah npm install pada diektori yang terdapat file package.json npm akan melihat list dependensi yang ada pada package.json dan mendownload versi terbaru dari package tersebut sesuai dengan ketentuan semantic versioning
- Untuk melihat package yang terinstall secara lokal dapat dengan perintah npm ls
- Untuk melihat package apa saja yang perlu diupdate gunakan perintah npm outdated

- Untuk melakukan proses update, gunakan perintah npm update

Uninstalling local package

- Untuk melakukan proses uninstall terhadap package lokal, dapat dilakukan dengan perintah npm uninstall <package-name>, seperti contoh berikut:

  ```
  npm uninstall lodash
  ```

- Untuk melukan uninstall sekaligus menghilangkan package dalam list dependensi pada package.json tambahkan argumen --save atau --save-dev untuk menghilangkan dari devDependencies

  ```
  npm uninstall --save lodash
  ```

Creating Node.js modules

- Buatlah sebuah folder project yang berisi file package.json dan file index.js
- Untuk memudahkan, dapat dilakukan melalui perintah npm init
- Contoh, pada file index.js isilah dengan kode berikut:

```
exports.printMsg = function() {
     console.log("This is a message from the demo package");
}
```

- Untuk mengujinya, publish kode tersebut dan jalankan pada project lain

Publishing npm packages

Creating a user

- To publish, you must be a user on the npm registry
- If you don't have one, create it with `npm adduser`
- If you created one on the site, use `npm login` to store the credentials on the client.
- Use `npm config ls` to ensure that the credentials are stored on your client.

Publishing the package

- Use `npm publish` to publish the package.
- Also, make sure there isn't already a package with the same name owned by somebody else.

How npm works

Packages and Modules

- **A package is a file or directory that is described by a package.json.**
- A package is any of the following:

  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 <name>@<version> that is published on the registry with (c).
  e) A <name>@<tag> that points to (d).
  f) A <name> that has a latest tag satisfying (e).
  g) A git url that, when cloned, results in (a).

Packages and Modules

A module is anything that can be loaded with require() in a Node.js program. The following are all examples of things that can be loaded as modules:

- A folder with a package.json file containing a main field.
- A folder with an index.js file in it.
- A JavaScript file.

Package dependencies