NodeJS NPM

Rohmat Arief Budiarto

Agenda

- Pengenalan Node Package Manager
- Membuat Project
- Project Configuration
- Project Type
- Script
- Dependency Management
- Dan lain-lain

Pengenalan Node Package Manager

- Saat kita membuat aplikasi, biasanya kita akan buat dalam bentuk project
- Sederhananya, project adalah directory/folder yang berisikan kode program dan dependency (library) yang kita butuhkan
- Melakukan management kode program dan dependency management secara manual bukanlah hal yang menyenangkan
- Untungnya, NodeJS menyediakan NPM (Node Package Manager) yang bisa kita gunakan untuk melakukan management project NodeJS

Kegunaan NPM

- NPM tidak hanya digunakan untuk melakukan management project NodeJS
- NPM juga bisa digunakan untuk melakukan dependency management yang kita butuhkan dalam project yang kita buat
- NPM bisa digunakan untuk download dependency, update dan upgrade dependency secara otomatis, tanpa harus kita lakukan secara manual dengan cara download dependency secara manual

File package.json

- Saat kita melakukan management project menggunakan NPM
- NPM menyimpan semua konfigurasi project di file bernama package.json
- Semua konfigurasi project dan juga dependency kita simpan dalam file package.json yang terdapat di dalam directory project
- Kita bisa buat file package.json secara manual, atau menggunakan auto generate secara otomatis menggunakan NPM

NodeJS Dependency Repository

- Saat kita melakukan management dependency menggunakan NPM
- NPM akan melakukan pencarian dan download dependency secara otomatis dari website https://www.npmjs.com/
- Kita juga bisa mencari dependency di website https://www.npmjs.com/

Menginstall NPM

- Kita tidak perlu menginstall NPM secara manual
- Saat kita menginstall NodeJS, secara otomatis NPM akan terinstall juga
- Untuk mengecek versi NPM yang terinstall di komputer kita, kita bisa gunakan perintah : npm --version

Membuat Project

- Buat folder untuk project
- Masuk ke folder project melalui terminal/command line
- Buat project NPM baru dengan perintah :
  npm init

package.json

- Inti dari konfigurasi project NodeJS adalah package.json
- File package.json merupakan konfigurasi yang berupa format json untuk project NodeJS yang kita buat

Project Configuration (1)

Attribute	Description
name	Nama project / package
version	Versi project
description	Deskripsi project
homepage	Home page project
author	Author project

Project Configuration (2)

Attribute	Description
contributors	Nama kontributor project
main	Entry point package
keywords	Keyword project
license	Lisensi project
repository	Repository project

Menjalankan Project

- Untuk menjalankan project, sebenarnya kita lakukan buat sama seperti ketika kita menjalankan script NodeJS
- Kita bisa gunakan perintah
  node filescript
- Yang membedakan adalah, NodeJS sebelum menjalankan file nya, dia akan membaca konfigurasi dari package.json terlebih dahulu

Project Type

- Secara default, saat kita membuat project NodeJS, NodeJS menggunakan commonjs
- Oleh karena itu, ketika kita ingin menggunakan JavaScript Modules, kita harus mengubah file nya menjadi file mjs
- Namun, kita juga bisa mengubah default project type dari commonjs menjadi js modules, dengan cara mengubah type di package.json
- Sangat direkomendasikan sekarang menggunakan js modules dibanding commonjs, karena js modules sudah menjadi standard di JavaScript

Script

- NPM memiliki fitur yang bernama script, dimana kita bisa menyediakan perintah script yang nanti bisa digunakan untuk menjalankan perintah lainnya
- Penggunaan script ini biasanya digunakan untuk mempermudah ketika kita menjalankan perintah yang panjang
- Untuk menambahkan script, kita bisa tambahkan script nya di package.json

Menjalankan Script

- Untuk menjalankan script, kita bisa gunakan NPM dengan perintah :
  npm run-script namascript
- Untuk menjalankan script test dapat digunakan NPM dengan perintah:
  npm run-script test

Special Script

- Script di package.json terdapat beberapa yang spesial atau khusus
- Script tersebut tidak perlu dijalankan menggunakan npm run-script namascript, tapi bisa langsung dijalankan menggunakan perintah npm namascript
- Contoh special script yaitu : start, stop, test, restart, uninstall, version, dan lain-lain
- Selain itu, terdapat script spesial untuk script diatas, kita bisa gunakan gunakan prefix pre sebagai script yang akan dijalankan sebelumnya, dan prefix post sebagai script yang dijalankan setelahnya
- Misal ketika kita gunakan perintah npm start, maka akan menjalankan script prestart, start dan poststart

Modul Export

Main

- Sampai saat ini, kita tidak pernah membahas tentang attribute main di package.json
- Attribute main adalah entry point yang akan di-load ketika kita me-load NodeJS Project / Package
- Pada kasus ketika kita membuat aplikasi, mungkin tidak terlalu berguna, tapi pada kasus ketika kita membuat library yang akan digunakan di banyak project, baru attribute main ini akan terlihat kegunaanya

Export Module

- Problem ketika kita menggunakan attribute main adalah, kita cuma bisa mengekspos satu file JS, oleh karena itu penggunaan attribute main sebenarnya sudah tidak direkomendasikan lagi
- Sebagai penggantinya, terdapat attribute export yang bisa digunakan sebagai konfigurasi untuk mengekspos file JS
- Yang menarik dari fitur export ini, kita bisa membuat alias ketika mengekspos file JS, sehingga tidak perlu menggunakan nama file JS aslinya

Menggunakan Module

- Untuk menggunakan module yang sudah di export, kita cukup gunakan import dari nama module yang di export tersebut, namun ganti tanda . (titik) dengan nama package yang ada di package.json
- Misal
  “.” menjadi “belajar-nodejs-npm”
  “./write” menjadi “belajar-nodejs-npm/write”

Dependency Management

- Salah satu fitur yang sangat berguna dalam NPM adalah dependency management
- Saat kita membuat aplikasi, sering sekali kita akan membutuhkan dependency ke library atau package pihak lain, misal package open source, atau package yang kita buat sendiri
- NPM mendukung dependency management, sehingga kita tidak perlu download package yang kita butuhkan secara manual, termasuk tidak perlu melakukan update dependency secara manual ketika ada update terbaru

npmjs.com

- Secara default, NPM akan download dependency dari website https://www.npmjs.com/
- Kita bisa mencari open source package atau membuat open source package disana jika kita mau
- Untuk menginstall dependency, kita bisa gunakan perintah :
  npm install namadependency@version
- Atau bisa langsung tulis di dependencies di package.json

Download Dependency

- Untuk download dependency, kita bisa gunakan perintah :
  npm install
- Secara otomatis NPM akan download package yang ada di dependency ke dalam folder node_modules
- Selain itu, NPM juga akan membuat file package-lock.json yang berisikan informasi versi package yang di download, ini untuk mempermudah ketika kita melakukan download ulang library di komputer lain

Dependency Version

Semantic Version

- NodeJS merekomendasikan menggunakan semantic version dalam menentukan format version pada package yang kita buat
- https://semver.org/
- Jika kita perhatikan, kebanyakan package di https://www.npmjs.com/ menggunakan semantic version

Dependency

- Salah satu kegunaan menggunakan semantic version adalah, kita bisa menentukan versi package yang ingin kita gunakan secara dinamis, tanpa harus melakukan hardcode pada versi tertentu
- Ada beberapa aturan yang bisa kita gunakan ketika menentukan versi dependency yang ingin kita gunakan di package.json

Menentukan Versi Dependency (1)

Versi	Keterangan
x	Download versi terbaru dan update ke versi terbaru walaupun MAJOR berubah
1.x	Download versi 1 terbaru, update ke versi terbaru, tapi MAJOR tetap di 1
1.1.x	Download versi 1.1 terbaru, update ke versi terbaru, tapi MAJOR dan MINOR tetap di 1.1
1.1.1	Selalu download versi 1.1.1, tidak akan update walaupun ada versi baru

Menentukan Versi Dependency (2)

Versi	Keterangan
~1.1.1	Download versi 1.1.1, jika ada update, lakukan update, namun hanya update jika PATCH berubah
^1.1.1	Download versi 1.1.1, jika ada update, lakukan update, namun hanya update jika MINOR dan PATCH berubah

https://semver.npmjs.com/

Development Dependency

- Di package.json, terdapat dua jenis dependency, production dependency dan development dependency
- Dependency management yang sebelumnya kita bahas adalah production dependency, yaitu dependency yang dibutuhkan ketika aplikasi kita berjalan
- Sedangkan development dependency, adalah dependency yang dibutuhkan khusus ketika proses development, contoh yang sering misal dependency unit testing, yang cukup digunakan ketika development, tetapi tidak dibutuhkan ketika aplikasi berjalan

Menambah Development Dependency

- Untuk menambah development dependency, kita bisa tambahkan di bagian devDependencies di package.json
- Atau gunakan perintah :
  npm install namapackage --save-dev
- Untuk download dependency, sama seperti download production dependency, kita bisa gunakan perintah :
  npm install

Install Dependency Tanpa Development

- Secara default, saat kita gunakan perintah npm install, semua dependency akan di install, termasuk development
- Ada baiknya, ketika kita mau jalankan aplikasi di production, kita tidak perlu menginstall development dependency, caranya kita bisa gunakan perintah:
  npm install --production