# INTRODUCTION

Dalam section ini kita akan bahas NPM (Node Package Manager). Kemungkinan besar, node modules yang kalian butuhkan dalam pembuatan 1 aplikasi utuh, ada script open source nya disini.

Kemarin kita sudah sempat coba untuk lihat versi npm, dengan eksekusi command npm -v.

Just in case kalian mau install npm dengan versi lain, kalian bisa eksekusi npm i (which is singkatan dari install) -g (global) npm@version number. Di mac, kemungkinan kalian harus menambahkan prefix “sudo” karena kemungkinan permission di computer kalian ada yang belum di set up properly.

Klo di windows mengalami masalah, kalian bisa buka command prompt sebagai administrator privilege.

PACKAGE .JSON

Sekarang mari kita coba buat bikin 1 package. Buat 1 folder dengan nama npm-demo. Sebelum kalian tambahkan package node apapun kalian perlu membuat 1 file Namanya package.json. Package.json pada dasarnya adalah json file yang memiliki informasi tentang aplikasi kalian. Seperti nama aplikasi, versi, git repo, dll.

Untuk membuat package.json file, bisa menggunakan npm init.

A screenshot of a cell phone

Description automatically generated

Kalian akan melihat wizard yang akan membantu kalian dalam membuat package.json file. Nah, ad acara yang lebih mudah untuk membuat package.json file. Dengan mengetik package.json –yes. Perintah ini akan membuat package.json file dengan default value, sehingga kita tidak perlu menjalani wizard, satu per satu.

# INSTALL A NODE PACKAGE

Kali ini kita akan coba buat install 3rd party library di NPM ke dalam aplikasi node kita, underscore. Kita ke npmjs.org, lalu search underscore.

A screenshot of a social media post

Description automatically generated

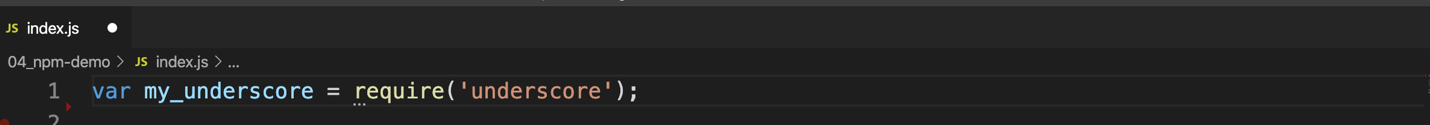
Ini adalah cara kita install underscore. Kembali ke terminal, kita coba install underscore package ke dalam node kita.

Setelah melakukan hal ini, 2 hal terjadi. Di package.json ada property baru, dependency, dan dibawah nya ada underscore. Jadi di package.json ini, akan di list, seluruh dependency yang digunakan dalam node app kalian.

Pada saat npm install dilakukan, maka node akan download package yang dibutuhkan, dan disimpan di dalam folder node\_modules.

# USING A PACKAGE

Sekarang kalian sudah install underscore. Kita akan coba gunakan 3rd party library itu di dalam node kita. Kembali ke code editor, kita buat 1 file dengan nama index.js di root folder npm-demo.



Saya terangkan sedikit mengenai bagaimana cara node membaca module require ini. Pada tahap awal, node akan membaca path require sebagai core module.

A picture containing light

Description automatically generated

Nah, seperti yang kita ketahui, underscore ini, kita dapat dari 3rd party library di npm. Node sendiri pada dasarnya tidak punya module ini. Dengan demikian, berarti node tidak menemukan core module dengan nama underscore.

Langkah kedua, node akan mencoba mencari file folder underscore dengan entry point index.js

Hal ini tidak ketemu juga. Dengan demikian, node akan pindah ke langkah ketiga.

Node akan mengasumsikan kalua underscore adalah module yang terdapat di node\_modules folder.

Sekarang kita coba akses website https://underscorejs.org/.  
yang akan kita coba gunakan adalah function contain. Mirip seperti method di php in\_array. Function contain ini akan mencari apakah sebuah value ada di dalam array. Kalau ada, return true, kalau tidak ada, return false.

Function contain ini memiliki 2 parameter. Parameter pertama adalah array utuh, parameter r kedua adalah value yang ingin dicari.

A screenshot of a cell phone

Description automatically generated

\*\* run app \*\*

Sekarang saya coba kasih 1 test kecil, coba kalian install 1 package Namanya mongoose. Mongoose ini adalah package yang bisa kita pakai untuk simpan data di mongodb database.

# PACKAGE DEPENDENCIES

Setelah kalian install mongoose, coba cek package.json file kalian. Seharusnya disini tertulis mongoose, dan underscore. Dan sekarang, coba kita cek folder node\_modules. Ternyata banyak banget folder yang di download oleh node, padahal kita Cuma perlu underscore dan mongoose. Ada yang punya gambaran, kira2 kenapa ini bisa terjadi?

Jawabannya adalah, karena mongoose dan underscore pun, dibuat oleh pihak ketiga. Dan module tersebut masing2, pasti punya dependency juga dengan modul lain. Itu lah kenapa, pada saat kita download 2 module ini, modul2 yang terkait dengan modul2 ini juga ikut terdownload, agar bisa digunakan.

Bayangkan kalua kalian ternyata pakai puluhan module atau belasan modul. Node modules folder berarti akan jadi sangat besar size nya. Tentu ini akan mempersulit kalian saat nanti mau kirim source code ke teman, atau bahkan upload source code ke server. Di node ini, kita punya cara aman nya.

Node modules bisa kita hapus dari root folder, dan simply jalanin lagi command npm i. Node akan otomatis download lagi seluruh package yang ada di dalam daftar dependency di package.json file kalian. Itulah sebabnya, package.json file sangatlah penting.

Ada cara kedua untuk ini, yaitu menggunakan .gitignore. Coba jalankan git.status. Seperti yang kita lihat, node modules masih ada dalam file yang dipantau oleh git. Sekarang kita mau exclude node\_modules. Caranya adalah dengan membuat 1 file .gitignore. masukan **node\_modules/**

Kalau kita buat git status, node modules akan hilang. Beginilah cara kita untuk exclode file ataupun folder dari git repository. Finally, kita jalankan perintah git add .. Lalu eksekusi git commit -m “pesan”.

# SEMANTIC VERSIONING (SAMVER)

Di semantic versioning, kita mengenal 3 angka. Angka pertama adalah major version, angka kedua minor version, angka ketiga adalah patch release.

Apabila ada bugs yang diperbaiki, yang ditingkatkan adalah angka patch nya.

Minor version digunakan untuk menambahkan fitur minor baru yang tidak mengganggu existing API.

Saat upgrade minor feature, maka angka jadi 4.13.0. Saat angka 0, berarti maksudnya, mereka belum menemukan bugs apapun. Ini berarti, bisa jadi versi ini unstable. Saat mereka menemukan bugs dan release patch baru, versi naik jadi 1.

Nah, di saat mereka ada melakukan penambahan fitur atau perubaha, yang berpotensi mengganggu fungsi2 lainnya yang sudah ada, maka mereka akan meningkatkan major version. Dan minor version serta patch versionnya, reset jadi 0.

Di semantic versioning, kita mengenai 2 simbol.

^ Caret symbol 4.x.x

~ Tilde symbol 1.8.x

A screenshot of a cell phone

Description automatically generated

Nah kalua kalian mau menggunakan dependency yang sama persis, maka kalian tinggal hilangkan saja simbolnya.

Nah, pertanyaan nya adalah, bagaimana sebaiknya kita menyingkapi ini? Haruskah kita pakai tanda Caret, Tilde, atau sama persis? Jawabannya, tidak ad acara yang pasti. Kalian harus memantau perkembangan masing2 dependency yang kalian gunakan dari 3rd party library.

Tapi pada dasarnya, biasanya perkembangan dari sebuah module / package, pasti akan diimbangi dengan perkembangan dari dependency nya, tinggal tergantung waktu saja. Jadi, tetap pantau saja perkembangannya. Dan 1 hal yang kalian perlu waspadai. Hilangkan kebiasaan untuk update module, tanpa baca dulu apa yang berubah. Karena bisa jadi, kalian update ke versi yang incompatible dengan yang kalian buat, terutama pada saat kalian sudah punya live app yang customer gunakan.

Ada cara cepat ga, untuk tau ada update atau ga? Simple. Kita bisa pakai command outdated di root folder node app kalian.

A screen shot of a social media post

Description automatically generated

Bagaimana kalua mau uninstall

Tinggal ketik npm uninstall atau npm un, dan nama dari package let’s say mongoose.

Dan bagaimana kalau mau update?

Tinggal ketik npm update