NPM - Package

Giới thiệu chung về NPM

NPM viết tắt của **N**ode **P**ackage **M**anager là một công cụ tạo và quản lý các thư viện lập trình Javascript cho NodeJS. Trong cộng đồng Javascript, các lập trình viên chia sẻ hàng trăm nghìn các thư viện với các đoạn code đã thực hiện sẵn một chức năng nào đó. Nó giúp cho các dự án mới tránh phải viết lại các thành phần cơ bản, các thư viện lập trình hay thậm chí cả các framework.

Nếu bạn vẫn chưa tưởng tượng ra lợi ích của việc sử dụng npm thì ta sẽ lấy ví dụ cụ thể hơn.(Các ví dụ dưới đây đều không bao gồm các trường hợp các bạn sử dụng 1 package manager nào khác, ví dụ như yarn, maven…)

Ví dụ 1: Nếu bạn không sử dụng npm, bạn sẽ phải download toàn bộ các thư viện một cách thủ công, sau đó include vào project của bạn, việc này rất mất thời gian, trong khi với npm, bạn chỉ cần 1 dòng lệnh là xong.

Ví dụ 2: Khi làm 1 dự án, bạn phải chia sẻ code cùng với các cộng sự của mình, nếu không sử dụng npm (hoặc bất kì trình quản lí package nào) thì khi commit code ta phải commit cả thư viện vào, rất nặng. Khi deploy ta cũng phải copy thư viện lên, rất chậm và tốn thời gian.

Cài đặt và sử dụng NPM

Npm được tích hợp sẵn có trong NodeJS, để kiểm tra xem trên hệ thống của bạn đã được cài npm chưa chúng ta sử dụng lệnh **npm -v**, nếu một phiên bản hiện ra thì hệ thống của bạn đã được cài đặt npm.

Nếu bạn tạo mới 1 project và muốn sử dụng npm, tốt hơn hết bạn nên bắt đầu với câu lệnh:

```
npm init
```

Câu lệnh trên đơn giản là sẽ tạo ra 1 file có tên là `package.json` – thành phần này được gọi là `Local Package Database`, lưu trữ thông tin (tên package, phiên bản, các dependencies) mà project của bạn sử dụng.

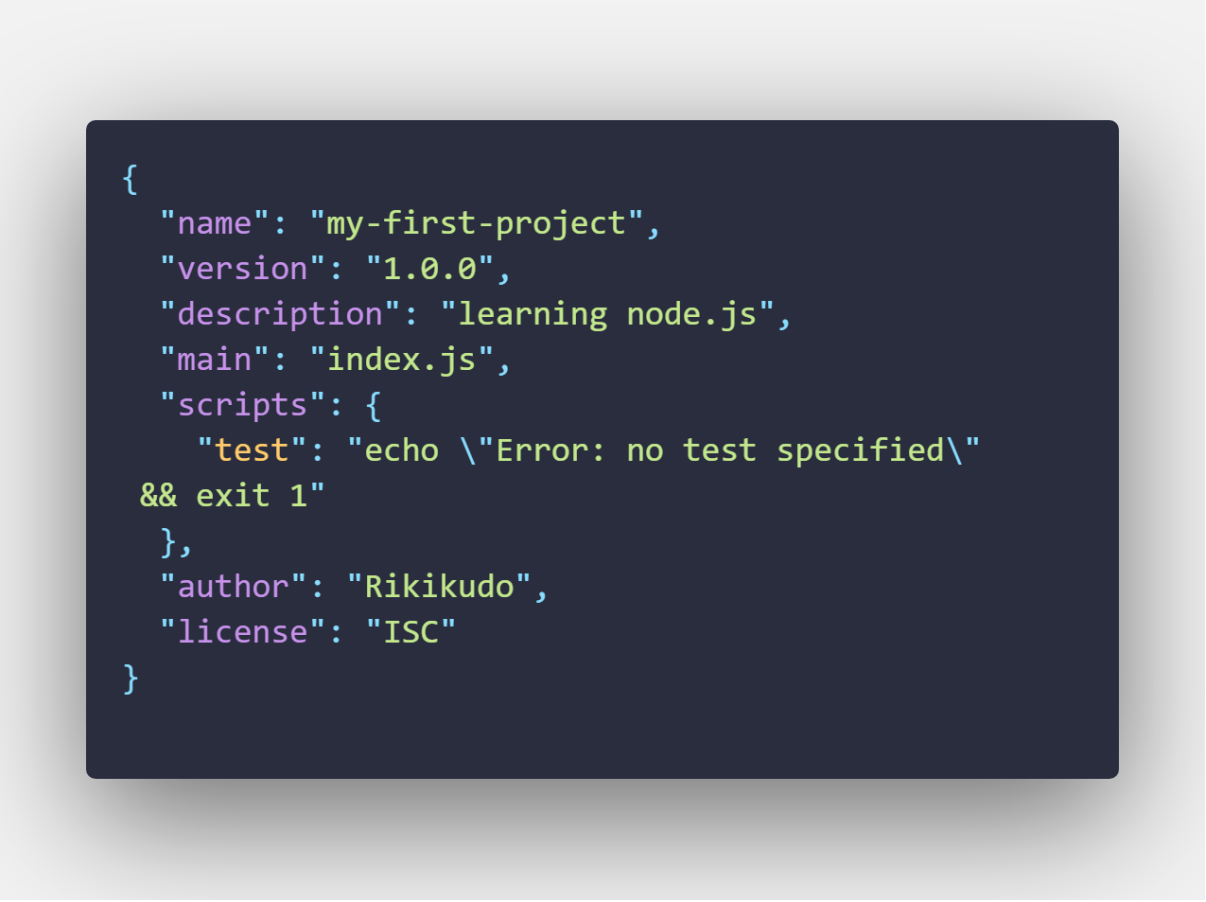
Sau khi chạy câu lệnh này, Npm sẽ hỏi chúng ta một vài câu hỏi về project của bạn.

- Đầu tiên là về package name: Đây là tên dự án của bạn

- Version: Đây là phiên bản dự án của bạn, mặc định sẽ là 1.0.0
- Description: Mô tả về dự án của bạn.
- Entry point: Entry point sẽ quy định root của Node, file này sẽ chứa các câu lệnh về server, mặc định sẽ là index.js. Chú ý là khi bạn chạy server, Node sẽ đọc dữ liệu từ file này để khởi tạo. Tên này có thể thay đổi được, nhưng hãy chắc rằng file đó tồn tại ở root của dự án.
- Test command: Đây là dòng lệnh mà sẽ chạy khi bạn gọi **npm test** Mặc định để trống
- Git repository: Như tên gọi, nó là git repository ứng với dự án của bạn. Mặc định để trống.
- Keywords: Các từ khóa tương ứng với dự án của bạn. Mặc định để trống
- Author: Tên tác giả của dự án. Mặc định để trống
- License: Giấy phép. Mặc định là ISC.

Khi kết thúc sẽ hỏi 1 câu là Is this OK: bạn có chắc không, yes hoặc no.

Sau khi kết thúc, project của các bạn sẽ có thêm 1 file có tên là **package.json**. Khi bạn vào trong đó ban đầu sẽ file hiển thị những thông tin tương tự như dưới đây.



Các thông tin trong file này ban đầu chính là những thông tin bạn nhập trước đó trong phần **npm init.** Trong tương lai bạn có thể add rất nhiều thứ vào file này nữa. Và giờ bạn có thể sử dụng npm để cài đặt các package mà mình muốn include vào trong dự án của mình rồi.

Các loại package

Dựa theo chức năng mà ta chia package ra làm 2 loại, đó là **Simple dependencies** (Để cho ngắn gọn thì về sau mình sẽ chỉ gọi nó là **Dependecies** thôi) và **Development dependencies.** Về sự khác nhau giữa 2 loại này ra sao.

Dependencies là những package **bắt buộc phải có trong quá trình chạy sản phẩm**, kiểu như một thư viện cung cấp các hàm mà code của bạn cần.

Khi cài đặt dependencies, Npm sẽ tự động cài đặt tất cả các dependencies cần thiết.

Ví dụ như nếu package A phụ thuộc vào package B, package B lại phụ thuộc vào package C thì Npm sẽ cài đặt đồng thời cả A,B,C.

Đó cũng chính là lí do vì sao khi bạn cài đặt 1 package và vào folder node_modules sẽ thấy cả đống các package khác mà bạn không hề cài đặt

Còn đối với **Development dependencies là những package bắt buộc khi phát triển cũng như phát hành sản phẩm.** Kiểu như các trình biên dịch giúp biên dịch đoạn code của bạn về javascript, rồi các framework phục vụ cho việc kiểm thử,…

Khi cài đặt Development Dependencies, Npm sẽ chỉ cài đặt các dependencies mà cần thiết.

Ví dụ như nếu package A phụ thuộc vào package B, package B lại phụ thuộc dev vào package C thì Npm sẽ chỉ cài đặt A và B.

Giờ chúng ta sẽ xem chúng ta cài đặt các package như thế nào nhé.

- **Cài đặt Simple dependencies package:**

Cú pháp như sau: `npm install [tên package]`

hoặc đơn giản hơn là `npm i [tên package]`

- Cài đặt **Development dependencies package:**

Cú pháp như sau**:** `npm install [tên package] --save-dev`

hoặc đơn giản hơn là**:** `npm i [tên package] --save-dev`

Cùng với việc thêm file **package.json**, khi bạn cài đặt package, project của bạn sẽ có thêm 1 folder có tên là node_modules, trong đó sẽ chứa tất cả các package đã được cài đặt.

Các kiểu cài đặt package

Chúng ta có 2 kiểu cài đặt package: **Local và Global**

Kiểu cài đặt ở trên là cài đặt **local**, tức là các package ở trên được cài đặt và chỉ sử dụng được trong project hiện tại mà nó được cài đặt, với các project khác sẽ không sử dụng được.

Kiểu cài đặt thứ 2 là **global**, các package sẽ được cài đặt global, đúng như tên gọi của nó, tức là bạn có thể sử dụng nó ở không chỉ project hiện tại mà ở các project khác mà không cần phải cài đặt lại nữa.

Các package được cài đặt kiểu global cũng sẽ được lưu trong node_module nhưng ở 1 đường dẫn khác.

Để xem các package được cài đặt global, ta có thể sử dụng lệnh:

`npm list -g`

Để xem đường dẫn nơi chứa các package này, ta sử dụng lệnh

`npm root -g`

Để cài đặt package kiểu global, bạn viết câu lệnh giống như của Simple Dependencies thêm flag -g hoặc –global

Package Version và Cập nhật package

Giờ chúng ta sẽ đi vào chi tiết hơn về Package Version:

Quay trở lại với hình ảnh ở trên:

Ví dụ với package này:

Tên package: `slugify`

Version của package: `1.3.6`

Version của các package thường sẽ có dạng x.x.x

Mỗi 1 số (được ngăn cách bởi dấu '.') đều có ý nghĩa riêng.

Trong ví dụ trên, xét từ phải qua trái.

+ Số đầu tiên được gọi là `patch version`, patch version là gì? có thể hiểu đơn giản nó là số thứ tự bản vá lỗi. Ví dụ như ở version 1.3 này, ai đó phát hiện có lỗi và sửa lại nó, sau khi sửa lại, bản mới này sẽ có tên là 1.3.1, cứ như vậy, bản vá thứ 2 sẽ có tên là 1.3.2, ….. Ở ví dụ trên thì đây là bản vá lỗi thứ 6 của version 1.3

+ Số thứ 2 được gọi là `minor version`, minor version bao gồm những tính năng mới, update nhỏ mà không làm thay đổi những phần trước đó đã có. Giả sử 1 ngày đẹp trời, team phát triển slugify tìm tòi sáng tạo ra 1 vài tính năng mới, họ sẽ cho ra lò slugify bản 1.4, bản mới này chỉ có thêm tính năng, không hề ảnh hưởng đến những project sử dụng các bản cũ hơn (1.1, 1.2..)

+ Số cuối cùng được gọi là `major version`, major version sẽ có những thay đổi, có thể là update, thêm tính năng, … và những thay đổi đó có thể khiến những project dùng bản trước đó bị lỗi. Chẳng hạn 1 ngày đẹp trời khác, team của slugify không thích sử dụng những tên hàm, tên các option đã có, họ muốn thay đổi chúng, giả sử bản hiện tại đang là 1.3.5, nếu muốn thực hiện các thay đổi đó, họ không thể cho ra lò bản 1.4 được vì giữa 2 bản này sẽ có sự khác biệt quá xa nhau mà phải cho ra lò version 2.

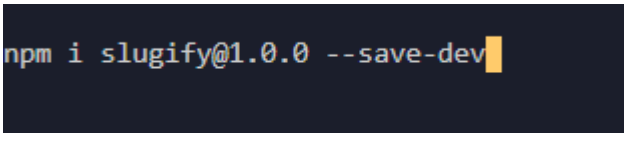
Ta phân biệt giữa minor và major ở chỗ này.

- Ở minor, giữa bản 1.3.x và 1.4.x, 1.3.x có gì thì 1.4.x sẽ có cái đó, ngoài ra sẽ bổ sung 1 vài tính năng mới.
- Còn với major, giữa bản 1.x.x và 2.x.x có thể có sự khác biệt không chỉ về tính năng, mà còn cả về tên hàm, tên các options có thể thay đổi… code bạn chạy ngon khi dùng bản 1.x.x không có nghĩa là update lên bản 2.x.x code bạn vẫn chạy ngon.

Npm cũng cho phép chúng ta được phép cài đặt package với version cụ thể. Cú pháp như sau:

npm install [tên package]@[version_bạn_muốn_cài] [flag_nếu_có]

flag ở đây là `--save-dev` hoặc `--save` hoặc `--g`



Câu lệnh trên sẽ cài đặt package slugify version 1.0.0

Các bạn phải chú ý khi cài đặt version vì như đã nói ở trên, giữa các version với nhau đều có những thay đổi ít nhiều có thể làm code của bạn không hoạt động như mong muốn nữa.

Update package

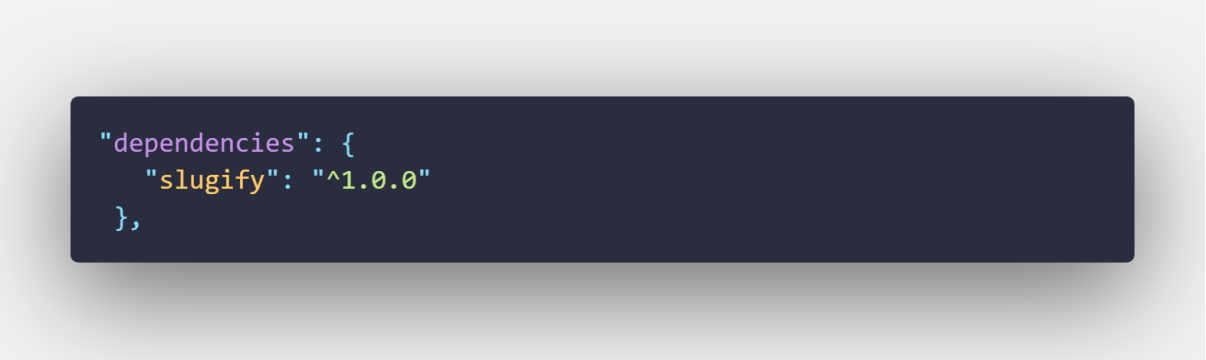
Để update package, các bạn sử dụng cú pháp:

`npm update [tên_package]`

Nhưng các bạn phải chú ý, việc update đến version nào còn phụ thuộc vào một điểm nữa mà mình chưa nhắc tới ở phần giới thiệu về package version, đó là kí hiệu ở trước version.

Có 2 kí hiệu tiền tố phổ biến nhất đó là dấu ^ và dấu ~

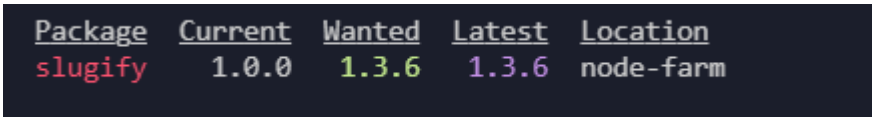
Khi bạn cài đặt package thì mặc định tiền tố sẽ là dấu ^



Với tiền tố ^. Khi update, bạn sẽ được update đến bản minor mới nhất.

Lấy ví dụ package ở trên, khi ta update package slugify, nó sẽ được update tới version mới nhất dạng `1.x.x` trong đó `1.0.0 <= 1.x.x < 2.0.0`

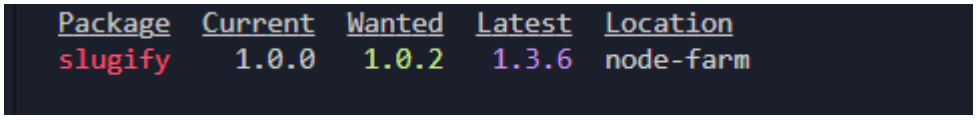
Nếu bạn sử dụng lệnh npm outdated sẽ thấy kết quả như sau:



Như có thể thấy, version mới nhất mà bạn có thể cập nhật được nếu sử dụng ^ là version 1.3.6, vì nó là bản minor mới nhất hiện có và thỏa mãn 1.0.0 < 1.3.6 < 2.0.0. Tuy nhiên, nếu chúng ta thay đổi kí tự ^ thành ~ thì sẽ khác:

```
"dependencies": {
    "slugify": "~1.0.0"
},
```

Khi chạy lệnh npm outdated sẽ cho ra kết quả:



Package	Current	Wanted	Latest	Location
slugify	1.0.0	1.0.2	1.3.6	node-farm

Sử dụng kí tự ~ có nghĩa là khi update, bạn sẽ được update đến bản vá lỗi mới nhất. Trong ví dụ package ở trên, khi ta update package slugify, nó sẽ được update tới version với dạng `1.0.x` trong đó `1.0.0 <= 1.0.x < 1.1.0`, vậy version mới nhất mà thỏa mãn điều kiện trên là version 1.0.2

Ngoài 2 kí hiệu đó ra, còn có 1 số kí hiệu khác, tuy nhiên không phổ biến bằng, có thể kể đến như:

`[version]`: giữ nguyên version hiện tại kể cả khi update

`>[version]`: version sau khi update phải > version hiện tại

`*[version]`: update đến version mới nhất không có ràng buộc gì.

`>= [version]`: version sau khi update phải >= version hiện tại

Bạn có thể xem thêm tại đây:

https://docs.npmjs.com/files/package.json

Chú ý là sau khi update, kí hiệu tiền tố sẽ trở về mặc định đó là kí tự ^. Cá nhân mình thì khuyên các bạn nên sử dụng kí tự tiền tố là ~ .

Gỡ cài đặt package

Cú pháp:

```
npm uninstall [tên_package]
```

Một điều nữa mà mình muốn nhắc tới đó là về folder node_module, folder đó chứa toàn bộ các files mà của các package bạn đã cài đặt, có thể có vài trăm thậm chí hang nghìn file, vì vậy bạn không nên share folder đó. Vậy nếu không share thì làm sao người khác lấy code bạn về có thể chạy được? Đơn giản chỉ cần 1 lệnh thôi, đó cũng chính là lệnh mà mình nhắc tới trong phần ví dụ về lợi ích của npm.

Đó là lệnh: `npm install`

Chỉ cần chạy lệnh này, npm sẽ tự động tìm và install tất cả các package được liệt kê ra trong file package.json, tất cả diễn ra 1 cách tự động và bạn chỉ việc ngồi chờ cho đến khi nó hoàn thành xong mà thôi. Rất tiện lợi, đúng không?

Một chú ý nho nhỏ trong phần này nữa là khi chạy lện npm install, npm sẽ tự động install tất cả các dependencies kể cả devdependencies, nếu bạn không muốn npm cài đặt devdependencies thì có thể sử dụng câu lệnh:

```
npm install --product
```