

# NPM

NodeJs

# NPM

Node **P**ackage **M**anager est un outil gérant les bibliothèques de programmation **Javascript** pour **Node.js**.



# NPM

## Npm init

Indispensable pour pouvoir utiliser npm avec votre projet .

Cette commande va générer un fichier [package.json](#) qui décrit la configuration de votre projet.

Chaque module de npm est configuré ainsi, ce qui fait que votre projet est par définition un paquet au même titre que les autres et donc potentiellement publiable sur npm .

```
PS C:\Cours\NodeJs\Exemples\NPMTEST> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.
```

```
See `npm help json` for definitive documentation on these fields
and exactly what they do.
```

```
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.
```

```
Press ^C at any time to quit.
package name: (npmtest) npmtest
version: (1.0.0) 1.0.0
description: Ceci est un test
entry point: (index.js) app.js
test command:
git repository: http://gitlab.com/mperson/npmtest.git
keywords: npm, command, samples
author: Mike Person
license: (ISC)
About to write to C:\Cours\NodeJs\Exemples\NPMTEST\package.json:
```

```
{
  "name": "npmtest",
  "version": "1.0.0",
  "description": "Ceci est un test",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "http://gitlab.com/mperson/npmtest.git"
  },
  "keywords": [
    "npm",
    "command",
    "samples"
  ],
  "author": "Mike Person",
  "license": "ISC"
}
```

```
Is this OK? (yes) █
```

# NPM

## npm -v

Permet d'obtenir la version installée

```
PS C:\Cours\NodeJs\Exemples\NPMTEST> npm -v  
6.13.4
```

## Npm install <nom du paquet>

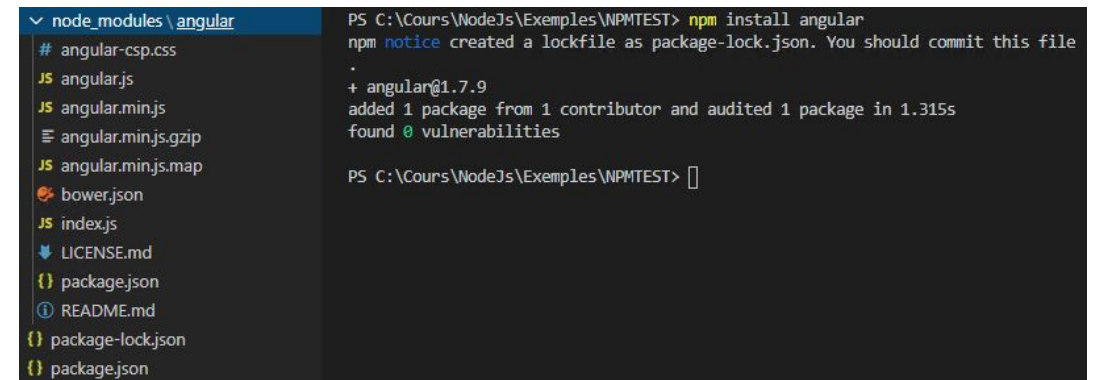
Cette commande permet l'installation d'un paquet et de ses dépendances.

Npm considère comme paquet valide :

- Un dossier contenant un fichier package.json
- Une archive gzip (contenant un dossier et le package.json associé )
- Une url vers une archive gzip
- Un paquet publié sur npm ( <nom\_du\_paquet>@<version>  
ou  
<nom\_du\_paquet>@<tag> ou <nom\_du\_paquet> )
- Une adresse git qui fournit une archive

## Npm uninstall <nom du paquet>

Cette commande permet de désinstaller un paquet



```
node_modules \angular  
# angular-csp.css  
JS angular.js  
JS angular.min.js  
angular.min.js.gzip  
JS angular.min.js.map  
bower.json  
JS index.js  
LICENSE.md  
package.json  
README.md  
package-lock.json  
package.json  
  
PS C:\Cours\NodeJs\Exemples\NPMTEST> npm install angular  
npm notice created a lockfile as package-lock.json. You should commit this file  
.  
+ angular@1.7.9  
added 1 package from 1 contributor and audited 1 package in 1.315s  
found 0 vulnerabilities  
  
PS C:\Cours\NodeJs\Exemples\NPMTEST>
```

# NPM

## Remarque :

Un paquet peut être installé globalement avec l'option -g ce qui permet une utilisation à travers la ligne de commande de n'importe où.

## **Attention à l'usage de l'installation globale !**

Si il peut sembler pratique d'avoir accès à des paquets de n'importe où sur le système sans devoir les réinstaller, ***la portabilité de votre module/programme ne sera plus assurée.***

# NPM

## Déclarer ses dépendances

NPM nous propose un gestionnaire de dépendance qui nous permet facilement de passer notre projet d'un environnement à un autre tout en gardant les dépendances à jour et de manière propre.

Npm distingue principalement deux types de dépendances renseignées sous forme d'objet dans le fichier package.json :

- Production
- Développement

```
"dependencies":  
{  
  "angular": "1.7.9"  
},  
"devDependencies":  
{  
  "angular": "1.7.9",  
  "webpack-dev-server": "3.10.3"  
}
```

Grâce à ce fichier, lors de la distribution de notre projet, un simple **npm install** et les dépendances s'installeront.

```
PS C:\Cours\NodeJs\Exemples\NPMTEST> npm install  
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#de  
precated  
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#depre  
cated  
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.2.7 (node_modules\c  
hokidar\node_modules\fsevents):  
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents
```

# NPM

Pour mettre à jour ses dépendances :

- `npm update <nom du paquet>` ☐ Mets à jour le paquet cité
- `npm update` ☐ Mets à jour toutes les dépendances

# NPM

## Semantic Versioning (SemVer)

L'écosystème nodejs utilise majoritairement la méthode **SemVer** pour numéroter les versions des paquets.

4 . 2 . 0

Major Minor Patch

Npm introduit un certain nombre de patterns permettant de gérer plus finement les versions qu'avec une numérotation exacte :

- Les opérateurs `<` `<=` `>` `>=` `=` peuvent être combinés avec `||` ou un espace pour former une expression définissant un ensemble de versions

`1.2.7 || >=1.2.9 <2.0.0` correspond aux versions 1.2.7 , 1.2.9 et 1.4.6 mais pas 1.2.8 ni 2.0.0

- `*` ( ou `X` ou `x` ) prendra toujours la dernière version disponible

`1*` équivaut à la dernière version de la branche 1 soit `>=1.0.0 <2.0.0`

- `-` détermine un intervalle inclusif

`1.2.3 - 2.3.4` équivaut à `>=1.2.3 <=2.3.4`



# NPM

- Les opérateurs ~ ^

- ~ inclus la version mineure la plus récente mais sans changer de branche

`~1.2.3 équivaut à >=1.2.3 <1.3.0`

- ^ inclus la version majeur la plus récente

`^1.2.3 équivaut à >=1.2.3 <2.0.0`

Il existe un site permettant de s'y retrouver lors de la recherche d'une version pour un package particulier : <https://semver.npmjs.com/>

Ce site mettra en surbrillances les versions incluent suite aux wildcard utilisés

Pour aller plus loin : <https://docs.npmjs.com/misc/semver> et <https://maxlab.fr/javascript/comprendre-npm-astuces-et-configuration/>