eBook Gratuit

APPRENEZ npm

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#npm

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: npm

It is an unofficial and free npm ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official npm.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec npm

Remarques

npm est le gestionnaire de paquets par défaut pour Node.js. Il est écrit entièrement en JavaScript et permet d'installer et de distribuer principalement des modules JavaScript sur le registre. Les packages sont traités au format CommonJS et sont reconnaissables à la présence d'un fichier `package.json` .

Versions

Version	Date de sortie
v4.0.0	2016-10-21
v3.10.0	2016-06-17
v3.9.0	2016-05-06
v2.15.0	2016-03-11
v3.8.0	2016-02-26
v3.7.0	2016-01-29
v3.6.0	2016-01-21
v3.5.0	2015-11-20
v3.4.0	2015-11-06
v3.3.0	2015-08-14
v2.14.0	2015-08-14
v3.2.0	2015-07-25
v3.1.0	2015-07-03
v2.13.0	2015-07-03
v3.0.0	2015-06-26
v2.12.0	2015-06-19
v2.11.0	2015-05-22
v2.10.0	2015-05-08

Version	Date de sortie
v2.9.0	2015-04-24
v2.8.0	2015-04-10
v2.7.0	2015-02-27
v2.6.0	2015-02-13
v2.5.0	2015-01-30
v2.4.0	2015-01-23
v2.3.0	2015-01-16
v2.2.0	2015-01-09
v2.1.0	2014-09-26
v2.0.0	2014-09-13
v1.4.0	2014-02-13

Examples

Installation ou configuration

Installer

`npm` est fourni avec Node.js , donc si vous installez Node.js, vous aurez automatiquement installé `npm` . Vous pouvez choisir entre une version **actuelle** et une version **LTS**

les fenêtres

Pour Microsoft Windows, vous pouvez télécharger un **programme** d' **installation MSI** à l' adresse https://nodejs.org/en/download/ .

OS X

Pour Apple OS X, vous pouvez télécharger un **programme** d' **installation PKG** au même endroit https://nodejs.org/en/download/ .

Linux

Pour Linux, vous pouvez utiliser votre gestionnaire de paquets pour installer Node.js et npm.

Vous pouvez également compiler Node.js depuis le source et vous obtiendrez toujours `npm` . Il y a

aussi un script que vous pouvez exécuter pour installer `npm` :

```
curl -L https://www.npmjs.com/install.sh | sh
```

Mettre à niveau npm vers la dernière version

La méthode recommandée pour mettre à jour votre installation `npm` est de simplement installer `npm` :

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

Vous pouvez également mettre à niveau vers la version LTS actuelle plutôt que la version la plus récente:

```
npm install -g npm@lts
```

Vous pouvez également installer n'importe quelle version de Node (et npm) avec `nvm` . Lorsque vous installez globalement avec `npm` avec une installation `nvm` , vous n'avez pas besoin d'utiliser `sudo` (ou Exécuter en tant qu'administrateur sous Windows).

Installer des paquets

Notez que les packages peuvent être installésCette commande installe la version la plus récente des packages nommés:

à la fois localement ou globalement.

L'installation locale signifie que **npm** installe votre paquet dans le répertoire de travail en cours. Les modules de nœuds vont dans `./node_modules` , les exécutables vont dans `./node_modules/.bin/` . En général, vous souhaiterez installer des modules locaux pour une utilisation dans votre programme, en tant que dépendance, et ils fonctionneront uniquement là où ils sont installés.

```
npm install <package names>
```

Sténographie:

```
npm i <package names>
```

`npm` peut interagir avec un fichier `package.json` dans le répertoire en cours de diverses manières utiles, via les `dependencies` objets et les `devDependencies` stockées dans `package.json` (installation de plusieurs modules):

La commande d' `npm install` sans paramètres

```
npm install
```

installe tous les paquets nommés en tant que clés d'objet dans les objets `dependencies` et

`devDependencies` de `package.json` , en utilisant les restrictions de `devDependencies` version sémantique indiquées par les valeurs d'objet.

Lors du développement de nouveaux logiciels:

Utilisez l'option `-S` pour ajouter les `<package names>` et les versions des modules npm que vous installez, qui doivent toujours être inclus avec votre module. Ajoute à la liste des `dependencies` suivies dans le fichier `package.json` , après l'installation.

```
npm i <package names> -S
```

Utilisez l'option `-D` pour ajouter les `<package names>` et les versions des modules npm que vous installez et que d'autres développeurs ont besoin pour développer ou tester votre module. Ajoute à la liste des `devDependencies` suivis dans le fichier `package.json` , après l'installation.

```
npm i <package names> -D
```

Où `lodash` et `mocha` sont des noms de paquets.

Installation de packages globaux

Installer un paquet global

Les paquets installés globalement suppriment les modules dans `{prefix}/lib/node_modules` les fichiers exécutables dans `{prefix}/bin` , où `{prefix}` est généralement quelque chose comme `/usr/local` . L'installation d'un module global signifie que ses fichiers binaires se retrouvent dans votre variable d'environnement `PATH` . Habituellement, vous voudrez installer un module global s'il s'agit d'un outil de ligne de commande ou de quelque chose que vous souhaitez utiliser dans votre shell.

```
npm install --global package-name
```

Supprimer un package global

```
npm uninstall --global package-name
```

Note: l'argument `--global` peut être simplifié à `-g` . Ainsi, par exemple, la première commande pourrait être `npm install -g package-name` , avec exactement le même résultat.

Remarque: dans les systèmes * nix, l'installation de packages globaux peut nécessiter des autorisations de superutilisateur. Ne pas le faire échouera avec: `EACCES` . Dans ce cas, lancez:

```
sudo npm install --global package-name
```

Mise à jour des paquets

Dans chaque application, le cycle de vie survient un jour où ses composants doivent être mis à

jour. Tout le monde connaît la peine de mettre à jour chaque dépendance individuellement. Eh bien, il vous suffit de lancer la commande:

```
npm update (-g)
```

Si le "-g" est là, alors npm mettra à jour les paquets globaux.

Utiliser npm pour gérer les dépendances

Vous souhaitez donc déployer votre application sur plusieurs sites? et votre projet a trop de dépendances pour les installer un par un? Npm a une solution, lancez simplement la commande suivante:

```
npm init
```

Dans le dossier racine du projet, suivez les instructions à l'écran (saisissez la valeur souhaitée puis appuyez sur Entrée), puis, si vous souhaitez enregistrer une dépendance, ajoutez:

```
--save
```

après ton

```
npm install
```

commandes par exemple:

```
npm install mypackagename --save
```

Et puis cette dépendance sera sauvegardée alors vous n'avez pas à déplacer le dossier "node_modules". Pour installer tous les problèmes de dépendance enregistrés:

```
npm install
```

et toutes les dépendances enregistrées seront installées.

Lire Commencer avec npm en ligne: https://riptutorial.com/fr/npm/topic/2061/commencer-avec-npm

Chapitre 2: Édition

Examples

Publier le référentiel public

Pour publier des référentiels publics avec un compte npm gratuit, vous devez d'abord publier le module avec accès public. Une façon de le faire est de définir la configuration pour `npm` à lire dans `packages.json` comme suit:

```
"publishConfig": {
  "access": "public"
},
```

Vous pouvez également utiliser l'indicateur `--access=public` avec la commande `npm publish` . Sinon, vous obtiendrez le message d'erreur légèrement confus, "vous avez besoin d'un compte payant pour effectuer cette action".

```
npm publish // if you modified packages.json
```

ou

```
npm publish --access=public
```

Signe Git Tag

Pour que la signature fonctionne, vous devez configurer une clé GPG par défaut. Vous pouvez l'activer ou le désactiver comme suit:

```
npm config set sign-git-tag <true or false>
```

Lire Édition en ligne: https://riptutorial.com/fr/npm/topic/3025/edition

Chapitre 3: Qu'est-ce que NPM fait?

Introduction

Il vous permet de gérer facilement différents packages (modules) et de garder une trace de la version que vous avez installée.

Examples

Initialiser un nouveau projet NPM

```
npm init
```

Cela initialisera un nouveau projet NPM pour vous - appuyez simplement sur Entrée jusqu'à ce qu'il ne vous pose plus de questions.

Maintenant, vous remarquerez que vous avez un nouveau fichier appelé *package.json* . Ce fichier conservera, entre autres, le suivi des modules ou modules que vous avez installés dans votre projet.

Lire Qu'est-ce que NPM fait? en ligne: https://riptutorial.com/fr/npm/topic/10586/qu-est-ce-que-npm-fait-

Chapitre 4: scripts npm

Syntaxe

- La propriété "scripts" de `package.json` vous permet d'exécuter des packages npm localement.
- Le script `"karma": "karma"` référence au script shell `karma` le `node_modules/.bin` . Cette référence doit être saisie et un alias doit lui être appliqué pour pouvoir être utilisé dans d'autres scripts npm, tels que `"test": "karma start"` .

Remarques

Scripts pré-reconnus

- `prepublish` : Exécuter avant la publication du paquet
- `publish` , `postpublish` : Exécuter après la publication du package
- `preinstall` : Exécuter avant l'installation du paquet
- `install` , `postinstall` : Exécuter après l'installation du package
- `preversion` , `version` : Exécuter avant de cogner la version du package
- `postversion` : Exécuter après bump la version du package
- `pretest` , `test` , `posttest test` : `posttest` par la commande `npm test`
- `prestop` , `stop` , `poststop` : `poststop` par la commande `npm stop`
- `prestart` , `start` , `poststart` : `poststart` par la commande `npm start`
- `prerestart` , `restart` , `postrestart` : Exécuter avec la commande `npm restart` . Remarque: `npm restart` exécute les scripts d'arrêt et de démarrage si aucun script de `restart` n'est fourni.

On peut en déduire que la propriété `"scripts"` de `package.json` est un outil très puissant. Il peut être utilisé comme un outil de construction, similaire à Grunt et Gulp, mais avec plus de 250 000 paquets disponibles. Les scripts NPM exécutent les packages npm installés localement sur votre projet à partir du `node_modules/.bin` .

Examples

Lancer le karma localement

extrait de `package.json`

```
{
    "scripts":
        "test": "karma start",
        "karma": "karma"
    }
}
```

Exécution de scripts npm

Il existe deux types de scripts npm, et la commande à exécuter est légèrement différente. Les premiers types de scripts npm sont des scripts "pré-reconnus". Ces scripts sont automatiquement reconnus par npm et n'ont pas besoin d'un préfixe spécial (comme vous le verrez pour l'autre type) pour les exécuter. Les autres types de scripts sont des scripts "personnalisés". Ces scripts ne sont pas pré-reconnus par npm et doivent être préfixés par une commande spéciale pour les exécuter. Il y a une liste de scripts pré-reconnus dans la section des remarques.

Pour exécuter des scripts pré-reconnus:

`npm start` ou `npm test`

Pour exécuter des scripts personnalisés, vous devez utiliser la commande `run` :

`npm run karma`

Quels sont les scripts npm et comment sont-ils déclenchés?

Les scripts npm sont des commandes que `npm` exécutera pour vous lorsqu'il est appelé avec les arguments appropriés. La puissance et le sens de ceci est de ne PAS installer les paquetages npm en globalisant votre environnement.

La différence entre les scripts pré-reconnus et les scripts personnalisés repose sur le mot d' `run` entre les balises, **`custom` scripts `custom` nécessiteront une `run` entre npm et le nom du script.**

Sur cette base, nous pouvons différencier et créer différentes tâches ou scripts à exécuter avec npm.

Étant donné l'exemple suivant sur le fichier `package.json` :

```
{
  "name": "MyApp",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "mocha --recursive ./tests/",
    "test:watch": "npm run test -- -w",
    "start": "nodemon --inspect ./app.js",
    "build": "rm -rf ./dist/ && gulp build"
  }
...
}
```

Nous pouvons voir différentes tâches à exécuter:

- `npm test` Cela fonctionnerait bien car c'est un script pré-reconnu
- `npm run test` Cela fonctionnerait bien car c'est un moyen valide d'exécuter un script npm
- `npm run test:watch` Fonctionne également, et appelle le test npm run à l'intérieur de lui-même
- `npm run build` Est-ce qu'avant d'exécuter `gulp build` supprimez le dossier `dist` qui se trouve

dans le répertoire (en supposant que vous soyez sous Linux ou que la commande `rm` soit reconnue)

Lire scripts npm en ligne: https://riptutorial.com/fr/npm/topic/4842/scripts-npm

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec npm	acdcjunior, Ana, Aurora0001, Cezar Augusto, Community, eppsilon, GilZ, Kenan, Konstantin A. Magg, KriszDev, Paul, Protectator, RationalDev, Tim Anstee
2	Édition	Pier Paolo Ramon, RationalDev
3	Qu'est-ce que NPM fait?	Kuhan
4	scripts npm	Alejandro Vales, DankestMemes