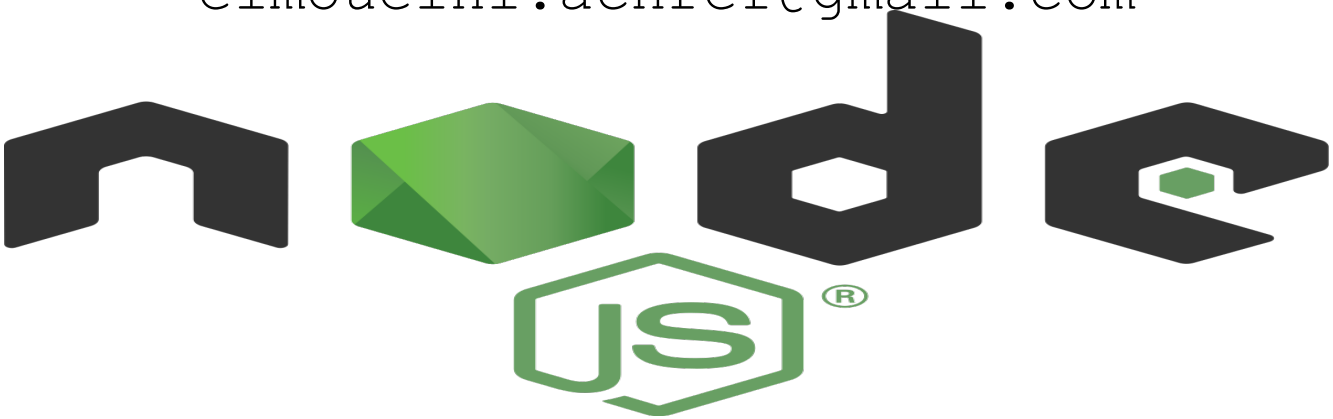
Node.js : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Plan

Plan

Node.js

On écrit

- **Node.js** et **node.js**
- ou aussi **NodeJS**
- ou encore **Node** et **node**

Node.js

Node.js

- Plateforme de développement (ou environnement d'exécution) : ensemble de librairies **JavaScript**
- Créé par **Ryan Lienhart Dahl** en 2010
- Permettant l'exécution de **JavaScript** côté serveur et de réaliser des tâches comme :
  - Persistance de données
  - Manipulation de fichiers
  - Sécurité
  - ...

Node.js

Node.js : histoire

- **2009** : Création de **Node.js** par **Ryan Dahl**, un développeur, en réponse aux limitations des modèles d'E/S bloquantes dans les applications web traditionnelles.
- **2010** : Première version stable de **Node.js** (v0.1.14)
- **2011** : Première version de **npm** (**N**ode **P**ackage **M**anager) simplifiant la gestion des dépendances dans les projets **Node.js**.
- **2012** : **Node.js** est utilisé par de grandes entreprises, notamment **PayPal** et **LinkedIn**.
- ...
- **2018** : Création de **Deno** par **Ryan Dahl** en réponse à certaines lacunes perçues de **Node.js**. **Deno** a été annoncé lors d'une présentation intitulée **10 Things I Regret About Node.js**
- ...

Node.js

Node.js : panorama

- **Utilisations courantes** :
  - développement de serveurs web et d'applications côté serveur, notamment des **API REST**,
  - des applications en temps réel (comme les chats en direct et les jeux en ligne),
  - des serveurs de fichiers,
  - des microservices.
- **Écosystème riche** : **Node.js** dispose d'une vaste bibliothèque de modules et de packages, gérés via **npm**.
- **Développement côté serveur** : Node.js a introduit un modèle asynchrone non bloquant qui permet de gérer efficacement de nombreuses connexions simultanées.
- **Framework et outils** : Plusieurs frameworks et outils ont été développés autour de Node.js pour simplifier le développement, notamment **Express.js**, **Nest.js**, **Ionic**, **Electron** (pour le développement d'applications de bureau multiplateformes).
- **Adoption par les grandes entreprises** : **Netflix**, **Airbnb**, **Uber** utilisent Node.js pour des applications critiques, profitant de ses performances et de sa flexibilité.
- ...

Node.js

Node.js : avantages

- **Performances élevées** : grâce à son modèle non bloquant et à sa gestion efficace des E/S
- **Évolutivité** : grâce à sa gestion efficace des connexions simultanées, Node.js est la plateforme idéale pour les applications en temps réel, les chats en direct, les jeux en ligne...
- **Écosystème riche et communauté active** : une vaste bibliothèque de modules et de packages tiers disponibles via **npm**
- **Utilisation du même langage côté client et côté serveur** : **JavaScript**
- ...

Node.js

Node.js : inconvénients

- **Complexité du code** : le modèle de callbacks peut conduire à une structure de code complexe et à des enchevêtrements de callbacks (callback hell).
- **Gestion des erreurs** : la gestion des erreurs peut être compliquée dans **Node.js** en raison de la nature asynchrone des opérations.
- **Monoprocessus** : par défaut, **Node.js** s'exécute dans un seul processus, ce qui signifie qu'il ne peut pas tirer pleinement parti des processeurs multi-cœurs sans l'utilisation de modules complémentaires.
- ...

Node.js

Remarque

- Pour vérifier la compatibilité de **Node.js** avec une version ES20** : aller à `https://node.green/#ES2015`.
- Pas besoin d'utiliser un traducteur comme **Babel**.

Node.js

Pour **Node.js**, il faut

- aller à `https://nodejs.org/en/`
- choisir la dernière version, télécharger et installer

Node.js

Pour **Node.js**, il faut

- aller à `https://nodejs.org/en/`
- choisir la dernière version, télécharger et installer

Pour vérifier la version installée

```
node --version
```

Node.js

Pour **Node.js**, il faut

- aller à `https://nodejs.org/en/`
- choisir la dernière version, télécharger et installer

Pour vérifier la version installée

```
node --version
```

Ou

```
node -v
```

Node.js

Commandes installées avec **Node.js**

- `node` : interpréteur **JavaScript** en ligne de commande.
- `npm` : gestionnaire de paquets officiel de **Node.js**.
- `npx` : exécute des paquets **Node.js** sans les installer globalement.

Node.js

Quel IDE (Environnement de développement intégré) pour **Node.js** ?

- **Visual Studio Code** (À ne pas confondre avec Visual Studio)
- Visual Studio
- Eclipse
- ...

Node.js

Visual Studio Code (ou VSC) , pourquoi ?

- Gratuit
- Multi-langage
- Multi-système d'exploitation
- Extensible via l'installation de quelques centaines d'extensions

Node.js

Visual Studio Code (ou VSC) , pourquoi ?

- Gratuit
- Multi-langage
- Multi-système d'exploitation
- Extensible via l'installation de quelques centaines d'extensions

VSC : téléchargement

`code.visualstudio.com/download`

Node.js

Quelques raccourcis **VSC**

- Pour activer la sauvegarde automatique : aller dans `File > AutoSave`
- Pour indenter son code : `Alt` `Shift` `f`
- Pour commenter/décommenter : `Ctrl` `:`
- Pour sélectionner toutes les occurrences : `Ctrl` `f2`
- Pour sélectionner l'occurrence suivante : `Ctrl` `d`
- Pour placer le curseur dans plusieurs endroits différents : `Alt`

Node.js

Démarche

- Créez un répertoire `cours-node` dans votre espace de travail
- Lancez **VSC** et allez dans `File > Open Folder...` et choisissez `cours-node`
- Dans `cours-node`, créez un fichier `main.js`

Node.js

Démarche

- Créez un répertoire `cours-node` dans votre espace de travail
- Lancez **VSC** et allez dans `File > Open Folder...` et choisissez `cours-node`
- Dans `cours-node`, créez un fichier `main.js`

Ajoutons le contenu suivant à `main.js`

```
Console.log("Hello world");
```

Node.js

Démarche

- Créez un répertoire `cours-node` dans votre espace de travail
- Lancez **VSC** et allez dans `File > Open Folder...` et choisissez `cours-node`
- Dans `cours-node`, créez un fichier `main.js`

Ajoutons le contenu suivant à `main.js`

```
Console.log("Hello world");
```

Pour exécuter, lancez la commande

```
node main.js
```

Node.js

npm : Node Package Manager

- Gestionnaire officiel de paquets pour **Node.js** (modules de la communauté)
- Utilisé aussi dans développement le front-end
- Lancé en 2010
- Créé par **Isaac Z. Schlueter**
- Installé automatiquement avec **Node.js**
- Documentation officielle : `www.npmjs.com`

Node.js

Autres gestionnaires de paquet pour **Node.js**

- **Yarn**
  - Lancé par **Facebook** en 2016
  - Plus rapide que les versions antérieures de **npm** mais l'écart a été réduit
  - Utilisant principalement le repository **npm**
- **Bower**
  - Créé en 2012 pour les projets Front-end
  - Ayant son propre repository
  - En 2017, l'équipe **Bower** a recommandé de migrer vers **Yarn**
- ...

Node.js

Pour vérifier la version installée de `npm`

```
npm --version
```

Node.js

Pour vérifier la version installée de `npm`

```
npm --version
```

Ou

```
npm -v
```

Node.js

Pour configurer (initialiser) un nouveau projet ou un projet existant

```
npm init
```

Ensuite

- Répondre aux différentes questions
- Vérifier la génération d'un fichier `package.json`

Node.js

Exemple du contenu de `package.json`

```
{
  "name": "cours-node",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Node.js

Pour initialiser un nouveau projet ou un projet existant sans répondre aux questions

```
npm init --yes
```

Node.js

Pour initialiser un nouveau projet ou un projet existant sans répondre aux questions

```
npm init --yes
```

Ou

```
npm init -y
```

Node.js

Deux types d'installation

- Installation globale d'un package ⇒
  - package ajouté dans `C:\Users\admin\AppData\Roaming\npm\node_modules`
  - package pouvant être utilisé dans tout autre projet **Node.js**.
- Installation locale d'un package ⇒
  - package ajouté dans `node_modules` du projet
  - package pouvant seulement être utilisé dans le projet actuel.
  - possibilité de déclarer le package installé dans `package.json`

Node.js

Pour installer globalement un package

```
npm install --global nom-package
```

Node.js

Pour installer globalement un package

```
npm install --global nom-package
```

Ou

```
npm install -g nom-package
```

Node.js

Avant `npm 5`, pour installer localement un package et le déclarer dans la section `dependencies` de `package.json`

```
npm install --save nom-package
```

Node.js

Avant `npm 5`, pour installer localement un package et le déclarer dans la section `dependencies` de `package.json`

```
npm install --save nom-package
```

Ou

```
npm install -S nom-package
```

Node.js

Avant `npm 5`, pour installer localement un package et le déclarer dans la section `dependencies` de `package.json`

```
npm install --save nom-package
```

Ou

```
npm install -S nom-package
```


Avant `npm 5`, sans les options `--save` et `-S`, le package installé ne sera pas déclaré dans `package.json`

```
npm install nom-package
```

Node.js

Depuis `npm 5`, pour installer localement un package et le déclarer dans la section `dependencies` de `package.json`

```
npm install --save-prod nom-package
```

Node.js

Depuis `npm 5`, pour installer localement un package et le déclarer dans la section `dependencies` de `package.json`

```
npm install --save-prod nom-package
```

Ou

```
npm install -P nom-package
```

Node.js

Depuis `npm 5`, pour installer localement un package et le déclarer dans la section `dependencies` de `package.json`

```
npm install --save-prod nom-package
```

Ou

```
npm install -P nom-package
```

Ou tout simplement

```
npm install nom-package
```

Node.js

A l'installation d'une première dépendance locale

- un `package-lock.json` sera généré
- un dossier `node_modules` sera ajouté pour stocker les dépendances

Node.js

A l'installation d'une première dépendance locale

- un `package-lock.json` sera généré
- un dossier `node_modules` sera ajouté pour stocker les dépendances

`package-lock.json`

- Introduit depuis **npm 5**
- Contenant l'arborescence exacte des dépendances installées : Si on installe la version 2 d'un package **X** qui dépend de la version 5 d'un package **Y**, alors **X** et **Y** seront tous les deux mentionnés dans `package-lock.json` ainsi que leur version.

Node.js

Pour installer localement un package sans le déclarer dans `package.josn`

```
npm install --no-save nom-package
```

Node.js

Pour installer localement un package sans le déclarer dans `package.josn`

```
npm install --no-save nom-package
```

Pour installer localement un package et le déclarer dans `devDependencies` **dans** `package.json`

```
npm install --save-dev nom-package
```

Node.js

Pour installer localement un package sans le déclarer dans `package.josn`

```
npm install --no-save nom-package
```

Pour installer localement un package et le déclarer dans `devDependencies` **dans** `package.json`

```
npm install --save-dev nom-package
```

Pour installer localement un package et le déclarer dans `devDependencies` **dans** `package.json`

```
npm install -D nom-package
```

Node.js

Remarque

La commande `install` a plusieurs alias autorisés : `add`, `i`, `in`, `ins`, `inst`, `insta`, `instal`, `isnt`, `isnta`, `isntal`, `isntall`.

Node.js

Pour installer plusieurs packages

```
npm install nom-package1 nom-package2 nom-package3
```

Node.js

Pour installer les dépendances listées dans les sections `dependencies` **et** `devDependencies` **du** `package.json`

```
npm install
```

Node.js

Pour désinstaller un package (et le supprimer de `package.json`)

```
npm uninstall nom-package
```

Node.js

Pour désinstaller un package (et le supprimer de `package.json`)

```
npm uninstall nom-package
```

Pour désinstaller un package (sans le supprimer de `package.json`)

```
npm uninstall nom-package --no-save
```

Node.js

Pour désinstaller un package (et le supprimer de `package.json`)

```
npm uninstall nom-package
```

Pour désinstaller un package (sans le supprimer de `package.json`)

```
npm uninstall nom-package --no-save
```

Pour désinstaller un package global

```
npm uninstall nom-package -g
```

Node.js

Comportement de `npm install A`

- Installe le package `A`.
- Installe aussi ses dépendances (ex. : `B`, `C`...).
- Ajoute `A` dans `package.json`.
- Enregistre `A`, `B`, `C`... dans `package-lock.json`.

Node.js

Comportement de `npm uninstall A` :

- Supprime `A` de `node_modules` et de `package.json`.
- Supprime aussi les dépendances de `A` (`B`, `C`...) **si elles ne sont utilisées nulle part ailleurs**.
- Sinon, les dépendances partagées sont conservées.

Node.js

Comportement de `npm uninstall A` :

- Supprime `A` de `node_modules` et de `package.json`.
- Supprime aussi les dépendances de `A` (`B`, `C`...) **si elles ne sont utilisées nulle part ailleurs**.
- Sinon, les dépendances partagées sont conservées.

Conseil pratique

`npm prune` permet de supprimer manuellement les dépendances non utilisées.

Node.js

Pour mettre à jour un package local

```
npm update nom-package
```

Node.js

Pour mettre à jour un package local

```
npm update nom-package
```

Pour mettre à jour tous les packages locaux (listés dans `package.json`

```
npm update
```

Node.js

Pour mettre à jour un package global

```
npm update nom-package -g
```

Node.js

Pour mettre à jour un package global

```
npm update nom-package -g
```

Pour mettre à jour tous les packages globaux

```
npm update -g
```

Node.js

Versions d'un module

- Tout projet (ou package) **Node.js** peut évoluer
- Et on marque cette évolution par des numéros de version souvent écrits : $x.y.z$
  - z : numéro de la correction (patch)
  - y : version mineure
  - x : version majeure

Node.js

Considérons le contenu suivant de la section `dependencies` de `package.json`

```
"dependencies": {
    "mongoose": "^4.6.7",
    "mysql": "~2.3.0"
}
```

Node.js

Considérons le contenu suivant de la section `dependencies` de `package.json`

```
"dependencies": {
    "mongoose": "^4.6.7",
    "mysql": "~2.3.0"
}
```

Explication

- `"^4.6.7"` : signifie que la version doit être supérieure ou égale à 4.6.7, mais inférieure à 5.0.0. Par exemple, cela autorise l'installation des versions 4.6.7, 4.7.0, 4.13.21, mais pas 5.0.0.
- `"~2.3.0"` : signifie que la version doit être supérieure ou égale à 2.3.0, mais inférieure à 2.4.0. Par exemple, cela autorise l'installation des versions 2.3.0, 2.3.1, 2.3.2, mais pas 2.4.0.

Node.js

Lancer la commande suivante

```
npm install
```

Node.js

Constats

- Aller dans `node_modules/mongoose/package.json` et vérifier que `4.13.21` est la version installée.
- Aller dans `node_modules/mysql/package.json` et vérifier que `2.3.2` est la version installée.

Node.js

Constats

- Aller dans `node_modules/mongoose/package.json` et vérifier que `4.13.21` est la version installée.
- Aller dans `node_modules/mysql/package.json` et vérifier que `2.3.2` est la version installée.

Explication

- `4.13.21` est la dernière version avant `5.0.0`
- `2.3.2` est la dernière version avant `2.4.0`

Node.js

Pour lister les modules qui ne sont pas à jour

```
npm outdated
```

Node.js

Pour lister les modules qui ne sont pas à jour

```
npm outdated
```

Résultat

```
Package   Current   Wanted  Latest  Location               Depended by
mongoose  4.13.21  4.13.21   8.0.1  node_modules/mongoose  cours-node
mysql       2.3.2    2.3.2  2.18.1  node_modules/mysql     cours-node
```

Node.js

Pour chercher un module selon un mot-clé

```
npm search mot-clé
```

Node.js

Pour chercher un module selon un mot-clé

```
npm search mot-clé
```


Remarque

On peut aussi chercher sur `https://www.npmjs.com/`.

Node.js

Pour lister les modules locaux

```
npm list
```

Node.js

Pour lister les modules locaux

```
npm list
```

Ou son alias

```
npm ls
```

Node.js

Pour lister les modules locaux

```
npm list
```

Ou son alias

```
npm ls
```


Pour lister les dépendances locales avec les sous dépendances du premier niveau

```
npm ls --depth=1
```

Node.js

Pour lister les dépendances globales

```
npm ls -g
```

Node.js

Considérons le contenu suivant de `index.js`

```
console.log("Hello from index.js");
```

Node.js

Considérons le contenu suivant de `index.js`

```
console.log("Hello from index.js");
```

Et le contenu de `public/script.js`

```
console.log("Hello from script.js");
```

Node.js

Considérons le contenu suivant de `index.js`

```
console.log("Hello from index.js");
```

Et le contenu de `public/script.js`

```
console.log("Hello from script.js");
```


Remarque

Pour associer une commande `npm` à l'exécution d'un script, il faut déclarer cette dernière dans la section `scripts` de `package.json`

Node.js

Dans `package.json`, déclarons les deux commandes dans la section `scripts`

```
"scripts": {
    "start": "node index.js",
    "serve": "node public/script.js"
}
```

Node.js

Dans `package.json`, déclarons les deux commandes dans la section `scripts`

```
"scripts": {
    "start": "node index.js",
    "serve": "node public/script.js"
}
```


Pour exécuter la commande associée à la section `serve`

```
npm run serve
```

Node.js

Pour exécuter la commande associée à la section `start`

```
npm run start
```

Node.js

Pour exécuter la commande associée à la section `start`

```
npm run start
```

Pour la section `start`, on peut tout simplement exécuter

```
npm start
```

Node.js

Pour exécuter la commande associée à la section `start`

```
npm run start
```

Pour la section `start`, on peut tout simplement exécuter

```
npm start
```


Pour exécuter la commande définie dans la section `stop` (définie dans `scripts`) de `package.json`

```
npm stop
```

Node.js

Pour obtenir une liste complète des commandes principales

```
npm help
```

Node.js

Pour obtenir une liste complète des commandes principales

```
npm help
```

Pour avoir de l'aide sur une commande

```
npm help nom-commande
```

Node.js

Pour obtenir une liste complète des commandes principales

```
npm help
```

Pour avoir de l'aide sur une commande

```
npm help nom-commande
```

Pour avoir de l'aide sur un package

```
npm view nom-package
```

Node.js

`npm ci` (**C**lean **I**nstall)

- Installe les dépendances à partir de `package-lock.json` uniquement
- Supprime `node_modules/` avant installation
- Ne modifie jamais les fichiers `package.json` ou `package-lock.json`
- Plus rapide et fiable que `npm install` pour les environnements **CI** (**C**ontinuous **I**ntegration)

Node.js

`npm audit` (audit de sécurité)

- Analyse les dépendances pour détecter les vulnérabilités connues
- `npm audit fix` applique automatiquement les correctifs disponibles
- `npm audit fix --force` met à jour même avec des changements majeurs (à utiliser avec prudence)

Node.js

Quelques commandes `npm` et leurs alias

Commande Complète	Alias	Description
`npm install`	`npm i`	Installe les dépendances du `package.json`
`npm install paquet`	`npm i paquet`	Installe un paquet localement
`npm install paquet --save`	`npm i paquet -S`	(Obsolète) Ajoute le paquet aux dépendances
`npm install paquet --save-dev`	`npm i paquet -D`	Ajoute le paquet aux dépendances de développement
`npm uninstall paquet`	`npm remove paquet`	Supprime un paquet installé
`npm uninstall paquet`	`npm rm paquet`	Supprime un paquet installé (forme abrégée)
`npm list`	`npm ls`	Liste les paquets installés

Node.js

Après installation de certains paquets (globalement ou localement) :

- `nodemon` : redémarre l'application automatiquement à chaque changement.
- `ts-node` : exécute du **TypeScript** directement.
- `tsc` : transpile des fichiers **TypeScript** en **JavaScript**.
- `eslint` : analyse statique du code **TypeScript**/**JavaScript**.
- `vite`, `ng`, `next`... : exécutables liés à des frameworks.

Node.js

nvm : Node Version Manager

- Gestionnaire de versions pour **Node.js** : permettant de travailler avec plusieurs versions de **Node.js**
- Disponible pour **Windows**, **Mac OS** et **Linux**
- Conçu initialement pour **Mac OS** et **Linux** ensuite pour **Windows**
- Deux dépôts **GitHub** différents :
  - Pour **Mac OS** et **Linux** : `https://github.com/nvm-sh/nvm`
  - Pour **Windows** : `https://github.com/coreybutler/nvm-windows`

Node.js

Autres gestionnaires de versions pour **Node.js**

- **nvs** (**N**ode **V**ersion **S**witcher) : `https://github.com/jasongin/nvs`
- **Volta** : `https://github.com/volta-cli/volta#installing-volta`
- ...

Node.js

Installation sous **Windows**

- Aller à `https://github.com/coreybutler/nvm-windows/releases`
- Choisir un format, télécharger puis installer

Node.js

Installation sous **Mac OS** et **Linux** : 3 solutions

- Soit en téléchargeant et exécutant le script **Shell** du lien suivant : `https://github.com/nvm-sh/nvm/blob/v0.39.3/install.sh`
- Soit en exécutant la commande `curl`
- Soit en exécutant la commande `wget`

Node.js

Deuxième solution avec tt curl

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

Node.js

Deuxième solution avec tt curl

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

Troisième solution avec `wget`

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

Node.js

Deuxième solution avec tt curl

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

Troisième solution avec `wget`

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```


Quel que soit votre OS

Redémarrer le terminal (console)

Node.js

Pour connaitre la version installée de `nvm`

```
nvm --version
```

Node.js

Pour connaitre la version installée de `nvm`

```
nvm --version
```

Ou

```
nvm -v
```

Node.js

Remarque

Installer `nvm` n'installe pas automatiquement une version de **Node.js**.

Node.js

Pour installer la dernière version (lts) de Node.js (sous Windows)

```
nvm install lts
```

Node.js

Pour installer la dernière version (lts) de Node.js (sous Windows)

```
nvm install lts
```

Pour installer la dernière version (lts) de Node.js (sous Mac et Linux)

```
nvm install --lts
```

Node.js

Pour installer la dernière version (lts) de Node.js (sous Windows)

```
nvm install lts
```

Pour installer la dernière version (lts) de Node.js (sous Mac et Linux)

```
nvm install --lts
```


Pour installer une version spécifique

```
nvm install 14.7.0
```

Node.js

Pour désinstaller une version spécifique

```
nvm uninstall 14.7.0
```

Node.js

Pour lister les versions installer de Node.js

```
nvm list
```

Node.js

Pour lister les versions installer de Node.js

```
nvm list
```

Ou

```
nvm ls
```

Node.js

Pour lister les versions installer de Node.js

```
nvm list
```

Ou

```
nvm ls
```


Résultat (exemple)

```
* 18.12.1 (Currently using 64-bit executable)
  14.7.0
```

Node.js

Pour consulter la liste des versions disponibles pour installation

```
nvm list available
```

Node.js

Pour utiliser une version spécifique

```
nvm use 14.7.0
```

Node.js

Pour utiliser une version spécifique

```
nvm use 14.7.0
```

Pour vérifier

```
  18.12.1
* 14.7.0 (Currently using 64-bit executable)
```

Node.js

Pour utiliser la dernière version (lts)

```
nvm use latest
```

Node.js

Pour utiliser la dernière version (lts)

```
nvm use latest
```

Ou

```
nvm use lts
```

Node.js

Pour connaître la version courante utilisée

```
nvm current
```

Node.js

npx : Node Package eXecute

- Outil d'exécution de paquets **Node.js**
- Automatiquement installé avec **Node.js**

Node.js

Commençons par installer globalement le paquet `cowsay`

```
npm install -g cowsay
```

Node.js

Commençons par installer globalement le paquet `cowsay`

```
npm install -g cowsay
```

Pour vérifier que le paquet a bien été installé

```
npm list -g
```

Node.js

Commençons par installer globalement le paquet `cowsay`

```
npm install -g cowsay
```

Pour vérifier que le paquet a bien été installé

```
npm list -g
```

Pour lancer le paquet

```
cowsay salut
```

Node.js

Désinstallons ce paquet `cowsay`

```
npm uninstall -g cowsay
```

Node.js

Désinstallons ce paquet `cowsay`

```
npm uninstall -g cowsay
```

Pour vérifier que le paquet a bien été désinstallé

```
npm list -g
```

Node.js

Pour lancer le paquet `cowsay` sans qu'il soit installé

```
npx cowsay salut
```

Node.js

Pour lancer le paquet `cowsay` sans qu'il soit installé

```
npx cowsay salut
```

Vérifier que le paquet n'a pas été réinstallé en global

```
npm list -g
```

Node.js

Pour lancer le paquet `cowsay` sans qu'il soit installé

```
npx cowsay salut
```

Vérifier que le paquet n'a pas été réinstallé en global

```
npm list -g
```

Ni en local

```
npm list
```

Node.js

Question

Peut-on utiliser **npx** avec tous les packages **Node.js** ?

Node.js

Question

Peut-on utiliser **npx** avec tous les packages **Node.js** ?

Réponse

On peut utiliser la commande **npx** avec les packages **npm** incluant des scripts spécifiés dans leur fichier package.json qui peuvent être exécutés à l'aide de la commande `npm run`.

Node.js

Sites de comparaison npm

- `npm-compare.com` :
  - Comparaison multi-critères : techniques et métriques
  - Compare des packages npm sur plusieurs critères comme la taille, les dépendances, la popularité.
- `npmtrends.com` :
  - Comparaison axée sur la popularité.
  - Affiche des graphiques de téléchargements à partir des données de **npm** et **GitHub**.
- `npms.io`
- `npm-stat.com`
- `openbase.com`
- ...

Node.js

Ressources utiles

- https://nodejs.dev
- https://www.npmjs.com/
- https://node.green/