npm

npm - Gestion de dépendances

- Un gestionnaire de dépendances est un programme qui lance le téléchargement d'une bibliothèque dont dépend votre code, mais également de manière récursive toutes les bibliothèques dont dépendent la bibliothèque installée.
- Equivalent pour du code JavaScript à apt-get
- Principaux gestionnaires de dépendances :
  - Java : Maven, Gradle
  - Ruby : Bundler, gem
  - Python : pip
  - C# : nuget
  - PHP : composer, PEAR
  - Swift / Objective C : CocoaPods
  - JavaScript : npm, yarn, ~~Bower~~, ~~jspm~~, pnpm

npm - Gestion de dépendances

- Il existe plusieurs programme pour la gestion de dépendances en JavaScript
  - npm
    https://www.npmjs.com/
    La norme, s'installe en même temps de Node.js.
  - Yarn
    https://yarnpkg.com/
    Développé par Facebook, a une époque considéré comme plus sécurisé et moins exposé au bugs que npm. La version 5 de npm a comblé son retard. Yarn garde l'avantage de paralléliser les téléchargements.
  - pnpm
    https://pnpm.js.org/
    Ultra-rapide et moins gourmand en disque, n'installe les dépendances qu'une fois par machine, puis des liens symboliques dans les projets
- Benchmarks
  https://github.com/pnpm/node-package-manager-benchmark

npm - Gestion de dépendances

- jspm
  https://jspm.org/
  Permet historiquement d'utiliser facilement des modules ES6 avec SystemJS.
  Pas recommandé pour de nouveaux projets.

- bower
  https://bower.io/
  Développé par Twitter, historiquement adapté aux dépendances front-end. Pas recommandé pour de nouveaux projets.

- Registres
  - Il existe 2 registres de dépendances, npm et bower. Yarn, pnpm et jspm utilisent celui de npm.

npm - Où trouver des bibliothèques ?

- Sur le registre npm
  - Moteur de recherche
    https://www.npmjs.com
  - Regarder les stats d'un paquet :
    https://www.npmjs.com/package/bootstrap
  - ~~Paquets npm les plus utilisées (en nombre de dépendances)~~
    ~~https://www.npmjs.com/browse/depended~~
- Sur GitHub
  - Recherche par nombre d'étoiles :
    https://github.com/search?q=stars%3A%3E0
  - Explorer les projets mis en avant par GitHub
    https://github.com/explore
  - Projets qui reçoivent en ce moment le plus d'étoiles
    https://github.com/trending

npm - Où trouver des bibliothèques ?

- Awesome Lists
  https://github.com/sindresorhus/awesome
  - Awesome Node.js
    https://github.com/sindresorhus/awesome-nodejs
  - Awesome Frontend
    https://github.com/dypsilon/frontend-dev-bookmarks
  - Awesome React
    https://github.com/enaqx/awesome-react
  - Awesome Angular
    https://github.com/gdi2290/awesome-angular
- Autres
  - https://bestofjs.org/
  - https://www.npmtrends.com
  - https://jstools.substack.com/about

npm - Présentation

- Gestionnaire de dépendance de Node.js (en général installé en même temps que Node.js)
- A l'origine, plutôt destiné à du code console ou serveur, bien que des bibliothèques comme jQuery ou Bootstrap y soient présentes depuis toujours
- Aujourd'hui le programme le plus complet mais pas le plus rapide
- Les programmes npm, yarn, pnpm sont incompatibles entre eux, il faut en choisir un dans un projet et s'y tenir

npm - Le fichier package.json

- Aujourd'hui le fichier package.json est le coeur de la configuration d'un projet JavaScript
- On y retrouve principalement
  - Nos dépendances de prod
  - Nos dépendances de dev
  - Les scripts permettant d'interagir avec le projet (build, serveur de dev, tests...)
  - Des noms, descriptions, auteurs, versions, dépôt git en cas de publication sur npm
  - De la config pour le projet où des outils du projets (linters, tests, builders...)
  - ...

npm - Le fichier package.json

- Création d'un fichier package.json minimal
- PAS DE COMMENTAIRES DANS UN FICHIER JSON !

```
{}
```

- Pour le créer en ligne de commande
  - Dans un projet (en mode interactif)
    ```
    npm init
    yarn init
    ```
  - Dans un projet (en forçant yes à toutes les questions)
    ```
    npm init -fy
    ```
  - Pour créer un projet depuis un paquet create-* (ici un paquet create-react-app)
    ```
    npm init react-app mon-app-react
    ```
  - Pour créer un projet en lançant une commande
    ```
    npx create-react-app mon-app-react
    ```

npm - Installation d'un nouveau paquet

- Installer un paquet
  ```
  npm install react
  npm i react
  yarn add react
  ```

- Installer un paquet de dev
  ```
  npm i prettier --save-dev
  npm i prettier -D
  yarn add prettier --dev
  ```

- Installer une version ancienne d'un paquet
  ```
  npm i react@16.0.0
  yarn add react@^16.0.0
  ```

- Installer ou réinstaller la dernière version d'un paquet (dist-tag latest)
  ```
  npm i react@latest
  ```

- D'autre dist-tags peuvent exister (cf la doc du paquet)
  ```
  npm i react@next
  yarn add react@next
  ```

npm - Réinstallation du node_modules

- Commande à lancer quand on récupère un projet sur Github (par exemple) ou quand on récupère des commits qui ont modifié le lockFile
- Pour réinstaller les dépendances telles que demandées par le lockFile (package-lock.json ou yarn.lock
  ```
  npm install
  yarn install
  ```
- Pour ne pas installer les devDependencies (qui ne servent que sur un poste de dev : développeur, CI...)
  ```
  npm install --omit dev
  yarn install --production
  ```
- On peut aussi le faire via une variable d'environnement (recommandé) :
  ```
  NODE_ENV=production
  ```

npm - Lister les dépendances

- Lister les dépendances

```
npm list --all
yarn list
```

```
├─┬ react@16.4.2
| ├─┬ fbjs@0.8.17
| | ├── core-js@1.2.7
| | ├─┬ isomorphic-fetch@2.2.1
| | | ├─┬ node-fetch@1.7.3
| | | | ├─┬ encoding@0.1.12
| | | | | └── iconv-lite@0.4.24 deduped
| | | | └── is-stream@1.1.0
| | | └── whatwg-fetch@2.0.4
| | ├── loose-envify@1.4.0 deduped
| | ├── object-assign@4.1.1 deduped
| | ├─┬ promise@7.3.1
| | | └── asap@2.0.6 deduped
```

- deduped signifie que la dépendance est partagée par d'autre (se retrouve à la racine de node_modules)

- Lister que nos dépendances directes

```
npm list
yarn list --depth 1
```

npm - Détecter des dépendances à mettre à jour

- Savoir quoi mettre à jour
  `npm outdated`
- 3 cas possibles
  - rouge : mise à jour dispo
  - jaune : migration dispo
  - à jour : n'apparait pas

```
MacBook-Pro:front-end-scs-angular romain$ npm outdated
Package                                Current   Wanted   Latest  Location
@angular-devkit/build-angular            0.6.3    0.6.8    0.7.5  front-end-scs-angular
@angular/animations                      6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/cli                             6.0.3    6.0.3    6.1.5  front-end-scs-angular
@angular/common                          6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/compiler                        6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/compiler-cli                    6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/core                            6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/forms                           6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/http                            6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/language-service                6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/platform-browser                6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/platform-browser-dynamic        6.0.2    6.0.2    6.1.6  front-end-scs-angular
@angular/router                          6.0.2    6.0.2    6.1.6  front-end-scs-angular
@ng-select/ng-select                     2.0.3    2.6.0    2.6.0  front-end-scs-angular
@ngx-translate/core                     10.0.1   10.0.2   10.0.2  front-end-scs-angular
```

- Versionnage sémantique
  https://semver.org/
- MAJOR.MINOR.PATCH (ex : 1.2.3)
- New PATCH : pas de changement de comportement (API ou interface identiques)
- New MINOR : changements rétro-compatibles
- New MAJOR : changements rétro-incompatibles (lire le guide de migration)

npm - Lockfile

- Le fichier package.json permet de verrouiller
  - Une version majeure : ^1.2.3
  - Une version mineure : ~1.2.3
  - Une version patch : 1.2.3
- Le fichier package-lock.json permet de verrouiller également les dépendances de ce paquet de telle sorte que tous les environnements (postes de dev, prod...) partage les mêmes versions
- Le lockfile permet également de garder l'URL et le checksum de chaque dépendance du projet (perf + sécurité)
- npm maintient un cache sur le disque, les installations ultérieures seront plus rapides
- Les lockfiles de npm, yarn et pnpm sont incompatibles entre eux (parfois même entre 2 versions d'npm !)

npm - Mettre à jour

- Mettre à jour tous les paquets packages installés
  ```
  npm update
  yarn upgrade
  yarn upgrade-interactive
  ```
- Mettre à jour un seul paquet
  ```
  npm update jquery
  yarn upgrade jquery
  ```
- Migrer un paquet vers une version majeure
  ```
  npm i jquery@2
  yarn add jquery@2
  ```
- Migrer un paquet vers la dernière version majeure
  ```
  npm i jquery@latest
  yarn add jquery
  ```
- Supprimer un paquet
  ```
  npm uninstall react
  npm rm react
  yarn remove react
  ```

npm - Lancer un script

- Lancer un script
  ```
  npm run-script nom-du-script
  npm run nom-du-script
  yarn nom-du-script
  ```
- Certains script on des alias (start, test...)
  ```
  npm run-script test
  npm run test
  npm test
  npm t
  ```
- Certains script peuvent s'exécuter automatiquement
  postinstall, prepublish, pretest...
- Lancer un script permet d'exécuter un programme local (dans node_modules/.bin)

npm - Configuration

- Utilisation d’un proxy
  npm config set proxy http://host:8080
  npm config set proxy http://user:pass@host:8080
- Supprimer une config
  npm config rm proxy
- Lister les configs
  npm config list