

# TP2 : GIT et NPM

---



Ce TP a pour but de vous initier à GIT qui permettra de versionner, sauvegarder, ... le projet fil rouge du module. Nous allons aussi voir ensemble le gestionnaire de paquet NPM.

## GIT

---

Pour la partie GIT, je vous invite à vous créer un compte sur <https://github.com> qui est le site le plus connu et surtout le plus utilisé par de potentiels efuturs employeurs afin de voir vos différents projets.

Ensuite, je vous invite à réaliser le tutoriel [nodeschool.io](https://nodeschool.io) suivant :

- `git-it`

La partie Git est très importante pour les différentes libraries Node.JS disponibles sur NPM. En effet, elles sont toutes disponibles sur un repository Git (privé ou public). De plus, cela servira pour la suite du TP.

# NPM

---

## Introduction

NPM pour **N**ode **P**ackage **M**anager sert à installer différents packages au sein de votre projet ou de votre système. Avant de rentrer dans les détails, voici le tutoriel [nodeschool.io](https://nodeschool.io) qui se rattache à ce que nous allons voir :

- `how-to-npm`

## Le scope

Le **scope** permet de publier un package au sein d'un "nom de domaine" sous la forme `@votrenom`. Au lieu par exemple d'installer le package via :

```
npm install monpackage
```

vous l'installerez via

```
npm install @votrenom/monpackage
```

## Package.json

Le fichier `package.json` de notre projet ressource toutes les libraries externes de notre projet, les différentes commandes disponibles via `npm` et pleins d'autres informations propres au build, publication, ...

Nous allons nous concentrer ici sur les champs `devDependencies` , `dependencies` et `peerDependencies` .

- `dependencies` : Ce sont les dépendances principales de votre module. Elles permettent à celui-ci de fonctionner correctement en production lorsqu'il est compilé (si besoin).
- `devDependencies` : Ce sont toutes les dépendances dont vous avez besoin pour développer et tester votre projet (par exemple toutes les extensions gulp propre à la compilation de votre projet front).
- `peerDependencies` : Ici cela permet de valider la résolution des dépendances entre votre package et un autre.

Pour plus d'informations sur `peerDependencies` , je vous invite à lire cet article (en anglais) : <https://nodejs.org/en/blog/npm/peer-dependencies/>.

Bien entendu, outre la résolution des dépendances, vous avez aussi la résolution des versions des dépendances. De base, npm installe toujours la version avec un `^` juste devant.

Ex : `"async" : "^1.8.1"` peut vous installer, lors du premier `npm install` une version de `async` entre la `1.8.1` et la `2.0.0` .

Au contraire `"async" : "^0.4.0"` peut vous installer `async` entre la version `0.4.0` et `0.5.0` .

Pour vérifier tout ceci, je vous invite à jeter un oeuil à cet article chez NPM : <https://docs.npmjs.com/misc/semver>

## Réalisation de notre premier module

Réalisation d'un module de cryptage de mots de passe permettant de simplifier les appels du module `encrypt` fourni par Node.JS.

Vous devrez créer le projet sur github afin de le publier ensuite au sein de NPM via un scope qui devra être le pseudo github de celui hébergeant le projet.

Le nom du package sera `iut-encrypt` .

Vous devrez pouvoir encoder un mot de passe en sha1 mais pensez votre module comme évolutif pour pouvoir rajouter d'autres méthodes dans le futur.

