NPM

Initialiser un projet avec npm :

Avec le terminal, aller dans le dossier du futur projet, puis =>

npm init

Cette commande va créer un fichier package.json .

Npm install <nomDuPackage>

Ceci installe une nouvelle dépendance.

Npm ensuite nous demande le nom du package =>

Package name : Le nom de notre package

Version : 0.0.1

Description : description du projet

Entry point : nom du fichier principal

Test command :

Git repository : nom du répertoire, pour faire du versioning

Keywords : node.js , javascript , es6+

Author : moi

License :

Une fois fait, cela créer un fichier package.json avec de nombreuses info concernant le future projet.

Puis on valide avec ok.

{

  "name": "nodejs",

  "version": "0.0.1",

  "description": "faire une api avec NodeJs, expressjs et mongodb",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "keywords": [

    "nodejs",

    "es6+",

    "mongodb",

    "expressjs"

  ],

  "author": "Yan Coquoz",

  "license": "ISC"

}

Apres avoir fait un console.log(‘hello world’) dans le dossier index.js, on écrit dans le terminal =>

node index.js . Là hello world s’affiche. Pour éviter de taper cette ligne de commande à chaque fois, dans le fichier package.json on va ajouter un ligne dans la partie ‘scripts’

"start": "node index.js"

Maintenant dans le terminal, on peut écrire => npm run start. Ou bien npm start.

Pour qu’un serveur écoute (qu’il soit attentif au changement, et qu’il ne plante plus) en permanence il faut installer une dépendance du nom de backpack ou nodemon.

* npm install –save backpack-core

Une fois installer, il faut changer le point d’entrer du serveur au niveau du scripts.

"scripts": {

    "start": "backpack"

  },

Cependant attention, il est fortement conseiller de mettre son fichier index.js dans un dossier src.

Ensuite nous pouvons exécuter la commande npm start.

Puis pour que tous soit optimal, (que les dernières avancée de javascript soit prise en compte) il faut aussi installer Babel.

* npm install –save babel-preset-stage-3

A la racine du projet, on créer un fichier ‘.babelrc’ . A l’intérieur on met :

{

    "presets":[

        "backpack-core/babel",

        "stage-3"

    ]

}

Ce qui correspond à l’utilisation de l’ES6 à ES8 grâce à babel, et certaines options comme le spread operator avec le stage-3 (mise à jour des ecmascript, commence à 0 fini à 4, 4 étant la version final).