# 🪛Qu’est-ce que npm ?

**npm** est le gestionnaire de package par défaut de Node.js. Le Gestionnaire de package permet aux programmeurs de publier et partager plus facilement le code source des bibliothèques Node.js. Il est conçu pour simplifier l’installation, la mise à jour et la désinstallation des bibliothèques.

## Création d'une application

Nous allons donc voir comment démarrer une application sereinement avec l'environnement Node.js dans Visual Studio. VS est un environnement de développement Node.js puissant.

Sous windows, lancez l'invite de commande à l'aide de la touche windows.


Voici les étapes à suivre pour créer une application[1].

De façon classique, tapez dans l'invite de commande

1. cmd

Puis dans la nouvelle fenêtre :

1. >mkdir 📁start-npm (le signe > désigne le prompt, vous ne le tapez pas)

[1] https://code.visualstudio.com/docs/nodejs/nodejs-tutorial#_hello-world

2. >cd start-npm
3. >code .

Explications :

1. Vous créez un📁répertoire start-npm,

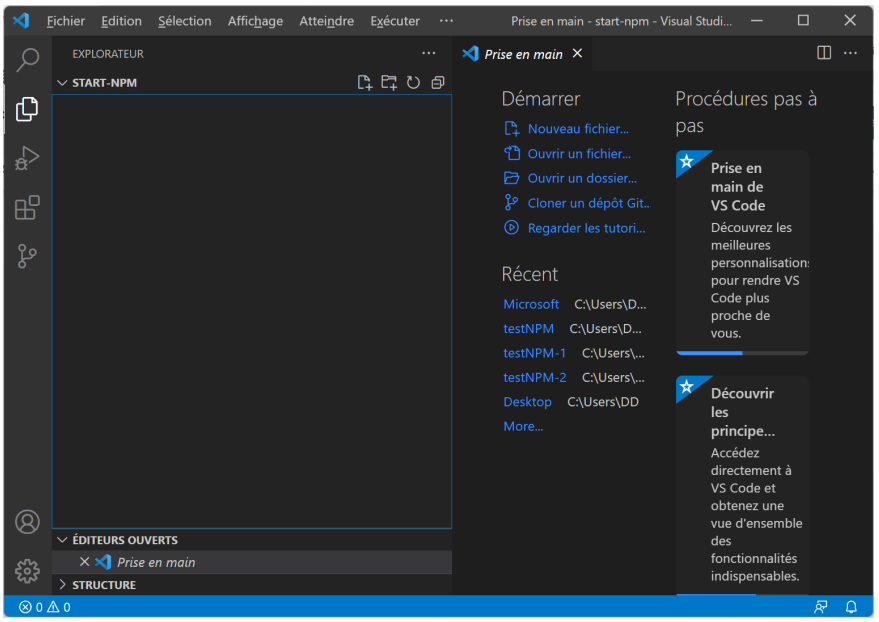
2. Vous allez dans ce répertoire et

3. Vous lancez VS.

Vous pouvez sinon exécuter directement la ligne de commande suivante,

C:\Users\DD\Desktop>mkdir start-npm && cd start-npm && code .

Votre éditeur est lancé, comme le montre la figure suivante.




## 1. Création d'un fichier *package.json*

Mais c'est quoi un package ?

Une recherche rapide, montre par exemple que "jest"[2] existe dans nodejs sous forme de package.

Finalement, je comprends vite que la commande magique pour utiliser le code est : `npm install  jest --save-dev`

Mais, pas si vite, il nous faut avant tout créer un fichier de configuration[3]. Ce fichier a pour nom : package.json.

## package.json ?

Voici une réponse à la question : "ça sert à quoi ce fichier json, un extrait de la documentation"[4].

***"You can add a package.json file to your package to make it easy for others to manage and install. Packages published to the registry must contain a package.json file."***

Ce fichier magique va indiquer les dépendances entre les autres packages pour que **npm** puisse gérer l'ensemble des dépendances.

## Initialisation

Avant de commencer le processus d'initialisation, observez bien le contenu vide du répertoire que vous venez de créer. A l'issue de la phase d'initialisation, il contiendra le fichier de configuration.

---

[2] A comprehensive JavaScript testing solution. Works out of the box for most JavaScript projects. La documentation est pour l'instant très peu compréhensible (cela viendra).

[3] Dans le cas du débogger la nécessité de créer également un fichier de configuration.

[4] https://docs.npmjs.com/creating-a-package-json-file

Pour créer le fichier package.json, nous facilite la tâche.

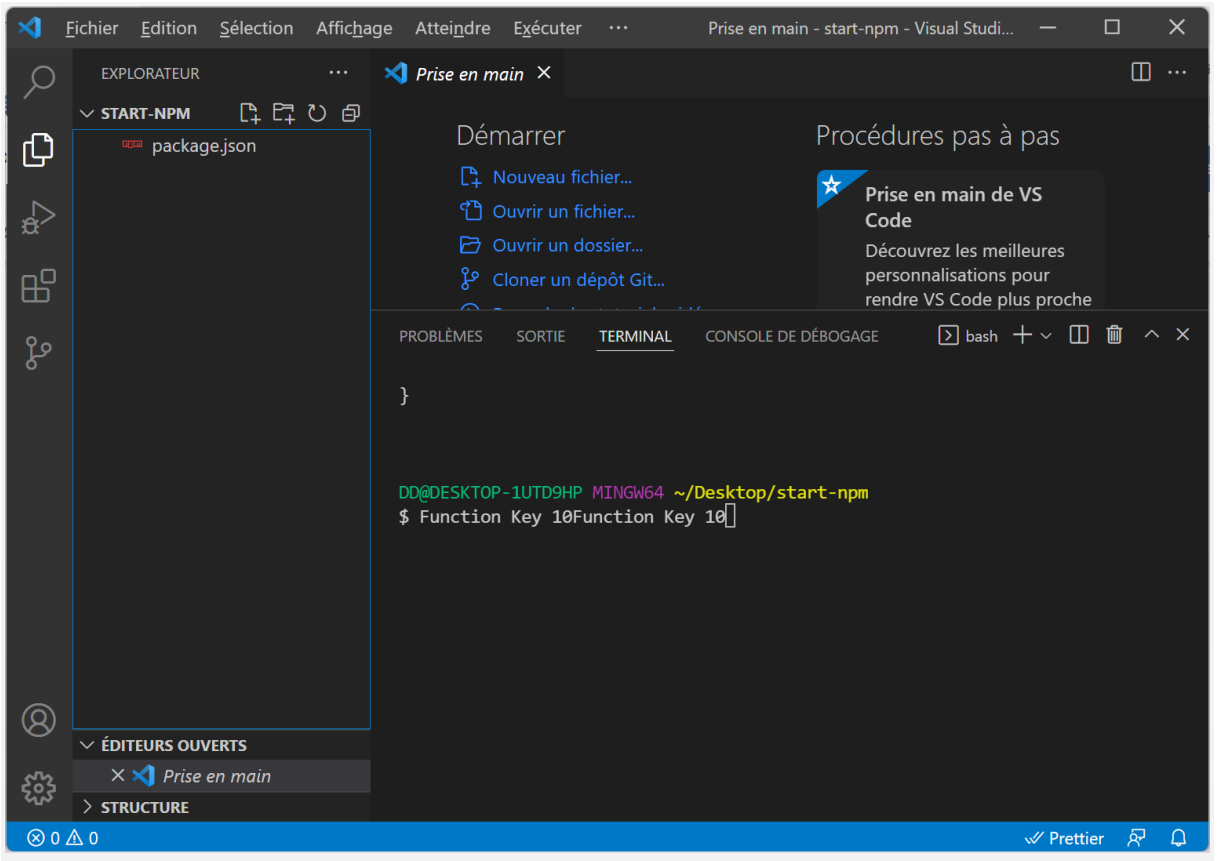
En effet, npm monte la structure.

Dans le terminal taper la commande : npm init.

```
$ npm init
```

Vous répondez par la touche **entrée** ↩ à chaque question posée. Nous reviendrons sur la valeur des attributs.

Dans l'explorateur de fichier, observez bien l'apparition de votre fichier package.json.

1. 

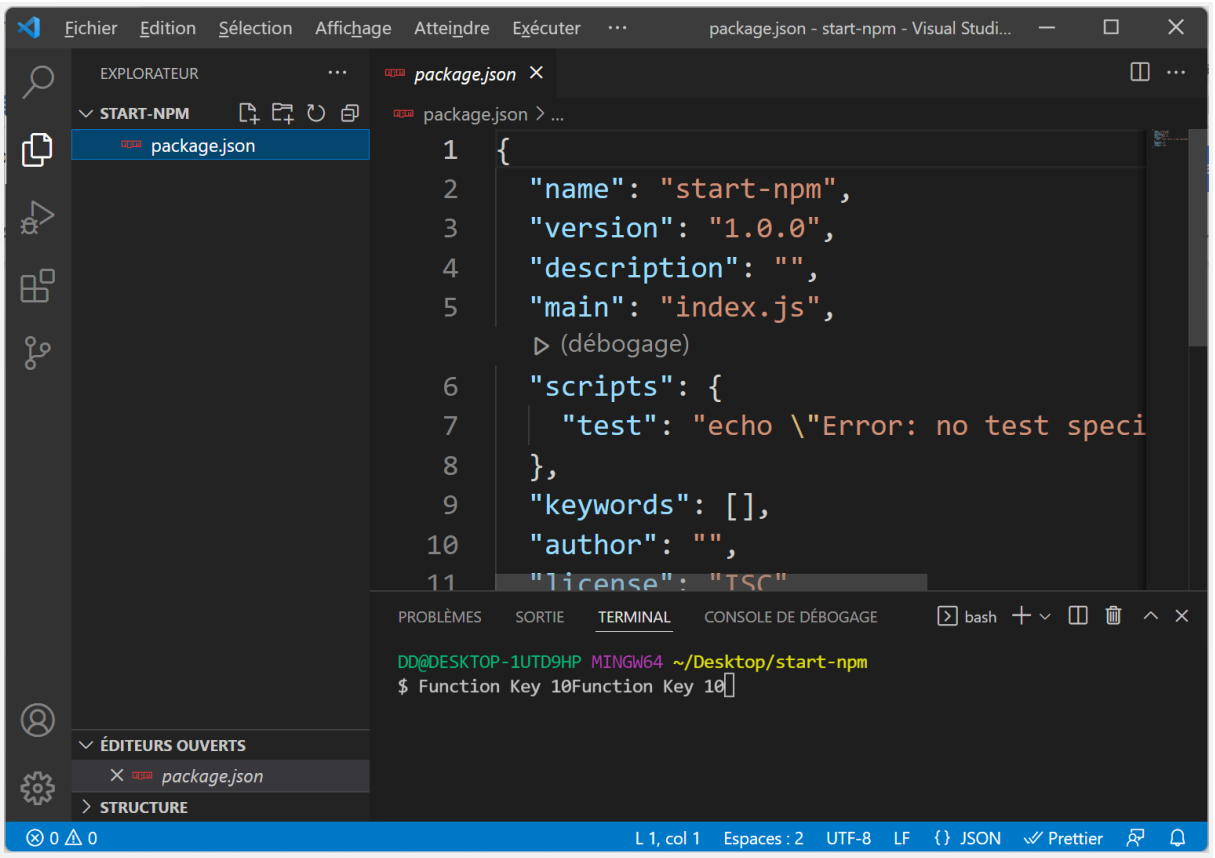
Vous pouvez éditer ce fichier[5] :

 package.json.

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

[5] Sa lecture reste encore floue.

Vous pouvez **modifier** des informations dans le fichier, donnez par exemple un nom d'auteur au projet.




🥷Vous pouvez **ajouter** des informations dans le fichier,

Ajoutez l'entrée suivante à la section scripts : "start": "node ./src/index.js"

package.json.

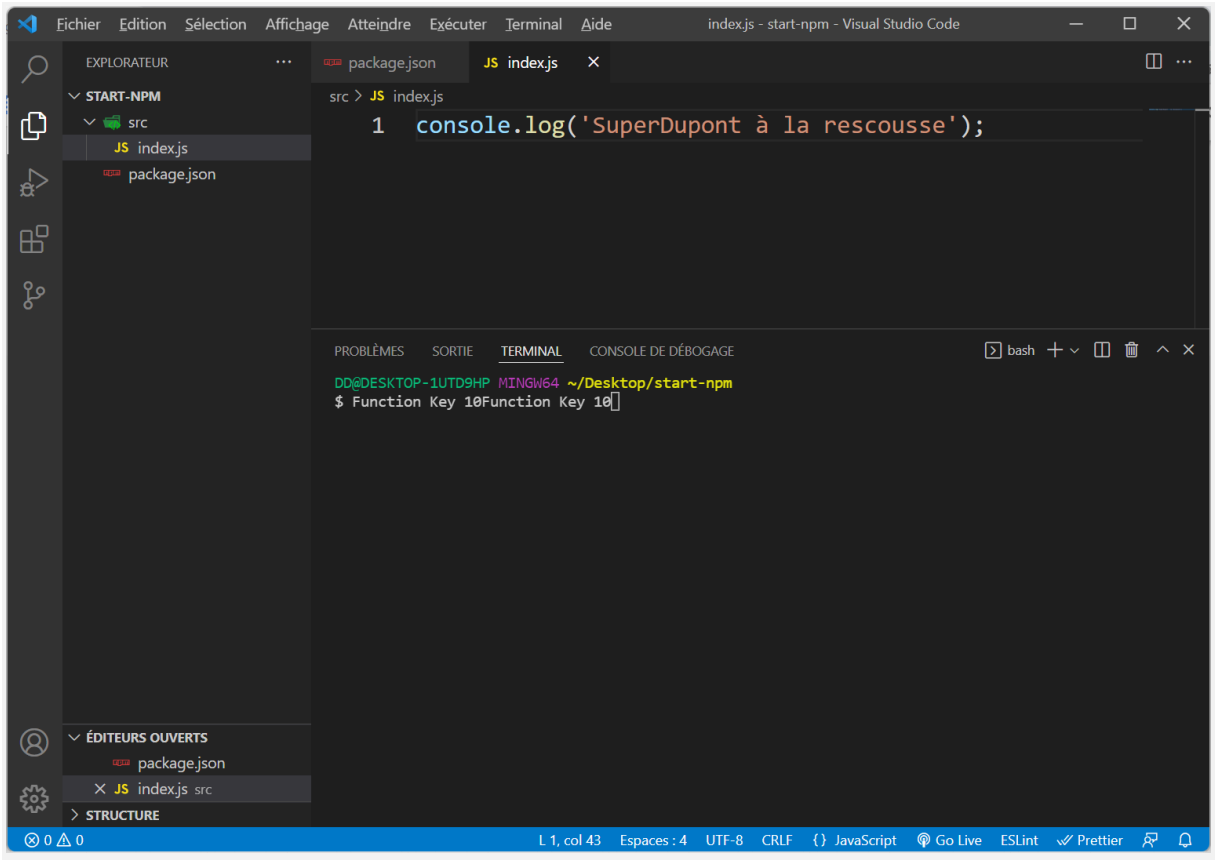
6. "scripts": {
7. "start": "node ./src/index.js",
8. "test": "echo \"Error:"
9. },

Créez le répertoire 📁src et ajoutez-y un fichier index.js.

Ajoutez dans ce fichier un simple affichage.

## index.js

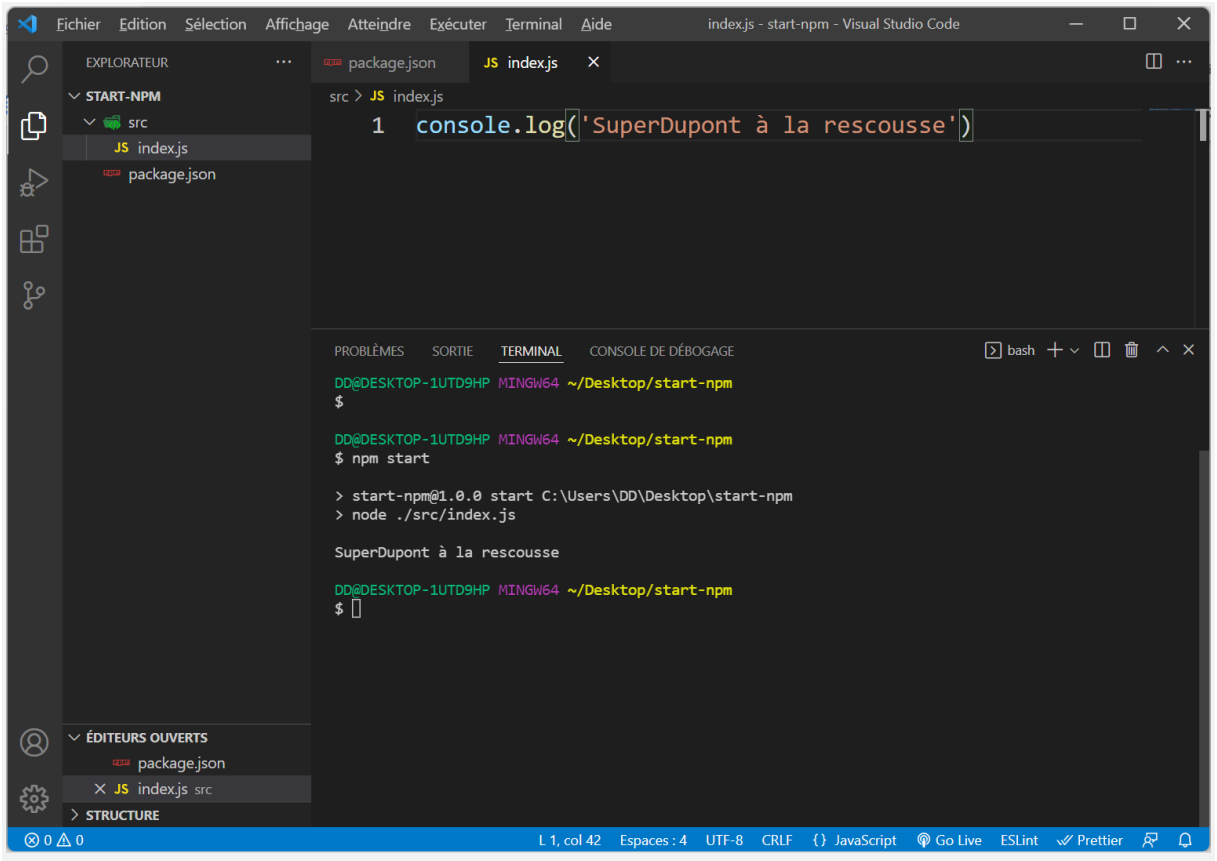
1. console.log('SuperDupont à la rescousse');




🪛Exécutez la commande :

```
>npm start
```

Vous devriez avoir un message de SuperDupont




Nous avons une application qui est correctement mise en place. Nous allons installer un paquet. Ce paquet a pour rôle de nous aider à **tester** une application.

## Installation de paquets !

Commencez par créer un fichier parser.js et ajoutez le code suivant :

parser.js

```
1. exports.parse = function parseOrder(order) {
2.   const match =
     order.match(/name:\s(?<name>\w+\s\w+).*address:\s(?<address>\
     w+\s\w+\s\w+).*city:\s(?<city>\w+\s\w+).*/)
```

```
3.    return match.groups;
4.  }
```

👹Lig. 2 : le code est incompréhensible pour les novices en regexp. Disons simplement que nous testons le format de l'argument[6].

## Explications

Prenons la partie de l'expression régulière suivant :
name:\s(?<name>\w+\s\w+) où :

name : correspond aux caractères name : littéralement (sensible à la casse)

\s: correspond à n'importe quel caractère d'espacement (équivalent à [\r\n\t\f\v ])

(): création d'un groupe

Nom du groupe: de capture nommé (?<name>\w+\s\w+)

\w: correspond à n'importe quel caractère verbal (équivalent à [a-zA-Z0-9_])

+: correspond à l'élément précédent entre une et un nombre illimité de fois, autant de fois que possible, en restituant si nécessaire (gourmand)

\s: correspond à n'importe quel caractère d'espacement (équivalent à [\r\n\t\f\v ])

\w: correspond à n'importe quel caractère de mot (équivalent à [a-zA-Z0-9_])

+: correspond au jeton précédent entre une et un nombre illimité de fois, autant de fois que possible, en restituant si nécessaire (gourmand)

[6] https://regex101.com/r/Qkt5uS/1

Quelle expression !

Un outil en ligne pourrait vous aider : https://regex101.com/

Cliquez sur la figure pour voir le code en direct[7] :


Mais revenons au code de parser.js.

Il faut pour l'instant se focaliser sur le mot exports en lig.1.

Nous reviendrons sur ce mot magique qui permet à votre application de disposer de ce code dans d'autres fichiers.

## Installez Jest

Jest est une bibliothèque qui permet de tester votre code ! Des milliers de lignes de code sont mises à disposition de tous gratuitement.

Pour installer Jest exécutez la commande :

```
npm install jest --save-dev
```

Ouvrez le fichier package.json et recherchez la section devDependencies. (nous aurons certainement un autre numéro de version )

```
"devDependencies": {
    "jest": "^27.5.1"
  }
```

[7] https://regex101.com/r/Qkt5uS/2

Dans le fichier package.json ajoutez à script :

```
"scripts": {
  "start": "node ./src/index.js",
  "test": "jest"
},
```

## Mise en place du test

Créez un répertoire __tests__. Notez bien son orthographe car *jest* recherche les fichiers de test dans ce répertoire automatiquement.

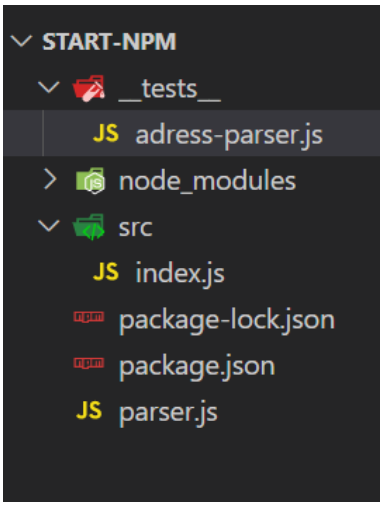
Ajoutez le fichier adress-parser.js

adress-parser.js

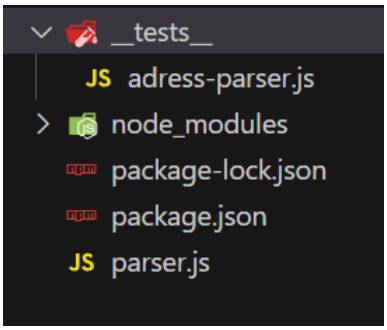
```
1.  const { parse } = require("../parser");
2.
3.  describe('Format parser', () => {
4.    test('Vérification du format', () => {
5.      expect(
6.        parse(
7.          "name: Dupont Denis dit SuperDupont address: 44 rue Lebel (ligne
        1) city: 94300 Vincennes "
8.        )
9.        ).toEqual({
10.         name: "Dupont Denis",
11.         address: "44 rue Lebel",
12.         city: "94300 Vincennes"
```

```
13.      });
14.  })
15.})
```

Votre architecture doit ressembler à

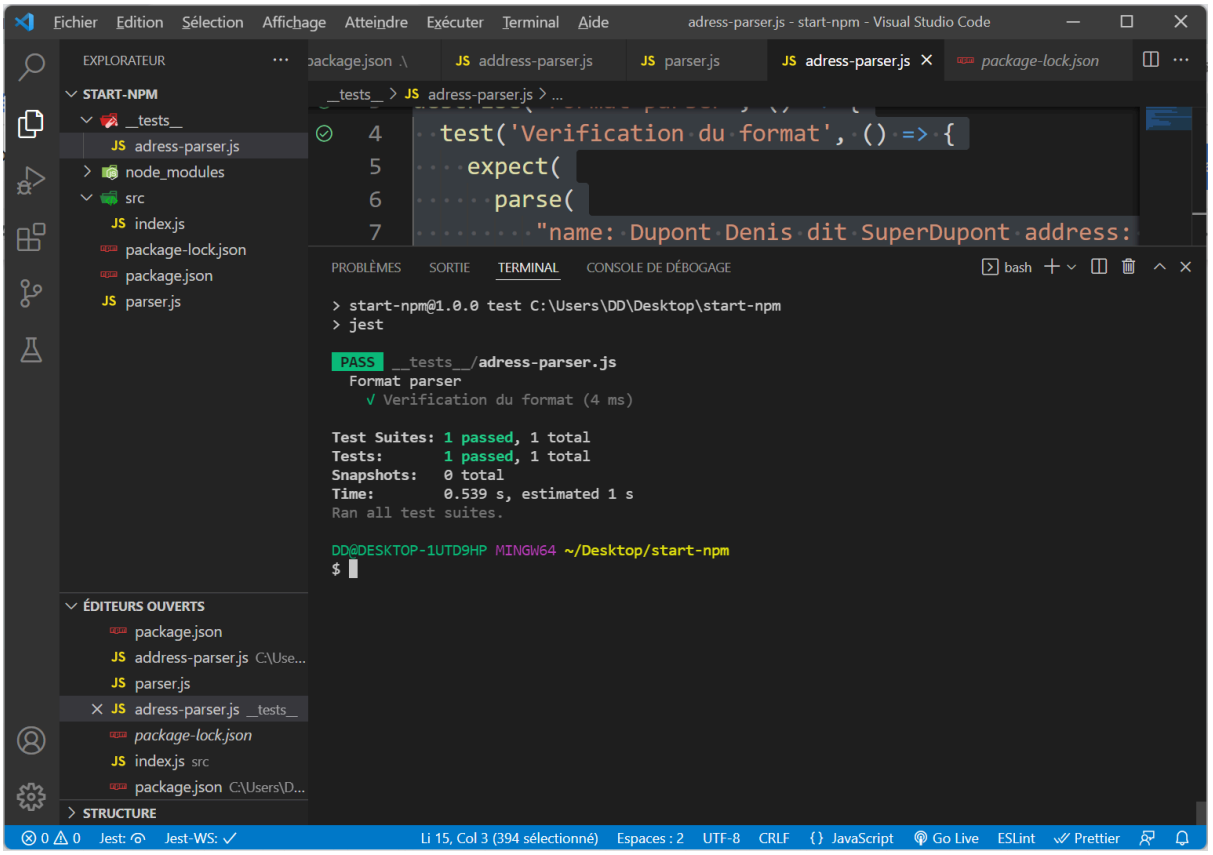



Ou si vous recréez une nouvelle application, vous aurez la structure suivante.




Lancez le test en exécutant la commande npm test

```
>npm test
```




Amusez-vous à tester votre propre adresse !

Elle devra vérifier un format strict :

"name: item1 item2 **blabla** address: item1 item2 item3 **blabla** city: item1 item2 **blabla** "

- les noms clés name, address et city sont obligatoires
- les items sont obligatoires
- **blabla** est quelconque est peut être ignoré.

Par exemple, modifiez le fichier adress-parser.js en changeant le code postal

adress-parser.js

1. const { parse } = require("../parser");

```
2.
3.  describe('Format parser', () => {
4.    test('Vérification du format', () => {
5.      expect(
6.        parse(
7.          "name: Dupont Denis dit SuperDupont address: 44 rue Lebel (ligne
    1) city: 94300 Vincennes en île de France"
8.        )
9.        ).toEqual({
10.         name: "Dupont Denis",
11.         address: "44 rue Lebel",
12.         city: "94000 Vincennes"
13.       });
14.  })
15.})
```