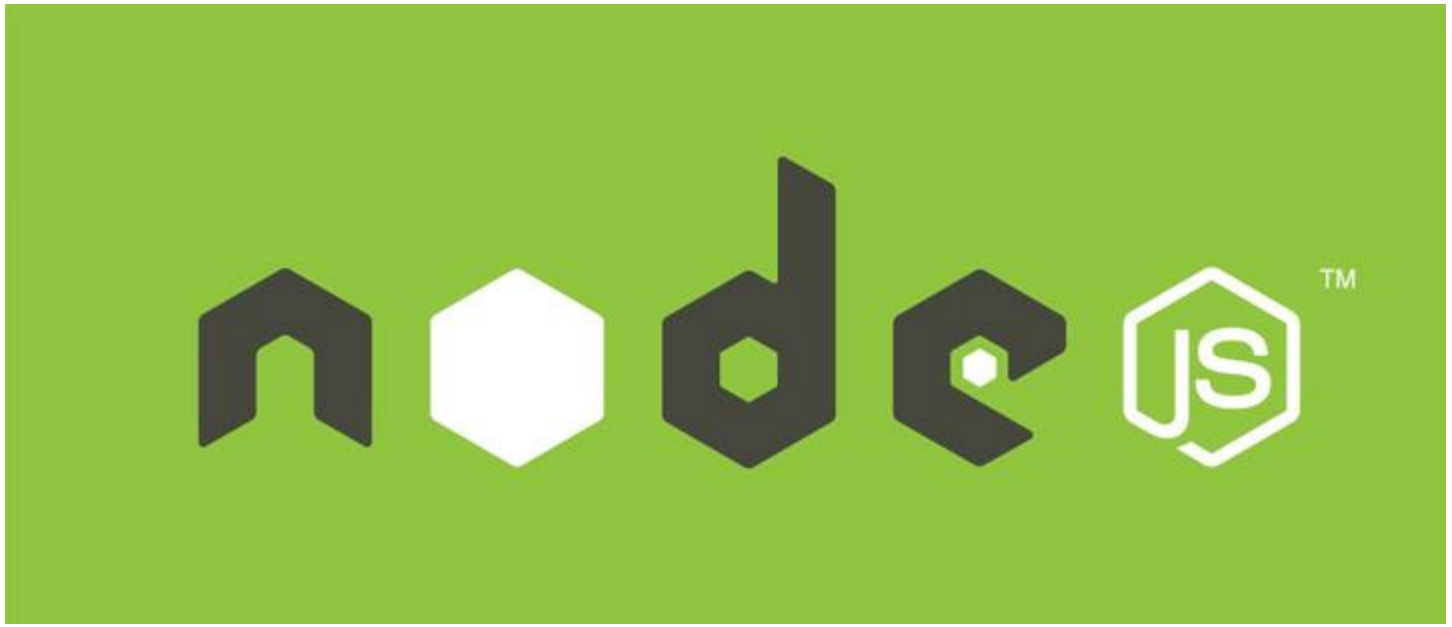


Chapitre 1 : Introduction à Node.js

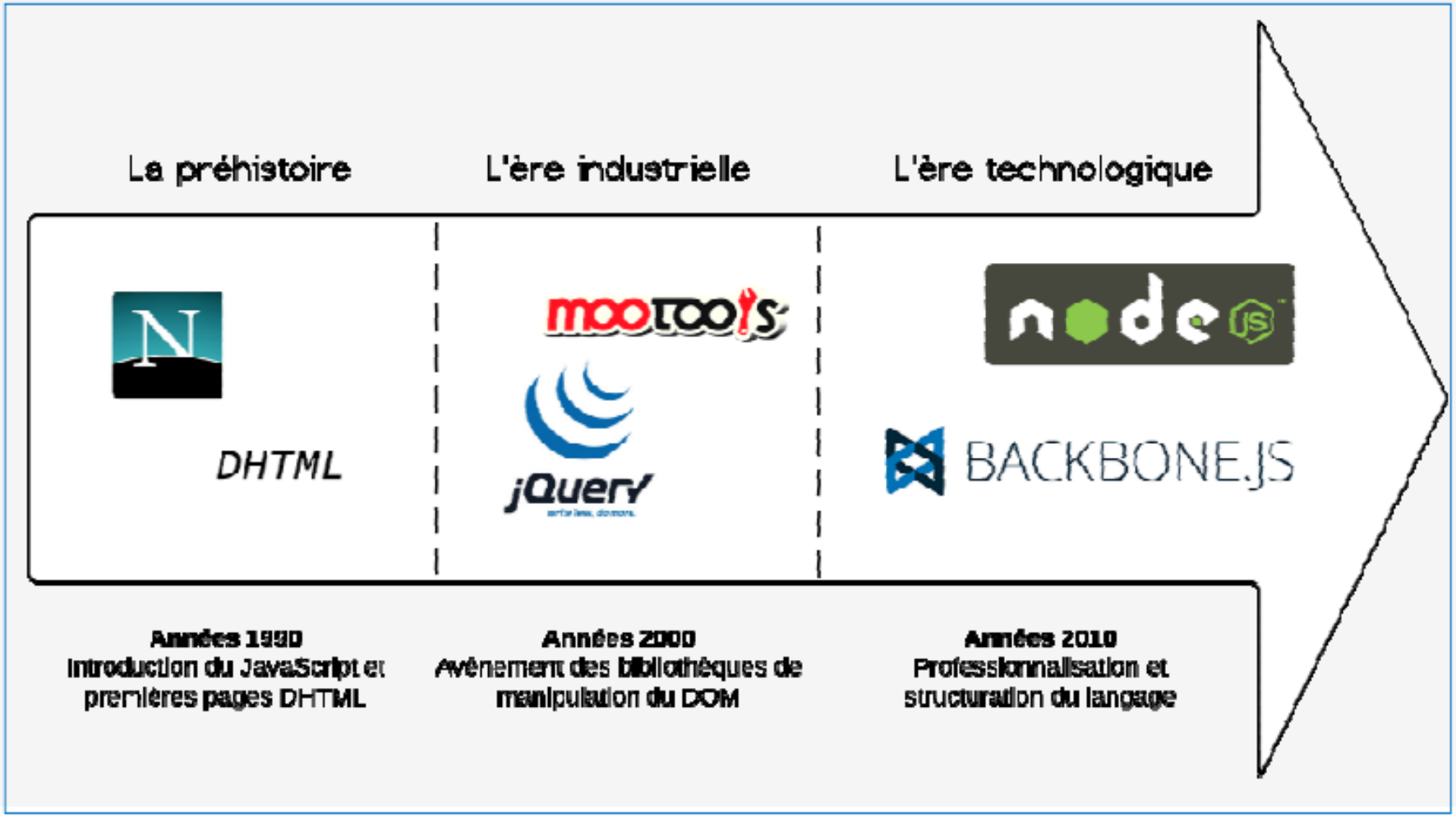
Hachicha Hassan ISET Sfax

Initiation à Node JS

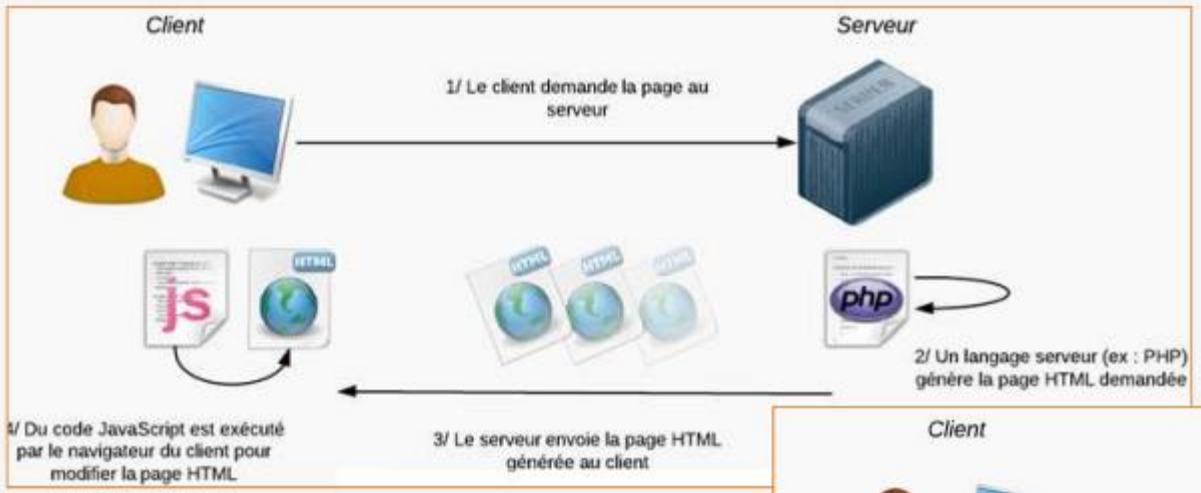
- **Node JS :**
- Plateforme de développement : ensemble de librairies JavaScript
- Permettant l'exécution de JavaScript côté serveur et de réaliser des tâches comme :
  - Persistance de données
  - Manipulation de fichiers
  - Sécurité
  - ...
- Créée par Ryan Lienhart Dahl en 2010

Initiation à Node JS

- **Du JavaScript à Node JS :**



Node JS : du JavaScript sur le serveur



Modèle bloquant vs modèle non bloquant

Imaginez un programme dont le rôle est de télécharger un fichier puis de l'afficher. Voici comment on écrirait le code dans un **modèle bloquant** :

```
Télécharger un fichier
Afficher le fichier
Faire autre chose
```

Les actions sont effectuées dans l'ordre. Il faut lire les lignes de haut en bas :

1.Le programme va télécharger un fichier sur Internet

2.Le programme affiche le fichier à l'utilisateur

3.Puis ensuite le programme peut faire d'autres choses (effectuer d'autres actions)

Modèle bloquant vs modèle non bloquant

Maintenant, on peut écrire le même code sur un modèle non bloquant :

Télécharger un fichier

Dès que c'est terminé,

afficher le fichier

Faire autre chose

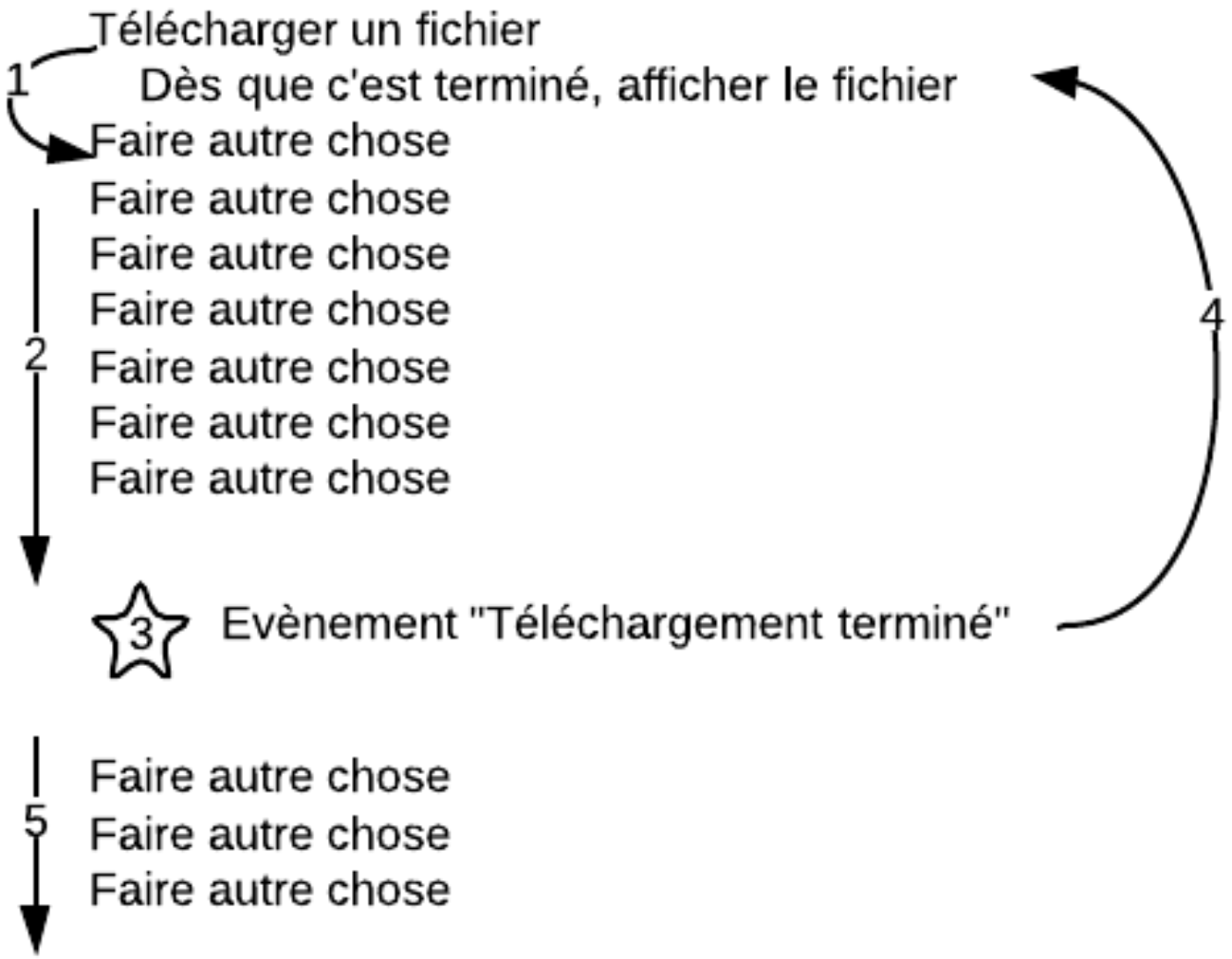
Le programme n'exécute plus les lignes dans l'ordre où elles sont écrites. Il fait ceci :

Le programme lance le téléchargement d'un fichier sur Internet

Le programme fait d'autres choses (le programme suit son cours)

Dès que le téléchargement est terminé, le programme effectue les actions qu'on lui avait demandées : il affiche le fichier

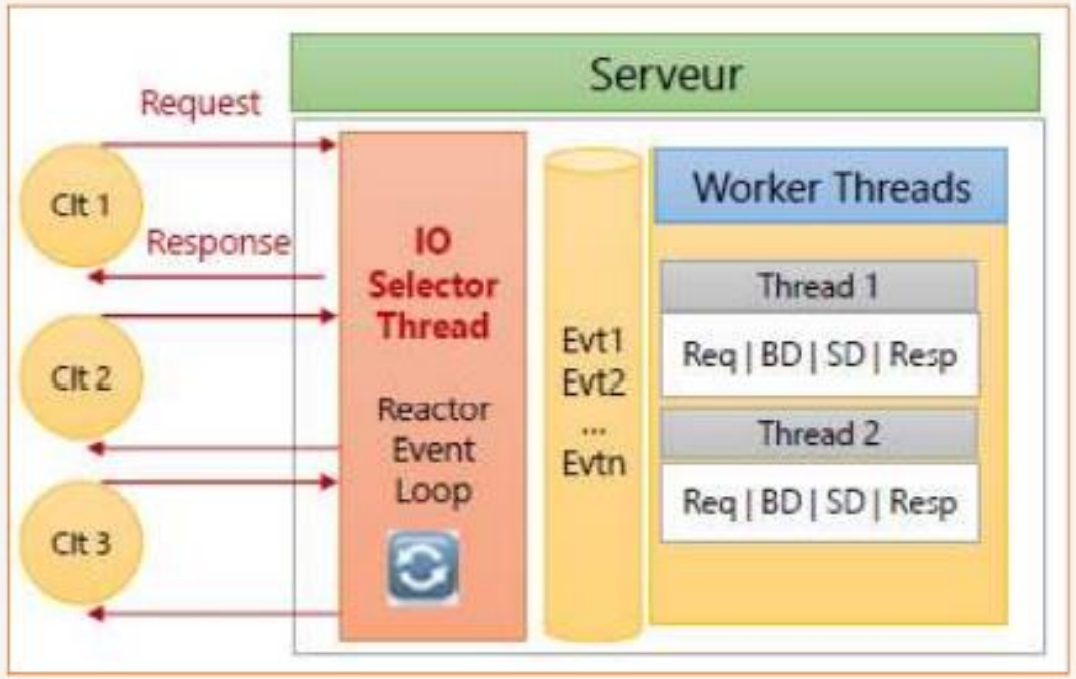
Modèle bloquant vs modèle non bloquant



Node JS : Quelques caractéristiques

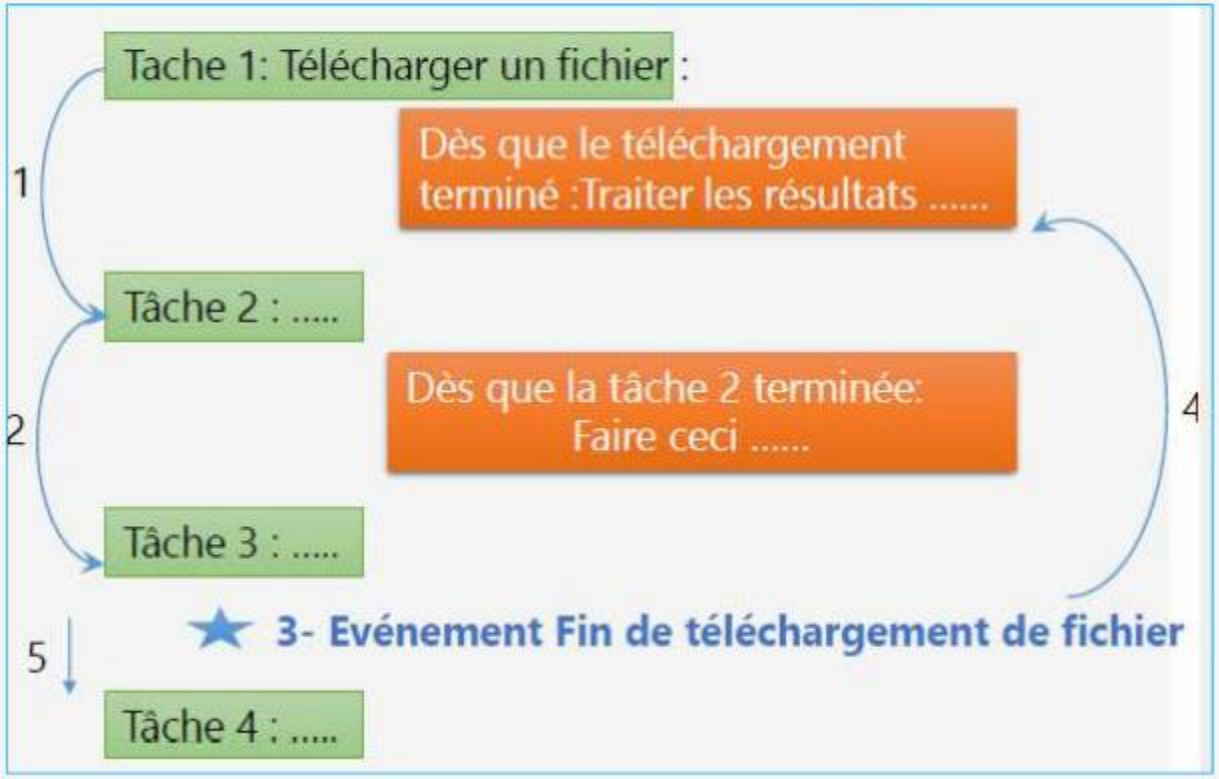
Modèle « Single Thread non bloquant »

- **Node** n'utilise qu'un seul thread pour gérer les requêtes en utilisant des E/S non bloquantes.
- Le code NodeJS est asynchrone piloté par des événements et en utilisant des callbacks pour chaque action.
- Ce qui permet qu'avec un seul Thread, on peut traiter simultanément plusieurs requêtes.
- **Node** permet de développer très simplement des applications scalables.
- S'appuie sur V8, le moteur JavaScript de Google utilisé dans Chrome, qui fait partie des moteurs JavaScript les plus puissants.



Node JS : modèle non bloquant

- **Node JS** : architecture de code entièrement non bloquante



Node JS : installation

- **Installation :**
  - https://nodejs.org/fr/download/
  - **node -v** : version de node

- Après installation de NodeJS, vous disposez de :
  - L'outil **Node** qui permet d'exécuter une application NodeJS
  - L'outil **npm** (Node Package Manager) qui est un gestionnaire de paquets officiels de NodeJS.

Node JS : installation

- **Initialisation d'un projet NodeJS :**
  - **`npm init`** ou encore **`npm init -y`** (sans le mode interactif)
  - Permet de générer le fichier ***package.json***
    - **package.json** :
      - les infos sur le projet,
      - déclarer les dépendances du projet,
      - déclarer les scripts, Etc…

```
var http = require('http');

var server = http.createServer((req, res) =>{
    http.request('http://www.site.com/fichier.zip', function (
error, res, body) {
    console.log("Fichier téléchargé !");
});
console.log("Je fais d'autres choses en attendant...");
res.end('Hello World de Node.js');
});

    server.listen(3000);
```

Framework Express

- **Quelques frameworks NodeJS :**
  - **ExpressJS**
  - Ionic
  - NestJS, ...
- **Express.js (ExpressJS ou Express) :** micro-Framework pour Node.js.
  - Fournir des outils de base pour aller plus vite dans la création d'applications Node.js.
  - Offrir des fonctionnalités pour :
    - La gestion des routes (système de navigation)
    - Un moteur de Templates (Les vues de l'application)
    - Les middlewares : fonctions ayant accès à l'objet *request* et *response*

Mise en place d'un serveur express

- Installation :
  - `npm install --save express`
- Tester le démarrage du serveur :

```
const express = require('express')
const app = express()
const port = 4000

app.listen(port, () => {
  console.log(`Server is running on port ${port}`)
})
```

Gestion des routes

- **Express.js :** Gestion des routes simples

```
const express = require('express')
const app = express()
const port = 4000

app.get('/', function(req, res) {
  res.send('Vous êtes à la page d\'accueil du serveur');
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`)
})
```

Gestion des routes

- **Express.js :** Gestion des routes dynamiques

```
app.get('/infos/:code', function(req, res) {
    res.setHeader('Content-Type', 'text/plain');
    res.send('Vous êtes la personne ayant le code' + req.params.code);
});
```

- Au niveau du serveur node JS, localhost:4000; en évitant de passer le suffixe "?variable=valeur", les urls s'écrivent comme suit :

  /infos/1
  /infos/2
  /infos/3

Package nodemon

- Pour éviter le redémarrage du serveur à chaque fois, vous pouvez installer :
  - **Nodemon :** (https://www.npmjs.com/package/nodemon)
    - **`> npm install -g nodemon`**
    - **`> npm install --save-dev nodemon`**
  - Au niveau de "start " de " scripts ", mettez :
    - ***nodemon*** à la place de ***node***

```
> nodemon server.js

[nodemon] 2.0.12
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server is running on port 4000
```

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "nodemon server.js"
},
"keywords": [],
"author": "",
"license": "ISC",
"dependencies": {
  "express": "^4.17.1"
},
"devDependencies": {
  "nodemon": "^2.0.12"
}
}
```