

NodeJS : notions de base

- Introduction
(historique, concepts de base, avantages et inconvénients,...)
- Liens utiles
- Installation
- Hello World
- Modules NodeJS
- package.json
- npm : commandes et options

Evolution de JavaScript

- Langage créé en 1995
- Dans les **années 1990**, utilisé surtout pour créer des petits effets dans les pages Web (DHTML)
- Dans les **années 2000**, utilisé dans des bibliothèques pour créer des interfaces client (jQuery, Mootools, Dojo, Prototype,...)
- Dans les **années 2010**, sert aussi du côté serveur (NodeJS) et dans des frameworks (Angular, ReactJS, Vue.js, Ember.js,...)

Les origines de NodeJS

- **NodeJS** a été créé en 2009 par Ryan Dahl mais les **premières applications** utilisables en **production** ne datent que de **2015**
- C'est un programme **open-source** qui est basé sur le **moteur V8 de Google** pour interpréter le JavaScript (« **server-side** »)
- Il existe sur Windows, Linux, OS X,...
- Il est utilisé notamment par IBM, Microsoft, Yahoo, SAP, PayPal, LinkedIn, Netflix,...

Concepts de NodeJS

- Comme JavaScript est basé sur les événements, NodeJS est aussi basé sur une **architecture "event-driven"** : le code va soit réagir à un événement soit lever un événement
- NodeJS n'est pas un langage, ce n'est pas un framework, c'est un **environnement d'exécution bas niveau**
- NodeJS est **mono-thread**, alors que les autres serveurs (PHP, .NET, Java,...) sont multi-threads
- NodeJS sur un modèle **asynchrone non-bloquant**

Monothread vs. Multithreads

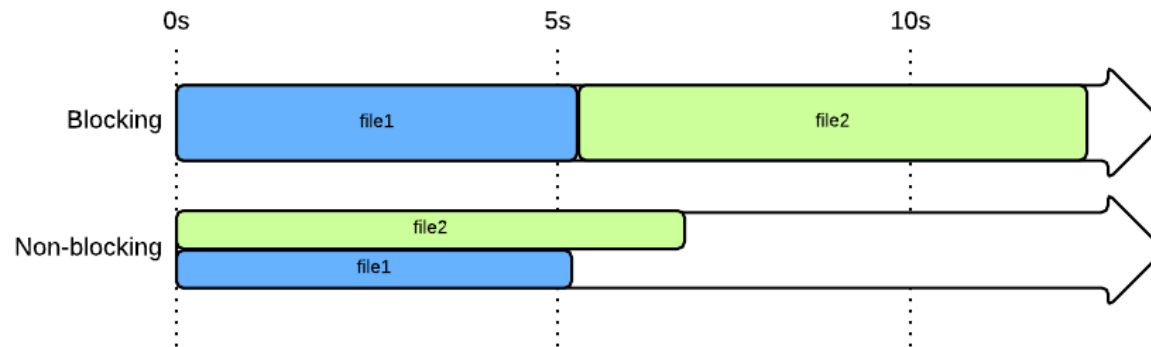
- NodeJS gère **un seul processus** sur un seul CPU et sans parallélisme. Si une tâche prend trop de temps, il démarre la suivante et revient quand la première est terminée.

ex : Le serveur d'un resto passe de table en table

- Les autres serveurs gèrent **plusieurs processus** en parallèle. Mais ils traitent la tâche jusqu'au bout. Si elle prend du temps, on doit attendre ou on doit augmenter les ressources en mémoire et CPU.

ex : Les clients font la file pour passer commande

Non-blocking I/O



- Ex : on télécharge 2 fichiers « file1 » et « file2 »
 - En mode bloquant (ex : Apache) on attend la fin du téléchargement de file1 pour débiter file2
 - En mode non-bloquant (ex : NodeJS) l'application continue de tourner pendant que file1 et file2 sont téléchargés. On est averti à la fin

Moteur V8

- Le moteur V8 a été créé par **Google** en 2008 pour Chromium. Depuis, il est utilisé sur d'autres projets (NodeJS, MongoDB,...)
- **V8** transforme le JavaScript en code machine avant de l'exécuter (compilation JIT au lieu d'être interprété). Ce code est ensuite optimisé, d'où sa performance.
- V8 gère le standard **ECMAScript 6**

Avantages et Inconvénients

- Il est conçu pour optimiser les possibilités et les évolutions d'une application web en temps réel
- Il peut fonctionner hors navigateur. Mais comme il est bas niveau, on doit gérer le serveur Web soi-même
- L'utilisation des événements et des call-backs est plus complexe à comprendre
- Il dispose d'une grande communauté
- Il est modulaire, rapide, évolutif (scalable)

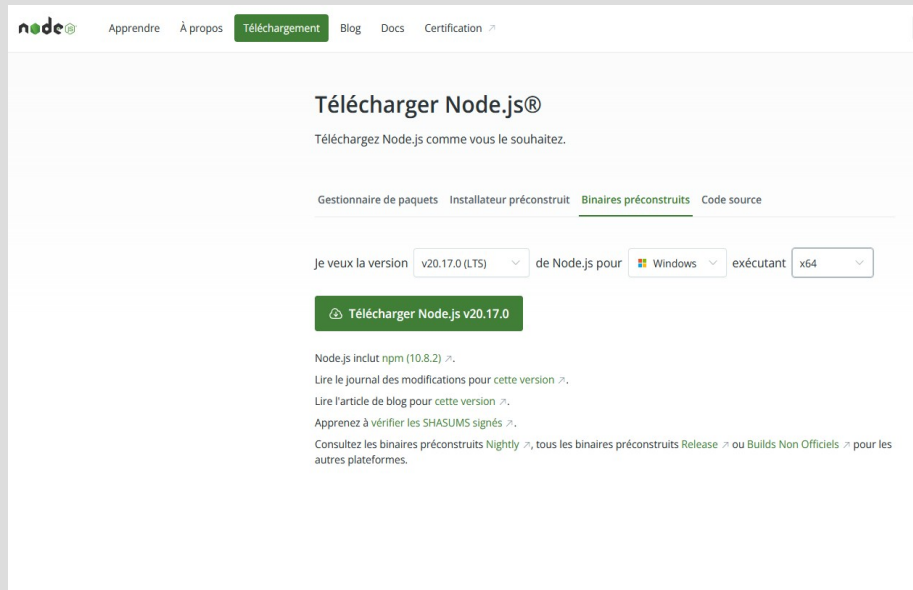
Liens utiles

- Site officiel

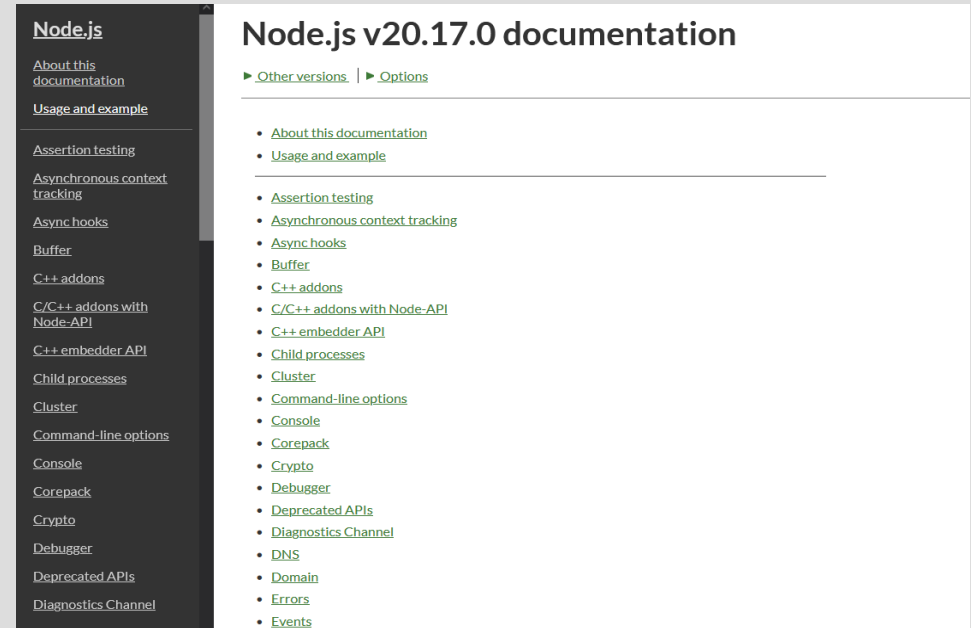
<https://nodejs.org/fr/download/>

- Documentation
(ex: version 20.17.0 LTS)

<https://nodejs.org/docs/latest-v20.x/api/>



The screenshot shows the Node.js download page in French. The header includes the Node.js logo and navigation links: Apprendre, À propos, Téléchargement (highlighted), Blog, Docs, and Certification. The main heading is "Télécharger Node.js®" with the subtext "Téléchargez Node.js comme vous le souhaitez." Below this, there are tabs for "Gestionnaire de paquets", "Installateur préconstruit", "Binaires préconstruits" (selected), and "Code source". A form allows selecting the version (v20.17.0 LTS), platform (Windows), and architecture (x64). A green button labeled "Télécharger Node.js v20.17.0" is prominent. Below the button, there are links for "Node.js inclut npm (10.8.2)", "Lire le journal des modifications pour cette version", "Lire l'article de blog pour cette version", "Apprenez à vérifier les SHASUMS signés", and "Consultez les binaires préconstruits Nightly, tous les binaires préconstruits Release ou Builds Non Officiels pour les autres plateformes."



The screenshot shows the Node.js v20.17.0 documentation page. The header includes the Node.js logo and navigation links: About this documentation, Usage and example, Assertion testing, Asynchronous context tracking, Async hooks, Buffer, C++ addons, C/C++ addons with Node-API, C++ embedder API, Child processes, Cluster, Command-line options, Console, Corepack, Crypto, Debugger, Deprecated APIs, and Diagnostics Channel. The main heading is "Node.js v20.17.0 documentation" with sublinks for "Other versions" and "Options". A list of links is provided, including "About this documentation", "Usage and example", "Assertion testing", "Asynchronous context tracking", "Async hooks", "Buffer", "C++ addons", "C/C++ addons with Node-API", "C++ embedder API", "Child processes", "Cluster", "Command-line options", "Console", "Corepack", "Crypto", "Debugger", "Deprecated APIs", "Diagnostics Channel", "DNS", "Domain", "Errors", and "Events".

Installation de NodeJS

- Versions **LTS** (Long Term Support : 18 mois)
- Autres versions (stable, active, maintenance)
- Actuellement, c'est la version 20.17.0 LTS
 - Calendrier des versions
- Installation à partir d'un installeur ou des sources
- Vérification dans une fenêtre de ligne de commande : **node --version** ou **node -v**

Livre Node.js – Apprendre par la pratique

Hello World

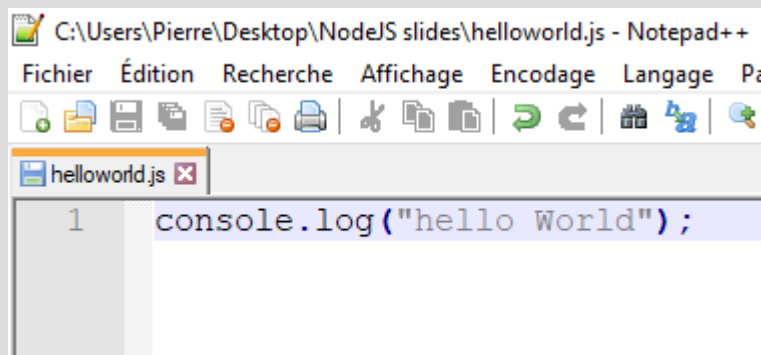
- NodeJS REPL (Read-Eval-Print-Loop)

- On peut tester des petites portions de code

```
I:\nodeProjets>
I:\nodeProjets>node --version
v20.17.0

I:\nodeProjets>node
> console.log("Hello World")
Hello World
```

- Créer un fichier helloworld.js et l'exécuter avec la commande : **node helloworld.js**



Utilisation d'un IDE

- **IDE** (Integrated Development Environment) est un logiciel facilitant le développement
- Il est souvent composé d'un éditeur de code, d'un debugger et de plusieurs fonctionnalités telles que « autocomplétion », « coloration syntaxique », « plugins d'extension »,...
- Clients lourds (Notepad++, VSCode, Atom, Brackets, SublimeText, NetBeans, Eclipse,...)
- En ligne (Codeanywhere.com, Koding.com, CodeBox,...)

Modules de NodeJS

- Le **cœur de NodeJS** est petit mais sa richesse, ce sont les modules autour.
- Un **module** encapsule le code d'une partie fonctionnelle d'une application
- Un module peut exporter un autre module
- On utilise « **require** » pour importer un module et « **exports** » pour exporter un module
- Il existe des modules **natifs**, dans NodeJS et des modules **communautaires** à récupérer via NPM

Modules natifs

- La liste des modules natifs et leurs API sont disponibles sur <https://nodejs.org/api/> ou sur <https://nodejs.org/dist/latest-v20.x/docs/api/>
- Par exemple : Process, OS, Path, Util, Sys, FileSystem, ReadFile, Stream, Console, Errors, Timers, Events, URL, Http, QueryString,...
- Un **module peut exporter** : une String, un nombre, un objet, une fonction, une classe,...

NPM

(Node Package Management)

- **NPM** est un outil qui permet de **gérer les modules et leurs dépendances**
- NPM fonctionne en **ligne de commande**
- NPM permet de rechercher, installer et gérer les modules
- **NPM est installé avec NodeJS** et on trouve toutes les infos sur <https://www.npmjs.com/>

```
I:\nodeProjets>  
I:\nodeProjets>npm -v  
10.8.1
```

Modules communautaires

- Pour ajouter un module qui n'est pas installé par défaut dans NodeJS

npm install <nom_du_module>

- Ce module est installé dans un dossier **node_modules** dans le dossier du projet
- **Avant d'installer** des modules dans un nouveau projet

npm init

pour créer un fichier **package.json**

npm init

```
Node.js command prompt
E:\nodeProjects\debutant>dir
Le volume dans le lecteur E s'appelle DATA
Le numéro de série du volume est E48F-F0C4

Répertoire de E:\nodeProjects\debutant

16-02-21 13:55 <DIR>      .
16-02-21 13:55 <DIR>      ..
                0 fichier(s)          0 octets
                2 Rép(s) 114.098.974.720 octets libres

E:\nodeProjects\debutant>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (debutant)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to E:\nodeProjects\debutant\package.json:

{
  "name": "debutant",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
E:\nodeProjects\debutant>
```

Fichier package.json

- Ce fichier au format **JSON** (JavaScript Object Notation) contient les informations du projet
 - Nom, description, version du projet, mots-clés,...
 - Point d'entrée (fichier JS)
 - Dépendances (production et développement) avec numéros de version
 - Tests
 - Auteur, Licences,...
 - etc
- Voir : <https://docs.npmjs.com/>

Commandes NPM

- **npm init** : initialiser un nouveau projet et créer le fichier package.json
- **npm install <nom_du_module>** : installer un module au niveau du projet
- **npm uninstall <nom_du_module>** : désinstaller un module
- **npm update <nom_du_module>** : mise à jour d'un module
- **npm config <parametre>** : configurer NPM

Options NPM

- Les commandes install, uninstall, update avec l'option **-g** traitent les modules au niveau **global** (pour tous les projets) plutôt que local
- L'option **--save** ajoute le module dans le fichier package.json (**dependencies**)
- L'option **--save-dev** ajoute le module dans le fichier package.json (**devDependencies**)
- Les **versions des modules** sont notées dans **package.json** (numérotation à 3 chiffres)