Installation de NodeJS

Joseph AZAR – Sept 2024

Node.js suit un calendrier de publication et adopte une politique de support à long terme (LTS: Long Term Support). Le calendrier de publication est basé sur la norme Semantic Versioning (https://semver.org/).

La politique de publication de Node.js indique qu'il existe deux versions majeures de Node.js par an, une en avril et une en octobre. Les versions majeures incluent des modifications d'API non conformes ou incompatibles, bien que le projet Node.js essaie de minimiser le nombre et l'impact des modifications de rupture afin de réduire les perturbations pour les utilisateurs.

Les versions majeures paires de Node.js sont promues en LTS après 6 mois. Les versions paires sont prévues pour avril et promues en LTS en octobre. Les versions LTS sont prises en charge pendant 30 mois. **Il est recommandé d'utiliser les versions LTS de Node.js** pour les applications de production. L'objectif de la politique LTS est de fournir une stabilité aux utilisateurs et également de fournir un calendrier prévisible des versions afin que les utilisateurs puissent gérer leurs mises à jour de manière appropriée. Toutes les versions LTS de Node.js reçoivent des noms de code, nommés d'après des éléments. Node.js 16 porte le nom de code Maintenance LTS **Gallium**. Node.js 18 porte le nom de code Active LTS **Hydrogen**.

Download Node.js®

Download Node.js the way you want.

Package Manager | Prebuilt Installer | Prebuilt Binaries | Source Code

Install Node.js v20.17.0 (LTS) on macOS using nvm

```bash
# installs nvm (Node Version Manager)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash

# download and install Node.js (you may need to restart the terminal)
nvm install 20

# verifies the right Node.js version is in the environment
node -v # should print `v20.17.0`

# verifies the right npm version is in the environment
npm -v # should print `10.8.2`
```

Bash

Copy to clipboard

Package managers and their installation scripts are not maintained by the Node.js project.
If you encounter issues, please reach out to the package manager's maintainers.

Node.js includes npm (10.8.2) ↗.

Read the changelog for this version ↗

Read the blog post for this version ↗

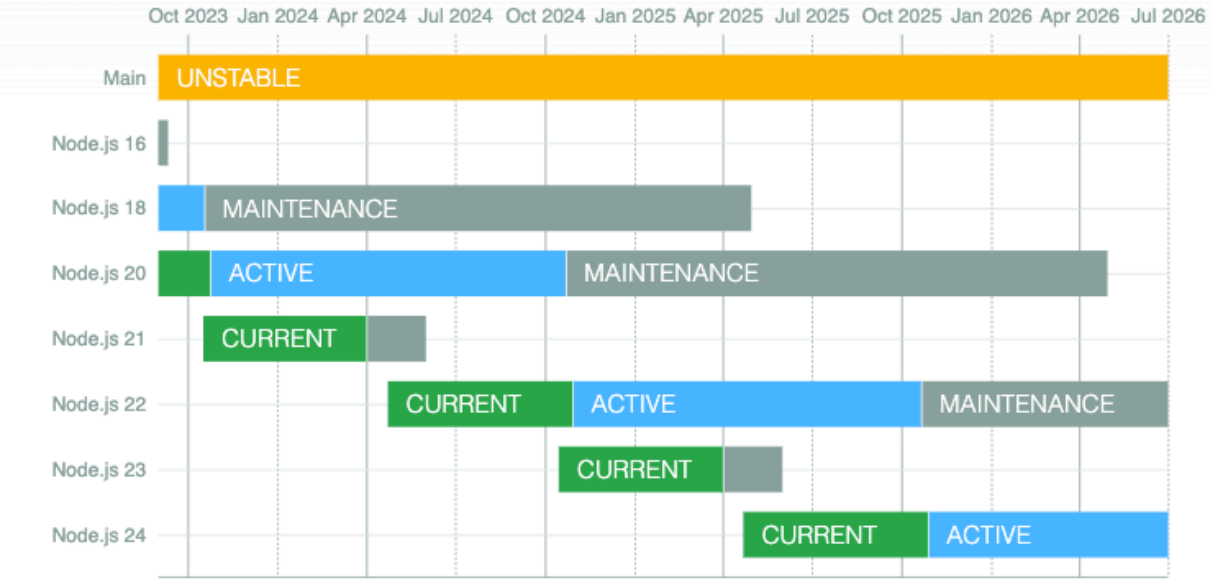
Learn how to verify signed SHASUMS ↗

Check out other community supported package managers ↗

Node.js Releases

Major Node.js versions enter *Current* release status for six months, which gives library authors time to add support for them. After six months, odd-numbered releases (9, 11, etc.) become unsupported, and even-numbered releases (10, 12, etc.) move to *Active LTS* status and are ready for general use. *LTS* release status is "long-term support", which typically guarantees that critical bugs will be fixed for a total of 30 months. Production applications should only use *Active LTS* or *Maintenance LTS* releases.

Release Schedule



Looking for latest release of a version branch?

Node.js Version	Codename	Release Date	npm	
v22.7.0	-	2024-08-21	v10.8.2	Releases Changelog Docs
v21.7.3	-	2024-04-10	v10.5.0	Releases Changelog Docs
v20.17.0	Iron	2024-08-21	v10.8.2	Releases Changelog Docs
v19.9.0	-	2023-04-10	v9.6.3	Releases Changelog Docs
v18.20.4	Hydrogen	2024-07-08	v10.7.0	Releases Changelog Docs
v17.9.1	-	2022-06-01	v8.11.0	Releases Changelog Docs
v16.20.2	Gallium	2023-08-08	v8.19.4	Releases Changelog Docs
v15.14.0	-	2021-04-06	v7.7.6	Releases Changelog Docs
v14.21.3	Fermium	2023-02-16	v6.14.18	Releases Changelog Docs
v13.14.0	-	2020-04-29	v6.14.4	Releases Changelog Docs
v12.22.12	Erbium	2022-04-05	v6.14.16	Releases Changelog Docs
v11.15.0	-	2019-04-30	v6.7.0	Releases Changelog Docs
v10.24.1	Dubnium	2021-04-06	v6.14.12	Releases Changelog Docs
v9.11.2	-	2018-06-12	v5.6.0	Releases Changelog Docs
v8.17.0	Carbon	2019-12-17	v6.13.4	Releases Changelog Docs
v7.10.1	-	2017-07-11	v4.2.0	Releases Changelog Docs
v6.17.1	Boron	2019-04-03	v3.10.10	Releases Changelog Docs
v5.12.0	-	2016-06-23	v3.8.6	Releases Changelog Docs
v4.9.1	Argon	2018-03-29	v2.15.11	Releases Changelog Docs
v0.12.18	-	2017-02-22	v2.15.11	Releases Changelog Docs

Les versions majeures impaires sont publiées en octobre et ne sont prises en charge que pendant 6 mois. Les versions impaires sont censées être utilisées pour essayer de nouvelles fonctionnalités et tester le chemin de migration, mais ne sont généralement pas recommandées pour une utilisation dans les applications de production.

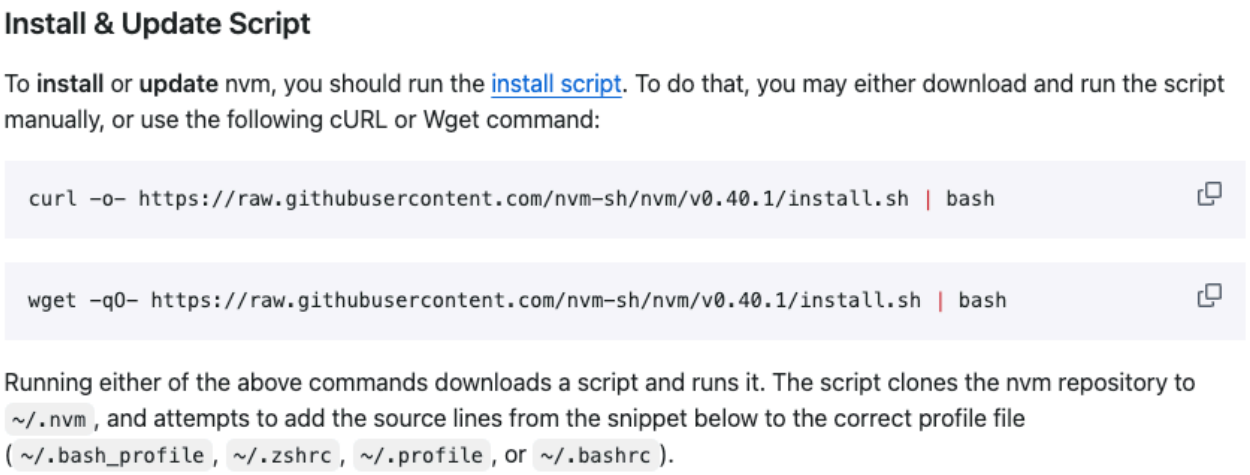
Le groupe de travail sur la publication de Node.js (Release Working Group) a autorité sur le calendrier et les processus de publication de Node.js. Le calendrier de publication de Node.js et la documentation sur les politiques sont disponibles à l'adresse *https://nodejs.org/fr/about/releases/*

Installation de Node.js avec nvm

- Installation de NVM sur Windows: https://www.freecodecamp.org/news/nvm-for-windows-how-to-download-and-install-node-version-manager-in-windows-10/
- Installation de NVM sur MacOS/Linux: https://www.freecodecamp.org/news/node-version-manager-nvm-install-guide/
- NVM Github: https://github.com/nvm-sh/nvm

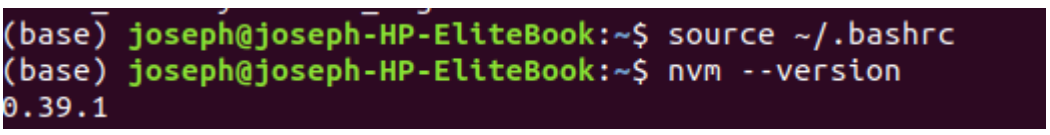
Ce qui suit expliquera comment installer Node.js à l'aide du gestionnaire de versions node (**nvm: node version manager**). Au moment de la rédaction, nvm est un projet d'incubation de la Fondation OpenJS et fournit un moyen simple d'installer et de mettre à jour les versions de Node.js. Tout d'abord, nous devons installer nvm. nvm fournit un script qui gère le téléchargement et l'installation de nvm. Saisissez la commande suivante dans votre terminal pour exécuter le script d'installation **nvm**:

```
curl -o-
https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh |
bash
```



nvm tentera automatiquement de s'ajouter à votre chemin (***path***). Fermez et rouvrez votre Terminal pour vous assurer que les changements ont eu lieu. Ensuite, entrez la commande suivante pour répertorier la version nvm que nous avons installée ; cela confirmera également que nvm est disponible dans notre chemin :

```
nvm --version
```



Pourquoi utiliser NVM (Node Version Manager) [https://github.com/nvm-sh/nvm] ? Vous pouvez utiliser le gestionnaire de versions Node.js (NVM) pour gérer votre installation Node.js et gérer une ou plusieurs versions de Node.js sur votre ordinateur. L'avantage

d'utiliser un gestionnaire de version est que vous pouvez tester les nouvelles versions de Node.js au fur et à mesure de leur sortie tout en ayant des versions plus anciennes et plus stables installées en cas de problèmes de compatibilité.

Vous manquez la commande **nvm** après avoir exécuté le script d'installation ?
Si vous utilisez **macOS**, il vous manque peut-être un fichier ~/**.bash_profile .**
Pour résoudre ce problème, vous pouvez exécuter
touch ~/.bash_profile
dans votre ligne de commande et relancer le script d'installation.
Si le problème persiste après cela, vous pouvez ouvrir le fichier .bash_profile existant (à l'aide de votre éditeur de texte préféré) et y ajouter la ligne suivante :
source ~/.bashrc
Assurez-vous que le fichier **.bashrc** existe.
Si vous rencontrez toujours des problèmes, vous pouvez jeter un coup d'œil à ce problème pour trouver une discussion sur le problème et une collection de solutions possibles:
https://github.com/nvm-sh/nvm/issues/576
https://github.com/nvm-sh/nvm

Pour installer Node.js 16, 18 ou 20, nous utilisons la commande **$ nvm install**. Nous pouvons fournir soit la version spécifique que nous souhaitons installer, soit le numéro de version principale. Si nous spécifions uniquement le numéro de version majeure, nvm installera la dernière version de cette ligne de version majeure. Saisissez la commande suivante pour installer la dernière version de Node.js 16:

```
nvm install 16
```

La dernière version de Node.js 16 devrait maintenant être installée et disponible dans votre chemin. Vous pouvez le confirmer en entrant la commande suivante :

```
node --version
```

nvm installera également la version de npm fournie avec la version Node.js que vous avez installée. Entrez ce qui suit pour confirmer quelle version de npm est installée :

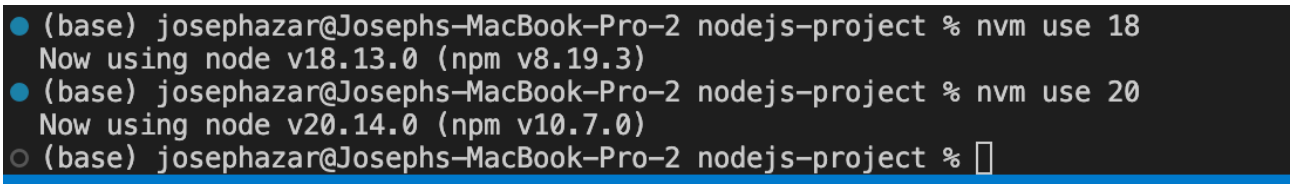
```
npm --version
```

nvm facilite l'installation et le basculement entre plusieurs versions de Node.js. Nous pouvons entrer la commande suivante pour installer et passer à la dernière version de Node.js 18:

```
nvm install 18
```

Une fois les versions installées, nous pouvons utiliser la commande **$ nvm use** pour basculer entre elles :

```
nvm use 16
```



Utilisez la dernière version LTS avec: **nvm use --lts**

```
nvm use --lts
```

nvm est un gestionnaire de version pour Node.js sur des plates-formes de type UNIX et prend en charge les shells compatibles POSIX. POSIX est un ensemble de normes pour la compatibilité des systèmes d'exploitation, défini par l'IEEE Computer Society.

Dans la première étape du document, nous avons téléchargé et exécuté le script d'installation de nvm. Le script d'installation nvm effectue les opérations suivantes :

- Il clone le référentiel nvm GitHub (https://github.com/nvm-sh/nvm) vers **~/.nvm/**.
- Il tente d'ajouter les lignes source pour importer et charger nvm dans le fichier de profil approprié, où le fichier de profil est soit **~/.bash_profile**, **~/.bashrc**, **~/.profile** ou **~/.zshrc**.

nvm prend en charge les alias qui peuvent être utilisés pour installer les versions LTS de Node.js. Par exemple, vous pouvez utiliser la commande **$ nvm install --lts** pour installer la dernière version de LTS.

Pour désinstaller une version de Node.js, vous pouvez utiliser la commande **$ nvm uninstall**. Pour modifier la version par défaut de Node.js, utilisez la commande

$ nvm alias default <version>. La version par défaut est la version qui sera disponible par défaut lors de l'ouverture de votre Terminal.

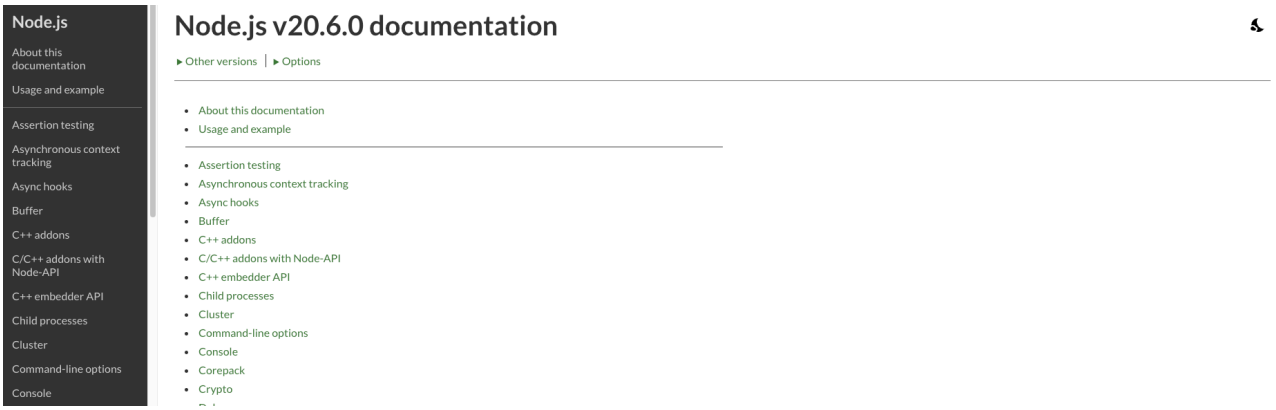
Accéder à la documentation de l'API Node.js

Le projet Node.js fournit une documentation de référence complète sur l'API (Application Programming Interface). La documentation de l'API Node.js est une ressource essentielle pour comprendre quelles APIs sont disponibles dans la version de Node.js que vous utilisez. La documentation Node.js décrit également comment interagir avec les APIs, y compris les arguments qu'une méthode accepte et la valeur de retour de la méthode.

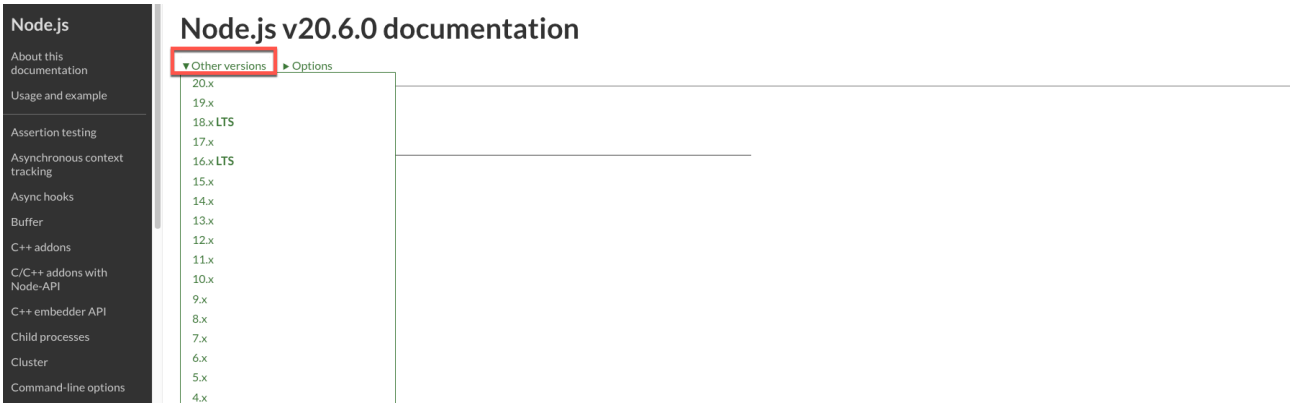
Ce qui suit montre comment accéder et naviguer dans la documentation de l'API Node.js.

Tout d'abord, accédez à https://nodejs.org/api/ dans votre navigateur.

Attendez-vous à voir la documentation de l'API Node.js pour la version la plus récente de Node.js :



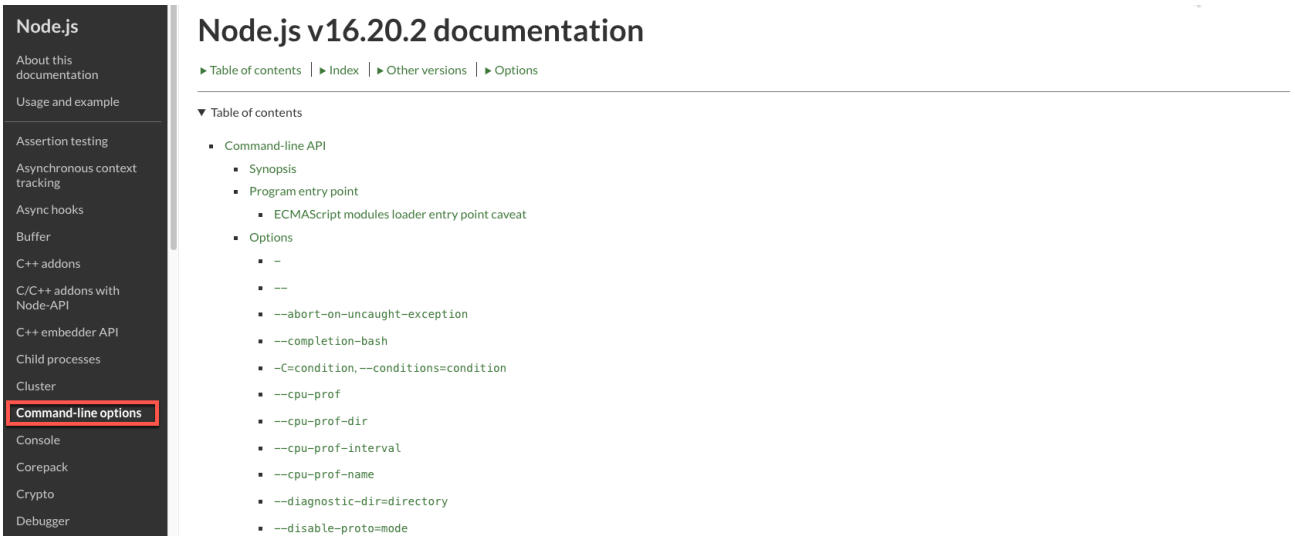
Survolez le lien “***Other versions***” et attendez-vous à voir les autres lignes de version de Node.js répertoriées. Voici comment vous pouvez changer la version de Node.js pour laquelle vous consultez la documentation :



Supposons maintenant que nous voulions trouver la documentation de la méthode **fs.readFile()**. La méthode **fs.readFile()** est exposée via le module principal du système de fichiers (File system). Nous devons d'abord localiser et cliquer sur **File System** dans le volet de navigation de gauche. En cliquant sur Système de fichiers, nous accéderons à la table des matières de la documentation de l'API du module de base du système de fichiers :

Faites défiler jusqu'à ce que vous trouviez la méthode **fs.readFile()** répertoriée dans la table des matières. Lorsque vous recherchez une API spécifique, il peut être utile d'utiliser la fonction de recherche de votre navigateur pour localiser la définition de l'API. Cliquez sur le lien **fs.readFile()** dans la table des matières. Cela ouvrira la définition de l'API.

Maintenant, cliquez sur “Command line options” dans le volet de navigation de gauche. Cette page détaille toutes les options de ligne de commande disponibles qui peuvent être transmises au processus Node.js :



Cette documentation de l'API Node.js est une ressource de référence vitale lors de la création d'applications Node.js. La documentation est spécifique à chaque version de Node.js.

La documentation de l'API détaille l'utilisation des API Node.js, notamment :

- Les paramètres acceptés et leurs types
- La valeur et le type que l'API renvoie

Dans certains cas, la documentation fournira des informations supplémentaires, y compris un exemple d'utilisation ou un exemple de code démontrant l'utilisation de l'API.

Notez qu'il existe des API non documentées. Certaines API Node.js sont intentionnellement non documentées. Certaines des API non documentées sont considérées comme internes uniquement et ne sont pas destinées à être utilisées en dehors de l'environnement d'exécution principal de Node.js.

La documentation de l'API détaille également la stabilité des API. Le projet Node.js définit et utilise les trois indices de stabilité suivants :

0 – Obsolète (Deprecated) : l'utilisation de ces API est déconseillée. Des avertissements peuvent être émis lors de l'utilisation de ces API. Les API obsolètes seront également répertoriées sur https://nodejs.org/dist/latest-v14.x/docs/api/deprecations.html.

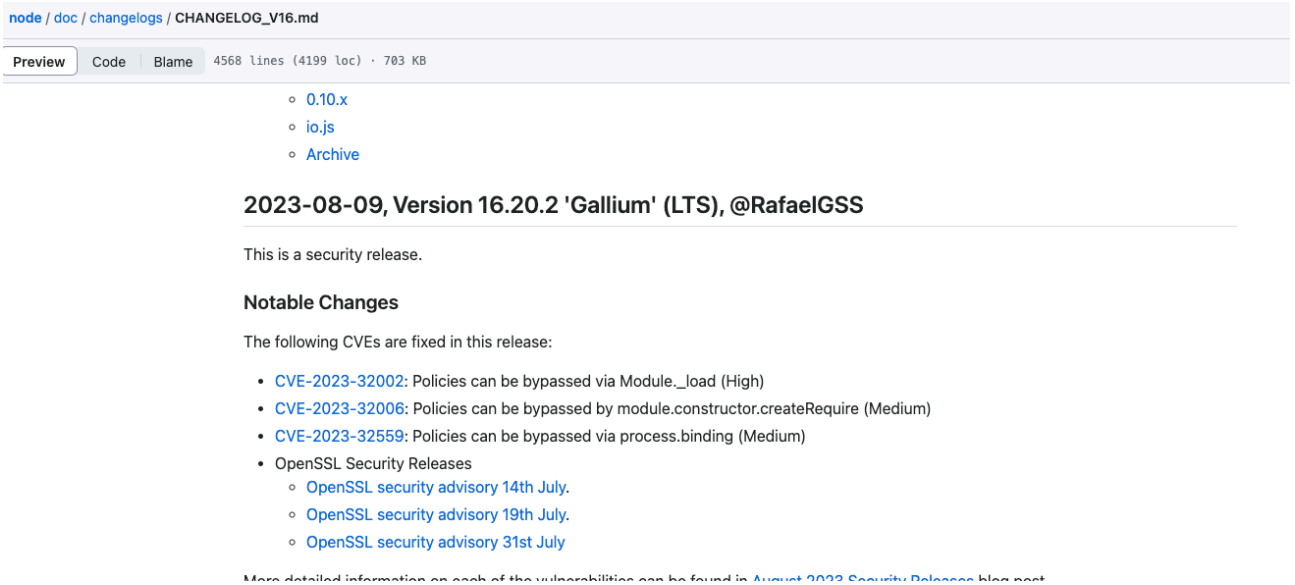
1 – Expérimental : Ces API ne sont pas considérées comme stables et peuvent faire l'objet de modifications non rétrocompatibles. Les API expérimentales ne sont pas soumises aux règles de gestion des versions sémantiques. Ces API doivent être utilisées avec prudence, en particulier dans les environnements de production.

2 – Stable : Avec des API stables, le projet Node.js tentera d'assurer la compatibilité.

La documentation Node.js est maintenue par le projet Node.js dans le référentiel principal Node.js. Toute erreur ou suggestion d'amélioration peut être signalée en tant que problème sur https://github.com/nodejs/node.

Le projet Node.js gère un fichier **CHANGELOG.md** pour chaque ligne de version de Node.js, détaillant les commits individuels qui ont lieu dans chaque version. Le fichier **CHANGELOG.md** pour Node.js 16 se trouve à l'adresse https://github.com/nodejs/node/blob/master/doc/changelogs/CHANGELOG_V16.md

Ce qui suit est un extrait du fichier Node.js 16 **CHANGELOG.md** :



Le projet Node.js s'efforce de mettre en évidence les changements notables dans chaque version. Le fichier CHANGELOG.md indique quels commits ont été déterminés comme étant SEMVER-MINOR selon la norme Semantic Versioning (https://semver.org/). Les entrées marquées comme SEMVER-MINOR indiquent généralement des ajouts de fonctionnalités. Le fichier CHANGELOG.md indiquera également quand une version est considérée comme une version de sécurité (résolution d'un problème de sécurité). Dans le cas d'une version de sécurité, la section “Notable Changes” commencera par la phrase “This is a security release”.

Les fichiers Node.js CHANGELOG.md peuvent être utilisés comme référence lors de la mise à niveau de Node.js, pour aider à comprendre quelles mises à jour et modifications sont incluses dans la nouvelle version.