Ryan Dahl a eu la brillante idée de prendre le moteur Javascript V8, c'est celui qu'on trouve dans le navigateur Chrome, et de l'utiliser en dehors du navigateur. Il a créé la plateforme Node.js !

Node.js est très fréquemment utilisé pour écrire des services côté serveur appelés API (Application Programming Interface)

Et au final, on peut dire que Node.js représente une alternative à des langages serveur comme PHP, Java ou Python.

Cette alternative est devenue populaire dans pas mal d'entreprises et a même révolutionné pas mal de choses dans des grosses entreprises comme PayPal.

PayPal qui utilisait du Java a décidé de tester Node.js sur une petite partie de son code et les résultats ont été plutôt bluffants.

- Il a fallu 2 développeurs Node.js au lieu de 5 développeurs Java pour construire la même chose
- Les développeurs de l'équipe Node.js ont fait le même travail que l'équipe Java en la moitié du temps
- 33% de lignes de code en moins et 40% de fichiers en moins en Node.js par rapport à Java
- La version Node.js est 35% plus rapide que celle de Java

Alors bien sûr, il ne faut pas s'emballer non plus et ces résultats peuvent être dus à plein d'autres facteurs, peut-être que les développeurs Node.js étaient excellents et les devs Java étaient nazes, on n'en sait rien.

Mais ça fait tout de même réfléchir et ça place clairement Node.js comme une alternative de choix pour le back-end de nos applis web.

# Définitions

Maintenant je vais t'expliquer les avantages de Node.js mais j'ai pas envie de te balancer plein de termes techniques sans que tu les comprennes.
Alors accroche-toi bien ! C'est le moment définition !

Je vais aborder pas mal de notions qui te sont peut-être inconnues : On va parler de RTA, de SPA, de Single Thread et Multi Thread et de systèmes Blocking et Non Blocking.

**Alors déjà c'est quoi RTA ?**

RTA ça veut dire Real Time Applications, ce sont les applications en temps réel, ce sont ces applications qui ont besoin de se mettre à jour super fréquemment.

Je te donne un exemple : les messageries instantanées style WhatsApp sont des RTA. Tu as besoin de connaitre les nouveaux messages immédiatement parce-que si tu reçois les messages de tes correspondants 10 minutes plus tard ça en ferait une très mauvaise appli….

**Et c'est quoi SPA ?**

Ce sont les initiales de Single Page Applications. Ce sont des applis dans lesquelles il n'y a qu'une page html et le contenu de cette page change en fonction des actions de l'utilisateur. C'est différent du modèle ou il y a plusieurs pages html. En général, on utilisera des frameworks comme Angular, Vue ou React pour créer ce genre d'applications web.

**Parlons maintenant de Multithread et Single thread !**

Multithread c'est la capacité à effectuer plusieurs tâches en même temps, en parallèle et bien sûr Single thread c'est le fait que tu ne t'occupes que d'une tâche à la fois.

En gros tu peux représenter ça dans ta tête en imaginant qu'un système multithread a plusieurs petits bonhommes et que tu peux confier à chacun d'eux une tâche et ils travaillent en parallèle, donc au final ton système est multitâche.
Alors que le système single thread lui n'a qu'un petit bonhomme et tu peux donc lui confier qu'une seule tâche à la fois.

# Pourquoi Node.js ?? Les avantages de Node.js

## Super adapté aux RTA et SPA

Node.js est un système single thread non bloquant !

En effet comment fonctionne Node.js ? Si un client 1 fait une requête d'un fichier au serveur alors il lance cette requête sans attendre le résultat.
Il n'attend pas car il n'est pas bloquant. Si un autre client vient faire une autre requête, il est tout de suite capable de traiter cette requête également.

Au final ça rend les choses super rapides pour les applis qui font beaucoup de requêtes de fichiers car Node.js est capable de gérer énormément de requêtes en parallèle sans les faire attendre les unes les autres et c'est grâce à ça qu'il est particulièrement bien adapté aux SPA et RTA.
Les RTA font énormément de requêtes pour sans cesse mettre à jour les données de l'appli.

Et pour les SPA, la complexité du code réside souvent du côté client. Il n'y a pas généralement pas d'opérations complexes du côté serveur, les clients font pas mal de requêtes au serveur pour récupérer les données et le serveur se contente de procurer ces données pour alimenter l'appli.

Maintenant je veux que tu saches quand même que le fait que Node.js soit single thread alors que la plupart des systèmes classiques sont multithreads peut-être un inconvénient pour certains projets. Si tu as des tâches qui sont longues à traiter du côté serveur, des tâches qui demandent beaucoup de ressources car il y a beaucoup de calcul par exemple comme l'encodage vidéo ou la manipulation d'image alors Node.js ne sera pas du tout adapté ! Il est single thread et si son petit bonhomme est occupé sur une tâche alors il ne pourra pas s'occuper de toutes les autres tâches qui attendent, ce qui rendra tout le système très lent.

## Flexible

Node.js est très light comme plateforme et n'a pas beaucoup de fonctionnalité déjà intégré. C'est toi qui choisis quels sont les modules que tu veux lui greffer pour l'utiliser.

Il n'y a pas de conventions strictes sur comment tu dois t'y prendre et ça te laisse pas mal de marge de manœuvre pour décider de tout. Ce qui fait que tu peux commencer avec quasiment rien et avancer petit à petit dans la direction que tu penses être la meilleure au fur et à mesure que tu avances dans ton projet.

Pour avancer, tu disposes d'un écosystème très large de librairies open-source que tu pourras utiliser directement dans ton projet avec le NPM (Node Package Manager). Pas

besoin de réinventer la roue ! Il y a toujours des gens qui se sont casse la tête pour toi avant toi.

## C’est du Javascript !

Et oui mon pote ! Si tu connais déjà le Javascript et bien ça sera relativement facile de commencer avec Node.js car c’est le même langage… Notre copain Javascript ! Donc pas besoin d’apprendre un nouveau langage.

Et surtout le fait que ça soit du Javascript c’est très pratique pour les développeurs Full-stack ! C’est souvent un peu déstabilisant de coder avec un langage du côté front-end et en autre langage du côté back-end.

Et bien avec Node.js ce casse-tête c’est fini parce que tu restes en Javascript partout… Même au niveau organisation de ton code, c’est mieux ! Tu pourras avoir les mêmes conventions de noms, les mêmes bonnes pratiques dans tout le code de ton projet, aussi bien en Front qu’en Back !