المدرسة الحسنية للأشغال العمومية
ECOLE HASSANIA DES TRAVAUX PUBLICS

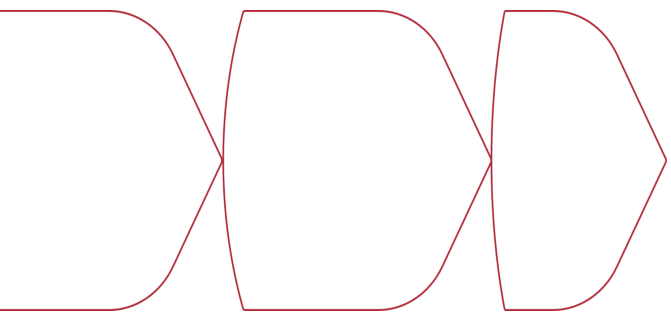
Introduction à Node.js

Comprendre les bases du développement serveur avec JavaScript

Slimani Mohamed Amine

EHTP

February 10, 2025



Sommaire

Qu'est-ce que Node.js ?

- **Définition** : Node.js est une plateforme JavaScript côté serveur basée sur le moteur V8 de Chrome.
- **Objectif** : Permettre de créer des applications web rapides et scalables.
- **Avantages** : Modèle non bloquant (asynchrone), grande communauté, riche écosystème de modules.

Pourquoi utiliser Node.js ?

- **Performance** : Modèle non bloquant pour des applications rapides.
- **Scalabilité** : Facile à scaler horizontalement et verticalement.
- **JavaScript partout** : Utiliser le même langage côté client et serveur.

Composants de Node.js

- **Moteur V8** : Exécute le code JavaScript.
- **Libuv** : Bibliothèque pour la gestion des entrées/sorties asynchrones.
- **NPM (Node Package Manager)** : Gestionnaire de packages pour Node.js.

Architecture de Node.js

- **Boucle d'événements (Event Loop)** : Gère les opérations asynchrones.
- **Thread Pool** : Exécute les tâches bloquantes en parallèle.
- **Modules natifs** : Fournit des fonctionnalités de base (ex : ‘http‘, ‘fs‘).

Exemple de code Node.js

Serveur HTTP simple

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

const server = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello, World!\n');
});

Comment Code
server.listen(3000, '127.0.0.1', () => {
    console.log('Server running at http://127.0.0.1:3000/');
});
```

Bonnes pratiques

- **Utiliser des modules** : Diviser le code en modules réutilisables.
- **Gestion des erreurs** : Utiliser des middlewares pour capturer les erreurs.
- **Optimisation des performances** : Utiliser des techniques de caching et de compression.

Outils pour travailler avec Node.js

- **Express.js** : Framework web pour Node.js.
- **Socket.IO** : Bibliothèque pour les applications en temps réel.
- **PM2** : Gestionnaire de processus pour Node.js.

Exemple d'application avec Express.js

Application web simple

```
const express = require('express');
const app = express();

Comment Code
app.get('/', (req, res) => {
    res.send('Hello, World!');
});

Comment Code
app.listen(3000, () => {
    console.log('Server running at http://localhost:3000/');
});
```

Défis de Node.js

- **Gestion des erreurs** : Les erreurs non capturées peuvent faire planter l'application.
- **Performance CPU-intensive** : Node.js n'est pas idéal pour les tâches gourmandes en CPU.
- **Compatibilité** : Certains modules ne sont pas compatibles avec les dernières versions de Node.js.

Pourquoi c'est important ?

- Node.js est une plateforme puissante pour créer des applications web rapides et scalables.
- Il permet d'utiliser JavaScript côté serveur, ce qui simplifie le développement full-stack.
- Comprendre Node.js est essentiel pour les développeurs web et les architectes logiciels.

Résumé

Node.js est une plateforme révolutionnaire pour le développement d'applications serveur rapides et scalables. Développez mieux, plus vite, et avec JavaScript partout !