Programme de formation

Node.js avancé

Optimisez la performance et la qualité de vos applications Node.JS

Durée

3 jours / 21 heures

Description

Node.JS est un projet open-source se basant sur le moteur V8 de Chrome, qui permet d'exécuter du JavaScript côté serveur, contrairement à ce qu'on a l'habitude de voir avec le JavaScript côté client. Il s'agit d'un interpréteur JavaScript exécutable, et enrichissant le langage avec sa propre API. Sa spécificité vient de son API, entièrement orientée vers le non bloquant, qui permet d'écrire des applications avec d'excellents temps de réponse. Cette formation Node.js perfectionnement de 3 jours, vous permettra de maîtriser les concepts avancés de Node.Js , d'un point de vue objets avancés JavaScript, programmation fonctionnelle, paradigme asynchrone, ES6 mais aussi performance et qualité des applications Node.JS .

Objectifs pédagogiques

- Maîtriser le cœur de la technologie Node.JS
- Optimiser les performances de leurs applications
- Améliorer la qualité des applications

Public

Développeurs, Architectes ou chefs de projet

Pré-requis

Maîtrise de JavaScript et 1ères expériences avec Node.JS

Méthodes pédagogiques

30 % théorie / 70 % pratique

Profil intervenant

L'ensemble de nos formations sont animées par des formateurs expérimentés possédant une expérience terrain éprouvée.

Modalités d'évaluation

L'évaluation des acquis se fait tout au long de la session au travers d'ateliers de mise en pratique des notions et concepts abordés pendant la formation.

Programme

Rappels sur node.js

- Installation et utilisation du REPL : présentation nvm&co
- JavaScript côté serveur : démonstrations et premiers travaux pratiques
- La "single-threadedeventloop"
- API non bloquante : intérêts
- Aller plus loin avec les objets JavaScript?: Object.create,Object.defineProperty
- La programmation fonctionnelle?: map, reduce, currying (illustration avec lodash)

La programmation asynchrone

- Dompter le paradigme asynchrone?
- Les différentes API : callback et librairies associées, fibers, promesses
- Les avantages et pièges à éviter
- Gérer la soupe de callbacks avec Async

Les modules node.js

- Description
- Fonctionnement
- Structure

Ecrire un serveur avec node.js

- Présentation
- Démonstration

Les librairies d'accès aux bases de données

- Présentation
- NoSQL: MongoDB et Redis
- Modélisation avec NoSQL
- Différentes utilisations de Redis
- Implémentation du modèle

Communication en temps réel

- Définition et problématiques
- Les technologies à disposition
- Intégration des WebSockets HTML5 avec Socket.IO

Communication inter-process en temps réel

- Le pub/sub avec Redis
- AMQP dans Node JS
- Présentation RabbitMQ et ZeroMQ
- Event-loop distribuée

Les tests avec nodejs

- Tests unitaires avec Mocha (atelier)
- Tests fonctionnels avec les headless browsers
- Intégration avec npm

Travaux pratiques

- Exercices sur les server-sideevents, websockets avec socket.io

Gestion de la performance avec javascript et node.js

- Écrire du JavaScript performant pour V8 : les bonnes pratiques

Gestion de la mémoire : la pile et la mémoire totale, comment les gérer

- Anticiper et trouver les fuites mémoire
- Effectuer des calculs lourds : pool de workers, amqp…
- Les Cluster
- Utiliser tous les processeurs de sa machine
- Cluster et données partagées
- La solution haute performance Redis
- Bonus spécial troll
- Comparaison avec l'équivalent Apache/PHP

Qualité

- Déboguer son application : utilisation du débuggeur v8
- Profiling : état des lieux, outils
- Bonnes pratiques : les meilleurs outils de test, contrôle de qualité du
- code, conventions…
- Discussion libre

Conclusion et conseils

- Synthèse des bonnes pratiques de développement NodeJS
- Veille : comment faire le tri dans les milliers de modules disponibles ?
- Le futur avec ES6?: let, const, arrows, classes, destructuration, proxy, observation, generators, comprehensivelists, collections, promesses…
- Le futur présent : fonctionnalités ES6 déjà utilisables dans Node