Optimiser les Applications Web avec WebAssembly et Emscripten

Lien : https://innov-systems.com/formation/optimiser-les-applications-web-avec-webassembly-et-emscripten

DURÉE
4 jours (28h)

RÉFÉRENCE
TWC256

CATÉGORIE
Conception et Développement Web Pour Mobiles

Innov Systems

OBJECTIFS DE LA FORMATION

À l'issue de cette formation, vous serez capable de :

- Comprendre les fondements, la logique et les avantages du standard WebAssembly (WASM) dans l'écosystème web moderne
- Maîtriser la création, la compilation et l'intégration de modules WebAssembly dans des applications JavaScript, Node.js et navigateur
- Développer des modules performants avec AssemblyScript, Rust et C/C++ via Emscripten
- Savoir porter, déboguer et optimiser des bibliothèques natives pour un usage web
- Explorer les bonnes pratiques et les outils de performance liés à l'utilisation de WASM dans des projets réels

POUR QUI ?

- Développeurs web front-end et back-end
- Architectes techniques
- Ingénieurs logiciels
- Responsables techniques souhaitant optimiser leurs applications web

Programme détaillé

1 / COMPRENDRE LES ENJEUX ET LES PRINCIPES DE WEBASSEMBLY

- Les limites des langages interprétés et la promesse du code natif sur le web
- Historique et architecture du standard W3C WebAssembly
- Concepts fondamentaux : portabilité, sandboxing, sécurité et performance
- Les acteurs clés : Bytecode Alliance, WASI, navigateurs compatibles

2 / STRUCTURE ET CYCLE DE VIE D'UN MODULE WASM

- Structure interne d'un module binaire (sections, imports, exports)
- Représentation textuelle WAT et compilation vers WASM
- Interprétation, exécution et intégration dans un environnement Node.js
- Travaux pratiques : création d'un premier module simple en WAT et exécution avec Node

3 / INTERFACES ET API JAVASCRIPT DE WEBASSEMBLY

- Chargement et instanciation de modules avec l'API WebAssembly
- Interaction entre WASM et JavaScript : mémoire partagée, fonctions import/export
- Gestion des erreurs et du typage
- Travaux pratiques : manipulation d'un module WASM dans une page web et mesure des performances

4 / DÉCOUVERTE DU LANGAGE ASSEMBLYSCRIPT

- Principes et positionnement du langage
- Installation et configuration de l'environnement AssemblyScript
- Compilation d'un projet vers WASM
- Travaux pratiques : création d'un module simple et intégration dans une application web

5 / GESTION DE LA MÉMOIRE ET OPTIMISATION EN ASSEMBLYSCRIPT

Innov Systems

- Compréhension du modèle mémoire de WASM
- Pointeurs, heap, garbage collection et limitations
- Stratégies d’optimisation et débogage de modules AssemblyScript
- Travaux pratiques : génération d’une visualisation graphique (ex. fractale de Mandelbrot)

6 / INTRODUCTION AU DÉVELOPPEMENT RUST VERS WASM

- Pourquoi Rust pour le WebAssembly ?
- Mise en place de l’environnement : wasm-pack et cargo
- Compilation et utilisation d’un module Rust dans un navigateur
- Travaux pratiques : création d’une librairie Rust exportant des fonctions de calcul haute performance

7 / INSTALLATION ET CONFIGURATION DE EMSCRIPTEN

- Présentation du SDK Emscripten et de ses outils (emcc, emmake, emconfigure)
- Processus de compilation d’un projet C/C++ en module WASM
- Les fichiers générés : .wasm, .js et .html
- Travaux pratiques : compilation et exécution d’un programme simple C en WASM

8 / INTÉGRATION AVANCÉE ENTRE C/C++ ET JAVASCRIPT

- Les appels inter-langages : ccall, cwrap, EXPORTED_FUNCTIONS
- Le système de fichiers virtuel et les variables d’environnement
- Gestion des entrées/sorties et optimisation de la taille du module
- Travaux pratiques : intégration d’une bibliothèque C mathématique dans une application web interactive

9 / OPTIMISATION, DEBUG ET MAINTENANCE DE MODULES WASM

- Techniques d’optimisation des performances
- Outils de profiling et d’analyse de la mémoire
- Bonnes pratiques de maintenance et d’évolution de projets WASM
- Travaux pratiques : analyse comparative entre exécution JavaScript et exécution WASM

Innov Systems

10 / EXÉCUTION HORS NAVIGATEUR ET RUNTIMES WASM

- Comprendre les runtimes : wasmtime, wasm3, Wazero
- Installation et utilisation des runtimes sur différents environnements
- Utilisation de WASI pour l'accès au système
- Travaux pratiques : exécution d'un module Rust sur wasmtime et comparaison de performance

11 / PORTAGE ET COMPILATION DE LIBRAIRIES EXISTANTES

- Adapter des bibliothèques natives vers le web : contraintes et compatibilités
- Configuration avancée d'Emscripten (Autoconf, Emmake, Makefile)
- Utilisation des options de compilation : MODULARIZE, EXPORTED_RUNTIME_METHODS, EXPORTED_FUNCTIONS
- Travaux pratiques : portage d'une bibliothèque open source C++ vers le web

12 / BONNES PRATIQUES ET PERSPECTIVES D'UTILISATION DE WEBASSEMBLY

- Stratégies de déploiement et intégration continue
- Compatibilité navigateur et rétrocompatibilité
- WebAssembly et l'avenir du développement full-stack : WASI, WASM Cloud, WebGPU
- Travaux pratiques : mini-projet de synthèse — intégration d'un module WASM optimisé dans une application complète (front + back)

Inov Systems

Approche pédagogique

- Support Ecrit et Projection
- Exposés Interactifs, Podcasts et Vidéos
- Brainstorming et Jeux de Rôle
- Cas Pratiques et Labs inclus pour leur impact opérationnel
- Test de Validation des Acquis des Connaissances

Prochaines dates programmées

24 au 27 Nov. 2026	Distanciel
12 au 15 Jan. 2027	Présentiel - Casablanca
09 au 12 Mars 2027	Présentiel - Casablanca
04 au 07 Mai 2027	Présentiel - Casablanca
29 Juin au 02 Juil. 2027	Présentiel - Casablanca

Autres dates possibles sur demande. Contactez-nous pour organiser une session intra-entreprise.

Réservation & Renseignements

Téléphone : +212 522 247 210
Email : contact@innov-systems.com
Web : https//www.innov-systems.com

Innov Systems — Tous droits réservés

Innov Systems