Smartfuture
Certifications | Training | Technology solutions

Formation

JavaScript – Programmation Avancée

Formation en présentiel et en ligne à la demande. Approfondissez JavaScript moderne (ES6+), maîtrisez les mécaniques avancées du langage (closures, prototypes, asynchronisme) et développez des applications web robustes et maintenables.

Formation 100% pratique | Exercices & cas d'usage | Formateurs expérimenté

4 jours 35 h	**Format** Atelier pratique
Modalité Présentiel ou en ligne	**Niveau** Intermédiaire

INSCRIPTION / RÉSERVATION

Je m'inscris maintenant

OBJECTIFS PÉDAGOGIQUES

- Maîtriser les concepts avancés du langage JavaScript (ES6+).
- Comprendre en profondeur le modèle d'exécution, l'event loop et l'asynchronisme.
- Structurer une base de code JavaScript professionnelle, claire et maintenable.
- Manipuler le DOM, les événements et les APIs du navigateur avec des techniques avancées.
- Réaliser un projet complet en JavaScript moderne en appliquant les bonnes pratiques.

PUBLIC CIBLE

- Développeurs ayant des bases en JavaScript et souhaitant progresser.
- Intégrateurs web évoluant vers un développement front-end avancé.
- Développeurs back-end voulant renforcer leurs compétences en JavaScript côté front.
- Personnes impliquées dans la maintenance ou la refonte d'applications JavaScript.

PRÉREQUIS

- Bonne maîtrise des bases de JavaScript (variables, fonctions, tableaux, objets).
- Connaissances de base en HTML/CSS.
- Expérience pratique sur un projet web simple est un plus.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Rappels structurés & JavaScript moderne (ES6+)

Rappel des fondamentaux utiles pour l'avancé

- Syntaxe de base et bonnes pratiques de lisibilité.
- Portée des variables avec var, let, const.
- Types primitifs et objets : rappel et nuances.

Syntaxe moderne ES6 et au-delà

- Fonctions fléchées (arrow functions) et leurs spécificités (this implicite).
- Template literals, interpolation et chaînes multi-lignes.
- Opérateurs spread et rest, décomposition (destructuring) d'objets et de tableaux.
- Paramètres par défaut, opérateur de coalescence nulle (??) et optionnel chaining (?.).

Fonctions avancées, closures et contexte d'exécution

Fonctions en tant que citoyens de première classe

- Fonctions comme valeurs : passage en paramètre, retour de fonctions.
- Fonctions d'ordre supérieur (higher-order functions).
- Utilisation avancée de map, filter, reduce pour transformer les données.

Closures et portée

- Comprendre la portée lexicale en JavaScript.
- Closures : définition, exemples concrets et cas d'usage (encapsulation, fonctions fabriques).
- Pièges fréquents liés aux closures et comment les éviter.

Contexte d'exécution et mot-clé this

- Appels simples, méthodes d'objet, fonctions fléchées et this.
- call, apply, bind : forcer le contexte.
- Cas pratiques : gestion de this dans des callbacks et écouteurs d'événements.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Objets, prototypes et classes ES6

Modèle objet et chaîne de prototypes

- Objets littéraux, propriétés, getters et setters.
- Notion de prototype, Object.create et chaîne de prototypes.
- Héritage prototypal : compréhension et exemples.

Classes ES6

- . Déclaration de classes, constructeur et méthodes d'instance.
- Héritage avec extends et utilisation de super.
- Propriétés et méthodes statiques.
- Patrons de conception simples (factory, singleton, strategy) en JavaScript.

Encapsulation et organisation du code

- Simulation de la visibilité privée (closures, symboles, champs privés).
- Structurer des modules d'objets pour un projet.

Asynchronisme et modèle d'exécution JavaScript

Event loop, call stack et file de messages

- Comment JavaScript exécute le code : pile d'appels, heap, queue.
- Timers, callbacks et APIs du navigateur.
- Micro-tâches vs macro-tâches (promises, mutation observers, timers…).

Promises

- Création et consommation de Promises (then, catch, finally).
- Chaînage de Promises et gestion d'erreurs.
- Promise.all, Promise.race, Promise.allSettled.

async/await et accès aux APIs

- Réécrire du code asynchrone lisible avec async/await.
- Gestion des erreurs (try/catch) et bonnes pratiques.
- Consommer une API REST avec fetch.
- Cas pratiques : formulaires, recherche en temps réel, auto-complétion.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Manipulation avancée du DOM & APIs du navigateur

DOM avancé

- Sélection, création et insertion dynamique d'éléments.
- Modification de classes, styles et attributs de manière performante.
- Templates HTML, fragments de document et optimisation.

Gestion des événements

- Modèle d'événements (capture, bubbling).
- Délégation d'événements et code maintenable.
- Gestion des formulaires et validation côté client.

APIs utiles du navigateur

- Web Storage : localStorage, sessionStorage.
- API History et navigation côté client (notions).
- API Fetch, File API, aperçu d'autres APIs selon le temps (Notifications…).

Modules JavaScript, outillage et qualité de code

Modules JavaScript

- Modules ES (import / export) : organisation du code en fichiers.
- Différences avec les anciens patterns (IIFE, namespaces globaux).
- Structurer un projet : modules « services », « UI », « utils »…

Outils modernes

- Gestion des dépendances avec npm / yarn.
- Notions de bundlers (Webpack, Vite, Parcel) et transpilation (Babel).
- Linting et formatage (ESLint, Prettier) pour un code homogène.

Tests et robustesse

- Introduction aux tests unitaires (Jest ou équivalent).
- Tester des fonctions pures et gérer les dépendances.
- Bonnes pratiques de gestion d'erreurs et de logs.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Projet de synthèse en JavaScript avancé

Conception de l'application

- Choix d'un mini-projet : application de gestion (tâches, contacts, produits…), tableau de bord, etc.
- Spécifications fonctionnelles et découpage en composants / modules.

Implémentation

- Architecture des fichiers et modules.
- Intégration d'une API externe (si disponible).
- Gestion de l'état et des interactions utilisateur.
- Stockage des données (localStorage) et restauration de session.

Finalisation et bonnes pratiques

- Refactoring et amélioration des performances.
- Revue de code guidée par le formateur.
- Pistes d'évolution (frameworks front, TypeScript, Node.js…).