DÉVELOPPEMENT FRONTEND

JavaScript avancé : async, POO et patterns modernes

Approfondir JavaScript avec la programmation orientée objet, l'asynchrone avancé, les design patterns et les bonnes pratiques professionnelles.

DURÉE	TARIF HT	NIVEAU	LANGUE	GROUPE	FORMAT
3j (21h)	**990.00 €**	**Intermédiaire**	**Français**	**3-8**	**Formation**

1 PUBLIC VISÉ

- Développeurs JavaScript ayant déjà les bases et souhaitant progresser vers un niveau professionnel
- Développeurs autodidactes voulant structurer et fiabiliser leur code
- Personnes ayant suivi la formation JavaScript fondamentaux : le langage du web ou disposant d'une expérience équivalente

2 PRÉREQUIS

- Bonne maîtrise des fondamentaux JavaScript : variables, fonctions, tableaux, objets, DOM, événements
- Avoir suivi la formation JavaScript fondamentaux : le langage du web ou équivalent
- Savoir utiliser le terminal et VS Code

3 OBJECTIFS PÉDAGOGIQUES

- Comprendre le modèle objet de JavaScript : prototypes et classes ES6
- Maîtriser l'héritage et le polymorphisme en JavaScript
- Gérer des scénarios asynchrones complexes avec async/await et les promesses avancées
- Utiliser les concepts fonctionnels : closures, currying, composition de fonctions
- Appliquer les design patterns les plus courants dans un projet réel
- Écrire des tests unitaires pour fiabiliser son code
- Configurer un outillage professionnel : linting, formatage, bundler

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) – Orienté objet et programmation fonctionnelle

Module 1 – Prototypes et classes ES6 (3h)

Le modèle objet de JavaScript (1h)

- JavaScript n'est pas Java : comprendre l'orienté objet par prototypes
- La chaîne de prototypes : comment JavaScript cherche une propriété
- La différence entre une instance et son prototype
- Pourquoi comprendre les prototypes même quand on utilise les classes

Les classes ES6 (1h)

- Déclarer une classe, un constructeur, des méthodes
- Propriétés publiques, privées et statiques
- Getters et setters
- Cas pratique : modéliser un système de gestion d'utilisateurs

Héritage et polymorphisme (1h)

- Étendre une classe : hériter des propriétés et méthodes
- Surcharger une méthode dans une classe enfant
- Le polymorphisme en pratique : même interface, comportements différents
- Cas pratique : hiérarchie de classes pour une application de e-commerce

Module 2 – Programmation fonctionnelle (4h)

Les closures (1h30)

- Qu'est-ce qu'une closure ? La portée lexicale expliquée
- Cas d'usage concrets : compteurs, mémoïsation, encapsulation privée
- Les pièges courants : closures dans les boucles
- Cas pratique : construire un module avec des données privées via closure

Currying et composition (1h30)

- Le currying : transformer une fonction multi-arguments en fonctions unaires
- La composition de fonctions : enchaîner des transformations
- Différence entre composition et chaînage
- Cas pratique : pipeline de traitement de données

Les itérateurs et générateurs (1h)

- Le protocole itérable de JavaScript
- Créer un itérateur personnalisé

Les générateurs : produire des valeurs à la demande

- Cas d'usage : paginer des résultats, générer des identifiants uniques

Jour 2 (7h) – Asynchrone avancé et design patterns

Module 3 – Asynchrone avancé (3h)

Maîtriser les promesses (1h)

- Rappel : le cycle de vie d'une promesse
- Enchaîner des opérations asynchrones sans perdre la lisibilité
- Lancer plusieurs requêtes en parallèle et attendre toutes les réponses
- Prendre la première réponse disponible parmi plusieurs sources
- Gérer les timeouts et les annulations de requêtes

Gestion avancée des erreurs asynchrones (1h)

- Les pièges de la gestion d'erreurs avec async/await
- Centraliser la gestion des erreurs dans un flux asynchrone
- Réessayer automatiquement une opération qui échoue
- Logger et tracer les erreurs sans bloquer le flux
- Cas pratique : client HTTP robuste avec retry et timeout

Patterns asynchrones avancés (1h)

- La file d'attente de tâches asynchrones
- Limiter le nombre d'opérations parallèles
- Le pattern circuit breaker : couper un service défaillant
- Cas pratique : téléchargeur de fichiers avec concurrence limitée

Module 4 – Design patterns (4h)

Pourquoi les design patterns ? (30min)

- L'origine des patterns : le Gang of Four
- Les trois familles : création, structure, comportement
- Comment identifier quel pattern utiliser face à un problème

Patterns de création (1h)

- Le pattern Module : encapsuler et exposer une API propre
- Le pattern Factory : déléguer la création d'objets
- Le pattern Singleton : garantir une instance unique
- Cas pratique : gestionnaire de configuration applicative

Patterns de comportement (1h30)

Le pattern Observer : réagir à des événements sans couplage fort

- Le pattern Strategy : choisir un algorithme à l'exécution
- Le pattern Command : encapsuler une action pour l'annuler ou la rejouer
- Cas pratique : système de notifications découplé

Patterns de structure (1h)

- Le pattern Decorator : enrichir un objet sans le modifier
- Le pattern Proxy : contrôler l'accès à un objet
- Le pattern Adapter : faire communiquer deux interfaces incompatibles
- Cas pratique : système de cache transparent avec Proxy

Jour 3 (7h) – Tests, outillage et projet de synthèse

Module 5 – Tests unitaires avec Jest (3h)

Pourquoi tester son code ? (30min)

- Les bugs en production : coût humain et technique
- Les différents types de tests : unitaire, intégration, end-to-end
- La pyramide des tests : où investir son effort
- TDD : écrire le test avant le code

Premiers pas avec Jest (1h)

- Installer et configurer Jest dans un projet
- Écrire son premier test : structure d'un test, assertions de base
- Organiser ses tests : fichiers, suites, cas limites
- Lancer les tests et lire les rapports

Tester des cas complexes (1h30)

- Tester des fonctions asynchrones
- Les mocks : simuler des dépendances externes
- Tester une classe et ses méthodes
- Mesurer la couverture de code et l'interpréter
- Cas pratique : écrire une suite de tests pour le module développé en jour 2

Module 6 – Outillage professionnel (2h)

ESLint et Prettier (1h)

- ESLint : détecter les erreurs et imposer des règles de style
- Configurer ESLint pour un projet JavaScript moderne
- Prettier : formater le code automatiquement à la sauvegarde

Faire cohabiter ESLint et Prettier sans conflits
- Partager la configuration avec l'équipe

Vite et l'écosystème moderne (1h)
- Pourquoi un bundler ? Le problème des modules en production
- Vite : démarrage rapide, rechargement instantané, build optimisé
- Configurer un projet JavaScript avec Vite
- Les variables d'environnement dans un projet front-end
- Cas pratique : migrer le projet de la formation vers Vite

Module 7 – Projet de synthèse (2h)

Réalisation d'une application JavaScript structurée
- Cahier des charges : application de gestion de tâches avec architecture en couches
- Fonctionnalités : POO pour les modèles, pattern Observer pour les événements, appels API asynchrones, tests unitaires sur les modules critiques
- Revue de code collective : architecture, lisibilité, couverture de tests
- Retour formateur individualisé

5 COMPÉTENCES VISÉES

- Concevoir et structurer une application en orienté objet avec JavaScript
- Gérer des flux asynchrones complexes sans perdre le contrôle
- Écrire du code maintenable, testable et conforme aux standards professionnels
- Identifier et appliquer le bon design pattern selon le contexte
- Mettre en place un environnement de développement professionnel complet

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou distanciel (visioconférence). Le formateur alterne entre méthode démonstrative (live coding commenté et progressif), méthode interrogative (analyse critique de code existant) et méthode active (exercices de refactoring, écriture de tests, mini-projets par module). L'accent est mis sur la compréhension des mécanismes sous-jacents plutôt que sur la mémorisation de recettes.

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition des apprenants
- Dépôt GitHub de démonstration avec exercices et corrections par module
- Environnement de développement : VS Code + Node.js + Jest + Vite
- Pour le distanciel : visioconférence (Zoom ou équivalent), partage d'écran, chat en direct
- Accès à la plateforme pédagogique LaPolaris (supports, ressources, émargement)

8 MODALITÉS D'ÉVALUATION

- En cours de formation : exercices de refactoring et d'écriture de tests corrigés à chaque module
- En fin de formation : réalisation d'une application structurée avec POO, patterns et tests
- Questionnaire d'auto-évaluation des acquis en fin de parcours

9 CRITÈRES D'ÉVALUATION

- Conception correcte d'une hiérarchie de classes adaptée au problème
- Gestion fluide et sécurisée de scénarios asynchrones complexes
- Application pertinente d'au moins deux design patterns dans le projet final
- Couverture de tests suffisante sur les modules critiques
- Qualité et lisibilité du code produit : nommage, organisation, absence de redondances

10 MODALITÉS DE VALIDATION

Attestation de fin de formation délivrée à l'issue du parcours, conditionnée à une assiduité d'au moins 80 % et à la réalisation du projet de synthèse. L'attestation précise les objectifs atteints et les compétences acquises.

11 SUIVI ET ACCOMPAGNEMENT

- Feuilles d'émargement signées par demi-journée (présentiel) ou émargement numérique (distanciel)
- Traçabilité des activités pédagogiques réalisées
- Attestation d'assiduité délivrée en fin de formation
- Suivi individuel via les exercices corrigés et le projet de synthèse

12 CONDITIONS D'ACCÈS

Formation accessible sur inscription directe, sans prérequis administratif particulier. Le financement peut être pris en charge par l'employeur dans le cadre d'un plan de développement des compétences, ou en autofinancement avec possibilité de paiement en plusieurs fois.

13 DÉLAIS D'ACCÈS

Inscription possible jusqu'à **5 jours ouvrés** avant le début de la session. Pour toute demande urgente, nous contacter directement.

ACCESSIBILITÉ · HANDICAP

Nos formations sont accessibles aux personnes en situation de handicap. Pour toute situation nécessitant un aménagement (matériel, temporel ou pédagogique), nous vous invitons à nous contacter avant l'inscription afin d'étudier les adaptations possibles. Référent handicap : contact@lapolaris.fr

LaPolaris

TÉL. +33762584798

EMAIL contact@lapolaris.fr

WEB lapolaris.fr