REACTJS & TAILWIND CSS

Public	Développeurs front-end ayant déjà pratiqué les bases de React. Profils ayant une expérience concrète avec les composants fonctionnels, les événements, useState et useRef.
Durée	3 jours - 21 heures
Pré-requis	Maîtrise de JavaScript ES2020+ (fonctions fléchées, destructuring, spread, async/await, modules). Connaissance des composants React fonctionnels : JSX, props, useState, useRef, gestion des événements, rendu conditionnel. Notions de base en HTML/CSS
Objectifs	Consolider les fondamentaux React avant d'aborder les notions avancées Maîtriser les hooks essentiels : useEffect, useCallback, useMemo, useReducer et useContext Concevoir une architecture de projet React maintenable (routing, organisation des dossiers, séparation des responsabilités) Mettre en place une gestion d'état robuste avec useReducer, Context API et Redux Toolkit Connecter une application React à une API REST et gérer les données asynchrones Construire des interfaces modernes et responsives avec Tailwind CSS v4 Créer et organiser des composants réutilisables stylisés avec Tailwind CSS Appliquer les bonnes pratiques de performance : memoïsation, code splitting, lazy loading Intervenir directement sur un projet React existant et y appliquer les notions vues en formation
Méthodes pédagogiques	Pour bien préparer la formation, le stagiaire remplit une évaluation de positionnement et fixe ses objectifs à travers un questionnaire. La formation est délivrée en présentiel ou distanciel (e-learning, classe virtuelle, présentiel et à distance). Le formateur alterne entre méthodes démonstratives, interrogatives et actives (via des travaux pratiques et/ou des mises en situation). La validation des acquis peut se faire via des études de cas, des quiz et/ou une certification. Cette formation est animée par un consultant-formateur dont les compétences techniques, professionnelles et pédagogiques ont été validées par des diplômes et/ou testées et approuvées par l'éditeur et/ou par Audit Conseil Formation.
Moyens techniques	1 poste de travail complet par personne De nombreux exercices d'application Mise en place d'ateliers pratiques Remise d'un support de cours Passage de certification(s) dans le cadre du CPF Remise d'une attestation de stage
Modalité d'évaluation des acquis	Evaluation des besoins et objectifs en pré et post formation Evaluation technique des connaissances en pré et post formation Evaluation générale du stage
Délai d'accès	L'inscription à cette formation est possible jusqu'à 5 jours ouvrés avant le début de la session
Accessibilité handicapés	Au centre d'affaires ELITE partenaire d'ACF à 20 m. Guide d'accessibilité à l'accueil.

JOUR 1 — FONDAMENTAUX RENFORCÉS & HOOKS ESSENTIELS

1. MODULE 1 — REMISE À NIVEAU ET CONSOLIDATION DES BASES REACT (1H30)

- Rappel de l'architecture d'un composant fonctionnel React moderne
- JSX avancé : fragments, rendu de listes avec key, rendu conditionnel ternaire et &&
- Props : valeurs par défaut, destructuring, propagation avec le spread operator
- useState en profondeur : mise à jour fonctionnelle, état complexe (objets, tableaux imbriqués)
- useRef : accès au DOM, stockage de valeur persistante sans déclencher de re-render
- Gestion des événements : event object, prévention du comportement par défaut, délégation
- **Atelier pratique :** Exercice : refactorisation d'un composant de formulaire existant en appliquant les bonnes pratiques

2. MODULE 2 — USEEFFECT : SYNCHRONISATION ET EFFETS DE BORD (2H)

- Comprendre le cycle de vie d'un composant fonctionnel et le rôle de useEffect
- Les 3 formes de useEffect : sans dépendances, avec dépendances, avec fonction de cleanup
- Cas d'usage courants : appels API, synchronisation avec le DOM, abonnements, timers
- Pièges classiques : dépendances manquantes, boucles infinies, mise à jour sur composant démonté
- La fonction de cleanup en pratique : AbortController, clearInterval, removeEventListener
- useLayoutEffect vs useEffect : différences et cas d'usage spécifiques
- **Atelier pratique :** Atelier : connexion d'un composant de liste à une API REST avec gestion complète des états loading / error / success

3. MODULE 3 — USECALLBACK ET USEMEMO : OPTIMISER SANS SUR-OPTIMISER (1H30)

- Comprendre pourquoi React re-rend les composants et à quel coût réel
- useMemo : mémoriser le résultat d'un calcul coûteux, gestion des dépendances
- useCallback : mémoriser une référence de fonction stable pour les props de callbacks
- React.memo : empêcher le re-render d'un composant enfant si ses props sont inchangées
- Règle d'or : mesurer avant d'optimiser — utiliser le Profiler React DevTools
- Exemples concrets où ces hooks font une vraie différence vs cas où ils sont inutiles
- **Atelier pratique :** Atelier : identification et correction de re-renders inutiles sur un composant de tableau de données

4. MODULE 4 — USEREDUCER : GÉRER UN ÉTAT COMPLEXE (2H

- Limites de useState pour les états complexes et interdépendants
- useReducer : reducer, action, dispatch — comprendre le pattern par analogie avec Redux
- Structure d'une action typée : { type, payload } et gestion en switch/case
- Immutabilité en pratique : mise à jour d'objets et de tableaux imbriqués sans mutation
- Combiner useReducer avec useContext pour un state global léger sans librairie
- Comparatif useReducer vs useState : quand choisir l'un ou l'autre
- **Atelier pratique :** Atelier : remplacement de plusieurs useState par un useReducer sur un formulaire multi-étapes

JOUR 2 — ARCHITECTURE, ROUTING & STATE MANAGEMENT

1. MODULE 5 — ARCHITECTURE D'UN PROJET REACT (1H30)

- Structure de dossiers recommandée : organisation par fonctionnalité ou par couche
- Rôle et organisation des dossiers clés : /components, /hooks, /pages, /services, /lib, /routes, /data
- Séparation des responsabilités : composants de présentation vs composants logiques
- Création de hooks personnalisés (custom hooks) pour extraire et réutiliser la logique
- Exemples concrets : useFetch, useLocalStorage, useDebounce, useForm
- Conventions de nommage, barrel exports (index.js) et bonnes pratiques d'import
- Revue d'architecture live : analyse d'un projet existant et identification des axes d'amélioration
- **Atelier pratique :** Atelier : restructuration de l'arborescence d'un projet et extraction de la logique métier dans des hooks custom

2. MODULE 6 — REACT ROUTER V6 : ROUTING AVANCÉ (1H30)

- Rappel des concepts de base : BrowserRouter, Routes, Route, Link, NavLink
- Routes imbriquées (nested routes) et composant Outlet pour les layouts partagés
- Paramètres d'URL dynamiques avec useParams et query strings avec useSearchParams
- Navigation programmatique avec useNavigate
- Routes protégées (PrivateRoute) : gestion de l'authentification côté client
- Lazy loading des pages avec React.lazy + Suspense pour optimiser le chargement initial
- Gestion des erreurs de routing avec errorElement
- **Atelier pratique :** Atelier : mise en place du routing complet d'une application multi-pages avec une route protégée

3. MODULE 7 — CONTEXT API ET GESTION D'ÉTAT GLOBALE (2H)

- useContext : partager des données dans l'arbre de composants sans prop drilling
- Créer un Context robuste : Provider, valeur par défaut, bonne organisation
- Pattern « Context + Reducer » : un store global léger sans librairie externe
- Optimiser les contextes : éviter les re-renders inutiles avec useMemo et la découpe des contextes
- Introduction à Redux Toolkit : configureStore, createSlice, useSelector, useDispatch
- RTK Query : fetching de données avec cache intégré, invalidation, refetch automatique
- Quand utiliser Context, quand passer à Redux : critères de décision concrets
- **Atelier pratique :** Atelier : implémentation d'un contexte d'authentification global, puis extension avec Redux Toolkit

4. MODULE 8 — APPELS API ET DONNÉES ASYNCHRONES (2H)

- Fetch API et Axios : comparatif, gestion des erreurs HTTP, configuration d'une instance globale
- Pattern service layer : centraliser les appels API dans des fichiers dédiés (/services)
- Gestion cohérente des états asynchrones dans toute l'application : loading, error, data
- Introduction à React Query (TanStack Query) : useQuery, useMutation, cache, invalidation
- Comparatif useEffect + fetch vs React Query : gains en lisibilité et en fiabilité
- Pagination simple et chargement infini avec React Query
- Variables d'environnement Vite (.env) pour les URLs d'API
- **Atelier pratique :** Atelier : remplacement des useEffect de fetching par React Query sur les pages de liste et de détail

JOUR 3 — TAILWIND CSS V4, PERFORMANCE & BONNES PRATIQUES

1. MODULE 9 — TAILWIND CSS V4 : FONDAMENTAUX ET INTÉGRATION REACT (2H)

- Philosophie utility-first : écrire le style directement dans le JSX, sans fichiers CSS séparés
- Tailwind CSS v4 : nouveautés clés — moteur Oxide (Rust), 5× plus rapide, installation simplifiée
- Configuration CSS-first : plus de tailwind.config.js, tout se définit avec la directive @theme
- Installation dans un projet Vite + React : plugin @tailwindcss/vite, import en une seule ligne
- Les utilitaires essentiels : spacing, typography, colors, flexbox, grid, borders, shadows
- Responsive design avec les préfixes de breakpoints : sm:, md:, lg:, xl: — approche mobile-first
- Dark mode avec la variante dark:
- États interactifs : hover:, focus:, active:, disabled:, group-hover:
- **Atelier pratique :** Atelier : intégration de Tailwind CSS v4 dans le projet, remplacement des styles CSS par des classes utilitaires

2. MODULE 10 — TAILWIND CSS V4 : COMPOSANTS, DESIGN SYSTEM ET BONNES PRATIQUES (2H)

- Créer des composants React réutilisables et stylisés avec Tailwind
- Gérer les variantes de style via les props : Button (primary / secondary / danger), Badge, Card
- La directive @layer et @utility : créer des classes composées réutilisables dans le CSS
- Introduction à shadcn/ui : installer et customiser des composants accessibles prêts à l'emploi
- Animations et transitions : transition-, duration-, ease-, animate- et keyframes personnalisées
- Organisation et conventions pour maintenir un projet Tailwind lisible sur le long terme
- **Atelier pratique :** Atelier : construction d'une bibliothèque de composants UI cohérente — Bouton, Carte, Badge, Champ de formulaire

3. MODULE 11 — PERFORMANCES ET OPTIMISATION REACT (1H30)

- Comprendre le cycle de rendu React : quand et pourquoi un composant re-rend
- Profiler une application avec React DevTools : identifier les composants lents
- Récapitulatif des outils de memoïsation : React.memo, useMemo, useCallback — quand les utiliser
- Code splitting par route avec React.lazy et Suspense
- Optimisation des listes longues : virtualisation avec TanStack Virtual
- Chargement paresseux des images : attribut loading="lazy" et Intersection Observer
- Mesurer l'impact réel : Lighthouse, Core Web Vitals (LCP, INP, CLS)
- **Atelier pratique :** Atelier : audit de performance puis application des optimisations sur un composant de liste chargé depuis une API

4. MODULE 12 — SYNTHÈSE ET MISE EN APPLICATION SUR PROJET RÉEL (1H30)

- Récapitulatif des patterns vus en formation : bonnes pratiques React et Tailwind
- Checklist de qualité pour un projet React en production : linting, formatage, conventions Git
- Erreurs courantes et comment les éviter : dépendances useEffect, mutations d'état directes, clés React
- Mise en application guidée : chaque participant implémente une fonctionnalité complète sur son propre projet en mobilisant les notions de la formation
- Revue de code collective et retours personnalisés du formateur
- Questions / réponses et ressources pour continuer à progresser
- Bilan de fin de formation et QCM de validation des acquis

NOUS CONTACTER

Siège social
16, ALLÉE FRANÇOIS VILLON
38130 ÉCHIROLLES

Téléphone
04 76 23 20 50 - 06 81 73 19 35

Suivez-nous sur les réseaux sociaux, rejoignez la communauté !

ACF Audit Conseil Formation

@ACF_Formation

Dernière mise à jour : 17/04/2026

PROFIL Formateur : Les formateurs sont recrutés selon plusieurs critères :
Expérience, pédagogie, dynamisme et prévoyance.