

Formation Tailwind CSS : créer des interfaces modernes avec l'approche utility-first

Réf. SF34595

Maîtrisez Tailwind CSS pour concevoir des interfaces web responsives, personnalisées et prêtes pour la production.

3 j Durée	2 250 € HT Inter / participant	5 250 € HT Intra (≤12 pers.)	0 Sessions à venir
--------------	-----------------------------------	---------------------------------	-----------------------

Objectifs pédagogiques

- ✓ Expliquer la philosophie utility-first de Tailwind CSS et ses avantages par rapport aux approches CSS traditionnelles
- ✓ Concevoir des composants réutilisables et structurer un mini design system
- ✓ Configurer un environnement de développement Tailwind CSS avec les outils modernes (Vite, npm)
- ✓ Personnaliser le thème Tailwind (couleurs, typographies, animations) selon une charte graphique
- ✓ Utiliser les classes utilitaires pour gérer la mise en page, la typographie, les couleurs et les espacements
- ✓ Optimiser le CSS généré pour un déploiement en production
- ✓ Créer des interfaces responsives en exploitant le système de breakpoints et le dark mode
- ✓ Intégrer Tailwind CSS dans un projet React, Vue ou Next.js

Public visé & prérequis

Public visé Développeurs

Prérequis- Maîtrise de HTML et CSS - Notions de responsive design - Bases en JavaScript - Connaissance d'au moins un framework JS (React, Vue ou Next.js) recommandée

Programme détaillé

Découverte de Tailwind CSS et philosophie utility-first

- Approche utility-first : principes fondamentaux et différences avec le CSS classique et les frameworks comme Bootstrap
- Comparaison des approches : avantages et limites de Tailwind CSS face aux méthodologies BEM, CSS-in-JS et aux frameworks orientés composants
- Design system : comprendre comment Tailwind structure un système de design cohérent dès la configuration
- Documentation officielle : naviguer efficacement dans la référence Tailwind pour gagner en autonomie

Activités pratiques :

- Analyse comparative d'une même interface réalisée en CSS classique et en Tailwind CSS
- Exploration guidée de la documentation officielle pour résoudre des cas concrets

Installation et configuration de l'environnement

- Méthodes d'installation : mise en place via CDN pour le prototypage et via npm/Vite pour les projets professionnels
- Fichier tailwind.config.js : rôle, structure et options de configuration de base
- Structure CSS : organisation des couches @tailwind base, components et utilities
- Compilation Just-In-Time (JIT) : fonctionnement du moteur de génération à la demande et impact sur les performances de développement
- Outillage éditeur : configuration du plugin VS Code Tailwind CSS IntelliSense pour l'autocomplétion et la prévisualisation

Activités pratiques :

- Initialisation d'un projet Tailwind CSS avec Vite et npm
- Configuration de l'autocomplétion dans VS Code et vérification du fonctionnement JIT

Maîtrise des classes utilitaires

- Espacements : gestion fine du padding, margin, gap et space entre éléments
- Typographie : tailles de texte, poids de police, interlignage et couleurs de texte
- Couleurs et arrière-plans : palette par défaut, opacité et dégradés
- Bordures et effets visuels : border-radius, ombres portées et ring pour les focus states
- Mise en page Flexbox : alignement, distribution et ordonnancement des éléments
- Mise en page Grid : grid-cols, gap, auto-fit et construction de layouts complexes
- Dimensionnement : container, max-w, aspect-ratio et gestion des tailles responsives

Activités pratiques :

- Construction d'un header avec navigation en Flexbox
- Réalisation d'une grille de cartes produits avec CSS Grid et les utilitaires Tailwind

Responsive design et dark mode

- Breakpoints Tailwind : système mobile-first avec les préfixes sm, md, lg, xl et 2xl
- Classes conditionnelles : application de styles selon le contexte (hover, focus, group, peer)
- Dark mode automatique : activation via les préférences système (prefers-color-scheme)
- Dark mode manuel : implémentation d'un toggle clair/sombre en JavaScript avec la classe dark

Activités pratiques :

- Adaptation responsive d'un layout multi-colonnes (mobile → desktop)
- Implémentation d'un switch de thème clair/sombre avec persistance du choix utilisateur

Composants réutilisables et design system

- Directive @apply : extraction de patterns récurrents en classes CSS réutilisables
- Classes conditionnelles : gestion des variantes d'un composant (tailles, couleurs, états)
- Bonnes pratiques de structuration : quand utiliser @apply vs composition de classes vs composants framework
- UI libraries : intégration de shadcn/ui et Headless UI pour accélérer le développement
- Mini design system : organisation des tokens (couleurs, espacements, typographies) et documentation des composants

Activités pratiques :

- Création d'une bibliothèque de composants de base (boutons, cartes, badges, alertes)
- Structuration d'un mini design system documenté avec les tokens du projet

Configuration avancée et optimisation

- Personnalisation du thème : extension et surcharge des couleurs, typographies et breakpoints pour respecter une charte graphique
- Utilitaires personnalisés : création de classes spécifiques au projet
- Directives Tailwind 4.x : utilisation de @theme, @layer et @plugin pour une configuration moderne
- Animations personnalisées : définition de keyframes et transitions dans la configuration Tailwind
- Safelists : conservation de classes générées dynamiquement lors de la purge
- Optimisation production : purge CSS automatique, minification et analyse de la taille du bundle

Activités pratiques :

- Adaptation complète du thème Tailwind à une charte graphique fournie
- Création d'animations personnalisées (transitions, apparitions) via la configuration
- Audit et optimisation du poids CSS avant déploiement

Intégration dans les frameworks modernes

- Tailwind avec React : configuration, approche par composants et gestion des classes dynamiques
- Tailwind avec Next.js : setup spécifique, App Router et optimisations SSR
- Tailwind avec Vue : intégration dans les Single File Components et les composables
- Composants dynamiques : gestion des styles conditionnels avec clsx/cn et template literals
- Système de layout global : mise en place d'une structure de page cohérente et réutilisable
- Accessibilité : utilisation des classes sr-only, focus-visible et bonnes pratiques ARIA avec Tailwind

Activités pratiques :

- Intégration de Tailwind dans un projet React ou Vue existant

- Construction d'un layout d'application complet (sidebar, header, contenu principal) avec gestion responsive

Pourquoi choisir Sparks Formation ?

20 ans d'expertise IT, des formateurs experts et des sessions garanties contre l'annulation.

20 ans
d'expertise

1 068+
formations

11 178
apprenants formés/an

+67
Net Promoter Score

Avis des participants

Note moyenne des participants de la filière Développement Web & Fullstack : 4,9/5 — 39 avis

Maxime V. — session très agréable, dense mais toujours agréable. formateur très à l'écoute, prends le temps de répondre aux questions. on apprend l'utilisation des fonctions mais aussi les processus mis...

Cécile L. — Très bon instructeur qui explique toujours en montrant du code, aucune slide et une présentation vivante ce qui permet de rester intéressé et concentré tout au long !...

Ils nous font confiance

Cette année notre filière Développement Web & Fullstack a accompagné plus de 111 entreprises dont :

ARPEGE APSIDE TOP CEGEDIM CONSERVATOIRE NATIONAL DES ARTS ET METIERS ECONOCOM APPS, CLOUD & DATA ORANGE BUSINESS
ORANGE BUSINESS SERVICES SA SOPRA STERIA ADVENTIEL AGATE AGENCE ALPINE DES TERRITOIRES +101 autres

Financement & certifications

Cette formation est éligible aux financements OPCO et FNE-Formation. Nos équipes vous accompagnent dans le montage de votre dossier.



Réservez votre place dès maintenant

20 ans d'expertise · 1 000+ formations · NPS +67 · réponse sous 24h ouvrées

0 805 950 800

demande@sparks-formation.com