Formation **Angular : développer des applications web modernes**

Angular est le framework front-end de référence maintenu par Google, utilisé par des milliers d'entreprises pour construire des applications web robustes et maintenables. Cette formation de 4 jours vous guide pas à pas dans le développement d'une application Angular complète, en suivant les pratiques recommandées par l'équipe Angular. Vous commencerez par les fondamentaux — composants, templates, directives — puis vous progresserez vers la programmation réactive avec RxJS et les Signals, le routage, les formulaires réactifs et la communication avec des API REST. Chaque concept est mis en pratique immédiatement à travers une application fil rouge que vous enrichissez tout au long de la formation. Vous terminerez par les tests unitaires avec Vitest et les bonnes pratiques de déploiement.

Durée

4 jours

Objectifs pédagogiques

- Créer et structurer une application Angular avec les standalone components et Angular CLI
- Développer des composants réutilisables avec le système de templates, directives et pipes
- Implémenter la réactivité avec RxJS et les Signals (signal, computed, effect)
- Gérer la navigation et le chargement différé avec le Router Angular
- Communiquer avec un backend via HttpClient et consommer des API REST
- Construire des formulaires réactifs avec validation typée

Public

Développeurs, architectes, chefs de projet techniques

Prérequis

- Maîtrise de JavaScript (ES6+) ou TypeScript- Connaissances solides en HTML5 et CSS3- Notions de programmation orientée objet- Familiarité avec la ligne de commande (npm, terminal)

Programme de formation

Rappels TypeScript et JavaScript moderne

- Évolutions ECMAScript (ES6 à ES2024) : fonctions fléchées, destructuring, modules, async/await et les apports récents du langage
- TypeScript pour Angular : typage statique, interfaces, classes, génériques et décorateurs pour écrire du code robuste
- Outillage moderne : configuration de l'environnement de développement avec Node.js, Angular CLI et un éditeur de code

Exemples d'activités pratiques :
- Installation et configuration de l'environnement de développement Angular
- Refactorisation d'un script JavaScript vers TypeScript avec typage strict

Fondamentaux d'Angular et premiers composants

- Architecture d'une application Angular : structure de projet, standalone components et rôle de l'Angular CLI basé sur esbuild/Vite
- Composants : décorateur @Component, templates inline et externes, styles encapsulés et cycle de vie
- Syntaxe des templates : interpolation, property binding, event binding et two-way binding
- Control flow moderne : directives @if, @for et @switch pour structurer les vues de manière déclarative
- Directives et pipes : directives d'attributs, directives structurelles personnalisées et pipes de transformation intégrés ou custom

Exemples d'activités pratiques :
- Création d'une application Angular avec l'Angular CLI et exploration de la structure générée
- Développement d'un composant interactif exploitant les différentes formes de data binding
- Création d'une directive personnalisée et d'un pipe de formatage

Programmation réactive avec RxJS et Signals

- Programmation réactive : principes fondamentaux, Observables, opérateurs RxJS et gestion de l'asynchronisme
- Signals Angular : signal(), computed() et effect() comme nouvelle API réactive native du framework
- Complémentarité RxJS / Signals : quand utiliser l'un ou l'autre, interopérabilité entre les deux approches
- Mode Zoneless : comprendre Zone.js et l'évolution vers une détection de changements pilotée par les Signals

Exemples d'activités pratiques :
- Implémentation de flux de données réactifs avec des Observables et des opérateurs RxJS
- Refactorisation d'un composant pour utiliser les Signals à la place des Observables
- Observation du comportement de la détection de changements avec et sans Zone.js

Composants avancés et injection de dépendances

- Communication entre composants : Input et Output basés sur les Signals pour une communication parent/enfant typée et réactive
- Cycle de vie des composants : hooks et bonnes pratiques pour gérer l'initialisation, les mises à jour et la destruction
- Injection de dépendances : fonction inject(), hiérarchie des injecteurs et providers dans un contexte standalone
- Services Angular : création, portée d'injection et intégration optimisée avec les standalone components

Exemples d'activités pratiques :

- Structuration d'une application multi-composants avec communication par Signals
- Création d'un service partagé avec injection hiérarchique

Formulaires réactifs

- Reactive Forms : FormControl, FormGroup et FormBuilder pour construire des formulaires dynamiques
- Formulaires typés : exploitation du typage fort introduit dans les versions récentes d'Angular (TypedFormGroup, FormControl typé)
- Validation : validators natifs, validators personnalisés synchrones et asynchrones, gestion des erreurs et affichage dynamique
- Template-driven vs Reactive : critères de choix selon le contexte du projet

Exemples d'activités pratiques :
- Construction d'un formulaire de saisie réactif avec validation cross-field
- Implémentation d'un validator asynchrone vérifiant la disponibilité d'un identifiant

Routage et communication HTTP

- Router Angular : configuration avec provideRouter(), définition des routes, navigation par routerLink et paramètres dynamiques
- Chargement différé : lazy loading de composants standalone avec loadComponent() pour optimiser le bundle initial
- Guards et resolvers : protection de routes et pré-chargement de données avant navigation
- HttpClient : requêtes HTTP (GET, POST, PUT, DELETE) avec provideHttpClient(), interceptors pour la gestion transversale (authentification, logs, erreurs)
- Consommation d'API REST : communication avec un backend, gestion des réponses et traitement des erreurs

Exemples d'activités pratiques :
- Mise en place d'une navigation multi-pages avec lazy loading et paramètres de route
- Consommation d'une API REST depuis un service Angular avec gestion des erreurs via interceptor

Tests unitaires et qualité du code

- Écosystème de test Angular : passage de Karma (deprecated) à Vitest comme runner de tests moderne
- Tests de composants : configuration de TestBed, rendu de composants et assertions sur le DOM
- Tests de services : injection de dépendances dans les tests, mocking de HttpClient
- Couverture de code : mesure et interprétation de la couverture pour guider l'effort de test

Exemples d'activités pratiques :
- Écriture de tests unitaires pour un composant et un service de l'application fil rouge
- Configuration de Vitest et exécution d'un rapport de couverture

Déploiement et bonnes pratiques

- Build de production : commandes de build Angular CLI, optimisations automatiques (tree-shaking, minification, esbuild)
- Variables d'environnement : gestion de la configuration par environnement (développement, staging, production)
- Introduction au SSR : Server-Side Rendering et hydration progressive pour améliorer le temps de chargement initial
- Accessibilité et performance : bonnes pratiques A11y, audit avec Angular DevTools, directives @defer pour le chargement différé de contenu

Exemples d'activités pratiques :
- Build et déploiement de l'application fil rouge en mode production
- Audit de performance avec Angular DevTools et identification des optimisations

Moyens et méthodes pédagogiques

- La formation alterne entre présentations des concepts théoriques et mises en application à travers d'ateliers et exercices pratiques (hors formation de type séminaire).
- Les participants bénéficient des retours d'expérience terrains du formateur ou de la formatrice
- Un support de cours numérique est fourni aux stagiaires

Modalités d'évaluation

- **En amont de la session de formation**, un questionnaire d'auto-positionnement est remis aux participants, afin qu'ils situent leurs connaissances et compétences déjà acquises par rapport au thème de la formation.

- **En cours de formation**, l'évaluation se fait sous forme d'ateliers, exercices et travaux pratiques de validation, de retour d'observation et/ou de partage d'expérience, en cohérence avec les objectifs pédagogiques visés.

- **En fin de session**, le formateur évalue les compétences et connaissances acquises par les apprenants grâce à un questionnaire reprenant les mêmes éléments que l'auto-positionnement, permettant ainsi une analyse détaillée de leur progression.