

# Angular 21 et 22 – Développement d'applications Web

3 jours (21 heures)

Initiation

Présentiel ou classe à distance

à partir de

**1905 € HT** par personne (TVA en vigueur)

Repas inclus en présentiel

Contenu mis à jour le 30/09/2026. Document téléchargé le 02/10/2026.

## Objectifs

A l'issue de cette formation, vous serez capable de :

- Utiliser les versions 21 et 22 du framework Angular
- Maîtriser l'architecture Signals-first et le mode zoneless
- Développer et tester complètement une application avec Vitest
- Concevoir des formulaires modernes (Reactive Forms et Signal Forms)
- Mettre en pratique les outils d'Intelligence Artificielle (IA) officiels d'Angular pour développer plus efficacement
- Appliquer les bonnes pratiques de développement.

Les compétences visées par cette formation font l'objet d'une évaluation.

## À qui s'adresse cette formation ?

Développeurs et chefs de projets.

### Prérequis

Avoir une bonne connaissance des langages du Web comme HTML, CSS et JavaScript.

# Programme

## 1 Pendant votre formation

### Angular 21 et 22 – Développement d'applications Web

Jour 1 – Matin

#### Architecture, installation et premier projet Angular 21 / 22

- Architecture moderne d'une application Angular :
  - Tout est standalone par défaut depuis Angular 19
  - Disparition progressive des NgModule (encore supportés mais plus recommandés)
  - Simplification du bootstrapping avec bootstrapApplication() et app.config.ts
- Installation de l'environnement avec Angular CLI v21 / v22
- Premier projet : création via CLI et déploiement local avec Vite / esbuild
- Le mode zoneless par défaut (Angular 21) :
  - Suppression de zone.js des polyfills
  - Bundle plus léger, change detection prévisible et plus performante
  - provideZoneChangeDetection() à ajouter explicitement pour les anciennes applications
- TypeScript 5.9+ : types, classes, interfaces, modules ES, async / await
- Architecture MVVM Signals-first, sans Zone.js
- A noter pour Angular 22 : OnPush deviendra la stratégie de change detection par défaut

*Exemple de travaux pratiques (à titre indicatif)*

- Création d'un projet Angular 21 / 22 zoneless et exploration du code généré

Jour 1 – Après-midi

#### Les Signals : le cœur réactif d'Angular 21 / 22

- Pourquoi les Signals ? Le passage de Zone.js à un modèle réactif explicite
- Les trois primitives fondamentales :
  - signal() – valeur réactive modifiable
  - computed() – valeur dérivée, recalculée à la demande
  - effect() – réaction à un changement (logs, persistance, side-effects)
- Lecture / écriture : (), .set(), .update()
- Comparaison rapide Signals vs Observables (Promises mises de côté pour le moment)
- Les Signals dans le template : binding automatique sans async pipe

*Exemple de travaux pratiques (à titre indicatif)*

- Création d'un compteur réactif puis d'une mini todo-list utilisant signal(), computed(), effect()

Jour 2 – Matin

## Composants standalone, templates modernes et communication

- Démarrer "from scratch" avec ng new et composants standalone
- Création d'un composant à la main (@Component + bootstrapApplication)
- Templates syntaxiques modernes :
  - Interpolation {{ }}
  - Property binding [prop]
  - Event binding (event)
  - Two-way binding [(ngModel)]
- Directives structurales intégrées @if, @for, @switch, @let
- Remplacement définitif de \*ngIf, \*ngFor, \*ngSwitch
- Pipes intégrés et création de pipes personnalisés
- Communication entre composants avec les Signals :
  - input() – remplace @Input() (signal en lecture seule côté enfant)
  - output() – remplace @Output()
  - model() – two-way binding signal-natif
  - viewChild() / contentChild() à base de Signals

### Exemples de travaux pratiques (à titre indicatif)

- Création de plusieurs composants communicants avec input(), output(), model()
- Liste filtrable utilisant @for et computed()

Jour 2 – Après-midi

## Services, HTTP et accès au backend

- Création et injection de services avec inject() (recommandé)
- provideHttpClient() activé par défaut depuis Angular 21
- Différences entre Promises, Observables et Signals (rappel et choix selon les cas)
- Présentation rapide de RxJS (encore utile pour les flux asynchrones)
- Requêtes HTTP via HttpClient :
  - HttpClient.get, post, put, delete
  - Intercepteurs basiques
  - Gestion d'erreurs avec catchError
- Intégration avec les Signals : convertir un Observable HTTP en Signal avec toSignal()
- Introduction à httpResource() : chargement asynchrone signal-natif (approfondi en formation avancée)
- async / await vs subscribe dans les composants
- Mise en place rapide de Tailwind CSS via ng add @angular/cli:tailwind

### Exemple de travaux pratiques (à titre indicatif)

- Intégration d'une API REST externe dans un service, exposition via Signals et affichage dans le composant avec @if / @for

Jour 3 – Matin

### Formulaires Angular : Reactive Forms et Signal Forms

- Types de formulaires : Template-driven vs Reactive Forms
- Création de formulaires "classiques":
  - FormsModule
  - ReactiveFormsModule et FormBuilder
  - FormGroup, FormControl typés (TypedFormGroup)
- Validation, gestion des erreurs, affichage dynamique
- Signal Forms – la grande nouveauté Angular 21 / 22 :
  - Nouvelle API expérimentale (Angular 21) > stable (Angular 22)
  - Création d'un formulaire avec la fonction form() à partir d'un signal modèle
  - Synchronisation automatique entre modèle (signal) et DOM
  - Validation déclarative basée sur des schémas
  - Type-safety complet
  - Mises à jour fines : seul le champ modifié déclenche un re-render
  - Migration progressive avec @angular/forms/signals/compat

#### Exemples de travaux pratiques (à titre indicatif)

- Création d'un formulaire de login avec Reactive Forms typés
- Refactorisation du même formulaire en Signal Forms

Jour 3 – Après-midi

### Routing, tests Vitest et développement assisté par Intelligence Artificielle (IA)

- Routing modulaire :
  - Configuration via provideRouter()
  - Définition des routes dans des fichiers séparés
  - Lazy loading via loadComponent() et loadChildren()
  - Navigation avec routerLink, RouterOutlet, paramètres dynamiques
  - Notions de guards et resolvers (approfondies en formation avancée)
- Les tests dans Angular 21 / 22 :
  - Vitest : runner officiel et stable depuis Angular 21
  - Remplace Karma / Jasmine pour les nouveaux projets (Karma / Jasmine restent disponibles)
  - Tests unitaires de composants standalone
  - En mode zoneless : await fixture.whenStable() au lieu de fixture.detectChanges()
  - E2E testing avec Playwright (préconisé)
- Développement Angular assisté par IA :
  - Pourquoi Angular investit autant sur l'IA : framework qui évolue vite, LLM qui ont du retard
  - Les fichiers llms.txt et llms-full.txt officiels (angular.dev/context/) :

- Index et corpus complet de la doc Angular optimisé pour les LLM
- A donner en contexte à Claude, ChatGPT, Gemini... pour générer du code Angular à jour
- best-practices.md : fichier de règles officielles à inclure dans son IDE
- Fichiers de règles spécifiques aux IDE :
- GitHub Copilot : .github/copilot-instructions.md
- Cursor : .cursor/rules/
- MCP Server Angular CLI (introduction) :
- ng mcp – serveur Model Context Protocol
- Permet à un assistant IA d'interagir directement avec votre projet Angular
- Découverte des principaux outils (vue détaillée en formation avancée)

#### Exemples de travaux pratiques (à titre indicatif)

- Configuration d'un router avec plusieurs vues (home, about, produit) et lazy loading
- Tests unitaires d'un service et d'un composant standalone avec Vitest
- Configuration de Claude / Cursor / Copilot avec les fichiers llms.txt officiels d'Angular et génération assistée d'un composant Signals-first

## Financement

Plusieurs dispositifs de financement sont accessibles via les OPCO (Opérateurs de Compétences), organismes agréés par le ministère du Travail dont le rôle est d'accompagner, collecter et gérer les contributions des entreprises au titre du financement de la formation professionnelle.

Pour plus d'information, une équipe de gestionnaires SKOLAE Formation spécialisée vous accompagne dans le choix de vos formations et la gestion administrative.

[Nous contacter pour en savoir plus →](#)

## Modalités pédagogiques

Formation délivrée en présentiel ou distanciel\* (blended-learning, e-learning, classe virtuelle, présentiel à distance).

Le formateur alterne entre méthode\*\* démonstrative, interrogative et active (via des travaux pratiques et/ou des mises en situation).

Variables suivant les formations, les moyens pédagogiques mis en oeuvre sont :

- Ordinateurs Mac ou PC (sauf pour certains cours de l'offre Management), connexion internet fibre, tableau blanc ou paperboard, vidéoprojecteur ou écran tactile interactif (pour le distanciel)

- Environnements de formation installés sur les postes de travail ou en ligne
- Supports de cours et exercices

En cas de formation intra sur site externe à M2I, le client s'assure et s'engage également à avoir toutes les ressources matérielles pédagogiques nécessaires (équipements informatiques...) au bon déroulement de l'action de formation visée conformément aux prérequis indiqués dans le programme de formation communiqué.

\* nous consulter pour la faisabilité en distanciel

\*\* ratio variable selon le cours suivi

### Accessibilité aux personnes en situation de handicap

## Détails de la session

Du 7 décembre 2026 au 9 décembre 2026 (3 jours / 21 heures)

Reims

Du 7 décembre 2026 au 9 décembre 2026 **Reims**

- 7 décembre 2026 ..... 9h00 – 16h00
- 8 décembre 2026 ..... 9h00 – 16h00
- 9 décembre 2026 ..... 9h00 – 16h00