

Angular 21 et 22 – Fonctionnalités avancées

3 jours (21 heures)

Perfectionnement

Présentiel ou classe à distance

à partir de

2 085 € HT par personne (TVA en vigueur) **Repas inclus en présentiel**

Contenu mis à jour le 25/09/2026. Document téléchargé le 02/10/2026.

Réservez la formation Angular 2 à 6 – Fonctionnalités avancées, disponible dans 35 villes en France.

Objectifs

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

- Maîtriser en profondeur l'architecture Signals-first et zoneless
- Concevoir des applications avec gestion d'état avancée (NgRx Signal Store et NgRx classique)
- Sécuriser une application avec OAuth2 / OpenID Connect (Keycloak)
- Implémenter du SSR avec hydratation incrémentale
- Construire des composants accessibles avec Angular Aria
- Internationaliser une application et la déployer en PWA.

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

À qui s'adresse cette formation ?

Développeurs Angular et chefs de projets.

Prérequis

Avoir suivi la formation JVS-ANGU "Angular 21 et 22 – Développement d'applications Web" ou avoir les connaissances équivalentes (composants standalone, Signals de base, Reactive Forms, routage simple, tests Vitest).

Programme

1 Pendant votre formation

Angular 21 et 22 – Fonctionnalités avancées

Jour 1 – Matin

Architecture moderne et patterns avancés Angular 21 / 22

- Vue d'ensemble de l'architecture actuelle :
 - Tout est standalone par défaut, disparition progressive des NgModule
 - Simplification continue : `provideHttpClient()` par défaut, schématic common-to-standalone
- Etat de l'écosystème : composants, directives, pipes, services adaptés à Angular 21 / 22

Zone.js et mode Zoneless en profondeur

- Rappel du rôle historique de Zone.js
- Zoneless par défaut en Angular 21 :
 - Suppression de zone.js des polyfills, bundle plus léger
 - Change detection pilotée par les Signals et événements de template
 - `provideBrowserGlobalErrorListeners()` et gestion d'erreurs
- Stratégies de migration vers le mode zoneless
- Angular 22 : `ChangeDetectionStrategy.OnPush` deviendra la stratégie par défaut (Eager reste accessible explicitement)

Patterns avancés et nouveautés des versions 21 et 22

- API Signal-first : `input()`, `output()`, `model()`, `viewChild()`, `contentChild()`
- API de création dynamique de composants stabilisée (`createComponent()`, `inputBinding()`, `outputBinding()`)
- Syntaxe de template enrichie (`@let`, opérateur `in`, template literals)
- Selectorless components (Angular 22) : import direct des classes dans les templates

Exemple de travaux pratiques (à titre indicatif)

- Audit d'une application Angular 17 / 18 et identification des points de migration vers Angular 21 zoneless

Jour 1 – Après-midi

Signals avancés et Resource API

Approfondissement des primitives Signals

- Signaux modifiables – patterns d'usage (état dérivé, mutations imbriquées)
- Signaux calculés (`computed`) – réactivité fine, dépendances pures, mémoïsation
- `linkedSignal()` – signaux calculés modifiables (cas d'usage : `reset`, `default values`)
- Effets (`effect()`) et `untracked()` – quand et comment les utiliser sans piège
- Gestion du cycle de vie : `effect cleanup`, `DestroyRef`

Resource API : le chargement asynchrone signal-natif

- `resource()` et `httpResource()` :
 - Paramètres réactifs, `value()`, `status()`, `isLoading()`, `error()`
 - Annulation automatique des requêtes obsolètes
 - Reload manuel et invalidation
- `debounced()` (Angular 22) – debounce signal-natif sans RxJS
- Patterns : barre de recherche avec autocomplétion, pagination réactive

Le pont avec RxJS – quand et pourquoi

- `toSignal()` / `toObservable()` : conversions et pièges courants
- Cas où RxJS reste pertinent : WebSockets, streams complexes, opérateurs métier
- Cas où les Signals suffisent : état UI, dérivations simples, HTTP standard

Exemples de travaux pratiques (à titre indicatif)

- Construction d'une barre de recherche avec `debounced()` et `httpResource()`
- Gestionnaire d'état local complet avec `signal`, `computed`, `linkedSignal` et `effect`

Jour 2 – Matin

Gestion d'état globale : NgRx Signal Store et NgRx classique

Choisir sa stratégie d'état

- Signals locaux pour l'état d'un composant
- Services et Signals pour l'état partagé léger
- NgRx Signal Store pour des stores complexes signal-natifs
- NgRx classique (Store + Effects) pour les architectures Flux / Redux à grande échelle

NgRx Signal Store

- `signalStore`, `withState`, `withMethods`, `withComputed`, `withHooks`
- Plug-ins : `withEntities`, `rxMethod`
- Intégration native avec les Signals d'Angular 21 / 22
- Stores feature et stores root

NgRx 19+ classique (compatible Angular 21 / 22)

- Pattern Flux revisité avec API standalone : `provideStore`, `provideEffects`, `provideState`
- Stores, states, actions, reducers, effects, selectors
- `createFeature()` et `createActionGroup()`
- Quand préférer NgRx classique à Signal Store (effets complexes, time-travel debugging, équipes habituées au pattern)

Exemple de travaux pratiques (à titre indicatif)

- Implémentation d'un panier e-commerce avec NgRx Signal Store, puis comparaison avec une approche NgRx classique

Jour 2 – Après-midi

Routage avancé et sécurisation OAuth2 / Keycloak

Routage avancé

- API de routage moderne :
 - `provideRouter()` avec features (`withComponentInputBinding`, `withViewTransitions`, `withInMemoryScrolling`)
 - Organisation modulaire facilitée, fichiers `*.routes.ts` dédiés
 - Routes externes lazy-loaded (Angular 22)
- Guards et resolvers fonctionnels :
 - Fonction `inject()` dans les guards / resolvers
 - `CanActivateFn`, `CanMatchFn`, `ResolveFn`
 - Redirections asynchrones (retour Promise ou Observable)
- Lazy loading granulaire avec `loadComponent()` et `loadChildren()`
- Routes secondaires (named outlets), routes relatives, paramètres typés
- Préchargement personnalisé (`PreloadAllModules`, stratégies custom)

Sécurisation avec Keycloak / OAuth2

- Présentation des protocoles OAuth2 et OpenID Connect
- Bibliothèques : `keycloak-angular` et `oidc-client-ts`
- HTTP interceptor signal-aware pour l'injection du token
- Patterns guard et resolver pour les routes protégées
- Refresh token et gestion de session
- Gestion des rôles et permissions

Exemple de travaux pratiques (à titre indicatif)

- *Application multipage avec guards fonctionnels, lazy loading et flow d'authentification OIDC complet (login, refresh token, logout, route protégée par rôle)*

Jour 3 – Matin

SSR et hydratation incrémentale

Server-Side Rendering avec Angular

- SSR activé par défaut pour les nouveaux projets (`ng new --ssr`)
- Architecture : Express server, `provideServerRendering()`, `provideClientHydration()`
- Routes statiques (SSG) vs SSR dynamique vs CSR – comment choisir
- Pré-rendu au build avec `prerender`

Hydratation incrémentale (stable en Angular 21)

- `@defer` (hydrate on viewport) – hydrater quand le composant entre dans le viewport
- `@defer` (hydrate on interaction) – hydrater au premier clic / hover
- `@defer` (hydrate on idle) – hydrater quand le navigateur est libre
- Event replay automatique pendant l'hydratation

- Mesurer l'impact : Core Web Vitals (LCP, INP, CLS)

Stratégies de déploiement

- Node.js standard (Docker, PM2)
- Edge runtimes : Cloudflare Workers, Vercel, Netlify
- Mise en cache et stratégies CDN

Pièges classiques et debug

- Différences DOM serveur vs client (hydratation mismatch)
- Accès window, document, localStorage dans un contexte SSR
- Gestion des cookies et de l'authentification côté serveur

Exemple de travaux pratiques (à titre indicatif)

- Activation du SSR avec hydratation incrémentale sur l'application du jour précédent, mesure des Core Web Vitals avant / après

Jour 3 – Après-midi

Accessibilité, internationalisation et PWA

Angular Aria (@angular/aria) — Developer Preview en Angular 21

- Bibliothèque de directives headless accessibles construite par l'équipe Angular Material
- Conformité WAI-ARIA : gestion clavier, attributs ARIA, focus management, lecteurs d'écran
- Composants disponibles : Accordion, Autocomplete, ComboBox, Grid, ListBox, Menu, Multiselect, Select, Tabs, Toolbar, Tree
- Construction de composants stylés sur mesure à partir d'Angular Aria
- Comparaison avec Angular Material et le CDK (a11y, overlay, drag & drop)
- Conformité réglementaire : EAA (European Accessibility Act, juin 2025), RGAA 4.x, WCAG 2.2 AA
- Tests d'accessibilité : axe-core, jest-axe

Internationalisation (i18n)

- Pipeline @angular/localize :
 - Extraction automatisée (ng extract-i18n)
 - Formats supportés : XLIFF 1.2 / 2.0, XMB, JSON
 - Workflow de traduction et build multi-locales
- Pipes de régionalisation : DatePipe, CurrencyPipe, NumberPipe
- \$localize runtime vs i18n compile-time
- Alternatives populaires : ngx-translate, transloco

Progressive Web Apps (PWA)

- Génération via ng add @angular/pwa
- Service worker Angular : stratégies de cache (prefetch, lazy, freshness)
- SwUpdate : notification de nouvelle version aux utilisateurs

- SwPush : push notifications
- Stratégies hors-ligne, fallback, gestion des assets dynamiques

Exemples de travaux pratiques (à titre indicatif)

- Construction d'un ComboBox accessible custom à partir d'Angular Aria
- Internationalisation d'une application en 3 langues
- Création d'une PWA avec stratégie de cache custom et SwUpdate

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 16 décembre 2026 au 18 décembre 2026 (3 jours / 21 heures)

Strasbourg

Du 16 décembre 2026 au 18 décembre 2026

Strasbourg

- 16 décembre 2026 ----- 9h00 – 16h00
- 17 décembre 2026 ----- 9h00 – 16h00
- 18 décembre 2026 ----- 9h00 – 16h00