Formation Angular - Avancé

Code : DEV-ANGULAR-200 Durée : 3 jours (21 heures)

Public visé

Développeurs Angular et chefs de projets.

{{< formation-prerequis >}}

Objectifs pédagogiques

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

- Définir le framework Angular, de manière approfondie

Programme

Jour 1 - Matin

Rappel sur le fonctionnement d'Angular

- Vue d'ensemble de l'architecture actuelle :
- Préférence pour les composants autonomes ("standalone")
- Simplification du bootstrapping
- Présentation de l'état de l'écosystème :
- Usages actuels des modules
- Directives
- Pipes
- Services adaptés à Angular 20

Zone.js et Mode "Zoneless"

- Zone.js : explication de son rôle historique pour la détection de changements
- Mode zoneless désormais au coeur de l'architecture Angular 20 :
- Angular permet de s'affranchir de Zone.js pour un rendu plus rapide, moins de re-rendus inutiles et un contrôle manuel de la détection de changement grâce aux signals ou en déclenchant explicitement la détection
- Zoneless devient le mode recommandé (encore en "developer preview" mais déployable sur de vrais projets)

Nouveautés de la version 20

- Signal-based reactivity stable :

- Signaux réactifs
- Gestion manuelle et efficace du cycle de vie et des mises à jour d'UI sans Zone.js
- API de création dynamique de composants stabilisée createComponent(), directement compatible avec la nouvelle réactivité et les standalone components
- Syntaxe de template enrichie (opérateur **, template literals, opérateur in dans les expressions)

Rappel sur les "standalone components"

- Les "standalone components" sont pleinement matures depuis Angular 17+ :
- Exclusion de NgModule
- Simplification des dépendances
- Autonomie maximale

Exemple de travaux pratiques

- Création d'une application sur Angular 20, exploration des composants « standalone »

Jour 1 - Après-midi

Gestion avancée de l'état avec NgRx

- NgRx 19 (compatible Angular 20) est la version actuelle, v20 est en préparation mais la v19 reste 100% compatible
- Pattern Flux revisité avec la nouvelle syntaxe "standalone" (API provideStore, provideEffects...)
- Stores, states, actions, effects : intégration simplifiée aux nouveaux composants autonomes et signaux
- Recommandation d'utiliser signals localement, et NgRx pour le state partagé / applications complexes

Exemple de travaux pratiques

- Développement d'un petit gestionnaire d'état avec NgRx, nouvelle intégration sans NgModule via les nouveaux providers

Jour 2 - Matin

Routage avancé

- API de routage évoluée :
- Déclaration avec provideRouter
- Organisation modulaire facilitée
- Prise en charge étendue des secondary routes, routes relatives, options avancées (scroll smooth, options natives)
- Guards, resolvers accessibles et customisables
- Asynchronous redirects : fonction de redirection asynchrone (retourne une Promise ou un Observable)

Exemple de travaux pratiques

- Création d'une application multipage intégrant des guards et le chargement asynchrone

Jour 2 - Après-midi

Angular Signals

- Signaux modifiables (writable signals) :
- Remplacement fluide de la gestion d'état locale au composant
- Signaux calculés (computed signals) :
- Réactivité fine

- Dépendance propre
- Exclu les abonnements RxJS inutiles
- Inputs signalés entre composants (communication parent / enfant efficace, sans "@Input" classique)
- Les signaux gèrent la notification de template de façon automatique, sans surconsommation de ressources

Exemple de travaux pratiques

- Conception d'une application manipulant signaux modifiables, calculés, et passages des signaux

SSR (Server-Side Rendering)

- SSR activé par défaut sur projets nouveaux, prise en charge complète de l'hydratation incrémentale, event replay automatique pour une UX fluide même en SSR
- Configuration simplifiée du serveur pour le rendu dynamique et des pages statiques optimisées

Exemple de travaux pratiques

- Mise en place d'une application SSR, gestion d'évènements réhydratés côté client

Jour 3 - Matin

Les Progressive Web Apps (PWA)

- Génération PWA via le schematic @angular/pwa, assets et manifest gérés automatiquement
- Mise à jour du service worker :
- Nouveaux outils de notification aux utilisateurs
- Stratégies de cache configurables
- Meilleure intégration avec Angular CLI

Exemple de travaux pratiques

- Création d'une PWA avec gestion de cache évoluée, détection de nouvelle version et stratégie hors-ligne

Jour 3 - Après-midi

Internationalisation (i18n) et accessibilité

- Pipeline d'i18n maintenu et simplifié dans Angular 20 (extraction automatisée, support natif xlf, json...)
- Nouvelle gestion du pipe de régionalisation, meilleure gestion des formats (dates, monnaies...)
- Fonctionnement sur Ivy, compatibilité totale avec les composants standalone
- Renforcement de la prise en charge de l'accessibilité (composants natifs, directives, hooks d'accessibilité)

Exemple de travaux pratiques

- Internationalisation d'une application avec extraction, gestion de plusieurs langues, tests de compatibilité

Modalités d'évaluation des acquis

En cours de formation, par des études de cas ou des travaux pratiques. En fin de formation, par un questionnaire d'auto-évaluation.

{{< formation-require >}}

{{< formation-seealso >}}