EduGroupe
Accompagner pour réussir

Angular 20 et versions antérieures - Développement d'applications Web

Référence : **OPS006D** Durée : **3 jours (21 heures)** Certification : **Aucune**

CONNAISSANCES PRÉALABLE

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

PROFIL DES STAGIAIRES

- Développeurs et chefs de projets

OBJECTIFS

- Utiliser la version 20 du framework Angular
- Développer et tester complètement une application
- Appliquer les bonnes pratiques de développement

CERTIFICATION PRÉPARÉE

- Aucune

MÉTHODES PÉDAGOGIQUES

- Mise à disposition d'un poste de travail par stagiaire
- Remise d'une documentation pédagogique papier ou numérique pendant le stage
- La formation est constituée d'apports théoriques, d'exercices pratiques et de réflexions
- Le suivi de cette formation donne lieu à la signature d'une feuille d'émargement

FORMATEUR

- Consultant-formateur expert d'Angular

MÉTHODES D'ÉVALUATION DES ACQUIS

- Auto-évaluation des acquis par le stagiaire via un questionnaire
- Attestation des compétences acquises envoyée au stagiaire
- Attestation de fin de stage adressée avec la facture

ACCESSIBILITÉ DE LA FORMATION

- EduGroupe met en place un ensemble de dispositifs pour accueillir, accompagner et adapter ses formations aux personnes en situation de handicap (PSH).
- Notre référent(e) handicap se tient à votre disposition au 01.71.19.70.30 ou par mail à referent.handicap@edugroupe.com pour tout besoin d'aménagement, afin de vous offrir la meilleure expérience possible.

CONTENU DU COURS

1. JOUR 1 - Matin

2. Architecture, installation et premier test avec Angular 20

- Architecture moderne d'une application Angular 20 / Application standard (Modules) vs approche avec Standalone Components
- Installation de l'environnement avec Angular CLI v20
- Premier projet : création via CLI et déploiement local avec Vite (préconfiguré depuis Angular 17)
- TypeScript : Présentation rapide de TypeScript 5.x ; Types, classes, interfaces et modules ES ; Asynchrone : promesses et async / await
- Rôle des composants autonomes (standalone) - désormais fortement encouragés
- Introduction aux directives : @Component ; @Input ; @Output
- Architecture MVVM + zones d'exécution sans Zone.js (zone-less rendering activé par défaut)

3. Exemple de travaux pratiques (à titre indicatif)

- 💡 *Création d'un projet Angular 20 et exploration du code généré (fonctionnement avec main.ts, bootstrapApplication...)*

4. JOUR 1 - Après-midi

5. Création de la première application et initiation aux templates

- Démarrer "from scratch" avec ng new et composants standalone
- Création d'un composant à la main (@Component + bootstrapApplication)
- Les nouveaux templates syntaxiques : Interpolation ({{ }}) ; Property binding [prop] ; Event binding (event) ; Two-way binding [(ngModel)]
- Nouvelles directives structurelles @if, @for, @switch : Syntaxe plus intuitive (remplaçant *ngIf, *ngFor)
- Pipes intégrés et création de pipes personnalisés
- Création et injection de services (avec inject () recommandé)
- Injection contextuelle via les options de providers (provideService, provideHttpClient...)

6. Exemples de travaux pratiques (à titre indicatif)

- 💡 *Création d'une application avec des composants autonomes*
- 💡 *Manipulations de données via services et affichage via binding avancé*

7. JOUR 2 - Matin

8. Formulaires Angular avec typage fort

- Types de formulaires :Template-driven vs Reactive
- Création de formulaires avec : FormsModule ; ReactiveFormsModule et FormBuilder
- Nouveauté Angular 17 / 18+ : formulaires fortement typés avec TypedFormGroup, FormControl
- Utilisation optimisée : Gestion de la validation ; Erreurs ; Affichage dynamique
- Mise en pratique : formulaire de login ou entrée produit

9. Exemple de travaux pratiques (à titre indicatif)

- 💡 *Création de formulaires réactifs modernes avec typage strict*

10. JOUR 2 - Après-midi

11. HTTP, RxJS et accès au backend

- Présentation de RxJS 8+ : Nouveaux opérateurs simplifiés ; Différences entre Promises, Observables et Signals
- Requêtes HTTP via la nouvelle API HttpClient avec provideHttpClient ()
- Utilisation de HttpClient.get, post, interceptors, catchError.
- Présentation de httpResource
- Accès aux API REST
- Présentation de async / await vs subscribe dans les composants

12. Exemple de travaux pratiques (à titre indicatif)

- 💡 *Intégrer une API REST externe dans un service et affichage dans le composant (avec ngIf / @if)*

13. JOUR 3 - Matin

14. Routage modulaire et composants autonomes

- Le Router Angular 20 : Déclaration avec provideRouter () ; Définition des routes dans fichiers séparés ; Techniques d'organisation modulaire des routes (lazy loading, guards, resolvers...)
- Rappel : avec les composants standalone, le routage est simplifié
- Le concept de SCAM (Single Component Angular Module) : obsolète depuis Angular 15+
- Chargement différé de composants via loadComponent ()

15. Exemples de travaux pratiques (à titre indicatif)

- 💡 *Configuration d'un router avec plusieurs vue (home, about, produit)*
- 💡 *Navigation avec routerLink et paramètres dynamiques*

16. JOUR 3 - Après-midi

17. Les signals (Angular Reactive Signals API)

- Présentation des signals introduits depuis Angular 16+ : Signal(), computed(), effect() ; Réactivité sans RxJS, plus fluide et plus ergonomique
- Comparaison Signals vs Observables
- Cas d'usage : Intégration dans les composants ; Gestion de l'état

18. Les tests dans Angular 20

- Unités : avec Jasmine / Karma ou Jest (optionnel)
- Tests de composants standalone
- Nouveautés Angular 20 : simplification de la configuration des tests pour composants autonomes
- E2E testing amélioré avec Playwright (préconisé) au lieu de Protractor

19. Exemples de travaux pratiques (à titre indicatif)

- *Création d'un signal et observation des résultats dynamiques*
- *Tests unitaires de service et composant*
- *Test E2E avec Playwright : script de login ou navigation*