Formation : Angular, maîtriser le Framework Front-End de Google

Angular est le framework javascript de référence de Google. Il utilise tous les standards du Web. Il offre des performances accrues avec une conception modulaire adaptée à la mobilité ainsi qu'une amélioration de la productivité de vos équipes de développement. Angular bénéficie immédiatement d'un écosystème riche et d'une communauté toujours plus grande.

Objectifs pédagogiques

- À l'issue de la formation, le participant sera en mesure de :
- Organiser, modulariser et tester ses développements JavaScript
- Maîtriser les fondamentaux du Framework Angular
- Créer rapidement des applications Web complexes
- Savoir intégrer les tests unitaires au développement
- Connaître les bonnes pratiques de développement et de mise en production

Public concerné

Architectes, développeurs et chefs de projets Web.

Prérequis

Bonnes connaissances des technologies du Web et des outils modernes de développement Front-End. Connaissances de JavaScript.

Vérifiez que vous avez les prérequis nécessaires pour profiter pleinement de cette formation en faisant ce test.

Programme de la formation

Développement JavaScript : rappels

- Historique évolution de JavaScript.
- Bonnes pratiques ECMAScript 5.
- ES13/ES6/ES2015, présentation générale.
- Nouveautés syntaxiques : collection, template string, affectation destructurée, arrow functions, promesses, boucles...
- La programmation objet : La syntaxe de class.
- L'instruction spread.
- Le pattern observer/observable (ES7).

- TypeScript en détail, configuration.
- Les décorateurs en TypeScript.
- Les Webcomponents.
- "Modules Loaders" : Webpack, "import/export".

Travaux pratiques

- Mise en œuvre de l'environnement.

Les fondamentaux d'Angular

- Structure de l'application.
- Les composants.
- Les modules Angular, "core" et principaux modules.
- Les templates et les styles.
- Utiliser Angular CLI.
- Les différents types de binding.
- Encapsulation des vues et des styles.
- Les filtres, filtres personnalisés.
- Les directives, directives personnalisées.
- Variables de template.
- Les décorateurs et leurs propriétés en détails.
- Les évènements.
- Les formulaires : Reactive Forms, FormControl et FormGroup.
- La validation des formulaires.

Travaux pratiques

- Réalisation d'une application interactive. Exemple du Tchat.

La programmation réactive

- Configuration d'une application RxJS.
- Les observables.
- Les souscriptions.
- Les opérateurs.
- Les Hots observables.
- Les observable d'ordre 2.

Travaux pratiques

- Créer une page RxJs pour tester chacun des points ci-dessus.

Approfondissement des composants

- Les décorateurs @Host, @ContentChild, @ViewChild.

- Echanges de paramètres entre composants.
- Le cycle de vie des composants et la communication entre composants.
- Les Inputs et les Outputs.
- L'encapsulation et L'internationalisation : component locale.
- Le Bootstrapping.
- Les services et l'injection de dépendance.
- Providers et hiérarchie des injecteurs.
- Injection de service dans un service.
- Utilisation des services : Communication entre composants.
- Les observables et les subjects et les Modules, RouterModule, SharedModule.

Travaux pratiques

- Tester dans l'application interactive chacun des points ci-dessus

"Routing" et requête HTTP

- Introduction aux requêtes HTTP.
- Créer des requêtes HTTP .
- Utilisation d'un cloud externe.
- Les intercepteurs.
- Headers et gestion des erreurs.
- Gestion de routes dans l'arbre des composants.
- Les directives RouterLink et RouterLinkActive.
- Les services ActivateRoute et Router.
- Transmission de paramètres dans les routes.
- Utilisation de QueryParams et Fragment et Imbrication de routes.
- Intercepter les paramètres de routage et wildcard.
- Configuration des "Guard" pour le l'initialisation des routes.

Travaux pratiques

- Reprendre l'application, intégrer les points détaillés dans le chapitre. Consommation d'une API REST.

Tests unitaires. Bonnes pratiques et outils

- Configurer l'environnement de test.
- Présentation de Karma et de Jasmine.
- Ecrire les tests avec Jasmine.
- Ecrire des tests d'intégration avec protractor.
- Couverture du code. Indices du code-coverage.
- BDD Behaviour driven development, TDD Test driven development.
- Cas de test : pipe, composant, application.

Travaux pratiques

- Reprendre l'application interactive, intégrer les points détaillés dans le chapitre. Mise en œuvre du Test driven development.

Déployer une application avec Angular

- Gestion de l'authentification et des habilitations.
- Application isomorphique.
- Préparer les fichiers pour le déploiement.
- Une commande pour les préparer tous.
- Déployer sur un serveur web.
- Optimiser les fichiers en production.

Travaux pratiques

- Livraison en production du projet interactif.