Tél : +352 27 39 351
Fax : +352 27 39 52 68
Email : formation@oxiane.lu

Testing Angular Pragmatique

Apprendre à tester vos applications Angular avec une stratégie pragmatique et efficace

L'unit-testing ou plus particulièrement l'unit-testing front-end est indéniablement dans le top 3 des sujets les plus complexes associés au développement d'applications Angular. Face à cette complexité, il est tentant de renoncer mais à quel prix ? Certes, il est possible de développer une application sans tests unitaires mais est-ce réellement plus rapide ? Qu'advient-il de la maintenabilité du code quelques mois ou à peine quelques semaines plus tard ? Pourrons-nous refactoriser l'application et la déployer régulièrement en toute sérénité au fur et à mesure qu'elle se complexifie et qu'elle vieillit entre nos mains ?

Ce workshop présente à travers des exercices pratiques, des techniques pragmatiques d'unit-testing de composants et de services Angular. Vous y découvrirez également comment écrire des tests compréhensibles, maintenables et surtout rentables.

Détails

- Code : DW-TAP
- Durée : 2 jours (14 heures)

Public
- Développeurs Angular

Pré-requis
- Une expérience de testing sera bénéfique afin de bien saisir les bonnes pratiques et astuces exposées lors de cette formation

Objectifs
- Maîtriser le coût des tests et adopter une stratégie de testing pragmatique et rentable
- Implémenter des tests unitaires pour vos applications Angular avec l'approche Test-Driven Development
- Choisir le bon type de tests à implémenter en fonction du contexte
- Découpler les tests du code pour faciliter le refactoring
- Implémenter des tests unitaires maintenables et compréhensibles

Programme

Introduction au testing
- Pourquoi tester ?
- Les différents types de test
- Définition d'une unité de code
- Un exemple de test unitaire

Jest
- Avantages et inconvénients
- Mise en place de Jest

Test-Driven Development
- Origines et intérêt du Test-Driven Development
- Progressive TDD
- *Exercice : Tester un service Angular avec l'approche Progressive TDD*
- *Exercice : Debugging avec Jest*
- Astuces et bonnes pratiques

Timeboxed TDD & TCR
- Refactor vs. Rewrite: retour aux définitions
- Timeboxed TDD
- Limbo
- TCR : Test && Commit || Revert

Test Doubles
- Mocks vs Spies vs Stubs vs Fakes
- Classicists vs Mockists
- Choisir la bonne approche pour "overrider" l'injection de dépendance Angular
- Type-Safe testing
- Choosing the right way to override Angular dependency injection
- *Exercice : Simuler le comportement d'une dépendance avec Mocks, Spies & Stubs*

Component Testing
- Tester un composant
- Les différents types de testing de composant : Integration vs. Shallow vs. Isolated
- Interaction avec le DOM
- *Exercice : Shallow testing*
- *Exercice : Integration testing*
- *Exercice : Testing des inputs / outputs*
- *Exercice : Interaction avec les formulaires*

Angular CDK Test Harness
- L'histoire d'Angular CDK Test Harness
- Fonctionnement
- *Exercice: Utilisation d'un test harness*
- *Exercice: Implémentation d'un test harness*

Tél : +352 27 39 351
Fax : +352 27 39 52 68
Email : formation@oxiane.lu

Cypress Component Testing

- Brisons les frontières de l'"isolated testing" avec Cypress Component Testing
- *Exercice : Isoler et tester un composant avec Cypress*
- *Exercice : Réutiliser le test harness*

Visual Regression Testing

- Tester la présentation
- Détecter les régressions visuelles avec Cypress Component Testing & Percy

Définir une Stratégie de Testing

- Que faut-il tester et comment ?

Modalités

- Type d'action :Acquisition des connaissances
- Moyens de la formation :Formation présentielle – 1 poste par stagiaire – 1 vidéo projecteur – Support de cours fourni à chaque stagiaire
- Modalités pédagogiques :Exposés – Cas pratiques – Synthèse
- Validation :Exercices de validation – Attestation de stages