Ducin IT Consulting

JavaScript Automated Testing

Ducin IT Consulting - Program szkolenia

Czas trwania: **2-3 dni**
Formuła: **25% teoria, 75% praktyka**

Szkolenie ma na celu przybliżyć - w zależności od potrzeb grupy - podstawowe oraz zaawansowane zagadnienia związanie z testowaniem automatycznym kodu JavaScriptowego. Kładzie nacisk nie tylko na poprawne pisanie testów, ale i wychwytywanie krytycznych fragmentów kodu, które najbardziej warto pokryć testami.

Praktyczne przykłady realizowane są m.in. w oparciu o TDD oraz BDD. Uczestnicy uczą się technik testowania Unitowego, Integracyjnego oraz End- to- End oraz analizują, dlaczego tzw. "piramida testów" ma taki, a nie inny kształt. Testowane są różnorodne elementy systemów - takie jak logika biznesowa, asynchroniczne żądania, timery, warstwa wizualna DOM - a także zaawansowane techniki takie jak keszowanie, restartowanie żądań i wiele, wiele innych. Testowanie End- to- End uwzględniają pokrycie testami istniejącej, popularnej aplikacji.

Szkolenie zakłada co najmniej podstawową wiedzę z zakresu JavaScriptu.

Kluczowe Aspekty:

- Opracowywanie strategii przekrojowego testowania aplikacji
- Techniki pisania testów skutecznych i relatywnie tanich w utrzymaniu
- Kluczowe techniki mockowania, stubowania - wspomagające pisanie szybkich i precyzyjnych testów unitowych
- ...po tym szkoleniu polubisz testowanie!

Ducin IT Consulting

Program szkolenia:

1. Piramida Testów
    1.1. Testy: unitowe, integracyjne, end-to-end
    1.2. Aberracje klasycznej piramidy testów
    1.3. Testing Trophy
    1.4. Smote tests, Regression tests, Acceptance tests
2. Implementacja
    2.1. Struktura testów
    2.2. Setup, Teardown
    2.3. Arrange, Act, Assert
    2.4. Testy sync i async
    2.5. Test-Driven Development
    2.6. Data-Driven Tests
    2.7. Snapshot Tests
    2.8. Assert Object Pattern
3. Automatyzacja/Tooling
    3.1. Code Coverage
    3.2. CI/CD
    3.3. Testowanie statyczne: ESLint rules
4. Asercje
    4.1. chai assertions: should, expect, assert
    4.2. testing-library assertions
    4.3. custom assertions in jest/jasmine
5. Kryteria wartościowych testów

Ducin IT Consulting

- 5.1. Zaufanie do testów
- 5.2. FIRST
- 5.3. Testowanie zachowań, zamiast stanu i implementacji
- 5.4. Wyszukiwanie obszarów, które warto testować
- 5.5. Czytelność, intencja
- 5.6. Kruchość/stabilność testów
- 5.7. Happy & Sad Paths, Exceptions
- 5.8. False Positives, False Negatives
- 5.9. Behavior-Driven Development
- 5.10. "System Under Test"

6. Zaawansowane techniki testowania

- 6.1. Property Testing
- 6.2. Testy Mutacyjne

7. Testowanie vs zależności

- 7.1. spies, stubs, mocks
- 7.2. mocked imports
- 7.3. mocked HTTP: fetch, axios, msw
- 7.4. Don't care, don't specify

8. Testowanie

- 8.1. testowanie DOM
- 8.2. testowanie operacji asynchronicznych
- 8.3. testowanie logiki biznesowej
- 8.4. testowanie komponentów UI
- 8.5. testowanie UI: unitowe vs integracyjne vs E2E
- 8.6. testowanie integracyjne procesów biznesowych

9. Testowanie End-to-End

- 9.1. Cypress
- 9.2. Puppeteer
- 9.3. Page Objects

10. Testowanie Accessibility

- 10.1. Cele, korzyści, koszty
- 10.2. A11Y: WCAG
- 10.3. Przykłady złamanych reguł
- 10.4. aXe

11. Visual Regression Testing

- 11.1. Cele, korzyści, koszty
- 11.2. Storybook
- 11.3. Pixelmatch, Resemble
- 11.4. Snapshots, BackstopJS

12. Narzędzia

- 12.1. Istanbul/NYC
- 12.2. Mocha
- 12.3. Jest
- 12.4. JSDOM
- 12.5. Karma
- 12.6. Jasmine
- 12.7. Sinon.js
- 12.8. Chai
- 12.9. ajv
- 12.10. Puppeteer
- 12.11. Cypress
- 12.12. aXe