

JavaScript dla Programistów Backend

Ducin IT Consulting - Program szkolenia

Czas trwania: 3-5 dni

Formuła: 50% wykłady, 50% ćwiczenia

Uczestnicy po szkoleniu:

- Będą doskonale rozumieli działanie języka JavaScript oraz jego różnice względem większości innych języków.
- Będą omijali pułapki, w które najczęściej wpadają początkujący JavaScriptowcy; będą znali dobre praktyki.
- Będą rozumieli architektury oraz nowoczesne podejścia do tworzenia aplikacji JavaScriptowych; znając różnice pomiędzy bibliotekami i frameworkami.
- Będą umieli wybrać właściwe narzędzia dla określonego projektu.
- Będą umieli zaprojektować duże, skalowalne aplikacje, wskazać ich wrażliwe punkty, optymalizować – a także efektywnie pracować z kodem legacy.
- Będą mieli rozeznanie w narzędziach JavaScriptowych.

Kluczowe Aspekty:

- Zawiera wzorce i najlepsze praktyki
- Nacisk na myślenie funkcyjne zgodnie z filozofią JS
- Trener jest praktykiem z wieloletnim wcześniejszym doświadczeniem w backend

Program szkolenia:

1. Język JavaScript

1.1. Pułapki JavaScriptowe

- 1.1.1. Najczęściej napotymane błędy
- 1.1.2. Funkcje: zakres leksykalny vs blokowy
- 1.1.3. Koercja - rzutowanie typów
- 1.1.4. OOP: prototypy vs klasy
- 1.1.5. Asynchroniczne wykonywanie kodu

1.2. Funkcje - kluczowy element języka

- 1.2.1. Context (this)
- 1.2.2. Hoisting
- 1.2.3. Closures/domknięcia - mechanika, cele, korzyści

1.3. Asynchroniczność

- 1.3.1. 3 modele programowania: synchroniczny, asynchroniczny, współbieżny
- 1.3.2. Synchroniczne oraz asynchroniczne callbacki
- 1.3.3. Event Loop, Run to Completion Rule
- 1.3.4. Race Conditions
- 1.3.5. Patterns: Callbacks, Events, Promises, Coroutines, RxJS

1.4. Standard ECMAScript6+ - najważniejsze elementy

- 1.4.1. Rozszerzenia składni
- 1.4.2. Moduły
- 1.4.3. Generatory

2. Architektura

2.1. Style architektoniczne

- 2.1.1. WebComponents, dyrektywy angularowe
- 2.1.2. Nowoczesna architektura komponentowa
- 2.1.3. Komunikacja poprzez eventy
- 2.1.4. Sposoby zarządzania stanem

2.2. Wzorce projektowe

- 2.2.1. Dependency Injection
- 2.2.2. Model-View-ViewModel
- 2.2.3. Data binding (angular, knockout)
- 2.2.4. Alternatywne implementacje warstwy modelu
- 2.2.5. Modularność aplikacji

2.3. Przegląd architektur frameworków/bibliotek JavaScriptowych

- 2.3.1. jQuery
- 2.3.2. AngularJS
- 2.3.3. React / Redux
- 2.3.4. RxJS

2.4. Programowanie funkcyjne

- 2.4.1. Podstawowe operatory
- 2.4.2. Pure functions
- 2.4.3. Data immutability

3. Opcjonalne moduły

3.1. Przeglądarki

- 3.1.1. Zaawansowane użycie devtools
- 3.1.2. Wydajność aplikacji - analiza, optymalizacja
- 3.1.3. Wsparcie standardów, polyfills

3.2. Node.js, automatyzacja, bundling

- 3.2.1. Ekosystem współczesnych projektów JavaScriptowych
- 3.2.2. npm
- 3.2.3. grunt, gulp
- 3.2.4. AMD: require.js
- 3.2.5. CommonJS: browserify
- 3.2.6. Webpack
- 3.2.7. ES6+: Babel
- 3.2.8. Automated tests: karma, jasmine, mocha, jest

3.3. TypeScript - statyczne typowanie w JavaScript

3.3.1. Kluczowe założenia TypeScripta

3.3.2. Podstawowe typy

3.3.3. Type inference

3.3.4. Duck typing

3.3.5. Wsparcie narzędzi

3.3.6. Problemy jakie TS rozwiązuje, jakich nie rozwiązuje oraz jakie tworzy

3.3.7. Specyficzne dla TypeScripta wzorce projektowe

3.4. Zaawansowane promisy

3.4.1. Łączenie

3.4.2. Obsługa błędów

3.4.3. Zaawansowane operacje (all, race, any, some)

3.4.4. Wzorce projektowe oparte o promisy

3.4.5. Antywzorce

4. Testowanie

4.1. Statyczne

4.2. Unitowe

4.3. E2E