

# Reactive Functional Programming

# with RxJS

Ducin IT Consulting - Program szkolenia

Czas trwania: 3 dni

Formuła: 40% wykłady, 60% ćwiczenia

Szkolenie przeznaczone dla osób mających przynajmniej podstawową wiedzę o JavaScripcie. Poświęcone jest nowoczesnemu podejściu do tworzenia asynchronicznego kodu JavaScript – Reaktywnemu Programowaniu Funkcyjnemu na bazie biblioteki RxJS. Kładzie nacisk na zrozumienie filozofii tego podejścia, z uwagi na znaczące różnice z programowaniem imperatywnym i obiektowym. Uczestnicy szkolenia uczą się projektować aplikacje i rozwiązania z użyciem strumieni, dokonując na nich mnóstwo rozmaitych operacji.

W trakcie szkolenia "krystalizują się" klasy problemów i dobierane są do nich odpowiednie rozwiązania: operatory i ich kompozycje, subjecty oraz sposoby przepływu kontroli aplikacji. Zrozumienie istoty FRP oraz realizacja wielu ćwiczeń umożliwia pójście krok dalej – przegląd wzorców rozwiązań oraz architektur aplikacji opartych o strumienie reaktywne.

Podczas szkolenia kładziemy duży nacisk zarówno na zrozumienie istoty omawianych zagadnień, kodowanie własnych rozwiązań, jak i pracę w grupie.

# Kluczowe Aspekty:

- Paradygmat programowania funkcyjnego i reaktywnego
- Poznawanie RxJS od strony praktycznej liczne ćwiczenia ze strumieniami, operatorami i subjectami
- Architektura aplikacji opartych o RxJS oraz wzorce rozwiązań



# Program szkolenia:

# 1. JavaScript Functional Programming

- 1.1. Functions, Function Objects
- 1.2. Contexts
- 1.3. Scopes: Function vs Lexical
- 1.4. Closures, Currying
- 1.5. Reducers, Higher Order Functions
- 1.6. Pure Functions, Side Effects

# 2. (opcjonalnie) Asynchrony

- 2.1. 3 Programming Models: Synchronous, Asynchronous, Parallel
- 2.2. JavaScript inside Browsers and Node.js
- 2.3. Concurrency in JavaScript
- 2.4. Event Loop, WEB APIs
- 2.5. Run to Completion Rule
- 2.6. Race Conditions
- 2.7. Patterns: Callbacks, Events, Promises, RxJS

## 3. Stream-based FRP

- 3.1. Functional Reactive Programming
- 3.2. Pull-based vs Push-based
- 3.3. Observable Pattern
- 3.4. Marble Diagrams

#### 4. Streams

- 4.1. Creating Streams
- 4.2. Managing Subscriptions



#### 4.3. Operators

- 4.3.1. Manipulating Data
- 4.3.2. Manipulating Time
- 4.3.3. Backpressure
- 4.3.4. Flattening
- 4.4. Higher Order Observables
- 4.5. Error Handling
- 4.6. Debugging
- 4.7. RxJS 5.5
  - 4.7.1. let operator
  - 4.7.2. pipe operrator

# 5. Subjects

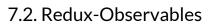
- 5.1. Hot and Cold Observables
- 5.2. Współdzielenie subskrypcji
- 5.3. Multicast, Unicast
- 5.4. Different Subjects
  - 5.4.1. Subject
  - 5.4.2. Behavior Subject
  - 5.4.3. ReplaySubject
  - 5.4.4. AsyncSubject

### 6. Architecture

- 6.1. Inversion of Control w RxJS
- 6.2. Real-time Apps
- 6.3. Strumienie vs Webservices (HTTP oraz Websockety)
- 6.4. Best Practices
- 6.5. Antipatterns

# 7. (opcjonalnie) RxJS + Redux

7.1. Redux Effects, effect-based application flow



# Consulting

- **7.3. NGRX**
- 7.4. Budowa aplikacji w oparciu o RxJS + Redux-Observables lub NGRX

# 8. (opcjonalnie) RxJS + nowy Angular

- 8.1. Callback-based components vs stream-based components (React vs Angular)
- 8.2. Component Inputs & Outputs jako strumienie
- 8.3. EventEmitter
- 8.4. Change Detection, strategia On Push
- 8.5. Observables vs Promises: differences & similarities
- 8.6. Budowa aplikacji w oparciu o RxJS + Angular

Ducin IT Consulting | tel. +48 500 260 759 | ducin.it | tomasz@ducin.it | Strona: 4/4