BOTTEGA IT minds

# Program szkolenia:

# Functional Reactive Programming z RxJS

## Informacje:

| | |
|---|---|
| **Nazwa:** | **Functional Reactive Programming z RxJS** |
| **Kod:** | **frp-rxjs** |
| **Kategoria:** | JavaScript |
| **Odbiorcy:** | developerzy, architekci |
| **Czas trwania:** | 3 dni |
| **Forma:** | 40% wykłady / 60% warsztaty |

Szkolenie przeznaczone dla osób majacych przynajmniej podstawowa wiedze o JavaScripcie. Poświecone 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 znaczace róznice z programowaniem imperatywnym i obiektowym.

Uczestnicy szkolenia ucza sie projektować aplikacje i rozwiazania z uzyciem strumieni, dokonujac na nich mnóstwo rozmaitych operacji. W trakcie szkolenia „krystalizuja sie" klasy problemów i dobierane sa do nich odpowiednie rozwiazania: operatorami, subjecty oraz sposoby przepływu kontroli aplikacji.

Zrozumienie istoty FRP oraz realizacja wielu ćwiczeń umozliwia pójście krok dalej – przeglad wzorców rozwiazań oraz architektur aplikacji opartych o strumienie.

Podczas szkolenia kładziemy duzy nacisk zarówno na zrozumienie istoty omawianych zagadnień, kodowanie własnych rozwiazań, jak i prace w grupie.

# Zalety szkolenia:

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

BOTTEGA
IT minds

# Szczegółowy program:

## 1. JavaScript Functional Programming

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

## 2. Asynchrony

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

## 3. Stream-based FRP

- 3.1. Functional Reactive Programming
- 3.2. Observable Pattern
- 3.3. Marble Diagrams

## 4. Streams

- 4.1. Creating Streams
- 4.2. Managing Subscriptions
- 4.3. Operators: Manipulating Data
- 4.4. Operators: Manipulating Time

BOTTEGA IT minds

- 4.5. Operators: Backpressure
- 4.6. Operators: Flattening
- 4.7. Higher Order Observables
- 4.8. Error Handling
- 4.9. Debugging
- 4.10. RxJS 5.5: let, pipe

**5. Subjects**

- 5.1. Hot and Cold Observables
- 5.2. Współdzielenie subskrypcji
- 5.3. Multicast, Unicast
- 5.4. Subject classes:
  - 5.4.1. Subject
  - 5.4.2. BehaviorSubject o ReplaySubject
  - 5.4.3. 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. RxJS + Redux**

- 7.1. Redux Effects, effect-based application flow
- 7.2. redux-observables
- 7.3. ngrx
- 7.4. Budowa aplikacji w oparciu o RxJS + redux-observables lub ngrx

BOTTEGA IT minds

## 8. RxJS + Angular 2+

8.1. Callback-based components vs stream-based components (React vs Angular)

8.2. Component Inputs and Outputs – jako strumienie

8.3. EventEmitter

8.4. ChangeDetection, strategia OnPush

8.5. Observables vs Promises: differences and similarities

8.6. Budowa aplikacji w oparciu o RxJS + Angular2+