BO·TT·EGA
IT minds

# Program szkolenia:

# Asynchronous JavaScript

## Informacje:

| | |
|---|---|
| **Nazwa:** | **Asynchronous JavaScript** |
| **Kod:** | **JS-asynch** |
| **Kategoria:** | JavaScript |
| **Grupa docelowa:** | developerzy<br>architekci |
| **Czas trwania:** | 3 dni |
| **Forma:** | 50% wykłady / 50% warsztaty |

Szkolenie przeznaczone dla osób mających przynajmniej podstawową wiedzę o JavaScripcie. Poświęcone jest kluczowemu zagadnieniu języka - asynchroniczności, jej obsłudze, projektowaniu przepływu kontroli, skalowaniu. Przydatne jest również dla programistów z backgroundem backendowym, którzy nigdy nie zetknęli się (lub w niewielkim stopniu) z zagadnieniami, które w JavaScripcie są chlebem powszednim. Po treningu uczestnicy potrafią rozumieją analizowane problemy, potrafią je celnie zdiagnozować i dobrać odpowiednie rozwiązanie, zgodnie z tzw. best practices. Potrafią także zaprojektować skalowalne rozwiązania dla (bardzo) dużych aplikacji.

Zakres obejmuje przedstawienie klas typowych problemów asynchroniczności, wzorców rozwiązań, dobór narzędzi i wreszcie implementację.

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

# Zalety szkolenia:

- Podstawy programowania asynchronicznego w JavaScript
- Problemy asynchroniczności – kompleksowy przegląd rozwiązań
- Wzorce projektowe

BOTTEGA IT minds

# Szczegółowy program:

## 1. JavaScript Functional Programming

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

## 2. Asynchrony

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

## 3. Callbacks

- 3.1. Synchronous and Asynchronous Callbacks
- 3.2. Callback Hell
- 3.3. Events

## 4. Promises

- 4.1. Promise Design Pattern
- 4.2. States, State Transitions
- 4.3. Chaining
- 4.4. Values
- 4.5. Error Handling

BOTTEGA IT minds

4.6. Advanced Patterns and Usecases (Combining Promises)

4.7. Promise Anti-patterns

## 5. Promise Implementations

5.1. Promises/A+ Specification

5.2. ECMAScript 6: Promise, Arrow Functions, Destructuring

5.3. Bluebird Promises

5.4. jQuery Promises and Deferreds

## 6. RxJS

6.1. Reactive Functional Programming

6.2. Observable Pattern

6.3. Sequences, Operators, Marble Diagrams

6.4. Error Handling

6.5. Subjects

6.6. Schedulers

6.7. Observables vs Promises: differences and similarities