

# Expert TypeScript Workshop

# Static Typing for FE/BE JavaScript

Ducin IT Consulting - Program Szkolenia

Czas trwania: 3 dni

Formuła: 50% ćwiczenia, 25% teoria, 25% praca w grupie

Trening przygotowuje programistów do efektywnego stosowania TypeScriptu w projektach poprzez szukanie balansu pomiędzy wysokim poziomem bezpieczeństwa typów w kodzie, praktycznością rozwiązań i prostotą. Szkolenie przeznaczone zarówno dla programistów frontendowych jak i backendowych. Zakres obejmuje - w zależności od potrzeb grupy - od podstawowych problemów i ich rozwiązań, poprzez zaawansowane, aż po zagadnienia eksperckie.

Szkolenie kładzie nacisk na statyczne typowanie jako alternatywę do dynamiczności środowiska JavaScript - języka i budowanych w nim bibliotek. Poznajemy wzorce projektowe stosowane we frontendzie oraz implementują niektóre z nich. Wiele ćwiczeń realizowanych jest na podstawie zautomatyzowanych testów w stylu TDD.

Rozpoczynamy od ugruntowania wiedzy, wskazania elementów języka, które w dalszym ciągu nie są type- safe oraz wyjaśnienia reguł i ograniczeń, do których TS musi się dostosować. Następnie, tę wiedzę stosujemy do modelowania domeny oraz architektury aplikacji. Wskazujemy także dobre praktyki stosowania TS, dzięki którym w zgodzie z Open/Closed Principle, kod nawet po latach będzie otwarty na dodawanie nowych funkcjonalności, jednocześnie nie wymagając każdorazowej przebudowy przy okazji zmieniających się wymagań. Uwzględniane czynniki to elastyczność, łatwe refaktoringi w przyszłości i testowalność.

# Kluczowe Aspekty:

- Praktyczne stosowanie TypeScripta dla szybszych refaktorów i developmentu
- Poprawne rozumienie TypeScripta w odniesieniu do JavaScriptu, a także Javy czy C#
- Myślenie typami i programowanie przy uzyciu typów
- Wzorce projektowe, elementy DDD, architektura
- Najlepsze praktyki, częste błędy

Strona: 1/5



# **Program Szkolenia:**

#### 1. Introduction

- 1.1. TypeScript Principles, Design Goals and Limitations
- 1.2. JS evolution & TS evolution
- 1.3. TS Breaking Changes
- 1.4. Compile-time & Runtime
- 1.5. Type Safety

### 2. Type System

- 2.1. Static vs Dynamic typing
- 2.2. Strong vs Weak typing
- 2.3. Structural vs Nominal typing
- 2.4. Explicit vs Implicit typing
- 2.5. Type Inference
- 2.6. Duck typing
- 2.7. Control Flow Analysis
- 2.8. Type narrowing, Type Guards
- 2.9. Types as Sets

# 3. Built-in types & Type Compatibility

- 3.1. Primitive Types
- 3.2. Object Types
- 3.3. Function Types
- 3.4. Unions and Intersections
- 3.5. Aliases, Interfaces, Classes, Objects
- 3.6. Enums, Literals, Tuples, Index Signatures

Ducin IT Consulting | tel. +48 500 260 759 | ducin.dev | tomasz@ducin.dev | Strona: 2/5

Consulting

#### 3.7. Top and Bottom Types

### 4. Type Language

- 4.1. Mapped Types
- 4.2. Conditional Types
- 4.3. Generics
  - 4.3.1. Generic Types
  - 4.3.2. Class, Interface & Type Generics
  - 4.3.3. Function Generics
  - 4.3.4. Required vs Inferred Generics
  - 4.3.5. Parametrized Types vs Generic Types

#### 5. Type Programming

- 5.1. Filtering object type keys and values
- 5.2. Distributing unions
- 5.3. Recursive Generics
- 5.4. Combining Mapped, Conditional, Generics and Unions altogether

### 6. Type-safety

- 6.1. Soundness and Completeness
- 6.2. Type-unsafe operations in TS
- 6.3. Compiler "strict" flags
- 6.4. Variance (invariance, covariance, contravariance, bivariance)
- 6.5. Type-safe operation on external sources (e.g. HTTP)

# 7. TypeScript Classes

- 7.1. Interfaces vs Types
- 7.2. Classes, Decorators
- 7.3. OOP: abstraction, polymorphism, inheritance, encapsulation

Ducin IT Consulting | tel. +48 500 260 759 | ducin.dev | tomasz@ducin.dev | Strona: 3/5

# 8. Type Compatibility

Ducin IT Consulting

- 8.1. Primitives
- 8.2. Object Types
- 8.3. Function Types
- 8.4. Unions and Intersections, Mixed Types
- 8.5. Aliases, Interfaces, Classes, Objects

### 9. Object-Oriented Programming with TS

- 9.1. Applying SOLID Principles to TS
- 9.2. Encapsulation
- 9.3. Structural Polymorphism
- 9.4. Classes and Inheritance
- 9.5. Reusable Decorators (Parameters, Properties, Methods, Classes)

## 10. Functional Programming with TS

- 10.1. Typing Higher Order Functions with Generics
- 10.2. Functional Composition
- 10.3. Overloading Function Signatures

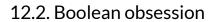
#### 11. Patterns and Practices

- 11.1. Type-safe State Machines
- 11.2. Domain Logic in Client Apps
- 11.3. Opaque/Brand types
- 11.4. VOs (Value Objects)
- 11.5. DTOs (Data Transfer Objects)
- 11.6. Single Source of Truth for types

# 12. Anti-patterns

12.1. Primitive obsession

Ducin IT Consulting | tel. +48 500 260 759 | ducin.dev | tomasz@ducin.dev | Strona: 4/5



12.3. Leaking types



Ducin IT Consulting | tel. +48 500 260 759 | ducin.dev | tomasz@ducin.dev | Strona: 5/5