Ducin IT Consulting

# Warsztat Ekspercki TypeScript

## Statyczne Typowanie na Frontendzie

Ducin IT Consulting - Program szkolenia

Czas trwania: **3 dni**
Formuła: **30% teoria, 50% ćwiczenia, 20% praca w grupie**

Trening przygotowuje programistów do efektywnego stosowania TypeScriptu w projektach poprzez szukanie balansu pomiędzy prostotą, elastycznością i bezpieczeństwem kodu.

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 dynamicznego JavaScriptu. Porównujemy implementacje dynamiczne i statyczne, przyglądając się zaletom płynącym z większej kontroli nad typami danych oraz kosztom, jakie niesie ze sobą stosowanie TypeScriptu w projektach o różnej skali. Poznajemy wzorce projektowe stosowane we frontendzie oraz implementują niektóre z nich. Wiele ćwiczeń realizowanych jest na podstawie zautomatyzowanych testów w stylu TDD.

Szkolenie przewiduje także zadania związane z projektowaniem aplikacji. Uwzględniane czynniki to elastyczność, łatwe refaktoringi w przyszłości i testowalność.

## Kluczowe Aspekty:

- Zrozumienie celów i strategii TypeScripta, ułatwiające intuicyjne stosowanie języka
- Poprawne rozumienie TypeScripta w odniesieniu zarówno do JavaScriptu, jak i Javy czy C#
- Wzorce projektowe, elementy DDD, architektura
- Najlepsze praktyki, częste błędy

Ducin IT Consulting

# Program szkolenia:

## 1. (Optional) Prerequisites

### 1.1. Chosen elements of ECMAScript

## 2. Introduction

### 2.1. TypeScript Design Goals and Philosophy

### 2.2. Compile-time & Runtime

### 2.3. Responsibilities of TypeScript: problems solved & unsolved

## 3. Type System

### 3.1. Static vs Dynamic typing

### 3.2. Strong vs Weak typing

### 3.3. Implicit vs Explicit typing

### 3.4. Type inference

### 3.5. Structural vs Nominal typing

### 3.6. Duck typing

### 3.7. Soundness and Completeness

### 3.8. Variance (invariance, covariance, contravariance, bivariance)

## 4. Type Compatibility

### 4.1. Primitives

### 4.2. Object Types

### 4.3. Function Types

### 4.4. Unions and Intersections, Mixed Types

### 4.5. Aliases, Interfaces, Classes, Objects

## 5. Fundamental Types

### 5.1. Primitive Types

Ducin IT Consulting

## 10. Patterns

- 10.1. Domain Logic in frontend layer
- 10.2. Data Transfer Objects
- 10.3. Value Objects
- 10.4. Typed Templates
- 10.5. Typed Promises, Events, Streams (async ops)

## 11. Ecosystem

- 11.1. Editors/IDEs
- 11.2. Compilation
  - 11.2.1. tsconfig.json file
  - 11.2.2. Compiler Flags
  - 11.2.3. Compilation Target
- 11.3. Handling dependencies
  - 11.3.1. .d.ts files
  - 11.3.2. DefinitelyTyped, typings, npm @types
  - 11.3.3. writing custom declarations

## 12. Bundles

- 12.1. TS Namespaces
- 12.2. TS Modules
- 12.3. Automation: Webpack, Parcel

## 13. (Optional) TypeScript and legacy code

- 13.1. Project Remake Strategy: one-big-shot vs step-by-step
- 13.2. Moving logic between server & client
- 13.3. Old & new code co-existing
- 13.4. Study case

## 14. (Optional) Contract-First Design (API Contracting)

14.1. TS interfaces contracts

14.2. RAML/swagger-based contracts

14.3. JSON format, JSON Schema