Ducin IT Consulting

Warsztat Ekspercki TypeScript

Statyczne Typowanie na Frontendzie

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 prostotą, elastycznością i bezpieczeństwem typów w kodzie.

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:

- 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

Ducin IT Consulting

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

8. Type Compatibility

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

12. Anti-patterns

- 12.1. Primitive obsession
- 12.2. Boolean obsession

12.3. Leaking types

13. Optional topics

13.1. Bundling with webpack/parcel

13.2. TS tooling & ecosystem

13.3. JS-TS project migration strategies

13.4. Contract-first design (contracts: RAML, swagger, JSON Schema)

13.5. Training Prerequisites: Chosen elements of ECMAScript