Ducin IT Consulting

Zaawansowany TypeScript

Statycznie Typowany JavaScript

Ducin IT Consulting - Program szkolenia

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

Szkolenie przeznaczone zarówno dla programistów frontendowych jak i backendowych. Zakres obejmuje - w zależności od potrzeb grupy - zarówno podstawy jak i tematy bardzo zaawansowane.

Szkolenie kładzie nacisk na statyczne typowanie jako alternatywę do dynamicznego JavaScriptu. Uczestnicy poznają zalety płynące z większej kontroli nad typami danych oraz koszty, jakie niesie ze sobą stosowanie TypeScriptu w projektach o różnej skali. Poznają wzorce projektowe stosowane we frontendzie (oraz implementują niektóre z nich). Szkolenie przewiduje także zadania związane z projektowaniem aplikacji, a nie tylko samym kodowaniem.

Kluczowe Aspekty:

- Poprawne rozumienie TypeScriptu 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. Introduction

1.1. Chosen elements of ECMAScript

1.2. Compile-time & runtime

1.3. Responsibilities of TypeScript: problems solved & unsolved

2. Fundamental Types

2.1. Primitive types

2.2. The any type

2.3. Enums, String literals, Tuples

2.4. Unions, Intersections, Index types

2.5. Function types

3. (opcjonalnie) Functional Programming with TS

3.1. Functional Programming Recap

3.2. Functional Composition

3.3. Overloading Function Signatures

4. Type system

4.1. Static vs Dynamic typing

4.2. Strong vs Weak typing

4.3. Duck typing

4.4. Type inference

5. TypeScript Classes

5.1. Interfaces

5.2. Classes

5.3. Mixins

- 5.4. OOP: abstraction, polymorphism, inheritance, encapsulation

6. Ecosystem

- 6.1. Editors/IDEs
- 6.2. Compiler, compilation target
- 6.3. Handling dependencies
  - 6.3.1. .d.ts files
  - 6.3.2. DefinitelyTyped, typings, npm @types
  - 6.3.3. writing custom declarations

7. Advanced Concepts

- 7.1. TypeScript Generics
  - 7.1.1. Class Generics
  - 7.1.2. Function Generics
  - 7.1.3. Required vs Inferred Generics
- 7.2. TS Decorators
  - 7.2.1. Parameters
  - 7.2.2. Properties
  - 7.2.3. Methods
  - 7.2.4. Classes

8. Bundles

- 8.1. TS Namespaces
- 8.2. TS Modules
- 8.3. Automation: Webpack, Parcel

9. TypeScript and legacy code

- 9.1. Project Remake Strategy: one-big-shot vs step-by-step
- 9.2. Moving logic between server & client
- 9.3. Old & new code co-existing
- 9.4. Study case

Ducin IT Consulting

10. More usecases

10.1. Typed templates

10.2. Typed promises and other async operations

11. DDD elements

11.1. Domain logic in frontend layer

11.2. Data Transfer Object

11.3. Value Object

12. Contract-First Design (API Contracting)

12.1. TS interfaces contracts

12.2. RAML/swagger-based contracts

12.3. JSON format, JSON Schema

13. Backend-less development

13.1. Organizational and Business background

13.2. BLD implementations