BOTTEGA
IT minds

Program szkolenia:

Warsztaty: Angular, TypeScript, RxJS, NGRX

Informacje:

Nazwa:	**Warsztaty: Angular, TypeScript, RxJS, NGRX**
Kod:	**ang-workshop**
Kategoria:	Angular
Odbiorcy:	developerzy, architekci
Czas trwania:	3 dni
Forma:	30% wykłady / 70% warsztaty

Szkolenie jest kompleksowym warsztatem, który ma na celu doprowadzenie do świadomego wykorzystania z technologii angular/ngrx.

W trakcie szkolenia będziemy pracować nad wymaganiami jednej spójnej aplikacji oraz wykonywać kilka pobocznych ćwiczeń.

Pełny program trwa 5 dni. W zależności od potrzeb grupy możemy modyfikować zakres szkolenia tak aby dopasować do mniejszej ilości dni.

Zalety szkolenia:

- zrozumienie i spójne połączenie filozofii reduxa oraz strumieni reaktywnych
- przykłady o złożoności odpowiadającej realnym problemom
- dobre praktyki oraz rozwiązania często spotykanych problemów

BOTTEGA IT minds

Szczegółowy program:

1. TypeScript

1.1. Types

1.1.1. Type overview

1.1.2. Bottom and Top types

1.1.3. Enums, String literals, Tuples

1.1.4. Unions, Intersections, Index types

1.1.5. Function types

1.1.6. Generic Types

1.1.7. Mapped Types

1.1.8. Conditional Types

1.2. TypeScript Classes

1.2.1. Interfaces

1.2.2. Classes

1.2.3. Mixins

1.2.4. Decorators

1.2.5. OOP: abstraction, polymorphism, inheritance, encapsulation

1.3. Advanced Concepts:

1.3.1. Covariance vs Contravariance

1.3.2. Structural vs Nominal

2. Angular

2.1. Architecture

2.1.1. Overview

2.1.2. Building blocks

BOTTEGA IT minds

- 2.2. Components
  - 2.2.1. Responsibilities
  - 2.2.2. Custom components definitions
  - 2.2.3. Built-in directives
  - 2.2.4. Lifecycle hooks
  - 2.2.5. Binding types: property, event, two-way
  - 2.2.6. Cross components communication
  - 2.2.7. Inputs, outputs
  - 2.2.8. Pure Components vs Containers (aka Dumb vs Smart)
  - 2.2.9. Forms: template-driven vs reactive
- 2.3. Services
  - 2.3.1. Responsibilities
  - 2.3.2. Built in services
  - 2.3.3. Custom services definition
  - 2.3.4. Stateless vs stateful services
- 2.4. Modules
  - 2.4.1. In depth definition: declarations, exports, imports, providers
  - 2.4.2. Designing custom Modules
  - 2.4.3. "Components" modules
  - 2.4.4. "Services" modules
  - 2.4.5. "Mixed" modules
- 2.5. Dependency Injection
  - 2.5.1. Injectors
  - 2.5.2. Providers
  - 2.5.3. Hierarchical injectors

BOTTEGA IT minds

- 2.6. Pipes
  - 2.6.1. Built-in
  - 2.6.2. Custom pipes
- 2.7. Routing and Navigation
  - 2.7.1. Router config
  - 2.7.2. Router navigation: declarative and imperative
  - 2.7.3. Router guards
  - 2.7.4. Modules loading: lazy vs eager
- 2.8. Unit Testing
  - 2.8.1. Components
  - 2.8.2. Services
  - 2.8.3. Pipes

3. RxJS

- 3.1. Reactive programming paradigm
- 3.2. Overview: Observable streams, Operators, Observers, Subscriptions, Subjects
- 3.3. In depth
  - 3.3.1. Observable streams
  - 3.3.2. stream Operators
  - 3.3.3. stream Observers
  - 3.3.4. stream life cycle
  - 3.3.5. stream Subscriptions
  - 3.3.6. data flow
- 3.4. Defining custom RxJS building blocks (exercises)
  - 3.4.1. streams
  - 3.4.2. operators

BOTTEGA IT minds

- 3.4.3. subscription
- 3.4.4. observers
- 3.5. How to work with RxJS built-ins (exercises)
- 3.5.1. stream factory functions
- 3.5.2. built in operators
- 3.6. Combining multiple streams
- 3.6.1. in depth comparison of built-in operators and factory functions
- 3.6.2. real world examples and exercises
- 3.7. Higher Order Observables (HOO)
- 3.7.1. definition
- 3.7.2. how to flatten HOO
- 3.7.3. how to use them effectively
- 3.7.4. real world examples and exercises
- 3.8. Subjects
- 3.8.1. unicasted vs multicasted Observables
- 3.8.2. managing Subject state and behaviour
- 3.8.3. built-in Subjects comparison

4. ngrx, Redux

- 4.1. When should I use it?
- 4.2. ngrx/Redux architecture overview
- 4.3. In depth analysis
- 4.3.1. Store
- 4.3.2. State
- 4.3.3. Selectors
- 4.3.4. Actions, Action Types

BO·TT·EGA
IT minds