Ducin IT Consulting

Nowoczesne Aplikacje oparte o React/Redux

Ducin IT Consulting - Program szkolenia

Czas trwania: **3 dni**
Formuła: **25% wykłady, 75% ćwiczenia**

Szkolenie przeznaczone dla osób mających przynajmniej podstawową wiedzę o JavaScripcie. Poświęcone jest zastosowaniu programowania funkcyjnego do tworzenia nowoczesnych aplikacji opartych o biblioteki React oraz Redux. Kładzie nacisk na zrozumienie fundamentów paradygmatu funkcyjnego oraz filozofii obu narzędzi – a potem zastosowanie ich w praktyce do stworzenia własnej aplikacji z czytelnym i krótkim kodem.

W trakcie szkolenia porównujemy React/Redux z innymi bibliotekami i frameworkami, analizując mocne i słabe strony różnych rozwiązań. Uczestnicy mają za zadanie nie tylko poprawnie zakodować określone funkcjonalności aplikacji, ale także zaproponować wiele rozwiązań i – wskutek analizy – wybrać optymalne w danej sytuacji. Poruszane zagadnienia związane z designem i architekturą to m.in. granularność komponentów, zakres ich odpowiedzialności, rola statycznego typowania oraz separowanie komponentów od zmian stanu.

Podczas szkolenia kładziemy duży nacisk zarówno na zrozumienie istoty omawianych zagadnień, kodowanie własnych rozwiązań, jak i pracę w grupie.

Kluczowe Aspekty:

- architektura komponentowa i folozofia aplikacji opartych o React/Redux
- zastosowanie nowoczesnego JavaScriptu i potrzeba jego użycia
- nowoczesny toolstack z wykorzystaniem Redux-toolkit
- dobre praktyki, częste błędy

Ducin IT Consulting

Program szkolenia:

1. Components

1.1. Component as a function

1.2. State, Props, Callback Props

1.3. One-way data flow

1.4. JSX, Virtual DOM (vs Shadow DOM)

1.5. Hooks, Rules of hooks, Custom Hooks

1.6. React's Reactivity Model

2. Component Architecture

2.1. Component Composition

2.2. Stateful & Stateless Components

2.3. Controlled components vs Uncontrolled Components

2.4. State management: private vs shared

2.5. Passing callbacks down

2.6. Declarative components, IoC

3. Redux

3.1. Building blocks: state, store, actions, reducers

3.2. Inversion of Control

3.3. Async Flow

3.4. Middlewares

3.5. Redux-Toolkit, slices

4. Performance

4.1. reasons to rerender

4.2. memoization (useCallback, useMemo, memo)

- 4.3. props.children
- 4.4. context-related performance
- 4.5. redux-related performance

5. Usage with TypeScript

- 5.1. Typing state & props
- 5.2. Typing hooks
- 5.3. Typing React Context
- 5.4. Typing Redux: state, reducers, actions, slices

6. Testing

- 6.1. jest, RTL, assertions, mocks
- 6.2. component unit testing
- 6.3. multi-component integration testing
- 6.4. testing contexts & providers interaction
- 6.5. snapshot testing
- 6.6. non-fragile tests

7. Infrastructure

- 7.1. Storybook, Component Libraries
- 7.2. Monorepos
- 7.3. create-react-app (js, ts)
- 7.4. webpack

8. (Optional) Functional Programming recap

- 8.1. FP vs OOP
- 8.2. Immutability
- 8.3. Pure Functions, Side Effects
- 8.4. Higher Order Functions
- 8.5. Reducers