Diving into functional programming with Elm

...

Pragmatic web functional development

By Adrian Magdas

Co-Founder & Lead Developer @ Crafting Software Innovation
adrian@craftingsoftware.com

Agenda

1. Why Functional Programming
2. Short introduction to Elm language
3. Elm Application Architecture (TEA)
4. Elm system and how to manage side-effects
5. Take aways

Why Functional Programming

- **Handling concurrency in imperative languages with shared mutable state is hard to get right, a lot of well known issues: race conditions, deadlocks, state management**
- **FP avoids the problems associated with shared mutable state by forcing immutability and separation of data and data transformations**
- **Usual properties for a functional language**: functions as first class citizens, referential transparency, immutability, persistent data structures, pattern matching, pure functions, data separated from transformations

Short introduction to Elm

Let’s see some syntax and language data types to be able to understand the examples

CODE

Elm Application Architecture

- provides a simple way to structure any web application

- Triplet: Init, Update, View

- clear separation between state and state changing operations

- influence for well known libraries like Redux

Elm System

- provides the runtime for Elm apps

- acts as a side effect manager so there is a clear barrier between JS and Elm

- ports

- subscriptions

- Tasks

Take-aways

Functional programming: functions as first class citizens, referential transparency, immutability, persistent data structures, pattern matching, pure functions, data separated from transformations

State management: managing state is much more simple when we have a clear separation between state and operations on state

Elm: offers a pragmatic way to manage application state

There are no silver bullets :)

Thank you

Develop with passion!

Resources