Dive into web functional programming with Elm

...

Pragmatic web functional development

Adrian Magdas

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

Agenda

1. Why Functional Programming
2. Introduction to Elm language
3. Elm runtime system
4. Elm Application Architecture (TEA)
5. Take-aways

WE ARE CRAFTING SOFTWARE

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

WE ARE CRAFTING SOFTWARE

Why Elm?

- No runtime errors in practice. No null. No undefined is not a function.
- Friendly error messages that help you add features more quickly.
- Well-architected code that stays well-architected as your app grows.
- Automatically enforced semantic versioning for all Elm packages.

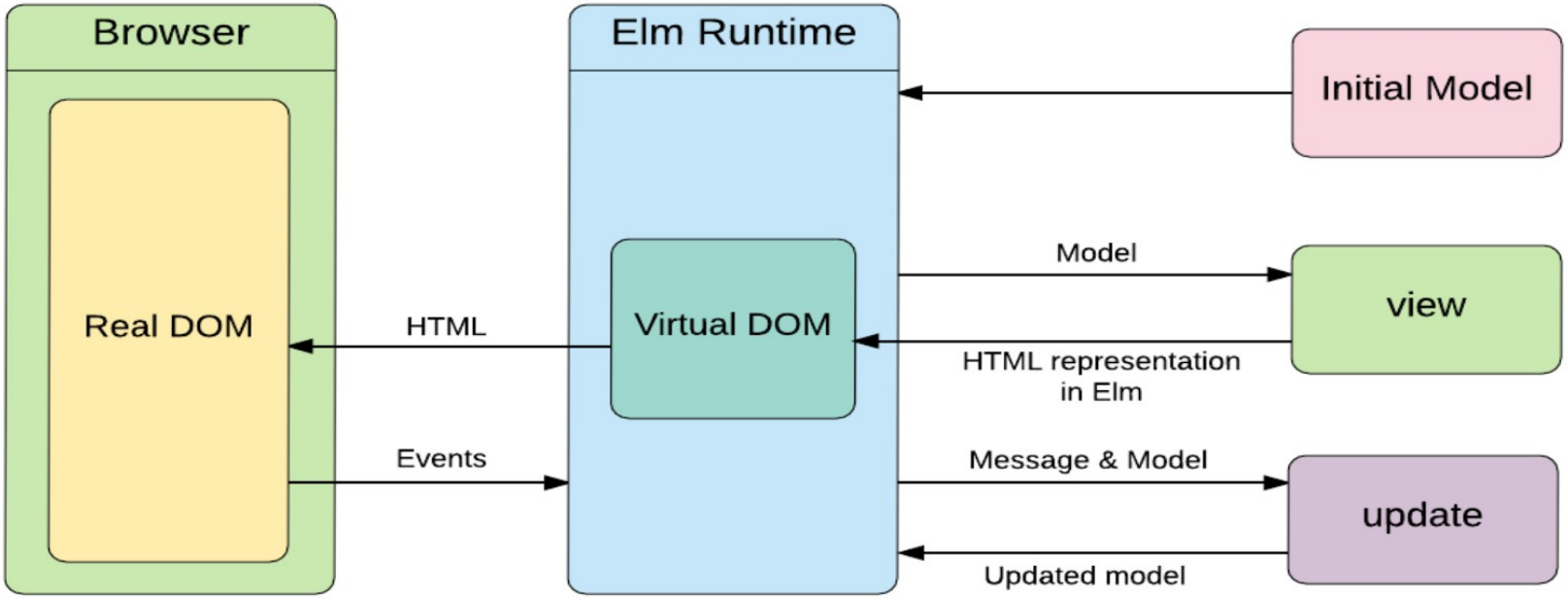
WE ARE CRAFTING SOFTWARE

Short introduction to Elm

- Values
- Functions
- Lists
- Tuples
- Records
- Types

WE ARE CRAFTING SOFTWARE

Elm System


Browser
Real DOM
HTML
Events
Elm Runtime
Virtual DOM
Initial Model
Model
view
HTML representation
in Elm
Message & Model
update
Updated model

Elm system

- provides the runtime for Elm apps
- acts as a side effect manager so there is a clear barrier between JS and Elm, any communication with JS has to be done via ports
- ports
- Subscriptions
- Tasks

WE ARE CRAFTING SOFTWARE

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

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 but some bullets are better than others :)

Thank you

Develop with passion!

Resources

- https://dev.to/rtfeldman/tour-of-an-open-source-elm-spa
- http://rundis.github.io/blog/2017/elm_bootstrap_launch.html
- https://discourse.elm-lang.org
- https://guide.elm-lang.org/
- http://package.elm-lang.org/packages/elm-lang/core/latest

WE ARE
CRAFTING SOFTWARE