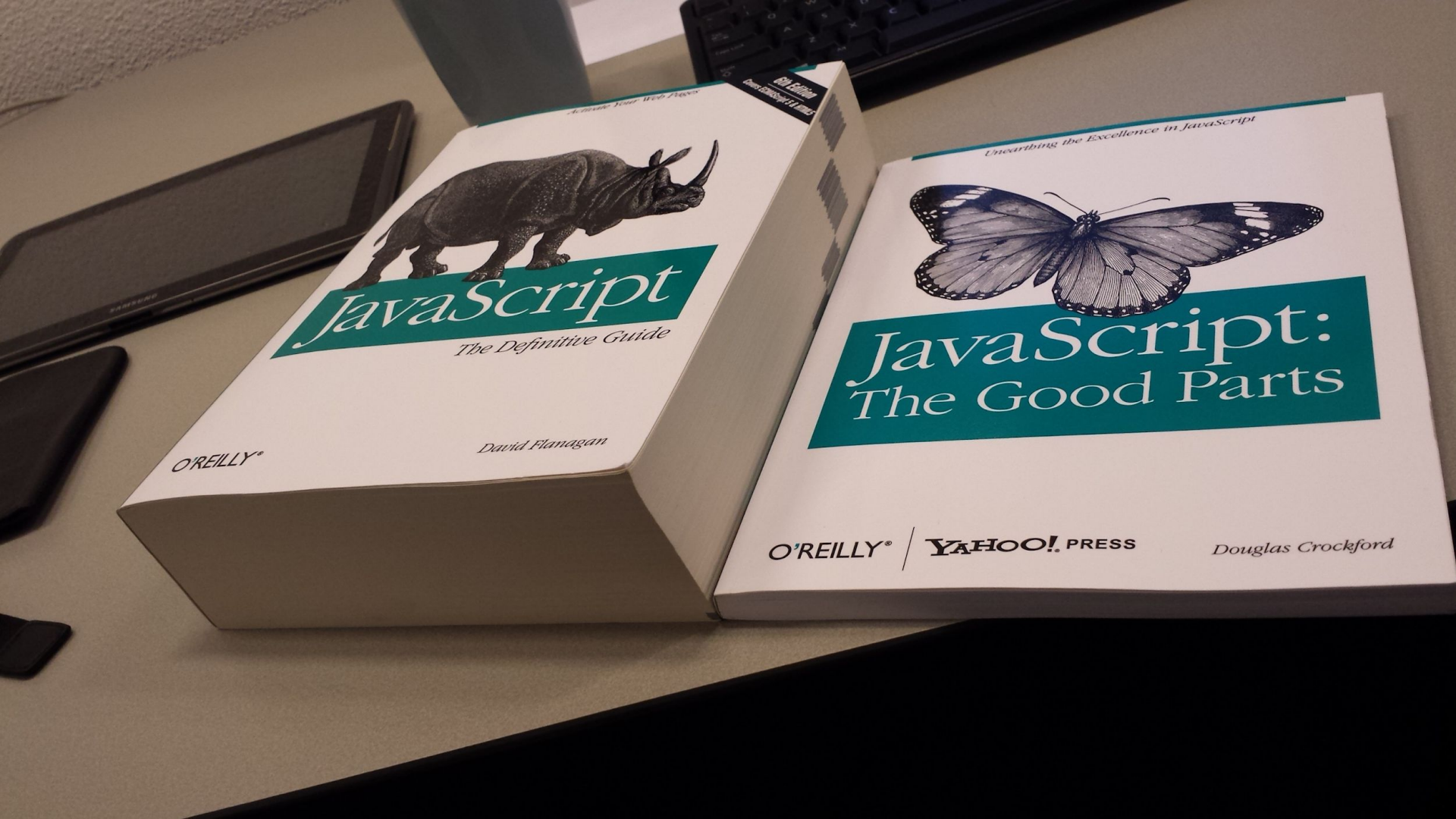
WPROWADZENIE DO ELM

funkcyjny frontend

Jakub Stańczak
Patryk Scheffler


Activate Your Web Pages
JavaScript
The Definitive Guide
David Flanagan
O'REILLY
Unearthing the Excellence in JavaScript
JavaScript:
The Good Parts
O'REILLY
YAHOO! PRESS
Douglas Crockford

ELM

- Rok powstania - 2012
- Autor - Evan Czaplicki
- Czysty język funkcyjny, bazowany na Haskellu

Forget what you have heard about functional programming. Fancy words, weird ideas, bad tooling. Barf.

- Evan Czaplicki

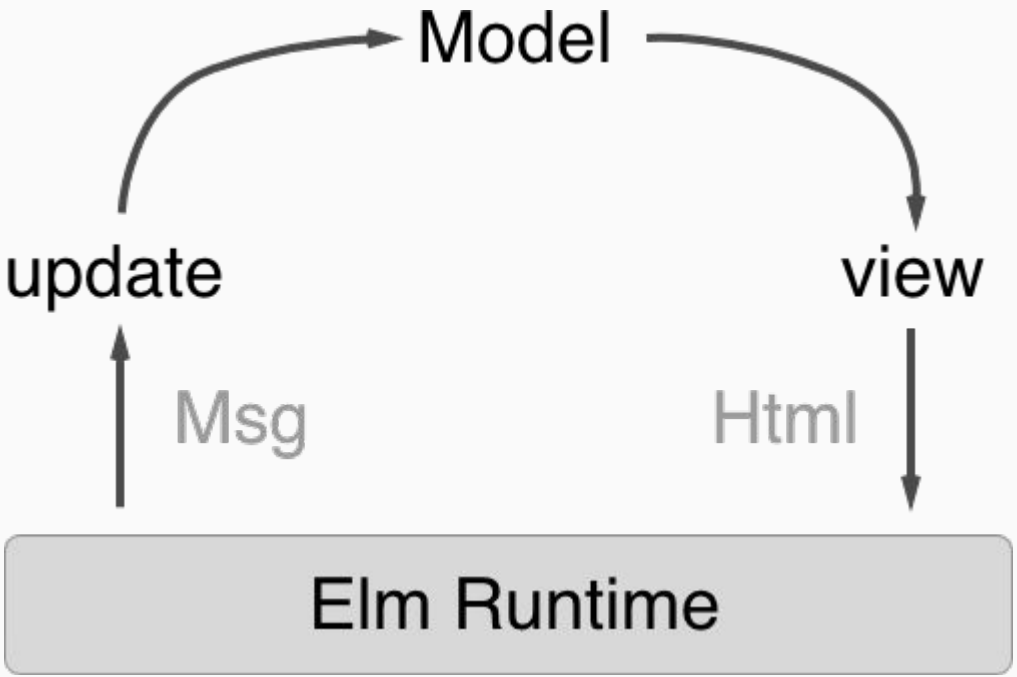
GŁÓWNE ZALETY

No Runtime Exceptions	Great Performance
Enforced Semantic Versioning	Small Assets

CO ELM ZAWIERA

- Pure function (Purescript.js)
- Immutable data structures (Immutable.js)
- Currying (Ramda.js)
- Static type checking (Flow)
- No concept of Null (Folktale)
- The Elm Architecture (Redux)
- Declarative UI (React.js)
- Virtual DOM (React.js)

ELM ARCHITECTURE



JAK ZACZĄĆ?

JĘZYKI FUNKCYJNE

ERLANG

elixir

Haskell

OCaml

elm

Scala

Clojure

JĘZYKI FUNKCYJNE

- Pierwszy język funkcyjny: LISP ~1958

- Funkcje jako model programowania
- Rachunek lambda
- Dopasowanie wzorca
- Brak efektów ubocznych
- Nadawanie typów
- Rekursja
- Leniwa ewaluacja
- Funkcje wyższego rzędu

WSZYSTKO JEST FUNKCJĄ!

Funkcyjne operacje na kolekcjach

- map - [1,4,9] = map(x -> x^2, [1,2,3])
- filter - [1,3] = filter(x -> x!=2, [1,2,3])
- reduce - 6 = reduce(acc, x -> acc + x, [1,2,3])
- pattern matching -

  case someStringInfo of

  “pattern1” -> print “WoW”

  _ -> print “Unrecognized”

Monady

A monad is just a monoid in the category of endofunctors, what's the problem?

```
\a -> (f a) >>= \b -> (g b)
```

https://stackoverflow.com/questions/3870088/a-monad-is-just-a-monoid-in-the-category-of-endofunctors-whats-the-problem

Monady

- Wymagania:
  - Kontekst
  - Sposób łączenia

Oznaczenia

- 2 + 3 == (+) 2 3
- (+) 3 == (Int -> Int)
- f : (Int -> Int)
- g : (a -> a)

Kontekst

type Maybe = Nothing | Just a

gdzie “a” oznacza dowolny typ zmiennej

Nothing - wiadomość oznaczająca brak wartości zmiennej

Just - “wiadomość / pojemnik na zmienną z wartością”

Functors

(a -> b) -> f a -> f b

dla Maybe

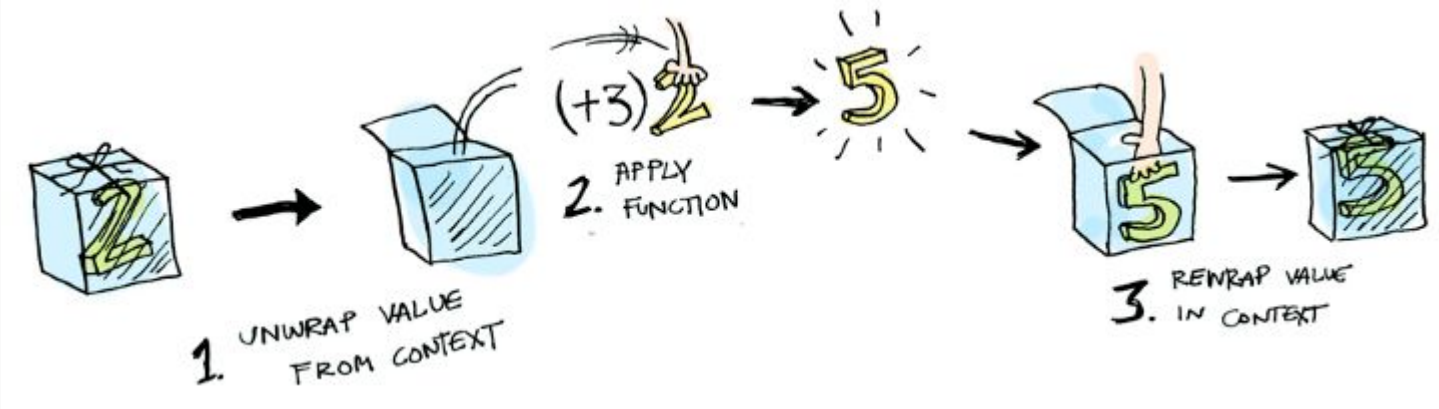
(a -> b) -> Maybe a -> Maybe b

gdzie a, b są to typy zmiennych jak Int, String

Maybe functor

```
map func maybe =
    case maybe of
        Just value -> Just( func value )
        Nothing -> Nothing
```

Generalne działanie



https://cdn-images-1.medium.com/max/720/0*b6bZ8NXtuk3O2QO8.png

Monada

(a -> f b) -> f a -> f b

dla Maybe

(a -> Maybe b) -> Maybe a -> Maybe b

gdzie a, b są to typy zmiennych jak Int, String

Maybe monad

```
andThen func maybe =
    case maybe of
        Just value -> func value
        Nothing -> Nothing
```

Maybe

```
> String.toFloat
<function> : String -> Maybe Float

> String.toFloat "3.1415"
Just 3.1415 : Maybe Float

> String.toFloat "abc"
Nothing : Maybe Float
```

Jak to wygląda w elm

Operacje w elm

- 1 + 2
- 1 < 2
- “Hello” ++ “World!”
- [“Hello”] ++ [“World”]
- type Boolean = YES | NO
- type alias Position =

```
{ x : Int,
  y : Int }
```

Funkcje

- Definicja

  greetFunction : String -> String
  greetFunction name =
  	“Hello ” ++ name

- Wywołanie

  > greetFunction “World!”
  Hello World!

Pattern matching

```
answer : String -> Maybe Int
answer question =
    case question of
        "universe?" ->
            Just 42

        _ ->
            Nothing
```

Pipe operator

```
fingerprint : String -> List Char
fingerprint word =
    word
    |> String.toLower
    |> String.toList
    |> List.sort

fingerprint_ : String -> List Char
fingerprint_ word =
    List.sort
    <| String.toList
    <| String.toLower word
```

Compiler

```
-- NAMING ERROR ------------------------------------------------------ Main.elm

Cannot find variable `List.nap`.

4| foo = List.nap ((+) 1) things
          ^^^^^^^^
`List` does not expose `nap`. Maybe you want one of the following?

    List.map
    List.any
    List.map2
    List.map3

Detected errors in 1 module.
```

```
The definition of `greet` does not match its type annotation.

11| greet : String -> Int
12| greet name =
13|>    hello ++ ", " ++ name ++ "!"

The type annotation for `greet` says it always returns:

    Int

But the returned value (shown above) is a:

    String

Detected errors in 1 module.
```

Live demo

Bibliografia

- https://en.wikipedia.org/wiki/Functional_programming
- https://www.tutorialspoint.com/functional_programming/functional_programming_introduction.htm
- https://elm-lang.org/
- https://en.wikipedia.org/wiki/Elm_(programming_language)
- https://guide.elm-lang.org/
- https://github.com/tastejs/todomvc/tree/gh-pages/examples/elm
- https://medium.com/@l.mugnaini/functors-applicatives-and-monads-in-pictures-784c2b5786f7