

# Elm | > Real Life

[jiri.sliva@newired.com](mailto:jiri.sliva@newired.com)



# In the beginning ... (2016)

## Needs

- Pure JavaScript
- Complex UI
- REST
- Electron, NW.js, WebExtension ...
- a lot of Prototyping
- a lot of Refactoring
- GWT
- Polymer
- ClojureScript
- AngularJS / Angular 2.0 RC
- React, Redux, Immutable-js
- Flow, TypeScript
- **Elm**

# Elm - Pure Functional

- Immutable data, only pure functions, ...
- ML family syntax (*Haskel for kids*)
- Static Typing

```
view : List Article -> Html Msg
view articles =
  let
    viewArticle article =
      div [ class "article" ] [ text article.title ]
  in
  Html.ul
    [ class "article-list" ]
    (List.map viewArticle articles)
```

# Types

```
String, Int, Float, Bool, List, (a, b, c)
```

```
type alias PersonId = Int
```

```
type alias Person =  
  { name : String  
    , age : Int  
  }
```

```
type Role  
  = Admin  
  | Sales  
  | Customer
```

```
type Msg  
  = ButtonClicked  
  | NameChanged String  
  | SetRole Person Role
```

```
isJunior : Person -> Bool  
isJunior person =  
  person.age < 20
```

```
isJunior : { a | age: Int } -> Bool  
isJunior x =  
  x.age < 20
```

# Pattern Matching

```
case msg of
  ButtonClicked ->
    ..
  NameChanged name ->
    ..
  SetRole person role ->
    ..
```

```
case aList of
  [] ->
    "matches the empty list"
  [x]->
    "matches a list of exactly one item, " ++ toString x
  x::xs ->
    "matches a list of at least one item whose head is " ++ toString x
```

# Pattern Matching .. more

```
type Page
  = User UserSection
  | Invoices
```

```
type UserSection
  = Profile
  | History
```

```
access : Role -> Page -> String
```

```
access role page =
  case ( role, page ) of
    ( Sales, User _ ) ->
      ..
    ( Sales, Invoices ) ->
      ..
    ( Customer, User Profile ) ->
      ..
    ( Customer, User History ) ->
      ..
    ( Admin, _ ) ->
      ..
```

-- MISSING PATTERNS ----- src/Main.elm

This `case` does not have branches for all possibilities:

```
41|>   case ( role, page ) of
42|>     ( Sales, User _ ) ->
43|>       ..
44|>
45|>     ( Sales, Invoices ) ->
46|>       ..
47|>
48|>     ( Customer, User Profile ) ->
49|>       ..
50|>
51|>     ( Customer, User History ) ->
52|>       ..
53|>
54|>     ( Admin, _ ) ->
55|>       ..
```

Missing possibilities include:

```
( Customer, Invoices )
```

# null? .. Maybe not

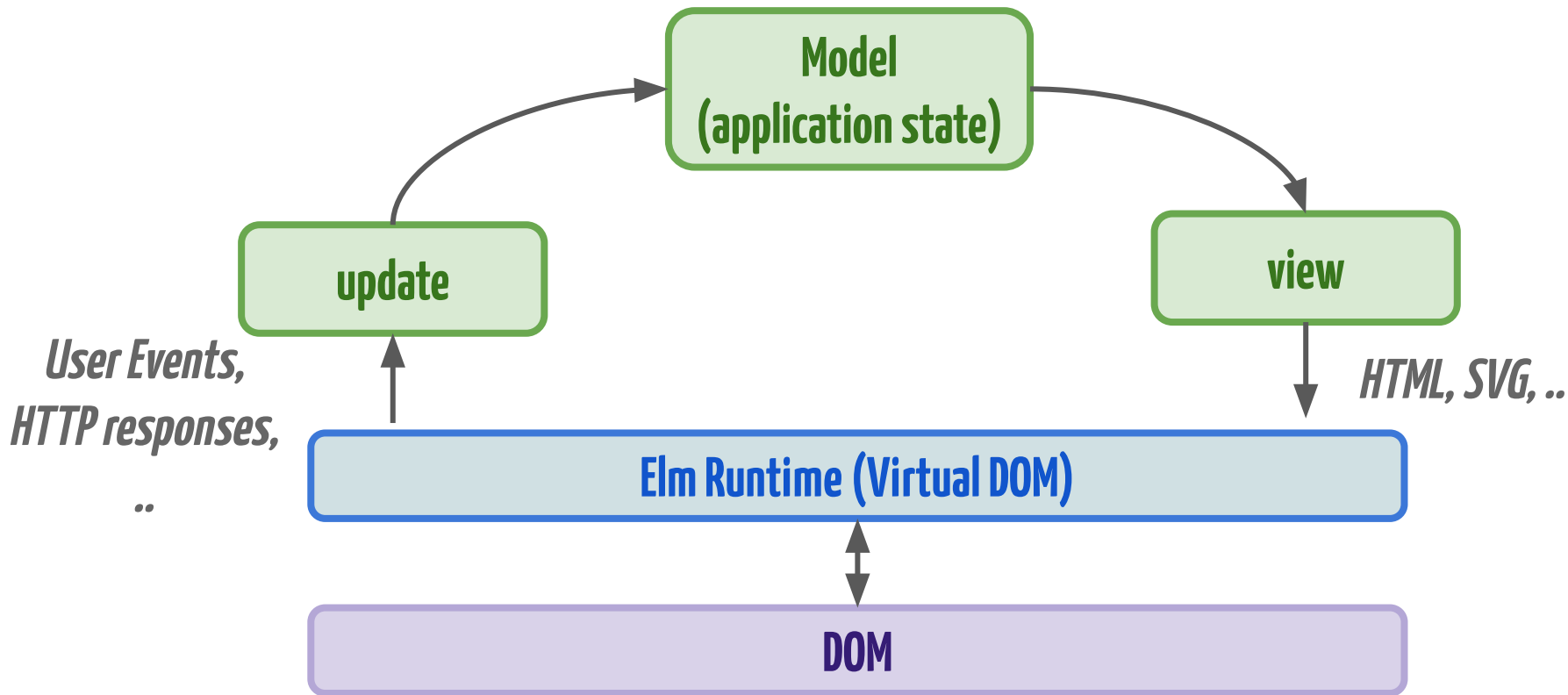
```
type Maybe a
  = Just a
  | Nothing
```

```
type alias Person =
  { name : String
  , age : Int
  , email: Maybe String
  }
```

```
personEmail person =
  case person.email of
    Just email ->
      email
    Nothing ->
      "no e-mail address"
```

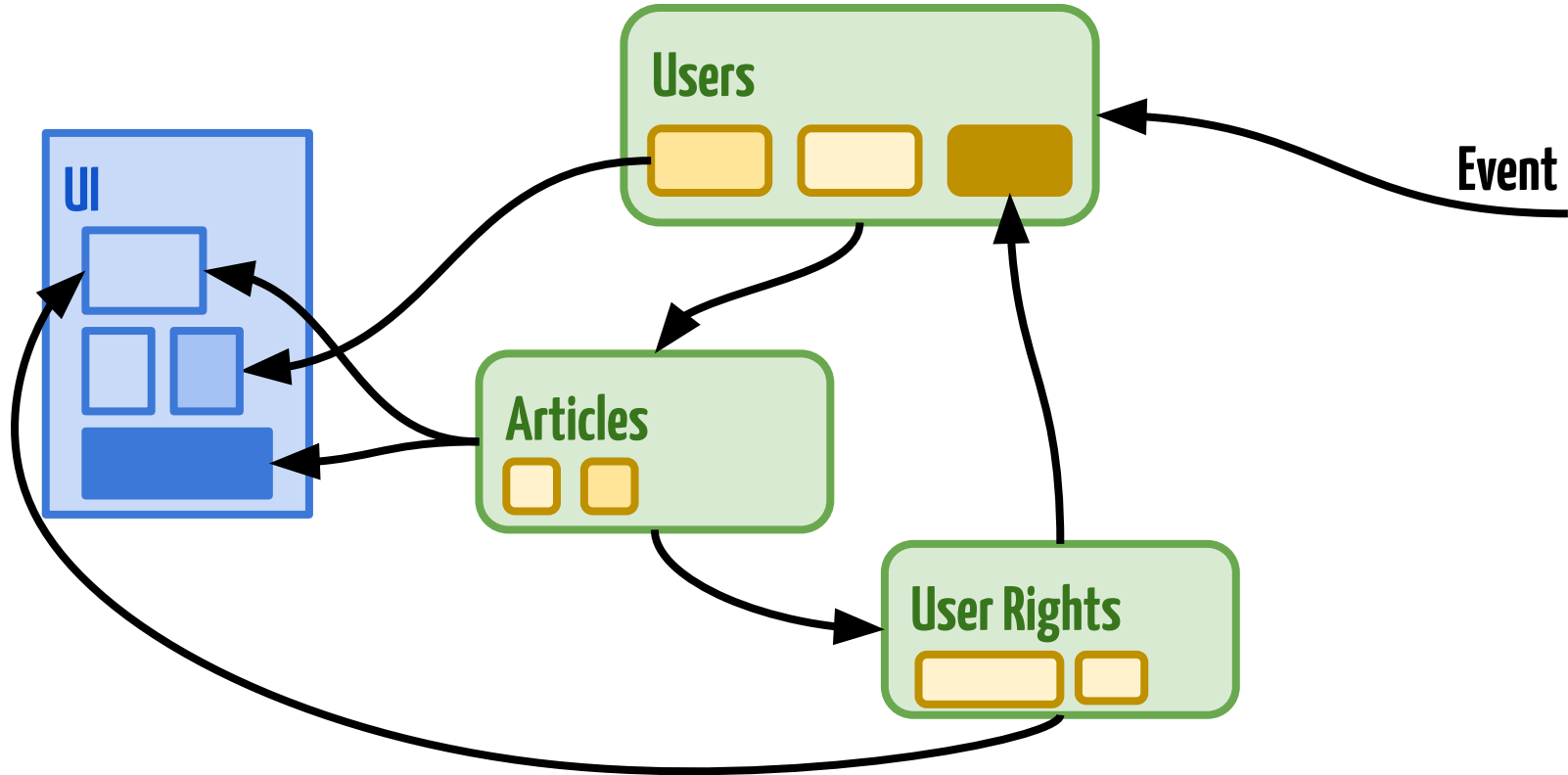
```
personEmail person =
  person.email |> Maybe.withDefault "no e-mail address"
```

# Model – Update – View

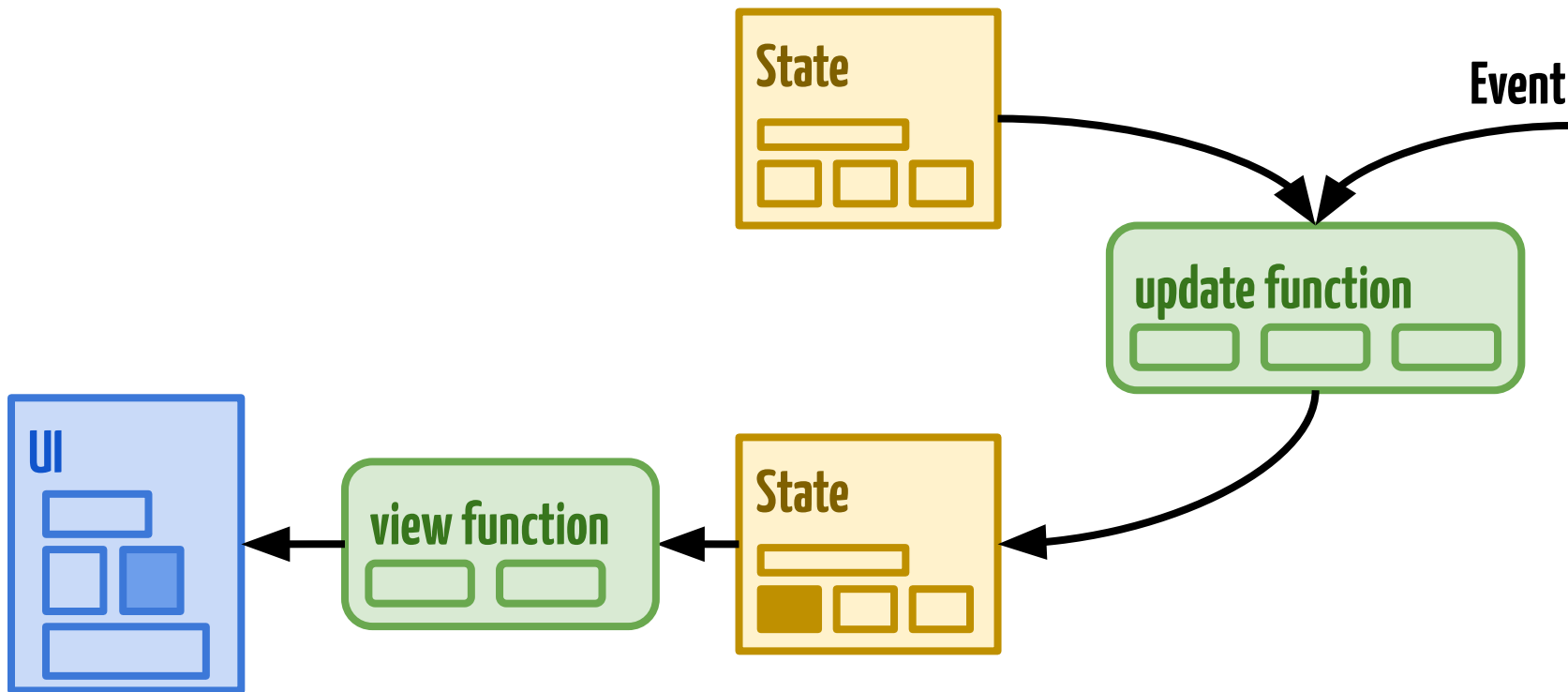




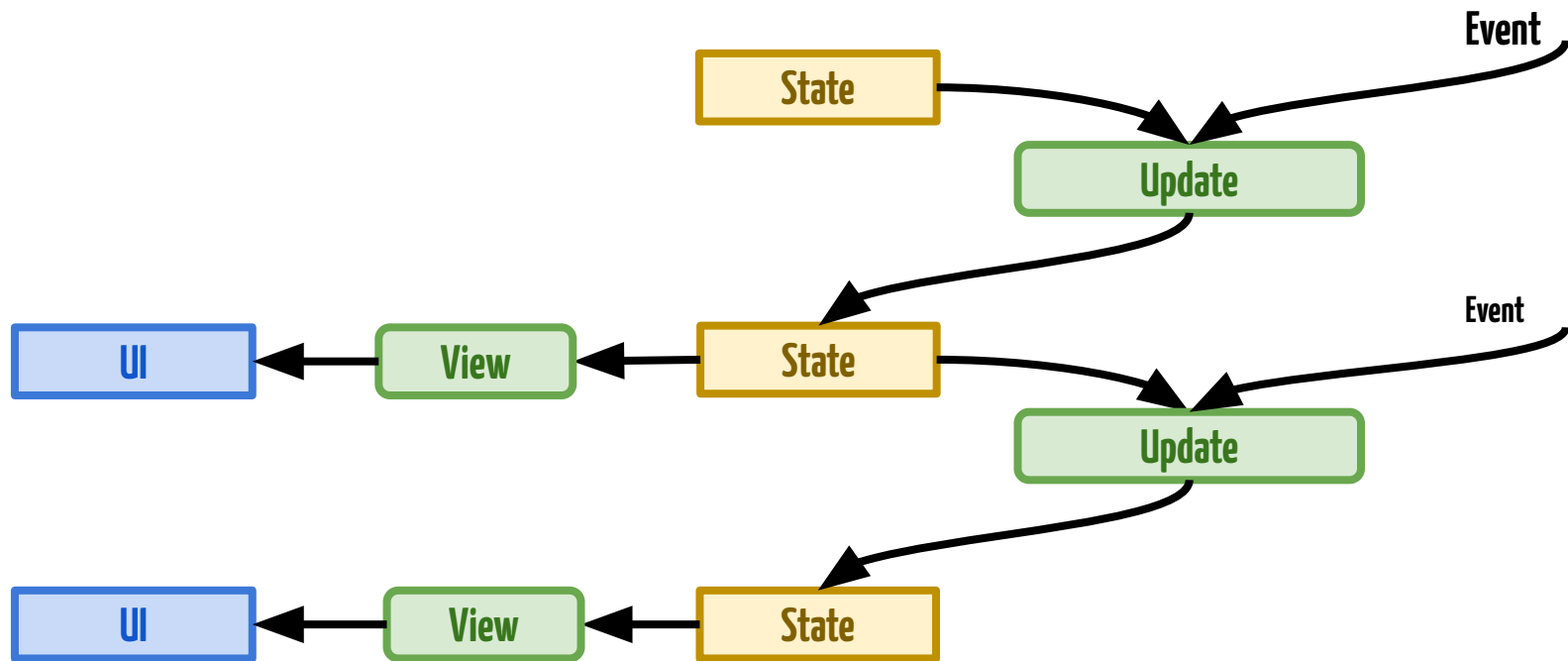
# Inside *common application*™



# Inside Elm Application



# Elm Application Timeline



# Demo!

## Proč Elm?

- Rychlé první kroky
- Minimalistická syntaxe
- Přátelský kompilátor
- Bezpečný refactoring
  - Single source of truth
  - Immutability
  - Static Typing
- “Něco jako React + Redux + TypeScript + Immutable-js ...” v jednom nástroji
- No runtime errors

# Compiler

## Friendly

**Detected** errors in 1 module.

-- TYPE MISMATCH ----- src/Main.elm

The `model` record does not have a `nane` field:

```
43|      { model | nane = "John" }  
                      ^^^^^
```

This is usually a typo. Here are the `model` fields that are most similar:

```
{ name : String  
  , greetings : String  
}
```

So maybe `nane` should be `name`?

## Fast

114 files,  
29 173 lines => **0.79** sec.


# Tooling

- **Elm-Format** – Accepted by community as *the only one proper format* (used in 99% packages)
- **Elm Analyse** – Static analyze tool. Interactive web application or JSON output
- **Elm debugger** – Just compile with `--debug`
- Elm-monitor - Monitor your Elm program with redux-devtools (experimental)
- [ellie-app.com](http://ellie-app.com)
- IDEs support
  - **Intellij IDEA** – Refactoring, Find usages, Code completion, Type Inference and Type Checking
  - **Visual Studio Code**
  - **Atom**
  - **Vim**

# Ecosystem

[package.elm-lang.org](http://package.elm-lang.org)

- [vega-lite](#)
- [elm-graphql](#)
- [Elm Bootstrap](#) - Elm & Bootstrap 4
- [elm-mdl](#) - Material Design Library

elm  
packages

[elm/browser](#) ... 1.0.1 — Overview  
Run Elm in browsers, with access to browser history for single-page apps (SPAs)

[elm/bytes](#) ... 1.0.8 — Overview  
Work with sequences of bytes (a.k.a. ArrayBuffer, typed arrays, DataView)

[elm/core](#) ... 1.0.2 — Overview  
Elm's standard libraries

[elm/file](#) ... 1.0.5 — Overview  
Select files. Download files. Work with file content.

### Popular Packages

- [elm/core](#)
- [elm/html](#)
- [elm/json](#)
- [elm/browser](#)
- [elm/url](#)
- [elm/http](#)

### Resources

- [Fancy Search](#)
- [Using Packages](#)
- [API Design Guidelines](#)
- [Write great docs](#)
- [Elm Website](#)



## Kdo to používá?

- Microsoft – [Microsoft/elm-json-tree-view](#)
  - IBM – [Decision Composer](#)
  - Ableton – [learningmusic.ableton.com](#)
  - Gizra
  - NoRedInk
  - Newired ;-)
- Chceš programovat v Elmu?  
Stále hledáme nové kolegy do týmu.

To be continued ...

## Appendix 1: Elm & JavaScript Interoperability

# Interoperability with JavaScript – Ports (1)

## Elm

```
port setProfile : String -> Cmd msg

update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        SetProfile ->
            ( model, setProfile "pythagoras" )
        ..
```

## JavaScript

```
var app = Elm.Main.init({
    node: document.getElementById('elm')
});

app.ports.setProfile.subscribe(function(id) {
    console.log("Elm is asking for saving Profile with id: " + id)
    localStorage.setItem("profile", id);
});
```

# Interoperability with JavaScript – Ports (2)

Elm

```
port onLastAccess : (String -> msg) -> Sub msg

subscriptions : Model -> Sub Msg
subscriptions _ =
    onLastAccess OnLastAccess

type Msg
    = OnLastAccess String

update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        OnLastAccess lastAccess ->
            ( { model | lastAccess = lastAccess }, Cmd.none )
        ..
```

JavaScript

```
app.ports.onLastAccess.send(lastAccess);
```

# WebComponents – Dispatch Custom Events

```
class CustomEditor extends HTMLElement {  
  constructor() {  
    const shadow = this.attachShadow({ mode: 'open' })  
    const input = document.createElement("input")  
    shadow.appendChild(input)  
  
    const _this = this  
    input.addEventListener("input", function () {  
      _this.dispatchEvent(new CustomEvent("editorChanged", { detail: input.value }));  
    })  
  }  
}  
  
customElements.define('custom-editor', CustomEditor)
```

```
<custom-editor></custom-editor>
```

# WebComponents – Handle Custom Events

```
node "custom-editor"  
  [ property "editorValue" (Encode.string model.greetings)  
    , on "editorChanged" editorChangedDecoder  
  ]  
  []
```

```
editorChangedDecoder : Decode.Decoder Msg  
editorChangedDecoder =  
  Decode.map EditorChanged <|  
    Decode.at [ "detail" ] <|  
      Decode.string
```

```
input.addEventListener("input", function () {  
  _this.dispatchEvent(new CustomEvent("editorChanged", { detail: input.value }));  
})
```

