Fun with functional programming using Elm

Learn the paradigm of functional programming in a fun way by creating web apps in a pure functional language - Elm

Today

Why learn this way?

Why choose functional programming?

What is functional programming?

Introduction to the Elm language

crisp.

About me

40 years as a developer

Startup during 2019 using Elm

crisp.

Per Lundholm

Why learn this way?

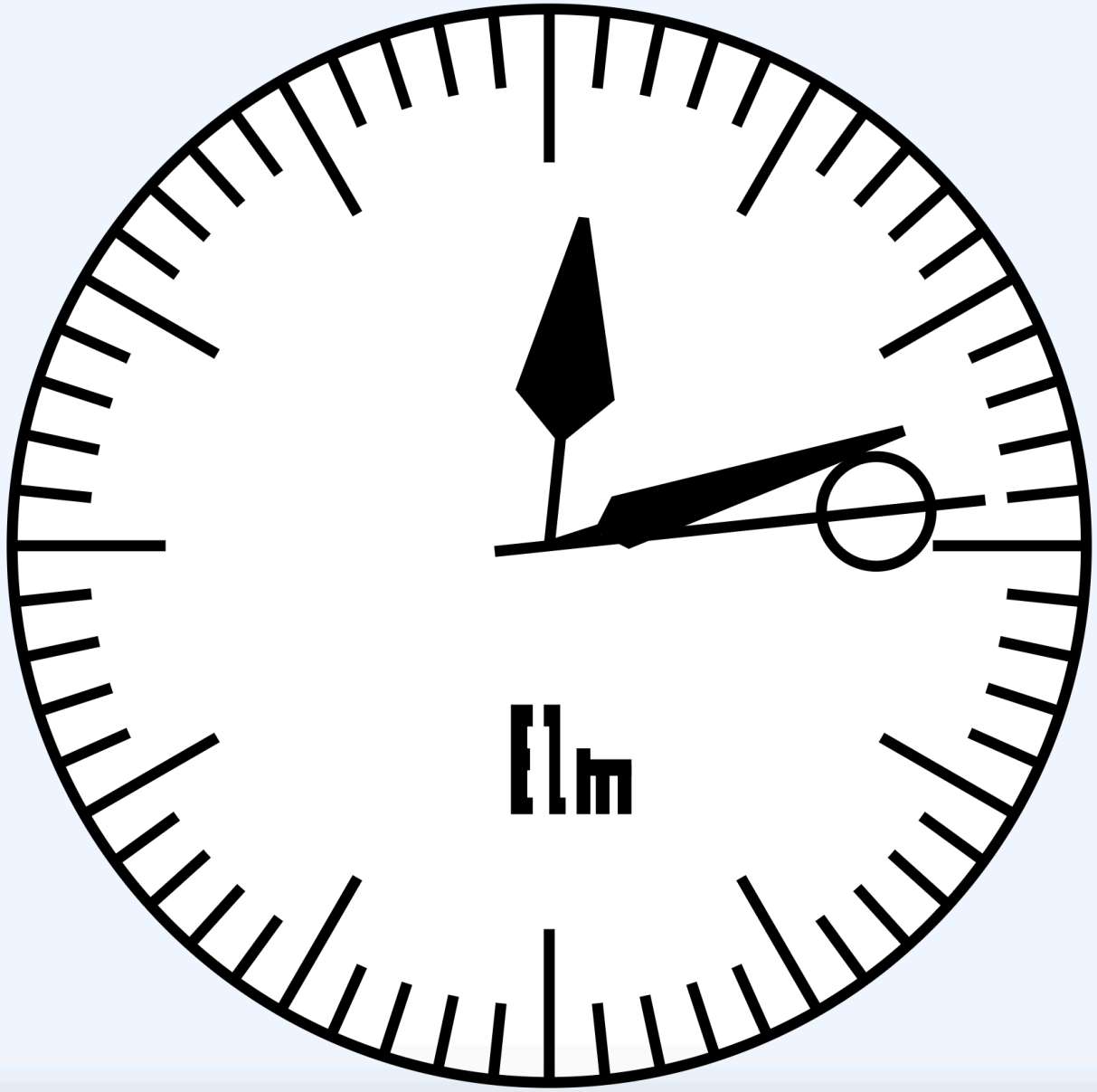
Learning should be fun

Interacting with things on screen is fun

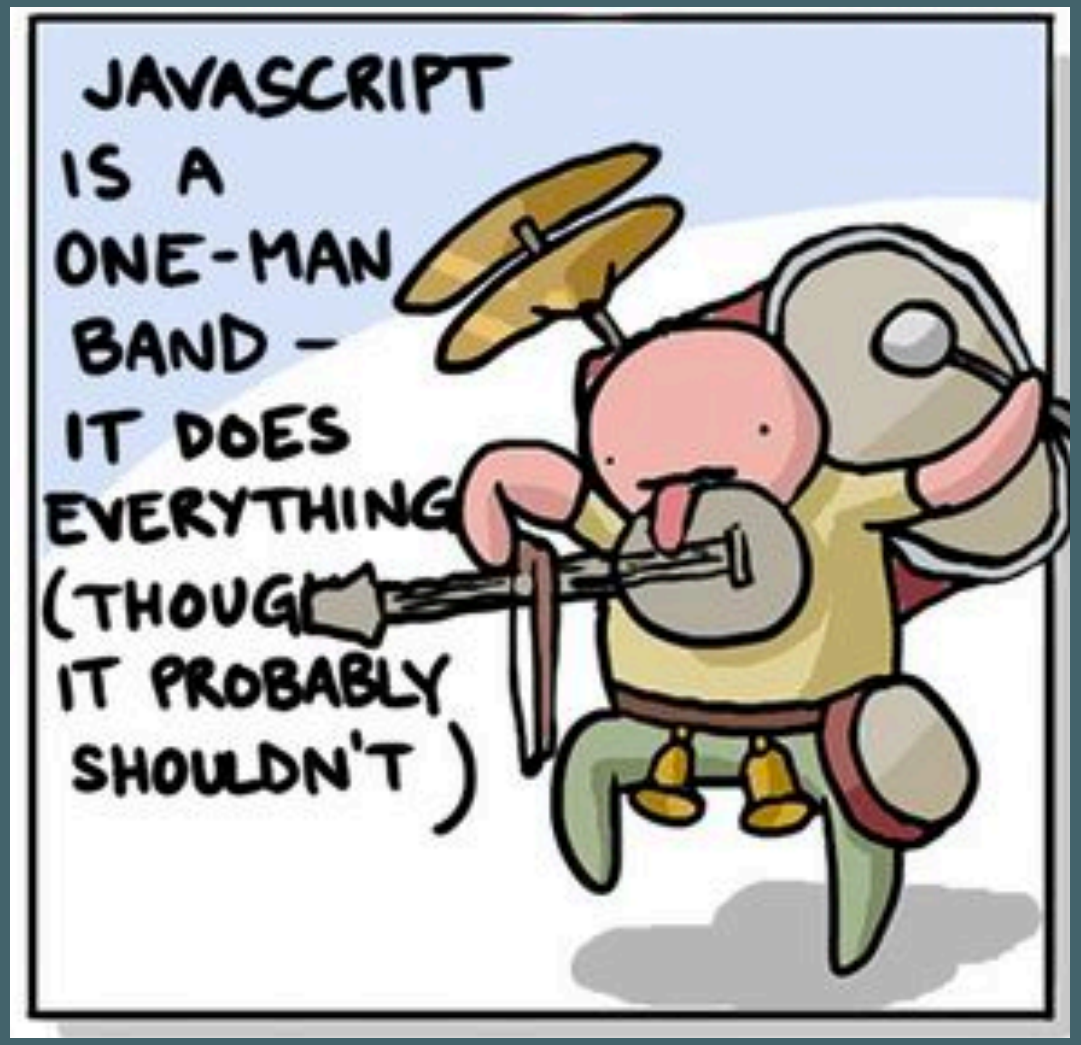
Friendly and helpful compiler is fun

Use a pure language to get it right

Avoid multi-paradigm



Programming languages


JAVASCRIPT
IS A
ONE-MAN
BAND -
IT DOES
EVERYTHING
(THOUGH
IT PROBABLY
SHOULDN'T)


HASKELL IS JAZZ -
COMPLEX, BUT INCREDIBLY
POWERFUL AND FLEXIBLE -
(IF YOU KNOW
HOW TO USE
IT.)

What is functional programming?

Function is the main abstraction

Immutable state

Declarative

Aspect	Functional	Object oriented
Main abstraction	Function	Object
Language class	Declarative	Imperative
State	Immutable	Mutable

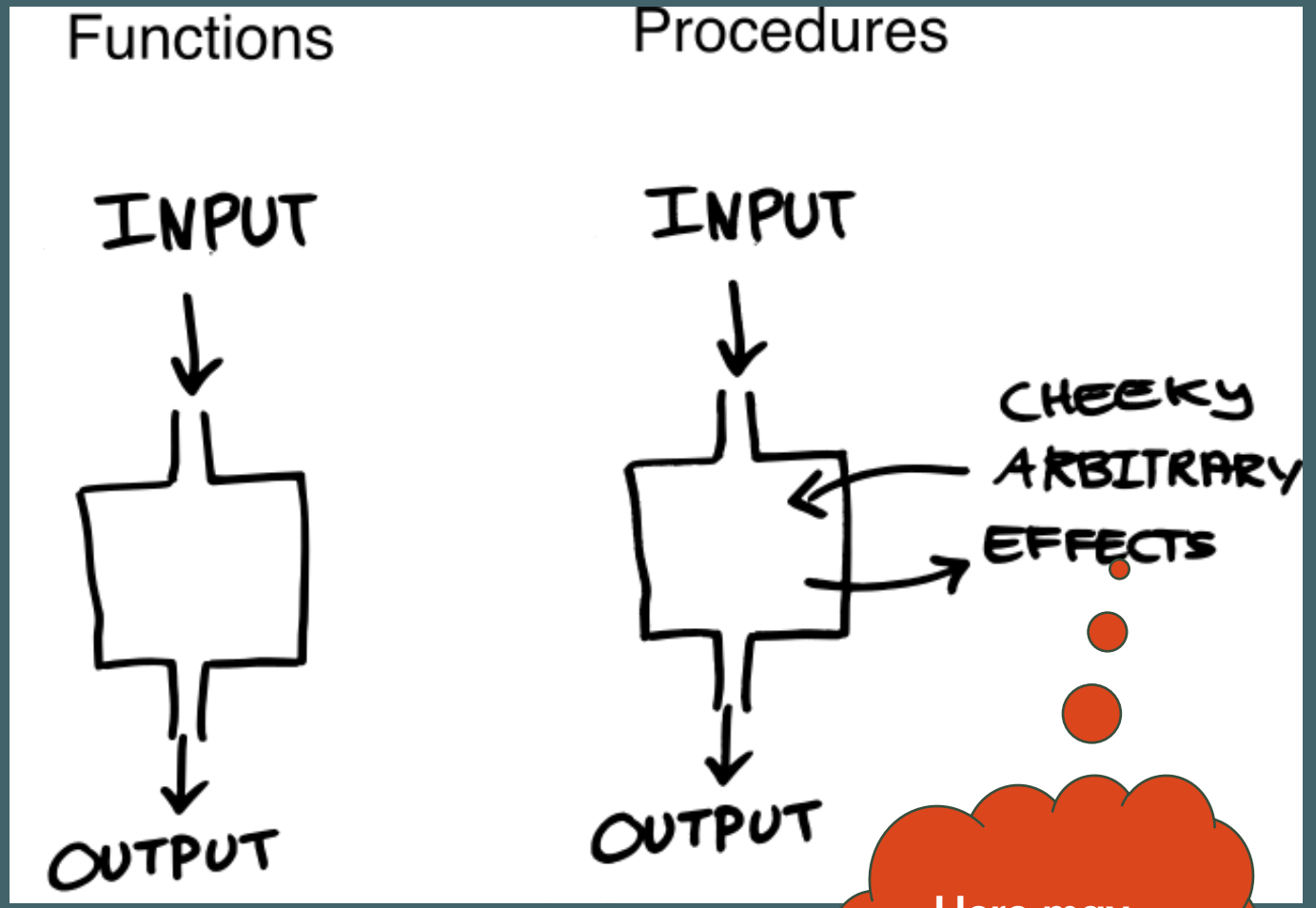
Why choose functional programming?

Easy to understand code

No need to consider rest of system

Cache result

Parallel execution of independent functions



```
print(String s)
random()
dateTime()
```

What is a *pure* function?

1. No side effects
2. Always the same result, given same arguments

```
add(Integer x, Integer y)
toHtml(Model m)
```

But wait... no side effects?

Anything useful has side effects, instead let's talk about:

Actions

Depends on when they are called. E.g. `print()`.

Calculations

Convert input to output (pure). E.g. `sum()` .

Data

Facts `"{"playerId":"dfedb43e-7194-4348-a615-9e09744b1b51"}"`

Functional thinking

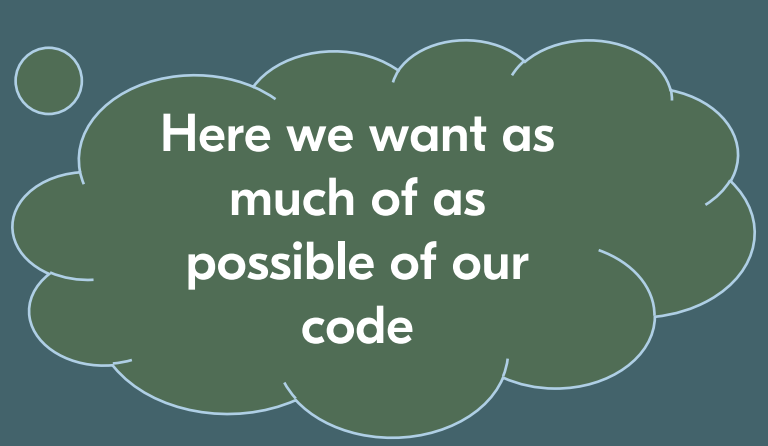
How does that help?

Actions

depends on number of calls, timing and input

Calculations

depends on input only



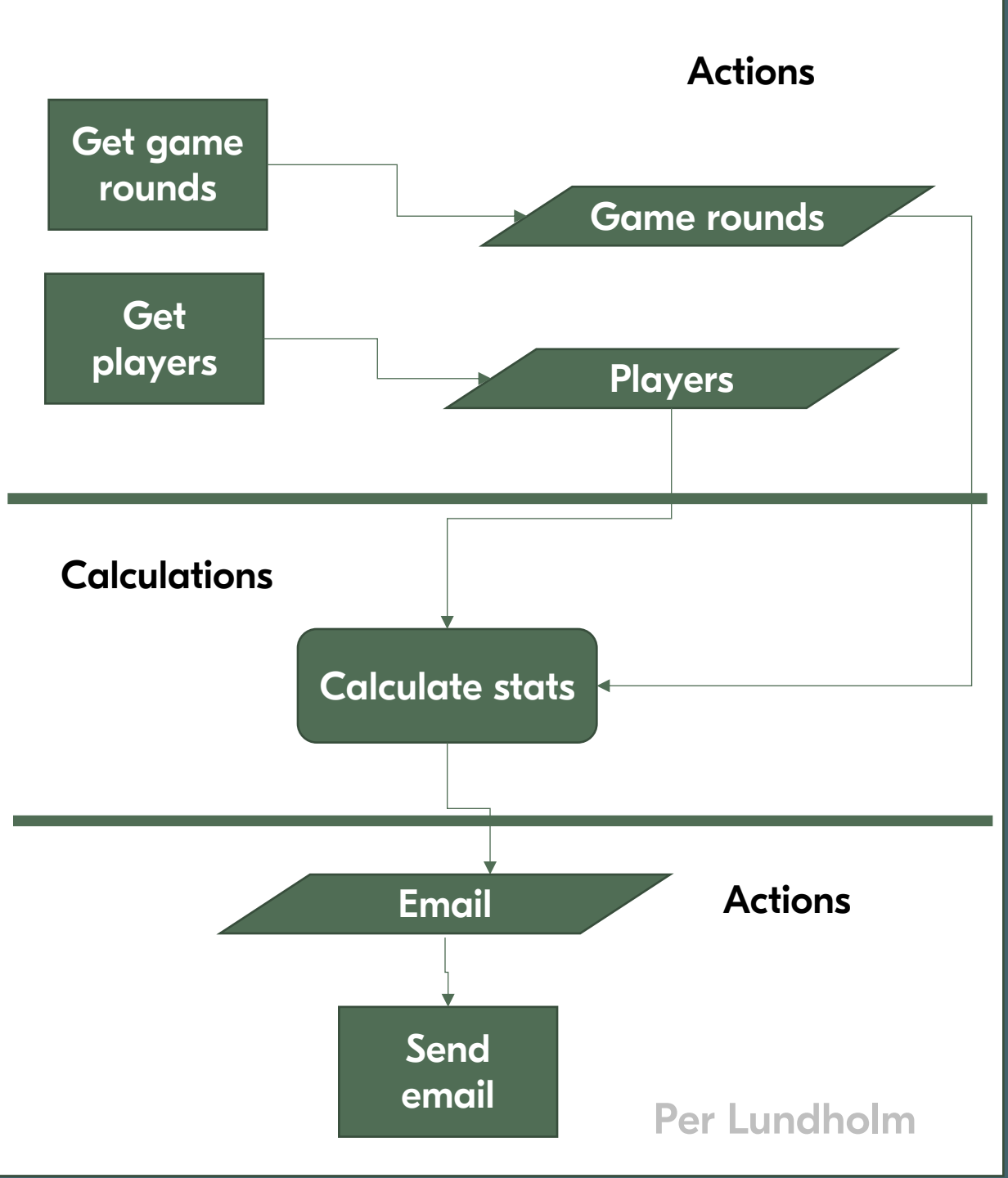
Example:
Send players their stats

Getting data are **actions**

You want to minimize logic here

Calculating stats is the workhorse

It depends only on input — easier to test



Introduction to the Elm language

Pure functional language

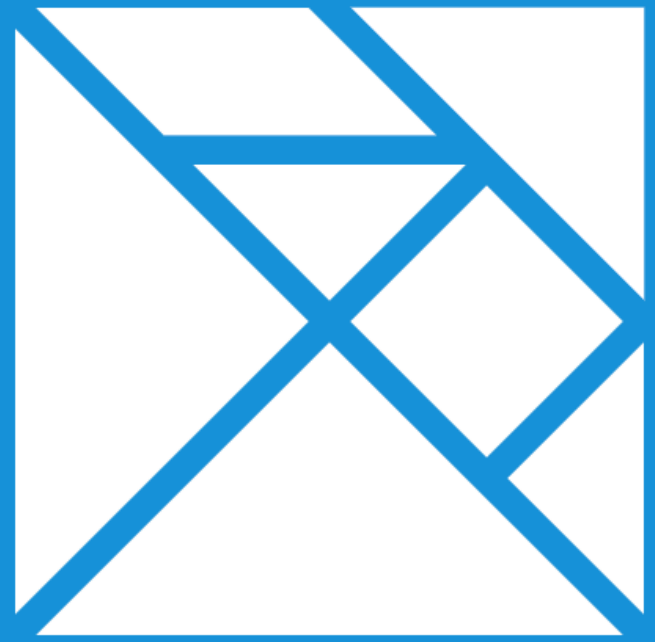
Strong *but* nice type system

Dedicated to the web

Compiles to JavaScript

crisp.

elm



Per Lundholm

Elm's selling points

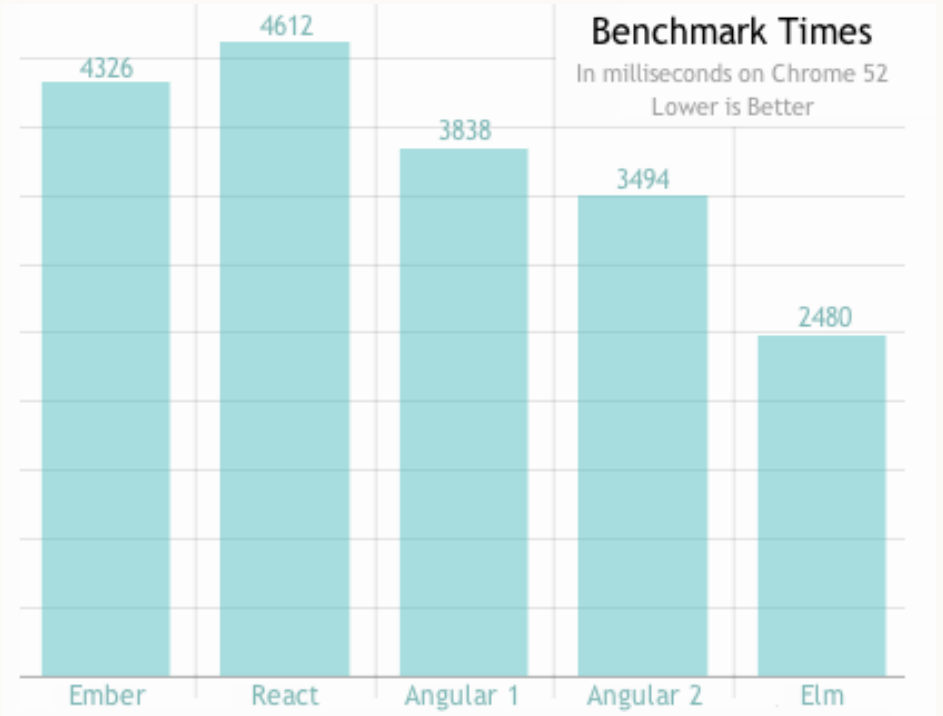
No Runtime Exceptions

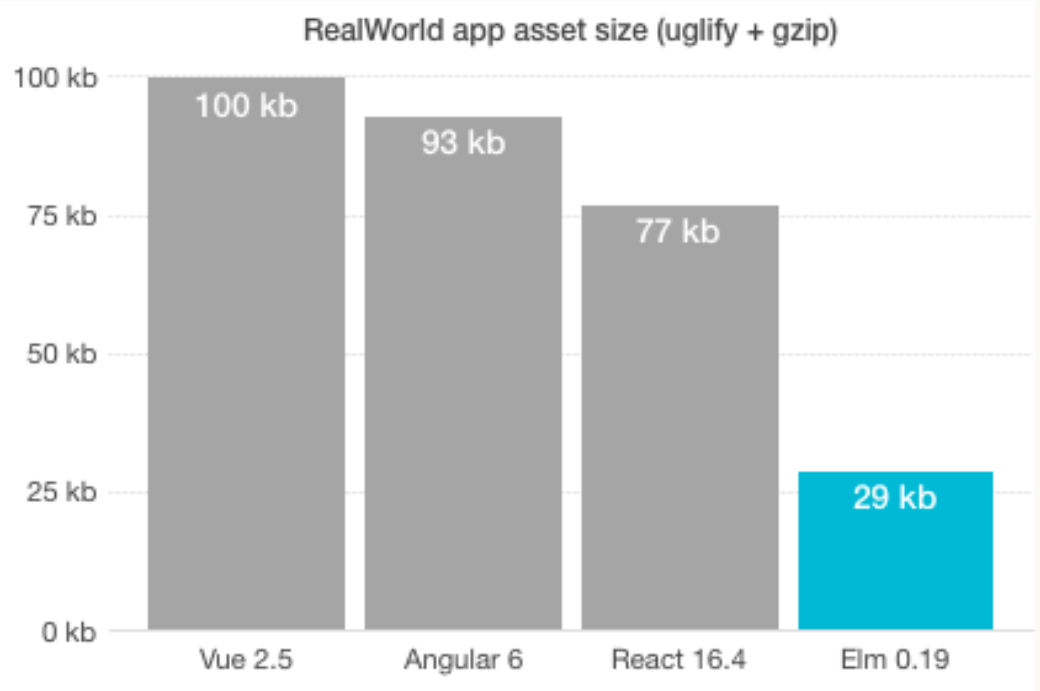
Great Performance

Enforced Semantic Versioning

Small Assets

JavaScript Interop





crisp.

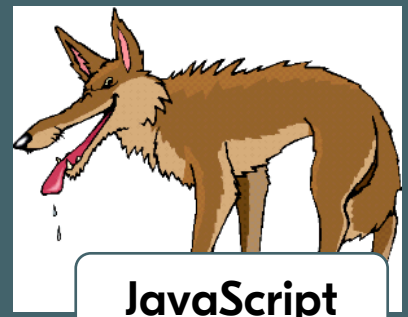
The nice clean world of Elm

REST

JSON Decode/Encode

Safe

Port



JavaScript

Per Lundholm

Elm has guard rails

Types!
No null value
An if has an else
Modules for isolation
A switch handles every case
Friendly and helpful compiler

Friendly compiler

```
-- TYPE MISMATCH  /Users/perty/IdeaProjects/elm-currency-adder/src/CompilerError.elm

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

36|              { model | fistName = string }
                         ^^^^^^^^

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

    { firstName : String
    , lastName : String
    }

So maybe fistName should be firstName?
```

What does Elm look like?

A bit like Haskell

A function has an optional type declaration

The definition is an expression

```
update : Msg -> Model -> Model
update msg model =
    case msg of
        LeftValue string ->
            { model | leftValue = string }

        RightValue string ->
            { model | rightValue = string }


view : Model -> Html Msg
view model =
    div []
        [ h1 []
            [ text "Add money of different currencies"
            ]
        , div []
            [ input [ onInput LeftValue ] []
            , text " + "
            , input [ onInput RightValue ] []
            ]
        ]
```

Elm types

Records

Custom types

Phantom types

Records

```
type alias TimeOfDay =
    { hour : Int
    , minute : Int
    , second : Int
    }


type alias Model =
    { time : TimeOfDay
    , timeZone : Time.Zone
    , clockWidth : String
    }
```

Elm types

Records

Custom types

Phantom types

Custom

```
type User
    = Regular String Int
    | Visitor String


r =
    Regular "TheMightyMouse" 95


v =
    Visitor "Guest1337"
```

```elm
type Eur
    = Eur


type Sek
    = Sek
```

Phantom

Elm types

Records

Custom types

Phantom types

```elm
-- Demonstrate phantom types by the problem of mixing currencies.
-- The `c` is not used by any of the type's data
-- constructors so `Currency` is called a "phantom" type.


type Currency c
    = Currency Int



-- Note that values like Currency 500 don't tell you what kind of currency
-- we're dealing with.
-- The only way to know is by adding a type signature.


price : Currency Eur
price =
    Currency 500



-- Can only add two currencies of the same kind


add : Currency a -> Currency a -> Currency a
add (Currency c1) (Currency c2) =
    Currency (c1 + c2)
```

crisp.

The Elm Architecture

Model = application state

View = function of the current state

Update = function of current state and a Message

```elm
module Counter exposing (main)

import ...


main =
    Browser.sandbox
        { init = 0
        , update = update
        , view = view
        }


type alias Model =
    Int


type Msg
    = Increment
    | Decrement


view : Model -> Html Msg
view model =
    div []
        [ button [ onClick Decrement ] [ text "-" ]
        , div [] [ text (String.fromInt model) ]
        , button [ onClick Increment ] [ text "+" ]
        ]


update : Msg -> Model -> Model
update msg model =
    case msg of
        Increment ->
            model + 1

        Decrement ->
            model - 1
```

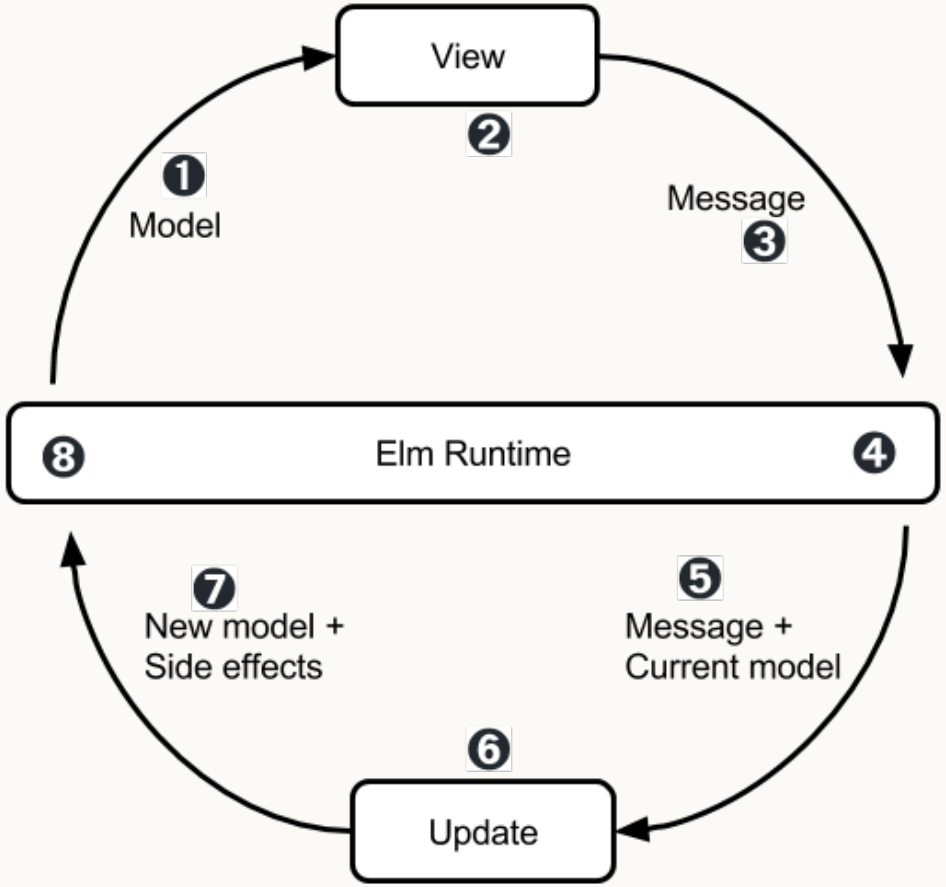
The Elm Architecture

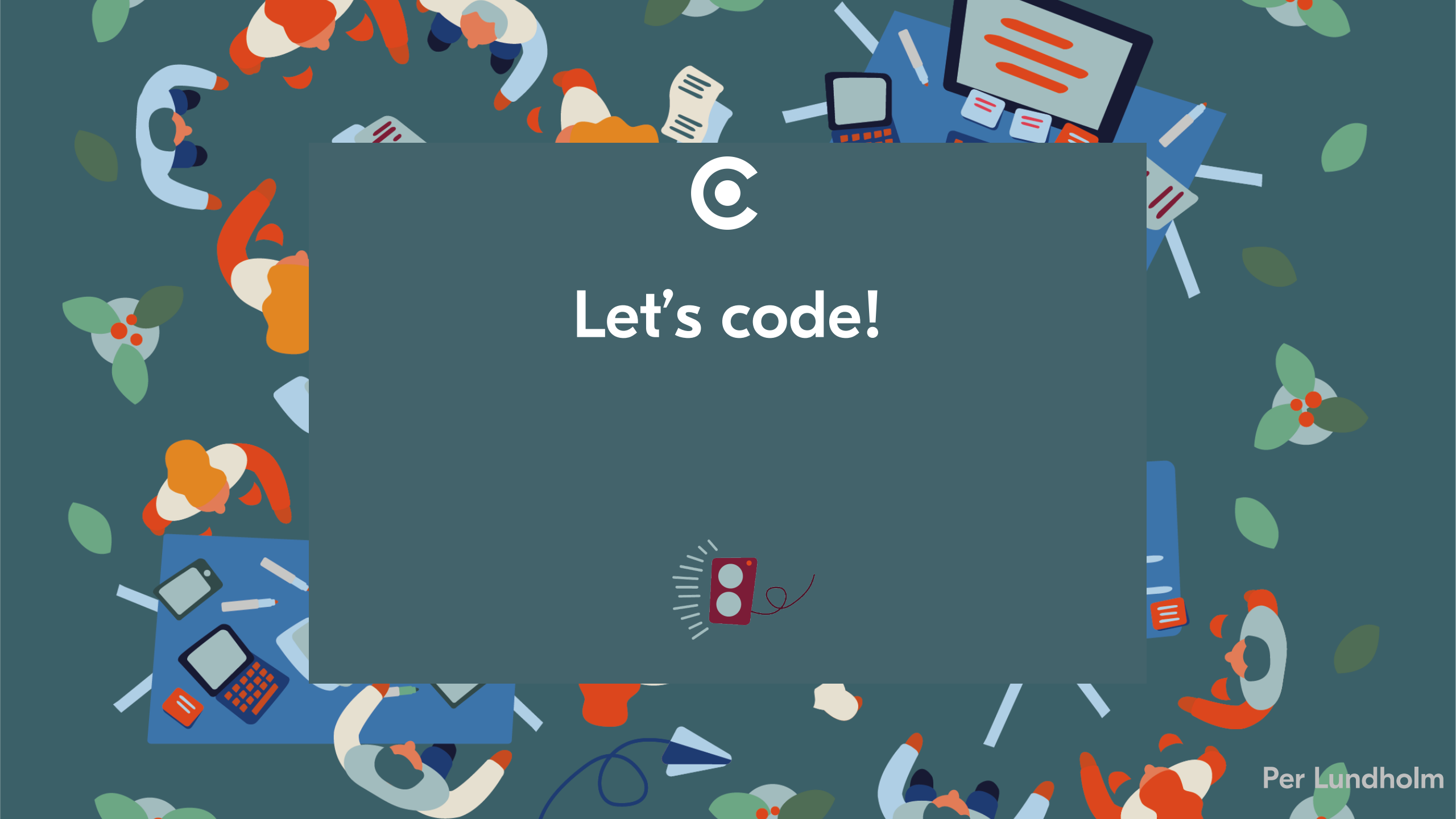
The view takes the model and creates HTML

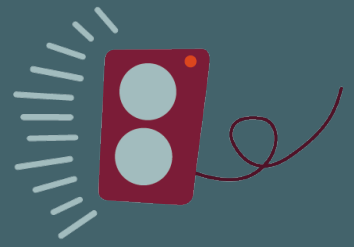
User activity emits a Message

Runtime pass Message and Model to Update

Update creates a new model and initiates side effects (e.g. REST call)



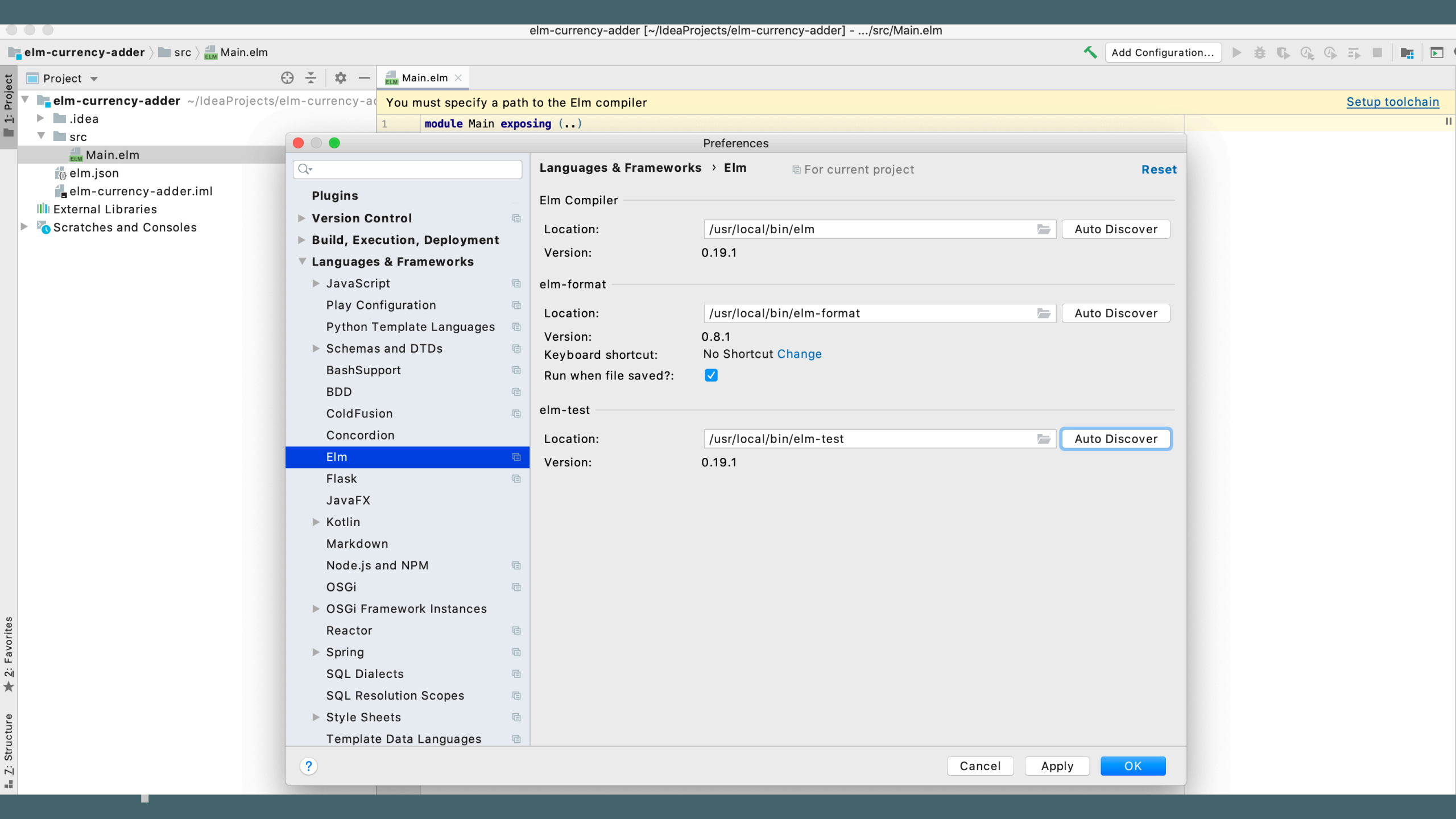

Let’s code!
Per Lundholm



Use elm-init

```
mkdir my-elm-project
cd my-elm-project
elm init
```

The directory gets a starting `elm.json` and an empty `src` folder.

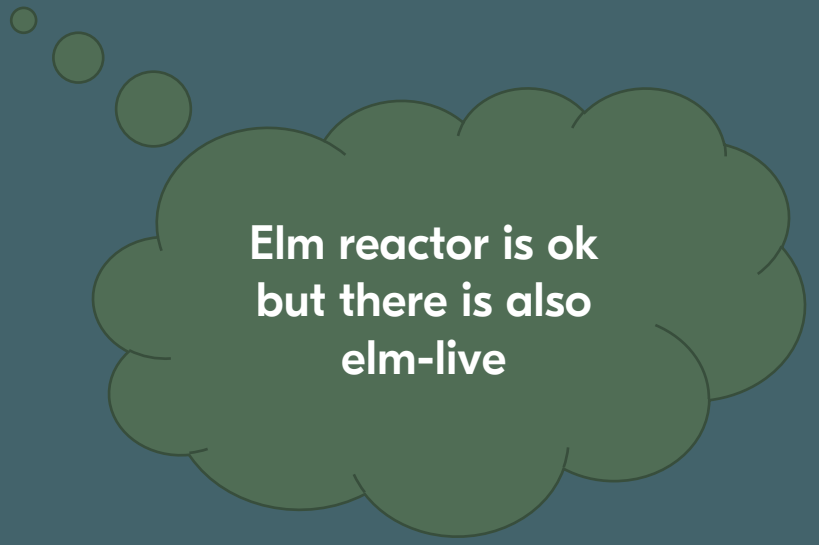

elm-currency-adder [~/IdeaProjects/elm-currency-adder] - .../src/Main.elm
elm-currency-adder
src
Main.elm
Add Configuration...
Project
Main.elm
elm-currency-adder ~/IdeaProjects/elm-currency-ad
.idea
src
Main.elm
elm.json
elm-currency-adder.iml
External Libraries
Scratches and Consoles
You must specify a path to the Elm compiler
Setup toolchain
1 module Main exposing (..)
Preferences
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
JavaScript
Play Configuration
Python Template Languages
Schemas and DTDs
BashSupport
BDD
ColdFusion
Concordion
Elm
Flask
JavaFX
Kotlin
Markdown
Node.js and NPM
OSGi
OSGi Framework Instances
Reactor
Spring
SQL Dialects
SQL Resolution Scopes
Style Sheets
Template Data Languages
Languages & Frameworks › Elm
For current project
Reset
Elm Compiler
Location: /usr/local/bin/elm
Auto Discover
Version: 0.19.1
elm-format
Location: /usr/local/bin/elm-format
Auto Discover
Version: 0.8.1
Keyboard shortcut: No Shortcut Change
Run when file saved?:
elm-test
Location: /usr/local/bin/elm-test
Auto Discover
Version: 0.19.1
Cancel
Apply
OK
1: Project
2: Favorites
7: Structure

Hello world

```elm
module Main exposing (main)

import Browser
import Html exposing (Html, div, h1, text)


main =
    Browser.sandbox
        { init = init
        , update = update
        , view = view
        }


type Msg
    = NoOp


type alias Model =
    String
```

```elm
init : Model
init =
    "some string"

update : Msg -> Model -> Model
update msg model =
    case msg of
        NoOp ->
            model


view : Model -> Html Msg
view model =
    div []
        [ h1 []
            [ text "Add money of different curr
            ]
        ]
```

```
> elm reactor
Go to http://localhost:8000 to see your
project dashboard.
```



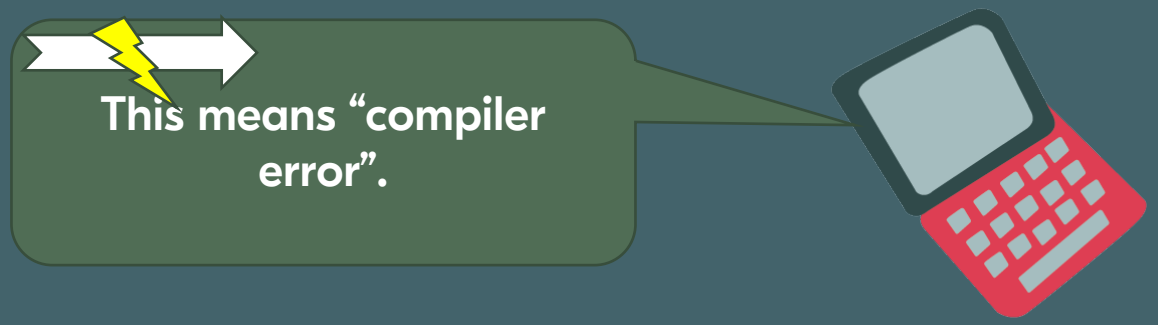
Let's add input

```elm
view : Model -> Html Msg
view model =
    div []
        [ h1 []
            [ text "Add money of different currencies"
            ]
        , div []
            [ input [ onInput LeftValue ] []
            , text " + "
            , input [ onInput RightValue ] []
            ]
        ]
```

```elm
type Msg
    = LeftValue String
    | RightValue String
```

```elm
update : Msg -> Model -> Model
update msg model =
    case msg of
        LeftValue string ->
            { model | leftValue = string }

        RightValue string ->
            { model | rightValue = string }
```

```elm
type alias Model =
    { leftValue : String
    , rightValue : String
    }
```



Add money of different currencies

[ ] + [ ]

Let's show the result

```elm
view : Model -> Html Msg
view model =
    div []
        [ h1 []
            [ text "Add money of different currencies"
            ]
        , div []
            [ input [ onInput LeftValue ] []
            , text " + "
            , input [ onInput RightValue ] []
            , text " = "
            , viewResult model
            ]
        ]
```

```elm
viewResult : Model -> Html Msg
viewResult model =
    case String.toFloat model.leftValue of
        Just left ->
            case String.toFloat model.rightValue of
                Just right ->
                    text <| String.fromFloat (left + right)

                Nothing ->
                    text "?"

        Nothing ->
            text "?"
```

Add money of different currencies

1330 + 7 = 1337

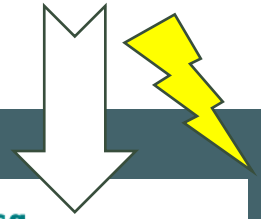
Per Lundholm

Let's make a table with currencies
Note the use of `map` to divide and conquer the problem

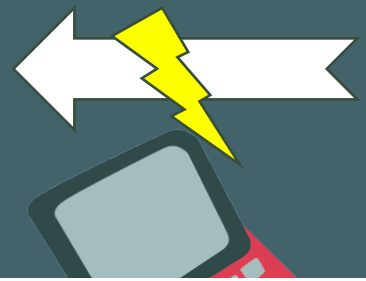
```elm
-- Data


currencies : List Currency
currencies =
    [ Currency "EUR" 10.57
    , Currency "USD" 9.5
    , Currency "CHF" 9.9
    , Currency "INR" 0.13
    ]
```

```elm
type alias Currency =
    { code : String
    , rate : Float
    }
```

```elm
view : Model -> Html Msg
view model =
    div []
        [ h1 []
            [ text "Add money of different currencies"
            ]
        , div []
            [ input [ onInput LeftValue ] []
            , text " + "
            , input [ onInput RightValue ] []
            , text " = "
            , viewResult model
            , viewCurrenciesTable model
            ]
        ]
```



```elm
viewCurrenciesRows : List Currency -> List (Html Msg)
viewCurrenciesRows c =
    List.map viewCurrencyRow c
```



```elm
viewCurrencyRow : Currency -> Html Msg
viewCurrencyRow c =
    tr [] [ td [] [ text c.code ], td [] [ text (String.fromFloat c.rate) ] ]
```

```elm
viewCurrenciesTable : Model -> Html Msg
viewCurrenciesTable model =
    table
        []
        ([ thead []
            [ th [] [ text "Code" ]
            , th [] [ text "Rate" ]
            ]
        ]
            ++ viewCurrenciesRows model.currencies
        )
```

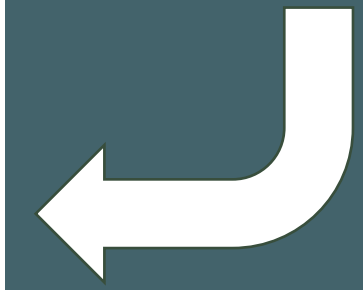
crisp.

Per Lundholm

```elm
viewCurrenciesTable : Model -> Html Msg
viewCurrenciesTable model =
    table
        []
        ([ thead []
            [ th [] [ text "Code" ]
            , th [] [ text "Rate" ]
            ]
         ]
            ++ viewCurrenciesRows model.currencies
        )
```

```elm
type alias Model =
    { leftValue : String
    , rightValue : String
    , currencies : List Currency
    }


init : Model
init =
    Model "" "" currencies
```

```elm
-- Data


currencies : List Currency
currencies =
    [ Currency "EUR" 10.57
    , Currency "USD" 9.5
    , Currency "CHF" 9.9
    , Currency "INR" 0.13
    ]
```

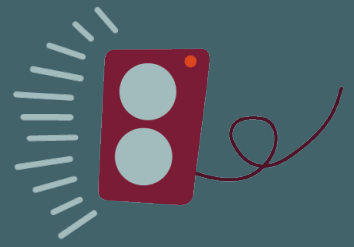


Add money of different currencies

[ ] + [ ] = ?

Code	Rate
EUR	10.57
USD	9.5
CHF	9.9
INR	0.13

Per Lundholm



Please stand by

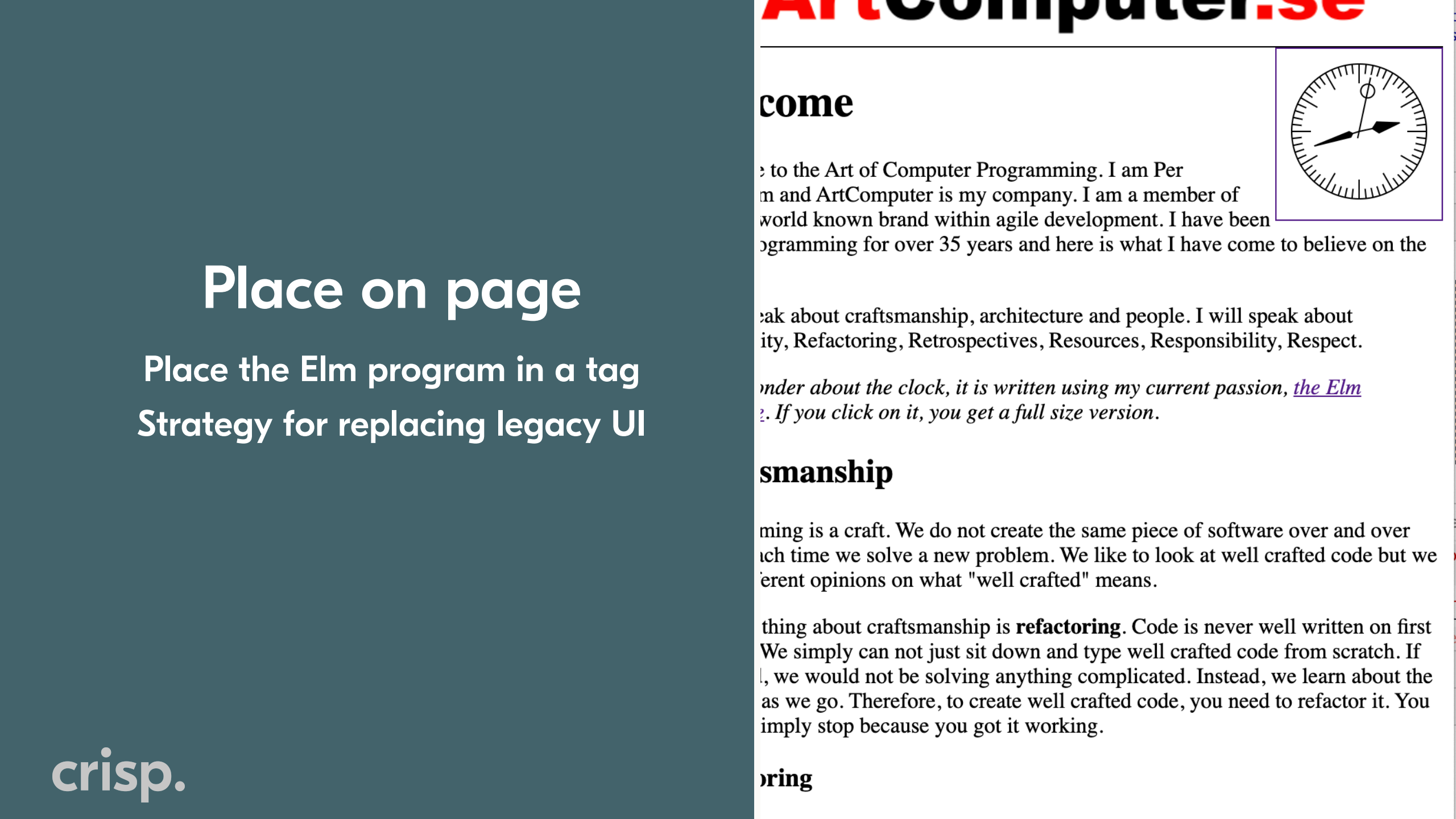
We will continue after this message from the HTML world.

Place on page

Place the Elm program in a tag

Strategy for replacing legacy UI

crisp.



```
<script type="text/javascript" src="elm.js"></script>
```

```
elm make src/Clock.elm --output=elm.js
```

```
<div id="clock"></div>
```

```
<script type="text/javascript">
    var clockDiv = document.getElementById('clock');
    Elm.Clock.init({node: clockDiv});
</script>
```

Thank you!

Ask me about Elm

https://github.com/perty/elm-currency-adder

crisp.

Per Lundholm

Continuing the coding

Select currencies

```elm
-- Data


currencies : List Currency
currencies =
    [ Currency "EUR" 10.57
    , Currency "USD" 9.5
    , Currency "CHF" 9.9
    , Currency "INR" 0.13
    ]
```

```elm
view : Model -> Html Msg
view model =
    div []
        [ h1 []
            [ text "Add money of different currencies"
            ]
        , div []
            [ input [ onInput LeftValue ] []
            , select [ onInput LeftCurrency ] currencyOptions
            , text " + "
            , input [ onInput RightValue ] []
            , select [ onInput RightCurrency ] currencyOptions
            , text " = "
            , viewResult model
            , viewCurrenciesTable model
            ]
        ]
```

```elm
currencyOptions : List (Html Msg)
currencyOptions =
    [ option [] [ text "--" ] ] ++ (List.map currencyOption <| List.map .code currencies)


currencyOption : String -> Html Msg
currencyOption currency =
    option [] [ text currency ]
```

Per Lundholm

Select currencies

```
view : Model -> Html Msg
view model =
    div []
        [ h1 []
            [ text "Add money of different currencies"
            ]
        , div []
            [ input [ onInput LeftValue ] []
            , select [ onInput LeftCurrency ] currencyOptions
            , text " + "
            , input [ onInput RightValue ] []
            , select [ onInput RightCurrency ] currencyOptions
            , text " = "
                                    ult model
                                    renciesTable model
```

```
update : Msg -> Model -> Model
update msg model =
    case msg of
        LeftValue string ->
            { model | leftValue = string }

        LeftCurrency string ->
            { model | leftCurrency = string }

        RightValue string ->
            { model | rightValue = string }

        RightCurrency string ->
            { model | rightCurrency = string }
```

```
type Msg
    = LeftValue String
    | LeftCurrency String
    | RightValue String
    | RightCurrency String
```

crisp.

Per Lundholm

Select currencies

```elm
update : Msg -> Model -> Model
update msg model =
    case msg of
        LeftValue string ->
            { model | leftValue = string }

        LeftCurrency string ->
            { model | leftCurrency = string }

        RightValue string ->
            { model | rightValue = string }

        RightCurrency string ->
            { model | rightCurrency = string }
```

```elm
type alias Model =
    { leftValue : String
    , leftCurrency : String
    , rightValue : String
    , rightCurrency : String
    , currencies : List Currency
    }
```

```elm
init : Model
init =
    { leftValue = ""
    , leftCurrency = ""
    , rightValue = ""
    , rightCurrency = ""
    , currencies = currencies
    }
```

Add money of different currencies

55 EUR + 6.5 USD = 61.5

Code	Rate
EUR	10.57
USD	9.5
CHF	9.9
INR	0.13

crisp.

Per Lundholm

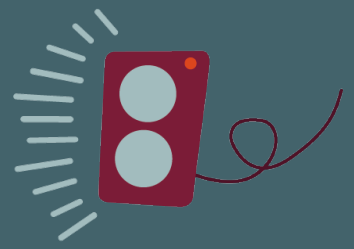
Use currencies. Add two amounts of different currencies. Result in 2nd currency.

```elm
viewResult : Model -> Html Msg
viewResult model =
    case String.toFloat model.leftValue of
        Just left ->
            case String.toFloat model.rightValue of
                Just right ->
                    text <|
                        (String.fromFloat <|
                            addTwoCurrencies left model.leftCurrency right model.rightCurrency
                        )
                            ++ " "
                            ++ model.rightCurrency

                Nothing ->
```

```elm
addTwoCurrencies : Float -> String -> Float -> String -> Float
addTwoCurrencies v1 c1 v2 c2 =
    let
        rate1 =
            findCurrencyRate c1

        rate2 =
            findCurrencyRate c2
    in
    ((rate1 * v1) + (rate2 * v2)) / rate2
```

```elm
-- Return the currency rate. We can be confident that the user c
-- a none-existing currency so a default of 1.0 is safe.


findCurrencyRate : String -> Float
findCurrencyRate codeName =
    let
        maybeRate =
            List.filter (\c -> c.code == codeName) currencies
                |> List.map .rate
                |> List.head
    in
    Maybe.withDefault 1.0 maybeRate
```

Per Lundholm



Improve this code

There is too many primitives, Float and String everywhere. We should have domain modelled types.

There is no way of telling which field is problematic. We should give the user feedback, e.g. marking field that is not correct. Change the attributes for that field to make it visually apparent.