ELM

FUNCTIONAL REACTIVE PROGRAMMING IN THE BROWSER

WHO'S TALKING?

BASTIAN KROL

Developer at codecentric

@bastiankrol

basti1302

bastian.krol@codecentric.de

- Software Development
- Agile Coaching
- Consulting
- Continuous Delivery
- Agile Software Factory
- Software Architecture
- DevOps
- Performance Tuning
- Big Data
- Operations

WE'RE HIRING!

- Karlsruhe
- Stuttgart
- Frankfurt
- München
- Berlin
- Hamburg
- Solingen
- Düsseldorf
- Münster
- Dortmund

https://www.codecentric.de/karriere/offene-stellen/

WHAT IS ELM

- Purely Functional
- Static Type System
- Reactive (FRP)
- Compiles to JavaScript
- Open Source
- Makes Web Development Delightful

WHY BOTHER?

MARKETING BLURBS

- Clean syntax
- No runtime exceptions
- Blazing fast rendering
- Libraries with SemVer guarantees
- Smooth JavaScript interop
- Time-traveling debugger

REDUCE BUG HUNTING

Bug by Nestor Ferraro / CC BY 2.0

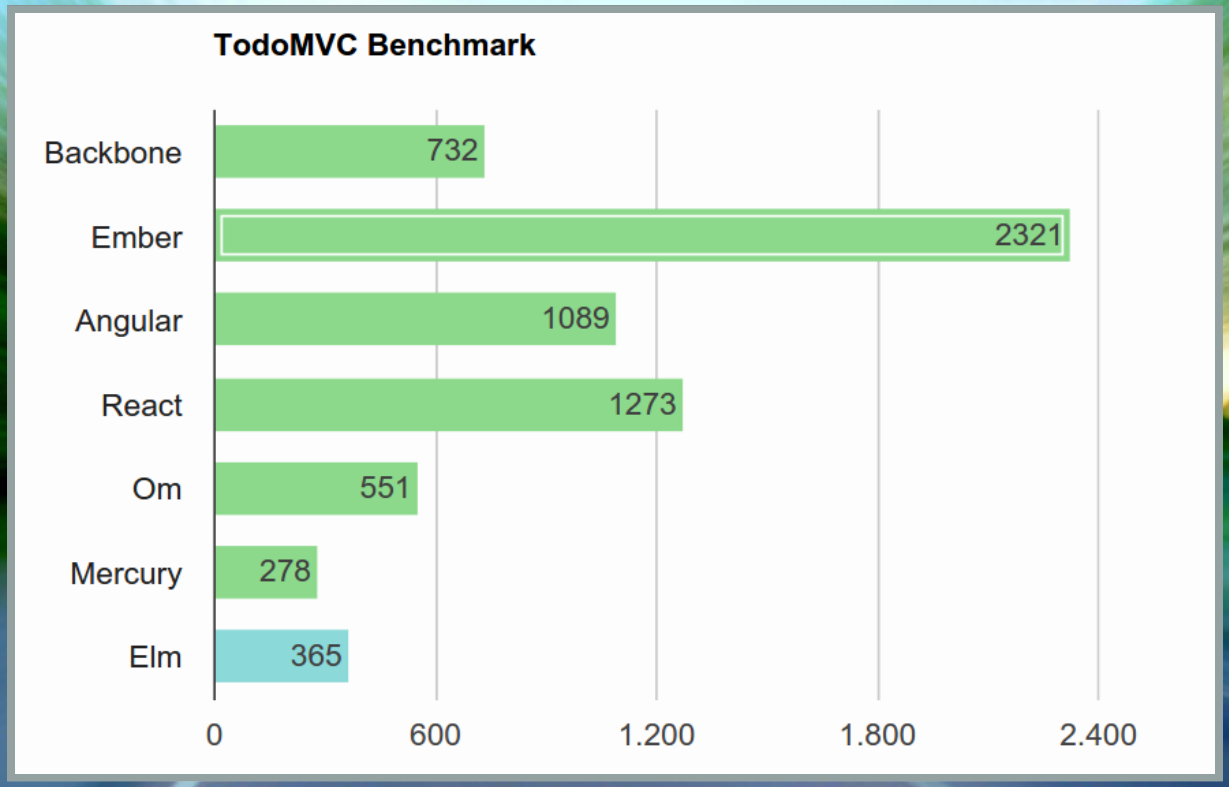
EVERYTHING IS STATELESS

Carved Stone" by Kit Carruthers / CC BY 2.0

swat by Marc Dalio / CC BY-NC-ND 2.0


ANCIENT ELM PROVERB
Once it compiles, it just works!

FAST RENDERING



SEMVER GUARANTEES

- Change public API of library
- Try to publish as minor/bugfix
- =>Nope
- Powered by Static Typing

SYNTAX BASICS

HELLO WORLD

Elm

```elm
import Html

main = Html.text "Hello Karlsruhe!"
```

Result

Hello Karlsruhe!

FUNCTIONS

Elm

```
import Html

greet str = Html.text str

main =
  greet "Hello Karlsruhe!"
```

Result

Hello Karlsruhe!

TYPE ANNOTATIONS

Elm

```elm
import Html exposing (Html)

greet : String -> Html
greet str =
  Html.text str

main : Html
main =
  greet "Hello Karlsruhe!"
```

Result

PARAMETER TYPES & RETURN TYPES

Elm

```
import Html exposing (Html)
import String

greet : Int -> String -> Html
greet number string =
  Html.text
    (String.repeat number string)

main : Html
main =
  greet 3 "Hello Karlsruhe! "
```

Result

Hello Karlsruhe!
Hello Karlsruhe!
Hello Karlsruhe!

FUNCTION APPLICATION

AKA DATA PIPELINES

Elm

```
greet : Int -> String -> Html
greet number string =
  string
  |> String.repeat number
  |> Html.text

main : Html
main =
  greet 3 "Hello Karlsruhe! "
```

Result

Hello Karlsruhe!
Hello Karlsruhe!
Hello Karlsruhe!

FUNCTIONAL REACTIVE PROGRAMMING

IN ELM

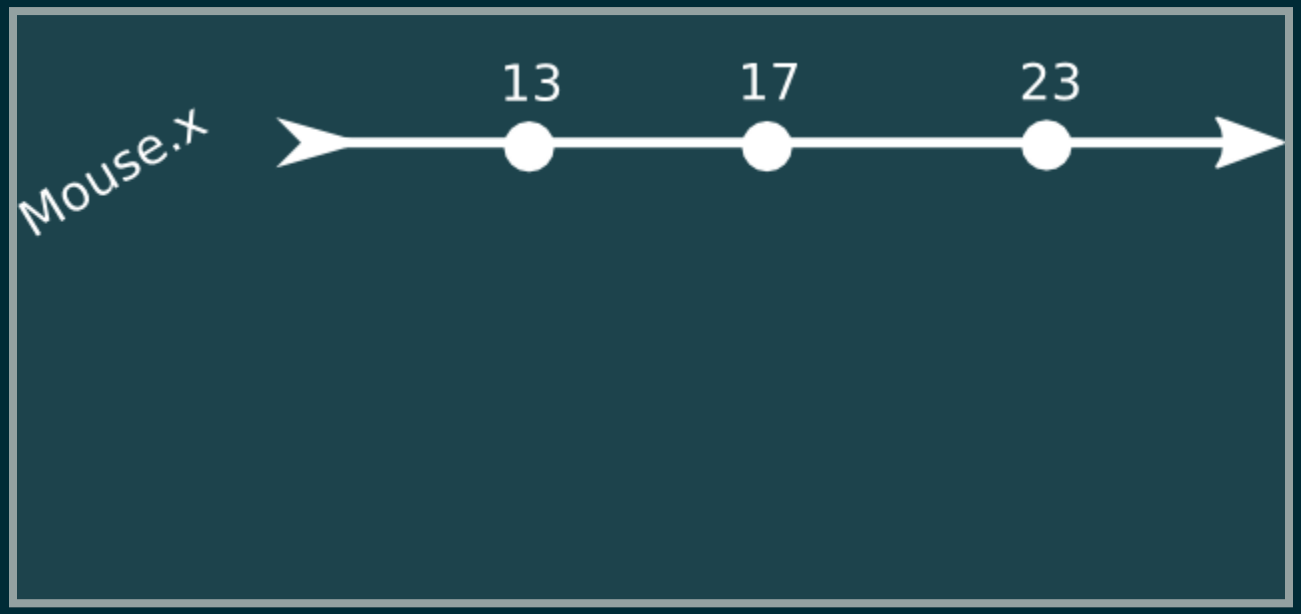
SIGNALS

Elm

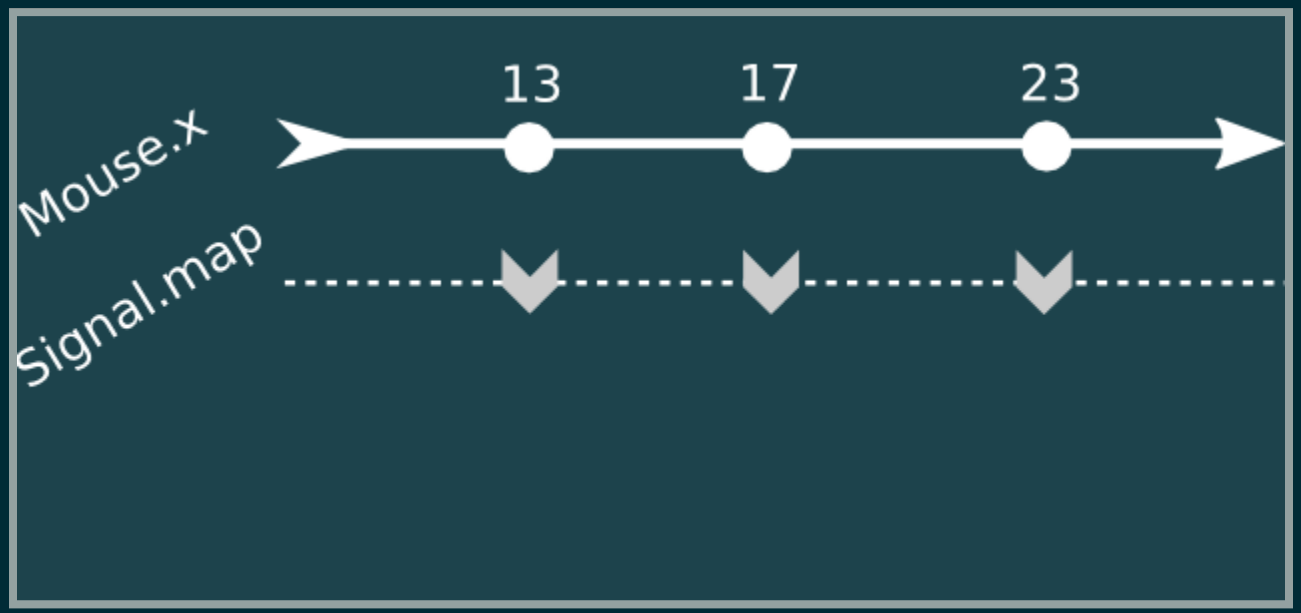
```
import Html exposing (Html)
import Mouse

toHtml : Int -> Html
toHtml x =
  Html.text (toString x)

main : Signal Html
main =
  Signal.map toHtml Mouse.x
```

Result

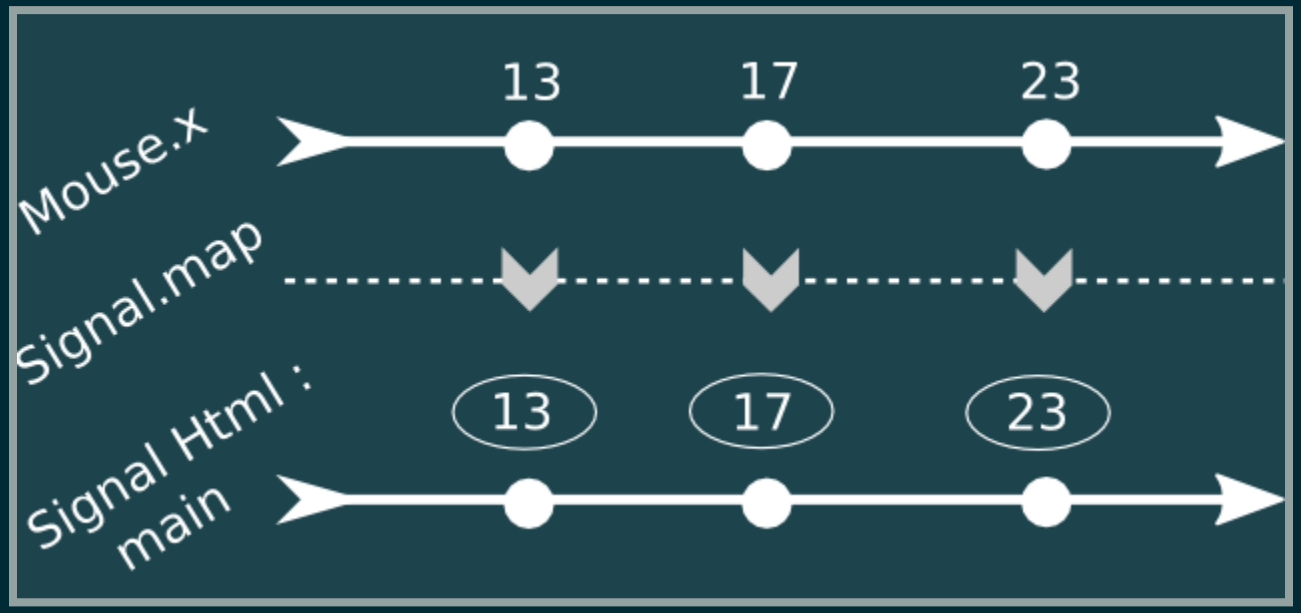
SIGNALS (CONT.)



SIGNALS (CONT.)



SIGNALS (CONT.)



STUFF YOU CAN DO WITH SIGNALS

- `merge` multiple signals
- `map` signals (transform each value)
- `filter` signals (drop events)
- ... lots of other interesting stuff ... :)

MORE ON SIGNALS

http://elm-lang.org/guide/reactivity
https://github.com/evancz/elm-architecture-tutorial

THE ELM COMPILER IS YOUR FRIEND

The dog and his man by C. I. Ştefănescu / CC BY 2.0

TYPOS

```elm
myFunction str =
  String.repeat 3 str

main : Html
main =
  Html.text (myFunctino "Elm")
```

```
-- NAMING ERROR ---------------------- errors/Spelling.elm

Cannot find variable `myFunctino`

9|    Html.text (myFunctino "Elm")
                 ^^^^^^^^^^
Maybe you want one of the following?

    myFunction
```

TYPE MISMATCH

```
subMismatch =
{ name = "Alice", age = 24 } == { name = "Bob", age = "30" }
```

```
-- TYPE MISMATCH ---------------------- errors/SubMismatch.elm

The right argument of (==) is causing a type mismatch.

2| { name = "Alice", age = 24 } == { name = "Bob", age = "30"
                                    ^^^^^^^^^^^^^^^^^^^^^^^^^^
(==) is expecting the right argument to be a:

    { ..., age : number }

But the right argument is:

    { ..., age : String }
```

FORGOT A CASE?

```
type Colour = Red | Green | Blue

colourToHex : Colour -> String
colourToHex colour =
  case colour of
    Red   -> "#f00"
    Green -> "#0f0"
```

FORGOT A CASE?

```
-- MISSING PATTERNS --------------- errors/ForgottenCase.elm

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

5|>  case colour of
6|>    Red   -> "#f00"
7|>    Green -> "#0f0"

You need to account for the following values:

    Main.Blue

Add a branch to cover this pattern!
```

RUNTIME EXCEPTIONS?

JavaScript

```
function repeatFirst(list) {
  var repeated = list[0].repeat(3);
  alert(repeated);
}
```

```
var list1 =
  [ 'hip', 'hop', 'hooray!' ];
...
repeatFirst(list1);
```

Run

```
var list2 = [];
```

Run

```
...
repeatFirst(list2);
```

COMPILE TIME > RUNTIME

```
repeatFirst list =
  let elem = List.head list
  in String.repeat 3 elem
```

```
> elm-make RepeatFirst.elm --output repeat-first.html
-- TYPE MISMATCH ---------------------------- RepeatFirst.elm

The 2nd argument to function repeat is causing a mismatch.

8|      String.repeat 3 elem
                       ^^^^
Function `repeat` is expecting the 2nd argument to be:
    String
But it is:
    Maybe a

Hint: I always figure out the type of arguments from left
to right. If an argument is acceptable when I check it, I
assume it is "correct" in subsequent checks. So the problem
```

CORRECTED VERSION

```
repeatFirst list =
  let elem = List.head list |> Maybe.withDefault ""
  in String.repeat 3 elem
```

```
> elm-make RepeatFirst.elm --output repeat-first.html
Success! Compiled 1 modules.
Successfully generated repeat-first.html
```

PROTECTION FROM RUNTIME EXCEPTIONS

SUMMARY

- *No runtime exceptions!*
- Refactoring is *easy* and *safe*
- Coherent approach to user interaction with *signals*
- *Fast* rendering
- Can be introduced into *existing* JS code bases
- Uniform Structure for Apps (the Elm Architecture, not covered today)


Q. E. D.
ELM MAKES
WEB DEVELOPMENT
DELIGHTFUL

LINKS:

- http://elm-lang.org/
- Online Editor/Sandbox
- Elm Friday (Tutorial/Blog Series)

QUESTIONS?

THANK YOU!

 @bastiankrol

 basti1302

 bastian.krol@codecentric.de