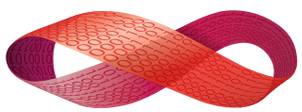
# Building a Graphical IDE in Elm/Purescript

## for a Distributed PLC Language Compiling to BEAM

by @doppioslash

05/05/2017 - ElixirConf EU - Barcelona



DIPL. PHYS. PEER STRITZINGER GMBH

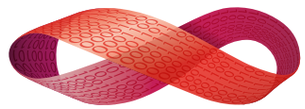
Hi, I'm

# Claudia Doppioslash

**Functional Programmer** & **Game Developer**

@doppioslash

www.lambdacat.com

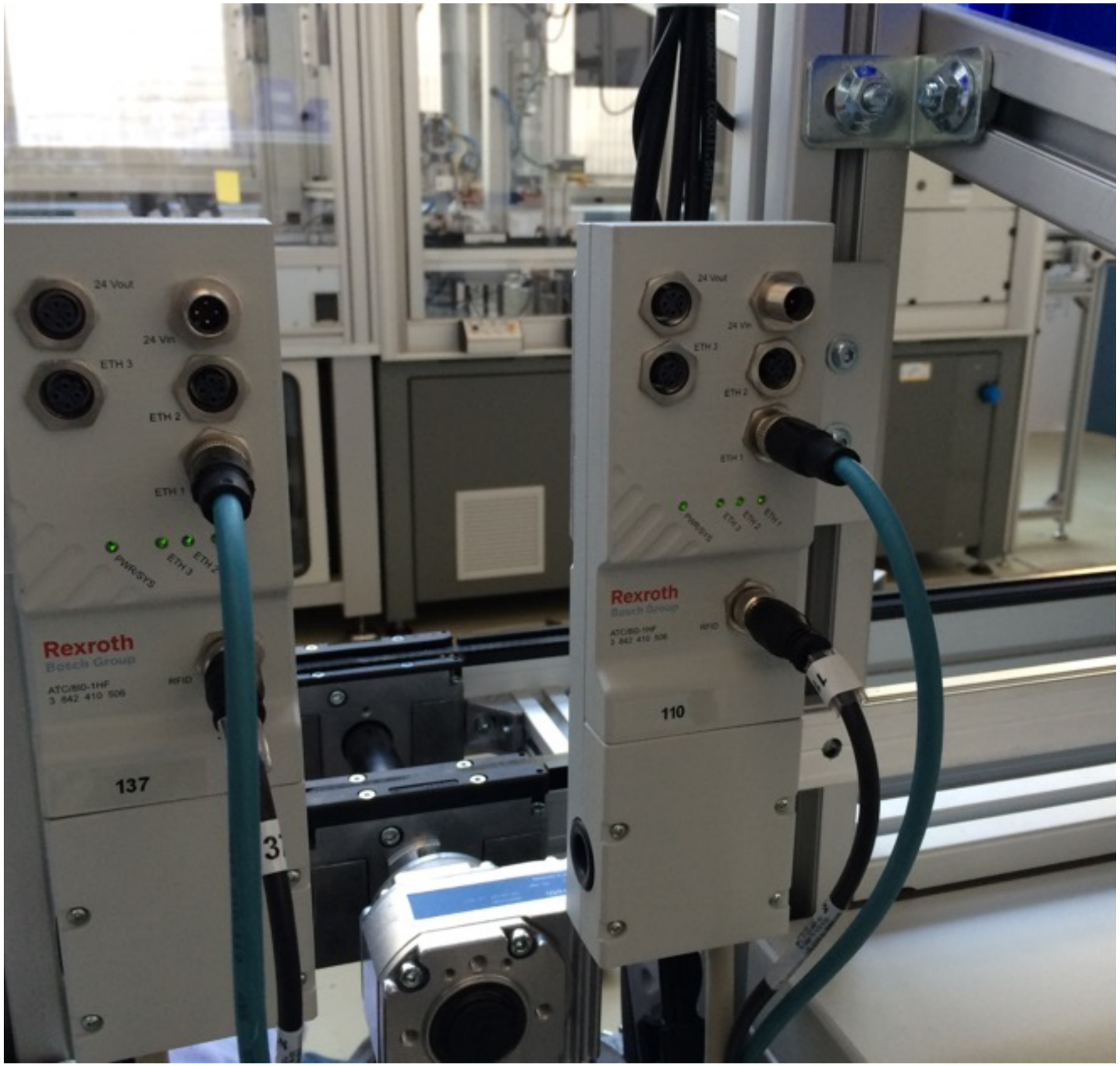


DIPL. PHYS. PEER STRITZINGER GMBH

# Peer Stritzinger GmbH

Functional and Failure Tolerant
Programming for Embedded,
Industrial Control and Automotive

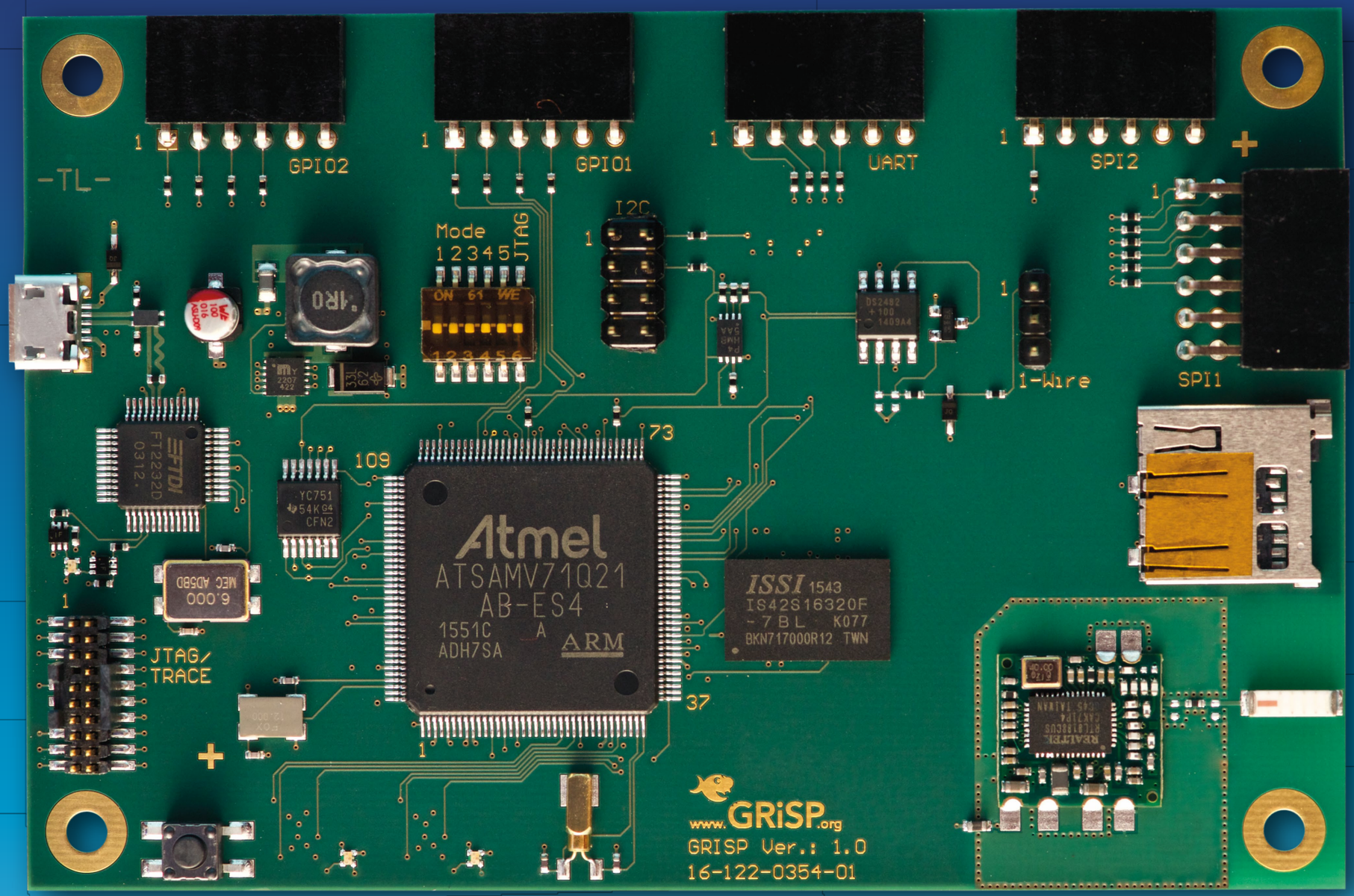

**www.stritzinger.com**


24 Vout
24 Vin
ETH 3
ETH 2
ETH 1
PWR/SYS
Rexroth
Bosch Group
ATC/BD-1HF
3 842 410 506
RFID
137
110


GEBIT
future talk
GEBIT


Rexroth

DIPL. PHYS. PEER STRITZINGER GMBH




# www.grisp.org

LISP
*flavored*
ERLANG


GRiSP

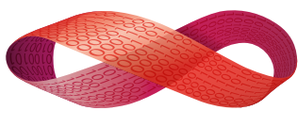

luerl

# Why are you here?

"I need to get some frontend code done,
and I hate Javascript"

Interested in Haskell-like languages

Undecided between Elm and Purescript



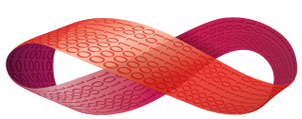
DIPL. PHYS. PEER STRITZINGER GMBH

# What are you getting

This is a WIP-mortem:

- why we made the choices we made
- what went right/wrong
- enough Elm to understand what's going on
- our experience of porting from Elm to Purescript

Not an Elm or Purescript guide, also not latest Elm version.



DIPL. PHYS. PEER STRITZINGER GMBH

# 0.16? 0.17?

The jump from 0.16 and 0.17 in Elm

## 0.16

FRP

mailboxes

addresses

signals

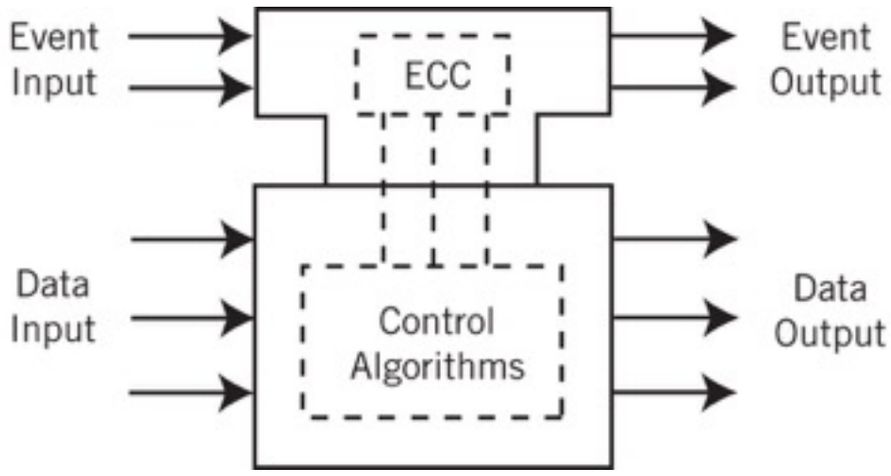
foldp

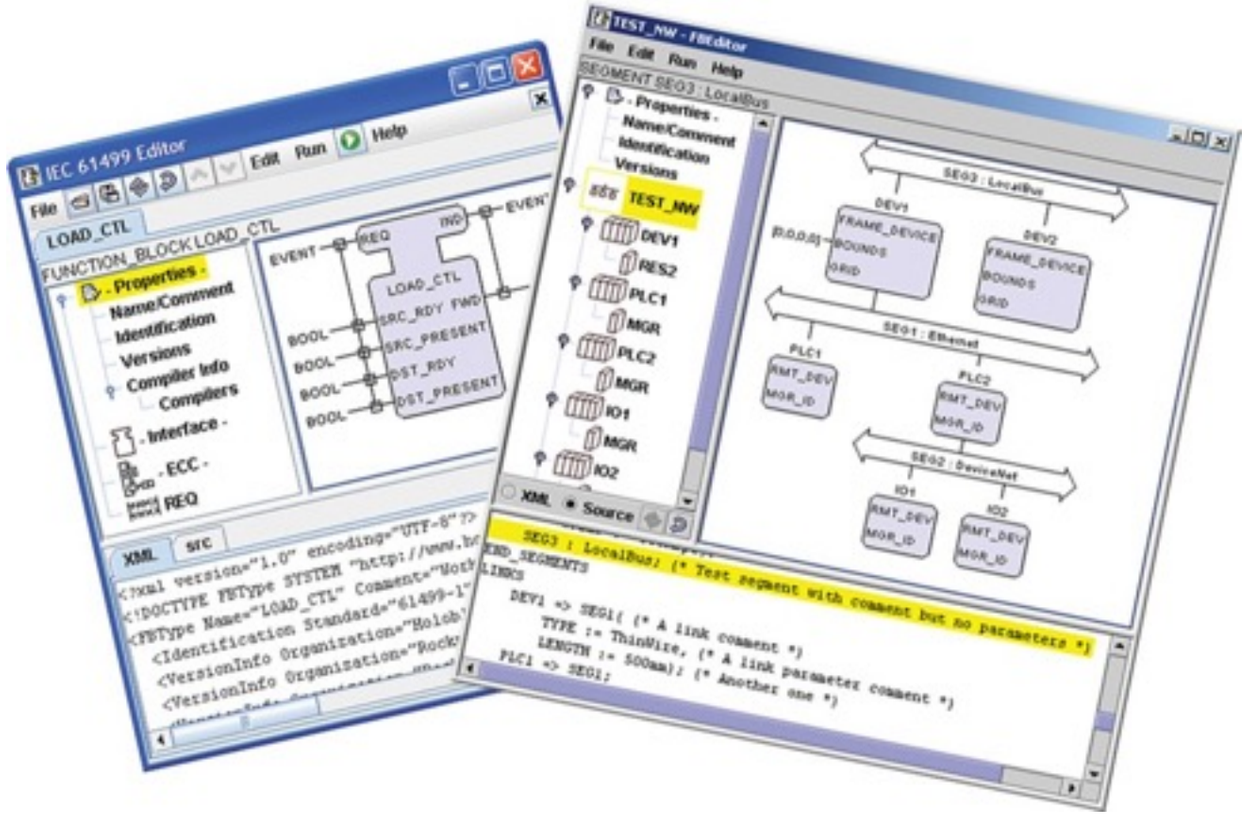
## 0.17

# Our Project

Visual IDE for PLC language **IEC61499**

“A programmable logic controller, PLC, or programmable controller is a digital computer used for automation"



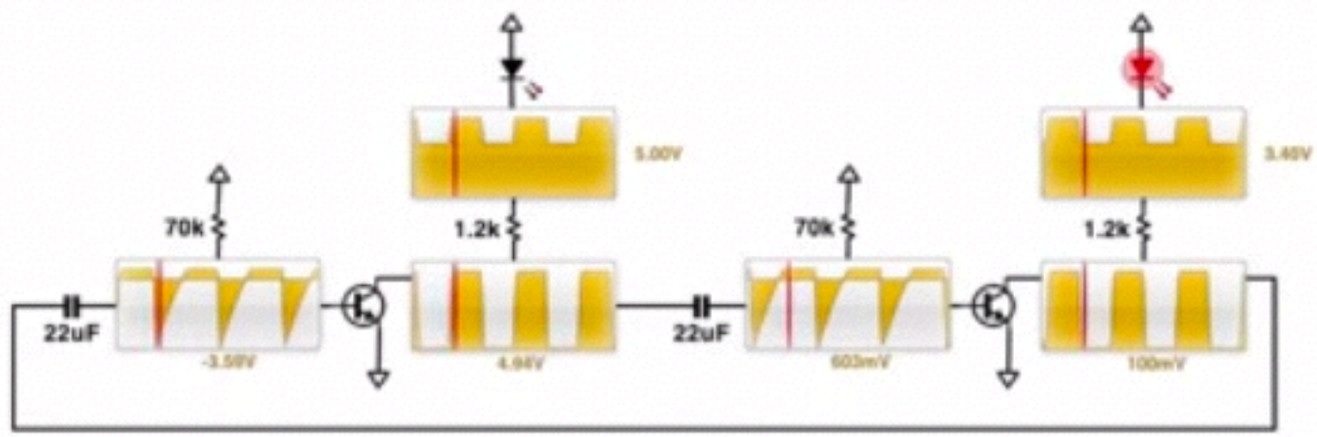




(images from http://www.controldesign.com/articles/2007/202/?show=all)

DIPL. PHYS. PEER STRITZINGER GMBH

# Our Project

Inspired by Bret Victor's "Inventing on Principle" talk:

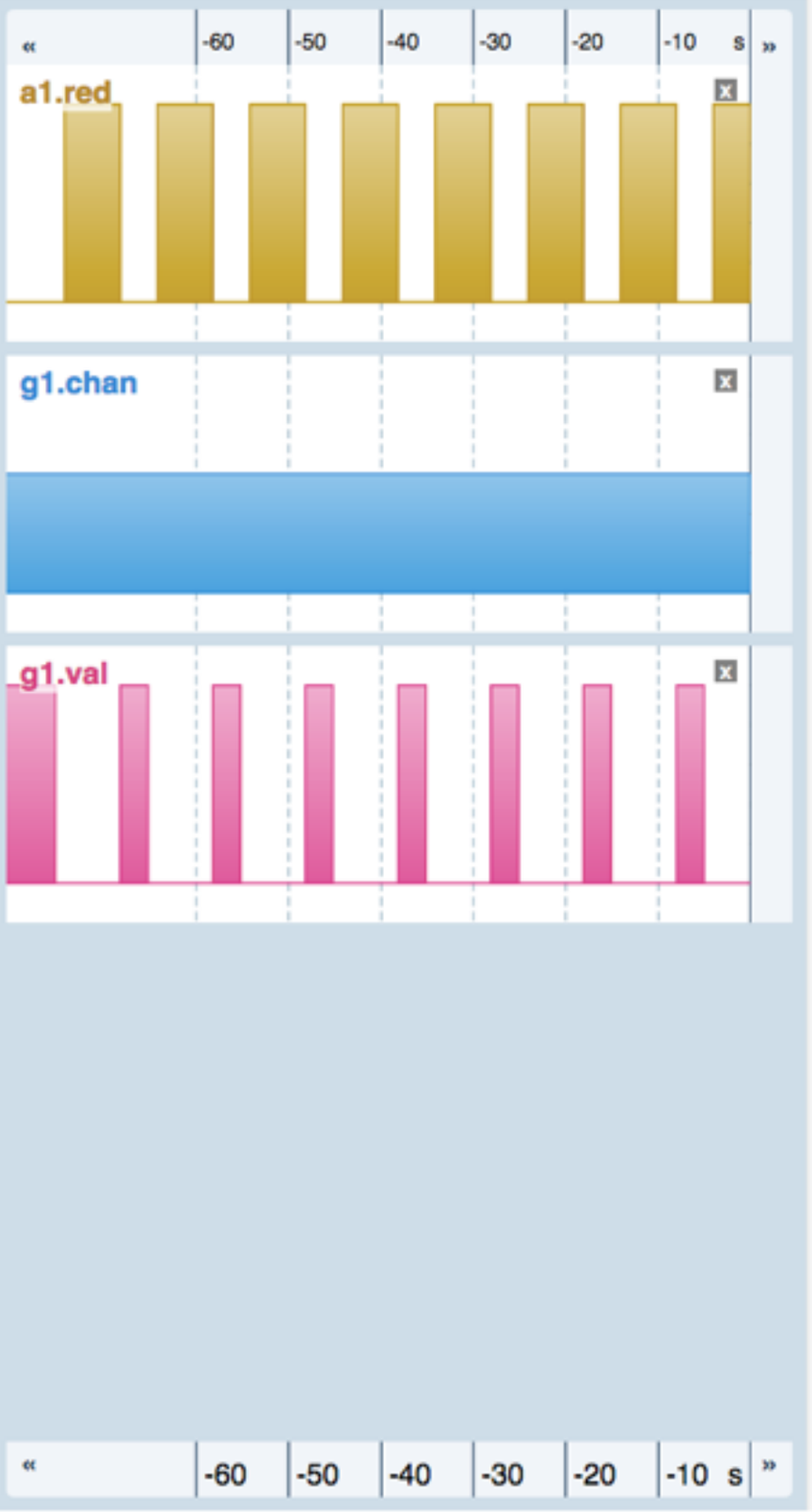



DIPL. PHYS. PEER STRITZINGER GMBH

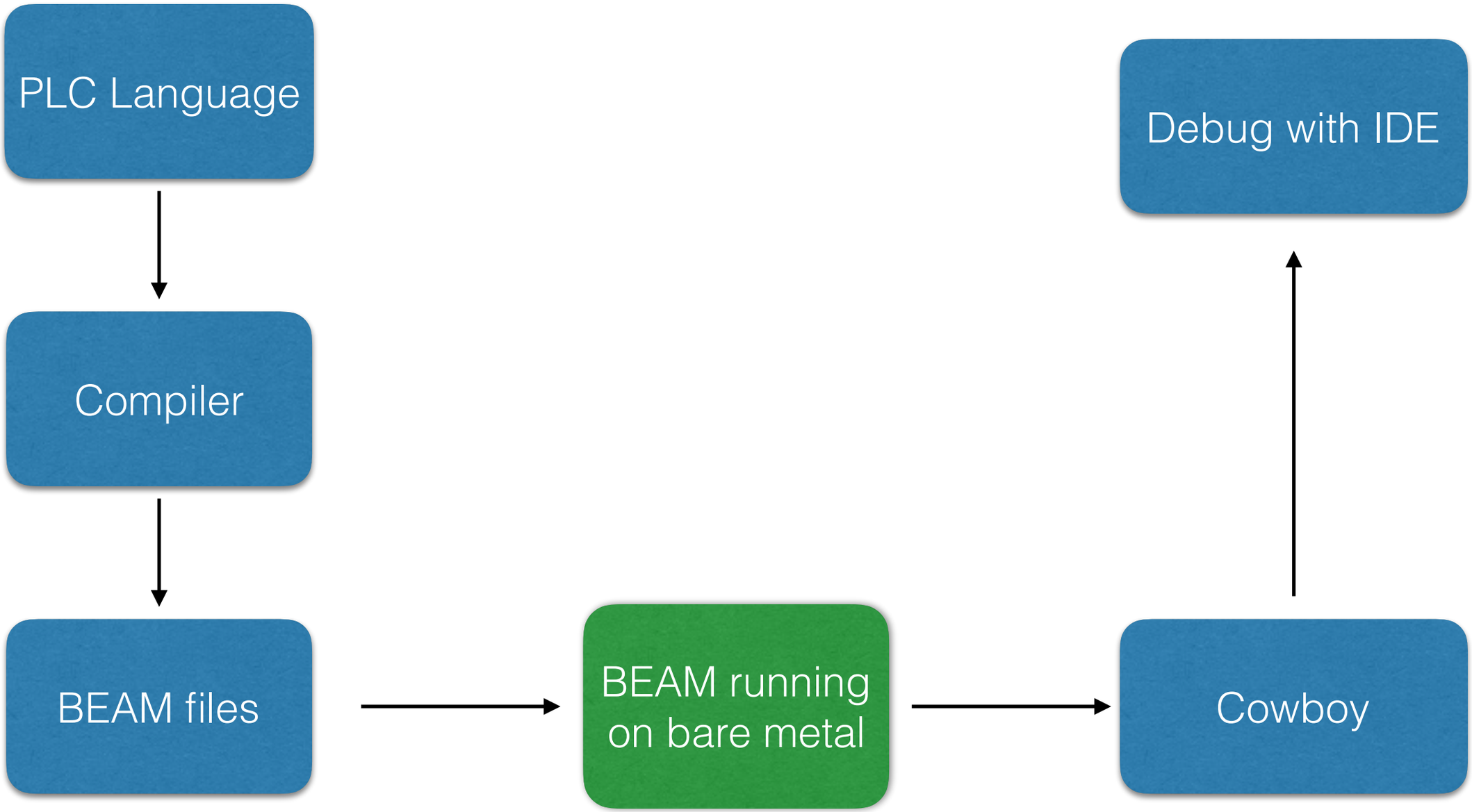
# Our Project

ATCnet | ampel_app

Rexroth
Bosch Group




# Our Project




DIPL. PHYS. PEER STRITZINGER GMBH

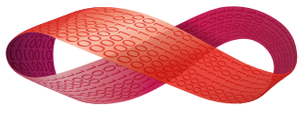
# Requirements

## Many platforms to support

All PC OSs & iPad Pro

## Decent performance

Needs to be interactive
~30fps should be fine

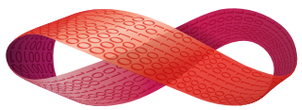


DIPL. PHYS. PEER STRITZINGER GMBH

# Frontend Tech Choice

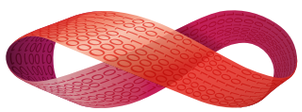
**Web Technologies** because cross-platform

Hence: **Javascript, CSS, Svg**



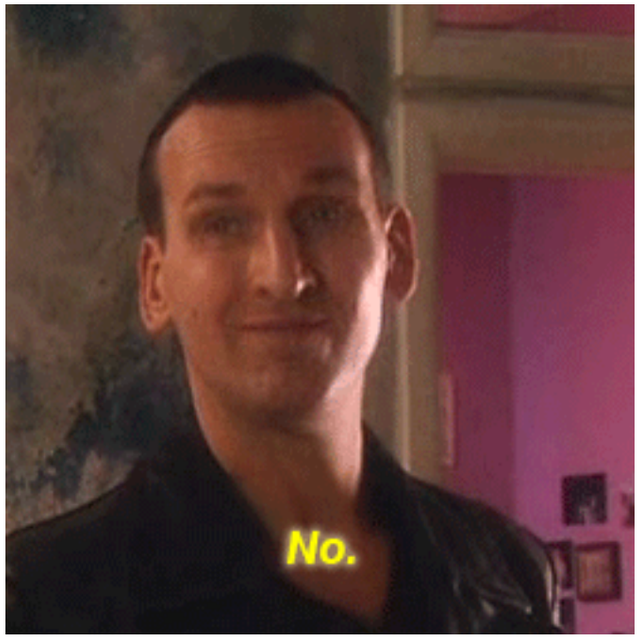
DIPL. PHYS. PEER STRITZINGER GMBH

# Wait a minute, Javascript?



DIPL. PHYS. PEER STRITZINGER GMBH

# Wait a minute, Javascript?




# …let’s not.

# Possible Choices, Then

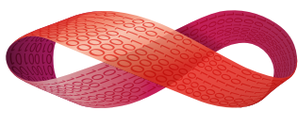
Ready at the time:

Clojurescript 


Elm 

CoffeScript 

Typescript 

DIPL. PHYS. PEER STRITZINGER GMBH

# Why did we chose Elm?

**Functional Reactive Programming**

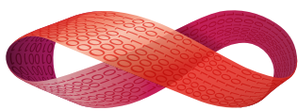
(it's gone now though)

**Good error messages**

(so good everyone is imitating them)

**Some concepts somewhat similar to Erlang**

(e.g. Mailboxes)



DIPL. PHYS. PEER STRITZINGER GMBH

# What is Elm?

Pure Functional

Strongly Typed

Eagerly evaluated

Compiles to **Javascript**

**Functional Reactive Programming** ( < 0.17 )

Haskell-like syntax

Very **small**

Optimised for **learning curve** (>0.16)

Similar to Haskell but no advanced types

DIPL. PHYS. PEER STRITZINGER GMBH

# Elm Pros compared to JS

**If it compiles, it works** (90% of the time)

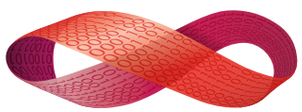
Confident **refactoring**

Clean

Much fewer LOC

The famous great error messages

(certainly better than undefined is not a function)

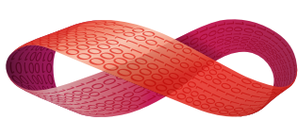


DIPL. PHYS. PEER STRITZINGER GMBH

# The famous Elm errors

- contextual
- correct common errors

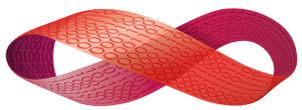
```
-- MISSING PATTERNS ---------------------------------------------- tmp.elm

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

4|>    case list of
5|>        [x] ->
6|>            x
7|>
8|>        _ :: rest ->
9|>            last rest

You need to account for the following values:

    []

Add a branch to cover this pattern!

If you are seeing this error for the first time, check out these hints:
<https://github.com/elm-lang/elm-compiler/blob/0.16.0/hints/missing-patterns.md>
The recommendations about wildcard patterns and `Debug.crash` are important!
```



# The famous Elm errors

How do they do it?

- make it a priority
- carefully tracked on a git repo
- type system complexity
  (simpler = easier to have good errors)



DIPL. PHYS. PEER STRITZINGER GMBH

# Elm Pros compared to JS

Elm actually makes sense (seen the '**Wat**' talk?)

```
failbowl:~(master!?) $ jsc
> Array(16)
,,,,,,,,,,,,,,,
```

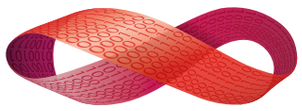
# Elm Cons compared to JS

Javascript **interop inflexible**

(less in 0.17)

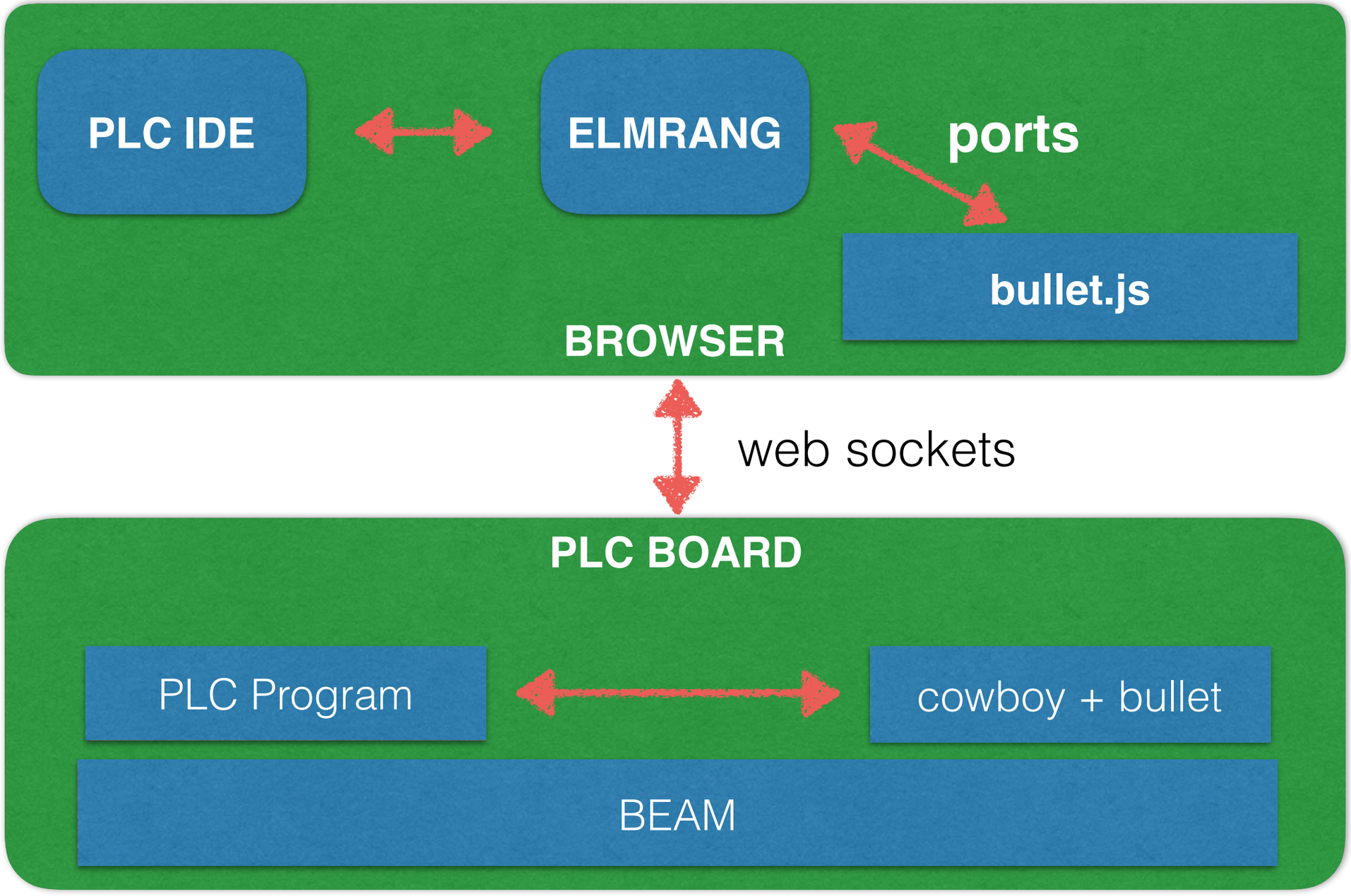
new language, still 0.x

…so, not that much.



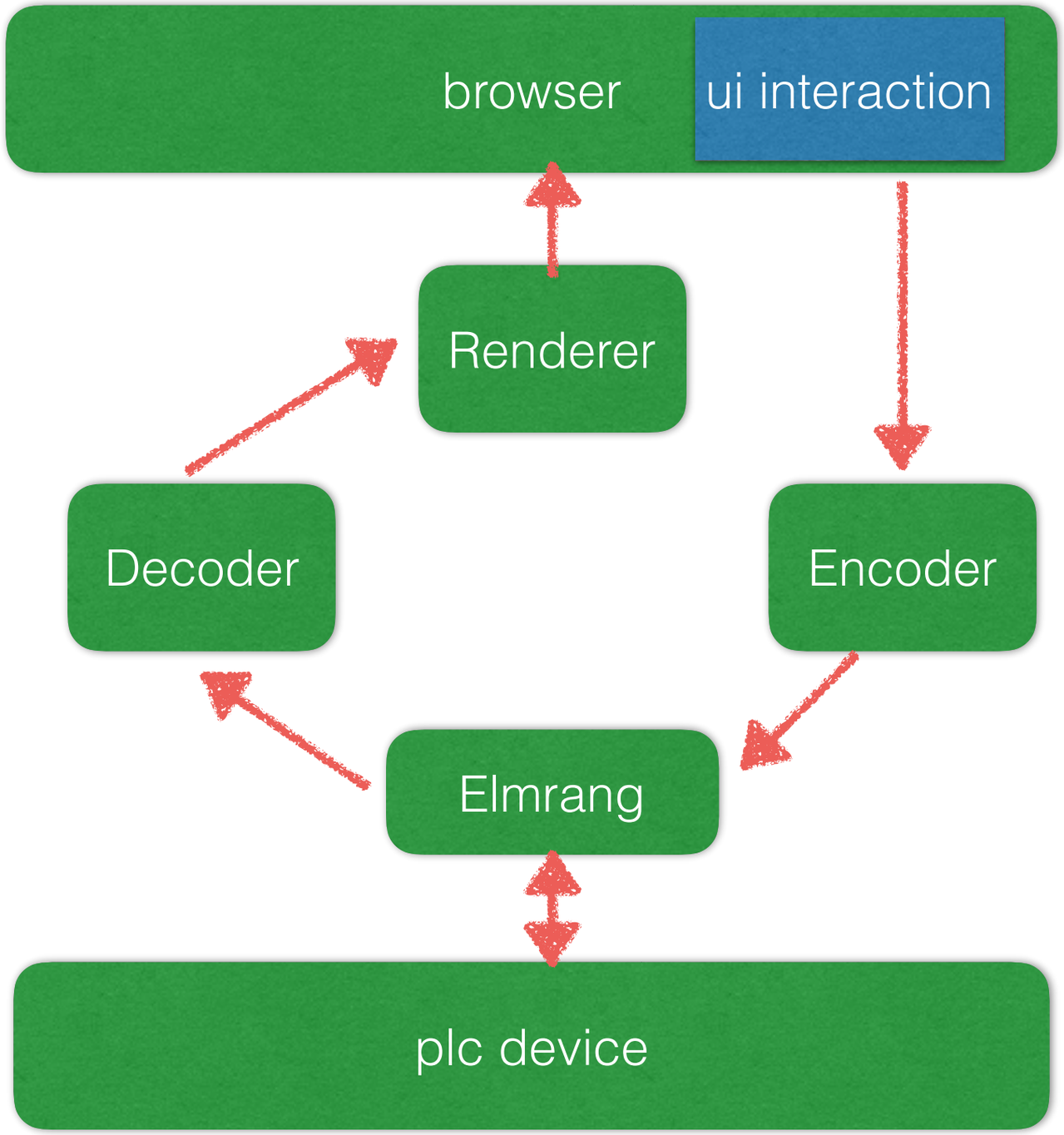
DIPL. PHYS. PEER STRITZINGER GMBH

# Our Project Structure



# Demo

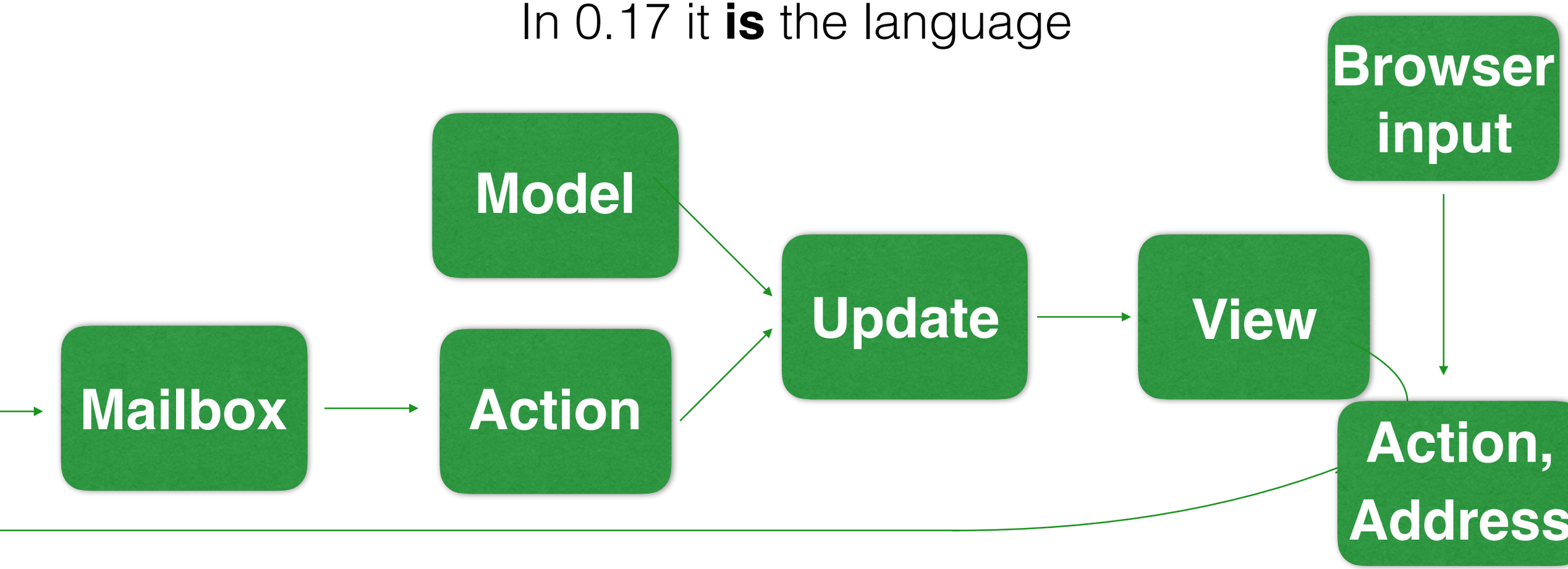
# PLC IDE Structure



# What is StartApp?

Implementation of **The Elm Architecture** for **0.16**

In 0.17 it **is** the language




**Beware: this is different in 0.17**

# PLC IDE Structure

Four **StartApp** connected by **Mailboxes**

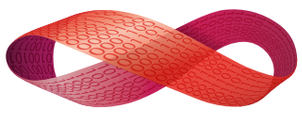
Wired into a parent StartApp, so nested StartApps

As in the structure invented by **foxdonut**

**Easy to expand**, add components

But no one ported it to 0.17 (may be impossible)

**Elmrang** can be a component using this structure



DIPL. PHYS. PEER STRITZINGER GMBH

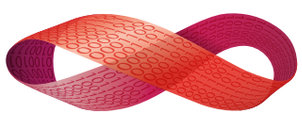
# Why are we still on 0.16?

We use **FRP** heavily

Porting code might not be **cost effective**

Frustrated with **lack of communication**
(e.g. no deprecation warnings)

Waiting for Elm evolution to **stabilise**



DIPL. PHYS. PEER STRITZINGER GMBH

# Production Problems

How to organise subcomponents in a big Elm app?

How to include an Elm project into an Erlang app?

How to store deps not on elm-package?

DIPL. PHYS. PEER STRITZINGER GMBH

# Organising Subcomponents

Every component has:

```
component/Action.elm
component/Model.elm
component/View.elm
component/Update.elm
component/Feature.elm
```

Wired in in App.elm and fed to Main.elm

DIPL. PHYS. PEER STRITZINGER GMBH

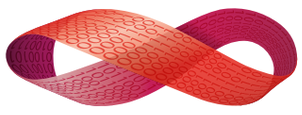
# Mixed Elm/Erlang Project

- `/elm` subdir in Erlang project
- compile Elm files to `/priv`
- add the .js to your html file
- we made a rebar3 plugin for this

DIPL. PHYS. PEER STRITZINGER GMBH

# Non elm-package deps

- fetch it from repo
- store it in a subdir of the erlang project
- move only the elm files to a subdir of the elm project
- not under `elm-stuff/`
- include the subdir in elm-package.json

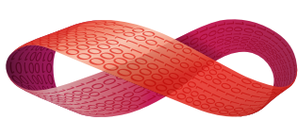


DIPL. PHYS. PEER STRITZINGER GMBH

# Rendering

Choices we had:

- WebGL (2d rendering engine)
- SVG (w or w/o CSS layout and animations)
- Html (not ideal)



DIPL. PHYS. PEER STRITZINGER GMBH

# Rendering

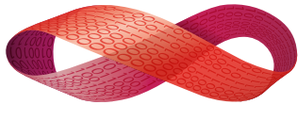
We use **Svg with CSS**

We try to do as much as we can with CSS

Animation in Elm can get complicated

CSS styles are in separate CSS files

We have an Svg & CSS expert on call



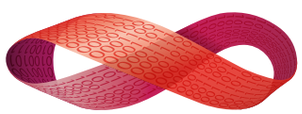
DIPL. PHYS. PEER STRITZINGER GMBH

# Rendering

**elm-html** and **elm-svg** have great syntax:

```
div [class “somecssclass”]
    [ p [] [text “a very well written paragraph”]
    , p [] [text “and another one”]
    ]
```

Based on virtualdom = fast



DIPL. PHYS. PEER STRITZINGER GMBH

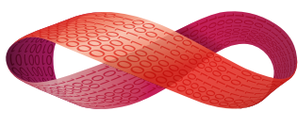
# Several words to the wise

## Be aware of what Elm is good for.

An Elm program has to fit the Elm Architecture
(which is good if it does fits, less if it doesn't)

## Wrapping Javascript libraries

There is no path to get a library that wraps a javascript library on elm-package (e.g. elm-d3)



DIPL. PHYS. PEER STRITZINGER GMBH

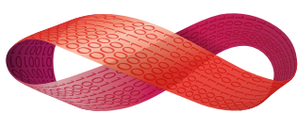
# Several words to the wise

## Elm is still experimental

Elm is still subject to big changes, expect to have to rewrite some of your code with a new version.

## Elm lacks a roadmap

There are short beta previews, and
Elm's author does semi-regular updates of what he's up to, in the elm-dev mailing list
(see: https://github.com/elm-lang/projects/blob/master/roadmap.md )

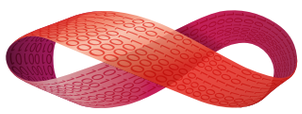


# What next?

We've skipped 0.17 and 0.18

Maybe come back when Elm is nearer to 1.0

Meanwhile taking another language for a spin,
porting a portion of our project to it



DIPL. PHYS. PEER STRITZINGER GMBH

# Possible Choices, Now

Ready now:

Bucklescript

Purescript 

Clojurescript 

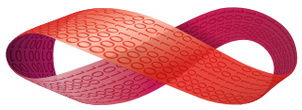
Elm 

CoffeScript 

Typescript 

Reason 

Fable 

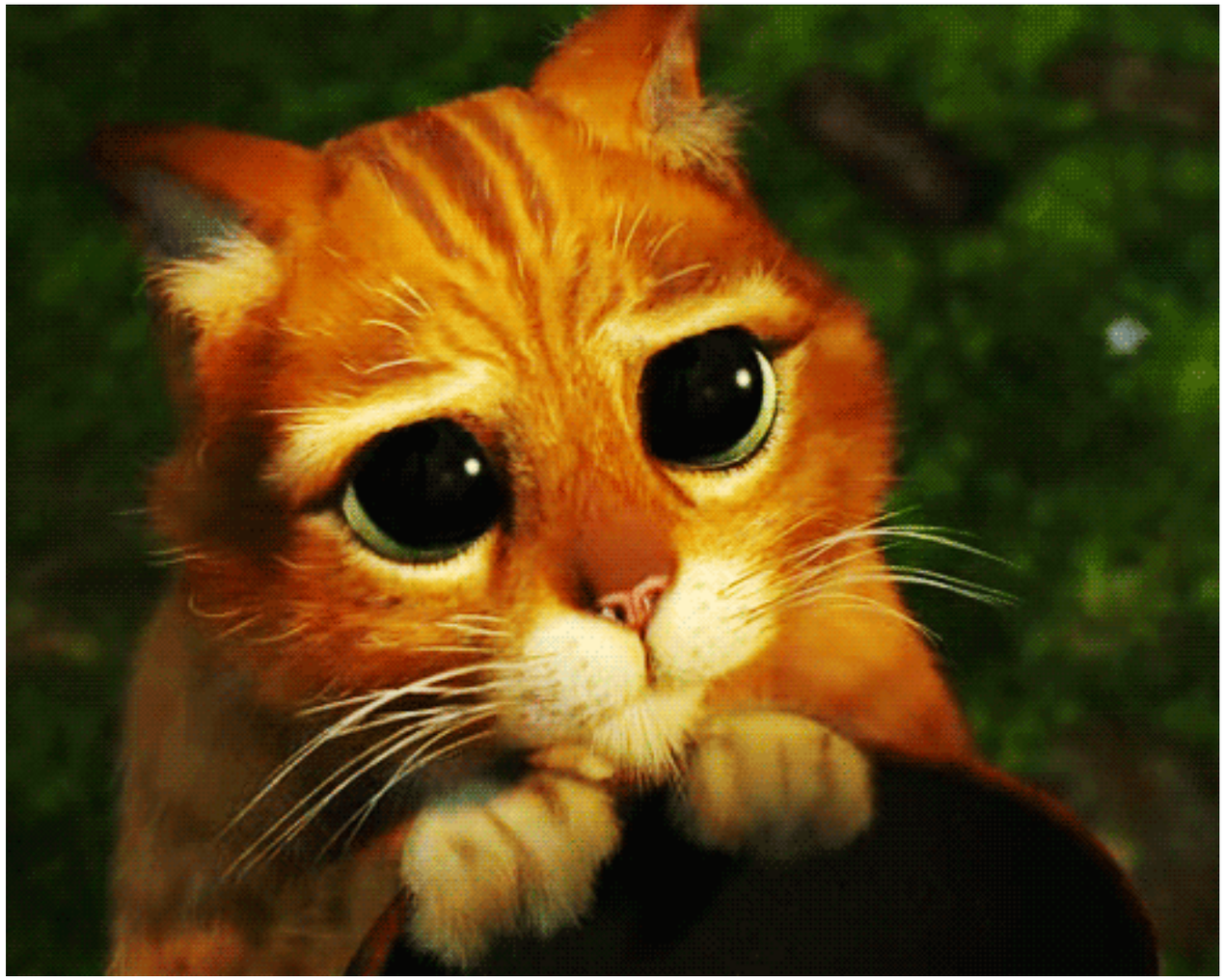


DIPL. PHYS. PEER STRITZINGER GMBH

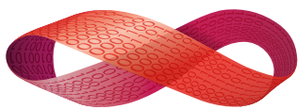

YOU MUST CHOOSE...
BUT CHOOSE WISELY.
quickmeme.com

# Our First Choice

**: “Please adopt me…”**



**“…I swear I won’t mention Monads”**



DIPL. PHYS. PEER STRITZINGER GMBH

# Our Second Choice

Purescript  : **“you’re free to do anything…**

**…if you can cope with the types”**

DIPL. PHYS. PEER STRITZINGER GMBH

# What is Purescript?

*Reminds you of anything?*

Pure Functional

Strongly Typed

Eagerly evaluated

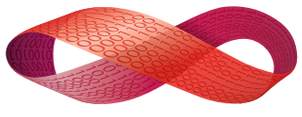
Compiles to Javascript

Advanced Types

Haskell-like syntax (with all the squiggles)

Generates readable Javascript, has no runtime

Open community, a bit of a roadmap

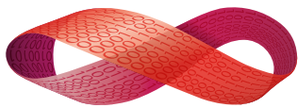


DIPL. PHYS. PEER STRITZINGER GMBH

# Philosophy Differences

Elm is made to be simple above anything else, have a quick learning curve

In Purescript you have most of the type features you have in Haskell, longer learning curve

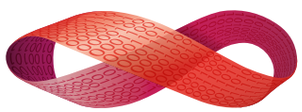


DIPL. PHYS. PEER STRITZINGER GMBH

# Philosophy Differences

Elm gives you only one possible program structure (Elm arch)

In Purescript there are many possible ways of structuring your app



# Why Purescript after Elm?

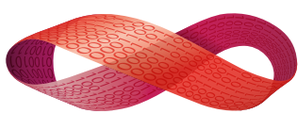
DIPL. PHYS. PEER STRITZINGER GMBH

# The Elm tradeoff

Preferring simpler types begets:

- smooth learning curve
- good error messages

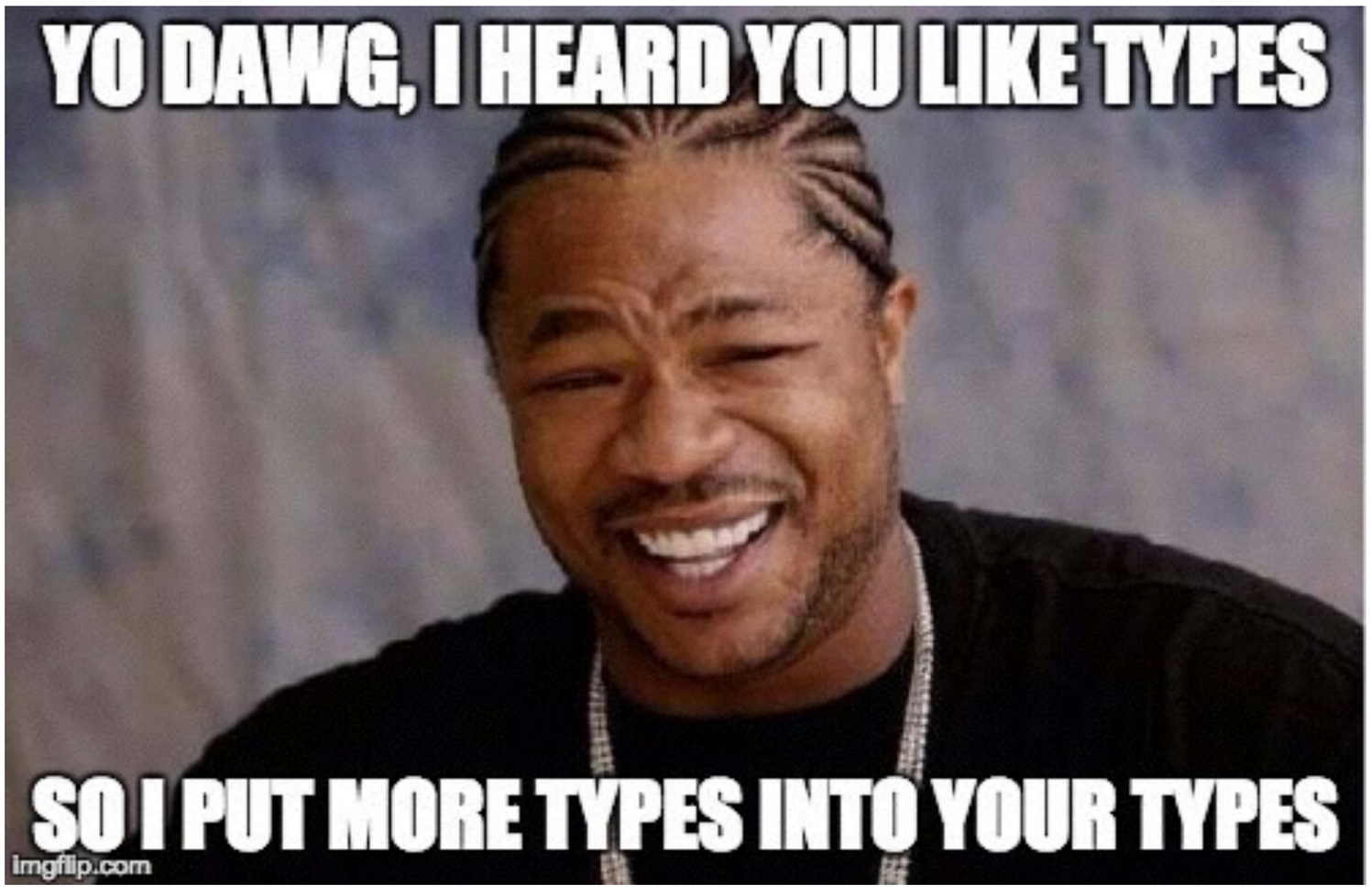
- more boilerplate
- components don’t compose



DIPL. PHYS. PEER STRITZINGER GMBH

# Why Purescript after Elm?

Exhibit 1: the type system is a great feature of Elm




Purescript's has more features. (Simplicity vs Power)

# Why Purescript after Elm?

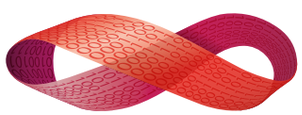
- once you get restless with Elm's boilerplate, you're likely ready for more powerful abstractions

- it's similar enough that porting code is relatively straightforward

- it's possible to implement Elm in it, but not the other way around

- it benefits from the hindsight of following Haskell from a distance

- Small, open community, communication still works



DIPL. PHYS. PEER STRITZINGER GMBH

# Pros compared to Elm

Pursuit (search libs by type signature)

Clearer direction

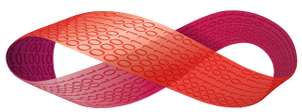
Can work a lot with REPL

Great workflow, including Type holes!!

All (well, many) of the cool abstractions

# Cons

Takes time to learn the cool abstractions



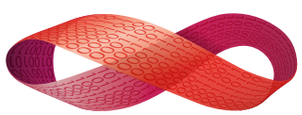
DIPL. PHYS. PEER STRITZINGER GMBH

# Reflection on
# Elm - Purescript - Haskell

- Simplest
- Focused on UX
- One way to do things
- Removes all historical baggage
- Great entry level language
- only targets web browsers

- Most sensible
- UX is fairly good
- Still a lot of power
- Eagerly evaluated, hence simpler
- many backends (C++, Erlang, Js)

- Research language
- Most powerful
- Least good UX
- Most historical baggage
- Laziness adds complexity
- Compiles to native code, llvm, C, etc



DIPL. PHYS. PEER STRITZINGER GMBH

# Higher Abstractions in Elixir

Algae

Bootstrapped
algebraic data types
for Elixir

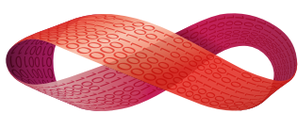
witchcraft

`Witchcraft` is a library providing common algebraic and categorical abstractions to Elixir. (Monoids, functors, monads, arrows, and categories)

# Get Started

At the moment Purescript is relying on **bower**, which makes the time after a new release particularly annoying

But Purescript's community is working on a new package manager: **psc-package**



DIPL. PHYS. PEER STRITZINGER GMBH

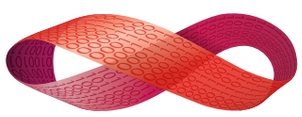
# Frameworks Overview

**Wrapping** 


- Pux
- Thermite
- purescript-react

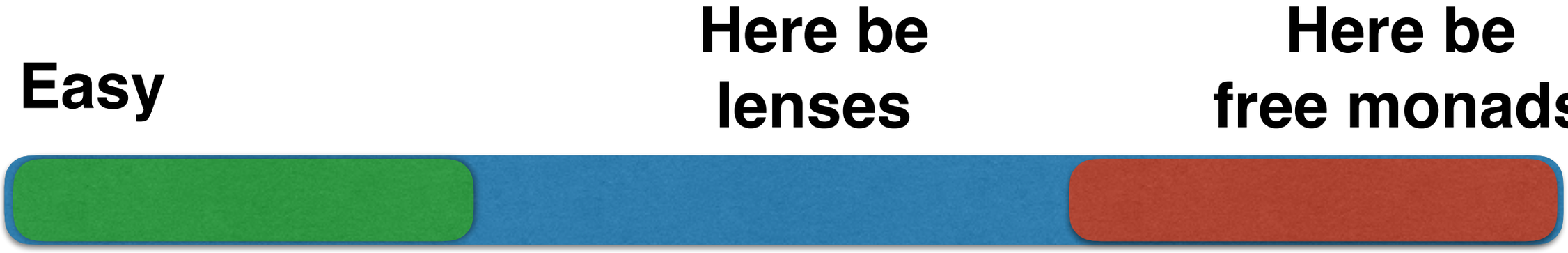
**Pure** 

- Halogen
- Flare
- Optic UI



# Frameworks

Type Complexity continuum




Flare

Pux

Thermite

Optic UI

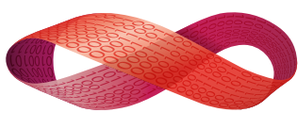
Halogen

# Why Flare?

- Great to start with
- Easy to make cool interactive graphs


# Why not?

- Limited to a specific use case
- Need to understand applicative functor syntax:
  `thing <$> thing <*> thing`



DIPL. PHYS. PEER STRITZINGER GMBH

# Why Pux?

Similar to the Elm architecture

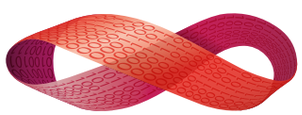
Svg support already included

Interactive React debugger can be wired in

Probably the simplest Purescript framework

**Why not?**

React dependencies /0\



DIPL. PHYS. PEER STRITZINGER GMBH

# On the pain of installing React

(Though the React interactive debugger is nice)

DIPL. PHYS. PEER STRITZINGER GMBH

# Pux Structure

State

Action

update
view

inputs
Aff

Compare with the Elm Architecture (0.16)

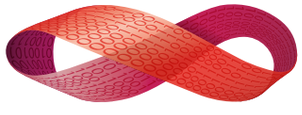
Model

Action

update
view

inputs
Effects

They are so similar, that…

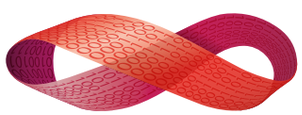


DIPL. PHYS. PEER STRITZINGER GMBH

# #1

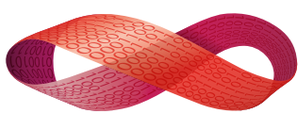
```
data Action = Increment | Decrement

type State = Int

update :: Action -> State -> State
update Increment state = state + 1
update Decrement state = state - 1
```

```
view :: State -> Html Action
view state =
  div []
      [ button [ onClick (const Increment) ]
               [ text "Increment" ]
      , span [] [ text (show state) ]
      , button [ onClick (const Decrement) ]
               [ text "Decrement" ]
      ]
```

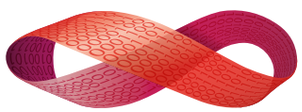


# #2

```
type alias Model = Int

type Action = Increment | Decrement

update : Action -> Model -> Model
update action model =
  case action of
    Increment -> model + 1
    Decrement -> model - 1

view : Signal.Address Action -> Model -> Html
view address model =
  div []
    [ button [ onClick address Decrement ] [ text "-" ]
    , div [ countStyle ] [ text (toString model) ]
    , button [ onClick address Increment ] [ text "+" ]
    ]
```



# Which one was Elm?



# it was…

# #2

# Thermite

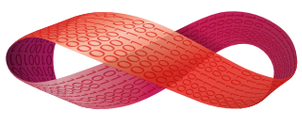
Wraps React

Lenses and stuff

# Optic UI

Pure Purescript

Lenses and stuff



DIPL. PHYS. PEER STRITZINGER GMBH

# Why Halogen?

I’d rather not have to install the 300 React tools

It’s used in production by Slamdata, on a pretty impressive app

> 1 people developing it

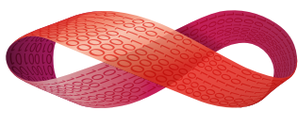
Nice Html DSL

v1.0.0 has arrived!

# Why not?

Argh, the types!! My eyes burn!

aka it’s just a bit hard



DIPL. PHYS. PEER STRITZINGER GMBH

# Halogen Structure

State

Query
action
request

Component

eval
render

main
HalogenEffects

Compare with StartApp (0.16)

Model

Action

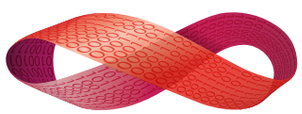
update
view

inputs
Effects

# Rendering

**halogen html** has great syntax too:

```
div [class_ (ClassName “somecssclass”)]
    [ p [] [text “a very well written paragraph”]
    , p [] [text “and another one”]
    ]
```



DIPL. PHYS. PEER STRITZINGER GMBH

# Halogen Structure

State

Query action request

Component

```
-- | The state of the component
type State = Boolean

-- | The query algebra for the component
data Query a
  = ToggleState a
  | IsOn (Boolean -> a)

data Message = Toggled Boolean

type Input = Unit

-- | The component definition
myButton :: forall m. H.Component HH.HTML Query Input Message m
myButton =
  H.component
    { initialState: const initialState
    , render
    , eval
    , receiver: const Nothing
    }
  where
```

DIPL. PHYS. PEER STRITZINGER GMBH

# Halogen Structure

render

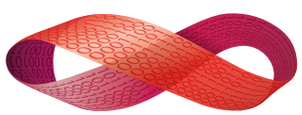
```
initialState :: State
initialState = false

render :: State -> H.ComponentHTML Query
render state =
  let
    label = if state then "On" else "Off"
  in
    HH.button
      [ HP.title label
      , HE.onClick (HE.input_ Toggle)
      ]
      [ HH.text label ]
```

DIPL. PHYS. PEER STRITZINGER GMBH

# Halogen Structure

**eval**

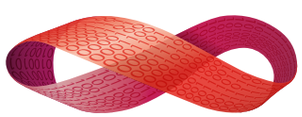
```
eval :: Query ~>
          H.ComponentDSL State Query Message m
eval = case _ of
  Toggle next -> do
    state <- H.get
    let nextState = not state
    H.put nextState
    H.raise $ Toggled nextState
    pure next
IsOn reply -> do
  state <- H.get
  pure (reply state)
```



DIPL. PHYS. PEER STRITZINGER GMBH

# Steps to get started with Purescript

1. get it from **npm** or **psvm**
2. start reading **Purescript by Example**
3. read **purescript-compat-elm**
4. try out **Pux** or **Flare**
5. come on **#purescript** on freenode
6. come to the video meetup
7. try out **Halogen**
8. ???
9. PROFIT!



DIPL. PHYS. PEER STRITZINGER GMBH

# Purescript Conclusion

Powerful

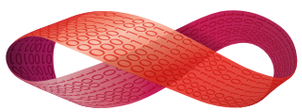
Sensible

With all your favourite abstractions, and more

It will take time to learn, but similar enough to Elm to get a headstart

But you don't have to know **everything** to start (with Pux)

It's not obsessed about language UX, but it's still good



DIPL. PHYS. PEER STRITZINGER GMBH

# tl;dr

Elm works fine with Erlang

If Elm compiles, it works (mostly)
boilerplate can get repetitive
never expect fancy types
Haskell syntax (with less squiggles)
there is no roadmap
Great entry level language into Haskell

Purescript works fine with Erlang
(it even compiles to it)
If Purescript compiles, it works (mostly)
types can get complicated
expect a longish learning curve
Haskell syntax, in all its squiggly glory
the roadmap is a thing
Great second step in your road to Haskell
maybe use Pux to start with

DIPL. PHYS. PEER STRITZINGER GMBH

www.stritzinger.com


@doppioslash

DIPL. PHYS. PEER STRITZINGER GMBH

**Win One of 3 Boards by subscribing to the Newsletter during the conference until May 7th**

GRiSP

**www.grisp.org**

DIPL. PHYS.
PEER STRITZINGER GMBH