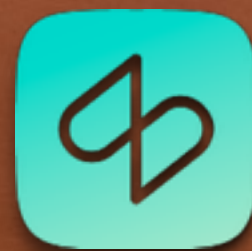
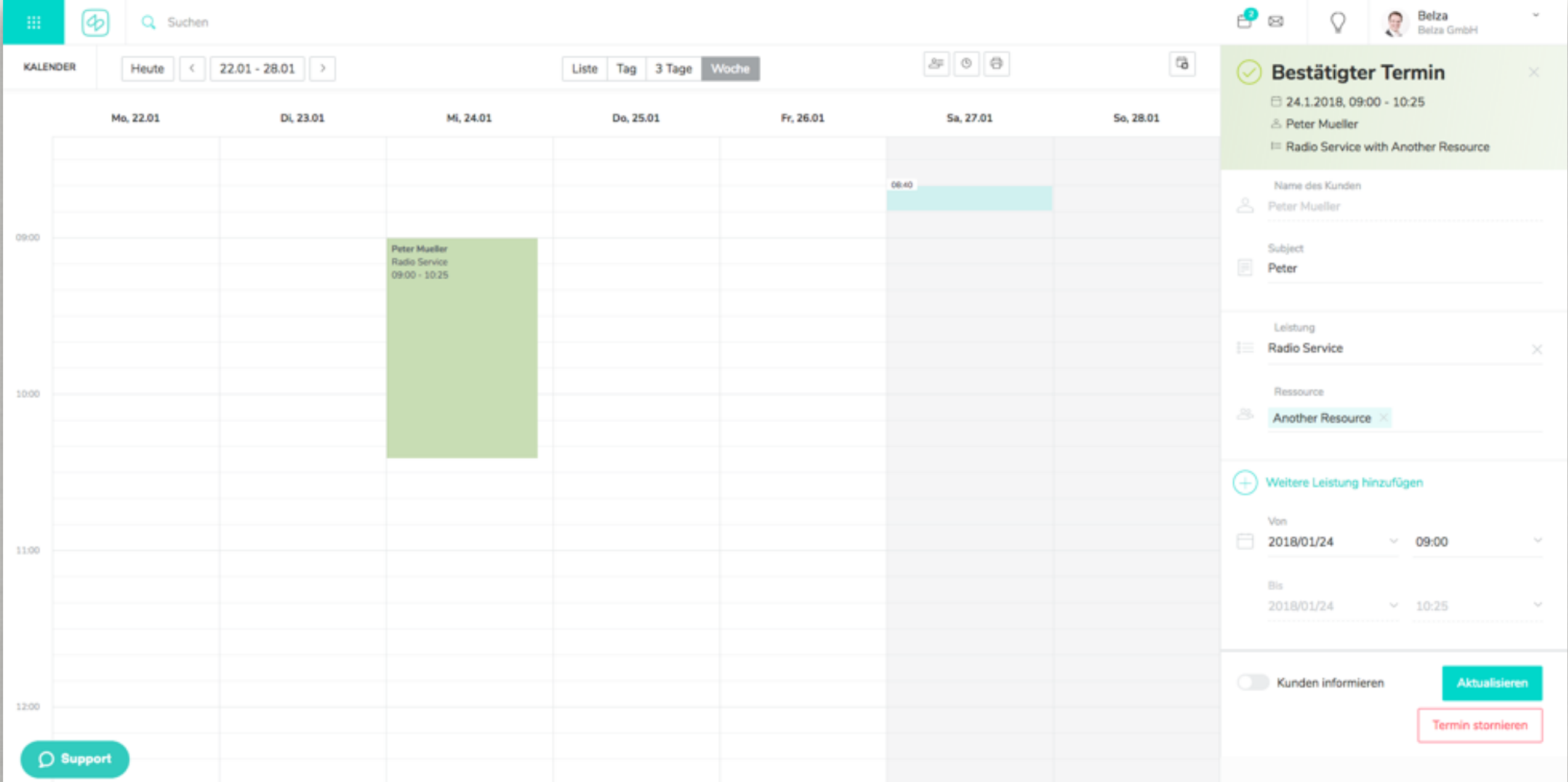


# FROM REACT TO ELM



shore

- Chris, Frontend Dev @  **shore**
- Our product helps SMBs digitize their business.



The screenshot displays the 'shore' calendar application interface. The top navigation bar includes a search icon and the text 'Suchen'. Below this, the 'KALENDER' tab is active, showing a weekly view for the period '22.01 - 28.01'. The calendar grid shows a green appointment for 'Peter Mueller Radio Service' on Wednesday, January 24th, from 09:00 to 10:25. A sidebar on the right provides details for the selected appointment, titled 'Bestätigter Termin'. The sidebar includes fields for 'Name des Kunden' (Peter Mueller), 'Subject' (Peter), and 'Leistung' (Radio Service). It also shows the 'Ressource' as 'Another Resource'. At the bottom of the sidebar, there are options to 'Kunden informieren' (toggle off) and buttons for 'Aktualisieren' and 'Termin stornieren'. A 'Support' button is located in the bottom left corner.

**KALENDER** Heute < 22.01 - 28.01 > Liste Tag 3 Tage Woche

Mo, 22.01 Di, 23.01 Mi, 24.01 Do, 25.01 Fr, 26.01 Sa, 27.01 So, 28.01

09:00

Peter Mueller  
Radio Service  
09:00 - 10:25

10:00

11:00

12:00

**Bestätigter Termin** ✕

24.1.2018, 09:00 - 10:25  
Peter Mueller  
Radio Service with Another Resource

Name des Kunden  
Peter Mueller

Subject  
Peter

Leistung  
Radio Service ✕

Ressource  
Another Resource ✕

+ Weitere Leistung hinzufügen

Von  
2018/01/24 09:00

Bis  
2018/01/24 10:25

☐ Kunden informieren **Aktualisieren**

**Termin stornieren**

**Support**



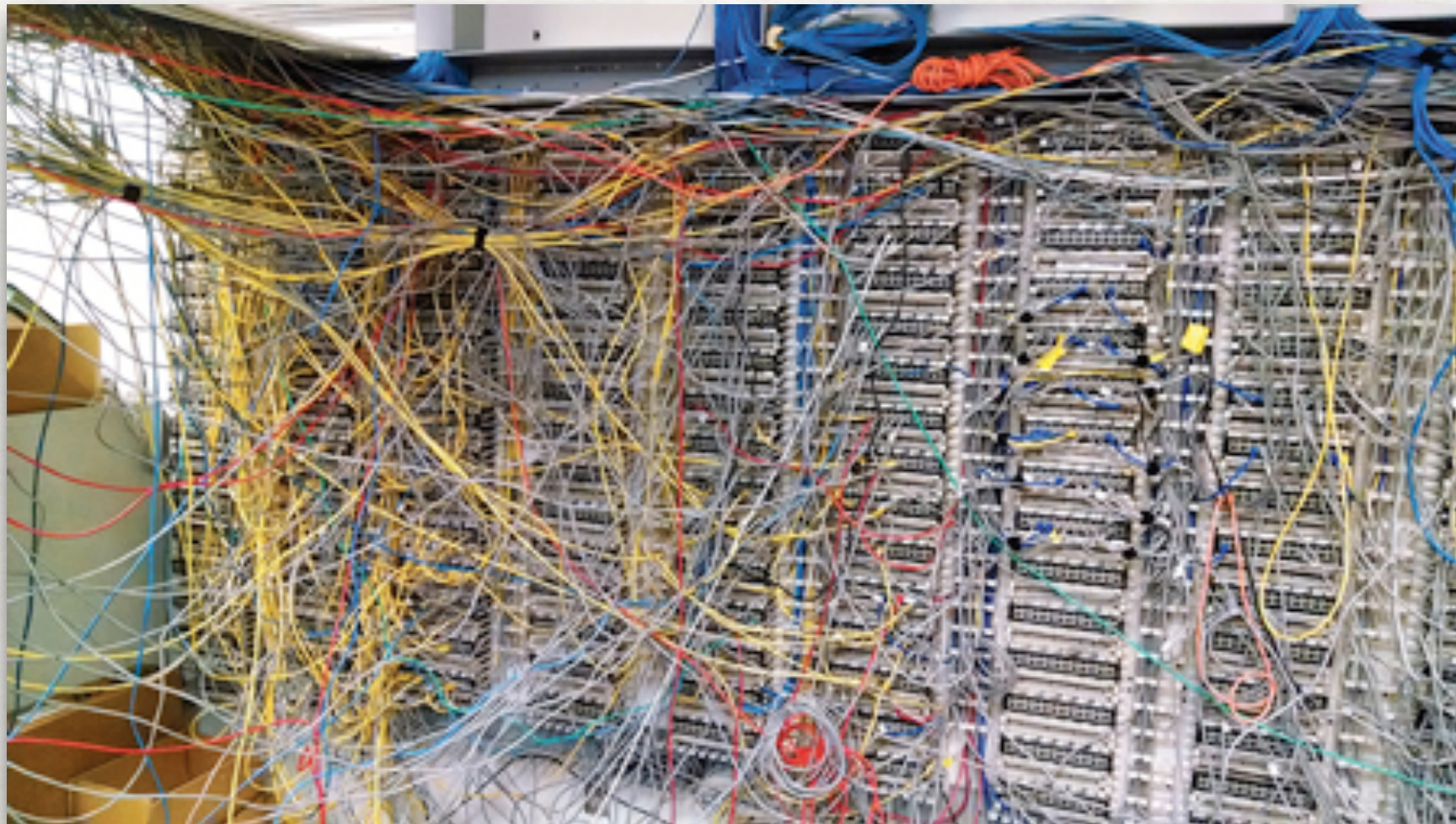
# ELM AT SHORE





# ELM AT SHORE

- Many bugs
- Often changing requirements -> patched up code







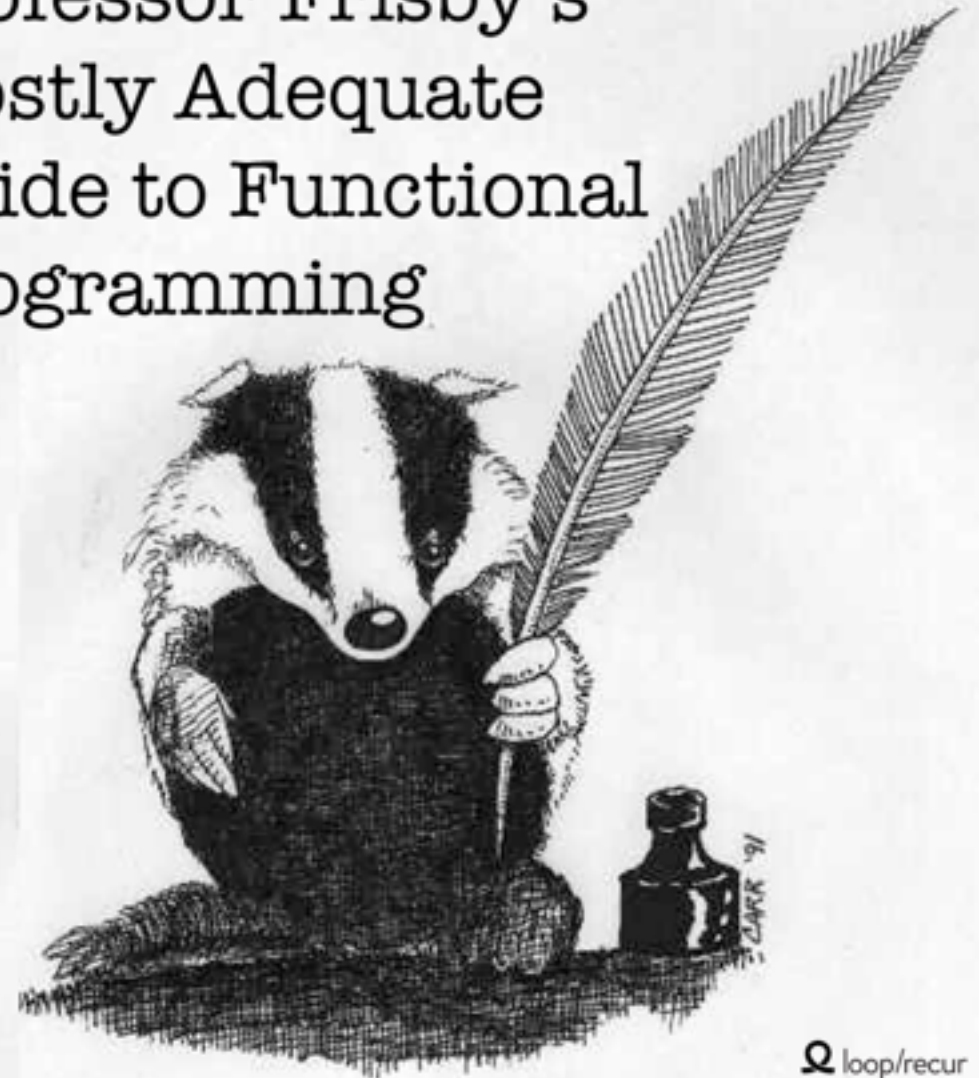
“Elm has no runtime errors”

- Talk at reactive conf in Bratislava
- WTF ??? :-) Can't be true

# TRYING IT

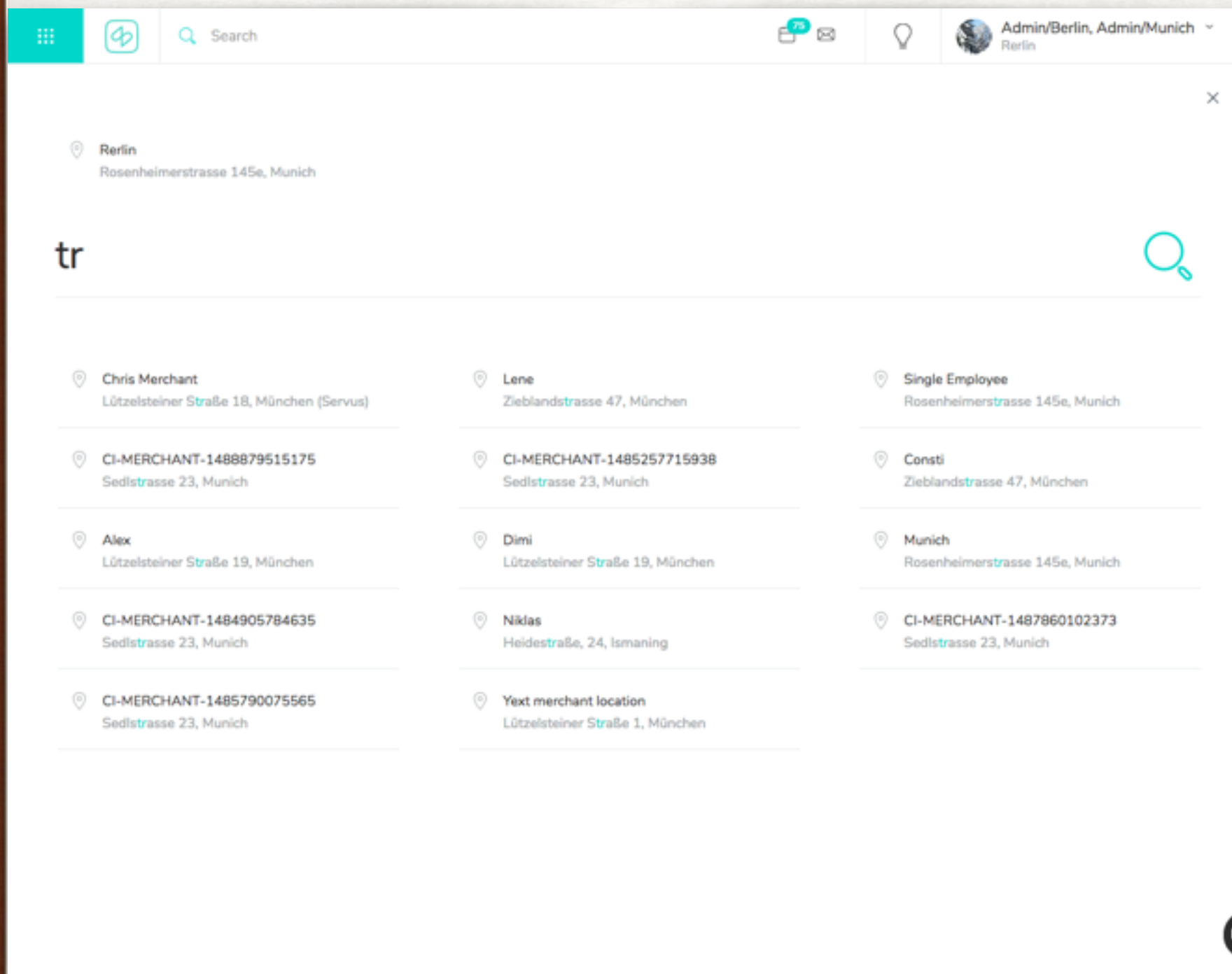
- 5 months went by until I tried it out
- I was amazed
- "How are not more people doing this"
- Makes many problems impossible that you are dealing with in JS every day.

Professor Frisby's  
Mostly Adequate  
Guide to Functional  
Programming





# FIRST PROJECT



- Worked out great
- Felt awesome
- Multiple refactoring very easy
- Forces good architecture
- 1/2 half bundle size

# MORE PROJECTS

- Initially a coworker and I started
- Other people got interested => they need to learn
- Integrating with more projects, legacy API Client we used ports

```
41 update : Msg -> Model -> ( Model, Cmd Msg )
42 update msg model =
43   case msg of
44     FeedbackLoaded (Ok { feedbacks, totalPages }) ->
45       ( { model
46         | feedbacks = model.feedbacks ++ feedbacks
47         , feedbackTotalPages = totalPages
48       }
49       , Cmd.none
50     )
51
52     FeedbackLoaded (Err error) ->
53       ( model, Cmd.none )
54
55     RowClick id ->
56       ( model, sendToJs (Ports.RowClick id) )
```

```
36 let app = Elm.Main.embed(node, [translations, companyBaseUrl]);
37
38 const handlePortMsg = ({ action, payload }) => {
39   switch (action) {
40     case 'RowClick':
41       onAction({
42         type: 'NAVIGATE_TO',
43         payload: { url: `#/booking/appointments/${payload}` },
44         meta: { proxy: true },
45       });
46       break;
47     default:
48       console.error(`Received unknown action type '${action}' from Elm.`);
49   }
50 };
51
52 app.ports.msgForJs.subscribe(handlePortMsg);
```



# IMPRESSIONS

- Puts your mind at ease
- Painless and confident refactoring
- Easy for beginners
- Great tools, no fighting with details
  - format
  - safe package manager
  - webpack integration



# MAIN COMPLAINTS

"Not backed by Facebook"

"I wanna use NPM"

"FP is just a phase"

"I need to learn this"

"Coding React is faster"

"It won't let me do ..."



# REAL WORLD PROBLEMS & SOLUTIONS

- Translations: I18Next
- Localization: Web Components
- Everything else: 5-10% Ports
- Unit test (fewer than with JS)
- Elm CSS (validity guarantee)



ChristophP / elm-i18next / 2.0.1

## Elm i18next - Load and use JSON translations files at runtime

Functions for working with dynamically loaded translations in Elm. PRs and suggestions welcome.

### Simple Example

```
elm package install ChristophP/elm-i18next
```

Then use the module in your app like this.

```
import Http
import Html exposing (Html)
import I18Next exposing
    ( t
    , tr
    , Translations
    , Delims(..)
    , initialTranslations
    , fetchTranslations
    )
```



# LOOKING BACK AFTER 9 MONTHS

- Still no runtime errors from Elm code
- 60 % of FE code is in Elm
- Much more maintainable code (even BE interested)
- Require less up front knowledge (guarantees)
- More interest in FP, Haskell, Elixir etc.
- Started "Elm |> Munich Meetup"



# THE BROADER PICTURE

- FP is awesome, plenty of other compile-to-JS solution
  - PureScript
  - GHCjs
  - ClojureScript
  - ReasonML
- It's about expressing the problem as easy as possible. Separating data and logic helps growth and composition
- Build stuff fast vs. stay flexible



# DIFFERENCE





# TAKE AWAYS

- Much Less Bugs
- Mostly Happier Devs
- Hiring Plus
- Code grows with the requirements
- Outlook on Elm 0.19
  - easier for SPAs (REALLY small bundlesizes, SSR, easy upgrade)

- Chris, Frontend Dev @ Shore
- Github: ChristophP
- [elmlang.slack.com](https://elmlang.slack.com): christophp

# Elm |> Munich

Next Wednesday, 24. January





**BACKUP**

# REACT TO ELM

Elm	React/Redux
Model	State
update	Reducer
Msg	Actions
Cmd	Sagas/Redux Loop/Redux Promises
view	render



# PAIN POINTS

- Translations -> I18next
- Localization
- Dates and local time
- Learning curve
- Slow release cycle sometimes frustrating
- Strictness but for a good reason

# TESTING

```
all : Test
all =
  describe "GitHub Response Decoder"
    [ test "it results in an Err for invalid JSON" <|
      \() ->
        let
          json =
            ""{ "pizza": [] }""

          isErrorResult result =
            False

        in
          json
            |> decodeString responseDecoder
            |> isErrorResult
            |> Expect.true "Expected decoding an invalid response to return an Err."
    , test "it successfully decodes a valid response" <|
      \() ->
        ""{ "items": [
          /* TODO: put JSON here! */
        ] }""
        |> decodeString responseDecoder
        |> Expect.equal
          (Ok
            [ { id = 5, name = "foo", stars = 42 }
              , { id = 3, name = "bar", stars = 77 }
            ]
          )
    ]
  ]
```