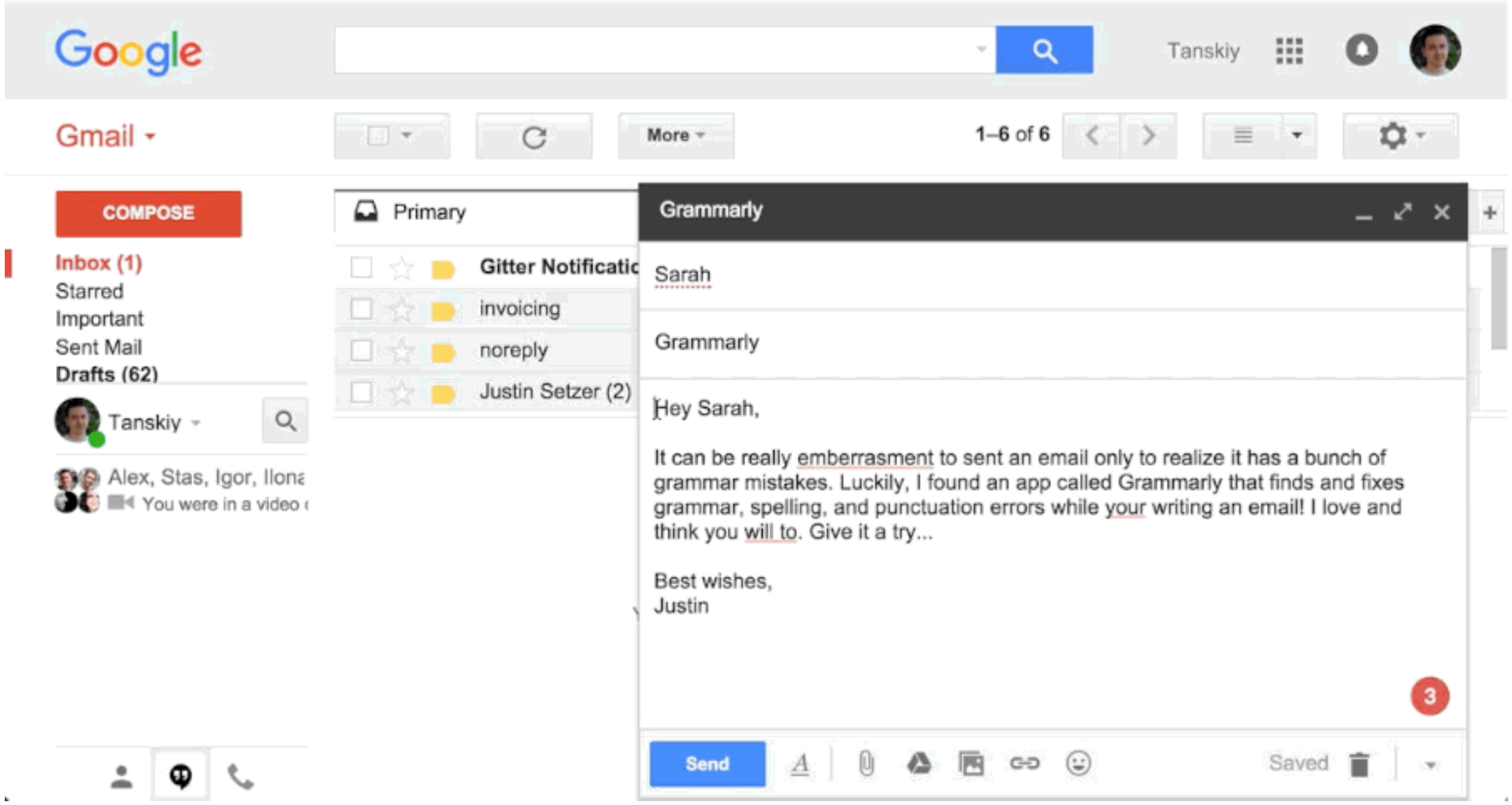
Shehet Gregory

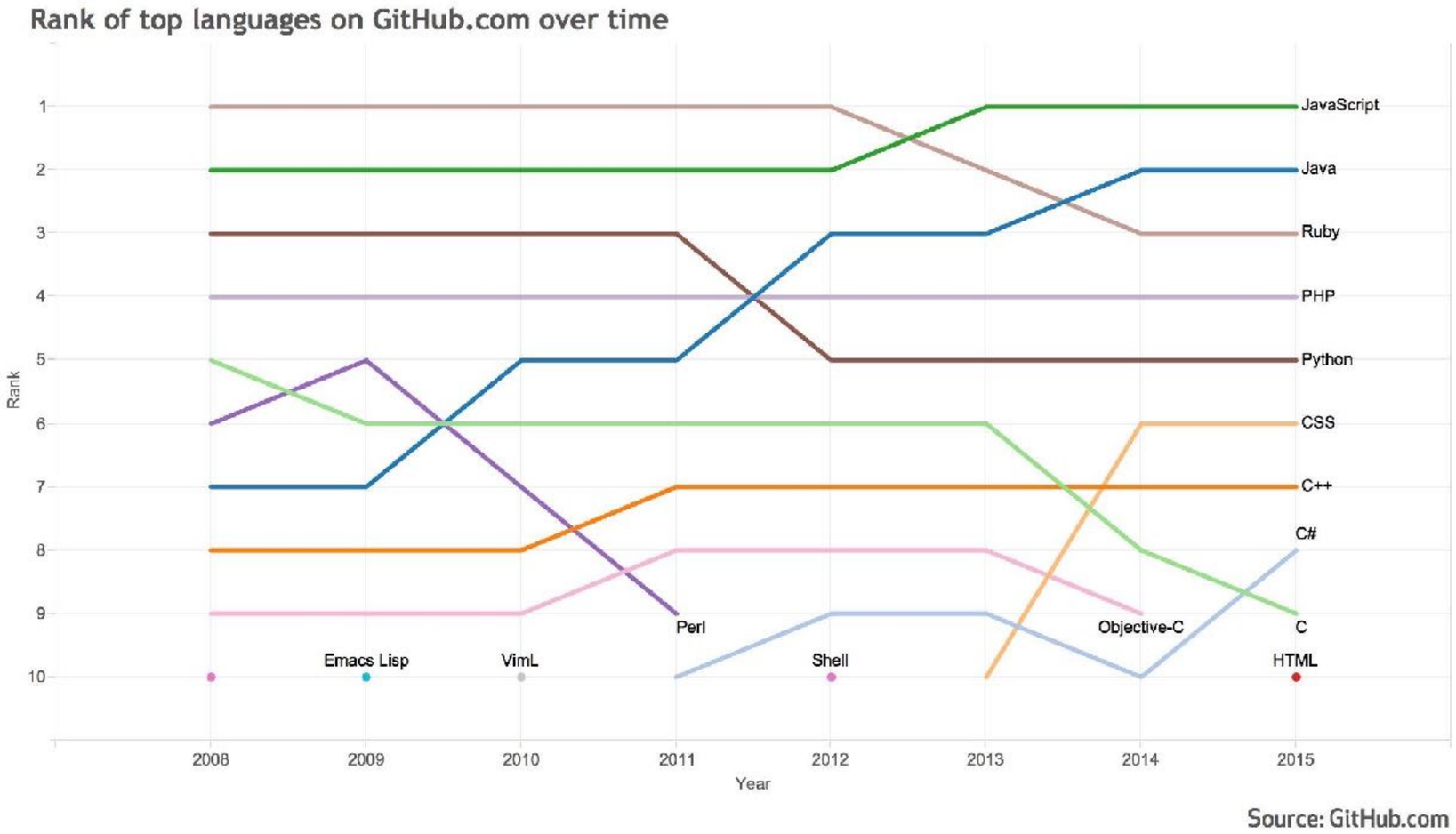
- Software Engineer at 
- Front-End Digest on DOU

Enjoy your code with ELM

Grammarly


Google
Tanskiy
Gmail
More
1–6 of 6
COMPOSE
Inbox (1)
Starred
Important
Sent Mail
Drafts (62)
Tanskiy
Alex, Stas, Igor, Ilona
You were in a video
Primary
Gitter Notificatic
invoicing
noreply
Justin Setzer (2)
Grammarly
Sarah
Grammarly
Hey Sarah,
It can be really emberrasment to sent an email only to realize it has a bunch of grammar mistakes. Luckily, I found an app called Grammarly that finds and fixes grammar, spelling, and punctuation errors while your writing an email! I love and think you will to. Give it a try...
Best wishes,
Justin
3
Send
Saved

Popular


Rank of top languages on GitHub.com over time
Rank
1
2
3
4
5
6
7
8
9
10
2008
2009
2010
2011
2012
2013
2014
2015
Year
JavaScript
Java
Ruby
PHP
Python
CSS
C++
C#
C
HTML
Objective-C
Shell
Perl
VimL
Emacs Lisp
Source: GitHub.com

Write For Everywhere

Write For Everywhere

Write For Everywhere

ELECTRON

Write For Everywhere

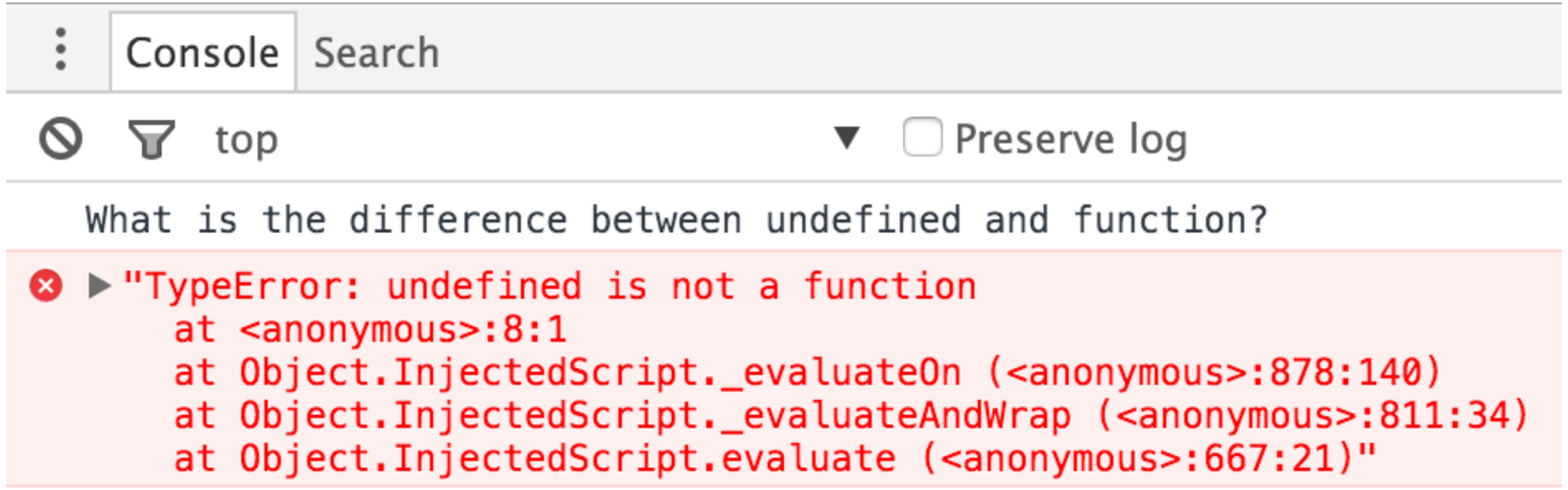
Applause to JS community

But...

History

15 days

Undefined


Console
Search
top
Preserve log
What is the difference between undefined and function?
"TypeError: undefined is not a function
at <anonymous>:8:1
at Object.InjectedScript._evaluateOn (<anonymous>:878:140)
at Object.InjectedScript._evaluateAndWrap (<anonymous>:811:34)
at Object.InjectedScript.evaluate (<anonymous>:667:21)"

Some JS function

```
function foo(a, b) {
  if (a > b) {
    return [a, b]
  }
}

foo(1, 2).join(',')
```

Some JS function

```
function foo(a, b) {
  if (a > b) {
    return [a, b]
  }
}

foo(1, 2).join(',')
```

Who it works?

```
> function foo(a, b){
    if (a > b) {
      return [a, b]
    }
  }
< undefined
> foo(1, 2).join(',')
✕ ▶ Uncaught TypeError: Cannot read property 'join' of undefined(…)    VM145:1
```


flow

Use Flow

```
code.js:9
  9: foo(1, 2).join(',')
     ^^^^^^^^^^^^^^^^^^^^ call of method `join`. Method
cannot be called on possibly undefined value
  9: foo(1, 2).join(',')
     ^^^^^^^^^ undefined


Found 1 error
```

Add to Git Hooks

Git
Hooks

Flow Hack

```
function foo(a, b) {
  if (a > b) {
    return [a, b]
  }
}

foo(1, 2).join(',')
```

Flow Hack

```
function foo(a, b):any {
  if (a > b) {
    return [a, b]
  }
}

foo(1, 2).join(',')
```

Reality

```
Found 0 errors
> foo(1, 2).join(',')
✕ ▶ Uncaught TypeError: Cannot read property 'join' of undefined(…)    VM145:1
```

Evan Czaplicki

@evancz

ELM

Let’s rewrite to ELM

Some function on ELM

```
foo : Int -> Int -> List Int
foo a b =
  if a > b then a :: b :: []

main =
  text (String.join "," (foo 1 2))
```

Some function on ELM

```
foo : Int -> Int -> List Int
foo a b =
  if a > b then a :: b :: []

main =
  text (String.join "," (foo 1 2))
```

Some function on ELM

```
foo : Int -> Int -> List Int
foo a b =
  if a > b then a :: b :: []

main =
  text (String.join "," (foo 1 2))
```

Some function on ELM

```
foo : Int -> Int -> List Int
foo a b =
  if a > b then a :: b :: []

main =
  text (String.join "," (foo 1 2))
```

Elm Compilation

```
Detected errors in 1 module.


-- SYNTAX PROBLEM -------------------------------------------------------------

I need whitespace, but got stuck on what looks like a new declaration. You are
either missing some stuff in the declaration above or just need to add some
spaces here:

6| main =
   ^

I am looking for one of the following things:

    whitespace
```

So why ELM?

Not Haskell or PureScript

Benefits?

- Pure functions -> Native

Pure Function

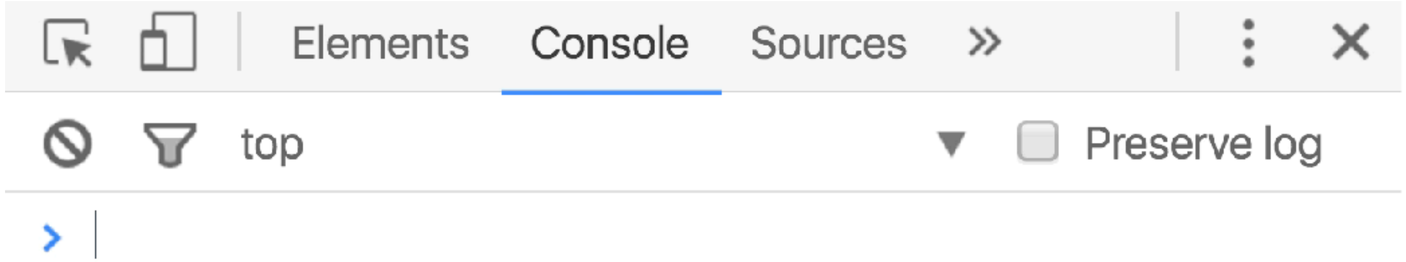
```
function firstFoo(x) {
  return x + x
}

function secondFoo(x) {
  calc()
  return x + x
  ...
}
```

Pure Function

```
function firstFoo(x) {
  return x + x
}

function secondFoo(x) {
  calc()
  return x + x
  ...
}
```

Pure Function

```
function firstFoo(x) {
  return x + x
}

function secondFoo(x) {
  calc()
  return x + x
  ...
}
```

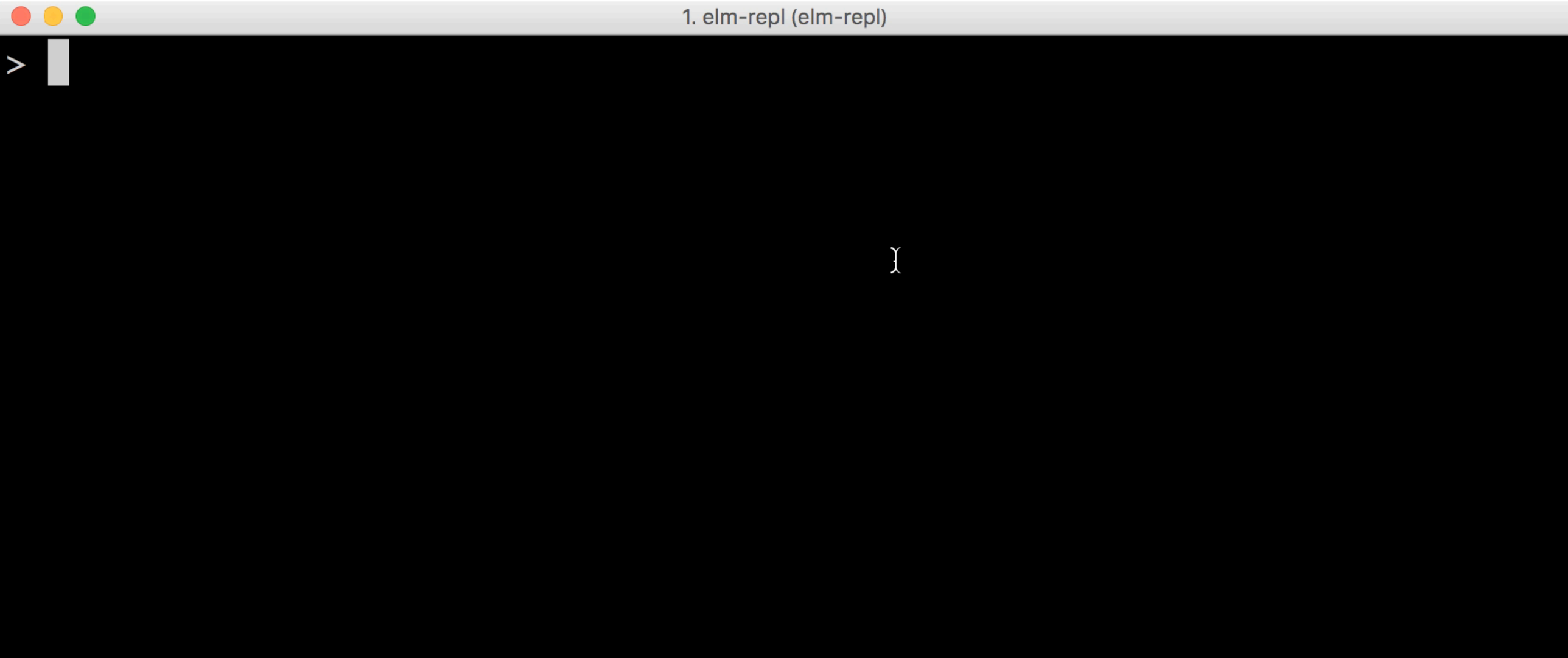


SO?

Benefits?

- Pure functions -> Native
- Immutable data structures -> Immutable.js

Stateless



Benefits?

- Pure functions -> Native
- Immutable data structures -> Immutable.js
- Static type checking -> Flow (Not JavaScript)

Static type checking

```
f : Int -> Int
f v =
  v + 3
```

Static type checking

```
f : Int -> String
f v =
  v + 3
```

Static type checking

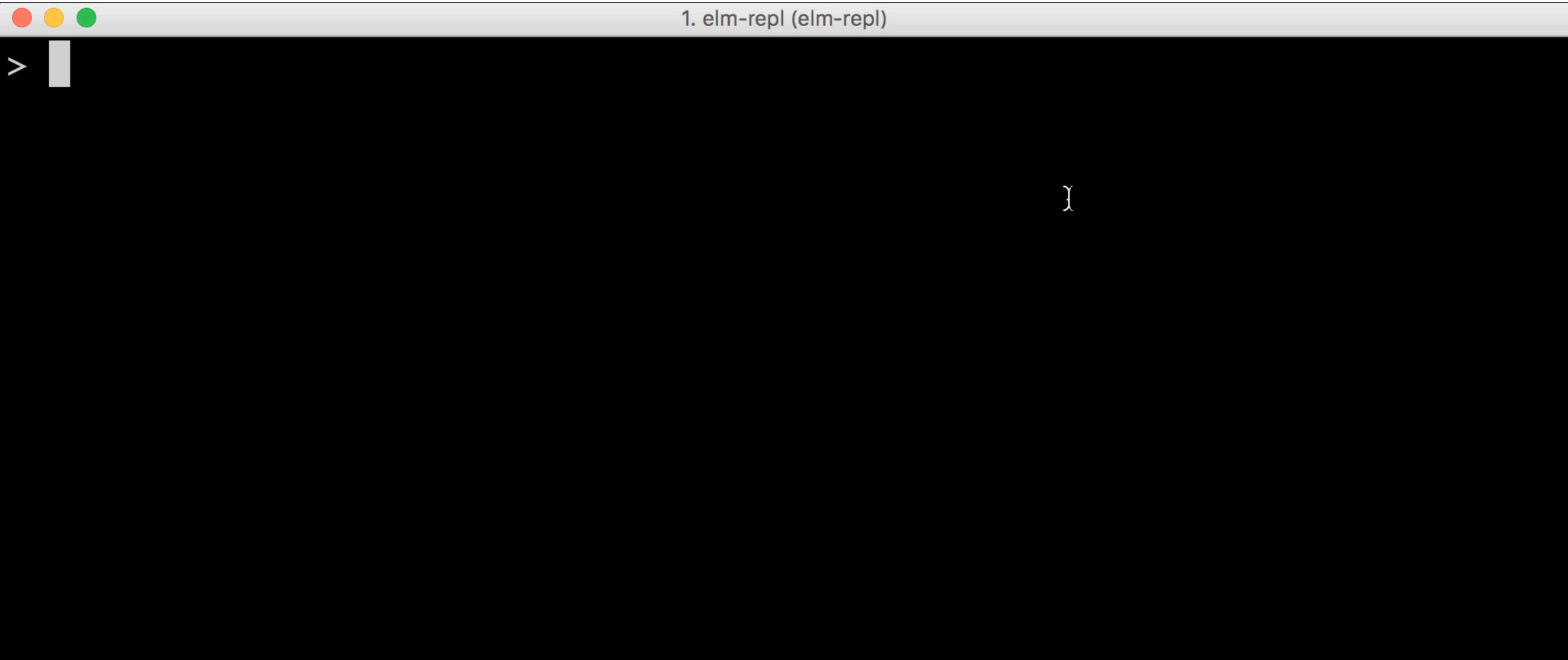
```
f : Int -> String
f v =
  v + 3
```

```
-- TYPE MISMATCH ---------------------------------------------------------

The type annotation for `f` does not match its definition.

8| f : Int -> String
       ^^^^^^^^^^^^^

The type annotation is saying:

    Int -> String

But I am inferring that the definition has this type:

    Int -> Int
```

Benefits?

- Pure functions -> Native
- Immutable data structures -> Immutable.js
- Static type checking -> Flow (Not JavaScript)
- No concept of Null -> folktale/data.maybe

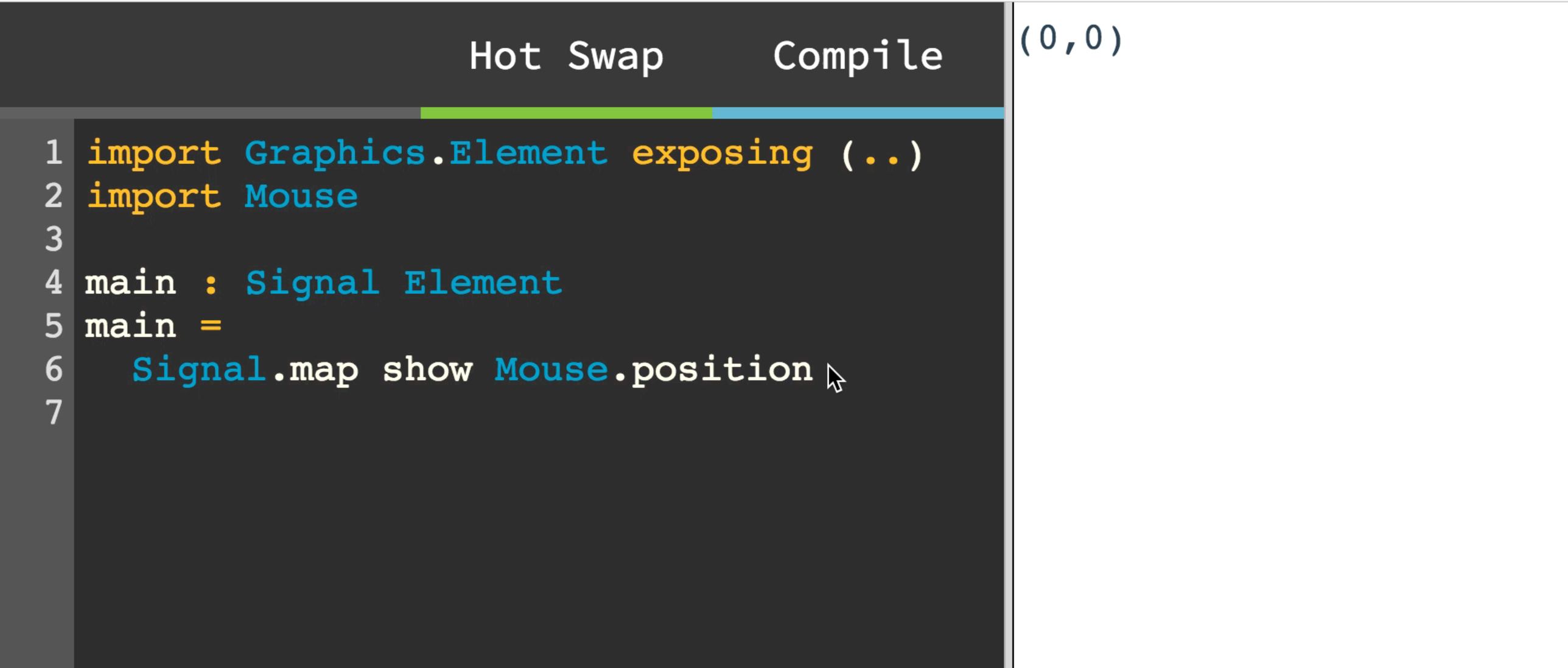
Maybe monad



Benefits?

- Pure functions -> Native
- Immutable data structures -> Immutable.js
- Static type checking -> Flow (Not JavaScript)
- No concept of Null -> folktale/data.maybe
- Reactivity -> Rx.js

Reactive


Hot Swap
Compile
1 import Graphics.Element exposing (..)
2 import Mouse
3
4 main : Signal Element
5 main =
6 Signal.map show Mouse.position
7
(0,0)

Benefits?

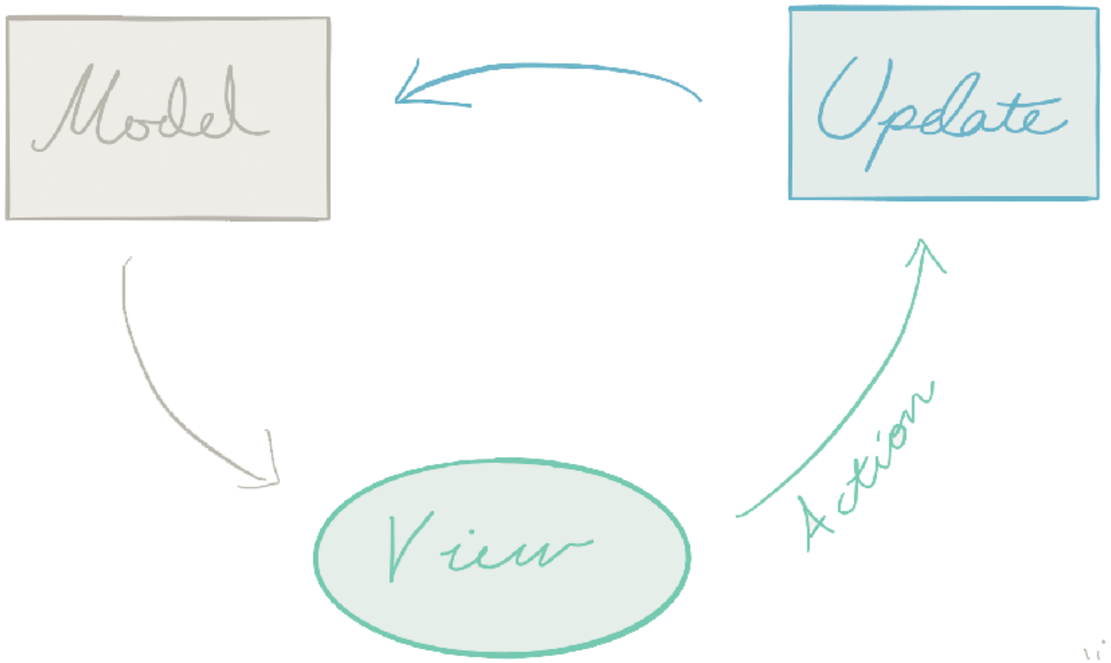
- Pure functions -> Native
- Immutable data structures -> Immutable.js
- Static type checking -> Flow (Not JavaScript)
- No concept of Null -> folktale/data.maybe
- Reactivity -> Rx.js
- The Elm Architecture -> Redux

Dan Abramov

@dan_abramov

Redux

ELM === Redux



Benefits?

- Pure functions -> Native
- Immutable data structures -> Immutable.js
- Static type checking -> Flow (Not JavaScript)
- No concept of Null -> folktale/data.maybe
- Reactivity -> Rx.js
- The Elm Architecture -> Redux
- Declarative UI -> React.js

Virtual DOM

```
view =
  div [ Props ] [ Children ]
```

Example
ELM vs Redux

Counter

Start App

```
// Redux

let initialState = 0
const store = createStore(
  reducer,
  initialState
)

const App () => (
  <Provider store={ store }>
    <View />
  </Provider>
)
```

```
-- ELM

type alias Model = Int

model : Model
model =
 0

main =
  Html.beginnerProgram
    { model = model
    , view = view
    , update = update
    }
```

Start App

```
// Redux

let initialState = 0
const store = createStore(
  reducer,
  initialState
)

const App () => (
  <Provider store={ store }>
    <View />
  </Provider>
)
```

```
-- ELM

type alias Model = Int

model : Model
model =
 0

main =
  Html.beginnerProgram
    { model = model
    , view = view
    , update = update
    }
```

Start App

```
// Redux

let initialState = 0
const store = createStore(
  reducer,
  initialState
)

const App = () => (
  <Provider store={ store }>
    <View />
  </Provider>
)
```

```
-- ELM

type alias Model = Int

model : Model
model =
 0

main =
  Html.beginnerProgram
    { model = model
    , view = view
    , update = update
    }
```

Start App

```
// Redux

let initialState = 0
const store = createStore(
  reducer,
  initialState
)

const App () => (
  <Provider store={ store }>
    <View />
  </Provider>
)
```

```
-- ELM

type alias Model = Int

model : Model
model =
 0

main =
  Html.beginnerProgram
    { model = model
    , view = view
    , update = update
    }
```

Start App

```
// Redux

let initialState = 0
const store = createStore(
  reducer,
  initialState
)

const App () => (
  <Provider store={ store }>
    <View />
  </Provider>
)
```

```
-- ELM

type alias Model = Int

model : Model
model =
 0

main =
  Html.beginnerProgram
    { model = model
    , view = view
    , update = update
    }
```

View

```
// Redux

const View = ({ dispatch, state }) => (
  <div>
    <button onClick={() => dispatch({type: 'DECREMENT'})} value='-' />
    <div>{ state }</div>
    <button onClick={() => dispatch({type: 'INCREMENT'})} value='+' />
  </div>
)

-- ELM

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

View

```
// Redux

const View = ({ dispatch, state }) => (
  <div>
    <button onClick={() => dispatch({type: 'DECREMENT'})} value='-' />
    <div>{ state }</div>
    <button onClick={() => dispatch({type: 'INCREMENT'})} value='+' />
  </div>
)

-- ELM

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

View

```
// Redux

const View = ({ dispatch, state }) => (
  <div>
    <button onClick={() => dispatch({type: 'DECREMENT'})} value='-' />
    <div>{ state }</div>
    <button onClick={() => dispatch({type: 'INCREMENT'})} value='+' />
  </div>
)

-- ELM

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

View

```
// Redux

const View = ({ dispatch, state }) => (
  <div>
    <button onClick={() => dispatch({type: 'DECREMENT'})} value='-' />
    <div>{ state }</div>
    <button onClick={() => dispatch({type: 'INCREMENT'})} value='+' />
  </div>
)

-- ELM

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

View

```
// Redux

const View = ({ dispatch, state }) => (
  <div>
    <button onClick={() => dispatch({type: 'DECREMENT'})} value='-' />
    <div>{ state }</div>
    <button onClick={() => dispatch({type: 'INCREMENT'})} value='+' />
  </div>
)

-- ELM

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

Update

```
// Redux

const reducer = (state = initialState, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return state + 1

    case 'DECREMENT':
      return state - 1

    default:
      return state
  }
}
```

```
-- ELM

type Msg
  = Increment
  | Decrement

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

    Decrement ->
      model - 1
```

Update

```
// Redux

const reducer = (state = initialState, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return state + 1

    case 'DECREMENT':
      return state - 1

    default:
      return state
  }
}
```

```
-- ELM

type Msg
  = Increment
  | Decrement

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

    Decrement ->
      model - 1
```

Update

```
// Redux

const reducer = (state = initialState, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return state + 1

    case 'DECREMENT':
      return state - 1

    default:
      return state
  }
}
```

```
-- ELM

type Msg
  = Increment
  | Decrement

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

    Decrement ->
      model - 1
```

Update

```
// Redux

const reducer = (state = initialState, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return state + 1

    case 'DECREMENT':
      return state - 1

    default:
      return state
  }
}
```

```
-- ELM

type Msg
  = Increment
  | Decrement

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

    Decrement ->
      model - 1
```

Update

```
// Redux

const reducer = (state = initialState, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return state + 1

    case 'DECREMENT':
      return state - 1

    default:
      return state
  }
}
```

```
-- ELM

type Msg
  = Increment
  | Decrement

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

    Decrement ->
      model - 1
```

Update

```
// Redux

const reducer = (state = initialState, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return state + 1

    case 'DECREMENT':
      return state - 1

    default:
      return state
  }
}
```

```
-- ELM

type Msg
  = Increment
  | Decrement

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

    Decrement ->
      model - 1
```

Debugging?


Chrome File Edit View History Bookmarks People Window Help
100%
Tue 5:34 PM
Airplane Mode
everything

So why ELM?

NoRedInk

50,000 lines of ELM code
1.5 year in production

0 Runtime Errors

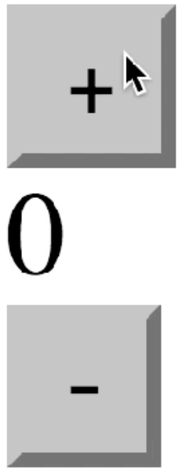
How I can use it?

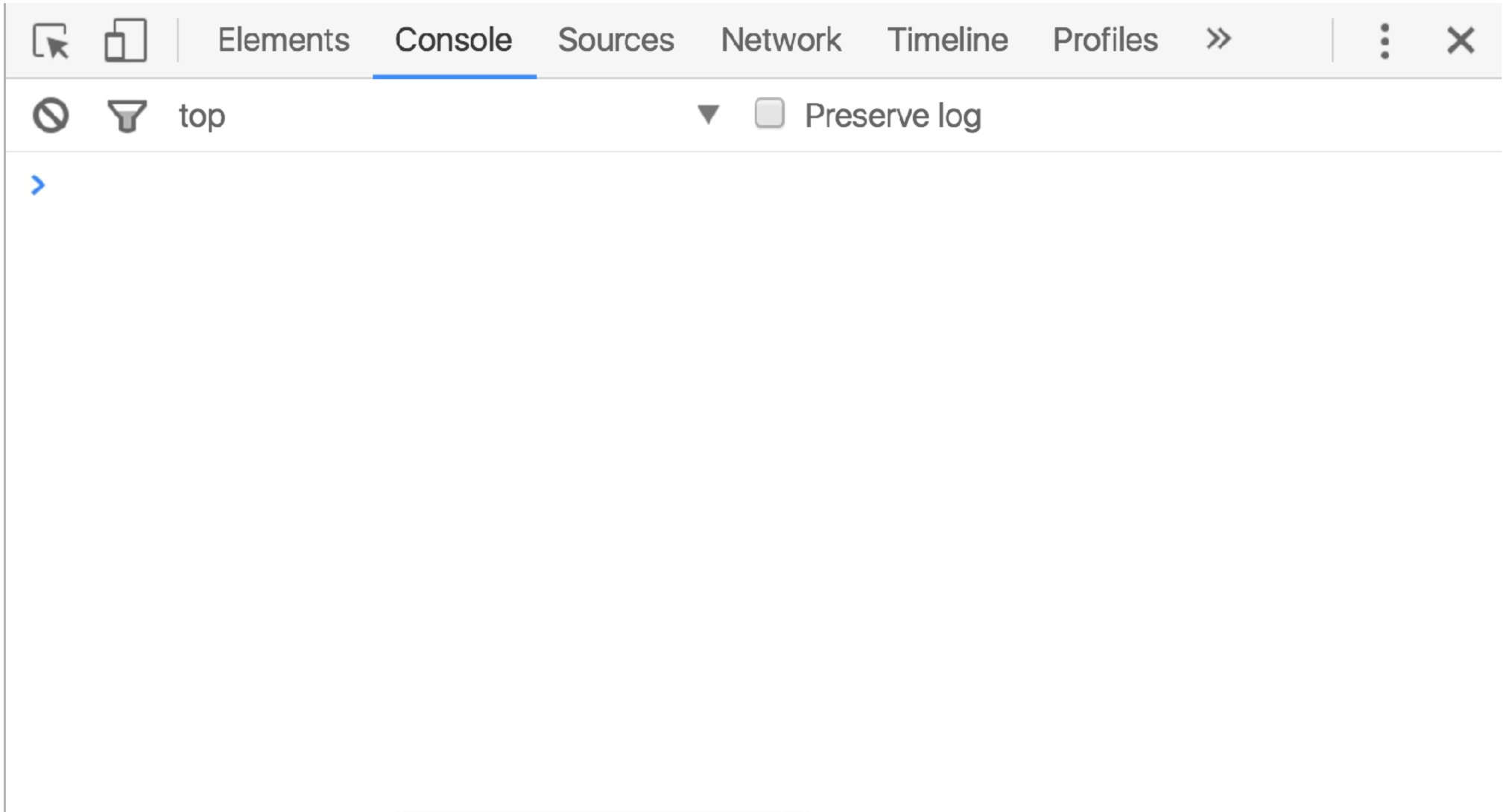
Ports & Subscriptions

HTML

```
<body>
  <input type="submit" value="+" onClick="sendEvent('Increment')">
  <div id="counter"></div>
  <input type="submit" value="-" onClick="sendEvent('Decrement')">
  <script>
   ...
  </script>
</body>
```

Result


+
0
-


Elements
Console
Sources
Network
Timeline
Profiles
top
Preserve log

Init App

```
main =
  Html.beginnerProgram
    { model = model
    , view = view
    , update = update
    }
```

```
main =
  Html.program
    { init = init
    , view = view
    , update = update
    , subscriptions = subscriptions
    }
```

Init App

```
main =
  Html.beginnerProgram
    { model = model
    , view = view
    , update = update
    }
```

```
main =
  Html.program
    { init = init
    , view = view
    , update = update
    , subscriptions = subscriptions
    }
```

Model

```
type alias Model = Int

model : Model
model =
 0
```

```
type alias Model = Int

init : (Model, Cmd Msg)
init =
  ( 0, Cmd.none )
```

Model

```
type alias Model = Int

model : Model
model =
 0
```

```
type alias Model = Int

init : (Model, Cmd Msg)
init =
  ( 0, Cmd.none )
```

Actions

```
type Msg
  = Increment
  | Decrement
```

```
type Msg
    = Update String
```

Ports

```
-- PORTS


port modelChanges : String -> Cmd msg
port modelValue : (String -> msg) -> Sub msg

-- SUBSCRIPTIONS


subscriptions : Model -> Sub Msg
subscriptions model =
  modelValue Update
```

Subscriptions

```
-- PORTS


port modelChanges : String -> Cmd msg
port modelValue : (String -> msg) -> Sub msg

-- SUBSCRIPTIONS


subscriptions : Model -> Sub Msg
subscriptions model =
  modelValue Update
```

Update

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

    Decrement ->
      model - 1
```

```elm
update : Msg -> Model -> (Model, Cmd Msg)
update msg model =
  case msg of

    Update action ->
      case action of
        "Increment" ->
          (model + 1, modelChanges (toString (model + 1)))
        "Decrement" ->
          (model - 1, modelChanges (toString (model - 1)))
        _ ->
          (model, Cmd.none)
```

Update

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

    Decrement ->
      model - 1
```

```
update : Msg -> Model -> (Model, Cmd Msg)
update msg model =
  case msg of

    Update action ->
      case action of
        "Increment" ->
          (model + 1, modelChanges (toString (model + 1)))
        "Decrement" ->
          (model - 1, modelChanges (toString (model - 1)))
        _ ->
         (model, Cmd.none)
```

Update

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

    Decrement ->
      model - 1
```

```
update : Msg -> Model -> (Model, Cmd Msg)
update msg model =
  case msg of

    Update action ->
      case action of
        "Increment" ->
          (model + 1, modelChanges (toString (model + 1)))
        "Decrement" ->
          (model - 1, modelChanges (toString (model - 1)))
        _ ->
          (model, Cmd.none)
```

HTML

```
<body>
  <input type="submit" value="+" onClick="sendEvent('Increment')">
  <div id="counter"></div>
  <input type="submit" value="-" onClick="sendEvent('Decrement')">
  <script>
      var node = document.getElementById('counter');
      var app = Elm.Counter.embed(node);
      app.ports.modelChanges.subscribe(function(model) {
          console.log(model);
      })
      function sendEvent(action) {
          app.ports.modelValue.send(action);
      }
  </script>
</body>
```

HTML

```
<body>
  <input type="submit" value="+" onClick="sendEvent('Increment')">
  <div id="counter"></div>
  <input type="submit" value="-" onClick="sendEvent('Decrement')">
  <script>
      var node = document.getElementById('counter');
      var app = Elm.Counter.embed(node);
      app.ports.modelChanges.subscribe(function(model) {
          console.log(model);
      })
      function sendEvent(action) {
          app.ports.modelValue.send(action);
      }
  </script>
</body>
```

Init Component

```
<body>
  <input type="submit" value="+" onClick="sendEvent('Increment')">
  <div id="counter"></div>
  <input type="submit" value="-" onClick="sendEvent('Decrement')">
  <script>
      var node = document.getElementById('counter');
      var app = Elm.Counter.embed(node);
      app.ports.modelChanges.subscribe(function(model) {
          console.log(model);
      })
      function sendEvent(action) {
          app.ports.modelValue.send(action);
      }
  </script>
</body>
```

Subscribe/Send

```
<body>
  <input type="submit" value="+" onClick="sendEvent('Increment')">
  <div id="counter"></div>
  <input type="submit" value="-" onClick="sendEvent('Decrement')">
  <script>
      var node = document.getElementById('counter');
      var app = Elm.Counter.embed(node);
      app.ports.modelChanges.subscribe(function(model) {
          console.log(model);
      })
      function sendEvent(action) {
          app.ports.modelValue.send(action);
      }
  </script>
</body>
```

What about React?

React Elm Component

Import

```
import Elm from 'react-elm-components'
import { Counter } from './Counter'
```

```
class CounterWrapper extends Component ({
  constructor(props) {
    super(props)
    this.ports = null
  }

  setupPorts = ports => {
    this.ports = ports
    this.ports.modelChanges.subscribe(model => console.log(model))
  }

  sendEvent(action) {
    this.ports.modelValue.send(action)
  }

  render() {
    return (
      <div>
        <input type='submit' value='+' onClick={() => this.sendEvent('Increment')} />
        <Elm src={ Counter } ports={ this.setupPorts } />
        <input type='submit' value='-' onClick={() => this.sendEvent('Decrement')} />
      </div>
    )
  }
})
```

```
class CounterWrapper extends Component ({
  constructor(props) {
    super(props)
    this.ports = null
  }

  setupPorts = ports => {
    this.ports = ports
    this.ports.modelChanges.subscribe(model => console.log(model))
  }

  sendEvent(action) {
    this.ports.modelValue.send(action)
  }

  render() {
    return (
      <div>
        <input type='submit' value='+' onClick={() => this.sendEvent('Increment')} />
        <Elm src={ Counter } ports={ this.setupPorts } />
        <input type='submit' value='-' onClick={() => this.sendEvent('Decrement')} />
      </div>
    )
  }
})
```

```
class CounterWrapper extends Component ({
  constructor(props) {
    super(props)
    this.ports = null
  }

  setupPorts = ports => {
    this.ports = ports
    this.ports.modelChanges.subscribe(model => console.log(model))
  }

  sendEvent(action) {
    this.ports.modelValue.send(action)
  }

  render() {
    return (
      <div>
        <input type='submit' value='+' onClick={() => this.sendEvent('Increment')} />
        <Elm src={ Counter } ports={ this.setupPorts } />
        <input type='submit' value='-' onClick={() => this.sendEvent('Decrement')} />
      </div>
    )
  }
})
```

```
class CounterWrapper extends Component ({
  constructor(props) {
    super(props)
    this.ports = null
  }

  setupPorts = ports => {
    this.ports = ports
    this.ports.modelChanges.subscribe(model => console.log(model))
  }

  sendEvent(action) {
    this.ports.modelValue.send(action)
  }

  render() {
    return (
      <div>
        <input type='submit' value='+' onClick={() => this.sendEvent('Increment')} />
        <Elm src={ Counter } ports={ this.setupPorts } />
        <input type='submit' value='-' onClick={() => this.sendEvent('Decrement')} />
      </div>
    )
  }
})
```

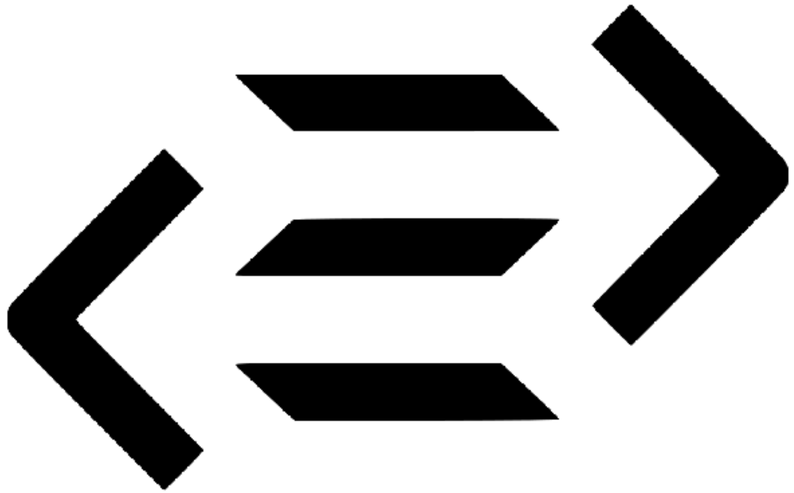
```
class CounterWrapper extends Component ({
  constructor(props) {
    super(props)
    this.ports = null
  }

  setupPorts = ports => {
    this.ports = ports
    this.ports.modelChanges.subscribe(model => console.log(model))
  }

  sendEvent(action) {
    this.ports.modelValue.send(action)
  }

  render() {
    return (
      <div>
        <input type='submit' value='+' onClick={() => this.sendEvent('Increment')} />
        <Elm src={ Counter } ports={ this.setupPorts } />
        <input type='submit' value='-' onClick={() => this.sendEvent('Decrement')} />
      </div>
    )
  }
})
```

So why ELM?

Redux Elm Middleware

https://github.com/stoeffel/redux-elm-middleware

Problem



PureScript

http://www.purescript.org/

What Next?

http://elm-lang.org/
https://www.reddit.com/r/elm
http://elmlang.herokuapp.com/ - Slack
https://medium.com/@diamondgfx - ELM tutorial
https://github.com/isRuslan/awesome-elm

Thank you!
Questions are welcome