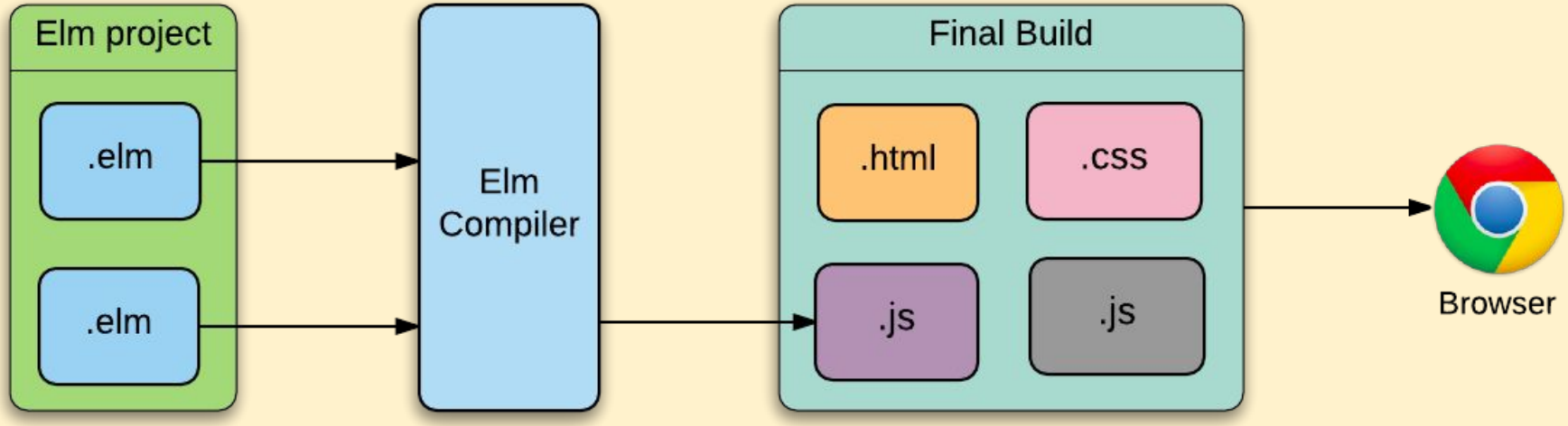
ELM

joy of programming

DOMAIN


Elm project
.elm
.elm
Elm
Compiler
Final Build
.html
.css
.js
.js
Browser

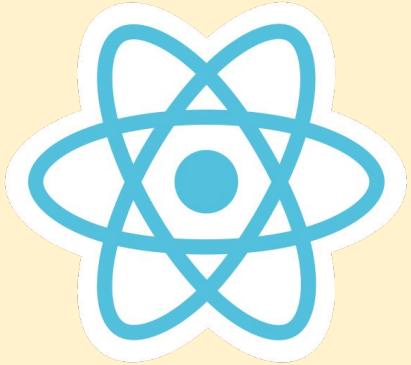
PROBLEM BEING SOLVED

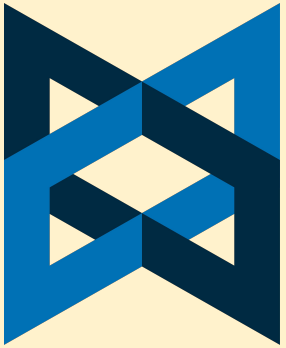
Runtime exceptions

a) introduced when adding new code

b) introduced by refactoring

THESE WON'T HELP MUCH



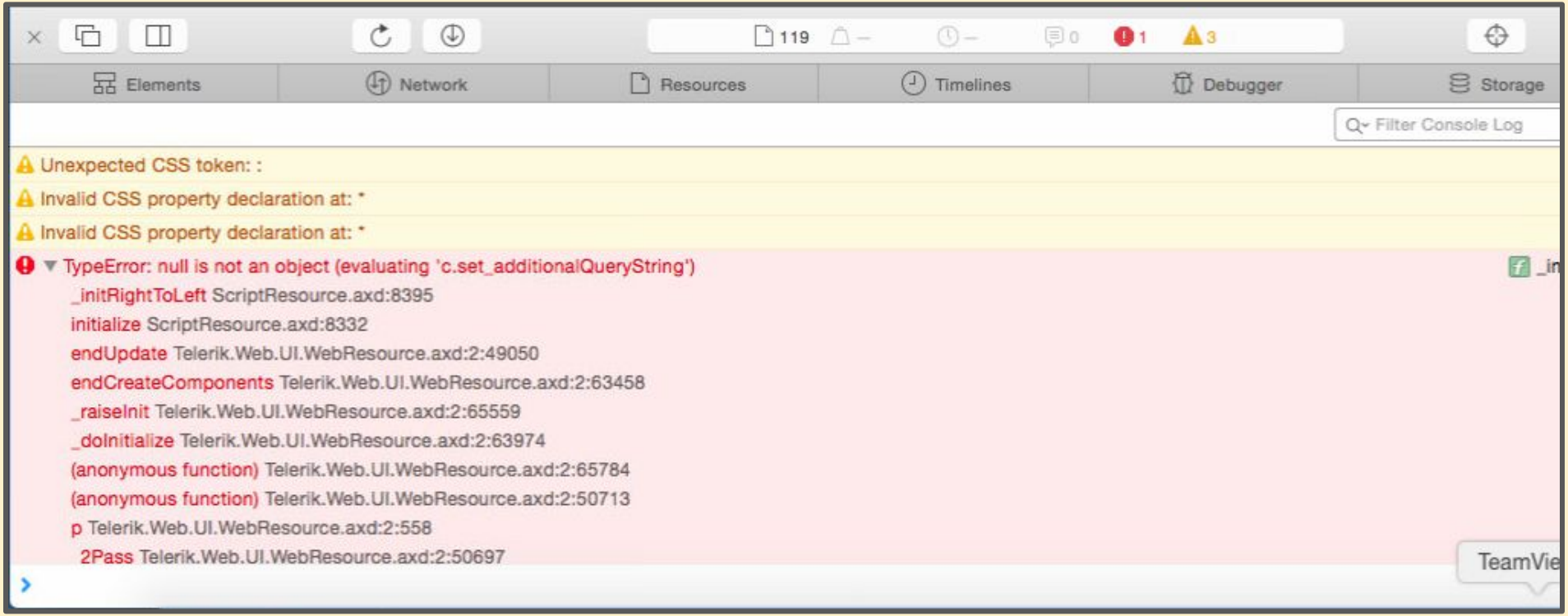


All of them are JavaScript products. JavaScript is not type safe language. Hence, these frameworks don't bring you close to safety.

RUNTIME EXCEPTIONS

```
Uncaught TypeError: undefined is not a function
    Ext.cmd.derive.applyTitle
    j
    b.implement.setConfig
    Ext.cmd.derive.show
    Ext.cmd.derive.alert
    Ext.cmd.derive.loginClick
```

RUNTIME EXCEPTIONS



WANT PROOFS?

No runtime exceptions

“NoRedInk has 36k lines of Elm, and after more than a year in production, it still has not produced a single runtime exception.”

Elm online documentation

http://elm-lang.org/

MORE PROOFS


Reactive
50,000 lines of Elm code
1.5 years in production
0 runtime exceptions
Elm

AND SOME MORE PROOFS

Ossi Hanhinen
@ohanhi

Follow

Some stats of our @elmlang project so far:
- 2-3 devs, 3mo
- 148 #Elm modules
- 1 JS file
- 3 major refactors
- 0 runtime exceptions 🎉

7:21 AM - 9 Oct 2015

90 132

HOW IS IT ACCOMPLISHED?

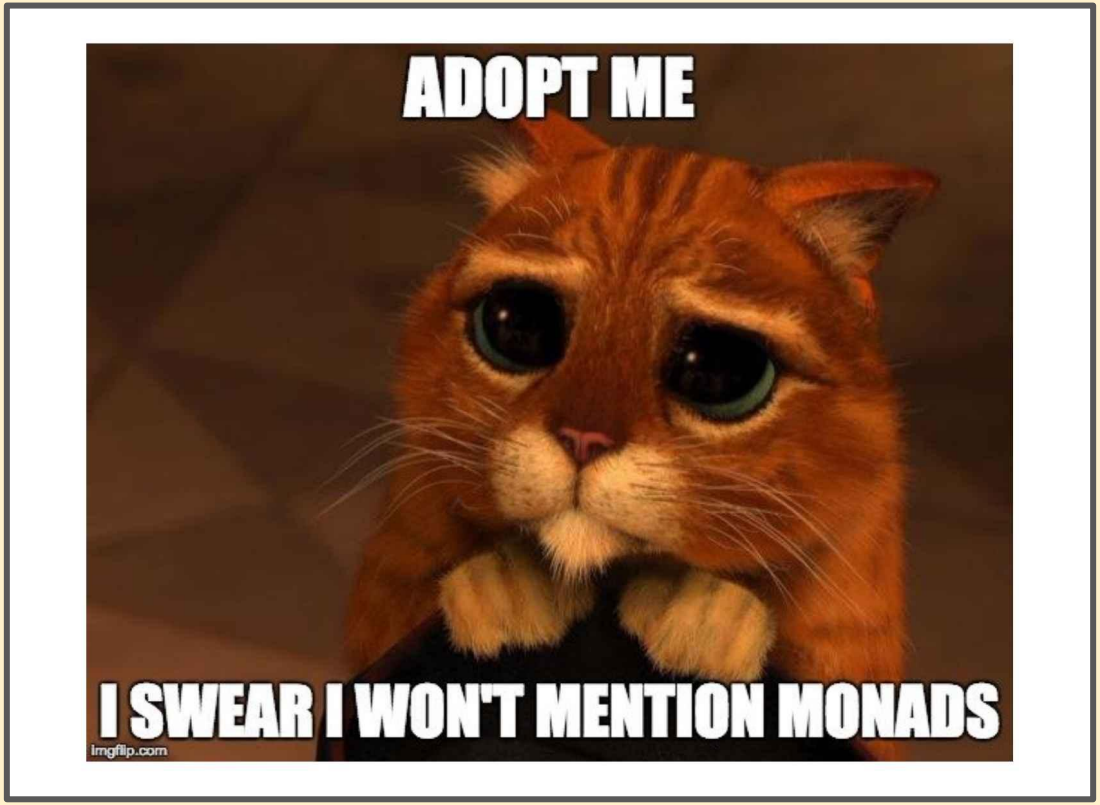
ELM is:

a) Functional

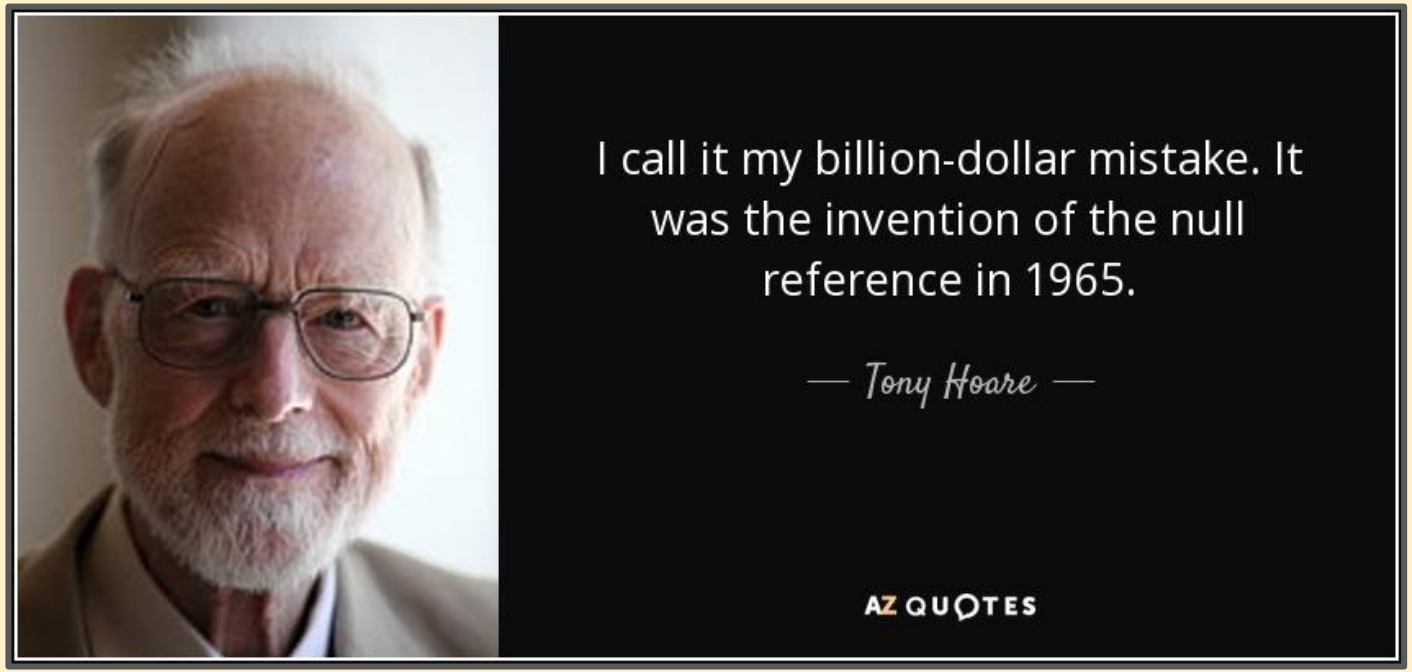
b) Immutable

c) Typed (Strong, Static, Inferred)

FUNCTIONAL BUT NO MONADS



NO NULL



JavaScript has *null* and *undefined*.

COMPILER IS NICE TO YOU

```
» elm-make test.elm --output=test.html
-- TYPE MISMATCH ---------------------------------------------------- test.elm

The branches of this `if` produce different types of values.

 8 ▷     if n < 0 then
 9 ▷         "negative"
10 ▷     else
11 ▷         n

The `then` branch has type:

    String

But the `else` branch is:

    number

Hint: These need to match so that no matter which branch we take, we always get
back the same type of value.

Detected errors in 1 module.
```

EASIER TESTING

You test pure functions. No mocks or stabs needed. Boils down to essentially comparing input with output.

TYPE INFERENCE

```elm
hermann =
  { first = "Hermann"
  , last = "Hesse"
  }


isOver50 person =
  person.age > 50


answer =
  isOver50 hermann
```

```
-- TYPE MISMATCH ------------------------------------ types/missing-field.elm

The 1st argument to function `isOver50` has an unexpected type.

13|   isOver50 hermann
              ^^^^^^^

Looks like a record is missing the field `age`

As I infer the type of values flowing through your program, I see a conflict
between these two types:

    { a | age : comparable }

    { first : String, last : String }
```

IMMUTABILITY

```
Detected errors in 1 module.


-- DUPLICATE DEFINITION ------------------------------ elm-examples/Playground.elm

Naming multiple top-level values `multiplier` makes things ambiguous. When you
say `multiplier` which one do you want?

199| multiplier =
   ^^^^^^^^^^

Find all the top-level values named `multiplier` and do some renaming. Make sure
the names are distinct!
```

CLEAN ARCHITECTURE

ELM is not just a language. It has TEA (The Elm Architecture) baked into it.

- Model — the state of your application
- Update — a way to update your state
- View — a way to view your state as HTML

JAVASCRIPT INTEROPERABILITY

When in time of need for some JS library, ELM makes it:

a) Possible

b) Clear that there are safe (ELM) and unsafe (JavaScript) parts of code

SO WHAT?

So, couple of things:

a) Code that is easier to reason about

b) Scaling applications is easier

c) Scaling teams is easier

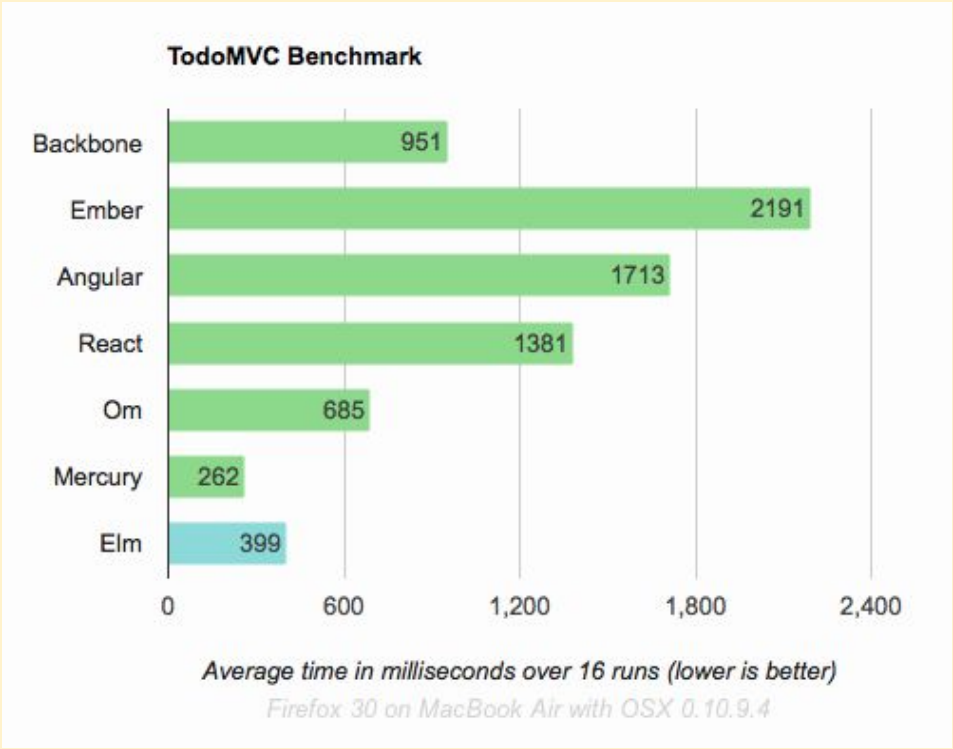
I don't need Elm!

Functional programming	→	Modern JS is inspired by FP
Components	→	React
Effect as data	→	Flux / Redux
Type checking	→	Flow.js / Typescript
Immutability	→	ImmutableJS / Ramda / Lodash
Modules	→	Webpack / Babel

Javascript

Please Just Work

WHAT ABOUT PERFORMANCE?



Thanks to:

a) Virtual DOM

b) Purity

c) Immutability

THANKS !