PROGRAMMING IS PAIN AND SUFFERING

goto; conference

Avoiding the Nightmare on Elm Street

Speaker Thomas Anagrius

goto; conference

Click ‘Rate Session’ to rate session and ask questions.

Thomas Anagrius

Developer at Humio in Sweden

M.Sc. in Computer Science from Aarhus University
majoring in Computer Graphics

Originally from Aarhus, Denmark.
Now living in Stockholm, Sweden.

Geeky Interests

- Web Tech (JavaScript, Elm, NodeJS, React, WebSockets)
- Automation (Continuous Delivery, TTD)
- Functional Programming (Clojure, Elm, Swift, Rust, Haskell)
- Microservices and Immutable Infrastructure (Terraform, Mesos, Kubernetes)
- The list goes on... no more space.

This is **not**
an introduction to Elm

Elm at a glance

- Typed Functional Language
- Based on ML
- Compiles to JavaScript
- Made for front-end
- No Runtime Exceptions
- Very Fast

```elm
initialModel : Model
initialModel =
    { speaker =
        { name = "Thomas Anagrius"
        , topic = "Pain"
        , rating = 3
        , mood = "Great"
        }
    , attendance = 10
    }



-- UPDATE


type Msg
    = EjectSpeaker
    | ThrowRottenTomato


update : Msg -> Model -> Model
update msg ({ speaker, attendance } as model) =
    case msg of
        EjectSpeaker ->
            { model | speaker = { speaker | mood = "Dead" } }

        ThrowRottenTomato ->
            { model
```

I’m neither trying to get you to use Elm or **not** to use Elm

Our Product: Humio

What

Log Aggregation / Monitoring - Startup

Stack

Skala, Akka, Kafka, Elm

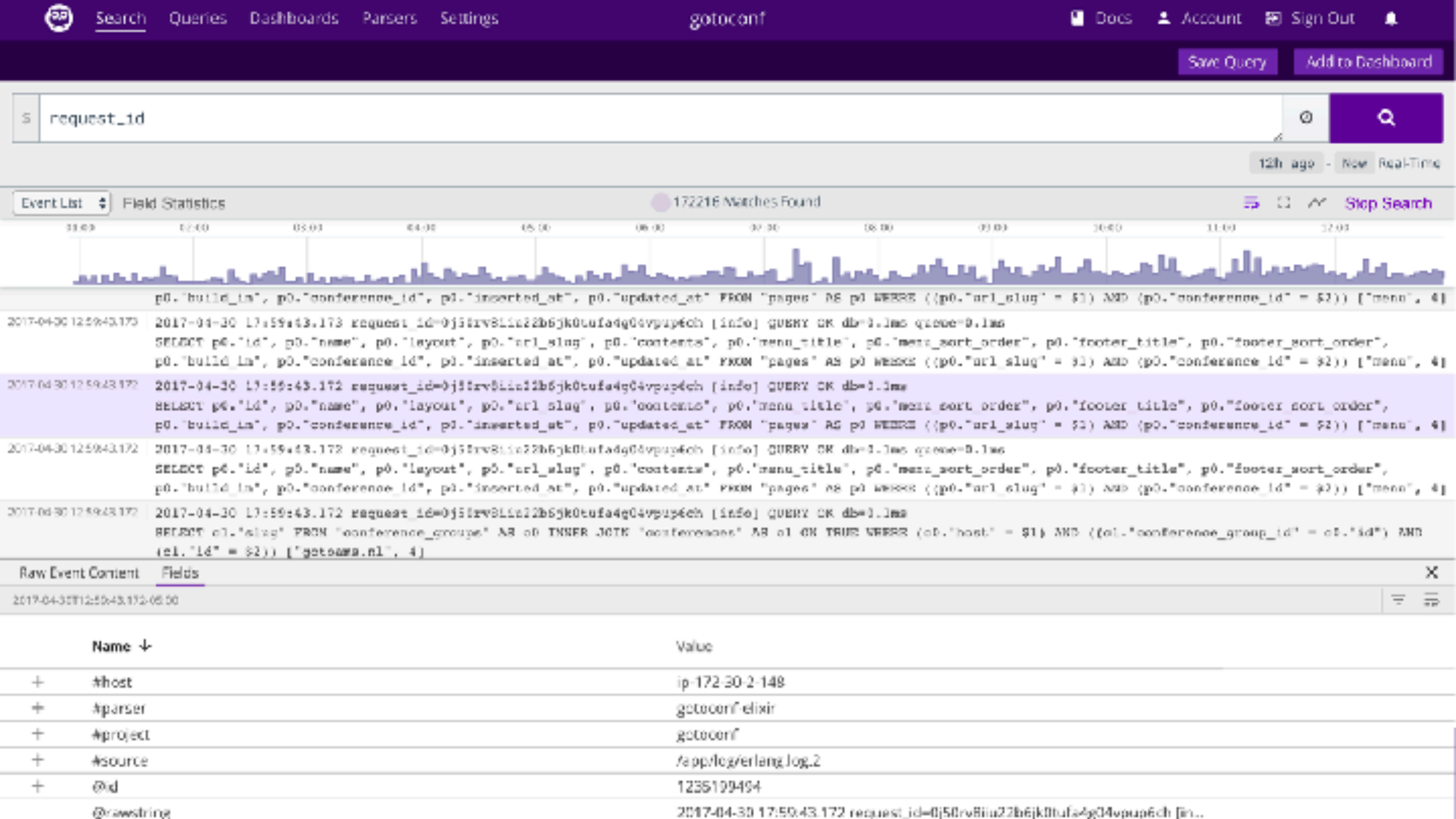
Elm and Humio

Production

1 Year

Lines

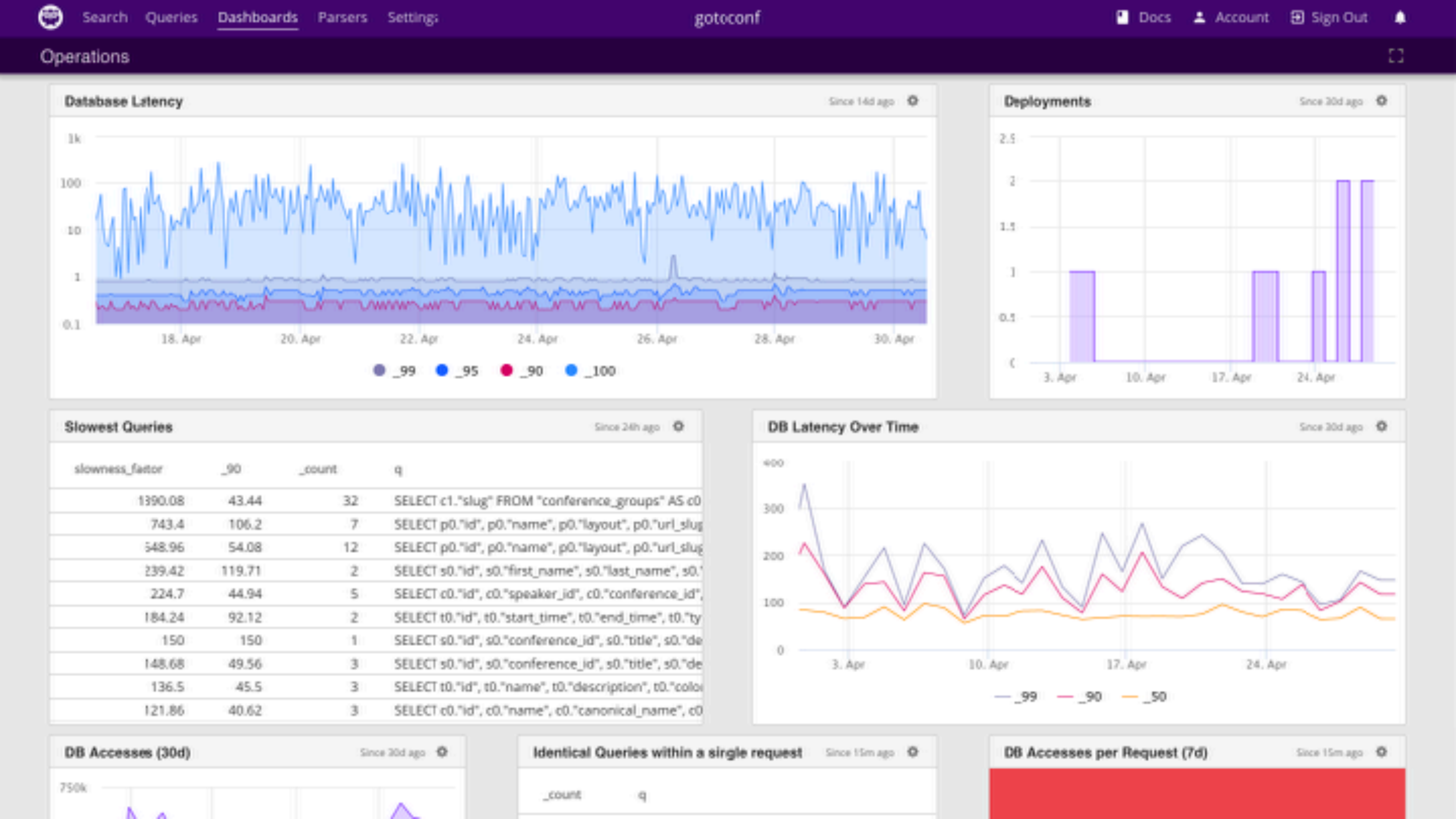
ca 50.000



```
p0."build_in", p0."conference_id", p0."inserted_at", p0."updated_at" FROM "pages" AS p0 WHERE ((p0."url_slug" = $1) AND (p0."conference_id" = $2)) ["menu", 4]
2017-04-30 12:59:43.173  2017-04-30 17:59:43.173 request_id=0j50rv8iiu22b6jk0tufa4g04vpup6ch [info] QUERY OK db=1.1ms queue=0.1ms
SELECT p0."id", p0."name", p0."layout", p0."url_slug", p0."contents", p0."menu_title", p0."menu_sort_order", p0."footer_title", p0."footer_sort_order",
p0."build_in", p0."conference_id", p0."inserted_at", p0."updated_at" FROM "pages" AS p0 WHERE ((p0."url_slug" = $1) AND (p0."conference_id" = $2)) ["menu", 4]
2017-04-30 12:59:43.172  2017-04-30 17:59:43.172 request_id=0j50rv8iiu22b6jk0tufa4g04vpup6ch [info] QUERY OK db=1.1ms
SELECT p0."id", p0."name", p0."layout", p0."url_slug", p0."contents", p0."menu_title", p0."menu_sort_order", p0."footer_title", p0."footer_sort_order",
p0."build_in", p0."conference_id", p0."inserted_at", p0."updated_at" FROM "pages" AS p0 WHERE ((p0."url_slug" = $1) AND (p0."conference_id" = $2)) ["menu", 4]
2017-04-30 12:59:43.172  2017-04-30 17:59:43.172 request_id=0j50rv8iiu22b6jk0tufa4g04vpup6ch [info] QUERY OK db=1.1ms queue=0.1ms
SELECT p0."id", p0."name", p0."layout", p0."url_slug", p0."contents", p0."menu_title", p0."menu_sort_order", p0."footer_title", p0."footer_sort_order",
p0."build_in", p0."conference_id", p0."inserted_at", p0."updated_at" FROM "pages" AS p0 WHERE ((p0."url_slug" = $1) AND (p0."conference_id" = $2)) ["menu", 4]
2017-04-30 12:59:43.172  2017-04-30 17:59:43.172 request_id=0j50rv8iiu22b6jk0tufa4g04vpup6ch [info] QUERY OK db=1.1ms
SELECT c1."slug" FROM "conference_groups" AS c0 INNER JOIN "conferences" AS c1 ON TRUE WHERE (c0."host" = $1) AND ((c1."conference_group_id" = c0."id") AND
(c1."id" = $2)) ["gotoams.nl", 4]
```

2017-04-30T12:59:43.172-05:00

Name	Value
#host	ip-172-30-2-148
#parser	gotoconf-elixir
#project	gotoconf
#source	/app/log/erlang.log.2
@id	1235109494
@rawstring	2017-04-30 17:59:43.172 request_id=0j50rv8iiu22b6jk0tufa4g04vpup6ch [in...



slowness_factor	_90	_count	q
1390.08	43.44	32	SELECT c1."slug" FROM "conference_groups" AS c0
743.4	106.2	7	SELECT p0."id", p0."name", p0."layout", p0."url_slug
648.96	54.08	12	SELECT p0."id", p0."name", p0."layout", p0."url_slug
239.42	119.71	2	SELECT s0."id", s0."first_name", s0."last_name", s0."
224.7	44.94	5	SELECT c0."id", c0."speaker_id", c0."conference_id",
184.24	92.12	2	SELECT t0."id", t0."start_time", t0."end_time", t0."ty
150	150	1	SELECT s0."id", s0."conference_id", s0."title", s0."de
148.68	49.56	3	SELECT s0."id", s0."conference_id", s0."title", s0."de
136.5	45.5	3	SELECT t0."id", t0."name", t0."description", t0."colo
121.86	40.62	3	SELECT c0."id", c0."name", c0."canonical_name", c0

_count	q

Short Demo

Reaction

WOW! You’re using Elm!

That’s Awesome

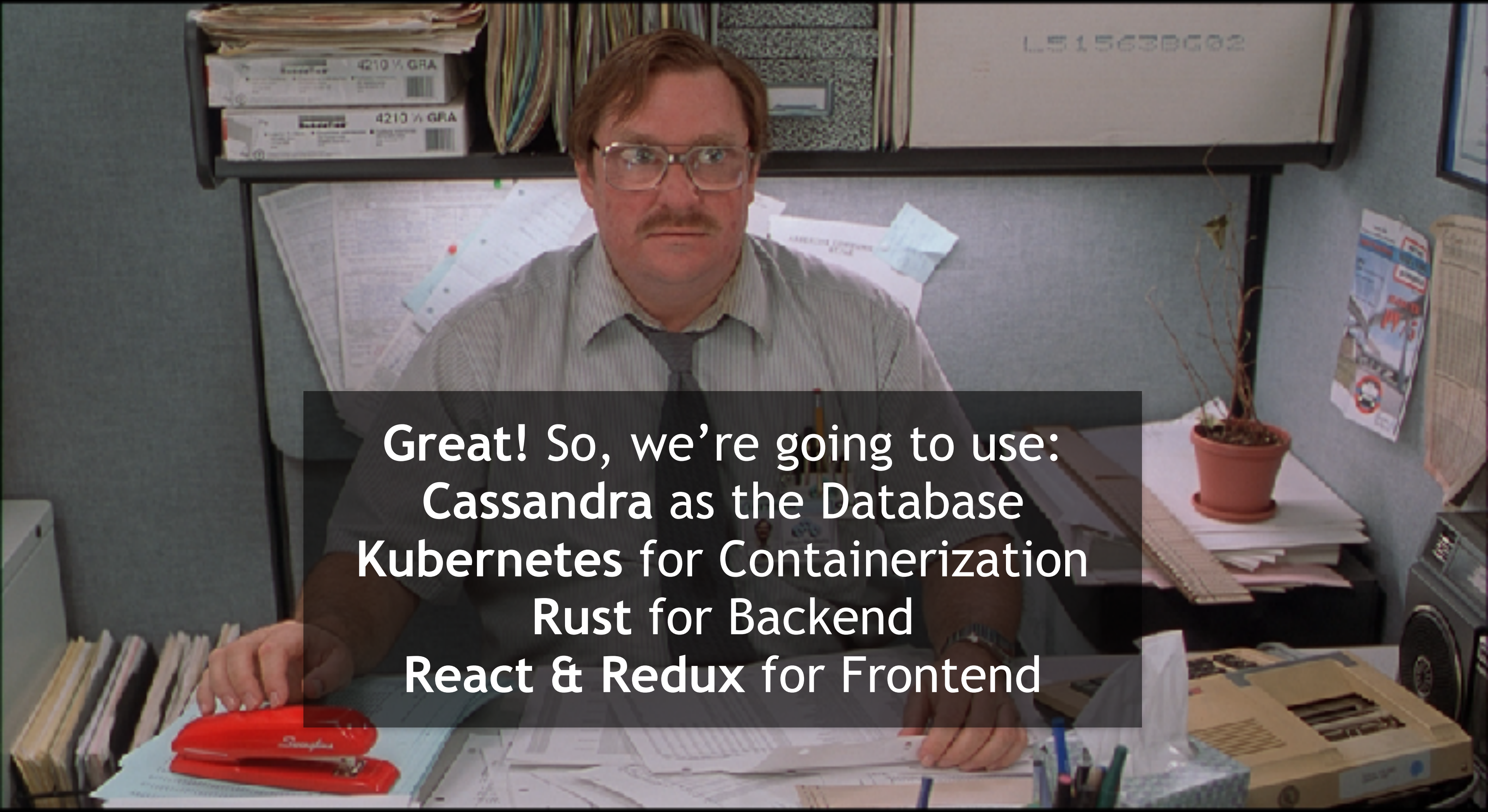
How did you dare?

When starting every project you have to make some tech decisions

How do you decide?

The Normal Scenario:

Thomas, I’ve picked you as the tech lead for a new IT project



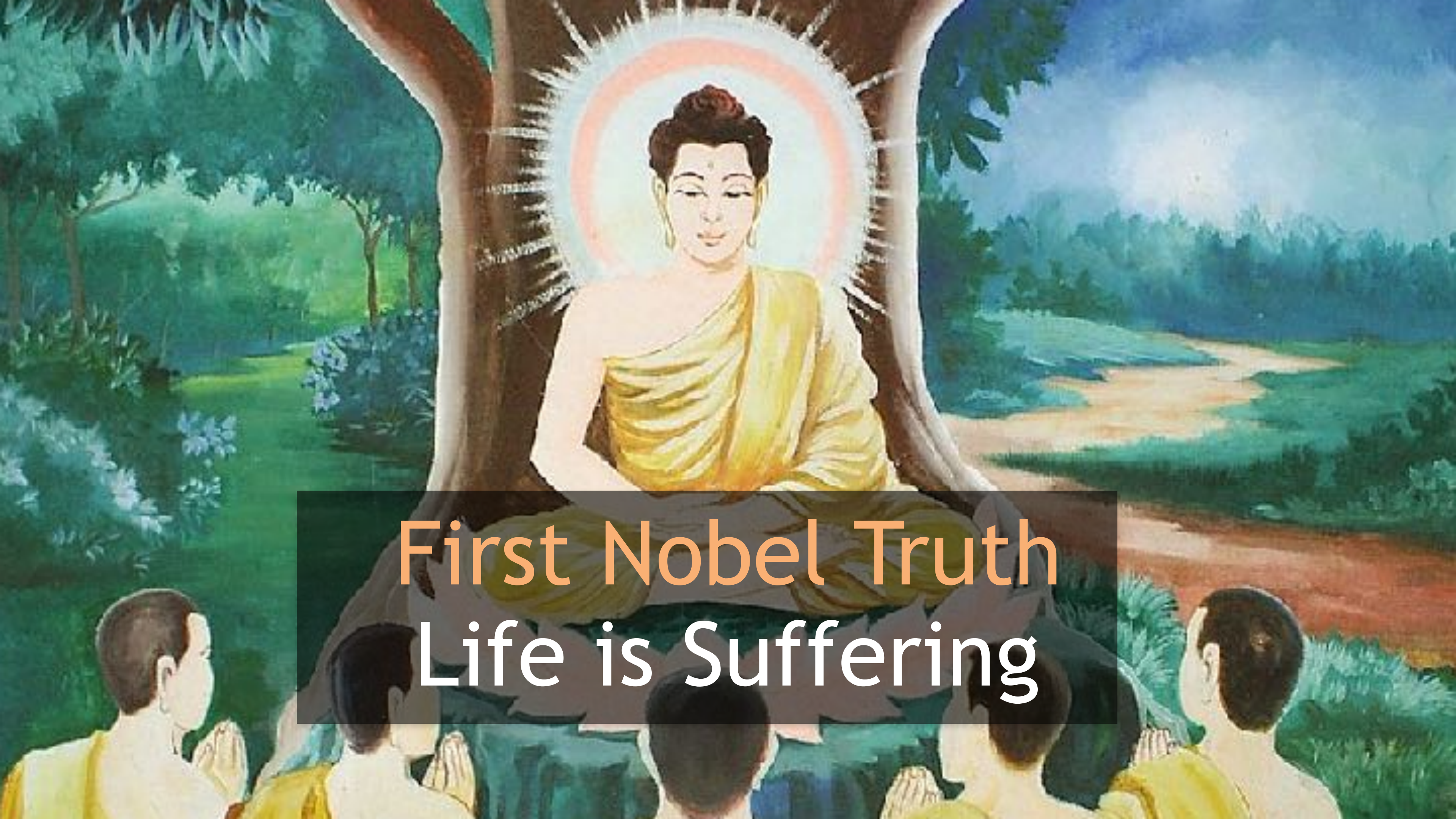
Great! So, we’re going to use:
Cassandra as the Database
Kubernetes for Containerization
Rust for Backend
React & Redux for Frontend

Um… what’s the project about again?

The Best Tool for the Job™

Best How?	Best Performance
	Best to Write
	Best Concurrency Support
	Best Cultural Fit
	Best Documentation
	Best to Debug

PROGRAMMING IS PAIN AND SUFFERING



First Nobel Truth

Life is Suffering

The [Modified] First Nobel Truth

All programming is suffering. To program, you must suffer.

It is impossible to program without experiencing some kind of suffering.

Your choice is
which pains

Every Choice has its Tradeoffs

JavaScript is pain.

Anyone who says differently is selling something.

Coding Pains: JavaScript

- Just Plain Bad Language
- Hard to Maintain in bigger teams
- Extremely aggressive type coercion
- Strange scoping (`this`)
- Many devs hate it

Only one Example Needed

```
NaN == NaN → false
```

Elm is pain.

Anyone who says differently is selling something.

Coding Pains: Elm

- Esoteric Syntax
- Not Mainstream
- Steep Learning Curve
- Compiled / Build Phase
- Hard to write
- Low Tool Support (Yet)
- Lacks Documentation
- No Examples of Large Code Bases
- Few 3rd party libs
- Slower Time-to-Feature

Weigh Tech Decisions like you would any other Business Decision.
Base it on ***strategy*** and the **problem at hand**.

Keeping your developers
happy and excited
is a business strategy.

For Humio the Considerations were:

Our Problem		Elm as Pain Reliever
Lot of data transformations	->	Functional Language
High volumes of data	->	Must be fast
Very interactive under load	->	Must be fast
Distributed Team	->	Typed Language
Company Culture	->	Language Nerds
Dashboards run for days	->	No Runtime Errors

Pains in Elm

1 year in

Our Pains with Elm

Early Adopter

Lots of Boilerplate

Routing Messages

Lack of Libraries

Architecture

Native Code

Refactoring

Learning Curve

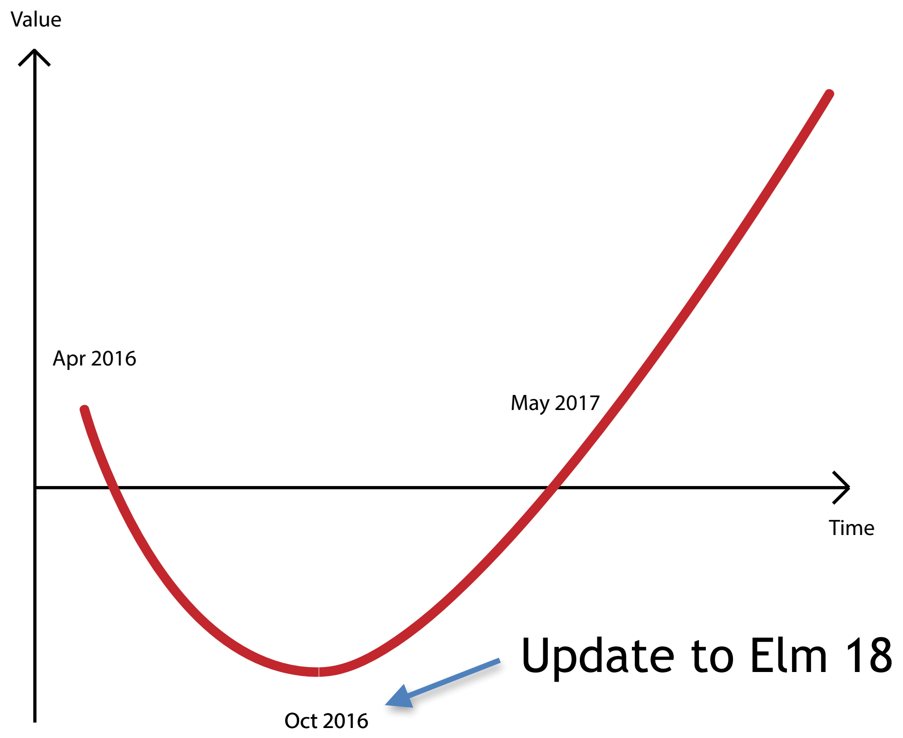
Bad Modeling

Evil Compiler

1. Being an Early Adopter

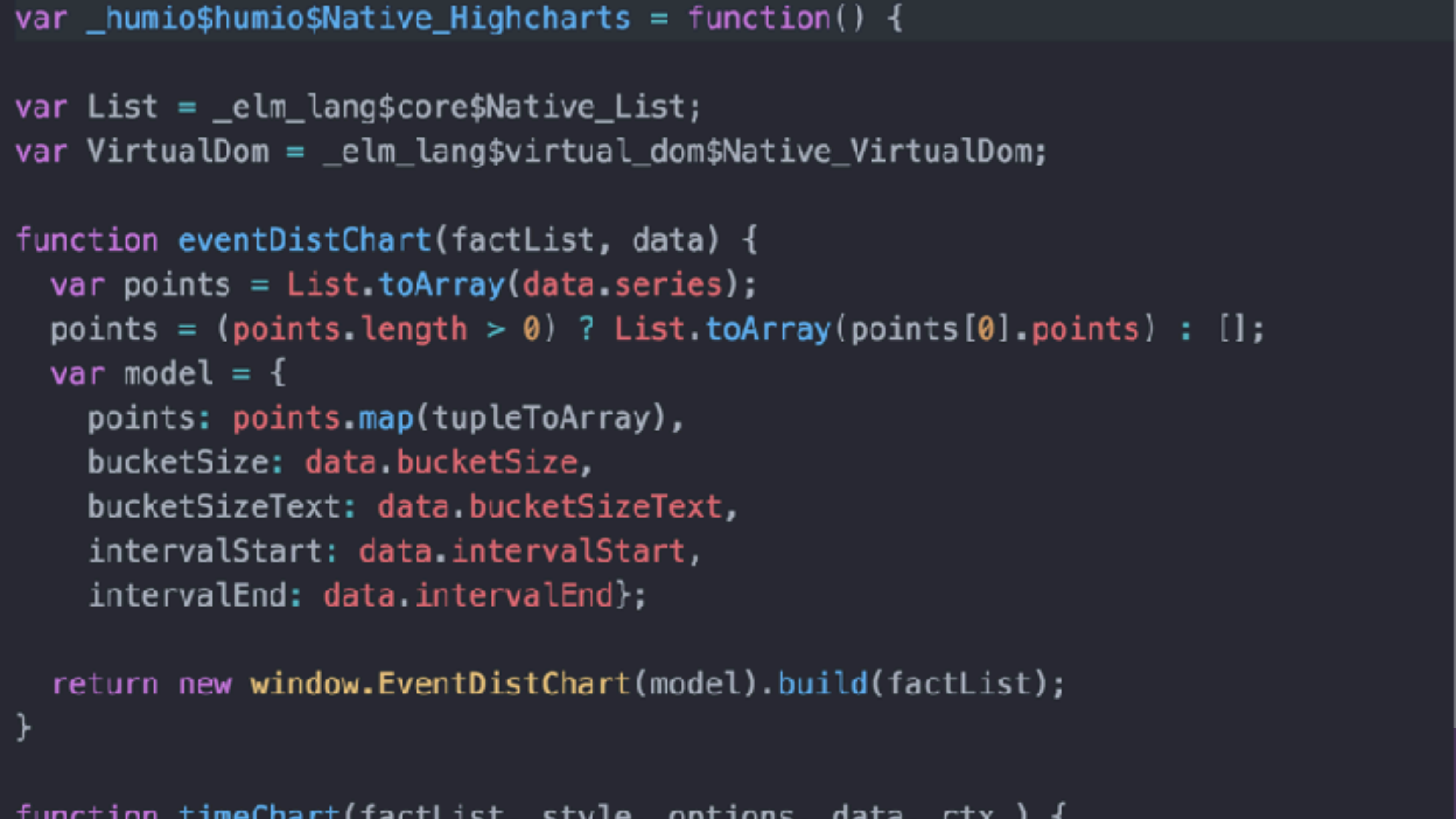
No one has blogged about it.
No one has written on Stack Overflow
You have to figure it out yourself

Investment is a J-curve



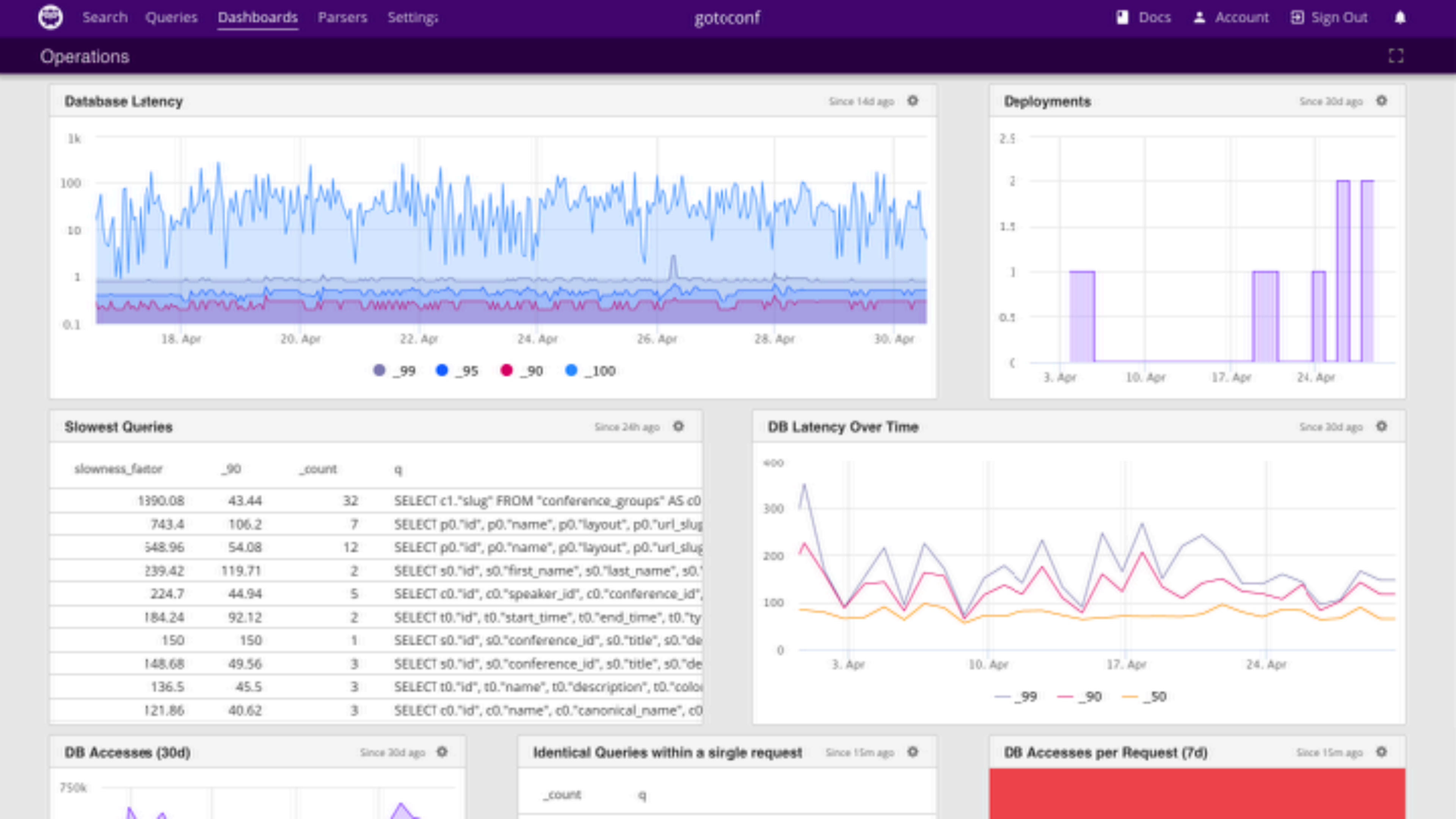
2. Be very careful with Native

At lease half of our troubles have come dealing with Native JS Code.



```
var _humio$humio$Native_Highcharts = function() {

var List = _elm_lang$core$Native_List;
var VirtualDom = _elm_lang$virtual_dom$Native_VirtualDom;

function eventDistChart(factList, data) {
  var points = List.toArray(data.series);
  points = (points.length > 0) ? List.toArray(points[0].points) : [];
  var model = {
    points: points.map(tupleToArray),
    bucketSize: data.bucketSize,
    bucketSizeText: data.bucketSizeText,
    intervalStart: data.intervalStart,
    intervalEnd: data.intervalEnd};

  return new window.EventDistChart(model).build(factList);
}

function timeChart(factList, style, options, data, ctx ) {
```

We had no choice. :(
Needed lots of charts and
no fully featured Elm Chart lib.



slowness_factor	_90	_count	q
1390.08	43.44	32	SELECT c1."slug" FROM "conference_groups" AS c0
743.4	106.2	7	SELECT p0."id", p0."name", p0."layout", p0."url_slug
548.96	54.08	12	SELECT p0."id", p0."name", p0."layout", p0."url_slug
239.42	119.71	2	SELECT s0."id", s0."first_name", s0."last_name", s0."
224.7	44.94	5	SELECT c0."id", c0."speaker_id", c0."conference_id",
184.24	92.12	2	SELECT t0."id", t0."start_time", t0."end_time", t0."ty
150	150	1	SELECT s0."id", s0."conference_id", s0."title", s0."de
148.68	49.56	3	SELECT s0."id", s0."conference_id", s0."title", s0."de
136.5	45.5	3	SELECT t0."id", t0."name", t0."description", t0."colo
121.86	40.62	3	SELECT c0."id", c0."name", c0."canonical_name", c0

_count	q

Say “Goodbye” to

Type Safety
No Errors At Runtime
Developer Sanity

Use Ports, Except when you can’t

They are like JNI for Java.
You talk to javascript through “an API”.
But they can be cumbersome.

3. Minimize Maybes

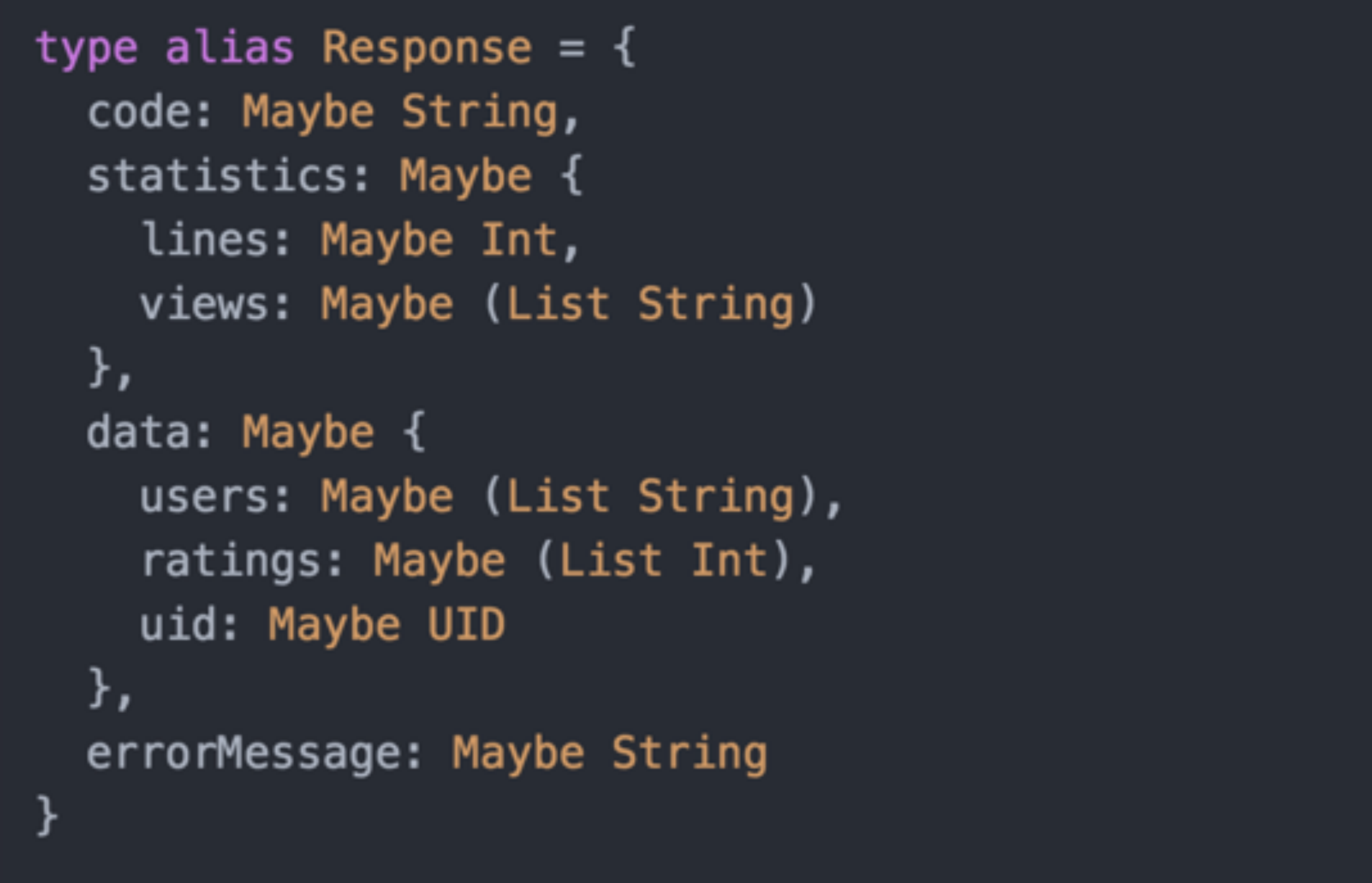
Instead of **nulls**, we have maybes in Elm.

```
x : Maybe Int
x =
  Just 100

y : Maybe Int
y =
  Nothing
```

Coming from JavaScript

We’re used to dealing with JSON directly from APIs. Implicitly knowing that values “will be set”.

Bad



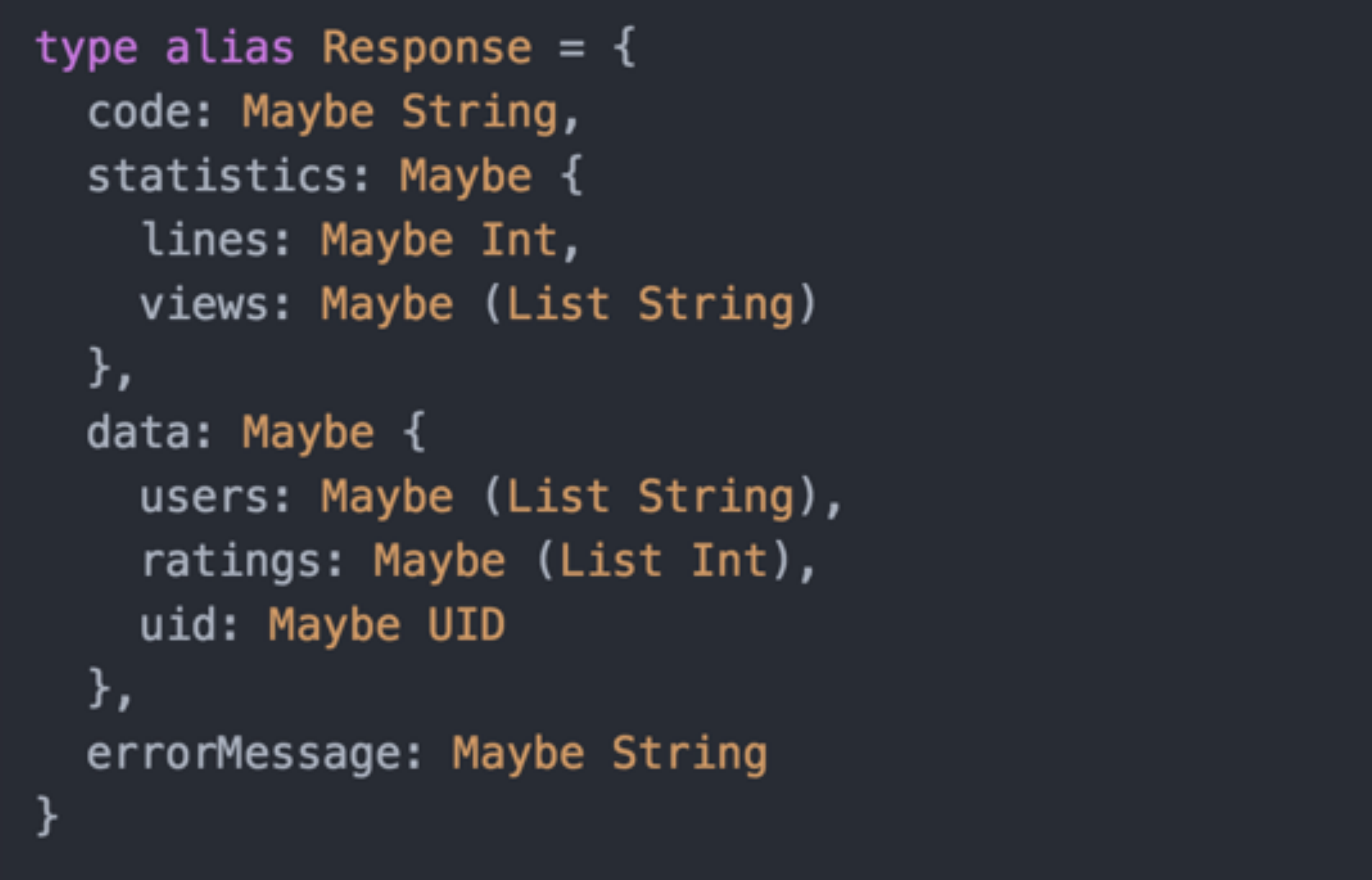
```elm
type alias Response = {
  code: Maybe String,
  statistics: Maybe {
    lines: Maybe Int,
    views: Maybe (List String)
  },
  data: Maybe {
    users: Maybe (List String),
    ratings: Maybe (List Int),
    uid: Maybe UID
  },
  errorMessage: Maybe String
}
```

```
case response.code of
  Just "OK" ->
    Accept
  Just "Error" ->
    Deny
  _ ->
    Deny
```

Possible Configurations

$$2^{\#maybe}$$

Bad



```elm
type alias Response = {
  code: Maybe String,
  statistics: Maybe {
    lines: Maybe Int,
    views: Maybe (List String)
  },
  data: Maybe {
    users: Maybe (List String),
    ratings: Maybe (List Int),
    uid: Maybe UID
  },
  errorMessage: Maybe String
}
```

Combinatorial Explosion!

$$2^9 = 512$$

And Elm will force you to deal will all cases!

Better

```
type alias Response = {
  code: String,
  statistics: Maybe {
    lines: Int,
    views: Maybe (List String)
  },
  data: Maybe {
    users: List String,
    ratings: Maybe (List Int),
    uid: Maybe UID
  },
  errorMessage: Maybe String
}
```

Possible Configurations

$$2^6 = 64$$

Modeling

The most important skill
you need to pick up in Elm is
representing your problem properly.

Good

```
type Response =
  StatisticsResponse { lines: Int, views: Maybe (List String)}
  | DataResponse { users: List String, uid: Maybe UID }
  | ErrorResponse String
```

Possible Configurations

$$2^1 + 2^1 + 2^0 = 5$$

Live Coding
Modelling

You reduce logical errors and complexity enormously

Recommended

Watch Richard Feldman's Presentation:

Making Impossible States Impossible

Last Thing

The Compiler

Compiler, you are the bane of my existence.

Compiler, you are the joy of my life.

I have had days where I would not try my implementation in the browser for hours. **Yet, I was sure it would work.**

COMPILES AND WORKS FIRST TIME.
WHAT DID I DO WRONG?
quickmeme.com

Elm lets you sleep well at night.

Strong sense of

If it compiles it works.

The Dream

Strongly Type Language

No Errors at Runtime

Time Traveling Debugger

Our Nightmare on Elm Street

HIGHCHARTS

goto; conference

Please

Remember to rate this session

Thank you!