Code BEAM SF
2018

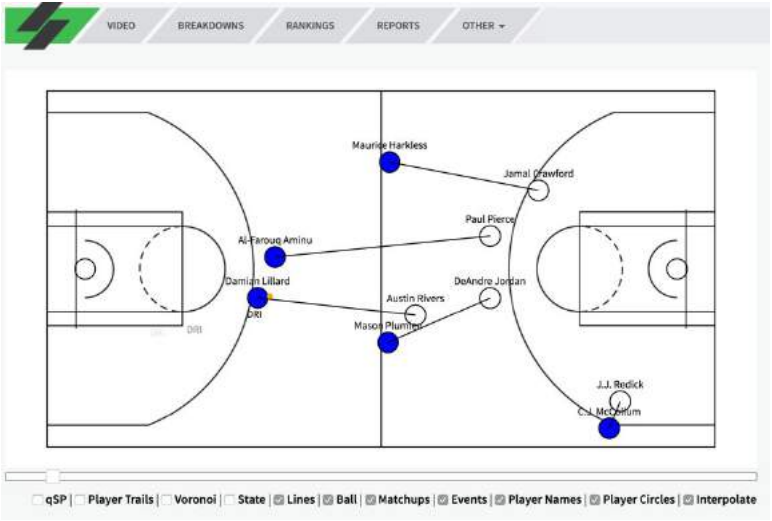
Hype For Types

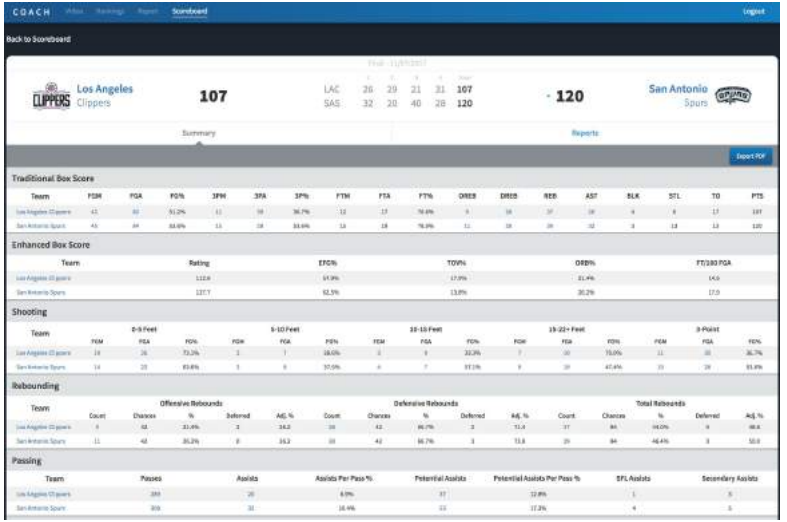
Using Dialyzer to bring type checking to your Elixir code

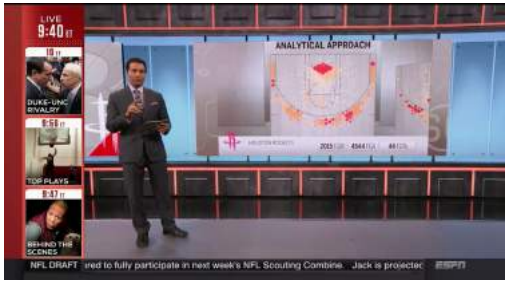
Hi, I'm Emma Cunningham! 

@emmatcu
#CodeBEAMSF

@emmatcu
#CodeBEAMSF







@emmatcu
#CodeBEAMSF

By the end of this talk, we will be able to...

- ★ understand & appreciate the power of type theory
- ★ be able to apply these concepts to our Elixir development practices
- ★ live slightly more free of stress knowing that we've got a type checker that has our back!

@emmatcu
#CodeBEAMSF

Type theory & me

@emmatcu
#CodeBEAMSF

Portrait of a type-loving functional programmer (c. 2008)

@emmatcu
#CodeBEAMSF

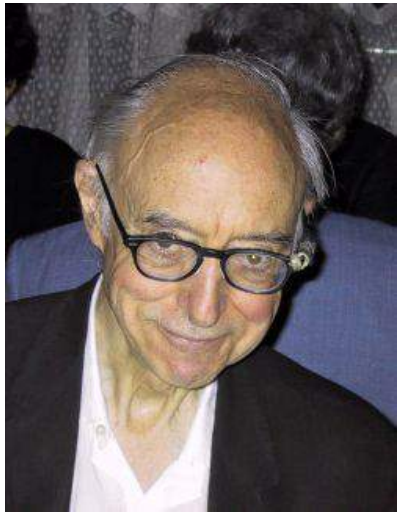
The Curry-Howard isomorphism

@emmatcu
#CodeBEAMSF

The Curry-Howard isomorphism?

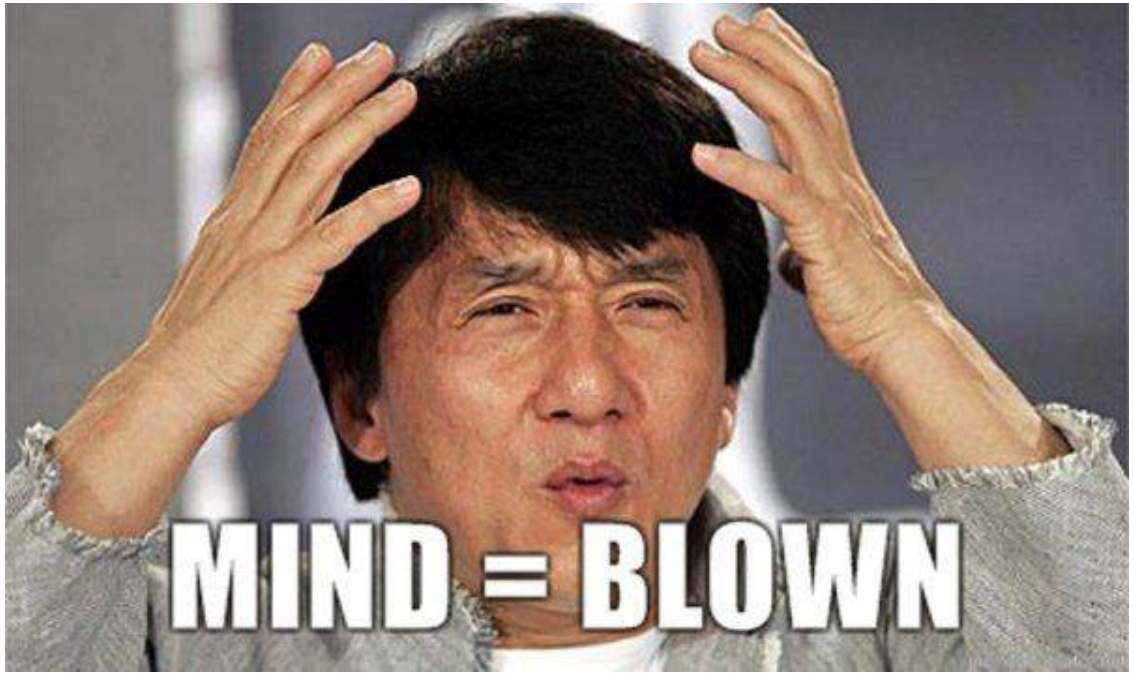
@emmatcu
#CodeBEAMSF

The Curry-Howard isomorphism!



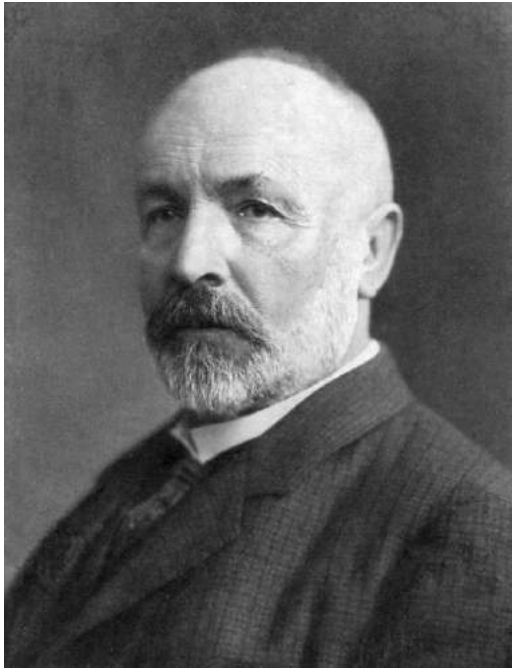
@emmatcu
#CodeBEAMSF

The Curry-Howard isomorphism!

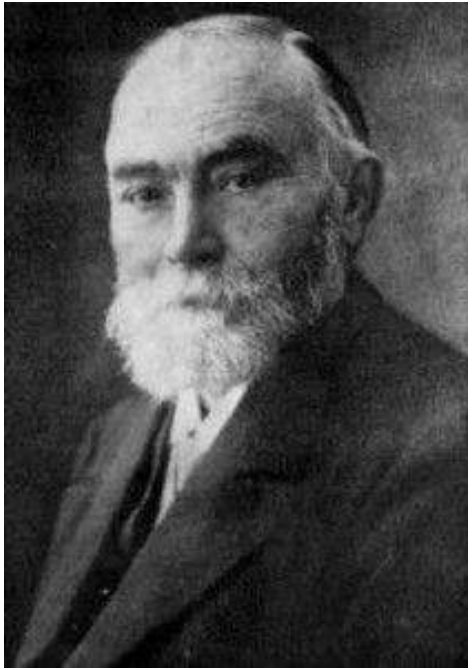


@emmatcu
#CodeBEAMSF

The origins of type theory



Georg Cantor



Gottlob Frege

@emmatcu
#CodeBEAMSF

Russell's paradox

There's a barber who shaves all people who do not shave themselves...

Who shaves the barber??

@emmatcu
#CodeBEAMSF

Types were created to avoid ***paradoxes****!*

The Curry-Howard isomorphism

@emmatcu
#CodeBEAMSF

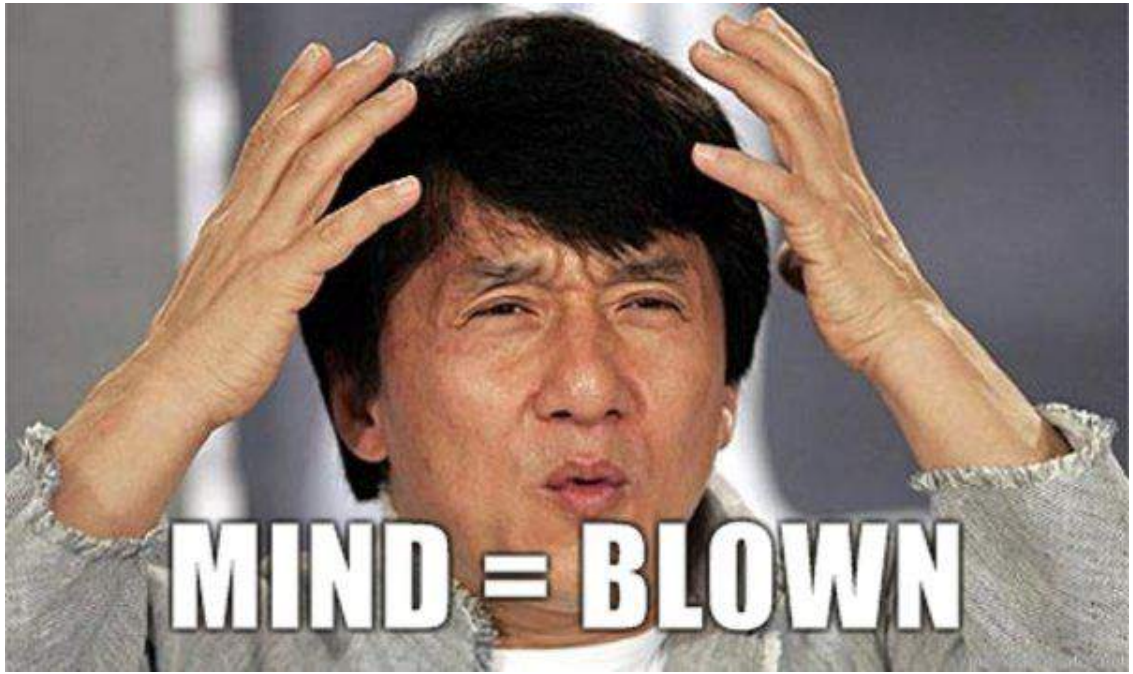
The Curry-Howard isomorphism, revisited!

Logic	**CS**
Proofs	Programs
Formula	Types
A implies B	function from A to B
Axiom	System primitive
Soundness theorem	Compiler
Completeness theorem	Debugger
Incompleteness theorem	Infinite loop

Paradoxes in logical theory are like bugs in software.

If types can save us from paradoxes, they can also save us from bugs.

The Curry-Howard isomorphism!



@emmatcu
#CodeBEAMSF

Type systems

Type systems

a set of formalized rules that assign types to units of meaning within a language (variables, functions, etc.) and dictate what constitutes a type error

Type systems

Static

type values known at **compile time**, either by specification or by inference

Dynamic

types are associated with **run-time** values, no need for specification

@emmatcu
#CodeBEAMSF

Type systems

Strong

errors when there are type conflicts (e.g. when a function is called w/ an argument of the wrong type)

Weak

may perform implicit type conversion or sometimes unpredictable results as the product of a type conflict

@emmatcu
#CodeBEAMSF

Type systems

A **type-safe** language does not allow violations of the language's type system

Type systems

Type-checking is the process of verifying and enforcing the constraints of types; this process may occur at compile-time or run-time.

Elixir and types

@emmatcu
#CodeBEAMSF

You already know about types in Elixir!

@emmatcu
#CodeBEAMSF

Some types in Elixir

```
1                       # integer
1.0                     # float
true                    # boolean
:foo                    # atom
"bar"                   # string
[1, 2, 3]               # list
%{foo: “bar”}           # map
```

@emmatcu
#CodeBEAMSF

And you may already be leveraging some of the power of types in Elixir

```
def foo(n) when is_integer(n), do: n
def foo(n) when is_float(n), do: round(n)
```

@emmatcu
#CodeBEAMSF

Strongly and dynamically typed

```
def foo(n), do: n * 2

def bar(), do: foo("2")
```

@emmatcu
#CodeBEAMSF

Imagine the horror (at runtime)

@emmatcu
#CodeBEAMSF

Success typing with Dialyzer

a type checker for a language like Erlang:

- should work without type declarations being there (can accept hints),
- should be simple and readable,
- should adapt to the language (and not the other way around),
- only complain on type errors that would guarantee a crash.

"Practical Type Inference Based on Success Typings", Lindahl & Sagonas 2006

Dialyzer: Discrepancy Analyzer

```
03:25:33hype (master)$ mix dialyzer
Checking PLT...
[:asn1, :compiler, :connection, :cowboy, :cowlib, :crypto, :db_connection,
 :decimal, :dialyxir, :ecto, :eex, :elixir, :file_system, :gettext, :kernel,
 :logger, :mime, :phoenix, :phoenix_ecto, :phoenix_html, :phoenix_live_reload,
 :phoenix_pubsub, :plug, :poison, :poolboy, :postgrex, :public_key, :ranch,
 :runtime_tools, :ssl, :stdlib]
PLT is up to date!
Starting Dialyzer
dialyzer args: [check_plt: false,
 init_plt: '/Users/emmacunningham/Documents/conf/elixir-dialyzer-demo/hype/_build/dev/dialyxir_erlang-20.1.4_elixir-1.5.2_deps-dev.plt',
 files_rec: ['/Users/emmacunningham/Documents/conf/elixir-dialyzer-demo/hype/_build/dev/lib/hype/ebin'],
 warnings: [:unknown]]
done in 0m2.3s
lib/hype_web/controllers/page_controller.ex:6: Function index/2 has no local return
lib/hype_web/controllers/page_controller.ex:7: The call 'Elixir.HypeWeb.PageController':foo(<<_:8>>) will never return since it differs in the 1st
 argument from the success typing arguments: (number())
done (warnings were emitted)
```

@emmatcu
#CodeBEAMSF

Add Dialyzer/Dialyxir to your Elixir project

`mix.exs`

```elixir
def project do
  [
    # ...
    dialyzer: [plt_add_deps: :transitive]
  ]
end

defp deps do
  [
    # ...
    {:dialyxir, "~> 0.5.0", only: [:dev], runtime: false}
  ]
end
```

@emmatcu
#CodeBEAMSF

Get/compile the dep & generate plt

```
$ mix do deps.get, deps.compile, dialyzer --plt
```

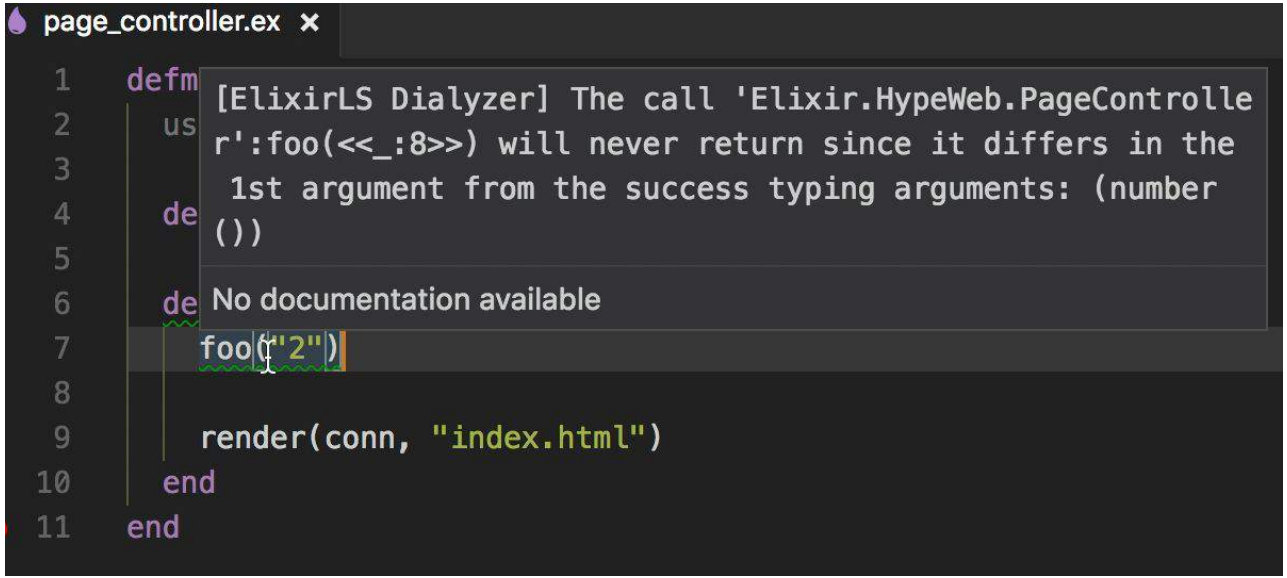
n.b.: This may take a very, very long time. You only have to do this the first time you run Dialyzer on a project

Run Dialyzer

`mix dialyzer`

```
lib/hype_web/controllers/page_controller.ex:6: Function index/2 has no
local return
lib/hype_web/controllers/page_controller.ex:7: The call 'Elixir.HypeWeb
.PageController':foo(<<_:8>>) will never return since it differs in the
 1st argument from the success typing arguments: (number())
```

Elixir LS support



@emmatcu
#CodeBEAMSF

@spec

```
@spec foo(integer) :: integer
def foo(n), do: n * 2

def index(conn, _params) do
  foo("2")

  render conn, "index.html"
end
```

@emmatcu
#CodeBEAMSF

But what about testing?

But what about testing?

Type checking frees your tests from needing to check these low-level concerns and instead lets them focus on business logic.

Some benefits

- ★ Reduce runtime errors
- ★ Type annotations help w/ documentation
- ★ Project maintainability improved
- ★ Sleep easy at night

h.t. @bodil

@emmatcu
#CodeBEAMSF

Code BEAM
SF 2018

💖Thank you!💖

Emma Cunningham

@emmatcu
the.cunning.ham@gmail.com