

Code BEAM SF

2018

# *Hype For Types*

---

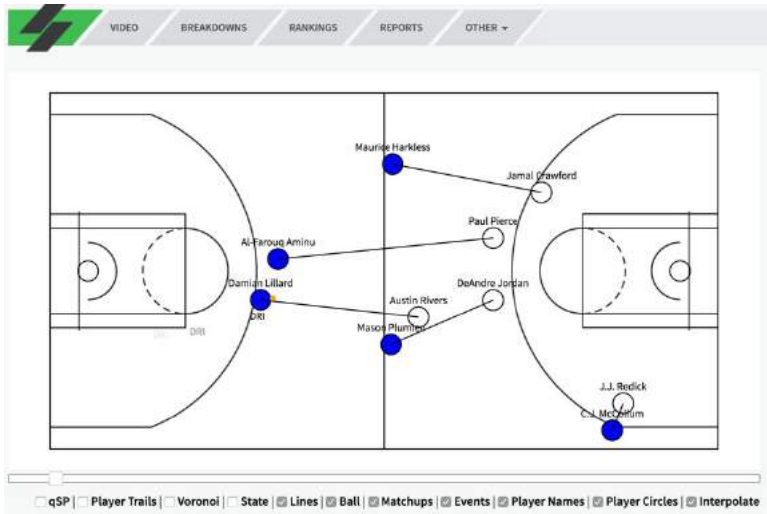
Using Dialyzer to  
bring type  
checking to your  
Elixir code

*Hi, I'm Emma Cunningham!* 🙌

@emmatcu  
#CodeBEAMSF



@emmatcu  
#CodeBEAMSF



COACH | [Play](#) | [Settings](#) | [Reports](#) | [Scoreboard](#) | [Logout](#)

Back to Scoreboard

Feb - 1, 2017

Los Angeles Clippers **107** | LAC 26 29 21 31 187 | SAS 32 30 40 28 120 | San Antonio Spurs **120**

Summary | Reports | [Export PDF](#)

### Traditional Box Score

Team	FSR	FGA	FG%	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST	BLK	STL	TO	PTS
Los Angeles Clippers	45	80	51.2%	13	38	36.7%	12	17	70.6%	9	34	37	8	8	17	107
San Antonio Spurs	45	89	53.9%	15	28	53.6%	13	28	76.9%	11	39	50	10	3	14	120

### Enhanced Box Score

Team	Rating	FG%	TO%	ORB%	FT/3P%
Los Angeles Clippers	110.8	51.9%	17.9%	81.4%	14.5
San Antonio Spurs	122.7	52.3%	13.8%	80.2%	17.0

### Shooting

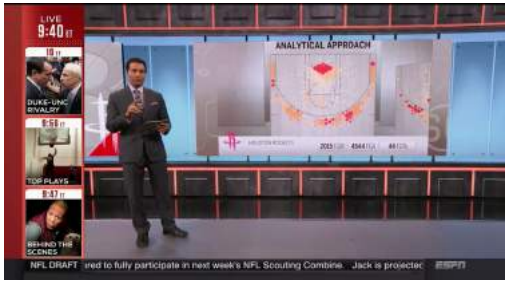
Team	FGM	0-9 Feet				9-16 Feet				16-23 Feet				3-Point	FGM	FG%
		FGM	FGA	FG%	Def.	FGM	FGA	FG%	Def.	FGM	FGA	FG%	Def.			
Los Angeles Clippers	24	26	92.3%	1	1	80.0%	3	3	100%	7	10	70.0%	11	30	36.7%	
San Antonio Spurs	14	23	60.9%	5	9	55.6%	0	0	0%	8	10	80.0%	21	28	75.0%	

### Rebounding

Team	Count	Offensive Rebounds				Defensive Rebounds				Total Rebounds					
		Count	Chances	%	Def.	Count	Chances	%	Def.	Count	Chances	%	Def.		
Los Angeles Clippers	9	18	33.3%	3	34.2	19	42	45.2%	3	11.4	37	84	84.0%	9	88.4
San Antonio Spurs	11	42	26.2%	9	24.2	20	42	47.6%	3	13.8	29	84	46.4%	9	10.9

### Passing

Team	Passes	Assists	Assists Per Pass %	Potential Assists	Potential Assists Per Pass %	STL Assists	Secondary Assists
Los Angeles Clippers	289	29	9.7%	37	12.8%	1	5
San Antonio Spurs	319	32	10.0%	33	10.3%	4	0



@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?*



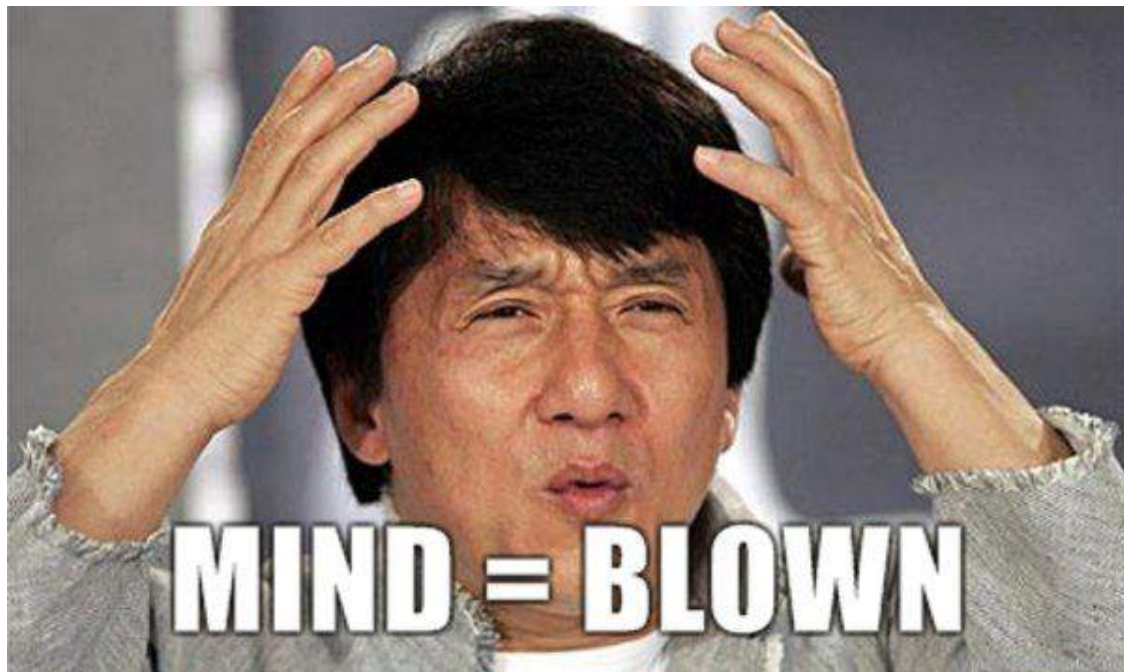
@emmatcu  
#CodeBEAMSF

# *The Curry-Howard isomorphism!*



@emmatcu  
#CodeBEAMSF

# *The Curry-Howard isomorphism!*

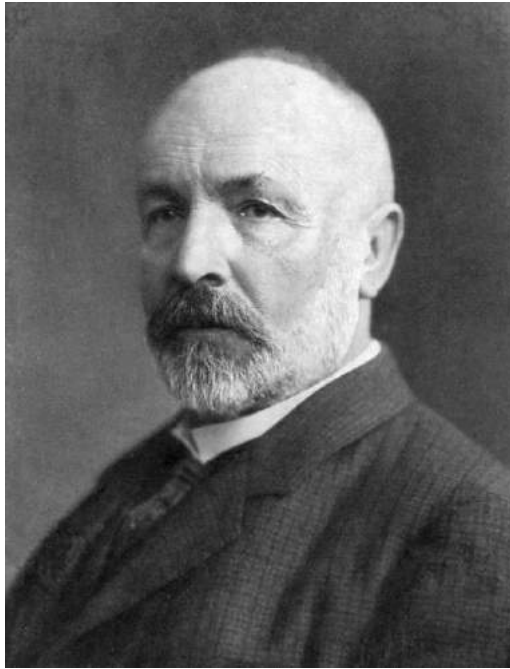


@emmatcu

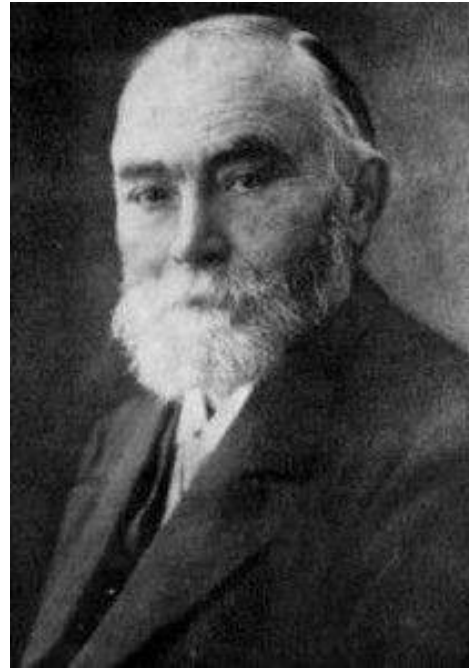
#CodeBEAMSF

# *The origins of type theory*

@emmatcu  
#CodeBEAMSF



Georg Cantor



Gottlob Frege

@emmatcu  
#CodeBEAMSF



# Russell's paradox

@emmatcu  
#CodeBEAMSF



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!***

@emmatcu  
#CodeBEAMSF



# *The Curry-Howard isomorphism*

@emmatcu  
#CodeBEAMSF

# *The Curry-Howard isomorphism, revisited!*

## Logic

Proofs

Formula

A implies B

Axiom

Soundness theorem

Completeness theorem

Incompleteness theorem

## CS

Programs

Types

function from A to B

System primitive

Compiler

Debugger

Infinite loop

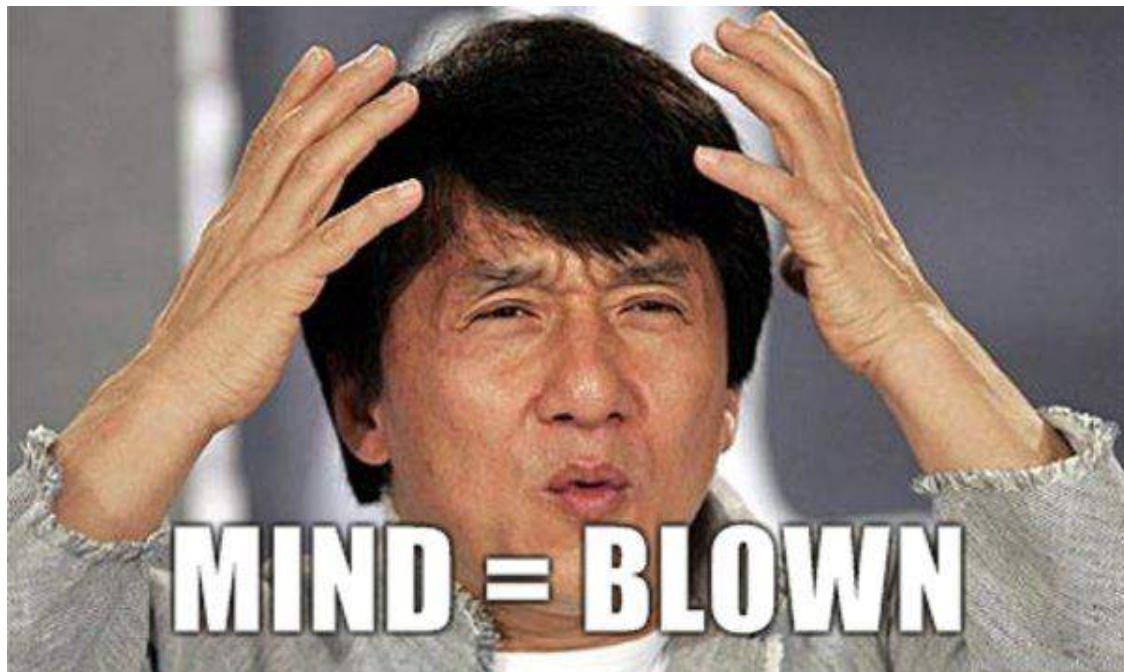
@emmatcu  
#CodeBEAMSF

*Paradoxes in logical theory are like  
bugs in software.*

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

@emmatcu  
#CodeBEAMSF

# *The Curry-Howard isomorphism!*



@emmatcu

#CodeBEAMSF

# *Type systems*

@emmatcu  
#CodeBEAMSF

# *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

@emmatcu  
#CodeBEAMSF

# *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

# *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



# *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

<code>1</code>	<code># integer</code>
<code>1.0</code>	<code># float</code>
<code>true</code>	<code># boolean</code>
<code>:foo</code>	<code># atom</code>
<code>"bar"</code>	<code># string</code>
<code>[1, 2, 3]</code>	<code># list</code>
<code>%{foo: "bar"}</code>	<code># map</code>

*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")
```

# I thought I met the h0r0r0r (at trum+time)

```
[error] #PID<0.395.0> r  
Server: localhost:4000  
Request: GET /  
** (exit) an exception v  
** (ArithmeticError)  
  (hype) lib/hype_  
  (hype) lib/hype_  
  (hype) lib/hype_  
  (hype) lib/hype_  
  (hype) lib/hype_  
  (phoenix) lib/pt  
  (hype) lib/hype_  
  (hype) lib/plug_  
  (hype) lib/hype_  
  (plug) lib/plug_  
  (cowboy) /Users/
```

4



:cowboy\_protocol.execute/

@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

@emmatcu  
#CodeBEAMSF

# 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

```
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*

@emmatcu  
#CodeBEAMSF

# *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())
```

@emmatcu  
#CodeBEAMSF

# Elixir LS support

```
page_controller.ex x
1 defm [ElixirLS Dialyzer] The call 'Elixir.HypeWeb.PageControlle
2   us r':foo(<<_:8>>) will never return since it differs in the
3     de 1st argument from the success typing arguments: (number
4       de ())
5     de No documentation available
6     de
7     de foo("2")
8     de
9     de render(conn, "index.html")
10    end
11  end
```

@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?*

@emmatcu  
#CodeBEAMSF



## *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.

@emmatcu  
#CodeBEAMSF

## *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