Killed by 'Worse is Better'

About me:

Name: Adam McCullough
@TheWizardTower on X
TheWizardTower@protonmail.com
Espresso nerd
Board Game Czar
Tango Dancer
Sometimes paid to argue about software

These slides live at:
github.com/TheWizardTower/killed-by-worse-is-better

Some preliminaries:

Some preliminaries:

- This isn't a technical talk.

Some preliminaries:

- This isn't a technical talk.
  - More about philosophy, history, business, and human nature

Some preliminaries:

- This isn't a technical talk.
  - More about philosophy, history, business, and human nature
- This isn't a cheerful talk

Overview:

Overview:

1. "Why Isn't FP On Top?"
2. The past:
    1. LISP
    2. Unix
3. Observations?
4. Haskell
5. Golang
6. Prod Standard
7. Diagnosis
8. Prescription

Why isn't FP on top?

Why Isn't Functional Programming the Norm? – Richard Feldman

1.5M views • 5 years ago

Metosin

Richard is a member of the Elm core team, the author of Elm in Action from Mannin...

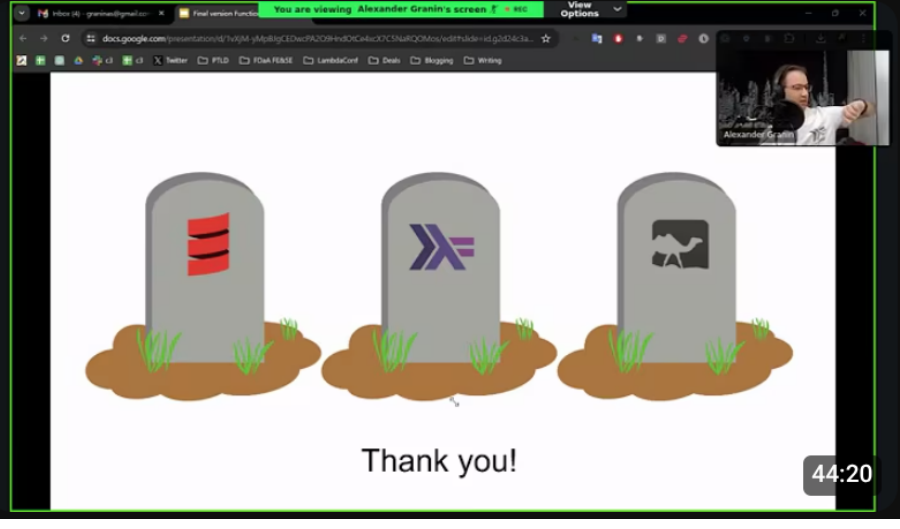
28 chapters Introduction | Language | Killer Apps | Ruby Rails...

Is Functional Programming DEAD Already?

83K views • 6 months ago

Modern Software Engineering

Functional Programming was the bright new thing, but the noise about it seems to have quietened down lately, is this because it ...

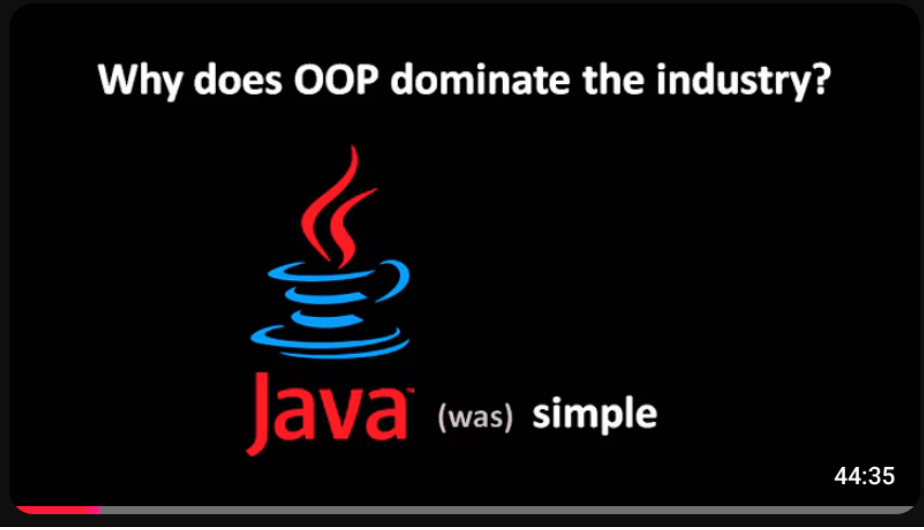


Functional Programming: Failed Successfully by Alexander Granin

6K views • 10 months ago

LambdaConf

Functional Programming: **Failed** Successfully by Alexander Granin at #LambdaConf2024 Get your ticket for #LambdaConf2025 ...



Object-Oriented Programming is Bad

2.4M views • 9 years ago

Brian Will

An explanation of why you should favor procedural programming over **Object-Orien...**

14 chapters Introduction | Outline | Problems | Functional...

It seems to be at the top of everyone's mind.

It seems to be at the top of everyone's mind.

Because, if you're already initiated, all you see are advantages

It seems to be at the top of everyone's mind.

Because, if you're already initiated, all you see are advantages

So what are we missing?

LISP!

LISP!

What did LISP get right?

LISP!

What did LISP get right?

- Elegant

LISP!

What did LISP get right?

- Elegant
  - "Maxwell's Equations for software" -- Alan Kay

LISP!

What did LISP get right?

- Elegant
  - "Maxwell's Equations for software" -- Alan Kay
  - Lithe instantiation of the Lambda Calculus

LISP!

What did LISP get right?

- Elegant
  - "Maxwell's Equations for software" -- Alan Kay
  - Lithe instantiation of the Lambda Calculus
- Defined a pile of FP Idioms

LISP!

What did LISP get right?

- Elegant
  - "Maxwell's Equations for software" -- Alan Kay
  - Lithe instantiation of the Lambda Calculus
- Defined a pile of FP Idioms
- Also the first (to my knowledge) to call out "this function does IO" vs "This is a pure function"

LISP!

What did LISP get right?

- Elegant
  - "Maxwell's Equations for software" -- Alan Kay
  - Lithe instantiation of the Lambda Calculus
- Defined a pile of FP Idioms
- Also the first (to my knowledge) to call out "this function does IO" vs "This is a pure function"
  - Functions vs "pseudo-functions"

Also pioneered things like:

Also pioneered things like:

1. Garbage Collection
2. Dynamic Typing (but we won't hold that against it)
3. REPLs!
4. Tree Data Structures
5. Self-Hosted Compiler
6. Macros

Okay, what did UNIX get right?

Okay, what did UNIX get right?

- "Never bet against the cheap plastic model" --ESR

Okay, what did UNIX get right?

- "Never bet against the cheap plastic model" --ESR
- Unix could run on machines that cost $10-20k, or up to a million

Okay, what did UNIX get right?

- "Never bet against the cheap plastic model" --ESR
- Unix could run on machines that cost $10-20k, or up to a million
- So a well-off enthusiast could tinker with it at home, and apply those skills at work or Uni

Okay, what did UNIX get right?

- "Never bet against the cheap plastic model" --ESR
- Unix could run on machines that cost $10-20k, or up to a million
- So a well-off enthusiast could tinker with it at home, and apply those skills at work or Uni
- Aimed at correctness, but wasn't a slave to theory

…but what did Unix innovate?

Innovations:

Innovations:

- "Everything is a file"
- Configuration in flat text files, rather than bespoke binary formats
  - ...which nurtured a whole pile of text processing tools
- Return codes were useful
- Pipes are neat

Which, to be honest, is not a ton (aside from pipes).

Which, to be honest, is not a ton (aside from pipes).

It iterated much more rapidly, producing half a dozen standards from different outfits before Lisp produced one

Which, to be honest, is not a ton (aside from pipes).

It iterated much more rapidly, producing half a dozen standards from different outfits before Lisp produced one

But didn't do anything tremendously groundbreaking

Which, to be honest, is not a ton (aside from pipes).

It iterated much more rapidly, producing half a dozen standards from different outfits before Lisp produced one

But didn't do anything tremendously groundbreaking

It just did it in a way that became the lingua franca for CS departments in the 70s

Which, to be honest, is not a ton (aside from pipes).

It iterated much more rapidly, producing half a dozen standards from different outfits before Lisp produced one

But didn't do anything tremendously groundbreaking

It just did it in a way that became the lingua franca for CS departments in the 70s

...and then industry in the 80s

Which, to be honest, is not a ton (aside from pipes).

It iterated much more rapidly, producing half a dozen standards from different outfits before Lisp produced one

But didn't do anything tremendously groundbreaking

It just did it in a way that became the lingua franca for CS departments in the 70s

...and then industry in the 80s

...and then hacker culture in the 90s

The obvious comparison notices that the Unix slide has much less "ground-breaking firsts" than the LISP one had.

Unix seemed to take a more iterative, fox-like/ generalist approach, while Lisp was very much a specialist hedgehog.

Early personal Unix machines were hardly cheap, between 10-20k in 1980s dollars

Early personal Unix machines were hardly cheap, between 10-20k in 1980s dollars

...but Lisp machines were $50-80k

Early personal Unix machines were hardly cheap, between 10-20k in 1980s dollars

...but Lisp machines were $50-80k

...and slower.

Early personal Unix machines were hardly cheap, between 10-20k in 1980s dollars

...but Lisp machines were $50-80k

...and slower.

For context:
The TRS-80 was released in 1977 for $600

BBC Micro Model B: 1982, £335

Commodore 64: 1982, $595

Apple II: 1977, $1,300 (no monitor or disk drive)

New car in 1984: $6,000

House in 1984: ~$80-100,000

...and then there's all the Drama!

Context:

Context:

Lisp innovation HQ was MIT's AI Lab

Context:

Lisp innovation HQ was MIT's AI Lab

In 1979, Russel Noftsker (Administrator of the AI Lab)

Context:

Lisp innovation HQ was MIT's AI Lab

In 1979, Russel Noftsker (Administrator of the AI Lab)

approached Richard Greenblatt (one of the hackers in the Lab)

Context:

Lisp innovation HQ was MIT's AI Lab

In 1979, Russel Noftsker (Administrator of the AI Lab)

approached Richard Greenblatt (one of the hackers in the Lab)

and said "Let's commercialize Lisp."

Context:

Lisp innovation HQ was MIT's AI Lab

In 1979, Russel Noftsker (Administrator of the AI Lab)

approached Richard Greenblatt (one of the hackers in the Lab)

and said "Let's commercialize Lisp."

The bet was on hardware getting fast enough that high-level langs like Lisp would gain market share

Faction one: LMI

Faction one: LMI

- Short for Lisp Machines, Inc
- Founded in 1979
- CEO Richard Greenblatt
- Wanted to keep MIT's AI Lab atmosphere - open, informal, productive
- Wanted to fund the lab Kickstarter-style, where customers buy the product, then they develop it
  - To be fair, he found several people who were game...

Basically, they wanted to spread the hacker ethos to the world, and thought it was good enough to produce products that would sell immediately

Basically, they wanted to spread the hacker ethos to the world, and thought it was good enough to produce products that would sell immediately

So, flat hierarchy, focus on excellence above all else, etc...

Faction two: Symbolics, Inc

Faction two: Symbolics, Inc

- Founded in 1980
- CEO Russel Noftsker
- Wanted to keep the MIT AI Lab going, felt that a commercial endeavor was the best way to fund it
- Wanted to run a sales focused company
- But in service of the same goal

How'd it go?

How'd it go?

LMI got 3-4 of the Lab members

How'd it go?

LMI got 3-4 of the Lab members

Symbolics got 14

How'd it go?

LMI got 3-4 of the Lab members

Symbolics got 14

Two abstained from either. Martin Minsky and RMS

Noftsker had a problem: He didn't have an office for his people to work in, or hardware for them to work on.

Noftsker had a problem: He didn't have an office for his people to work in, or hardware for them to work on.

He made a deal with the AI Lab director, Patrick Winston, to let them use AI lab facilities and hardware.

Noftsker had a problem: He didn't have an office for his people to work in, or hardware for them to work on.

He made a deal with the AI Lab director, Patrick Winston, to let them use AI lab facilities and hardware.

In exchange, MIT would get free licenses to any software Symbolics developed.

Noftsker had a problem: He didn't have an office for his people to work in, or hardware for them to work on.

He made a deal with the AI Lab director, Patrick Winston, to let them use AI lab facilities and hardware.

In exchange, MIT would get free licenses to any software Symbolics developed.

Trouble was, Stallman was still at the lab, and actively did not want to see a single company get a monopoly. So he started re-implementing their software himself.

Noftsker had a problem: He didn't have an office for his people to work in, or hardware for them to work on.

He made a deal with the AI Lab director, Patrick Winston, to let them use AI lab facilities and hardware.

In exchange, MIT would get free licenses to any software Symbolics developed.

Trouble was, Stallman was still at the lab, and actively did not want to see a single company get a monopoly. So he started re-implementing their software himself.

Which lead to a lot of pernicious accusations of code theft, because of the aforementioned deal

So you had a language that was blindingly, staggeringly beautiful language that innovated so hard we're still cribbing notes from it, almost 70 years later...

So you had a language that was blindingly, staggeringly beautiful language that innovated so hard we're still cribbing notes from it, almost 70 years later...

...stuck on a machine that cost 3x as much as a more performant model

So you had a language that was blindingly, staggeringly beautiful language that innovated so hard we're still cribbing notes from it, almost 70 years later...

...stuck on a machine that cost 3x as much as a more performant model

So what happened is that it became monumentally influential, but itself never captured the mainstream

Okay, next up: Haskell

Okay, next up: Haskell

What were Haskell's goals?

Okay, next up: Haskell

What were Haskell's goals?

- Effects without cheating
  - None of this "pseudo-function" black magic nonsense
- A common language for PLT research
  - That used to be Miranda, but that was proprietary
  - Also, Miranda took the pseudo-function route, and the folks behind Haskell wanted to make lazy IO work properly
  - (To the point that Haskell didn't get side-effects until Haskell Report 1.3 in May of 1996)

Cool. What was its *impact*?

Cool. What was its *impact*?

- They succeeded in their goals:
  - Common research language
  - Succeeded in making lazy IO concrete
  - More to the point, it's a tremendously fertile ground for new PLT research
    - Effect tracking systems, dependent types, STM, etc.
- QuickCheck seems to have leaked out into the zeitgeist
- It also is a benchmark for learning statically typed functional programming
- For a while, it looked like it might grab the mainstream, but alas, not meant to be, for a pile of

I'm not letting you off the hook that easily, what were those?

I'm not letting you off the hook that easily, what were those?

- Haskell didn't quite get their build tooling as airtight as it needs to be for industry use

I'm not letting you off the hook that easily, what were those?

- Haskell didn't quite get their build tooling as airtight as it needs to be for industry use
- "The Cabal Treadmill" is not a sign of success

I'm not letting you off the hook that easily, what were those?

- Haskell didn't quite get their build tooling as airtight as it needs to be for industry use
- "The Cabal Treadmill" is not a sign of success
- Stack helped for a while, but that turned into its own political turf war

I'm not letting you off the hook that easily, what were those?

- Haskell didn't quite get their build tooling as airtight as it needs to be for industry use
- "The Cabal Treadmill" is not a sign of success
- Stack helped for a while, but that turned into its own political turf war
- They also got a lot of architecture changes exactly backwards

I'm not letting you off the hook that easily, what were those?

- Haskell didn't quite get their build tooling as airtight as it needs to be for industry use
- "The Cabal Treadmill" is not a sign of success
- Stack helped for a while, but that turned into its own political turf war
- They also got a lot of architecture changes exactly backwards
  - We *absolutely* need to change the API on this foundational library because it isn't perfect!
  - We *absolutely* cannot fix head, map, fold, think of all the textbooks that we'd invalidate!

I'm not letting you off the hook that easily, what were those?

- Haskell didn't quite get their build tooling as airtight as it needs to be for industry use
- "The Cabal Treadmill" is not a sign of success
- Stack helped for a while, but that turned into its own political turf war
- They also got a lot of architecture changes exactly backwards
  - We *absolutely* need to change the API on this foundational library because it isn't perfect!
  - We *absolutely* cannot fix head, map, fold, think of all the textbooks that we'd invalidate!
- Nevermind the political purity tests that narrowed the funnel even further

Let's contrast this with Go.

Go's objectives:

Let's contrast this with Go.

Go's objectives:

- Like C, except without the bad parts

Let's contrast this with Go.

Go's objectives:

- Like C, except without the bad parts
- Like Python, but compiled and a bit faster

Let's contrast this with Go.

Go's objectives:

- Like C, except without the bad parts
- Like Python, but compiled and a bit faster
- Little-to-no extra tutorialization required

Let's contrast this with Go.

Go's objectives:

- Like C, except without the bad parts
- Like Python, but compiled and a bit faster
- Little-to-no extra tutorialization required
- No bikeshedding over formatting styles

Let's contrast this with Go.

Go's objectives:

- Like C, except without the bad parts
- Like Python, but compiled and a bit faster
- Little-to-no extra tutorialization required
- No bikeshedding over formatting styles
- Easy concurrency. Channels and share-by-communicating

How'd it go?

How'd it go?

Not great.

How'd it go?

Not great.

- It's a language specifically designed to accomodate fresh-outs, and not scare, intimidate, or confuse them
- It also got the semantics of its 'big killer feature' wrong.

How'd it go?

Not great.

- It's a language specifically designed to accomodate fresh-outs, and not scare, intimidate, or confuse them
- It also got the semantics of its 'big killer feature' wrong.
  - Published study of bugs in Go programs
  - They studied 171 concurrency bugs in things like Docker, K8s, gRPC, and others
  - MORE THAN HALF were because of "Non-traditional, go specific problems"

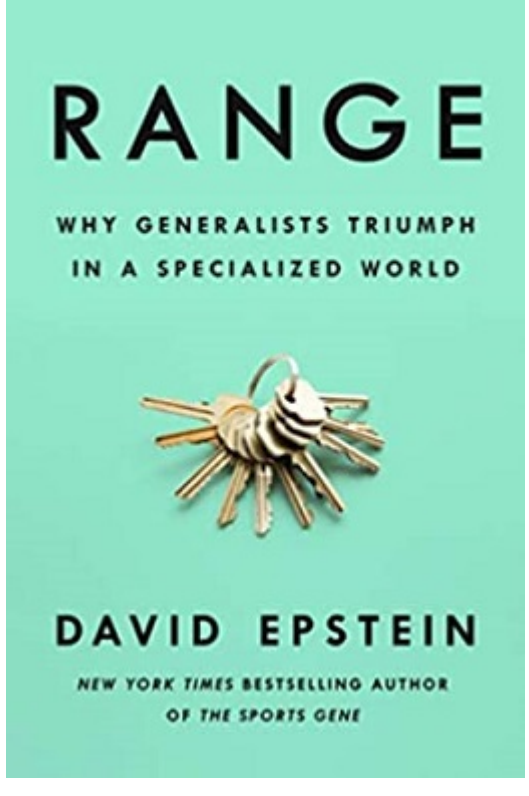
Is that it? Or is there more to it?

Is that it? Or is there more to it?

Hope you did the reading:

Is that it? Or is there more to it?

Hope you did the reading:



One of the ideas advanced in this book is the notion of 'vicious' or 'virtuous' learning environments.

An example of a virtuous learning environment is Chess. Another is Golf

Namely, the feedback you get is high-quality and can be taken at face-value

So, all things being equal, the more you study it, the better you get

The book provides an example of a vicious learning environment, too, which Epstein credits to Robin Hogarth

The book provides an example of a vicious learning environment, too, which Epstein credits to Robin Hogarth

A doctor had a sure-fire method of telling if someone was about to come down with Typhoid fever.

The book provides an example of a vicious learning environment, too, which Epstein credits to Robin Hogarth

A doctor had a sure-fire method of telling if someone was about to come down with Typhoid fever.

He'd feel their patients tongues

Business, it's worth noting, is a vicious learning environment

Business, it's worth noting, is a vicious learning environment

Meaning, things go absurdly wrong

Business, it's worth noting, is a vicious learning environment

Meaning, things go absurdly wrong

So to the suit's point, it's not fair to just say "make it right" and all will be well

Business, it's worth noting, is a vicious learning environment

Meaning, things go absurdly wrong

So to the suit's point, it's not fair to just say "make it right" and all will be well

...and to the devs point, "do people love our app" is not a terribly interesting question if the app is an unusable tire fire

We're confronting another issue, though, and that's best demonstrated with another anecdote

We're confronting another issue, though, and that's best demonstrated with another anecdote

Setting: 1846, Vienna General Hospital

We're confronting another issue, though, and that's best demonstrated with another anecdote

Setting: 1846, Vienna General Hospital

A doctor -- Ignaz Semmelweis -- noticed a strange thing:

We're confronting another issue, though, and that's best demonstrated with another anecdote

Setting: 1846, Vienna General Hospital

A doctor -- Ignaz Semmelweis -- noticed a strange thing:

Two clinics in the hospital handled deliveries, one run by nurses, the other run by doctors.

The one ran by nurses had a much better mortality rate than the one at the University.

The one ran by nurses had a much better mortality rate than the one at the University.

10% for the Doctor-run one vs less than 4% for the nurse-run one.

Doctor Semmelweis noticed this, and started doing systems analysis to figure out why

Doctor Semmelweis noticed this, and started doing systems analysis to figure out why

Was it that the nurses delivered babies on their side?

Doctor Semmelweis noticed this, and started doing systems analysis to figure out why

Was it that the nurses delivered babies on their side?

Was it the priest ringing the bell as he walked to a patient to give last rights?

He got burned out chasing the puzzle, so he took a vacation.

He got burned out chasing the puzzle, so he took a vacation.

He returned to discover a breakthrough!

He got burned out chasing the puzzle, so he took a vacation.

He returned to discover a breakthrough!

A colleague of his died.

He got burned out chasing the puzzle, so he took a vacation.

He returned to discover a breakthrough!

A colleague of his died.

Specifically, he died of the same symptoms as the mothers did, after childbirth.

He got burned out chasing the puzzle, so he took a vacation.

He returned to discover a breakthrough!

A colleague of his died.

Specifically, he died of the same symptoms as the mothers did, after childbirth.

His finger had been cut cut during an autopsy

Now we have a clue - is there something in the bodies that are getting autopsied that hurts people?

Now we have a clue - is there something in the bodies that are getting autopsied that hurts people?

Okay, assume so - maybe you should wash hands after autopsies, before a delivery.

The Doctors weren't happy about the suggestion, but he insisted.

The Doctors weren't happy about the suggestion, but he insisted.

Fatalities started dropping

So, happy ending, right? The riddle was solved, the solution found, heroes ride into the sunset, right?

So, happy ending, right? The riddle was solved, the solution found, heroes ride into the sunset, right?

Well, no.

So, happy ending, right? The riddle was solved, the solution found, heroes ride into the sunset, right?

Well, no.

Doctors didn't like the implication that they were making people sick

So, happy ending, right? The riddle was solved, the solution found, heroes ride into the sunset, right?

Well, no.

Doctors didn't like the implication that they were making people sick

Ignaz was also not the most polite or tactful person around

Put in a blender, pulse until smooth, and he gets:

Put in a blender, pulse until smooth, and he gets:

- Ignored
- Fired
- Sterilization policy was reversed
- Eventually committed to a mental asylum
- (probably/maybe) developed a mental condition, likely connected to syphilis or Alzheimer's
- Died of Sepsis

So. This suggests a grim question.

So. This suggests a grim question.

If we have documented evidence that Doctors, with lives on the line, will sacrifice those lives for the sake of their own egos

So. This suggests a grim question.

If we have documented evidence that Doctors, with lives on the line, will sacrifice those lives for the sake of their own egos

Why do we think software developers will behave any better?

Is it hopeless?

Is it hopeless?

Oh, probably. But there are some things that can be done.

Academia:

Academia:

- These ideas are clearly worth attention. So teach them in undergrad.
- Maybe we don't need to go all the way to depdendently typed effect tracking systems, but

Academia:

- These ideas are clearly worth attention. So teach them in undergrad.
- Maybe we don't need to go all the way to depdendently typed effect tracking systems, but
  - Sum types and Product types
  - Maybe/Either (or Optional/Result) as a solution to the billion dollar mistake
  - HOFs + map/fold/filter as a way to solve off-by-one loop errors
  - We can compromise and make Monadic IO an elective

Dynamic Failure in Mainstream Languages

Solved problems:

- Random memory overwrites
- Memory leaks

Solveable:

- Accessing arrays out-of-bounds
- Dereferencing null pointers
- Integer overflow
- Accessing uninitialized variables

50% of the bugs in Unreal can be traced to these problems!

"The Next Mainstream Programming Language: A Game Developer's Perspcetive", slide 30

Industry:

Industry:

- Invest in continual education for your talent
- "The only thing worse than training your employees, and having them leave, is not training them, and having them stay." -- Henry Ford

Industry:

- Invest in continual education for your talent
- "The only thing worse than training your employees, and having them leave, is not training them, and having them stay." -- Henry Ford
- First-To-Market is not the end-all be-all. It also has to work.

Industry:

- Invest in continual education for your talent
- "The only thing worse than training your employees, and having them leave, is not training them, and having them stay." -- Henry Ford
- First-To-Market is not the end-all be-all. It also has to work.
- It's easier to make things work if you use tools that don't lie to you.

Practitioners:

Practitioners:

- You'll have to invest in your own continued education, too.
- Be ruthless about real, quantifiable behavior of the systems and code you steward

Practitioners:

- You'll have to invest in your own continued education, too.
- Be ruthless about real, quantifiable behavior of the systems and code you steward
  - Is it safe?
  - Is it correct?
  - Is it Fast?
  - Is it a good servant to your customers?
  - Will the heirs of this system be able to manage it?
  - If not, what can be done to make it so?

Everyone:

Everyone:

Tooling isn't a nice-to-have, or something you can stick in an SEP field

Everyone:

Tooling isn't a nice-to-have, or something you can stick in an SEP field

- Rusts tooling proves 'YAGNI' is a cop-out and a lie
- A rust-grade error message only comes about when the team behind it is ruthless about treating "I read the message and I'm still confused" as a show-stopping bug.

We've also forgotten how to discard broken tools.

We've also forgotten how to discard broken tools.

When was the last time you used 'goto'?

We've also forgotten how to discard broken tools.

When was the last time you used 'goto'?

Not to say that we should stop teaching OO

We've also forgotten how to discard broken tools.

When was the last time you used 'goto'?

Not to say that we should stop teaching OO

But there's a pile of known problems, that other paradigms solve, and sending people into the industry unaware of them is irresponsible

We've also forgotten how to discard broken tools.

When was the last time you used 'goto'?

Not to say that we should stop teaching OO

But there's a pile of known problems, that other paradigms solve, and sending people into the industry unaware of them is irresponsible

...if not sadistic

Part of why Haskell got as popular as it did is there is clearly a desperate hunger for something other than 1970s-era language design.

Part of why Haskell got as popular as it did is there is clearly a desperate hunger for something other than 1970s-era language design.

And the tools are available. They've been available for decades, if not close to a century. Sum types are not new.

Part of why Haskell got as popular as it did is there is clearly a desperate hunger for something other than 1970s-era language design.

And the tools are available. They've been available for decades, if not close to a century. Sum types are not new.

But we've lost the ability to evaluate these things with data, rather than politics and fashion

Many things about business (and life) are uncertain, vicious, cruel, and capricious.

Many things about business (and life) are uncertain, vicious, cruel, and capricious.

If you can make something that *delights* someone, you're on good ground

Doing that requires laser focus, and a tool stack that treats you with honesty and respect -- enough to tell you when you're wrong

Doing that requires laser focus, and a tool stack that treats you with honesty and respect -- enough to tell you when you're wrong

Not like Mephistopheles chasing a sales quota.

Further reading: "Worse is Better for Better or Worse" Fin, a rant about stagnant systems level research (with many other excellent articles)

Q&A

Have I stunned you into silence or exisetential despair?