Best Of Curry On 2019

A shareback by Felix Dilke

Pro tip

If you've been to an interesting **event** or **conference**, **share the joy** by pointing out key insights to your colleagues!

Curry On - “An unusual non-profit conference focused on programming languages & emerging challenges in industry”

This threatens to be abstruse, so:

- What is it even about?
- How’s this relevant to SN?
- Just the highlights, please

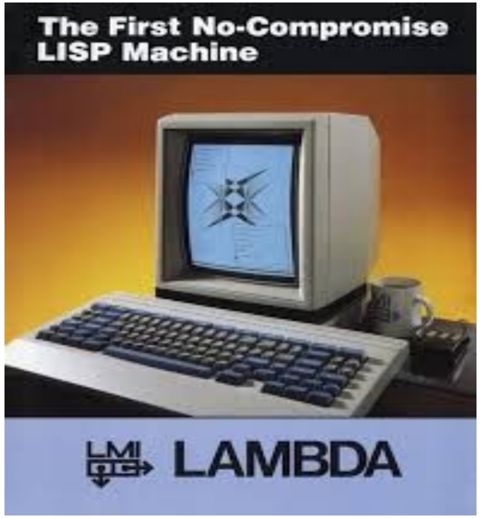
A curious observer asked me:

“If you don’t mind my asking… what is this conference actually about?”

Pro tip

Engage the audience’s attention with a challenging **question, insight** or **observation!**

Human-interest stories are also good.



My hastily improvised answer:

“It’s about languages for programming computers.

People are inventing new ones all the time.

It’s quite exciting.”

Pro tip

Ask a question, then answer it yourself!

This sets up a **rhythm** and **structure** for your presentation.

So there were talks about CARP (a hybrid of Lisp and Rust), Hackett, Idris, Pony, Elm, F#, F*, and WebAssembly.

The talks also covered development/static analysis tools, the history of computing, mobile apps, and cloud computing.

Pro tip

Don't be narrow - emphasize the sheer **breadth, sweep** and **wide applicability** of what you have to say.

Relevant to us how?

In terms of our day-to-day work, these talks seemed particularly to the point:

- ➔ Uber's build automation systems
- ➔ Fundamental principles of software engineering

and in terms of keeping up with the industry:

- ➔ Mozilla: WebAssembly outside the browser

There was also some fun stuff

...which maybe serves to inspire new ideas, provide context, and remind us why we joined the software industry in the first place

- ➔ Mobile development in F#
- ➔ Glamorous Toolkit
- ➔ Idris 2
- ➔ History of Logo

My notes about all these are online...

but here are some quick highlights

Uber’s build automation

They have a Programming Systems team - EE on steroids - focused on reducing the amount of manual effort. Several interesting tools

SubmitQueue: “keep **master** green at scale” - solving a problem that’s particularly chronic for large monorepos/dev teams. It filters and analyses commits to avoid conflicts, taking as input a queue of proposed changes, and generating patches. Takes into account which developer made the change(!)

Piranha: detect and remove unused code - automated cleanup, with smart handling of feature flags. Generates pull requests for manual review. Upcoming version detects stale tests, unused private methods, redundant wrappers.

Fundamental principles of software engineering

In CompSci, there are concepts like "unsolvability of the halting problem" - immutable, fundamental discoveries. In physics, Isaac Newton's "Opticks" (a magisterial 17thC tome about refraction, colour, lenses) is still relevant today.

What equivalents can we offer? Scrum, Agile, UML, still-evolving tools like Travis and Docker 😝

In the uncertain, shifting environment of most software projects, can we say anything certain at all? Can we at least guarantee cost and correctness?

Universities are not vocational (although their prospectuses suggest otherwise) and yet people are expected to emerge from them into industry. How could there be better links between the two?

In the history of software development, programming used to be idiosyncratic and hacky, contrasting painfully with the requirements of real life projects with budgets, scheduling and accountability… particularly in the military. Programmers were seen as expensive, poorly understood, undisciplined, and so on. There seem to be two cultures here, unable even to agree on the problem to be solved.

The NATO 1968 conference spelt this out and introduced the term "the software crisis"

In the 1969 follow-up, no consensus was achieved on anything, and no solutions emerged.

An interesting discussion with more questions than answers.

Mozilla: WebAssembly outside the browser

I hadn't quite realized Mozilla were such a powerhouse: besides Firefox and JavaScript, they also gave the world WebAssembly and Rust.

WebAssembly: enables fast, portable, secure, low-level programming of the browser. Although the Mozilla reps winced at this description, the aim of WA seems to be partly to "deliver the world from JavaScript" and its too-hasty standardization.

Learn from all the mistakes of the JVM / applets (inadequate security model)

Another use case: server-side Node apps: native modules are typically not portable or secure, and the ecosystem isn't adequate to support them.

The Node developers have in fact made a conscious decision NOT to do this, on the grounds that this lack of safety makes sense for some applications - leaving developers in a world of "dangerous toys".

The need is for an interface between Node/JS and WebAssembly: Enter WASI (WebAssembly System Interface), which is a sort of POSIX but for WebAssembly. Use cases include blockchain platforms, portable command-line tools, and 'lightweight sandbox' situations such as game engines.

Not incidentally, Rust includes cross-compilation to WebAssembly as a key feature.

THANK YOU

my notes:

github.com/springernature/felix-notes/tree/master/notes/curry-on

For the online videos:

www.curry-on.org/2019/#program

Below, there are slides about the fun stuff...

Mobile development in F#

F# is Microsoft's answer to Haskell, on the .NET platform - talk was by Don Syme, its principal designer

F# - open source, cross platform language with "immensely practical set of choices for applied FP". The Ionide add-in lets you use it with Visual Code

Using ideas from Elm, there is now Fabulous, a framework that lets you write portable mobile apps in F#, with Xamarin under the hood for cross-compilation.

Ideological discussion about why they decided to use MVU (model-view-update) instead of .NET's more cumbersome and less expressive MVVM (model-view-viewmodel) with its separate markup language Xaml.

To experiment with this, see sample app Elmish Contacts (uses most of the API).

Glamorous Toolkit

(attempts to do justice to this in one slide - somewhat like trying to describe an acid trip)

Open source IDE from software consultancy Feenk.com - based on Pharo

Pharo is a slimmed-down version of Squeak

Squeak is an environment for developing Smalltalk, written in Smalltalk

Smalltalk is a 1980s-flavoured VM-based language ultimately displaced by Java

Squeak/Pharo have amazing development tools

Glamorous Toolkit picks up where they leave off: Moldable Development - everything in the IDE can be viewed / hacked on

Idris 2

Edwin Brady's successor to his language Idris - a cleaned-up version of Haskell, with types as a first class citizen

As with Scala, architectural decision to facilitate language evolution by NOT being 100% compatible - but language is stable, with only minor aesthetic tweaks

Main motivation: Brady realized that the typechecker implementation was very inefficient, dissuading developers from making full use of it

Idris 1 is written in Haskell "because at that point no one knew how to write Idris"; Idris 2 is written in Idris and will eventually be written in Idris 2 i.e. self-hosting.

Better code completion! Done right, it can write half of your app for you.

He also introduces Quantitative Type Theory (QTT) which detects when there are 0 or 1 instances of something and optimises the code accordingly.

Logo much-loved 1960s teaching language in which you could program a physical "turtle" to draw on a 2-d surface

By Cynthia Solomon, computer pioneer who worked with MIT's Minsky and Papert, of Perceptrons fame (early implementation of neural networks) with input from Jean Piaget (child psychologist)

Reaction against rigid Q&A of Patrick Suppes' "Computer Aided Instruction" (CAI).

The idea was to invent a programming language for children, like BASIC but without the confusing "algebraic" syntax. Over the late 1960s, Logo became a complete environment and IDE. Besides moving turtles around, kids could write simple text-based games like Nim. IDE ported to Apple II, inspired Smalltalk.

Talk includes nostalgic montages of modems, teletypes, PDP mainframes, grinning kids writing simple apps where everything is in black and white uppercase.