Vince Casey
caseyv@tcd.ie
18327140

Simon Peyton-Jones

Biography of a Key Software Engineer - Software Engineering

I first learned of Simon Peyton-Jones earlier in the semester while on youtube. At the time, I was having a lot of trouble wrapping my head around Haskell, but also functional languages as a whole. Slowly becoming frustrated, I see a video titled *Simon Peyton Jones - Haskell is useless.* Eager to break up the monotony of dry youtube tutorials, I give in to my curiosity and click on the video.

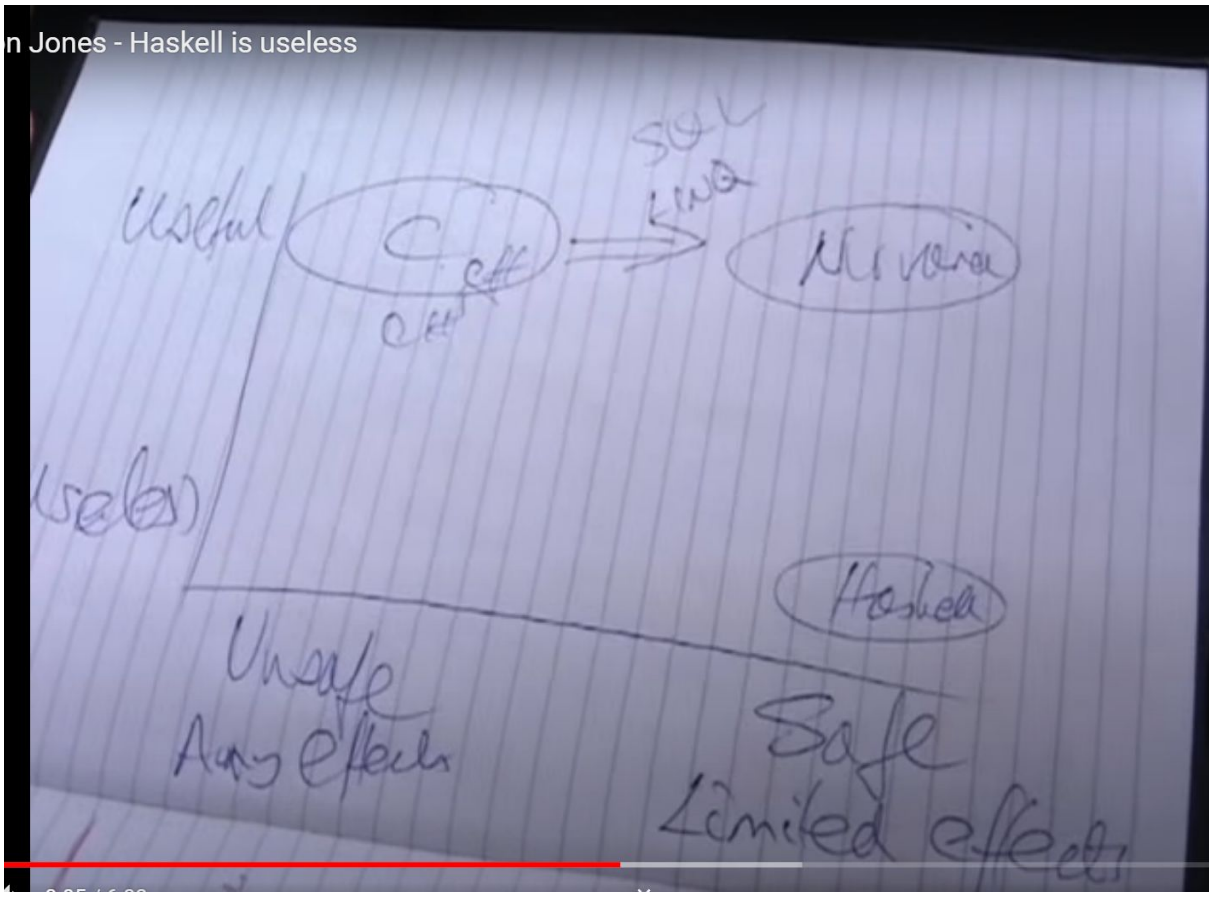
In the video, Jones begins by drawing a graph with 2 axes (Pictured Below). First was the Y axis, marking how useful a language was, then the X axis was a scale of how unsafe or safe a language was. He explained that safeness in this context referred to languages with more limited/controlled side effects. He explained that typical mainstream languages like for example, variants of C, are situated in the top left, making them very useful but also very unsafe. Then, in the opposing corner of the graph he placed haskell, defining it as a language that was “useless” but very safe.

As you might assume, there is a, as Jones calls it, a so-called “nirvana” region of the graph that the perfect language would be in would be a language that was both useful and safe. He explained that over the years, computing has trending more towards safer languages (albeit built up less safe ones) in order to get closer and closer to this perfect language.
This is where Haskell comes in.

Haskell began as an utterly useless language, why run a program if it is not going to have an effect on anything? Well Jones explains that overtime and lots of work, they did in fact manage to get these effectless languages and the regular effectful ones to work together. This has led to the paradigm featured in the graph where from both lands of languages, they can both approach the so-called “Nirvana” language. As they get closer to that goal, we are seeing more and more cross-pollination between the two sides where they take things from one another and use it to optimise or expand their own languages.

This video lit a spark under me and really opened my eyes to the uses and the cutting edge work being done in the world of functional programming. So much so that I have, as the Chair of the Computer Science Society, began plans to try and get him in as a speaker to speak with the college. It wasn’t long before my eyes went elsewhere into the story of both Jones and Haskell itself.

Vince Casey
caseyv@tcd.ie
18327140



(The graph in question)

Jones began studying Mathematics at Trinity College (The Cambridge Trinity College) in 1976. Notably he failed the course and ended up switching to the Electrical Sciences degree. It was here he met John Hughes, another Haskell researcher who notably did not fail his maths degree. They both followed their degrees up with a postgraduate in computer science. The college only had one computer at the time, named Phoenix, of which Jones and Hughes could only use between the hours of 1am and 4am as it was usually reserved for academic use. Here they met Arthur Norman, who introduced them to the world of functional programming, which at the time did not make much sense to them, they were both very much sold on the idea and began their journey into the world of functional programming.

This led to Jones writing his first paper that was published at the 1982 LISP and Functional Programming Conference entitled *Combinators & Language Expressions*. This paper was inspired by the papers of David Turner who took the idea of S K Combinators from Logic and implemented them into functional programming and can then execute. Also around this time, some of the greatest minds in computer science such as John Backus (who had just won the Turing award which is essentially the Nobel Prize for Computer Science) were endorsing the idea of as he described "liberating programming from the von Neumann Style" or in other words, moving away from imperative programming and instead developing the functional

Vince Casey
caseyv@tcd.ie
18327140

programming field. Jones among many others took this as a call to action to continue pursuing research in their field and developing new languages in functional programming and creating machines that could run them.

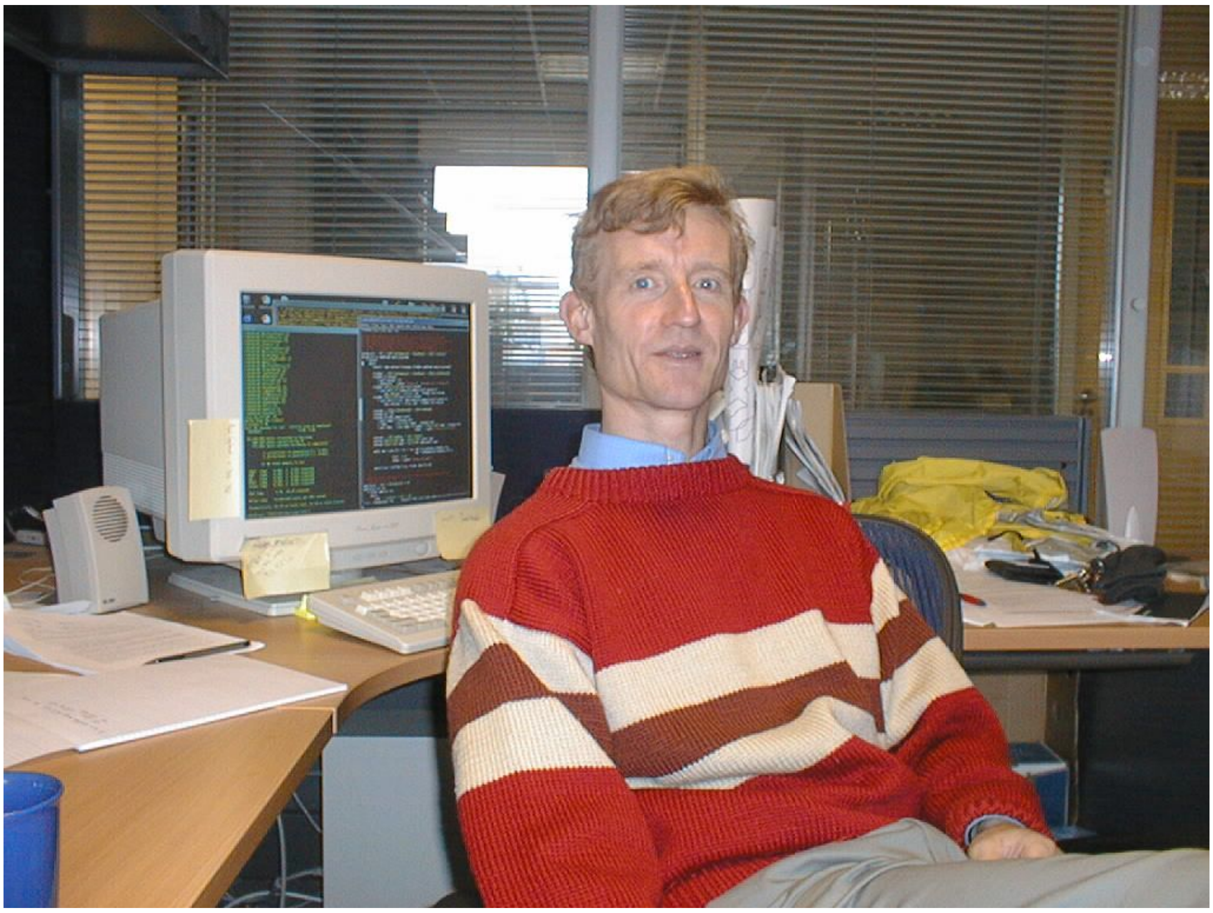
This all led to a period of what Jones describes as chaos where there was a huge period of growth with lots of new languages, developments and conferences. Then, in 1987 at the FPCA Conference (Functional Programming & Computer Architecture), leading minds in both the strict functional languages community (mainly based in edinburgh, the more established group) and the lazy functional programmers (Jones & co.) met and decided to create a common syntax or language in order to help the different groups work together much more effectively. They thought this would be a small task, taking at most 6 months, but it ended up taking 2 and a half years! After this was done, the Haskell 1.0 Report was ready and published in 1990.

In 1992, Jones alongside Phil Wadler, published the paper *Imperative Functional Programming* which brings us back to the graph mentioned earlier. Imperative programming languages are the standard ones we use mostly in the industry or to frame it like we did before, the “unsafe” but “useful” languages. This paper was revolutionary as it showed that Haskell could be as Jones says “The World's Finest Imperative Programming Language.” They were able to achieve this fusion of the two worlds thanks to Wadlers knowledge of Monads from the more theoretical side of Computer Science along with Jones based in Haskell. This allowed them to mix the two types of programming with very clear boundaries which enabled the haskell to stay “pure” or as described earlier “safe.”

It is hard to tell the full extent of Haskell’s impact as it is an open source language. However one statistic shows that on stack overflow and other similar sites, Haskell is the 5th most talked about language! Now, whether that’s because Haskell really is that popular or simply because functional programming is impossible to use, I really cannot say. Jones’ commitment to Haskell was so established in his family that they even named their cat *Haskell* after the language.

Throughout Jones’ career he has worked on many other things too of course. He co-created the C-- programming language with Norman Ramsey which rather than be designed to be used by humans, is instead designed to be generated by compilers for very high-level languages. He also lectured in both University College London and at the University of Glasgow. To this day, Jones is one of the heads behind GHC (The Glasgow Haskell Compiler) as well as a researcher at Microsoft Research in Cambridge. He was also elected Chair of the National Centre for Computing Education in 2019 which seeks to offer education in computer science to primary, secondary and third level education students. They work with both teachers and students and do both face to face and online teaching.

Vince Casey
caseyv@tcd.ie
18327140



(The man himself)