CHAPTER

7

Simon Peyton Jones

One of the instigators, back in 1987, of the project that led to the definition of the programming language Haskell, Simon Peyton Jones is a Principal Researcher at Microsoft Research's lab in Cambridge, England. He edited the Haskell 98 Revised Report, *the current stable definition of the language; he is the architect and lead developer of the Glasgow Haskell Compiler (GHC), the "de facto standard compiler" according to haskell.org; and he gave Haskell its widely cited unofficial motto: "Avoid success at all costs."*

A high-powered researcher and former professor who never got a PhD, Peyton Jones values both the practical and the theoretically beautiful. He learned to program on a machine with no permanent storage and only 100 memory locations, and in college he worked on both writing high-level compilers for the school's big iron and building his own primitive computers out of parts he could afford on a student's budget. But he was drawn to functional programming by a professor's demonstration of how to build doubly linked lists without using mutation and the beauty of the idea of lazy evaluation. Peyton Jones saw the ideas of functional programming "as a radical and elegant attack on the whole enterprise of writing programs": a way, rather than "just putting one more brick in the wall," to "build a whole new wall." In 2004 the Association for Computing

Machinery elected him a Fellow, citing his "contributions to functional programming languages."

Among the topics we covered in this interview are why he thinks functional programming shows increasing promise of changing the way software is written, why Software Transactional Memory is a much better way of writing concurrent software than locks and condition variables, and why it is so difficult, even at a place like Microsoft Research, to do real studies of whether different programming languages make programmers more or less productive.

Seibel: When did you learn to program?

Peyton Jones: When I was at school. Intel had just about produced the 4004—the world's first microprocessor. We didn't have a 4004 or anything like it—it was really a chip that hobbyists could barely get at that stage. The only computer they had available was an IBM schools computer, which was a strange machine built out of spare parts from mainframes. It had no permanent storage whatsoever so you had to type in your program every time you ran it.

It had 100 storage locations, total, which would each store, I think, eight-digit decimal numbers. And this stored both your program and your data. So the name of the game of programming that was to simply to fit the program into 100 storage locations. I can't quite remember how I got to write my first program. I think I and one other enthusiast at the school spent a lot of time on the schools computer. This would have been when I was about 15, 1974, '73—that kind of era.

Then after we'd been programming this machine for a little bit we discovered there was a computer at the technical college in Swindon. So we spent an hour on a very slow bus one afternoon a week and went to Swindon where there was this enormous machine—an Elliot 803—which lived in half a dozen large, white, fridge-sized cabinets in a room all its own with a white-coated operator.

After a bit the white-coated operator learned that we could figure out how to use the machine so she went away while we played with this vast engine.