FP101x - Functional Programming

Programming in Haskell - Introduction

Erik Meijer

The Software Crisis

How can we cope with the size and complexity of modern computer programs?

How can we reduce the time and cost of program development?

How can we increase our confidence that the finished programs work correctly?

Programming Languages

One approach to the software crisis is to design new programming languages that:

- Allow programs to be written clearly, concisely, and at a high-level of abstraction
- Support reusable software components
- Encourage the use of formal verification

Permit rapid prototyping;

Provide powerful problem-solving tools.

Functional languages provide a particularly elegant framework in which to address these goals.

What is a Functional Language?

Opinions differ, and it is difficult to give a precise definition, but generally speaking:

- Functional programming is style of programming in which the basic method of computation is the application of functions to arguments;

- A functional language is one that supports and encourages the functional style.

Example

Summing the integers 1 to 10 in Java:

```
total = 0;
for (i = 1; i ≤ 10; ++i)
    total = total+i;
```

The computation method is variable assignment.

Example

Summing the integers 1 to 10 in Haskell:

```
sum [1..10]
```

The computation method is function application.

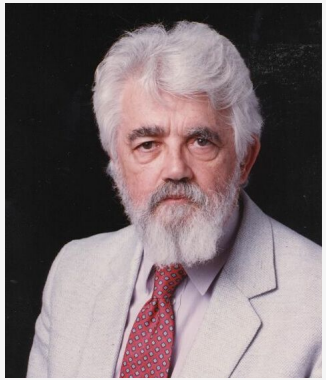
Historical Background

1930s:

Alonzo Church develops the lambda calculus, a simple but powerful theory of functions.

Historical Background

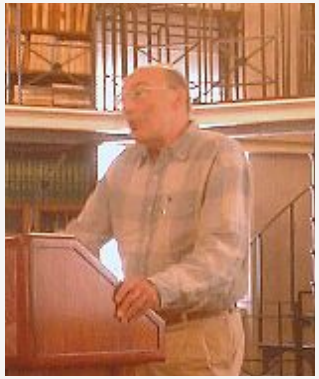
1950s:



John McCarthy develops Lisp, the first functional language, with some influences from the lambda calculus, but retaining variable assignments.

Historical Background

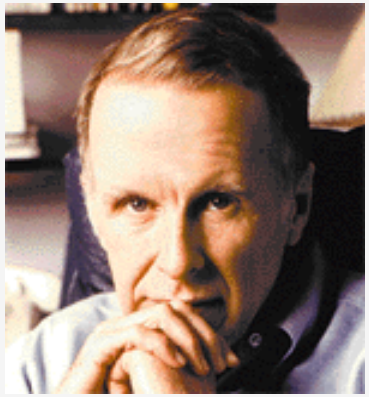
1960s:



Peter Landin develops ISWIM, the first *pure* functional language, based strongly on the lambda calculus, with no assignments.

Historical Background

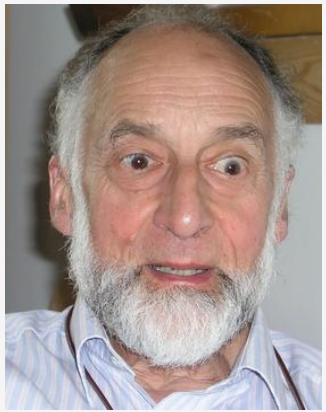
1970s:



John Backus develops FP, a functional language that emphasizes *higher-order functions* and *reasoning about programs*.

Historical Background

1970s:



Robin Milner and others develop ML, the first modern functional language, which introduced *type inference* and *polymorphic types*.

Historical Background

1970s - 1980s:

David Turner develops a number of *lazy* functional languages, culminating in the Miranda system.

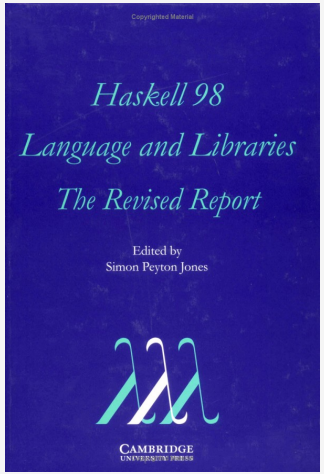
Historical Background

1987:

An international committee of researchers initiates the development of Haskell, a standard lazy functional language.

Historical Background

2003:



The committee publishes the Haskell 98 report, defining a stable version of the language.

Historical Background

2003-date:

Standard distribution, library support, new language features, development tools, use in industry, influence on other languages, etc.

A Taste of Haskell

```
f []     = []
f (x:xs) = f ys ++ [x] ++ f zs
           where
              ys = [a | a ← xs, a ≤ x]
              zs = [b | b ← xs, b > x]
```

?

Happy Hacking!