PROGRAMMING IN HASKELL

Chapter 1 - Introduction

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:

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

The computation method is variable assignment.

Example

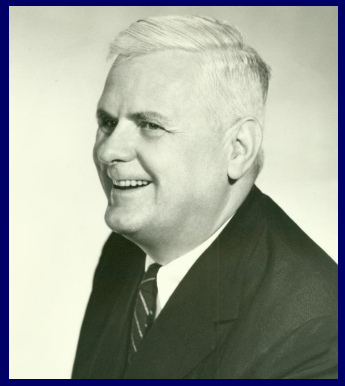
Summing the integers 1 to 10 in Haskell:

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

The computation method is function application.

Historical Background

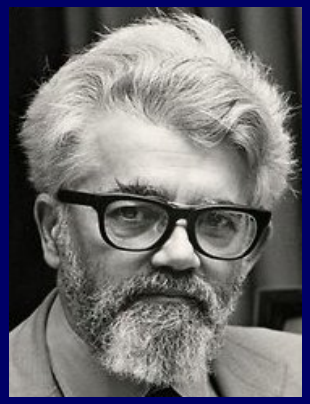
1930s:



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

Historical Background

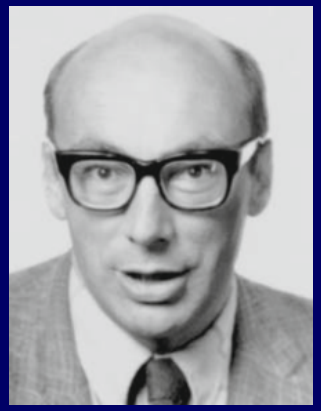
1950s:



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

Historical Background

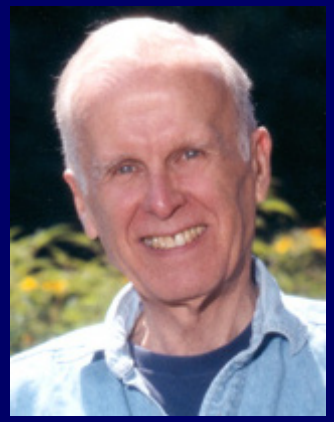
1960s:



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 starts the development of Haskell, a standard lazy functional language.

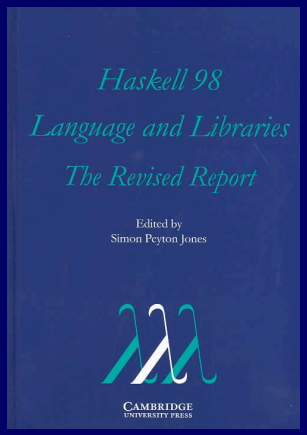
Historical Background

1990s:

Phil Wadler and others develop *type classes* and *monads*, two of the main innovations of Haskell.

Historical Background

2003:



The committee publishes the Haskell Report, defining a stable version of the language; an updated version was published in 2010.

Historical Background

2010-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]
```

?