

CAPHYON ⚡ LIGHTNING TALKS

FP in 10

July, 2019
Craiova



Victor Ciura
Technical Lead, Caphyon
www.caphyon.ro

Functional Programming

What is it all about ?



pipelines

ranges

optional

IO monad

Maybe | Just

algorithms

lifting

monoids

lambdas & closures

fold

values types

lazy evaluation

declarative vs imperative

monads

algebraic data types

map

higher order functions

composition

pattern matching

FP

expressions vs statements

pure functions

currying

category theory

recursion

partial application

Paradox of Programming

Machine/Human impedance mismatch:

- **Local/Global** perspective
- **Progress/Goal** oriented
- **Detail/Idea**
- **Vast/Limited** memory
- **Pretty reliable/Error prone**
- **Machine language/Mathematics**


Is it easier to think like a machine than to do math?

Semantics

- The meaning of a program
- Operational semantics: local, progress oriented
 - Execute program on an abstract machine in your brain
- Denotational semantics
 - Translate program to math
- Math: an ancient language developed for humans

What is Functional Programming ?

- Functional programming is a **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*

Let's address the  in the room...

 Haskell

A functional language is one that supports and encourages the **functional style**

What do you mean ?

Summing the integers 1 to 10 in C++/Java/C#

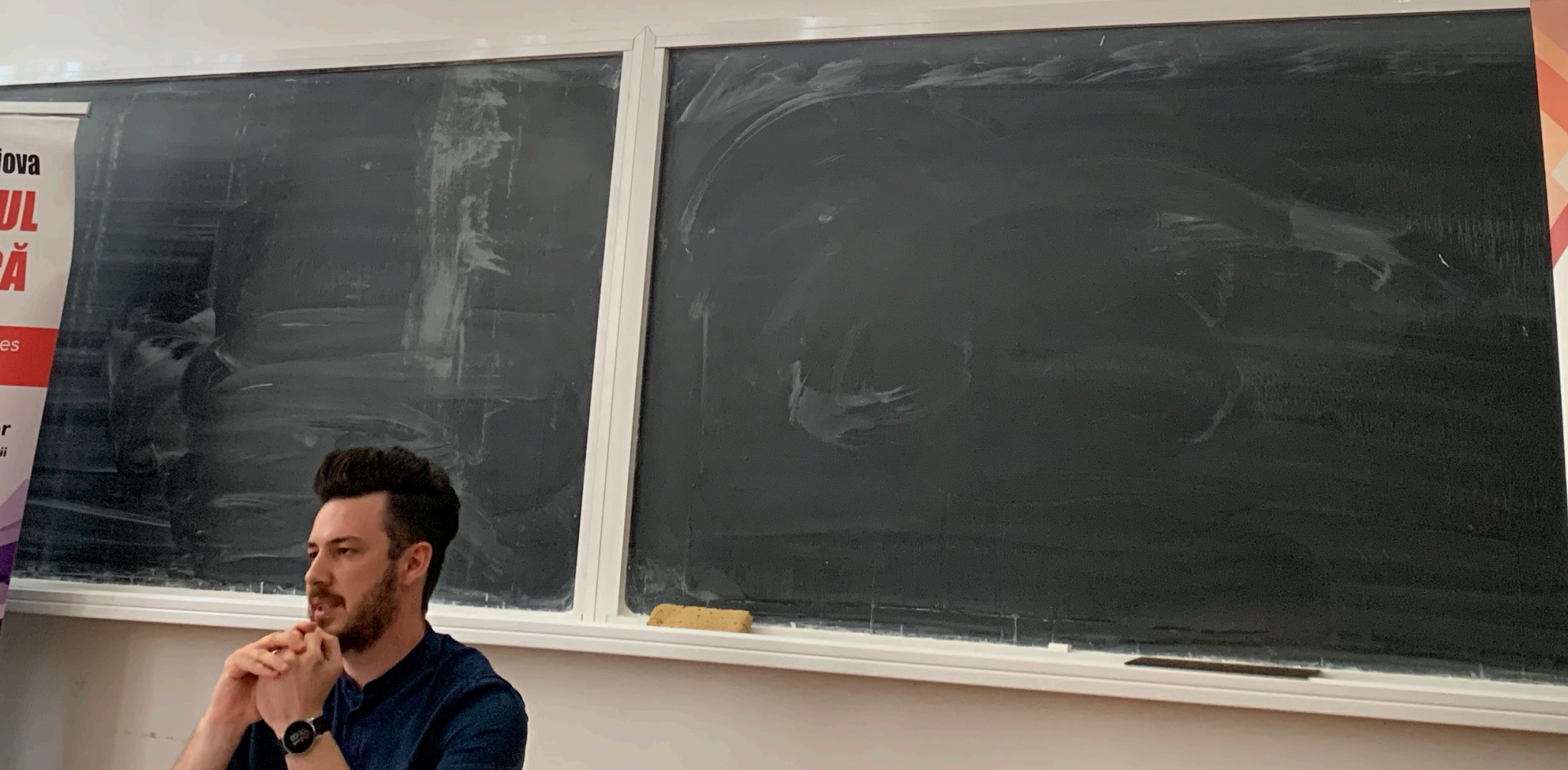
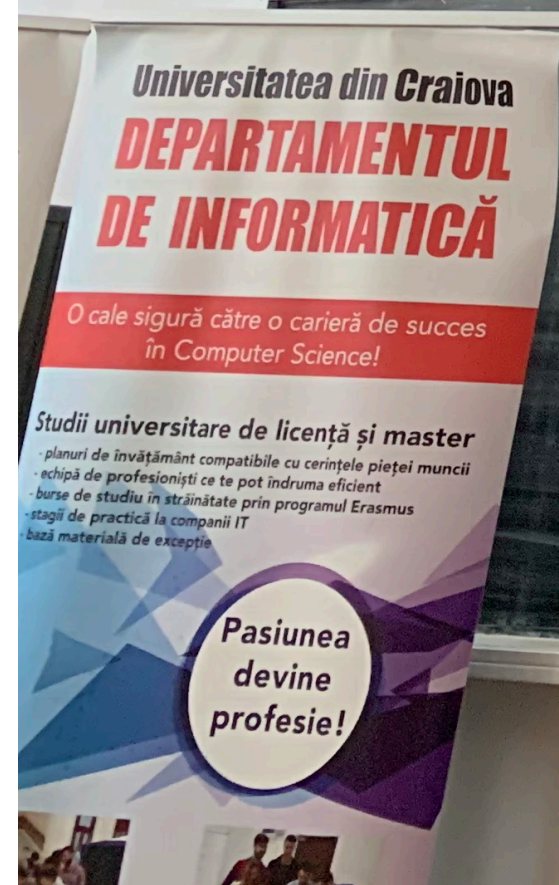
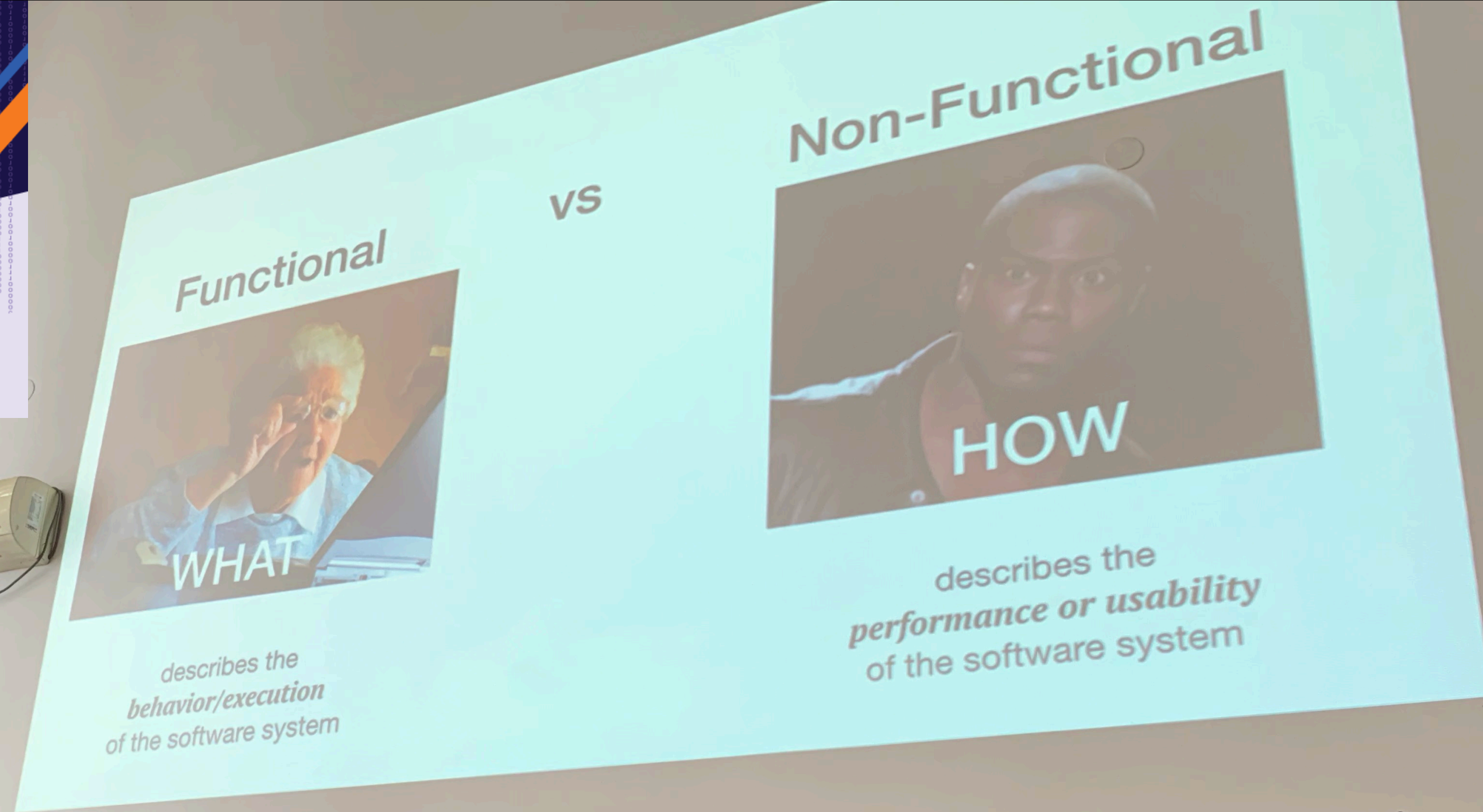
```
int total = 0;  
for (int i = 1; i ≤ 10; i++)  
    total = total + i;
```

The computation method is **variable assignment**.

Summing the integers 1 to 10 in **Haskell**

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

The computation method is **function application**.



Sneak Peek Into Next Level QA (Test Automation) - Antonio Valent

Historical Background



Historical Background

Most of the "new" ideas and innovations in modern programming languages are actually very old...



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

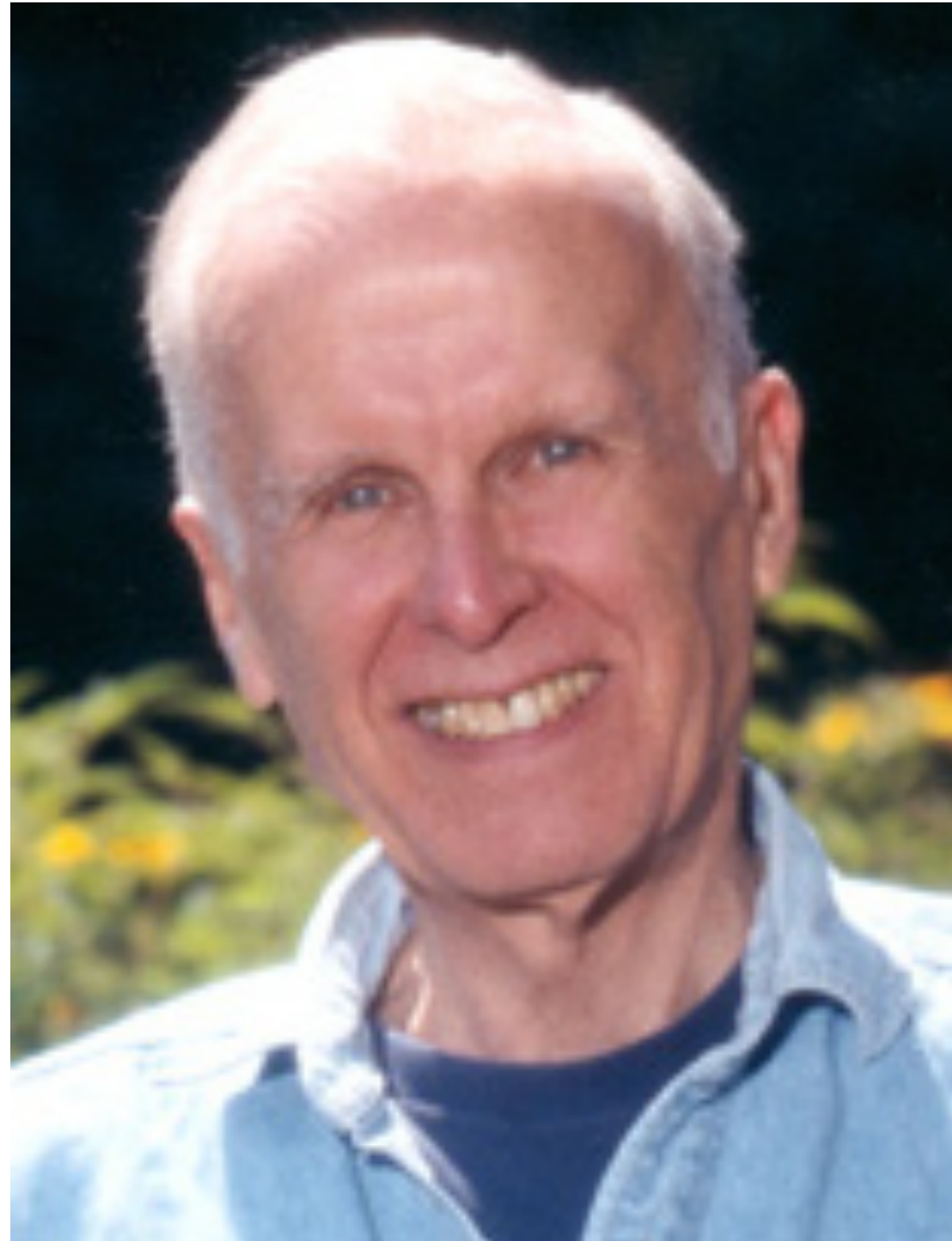
1960s



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

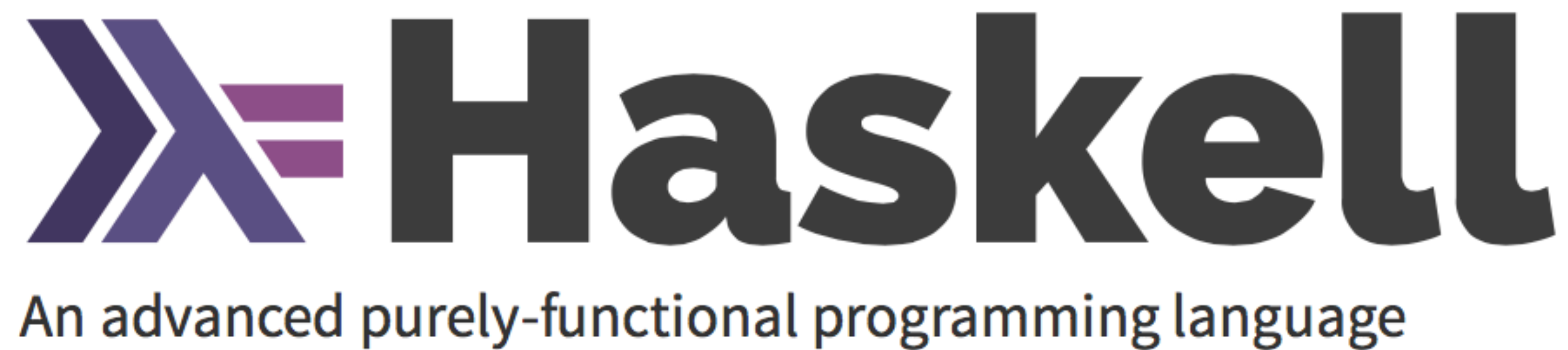
1970-80s



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

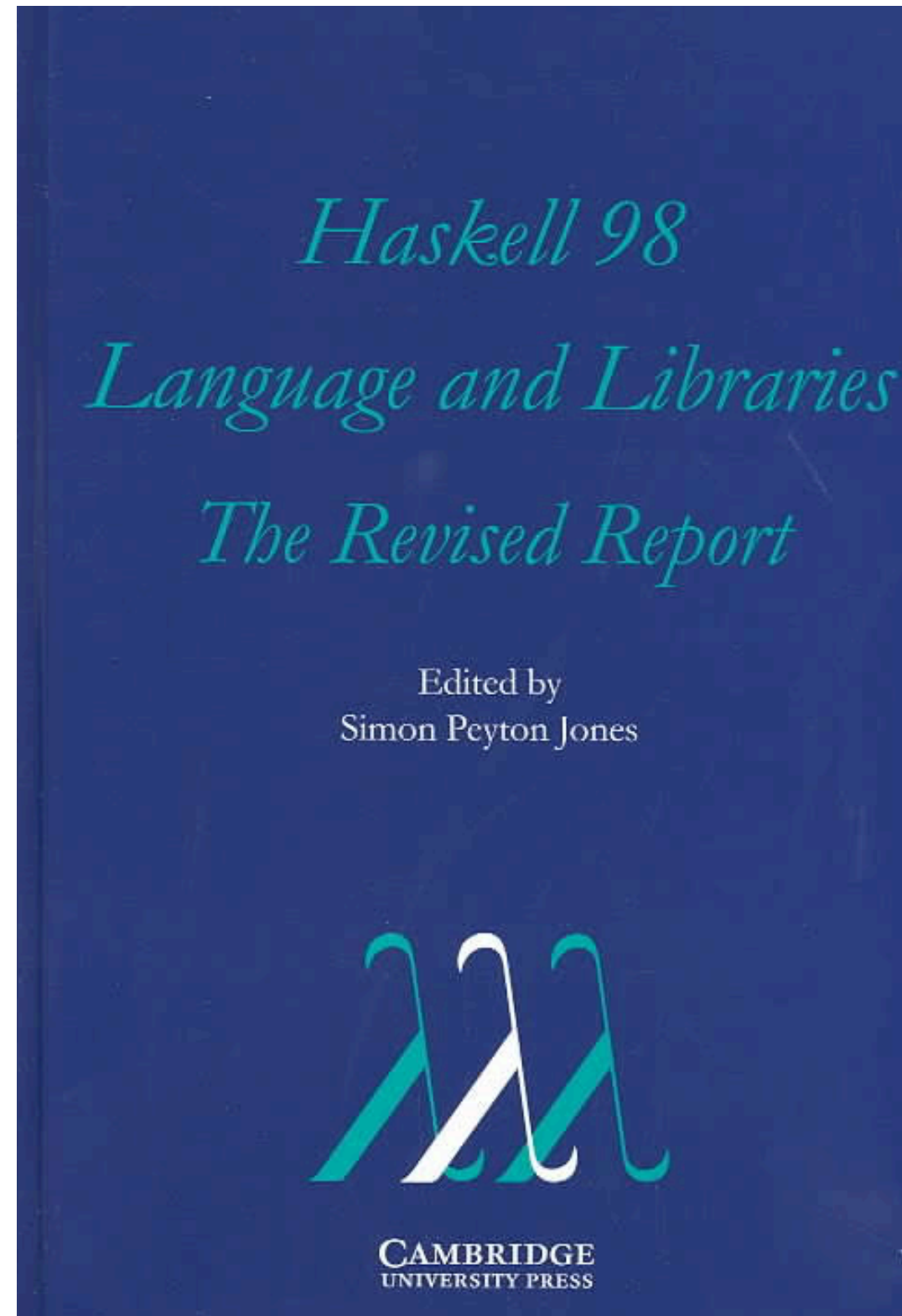


Phil Wadler and others develop **type classes** and **monads**,
two of the main innovations of Haskell

Historical Background

2003

2010



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

A Taste of Haskell

$$\begin{aligned} f [] &= [] \\ f (x:xs) &= f ys ++ [x] ++ f zs \\ &\text{where} \\ &\quad ys = [a \mid a \leftarrow xs, a \leq x] \\ &\quad zs = [b \mid b \leftarrow xs, b > x] \end{aligned}$$

What does **f** do ?

Quick Sort

`qsort :: Ord a => [a] -> [a]`

`qsort [] = []`

`qsort (x:xs) =`

`qsort smaller ++ [x] ++ qsort larger`

where

`smaller = [a | a ← xs, a ≤ x]`

`larger = [b | b ← xs, b > x]`

Quick Sort

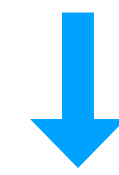
q [3,2,4,1,5]



q [2,1] ++ [3] ++ q [4,5]



q [1] ++ [2] ++ q [] q [] ++ [4] ++ q [5]



[1]

[]

[]

[5]

Quick Sort

```
void quickSort(arr[], low, high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[pi] is now
           at right place */
        pi = partition(arr, low, high);

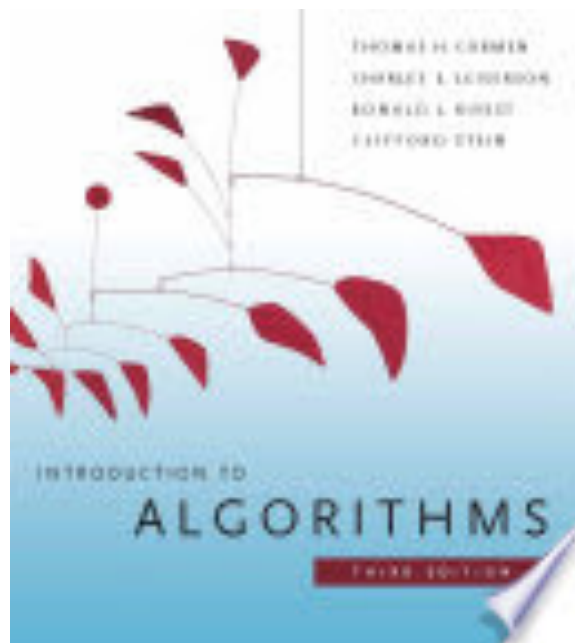
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

```
/* This function takes last element as pivot, places
   the pivot element at its correct position in sorted
   array, and places all smaller (smaller than pivot)
   to left of pivot and all greater elements to right
   of pivot */
```

```
partition (arr[], low, high)
{
    // pivot (Element to be placed at right position)
    pivot = arr[high];

    i = (low - 1) // Index of smaller element

    for (j = low; j <= high- 1; j++)
    {
        // If current element is smaller than or
        // equal to pivot
        if (arr[j] <= pivot)
        {
            i++; // increment index of smaller element
            swap arr[i] and arr[j]
        }
    }
    swap arr[i + 1] and arr[high]
    return (i + 1)
}
```



pseudo-code

True Story

1986:

Donald Knuth was asked to implement a program for the "*Programming pearls*" column in the **Communications of ACM** journal.

The task:

Read a file of text, determine the n most **frequently used words**, and print out a sorted list of those words along with their frequencies.

His solution written in **Pascal** was **10 pages** long.

True Story

Doug McIlroy

His response was a 6-line shell script that did the same:

```
tr -cs A-Za-z '\n' |  
tr A-Z a-z |  
sort |  
uniq -c |  
sort -rn |  
sed ${1}q
```

It's all about pipelines

Taking inspiration from **Doug McIlroy's UNIX shell script**,
write an algorithm in **your favorite programming language**,
that solves the same problem: **word frequencies**



Historical Background

1990s



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

Takeaway



"Make your code readable.
Pretend the next person who looks at your
code is a psychopath and they know where
you live."

Phil Wadler

CAPHYON ⚡ LIGHTNING TALKS

FP in 10

July, 2019
Craiova



Victor Ciura
Technical Lead, Caphyon
www.caphyon.ro