Functional programming introduction: What, why, how?

Architecture Forum, IBSO, 05 Mar 2020
DJ Adams

About me

Developer Advocate in the Developer Relations & Community Org (T&I)

Hacking on SAP software since 1987
Open Source and SAP communities
O'Reilly and SAP Press author

Current focus on SAP Cloud Platform as Business Technology Platform

Live streaming in the series "Hands-on SAP dev with qmacro"
Writing on blogs.sap.com, qmacro.org and langram.org

currying

lambda calculus

ML

higher-order functions

Haskell

recursion

Lisp

point-free coding

closures

referential transparency

functions as values

pure functions

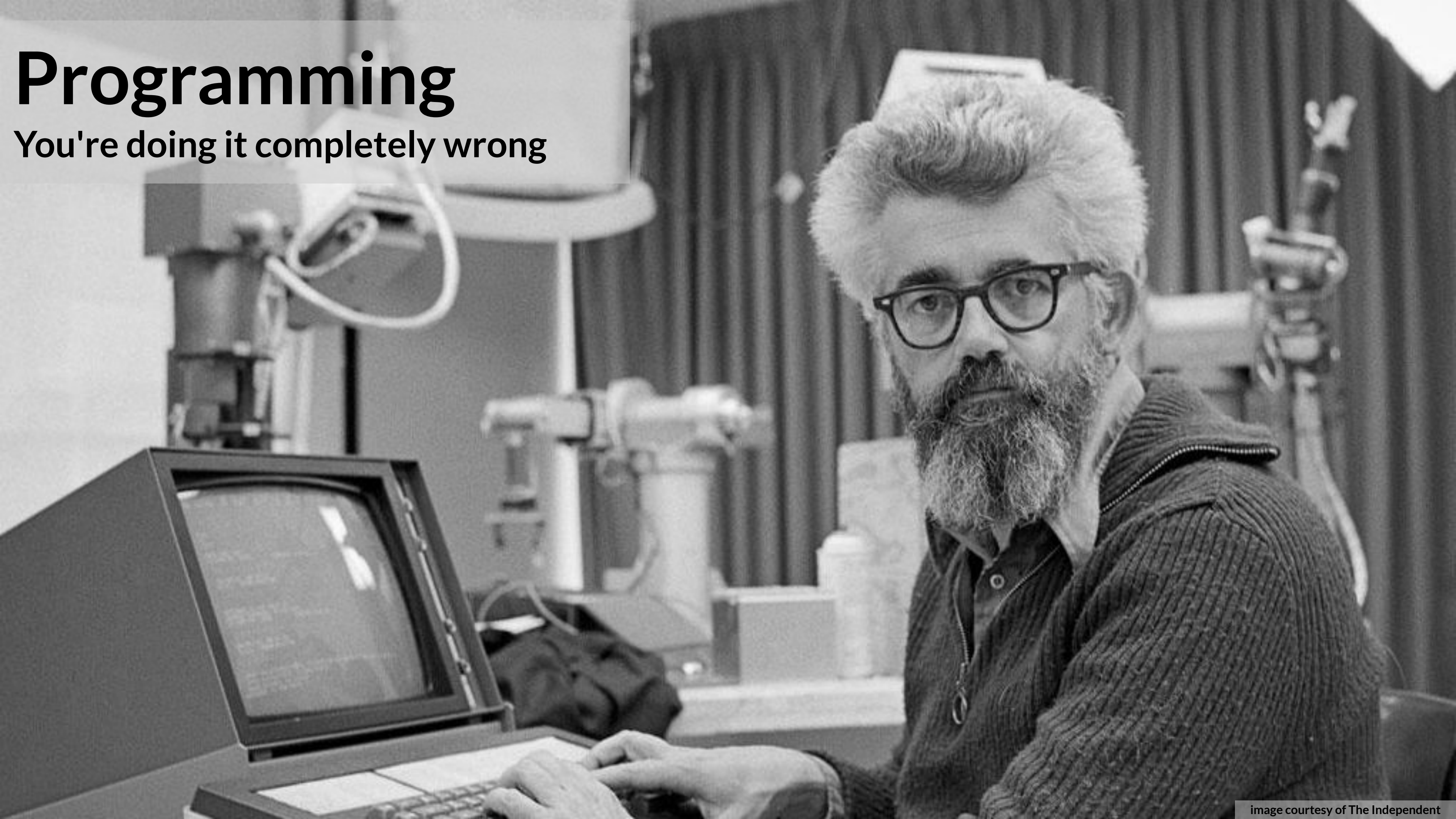
composability

Clojure

immutability

catamorphisms

SICP: Ableson, Sussman & Sussman



Programming

You're doing it completely wrong

image courtesy of The Independent

Agenda

What is it and what are some of the key concepts?

Why is it relevant to me?

How do I get started and how does it feel?

Where can I find out more?

```haskell
sum :: Num a => [a] -> a
sum []       = 0
sum (x:xs) = x + sum xs
```

What

Origins in Lambda Calculus

Computation as the evaluation of functions

Declarative programming paradigm

Avoidance of side effects and mutations

Focus on pure functions & referential transparency

Treats data as a first class citizen

Programming paradigms

	Imperative	**Object Oriented**	**Functional**
ABAP	Y	Y	
C	Y		
C++	Y	Y	
Haskell			Y
Clojure			Y
C#	Y	Y	
Java	Y	Y	
JavaScript	Y	Y (prototypal)	

Programming paradigms

	Imperative	**Object Oriented**	**Functional**
ABAP	Y	Y	Y (7.40SP5)
C	Y		
C++	Y	Y	Y (V11)
Haskell			Y
Clojure			Y
C#	Y	Y	Y (V3)
Java	Y	Y	Y (V8)
JavaScript	Y	Y (prototypal)	Y (always!)

Why

Broader horizon (avoiding the hammer and nail problem)

Fewer moving parts

Solid state

Reduced surface area for bugs to appear

Small pieces loosely joined

Easier to reason about

Higher level of abstraction (e.g. list machinery)

I M IN UR CLOUDZ
RUNNIN UR DATAZ

JS

Many functional programming aspects

More constructs & syntactic sugar with ES2015 / ES6

A first class language in the SAP ecosphere

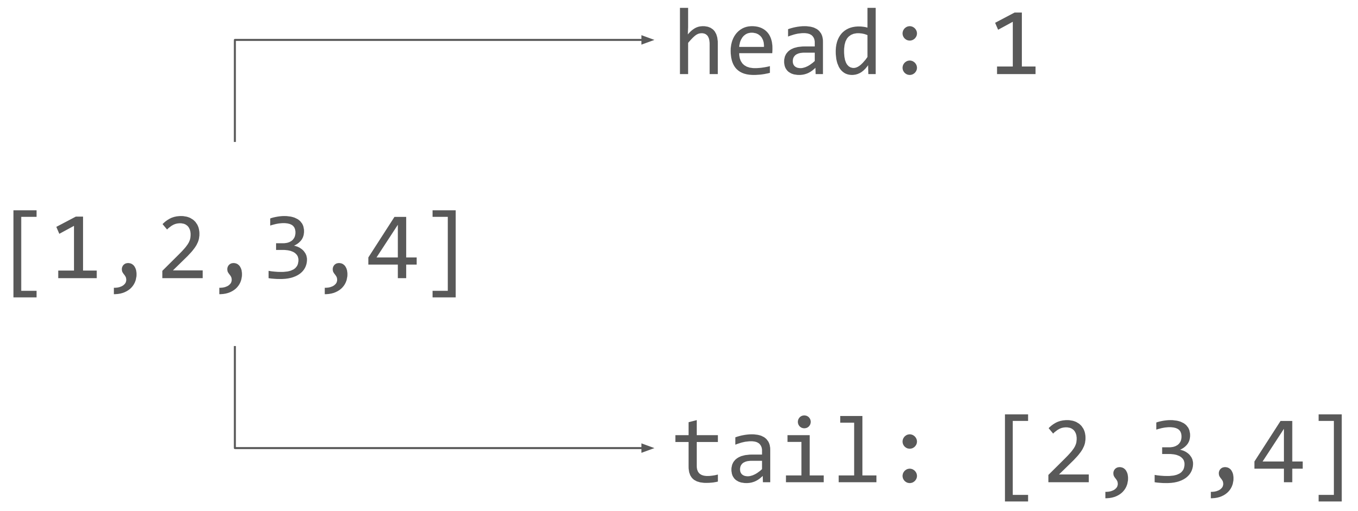
SAP Fiori on the frontend via UI5, backend with Node.js

A key runtime in our Cloud Foundry environment

JavaScript is here today and in your future also (also, remember Atwood's Law!)

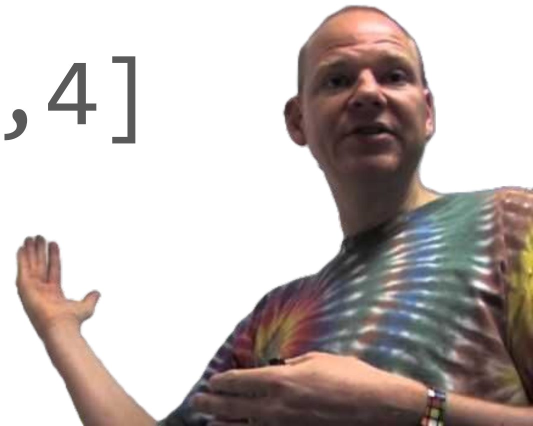
```haskell
sum :: Num a => [a] -> a
sum []       = 0
sum (x:xs) = x + sum xs
```

$$\begin{tikzcd}
& \texttt{car: 1} \\
\texttt{[1,2,3,4]} \arrow[ur] \arrow[dr] & \\
& \texttt{cdr: [2,3,4]}
\end{tikzcd}$$


head: 1
[1,2,3,4]
tail: [2,3,4]

$$\begin{tikzcd}
 & \text{first: 1} \\
{[1,2,3,4]} \arrow[ur] \arrow[dr] & \\
 & \text{rest: [2,3,4]}
\end{tikzcd}$$

$$\begin{tikzcd}
 & \texttt{x: 1} \\
\texttt{[1,2,3,4]} \arrow[ur] \arrow[dr] & \\
 & \texttt{xs: [2,3,4]}
\end{tikzcd}$$



```haskell
sum :: Num a => [a] -> a
sum []       = 0
sum (x:xs) = x + sum xs
sum :: Num a => [a] -> a
sum []     = 0
sum (x:xs) = x + sum xs

sum [1,2,3,4]
1 + (sum [2,3,4])
1 + (2 + (sum [3,4]))
1 + (2 + (3 + (sum [4])))
1 + (2 + (3 + (4 + (sum []))))
1 + (2 + (3 + (4 + (0))))
10
```

```haskell
sum :: Num a => [a] -> a
sum []       = 0
sum (x:xs) = x + sum xs
```

```
product :: Num a => [a] -> a
product []     = 1
product (x:xs) = x * product xs

product [1,2,3,4]
24
```

```
sum []             = 0
sum (x:xs)         = x + sum xs

product []         = 1
product (x:xs) = x * product xs

and []             = True
and (x:xs)         = x && and xs
```

Image courtesy of tolkeingateway.net

```
fold :: (a -> b -> b) -> b -> [a] -> b
fold f v []     = v
fold f v (x:xs) = f x (fold f v xs)

fold (+) 0 [1,2,3,4]
10
fold (*) 1 [1,2,3,4]
24
fold (&&) True [True, True, False, True]
False
```

Array.prototype.reduce()

From the Mozilla Developer Network docs:

The reduce() method executes a reducer function (that you provide) on each element of the array, resulting in a single output value.

Syntax:

```
arr.reduce(callback( accumulator, currentValue[, index[, array]] )[, initialValue])
```

How does it feel?

```
SLMIN .SYS    12P 20-Dec-85    VM    .SYS    3P 13-Aug-86
XL    .SYS     4P 20-Dec-85    LD    .SYS    8P 23-Aug-86
SP    .SYS     6P 13-Aug-86    DL    .SYS    5P 13-Aug-86
RT11SJ.SYS    78P 13-Aug-86    DU    .SYS    8P 13-Aug-86
NL    .SYS     2P 13-Aug-86    TT    .SYS    2P 13-Aug-86
SQ    .SYS     5P 31-May-85    RL02DG.SYS   71P 21-Nov-84
BASIC .SAV    56  24-May-79    BINCOM.SAV   24  20-Dec-85
DATIME.SAV     4  20-Dec-85    DIR   .SAV   19  20-Dec-85
DUMP  .SAV     9  20-Dec-85    DUP   .SAV   47  20-Dec-85
TSXMOD.SAV    78  27-Nov-92    FORTRA.SAV  206  21-May-85
HARRIS.SAV    41  12-Jun-85    LET   .SAV    5  20-Dec-85
START .P36     2  21-Dec-91    RETRO .OBJ 1519P 16-May-88
EM1A  .STM    19P 02-Feb-93    TSXUCL.TSX   22  07-Mar-92
STAND .LIN    12P 15-Aug-83    TSXV6 .MSG    1P 04-Sep-95
EM1B  .STM    19  11-Feb-93    JKFLIP.SAV   30  08-Mar-96
JKFLIP.FOR     3  08-Mar-96    RT11FB.SYS   93P 20-Dec-85
ANNOT .SAV    38  18-Apr-93    TSXP23.NEW 1200P 27-Nov-92
DUO   .DIR    18  16-Jul-96    DL    .DIR    7  16-Jul-96
DIR   .DIF    26  16-Jul-96    DEMOFG.OBJ    1    -BAD-
DEMOBG.OBJ     1    -BAD-      EVAN  .ID     1
 114 Files, 5949 Blocks
 14433 Free blocks

.o█demo time
```

Image courtesy of Wikimedia Commons

Where can I find more info?

map, filter, reduce (MDN reference)

Programming in a more functional style in JavaScript (article)

Functional programming in JavaScript (video playlist)

Ramda (functional library for JavaScript) (REPL link)

Hey Underscore, You're Doing It Wrong! (video)

Language Ramblings (blog)

Functional Programming Fundamentals - Dr. Erik Meijer (video playlist)

*For more information, please reread