DEV219:
Building More Stable Business Apps with Functional Techniques in JavaScript

DJ Adams, SAP Mentor & Principal, Bluefin Solutions
SAP TechEd EMEA, Barcelona 08 Nov 2016

An Introduction to Functional Techniques in JavaScript for UI5

DJ Adams, SAP Mentor & Principal, Bluefin Solutions
UI5con, Eindhoven 25 Nov 2016

Programming

You're doing it completely wrong

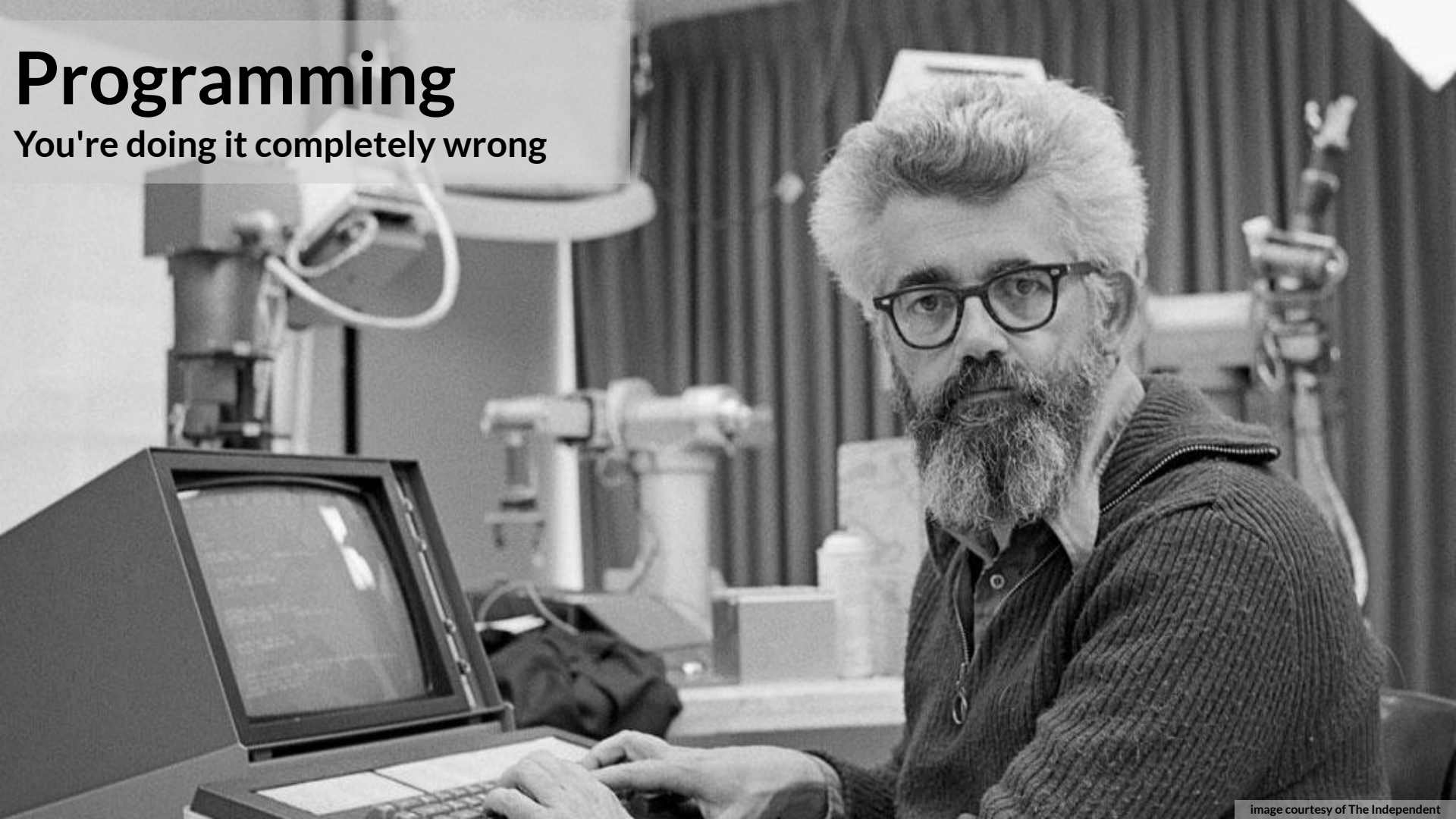

image courtesy of The Independent

Agenda

What is functional programming?

Why is it relevant to me?

What are some of the key concepts?

Let the dog see the rabbit!

Where could I apply functional techniques?

Where can I find out more?

What is functional programming?

A brief overview

Origins in Lambda Calculus

Computation as the evaluation of functions

Declarative programming paradigm

Avoidance of side effects and state changes

Focus on pure functions & referential transparency

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 is it relevant to me?

JavaScript today

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

JavaScript is here today and in your future also!

What are some of the key concepts?

Functions are values

First-class functions

Higher-order functions

Pure functions & referential transparency

Immutability

Composability

Function chaining

Closures

Recursion

Currying

Point-free code

What should I focus on?

Small pieces loosely joined

Avoid mutating state

Think in terms of lists and function chaining

Understand the power of closures

Practise using the higher-order functions **filter**, **map** and **reduce**

Let the dog see the rabbit!

Demo

Take data from UI5 Explored

Introduce filter, map and reduce

Grok function composition and chaining

Compare with imperative style

Rejoice!

Where could I apply these concepts?

Demo

Examine existing SAP app

Look at how data is processed there

Understand what is being done

Rewrite the section with a functional approach

Understand how this makes for a more robust codebase

/test-resources/sap/m/demokit/cart/view/Product.controller.js

Imperative

```
// find existing entry for product
var oEntry = null;
for (var i = 0 ; i < aCartEntries.length ; i ++) {
	if (aCartEntries[i].ProductId === oProduct.ProductId) {
		oEntry = aCartEntries[i];
		break;
	}
}

if (oEntry === null) {
	// create new entry
	oEntry = {
		Id : jQuery.sap.uid(),
		Quantity : 1,
		...
		PictureUrl : oProduct.PictureUrl
	};
	oCartData.entries[oCartData.entries.length] = oEntry;

} else {
	// update existing entry
	oEntry.Quantity += 1;
}
```

Functional

```
// find existing entry for product
let oEntry = aCartEntries.find(
	x => x.ProductId === oProduct.ProductId
);



if (oEntry) {
	// update existing entry
	oEntry.Quantity += 1;
} else {
	// create new entry
	oEntry = {
		Id : jQuery.sap.uid(),
		Quantity : 1,
		...
		PictureUrl : oProduct.PictureUrl
	};
	oCartData.entries[oCartData.entries.length] = oEntry;
}
```

/test-resources/sap/m/demokit/cart/view/Product.controller.js

Functional

```
// find existing entry for product
let oEntry = aCartEntries.find(
	x => x.ProductId === oProduct.ProductId
);


if (oEntry) {
	// update existing entry
	oEntry.Quantity += 1;
} else {
	// create new entry
	oEntry = {
		Id : jQuery.sap.uid(),
		Quantity : 1,
		...
		PictureUrl : oProduct.PictureUrl
	};
	oCartData.entries[oCartData.entries.length] = oEntry;
}
```

Functional (including currying)

```
let hasSame = R.curry((prop, ref, x) => ref[prop] === x[prop]),
    isSameProd = hasSame("ProductId", oProduct);

// find existing entry for product
let oEntry = aCartEntries.find(isSameProd);


if (oEntry) {
	// update existing entry
	oEntry.Quantity += 1;
} else {
	// create new entry
	oEntry = {
		Id : jQuery.sap.uid(),
		Quantity : 1,
		...
		PictureUrl : oProduct.PictureUrl
	};
	oCartData.entries[oCartData.entries.length] = oEntry;
}
```

/test-resources/sap/m/demokit/cart/view/Product.controller.js

Imperative

```
// recalculate total price
oCartData.totalPrice = 0;
for (var j = 0 ; j < oCartData.entries.length ; j ++) {
     oCartData.totalPrice += parseFloat(oCartData.entries[j].Price) * oCartData.entries[j].Quantity;
}
```

Functional

```
// recalculate total price
oCartData.totalPrice = aCartEntries.reduce((a, x) => a + parseFloat(x.Price * x.Quantity), 0);
```

Where can I find out more?

Resources

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)

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

Language Ramblings (blog)

*For more information, please reread