Software development with Scala

Vladyslav Pekker

Hochschule Konstanz
University of Applied Sciences

What is this course about?

80% Programming in Scala

- Syntax
- Semantics

What is this course about?

80% Programming in Scala

- Syntax
- Semantics

20% Software development

- Programming languages
- Programming disciplines

What is the goal of this course?

The goal is to give a head start on learning Scala.

What is the goal of this course?

The goal is to give a head start on learning Scala.

Scala's learning curve

Part I

The Java Ecosystem

Outline

1 Metal programming

2 A History Lesson...

Once upon a time...

Once upon a time...

in the land of computers there were 10101110010101101...

Once upon a time...

in the land of computers there were 10101110010101101...
then assemblers came along...

Once upon a time...

in the land of computers there were 10101110010101101...
then assemblers came along...
followed by compilers...

Once upon a time...

in the land of computers there were 10101110010101101...
then assemblers came along...
followed by compilers...
and they lived happily ever after...

Once upon a time...

in the land of computers there were 10101110010101101...
then assemblers came along...
followed by compilers...
and they lived happily ever after... NOT!

Portability

Platform Dependency

```
for(os <- List("linux", "mac", "windows"))
  println("If you compile a C program under " + os
          + "you can run it only on " + os + ".")
```

Portability

Platform Dependency

```
for(os <- List("linux", "mac", "windows"))
  println("If you compile a C program under " + os
          + "you can run it only on " + os + ".")
```

Sometimes a recompilation is not sufficient. The code has to be adapted!

Portability

Platform Dependency

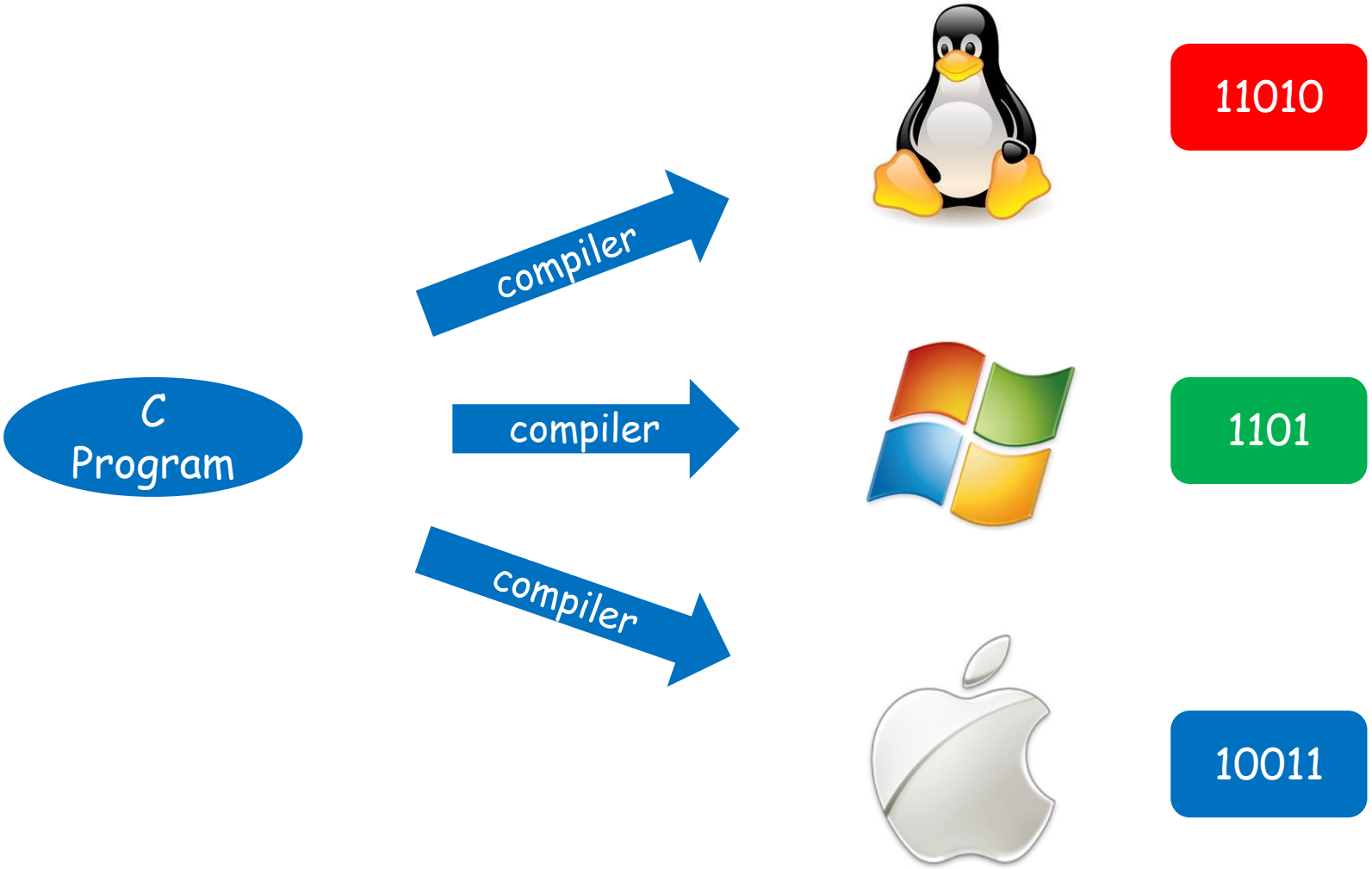
```
for(os <- List("linux", "mac", "windows"))
  println("If you compile a C program under " + os
          + "you can run it only on " + os + ".")
```

Sometimes a recompilation is not sufficient. The code has to be adapted!

Platform Independency

Write Once, Run Anywhere.

Problem - Unmanaged code



Problem - Unmanaged code 2

Why is unmanaged code a problem?

Problem - Unmanaged code 2

Why is unmanaged code a problem?

1. portability issues

Problem - Unmanaged code 2

Why is unmanaged code a problem?

1. portability issues
2. low level hazards like buffer overflows, segmentation faults etc.

Problem - Unmanaged code 2

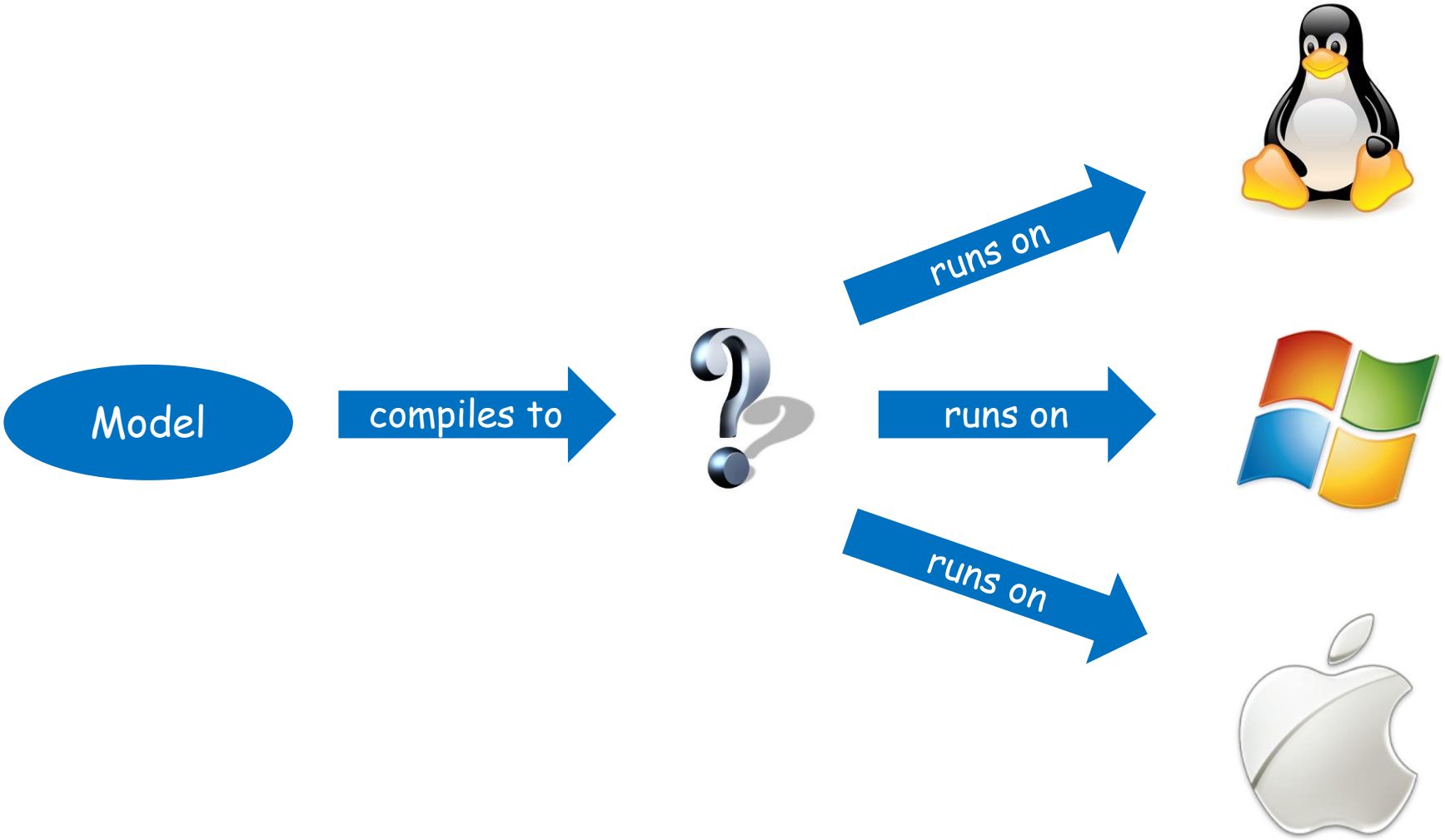
Why is unmanaged code a problem?

1. portability issues
2. low level hazards like buffer overflows, segmentation faults etc.

The problem, programming language designers are trying to solve:

Software development is not about programming the hardware, it is about abstracting over the hardware!

Solution - Managed code



Solution - Managed code 2

Why is managed code a solution?

Solution - Managed code 2

Why is managed code a solution?

1. portability issues are being taken care by VM vendors.

Solution - Managed code 2

Why is managed code a solution?

1. portability issues are being taken care by VM vendors.
2. low level hazards like buffer overflows, segmentation faults etc. are being handled by the VM.

Solution - Managed code 2

Why is managed code a solution?

1. portability issues are being taken care by VM vendors.
2. low level hazards like buffer overflows, segmentation faults etc. are being handled by the VM.
3. programs as-well as programming languages are changing every second; platforms are fairly stable!

Solution - Managed code 2

Why is managed code a solution?

1. portability issues are being taken care by VM vendors.
2. low level hazards like buffer overflows, segmentation faults etc. are being handled by the VM.
3. programs as-well as programming languages are changing every second; platforms are fairly stable!

Wind of Change

- it is feasible to change a program.
- it is hard to change a programming language.
- it is very hard to change a platform, but platforms are not required to be frequently changed.

Virtual Machines

Operating System

An operating system is an abstraction over a hardware machine.

Virtual Machines

Operating System

An operating system is an abstraction over a hardware machine.

Virtual Machine

A virtual machine is an abstraction over an operating system.

Virtual Machines

Operating System

An operating system is an abstraction over a hardware machine.

Virtual Machine

A virtual machine is an abstraction over an operating system.

Well-known VMs

Java	Java Virtual Machine (JVM)
.Net	Common Language Runtime (CLR)
Android	Dalvik virtual machine
	and *many more*

Managed Programming Languages

JVM - runs everywhere

Java Statically typed OO language

CLR - runs only on Windows

Managed Programming Languages

JVM - runs everywhere

Java Statically typed OO language

Scala Statically typed OO & FP language

CLR - runs only on Windows

Managed Programming Languages

JVM - runs everywhere

Java	Statically typed OO language
Scala	Statically typed OO & FP language
Clojure	Dynamically typed FP language (*Lisp* dialect)

CLR - runs only on Windows

Managed Programming Languages

JVM - runs everywhere

Java	Statically typed OO language
Scala	Statically typed OO & FP language
Clojure	Dynamically typed FP language (*Lisp* dialect)
Groovy	Dynamically typed scripting language

CLR - runs only on Windows

Managed Programming Languages

JVM - runs everywhere

Java	Statically typed OO language
Scala	Statically typed OO & FP language
Clojure	Dynamically typed FP language (*Lisp* dialect)
Groovy	Dynamically typed scripting language
	and *many more*

CLR - runs only on Windows

Managed Programming Languages

JVM - runs everywhere

Java	Statically typed OO language
Scala	Statically typed OO & FP language
Clojure	Dynamically typed FP language (*Lisp* dialect)
Groovy	Dynamically typed scripting language
	and *many more*

CLR - runs only on Windows

C#	Statically typed OO & FP language

Managed Programming Languages

JVM - runs everywhere

Java	Statically typed OO language
Scala	Statically typed OO & FP language
Clojure	Dynamically typed FP language (*Lisp* dialect)
Groovy	Dynamically typed scripting language
	and *many more*

CLR - runs only on Windows

C#	Statically typed OO & FP language
VB	Statically typed OO language

Managed Programming Languages

JVM - runs everywhere

Java	Statically typed OO language
Scala	Statically typed OO & FP language
Clojure	Dynamically typed FP language (*Lisp* dialect)
Groovy	Dynamically typed scripting language
	and *many more*

CLR - runs only on Windows

C#	Statically typed OO & FP language
VB	Statically typed OO language
F#	Statically typed FP language

Managed Programming Languages

JVM - runs everywhere

Java	Statically typed OO language
Scala	Statically typed OO & FP language
Clojure	Dynamically typed FP language (*Lisp* dialect)
Groovy	Dynamically typed scripting language
	and *many more*

CLR - runs only on Windows

C#	Statically typed OO & FP language
VB	Statically typed OO language
F#	Statically typed FP language
	and *many more*

Managed Programming Languages 2

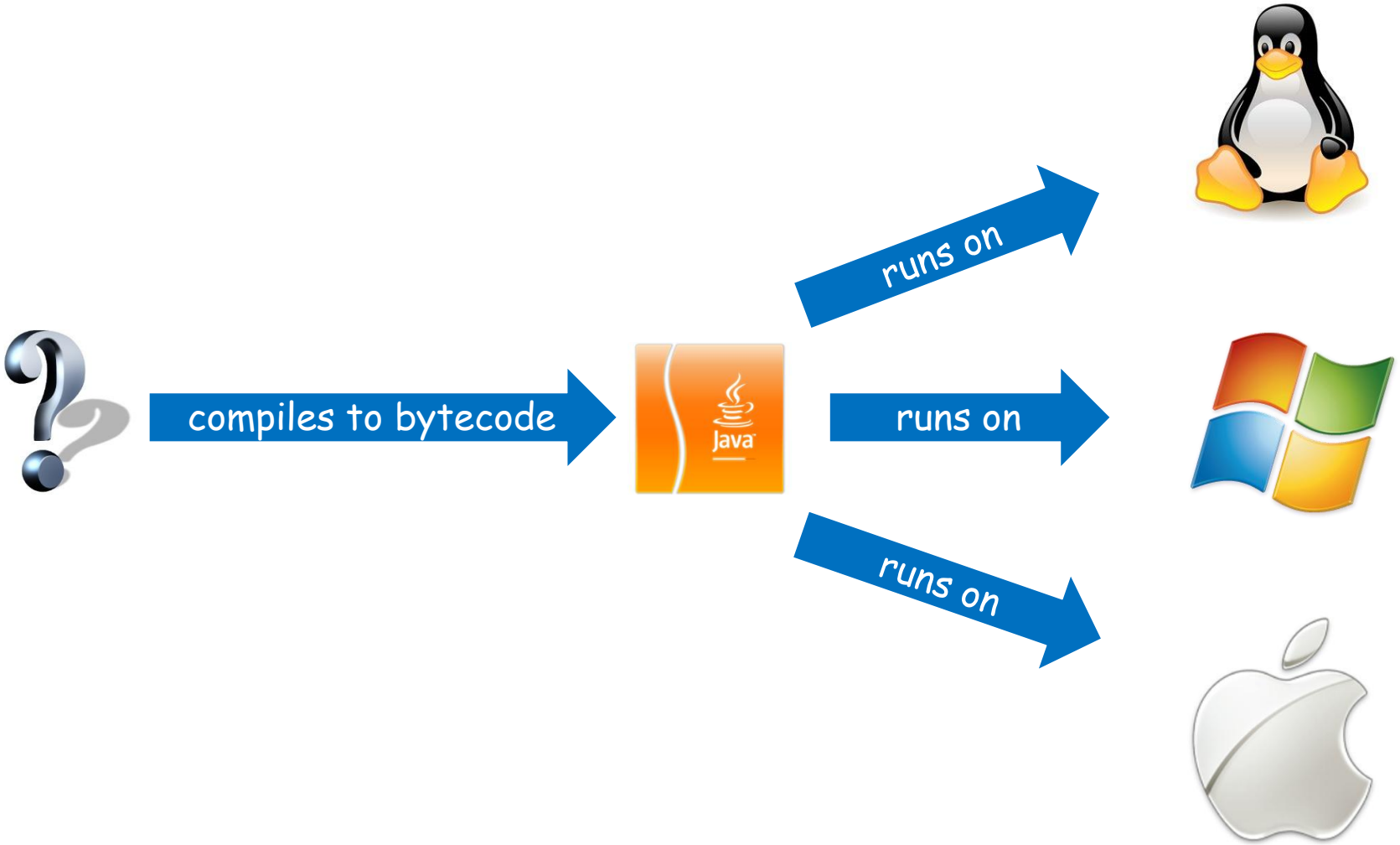
Dalvik - designed for Android

Java is officially supported. Every JVM language should theoretically work. Scala works. *Few others* work as-well.

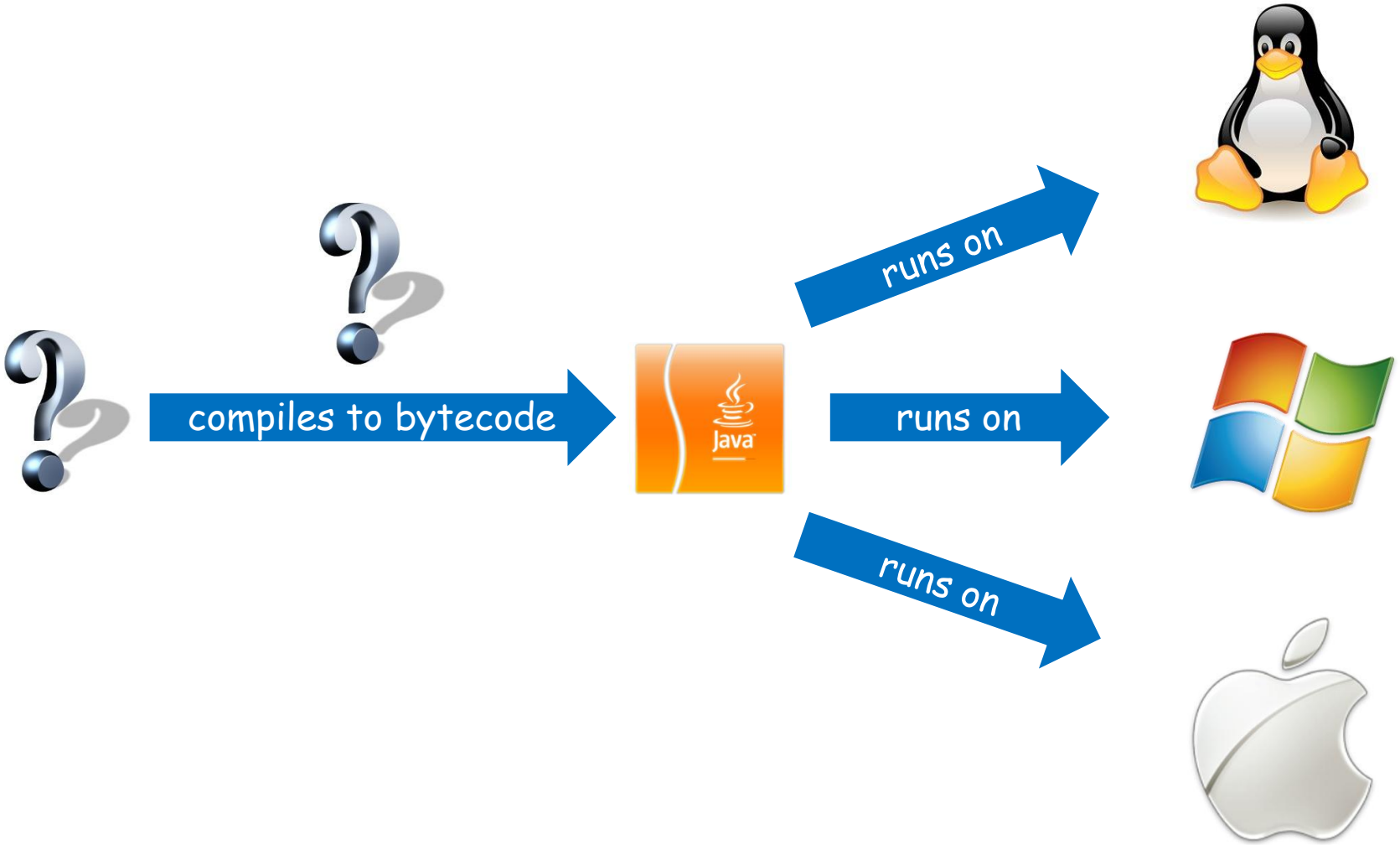
JVM - People's Choice

The code is not compiled to binary.
The code is compiled to the so called bytecode.
The JVM is a software component, which runs the bytecode.
A JVM implementation exists for almost every platform.
Java is the most used language on the JVM.
The most used Java compiler is called javac.

What language should we use for the JVM?



Can we reuse the java compiler for this language?



Outline

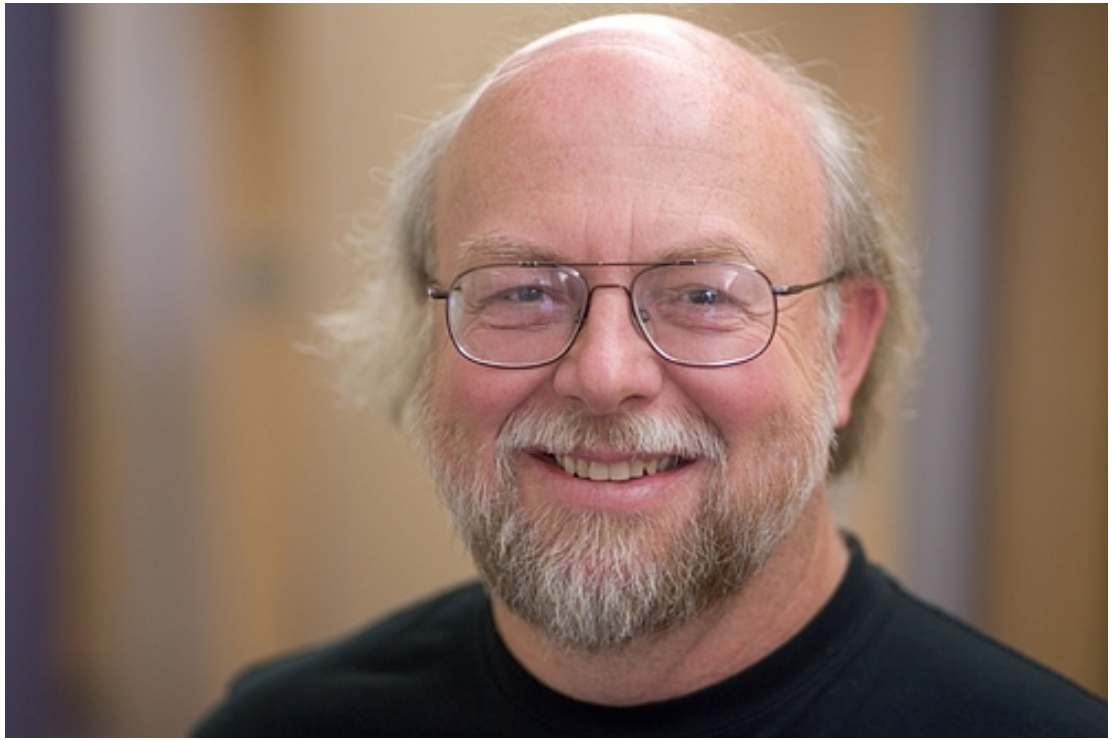
1 Metal programming

2 A History Lesson...

The Java Time Line

1996

Java 1.0 designed by James Gosling



The Java Time Line 2

1997

Marin Odersky and Philip Wadler start working on Pizza.

- Pizza is Java + extensions (generics, higher-order functions, etc.)
- Pizza compiler translates Pizza source code to Java source code so that the Java compiler can translate it to bytecode.

The Java Time Line 3

1998

Bracha, Odersky, Stoutamire and Wadler design Generic Java (GJ) for JavaSoft (later known as Sun Microsystems).

- GJ compiler translates Generic Java (Java + parts of Pizza) source code directly to bytecode.
- GJ adopts only generics from Pizza.

The Java Time Line 4

2000

GJ compiler becomes javac in JDK 1.3

- the new javac is more stable, correct and even performs better.
- javac (GJ compiler) still compiles Java (not GJ) to bytecode.

The Java Time Line 4

2000

GJ compiler becomes javac in JDK 1.3

- the new javac is more stable, correct and even performs better.
- javac (GJ compiler) still compiles Java (not GJ) to bytecode.

2004

Sun Microsystems releases Java 1.5

- generics from GJ finally make it to Java.

The Java Time Line 4

2000

GJ compiler becomes javac in JDK 1.3

- the new javac is more stable, correct and even performs better.
- javac (GJ compiler) still compiles Java (not GJ) to bytecode.

2004

Sun Microsystems releases Java 1.5

- generics from GJ finally make it to Java.

2005-2010

Java 1.6

- Java alternatives appear every day

The Java Time Line 4

2000

GJ compiler becomes javac in JDK 1.3

- the new javac is more stable, correct and even performs better.
- javac (GJ compiler) still compiles Java (not GJ) to bytecode.

2004

Sun Microsystems releases Java 1.5

- generics from GJ finally make it to Java.

2005-2010

Java 1.6

- Java alternatives appear every day

2011-present

Java 1.7

Meanwhile. . .

A long time ago in a galaxy far, far away....

Mid-1980s

A lot of research around functional programming languages. ML, OCaml, Miranda, Lisp dialects like Scheme.

- features like type inference, lazy evaluation and higher-ranked types.

Meanwhile. . .

A long time ago in a galaxy far, far away....

Mid-1980s

A lot of research around functional programming languages. ML, OCaml, Miranda, Lisp dialects like Scheme.

- features like type inference, lazy evaluation and higher-ranked types.

1987-1999

Development of Haskell by Simon Peyton-Jones.

- a pure, lazy, functional language with lots of new features
- advanced type system
- Martin Odersky contributes work on higher-ranked types to the Glasgow Haskell Compiler.

Meanwhile. . .

1999-2001

Martin Odersky works on Funnel language at EPFL in Switzerland

- basic principles, combines FP and Petri nets
- allows imperative, concurrent and OO programming
- compiles to JVM bytecode
- easy for academic folks, but too heavy for the industry

The Scala Time Line

2004

Scala - The fresh start on Funnel

- combines OO and FP
- static, but inferred typing
- features from Java and Pizza

The Scala Time Line

2004

Scala - The fresh start on Funnel

- combines OO and FP
- static, but inferred typing
- features from Java and Pizza

2008-2010

Scala 2.x - 2.7x

- Twitter takes Scala for a spin, and brings it into the Spotlight
- features from Haskell
- fine-grained qualification of access modifiers
- cutting edge features like extractor objects or abstract types

The Scala Time Line 2

2011

Scala 2.8x

- complete rework of the collections framework
- The Typesafe company - commercial support for Scala
- Serious attention from the industry

Scala 2.9x

- parallel collections
- own build tool - Simple Build Tool (SBT)
- better IDE and docs

The Scala Time Line 2

2011

Scala 2.8x

- complete rework of the collections framework
- The Typesafe company - commercial support for Scala
- Serious attention from the industry

Scala 2.9x

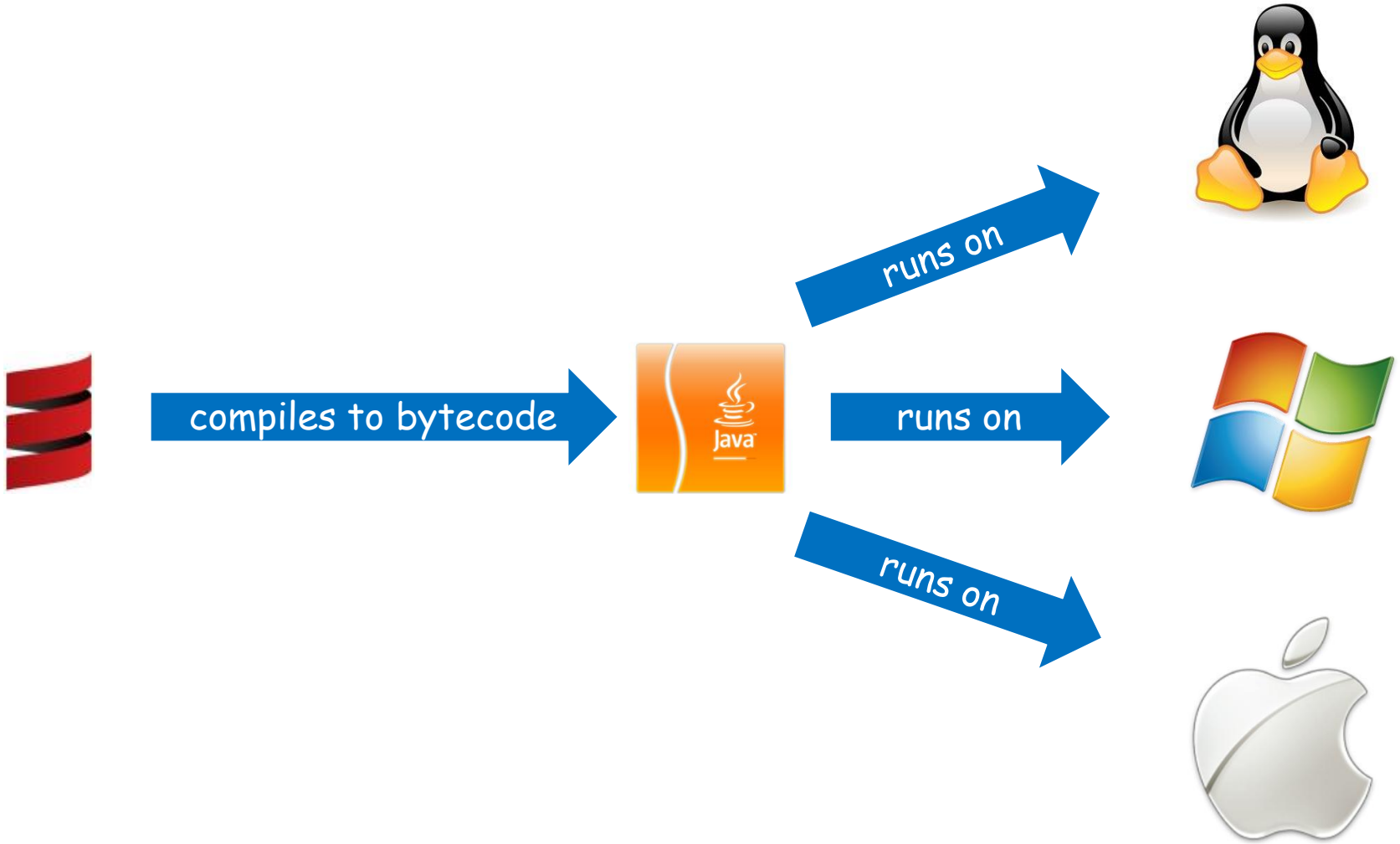
- parallel collections
- own build tool - Simple Build Tool (SBT)
- better IDE and docs

2012-present

Scala 2.10x (in development at the moment)

- new reflection framework
- dynamic typing (similar to C# 4)

Scala - People's Choice



Why should I learn Scala?

The Academic Point of View

- Cutting Edge language
- Pure OO language
- FP features
- Very advanced type system
- Joy
- You should learn a new programming language once in a year ;)

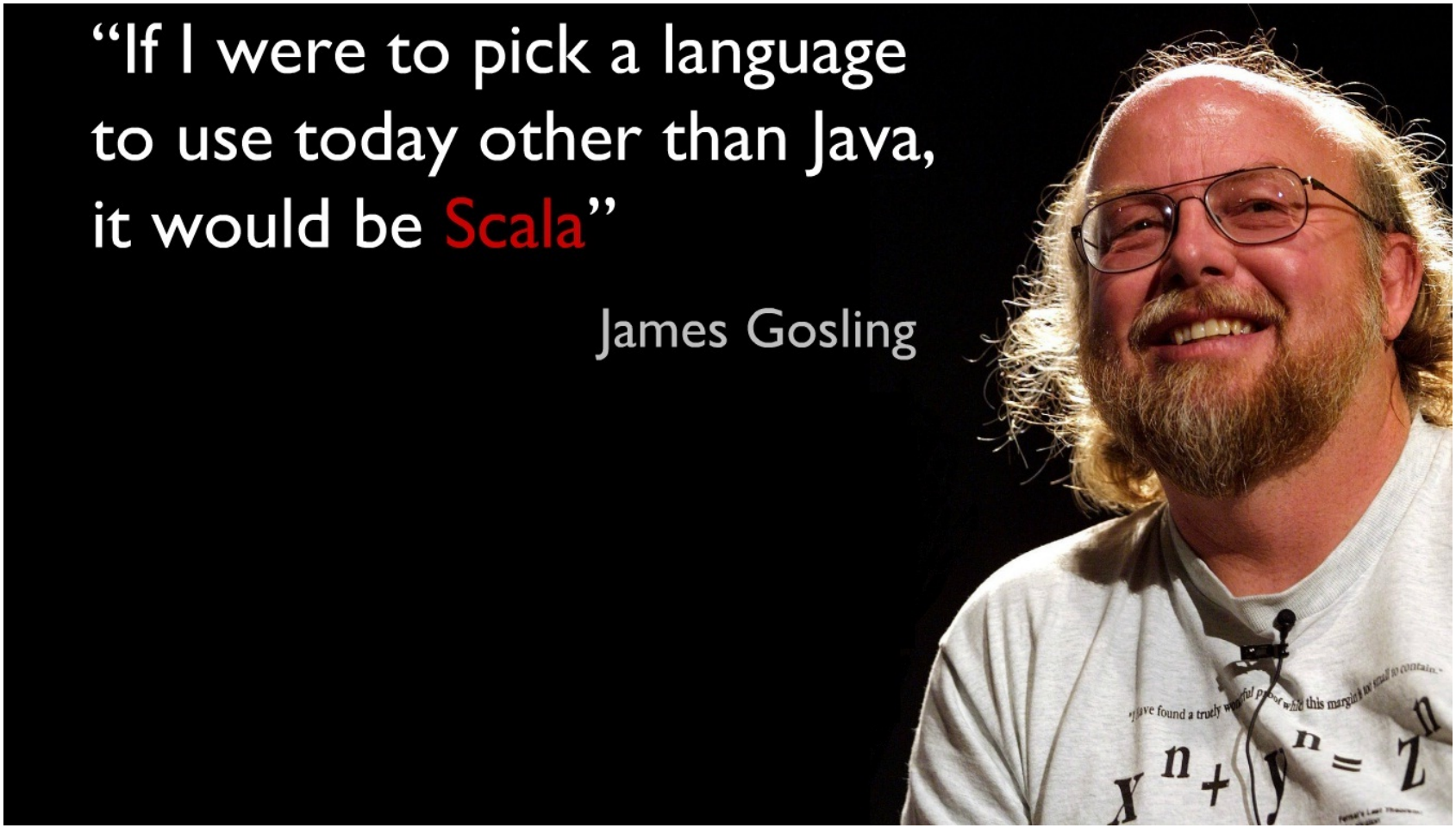
Why should I learn Scala?

The Industrial Point of View

- OO language
- Makes you more productive
- Provides a worthy alternative for solving concurrency problems
- Seamless interoperability with Java
- Lots of Scala jobs out there!

Scala in the Industry

Java is not going to be here forever



Java is not going to be here forever

Though my tip though for the long term replacement of javac is Scala. I'm very impressed with it! I can honestly say if someone had shown me the Programming in Scala book by by Martin Odersky, Lex Spoon & Bill Venners back in 2003 I'd probably have never created Groovy.

James Strachan - designer of Groovy

Summary

- JVM is a great platform
- Java is getting old
- Scala seems to be a great language
- Scala's interoperability with Java is seamless
- Scala combines OO with FP
- Scala is accepted in the industry

Part II

Introducing Scala

Outline

1 Reading Scala

Reading Scala