Why a new language

Peter van Rijn

History

- Assembly
  - Bootstrap switches
- Procedural
  - C, pascal
  - Language structures
    - Choosing, looping
- Object oriented
  - Smalltalk, Simula
  - C++, Java, C#

Where are we now?

- Administrative problems
  - To and from the database
    - That can't be difficult?
  - Meta programming
    - Ruby on rails
- General languages
  - If a hammer is the tool
    - Than every problem will became a nail
  - Domain Specific Languages

Where are we now?

- The multi core machines are coming
  - Parallel programming
  - Locking not a good solution
    - Java's synchronized and transactions are locking
  - The internet is parallel
- Frameworks are design pattern
  - Design pattern solve language issues
  - Solution that add a lot complexity
  - Java EE a whole lot of frameworks

OO

- Libraries
- Interface versus implementation
  - Plug-ins
- Administration is not OO
  - Dataflow
  - workflow
    - State machine

Java

- 12 years old
  - Solutions to problems 12 years ago
    - C++ too complex
    - Platform dependency
    - Vendors libraries
- Choses made for simplicity
  - C++ --
  - Java 1.1 smalltalk concepts, reflection

Java

- Java 1.0 the start
- Java 1.1 smalltalk concepts, reflection
- Java 1.2 swing
- Java 1.3 xml, j2ee
- Java 1.4 optimization
- Java 1.5 generics, enum, annotations (c++)
- Java 1.6 annotations + webservices
- Java 1.7 ??

Java strength

- JVM
  - Write once, run anywhere
    - It really works
  - Runtime optimization
- Libraries
  - For everything
  - Huge open source community
- Servlets
  - Fantastic technology

Java weakness

- Primitive types
  - Not classes
- Switch
- Not extensible /scalable
  - Only via new keywords
    - assert, enum, generics
- No inner functions and closures
- function pointers/literals
  - Only via interface

Solutions

- Dynamic languages
  - Groovy, Ruby
  - Annotations
  - generators
- Functional languages
  - Scala, haskell
    - Most functional languages do not allow side effects and imply immulability => Threadsafe

New language features

- On JVM
- Using java libraries
- Comparable with java

- Groovy, jruby
- scala

Dynamic typed

- Duck typing
  - No compile time type checking
  - No code completion
  - Tries it a runtime
- Generator
  - Meta programming
  - Easy to build a code generator

Static typed

- You have to type every variable
- Compile time checking
  - Type casting
- Code completion
- Scala is statically typed
  - But you don't have to write the type
    - The compiler finds out by itself (infers)
  - Feels like dynamic typing

scala

- Numbers are classes
- Operators are functions
- Leave out extra information
  - Wildcard parameter
- Inner functions
- Anonymous functions
- Closures
- Languages structures are functions

scala

- Case classes
- Pattern matching
- Traits
  - With on declaration
- Collections
  - Tuple
  - List, array, map
  - Option: Some, None
- For queries