First Programming Language in CS Education – The Arguments for Scala

WORLDCOMP 2011
By
Dr. Mark C. Lewis
Trinity University
I am writing a Scala textbook that is under contract with CRC Press.

- I would do this tutorial anyway, just because I like Scala that much.
http://www.cs.trinity.edu/~mlewis/ScalaLinks.html

- This is a collection of links for those who want more information on Scala.
Scala and Me

Grad schools and type systems
- Functional Programming and ML
- Interest in X10 and Fortress

Trinity curriculum update
- Was C → Java
- CS1 not object-early
- CS1 & CS2 use same language
Basics of Scala

“Scalable Language”

- Multi-Paradigm
- Productivity of scripting languages
- Expressivity of functional languages
- Scalability of standard OO languages
- Speed of compiled, statically-typed languages
- All OO
- Highly Functional
- Static-typing with local type inference
Too Complex?

Reasons for perception

- Scala is different
- Functional isn't broadly known
- Scalability → power
  - Bloggers show “cool” examples

Simpler in many ways

- Uniform syntax
### Shorter Language Specification

<table>
<thead>
<tr>
<th>Static Typed Language</th>
<th>Spec Length (pages)</th>
<th>Dynamic Typed Language</th>
<th>Spec Length (pages)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>552</td>
<td>Common LISP</td>
<td>1153</td>
</tr>
<tr>
<td>C++</td>
<td>1325</td>
<td>Ruby</td>
<td>341</td>
</tr>
<tr>
<td>C#</td>
<td>553</td>
<td>PHP</td>
<td>244</td>
</tr>
<tr>
<td>Java</td>
<td>684</td>
<td>JavaScript</td>
<td>252</td>
</tr>
<tr>
<td>VB.NET</td>
<td>597</td>
<td>Python 3.1</td>
<td>119</td>
</tr>
<tr>
<td>Scala</td>
<td>191</td>
<td>Scheme</td>
<td>50</td>
</tr>
<tr>
<td>F#</td>
<td>250</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fewer Keywords

Jobs

Job Trends from Indeed.com

- scala developer
- erlang developer
- scheme developer
- lisp developer

Percentage of Matching Job Postings

Jan '06 Jan '07 Jan '08 Jan '09 Jan '10 Jan '11
Benefits in CS 1

Programming in the small
- REPL
- Scripting Environment

Libraries allow for interesting code (JVM)
- Static type checking
- Uniform syntax
- Everything is a method call
- Powerful collections
Starting Off

“Hello, World” and such

- REPL
- Types - “lack” of primitives
  - tuples
- Statements/expressions
- Semicolon inference
- Simple input
- Variables
  - `val`
  - `var`
- Scripts
Conditions and Functions

- **Functional → expressions**
- **if expression**
- **Boolean logic**
- **Blocks of code**
- **Functions**
  - `def` (would make method in class)
- **Function literals**
  - `n => n+1`
  - `_+1`
Recursion for Iteration

- This is my style, even in C.
- Works well with functional
- Cements previous topics

Further with Scala
- Passing function argument → early abstraction

Matching idiom
- Introduce patterns
Collections (Array/List)

- Just for Scala
- Doesn't make sense before loops in most languages.
- One mutable, one immutable
- Many standard methods
- Many higher-order methods

Syntax
- Use () for indexing
- List also have ML style operations
- Creation, pass-by-name
Loops

- While loop
  - Not an expression
- For loop
  - Really for-each
- Ranges
- Many options
  - Multiple generators
  - If guards
  - Variables
  - Patterns
Files

- Can use Scanner
  - scala.io.Source
  - Scala Iterator[Char]
  - getLines : Iterator[String]
- Use with higher order methods
- Write with PrintWriter
- Introduce APIs?
Case Classes

- Immutable struct in simplest usage
- Simple syntax for grouping data
- Works as a pattern
- Copy method
**GUIs**

**scala.swing** wraps **javax.swing**

- Cleaner beginner syntax
- No explicit inheritance
- Reactions use partial functions
- **Drawbacks**
  - Currently no JTree
  - Tables complex
  - Button syntax uses companion object
Graphics

- Full Java2D
  - Really using Java
- Override paint method
- Use BufferedImage
- Events for animations
  - Keyboard
  - Mouse
  - Timer
Sorting & Searching

- Monomorphic at this point
- Can write your own visualization
- There are methods in collections
Other Stuff

- XML
  - Organizes data better than flat files
  - Serious recursion
  - Problems with high branching factor
- Object-Orientation
  - Normal classes
  - Including methods
Benefits in CS 2

- Pure OO
- Fewer quirks than Java
- Traits
- Rich collections
- Libraries again
- Can make things interesting/relevant
- Eclipse
- Scalable language
- Libraries as language
OO for Larger Programs

Review and extend from CS1

- Visibility/Private
- Special methods
  - Symbolic
  - Property assignment
- apply
- update

Object declarations
Eclipse

- Full apps need an IDE
- Automatic error checks
- Name completion
- Needs memory
Polymorphism

Inheritance/Subtyping
- Traits
- Protected visibility

Parametric Polymorphism
- Type bounds
- Parametric methods $\rightarrow$ polymorphic sorts
Other Collections

Sets
- Include expected methods
- Maps
  - (key, value)
  - key → value

Buffers
- Mutable
- Variable sized
- Many options mutable and immutable
Multithreading

Many approaches
- `java.lang.Thread`
- `java.util.concurrent`
- Parallel collections
  - Added in 2.9
- Actors

Simplified by functional
Networking

- Use java.net and java.io
- Streams
- Sockets
- Serialization
ADTs

I cover these:

- Stack/Queue
- Linked List
- Priority Queue
  - Sorted LL
  - Heap
- Trees
  - BST
- Spatial
- Manifests
- Inheriting traits!!!
Grammars

- RegEx
  - `.r` method on String
  - Triple quote strings
  - Patterns
  - Combinatorial Parsers
  - CF grammar → parser
  - Arithmetic example
Beyond CS 1&2

- Akka
- Non-JVM implementations
- GPGPU libraries
Conclusions

- Good early on
- REPL
- Scripting
- Grows well
- OO & static types
- IDE support
- Complete libraries
- Complexity not required