Scala, Your Next Programming Language
(or if it is good enough for Twitter, it is good enough for me)

WORLDCOMP 2011

By

Dr. Mark C. Lewis
Trinity University
Disclaimer

- 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.
How I got into Scala

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

Past Complaints

Documentation
- API
- Books
- Web resources
- Tools
- Improved IDE support
- Support
- Typesafe
Current Challenges

- **Compile Time**
  - Smart, multipass compiler
  - Optimization in progress

- **Binary Compatibility**
  - Changes in traits require recompilation

- **Uniform Style**
  - Many paradigms
  - Need guidelines
  - Tools coming
General rules

- All values are objects
- Operators are methods
- All expressions are valid statements
- Almost everything is an expression

Declarations

- Begin with keyword
  - val, var, def, type, class, object, trait
- Value parameters use ()
- Type parameters use []
Syntactic Sugar

Semicolon inference
- Newlines become semicolons when it fits
- Explicit works too

Operator notation
- Leave off dot and parens
- Methods with arity 0 or 1
Usage Methods

- REPL for simple tests
- Scripting for short programs
- Good for gluing things together

Applications
- Declare object with main
- Model is much like Java
Difference for Classes

- Take arguments
- No body required
- Code in body run at construction

Special methods
- Symbols
- Property assignment
- apply
- update
- Case classes
Object Declarations

- Creates singleton objects
- No static in Scala
- Companion objects
  - Apply method commonly used for object construction
  - No arguments
- Can inherit
Traits

- Not interfaces
- Allow method implementations and data
- No arguments
- Can inherit from many
- Call order is linearized

Key benefits
- Rich interfaces
- Extension of supertypes
Rich collection libraries
  - Array, List, Set, Map, etc.
  - Multiple varieties
    - Mutable/immutable
  - Creation
  - Companion object methods
    - fill, tabulate
  - Higher-Order Methods
    - map, filter, foreach, ...
  - Views
  - Streams
Standard
Immutable
- Match is not just switch
- Cases can be patterns and bind variables.
- Examples of patterns
  - Tuples
  - Array, List, etc.
  - RegEx
  - XML
  - much more
- No break
For Expressions

- Not your normal for
- Technically for-each
- More options
  - Multiple generators
  - Variable declarations
  - If guards
  - Patterns
- No break, continue, or goto
Implicit Conversions

- "Pimp my Interface"
- Strict rules
  - Must be in scope
  - Only one applied
- Allows extension of
  - java.lang.String
  - Arrays
- Any code you didn't write
Useful Scripting

- scala.sys.process added in 2.9
- Simple system calls
- Piping between programs
- Conditional calls
- Allows Java library calls
- Glue things together
Regular Expressions

- Triple quote string literals
- The `r` method on `String`
- Work as a pattern
- Combine with for loop
- Skip non-matches
XML

- Literals
- Pattern matching
- Loads DOM

**Xpath style searching**
- \ for immediate contents
- \ for deep search
- Use @ for properties
Combinatorial Parsers

- CF grammar → parser
- ^\^ to specify return
- Quick to parse trees
Parallelism

- Full access to Java libs
- Functional makes it easier
- Parallel collections
  - Added in 2.9
  - Call par method
- Fast conversions, O(1)
- Uses work stealing
- Works with for loops
More Parallel

- Actors
  - Communicate through messages
  - Single threaded in an actor
- Scala Actors
- Akka
DSLs

- Libraries look like language features
- Pass-by-name semantics
- Use implicits for built-in types
- Can use combinatorial parsers
Web Frameworks

- **Lift**
  - Written for Scala
  - Different approach

- **Play!**
  - Also for Java
  - Scala version uses Scala idioms
Conclusions

- Get most of the best of all worlds
- Less boiler plate
- Static type safety and speed
- High expressivity
- Keep current JVM functionality
- Easier parallel
- High level libraries
- Look like language features
- Easy to use
- DSLs