# Atomic Scala

An Introductory Tutorial

Bruce Eckel
from the book by
Bruce Eckel & Dianne marsh

# The Book

- Small chapters give a sense of accomplishment
- Instead of a "deep dive," just enough to cover a single concept
- The smallest possible amount: "atom"
- Only what you need to know for the next atom
- No chapter numbers to make them easy to shift around
- Book can be an intro for a dedicated learner

# Why another language?

- Parallelism is too hard -- threads and locks are basically impossible to get right (Heisenbugs)
- Functional Programming Languages are often too strange and restrictive
- Object/Functional Hybrid
  - More familiar syntax
  - More succinct: less code, more productivity ( > factor of two less code)
- Runs on JVM, interoperable with Java
- Statically typed, but with type inference
- Powerful & more consistent syntax

# More intros

- My blog posts:
  - Scala: The Static Language that Feels Dynamic
  - Scala, Patterns and The Perl Effect
  - Is Scala Only for Computer Scientists?
- Martin Odersky:
  - Scala: An Introduction
  - Odersky Explains Shared-Memory Concurrency

# Comments

```
47 * 42  // Perform multiplication
47 + 42

47 + 42 /* A multiline comment
doesn’t care
about newlines */
```

# Scripting

```
// ScriptDemo.scala
println("Hello, Scala!")

> scala ScriptDemo.scala

Hello, Scala!
```

# Values and Data Types

**val** *name*:*type* **=** *initialization*

(Values.scala)

# Variables

**var** *name:type* **=** *initialization*

# Type Inference

```
scala> val n = 1 + 1.2
n: Double = 2.2
```

# Expressions

*A statement changes state*.

*An expression expresses.*

(Expressions.scala)

```
scala> val e = println(5)
e: Unit = ()

scala> val f = {}
f: Unit = ()
```

# Conditional Expressions

(If.scala, If2.scala, If3.scala, If4.scala, If5.scala )

# For Loops

(For.scala)

# Objects

scala> val r = Range(0, 10)
scala> r.***(PRESS THE TAB KEY)***

# ScalaDoc

***http://www.scala-lang.org/api/current/index.html***

# Vectors

(VectorsAndObjects.scala)