The Feel of Scala

Bill Venners
President
Artima, Inc.
www.artima.com
Overall Presentation Goal

Give you a *feel* for what Scala programming is like, by showing Scala solutions to real problems.
Enough about me

- Run the Artima Developer website
- An old Java guy, who chose Scala for Artima's next language
- Coauthored *Programming in Scala*
- Primary developer of ScalaTest
My Problem

The verbosity of Java code adds accidental complexity to my software.
Scala lets me...

use and design libraries that enable clearer, more concise code.
What about Java?

- Can enhance Java
- But can only add; can't subtract
- Scala adds to *and subtracts from* Java
- Not source compatible with Java language
- But has seamless binary compatibility with JVM
What about JRuby, Jython, and Groovy?

- More concise; no type annotations
- Can add new methods to existing classes
- Can pass objects that share no common supertype to a method, which treats them uniformly
Dynamic vs. Static

- More concise; no type annotations
- Can add new methods to existing classes
- Can pass objects that share no common supertype to a method, which treats them uniformly

- Deterministic refactoring
- Fewer tests needed
- Documentation
- Code completion
- Performance, on JVM
Scala gives me...

- More concise; fewer type annotations
- Can *effectively* add new methods to existing classes
- Can pass objects that share no common supertype to a method, which treats them uniformly
- Deterministic refactoring
- Fewer tests needed
- Documentation
- Code completion
- Performance, on JVM
import scala.io.Source

def widthOfLength(s: String) = s.length.toString.length

if (args.length > 0) {

    val lines = Source.fromFile(args(0)).getLines.toList

    val longestLine = lines.reduceLeft(
        (a, b) => if (a.length > b.length) a else b
    )
    val maxWidth = widthOfLength(longestLine)

    for (line <- lines) {
        val numSpaces = maxWidth - widthOfLength(line)
        val padding = " " * numSpaces
        print(padding + line.length +" | " + line)
    }
}
else
    Console.err.println("Please enter filename")
Type inference
def factorial(x: BigInteger): BigInteger =
if (x == BigInteger.ZERO)
    BigInteger.ONE
else
    x.multiply(factorial(x.subtract(BigInteger.ONE)))

versus

def factorial(x: BigInt): BigInt =
    if (x == 0) 1 else x * factorial(x - 1)
Every value is an object;
Every operation a method call

factorial(x - 1)
factorial(x-1)

map.containsKey('a')
map containsKey 'a'
In dynamic languages, open classes let you add new methods to existing classes and objects.

Hamcrest matchers (Java)

```java
assertThat(map, hasKey('a'));
```

RSpec matchers (Ruby)

```ruby
map.should have_key('a')
```
Scala's implicit conversions lets you appear to add new methods to existing classes.

ScalaTest matchers (Scala)

map should have key ('a')
map should have

map.should(have)

map should have key ('a')

map.should(have).key('a')

1 + 2 − 3

(1).+(2).−(3)
public class Wrapper {
    private Object payload;
    public Wrapper(Object payload) {
        this.payload = payload;
    }
    public void should(HaveWord c) {
        // ...
    }
}

A wrapper class (in Java)
A conversion method (in Java)

```java
public static Wrapper wrap(Object obj) {
    return new Wrapper(obj);
}
```

```java
map.should(have) // ERROR
wrap(map).should(have)
```
An *implicit* conversion method (in Scala)

class Wrapper(payload: Any) {
    def should(HaveWord c) {
        // ...
    }
}

implicit def wrap(o: Any) = new Wrapper(o)
map should have key ('a')

map.should(have).key('a')

wrap(map).should(have).key('a')
ScalaTest matchers

map should have key ('a')

collection should contain element (1.0)

collection should have size (17)

string must equal ("done")

array must have length (9)
Scala:

assert(result == 17)
Scala:

assert(result == 17)

JUnit:

assertEquals(17, result);
Scala:

assert(result == 17)

JUnit:

assertEquals(17, result);

ScalaTest:

assert(result === 17)
The === operator
Which is actual? Which is expected?

```java
assertEquals(animalShelter.getCat(), myKitty);
```
A trick question

JUnit:
assertEquals(expected, actual);

TestNG:
assertEquals(actual, expected);
A basic control abstraction in Scala

JUnit:

assertEquals(expected, actual);

TestNG:

assertEquals(expected, actual);

ScalaTest:

expect(myKitty) {
   animalShelter.getCat()
}
Curly braces and parens
Currying
By-name parameters
Expect in action
JUnit 3:

String s = "hi";
try {
    s.charAt(-1);
    fail();
} catch (StringIndexOutOfBoundsException e) {
    // Expected, so continue
}

JUnit 4:

@Test(expected=StringIndexOutOfBoundsException.class)
public void testPassingANegativeToCharAt() {
    s.charAt(-1);
}
Testing for thrown exceptions (ScalaTest)

```scala
intercept[StringIndexOutOfBoundsException] {
  s.charAt(-1)
}
```
DEMO

Intercept in action

Test methods with Suite and TestNGSuite

Test functions with FunSuite

BDD with Spec
The need to test private methods

```java
expect("'c'") {
    FailureMessages.decorateToStringValue('c')
}
```
Symbol literals in Scala

'decorateToStringValue

""decorateToStringValue"

Symbol
Testing with invokePrivate (ScalaTest)

```scala
val decorateToStringValue = PrivateMethod[String](
  'decorateToStringValue
)

expect("\"Howdy!\"") {
  FailureMessages invokePrivate decorateToStringValue("Howdy!")
}
expect("'c'") {
  FailureMessages invokePrivate decorateToStringValue('c')
}
expect("1") {
  FailureMessages invokePrivate decorateToStringValue(1)
}
expect("true") {
  FailureMessages invokePrivate decorateToStringValue(true)
}
```
invokePrivate in action
Type checking in matchers

Map("hi" -> 1) should have key ("hi")

Map(1 -> "hi") should have key ("hi")

MatcherSpec.scala:1089: error: type mismatch;
  found    : java.lang.String("hi")
  required: Int
  Map(1 -> "hi") should have key ("hi")
What should I require about the type of object?

*object should have length (7)*
Duck typing lets you pass objects that have the same structure, but no common supertypes.

object should have length (7)
Duck typing lets you pass objects that have the same structure, but no common supertypes.

object should have length (7)

Array: public final int length;

String: public int length();

Document: public int getLength();
View bounds and the *can-be-treated-as* relationship

```
trait LengthWrapper {
  def length: Int
}
```

- `ArrayLengthWrapper` is a `LengthWrapper`:
  - `Array` can be treated as `ArrayLengthWrapper`

- `StringLengthWrapper` is a `LengthWrapper`:
  - `String` can be treated as `StringLengthWrapper`

- `DocumentLengthWrapper` is a `LengthWrapper`:
  - `Document` can be treated as `DocumentLengthWrapper`
Object can be any type that *can be treated as* a LengthWrapper

- **string** should have length (7)
- **array** should have length (7)
- **document** should have length (7)
length() - Method in class java.io.File
Returns the length of the file denoted by this abstract pathname.

length() - Method in interface java.lang.CharSequence
Returns the length of this character sequence.

length() - Method in class java.lang.String
Returns the length of this string.

length() - Method in class java.nio.CharBuffer
Returns the length of this character buffer.

length() - Method in class java.nio.charset.CoderResult
Returns the length of this character buffer.

length() - Method in interface java.sql.Blob
Returns the number of bytes in the Blob object.

length() - Method in interface java.sql.Clob
Returns the number of characters in the Clob object.

length() - Method in class java.util.BitSet
Returns the number of elements in the BitSet.

length() - Method in class java.util.concurrent.atomic.AtomicIntegerArray
Returns the length of the array.

length() - Method in class java.util.concurrent.atomic.AtomicReferenceArray
Returns the length of the array.

length() - Method in class java.nio.file.FileCacheImageOutputStream
Returns the length of the file denoted by this abstract pathname.

length() - Method in class java.nio.file.FileImageInputStream
Returns the length of the file denoted by this abstract pathname.

length() - Method in class javax.imageio.stream.FileImageInputStream
Returns the length of the file denoted by this abstract pathname.

length() - Method in class javax.imageio.stream.FileNameInputStream
Returns the length of the file denoted by this abstract pathname.

length() - Method in class javax.imageio.stream.InputStreamImpl
Returns the length of the stream.

length() - Method in class javax.imageio.stream.OutputStreamImpl
Returns the length of the stream.

length() - Method in class javax.imageio.stream.ImageInputStreamImpl
Returns the length of the stream.

length() - Method in class javax.imageio.stream.MemoryImageOutputStream
Returns the length of the stream.

length() - Method in class javax.sound.midi.MidiMessage
Returns the length of the MidiMessage object.

length() - Method in class org.omg.CORBA.Interface
Returns the length of the interface object.

length() - Method in class org.omg.CORBA.Value
Returns the length of the Value object.

length() - Method in interface org.w3c.dom.Node
Returns the length of the nodes in the list.

length() - Method in interface org.w3c.dom.NamedNodeMap
Returns the length of the nodes in the list.

length() - Method in interface org.w3c.dom.Document
Returns the number of nodes in the document.

length() - Method in interface org.w3c.dom.Element
Returns the number of attributes in the element.

length() - Method in interface org.w3c.dom.Node
Returns the number of nodes in the list.

length() - Method in interface org.w3c.dom.NodeList
Returns the number of nodes in the list.

length() - Method in interface org.xml.sax.Attributes
Returns the number of attributes in the list.

length() - Method in interface org.xml.sax.ContentHandler
Returns the number of characters in the content currently in the document.

length() - Method in interface org.xml.sax.DocumentHandler
Returns the number of DOMImplementations in the list.

length() - Method in interface org.xml.sax.DocumentFactory
Returns the number of DOMImplementations in the list.

length() - Method in interface org.xml.sax.DTMIFactory
Returns the number of DOMImplementations in the list.

length() - Method in interface org.xml.sax.XMLReader
Returns the number of nodes in the document.

length() - Method in interface org.xml.sax.XMLStreamHandler
Returns the number of nodes in the list.
Structural types in Scala

LengthWrapper

« trait »

def length: Int

LengthFieldWrapper

is a

can be treated as

{ val length: Int }

LengthMethodWrapper

is a

can be treated as

{ def length(): Int }

GetLengthMethodWrapper

is a

can be treated as

{ def getLength(): Int }
Object can be any type that can be treated as a LengthWrapper

string should have length (7)
array should have length (7)
document should have length (7)
file should have length (7)
blob should have length (7)
atomicIntegerArray should have length (7)
gapContent should have length (7)
midiMessage should have length (7)
datagramPacket should have length (7)
iioByteBuffer should have length (7)
documentEvent should have length (7)
characterData should have length (7)
nodeList should have length (7)
attributes should have length (7)
yourNewClass should have length (7)
testing length

optional type systems
Summary: Scala decreases the cost and increases the benefit of static typing.

- More concise; *fewer* type annotations
- Can *effectively* add new methods to existing classes
- Can pass objects that share no common supertype to a method, which treats them uniformly
- Deterministic refactoring
- Fewer tests needed
- Documentation
- Code completion
- Performance, on JVM

Website: www.devoxx.com
Programming in Scala is available in the Devoxx bookstore.