Groovy

Introduction

What’s Groovy?

Groovy:

“marvelous, wonderful, excellent, hip, trendy”

Language features:

- Lightweight
- Dynamic
- Object-oriented
- Open sourced under the Apache License
- Compiles into Java bytecode
- Extends the Java API and libraries

Bibliography

- Venkat Subramaniam, Programming Groovy 2, The Pragmatic Programmers, LLC, 2013
- https://github.com/ericraio/Groovy

Dynamic languages

- Have the ability to extend a program at runtime
  - Changing types
  - Behaviors
  - Object structures

Dynamic languages

- Give me 5% raise

```
5.percentRaise(80000)
```

- We can easily modify the behaviour of the built-in integer type:

```
Integer.metaClass.percentRaise = { amount -> amount * (1 + delegate / 100.0) }
```

- *Code synthesis* vs. *code generation*

Unit testing

- Most dynamic languages are dynamically typed.
  - The types are often inferred based on the context.
- There are no compilers to flag type-casting violations at compile time.
- Quite a bit of code may be synthesized
  - Our program can be extended at runtime
  - We can’t rely upon coding-time verification alone.
- Writing code in dynamic languages requires greater discipline than writing in statically typed languages.

Groovy classes

- Groovy classes extend *java.lang.Object*

```
groovy:000> println XmlParser.class
class groovy.util.XmlParser
===> null
groovy:000> println XmlParser.class.superclass
class java.lang.Object
===> null
groovy:000>
```

GDK

- Groovy extends JDK with convenience methods and closure support through the Groovy JDK (GDK).

```
groovy:000> lst = ['Groovy', 'is', 'hip']
===> [Groovy, is, hip]
groovy:000> println lst.join(' ')
Groovy is hip
===> null
groovy:000> println lst.getClass()
class java.util.ArrayList
===> null
groovy:000>
```

From Java to Groovy

```
// Java code
public class Greetings {
  public static void main(String[] args) {
    for(int i = 0; i < 3; i++) {
      System.out.print("ho ");
    }
    System.out.println("Merry Groovy!");
  }
}
```

From Java to Groovy

- Groovy automatically imports the following Java packages:
  - *java.lang*, *java.util*, *java.io*, and *java.net*.
- It also imports classes *java.math.BigDecimal* and *java.math.BigInteger*.
- Groovy packages groovy.lang and groovy.util are also imported.

```
for(int i = 0; i < 3; i++) {
  System.out.print("ho ")
}
System.out.println("Merry Groovy!")
```

From Java to Groovy

- *println()* has been added to *java.lang.Object*.
- Grovy has a lighter form of loops using *Range* object
- Groovy is lenient with parentheses

```
for(i in 0..2) { print 'ho ' }

println 'Merry Groovy!'
```

Ways to loop

- If the closure expects only one parameter, we can use the default name *it* for it in Groovy.

```
0.upto(2) { print "$it "}

3.times { print "$it "}

0.step(10, 2) { print "$it "}



3.times { print 'ho ' }

println 'Merry Groovy!'
```

Examples 03-08

Other features

- The return statement is almost always optional
- The semicolon (;) is almost always optional, though we can use it to separate statements
- Methods and classes are public by default.
- We can initialize JavaBeans using named parameters

Examples 09-16

Implementing Interfaces

```
// Java code
button.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent ae) {
JOptionPane.showMessageDialog(frame, "You clicked!");
}
});

button.addActionListener(
{ JOptionPane.showMessageDialog(frame, "You clicked!") } as ActionListener
)
```

Implementing Interfaces

```
DisplayMouseLocation = { positionLabel.setText("$it.x, $it.y") }
frame.addMouseListener(displayMouseLocation as MouseListener)
frame.addMouseMotionListener(displayMouseLocation as MouseMotionListener)

handleFocus = [
focusGained : { msgLabel.setText("Good to see you!") },
focusLost : { msgLabel.setText("Come back soon!") }
]

button.addFocusListener(handleFocus as FocusListener)

events = ['WindowListener', 'ComponentListener']
// Above list may be dynamic and may come from some input
handler = { msgLabel.setText("$it") }
for (event in events) {
  handlerImpl = handler.asType(Class.forName("java.awt.event.${event}"))
  frame."add${event}"(handlerImpl)
}
```

- GroovyForJavaEyes/Swing.groovy

Groovy boolean evaluation

Type	Condition for Truth
Boolean	True
Collection	Not empty
Character	Value not 0
CharSequence	Length greater than 0
Enumeration	Has more elements
Iterator	Has next
Number	Double value not 0
Map	Not empty
Matcher	At least one match
Object[]	Length greater than 0
Any other type	Reference not null

```
str = 'hello'
if (str) { println 'hello' }

lst0 = null
println lst0 ? 'lst0 true' : 'lst0 false'
lst1 = [1, 2, 3]
println lst1 ? 'lst1 true' : 'lst1 false'
lst2 = []
println lst2 ? 'lst2 true' : 'lst2 false'
```

Examples 17-41