INTRODUCTION TO GROOVY

GABOR BATA

APRIL 20, 2015

AGENDA

1. Groovy overview
2. Language characteristics
3. From Java to Groovy
4. Essential Groovy features – lists, maps, closures etc.
5. Summary
6. Questions

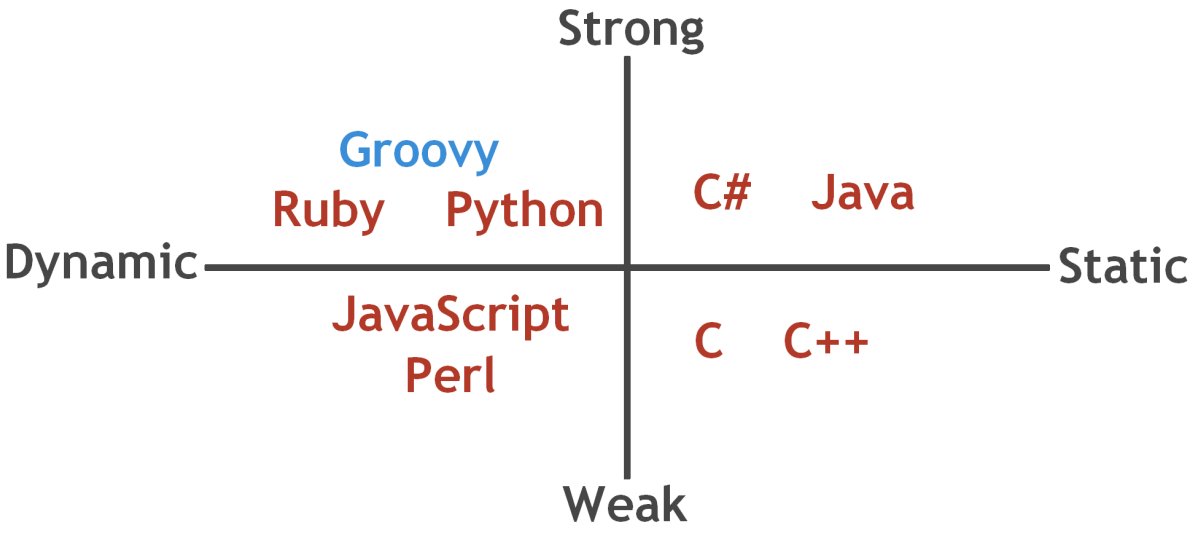
GROOVY OVERVIEW

Groovy is a dynamic language for the JVM. It shines with

- full object-orientation
- scriptability
- optional typing
- operator customization
- lexical declarations for the most common data types
- closures
- compact property syntax
- seamless Java integration

Groovy can largely be viewed as a superset of Java as *most valid Java files are also valid Groovy files*, but Groovy code can be more compact.

LANGUAGE CHARACTERISTICS



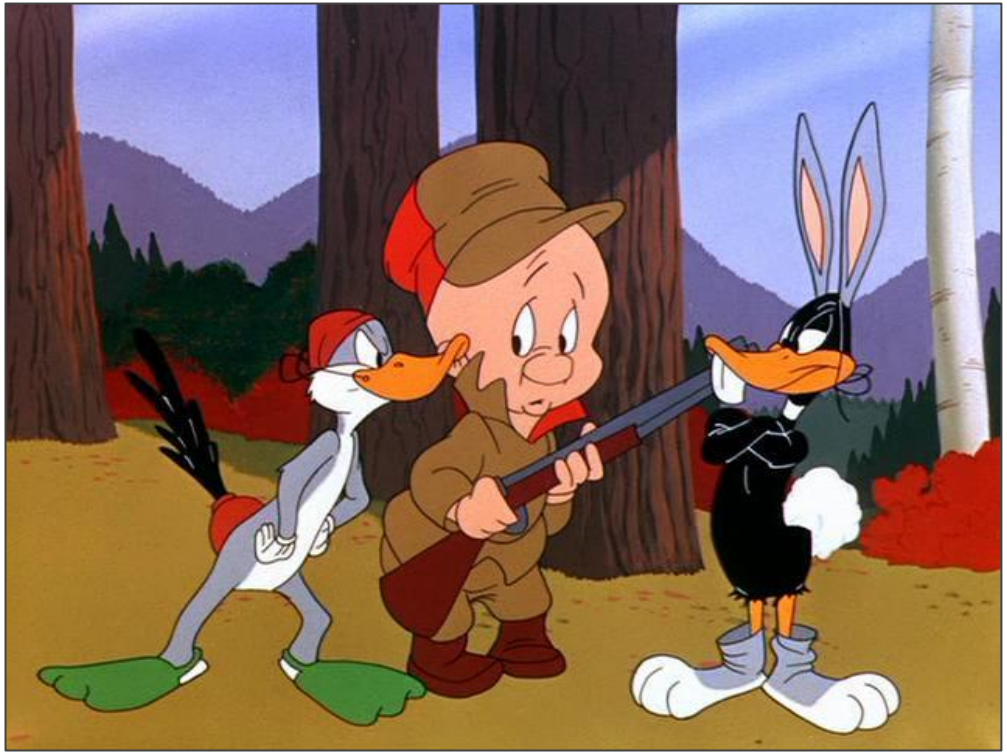
PARADIGM	Object-oriented, imperative, scripting, functional
TYPING DISCIPLINE	Dynamic, Static, Strong, Duck

LANGUAGE CHARACTERISTICS – CONTINUED

DUCK TYPING

"If it walks like a duck and talks like a duck, it must be a duck".

In other words, in a language that supports duck typing an object of a particular type is compatible with a method or function, if it supplies all the methods/method signatures asked of it by the method/function at run time.



FROM JAVA TO GROOVY

```java
01. package com.acme.java.example;
02.
03. public class HelloWorld {
04.   private String name;
05.
06.   public void setName(String name) {
07.     this.name = name;
08.   }
09.
10.   public String getName() {
11.     return name;
12.   }
13.
14.   public String greet() {
15.     return "Hello " + name;
16.   }
17.
18.   public static void main(String[] args) {
19.     HelloWorld helloWorld = new HelloWorld();
20.     helloWorld.setName("Austin Powers");
21.     System.out.println(helloWorld.greet());
22.   }
23. }
```

```groovy
01. package com.acme.groovy.example
02.
03. class HelloWorld {
04.   def name
05.   def greet() { "Hello ${name}" }
06. }
07.
08. def helloWorld = new HelloWorld()
09. helloWorld.name = "Austin Powers"
10. println helloWorld.greet()
```

- typing is optional (‘String’ vs. ‘def’)
- semi-colons/parentheses are optional
- public visibility is the default
- string interpolation with GString
- ‘return’ can be omitted (the last expression is returned)
- getter/setter is provided automatically
- fields (properties) can be accessed by dot notation
- handy methods and shortcuts (e.g. println)

ESSENTIAL GROOVY FEATURES – 1 / 5

ASSERT STATEMENT Verify pre- and post-conditions

```
def version = 42
assert version == 42  // ok
version++
assert version == 42  // assertion failure
```

OPTIONAL PARENTHESES Can be omitted if the method signature requires at least one parameter (DSL friendly).

```
// parentheses is optional
Math.max(1, 2)
Math.max 1, 2

println('Groovy is awesome!')
println 'Groovy is awesome!'
```

```
// parentheses is mandatory
list.size()
```

ESSENTIAL GROOVY FEATURES – 2 / 5

STRINGS

```
def single  = 'This is a \'single-quoted\' string'  // java.lang.String
def gstring = "Today is: ${new Date()}" // interpolation, groovy.lang.GString
def multiline = """
   This is a
   multiline string
"""
```

LISTS

```
def names = ['Austin Powers', 'Dr. Evil'] // ArrayList
assert names.size() == 2
assert names[1] == 'Dr. Evil'

names << 'Mini-me'
assert names.size() == 3

names.each { name -> println name }  // iterate over
```

```
// Change default list type
def x = [1, 2, 3] as LinkedList

// Ranges
println names[1..3]

// Outputs:
// [Dr. Evil, Mini-me]
```

ESSENTIAL GROOVY FEATURES – 3 / 5

MAPS

```
def releaseYears = ['Java': 1995, 'Groovy': 2003]  // java.lang.LinkedHashMap
assert releaseYears.size() == 2
assert releaseYears.Java == 1995
assert releaseYears['Java'] == 1995
releaseYears['Ruby'] = 1995
assert releaseYears.size() == 3
releaseYears.each { lang, year -> println "${lang} was first released in ${year}" }
```

REGEX

```
def twister = 'she sells sea shells'
assert twister =~ 'she'                    // contains 'she' (i.e. find)
assert twister ==~ /she.*shells/           // starts with 'she' and ends with 'shells'

def pattern = ~/she.*shells/               // the same example precompiled
assert pattern.matcher(twister).matches()
pattern.matcher(twister).each { ... }      // matchers are iterable
```

ESSENTIAL GROOVY FEATURES – 4 / 5

CLOSURES

A closure captures a piece of logic and the enclosing scope. They are first-class objects and can receive messages, can be returned from method calls, stored in fields, and used as arguments to a method call. This is a similar concept of lambdas in Java.

```
def name = "Austin"
def printClosure = { println "Hello, ${name}" }
printClosure()  // Outputs: Hello, Austin
name = "Dr. Evil"
printClosure()  // Outputs: Hello, Dr. Evil

def printClosure = { name -> println "Hello, ${name}" }
printClosure "Austin"
printClosure "Dr. Evil"

def printClosure = { name1, name2 -> println "Hello, ${name1}, ${name2}" }
printClosure "Austin", "Dr. Evil"
```

ESSENTIAL GROOVY FEATURES – 5 / 5

ITERABLES

Every object is iterable in Groovy, even if it was implemented in Java.
You can apply the following iterative object methods on them (this is not a complete list):

Returns	Purpose
`List`	`collect { ... }`
`(void)`	`each { ... }`
`(void)`	`eachWithIndex { item, index -> ... }`
`Object`	`find { ... }`
`List`	`findAll { ... }`
`Integer`	`findIndexOf { ... }`
`Integer`	`findLastIndexOf() { ... }`

SUMMARY

GROOVY IS ACTUALLY JAVA (ON STEROIDS)

- JVM based language
- Short learning curve
- Existing libraries for Java can be used
- Dynamic typing (with optional static typing)
- Compact syntax
- Closure is first class citizen
- Syntactic sugar – collections, property access, etc.
- Regex is first class citizen
- Ideal language for creating DSLs (e.g. Gradle, Grails)

USEFUL RESOURCES

WEBSITES

- *Groovy Programming Language*
  http://www.groovy-lang.org/
- *Groovy web console*
  http://groovyconsole.appspot.com/

RECOMMENDED READING

- *Groovy in Action* – Dierk König (Manning)
- *Programming Groovy* – Venkat Subramaniam (The Pragmatic Bookshelf)
- *Beginning: Groovy and Grails* – Christopher M. Judd, Joseph Faisal Nusairat, James Shingler (Apress)

THANK YOU