



Introduction to Groovy

About the speaker

-  Java developer since the early days
-  Open source believer since 1997
-  Member of the Groovy Dev team since 2007
-  Co-founder of the Griffon project

Agenda

-  What is Groovy
-  From Java to Groovy
-  Java-like Features
-  Not-Java Features
-  Unique Features
-  Resources

What is Groovy

- Dynamic and agile language for the JVM
- Built upon the strenghts of the Java Language
- Brings modern features from Python, Ruby & Smalltalk
- Works perfectly with Java
- Supports DSLs and compact syntax
- Makes programming fun again

From Java to Groovy

HelloWorld.java

```
public class HelloWorld {  
    String name;  
  
    public void setName(String name)  
    { this.name = name; }  
    public String getName(){ return name; }  
  
    public String greet()  
    { return "Hello " + name; }  
  
    public static void main(String[] args){  
        HelloWorld helloWorld = new HelloWorld();  
        helloWorld.setName("Groovy");  
        System.out.println(helloWorld.greet());  
    }  
}
```







HelloWorld.groovy

```
public class HelloWorld {  
    String name;  
  
    public void setName(String name)  
    { this.name = name; }  
    public String getName(){ return name; }  
  
    public String greet()  
    { return "Hello " + name; }  
  
    public static void main(String[] args){  
        HelloWorld helloWorld = new HelloWorld();  
        helloWorld.setName("Groovy");  
        System.out.println(helloWorld.greet());  
    }  
}
```

GroovierHelloWorld.groovy

```
class HelloWorld {  
    String name  
    def greet() { "Hello $name" }  
}  
  
def helloWorld = new HelloWorld(name: "Groovy")  
println helloWorld.greet()
```


But how?

-  Remove noise
-  Remove boilerplate code
-  Introduce dynamic types
-  Use variable interpolation
-  Turbocharged POJOs
-  Script support

HelloWorld.groovy

```
public class HelloWorld {  
    String name;  
  
    public void setName(String name)  
    { this.name = name; }  
    public String getName(){ return name; }  
  
    public String greet()  
    { return "Hello " + name; }  
  
    public static void main(String[] args){  
        HelloWorld helloWorld = new HelloWorld();  
        helloWorld.setName("Groovy");  
        System.out.println(helloWorld.greet());  
    }  
}
```

Java-like Features

Close to home

Java-like Features

- A Java class is a Groovy and viceversa
- Full JDK5 support: annotations, generics, varargs, enums
- 98% of Java code is valid Groovy code
- Rename *.java to *.groovy and compile!

Varargs in action

```
class Calculator {  
    def addAllGroovy(Object[] args) {  
        int total = 0  
        for(i in args) { total += i }  
        total  
    }  
  
    def addAllJava(int... args) {  
        int total = 0  
        for(int i : args) { total += i }  
        total  
    }  
}
```

```
Calculator c = new Calculator()  
assert c.addAllGroovy(1,2,3,4,5) == 15  
assert c.addAllJava(1,2,3,4,5) == 15
```

Scott Davis' 1st Mantra

Groovy is Java & Java is Groovy



Not-Java Features

Exploring the Neighborhood

Assorted Goodies

- 🟢 Default parameter values as in PHP
- 🟢 Name parameters as in Ruby
- 🟢 Operator overloading (through naming convention)
- 🟢 Native syntax for Maps and Lists
- 🟢 New types like Ranges and Closures (!!)
- 🟢 Regexpes are first-class citizens

Closures, Closure, Closures!

- ➊ Reusable blocks of code (think JavaScript functions)
- ➋ They substitute inner classes in almost all cases
- ➌ Can serve as proxies of single-method interfaces
- ➍ Default parameter named it if not supplied

Examples of Closures

```
greet1 = { name -> println "hello $name" }  
greet1("Groovy")  
  
greet2 = { println it }  
greet2("Java")  
  
iCanHaveTypedParametersToo = { int x, int y ->  
    println "coordinates are ($x, $y)"  
}  
iCanHaveTypedParametersToo(42, 21)  
  
myActionListener = { event ->  
    // do something cool with event here  
} as java.awt.event.ActionListener
```

Curried Closures

```
getSlope = { x, y, b = 0 ->  
    println "x: $x, y: $y, b: $b"  
    (y - b) / x  
}
```

```
assert 1 == getSlope(2, 2)  
getSlopeX = getSlope.curry(5)
```


```
assert 1 == getSlopeX(5)  
assert 0 == getSlopeX(2.5, 2.5)
```

Native Syntax for Maps/Lists

```
Map map = [:]
assert map instanceof java.util.Map
map["key1"] = "value1"
map.key2 = "value2"
println map
assert map.size() == 2
assert map.key1 == "value1"
assert map["key2"] == "value2"

List list = []
assert list instanceof java.util.List
list.add("One")
list << "Two"
println list
assert list.size() == 2
assert ["One", "Two"] == list
```

Iterators Everywhere

 You can use iterator methods with any object

 Iterators harness the power of closures

 Examples: each, collect inject, every, any, find

Iterators in Action

```
printIt = { println it }  
// 3 ways to iterate from 1 to 5, there are more!  
[1,2,3,4,5].each printIt  
1.upto 5, printIt  
(1..5).each printIt  
  
// compare to a regular loop  
for(i in [1,2,3,4,5]) printIt(i)  
  
[1,2,3,4,5].eachWithIndex { v, i -> println "list[$i] => $v" }
```

Scott Davis' 2nd Mantra

Groovy is Java & Groovy is NOT Java



Unique Features

To infinite.. and beyond!

The 'as' keyword






```
import javax.swing.table.DefaultTableCellRenderer as DTCR

def myActionListener = { event ->
    // do something cool with event
} as java.awt.event.ActionListener

def renderer = [
    getTableCellRendererComponent = { t, v, s, f, r , c ->
        // cool renderer code goes here
    }
] as DTCR

// note that this technique is like creating objects in
// JavaScript using JSON notation
```

New Operators

-  ?: (elvis) – a refinement on the ternary operator
-  ?. safe dereference – kiss NPEs goodbye!
-  <=> (spaceship) – compare two values
-  * (spread) – explodes contents of a list/array
-  .* (spread-dot) – apply method on all list elements

Traversing Object graphs

- GPath is to objects what XPath is to XML
- `?.` and `*.` come in handy in many situations
- Take advantage of Maps as POGOs
- Short syntax for property access a plus

Sample GPath Expressions

```
class Person {
    String name
    int id

    String toString() {
        "name: $name, id: $id"
    }
}

def persons = [
    new Person(name: "Duke", id: 1),
    [name: "Tux", id: 2] as Person
]

println persons
assert [1, 2] == persons.id
assert ["Duke", "Tux"] == persons*.getName()
assert null == persons[2]?.name
assert "Duke" == persons[0].name ?: "Groovy"
assert "Groovy" == persons[2]?.name ?: "Groovy"
```

Metaprogramming

- Groovy does not support open classes like Ruby does
- Change a class' behavior at runtime
- Since 1.6 change them at buildtime too (AST xforms)
- Intercept method calls and property access

Category Example

```
class Pouncer {
    static pounce(Integer self) {
        def s = "Boing!"
        1.upto(self - 1) { s += " boing!" }
        s
    }
}

use(Pouncer) {
    assert 3.pounce() == "Boing! boing! boing!"
    println 4.pounce()
}
```

Metaclass Example

```
Integer.metaClass.pounce = {  
    def s = "Boing!"  
    1.upto(delegate - 1) { s += " boing!" }  
    s  
}  
  
assert 3.pounce() == "Boing! boing! boing!"  
println 4.pounce()
```

AST Transformations

```
import java.text.SimpleDateFormat
class Event {
    @Delegate Date when
    String title, url

    String toString() {
        "title: $title, url: $url
when: $when"
    }
}

df = new SimpleDateFormat("MM/dd/yyyy")
so2gx = new Event(title: "SpringOne2GX",
    url: "http://springone2gx.com",
    when: df.parse("10/19/2009"))
oredev = new Event(title: "Oredev",
    url: "http://oredev.org",
    when: df.parse("11/02/2009"))
println so2gx
println oredev
assert oredev.after(so2gx.when)
```



Groovy is what the Java language would look like had it be written in the 21st century



Resources

 <http://groovy.codehaus.org>

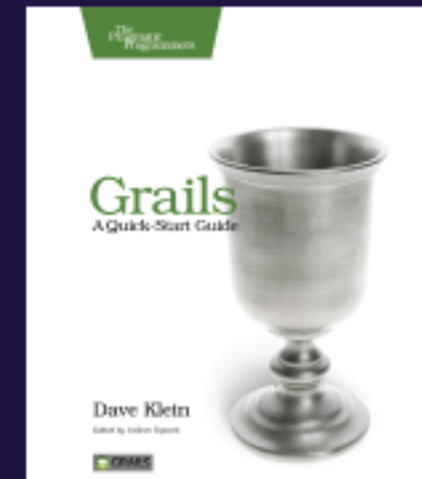
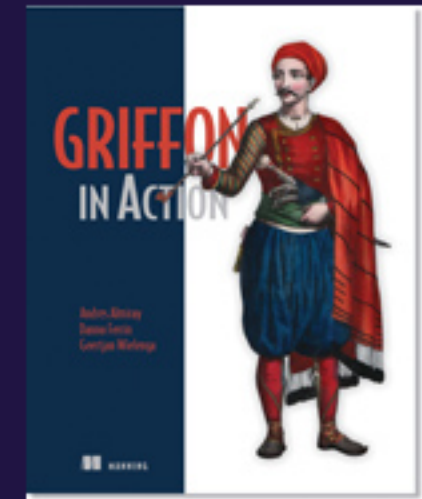
 <http://groovyblogs.org>

 <http://groovy.dzone.org>

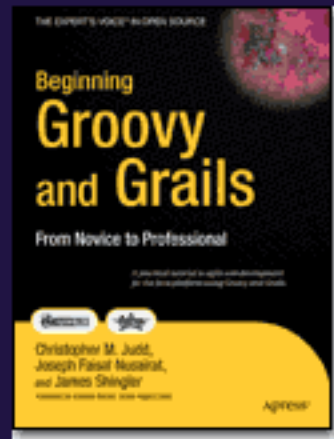
 <http://jroller.com/aalmiray>

 @aalmiray

Books



More Books



This presentation made with



Q & A

Thank you!