Groovy For Java Programmers

QCONSF 2010

Jeff Brown
Core Grails Developer
jeff.brown@springsource.com
SpringSource - A Division Of VMware
http://springsource.com/

Copyright 2010 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

What Is Groovy?

- Agile Dynamic Language For The JVM
- Inspired By Languages Such As...
  - Python
  - Ruby
  - Smalltalk

Why Groovy

- Powerful Dynamic Language
- Relatively Easy To Learn
- Familiar Syntax For Java Programmers
- Integrates Really Well With Java
  - Containers, Libraries, Existing Java Code

Installing Groovy

- Download Latest Release
  - http://groovy.codehaus.org/
- Extract Archive
- Set $GROOVY_HOME
- Add $GROOVY_HOME/bin to PATH

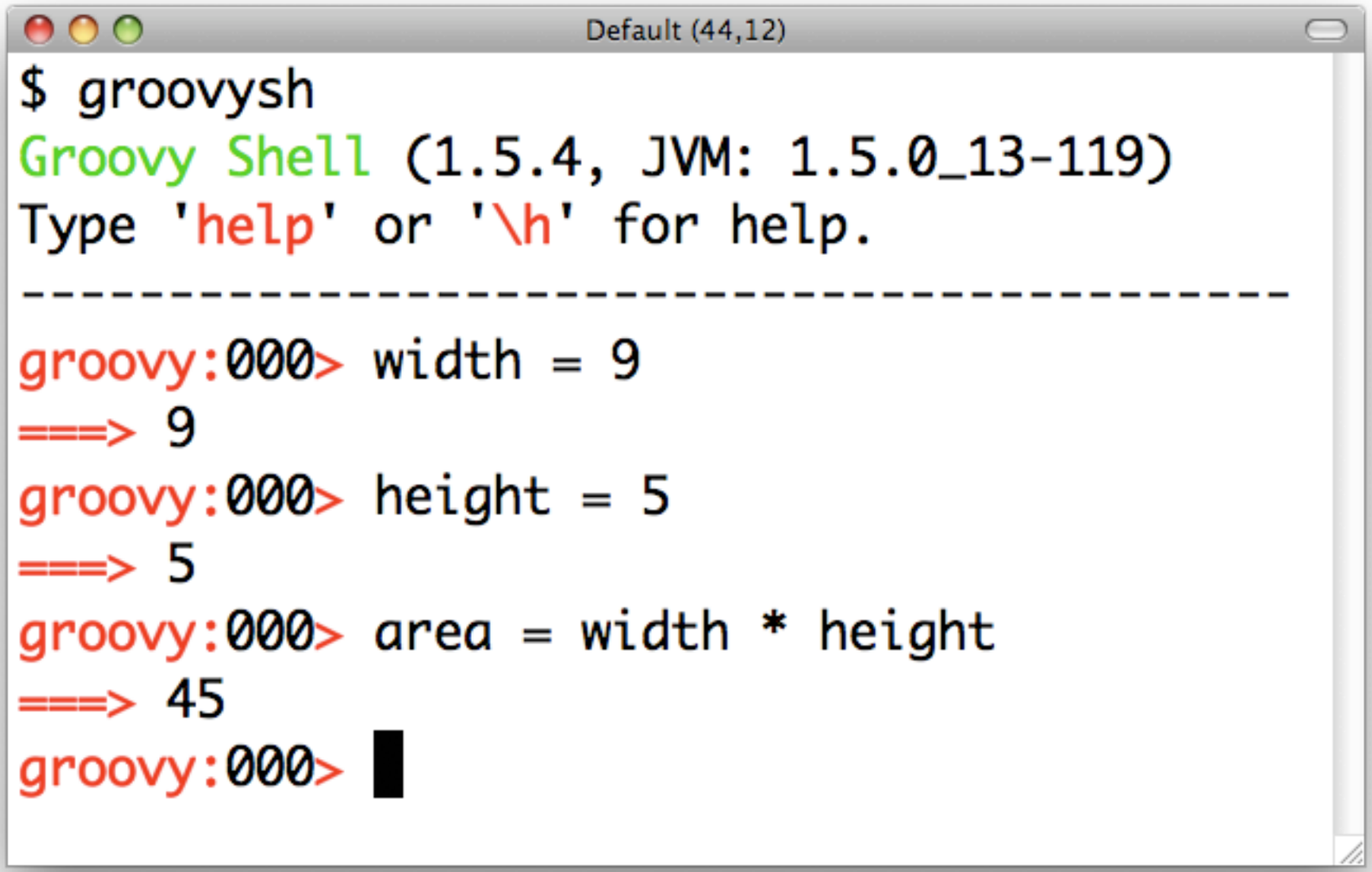
Groovy Tools

- groovy - Interpreter
- groovyc - Compiler
- groovysh - Shell
- groovyConsole - Swing Console

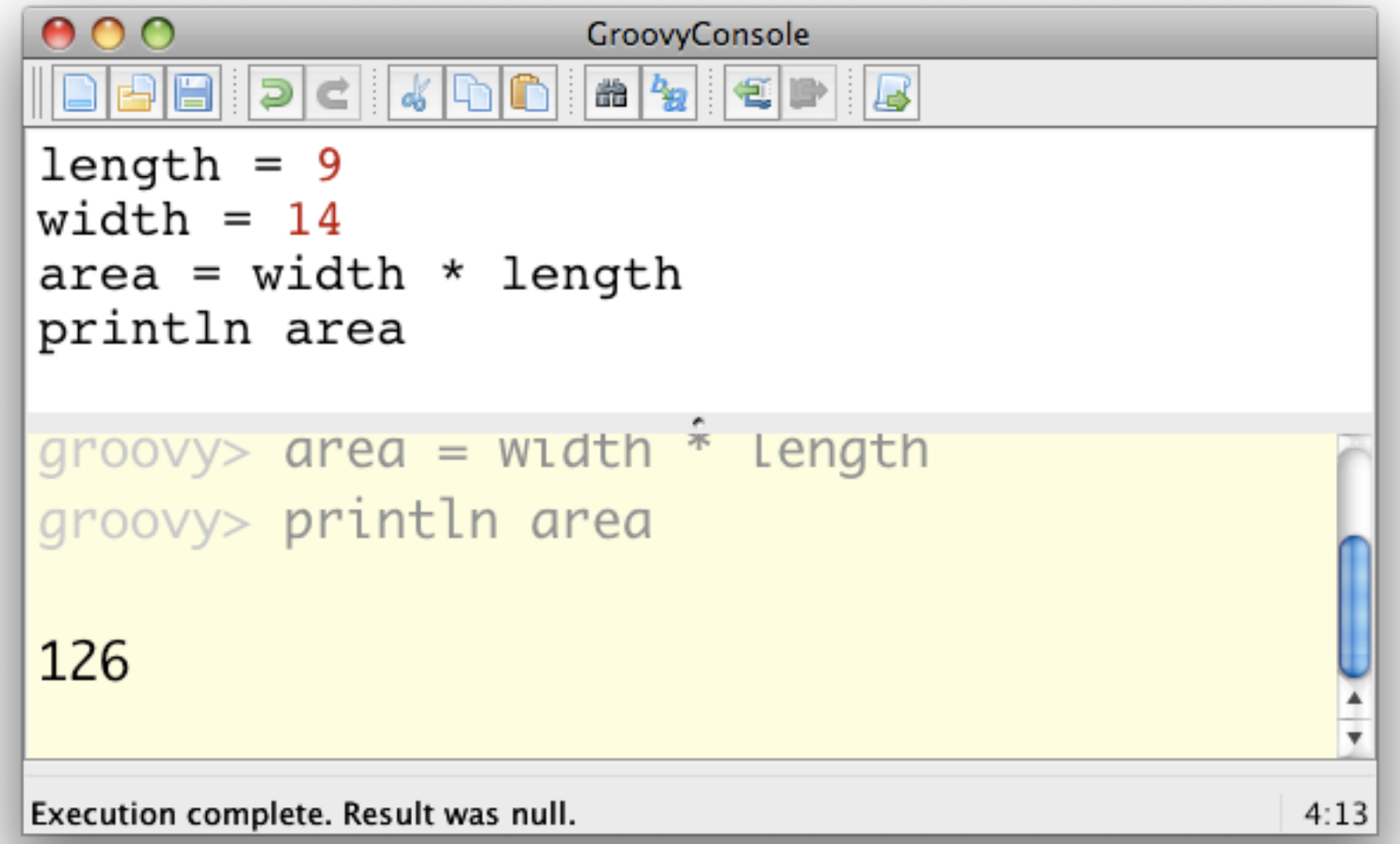
Give It A Spin

```
$ groovy -version
Groovy Version: 1.5.4 JVM: 1.5.0_13-119

$ groovy -e "println 'Groovy Rocks.'"
Groovy Rocks.

$ groovy -e "x=5; y=10; z=x*y; println z"
50
```

groovysh


Default (44,12)
$ groovysh
Groovy Shell (1.5.4, JVM: 1.5.0_13-119)
Type 'help' or '\h' for help.
groovy:000> width = 9
===> 9
groovy:000> height = 5
===> 5
groovy:000> area = width * height
===> 45
groovy:000>

groovyConsole



Groovy Class

```groovy
class GroovyPerson {

    // dynamically typed property
    def age

    // statically typed property
    String name

    def printName() {
        println name
    }

    static void main(String[] args) {
        def person = new GroovyPerson(age:7,
                                      name:'Jake')
        person.printName()
    }
}
```

more on Groovy properties later...

Groovy Scripts

- Scripts Do Not Require A Class Definition
  - no main method

```
// MyGroovyScript.groovy

println 'This is an executable script!'
```

Print Independence Day


```
// PrintIndependenceDay.java

import java.util.Calendar;
import java.util.Date;

public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.clear();
        calendar.set(Calendar.MONTH, Calendar.JULY);
        calendar.set(Calendar.DATE, 4);
        calendar.set(Calendar.YEAR, 1776);

        Date time = calendar.getTime();

        System.out.println(time);
    }
}
```

Print Independence Day


```
// PrintIndependenceDay.groovy

import java.util.Calendar;
import java.util.Date;

public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.clear();
        calendar.set(Calendar.MONTH, Calendar.JULY);
        calendar.set(Calendar.DATE, 4);
        calendar.set(Calendar.YEAR, 1776);

        Date time = calendar.getTime();

        System.out.println(time);
    }
}
```

No Utility Imports...

```
// PrintIndependenceDay.groovy



public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.clear();
        calendar.set(Calendar.MONTH, Calendar.JULY);
        calendar.set(Calendar.DATE, 4);
        calendar.set(Calendar.YEAR, 1776);

        Date time = calendar.getTime();

        System.out.println(time);
    }
}
```

No Semicolons...

```groovy
// PrintIndependenceDay.groovy



public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance()
        calendar.clear()
        calendar.set(Calendar.MONTH, Calendar.JULY)
        calendar.set(Calendar.DATE, 4)
        calendar.set(Calendar.YEAR, 1776)

        Date time = calendar.getTime()

        System.out.println(time)
    }
}
```

No Getters...

```
// PrintIndependenceDay.groovy



public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.instance
        calendar.clear()
        calendar.set(Calendar.MONTH, Calendar.JULY)
        calendar.set(Calendar.DATE, 4)
        calendar.set(Calendar.YEAR, 1776)

        Date time = calendar.time

        System.out.println(time)
    }
}
```

No Static Typing...

```
// PrintIndependenceDay.groovy



public class PrintIndependenceDay {

    public static void main(String[] args) {
        def calendar = Calendar.instance
        calendar.clear()
        calendar.set(Calendar.MONTH, Calendar.JULY)
        calendar.set(Calendar.DATE, 4)
        calendar.set(Calendar.YEAR, 1776)

        def time = calendar.time

        System.out.println(time)
    }
}
```

No System.out.blah.blah...

```groovy
// PrintIndependenceDay.groovy



public class PrintIndependenceDay {

    public static void main(String[] args) {
        def calendar = Calendar.instance
        calendar.clear()
        calendar.set(Calendar.MONTH, Calendar.JULY)
        calendar.set(Calendar.DATE, 4)
        calendar.set(Calendar.YEAR, 1776)

        def time = calendar.time

        println(time)
    }
}
```

No Class...


```groovy
// PrintIndependenceDay.groovy

def calendar = Calendar.instance
calendar.clear()
calendar.set(Calendar.MONTH, Calendar.JULY)
calendar.set(Calendar.DATE, 4)
calendar.set(Calendar.YEAR, 1776)

def time = calendar.time

println(time)
```

Optional Parens...


```groovy
// PrintIndependenceDay.groovy

def calendar = Calendar.instance
calendar.clear()
calendar.set Calendar.MONTH, Calendar.JULY
calendar.set Calendar.DATE, 4
calendar.set Calendar.YEAR, 1776

def time = calendar.time

println time
```

Lets Go Meta...

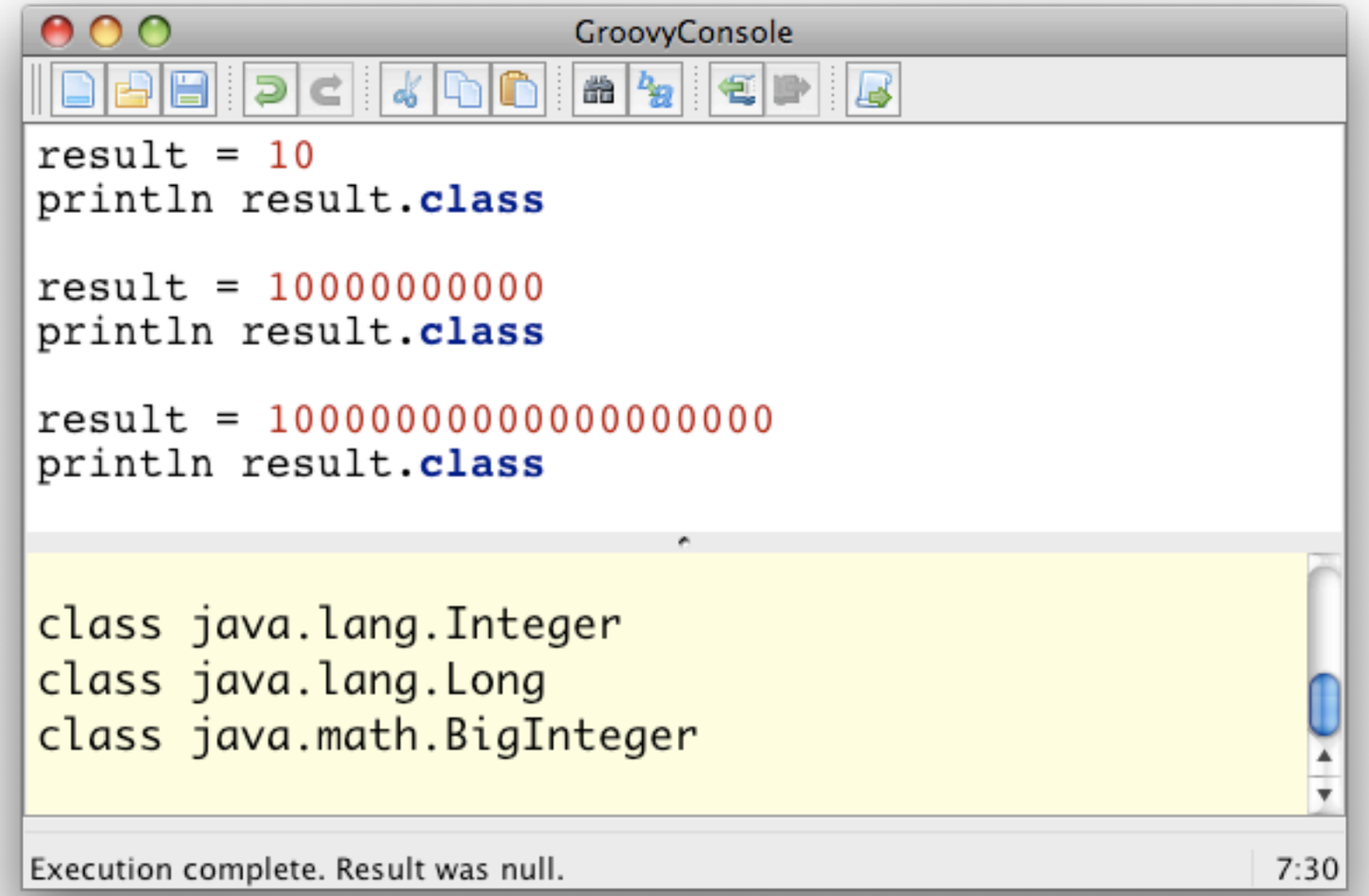

```groovy
// PrintIndependenceDay.groovy

def calendar = Calendar.instance
calendar.with {
    clear()
    set MONTH, JULY
    set DATE, 4
    set YEAR, 1776
    println time
}
```

Lets Compare...

```java
// PrintIndependenceDay.java

import java.util.Calendar;
import java.util.Date;

public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.clear();
        calendar.set(Calendar.MONTH, Calendar.JULY);
        calendar.set(Calendar.DATE, 4);
        calendar.set(Calendar.YEAR, 1776);

        Date time = calendar.getTime();

        System.out.println(time);
    }
}
```

```groovy
// PrintIndependenceDay.groovy

def calendar = Calendar.instance
calendar.with {
    clear()
    set MONTH, JULY
    set DATE, 4
    set YEAR, 1776
    println time
}
```

Everything Is An Object



Groovy Strings

- Single quoted Strings are java.lang.String
- Double quoted Strings are "GStrings"
  - may contain embedded Groovy code

```
aString = 'This is a String'

answer = 42
aGString = "The answer is ${answer}."
```

Groovy Strings

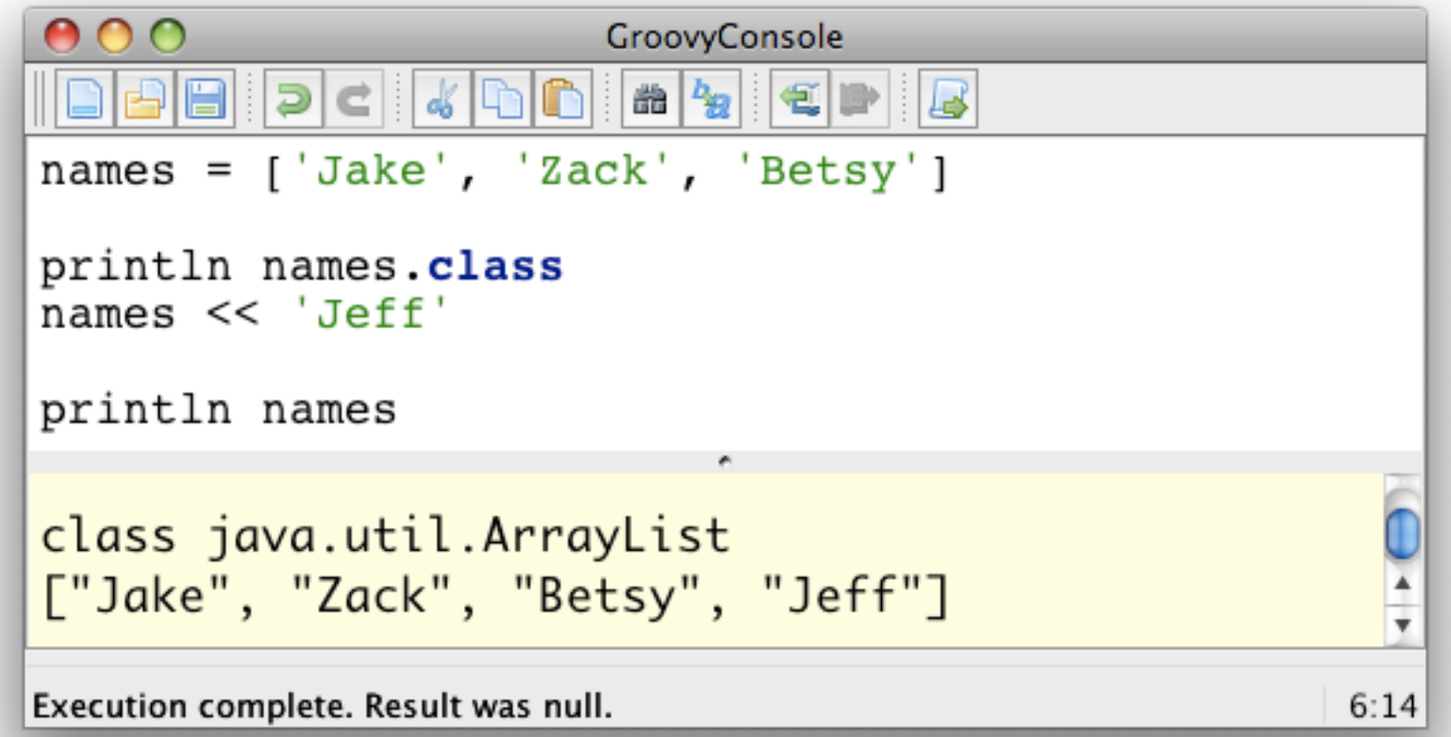
- Strings May Be Referenced Using [ ]

```
message = 'Groovy Is Cool'

println message[0]       // G
println message[-4]      // C
println message[0..5]    // Groovy
println message[-4..-1] // Cool
println message[-1..-4] // looC
```

Groovy Collections

- Groovy Collections Are Standard java.util.Collections
- Groovy Adds Many Useful Methods To Existing Collections
- Many Common Tasks Are Much More Simple In Groovy Compared To Java

Groovy List


GroovyConsole
names = ['Jake', 'Zack', 'Betsy']
println names.class
names << 'Jeff'
println names
class java.util.ArrayList
["Jake", "Zack", "Betsy", "Jeff"]
Execution complete. Result was null.
6:14

Groovy Maps

GroovyConsole

```
myMap = [name:'Jeff', language:'Groovy']
myMap.town = 'St. Charles'
myMap['company'] = 'G2One'

println myMap.getClass()
println myMap.company

// what is happening with this next line?...
println myMap.class
```

```
class java.util.LinkedHashMap
G2One
null
```

Execution complete. Result was null. 8:37

Groovy Beans

- Groovy Beans / POGOs
- Similar To POJOs
  - ...but groovier
  - eliminates boilerplate code

POJO

```java
public class Person {
    private String firstName;
    private String lastName;

    public Person() {
    }

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
```

POJO

```
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}
```

POJO

- Modern Java IDEs Generate Most Of That Code
  - developer declares fields
  - IDE generates constructors
  - IDE generates getters/setters

If the IDE can generate all of that code, why can't the compiler or the

Groovy Beans

- Groovy Beans Eliminate All Of The Boilerplate Code
- No Need To Write Getters/Setters
- Seldom Need To Write Constructors

Groovy Beans

```
class BaseballTeam {
    def cityName
    def teamName
}
```

```
myTeam = new BaseballTeam(teamName:'Cardinals',
                          cityName:'St. Louis')

println myTeam.teamName
println myTeam.cityName
```

Groovy Beans

- Property Access Looks Like Field Access

```
myTeam = new BaseballTeam()

// myTeam.setTeamName('Cardinals')
myTeam.teamName = 'Cardinals'

// myTeam.setCityName('St. Louis')
myTeam.cityName = 'St. Louis'

// println myTeam.getTeamName()
println myTeam.teamName
```

Groovy Beans

```groovy
class BaseballTeam {
    def cityName
    def teamName

    def getDisplayName() {
        "${cityName} ${teamName}"
    }
}
```

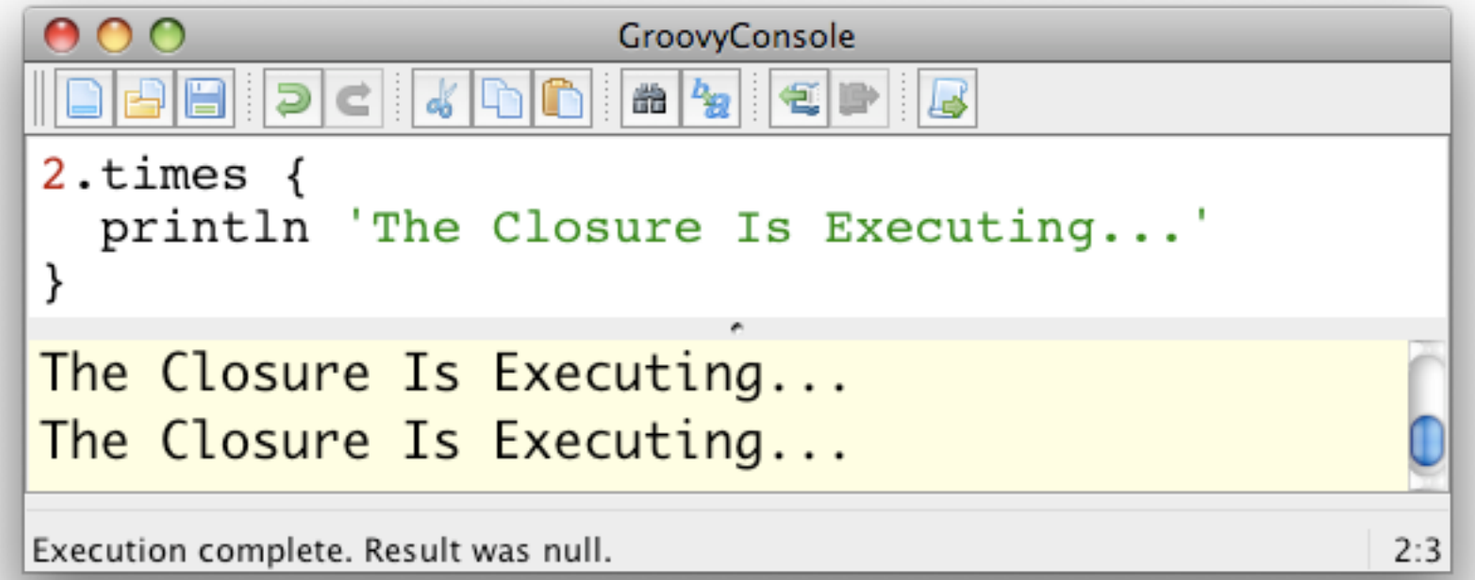
```groovy
myTeam = new BaseballTeam()

myTeam.teamName = 'Cardinals'
myTeam.cityName = 'St. Louis'

// println myTeam.getDisplayName()
println myTeam.displayName
```

Closures

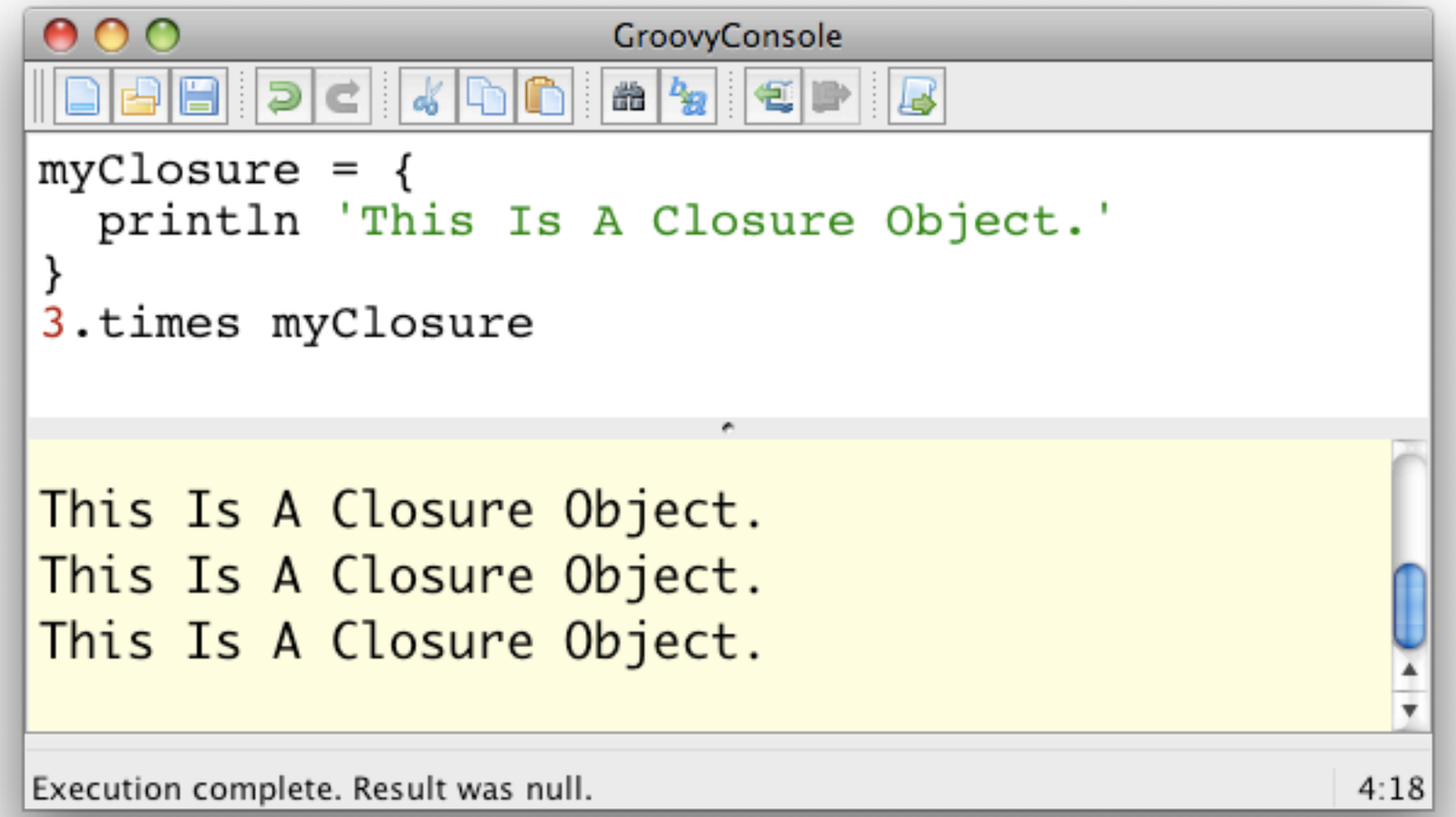
- A Block Of Code
- May Be Passed As Arguments
- May Accept Parameters
- May Return A Value
- Much More Powerful Than Anonymous Inner Classes

Closures

- Groovy Adds A 'times' Method To Number
- The 'times' Method Accepts A Closure As An Argument



Closures


GroovyConsole
myClosure = {
println 'This Is A Closure Object.'
}
3.times myClosure
This Is A Closure Object.
This Is A Closure Object.
This Is A Closure Object.
Execution complete. Result was null.
4:18

Closures

- Closures May Declare An Argument List

GroovyConsole

```
3.times { firstArg ->
  println "The argument is ${firstArg}"
}
```

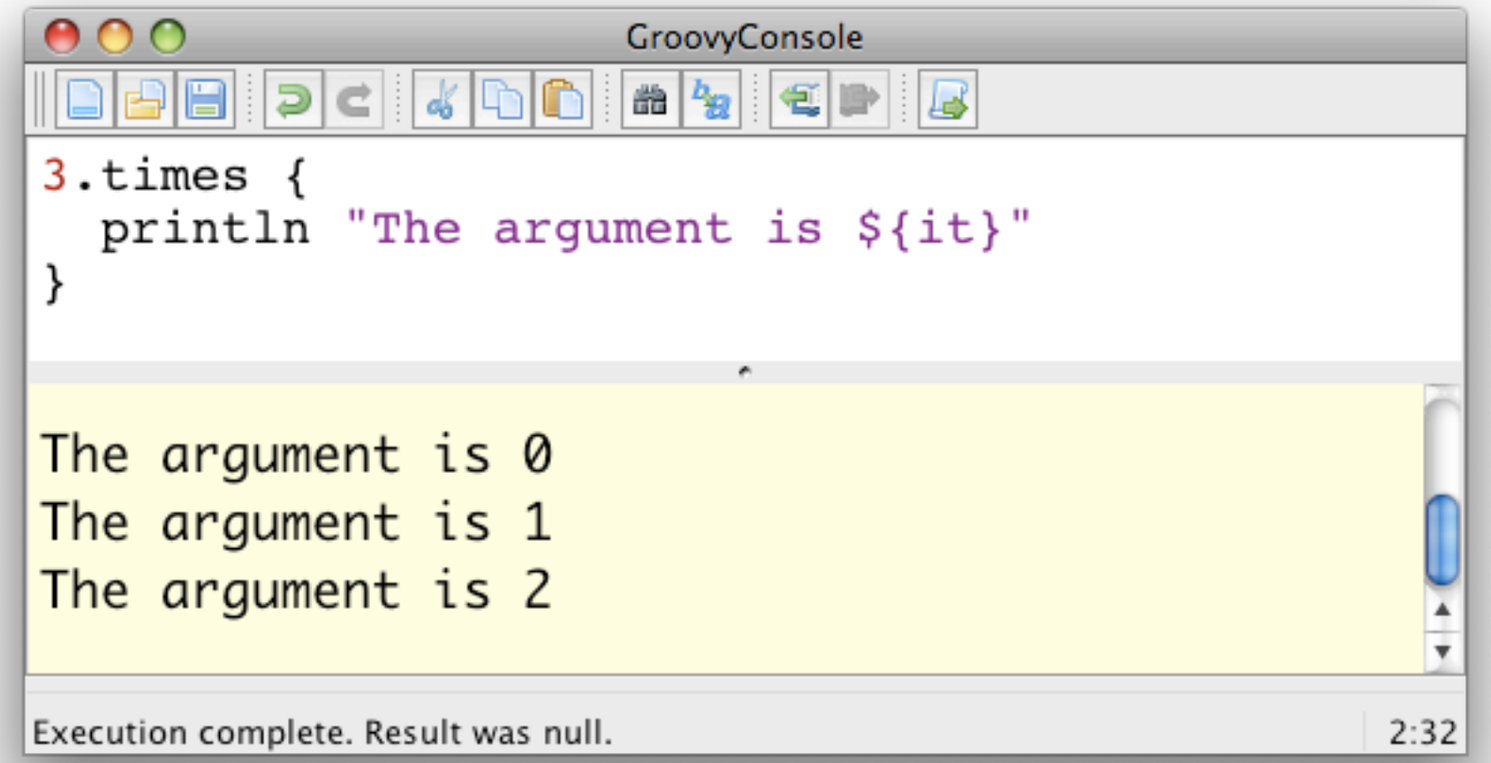
```
The argument is 0
The argument is 1
The argument is 2
```

Execution complete. Result was null. 3:2

the times method is passing an argument into the closure

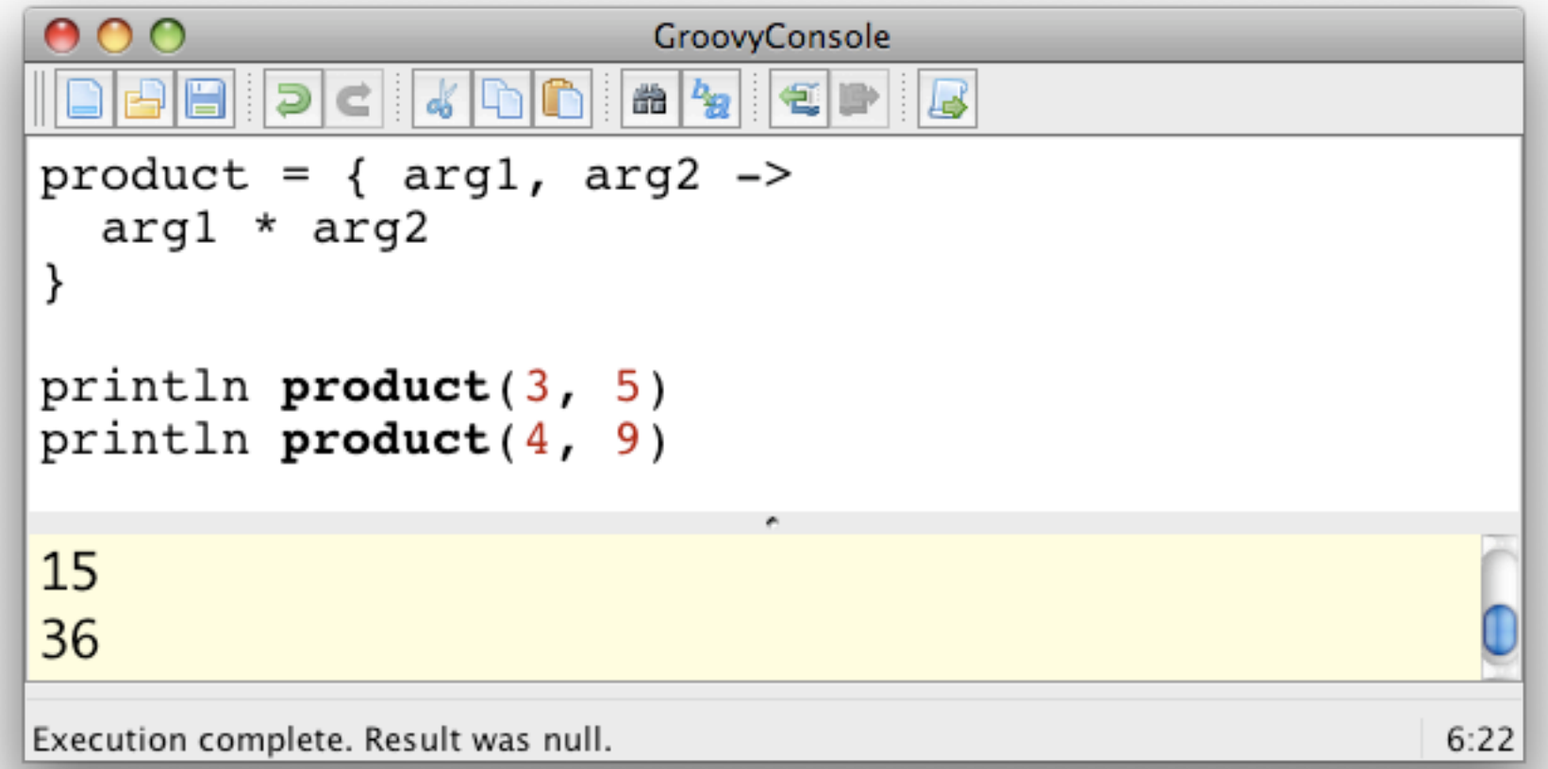
Closures

- The Implicit 'it' Argument



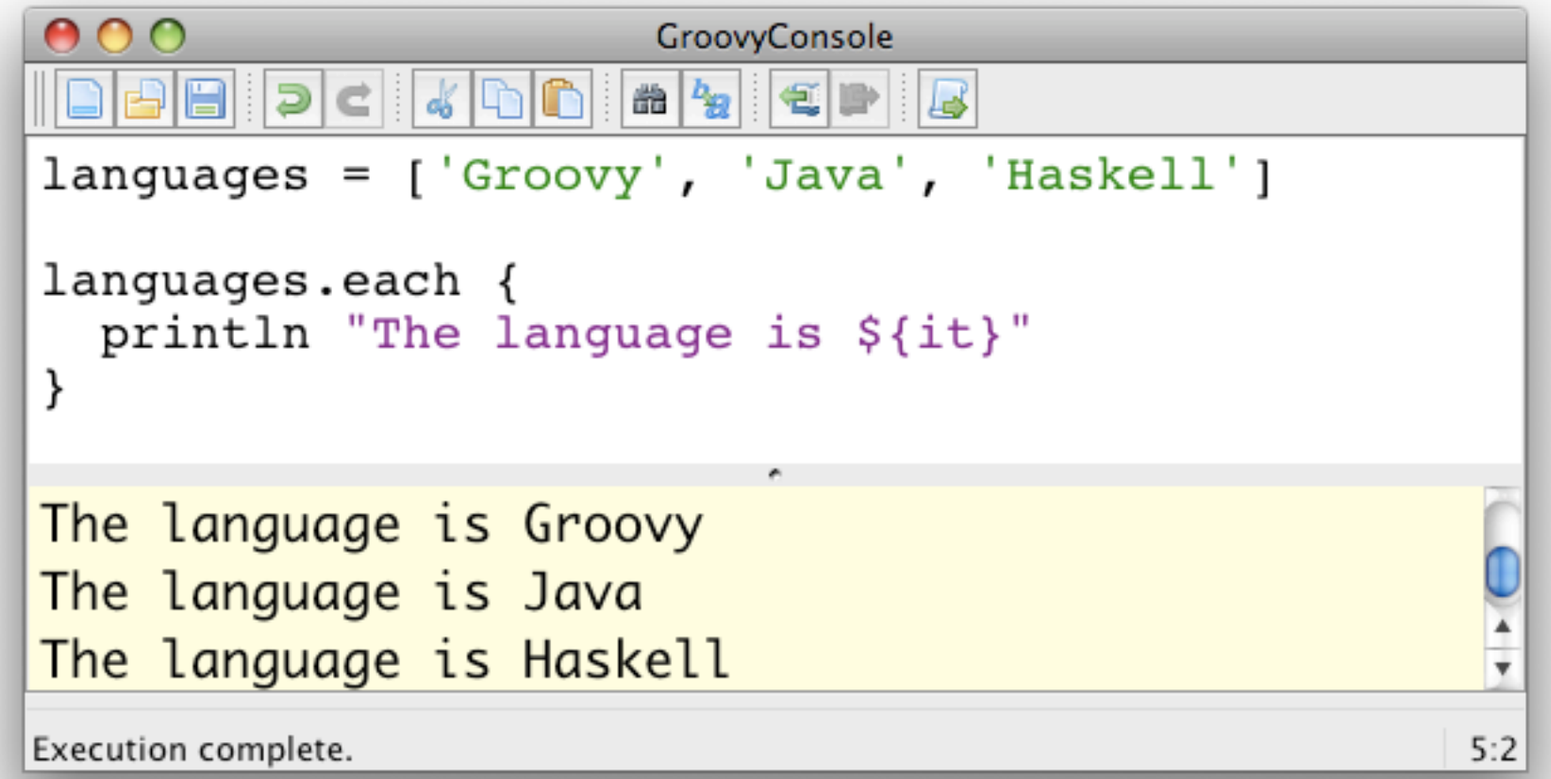
Closures

- Closures May Accept Multiple Arguments

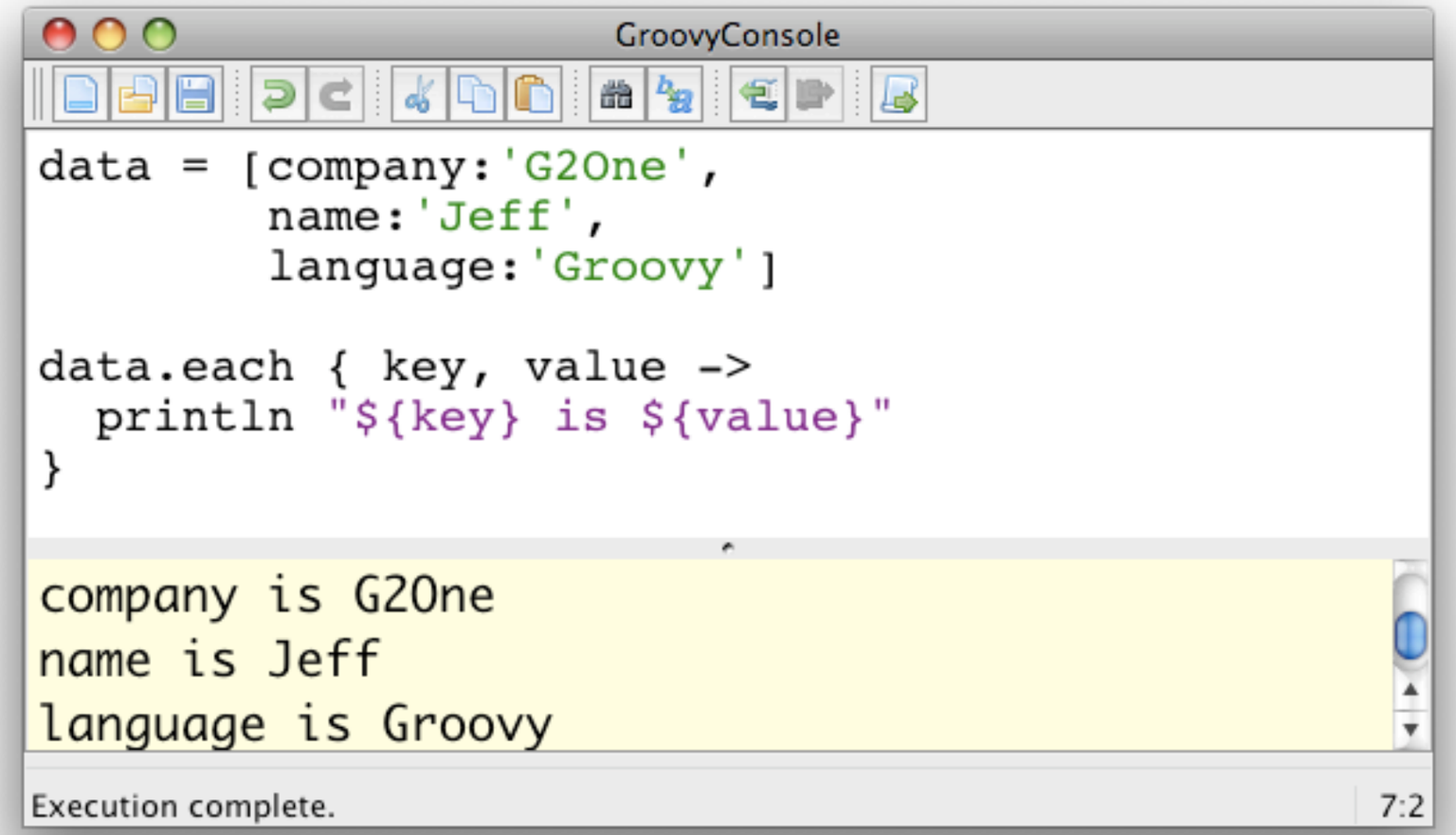


Closures

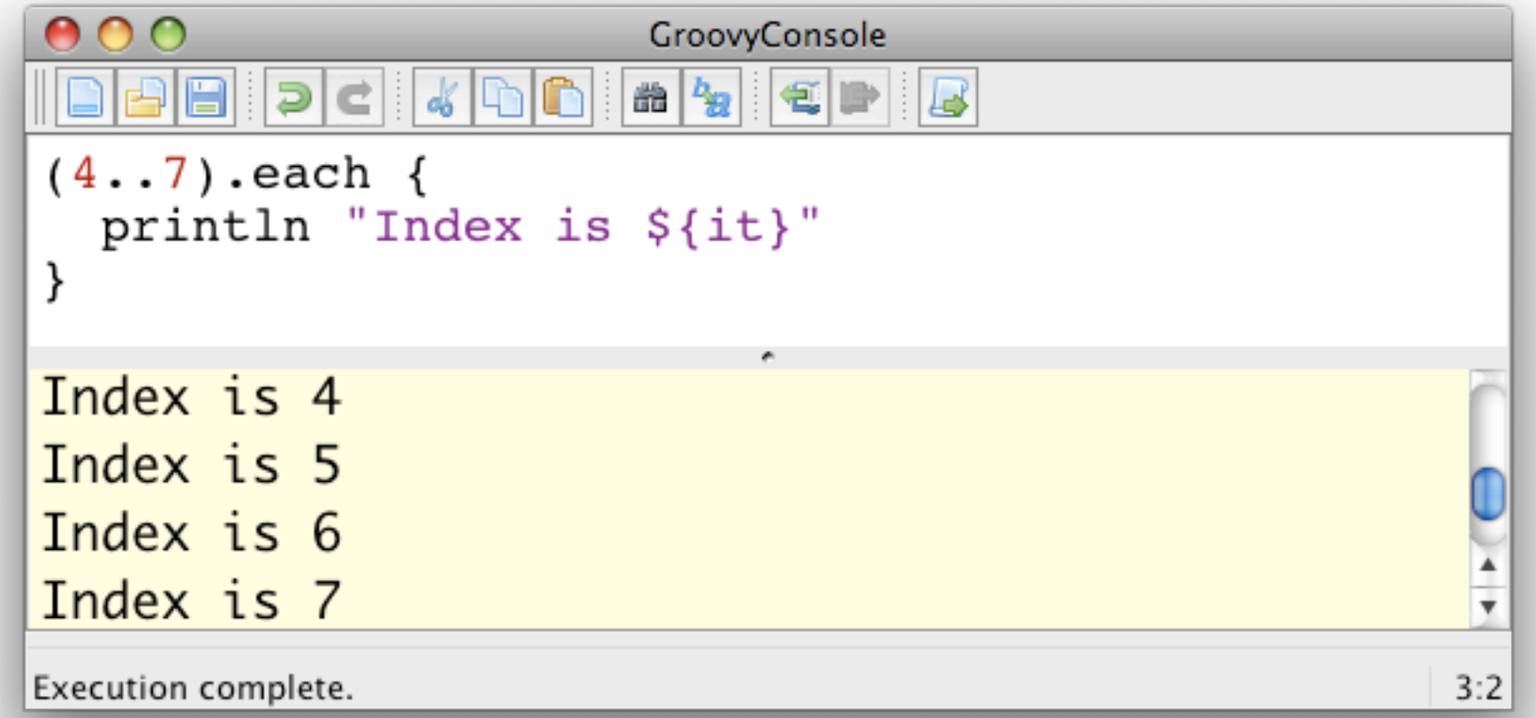
- Closures Simplify Collection Iteration



Closures


GroovyConsole
data = [company:'G2One',
name:'Jeff',
language:'Groovy']
data.each { key, value ->
println "${key} is ${value}"
}
company is G2One
name is Jeff
language is Groovy
Execution complete.
7:2

Closures


GroovyConsole
(4..7).each {
println "Index is ${it}"
}
Index is 4
Index is 5
Index is 6
Index is 7
Execution complete.
3:2

Builders

- Builders Are A Powerful Concept
- Metaprogramming Makes Builders A Snap In Groovy
- Several Builders Are Bundled With Groovy
  - SwingBuilder, MarkupBuilder, etc...
- You Can Write Your Own

MarkupBuilder

```groovy
def xmlBuilder =
  new groovy.xml.MarkupBuilder()

xmlBuilder.teams {
  teams {
    team(name: 'Cardinals') {
      player('Albert Pujols')
      player('Ozzie Smith')
    }
    team(name:'Rams') {
      player('Torry Holt')
    }
  }
}
```

```xml
<teams>
  <teams>
    <team name='Cardinals'>
      <player>Albert Pujols</player>
      <player>Ozzie Smith</player>
    </team>
    <team name='Rams'>
      <player>Torry Holt</player>
    </team>
  </teams>
</teams>
```

Q & A