Groovy Workshop 2016

Daniel Hyun

2016 July 27

Table of Contents

1. Intro

1.1. What is Groovy

- Questionable name
- Awesome language
- Flat learning curve (from java)

1.2. Install Groovy

Requirements

- Download and install JDK
- Set Environment variable `JAVA_HOME` to location of JDK path
- Download and install SDKMAN!

Installing Groovy via SDKMAN!:

bash

```
$ sdk install groovy
```

1.3. Hello World

groovySh

```
$ groovySh
Groovy Shell (2.4.6, JVM: 1.8.0_72)
Type ':help' or ':h' for help.
-------------------------------------------------------------------------------
groovy:000> println 'Hello, World!'
Hello, World!
===> null
```

Groovy Console

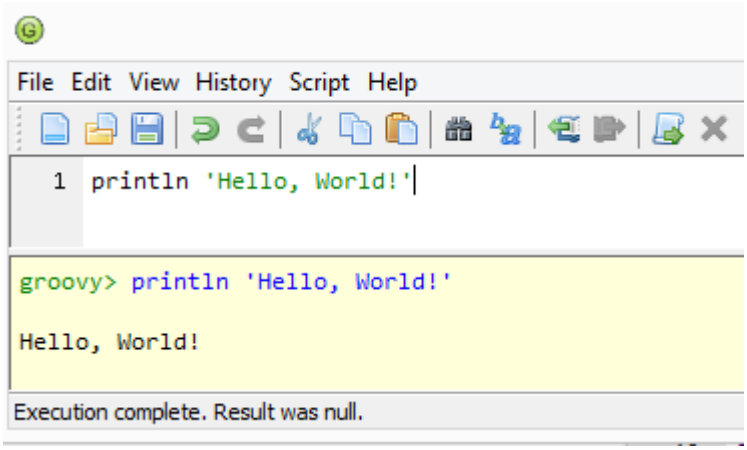

Figure 1. groovyConsole

2. Syntax

2.1. Java Like

```
public class HelloWorldGroovy {
  public static void main(String[] args) {
    System.out.println("Hello, Groovy!");
  }
}
```

This is 100% valid Groovy code

```
$ groovy HelloWorldGroovy.groovy
Hello, Groovy!
```

2.2. Public by default

```
class HelloWorldGroovy {
  void main(String[] args) {
    System.out.println("Hello, Groovy!");
  }
}
```

2.3. Semicolons optional

```
class HelloWorldGroovy {
  void main(String[] args) {
    System.out.println("Hello, Groovy!")
  }
}
```

2.4. Parenthesis are optional as well

```
class HelloWorldGroovy {
  void main(String[] args) {
    System.out.println "Hello, Groovy!"
  }
}
```

2.5. Just get to the point

```
class HelloWorldGroovy {
  void main(String[] args) {
    println "Hello, Groovy!"
  }
}
```

2.6. Just say what you need

```
println "Hello, Groovy!"
```

3. Strings

3.1. Declare a String

```
String hello = "Hello"
```

Double quoted Strings in Groovy are instances of `GString`. `GString` allows for String interpolation.

```
String hello = "Hello"
String greeting = "$hello World" // evaluates to "Hello World"
```

3.2. Alternative String declarations

Single Quote

```
String hello = 'Hello'
```

Single quote strings are *not* interpolated

Slashy String

```
String hello = /Hello/
```

Dollar Slashy String

```
String hello = $/Hello/$
```

3.3. Multiline Strings

You can declare multiline Strings using triple single quote, tripe double quote, slashy and dollary slashy strings.

```
String pizza = '''
Pizza in the morning
Pizza in the evening
Pizza at supper time
'''

String groovy = """
Groovy in the morning
Groovy in the evening
Groovy at supper time
"""

String rocket = /
Prepare for trouble
Make it double
/

String scifi = $/
Use the force Harry
-- Gandalf
/$

println pizza
println groovy
println rocket
println scifi
```

3.4. String manipulation

```
String hello = 'Hello'
String message = hello + ' ' + 'World!' ①

println message // Hello World!

String ello = hello - 'H' ②
message = "'$ello guvna" ③

println message //'ello guvna'
```

① Regular old String concatenation

② Remove first instance of 'H' from String `hello`

③ String interpolation

GNU tr like behavior

```
$ echo HELLO | tr [A-Z] [a-z] ①
hello
```

Groovy String `tr`

```
String hello = 'HELLO'.tr(/[A-Z]/, /[a-z]/) ①
```

① Replace all upper case to lower case

3.5. String formatting

```
String message = 'Welcome to GR8ConfUS'

String centered    = message.center(24, '-')   // --Welcome to GR8ConfUS--
String leftPadded  = message.padLeft(24, '-')  // ----Welcome to GR8ConfUS
String rightPadded = message.padRight(24, '-') // Welcome to GR8ConfUS----

println centered
println leftPadded
println rightPadded
```

3.6. Misc functions

```
String message = 'Hello'
String amplified = message * 10

println amplified // HelloHelloHelloHelloHelloHelloHelloHelloHelloHello
```

4. Lists

4.1. Creating lists is easy

```
List emptyList = []
List<String> conferences = ['Greach', 'GR8Conf', 'G3', 'KR8RConf']
```

4.2. Accessing elements of list

```
List<String> conferences = ['Greach', 'GR8Conf', 'G3', 'KR8RConf']

// index is size of list - 1
assert conferences[0] == 'Greach'
assert conferences[1] == 'GR8Conf'
assert conferences[2] == 'G3'
assert conferences[3] == 'KR8RConf'

// you can use negative indices as well
assert conferences[-1] == 'KR8RConf'
assert conferences[-2] == 'G3'
assert conferences[-3] == 'GR8Conf'
assert conferences[-4] == 'Greach'
```

4.3. Mutating the list

```
List<String> conferences = ['Greach', 'GR8Conf', 'G3', 'KR8RConf']

conferences << 'G3 Summit' ①
assert conferences[-1] == 'G3 Summit'
assert conferences == ['Greach', 'GR8Conf', 'G3', 'KR8RConf', 'G3 Summit']

conferences = conferences - 'Greach' ②

assert conferences[0] == 'GR8Conf'
assert conferences == ['GR8Conf', 'G3', 'KR8RConf', 'G3 Summit']
```

① Add `G3 Summit` to list

② Remove `Greach` from list and assign resulting list to original list

5. Maps

5.1. Creating maps is easy

```
Map emptyMap = [:]

Map<String, String> pokemon = [water: 'Squirtle', flying: 'Pidgey', bug: 'Pinsir'] ①
```

① Map literal notation with `key: 'value'` syntax

5.2. Accessing map entries

```
Map<String, String> pokemon = [water: 'Squirtle', flying: 'Pidgey', bug: 'Pinsir']

assert pokemon['water'] == 'Squirtle' ①
assert pokemon.water == 'Squirtle' ②
```

① Access using bracket notation

② Access using dot notation

5.3. Mutating the map

```
Map<String, String> pokemon = [water: 'Squirtle', flying: 'Pidgey', bug: 'Pinsir']

pokemon.water = 'Magikarp' ①

pokemon.electric = 'Pikachu' ②

println pokemon // [water:'Magikarp;, flying:'Pidgey', bug:'Pinsir',
electric:'Pikachu']
```

① Update `water` entry of map

② Add new entry to map

6. Range

A range represents a set of elements between a lower and upper bound.

6.1. Declaring a range

```
Range oneToTen = 1..10
assert oneToTen == [1, 2 ,3, 4, 5, 6, 7, 8, 9, 10]
```

6.2. Testing if element is within a range

```
Range oneToTen = 1..10

assert 5 in oneToTen
```

Ranges are discrete, not continuous! Must use < > to assess continuous values

```
Range oneToTen = 1..10

assert !(5.5 in oneToTen)

assert oneToTen.from < 5.5 && 5.5 < oneToTen.to
```

6.3. Using ranges with Lists

```
List<String> names = ['Kyle', 'Craig', 'Dane']

List<String> funny = names[0..1] ①
assert funny == ['Kyle', 'Craig']

List<String> unfunny = names[-1..-1] ②
assert unfunny == ['Dane']
```

① Take first two elements of list as a new sublist

② Make a new list from last element

7. Closures

A bit of code that captures environment in which it's created. It can take arguments.

7.1. Creating closures

```
Closure c = { println 'Hello' }

c() ①
c.call() ②
```

① Invoke closure using parenthesis operator

② Invoke by invoking `Closure#call` method

7.2. Closures can return values

```
Closure greeter = { return 'Hello' }

assert greeter() == 'Hello'
```

7.3. Closures are aware of enclosing context

```
String message = 'Hello'
Closure greeter = { return message }

assert greeter() == 'Hello'
```

7.4. Closures can take arguments

```
Closure greeter = { message -> return "$message to you" }

assert greeter('Hello') == 'Hello to you'
```

7.5. Closures can be curried

```
Closure greeter = { message -> return "$message to you" }

Closure englishGreeter = greeter.curry('Hello') ①

assert englishGreeter() == 'Hello to you' ②

Closure spanishGreeter = greeter.curry('Hola') ①

assert spanishGreeter() == 'Hola to you' ②
```

① Invoke `Closure#curry` to return a new Closure with argument frozen in place

② Invoke new Closure with frozen value

7.6. Closures can be composed

```
Closure greeter = { message -> return "$message to you" }
Closure formatter = { s -> s.center(20, '*') }

Closure greetThenFormat = formatter << greeter

assert greetThenFormat('Hello') == '****Hello to you****'

Closure formatThenGreet = formatter >> greeter

assert formatThenGreet('Hello') == '*******Hello******** to you'
```

7.7. Using Closures with Lists

```
List<String> names = ['Kyle', 'Craig', 'Dane']

names.each { name -> println name } ①

List<String> upperCased = names.collect { name -> name.toUpperCase() } ②
assert upperCased == ['KYLE', 'CRAIG', 'DANE']

List<String> funny = names.findAll { name -> name[0] in ['K', 'C'] } ③
assert funny == ['Kyle', 'Craig']
```

① Consume each element in List and print

② Map each element to upper case and return new List

③ Filter for names starting with K or C in List

7.8. Using Spread operator with List and Closures

For method or closures that take multiple parameters, you can apply a list as arguments to the method or closure

```
String conferenceAttender(String person, String conference, int year) {
 return "Welcome $person to $conference $year!"
}

List<String> me = ['Dan', 'GR8ConfUS', 2016]

assert conferenceAttender('Dan', 'GR8ConfUS', 2016) == conferenceAttender(*me)

assert conferenceAttender('Dan', 'GR8ConfUS', 2016) == 'Welcome Dan to GR8ConfUS
2016!'
assert conferenceAttender(*me) == 'Welcome Dan to GR8ConfUS 2016!'
```

8. Objects

Objects have nice defaults in Groovy

A complete POGO

```
class Pokemon {
  String name
  String type
}
```

This class definition creates setters and getters

```
class Pokemon {
  String name
  String type
}

Pokemon rattata = new Pokemon()
rattata.setName('Rattata')
rattata.setType('Normal')

assert rattata.getName() == 'Rattata'
assert rattata.getType() == 'Normal'
```

These getters and setters can be invoked via dot notation

```
class Pokemon {
  String name
  String type
}

Pokemon rattata = new Pokemon()
rattata.name = 'Rattata'
rattata.type = 'Normal'

assert rattata.name == 'Rattata'
assert rattata.type == 'Normal'
```

You can also use map constructor

```
class Pokemon {
  String name
  String type
}

Pokemon rattata = new Pokemon(name: 'Rattata', type: 'Normal')

assert rattata.name == 'Rattata'
assert rattata.type == 'Normal'
```

8.1. TupleConstructor

```groovy
import groovy.transform.*

@TupleConstructor
class Pokemon {
  String name
  String type
}

Pokemon rattata = new Pokemon('Rattata', 'Normal')

assert rattata.name == 'Rattata'
assert rattata.type == 'Normal'
```

8.2. ToString

```groovy
import groovy.transform.*

@ToString
@TupleConstructor
class Pokemon {
  String name
  String type
}

Pokemon rattata = new Pokemon('Rattata', 'Normal')

assert rattata.name == 'Rattata'
assert rattata.type == 'Normal'

assert rattata.toString() == 'Pokemon(Rattata, Normal)'
```

8.3. EqualsAndHashCode

```
import groovy.transform.*

@EqualsAndHashCode
@ToString
@TupleConstructor
class Pokemon {
  String name
  String type
}

Pokemon r1 = new Pokemon('Rattata', 'Normal')
Pokemon r2 = new Pokemon('Rattata', 'Normal')

assert r1 == r2
```

8.4. Canonical

Canonical == EqAndHashCode + ToString + TupleConstructor

```
import groovy.transform.*

@Canonical
class Pokemon {
  String name
  String type
}

Pokemon r1 = new Pokemon('Rattata', 'Normal')
Pokemon r2 = new Pokemon('Rattata', 'Normal')

assert r1 == r2
```

9. Useful annotations

9.1. CompileStatic

Enables compile time static checks

```
import groovy.transform.CompileStatic
import groovy.transform.Canonical

@Canonical
@CompileStatic
class Pokemon {
  String name
  String type
}

Pokemon r = new Pokemon('Rattata', 'Normal')
```

9.2. Memoized

Caches return value for a given set of arguments to a method or closure call

```
import groovy.transform.Memoized

@Memoized
int fibonacci(int i) {
  if (i == 0) return 0
  if (i == 1) return 1
  return fibonacci(i - 1) + fibonacci (i - 2)
}

def start = System.nanoTime()
fibonacci(15)
println System.nanoTime() - start // 1161994


start = System.nanoTime()
fibonacci(15)
println System.nanoTime() - start // 403949
```

10. File IO

Before Java 7's try-with-resources, Groovy had "execute around" patterns for handling File IO.

Groovy provides methods for handling resources like Writers or Readers and takes care of making sure they're closed no matter what.

10.1. Reading and writing a file

```
File secrets = new File('bank-secrets.txt')

secrets.withWriter { writer -> ①
  writer.writeLine 'E Corp: $1 billion'
  writer.writeLine 'E Corp: $2 billion'
  writer.writeLine 'E Corp: $3 billion'
}

secrets.withReader { reader -> ②
  while (line = reader.readLine()) {
    println "Found secret: $line"
  }
}

secrets.delete()
```

① Use `withWriter` with Closure that writes 3 lines to the file

② Use `withReader` to read lines from the file

11. Handy Operators

11.1. Null safe dereference

Given two classes we'd like to safely calculate attack probability

Before

```
class Pokemon {
  String name
  String type
  Attack attack
}

class Attack {
  String name
  String type
  int damage
}

double calculateAttack(Pokemon pkmn) {
  if (pkmn != null) {
    Attack attack = pkmn.attack
    if (attack != null) {
      return new Random().nextDouble() * attack.damage
    }
  }
  return -1
}

calculateAttack(new Pokemon(attack: new Attack(damage:5)))
```

After

```
class Pokemon {
  String name
  String type
  Attack attack
}

class Attack {
  String name
  String type
  int damage
}

double calculateAttack(Pokemon pkmn) {
  int baseDamage = pkmn?.attack?.damage
  return baseDamage ? baseDamage * new Random().nextDouble() : 0
}

calculateAttack(new Pokemon(attack: new Attack(damage:5)))
```

11.2. Elvis

A terse way to return a default value in face of a falsey value

```
String greeter (String message) {
  return message ?: 'Hello'
}

assert greeter() == 'Hello'
assert greeter('Howdy') == 'Howdy'
```

Groovier attack damage calculation

```
class Pokemon {
  String name
  String type
  Attack attack
}

class Attack {
  String name
  String type
  int damage
}

double calculateAttack(Pokemon pkmn) {
  int baseDamage = pkmn?.attack?.damage ?: 0
  return baseDamage * new Random().nextDouble()
}

calculateAttack(new Pokemon(attack: new Attack(damage:5)))
assert calculateAttack(new Pokemon()) == 0
```

11.3. Star dot

Executes method on every element in a List

```
class Pokemon {
  String name
  String type
  Attack attack
}

class Attack {
  String name
  String type
  int damage
}

double calculateAttack(Pokemon pkmn) {
  int baseDamage = pkmn?.attack?.damage ?: 0
  return baseDamage * new Random().nextDouble()
}

List<Pokemon> pkmn = (1..10).collect { i -> ①
  new Pokemon(
    name: 'Bulbasaur',
    type: 'Grass',
    attack: new Attack(
      name: 'Leaf Cutter',
      type: 'Grass',
      damage: i
    )
  )
}

List<String> names = pkmn*.name ②
assert names == ['Bulbasaur', 'Bulbasaur', 'Bulbasaur', 'Bulbasaur', 'Bulbasaur',
'Bulbasaur', 'Bulbasaur', 'Bulbasaur', 'Bulbasaur', 'Bulbasaur']

List<Double> damages = pkmn*.attack.damage ③

assert damages == [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

① Create a range (1..10) and transform each element into a Pokemon by setting the value as the damage for the attack

② Use Star dot to invoke `Pokemon#getName` on each element of the list and return as new list

③ Use Star dot to invoke `Pokemon#getAttack#getDamage` on each element of list and return as new list

11.4. Method reference

Methods and closures are first class objects in Groovy

Use `.&` to get a handle to the method or closure from an object

```
class Pokemon {
  String name
  String type
  Attack attack
}

Pokemon p = new Pokemon(name: 'Eevee')

Closure nameGetter = p.&getName ①

assert nameGetter() == 'Eevee'
```

① Use `.&` to dereference method as a closure to pass around

Closures can be passed by value of reference to methods

```
class Pokemon {
  String name
  String type
  Attack attack
}

class Attack {
  String name
  String type
  int damage
}

double calculateAttack(Pokemon pkmn) {
  int baseDamage = pkmn?.attack?.damage ?: 0
  return baseDamage * new Random().nextDouble()
}

List<Pokemon> pkmn = (1..10).collect { i ->
  new Pokemon(
    name: 'Bulbasaur',
    type: 'Grass',
    attack: new Attack(
      name: 'Leaf Cutter',
      type: 'Grass',
      damage: i
    )
  )
}

List<Double> damages = pkmn.collect(this.&calculateAttack) ①

println damages // [0.9729413212618172, 1.992348079720605, 1.0092590053588697,
3.4810779400638983, 3.955667167493159, 2.063303014744924, 5.943832726036444,
1.05018656250666, 6.461072896131895, 4.267121915844054]

assert damages
```

① In the absence of a owning class, you can use `this` to dereference the method

11.5. Spaceship

Used for comparisons

```
assert 1 <=> 1 == 0
assert 1 <=> 2 == -1
assert 2 <=> 1 == 1
```

Really handy for things like sorting lists

```
class Pokemon {
  String name
  String type
  Attack attack
}

class Attack {
  String name
  String type
  int damage
}

double calculateAttack(Pokemon pkmn) {
  int baseDamage = pkmn?.attack?.damage ?: 0
  return baseDamage * new Random().nextDouble()
}

List<Pokemon> pkmn = (1..10).collect { i ->
  new Pokemon(
    name: 'Bulbasaur',
    type: 'Grass',
    attack: new Attack(
      name: 'Leaf Cutter',
      type: 'Grass',
      damage: new Random().nextInt(10) + 1 ①
    )
  )
}

pkmn.sort { p1, p2 ->
 return p1.attack.damage <=> p2.attack.damage ②
}

println pkmn*.attack.damage // [1, 3, 3, 3, 4, 4, 5, 6, 6, 7]
```

① Randomize attack damage per pokemon

② Sort list by attack damage ascending

12. Training your Groovy-fu

http://www.groovy-koans.org/

https://github.com/nadavc/groovykoans

13. Real World

14. Web service consumption

Working with the web is a breeze in Groovy

Download source code for gr8conf.us

```
String html = 'http://gr8conf.us'.toURL().text

assert html.length() > 0

assert html.startsWith('<!doctype html> <html class="no-js" ng-app="gr8conf')
```

Scan HTML

```
@Grab('org.jsoup:jsoup:1.6.1')

import org.jsoup.Jsoup
import org.jsoup.nodes.Document

String html = 'http://gr8conf.us'.toURL().text

Document doc = Jsoup.parse(html) ①

List<String> links = doc.select('a').collect { link -> ②
  return link.attr('href')
}

println links
```

① Parse raw html to a structured Document

② Use CSS selector `a` to find all anchor nodes, iterate through them and extract the `href` attribute from each

Learn more about JSOUP

Consuming JSON

```
import groovy.json.JsonSlurper

String rawJson = 'http://www.reddit.com/r/pokemongo.json'.toURL().getText
(requestProperties: ['User-Agent': 'groovy-sample']) ①

JsonSlurper slurper = new JsonSlurper()

def json = slurper.parseText(rawJson) ②

List<String> summaries = json.data.children*.data.collect { post -> ③
  return """
  Author: $post.author
  Score: $post.score
  Title: $post.title
  """
}

println summaries.join('-' * 20)
```

① Set user-agent before getting response

② Use `JsonSlurper` to parse our raw text into a structured object graph

③ Use dot notation and spread dot operator to get a high level summary of endpoint

Sample output

```
  Author: Juxlos
  Score: 3885
  Title: Welcome to /r/PokemonGo!
  --------------------
  Author: PokemonGOmods
  Score: 148
  Title: "How do I..." and Bugs Megathread - 26/7
  --------------------
  Author: Kyurun
  Score: 5199
  Title: Guarantee 1000CP+ Evolutions
  --------------------
  Author: Brutal_Angel
  Score: 1294
  Title: Can we at least give credit that for being free app there aren't Ads being
jammed into our throat.
  --------------------
  Author: majorgoober
  Score: 4808
  Title: Casually browsing SDCC coverage when...
  --------------------
  Author: Xiiao
  Score: 5136
```

```
  Title: Your coffee is coming, please wait
  --------------------
  Author: Borgifornia
  Score: 4632
  Title: I found a geoduo.
  --------------------
  Author: GreyEagle08
  Score: 3426
  Title: How I feel playing in the middle of nowhere.
  --------------------
  Author: tanzanitetnn
  Score: 2853
  Title: Instinct Players be like:
  --------------------
  Author: nayfw
  Score: 736
  Title: This has just been posted to my local PoGo page...
  --------------------
  Author: Pearlshine1494
  Score: 202
  Title: When get an excellent throw, but it breaks out at the last second
  --------------------
  Author: adoseofdanta
  Score: 1158
  Title: Syther is the Pidgey of La Jolla Cove (San Diego)
  --------------------
  Author: dalcowboiz
  Score: 660
  Title: I got rid of this thing you guys keep complaining about
  --------------------
  Author: silverdollaflapjacks
  Score: 3189
  Title: My sister saw a familiar scene playing pokemon go at the park this weekend
  --------------------
  Author: sebs8
  Score: 1646
  Title: Found this while I was hunting for a Scyther today.
  --------------------
  Author: Sleepwalks
  Score: 412
  Title: My friends and I keep finding Sandshrew in the ocean, when we take the ferry
or go to the beach. We call them Seashrew, so I decided to draw one.
  --------------------
  Author: Whosdaman
  Score: 6026
  Title: My small rural town has one Pokémon Go hotspot, the PD just posted these all
around that area
  --------------------
  Author: Siyliss
  Score: 4910
  Title: So the game glitched out the other night and make Rhydon and Pinsir look like
```

```
they were in a Godzilla movie lol. I laughed pretty hard when it happened. Rhydon
nailed it.
  --------------------
  Author: Noticemenot
  Score: 5055
  Title: Life Before Pokemon GO and After
  --------------------
  Author: Smileynator
  Score: 3350
  Title: Why Pokemon Go pisses off the developer inside me.
  --------------------
  Author: shpitzX
  Score: 5196
  Title: Oh! it's a cute Jigly..AAAHHHHH!!
  --------------------
  Author: ZuneNebula
  Score: 3021
  Title: The Secret of being the strongest trainer ever!!
  --------------------
  Author: mihitnrun
  Score: 132
  Title: My girlfriend has started calling transferring Pokémon "harvesting" - I just
told her we're going to go to the park for a Poké-walk and this was her response
  --------------------
  Author: 7omo
  Score: 111
  Title: Pokéstops need to give higher level trainers more pokéballs!
  --------------------
  Author: marshmahlow
  Score: 2584
  Title: We've now had more days of PokemonGO with the three step bug than we had with
the original tracking method (U.S.)
  --------------------
  Author: Kraigius
  Score: 1610
  Title: The search for intelligent life
  --------------------
  Author: MacAtack3
  Score: 286
  Title: An explanation of egg distance. A compsci friend of mine explained this to me
yesterday and my eggs have been easier to hatch for it. He said the ping time was ~6
seconds.
```

14.1. Browser automation

Geb allows you to programmatically control a browser

```
@Grab('org.gebish:geb-core:0.13.1')
@Grab('org.seleniumhq.selenium:selenium-firefox-driver:2.53.1')
@GrabExclude('org.codehaus.groovy:groovy-all')

import geb.Browser

Browser.drive {
  go 'http://gr8conf.us' ①

  js.exec ''' ②
  window.scrollBy(0,700)
  '''

  assert $('h1')[1].text() == 'Welcome to GR8Conf US 2016!' ③

  $('a[href$="speakers"]')[1].click() ④

  assert $('h1')[1].text() == 'Speakers' ⑤
}
```

① Navigate to GR8ConfUS website

② Execute Javascript to scroll down by 700px

③ Find second `h1` via css selector and assert contents

④ Find second `a` via css selector for speakers and click

⑤ Verify that the page has changed

More on Geb at http://gebish.org/

14.2. Pokemon Go!

[drake go] | *images/drake-go.gif*

Project source

```
@GrabResolver(name='jitpack', root='https://jitpack.io', m2Compatible='true')
@Grab('com.github.Grover-c13:PokeGOAPI-Java:master-SNAPSHOT')

import com.pokegoapi.api.PokemonGo
import com.pokegoapi.auth.PtcLogin

import okhttp3.OkHttpClient

OkHttpClient http = new OkHttpClient() ①

def creds = ['user', 'password'] ②
def auth = new PtcLogin(http).login(*creds) ③

PokemonGo go = new PokemonGo(auth, http) ④
```

① Create a new client (comes with PokemonGO Java API)

② Enter user credentials

③ Login using Pokemon Trainer Club auth endpoint using spread operator

④ Instantiate new `PokemonGo`

`PokemonGo` is the gateway to interacting with the Pokemon Go API

pokemongoapi.groovy

```
@GrabResolver(name='jitpack', root='https://jitpack.io', m2Compatible='true')
@Grab('com.github.Grover-c13:PokeGOAPI-Java:master-SNAPSHOT')

import com.pokegoapi.api.PokemonGo
import com.pokegoapi.auth.PtcLogin

import okhttp3.OkHttpClient

OkHttpClient http = new OkHttpClient()

def creds = ['user', 'password']
def auth = new PtcLogin(http).login(*creds)

PokemonGo go = new PokemonGo(auth, http)

def coords = [44.9748004, -93.2776901, 0]
//
https://www.google.com/maps/place/44%C2%B058'29.3%22N+93%C2%B016'39.7%22W/@44.9748042,
-93.2798788,17z/data=!3m1!4b1!4m5!3m4!1s0x0:0x0!8m2!3d44.9748004!4d-93.2776901
go.setLocation(*coords) ①

println "Player: $go.playerProfile.username, $go.playerProfile.team, $
go.playerProfile.stats"
println "Inventory:"
go.inventories.itemBag.items.each { item -> ②
  println "$item.itemId, $item.count, $item.unseen"
}

def catchablePokemon = go.map.catchablePokemon ③

println "Pokemon in area:"

catchablePokemon.collect { mon ->
  [mon.encounterPokemon(), mon] ④
}.findAll { encounter, mon ->
  encounter.wasSuccessful() ⑤
}.each { encounter, mon ->
  println mon.pokemonId ⑥
}

def nearby = go.map.nearbyPokemon ⑦

println "Nearby pokemon:"

nearby.each { n ->
  println "$n.pokemonId, $n.distanceInMeters, $n.encounterId"
}
```

① Set current location using spread operator

② Iterate over your inventory

③ Find all catchablePokemon

④ Invoke `encounterPokemon()` and pass tuple of the encounter result and pokemon

⑤ Filter for any pokemon successfully encountered

⑥ Print each successfully encountered pokemon's id

⑦ Find all nearby pokemon

There are a lot more endpoints available for exploration in the source. Check the examples for more ideas.

14.3. Databases

```
import groovy.sql.Sql

Sql sql = Sql.newInstance(
  url: 'jdbc:h2:mem:db',
  driver: 'org.h2.Driver',
  user: 'sa', password: '')

sql.execute('create table beer (name char(255))')

def beers = ['Lager', 'Pale Ale', 'Saison', 'Black IPA']
beers.each { b ->
  sql.execute('INSERT INTO `beer` (name) VALUES (?)', b) (1)
}

def rows = sql.rows('select * from beer') (2)
assert rows.size() == beers.size()
assert rows.name == beers (3)
```

① Insert each beer name into the db

② Simple select, returns `List<GroovyRowResult>`

③ Use simple dot notation to access `name` key in each result

14.4. Desktop Fun

java.awt.Robot for some desktop fun

```
import java.awt.Robot

Robot r = new Robot() ①

(300..500).collect { i ->
  [i, i] ②
}.each { args ->
  Thread.sleep 30 ③
  r.mouseMove(*args) ④
}
```

① Create new instance of Robot

② Create pairs of x, y coordinates from 300 to 500

③ Sleep for 30 milliseconds

④ Move the cursor by pair of x, y coordinates

14.5. Creating quick webservices

Ratpack

```
@Grab('io.ratpack:ratpack-groovy:1.4.0-rc-2')

import static ratpack.groovy.Groovy.ratpack

ratpack {
  handlers {
    get { ①
      render 'hello, world' ②
    }
    get('foo') { ③
      render 'foo' ④
    }
  }
}
```

① Define a root handler for GET requests

② Render the String `hello, world` to the client

③ Define a handler for `GET /foo`

④ Render the String `foo` to the client

Ratpack binds to port 5050 by default

Try issuing some commands

```
$ curl localhost:5050 ①
hello, world

$ curl localhost:5050/foo ②
foo
```

Learn more at https://ratpack.io

15. Resources

- Apache Groovy Website
- Groovy In Action
- Groovy Mailing-lists
- Twitter