Groovy Scripting for the JVM

CONCEPTS

A script needs no class or main method: top-level statements just run, and args holds the arguments.

Script .groovy source
↓ compiled
Bytecode in memory
↓ runs on
JVM needs a JDK
↓ @Grab
Libraries Maven repos

Groovy is mostly Java-compatible: semicolons, parentheses and return are optional, and public is the default.

RUNNING

groovy *<file>*.groovy [args]
Run a script

groovy -e '*<code>*'
Run an inline script

groovy -v
Show Groovy and JVM versions

groovy -cp *<path>* *<file>*
Set the classpath

groovy -Dkey=value *<file>*
Define a system property

groovy -d *<file>*
Print full stack traces

groovy --help
List all options

ONE-LINERS

groovy -n -e '*<code>*' *<file>*
Run code per line, as line

groovy -p -e '*<code>*' *<file>*
Run per line and print result

groovy -a -n -e '*<code>*' *<file>*
Split each line into split

groovy -i .bak -p -e '*<code>*' *<file>*
Edit files in place with backup

SHELL

groovysh
Start the interactive shell

/help
List shell commands

/show
List shared variables

/load *<file>*
Load a file into the buffer

/grab *<g:m:v>*
Add a dependency to the shell

/exit
Leave the shell

DEPENDENCIES

@Grab('*<g>*:*<m>*:*<v>*')
Pull in a library

@GrabResolver(name='*<n>*', root='*<url>*')
Add a repository

SYNTAX

def x = 1
Declare a variable

println "Hi ${name}"
Interpolate in a GString

**'''line1
line2'''**
Multi-line string

/a\d+/
Slashy string for regex

def f = { a, b → a + b }
Closure

list.collect { it * 2 }
Map over a collection

a?.b
Safe navigation

a ?: b
Elvis: default if falsy

list*.name
Spread: property of each item

a == b
Value equality

class Foo { String name }
Class with auto getters

@CompileStatic
Static type checking

COLLECTIONS

[1, 2, 3]
List literal

[a: 1, b: 2]
Map literal

[] as Set
Convert to another type

(1..5)
Inclusive range

list.each { println it }
Loop over items

list.findAll { it > 1 }
Filter

list.find { it > 1 }
First match

list.inject(0) { s, x → s + x }
Fold to one value

list.sum()
Add up elements

list.groupBy { it.size() }
Group into a map

list.sort { it.name }
Sort by key

map.each { k, v → println "$k=$v" }
Loop over a map

FILES & TEXT

new File('*<path>*').text
Read a whole file

new File('*<path>*').eachLine { }
Read line by line

new File('*<path>*').text = '*<data>*'
Write a file

'echo hi'.execute().text
Run a shell command

str =~ /*<regex>*/
Find a match

new groovy.json.JsonSlurper().parseText(s)
Parse JSON

groovy.json.JsonOutput.toJson(obj)
Write JSON