Groovy Scripting for the JVM

CONCEPTS

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

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

ONE-LINERS

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

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

SHELL

`groovysh`
Start the interactive shell

DEPENDENCIES

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

SYNTAX

```
def x = 1
```
Declare a variable

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

```
def f = { a, b → a + b }
```
Closure

SYNTAX (cont.)

```
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

```
class Foo { String name }
```
Class with auto getters

COLLECTIONS

```
[1, 2, 3]
```
List literal

```
[a: 1, b: 2]
```
Map literal

```
(1..5)
```
Inclusive range

```
list.each { println it }
```
Loop over items

```
list.findAll { it > 1 }
```
Filter

```
list.find { it > 1 }
```
First match

```
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

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