

# Table of Contents

Introduction..... 1

Using GeoScript..... 1

    geoscript-groovy..... 3

# Introduction

The GeoScript Groovy Cookbook contains short recipes on how to use the GeoScript Groovy library.

GeoScript is a geospatial library written in Groovy. It provides modules for working with geometries, projections, features, layers, rasters, styles, rendering, and tiles. It is built on top of the Java Topology Suite (JTS) and GeoTools libraries. GeoScript Groovy is open source and licensed under the MIT license.

## Using GeoScript

To use GeoScript Groovy you need Java and Groovy installed and on your PATH. Next, download the latest stable release, the latest in development build, or build the code yourself. Then put the GeoScript Groovy bin directory on your PATH. You are now ready to use GeoScript Groovy!

GeoScript Groovy has three commands:

1. geoscript-groovy (which can run Groovy files)

A terminal window titled "1. bash" on a Mac. The user runs "cat script.groovy" showing a Groovy script that creates a Point, buffers it by 2.5 units, and prints the WKT. Then, the user runs "geoscript-groovy script.groovy", which outputs the WKT for the buffered point. The terminal text is as follows:

```
Jared-Ericksons-MacBook-Pro:~ jericks$ cat script.groovy
import geoscript.geom.Point

p = new Point(-123.181458, 47.036439)
println p.wkt

poly = p.buffer(2.5)
println poly.wkt
Jared-Ericksons-MacBook-Pro:~ jericks$ geoscript-groovy script.groovy
POINT (-123.181458 47.036439)
POLYGON ((-120.681458 47.036439, -120.72949479899194 46.54871319495968, -120.871
7591687218 46.07973041908728, -121.10278396924365 45.647513417450995, -121.41369
104703364 45.26867204703363, -121.792532417451 44.95776496924364, -122.224749419
08729 44.72674016872178, -122.69373219495968 44.584475798991924, -123.181458 44.
536439, -123.66918380504033 44.584475798991924, -124.13816658091272 44.726740168
72178, -124.57038358254901 44.95776496924364, -124.94922495296638 45.26867204703
363, -125.26013203075637 45.647513417450995, -125.49115683127822 46.079730419087
28, -125.63342120100808 46.54871319495968, -125.681458 47.036439, -125.633421201
00808 47.52416480504032, -125.49115683127822 47.993147580912726, -125.2601320307
5637 48.42536458254901, -124.94922495296638 48.80420595296637, -124.570383582549
01 49.11511303075637, -124.13816658091272 49.34613783127822, -123.66918380504032
49.48840220100808, -123.181458 49.536439, -122.69373219495968 49.48840220100808
, -122.22474941908727 49.34613783127821, -121.792532417451 49.11511303075636, -1
21.41369104703364 48.80420595296636, -121.10278396924363 48.425364582549, -120.8
7175916872178 47.99314758091272, -120.72949479899192 47.524164805040314, -120.68
1458 47.036439))
Jared-Ericksons-MacBook-Pro:~ jericks$
```

1. geoscript-groovysh (which starts a REPL shell)

```
1. java
Jared-Ericksons-MacBook-Pro:~ jericks$ geoscript-groovysh
Groovy Shell (2.4.10, JVM: 1.8.0_31)
Type ':help' or ':h' for help.

-----
groovy:000> import geoscript.geom.Point
=> geoscript.geom.Point
groovy:000> p = new Point(-123.181458,47.036439)
=> POINT (-123.181458 47.036439)
groovy:000> p.buffer(2.5)
=> POLYGON ((-120.681458 47.036439, -120.72949479899194 46.54871319495968, -120.8717591687218 46.07973041908728, -121.10278396924365 45.647513417450995, -121.41369104703364 45.26867204703363, -121.792532417451 44.95776496924364, -122.22474941908729 44.72674016872178, -122.69373219495968 44.584475798991924, -123.181458 44.536439, -123.66918380504033 44.584475798991924, -124.13816658091272 44.72674016872178, -124.57038358254901 44.95776496924364, -124.94922495296638 45.26867204703363, -125.26013203075637 45.647513417450995, -125.49115683127822 46.07973041908728, -125.63342120100808 46.54871319495968, -125.681458 47.036439, -125.63342120100808 47.52416480504032, -125.49115683127822 47.993147580912726, -125.26013203075637 48.42536458254901, -124.94922495296638 48.80420595296637, -124.57038358254901 49.11511303075637, -124.13816658091272 49.34613783127822, -123.66918380504032 49.48840220100808, -123.181458 49.536439, -122.69373219495968 49.48840220100808, -122.22474941908727 49.34613783127821, -121.792532417451 49.11511303075636, -121.41369104703364 48.80420595296636, -121.10278396924363 48.425364582549, -120.87175916872178 47.99314758091272, -120.72949479899192 47.524164805040314, -120.681458 47.036439))
groovy:000> 
```

1. geoscript-groovyConsole (which starts a graphical editor/mini IDE)



## geoscript-groovy

The geoscript-groovy command can run scripts file but it can also run inline scripts.

*Convert a shapefile to geojson*

```
geoscript-groovy -e "println new  
geoscript.layer.Shapefile('states.shp').toJSONString()"
```

*Get the Bounds of a Shapefile as a Geometry*

```
geoscript-groovy -e "println new  
geoscript.layer.Shapefile('states.shp').bounds.geometry"
```

*Count the number of Features in a Shapefile*

```
geoscript-groovy -e "println new geoscript.layer.Shapefile('states.shp').count"
```

*Render a Shapefile to an image*

```
geoscript-groovy -e "geoscript.render.Draw.draw(new  
geoscript.layer.Shapefile('states.shp'), out: 'image.png')"
```

*Pipe a Shapefile's geometry to another command line application that buffers each feature*

```
geoscript-groovy -e "new geoscript.layer.Shapefile('states.shp').eachFeature{ println  
it.geom.centroid}" | geom combine | geom buffer -d 1.5
```

*Pipe the results of buffering a point to convert it to KML*

```
echo "POINT (1 1)" | geom buffer -d 10 | geoscript-groovy -e "println  
geoscript.geom.Geometry.fromWKT(System.in.text).kml"
```