**The Groovy Story**

James Strachan and his wife were waiting for a late plane. While she went shopping, he visited an internet cafe and spontaneously decided to go to the Python website and study the language. In the course of this activity, he became more and more intrigued. Being a seasoned Java programmer, he recognized that his home language lacked many of the interesting and useful features Python had invented, such as native language support for common datatypes in an expressive syntax and, more important, dynamic behavior. The idea was born to bring such features to Java.

**What is Groovy?**

* Groovy is an optionally typed, dynamic language for the Java platform with many features that are inspired by languages like Python, Ruby, and Smalltalk, making them available to Java developers using a Java-like syntax.
* Groovy is often referred to as a scripting language, and it works very well for scripting. It’s a mistake to label Groovy purely in those terms, though. It can be precompiled into Java byte code, integrated into Java applications, power web applications, add an extra degree of control within build files, and be the basis of whole applications on its own. Groovy, obviously, is too flexible
* This is true in terms of both implementation (many parts of Groovy are written in Java, with the rest being written in Groovy itself) and interaction. When you program in Groovy, in many ways you’re writing a special kind of Java. All the power of the Java platform—including the massive set of available libraries
* Groovy: it runs inside the JVM and makes use of Java’s libraries (together called the Java Runtime Environment, or JRE). Groovy is only a new way of creating *ordinary* Java classes from a runtime perspective, Groovy *is* Java with an additional JAR file as a dependency.
* Calling Java from Groovy is a nonissue. When developing in Groovy, you end up doing this all the time without noticing. Groovy type is a subtype of java.lang.Object. Every Groovy object is an instance of type in the normal way.
* Integration in the opposite direction is just as easy. Suppose a Groovy class MyGroovyClass is compiled into MyGroovyClass.class and put on the classpath. You can use this Groovy class from within a Java class by typing

**new MyGroovyClass(); // create from Java**

You can then call methods on the instance, pass the reference as an argument to methods, and so forth.