Apache Groovy

[Introduction, What is Groovy,Soruce Code,Groovy Website & Documentation, Groovy Books & Additional Resources, How to get most of it ]

**Introduction to Apache Groovy**

In this lesson, we'll take a few minutes to explain **Groovy** for those new to the language. Even if you're already familiar, you might still learn a few things.

**What is Apache Groovy?**

* **Apache Groovy** is a **powerful, optionally typed, and dynamic** language for the **Java platform**, aimed at improving developer productivity.
* It’s inspired by **Python, Ruby, and Smalltalk**, offering Java developers a **Java-like syntax**.

**Key Features of Groovy**

1. **Flat learning curve**
   * **Concise, readable code** makes Groovy easy to learn.
   * Java developers can **quickly pick up Groovy**.
   * Existing **Java source code** can be compiled with the **Groovy compiler**.
   * **Easier than learning Java**, thanks to its **expressiveness and simplicity**.
2. **Smooth Java integration**
   * Works well with **third-party Java libraries**.
   * Easily integrates into Java development environments.
3. **Vibrant ecosystem**
   * Supports **web development, reactive apps, build tools, testing tools, and GUI building**.
   * A growing community with diverse projects.

**Powerful Features We'll Explore**

* **Closures** for cleaner and more functional code.
* **Builders** to create object graphs, JSON, and markup with simplified syntax.
* **Metaprogramming** for dynamic features at **compile-time and runtime**.
* **Functional programming** and **type inference**.
* **Static compilation** for performance improvements.

**Domain-Specific Languages (DSLs)**

* DSLs are **expressive, targeted solutions** for specific problems.
* Groovy’s syntax makes writing **custom DSLs** simple and effective.

**Scripting & Testing Capabilities**

* Great for writing **concise and maintainable tests**.
* Perfect for **build automation tasks**.
* Many Java developers integrate Groovy for its **powerful scripting and testing features**.

**Groovy’s Backing & Popularity**

* Supported by the **Apache Software Foundation**, ensuring **reliability and stability**.
* Listed as an **official project** on Apache.org.
* Used by **companies worldwide**, including mine, where **20+ developers** actively enjoy Groovy.
* Over **10 million downloads last year**—a testament to its growth.
* Ranked **#17 on the Top Programming Community Index**, showing its increasing popularity.

**Source Code**

[**https://github.com/danvega/apache-groovy-course**](https://github.com/danvega/apache-groovy-course)

**Exploring the Groovy Website & Documentation**

Now that we know a little about **Groovy**, let’s take a look at its **official website and documentation**. This will help us **navigate resources**, so we know where to **find answers** when needed.

**Groovy’s Main Website**

**Website:** groovy-lang.org

* Provides general **information about Groovy**.
* Lists **upcoming events** and **companies that use Groovy**.
* Contains a **Learn section** with books and presentations.
* Includes **community resources**, such as:
  + **Mailing lists**
  + **User groups**
  + **Groovy newsletter**
  + **Social media links**

**Open-Source Contributions**

🔹 Groovy is **open-source**, meaning anyone can **view, contribute, or improve** the code. 🔹 You can even find **source code** for the Groovy website if you want to see how it was built.

**Documentation Overview**

Groovy provides **detailed documentation** to help developers learn and troubleshoot issues.

🔹 **Single-page documentation:** Contains **everything in one place**. 🔹 **Version-specific documentation:** Helps check details for **older versions** of Groovy. 🔹 **API documentation:**

* **GroovyDoc API** → Reference for Groovy’s built-in features.
* **Groovy JDK API** → Shows additional **Groovy methods added to Java**.

**How to Use Documentation Effectively**

✔ You can **navigate by topic** instead of scrolling through long pages. ✔ **Bookmark key pages** to quickly access **API references**. ✔ **Search by package/class** to find methods **for specific tasks** (e.g., Groovy’s SQL class for database operations).

**Groovy Books & Additional Resources**

If you’re eager to **deepen your understanding of Groovy**, using **multiple resources** is key. This course will guide you, but there are **books, YouTube tutorials, articles, newsletters, podcasts**, and more to supplement your learning.

At the end of this course, I’ve compiled a list of **additional resources**—so be sure to check them out!

**Recommended Books**

**1️⃣ Groovy in Action (Second Edition)**

**Authors:** Multiple, including **Guillaume Laforgue** (Groovy Project Lead). ✔ A **comprehensive guide** covering all aspects of Groovy. ✔ Used as inspiration for structuring this course. ✔ A must-have for anyone **serious about mastering Groovy**.

**2️⃣ Making Java Groovy – Ken Cousin**

**Author:** Ken Cousin (Highly respected Groovy instructor). ✔ Perfect for **Java developers transitioning into Groovy**. ✔ Explains **differences and benefits** of Groovy over Java. ✔ Small and concise, but **packed with valuable insights**.

**3️⃣ Programming Groovy 2 – Venkat Subramaniam**

**Author:** Venkat Subramaniam (Well-known in the Groovy community). ✔ Covers **core Groovy concepts**, including **metaprogramming and closures**. ✔ Written by an **engaging speaker**—if you ever have a chance, attend his talks!

**Getting Started: Setting Up Your Development Environment**

Welcome to **Section Two**—this section focuses on **getting your development environment up and running**.

**Why This is Important**

✔ Ensuring you have the **right tools** for this course. ✔ Understanding how to **use these tools effectively**. ✔ Preparing your setup so everything runs **smoothly**.

**Before You Begin**

**Check the documentation** to explore Groovy’s download options.

Visit the **Download Section** for installation methods. Familiarize yourself with key tools such as:

* **Groovy Compiler**
* **Groovy Command-Line Shell**
* **Console & IDE Integration**

**Installing Java 8 on Windows**

In this tutorial, we’ll walk through **installing the Java JDK 8** on Windows. Many people confuse the **JRE (Java Runtime Environment)** and **JDK (Java Development Kit)**—let’s clarify the difference:

✔ **JRE** – Needed **only to run Java programs**. ✔ **JDK** – Includes compilers and tools **to create and compile Java programs**.

Since we want to **create Java programs**, we’ll be installing the **JDK** today.

**Step 1: Check If Java is Installed**

1️⃣ Open **Command Prompt**:

* Press Windows + R, type **cmd**, and hit **Enter**. 2️⃣ Type the command:
* java -version 3️⃣ If Java isn’t installed, you’ll see an error message.

**Step 2: Download & Install Java JDK 8**

1️⃣ Open a **browser** and search for "Java JDK download". 2️⃣ Click on **Java SE Downloads** and select **JDK 8**. 3️⃣ Accept the **license agreement**, then download the version matching your **system architecture** (32-bit or 64-bit). 4️⃣ Once downloaded, **double-click** the installer and follow these steps:

* Click **Next** through the installation prompts.

✔ **Java is now installed!**

**Step 3: Set Up Windows Environment Variables**

1️⃣ Open **System Properties**:

* Press Windows + Pause/Break OR **Right-click** "My Computer" → **Properties**. 2️⃣ Click **Advanced System Settings** → **Environment Variables**. 3️⃣ Locate where Java is installed (**e.g.,** C:\Program Files\Java\JDK1.8.0\_60). 4️⃣ Create a new **environment variable**:
* Name: JAVA\_HOME
* Value: **Paste your Java installation path**. 5️⃣.Edit the **Path variable**:
* Append %JAVA\_HOME%\bin at the end. 6️⃣ Click **OK** to save changes.

**Step 4: Verify Installation**

1️⃣ Open **Command Prompt** again (Windows + R, type **cmd**, hit Enter). 2️⃣ Run the command:

* java -version 3️⃣ If installed correctly, you’ll see **Java version 1.8.0\_60** displayed.

✔ **Java 8 is successfully installed on Windows!** 🎉

**Final Thoughts**

Now that Java is installed, you’re ready to **compile and run Java programs**! Let me know if you need help with any **additional configurations**.

**Installing Groovy on Windows**

In this tutorial, we’ll walk through **installing Groovy** on Windows. Before we begin, let’s check if **Groovy is already installed**.

**Step 1: Check If Groovy is Installed**

1️⃣ Open **Command Prompt** (Windows + R, type **cmd**, hit **Enter**). 2️⃣ Run the command:

* groovy -v 3️⃣ If Groovy **is not installed**, you’ll see an error message.

**Step 2: Install Groovy (Two Methods)**

**Method 1: Using Binary Distribution (Manual Install)**

1️⃣ Visit groovy-lang.org and go to **Download Section**. 2️⃣ Click on **Binary Distribution** to download the ZIP file. 3️⃣ Extract the ZIP file to a **folder on your C drive** (e.g., C:\Groovy). 4️⃣ Inside the extracted folder, you’ll see **Groovy version (e.g.,** Groovy 2.4.5**)**.

✔ **Groovy is installed**, but we need to set up environment variables.

**Step 3: Configure Environment Variables**

1️⃣ Open **System Properties** (Windows + Pause/Break OR **Right-click** "My Computer" → **Properties**). 2️⃣ Click **Advanced System Settings** → **Environment Variables**. 3️⃣ Create a new **variable**:

* Name: GROOVY\_HOME
* Value: **Path to Groovy folder (e.g.,** C:\Groovy\Groovy2.4.5**)** 4️⃣ Edit the **Path variable**:
* Append %GROOVY\_HOME%\bin at the end. 5️⃣ Click **OK** to save changes.

**Step 4: Verify Installation**

1️⃣ Open **Command Prompt** and run:

* groovy -v 2️⃣ If installed correctly, you’ll see **Groovy version, JVM version, and OS details**.

✔ **Groovy is successfully installed using the manual method!** 🎉

**Method 2: Using Installer (Automated Install)**

1️⃣ Visit **Groovy’s Download Page** and find the **Groovy Installer (**groovy-2.4.5-installer.exe**)**. 2️⃣ Download and **double-click the installer**. 3️⃣ Follow the setup prompts:

* **Select language** → Click **OK**.
* **Accept license agreement** → Click **Next**.
* **Choose installation path** → Click **Install**.
* **Add Groovy to Path automatically** → Click **Next**. 4️⃣ Finish the installation.

🔹 **The installer automatically sets up environment variables**, so no manual setup is needed!

✔ **Groovy is successfully installed using the installer!**

**Step 5: Verify Installation (Again)**

1️⃣ Open **Command Prompt** (Windows + R, type **cmd**, hit \*\*Enter`). 2️⃣ Run:

* groovy -v 3️⃣ You should see the **Groovy version, JVM version, and OS details**.

**Installing Groovy on Windows**

In this tutorial, we’ll walk through **installing Groovy** on Windows. Before we begin, let’s check if **Groovy is already installed**.

**Step 1: Check If Groovy is Installed**

1️⃣ Open **Command Prompt** (Windows + R, type **cmd**, hit **Enter**). 2️⃣ Run the command:

* groovy -v 3️⃣ If Groovy **is not installed**, you’ll see an error message.

**Step 2: Install Groovy (Two Methods)**

**Method 1: Using Binary Distribution (Manual Install)**

1️⃣ Visit groovy-lang.org and go to **Download Section**. 2️⃣ Click on **Binary Distribution** to download the ZIP file. 3️⃣ Extract the ZIP file to a **folder on your C drive** (e.g., C:\Groovy). 4️⃣ Inside the extracted folder, you’ll see **Groovy version (e.g.,** Groovy 2.4.5**)**.

✔ **Groovy is installed**, but we need to set up environment variables.

**Step 3: Configure Environment Variables**

1️⃣ Open **System Properties** (Windows + Pause/Break OR **Right-click** "My Computer" → **Properties**). 2️⃣ Click **Advanced System Settings** → **Environment Variables**. 3️⃣ Create a new **variable**:

* Name: GROOVY\_HOME
* Value: **Path to Groovy folder (e.g.,** C:\Groovy\Groovy2.4.5**)** 4️⃣ Edit the **Path variable**:
* Append %GROOVY\_HOME%\bin at the end. 5️⃣ Click **OK** to save changes.

**Step 4: Verify Installation**

1️⃣ Open **Command Prompt** and run:

* groovy -v 2️⃣ If installed correctly, you’ll see **Groovy version, JVM version, and OS details**.

✔ **Groovy is successfully installed using the manual method!** 🎉

**Method 2: Using Installer (Automated Install)**

1️⃣ Visit **Groovy’s Download Page** and find the **Groovy Installer (**groovy-2.4.5-installer.exe**)**. 2️⃣ Download and **double-click the installer**. 3️⃣ Follow the setup prompts:

* **Select language** → Click **OK**.
* **Accept license agreement** → Click **Next**.
* **Choose installation path** → Click **Install**.
* **Add Groovy to Path automatically** → Click **Next**. 4️⃣ Finish the installation.

🔹 **The installer automatically sets up environment variables**, so no manual setup is needed!

✔ **Groovy is successfully installed using the installer!** 🎉

**Step 5: Verify Installation (Again)**

1️⃣ Open **Command Prompt** (Windows + R, type **cmd**, hit \*\*Enter`). 2️⃣ Run:

* groovy -v 3️⃣ You should see the **Groovy version, JVM version, and OS details**.

**Final Thoughts**

✔ We’ve explored **two methods** of installing Groovy. ✔ Using the **binary distribution** allows **manual setup**, while the **installer automates everything**. ✔ Now, you’re ready to **start using Groovy on Windows!** 🚀

Let me know if you need any further clarifications!

This keeps everything **structured, clear, and concise** while ensuring completeness. Let me know if you’d like further refinements!

**Managing Development Tools with SDKMAN**

Today, we're exploring **SDKMAN**, a powerful tool for managing multiple versions of **software development kits (SDKs)** on **Unix-based systems**.

**What is SDKMAN?**

✔ **SDK Manager** for installing, switching, and managing multiple SDK versions. ✔ Previously called **Groovy Environment Manager (GVM)**. ✔ Inspired by **RVM & RBEnv** from the Ruby ecosystem. ✔ Works with languages & tools like **Groovy, Scala, Gradle, Grails, Spring Boot, and Maven**.

**Step 1: Install SDKMAN**

1️⃣ Open **Terminal** and run:

sh

curl -s "https://get.sdkman.io" | bash

2️⃣ Follow the **installation prompts**. 3️⃣ Restart your terminal and run:

sh

source "$HOME/.sdkman/bin/sdkman-init.sh"

4️⃣ Check the installed version:

sh

sdk version

✔ **SDKMAN is now installed!**

**Step 2: Using SDKMAN**

**Check Available Commands**

Run sdk help to view supported commands like:

* sdk list → Lists all available SDKs.
* sdk install <sdk-name> → Installs an SDK.
* sdk uninstall <sdk-name> <version> → Removes an SDK version.
* sdk use <sdk-name> <version> → Temporarily switch versions.
* sdk default <sdk-name> <version> → Set a permanent default version.

**List Available SDKs**

Run:

sh

sdk list

You’ll see options like **Groovy, Gradle, Maven, Spring Boot**, etc.

**Install & Switch SDK Versions**

1️⃣ Install **Groovy version 2.4.0**:

sh

sdk install groovy 2.4.0

2️⃣ Set **2.4.0 as default**:

sh

sdk default groovy 2.4.0

3️⃣ Switch back to **2.4.5**:

sh

sdk use groovy 2.4.5

**Why SDKMAN is Useful**

✔ **Easily manage multiple SDK versions** without manual installations.

✔ **Instantly switch** between tool versions for different projects.

✔ **Centralized control** for managing

**Getting Started with the Groovy Shell**

Now that we have **Groovy installed**, let’s take it for a **test drive** using the **Groovy Shell (aka GroovySH)**.

The Groovy Shell is a **command-line tool** that allows us to: ✔ Evaluate **Groovy expressions**. ✔ Define **classes and functions**. ✔ Run **quick experiments**—perfect for testing small snippets!

**Step 1: Launching GroovySH**

1️⃣ Open **Terminal** or **Command Prompt**. 2️⃣ Run:

* On Unix/macOS: groovysh
* On Windows: groovy.bat 3️⃣ You’ll see a **Groovy Shell prompt** (groovy:000>)—ready for input!

**Check Groovy Version** Run:

sh

groovysh --version

✔ Displays the installed Groovy version.

**Step 2: Testing Basic Expressions**

Try simple expressions directly in GroovySH:

groovy

1 + 1 // Outputs: 2

println "Hello, World!" // Outputs: Hello, World!

✔ The shell evaluates expressions **immediately**.

**Step 3: Multi-Line Expressions & Defining Classes**

Groovy allows **multi-line complex expressions**, like defining classes:

groovy

class Person {

def sayHello() {

println "Hello!"

}

}

✔ The shell waits until the **expression is complete** before compiling.

Now, create an instance and call the method:

groovy

def person = new Person()

person.sayHello() // Outputs: Hello!

✔ GroovySH allows quick **testing of class definitions**!

**Step 4: Defining & Calling Functions**

Define functions interactively:

groovy

def greet(name) {

println "Hello, $name!"

}

Call the function:

groovy

greet("Dan") // Outputs: Hello, Dan!

✔ GroovySH makes it easy to **define reusable functions** on the fly.

**Step 5: Useful GroovySH Commands**

GroovySH supports **rich interactive commands**:

* h → Displays **help menu**.
* clear → **Resets** the shell buffer.
* import <package> → **Imports Java/Groovy classes**.
* exit → **Quits GroovySH**.

Try listing all available commands:

sh

h

✔ Explore useful built-in functions!

**Final Thoughts**

✔ **GroovySH is an interactive playground** for experimenting with Groovy.

✔ Use it to **test code snippets** without writing full scripts.

✔ Now, you're ready to **start building with Groovy!**

**Compiling & Running Groovy Code**

In earlier lessons, we ran Groovy code **directly** without creating executable files. Now, let’s explore **Groovy compilation and execution** using **groovyc** (Groovy Compiler) and **groovy** (Groovy Executor).

✔ **Don't worry**—manual compilation isn't needed for every project! ✔ This section helps **understand what happens behind the scenes** when using an IDE.

**Step 1: Understanding groovyc (Groovy Compiler)**

* groovyc compiles Groovy source files into **Java bytecode**.
* Equivalent to javac in Java.
* Generates .class files, which **run on the JVM like Java programs**.

**Step 2: Compiling a Single File**

1️⃣ Open **Terminal/Command Prompt**. 2️⃣ Navigate to the folder containing your Groovy file (Person.groovy). 3️⃣ Run:

sh

groovyc Person.groovy

4️⃣ Verify the output:

sh

ls # (macOS/Linux) or dir # (Windows)

✔ You’ll see Person.class, the compiled Java bytecode file.

**Step 3: Compiling Multiple Files**

1️⃣ Compile all Groovy files in a directory:

sh

groovyc \*.groovy

✔ Creates .class files for **every Groovy file** in the directory.

2️⃣ Specify an **output directory** for compiled classes:

sh

groovyc -d classes \*.groovy

✔ Moves compiled .class files into a **new** classes **directory**.

**Step 4: Running Groovy Code**

**Use** groovy **to execute Groovy scripts without compilation**. 1️⃣ Run a Groovy file directly:

sh

groovy Application.groovy

✔ Works **without manually compiling**—Groovy **automatically interprets** and runs the code.

**Step 5: Running Compiled Groovy Code from Java**

Since Groovy compiles to **Java bytecode**, we can run Groovy-generated .class files in Java!

1️⃣ Ensure groovy-all.jar is included in the **classpath**. 2️⃣ Run compiled Groovy code using Java:

sh

java -cp /path/to/groovy-all.jar:. Application

✔ Java can execute **compiled Groovy** .class **files**, proving Groovy's **tight integration with Java**.

**Final Thoughts**

✔ Groovy code compiles **directly into Java bytecode**, making it **fully compatible** with the JVM. ✔ You can **run Groovy scripts dynamically** or **compile them for efficiency**.

✔ This approach works **just like Java**, making Groovy a **powerful addition** to Java projects!