Groovy Programming Language Cheatsheet

A concise cheat sheet covering essential aspects of the Groovy programming language, including syntax, data types, collections, closures, and common operations. This cheat sheet is designed to provide a quick reference for developers working with Groovy, offering a practical overview of the language's core features and capabilities.

Groovy Basics and Syntax

Basic Syntax

Comments	`//` Single-line comment `/* ... */` Multi-line comment `/** ... */` Groovydoc comment
Statements	Statements do not necessarily need to end with a semicolon (;)
Variables	Declared with `def`, `String`, `int`, etc. Groovy is dynamically typed, but static typing is also supported.
Strings	Single quotes (`'...'`) for simple strings. Double quotes (`"..."`) for strings with variable interpolation. Triple quotes (`'''...'''` or `"""..."""`) for multi-line strings.
Example	`def name = "Groovy"` `println "Hello, ${name}!" // Hello, Groovy!`

Data Types

Primitive Types	`int`, `long`, `float`, `double`, `boolean`, `char`
Objects	`Integer`, `Long`, `Float`, `Double`, `Boolean`, `Character`, `String`
Collections	`List`, `Set`, `Map`
Ranges	`1..10`, `'a'..'z'`

Operators

Arithmetic	`+`, `-`, `*`, `/`, `%`, `**` (power)
Assignment	`=`, `+=`, `-=`, `*=`, `/=`, `%=`
Comparison	`==`, `!=`, `<`, `>`, `<=`, `>=`
Logical	`&&`, `||`, `!`
Safe Navigation	`?.` (avoids NullPointerException)

Collections and Closures

Lists

Declaration	`def list = [1, 2, 3]`
Accessing Elements	`println list[0] // 1`
Adding Elements	`list << 4` `list.add(5)`
Iterating	`list.each { println it }`

Maps

Declaration	`def map = ['key1': 'value1', 'key2': 'value2']`
Accessing Values	`println map.key1 // value1` `println map['key2'] // value2`
Adding/Updating Values	`map.key3 = 'value3'` `map['key4'] = 'value4'`
Iterating	`map.each { key, value -> println "${key}: ${value}" }`

Closures

Definition	`def closure = { param -> println param }`
Calling	`closure('Hello') // Hello`
Implicit Parameter	`it` (when the closure has only one parameter)
Example	`def numbers = [1, 2, 3]` `numbers.each { println it * 2 } // 2, 4, 6`

Object-Oriented Programming

Classes

Definition	`class Person {`
	`String name`
	`int age`
	
	`Person(String name, int age) {`
	`this.name = name`
	`this.age = age`
	`}`
	`}`

Creating Instances	`def person = new Person('Alice', 30)` `println person.name // Alice`
Getters and Setters	Automatically generated for class properties.

Methods

Definition	`class Calculator {`
	`def add(int a, int b) {`
	`return a + b`
	`}`
	`}`

Calling	`def calc = new Calculator()` `println calc.add(5, 3) // 8`
Optional Parentheses	Parentheses can often be omitted. `println calc.add 5, 3 // 8`

Traits (Interfaces with Implementation)

Definition	`trait Loggable {`
	`def log(String message) {`
	`println "Logging: ${message}"`
	`}`
	`}`

Usage	`class MyClass implements Loggable {`
	`def doSomething() {`
	`log('Doing something...')`
	`}`
	`}`
	
	`def obj = new MyClass()`
	`obj.doSomething() // Logging: Doing something...`

Working with Files and I/O

File Operations

Creating a File	`def file = new File('example.txt')` `file.createNewFile()`
Writing to a File	`file.write('Hello, Groovy!')`
Reading from a File	`def content = file.text` `println content // Hello, Groovy!`
Appending to a File	`file << ' Appending text'` `println file.text // Hello, Groovy! Appending text`

Working with Streams

Reading from InputStream	`def inputStream = new FileInputStream('example.txt')`
	`inputStream.eachLine { println it }`
	`inputStream.close()`

Writing to OutputStream	`def outputStream = new FileOutputStream('output.txt')`
	`outputStream.write('Groovy Output'.getBytes())`
	`outputStream.close()`

Working with URLs

Reading Content from URL	`def url = new URL('http://example.com')`
	`def content = url.getText()`
	`println content`