



Groovy Basics and Syntax

Basic Syntax

Comments	<code>// Single-line comment</code> <code>/* ... */ Multi-line comment</code> <code>/** ... */ Groovydoc comment</code>
Statements	Statements do not necessarily need to end with a semicolon (;)
Variables	Declared with <code>def</code> , <code>String</code> , <code>int</code> , 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	<pre>def name = "Groovy" println "Hello, \${name}!" // Hello, Groovy!</pre>

Data Types

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

Operators

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

Collections and Closures

Lists

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

Maps

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

Closures

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

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'.getBytes())  
outputStream.close()
```

Working with URLs

Reading Content from URL

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