Docs

[User Manual](http://docs.google.com/userguide/userguide.html)

[Guides and Tutorials](https://guides.gradle.org)

[DSL Reference](http://docs.google.com/dsl/)

[Javadoc](http://docs.google.com/javadoc/)

[Release Notes](http://docs.google.com/release-notes.html)

[Forums](https://discuss.gradle.org/)

[Training](https://gradle.org/training/)

[Try Gradle Enterprise](https://gradle.com/enterprise)

[PDF](http://docs.google.com/userguide/userguide.pdf)

* [User Manual Home](http://docs.google.com/userguide/userguide.html)
* [Release Notes](http://docs.google.com/release-notes.html)
* [Installing Gradle](http://docs.google.com/userguide/installation.html)
* [Tutorials](https://guides.gradle.org/)

### Reference

* [Groovy DSL Reference](http://docs.google.com/dsl/)
* [Gradle API Javadoc](http://docs.google.com/javadoc/)
* [Core Plugins](http://docs.google.com/userguide/plugin_reference.html)
* [Gradle & Third-party Tools](http://docs.google.com/userguide/third_party_integration.html)

### Getting Started

* [Creating New Gradle Builds](https://guides.gradle.org/creating-new-gradle-builds/)
* [Creating Build Scans](https://guides.gradle.org/creating-build-scans/)
* [Migrating From Maven](https://guides.gradle.org/migrating-from-maven/)

### Running Gradle Builds

* [Command-Line Interface](http://docs.google.com/userguide/command_line_interface.html)
* [Customizing Execution](#gjdgxs)
  + [Configuring the Build Environment](http://docs.google.com/userguide/build_environment.html)
  + [Configuring the Gradle Daemon](http://docs.google.com/userguide/gradle_daemon.html)
  + [Initialization Scripts](http://docs.google.com/userguide/init_scripts.html)
* [Directory Layout](http://docs.google.com/userguide/directory_layout.html)
* [Executing Multi-Project Builds](http://docs.google.com/userguide/intro_multi_project_builds.html)
* [Gradle Wrapper](http://docs.google.com/userguide/gradle_wrapper.html)
* [Troubleshooting](http://docs.google.com/userguide/troubleshooting.html)
* [Using Build Scans](https://docs.gradle.com/build-scan-plugin)
* [Enabling and Configuring the Build Cache](http://docs.google.com/userguide/build_cache.html)
* [Integrating Separate Gradle Builds (Composite Builds)](http://docs.google.com/userguide/composite_builds.html)

### Authoring Gradle Builds

* [Fundamentals](#30j0zll)
  + [Introducing the Basics of Build Scripts](http://docs.google.com/userguide/tutorial_using_tasks.html)
  + [Working with Tasks](http://docs.google.com/userguide/more_about_tasks.html)
  + [Learning More About Build Scripts](http://docs.google.com/userguide/writing_build_scripts.html)
  + [Working with Files](http://docs.google.com/userguide/working_with_files.html)
  + [Creating Custom Task Types](http://docs.google.com/userguide/custom_tasks.html)
  + [Using Gradle Plugins](http://docs.google.com/userguide/plugins.html)
  + [The Standard Gradle Plugins](http://docs.google.com/userguide/standard_plugins.html)
  + [Understanding the Build Lifecycle](http://docs.google.com/userguide/build_lifecycle.html)
  + [Working with Logging](http://docs.google.com/userguide/logging.html)
  + [Configuring Multi-Project Builds](http://docs.google.com/userguide/multi_project_builds.html)
* [Best Practices](#1fob9te)
  + [Authoring Maintainable Build Scripts](http://docs.google.com/userguide/authoring_maintainable_build_scripts.html)
  + [Organizing Gradle Projects](http://docs.google.com/userguide/organizing_gradle_projects.html)
  + [Optimizing Build Performance](https://guides.gradle.org/performance/)
  + [Using the Build Cache](https://guides.gradle.org/using-build-cache/)
* [Dependency Management](#3znysh7)
  + [Introduction to Dependency Management](http://docs.google.com/userguide/introduction_dependency_management.html)
  + [Dependency Management Terminology](http://docs.google.com/userguide/dependency_management_terminology.html)
  + [Dependency Types](http://docs.google.com/userguide/dependency_types.html)
  + [Repository Types](http://docs.google.com/userguide/repository_types.html)
  + [Declaring Dependencies](http://docs.google.com/userguide/declaring_dependencies.html)
  + [Declaring Repositories](http://docs.google.com/userguide/declaring_repositories.html)
  + [Inspecting Dependencies](http://docs.google.com/userguide/inspecting_dependencies.html)
  + [Managing Dependency Configurations](http://docs.google.com/userguide/managing_dependency_configurations.html)
  + [Managing Transitive Dependencies](http://docs.google.com/userguide/managing_transitive_dependencies.html)
  + [Dependency Locking](http://docs.google.com/userguide/dependency_locking.html)
  + [Troubleshooting Dependency Resolution](http://docs.google.com/userguide/troubleshooting_dependency_resolution.html)
  + [Customizing Dependency Resolution Behavior](http://docs.google.com/userguide/customizing_dependency_resolution_behavior.html)
  + [Dependency Cache Internals](http://docs.google.com/userguide/dependency_cache.html)
  + [Working with Dependencies](http://docs.google.com/userguide/working_with_dependencies.html)
* [Publishing Artifacts](http://docs.google.com/userguide/artifact_management.html)
* [C++ Projects](#2et92p0)
  + [Building Native Software](http://docs.google.com/userguide/native_software.html)
  + [Software Model Concepts](http://docs.google.com/userguide/software_model_concepts.html)
  + [Rule-based Model Configuration](http://docs.google.com/userguide/software_model.html)
  + [Implementing Model Rules in a Plugin](http://docs.google.com/userguide/rule_source.html)
  + [Extending the Software Model](http://docs.google.com/userguide/software_model_extend.html)
* [Java Projects](#tyjcwt)
  + [Building Java & JVM projects](http://docs.google.com/userguide/building_java_projects.html)
  + [Testing Java & JVM projects](http://docs.google.com/userguide/java_testing.html)
* [Advanced Techniques](#3dy6vkm)
  + [Configuring Tasks Lazily](http://docs.google.com/userguide/lazy_configuration.html)
  + [Developing Parallel Tasks](https://guides.gradle.org/using-the-worker-api/)
  + [Testing Your Build with TestKit](http://docs.google.com/userguide/test_kit.html)
  + [Using Ant from Gradle](http://docs.google.com/userguide/ant.html)
* [Sample Gradle builds](#1t3h5sf)
  + [Groovy DSL Samples](https://github.com/gradle/gradle/tree/master/subprojects/docs/src/samples)
  + [Kotlin DSL Samples](https://github.com/gradle/kotlin-dsl/tree/master/samples)

### Extending Gradle

* [Writing Custom Plugins](http://docs.google.com/userguide/custom_plugins.html)
* [Plugin Development Guides](https://gradle.org/guides/?q=Plugin+Development)

[Edit this page](https://github.com/gradle/gradle/edit/master/subprojects/docs/src/docs/userguide/)

# Writing Build Scripts

Contents

[The Gradle build language](#4d34og8)

[The Project API](#2s8eyo1)

[The Script API](#17dp8vu)

[Declaring variables](#3rdcrjn)

[Configuring arbitrary objects](#26in1rg)

[Configuring arbitrary objects using an external script](#lnxbz9)

[Some Groovy basics](#35nkun2)

[Default imports](#1ksv4uv)

This chapter looks at some of the details of writing a build script.

[The Gradle build language](#4d34og8)

Gradle provides a *domain specific language*, or DSL, for describing builds. This build language is based on Groovy, with some additions to make it easier to describe a build.

A build script can contain any Groovy language element.[[1](#44sinio)] Gradle assumes that each build script is encoded using UTF-8.

[The Project API](#2s8eyo1)

In [the tutorial](http://docs.google.com/tutorial_java_projects.html#tutorial_java_projects) we used, for example, the apply() method. Where does this method come from? We said earlier that the build script defines a project in Gradle. For each project in the build, Gradle creates an object of type [Project](http://docs.google.com/dsl/org.gradle.api.Project.html) and associates this Project object with the build script. As the build script executes, it configures this Project object:

| **💡** | Getting help writing build scripts  Don’t forget that your build script is simply Groovy code that drives the Gradle API. And the [Project](http://docs.google.com/dsl/org.gradle.api.Project.html) interface is your starting point for accessing everything in the Gradle API. So, if you’re wondering what 'tags' are available in your build script, you can start with the documentation for the Project interface. |
| --- | --- |

* Any method you call in your build script which *is not defined* in the build script, is delegated to the Project object.
* Any property you access in your build script, which *is not defined* in the build script, is delegated to the Project object.

Let’s try this out and try to access the name property of the Project object.

[Example: Accessing property of the Project object](#2jxsxqh)

**build.gradle**

println name  
println project.name

**Output of** gradle -q check

> gradle -q check  
projectApi  
projectApi

Both println statements print out the same property. The first uses auto-delegation to the Project object, for properties not defined in the build script. The other statement uses the project property available to any build script, which returns the associated Project object. Only if you define a property or a method which has the same name as a member of the Project object, would you need to use the project property.

[Standard project properties](#z337ya)

The Project object provides some standard properties, which are available in your build script. The following table lists a few of the commonly used ones.

Table 1. Project Properties

| **Name** | **Type** | **Default Value** |
| --- | --- | --- |
| project | [Project](http://docs.google.com/dsl/org.gradle.api.Project.html) | The Project instance |
| name | String | The name of the project directory. |
| path | String | The absolute path of the project. |
| description | String | A description for the project. |
| projectDir | File | The directory containing the build script. |
| buildDir | File | *projectDir*/build |
| group | Object | unspecified |
| version | Object | unspecified |
| ant | [AntBuilder](http://docs.google.com/javadoc/org/gradle/api/AntBuilder.html) | An AntBuilder instance |

[The Script API](#17dp8vu)

When Gradle executes a script, it compiles the script into a class which implements [Script](http://docs.google.com/dsl/org.gradle.api.Script.html). This means that all of the properties and methods declared by the Script interface are available in your script.

[Declaring variables](#3rdcrjn)

There are two kinds of variables that can be declared in a build script: local variables and extra properties.

[Local variables](#3j2qqm3)

Local variables are declared with the def keyword. They are only visible in the scope where they have been declared. Local variables are a feature of the underlying Groovy language.

[Example: Using local variables](#1y810tw)

**build.gradle**

def dest = "dest"  
  
task copy(type: Copy) {  
 from "source"  
 into dest  
}

[Extra properties](#4i7ojhp)

All enhanced objects in Gradle’s domain model can hold extra user-defined properties. This includes, but is not limited to, projects, tasks, and source sets. Extra properties can be added, read and set via the owning object’s ext property. Alternatively, an ext block can be used to add multiple properties at once.

[Example: Using extra properties](#2xcytpi)

**build.gradle**

apply plugin: "java"  
  
ext {  
 springVersion = "3.1.0.RELEASE"  
 emailNotification = "build@master.org"  
}  
  
sourceSets.all { ext.purpose = null }  
  
sourceSets {  
 main {  
 purpose = "production"  
 }  
 test {  
 purpose = "test"  
 }  
 plugin {  
 purpose = "production"  
 }  
}  
  
task printProperties {  
 doLast {  
 println springVersion  
 println emailNotification  
 sourceSets.matching { it.purpose == "production" }.each { println it.name }  
 }  
}

**Output of** gradle -q printProperties

> gradle -q printProperties  
3.1.0.RELEASE  
build@master.org  
main  
plugin

In this example, an ext block adds two extra properties to the project object. Additionally, a property named purpose is added to each source set by setting ext.purpose to null (null is a permissible value). Once the properties have been added, they can be read and set like predefined properties.

By requiring special syntax for adding a property, Gradle can fail fast when an attempt is made to set a (predefined or extra) property but the property is misspelled or does not exist. Extra properties can be accessed from anywhere their owning object can be accessed, giving them a wider scope than local variables. Extra properties on a project are visible from its subprojects.

For further details on extra properties and their API, see the [ExtraPropertiesExtension](http://docs.google.com/dsl/org.gradle.api.plugins.ExtraPropertiesExtension.html) class in the API documentation.

[Configuring arbitrary objects](#26in1rg)

You can configure arbitrary objects in the following very readable way.

[Example: Configuring arbitrary objects](#1ci93xb)

**build.gradle**

task configure {  
 doLast {  
 def pos = configure(new java.text.FieldPosition(10)) {  
 beginIndex = 1  
 endIndex = 5  
 }  
 println pos.beginIndex  
 println pos.endIndex  
 }  
}

**Output of** gradle -q configure

> gradle -q configure  
1  
5

[Configuring arbitrary objects using an external script](#lnxbz9)

You can also configure arbitrary objects using an external script.

[Example: Configuring arbitrary objects using a script](#3whwml4)

**build.gradle**

task configure {  
 doLast {  
 def pos = new java.text.FieldPosition(10)  
 // Apply the script  
 apply from: 'other.gradle', to: pos  
 println pos.beginIndex  
 println pos.endIndex  
 }  
}

**other.gradle**

// Set properties.  
beginIndex = 1  
endIndex = 5

**Output of gradle -q configure**

> gradle -q configure  
1  
5

[Some Groovy basics](#35nkun2)

The [Groovy language](http://docs.groovy-lang.org/latest/html/documentation/index.html) provides plenty of features for creating DSLs, and the Gradle build language takes advantage of these. Understanding how the build language works will help you when you write your build script, and in particular, when you start to write custom plugins and tasks.

[Groovy JDK](#2bn6wsx)

Groovy adds lots of useful methods to the standard Java classes. For example, Iterable gets an each method, which iterates over the elements of the Iterable:

[Example: Groovy JDK methods](#qsh70q)

**build.gradle**

// Iterable gets an each() method  
configurations.runtime.each { File f -> println f }

Have a look at <http://groovy-lang.org/gdk.html> for more details.

[Property accessors](#3as4poj)

Groovy automatically converts a property reference into a call to the appropriate getter or setter method.

[Example: Property accessors](#1pxezwc)

**build.gradle**

// Using a getter method  
println project.buildDir  
println getProject().getBuildDir()  
  
// Using a setter method  
project.buildDir = 'target'  
getProject().setBuildDir('target')

[Optional parentheses on method calls](#49x2ik5)

Parentheses are optional for method calls.

[Example: Method call without parentheses](#2p2csry)

**build.gradle**

test.systemProperty 'some.prop', 'value'  
test.systemProperty('some.prop', 'value')

[List and map literals](#147n2zr)

Groovy provides some shortcuts for defining List and Map instances. Both kinds of literals are straightforward, but map literals have some interesting twists.

For instance, the “apply” method (where you typically apply plugins) actually takes a map parameter. However, when you have a line like “apply plugin:'java'”, you aren’t actually using a map literal, you’re actually using “named parameters”, which have almost exactly the same syntax as a map literal (without the wrapping brackets). That named parameter list gets converted to a map when the method is called, but it doesn’t start out as a map.

[Example: List and map literals](#3o7alnk)

**build.gradle**

// List literal  
test.includes = ['org/gradle/api/\*\*', 'org/gradle/internal/\*\*']  
  
List<String> list = new ArrayList<String>()  
list.add('org/gradle/api/\*\*')  
list.add('org/gradle/internal/\*\*')  
test.includes = list  
  
// Map literal.  
Map<String, String> map = [key1:'value1', key2: 'value2']  
  
// Groovy will coerce named arguments  
// into a single map argument  
apply plugin: 'java'

[Closures as the last parameter in a method](#23ckvvd)

The Gradle DSL uses closures in many places. You can find out more about closures [here](http://docs.groovy-lang.org/latest/html/documentation/index.html#_closures). When the last parameter of a method is a closure, you can place the closure after the method call:

[Example: Closure as method parameter](#ihv636)

**build.gradle**

repositories {  
 println "in a closure"  
}  
repositories() { println "in a closure" }  
repositories({ println "in a closure" })

[Closure delegate](#32hioqz)

Each closure has a delegate object, which Groovy uses to look up variable and method references which are not local variables or parameters of the closure. Gradle uses this for *configuration closures*, where the delegate object is set to the object to be configured.

[Example: Closure delegates](#1hmsyys)

**build.gradle**

dependencies {  
 assert delegate == project.dependencies  
 testCompile('junit:junit:4.12')  
 delegate.testCompile('junit:junit:4.12')  
}

[Default imports](#1ksv4uv)

To make build scripts more concise, Gradle automatically adds a set of import statements to the Gradle scripts. This means that instead of using throw new org.gradle.api.tasks.StopExecutionException() you can just type throw new StopExecutionException() instead.

Listed below are the imports added to each script:

**Gradle default imports**

import org.gradle.\*  
import org.gradle.api.\*  
import org.gradle.api.artifacts.\*  
import org.gradle.api.artifacts.component.\*  
import org.gradle.api.artifacts.dsl.\*  
import org.gradle.api.artifacts.ivy.\*  
import org.gradle.api.artifacts.maven.\*  
import org.gradle.api.artifacts.query.\*  
import org.gradle.api.artifacts.repositories.\*  
import org.gradle.api.artifacts.result.\*  
import org.gradle.api.artifacts.transform.\*  
import org.gradle.api.artifacts.type.\*  
import org.gradle.api.attributes.\*  
import org.gradle.api.capabilities.\*  
import org.gradle.api.component.\*  
import org.gradle.api.credentials.\*  
import org.gradle.api.distribution.\*  
import org.gradle.api.distribution.plugins.\*  
import org.gradle.api.dsl.\*  
import org.gradle.api.execution.\*  
import org.gradle.api.file.\*  
import org.gradle.api.initialization.\*  
import org.gradle.api.initialization.definition.\*  
import org.gradle.api.initialization.dsl.\*  
import org.gradle.api.invocation.\*  
import org.gradle.api.java.archives.\*  
import org.gradle.api.logging.\*  
import org.gradle.api.logging.configuration.\*  
import org.gradle.api.model.\*  
import org.gradle.api.plugins.\*  
import org.gradle.api.plugins.announce.\*  
import org.gradle.api.plugins.antlr.\*  
import org.gradle.api.plugins.buildcomparison.gradle.\*  
import org.gradle.api.plugins.osgi.\*  
import org.gradle.api.plugins.quality.\*  
import org.gradle.api.plugins.scala.\*  
import org.gradle.api.provider.\*  
import org.gradle.api.publish.\*  
import org.gradle.api.publish.ivy.\*  
import org.gradle.api.publish.ivy.plugins.\*  
import org.gradle.api.publish.ivy.tasks.\*  
import org.gradle.api.publish.maven.\*  
import org.gradle.api.publish.maven.plugins.\*  
import org.gradle.api.publish.maven.tasks.\*  
import org.gradle.api.publish.plugins.\*  
import org.gradle.api.publish.tasks.\*  
import org.gradle.api.reflect.\*  
import org.gradle.api.reporting.\*  
import org.gradle.api.reporting.components.\*  
import org.gradle.api.reporting.dependencies.\*  
import org.gradle.api.reporting.dependents.\*  
import org.gradle.api.reporting.model.\*  
import org.gradle.api.reporting.plugins.\*  
import org.gradle.api.resources.\*  
import org.gradle.api.specs.\*  
import org.gradle.api.tasks.\*  
import org.gradle.api.tasks.ant.\*  
import org.gradle.api.tasks.application.\*  
import org.gradle.api.tasks.bundling.\*  
import org.gradle.api.tasks.compile.\*  
import org.gradle.api.tasks.diagnostics.\*  
import org.gradle.api.tasks.incremental.\*  
import org.gradle.api.tasks.javadoc.\*  
import org.gradle.api.tasks.options.\*  
import org.gradle.api.tasks.scala.\*  
import org.gradle.api.tasks.testing.\*  
import org.gradle.api.tasks.testing.junit.\*  
import org.gradle.api.tasks.testing.junitplatform.\*  
import org.gradle.api.tasks.testing.testng.\*  
import org.gradle.api.tasks.util.\*  
import org.gradle.api.tasks.wrapper.\*  
import org.gradle.authentication.\*  
import org.gradle.authentication.aws.\*  
import org.gradle.authentication.http.\*  
import org.gradle.buildinit.plugins.\*  
import org.gradle.buildinit.tasks.\*  
import org.gradle.caching.\*  
import org.gradle.caching.configuration.\*  
import org.gradle.caching.http.\*  
import org.gradle.caching.local.\*  
import org.gradle.concurrent.\*  
import org.gradle.external.javadoc.\*  
import org.gradle.ide.visualstudio.\*  
import org.gradle.ide.visualstudio.plugins.\*  
import org.gradle.ide.visualstudio.tasks.\*  
import org.gradle.ide.xcode.\*  
import org.gradle.ide.xcode.plugins.\*  
import org.gradle.ide.xcode.tasks.\*  
import org.gradle.ivy.\*  
import org.gradle.jvm.\*  
import org.gradle.jvm.application.scripts.\*  
import org.gradle.jvm.application.tasks.\*  
import org.gradle.jvm.platform.\*  
import org.gradle.jvm.plugins.\*  
import org.gradle.jvm.tasks.\*  
import org.gradle.jvm.tasks.api.\*  
import org.gradle.jvm.test.\*  
import org.gradle.jvm.toolchain.\*  
import org.gradle.language.\*  
import org.gradle.language.assembler.\*  
import org.gradle.language.assembler.plugins.\*  
import org.gradle.language.assembler.tasks.\*  
import org.gradle.language.base.\*  
import org.gradle.language.base.artifact.\*  
import org.gradle.language.base.compile.\*  
import org.gradle.language.base.plugins.\*  
import org.gradle.language.base.sources.\*  
import org.gradle.language.c.\*  
import org.gradle.language.c.plugins.\*  
import org.gradle.language.c.tasks.\*  
import org.gradle.language.coffeescript.\*  
import org.gradle.language.cpp.\*  
import org.gradle.language.cpp.plugins.\*  
import org.gradle.language.cpp.tasks.\*  
import org.gradle.language.java.\*  
import org.gradle.language.java.artifact.\*  
import org.gradle.language.java.plugins.\*  
import org.gradle.language.java.tasks.\*  
import org.gradle.language.javascript.\*  
import org.gradle.language.jvm.\*  
import org.gradle.language.jvm.plugins.\*  
import org.gradle.language.jvm.tasks.\*  
import org.gradle.language.nativeplatform.\*  
import org.gradle.language.nativeplatform.tasks.\*  
import org.gradle.language.objectivec.\*  
import org.gradle.language.objectivec.plugins.\*  
import org.gradle.language.objectivec.tasks.\*  
import org.gradle.language.objectivecpp.\*  
import org.gradle.language.objectivecpp.plugins.\*  
import org.gradle.language.objectivecpp.tasks.\*  
import org.gradle.language.plugins.\*  
import org.gradle.language.rc.\*  
import org.gradle.language.rc.plugins.\*  
import org.gradle.language.rc.tasks.\*  
import org.gradle.language.routes.\*  
import org.gradle.language.scala.\*  
import org.gradle.language.scala.plugins.\*  
import org.gradle.language.scala.tasks.\*  
import org.gradle.language.scala.toolchain.\*  
import org.gradle.language.swift.\*  
import org.gradle.language.swift.plugins.\*  
import org.gradle.language.swift.tasks.\*  
import org.gradle.language.twirl.\*  
import org.gradle.maven.\*  
import org.gradle.model.\*  
import org.gradle.nativeplatform.\*  
import org.gradle.nativeplatform.platform.\*  
import org.gradle.nativeplatform.plugins.\*  
import org.gradle.nativeplatform.tasks.\*  
import org.gradle.nativeplatform.test.\*  
import org.gradle.nativeplatform.test.cpp.\*  
import org.gradle.nativeplatform.test.cpp.plugins.\*  
import org.gradle.nativeplatform.test.cunit.\*  
import org.gradle.nativeplatform.test.cunit.plugins.\*  
import org.gradle.nativeplatform.test.cunit.tasks.\*  
import org.gradle.nativeplatform.test.googletest.\*  
import org.gradle.nativeplatform.test.googletest.plugins.\*  
import org.gradle.nativeplatform.test.plugins.\*  
import org.gradle.nativeplatform.test.tasks.\*  
import org.gradle.nativeplatform.test.xctest.\*  
import org.gradle.nativeplatform.test.xctest.plugins.\*  
import org.gradle.nativeplatform.test.xctest.tasks.\*  
import org.gradle.nativeplatform.toolchain.\*  
import org.gradle.nativeplatform.toolchain.plugins.\*  
import org.gradle.normalization.\*  
import org.gradle.platform.base.\*  
import org.gradle.platform.base.binary.\*  
import org.gradle.platform.base.component.\*  
import org.gradle.platform.base.plugins.\*  
import org.gradle.play.\*  
import org.gradle.play.distribution.\*  
import org.gradle.play.platform.\*  
import org.gradle.play.plugins.\*  
import org.gradle.play.plugins.ide.\*  
import org.gradle.play.tasks.\*  
import org.gradle.play.toolchain.\*  
import org.gradle.plugin.devel.\*  
import org.gradle.plugin.devel.plugins.\*  
import org.gradle.plugin.devel.tasks.\*  
import org.gradle.plugin.management.\*  
import org.gradle.plugin.use.\*  
import org.gradle.plugins.ear.\*  
import org.gradle.plugins.ear.descriptor.\*  
import org.gradle.plugins.ide.\*  
import org.gradle.plugins.ide.api.\*  
import org.gradle.plugins.ide.eclipse.\*  
import org.gradle.plugins.ide.idea.\*  
import org.gradle.plugins.javascript.base.\*  
import org.gradle.plugins.javascript.coffeescript.\*  
import org.gradle.plugins.javascript.envjs.\*  
import org.gradle.plugins.javascript.envjs.browser.\*  
import org.gradle.plugins.javascript.envjs.http.\*  
import org.gradle.plugins.javascript.envjs.http.simple.\*  
import org.gradle.plugins.javascript.jshint.\*  
import org.gradle.plugins.javascript.rhino.\*  
import org.gradle.plugins.signing.\*  
import org.gradle.plugins.signing.signatory.\*  
import org.gradle.plugins.signing.signatory.pgp.\*  
import org.gradle.plugins.signing.type.\*  
import org.gradle.plugins.signing.type.pgp.\*  
import org.gradle.process.\*  
import org.gradle.swiftpm.\*  
import org.gradle.swiftpm.plugins.\*  
import org.gradle.swiftpm.tasks.\*  
import org.gradle.testing.base.\*  
import org.gradle.testing.base.plugins.\*  
import org.gradle.testing.jacoco.plugins.\*  
import org.gradle.testing.jacoco.tasks.\*  
import org.gradle.testing.jacoco.tasks.rules.\*  
import org.gradle.testkit.runner.\*  
import org.gradle.vcs.\*  
import org.gradle.vcs.git.\*  
import org.gradle.workers.\*

[1](#41mghml). Any language element except for statement labels.

Docs

* [User Manual](http://docs.google.com/userguide/userguide.html)
* [DSL Reference](http://docs.google.com/dsl/)
* [Release Notes](http://docs.google.com/release-notes.html)
* [Javadoc](http://docs.google.com/javadoc/)

News

* [Blog](https://blog.gradle.org/)
* [Newsletter](https://newsletter.gradle.com/)
* [Twitter](https://twitter.com/gradle)

Products

* [Build Scans](https://gradle.com/build-scans)
* [Build Cache](https://gradle.com/build-cache)
* [Enterprise Docs](https://gradle.com/enterprise/resources)

Get Help

* [Forums](https://discuss.gradle.org/c/help-discuss)
* [GitHub](https://github.com/gradle/)
* [Training](https://gradle.org/training/)
* [Services](https://gradle.org/services/)

Subscribe for important Gradle updates and news

Subscribe

By entering your email, you agree to our [Terms](https://gradle.org/terms/) and [Privacy Policy](https://gradle.org/privacy/), including receipt of emails. You can unsubscribe at any time.

© [Gradle Inc.](https://gradle.com) 2018 All rights reserved.

[Careers](https://gradle.com/careers) | [Privacy](https://gradle.org/privacy) | [Terms of Service](https://gradle.org/terms) | [Contact](https://gradle.org/contact/)