免费电子书

学习

Kotlin

Free unaffiliated eBook created from
Stack Overflow contributors.

#kotlin

1
1: Kotlin 2
2
Kotlin 2
2
Examples 2
3
Hello World 3
Hello WorldCompanion 3
varargs 4
Kotlin 4
4
2: DSL 6
6
Examples 6
DSLInfix 6
DSL 6
lambda 6
lambdas 6
3: Java 8 Stream 8
8
8
8
8
8
9
Examples 9
9
9
9
9

10

10

10

10

10

10

2 - 11

3 - 11

4 - 11

5 - Int 11

6 - Ints 12

7 - IntString 12

12

- 12

1 - 13

5 - 13

6 - 14

7a - 15

7b - SummarizingInt 15

4: JUnit 17

Examples 17

17

5: Kotlin Android 18

18

Examples 18

18

18

19

Kotlin 19

6: Kotlin for Java Developers 21

21

Examples 21

21
21
21
IFTRY 22
7: Kotlin 23
23
23
Examples 23
23
8: Kotlin 25
Examples 25
toString 25
9: 26
Examples 26
try-catch-finally 26
10: Vararg 27
27
Examples 27
vararg 27
Spread Operatorvararg 27
11: 29
29
29
Examples 29
29
Lambda 29
30
31
32
32
32
12: 33

33
Examples 33
1 33
13: 34
34
Examples 34
java/ 34
34
14: 36
36
36
Examples 36
36
36
Java 36
36
37
15: 39
39
39
Examples 39
39
16: KotlinRecyclerView 40
40
Examples 40
40
17: Lambdas 42
42
42
Examples 43
Lambda 43

Lambda ... 43
Lambda ... 43
18: ... 44
... 44
Examples ... 44
... 44
... 44
... 44
... 44
... 44
19: ... 46
Examples ... 46
... 46
... 46
... 46
... 47
20: ... 48
Examples ... 48
DTOPOJO / POCO ... 48
... 48
... 48
KotlinSerializableserialVersionUid ... 48
Kotlin ... 49
let ... 49
apply ... 50
21: ... 51
... 51
... 51
Examples ... 51
... 51
... 51
... 51

Java 7+ Path 52
..... 52
Java 8 TemporalISO 52
Companion 53
..... 53
..... 53
..... 53
..... 54
22: 55
..... 55
Examples 55
..... 55
..... 55
..... 55
..... 55
..... 56
..... 56
..... 57
23: 58
Examples 58
..... 58
..... 58
..... 58
..... 59
..... 59
..... 59
..... 59
24: 60
Examples 60
NullableNon Nullable 60
..... 60
null 60
.....

60

Iterable 61

Null Coalescing / Elvis 61

61

:) 61

25: 63

63

Examples 63

if 63

if 63

whenif-else-if 63

when 64

64

65

26: 66

66

Examples 66

66

66

66

66

27: 68

Examples 68

68

68

68

68

Kotlin 69

RegEx 69

69

69

findCharSequencestartIndexIntMatchResult 69

findAllinputCharSequencestartIndexIntSequence 70
matchEntireCharSequenceMatchResult 70
matchesCharSequenceBoolean 70
containsMatchIninputCharSequenceBoolean 70
splitCharSequencelimitIntList 70
replaceCharSequencereplacementStringString 71
28: 72
72
72
72
72
72
Examples 72
- 72
72
29: 74
Examples 74
74
74
30: 75
75
Examples 75
75
31: kotlin 76
76
Examples 76
kotlin.logging 76
32: 77
77
Examples 77
x 77

iterables ... 77
... 77
... 78
kotlin ... 78
... 78
... 78
33: ... 80
... 80
... 80
Kotlin ... 80
Examples ... 80
... 80
34: ... 82
... 82
... 82
... 82
Examples ... 82
'open' ... 82
... 82
... 82
... 82
... 83
... 83
... 83
... 83
Person ... 83
... 83
... 83
... 83
35: ... 85
... 85
Examples ... 85

85
downTo 85
step 85
85
36: 86
86
86
86
Examples 86
86
86
37: Kotlin 87
Examples 87
Gradle 87
JVM 87
Android 87
JS 87
Android Studio 88
88
88
Java 88
GroovyGradleKotlin 89
38: 91
91
91
Examples 91
91
91
set 91
92

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: kotlin

It is an unofficial and free Kotlin ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Kotlin.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: Kotlin

KotlinJetBrainsJVM。 KotlinJava 100。 KotlinJava。 KotlinJVMJavaJavaScript。

Kotlin

KotlinEclipseIntelliJIDE。 KotlinMaven Ant Gradle。

`$ kotlinc Main.kt`java`MainKt.class` Kt。 `$ java MainKt$ java MainKt`

```
Exception in thread "main" java.lang.NoClassDefFoundError: kotlin/jvm/internal/Intrinsics
    at MainKt.main(Main.kt)
Caused by: java.lang.ClassNotFoundException: kotlin.jvm.internal.Intrinsics
    at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:335)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    ... 1 more
```

JavaKotlin runt-time jar。

```
java -cp .:/path/to/kotlin/runtime/jar/kotlin-runtime.jar MainKt
```

1.0.0	2016215
1.0.1	2016316
1.0.2	2016513
1.0.3	2016630
1.0.4	2016922
1.0.5	2016118
1.0.6	20161227
1.1.0	201731
1.1.1	2017314
1.1.2	2017425
1.1.3	2017623

Examples

Kotlin`main`。 Kotlin“Hello World”

```
package my.program

fun main(args: Array<String>) {
    println("Hello, world!")
}
```

`Main.kt`

JVM。 `my.program.MainKt` 。

package

```
@file:JvmName("MyApp")
```

`my.program.MyApp` 。

- `@JvmName`。
-

Hello World

Kotlin 。

```
package my.program

object App {
    @JvmStatic fun main(args: Array<String>) {
        println("Hello World")
    }
}
```

`my.program.App` 。

`App` 。 `App`。

- `@JvmStatic`

Hello WorldCompanion

CompanionKotlin`main`。

```
package my.program

class App {
    companion object {
        @JvmStatic fun main(args: Array<String>) {
            println("Hello World")
        }
    }
}
```

`my.program.App` ◦

`App` ◦ `Object Declaration`◦

“hello”

```
class App {
    companion object {
        @JvmStatic fun main(args: Array<String>) {
            App().run()
        }
    }

    fun run() {
        println("Hello World")
    }
}
```

- @JvmStatic

varargs

varargs

```
package my.program

fun main(vararg args: String) {
    println("Hello, world!")
}
```

Kotlin

javaJava◦ Kotlin◦

`javac`java◦ `java`java◦

`kotlinc`kotlin`kotlin`kotlin◦

Kotlin◦ N1 2 3◦

Kotlin◦

```
fun main(args: Array<String>) {

    println("Enter Two number")
    var (a, b) = readLine()!!.split(' ') // !! this operator use for
NPE(NullPointerException).

    println("Max number is : ${maxNum(a.toInt(), b.toInt())}")
}
```

```
fun maxNum(a: Int, b: Int): Int {

    var max = if (a > b) {
        println("The value of a is $a");
        a
    } else {
        println("The value of b is $b")
        b
    }

    return max;

}
```

。

```
Enter Two number
71 89 // Enter two number from command line

The value of b is 89
Max number is: 89
```

!!Null Safety 。

Intellij。

Kotlin https://riptutorial.com/zh-CN/kotlin/topic/490/kotlin

2: DSL

KotlinDSL 。

Examples

DSLInfix

```
infix fun <T> T?.shouldBe(expected: T?) = assertEquals(expected, this)
```

DSL

```
@Test
fun test() {
  100.plusOne() shouldBe 101
}
```

DSL

```
class MyExample(val i: Int) {
  operator fun <R> invoke(block: MyExample.() -> R) = block()
  fun Int.bigger() = this > i
}
```

DSL

```
fun main2(args: Array<String>) {
    val ex = MyExample(233)
    ex {
        // bigger is defined in the context of `ex`
        // you can only call this method inside this context
        if (777.bigger()) kotlin.io.println("why")
    }
}
```

lambda

```
val r = Random(233)
infix inline operator fun Int.rem(block: () -> Unit) {
  if (r.nextInt(100) < this) block()
}
```

DSL

```
20 % { println("The possibility you see this message is 20%") }
```

lambdas

```
operator fun <R> String.invoke(block: () -> R) = {
  try { block.invoke() }
  catch (e: AssertException) { System.err.println("$this\n${e.message}") }
}
```

DSL

```
"it should return 2" {
   parse("1 + 1").buildAST().evaluate() shouldBe 2
}
```

shouldBeInfix approach to build DSLInfix approach to build DSL ◦

DSL https://riptutorial.com/zh-CN/kotlin/topic/10042/dsl

3: Java 8 Stream

Kotlin。 Sequence。

asSequence()Sequence 。 Sequence 。 toList() toSet() toMap()Sequence 。

```
// switch to and from lazy
val someList = items.asSequence().filter { ... }.take(10).map { ... }.toList()

// switch to lazy, but sorted() brings us out again at the end
val someList = items.asSequence().filter { ... }.take(10).map { ... }.sorted()
```

Kotlin。 Kotlin。 JavaNPE。 Kotlin

```
val someList = people.filter { it.age <= 30 }.map { it.name }
```

```
val someList: List<String> = people.filter { it.age <= 30 }.map { it.name }
```

Kotlinpeoplepeople.ageIntIntpeople.nameStringmapList<String> String List 。

peoplenull List<People>?

```
val someList = people?.filter { it.age <= 30 }?.map { it.name }
```

List<String>? *KotlinKotlinKotlin*

Kotlin。 Sequence“”start。 Java 8Kotlin

```
// Java:
Stream<String> stream =
Stream.of("d2", "a2", "b1", "b3", "c").filter(s -> s.startsWith("b"));

stream.anyMatch(s -> true);    // ok
stream.noneMatch(s -> true);   // exception
```

```
// Kotlin:
val stream = listOf("d2", "a2", "b1", "b3", "c").asSequence().filter { it.startsWith('b' ) }

stream.forEach(::println) // b1, b2

println("Any B ${stream.any { it.startsWith('b') }}") // Any B true
println("Any C ${stream.any { it.startsWith('c') }}") // Any C false

stream.forEach(::println) // b1, b2
```

Java

```
// Java:
Supplier<Stream<String>> streamSupplier =
```

```
    () -> Stream.of("d2", "a2", "b1", "b3", "c")
          .filter(s -> s.startsWith("a"));

streamSupplier.get().anyMatch(s -> true);   // ok
streamSupplier.get().noneMatch(s -> true);  // ok
```

Kotlin。 `SequenceSequenceconstrainOnce()`

```
val stream = listOf("d2", "a2", "b1", "b3", "c").asSequence().filter { it.startsWith('b' ) }
        .constrainOnce()

stream.forEach(::println) // b1, b2
stream.forEach(::println) // Error:java.lang.IllegalStateException: This sequence can be
consumed only once.
```

- Iterable API
- Array API
- List API
- Map API

Examples

```
// Java:
List<String> list = people.stream().map(Person::getName).collect(Collectors.toList());
```

```
// Kotlin:
val list = people.map { it.name }  // toList() not needed
```

```
// Java:
String joined = things.stream()
                      .map(Object::toString)
                      .collect(Collectors.joining(", "));
```

```
// Kotlin:
val joined = things.joinToString() // ", " is used as separator, by default
```

```
// Java:
int total = employees.stream()
                     .collect(Collectors.summingInt(Employee::getSalary)));
```

```
// Kotlin:
val total = employees.sumBy { it.salary }
```

```
// Java:
Map<Department, List<Employee>> byDept
     = employees.stream()
                .collect(Collectors.groupingBy(Employee::getDepartment));
```

```
// Kotlin:
```

```
val byDept = employees.groupBy { it.department }
```

```
// Java:
Map<Department, Integer> totalByDept
     = employees.stream()
                .collect(Collectors.groupingBy(Employee::getDepartment,
                     Collectors.summingInt(Employee::getSalary)));
```

```
// Kotlin:
val totalByDept = employees.groupBy { it.dept }.mapValues { it.value.sumBy { it.salary }}
```

```
// Java:
Map<Boolean, List<Student>> passingFailing =
     students.stream()
             .collect(Collectors.partitioningBy(s -> s.getGrade() >= PASS_THRESHOLD));
```

```
// Kotlin:
val passingFailing = students.partition { it.grade >= PASS_THRESHOLD }
```

```
// Java:
List<String> namesOfMaleMembersCollect = roster
    .stream()
    .filter(p -> p.getGender() == Person.Sex.MALE)
    .map(p -> p.getName())
    .collect(Collectors.toList());
```

```
// Kotlin:
val namesOfMaleMembers = roster.filter { it.gender == Person.Sex.MALE }.map { it.name }
```

```
// Java:
Map<Person.Sex, List<String>> namesByGender =
      roster.stream().collect(
        Collectors.groupingBy(
            Person::getGender,
            Collectors.mapping(
                Person::getName,
                Collectors.toList())));
```

```
// Kotlin:
val namesByGender = roster.groupBy { it.gender }.mapValues { it.value.map { it.name } }
```

```
// Java:
List<String> filtered = items.stream()
    .filter( item -> item.startsWith("o") )
    .collect(Collectors.toList());
```

```
// Kotlin:
val filtered = items.filter { item.startsWith('o') }
```

```
// Java:
String shortest = items.stream()
```

```
    .min(Comparator.comparing(item -> item.length()))
    .get();
```

```
// Kotlin:
val shortest = items.minBy { it.length }
```

2 -

```
// Java:
Stream.of("a1", "a2", "a3")
    .findFirst()
    .ifPresent(System.out::println);
```

```
// Kotlin:
sequenceOf("a1", "a2", "a3").firstOrNull()?.apply(::println)
```

3 -

```
// Java:
IntStream.range(1, 4).forEach(System.out::println);
```

```
// Kotlin:  (inclusive range)
(1..3).forEach(::println)
```

4 -

```
// Java:
Arrays.stream(new int[] {1, 2, 3})
    .map(n -> 2 * n + 1)
    .average()
    .ifPresent(System.out::println); // 5.0
```

```
// Kotlin:
arrayOf(1,2,3).map { 2 * it + 1}.average().apply(::println)
```

5 - Int

```
// Java:
Stream.of("a1", "a2", "a3")
    .map(s -> s.substring(1))
    .mapToInt(Integer::parseInt)
    .max()
    .ifPresent(System.out::println);  // 3
```

```
// Kotlin:
sequenceOf("a1", "a2", "a3")
    .map { it.substring(1) }
    .map(String::toInt)
    .max().apply(::println)
```

6 - Ints

```
// Java:
IntStream.range(1, 4)
    .mapToObj(i -> "a" + i)
    .forEach(System.out::println);

// a1
// a2
// a3
```

```
// Kotlin:  (inclusive range)
(1..3).map { "a$it" }.forEach(::println)
```

7 - IntString

```
// Java:
Stream.of(1.0, 2.0, 3.0)
    .mapToInt(Double::intValue)
    .mapToObj(i -> "a" + i)
    .forEach(System.out::println);

// a1
// a2
// a3
```

```
// Kotlin:
sequenceOf(1.0, 2.0, 3.0).map(Double::toInt).map { "a$it" }.forEach(::println)
```

```
// Java:
long count = items.stream().filter( item -> item.startsWith("t")).count();
```

```
// Kotlin:
val count = items.filter { it.startsWith('t') }.size
// but better to not filter, but count with a predicate
val count = items.count { it.startsWith('t') }
```

-

```
// Java:
List<String> myList = Arrays.asList("a1", "a2", "b1", "c2", "c1");

myList.stream()
      .filter(s -> s.startsWith("c"))
      .map(String::toUpperCase)
     .sorted()
     .forEach(System.out::println);

// C1
// C2
```

```
// Kotlin:
```

```
val list = listOf("a1", "a2", "b1", "c2", "c1")
list.filter { it.startsWith('c') }.map (String::toUpperCase).sorted()
        .forEach (::println)
```

1 -

```
// Java:
Arrays.asList("a1", "a2", "a3")
    .stream()
    .findFirst()
    .ifPresent(System.out::println);
```

```
// Kotlin:
listOf("a1", "a2", "a3").firstOrNull()?.apply(::println)
```

StringifPresent

```
// Kotlin:
inline fun String?.ifPresent(thenDo: (String)->Unit) = this?.apply { thenDo(this) }

// now use the new extension function:
listOf("a1", "a2", "a3").firstOrNull().ifPresent(::println)
```

`apply()`

`?.` http //stackoverflow.com/questions/34498562/in-kotlin-what-is-the-idiomatic-way-to-deal-with-nullable-values-referencing-o/34498563 34498563

5 -

```
// Java:
String phrase = persons
        .stream()
        .filter(p -> p.age >= 18)
        .map(p -> p.name)
        .collect(Collectors.joining(" and ", "In Germany ", " are of legal age."));

System.out.println(phrase);
// In Germany Max and Peter and Pamela are of legal age.
```

```
// Kotlin:
val phrase = persons
        .filter { it.age >= 18 }
        .map { it.name }
        .joinToString(" and ", "In Germany ", " are of legal age.")

println(phrase)
// In Germany Max and Peter and Pamela are of legal age.
```

Kotlin

```
// Kotlin:
// data class has equals, hashcode, toString, and copy methods automagically
```

```
data class Person(val name: String, val age: Int)

val persons = listOf(Person("Tod", 5), Person("Max", 33),
                     Person("Frank", 13), Person("Peter", 80),
                     Person("Pamela", 18))
```

6 -

```
// Java:
Map<Integer, String> map = persons
        .stream()
        .collect(Collectors.toMap(
                p -> p.age,
                p -> p.name,
                (name1, name2) -> name1 + ";" + name2));

System.out.println(map);
// {18=Max, 23=Peter;Pamela, 12=David}
```

Kotlin。/`Map`

```
// Kotlin:
val map1 = persons.map { it.age to it.name }.toMap()
println(map1)
// output: {18=Max, 23=Pamela, 12=David}
// Result: duplicates overridden, no exception similar to Java 8

val map2 = persons.toMap({ it.age }, { it.name })
println(map2)
// output: {18=Max, 23=Pamela, 12=David}
// Result: same as above, more verbose, duplicates overridden

val map3 = persons.toMapBy { it.age }
println(map3)
// output: {18=Person(name=Max, age=18), 23=Person(name=Pamela, age=23), 12=Person(name=David,
age=12)}
// Result: duplicates overridden again

val map4 = persons.groupBy { it.age }
println(map4)
// output: {18=[Person(name=Max, age=18)], 23=[Person(name=Peter, age=23), Person(name=Pamela,
age=23)], 12=[Person(name=David, age=12)]}
// Result: closer, but now have a Map<Int, List<Person>> instead of Map<Int, String>

val map5 = persons.groupBy { it.age }.mapValues { it.value.map { it.name } }
println(map5)
// output: {18=[Max], 23=[Peter, Pamela], 12=[David]}
// Result: closer, but now have a Map<Int, List<String>> instead of Map<Int, String>
```

```
// Kotlin:
val map6 = persons.groupBy { it.age }.mapValues { it.value.joinToString(";") { it.name } }

println(map6)
// output: {18=Max, 23=Peter;Pamela, 12=David}
// Result: YAY!!
```

`joinToStringPersonPerson.name`

。

7a -

```
// Java (verbose):
Collector<Person, StringJoiner, String> personNameCollector =
Collector.of(
        () -> new StringJoiner(" | "),          // supplier
        (j, p) -> j.add(p.name.toUpperCase()),  // accumulator
        (j1, j2) -> j1.merge(j2),               // combiner
        StringJoiner::toString);                // finisher

String names = persons
        .stream()
        .collect(personNameCollector);

System.out.println(names);  // MAX | PETER | PAMELA | DAVID

// Java (concise)
String names = persons.stream().map(p -> p.name.toUpperCase()).collect(Collectors.joining(" |
"));
```

```
// Kotlin:
val names = persons.map { it.name.toUpperCase() }.joinToString(" | ")
```

7b - SummarizingInt

```
// Java:
IntSummaryStatistics ageSummary =
    persons.stream()
           .collect(Collectors.summarizingInt(p -> p.age));

System.out.println(ageSummary);
// IntSummaryStatistics{count=4, sum=76, min=12, average=19.000000, max=23}
```

```
// Kotlin:

// something to hold the stats...
data class SummaryStatisticsInt(var count: Int = 0,
                                var sum: Int = 0,
                                var min: Int = Int.MAX_VALUE,
                                var max: Int = Int.MIN_VALUE,
                                var avg: Double = 0.0) {
    fun accumulate(newInt: Int): SummaryStatisticsInt {
        count++
        sum += newInt
        min = min.coerceAtMost(newInt)
        max = max.coerceAtLeast(newInt)
        avg = sum.toDouble() / count
        return this
    }
}

// Now manually doing a fold, since Stream.collect is really just a fold
val stats = persons.fold(SummaryStatisticsInt()) { stats, person ->
stats.accumulate(person.age) }
```

```
println(stats)
// output: SummaryStatisticsInt(count=4, sum=76, min=12, max=23, avg=19.0)
```

2Kotlin stdlib

```
// Kotlin:
inline fun Collection<Int>.summarizingInt(): SummaryStatisticsInt
        = this.fold(SummaryStatisticsInt()) { stats, num -> stats.accumulate(num) }

inline fun <T: Any> Collection<T>.summarizingInt(transform: (T)->Int): SummaryStatisticsInt =
        this.fold(SummaryStatisticsInt()) { stats, item -> stats.accumulate(transform(item)) }
```

summarizingInt

```
val stats2 = persons.map { it.age }.summarizingInt()

// or

val stats3 = persons.summarizingInt { it.age }
```

◦ Sequence◦

Java 8 Stream https://riptutorial.com/zh-CN/kotlin/topic/707/java-8-stream

4: JUnit

Examples

JUnit

```
@Rule @JvmField val myRule = TemporaryFolder()
```

`@JvmFieldmyRule` ◦ JUnit◦

JUnit https://riptutorial.com/zh-CN/kotlin/topic/6973/junit

5: Kotlin Android

KotlinAndroidButterKnife。。

Examples

gradle 。

Kotlin `build.gradle`。

```
buildscript {
    ...
}

apply plugin: "com.android.application"
...
apply plugin: "kotlin-android"
apply plugin: "kotlin-android-extensions"
...
```

`activity_main.xmlactivity_main.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="My button"/>
</LinearLayout>
```

Kotlin

```
import kotlinx.android.synthetic.main.activity_main.my_button

class MainActivity: Activity() {
    override fun onCreate(savedInstanceBundle: Bundle?) {
        super.onCreate(savedInstanceBundle)
        setContentView(R.layout.activity_main)
        // my_button is already casted to a proper type of "Button"
        // instead of being a "View"
        my_button.setText("Kotlin rocks!")
    }
}
```

*ID

```
// my_button can be used the same way as before
import kotlinx.android.synthetic.main.activity_main.*
```

//

```
import kotlinx.android.synthetic.main.activity_main.my_button

class NotAView {
    init {
        // This sample won't compile!
        my_button.setText("Kotlin rocks!")
    }
}
```

AndroidAndroid。 `build.gradle`

```
android {
    productFlavors {
        paid {
            ...
        }
        free {
            ...
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/buy_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Buy full version"/>
</LinearLayout>
```

```
import kotlinx.android.synthetic.free.main_activity.buy_button
```

Kotlin

```
mView.afterMeasured {
  // inside this block the view is completely drawn
  // you can get view's height/width, it.height / it.width
}
```

```
inline fun View.afterMeasured(crossinline f: View.() -> Unit) {
viewTreeObserver.addOnGlobalLayoutListener(object : ViewTreeObserver.OnGlobalLayoutListener {
    override fun onGlobalLayout() {
        if (measuredHeight > 0 && measuredWidth > 0) {
            viewTreeObserver.removeOnGlobalLayoutListener(this)
            f()
        }
    }
})
```

```
}
```

Kotlin Android https://riptutorial.com/zh-CN/kotlin/topic/9474/kotlin-android

6: Kotlin for Java Developers

KotlinJava。

JavaKotlinKotlinJava。

Examples

KotlinJava

```
val i : Int = 42
```

- `valvarfinal` “ **val** ue”**var** iable。
- `::`
- Kotlin

Java	
`int i = 42;`	`var i = 42  var i : Int = 42`
`final int i = 42;`	`val i = 42`

- `;`
- Kotlin
- Kotlin**100Java**
- Kotlin JVM
- Kotlin
- Kotlin`equals` / `hashCode`
- Kotlin
- Kotlin **`new`** 。 。
- Kotlin 。 `val a = someMap["key"]`
- KotlinJVM**Java Script** Kotlin
- Kotlin **Java 6** Android
- Kotlin**Android**
- Kotlin。
- Kotlin**Coroutines**

Kotlin`==equals ===`。

Java	
`a.equals(b);`	`a == b`
`a == b;`	`a === b`

Java	
`a != b;`	`a !== b`

https //kotlinlang.org/docs/reference/equality.html

IFTRY

Kotlin `if try`void。

KotlinJava*Elvis*

```
val i = if (someBoolean) 33 else 42
```

`try`

```
val i = try {
    Integer.parseInt(someString)
}
catch (ex : Exception)
{
    42
}
```

Kotlin for Java Developers https://riptutorial.com/zh-CN/kotlin/topic/10099/kotlin-for-java-developers

7: Kotlin

Kotlin。

1. Kotlin.kt。
2. KotlinAnyJavaObject。
3. val - var -
4. 。
5. Unit.It。 6.===。 aba === btrue。

Examples

1.。 。

```
    fun printHello(name: String?): Unit {
        if (name != null)
            println("Hello ${name}")
    }

    fun printHello(name: String?) {
        ...
    }
```

2.Single-Expression=

```
  fun double(x: Int): Int = x * 2
```

```
  fun double(x: Int) = x * 2
```

3.String。

```
In java:
    int num=10
    String s  = "i =" + i;

In Kotlin
    val num = 10
    val s = "i = $num"
```

4.Kotlinnullnullnull。 Stringnull

```
var a: String = "abc"
a = null // compilation error
```

String

```
var b: String? = "abc"
```

```
b = null // ok
```

5.Kotlin==。 = = b

```
    a?.equals(b) ?: (b === null)
```

Kotlin https://riptutorial.com/zh-CN/kotlin/topic/10648/kotlin

8: Kotlin

Examples

toString

Kotlin`toString`null`String?String?`。

Android`EditText`。

```
// Incorrect:
val text = view.textField?.text.toString() ?: ""
```

`"null"`。

```
// Correct:
val text = view.textField?.text?.toString() ?: ""
```

Kotlin https://riptutorial.com/zh-CN/kotlin/topic/6608/kotlin

9:

Examples

try-catch-finally

KotlinJava

```
try {
    doSomething()
}
catch(e: MyException) {
    handle(e)
}
finally {
    cleanup()
}
```

```
try {
    doSomething()
}
catch(e: FileSystemException) {
    handle(e)
}
catch(e: NetworkException) {
    handle(e)
}
catch(e: MemoryException) {
    handle(e)
}
finally {
    cleanup()
}
```

try

```
val s: String? = try { getString() } catch (e: Exception) { null }
```

Kotlin。

```
fun fileToString(file: File) : String {
    //readAllBytes throws IOException, but we can omit catching it
    fileContent = Files.readAllBytes(file)
    return String(fileContent)
}
```

https://riptutorial.com/zh-CN/kotlin/topic/7246/

10: Vararg

- **Vararg** vararg。
- **Spread Operator** * “”。

Examples

vararg

vararg。

```
fun printNumbers(vararg numbers: Int) {
    for (number in numbers) {
        println(number)
    }
}
```

。

```
printNumbers(0, 1)                // Prints "0" "1"
printNumbers(10, 20, 30, 500)     // Prints "10" "20" "30" "500"
```

Vararg。

Spread Operatorvararg

Spread Operator *vararg。

.....

```
fun printNumbers(vararg numbers: Int) {
    for (number in numbers) {
        println(number)
    }
}
```

...

```
val numbers = intArrayOf(1, 2, 3)
printNumbers(*numbers)

// This is the same as passing in (1, 2, 3)
```

...

```
val numbers = intArrayOf(1, 2, 3)
printNumbers(10, 20, *numbers, 30, 40)
```

```
// This is the same as passing in (10, 20, 1, 2, 3, 30, 40)
```

Vararg https://riptutorial.com/zh-CN/kotlin/topic/5835/vararg

11:

- ***Params*** = ...
- ***Params*** {...}
- ***PARAMS*** {...}
- < > ***PARAMS*** {...}
- {...}
- { ***ArgName*** ***ArgType*** - > ...}
- { ***ArgName*** - > ...}
- { ***ArgNames*** - > ...}
- { ***ArgName*** ***ArgType*** - > ...}

PARAMS	***Name*** : ***Type***
ArgName	
ArgType	***ArgName***
ArgNames	ArgName

Examples

“Lambda”。“”

```
# Takes no parameters and returns anything
() -> Any?

# Takes a string and an integer and returns ReturnType
(arg1: String, arg2: Int) -> ReturnType
```

`() -> Any?` lambda

```
fun twice(x: () -> Any?) {
    x(); x();
}

fun main() {
    twice {
        println("Foo")
    } # => Foo
      # => Foo
}
```

Lambda

Lambda。{braces} - ->。

```
{ name: String ->
    "Your name is $name" //This is returned
}
```

lambda。

lambda。

```
{ argumentOne:String, argumentTwo:String ->
    "$argumentOne - $argumentTwo"
}
```

lambdait。

```
{ "Your name is $it" }
```

lambda。

```
# These are identical
listOf(1, 2, 3, 4).map { it + 2 }
listOf(1, 2, 3, 4).map({ it + 2 })
```

::。 。

```
fun addTwo(x: Int) = x + 2
listOf(1, 2, 3, 4).map(::addTwo) # => [3, 4, 5, 6]
```

(ParamTypeA, ParamTypeB, ...) -> ReturnType ParamTypeA ParamTypeB ...`ReturnType1。

```
fun foo(p0: Foo0, p1: Foo1, p2: Foo2): Bar {
    //...
}
println(::foo::class.java.genericInterfaces[0])
// kotlin.jvm.functions.Function3<Foo0, Foo1, Foo2, Bar>
// Human readable type: (Foo0, Foo1, Foo2) -> Bar
```

。

```
class Foo
fun Foo.foo(p0: Foo0, p1: Foo1, p2: Foo2): Bar {
    //...
}
val ref = Foo::foo
println(ref::class.java.genericInterfaces[0])
// kotlin.jvm.functions.Function4<Foo, Foo0, Foo1, Foo2, Bar>
// Human readable type: (Foo, Foo0, Foo1, Foo2) -> Bar
// takes 4 parameters, with receiver as first and actual parameters following, in their order

// this function can't be called like an extension function, though
```

```
val ref = Foo::foo
Foo().ref(Foo0(), Foo1(), Foo2()) // compile error

class Bar {
    fun bar()
}
print(Bar::bar) // works on member functions, too.
```

。

```
object Foo
fun Foo.foo(p0: Foo0, p1: Foo1, p2: Foo2): Bar {
    //...
}
val ref = Foo::foo
println(ref::class.java.genericInterfaces[0])
// kotlin.jvm.functions.Function3<Foo0, Foo1, Foo2, Bar>
// Human readable type: (Foo0, Foo1, Foo2) -> Bar
// takes 3 parameters, receiver not needed

object Bar {
    fun bar()
}
print(Bar::bar) // works on member functions, too.
```

kotlin 1.1 。

1.1.0

```
fun makeList(last: String?): List<String> {
    val list = mutableListOf("a", "b", "c")
    last?.let(list::add)
    return list
}
```

。 。

。 。

```
class Foo
class Bar {
    fun Foo.foo() {}
    val ref = Foo::foo // compile error
}
```

`fun`。 `Unit` 。 `{}` 。 `Unit` return。

```
fun sayMyName(name: String): String {
    return "Your name is $name"
}
```

```
fun sayMyName(name: String): String = "Your name is $name"
```

```
fun sayMyName(name: String) = "Your name is $name"
```

◦ ◦

```
fun sayMyName(name: String): String = "Your name is $name"
```

inlineinlineC - ◦ lambdas◦

```
inline fun sayMyName(name: String) = "Your name is $name"
```

C

```
inline fun sayMyName() = "Your name is $name"

fun main() {
    val name = "Foo"
    sayMyName() # => Unresolved reference: name
}
```

Kotlin+* ◦ ◦ operator

```
data class IntListWrapper (val wrapped: List<Int>) {
    operator fun get(position: Int): Int = wrapped[position]
}

val a = IntListWrapper(listOf(1, 2, 3))
a[1] // == 2
```

https://riptutorial.com/zh-CN/kotlin/topic/1280/

12:

Kotlin

Examples

1

```
fun main(args: Array<String>) {
    launch(CommonPool) { // create new coroutine in common thread pool
        delay(1000L) // non-blocking delay for 1 second (default time unit is ms)
        println("World!") // print after delay
    }
    println("Hello,") // main function continues while coroutine is delayed
    Thread.sleep(2000L) // block main thread for 2 seconds to keep JVM alive
}
```

```
Hello,
World!
```

https://riptutorial.com/zh-CN/kotlin/topic/10936/

13:

`object`。 javaSingletons。 java。 Javakotlin。

Examples

java/

```
object CommonUtils {

    var anyname: String ="Hello"

    fun dispMsg(message: String) {
        println(message)
    }
}
```

```
CommonUtils.anyname
CommonUtils.dispMsg("like static call")
```

Kotlin。 `SomeSingleton.INSTANCE`。

java

```
public enum SharedRegistry {
    INSTANCE;
    public void register(String key, Object thing) {}
}

public static void main(String[] args) {
    SharedRegistry.INSTANCE.register("a", "apple");
    SharedRegistry.INSTANCE.register("b", "boy");
    SharedRegistry.INSTANCE.register("c", "cat");
    SharedRegistry.INSTANCE.register("d", "dog");
}
```

kotlin

```
object SharedRegistry {
    fun register(key: String, thing: Object) {}
}

fun main(Array<String> args) {
    SharedRegistry.register("a", "apple")
    SharedRegistry.register("b", "boy")
    SharedRegistry.register("c", "cat")
    SharedRegistry.register("d", "dog")
}
```

。

https://riptutorial.com/zh-CN/kotlin/topic/10152/

14:

。

。

JVMJAR `kotlin-reflect.jar` 。 JS。

Examples

`KClass`

```
val c1 = String::class
val c2 = MyClass::class
```

Kotlin。

```
fun isPositive(x: Int) = x > 0

val numbers = listOf(-2, -1, 0, 1, 2)
println(numbers.filter(::isPositive)) // [1, 2]
```

Java

Kotlin`KClass`Java`Class` `KClass.java`

```
val stringKClass: KClass<String> = String::class
val c1: Class<String> = stringKClass.java

val c2: Class<MyClass> = MyClass::class.java
```

`KClass`。

`BaseExampleExample`

```
open class BaseExample(val baseField: String)

class Example(val field1: String, val field2: Int, baseField: String):
    BaseExample(baseField) {

    val field3: String
        get() = "Property without backing field"

    val field4 by lazy { "Delegated value" }

    private val privateField: String = "Private value"
}
```

```
val example = Example(field1 = "abc", field2 = 1, baseField = "someText")
```

```
example::class.memberProperties.forEach { member ->
    println("${member.name} -> ${member.get(example)}")
}
```

◦ `private val privateField`private`member.get(example)`◦ ◦ Java getter◦ `private val`getter◦

```
fun isFieldAccessible(property: KProperty1<*, *>): Boolean {
    return property.javaGetter?.modifiers?.let { !Modifier.isPrivate(it) } ?: false
}

val example = Example(field1 = "abc", field2 = 1, baseField = "someText")

example::class.memberProperties.filter { isFieldAccessible(it) }.forEach { member ->
    println("${member.name} -> ${member.get(example)}")
}
```

```
example::class.memberProperties.forEach { member ->
    member.isAccessible = true
    println("${member.name} -> ${member.get(example)}")
}
```

```
class TestClass {
    val readOnlyProperty: String
        get() = "Read only!"

    var readWriteString = "asd"
    var readWriteInt = 23

    var readWriteBackedStringProperty: String = ""
        get() = field + '5'
        set(value) { field = value + '5' }

    var readWriteBackedIntProperty: Int = 0
        get() = field + 1
        set(value) { field = value - 1 }

    var delegatedProperty: Int by TestDelegate()

    private var privateProperty = "This should be private"

    private class TestDelegate {
        private var backingField = 3

        operator fun getValue(thisRef: Any?, prop: KProperty<*>): Int {
            return backingField
        }

        operator fun setValue(thisRef: Any?, prop: KProperty<*>, value: Int) {
            backingField += value
        }
    }
}
```

◦ ◦

```
val instance = TestClass()
TestClass::class.memberProperties
        .filter{ prop.visibility == KVisibility.PUBLIC }
        .filterIsInstance<KMutableProperty<*>>()
        .forEach { prop ->
            System.out.println("${prop.name} -> ${prop.get(instance)")
        }
```

`String"Our Value"`◦ KotlinJava VMType Erasure`List<String>`Properties`List<Any>`◦ ◦

```
val instance = TestClass()
TestClass::class.memberProperties
        .filter{ prop.visibility == KVisibility.PUBLIC }
        // We only want strings
        .filter{ it.returnType.isSubtypeOf(String::class.starProjectedType) }
        .filterIsInstance<KMutableProperty<*>>()
        .forEach { prop ->
            // Instead of printing the property we set it to some value
            prop.setter.call(instance, "Our Value")
        }
```

https://riptutorial.com/zh-CN/kotlin/topic/2402/

15:

Kotlin4。

。

。

。

。

- `<visibility modifier> val/var <variable name> = <value>`

Examples

```
public val name = "Avijit"

private val name = "Avijit"

protected val name = "Avijit"

internal val name = "Avijit"
```

https://riptutorial.com/zh-CN/kotlin/topic/10019/

16: KotlinRecyclerView

KotlinRecyclerView。

Examples

Kotlinactivity_main.xmlRecyclerView。

```
class MainActivity : AppCompatActivity(){

        lateinit var mRecyclerView : RecyclerView
        val mAdapter : RecyclerAdapter = RecyclerAdapter()

        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_main)
            val toolbar = findViewById(R.id.toolbar) as Toolbar
            setSupportActionBar(toolbar)

            mRecyclerView = findViewById(R.id.recycler_view) as RecyclerView

            mRecyclerView.setHasFixedSize(true)
            mRecyclerView.layoutManager = LinearLayoutManager(this)
            mAdapter.RecyclerAdapter(getList(), this)
            mRecyclerView.adapter = mAdapter
        }

        private fun getList(): ArrayList<String> {
            var list : ArrayList<String> = ArrayList()
            for  (i in 1..10) { // equivalent of 1 <= i && i <= 10
                println(i)
                list.add("$i")
            }
            return list
        }
    }
```

Recyclermain_item.xml

```
class RecyclerAdapter : RecyclerView.Adapter<RecyclerAdapter.ViewHolder>() {

    var mItems: ArrayList<String>  = ArrayList()
    lateinit var mClick : OnClick

    fun RecyclerAdapter(item : ArrayList<String>, mClick : OnClick){
        this.mItems = item
        this.mClick = mClick;
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val item = mItems[position]
        holder.bind(item, mClick, position)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
```

```
        val layoutInflater = LayoutInflater.from(parent.context)
        return ViewHolder(layoutInflater.inflate(R.layout.main_item, parent, false))
    }

    override fun getItemCount(): Int {
        return mItems.size
    }

    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val card = view.findViewById(R.id.card) as TextView
        fun bind(str: String, mClick: OnClick, position: Int){
            card.text = str
            card.setOnClickListener { view ->
                mClick.onClickListner(position)
            }
        }
    }
}
```

KotlinRecyclerView https://riptutorial.com/zh-CN/kotlin/topic/10143/kotlinrecyclerview

17: Lambdas

-
- {parameterNameParameterTypeotherParameterNameOtherParameterType - > anExpression}
-
- val additionIntInt - > Int = {ab - > a + b}
- `it`
- val squareInt - > Int = {it * it}
-
- - > ResultType
- InputType - > ResultType
- InputType1InputType2 - > ResultType

。

```
data class User(val fistName: String, val lastName: String) {
    fun username(userNameGenerator: (String, String) -> String) =
        userNameGenerator(firstName, secondName)
}
```

lambdalambda

```
val user = User("foo", "bar")
println(user.userName { firstName, secondName ->
     "${firstName.toUppercase}"_"${secondName.toUppercase}"
 }) // prints FOO_BAR
```

lambda

```
//valid:
val addition: (Int, Int) = { a, b -> a + b }
//valid:
val addition = { a: Int, b: Int -> a + b }
//error (type inference failure):
val addition = { a, b -> a + b }
```

lambda`it`。

```
listOf(1,2,3).map { it * 2 } // [2,4,6]
```

Examples

Lambda

```
val allowedUsers = users.filter { it.age > MINIMUM_AGE }
```

Lambda

```
val isOfAllowedAge = { user: User -> user.age > MINIMUM_AGE }
val allowedUsers = users.filter(isOfAllowedAge)
```

Lambda

```
object Benchmark {
    fun realtime(body: () -> Unit): Duration {
        val start = Instant.now()
        try {
            body()
        } finally {
            val end = Instant.now()
            return Duration.between(start, end)
        }
    }
}
```

```
val time = Benchmark.realtime({
    // some long-running code goes here ...
})
println("Executed the code in $time")
```

Lambdas https://riptutorial.com/zh-CN/kotlin/topic/5878/lambdas

18:

Kotlin。 。 。

Examples

```
val foo : Int by lazy { 1 + 1 }
println(foo)
```

2 。

```
var foo : Int by Delegates.observable("1") { property, oldValue, newValue ->
    println("${property.name} was changed from $oldValue to $newValue")
}
foo = 2
```

```
foo was changed from 1 to 2foo was changed from 1 to 2
```

```
val map = mapOf("foo" to 1)
val foo : String by map
println(foo)
```

```
1
```

```
class MyDelegate {
    operator fun getValue(owner: Any?, property: KProperty<*>): String {
        return "Delegated value"
    }
}

val foo : String by MyDelegate()
println(foo)
```

```
Delegated value
```

KotlinNull Type`WeakReference<T>` 。

`WeakReference`。

```
class MyMemoryExpensiveClass {
    companion object {
        var reference: WeakReference<MyMemoryExpensiveClass>? = null

        fun doWithReference(block: (MyMemoryExpensiveClass) -> Unit) {
            reference?.let {
                it.get()?.let(block)
            }
        }
    }

    init {
```

```
        reference = WeakReference(this)
    }
}
```

WeakReference。

```
class WeakReferenceDelegate<T>(initialValue: T? = null) : ReadWriteProperty<Any, T?> {
    var reference = WeakReference(initialValue)
        private set

    override fun getValue(thisRef: Any, property: KProperty<*>): T? = reference.get()

    override fun setValue(thisRef: Any, property: KProperty<*>, value: T?) {
        reference = WeakReference(value)
    }
}
```

`WeakReference`

```
class MyMemoryExpensiveClass {
    companion object {
        var reference: MyMemoryExpensiveClass? by
WeakReferenceDelegate<MyMemoryExpensiveClass>()

        fun doWithReference(block: (MyMemoryExpensiveClass) -> Unit) {
            reference?.let(block)
        }
    }

    init {
        reference = this
    }
}
```

https://riptutorial.com/zh-CN/kotlin/topic/10571/

19:

Examples

String`string[index]string[index]`。

```
val str = "Hello, World!"
println(str[1]) // Prints e
```

for。

```
for (c in str) {
    println(c)
}
```

Kotlin

-
-

。 。 `\t \b \n \r \' \" \\\$` 。 Unicode `\uFF00` 。

```
val s = "Hello, world!\n"
```

`"""`

```
val text = """
    for (c in "foo")
        print(c)
"""
```

trimMargin。

```
val text = """
    |Tell me and I forget.
    |Teach me and I remember.
    |Involve me and I learn.
    |(Benjamin Franklin)
    """.trimMargin()
```

`|` trimMargin;`trimMargin(">")` 。

。 。 `$`

```
val i = 10
val s = "i = $i" // evaluates to "i = 10"
```

```
val s = "abc"
```

```
val str = "$s.length is ${s.length}" // evaluates to "abc.length is 3"
```

```
val str = "\$foo" // evaluates to "$foo"
```

。 。

```
val price = """
${'$'}9.99
"""
```

Kotlin==。

```
val str1 = "Hello, World!"
val str2 = "Hello," + " World!"
println(str1 == str2) // Prints true
```

===。

```
val str1 = """
    |Hello, World!
    """.trimMargin()

val str2 = """
    #Hello, World!
    """.trimMargin("#")

val str3 = str1

println(str1 == str2) // Prints true
println(str1 === str2) // Prints false
println(str1 === str3) // Prints true
```

https://riptutorial.com/zh-CN/kotlin/topic/8285/

20:

Examples

DTOPOJO / POCO

kotlin◦ `data`

```
data class User(var firstname: String, var lastname: String, var age: Int)
```

`User`

- `val` S
- `equals()`
- `hashcode()`
- `toString()`
- `copy()`
- `componentN()` N

```
data class User(var firstname: String = "Joe", var lastname: String = "Bloggs", var age: Int =
20)
```

◦

```
val list = listOf(1,2,3,4,5,6)

//filter out even numbers

val even = list.filter { it % 2 == 0 }

println(even) //returns [2,4]
```

◦ ◦ ◦

```
class MyTable private constructor(table: Table<Int, Int, Int>) : Table<Int, Int, Int> by table
{

    constructor() : this(TreeBasedTable.create()) // or a different type of table if desired

}
```

`MyTable` `MyTable()` ◦ `MyTableTable<Int, Int, Int>` ◦ ◦

SO ◦

KotlinSerializableserialVersionUid

Kotlin`serialVersionUID` ◦

`private const val MySpecialCase`

```
class MySpecialCase : Serializable {
    companion object {
        private const val serialVersionUID: Long = 123
    }
}
```

getter / setter...

```
class MySpecialCase : Serializable {
    companion object {
        private val serialVersionUID: Long = 123
    }
}
```

`getSerialVersionUID`getter`getSerialVersionUID`。

```
class MySpecialCase : Serializable {
    companion object {
        @JvmStatic private val serialVersionUID: Long = 123
    }
}
```

`MySpecialCase`getter`getSerialVersionUID`。

`serialVersionUID`Serializable。

Kotlin

KotlinJava

```
fun doSomething() {
   someOtherAction()
   return this
}
```

```
fun <T: Any> T.fluently(func: ()->Unit): T {
    func()
    return this
}
```

```
fun doSomething() {
   return fluently { someOtherAction() }
}
```

let

`let` Kotlin。

```
val str = "foo"
str.let {
    println(it) // it
}
```

"foo"Unit ◦

*letalsolet◦ also*reutrn *Unit ◦*

null null letalso

```
val str: String? = someFun()
str?.let {
    println(it)
}
```

strnulllet◦ null ? ◦

apply

apply

thisthis◦

kdocapply◦ apply thisapply◦ ◦

```
File(dir).apply { mkdirs() }
```

```
fun makeDir(String path): File {
    val result = new File(path)
    result.mkdirs()
    return result
}
```

https://riptutorial.com/zh-CN/kotlin/topic/2273/

21:

- fun TypeName.extensionNameparams...{/ * body * /} //
- fun <TAny> TypeNameWithGenerics <T> .extensionNameparams...{/ * body * /} //
- myObj.extensionNameargs...//

◦ reference-type;◦ ◦

Examples

◦

```
fun IntArray.addTo(dest: IntArray) {
    for (i in 0 .. size - 1) {
        dest[i] += this[i]
    }
}
```

IntArrayIntArray ◦ this ◦

```
val myArray = intArrayOf(1, 2, 3)
intArrayOf(4, 5, 6).addTo(myArray)
```

◦ ◦

```
open class Super

class Sub : Super()

fun Super.myExtension() = "Defined for Super"

fun Sub.myExtension() = "Defined for Sub"

fun callMyExtension(myVar: Super) {
    println(myVar.myExtension())
}

callMyExtension(Sub())
```

"Defined for Super" myVarSuper ◦

IntLong

```
fun Long.humanReadable(): String {
    if (this <= 0) return "0"
    val units = arrayOf("B", "KB", "MB", "GB", "TB", "EB")
    val digitGroups = (Math.log10(this.toDouble())/Math.log10(1024.0)).toInt();
    return DecimalFormat("#,##0.#").format(this/Math.pow(1024.0, digitGroups.toDouble())) + "
" + units[digitGroups];
}
```

```
fun Int.humanReadable(): String {
    return this.toLong().humanReadable()
}
```

```
println(1999549L.humanReadable())
println(someInt.humanReadable())
```

Java 7+ Path

API◦ `exist` `notExists`deleteRecursivelyJava 7+ `Path`

```
fun Path.exists(): Boolean = Files.exists(this)
fun Path.notExists(): Boolean = !this.exists()
fun Path.deleteRecursively(): Boolean = this.toFile().deleteRecursively()
```

```
val dir = Paths.get(dirName)
if (dir.exists()) dir.deleteRecursively()
```

Kotlin

```
val x: Path = Paths.get("dirName").apply {
    if (Files.notExists(this)) throw IllegalStateException("The important file does not
exist")
}
```

apply◦ ◦

```
infix inline fun <T> T.verifiedBy(verifyWith: (T) -> Unit): T {
    verifyWith(this)
    return this
}

infix inline fun <T: Any> T.verifiedWith(verifyWith: T.() -> Unit): T {
    this.verifyWith()
    return this
}
```

```
val x: Path = Paths.get("dirName") verifiedWith {
    if (Files.notExists(this)) throw IllegalStateException("The important file does not
exist")
}
```

lambda◦

`TverifiedByT: Any?◦` `verifiedWith◦`

Java 8 TemporalISO

```
fun Temporal.toIsoString(): String = DateTimeFormatter.ISO_INSTANT.format(this)
```

```
val dateAsString = someInstant.toIsoString()
```

Companion

- SomethingfromString

```
class Something {
    companion object {}
}

class SomethingElse {
}

fun Something.Companion.fromString(s: String): Something = ...

fun SomethingElse.fromString(s: String): SomethingElse = ...

fun main(args: Array<String>) {
    Something.fromString("") //valid as extension function declared upon the
                             //companion object

    SomethingElse().fromString("") //valid, function invoked on instance not
                                   //statically

    SomethingElse.fromString("") //invalid
}
```

◦ this KotlinKT-9686KT-13053 ◦ ◦

color ◦ colorCache thislazy

```
class KColor(val value: Int)

private val colorCache = mutableMapOf<KColor, Color>()

val KColor.color: Color
    get() = colorCache.getOrPut(this) { Color(value, true) }
```

View◦

Anko

```
inline fun <reified T : View> View.find(id: Int): T = findViewById(id) as T
inline fun <reified T : View> Activity.find(id: Int): T = findViewById(id) as T
inline fun <reified T : View> Fragment.find(id: Int): T = view?.findViewById(id) as T
inline fun <reified T : View> RecyclerView.ViewHolder.find(id: Int): T =
itemView?.findViewById(id) as T

inline fun <reified T : View> View.findOptional(id: Int): T? = findViewById(id) as? T
inline fun <reified T : View> Activity.findOptional(id: Int): T? = findViewById(id) as? T
inline fun <reified T : View> Fragment.findOptional(id: Int): T? = view?.findViewById(id) as?
T
inline fun <reified T : View> RecyclerView.ViewHolder.findOptional(id: Int): T? =
itemView?.findViewById(id) as? T
```

```
val yourButton by lazy { find<Button>(R.id.yourButtonId) }
val yourText by lazy { find<TextView>(R.id.yourTextId) }
val yourEdittextOptional by lazy { findOptional<EditText>(R.id.yourOptionEdittextId) }
```

https://riptutorial.com/zh-CN/kotlin/topic/613/

22:

Kotlin

Examples

Kotlin。

```
interface MyInterface {
    fun bar()
}
```

```
class Child : MyInterface {
   override fun bar() {
       print("bar() was called")
   }
}
```

Kotlin

```
interface MyInterface {
    fun withImplementation() {
      print("withImplementation() was called")
    }
}
```

```
class MyClass: MyInterface {
    // No need to reimplement here
}
val instance = MyClass()
instance.withImplementation()
```

gettersetter

```
interface MyInterface2 {
    val helloWorld
        get() = "Hello World!"
}
```

```
interface MyInterface3 {
    // this property won't compile!
    var helloWorld: Int
        get() = field
        set(value) { field = value }
}
```

```
interface A {
    fun notImplemented()
    fun implementedOnlyInA() { print("only A") }
```

```
    fun implementedInBoth() { print("both, A") }
    fun implementedInOne() { print("implemented in A") }
}

interface B {
    fun implementedInBoth() { print("both, B") }
    fun implementedInOne() // only defined
}

class MyClass: A, B {
    override fun notImplemented() { print("Normal implementation") }

    // implementedOnlyInA() can by normally used in instances

    // class needs to define how to use interface functions
    override fun implementedInBoth() {
        super<B>.implementedInBoth()
        super<A>.implementedInBoth()
    }

    // even if there's only one implementation, there multiple definitions
    override fun implementedInOne() {
        super<A>.implementedInOne()
        print("implementedInOne class implementation")
    }
}
```

◦ ◦

```
interface MyInterface {
    val property: Int // abstract

    val propertyWithImplementation: String
        get() = "foo"

    fun foo() {
        print(property)
    }
}

class Child : MyInterface {
    override val property: Int = 29
}
```

◦ ◦ ◦

```
interface FirstTrait {
    fun foo() { print("first") }
    fun bar()
}

interface SecondTrait {
    fun foo() { print("second") }
    fun bar() { print("bar") }
}

class ClassWithConflict : FirstTrait, SecondTrait {
    override fun foo() {
```

```
        super<FirstTrait>.foo()  // delegate to the default implementation of FirstTrait
        super<SecondTrait>.foo() // delegate to the default implementation of SecondTrait
    }

    // function bar() only has a default implementation in one interface and therefore is ok.
}
```

```
interface MyInterface {
    fun funcOne() {
        //optional body
        print("Function with default implementation")
    }
}
```

super。

```
super.funcOne()
```

https://riptutorial.com/zh-CN/kotlin/topic/900/

23:

Examples

Kotlin`Array<T>` ◦

`emptyArray<T>()`

```
val empty = emptyArray<String>()
```

```
var strings = Array<String>(size = 5, init = { index -> "Item #$index" })
print(Arrays.toString(a)) // prints "[Item #0, Item #1, Item #2, Item #3, Item #4]"
print(a.size) // prints 5
```

`get(index: Int): Tset(index: Int, value: T)`

```
strings.set(2, "ChangedItem")
print(strings.get(2)) // prints "ChangedItem"

// You can use subscription as well:
strings[2] = "ChangedItem"
print(strings[2]) // prints "ChangedItem"
```

`Array<T>`◦

Kotlin		JVM
BooleanArray	booleanArrayOf(true, false)	boolean[]
ByteArray	byteArrayOf(1, 2, 3)	byte[]
CharArray	charArrayOf('a', 'b', 'c')	char[]
DoubleArray	doubleArrayOf(1.2, 5.0)	double[]
FloatArray	floatArrayOf(1.2, 5.0)	float[]
IntArray	intArrayOf(1, 2, 3)	int[]
LongArray	longArrayOf(1, 2, 3)	long[]
ShortArray	shortArrayOf(1, 2, 3)	short[]

`average()Byte Int Long Short Double FloatDouble`

```
val doubles = doubleArrayOf(1.5, 3.0)
print(doubles.average()) // prints 2.25

val ints = intArrayOf(1, 4)
println(ints.average()) // prints 2.5
```

`component1()` `component2()` ... `component5()`

index`getOrNull(index: Int)`null

`first()` `last()`

`toHashSet()``HashSet<T>`

`sortedArray()` `sortedArrayDescending()`current

`sort()` `sortDescending`

`min()` `max()`

Java`in`。

```
val asc = Array(5, { i -> (i * i).toString() })
for(s : String in asc){
    println(s);
}
```

for。

```
val asc = Array(5, { i -> (i * i).toString() })
for(s in asc){
    println(s);
}
```

```
val a = arrayOf(1, 2, 3) // creates an Array<Int> of size 3 containing [1, 2, 3].
```

```
val a = Array(3) { i -> i * 2 } // creates an Array<Int> of size 3 containing [0, 2, 4]
```

```
val a = arrayOfNulls<Int>(3) // creates an Array<Int?> of [null, null, null]
```

。 。

https://riptutorial.com/zh-CN/kotlin/topic/5722/

24:

Examples

NullableNon Nullable

`String ◦ ? String?`

```
var string        : String = "Hello World!"
var nullableString: String? = null

string = nullableString    // Compiler error: Can't assign nullable to non-nullable type.
nullableString = string    // This will work however!
```

◦

`?.` nullnull;

```
val string: String? = "Hello World!"
print(string.length)   // Compile error: Can't directly access property of nullable type.
print(string?.length)  // Will print the string's length, or "null" if the string is null.
```

null

nullKotlin`apply`

```
obj?.apply {
    foo()
    bar()
}
```

`objfoobar thisapplythis obj`null◦

`let`apply

```
nullable?.let { notnull ->
    notnull.foo()
    notnull.bar()
}
```

`notnull`lambda`it ◦`

null

```
var string: String? = "Hello!"
print(string.length)  // Compile error
if(string != null) {
    // The compiler now knows that string can't be null
```

```
    print(string.length)  // It works now!
}
```

null。

。

Iterable

`Collection<T?>Collections<T>。 filterNotNull。`

```
val a: List<Int?> = listOf(1, 2, 3, null)
val b: List<Int> = a.filterNotNull()
```

Null Coalescing / Elvis

if-else。 elvis`?:` Kotlin。

```
val value: String = data?.first() ?: "Nothing here."
```

`data?.first()datanull"Nothing here"data?.first()data?.first()。`

。

```
val value: String = data?.second()
    ?: throw IllegalArgumentException("Value can't be null!")
```

NullPointerExceptions`data!!.second()!!`

`!!`nullabilitynull。 `nullKotlinNullPointerException。`

```
val message: String? = null
println(message!!) //KotlinNullPointerException thrown, app crashes
```

:)

Kotlin`null reference。 a` “anullx ”

```
var a: String? = "Nullable String Value"
```

anull。 `a。 if...else。`

```
val b: Int = if (a != null) a.length else -1
```

`Elvis ?:if...else`Elvis

```
val b = a?.length ?: -1
```

`?: a?.length` nullelvis `-1` ◦ ◦

https://riptutorial.com/zh-CN/kotlin/topic/2080/

25:

Javaswitch when◦ break◦ bahaviors

```
when (x) {
    "foo", "bar" -> println("either foo or bar")
    else -> println("didn't match anything")
}
```

Examples

if

```
val str = "Hello!"
if (str.length == 0) {
    print("The string is empty!")
} else if (str.length > 5) {
    print("The string is short!")
} else {
    print("The string is long!")
}
```

else-branchesif◦

if

if

```
val str = if (condition) "Condition met!" else "Condition not met!"
```

elseif◦

else if◦

```
val str = if (condition1){
    "Condition1 met!"
   } else if (condition2) {
    "Condition2 met!"
   } else {
    "Conditions not met!"
   }
```

Kotlin val str: String =◦

whenif-else-if

whenelse-if-branchesif

```
when {
    str.length == 0 -> print("The string is empty!")
    str.length > 5  -> print("The string is short!")
    else            -> print("The string is long!")
}
```

if-else-if

```
if (str.length == 0) {
    print("The string is empty!")
} else if (str.length > 5) {
    print("The string is short!")
} else {
    print("The string is long!")
}
```

ifelse-branch。

```
when {
    condition -> {
        doSomething()
        doSomeMore()
    }
    else -> doSomethingElse()
}
```

when

`when` -statement。 `==equals。 。`

```
when (x) {
    "English" -> print("How are you?")
    "German" -> print("Wie geht es dir?")
    else -> print("I don't know that language yet :(")
}
```

when

```
val names = listOf("John", "Sarah", "Tim", "Maggie")
when (x) {
    in names -> print("I know that name!")
    !in 1..10 -> print("Argument was not in the range from 1 to 10")
    is String -> print(x.length) // Due to smart casting, you can use String-functions here
}
```

```
val greeting = when (x) {
    "English" -> "How are you?"
    "German" -> "Wie geht es dir?"
    else -> "I don't know that language yet :("
}
print(greeting)
```

whenelse。

whenenum

```
enum class Day {
    Sunday,
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday
}

fun doOnDay(day: Day) {
    when(day) {
        Day.Sunday ->     // Do something
        Day.Monday, Day.Tuesday ->     // Do other thing
        Day.Wednesday ->  // ...
        Day.Thursday ->   // ...
        Day.Friday ->     // ...
        Day.Saturday ->   // ...
    }
}
```

Tuedsay MondayTuedsay enum◦

◦ else

```
fun doOnDay(day: Day) {
    when(day) {
        Day.Monday ->     // Work
        Day.Tuesday ->    // Work hard
        Day.Wednesday ->  // ...
        Day.Thursday ->   //
        Day.Friday ->     //
        else ->           // Party on weekend
    }
}
```

if-then-elsewhenenum◦

kotlin enum

https://riptutorial.com/zh-CN/kotlin/topic/2685/

26:

JavaKotlin。 EnumClass

```
EnumClass.valueOf(value: String): EnumClass
EnumClass.values(): Array<EnumClass>
```

valueOf()IllegalArgumentException 。

```
val name: String
val ordinal: Int
```

Comparable。

Examples

```
enum class Color(val rgb: Int) {
    RED(0xFF0000),
    GREEN(0x00FF00),
    BLUE(0x0000FF)
}
```

。 ; 。

abstract 。

```
enum class Color {
    RED {
        override val rgb: Int = 0xFF0000
    },
    GREEN {
        override val rgb: Int = 0x00FF00
    },
    BLUE {
        override val rgb: Int = 0x0000FF
    }

    ;

    abstract val rgb: Int

    fun colorString() = "#%06X".format(0xFFFFFF and rgb)
}
```

```
enum class Color {
  RED, GREEN, BLUE
}
```

。 。

```
enum class Planet(var population: Int = 0) {
    EARTH(7 * 100000000),
    MARS();

    override fun toString() = "$name[population=$population]"
}

 println(Planet.MARS) // MARS[population=0]
 Planet.MARS.population = 3
 println(Planet.MARS) // MARS[population=3]
```

https://riptutorial.com/zh-CN/kotlin/topic/2286/

27:

Examples

“”。 when“” - 。

```
import kotlin.text.regex

var string = /* some string */

val regex1 = Regex( /* pattern */ )
val regex2 = Regex( /* pattern */ )
/* etc */

when {
    regex1.matches(string) -> /* do stuff */
    regex2.matches(string) -> /* do stuff */
    /* etc */
}
```

“”。 whenwhen。

```
import kotlin.text.regex

var string = /* some string */

when {
    Regex( /* pattern */ ).matches(string) -> /* do stuff */
    Regex( /* pattern */ ).matches(string) -> /* do stuff */
    /* etc */
}
```

---

whenwhen“argument-ful”。 whenwhenEntrywhen。 “immutable locals”“anonymous temporaries”。

```
import kotlin.text.regex

var string = /* some string */

when (RegexWhenArgument(string)) {
    Regex( /* pattern */ ) -> /* do stuff */
    Regex( /* pattern */ ) -> /* do stuff */
    /* etc */
}
```

when

```
class RegexWhenArgument (val whenArgument: CharSequence) {
    operator fun equals(whenEntry: Regex) = whenEntry.matches(whenArgument)
    override operator fun equals(whenEntry: Any?) = (whenArgument == whenEntry)
}
```

RegexRegexnull。

RegEx

KotlinRegex(pattern: String)regexfind(..)replace(..)。

Regexinputcdtrue

```
val regex = Regex(pattern = "c|d")
val matched = regex.containsMatchIn(input = "abc")    // matched: true
```

Regexpatterninput。 。 containsMatchIn(..)。

---

find(..)matchEntire(..)MatchResult?。 ? MatchResultKotlin null。

find(..)nullKotlinRegexnull

```
val matchResult =
    Regex("c|d").find("efg")           // matchResult: null
val a = matchResult?.value             // a: null
val b = matchResult?.value.orEmpty()   // b: ""
a?.toUpperCase()                       // Still needs question mark. => null
b.toUpperCase()                        // Accesses the function directly. => ""
```

orEmpty() bnull?b。

KotlinJava!!

```
a!!.toUpperCase()                      // => KotlinNullPointerException
```

KotlinJavaJava。

```
"""\d{3}-\d{3}-\d{4}""" // raw Kotlin string
"\\d{3}-\\d{3}-\\d{4}"  // standard Java string
```

findCharSequencestartIndexIntMatchResult

inputRegexpattern。 Matchresult?startIndexinputnull 。 MatchResult?MatchResult?value。 startIndex0
。

```
val phoneNumber :String? = Regex(pattern = """\d{3}-\d{3}-\d{4}""")
```

```
    .find(input = "phone: 123-456-7890, e..")?.value // phoneNumber: 123-456-7890
```

inputphoneNumbernull ◦

findAllinputCharSequencestartIndexInt Sequence

patterninput◦

```
val matchedResults = Regex(pattern = """\d+""").findAll(input = "ab12cd34ef")
val result = StringBuilder()
for (matchedText in matchedResults) {
    result.append(matchedText.value + " ")
}

println(result) // => 12 34
```

matchedResultsMatchResult◦ inputfindAll(..)◦

matchEntireCharSequenceMatchResult

inputpatterninput◦ null ◦

```
val a = Regex("""\d+""").matchEntire("100")?.value             // a: 100
val b = Regex("""\d+""").matchEntire("100 dollars")?.value     // b: null
```

matchesCharSequenceBoolean

true◦ ◦

```
val regex = Regex(pattern = """\d+""")
regex.matches(input = "50")             // => true
regex.matches(input = "50 dollars")     // => false
```

containsMatchIninputCharSequenceBoolean

true◦ ◦

```
Regex("""\d+""").containsMatchIn("50 dollars")       // => true
Regex("""\d+""").containsMatchIn("Fifty dollars")    // => false
```

splitCharSequencelimitIntList

◦

```
val a = Regex("""\d+""").split("ab12cd34ef")     // a: [ab, cd, ef]
val b = Regex("""\d+""").split("This is a test") // b: [This is a test]
```

◦ input◦ ◦

replaceCharSequencereplacementString String String

inputpattern◦

x

```
val result = Regex("""\d+""").replace("ab12cd34ef", "x") // result: abxcdxef
```

https://riptutorial.com/zh-CN/kotlin/topic/8364/

28:

◦ List◦

◦

- class *ClassName* < ***TypeName*** >
- class *ClassName* <*>
- *ClassName* <in ***UpperBound*** >
- *ClassName* <out ***LowerBound*** >
- class *Name* < ***TypeName*** ***UpperBound*** >

UPPERBOUND

Kotlin Generics`TAny?` ◦

```
class Consumer<T>
```

`TT: Any?` ◦ `T: Any` ◦

```
class Consumer<T: Any>
```

Examples

-

- ◦

```
  class Consumer<in T> { fun consume(t: T) { ... } }

  fun charSequencesConsumer() : Consumer<CharSequence>() = ...

  val stringConsumer : Consumer<String> = charSequenceConsumer() // OK since in-projection
  val anyConsumer : Consumer<Any> = charSequenceConsumer() // Error, Any cannot be passed

  val outConsumer : Consumer<out CharSequence> = ... // Error, T is `in`-parameter
```

- `List<out T>` `TComparator<in T>` `T`◦

Java

```
val takeList : MutableList<out SomeType> = ... // Java: List<? extends SomeType>

val takenValue : SomeType = takeList[0] // OK, since upper bound is SomeType

takeList.add(takenValue) // Error, lower bound for generic is not specified
```

```
val putList : MutableList<in SomeType> = ... // Java: List<? super SomeType>

val valueToPut : SomeType = ...
putList.add(valueToPut) // OK, since lower bound is SomeType

putList[0] // This expression has type Any, since no upper bound is specified
```

```
val starList : MutableList<*> = ... // Java: List<?>

starList[0] // This expression has type Any, since no upper bound is specified
starList.add(someValue) // Error, lower bound for generic is not specified
```

- JavaKotlinVariant Generics。

https://riptutorial.com/zh-CN/kotlin/topic/1147/

29:

Examples

。

```
annotation class Strippable
```

```
    @Target(AnnotationTarget.CLASS, AnnotationTarget.FUNCTION,
AnnotationTarget.VALUE_PARAMETER, AnnotationTarget.EXPRESSION)
    annotation class Strippable
```

```
annotation class Strippable(val importanceValue: Int)
```

- JavaIntLong;
-
- Foo :: class
-
-
-

- `@Target`
- `@Retention`。
- `@Repeatable`。
- `@MustBeDocumented`APIAPI。

```
@Target(AnnotationTarget.CLASS, AnnotationTarget.FUNCTION,
        AnnotationTarget.VALUE_PARAMETER, AnnotationTarget.EXPRESSION)
@Retention(AnnotationRetention.SOURCE)
@MustBeDocumented
annotation class Fancy
```

https://riptutorial.com/zh-CN/kotlin/topic/4074/

30:

Kotlin。。

Examples

```
interface Foo {
    fun example()
}

class Bar {
    fun example() {
        println("Hello, world!")
    }
}

class Baz(b : Bar) : Foo by b

Baz(Bar()).example()
```

```
Hello, world!
```

https://riptutorial.com/zh-CN/kotlin/topic/10575/

31: kotlin

Kotlin

Examples

kotlin.logging

```
class FooWithLogging {
  companion object: KLogging()

  fun bar() {
    logger.info { "hello $name" }
  }

  fun logException(e: Exception) {
    logger.error(e) { "Error occured" }
  }
}
```

kotlin.logging

kotlin https://riptutorial.com/zh-CN/kotlin/topic/3258/kotlin

32:

Kotlin。 plain int。

Examples

x

```
repeat(10) { i ->
    println("This line will be printed 10 times")
    println("We are on the ${i + 1}. loop iteration")
}
```

iterables

foriterable

```
val list = listOf("Hello", "World", "!")
for(str in list) {
    print(str)
}
```

Kotlin

```
for(i in 0..9) {
    print(i)
}
```

```
for((index, element) in iterable.withIndex()) {
    print("$element at index $index")
}
```

forEach

```
iterable.forEach {
    print(it.toString())
}
```

`it`Lambda

whiledo-while

```
while(condition) {
    doSomething()
}

do {
    doSomething()
```

```
} while (condition)
```

do-while。

。

```
while(true) {
    if(condition1) {
        continue // Will immediately start the next iteration, without executing the rest of
the loop body
    }
    if(condition2) {
        break // Will exit the loop completely
    }
}
```

breakcontinue

```
outer@ for(i in 0..10) {
    inner@ for(j in 0..10) {
        break       // Will break the inner loop
        break@inner // Will break the inner loop
        break@outer // Will break the outer loop
    }
}
```

forEach。

kotlin

```
//iterates over a map, getting the key and value at once

var map = hashMapOf(1 to "foo", 2 to "bar", 3 to "baz")

for ((key, value) in map) {
    println("Map[$key] = $value")
}
```

Kotlin。

```
fun factorial(n: Long): Long = if (n == 0) 1 else n * factorial(n - 1)

println(factorial(10)) // 3628800
```

factorial。

Kotlin。

map。

```
val numbers = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
```

```
val numberStrings = numbers.map { "Number $it" }
```

◦ ◦ filter◦

```
val numbers = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
val numberStrings = numbers.filter { it % 2 == 0 }.map { "Number $it" }
```

https://riptutorial.com/zh-CN/kotlin/topic/2727/

33:

。

3

1. 。
2. lambda。
3. 。

Kotlin

Kotlin

-
-
- Ktor
-

Examples

lambda。 `JFrame`

```
import javax.swing.*

fun JFrame.menuBar(init: JMenuBar.() -> Unit) {
    val menuBar = JMenuBar()
    menuBar.init()
    setJMenuBar(menuBar)
}

fun JMenuBar.menu(caption: String, init: JMenu.() -> Unit) {
    val menu = JMenu(caption)
    menu.init()
    add(menu)
}

fun JMenu.menuItem(caption: String, init: JMenuItem.() -> Unit) {
    val menuItem = JMenuItem(caption)
    menuItem.init()
    add(menuItem)
}
```

```
class MyFrame : JFrame() {
    init {
        menuBar {
            menu("Menu1") {
                menuItem("Item1") {
                    // Initialize MenuItem with some Action
                }
                menuItem("Item2") {}
```

```
            }
            menu("Menu2") {
                menuItem("Item3") {}
                menuItem("Item4") {}
            }
        }
    }
}
```

https://riptutorial.com/zh-CN/kotlin/topic/6010/

34:

◦

Apples OrangesPears ◦ ◦

Fruit getFruit()

Fruit ◦

- {}
- class {Derived Class}{Base Class}{Init Arguments}
- {}
- {DC-Object}{Base Class} == true

	Base Class
	Base Class
	Derived Class
DC-	“”

Examples

'open'

Kotlin **final**◦

open◦

```
open class Thing {
    // I can now be extended!
}
```

open◦

```
open class BaseClass {
    val x = 10
}
```

```
class DerivedClass: BaseClass() {
    fun foo() {
        println("x is equal to " + x)
    }
```

```
}
```

```
fun main(args: Array<String>) {
    val derivedClass = DerivedClass()
    derivedClass.foo() // prints: 'x is equal to 10'
}
```

```
open class Person {
    fun jump() {
        println("Jumping...")
    }
}
```

```
class Ninja: Person() {
    fun sneak() {
        println("Sneaking around...")
    }
}
```

Person

```
fun main(args: Array<String>) {
    val ninja = Ninja()
    ninja.jump()  // prints: 'Jumping...'
    ninja.sneak() // prints: 'Sneaking around...'
}
```

```
abstract class Car {
    abstract val name: String;
    open var speed: Int = 0;
}

class BrokenCar(override val name: String) : Car() {
    override var speed: Int
        get() = 0
        set(value) {
            throw UnsupportedOperationException("The car is bloken")
        }
}

fun main(args: Array<String>) {
    val car: Car = BrokenCar("Lada")
    car.speed = 10
}
```

```
interface Ship {
    fun sail()
    fun sink()
}

object Titanic : Ship {

    var canSail = true
```

```
    override fun sail() {
        sink()
    }

    override fun sink() {
        canSail = false
    }
}
```

https://riptutorial.com/zh-CN/kotlin/topic/5622/

35:

rangeTo..inin。

Examples

IntRangeLongRangeCharRange。 Javafor-loop

```
for (i in 1..4) print(i) // prints "1234"
for (i in 4..1) print(i) // prints nothing
```

downTo

。 downTo

```
for (i in 4 downTo 1) print(i) // prints "4321"
```

step

1step

```
for (i in 1..4 step 2) print(i) // prints "13"
for (i in 4 downTo 1 step 2) print(i) // prints "42"
```

enduntil

```
for (i in 1 until 10) { // i in [1, 10), 10 is excluded
println(i)
}
```

https://riptutorial.com/zh-CN/kotlin/topic/10121/

36:

◦ `(String) -> BooleanPair<Person, Person>` ◦

◦ ◦

- **typealias** *alias-name* **=** *existing-type*

◦ JVM◦ ◦

Examples

```
typealias StringValidator = (String) -> Boolean
typealias Reductor<T, U, V> = (T, U) -> V
```

```
typealias Parents = Pair<Person, Person>
typealias Accounts = List<Account>
```

https://riptutorial.com/zh-CN/kotlin/topic/9453/

37: Kotlin

Examples

Gradle

`kotlin-gradle-plugin`GradleKotlin。 Kotlin。 Kotlin `1.0.3` `kotlin-gradle-plugin` `1.0.3`。

`gradle.propertiesExtraPropertiesExtension`

```
buildscript {
   ext.kotlin_version = '1.0.3'

   repositories {
     mavenCentral()
   }

   dependencies {
     classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
   }
}
```

。

JVM

```
apply plugin: 'kotlin'
```

Android

```
apply plugin: 'kotlin-android'
```

JS

```
apply plugin: 'kotlin2js'
```

- kotlin `src/main/kotlin`
- java `src/main/java`
- kotlin `src/test/kotlin`
- java `src/test/java`
- `src/main/resources`
- `src/test/resources`

`SourceSets` 。

Kotlin

```
dependencies {
    compile "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
}
```

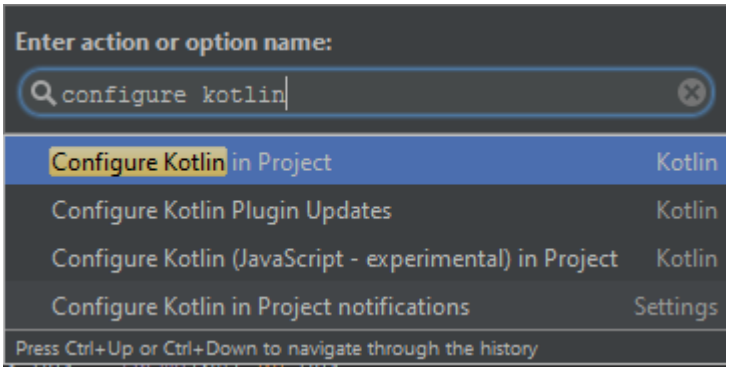
Kotlin Reflection`compile "org.jetbrains.kotlin:kotlin-reflect:$kotlin_version"`

Android Studio

Android StudioAndroidKotlin。

Kotlin>>>>JetBrains...> Kotlin>Android Studio。

Android Studio`Ctrl` + `Shift` + `A`.“Kotlin”Enter。



Android StudioGradle。

Java

JavaKotlin`Ctrl` + `Shift` + `A`“JavaKotlin”。 `.kt` Kotlin。

```java
package com.orangeflash81.myapplication;

public class Foo {
    private String name = "Joe Bloggs";

    String getName() { return name; }

    void setName(String value) { name = value; }
}
```

GroovyGradleKotlin

- gradle-script-kotlin
- /
    - `build.gradle.kts`
    - `gradlew`
    - `gradlew.bat`
    - `settings.gradle`
- `build.gradle.kts`
- IntellijGradle。
- Intellij`build.gradle.kts`。 /Intellij
- Gradle

Windows Gradle 3.3。 。

OSXUbuntu。

MavenJitpack Groovy。 。

Kotlin https://riptutorial.com/zh-CN/kotlin/topic/2501/kotlin

38:

Kotlin。API。

- listOfmapOfsetOf。
- arrayListOfhashMapOfhashSetOflinkedMapOfLinkedHashMaplinkedSetOfLinkedHashSet mutableListOfKotlin MultableListmutableMapOfThe Kotlin MultableMapmutableSetOfThe Kotlin MultableSet sortedMapOfsortedSetOf
- firstlastgetlambdafiltermapjoinreduce。

Examples

```
// Create a new read-only List<String>
val list = listOf("Item 1", "Item 2", "Item 3")
println(list) // prints "[Item 1, Item 2, Item 3]"
```

```
// Create a new read-only Map<Integer, String>
val map = mapOf(Pair(1, "Item 1"), Pair(2, "Item 2"), Pair(3, "Item 3"))
println(map) // prints "{1=Item 1, 2=Item 2, 3=Item 3}"
```

set

```
// Create a new read-only Set<String>
val set = setOf(1, 3, 5)
println(set) // prints "[1, 3, 5]"
```

https://riptutorial.com/zh-CN/kotlin/topic/8846/

S. No		Contributors
1	Kotlin	babedev, Community, cyberscientist, ganesshkumar, Ihor Kucherenko, Jayson Minard, mnoronha, neworld, Parker Hoyes, Ruckus T-Boom, Sach, Sean Reilly, Sheigutn, Simón Oroño, UnKnown, Urko Pineda
2	DSL	Dmitriy L, ice1000
3	Java 8 Stream	Brad, Gerson, Jayson Minard, Piero Divasto, Sam
4	JUnit	jenglert
5	Kotlin Android	Jemo Mgebrishvili, Ritave
6	Kotlin for Java Developers	Thorsten Schleinzer
7	Kotlin	Shinoo Goyal
8	Kotlin	Grigory Konushev, Spidfire
9		Brad Larson, jereksel, Sapan Zaveri
10	Vararg	byxor, piotrek1543, Sam
11		Aaron Christiansen, baha, Caelum, glee8e, Jayson Minard, KeksArmee, madhead, Spidfire
12		Jemo Mgebrishvili
13		Divya, glee8e
14		atok, Kirill Rakhman, madhead, Ritave, Sup
15		Avijit Karmakar
16	KotlinRecyclerView	Mohit Suthar
17	Lambdas	memoizr, Rich Kuzsma
18		Sam, Seaskyways
19		Januson, Sam
20		Aaron Christiansen, Adam Arold, Brad Larson, Héctor, Jayson Minard, Konrad Jamrozik, madhead, mayojava, razzledazzle, Sapan Zaveri, Serge Nikitin, yole

21		Dávid Tímár, Jayson Minard, Kevin Robatel, Konrad Jamrozik, olivierlemasle, Parker Hoyes, razzledazzle
22		Divya, Jan Vladimir Mostert, Jayson Minard, Ritave, Robin
23		egor.zhdan, Sam, UnKnown
24		KeksArmee, Kirill Rakhman, piotrek1543, razzledazzle, Robin, SerCe, Spidfire, technerd, Thorsten Schleinzer
25		Abdullah, Alex Facciorusso, jpmcosta, Kirill Rakhman, Robin, Spidfire
26		David Soroko, Kirill Rakhman, SerCe
27		Espen, Travis
28		hotkey, Jayson Minard, KeksArmee
29		Brad Larson, Caelum, Héctor, Mood, piotrek1543, Sapan Zaveri
30		Sam
31	kotlin	Konrad Jamrozik, olivierlemasle, oshai
32		Ben Leggiero, JaseAnderson, mayojava, razzledazzle, Robin
33		Slav
34		byxor, KeksArmee, piotrek1543, Slav
35		Nihal Saxena
36		Kevin Robatel
37	Kotlin	Aaron Christiansen, elect, madhead
38		Ascension