原 Kotlin入门实例大全(完结)

置顶 2018年10月23日 16:25:32 超级大黄狗Shawn 阅读数：99

Kotlin Cases(完结)

> Shawn qq:121857051
> 转载请留情

- IDE: IntelliJ IDEA 2018.2.3
- Kotlin: 1.3.0-release-IJ2018.2-1
- JDK: 1.8.0_161

经过了近一个月的编写更新，我认为这篇文章给新手拿来入门应该算是可以了。
这70余个实例如果能挨个手码一遍,我相信任何新手都是能入门的吧。
学习一门编程语言，其实最重要的是英语水平（笑）。
如果你的英语水平还可以，其实大部分编程语言是可以不学习就能粗略读懂的。
当然，如果你有编程基础，了解基本的模式，那么学习一门语言的速度就可以更快，比如一个java程序员估计只用一个小时就能基本熟练

这篇文章如果有任何错误、纰漏，请直接提出，我会及时更正。

https://blog.csdn.net/weixin_41084236/article/details/83308910
强烈推荐Kotlin官方文档
代码见github
注: 本文中实例部分参考于 runoobJava实例

Kotlin入门实例

- November 19, 2018 : update_ComposeApplyAndAlso
- November 19, 2018 : update_ComposeRun
- November 19, 2018 : update_ComposeWith
- November 19, 2018 : update_ComposeLet
- November 15, 2018 : update_ComposeFilter
- November 15, 2018 : update_ComposeFold
- November 14, 2018 : update_ComposeReduce
- November 13, 2018 : update_ComposeFlapMap
- November 12, 2018 : update_ComposeMap
- November 09, 2018 : update_Compose
- November 08, 2018 : update_Null
- November 07, 2018 : update_Component
- November 06, 2018 : update_Closure
- November 05, 2018 : update_Thread
- November 02, 2018 : update_This
- November 01, 2018 : update_Typealias
- November 01, 2018 : update_KDoc
- October 31, 2018 : update_DataClassAndSealedClass
- October 30, 2018 : update_TipsInfix
- October 29, 2018 : update_OperatorOverload
- October 26, 2018 : update_TipsRefEqual
- October 25, 2018 : update_TipsCases
- October 24, 2018 : update_ClassCases

---

1 字符串

1.1 字符串比较 StringCompare

- 功能: 通过compareTo和equals比较两个字符串
- 介绍: compareTo会比较两个字符串的首字母(字典序),equals会比较两个字符串,两个方法均有可选忽视大小写的参数.

```
fun main(args: Array<String>) {
    var str : String = "Hello World"
    var str1 : String = "Hello World"
    var str2 : String = "hello World"
    println(str.compareTo(str1))
    println(str.compareTo(str2))
    println(str.compareTo(str2,true))
    println(str.equals(str1))
    println(str.equals(str2))

```

```
    println(str.equals(str2,true))
}
```

1.2 查找最后一次出现位置 StringLastIndexOf

- 功能: 通过lastIndexOf查找给定字符串在目标字符串最后一次出现的位置.
- 介绍:
  - lastIndexOf是从后向前查找的.
  - startIndex参数可以指定从哪里开始向前查找.
  - 传递参数可以按位置传递,也可以按参数名传递.

```
fun main(args: Array<String>) {
    var str : String = "Hello, handsome boy."
    var str1: String = "boy? I am not a boy!"
    var lastHandsome : Int = str.lastIndexOf("handsome")
    var lastBoy : Int = str.lastIndexOf("BoY",ignoreCase = true,startIndex = str.length)
    var lastBoyInStr1 : Int = str1.lastIndexOf("boy",str1.length-5,true)
    println("The last index of \"handsome\" is "+lastHandsome)
    println("The last index of \"boy\" is "+lastBoy)
    println("Now I ignore the last 5 chr in str1: "+lastBoyInStr1)
}
```

1.3 删除字符 StringRemoveChr

- 功能: 做一个函数来实现删除字符串给定位置的字符.
- 介绍:
  - 构建返回非空函数的时候最好先声明返回类型.
  - PHP可能不是最好的语言.

```
fun main(args: Array<String>) {
    var str : String = "PHP, the best language."
    println(removeCharAt(str,1))
}

fun removeCharAt(s:String,pos:Int): String{
    return s.substring(0,pos)+s.substring(pos+1)
}
```

1.4 反转字符串 StringReverse

- 功能: 反转字符串
- 介绍: 无

```
fun main(args: Array<String>) {
    var str : String = "Shawn"
    var strReversed : String = str.reversed()
    println(str)
    println(strReversed)
}
```

1.5 查找字符串 StringIndexOf

- 功能: indexOf搜索字符串出现的位置.
- 介绍: 正向搜索,参考1.2,若没找到则返回-1.

```
fun main(args: Array<String>) {
    var str : String = "Python CPP Java Ruby Kotlin"
    printPos(str.indexOf("python"))
    printPos(str.indexOf("python",ignoreCase = true))
    printPos(str.indexOf("Java"))
}

fun printPos(pos:Int){
```

```
    if(pos == -1){
        println("Do not find that string.")
    }
    else
    {
        println("Find that string in "+pos)
    }
}
```

1.6 字符串分割 StringSplit

- 功能: 通过split分割字符串
- 介绍:
  - split可以指定以特定标志等来分割字符串
  - kotlin在创建对象的时候有一定的猜测(类型)能力

```
fun main(args: Array<String>) {
    var str : String = "Shawn-handsome-python-best"
    var splited : List<String> = str.split("-")
    println(splited)
}
```

1.7 字符串大小写 StringUpperAndLower

- 功能: 通过upper和lower可以将字符串做大小写变换
- 介绍: 无

```
fun main(args: Array<String>) {
    var str = "Shawn"
    println(str.toUpperCase())
    println(str.toLowerCase())
}
```

1.8 字符串区域匹配 StringRegionMatch

- 功能: 通过regionMatches来确定两个字符串给定区域是否相同
- 介绍:
  - 第四行翻译一下: 从str1的第4位与str2的第4位开始数,2位是否都相等.
  - 同样支持ignoreCases

```
fun main(args: Array<String>) {
    var str1 = "php Is the best."
    var str2 = "who is your daddy?"
    println(str1.regionMatches(4,str2,4,2))
    println(str1.regionMatches(4,str2,4,2,true))
}
```

1.9 字符串格式化 StringFormat

- 功能: 使用字符串格式化
- 介绍: kotlin中直接调用java的函数也是ok的.

```
import java.util.*

fun main(args: Array<String>) {
    var e = Math.E
    System.out.format("%f%n",e)
    System.out.format(Locale.CHINA, "%-10.4f%n%n", e)
}
```

1.10 连接字符串 StringAppend

- 功能: 用+连接字符串

- 介绍: 无

```
fun main(args: Array<String>) {
    var str1 = "No gods "
    var str2 = "Or kings "
    var str3 = "Only man!"
    println(str1+str2+str3)
}
```

2 数组

2.1 数组排序以及二分查找 ArraysSortAndSearch

- 功能: 数组的排序\搜索
- 介绍:
    - kotlin中生成整数数组用intArrayOf
    - 生成Array可以用arrayOf来进行.

```
fun main(args: Array<String>) {
    var array = intArrayOf(1,5,2,3,9,1)
    print("Original: ")
    printIntArray(array)
    array.sort()
    print("Sorted: ")
    printIntArray(array)
    println("'5' in the position: "+array.binarySearch(5))

}

fun printIntArray(intArray: IntArray){
    for(i in intArray){
        print(" "+i)
    }
    println()
}
```

2.2 添加元素 ArraysInsert

- 功能: 生成ArrayList并向其中添加元素.
- 介绍: 如果函数返回为空,可以声明Unit(也可以什么都不写)

```
fun main(args: Array<String>) {
    var array = arrayListOf(1,2,3,4,"dog")
    array.add(2.3)
    array.add(true)
    printArrayList(array)
}

public fun printArrayList(arrayList: ArrayList<out Any>):Unit{
    for(i in arrayList){
        println(i.toString())
    }
}
```

2.3 数组长度 ArraysLength

- 功能: 获取数组的长度
- 介绍: 无

```
fun main(args: Array<String>) {
    var array : Array<Any> = Array<Any>(4,{9})
    array.set(0,1)
    array.set(1,"hop")
    for(i in array){
        println(i)
```

```
    }
    println("The length of array is "+array.size)
}
```

2.4 数组反转 ArraysReverse

- 功能: 反转数组
- 介绍: reverse可以直接将ArrayList反转.

```
fun main(args: Array<String>) {
    var array = arrayListOf<Any>(1,2,3,4,5)
    array.reverse()
    printArrayList(array)//from ArraysInsert.kt
}
```

2.5 数组极值 ArraysMaxAndMin

- 功能: 获得数组的极值
- 介绍: 数组的max和min可以直接获得.

```
fun main(args: Array<String>) {
    var array = intArrayOf(2,3,1,6,-1,6,99)
    println("The maximum of array is "+array.max())
    println("The minimum of array is "+array.min())
}
```

2.6 数组合并 ArraysMerge

- 功能: 将两个数组合并
- 介绍: array和list可以直接用相加进行合并

```
fun main(args: Array<String>) {
    var array1 : Array<String> = arrayOf("S","h","a")
    var array2 : Array<String> = arrayOf("w","n")
    var list = array1.asList()
    list+=array2.asList()
    println(list)
    array1+=array2
    printArray(array1)
}

public fun printArray(array:Array<out Any>){
    for (i in array){
        println(i)
    }
}
```

2.7 数组填充 ArraysFill

- 功能: 初始化一个指定大小的数组,并用指定数字填充
- 介绍: printIntArray见ArraysSortAndSearch

```
fun main(args: Array<String>) {
    var array = IntArray(6)
    array.fill(2)
    printIntArray(array)
}
```

2.8 数组扩展 ArraysExtend

- 功能: 扩展数组
- 介绍: 创造一个容量合适的新数组,然后将需要的数据拷进去.

```
fun main(args: Array<String>) {
    var array = intArrayOf(1,2,3)
    var arrayExtended = IntArray(5)
    printIntArray(arrayExtended)
    arrayExtended[3] = 4
    arrayExtended[4] = 5
    System.arraycopy(array,0,arrayExtended,0,array.size)
    printIntArray(arrayExtended)
}
```

2.9 查找数组中的重复元素 ArraysDuplicates

- 功能: 查找重复元素
- 介绍: kotlin的in操作可以很轻松胜任这个工作

```
fun main(args: Array<String>) {
    var array = intArrayOf(1,1,2,3,4,4,5,5,5,5,6,7,8,8,1,1,9,9)
    var dup = ArrayList<Int>()
    for (i in 0 until array.size-1){
        for(j in i+1 until array.size){
            if (array[i]==array[j] && !(array[i] in dup)){
                println("Find duplicate: "+array[i])
                dup.add(array[i])
            }
        }
    }
}
```

2.10 删除元素 ArraysRemove

- 功能: 删除指定位置的元素
- 介绍: 用removeAt可以删除指定索引的元素

```
fun main(args: Array<String>) {
    var arrayList = ArrayList<String>()
    arrayList.add("The first element")
    arrayList.add("The second element")
    arrayList.add("The third element")
    arrayList.removeAt(1)
    printArrayList(arrayList)
}
```

2.11 差集 ArraysRemoveAll

- 功能: 获得差集
- 介绍: removeAll会将找到的全部元素都删掉,而不是删去相应的数量

```
fun main(args: Array<String>) {
    var arrayList1 = arrayListOf(1,2,2,3,4,4)
    var arrayList2 = arrayListOf(1,2,5,3,3)
    arrayList1.removeAll(arrayList2)
    printArrayList(arrayList1)
}
```

2.12 交集 ArraysRetainAll

- 功能: 获得交集
- 介绍: retainAll会将找到的元素保留,为找到的舍弃

```
fun main(args: Array<String>) {
    var arrayList1 = arrayListOf("Shawn","Doge","Doge","XueBi","NiShiLong","NiShiLong")
    var arrayList2 = arrayListOf("Shawn","Doge","Biu","WoShiLong","WoShiLong")
    arrayList1.retainAll(arrayList2)
    printArrayList(arrayList1)
}
```

2.13 查找元素 ArraysSearch

- 功能: 查找指定元素是否存在于集合中
- 介绍: 有很多种方法实现,用in是最省事的一种

```
fun main(args: Array<String>) {
    var arrayList = arrayListOf(1,3,5,2,4,6,"DFWill","Boom!")
    println("DFWill in the list? "+("DFWill" in arrayList))
    println("1 in the list? "+arrayList.contains(1))
}
```

2.14 数组相等 ArraysEquals

- 功能: 判断两个数组内容是否相等
- 介绍: 使用contentEquals来进行比较.

```
fun main(args: Array<String>) {
    var array1 = intArrayOf(1,2,3,4)
    var array2 = intArrayOf(1,2,3,4)
    var array3 = intArrayOf(1,2)
    println("array1 equals array2? "+array1.contentEquals(array2))
    println("array1 equals array3? "+array1.contentEquals(array3))
}
```

2.15 并集 ArraysUnionSet

- 功能: 获得两个数组的并集
- 介绍:
  - set中不存在重复元素,所以可以用来去重.
  - set的交并补等运算更贴近实际逻辑.

```
fun main(args: Array<String>) {
    var array1 = intArrayOf(1,2,2,3)
    var array2 = intArrayOf(3,4,4,5)
    var unionSet = emptySet<Int>()
    unionSet+=array1.toSet()
    unionSet+=array2.toSet()
    printSet(unionSet)
}
public fun printSet(s:Set<out Any>){
    for(i in s){
        print(i.toString()+" ")
    }
    println()
}
```

3 时间

格式化时间

- 功能: 格式化输出时间
- 介绍: 直接调用java的相关模块即可.

```
import java.text.SimpleDateFormat
import java.util.*

fun main(args: Array<String>) {
    val data = Date()
    val dateFormat = "yyyy-MM-dd HH:mm:ss"
    val simpleDateFormat = SimpleDateFormat(dateFormat)
    println(simpleDateFormat.format(data))
}
```

4 类与对象

4.1 创建类 ClassInit

- 功能: 构建并实例一个类
- 介绍:
  - 类的可以有一个主构造函数及一个或多个次构造函数,在类头的是主构造函数(constructor)
  - 类中可以有多个初始化块(init)
  - 所有类集成自超类Any(与java的Object不同)

```
class ClassInit constructor(info:String){
    init{
        println(info+" in the initializer block 1")
    }
    var str1 = "var str1".also(::println)
    init{
        println(info+" in the initializer block 2")
        this.str1 = info
    }
    fun printstr1(){
        println("str1: "+str1)
    }
    val str2 = "val str2".also(::println)
}

fun main(args: Array<String>) {
    println("Begin")
    var classInit = ClassInit("New")
    var classInit2 = ClassInit("New2")
    classInit.printstr1()
    classInit2.printstr1()
}
```

4.2 次构造函数 ClassSecondaryConstructor

- 功能: 次构造函数
- 介绍:
  - 如果既有主构造函数,又有次构造函数,则次构造函数需要委托给主构造函数
  - 主构造函数中可以直接声明类中的变量(var)
  - 多个次构造函数可以实现多级调用
  - 次级构造函数会在所有变量初始化\initBlock调用完后再进行调用
  - 调用次级函数会先调用委托的部分
  - 可以指定默认参数值是一件很幸福的事情

```
class ClassSecondaryConstructor constructor(var name:String){
    init{
        println("Init block "+name)
    }
    var age = 0
    var father = "father of ${name}".also(::println)
    constructor(name:String,age:Int):this(name){
        this.age = age
        println("First secondary constructor "+name)
    }
    constructor(name:String,age:Int = 18,father:String):this(name,age){
        this.father = father
        println("Second secondary constructor "+name)
    }
    init{
        println("Init block2 "+name)
    }
}

fun main(args: Array<String>) {
    var a = ClassSecondaryConstructor("Shawn")
    var b = ClassSecondaryConstructor("XiaoMing",24)

```

```
    var c = ClassSecondaryConstructor("Me",father = "Father")
}
```

4.3 继承 ClassDerived

- 功能: 继承父类
- 介绍:
  - 如果没有主构造函数,次构造函数不需要委托
  - 可继承的类,可覆写的方法,属性都要用open修饰
  - 覆写前要用override声明
  - 可以用var覆写val,反之不行
  - 用super同样可以调用父类

```
open class Father{
    public var age = 18
    private var name = ""
    constructor(name: String){
        this.name = name
    }
    open public fun printName(){
        println(name)
    }
    public fun printAge(){
        println(age)
    }
}

class Boy(name:String,age:Int):Father(name){
    init{
        this.age = age
    }
    override fun printName(){
        super.printName()
        println("from Boy")
    }

}

fun main(args: Array<String>) {
    var father = Father("Father")
    var boy = Boy("Boy",5)
    father.printName()
    father.printAge()
    boy.printName()
    boy.printAge()
}
```

4.4 抽象与接口 ClassAbstractAndInterface

- 功能: 继承抽象类,满足接口
- 介绍:
  - 抽象与接口的规则老一套
  - super<父类>可以实现对多个父亲的选择

```
abstract class ClassAbstract(){
    var i: Int? = null
    abstract fun a()
    abstract fun b()
}
interface ClassAbstractA{
    fun a(i: Int):Int{
        return i
    }
}
interface ClassAbstractB{
```

```
    fun b(s: String):String{
        return s
    }
}
class ClassAbstractClass:ClassAbstract(),ClassAbstractA,ClassAbstractB{
    override fun a(){
        println("a")
    }
    override fun a(i:Int):Int{
        println("a"+i)
        return i+1
    }
    override fun b(){
        println("b")
    }
    override fun b(s: String): String {
        println("b"+s)
        return super<ClassAbstractB>.b(s)
    }
}

fun main(args: Array<String>) {
    var abstract = ClassAbstractClass()
    abstract.a()
    abstract.a(1)
    abstract.b()
    abstract.b("test")
}
```

4.5 getter和setter ClassGetterAndSetter

- 功能: 重写类属性的set和get
- 介绍:
  - 通过重写这两个访问器,可以更灵活地实现功能
  - val只支持get重写

```
fun main(args: Array<String>) {
    var get = GetAndSet()
    for(i in 0..5){
        get.answer = i
        println(get.answer)
    }
}

class GetAndSet(){
    var data = 0
    var answer : Int
        get() {
            if(data!=0)return 42
            else return data
        }
        set(value) {data = value%2}
    }
```

4.6 幕后字段 ClassField

- 功能: 用Field访问属性自身
- 介绍:
  - 由于在类属性的set和get内访问自身会进行递归最后导致栈溢出,所以kotlin提供了yield标识符.
  - field不需要声明,在变量初始化后自动提供.

```
fun main(args: Array<String>) {
    var get = ClassField()
    for(i in 0..5){
        get.answer = i
        println(get.answer)
```

```
    }
}

class ClassField(){
    var answer : Int = 0
        set(value){
            field = value-1
        }
        get() {
            if(field==0)return 42
            else return field
        }
}
```

4.7 扩展函数 ClassExtend

- 功能: 使用扩展方法对类进行扩展
- 介绍:
    - 可以通过在类外定义函数来扩展类本身.
    - 扩展的函数不能访问类的私有属性

```
class ClassExtend{
    var str: String?= null
}
fun ClassExtend.printStr(){
    println(str)
}
fun ClassExtend.setStr(str:String){
    this.str = str
}

fun main(args: Array<String>) {
    var CE = ClassExtend()
    CE.printStr()
    CE.setStr("input something")
    CE.printStr()
}
```

4.8 伴生对象(静态成员) ClassCompanion

- 功能: 通过伴生对象来实现静态方法
- 介绍:
    - Kotlin中没有静态成员声明,但是有伴生方法可以实现
    - 虽然看起来像静态成员,但是伴生方法仍然是真是的实例成员,并且可以实现接口

```
class ClassCompanion{
    companion object {
        fun printHello(){
            print("Hello!")
        }
        val Dio: String = "Dio"
    }
}

fun main(args: Array<String>) {
    ClassCompanion.printHello()
    print(" "+ClassCompanion.Dio)
}
```

4.9 枚举类 ClassEnum

- 功能: 使用枚举类
- 介绍: 与Java中的枚举类区别不大

```
enum class ClassEnum(val rgb:Int,var stuff:String){
    RED(0xFF0000,"apple"){
        override fun print() {
            println(this.stuff)
        }
    },
    GREEN(0x00FF00,"Your hat"){
        override fun print() {
            println("That is ok "+this.stuff)
        }
    },
    BLUE(0x0000FF,"Sky"){
        override fun print() {
            println("I need some towers")
        }
    };
    abstract fun print()
}

fun main(args: Array<String>) {
    ClassEnum.RED.print()
    ClassEnum.GREEN.print()
    println(ClassEnum.BLUE.rgb)
    println(ClassEnum.BLUE.stuff)
    ClassEnum.BLUE.stuff = "Sea"
    println(ClassEnum.BLUE.stuff)
}
```

4.10 数据类 ClassData

- 功能: 用数据类来记录数据
- 介绍:
    - 数据类被设计来储存数据
    - 数据类的主构造函数至少要有一个参数
    - 主构造函数的多有参数必须有var或val修饰
    - 如果不特别实现,编译器会自动推导equals(),hashCode()和toString()
    - 用copy()可以快速实现同一份数据的复制\微调

```
data class ClassData(var aString: String,val bString: String){
    init {
        var sString = bString
    }
    fun print(){
        println("Print in ${this.javaClass}")
    }
}

fun main(args: Array<String>) {
    var banana = ClassData("aBanana","bBanana")
    var Banana = banana.copy(bString = "cBanana")
    println(banana)
    banana.print()
    println(Banana)
    Banana.print()
}
```

4.11 密封类 ClassSealed

- 功能: 使用密封类来让程序更顺眼
- 介绍:
    - 密封类自身是抽象的,不能直接实例化
    - 其构造函数为private类型的,且不允许有其他类型的构造函数
    - 密封类实现了一个有限的类集合
    - 密封类与when的结合非常严谨

- when中使用密封类如果覆盖了全部的情况,则不需要定义else子句
- when中使用了密封类,如果指明验证了其类型,则可以直接调用相应的方法.

```
sealed class ClassSealed
data class SealedA(var seal:String):ClassSealed()
class SealedB:ClassSealed(){
    fun print(){
        println("I am a little seal.")
    }
}
object SealedC:ClassSealed()

fun main(args: Array<String>) {
    var list = listOf<ClassSealed>(SealedB(),SealedA("I am here for basking."),SealedC)
    for (seal:ClassSealed in list){
        when(seal){
            is SealedA -> println(seal.seal)
            is SealedB -> seal.print()
            is SealedC -> println("And disappear again.")
        }
    }
}
```

5 流程控制

5.1 分支流程 ControlSwitch

- 功能: 用if/when来控制程序流程
- 介绍:
  - if-else如常
  - when语句取代了switch
  - when语句不会依次执行,所以不需要额外的break

```
fun main(args: Array<String>) {
    var a = 1
    var b = 2
    if(a>b)
    {
        println(a)
    }
    else
    {
        println(b)
    }
    var max = if(a>b) a else b
    println(max)
    fun max(a:Int,b:Int) = when(a>b){
        true -> a
        false -> b
        else -> null
    }
    fun isString(a:Any) = when(a){
        is String -> println("That is a string")
        !is String -> println("That is not a string")
        else -> println("I have no idea")
    }
    println(max(233,666))
    isString("Fish!")
}
```

5.2 循环 ControlLoop

- 功能: while循环与java相同,主要介绍for循环
- 介绍:
  - 0…5 包括5

- 0 until 5 不包括5
- 5 downTo 0 包括5和0

```
fun main(args: Array<String>) {
    var array = intArrayOf(1,2,3,4,5,6,7)
    for(i in array) print(i)
    println()
    for(i:Int in array) print(i)
    println()
    for(i in 0..5) print(i)
    println()
    for(i in 0 until 5) print(i)
    println()
    for(i in 0 until 5 step 2)print(i)
    println()
    for(i in 5 downTo 0)print(i)
    println()
    for(i in 5 downTo 0 step 2) print(i)
}
```

5.3 转跳 ControlJump

- 功能: break以及continue
- 介绍:
  - break未指定标签,则终止最近一个循环
  - continue未指定标签,则跳过最近一个循环的当次循环
  - 用标签可为以上两个命令指定控制的循环
  - 标签也同样可以用于return,最常见于直接从lambda返回到函数主体

```
fun main(args: Array<String>) {
    loop@ for(i in 0..5){
        loop2@ for(j in 6..10){
            println(""+i+" and "+j)
            if (i == 5) {
                println("loop break")
                break@loop
            }
            if (j == 9) {
                println("loop2 continue")
                continue@loop2
            }
            if ((i==0) and (j==7)) {
                println("loop continue")
                continue@loop
            }

        }
    }
}
```

6 异常 Exception

- 功能: Kotlin中的异常处理与Java相似
- 介绍:
  - 用try-catch来捕捉异常
  - 可以用多个catch针对多个异常
  - 接finally来运行无论是否发生异常均可执行的程序

```
import java.lang.Exception

fun main(args: Array<String>) {
    var list = intArrayOf(0)
    for (i in 0..1){
        try {
```

```
        list[i] = 1 / i
        println(list[i])
      }catch(e: ArithmeticException){
        e.printStackTrace()
      }catch(e: ArrayIndexOutOfBoundsException){
        e.printStackTrace()
      }catch(e: Exception){
        e.printStackTrace()
      }finally {
        println("Finally "+i)
      }
    }

}
```

7 运算符重载

7.1 一元运算符 OperatorOverloadUnary

- 功能: 重载类的一元运算符
- 介绍:
  - 类的运算符可以重载
  - 重载后的返回值与运算符原定义无关
  - 重载运算符要标注operator

```
class OperatorOverloadUnary(var i:Int){
    operator fun unaryPlus():Int{
        println("UnaryPlus: $i")
        return i
    }
    operator fun unaryMinus():Int{
        println("UnaryMinus: $i")
        return i-1
    }
    operator fun not():Int{
        println("not: $i")
        return i-2
    }

}

fun main(args: Array<String>) {
    val i = OperatorOverloadUnary(9)
    println(+i)
    println(-i)
    println(!i)
}
```

7.2 自运算 OperatorOverloadIncAndDec

- 功能: 重载自运算符
- 介绍:
  - 自运算如i++,i–的实现过程是:获取i初始值,获取i.inc()或i.dec()并赋给i,返回i初始值.
  - 自运算++i,–i的实现过程相似,只是返回的是i.inc()或i.dec()的值

```
class OperatorOverloadIncAndDec(var i:Int){
    init {
        println("Here is $i")
    }

    override fun toString(): String {
        return "toString: $i"
    }
    operator fun inc():OperatorOverloadIncAndDec{
        println("INC $i")
        return OperatorOverloadIncAndDec(i+10)
```

```
    }
    operator fun dec():OperatorOverloadIncAndDec{
        println("DEC $i")
        return OperatorOverloadIncAndDec(i-10)
    }
}

fun main(args: Array<String>) {
    var i = OperatorOverloadIncAndDec(42)
    i++
    i--
}
```

7.3 二元运算 OperatorOverloadBinary

- 功能: 重载二元运算符
- 介绍: 二元运算符要求有一个参数传入,类型不限

```
class OperatorOverloadBinary(var i: Int){
    operator fun plus(j:Int){
        println("plus $i and $j")
    }
    operator fun minus(s:String){
        println("minus $i and $s")
    }
    operator fun times(j:Int){
        println("times $i and $j")
    }
    operator fun div(j:Int){
        println("div $i and$j")
    }
    operator fun rem(j:Int){
        println("rem $i and $j")
    }
    operator fun rangeTo(s:String){
        println("rangeTo $i and $s")
    }
}

fun main(args: Array<String>) {
    var i = OperatorOverloadBinary(9527)
    i+1
    i-"Banana"
    i*3
    i/4
    i%5
    i.."Banana!"
}
```

7.4 赋值运算 OperatorOverloadAssign

- 功能: 重载赋值运算符
- 介绍: 如果相应函数不可用,则会尝试去调用二元运算

```
class OperatorOverloadAssign(var i: Int){
    operator fun plusAssign(s:String){
        println("plusAssign $i and $s")
    }
    operator fun minusAssign(i:Int){
        println("minusAssign ${this.i} and $i")
    }
    operator fun timesAssign(s:String){
        println("timesAssign $i and $s")
    }
    operator fun divAssign(i:Int){
        println("divAssign ${this.i} and $i")
    }
    operator fun remAssign(s:String){
```

```
15      println("remAssign $i and $s")
16    }
17  }
18
19  fun main(args: Array<String>) {
20    var i = OperatorOverloadAssign(666)
21    i+="banana"
22    i-=2
23    i*="Banana"
24    i/=3
25    i%="Banana!"
26  }
```

7.5 调用与访问 OperatorOverloadInvoke

- 功能: 重载调用与访问
- 介绍: 无

```
1   class OperatorOverloadInvoke(var i: Int){
2     operator fun get(i:Int){
3       println("get ${this.i} and $i")
4     }
5     operator fun set(i:Any,j:Any){
6       println("set ${this.i} $i and $j")
7     }
8     operator fun invoke(i:Any){
9       println("invoke ${this.i} and $i")
10    }
11  }
12
13  fun main(args: Array<String>) {
14    var i = OperatorOverloadInvoke(233)
15    i[12]
16    i["banana"] = "Banana"
17    i("Banana!")
18  }
```

7.6 逻辑运算 OperatorOverloadLogic

- 功能: 重载逻辑运算符
- 介绍:
  - !in和!=是in和==与!的复合运算
  - 所有的大小比较其实都是compareTo()与0的比较

```
1   class OperatorOverloadLogic(var i: Int){
2     operator fun contains(i:Any):Boolean{
3       println("contains ${this.i} and $i")
4       return false
5     }
6     override fun equals(other: Any?): Boolean {
7       println("equals $i and $other")
8       return true
9     }
10    operator fun compareTo(i:Any):Int{
11      println("compareTo ${this.i} and $i")
12      return 42
13    }
14  }
15
16  fun main(args: Array<String>) {
17    var i = OperatorOverloadLogic(450)
18    println("banana" in i)
19    println(i == OperatorOverloadLogic(12))
20    println(i>1)
21    println(i<2)
22    println(i>=3)
23
```

```
    println(i<="Banana")
}
```

8 代码文档 KDoc

- 功能: 编写Kotlin代码文档
- 介绍:
  - Kotlin的代码文档KDoc和java的javaDoc区别不大.
  - 支持markdown的语法,但是现在md除了基础语法,高级功能的语法并不统一.
  - 所以并不推荐用特别炫酷的md语法去写文档,实用就可以了.
  - 生成文档还需要依赖dokka

```
/**
 * This is a KDoc Demo
 * *Here* you can write doc in a markdown way.
 *
 * @param  KDoc a param of class KDoc
 * @author Shawn
 */

class KDoc(var KDoc:String){
    /**
     * @property varA a property of [KDoc]
     */
    var varA = 1
    fun funA(paramA:String):String{
        /**
         * @param paramA a param of funA
         * @return return the paramA
         */
        return paramA
    }
}
```

9 并发

9.1 创建新线程 ThreadNew

- 功能: 创建新线程
- 介绍: 同java

```
class ThreadNew : Thread(){
    override fun run(){
        super.run()
        Thread.sleep(5000)
        println("Hello")
    }
}

fun main(args: Array<String>) {
    val mThread = ThreadNew()
    mThread.run()
}
```

9.2 实现runnable接口 ThreadRunnable

- 功能: 通过runnable接口来创建新线程
- 介绍: 同java

```
fun main(args: Array<String>) {
    val mRunnable = Runnable {
        run {
            Thread.sleep(3000)
            println("Hello")
        }
```

```
    }
    Thread(mRunnable).start()
}
```

10 高级函数

10.1 调用 ComposeCall

- 功能: 调用函数
- 介绍:
  - 高级函数的复合原理: 把函数本身,而不是返回值,来作为参数或者返回值.
  - 包级函数可以直接用两个冒号调用.
  - 类函数可以用类名加两个冒号调用.

```
class ComposeCall{
    fun sayHi(){
        println("Hi!")
    }
    fun sayHiToAny(any: Any){
        println("Hi!"+any)
    }
}

fun main(args: Array<String>) {
    val hi = ComposeCall::sayHi
    val composeCall = ComposeCall()
    val arr = arrayOf("banana","Banana")

    arr.forEach(::println)
    arr.forEach(composeCall::sayHiToAny)
    println(arr.filter(String::isNotEmpty))

    println(hi)
    println(ComposeCall::sayHiToAny)
}
```

10.2 map ComposeMap

- 功能: map的使用
- 介绍:
  - forEach和map都能遍历数组,不同的是:forEach会对每个元素逐个进行处理,而map会统一处理完后返回一个新数组.
  - forEach和map中,用it都能代表遍历时的单个元素.
  - map函数可以通过传入函数来实现函数的复合.

```
fun main(args: Array<String>) {
    val raw = listOf(1,2,3,3,3,3)
    val newList = ArrayList<Int>()

    raw.forEach {
        newList.add(it+1)
    }

    println(newList)

    println(raw.map{it*10})
    println(raw.map(Int::toDouble))
}
```

10.3 flapMap ComposeFlapMap

- 功能: flapMap的使用
- 介绍:
  - 用flapMap可以将嵌套化的数组扁平化输出.

```
fun main(args: Array<String>) {
    val raw = listOf(
            1..5,
            99..101,
            999..1002
    )
    println(raw)
    println(raw.flatMap { it })
    println(
            raw.flatMap {
                intRange -> intRange.map{
                int -> "No.$int"
                }
            }
    )
}
```

10.4 reduce ComposeReduce

- 功能: reduce的使用
- 介绍:
    - 使用reduce来进行累计操作.
    - 第一个参数代表累计的最终结果.
    - 第二个参数代表遍历时的每个元素.
    - 后面连接的是要对元素进行的操作.

```
fun main(args: Array<String>) {
    val list = listOf(5,9,10)
    println(list.reduce{acc,i->acc*i})
}
```

10.5 fold ComposeFold

- 功能: fold的使用
- 介绍:
    - fold可以理解为带初始值的reduce
    - 将StringBuilder设置为初始值即可进行字符串拼接.

```
fun main(args: Array<String>) {
    val list = listOf(2,3,4,4,4)
    println(list.fold(1,{acc, i -> acc*i }))
    println(list.fold(StringBuilder(),{ acc, i -> acc.append(i) }))
}
```

10.6 filter ComposeFilter

- 功能: filter的使用
- 介绍:
    - 这里介绍三个函数: filter, filterIndexed 和 takeWhile
    - filter 可以执行遍历操作,将所有符合要求的要素全部提取
    - filterIndexed 在 filter 的基础上将索引也纳入了判断依据
    - takeWhile 则在出现第一个不合要求的要素的时候直接停止遍历

```
fun main(args: Array<String>) {
    val list = listOf(2,3,3,3,4,3,3)

    println(list.filter { it == 3 })

    println(list.filterIndexed { index, i -> index == i })

    println(list.takeWhile { it < 4 })
}
```

10.7 let ComposeLet

- 功能: let的使用
- 介绍:
  - 用let可以直接调用对象中的属性\函数,用it指代对象.
  - 用let也可以统一执行非空判断,当目标为null的时候不会执行let内的程序.

```
class ComposeLet(val s1: String, val num: Int){
    fun printS1(){
        println(s1)
    }
}

fun composeLet(i: Boolean):ComposeLet?{
    if(i){
        return ComposeLet("Banana",12)
    }else{
        return null
    }
}

fun main(args: Array<String>) {
    ComposeLet("banana",42).let {
        println(it.s1)
        it.printS1()
    }

    composeLet(true)?.let {
        it.printS1()
    }
    composeLet(false)?.let {
        println(it.num)
    }
}
```

10.8 with ComposeWith

- 功能: with的使用
- 介绍:
  - with中的this用来指代传入的对象.
  - with后接代码块的最后一行为with返回的结果.

```
class ComposeWith(val s: String){
    fun printS(){
        println(s)
    }
    override fun toString(): String {
        return s+" toString"
    }
}

fun main(args: Array<String>) {
    val result:String = with(ComposeWith("Banana")){
        this.printS()
        this.toString()
    }

    println(result)
}
```

10.9 run ComposeRun

- 功能: run的使用
- 介绍:
  - run可以视作let和with的结合体.

- 既不必用it来指代对象,又简化了with的非空判断问题.
- 任何能用let或者with的场景都能使用run.

```
class ComposeRun(val s: String, val s2: String){
    fun printS(){
        println(s)
    }
}

fun composeRun(i:Boolean):ComposeRun?{
    if (i){
        return ComposeRun("Banana","yes!")
    }else{
        return null
    }
}

fun main(args: Array<String>) {
    val s = composeRun(true)?.run {
        printS()
        "This will be returned."
    }
    composeRun(false)?.run{
        println("Do not work.")
    }

    println(s)

}
```

10.10 applyAndAlso ComposeApplyAndAlso

- 功能: apply和also的应用
- 介绍:
  - apply与run基本一致,只是run会闭包返回最后一行的代码,而apply会返回传入的对象.
  - also与let的关系与apply和run的关系一致.

```
class ComposeApplyAndAlso(val s: String){
    fun printS(){
        println(s)
    }
}

fun main(args: Array<String>) {
    val apply = ComposeApplyAndAlso("apply banana").apply {
        printS()
    }
    val also = ComposeApplyAndAlso("also banana").also {
        it.printS()
    }

    println(apply)
    apply.printS()
    println(also)
    also.printS()
}
```

PLUS 小技巧

P.1 可变长参数 TipsVarargs

- 功能: 用可变长参数来获取不定个数的参数
- 介绍: 无

```
fun tipsVarargs(vararg strings:String){
    for(s in strings){
```

```
    println(s)
  }
  for (i in 0 until strings.size){
    print(strings[i]+" ")
  }
}

fun main(args: Array<String>) {
  tipsVarargs("banana","Banana","banana","banana")
}
```

P.2 常量 Constant

- 功能: 试用常量
- 介绍:
  - val定义运行期常量(普通常量)
  - const val定义编译期常量
  - 编译期常量不能是可变对象
  - 用by lazy可以将普通常量的初始化推迟到第一次调用的时候

```
fun main(args: Array<String>) {
  var fruit = "banana"
  val Fruit by lazy { fruit }
  println(fruit)
  fruit = "Banana"
  println(Fruit)
}
const val TipsConst = "Only primitives and String are allowed"
```

P.3 获取class TipsGetClass

- 功能: 获得对象的class
- 介绍:
  - 用::class.java可以获取Java的class实例
  - 只用::class获取的是kotlin的class实例
  - 用实例的.javaClass也可以获取到class实例

```
class TipsGetClass

fun main(args: Array<String>) {
  val tipsGetClass = TipsGetClass()
  val cls = TipsGetClass::class.java
  val kls = TipsGetClass::class
  val tcls = tipsGetClass.javaClass
}
```

P.4 引用相等 TipsRefEqual

- 功能: 比较传统相等(==)和引用相等(===)
- 介绍:
  - 传统的相等很传统
  - 引用相等判断两个对象的引用是否是同一个

```
fun main(args: Array<String>) {
  var a = TipsRefEqual("You")
  var b = a
  var c = a
  var d = TipsRefEqual("You")
  println(a==b)
  println(b==c)
  println(a==d)
  println(a===b)
  println(b===c)
```

```
    println(a===d)
    println(b===d)
}

class TipsRefEqual(var name:String){
    override fun equals(other: Any?): Boolean {
        return this.name==other.toString()
    }
    override fun toString(): String {
        return this.name
    }
}
```

P.5 中缀表示法 TipsInfix

- 功能: 用infix修饰的函数可以使用中缀表示法调用
- 介绍:
    - 被修饰的函数必须是成员函数或者扩展函数
    - 被修饰的函数只能有一个参数
    - 这个参数不能是可变长参数,也不能有默认值
    - 使用中缀表示法a Bala b等同于a.Bala(b)
    - 使用中缀表示法必须指定接收者,设banana(String)是x类的infix函数:
        - 在x类内直接调用banana "Banana"是错误的
        - x类内可以调用this banana “Banana”
        - 或者调用banana(“Banana”)
    - 中缀表示法的优先级低于算数运算符,类型转换,rangeTo,但是高于逻辑运算符
    - 搞不懂优先级就直接用括号括起来.

```
class TipsInfix(var i: Int){
    infix fun banana(i:Int){
        println("infix banana ${this.i} and $i")
    }
}
infix fun TipsInfix.poi(i:Int){
    println("infix poi ${this.i} and $i")
}

fun main(args: Array<String>) {
    var i = TipsInfix(12)
    i banana 450
    i poi 12
}
```

P.6 类型别名 TipsTypealias

- 功能: 给类起别名
- 介绍:
    - typealias能够给类取别名.
    - 实际使用时类型仍是原类型,不会创造新的类型.
    - 注意: 无论如何,在起别名的时候一定要写好注释,不然忘了就很容易食翔.

```
typealias StrArrayList = ArrayList<String>
typealias SAL = StrArrayList


fun main(args: Array<String>) {
    var sal = SAL()
    sal.add("banana")
    sal.add("Banana")
    var strArrayList = StrArrayList()
    strArrayList.add("str")
    strArrayList.add("Str")
    println("The class of sal: "+sal.javaClass)
```

```
    println("The class of strArrayList: "+strArrayList.javaClass)
}
```

P.7 关键字this TipsThis

- 功能: 用this指代类
- 介绍:
  - 单用this可以指代当前位置所在的类
  - 在嵌套类时指代最近一层的嵌套类
  - 扩展函数中的this指代被扩展的类
  - this后面也可以通过加标签来指明指代的类

```
class Outside{
    var inside = Inside()
    inner class Inside{
        fun function(){
            var outside = this@Outside
            var inside = this@Inside
            var func = this
            println(outside.javaClass)
            println(inside.javaClass)
            println(func.javaClass)
        }
    }
}
fun Outside.func(){
    println(this.javaClass)
}

fun main(args: Array<String>) {
    var This = Outside()
    This.Inside().function()
    This.func()
}
```

P.8 闭包 TipsClosure

- 功能: 用闭包沟通函数内外
- 介绍:
  - 闭包最简单的一个例子就是"函数返回一个函数"
  - 通过这种方法,闭包可以获取函数的运行环境以及属性.

```
fun TipsClosure(init: Int): (Int)->Unit{
    var i = init
    return fun(add: Int){
        i+=add
        println("here the i is: "+i)
    }
}

fun main(args: Array<String>) {
    val add = TipsClosure(42)
    add(6)
    add(6)
    add(-11)
    add(18)
}
```

P.9 解构 TipsComponent

- 功能: 用解构来同时创建多个量
- 介绍:
  - 用数据量可以直接为解构的量进行初始化.

```
fun main(args: Array<String>) {
    val (a,b) = tipsComponent()
    println(a.toString()+b)
}

data class TipsComponent(val res1: Int, val res2: String)
fun tipsComponent():TipsComponent{
    return TipsComponent(23,"Banana")
}
```

P.10 空安全 TipsNull

- 功能: 介绍kotlin的空安全
- 介绍:
  - 声明量时在类型后加问号既是声明可空量(String?)
  - 安全调用可空量一般有三种方法:
    - 用?.调用是安全调用,在被调用的量为空时也返回null
    - 用!!.调用的是假定非空调用,在被调用的量为空时会发起空指针异常(NPE)
    - 或者直接用条件判断(if-else)来事先决定好空量的处理方案
  - 在转换类型的时候用as?可以进行安全转换,在转换失败时返回null而不是跳异常

```
fun main(args: Array<String>) {
    val a = "banana"
    val b: String? = null
    println(a)
    println(a?.length)
    println(b)
    println(b?.length)
    try{
        val c = b!!.length
    }catch (NPE: NullPointerException){
        NPE.printStackTrace()
    }
}
```

Kotlin学习（一）—— IDEAIntelliJ IDEA的安装配置及Kotlin的环境部署 - 刘桂林 320

作者：刘某人程序员博客：http://blog.csdn.net/qq_26787115声明：未经原作者允许请勿转载一.概括从这篇博客… 来自： 刘桂林

0

下载 **黑马2018最新kotlin项目实战。** 11-1

从入门到实战，全过程视频详细讲解。包含黑马外卖、手机影音、坦克大战等实战项目源码和讲解

Kotlin 从入门到实战（一） - CallMeSP的博客 1851

花了四五天的时间看完了kotlin-docs和kotlin-for-android-developers并写了一个小app实战一下。于是打算写两… 自： CallMeSP的博客

一个实例学习Kotlin 开发 Android App 的全过程（内有代码） - IT派 544

自 Google I/O 大会，Google 正式宣布 Kotlin 成为 Android 开发的官方语言的五个月以来，不少开发团队都开始使… 来自： IT派

文章热词 Electron入门 3Dmax入门学习 UG教程入门 Go语言入门学习 catia入门学习

Kotlin 入门-基本语法 - 花不掉泪的博客 3054

Kotlin 入门-基本语法kotlin 作为 java 类似的语言，学过 java 的同学，学 kotlin 应该很轻松。Package 和 Importp… 来自： 花不掉泪的博客

CoderLean 关注 172篇文章

湖前琴亭 关注 223篇文章

Q博士 关注 585篇文章

CallMeSP 关注 38篇文

Kotlin最简单的入门教程——类和继承 - 发现科技的专栏 1794

Kotlin 中类和java中的类声明都是用关键字class两者的区别主要在于构造函数：java中没有主构造函数次构造函数… 来自： 发现科技的专栏

Kotlin IntelliJ IDEA环境搭建 - Deng XianJun 449

IntelliJ IDEA 免费的社区版下载地址：https://www.jetbrains.com/idea/download/index.html 下载安装后，我们… 来自： Deng XianJun

IDEA创建Kotlin工程 - 行云间 8173

1.配置IDEA IDEA下载地址：https://www.jetbrains.com/idea/download/#section=windows 现IDEA最新的版本… 来自： 行云间

上海市辖区股王8年追涨停铁律“1272”曝光，震惊众人

陕西信息科技 · 燨燚

Kotlin搭建第一个Android程序（Intellij idea） - 在路上 5271

Kotlin 是Java语言的补充者，而非替代品，具体的对比可以参考文章：https://code.tutsplus.com/articles/java-vs… 来自： 在路上

使用IntelliJ IDEA创建基于Gradle的kotlin项目 - 熊本同学 732

一、首先打开Idea，选择Create New Project 二、选择Gradle项目并勾选 三、填写项目信息 四、然后一路默认下去… 来自： 熊本同学

使用IntelliJ IDEA创建第一个Kotlin程序 - lilongjiu的专栏 1027

使用IntelliJ IDEA创建第一个Kotlin程序 来自： lilongjiu的专栏

ASP.NET开发实例大全（基础篇）--1、目录 - jaylinhong的博客 820

ASP.NET快速入门篇 ## 1. 搭建ASP.NET开发环境 2. C#语言基础 3. 面向对象编程思想 4. ASP.NET内置对象 ## AS… 来自： jaylinhong的博客

下载 **Kotlin系统入门与进阶【完整视频+代码 推荐】 - jason_zhouz** 02-2

Kotlin系统入门与进阶 Kotlin系统入门与进阶 Kotlin系统入门与进阶

Kotlin系统入门与进阶（一） - Smile.L.B的博客 1

Kotlin中文文档：http://www.kotlindoc.cn/GettingStarted/Basic-Syntax.html https://www.kotlincn.net/docs/r... 来自： Smile.L.B的博客

下载 **Excel_VBA典型实例大全** 05-0

Excel_VBA初学者入门必读，附有经典实例供学习开发，举一反三。

0

下载 **Kotlin系统入门与进阶【百度网盘+完整视频+代码】** 07-1

第1章 课程介绍 第2章 数据类型 第3章 程序结构 第4章 面向对象 `````````````````````````` 第9章 Kotlin的应用与展望

Kotlin 系统入门到进阶 - JackChan 1933

Kotlin 系统入门到进阶 来自： JackChan

Kotlin从入门到精通 - wangning13ji的博客 1419

有空再写~~~蛤蛤蛤 wangning13ji的博客

下载 **Kotlin系统入门与进阶某课全套高清视频** 08-0

Kotlin系统入门与进阶，全套高清，快速入门kotlin的开发。

下载 **Android 60个经典例子 - Heroy非Hero** 05-0

Android基础代码，刚入门的小伙伴可以多看看

kotlin入门系列－第一个kotlin的android demo - BlackSmith 2217

google I/O后浏览新闻知道了ｋｏｔｌｉｎ已经是Ａｎｄｒｏｉｄ开发第一语言，很想试试这个新的语言开发有什… 来自： BlackSmith

Kotlin入门指南 - zh175578809的博客 756

从本篇博客开始，将介绍Kotlin官方文档的内容。本文翻译自Kotlin官方学习文档：https://kotlinlang.org/docs/ko… 来自： zh175578809的博客

下载 **C#开发实例大全（基础卷）完整版 - 遇水寒** 06-1

C#开发实例大全（基础卷）PDF完整版，只供学习交流，请下载后24小时内删除，如若侵权，立即删除。

下载 **PHP开发实例大全(基础版) - Suva667** 08-1

PHP开发从基础知识到高级应用各个层面的大量实例及源代码，共有600个左右

下载 **《kotlin从入门到精通快速上手》PDF** 08-3

《kotlin从入门到精通快速上手》

初级学软件之C#语言 - 胡延亮

主讲内容： rn第一讲 Hello World 实例(完结)rn第二讲 面向对象的实例（完结）rn第三讲 变量（完结）rn第四讲 常量（完结）rn第五讲 数据类型（完结…

下载 **Visual Basic开发实例大全 提高卷 （完整版 带书签） - suihongtian4774** 02-0

Visual Basic开发实例大全（提高卷）》筛选、汇集了Visual Basic开发从基础知识到高级应用各个层面的大量实例及源代码，共有600个左右，每个实例及源代码按实例说明、关键…

下载 **ASP.NET开发实例大全（基础卷）.pdf - 好名字占了** 12-0

ASP.NET开发实例大全（基础卷）筛选、汇集了ASP.NET开发从基础知识到高级应用各个层面约600个实例及源代码，大多数实例都按实例说明、关键技术、设计过程、详尽注释、…

Kotlin入门系列教程—Kotlin若只如初见 - 听海的博客 3

内容简介 前言 Kotlin的名字由来 Kotlin是什么 Kotlin的优势 Kotlin学习资源 总结 第一部分第二课预告... 来自： 听

下载 **ＨＦＳＳ仿真实例大全** 1

是您最好的学习例子！是您仿真的范例！共包括了２４个经典实例，如果需要还有中文说明书。

Kotlin入门系列教程—初始篇 - 听海的博客 1777

2017 Google IO 大会宣布了两项主要新闻: Google 正式宣布进入人工智能时代 Kotlin 成为 Android 官方编程语言 K… 来自： 听海的博客

0

书单丨学习Kotlin，这五本书够了 - Owen_Suen的博客 2549

Kotlin是2011年推出的全新编程语言使用Kotlin编写的程序可直接在JVM上运行谷歌已推荐Kotlin作为Android… ： Owen_Suen的博客

Kotlin系统入门与进阶 - soul梦的博客 37

第一章:课程介绍 1.什么是kotlin kotlin就是一门可以运行在Java虚拟机、Android、浏览器上的静态语言.它与Java… 来自： soul梦的博客

Kotlin总结4(完结) - TaoYuan 399

背景kotlin断断续续的学了一段时间，正好近期有个安卓项目，就直接上手了。kotlin的优势很明显，劣势也相对明… 来自： TaoYuan

下载 **Kotlin 语言官方中文文档 - 红叶岭谷** 05-2

kotlin 官方的中文文档大全

kotlin技巧和细节整理 - Aislli的博客 78

整理一些kotlin小技巧和遇到的一些从java转到kotlin需要注意的一些小细节。 1.扩展函数 用Java写android时一般… 来自： Aislli的博客

Kotlin入门学习心得 - weixin_36317299的博客 220

写在前面：作为一个安卓客户端的开发人员，现在Android的官方语言已经从java变成了kotlin。java稳稳占据了An… 来自： weixin_36317299的博客

下载 **java web开发实例大全（基础篇） - Z335134701** 07-2

java web开发实例大全（基础篇），有兴趣的朋友可以下载一下。

下载 **Kotlin程序开发入门精要-压缩包 - 刘_海洋** 02-0

Kotlin程序开发入门精要-试读样章，Kotlin程序开发入门精要-试读样章Kotlin程序开发入门精要-试读样章Kotlin程序开发入门精要-试读样章Kotlin程序开发入门精要-试读样章Kotli…

下载 **Kotlin程序开发入门精要高清版 - 狼牙_zyf** 03-3

新手入坑必备的书籍 我爱android 我爱开源 大家共同交流

C#开发实例大全 提高卷pdf - qq_38458882的博客 132

链接：https://pan.baidu.com/s/1blXwq6aFysopjyDD1iLLBg 密码：io5u 本书为完整版，以下为内容截图： … 来自： qq_38458882的博客

下载 **最新Kotlin for Android系列视频教程 - shinianrock** 01-0

最新Kotlin for Android系列视频教程 辅导机构全套教程 百度云资源

一周入门Kotlin(二) - 我知道经过这么多年IT你们很累 但我在砥砺前行.. 1249

本章主要介绍的内容有集合，运算符重载，控制流运算符等等。除了集合，你可以认为其他的内容比较杂和难运用… 来自： 我知道经过这么多年IT你…

JS基础小案例 - xlecho的博客 1857

title: JS基础小案例 date: 2015-12-19 10:28:40 categories: 前端基础总结 tags: 前端 xl_echo编辑整理，欢迎转载，… 来自： xlecho的博客

vue-cli（vue脚手架）超详细教程 - YinX 80850

都说Vue2简单上手容易，的确，看了官方文档确实觉得上手很快，除了ES6语法和webpack的配置让你感到陌… 来自

最新迅雷“应版权方要求，文件无法下载”的解决办法 - 徐奕的专栏 4

迅雷下载有的电影电视剧的时候会出现：应版权方要求，文件无法下载，或者显示迅雷任务包含违规内容 无法继续… 来自： 徐

IntelliJ IDEA IDEA 2018 激活注册码（可用） - 东陆之滇的技术博客 22436

K03CHKJCFT-eyJsaWNlbnNlSWQiOiJLMDNDSEtKQ0ZUIiwibGljZW5zZWVOYW1lIjoibNzIDEwMDEiLCJhc3NpZ… 东陆之滇的技术博客

Intellij IDEA的激活（使用破解补丁永久激活）（已更新） - shengshengshiwo 78582

你本文主要讲解使用破解补丁永久激活Intellij IDEA（到2099.12.31，时间也可以更长） 下载并安装IDEA 甩个链接… … shengshengshiwo

VSCode设置中文语言显示 - 飞扬的博客 40206

Vscode是一款开源的跨平台编辑器。默认情况下，vscode使用的语言为英文(us)，如何将其显示语言修改成中文… 来自： 飞扬的博客

IDEA类和方法注释模板设置（非常详细） - xiaoliulang0324的专栏 97240

IDEA自带的注释模板不是太好用，我本人到网上搜集了很多资料系统的整理了一下制作了一份比较完整的模板来… 来 aoliulang0324的专栏

IntelliJ IDEA全局内容搜索和替换 - 蝈蝈的博客 193780

在做项目时，有时会在整个项目里或指定文件夹下进行全局搜索和替换，这是一个很方便功能。使用方法如下：一… 来自： 蝈蝈的博客

Swagger教程二 - 愤怒的懒洋洋的博客 18042

Swagger搭建Restful接口教程二 一、前言 上一章节我们说的是swagger-ui也就是swagger1,接下来我们说的是… 来自： 愤怒的懒洋洋的博客

Ubuntu侧边栏和顶栏消失 (ps:亲测有效) - lzhiwei的博客 599

声明：未安装NVIDIA驱动， 正常时终端运行echo $TERM显示xterm，故障后显示linux， 侧边栏和顶栏消失但能进… 来自： lzhiwei的博客

Beyond Compare 4注册码|Beyond Compare 4注册激活码下载(附破解教) - weixin_39528277的博客 24799

下载来源出处：Beyond Compare 4注册码 Beyond Compare 4注册码是一款针对“Beyond Compare 4”而制作…

整理了10个干净、好用的BT、磁力链搜索网站给大家 - YXAPP的技术分享 63207

现在越来越流行在线看视频了，但是对于我得收藏癖爱好者，还是希望可以有比较好的资源网站的，尤其是种子、… 来自： YXAPP的技术分享

小米要用 AI + IoT 做年轻人的第一套智能家居 - CSDN资讯 2051

… 来自： CSDN资讯

史上最简单的 SpringCloud 教程 | 终章 - 方志朋的专栏 1200066

转载请标明出处： http://blog.csdn.net/forezp/article/details/70148833 本文出自方志朋的博客 错过了这一篇，… 来自： 方志朋的专栏

许朝军：啪啪如何解决用户的“寂寞” - duhaomin的专栏 7681

原文地址：http://tech.163.com/13/0112/14/8L1B332500094NCS.html 网易科技讯 1月12日消息，极客公园创新… 来自： duhaomin的专栏

webstorm 2018 激活破解方法大全 - 唐大帅的编程之路 649215

webstorm 作为最近最火的前端开发工具,也确实对得起那个价格,但是秉着勤俭节约的传统美德,我们肯定是能省则… 来自： 唐大帅的编程之路

CSDN真无耻，必须要登录才能看全文 - longwind09，多容寡欲，千里江河 3773

CSDN真无耻，必须要登录才能看全文–20171026 来自： longwind09，多容寡欲…

Postman 使用方法详解 - fxbin123的博客 163070

一、Postman背景介绍 用户在开发或者调试网络程序或者是网页B/S模式的程序的时候是需要一些方法来跟踪网页… 来自： fxbin123的博客

精选！15个必备的VSCode插件 - Dean 184293

Visual Studio Code 是由微软开发的一款免费、跨平台的文本编辑器。由于其卓越的性能和丰富的功能，它很快就… 来自： Dean

pyCharm最新2018激活码 - 昌昌 1172246

本教程对jetbrains全系列可用例：IDEA、WebStorm、phpstorm、clion等 因公司的需求，需要做一个爬取最近上… 来自： 昌昌

PhotoShop2018和PhotoShop2019安装与破解教程（含资源） - mieleizhi0522的博客 159616

大家有兴趣的可以加下我刚创建的一个PS学习交流的群825493164（PS摄影学习交流），里面答疑大家的问题。共… 来自： mieleizhi0522的博客

2018最新Web前端经典面试试题及答案 - wdlhao的博客 7

本篇收录了一些面试中经常会遇到的经典面试题以及自己面试过程中遇到的一些问题，并且都给出了我在网上收集… 来自： wdlha…

用一句话证明你是一名程序员——烫烫烫烫汤汤汤汤 - weixin_40876133的博客 3814

程序员的世界，给大家来普及下。当今最热门的职业属程序员莫属。互联网发展的迅速，市场的需求很大。薪资待… 来自： weixin_40876133的博客

今天凌晨，AWS一口气又双叒叕发布了N个新服务（随时更新中...) - 老孙的博客 3146

【CSDN记者美国拉斯维加斯现场报道】在北京时间今天凌晨刚刚结束的AWS re：invent 2018的Keynote上，在AW… 来自： 老孙的博客

SQL提升（二） - 愤怒的懒洋洋的博客 5389

Sql表操作提升 一、前言 Sql是最重要的关系数据库操作语言，现在基本上任何与数据库相关的操作都离不开sql… 愤怒的懒洋洋的博客

MyBatis——mapper.xml提升指南 - 愤怒的懒洋洋的博客 5483

MyBatis常见细节问题 一、前言 MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映… 愤怒的懒洋洋的博客

IntelliJ IDEA 最新注册码（截止到2019年5月4日） - 维C果糖的博客 128120

温馨提示：本教程的 GitHub 地址为「intellij-idea-tutorial」，欢迎感兴趣的童鞋Star、Fork，纠错。 IntelliJ IDEA… 来自： 维C果糖的博客

Proxyee-down的下载与安装教程 - shadandeajian的博客 114713

Proxyee-down是monkeyWie在Github上的一个开源项目，向作者致敬。 最新版的Proxyee-down为3.12（2018.1… shadandeajian的博客

office2016永久免费激活码（office2016密钥） - 老K的博客 776926

Microsoft Toolkit(Win10激活工具/Office2016激活工具) V2.6B4 绿色版人气:42008 下载 Microsoft Toolkit(Win10… 来自： 老K的博客

C++特征码定位 - woshilxq的专栏 3549

// BaseAddrTools.cpp : Defines the entry point for the DLL application. // #include #include #inc... 来自： woshilxq的专栏

Linux 学习之创建、删除文件和文件夹命令 - 大鹏小站的博客 137544

今天学习了几个命令，是创建、删除文件和文件夹的，在linux里，文件夹是目录，下面说下我学习的命令。 创建文… 来自： 大鹏小站的博客

军事理论课答案（西安交大版） - ling_wang的博客 835025

1.1 1 【单选题】我国陆地领土面积排名世界第几？（C） A、1 B、2 C、3 D、4 2 【单选题】以下哪个国家不属于… 来自： ling_wang的博客

Swagger使用指南 - sanyaoxu_3的博客 59130

1：认识SwaggerSwagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务。… 来自： sanyaoxu_3的博客

C#Winform窗口移动 - Maybe_ch的博客 15598

在我们将Winform自带的边框隐藏之后，我们需要自己编写窗口的移动。 思路就是1.获得点击左键时当前鼠标的坐… 来自： Maybe_ch的博客

windows版idea2018.2注册码激活，有效期至2099年 - s_eal的博客 52822

原文链接：https://blog.csdn.net/halen001/article/details/81137092 亲测有效 1.下载破解补丁：百度网盘：http… 来自： s_eal的博客

FFmpeg详解及常用命令使用 - qq_26464039的博客 5283

FFMPEG简介 FFMPEG堪称自由软件中最完备的一套多媒体支持库，它几乎实现了所有当下常见的数据封装格式、… 来自： qq_26464039的博客

你见过出身最奇特的码农是怎样的？ - qq_28263265的博客 4661

01 — 知乎上有这么一个问题：你见过出身最奇特的码农是什么样的？ 话题一出，引发了人民群众热烈的讨论。 网… 来自： qq_28263265的博客

SQL提升（一） - 愤怒的懒洋洋的博客 5361

Sql不常见关键字提升 一、前言 Sql是最重要的关系数据库操作语言，现在基本上任何与数据库相关的操作都离不开… 来自： 愤怒的懒洋洋的博客

最新最详细的黑苹果10.13.4安装教程 - qq_28735663的博客 144863

最新最全求详细的13.4的黑苹果安装教程，带安装工具和镜像！安装10.13的过程中，在论坛High Sierra版能查到的… 来自： qq_28735663的博客

史上最全Java面试题（带全部答案） - 林老师带你学编程 92429

今天要谈的主题是关于求职，求职是在每个技术人员的生涯中都要经历多次。对于我们大部分人而言，在进入自己… 来自： 林老师带你学编程

Java 枚举(enum) 详解7种常见的用法 - 请叫我大师兄 449897

JDK1.5引入了新的类型——枚举。在 Java 中它虽然算个“小”功能，却给我的开发带来了“大”方便。 大师兄我… 来自： 请叫我大师兄

没有更多推荐了，返回首页

超级大黄狗Shawn 关注

原创	粉丝	喜欢	评论
33	47	13	3

等级： 博客 2　　访问： 1万+

积分： 478　　排名： 12万+

勋章：

0

最新文章

工资条: Main部分部分

工资条: EmailSender部分

工资条: OneEmailInfo部分

工资条: ReadXLSX部分

工资条: TableMaker部分

个人分类

LeetCode	7篇
bodymovin	1篇
Kotlin	1篇
Android	1篇

归档

2018年11月	6篇
2018年10月	4篇
2018年9月	4篇
2018年8月	13篇
2018年7月	6篇

热门文章

快速入门（完整）：Python实例100个（基于最新Python3.7版本）
阅读量：8981

海龟绘图:Python3.7的turtle模块
阅读量：1067

Python中的与或非以及逻辑短路
阅读量：429

随机数：Python3.7的random模块详解
阅读量：322

数学运算模块：Python3.7的math模块与cmath模块
阅读量：298

最新评论

快速入门（完整）：Python实例...

weixin_41084236：[reply]weixin_41528941[/reply] 是我疏忽,已经修正,十分感谢.

快速入门（完整）：Python实例...

weixin_41528941：第12个结果有169，因为j的范围不对，应该改成 for j in range(2,roun...

快速入门（完整）：Python实例...

qq_40374604：:)