效率的抉择：用 Kotlin 做Android开发

Bennyhuo

个人简介

- 霍丙乾，Bennyhuo，
- 就职于腾讯地图
- Github: https://github.com/enbandari
- Kotlin 微信公众号
- Kotlin 社区：https://kotliner.cn
- Kotlin 博客：https://blog.kotliner.cn

Kotlin 简介

Kotlin 的基本情况

Kotlin 的基本情况

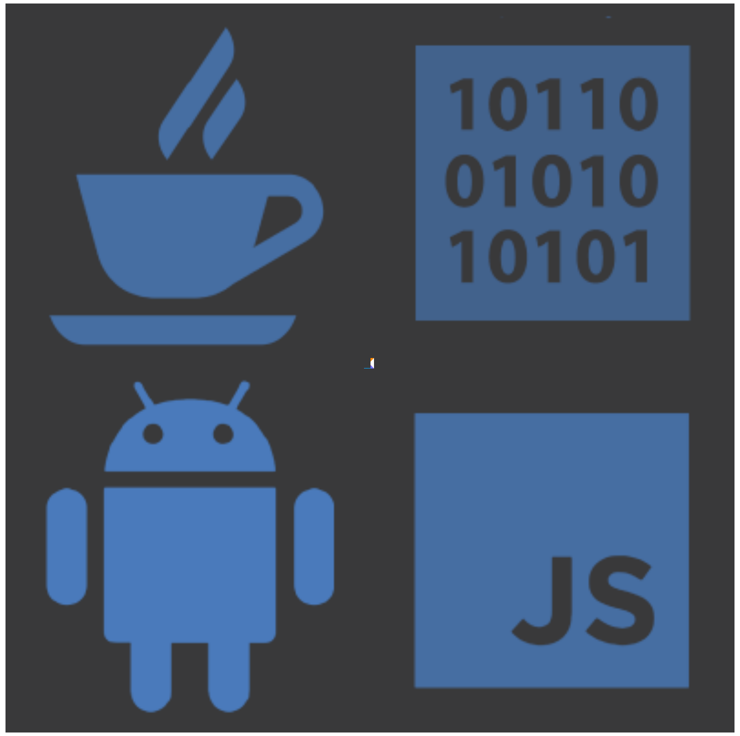
Kotlin 的基本情况

Andrey Breslav

Hadi Hariri

Kotlin 的基本情况

Kotlin 的基本情况



Kotlin 的基本情况

100% 兼容 Java

Kotlin 的基本情况

Kotlin 的基本情况

Kotlin 的基本情况

Android 官方开发语言

Kotlin 的基本情况

Kotlin 的基本情况

10110
01010
10101
JS

Kotlin 的基本情况

正式支持 JavaScript Target

Kotlin 的基本情况

Kotlin 的基本情况

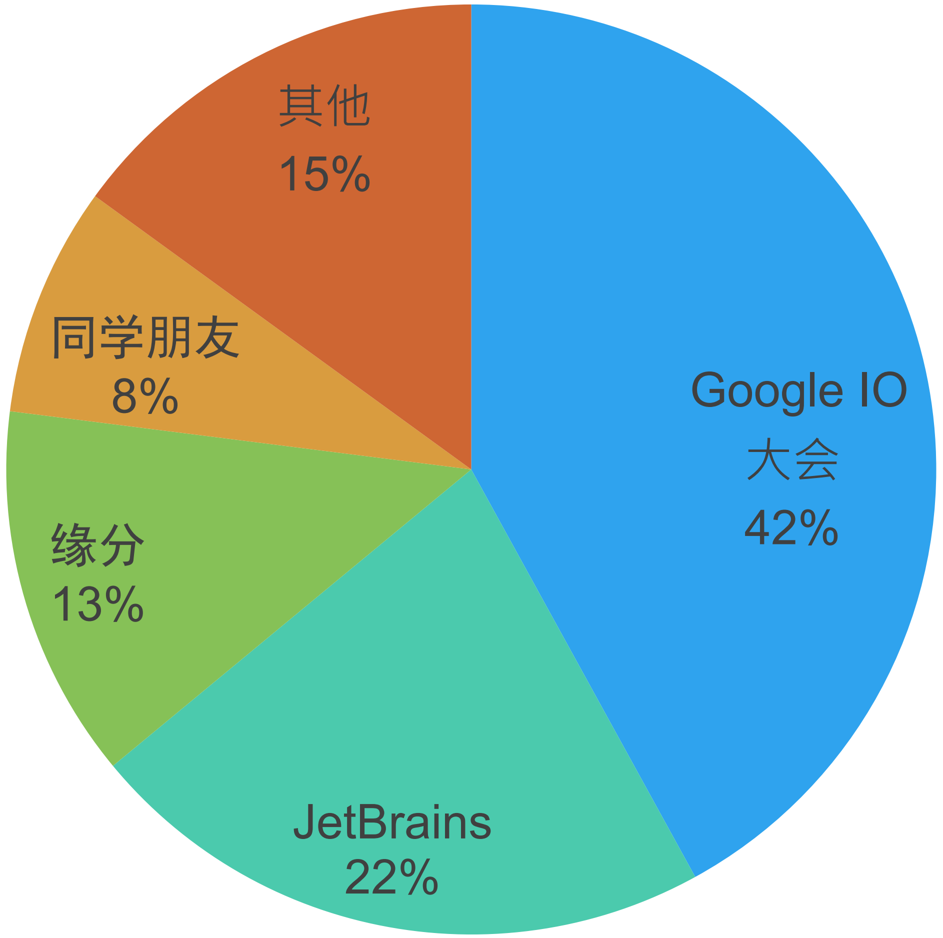
Kotlin 的基本情况

Native Target 预览版 0.4

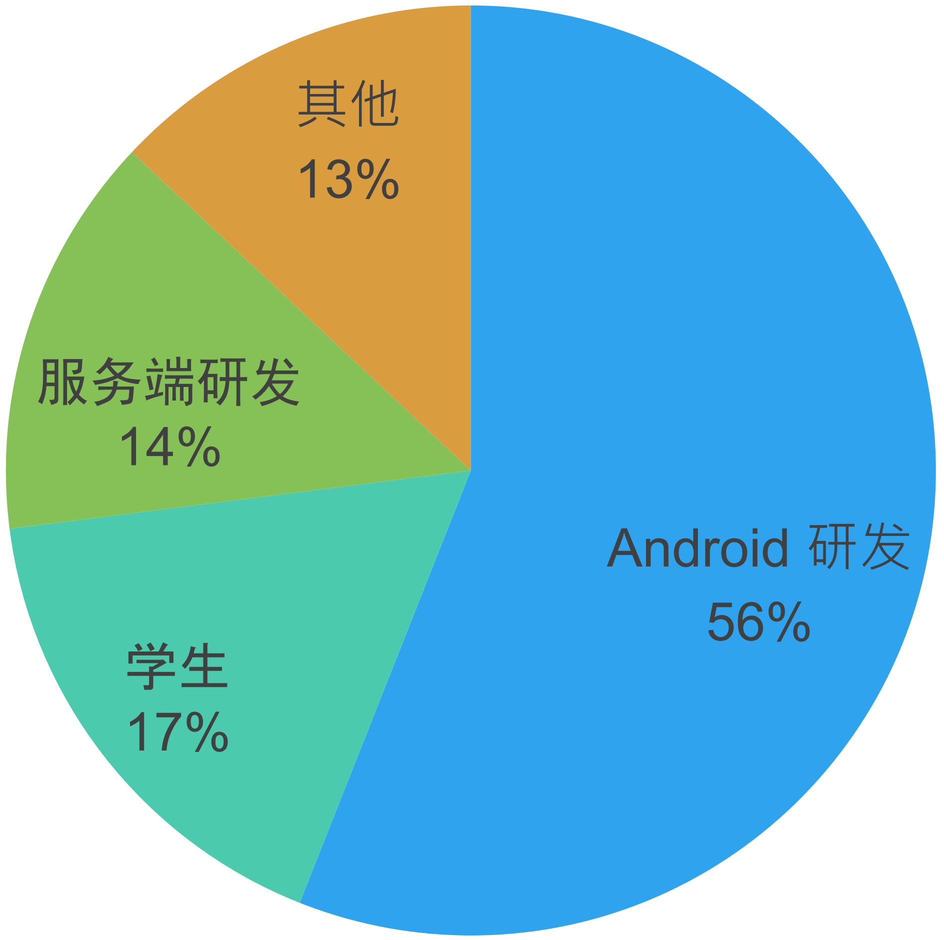
Kotlin 的基本情况

Kotlin 的基本情况

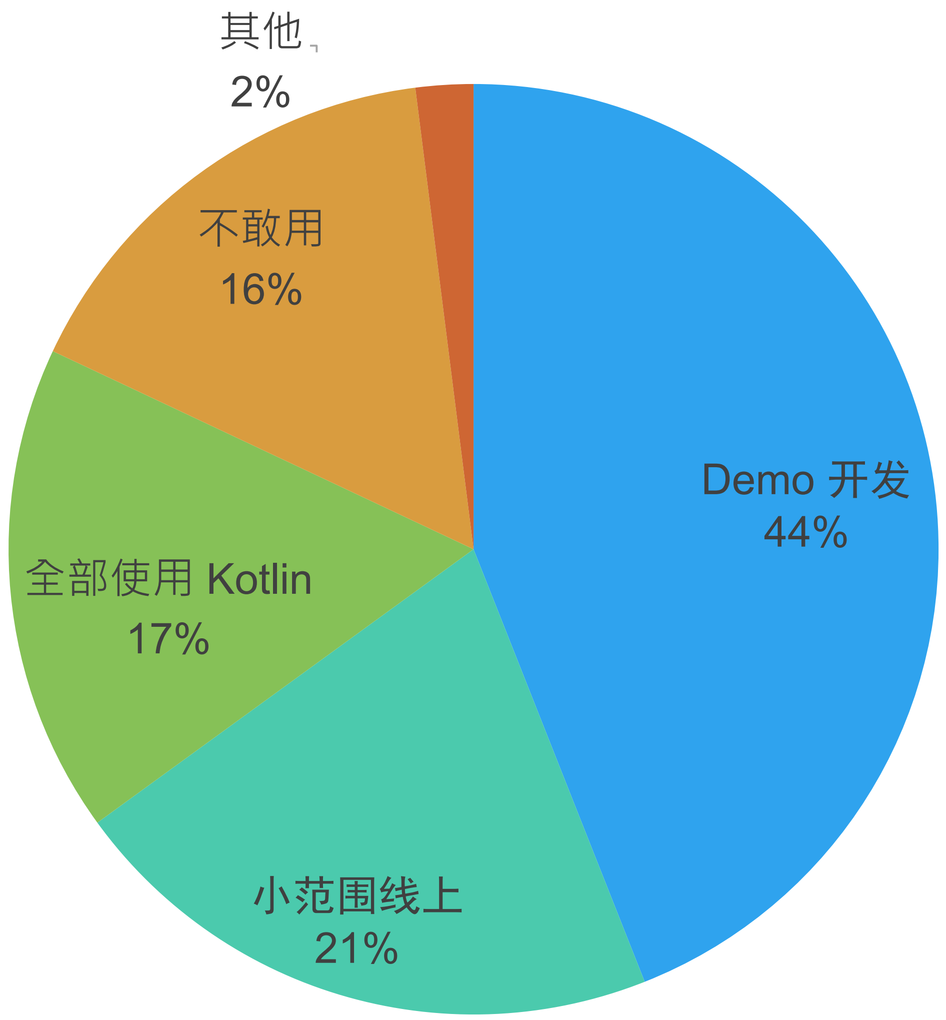
你是如何认识 Kotlin 的？



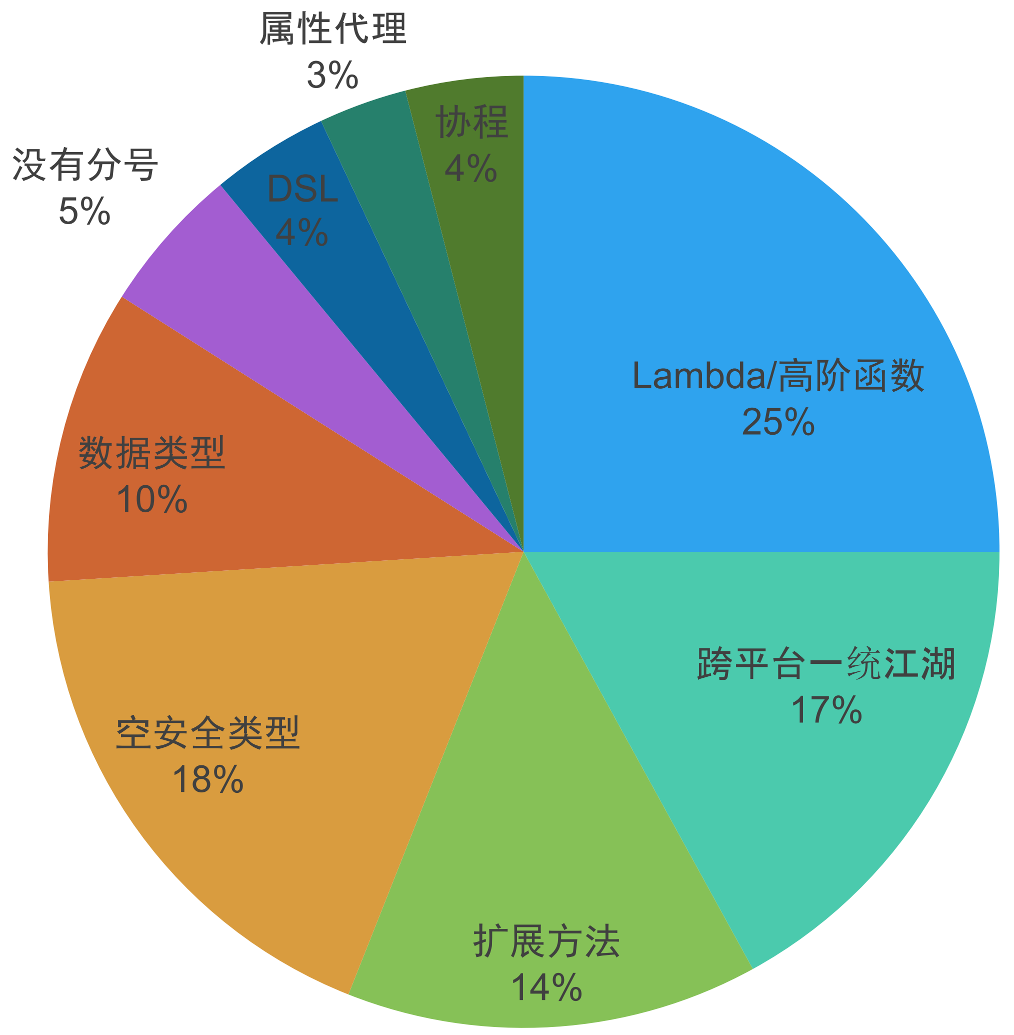
你从事何种职业？



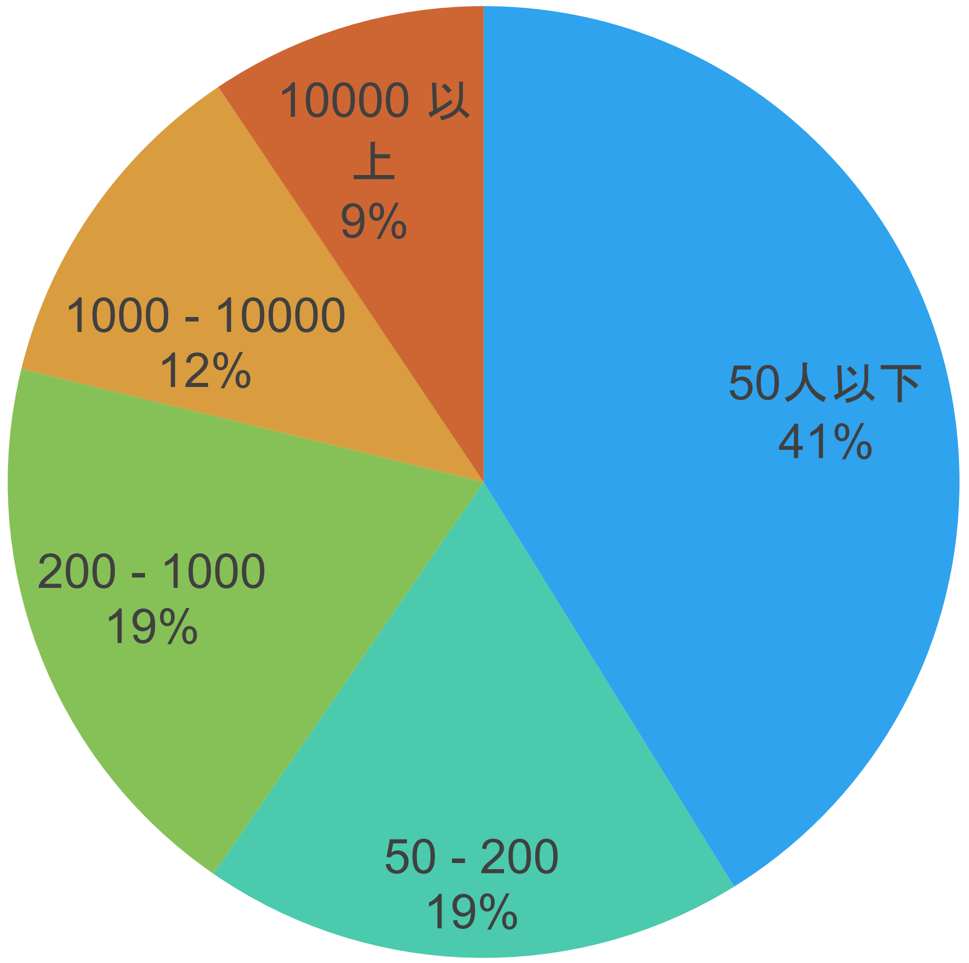
你或者你所在的项目如何使用 Kotlin 的？



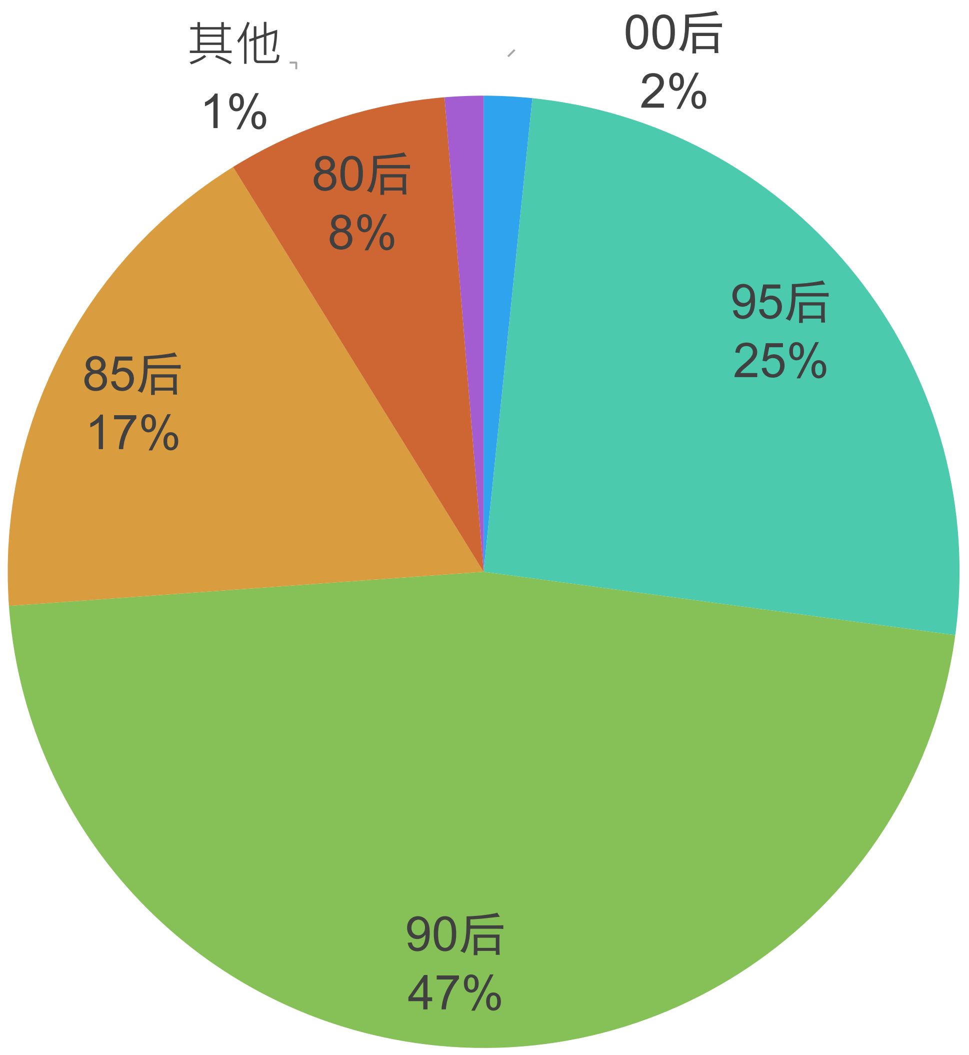
你觉得 Kotlin 最吸引你的特性是什么？



你所在的公司人数？



你的年龄范围？



Kotlin

高端大气上档次

♂	7	🐯
Gender	Age	Tiger
♏	11 8	
Scorpio	Birthday	

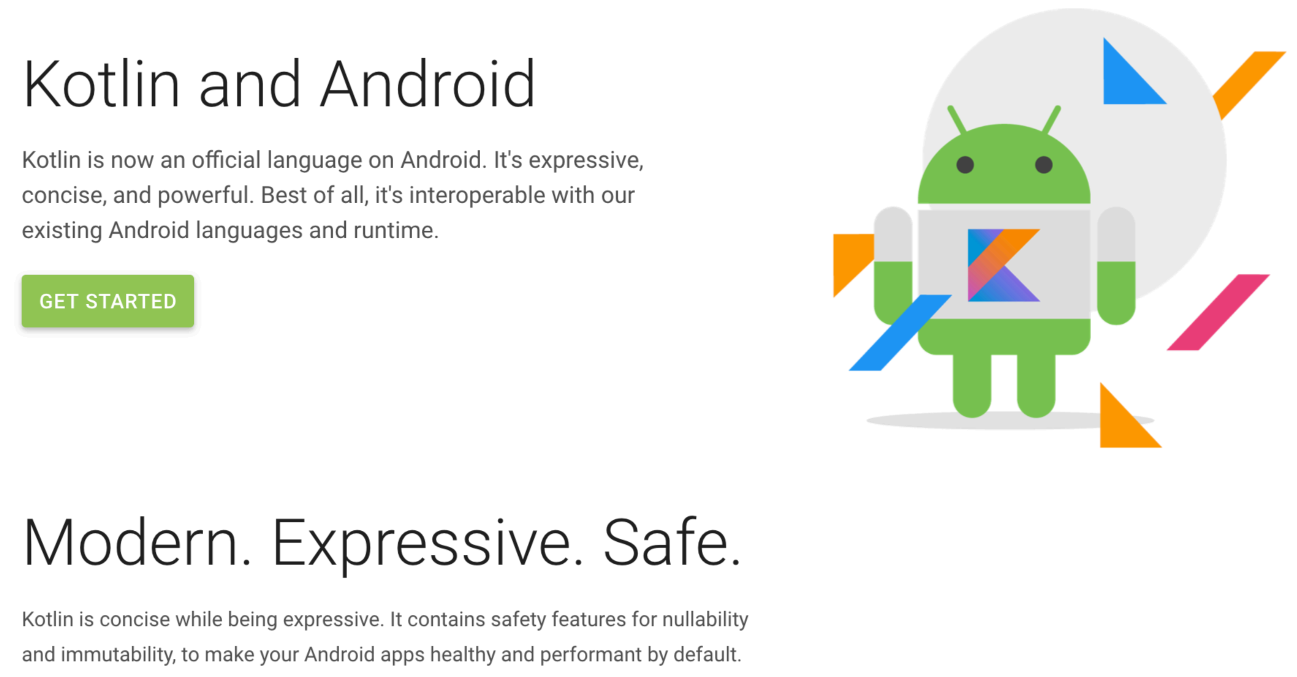
年轻 有实力 有潜力

爸爸有钱 家里有地

相关资料

Android developer

- 地址： https://developer.android.com/kotlin/index.html



Kotlin and Android

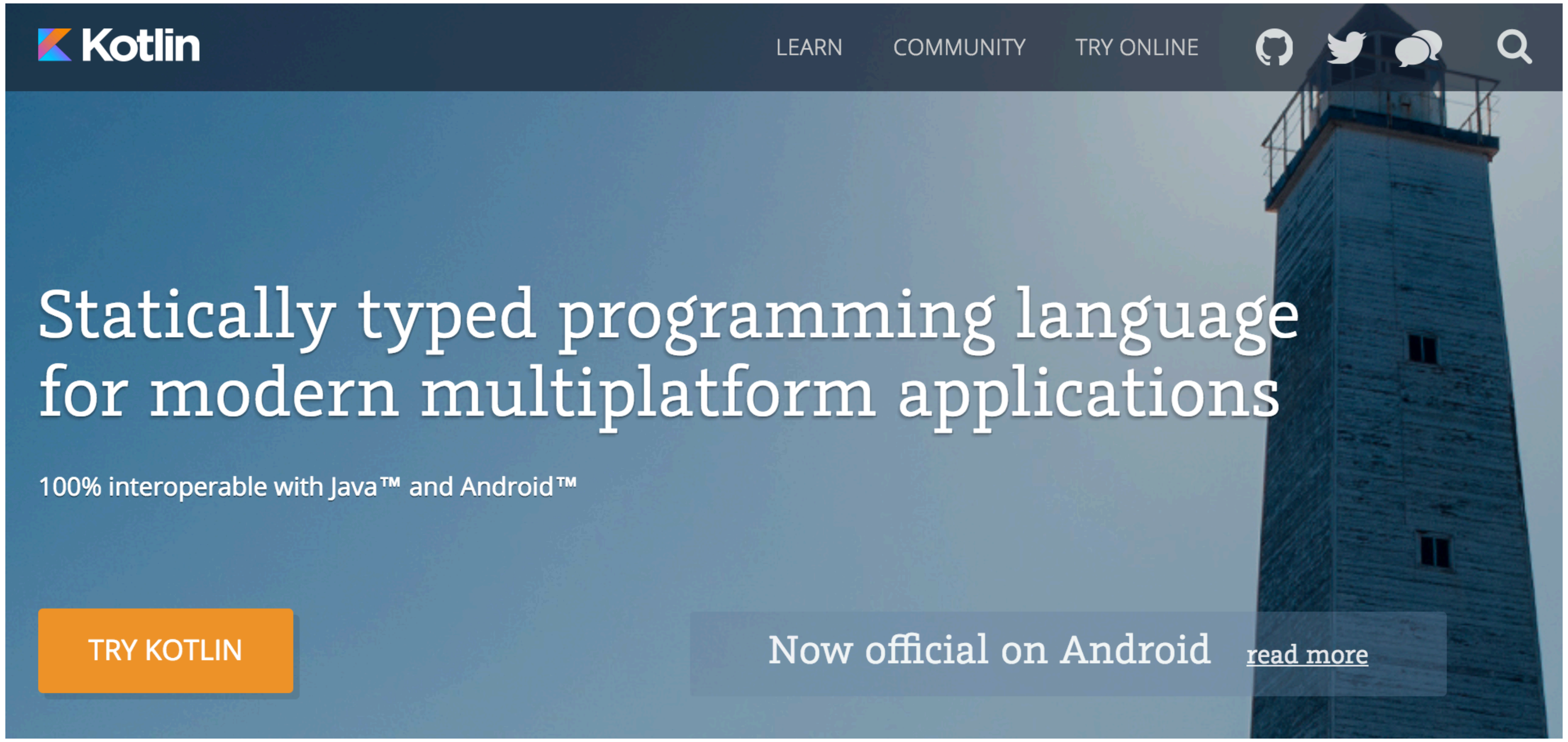
Kotlin is now an official language on Android. It's expressive, concise, and powerful. Best of all, it's interoperable with our existing Android languages and runtime.

Modern. Expressive. Safe.

Kotlin is concise while being expressive. It contains safety features for nullability and immutability, to make your Android apps healthy and performant by default.

Kotlin 官网

- 地址： http://kotlinlang.org/



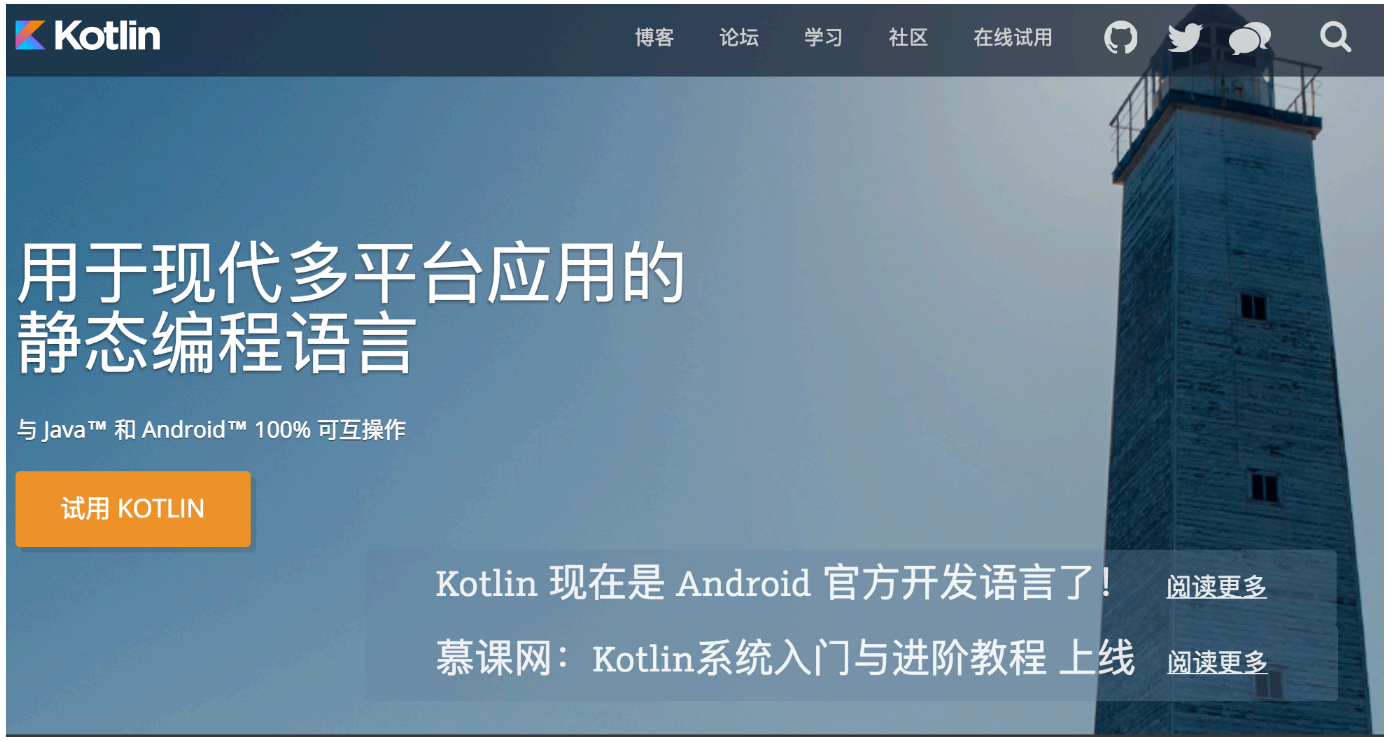
Statically typed programming language for modern multiplatform applications

100% interoperable with Java™ and Android™

Now official on Android

Kotlin 官网（中文版）

- 地址：https://www.kotlincn.net/



Kotlin 博客 论坛 学习 社区 在线试用

用于现代多平台应用的
静态编程语言

与 Java™ 和 Android™ 100% 可互操作

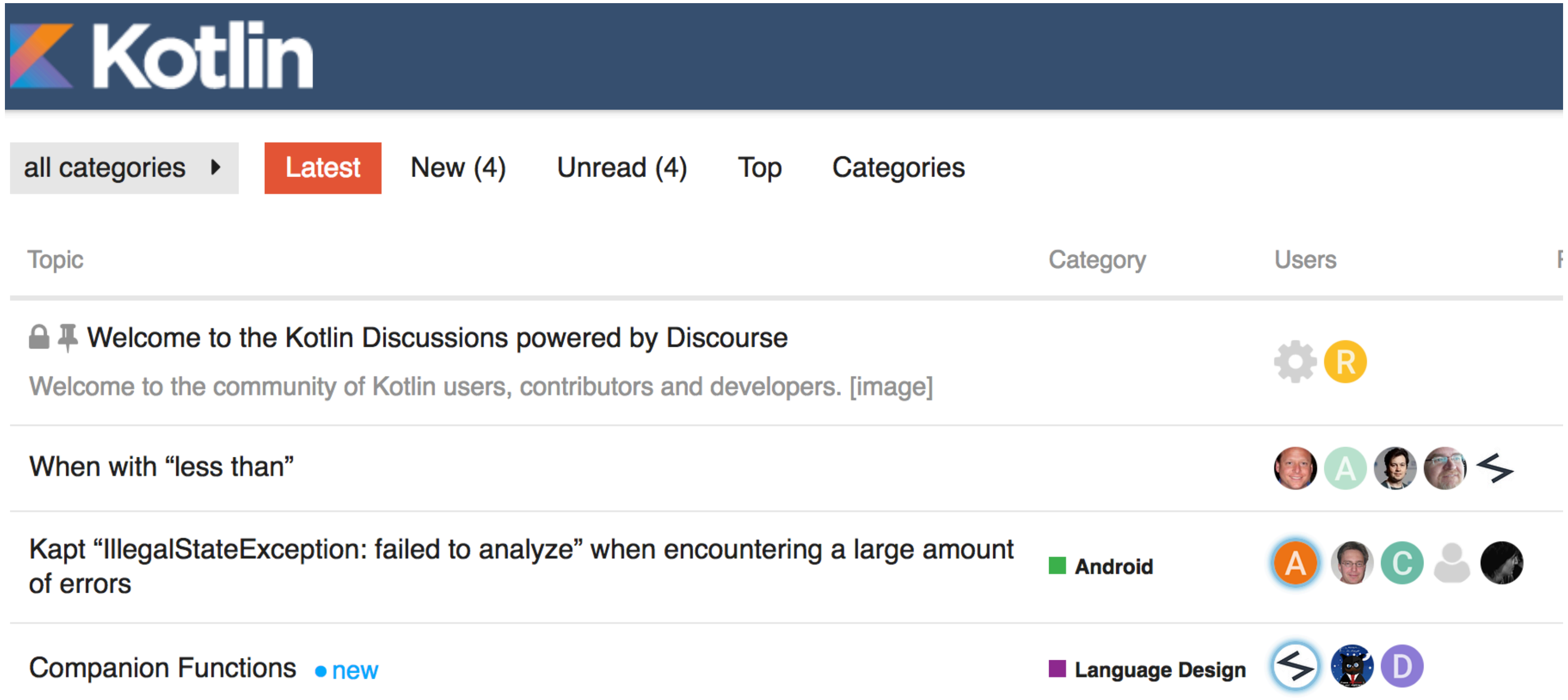
试用 KOTLIN

Kotlin 现在是 Android 官方开发语言了！ 阅读更多

慕课网：Kotlin系统入门与进阶教程 上线 阅读更多

Kotlin 论坛

- 地址：https://discuss.kotlinlang.org/



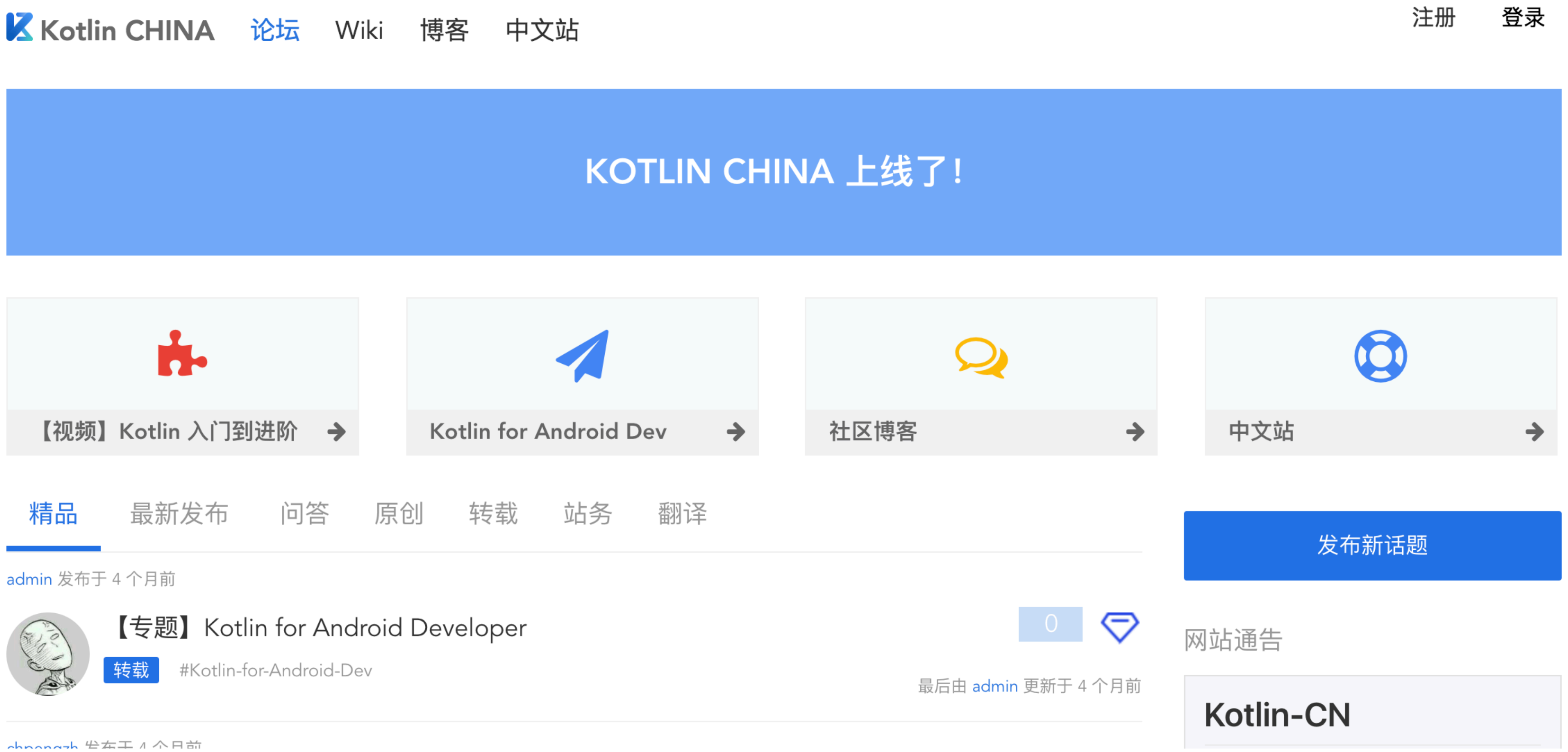
Kotlin

all categories ▸ | Latest | New (4) | Unread (4) | Top | Categories

Topic	Category	Users
Welcome to the Kotlin Discussions powered by Discourse Welcome to the community of Kotlin users, contributors and developers. [image]		
When with “less than”		
Kapt “IllegalStateException: failed to analyze” when encountering a large amount of errors	🟩 Android	
Companion Functions • new	🟪 Language Design	

Kotlin 论坛（中文版）

- 地址：https://kotliner.cn/


Kotlin CHINA 论坛 Wiki 博客 中文站 注册 登录

KOTLIN CHINA 上线了！

【视频】Kotlin 入门到进阶
Kotlin for Android Dev
社区博客
中文站

精品 最新发布 问答 原创 转载 站务 翻译

发布新话题

admin 发布于 4 个月前

【专题】Kotlin for Android Developer

转载 #Kotlin-for-Android-Dev

0

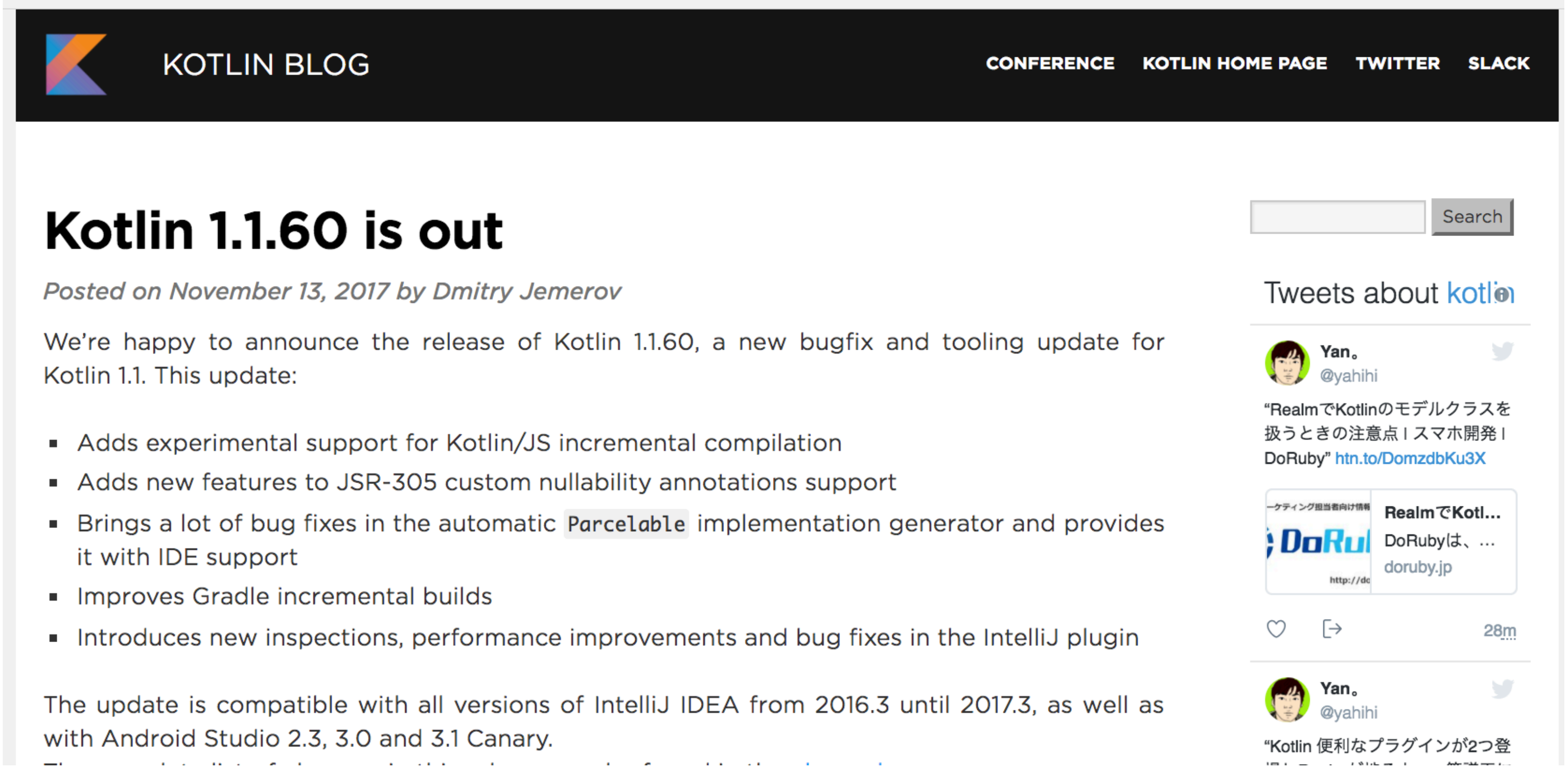
最后由 admin 更新于 4 个月前

网站通告

Kotlin-CN

Kotlin 博客

- 地址：https://blog.jetbrains.com/kotlin



Kotlin 1.1.60 is out

Posted on November 13, 2017 by Dmitry Jemerov

We're happy to announce the release of Kotlin 1.1.60, a new bugfix and tooling update for Kotlin 1.1. This update:

- Adds experimental support for Kotlin/JS incremental compilation
- Adds new features to JSR-305 custom nullability annotations support
- Brings a lot of bug fixes in the automatic `Parcelable` implementation generator and provides it with IDE support
- Improves Gradle incremental builds
- Introduces new inspections, performance improvements and bug fixes in the IntelliJ plugin

The update is compatible with all versions of IntelliJ IDEA from 2016.3 until 2017.3, as well as with Android Studio 2.3, 3.0 and 3.1 Canary.

Tweets about kotlin

Yan。 @yahihi
"RealmでKotlinのモデルクラスを扱うときの注意点 | スマホ開発 | DoRuby" htn.to/DomzdbKu3X

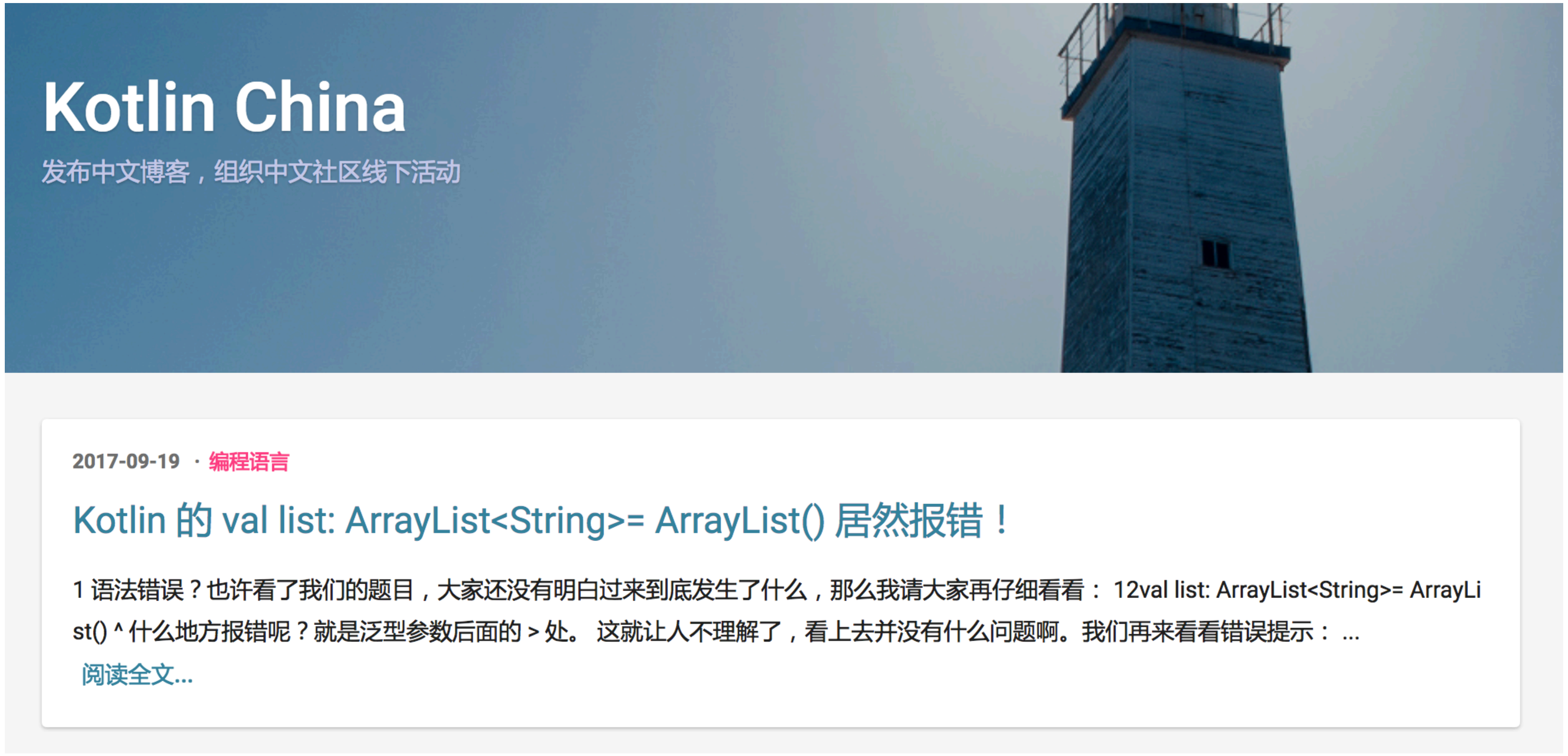
RealmでKotl... DoRubyは、... doruby.jp

28m

Yan。 @yahihi
"Kotlin 便利なプラグインが2つ登

Kotlin 博客（中文版）

- 地址：https://blog.kotliner.cn/



Kotlin China

发布中文博客，组织中文社区线下活动

2017-09-19 · 编程语言

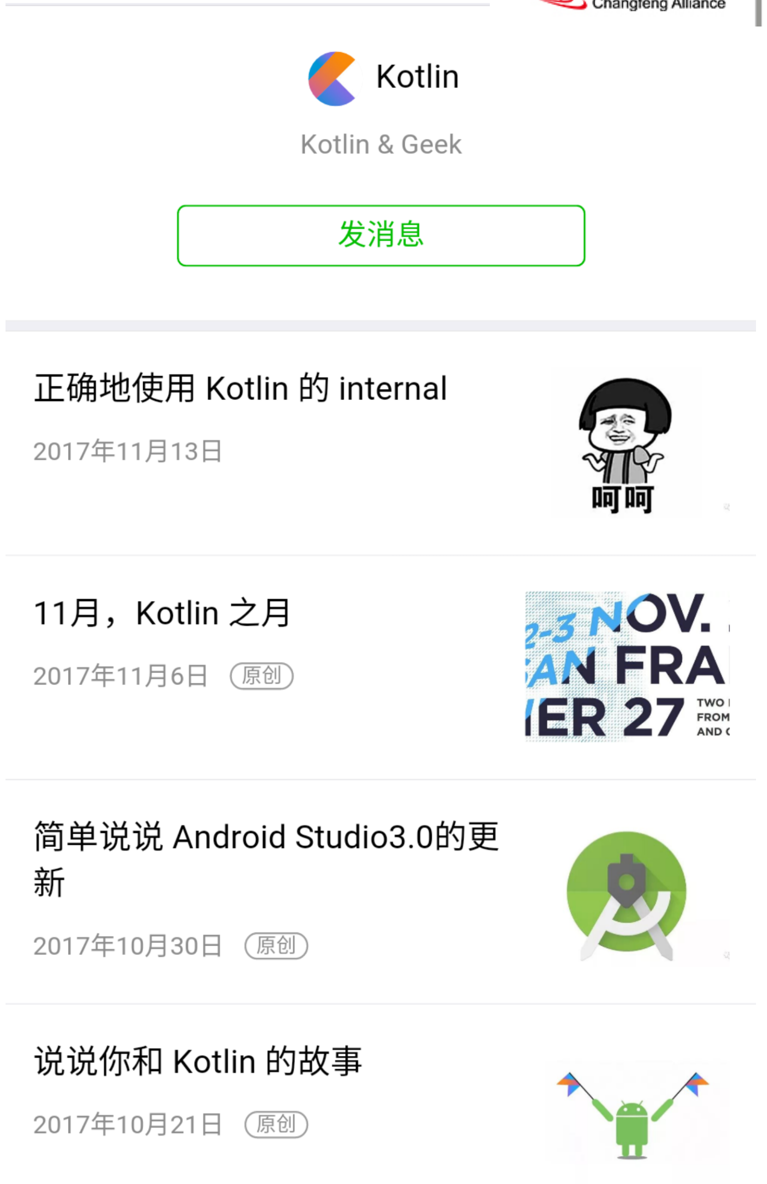
Kotlin 的 val list: ArrayList<String>= ArrayList() 居然报错！

1 语法错误？也许看了我们的题目，大家还没有明白过来到底发生了什么，那么我请大家再仔细看看： 12val list: ArrayList<String>= ArrayList() ^ 什么地方报错呢？就是泛型参数后面的 > 处。 这就让人不理解了，看上去并没有什么问题啊。我们再来看看错误提示： ...

阅读全文...

Kotlin 公众号

- 微信公众号：Kotlin



Kotlin

Kotlin & Geek

发消息

正确地使用 Kotlin 的 internal

2017年11月13日

11月，Kotlin 之月

2017年11月6日 原创

简单说说 Android Studio3.0的更新

2017年10月30日 原创

说说你和 Kotlin 的故事

2017年10月21日 原创

Kotlin 教学视频

- O'Reilly **Introduction to Kotlin Programming**：http://hadihariri.com/2016/11/01/oreilly-kotlin-course/
- O'Reilly **Advanced Kotlin Programming**：http://hadihariri.com/2016/11/01/oreilly-kotlin-course/

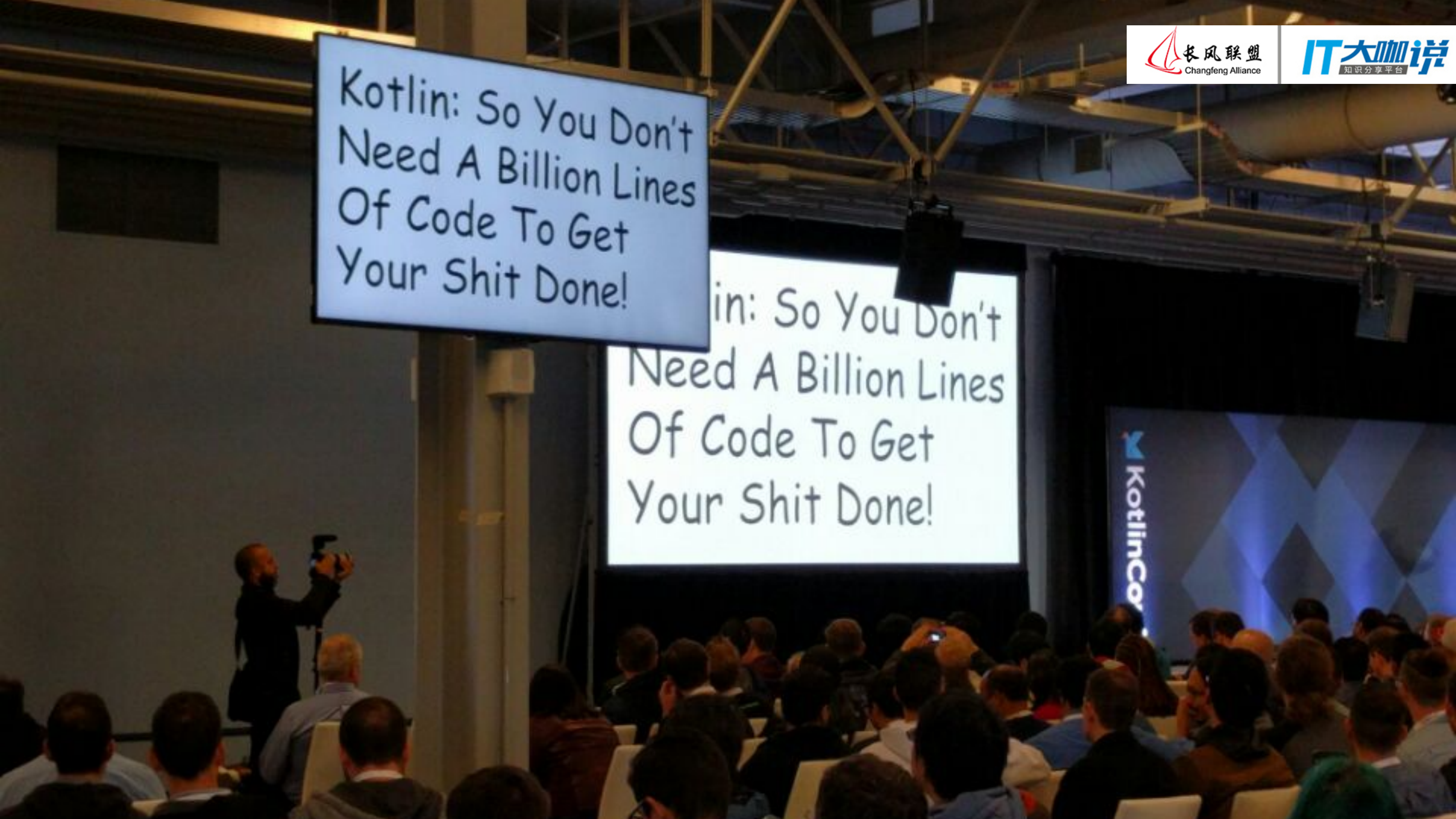
- Github **Kotlin 入门到“放弃”** ：https://github.com/enbandari/Kotlin-Tutorials
- 慕课网 **Kotlin 入门到进阶**：http://coding.imooc.com/class/108.html

培训目标

- 认识 Kotlin
- 了解 Kotlin 的基本语法
- 熟悉 Kotlin 特性的一些应用场景
- 最后把培训的内容都忘了

腾讯视频
不坏不坏 忘得真快呀

下面开始讲代码


Kotlin: So You Don't Need A Billion Lines Of Code To Get Your Shit Done!
in: So You Don't Need A Billion Lines Of Code To Get Your Shit Done!
KotlinCo

“

简洁，就要少写废话

”

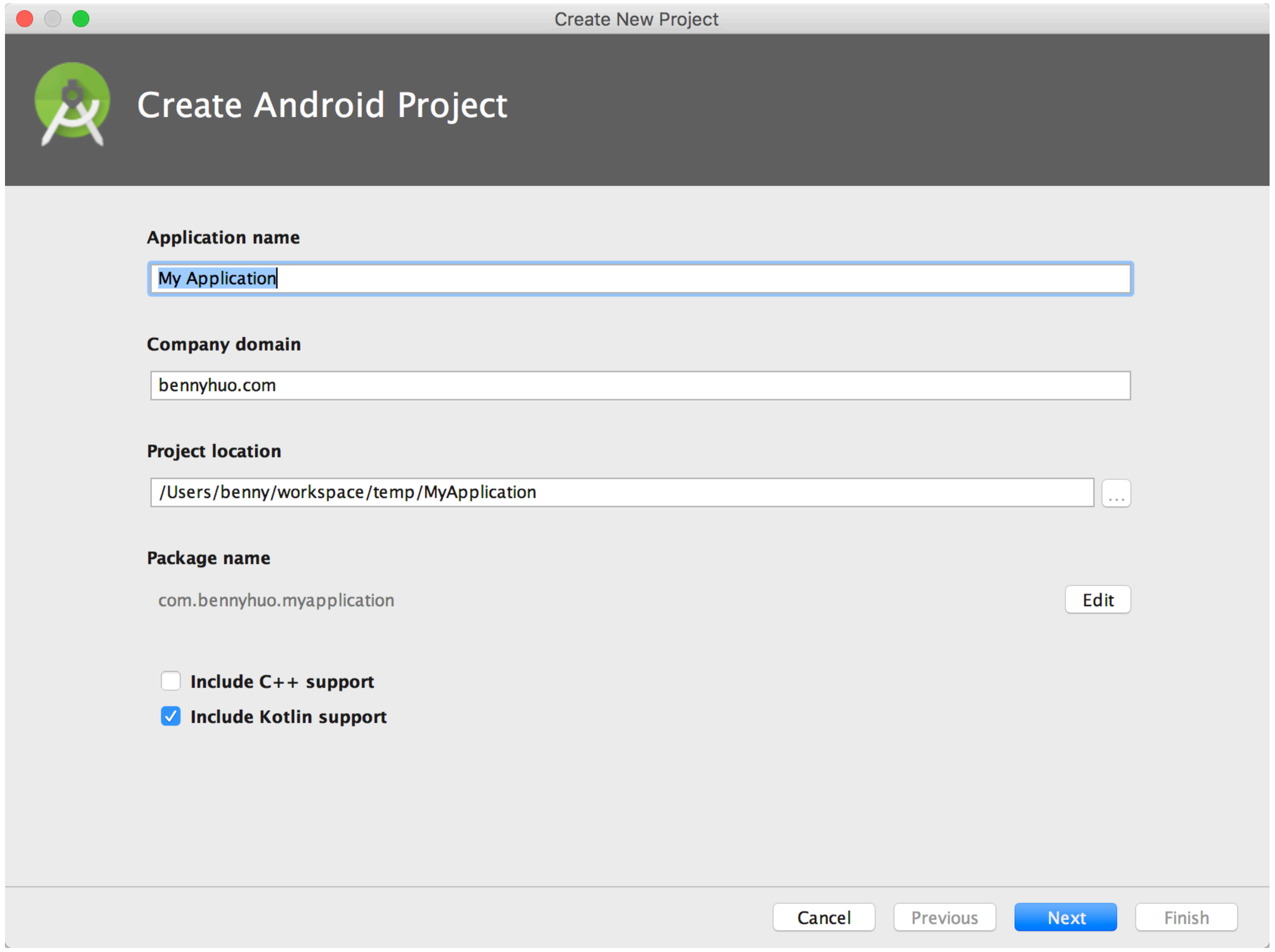
长风联盟 Changfeng Alliance | IT大咖说 知识分享平台

“

高效，就要少犯错误

”

创建工程


Create New Project

Create Android Project

Application name

My Application

Company domain

bennyhuo.com

Project location

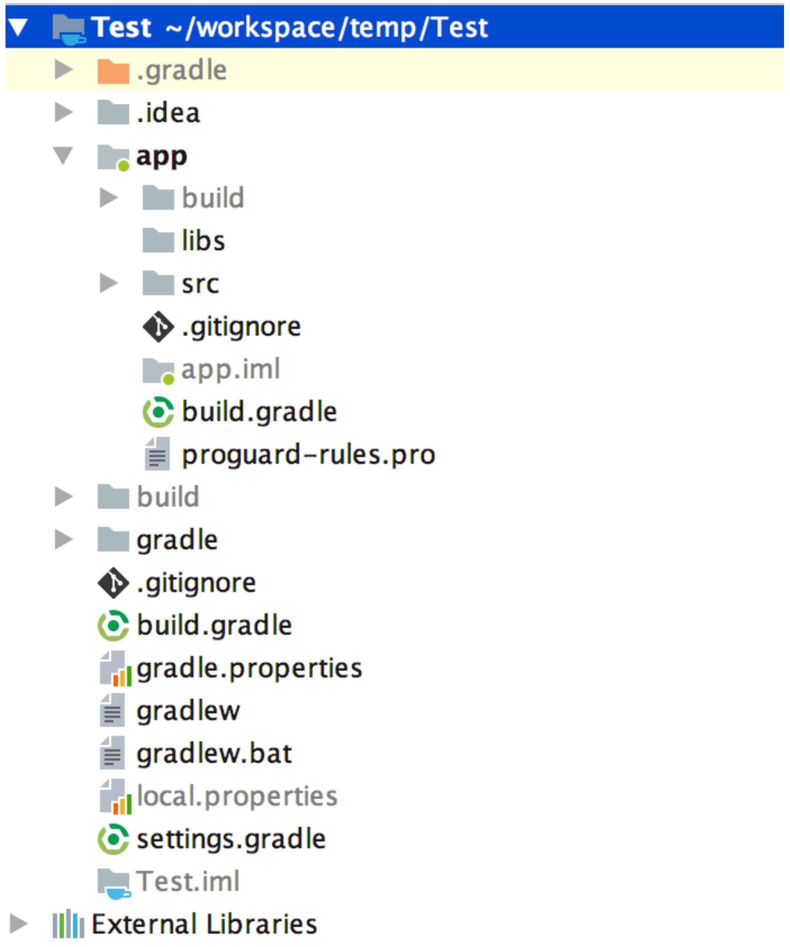
/Users/benny/workspace/temp/MyApplication

Package name

com.bennyhuo.myapplication

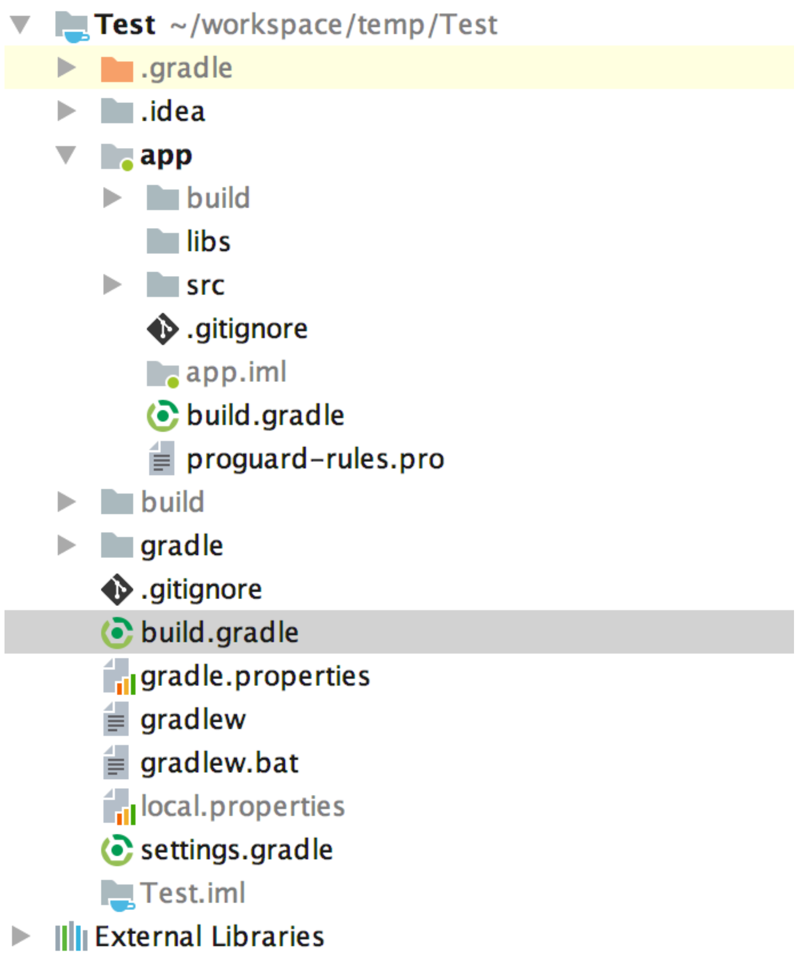
- [ ] Include C++ support
- [x] Include Kotlin support

创建工程



```
▼ Test ~/workspace/temp/Test
    ▶ .gradle
    ▶ .idea
    ▼ app
        ▶ build
          libs
        ▶ src
          .gitignore
          app.iml
          build.gradle
          proguard-rules.pro
    ▶ build
    ▶ gradle
      .gitignore
      build.gradle
      gradle.properties
      gradlew
      gradlew.bat
      local.properties
      settings.gradle
      Test.iml
▶ External Libraries
```

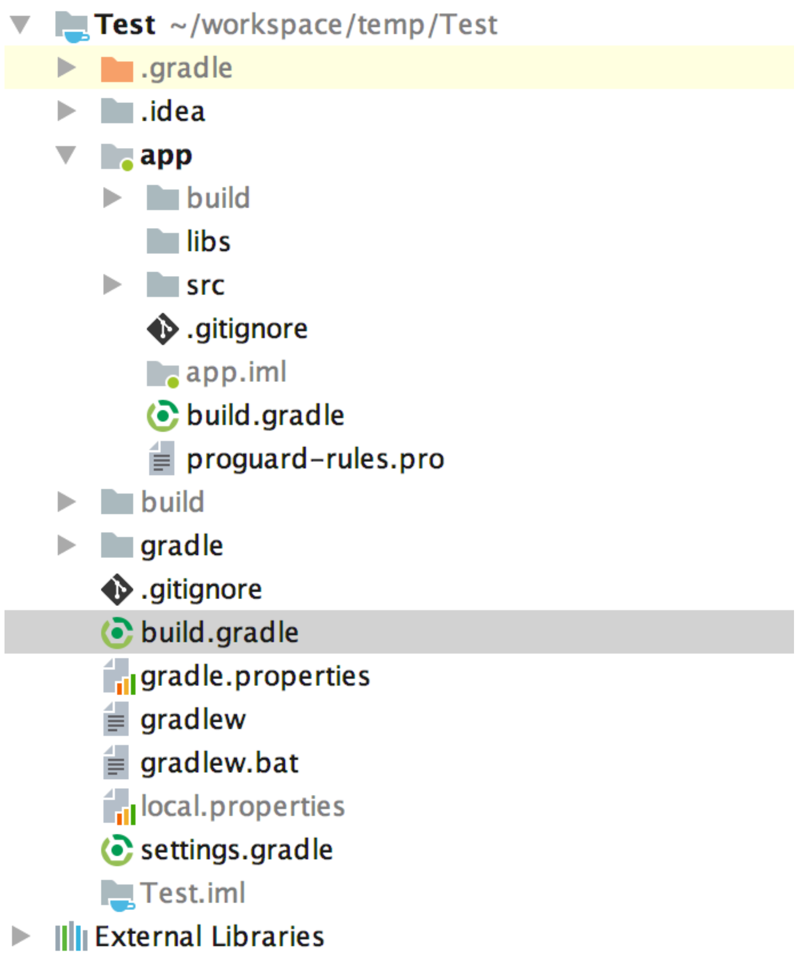
配置工程



```
Test ~/workspace/temp/Test
  .gradle
  .idea
  app
    build
    libs
    src
    .gitignore
    app.iml
    build.gradle
    proguard-rules.pro
  build
  gradle
  .gitignore
  build.gradle
  gradle.properties
  gradlew
  gradlew.bat
  local.properties
  settings.gradle
  Test.iml
External Libraries
```

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.0.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

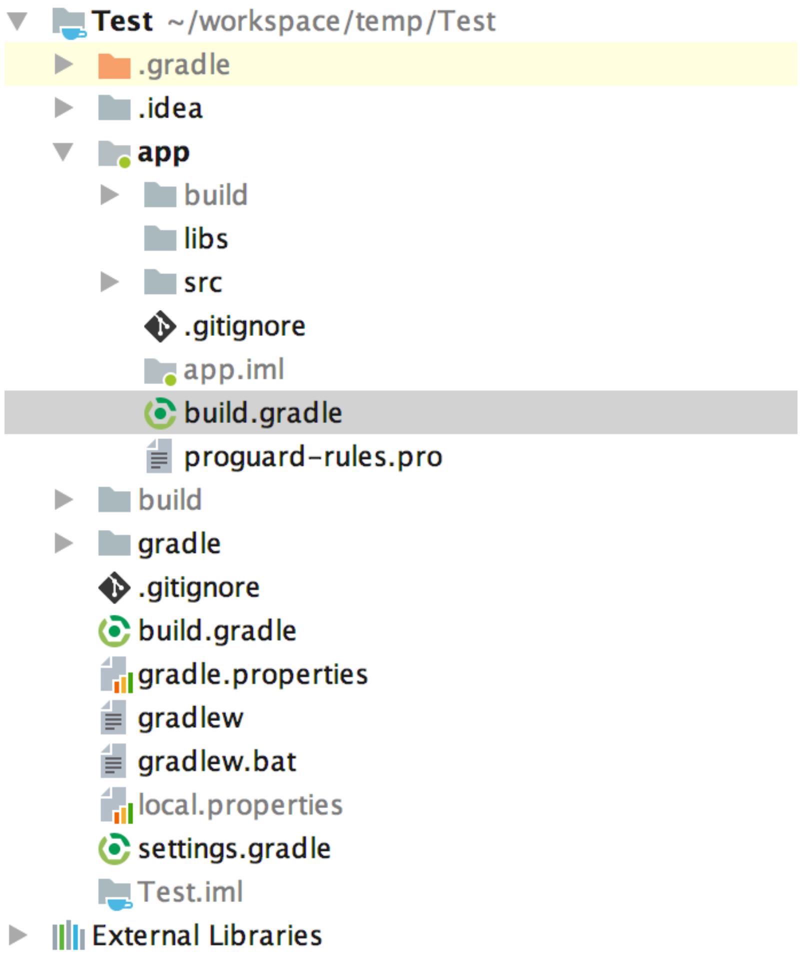
配置工程



```
Test ~/workspace/temp/Test
    .gradle
    .idea
    app
        build
        libs
        src
        .gitignore
        app.iml
        build.gradle
        proguard-rules.pro
    build
    gradle
    .gitignore
    build.gradle
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle
    Test.iml
External Libraries
```

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.0.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

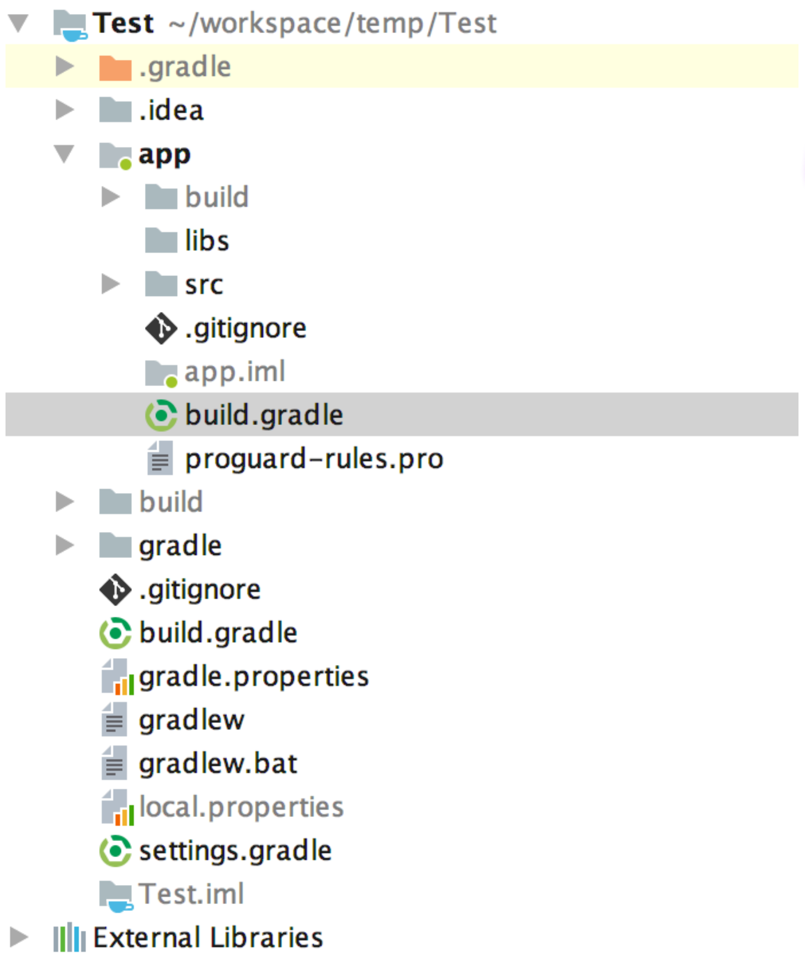
配置工程



```
▼ Test ~/workspace/temp/Test
    ▶ .gradle
    ▶ .idea
    ▼ app
        ▶ build
          libs
        ▶ src
          .gitignore
          app.iml
          build.gradle
          proguard-rules.pro
    ▶ build
    ▶ gradle
      .gitignore
      build.gradle
      gradle.properties
      gradlew
      gradlew.bat
      local.properties
      settings.gradle
      Test.iml
▶ External Libraries
```

```
apply plugin: 'com.android.application'

apply plugin: 'kotlin-android'

android {
    ...
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jre7:$kotlin_version
    ...
}
```

配置工程



```
▼ Test ~/workspace/temp/Test
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▶ build
      libs
    ▶ src
      .gitignore
      app.iml
      build.gradle
      proguard-rules.pro
  ▶ build
  ▶ gradle
    .gitignore
    build.gradle
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle
    Test.iml
▶ External Libraries
```

```
apply plugin: 'com.android.application'

apply plugin: 'kotlin-android'

android {
    ...
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jre7:$kotlin_version
    ...
}
```

如何定义变量

```
int anInt = 2;
```

如何定义变量

```
val anInt: Int = 2
```

如何定义变量

```
val anInt: Int = 2
```

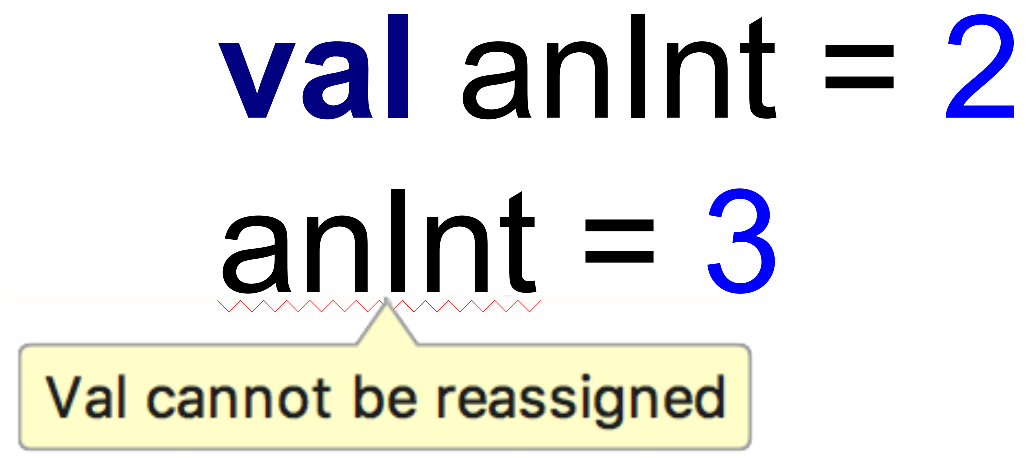
如何定义变量

```
val anInt = 2
```

如何定义变量

```
val anInt = 2
anInt = 3
```

如何定义变量



```
val anInt = 2
anInt = 3
```

Val cannot be reassigned

如何定义变量

```
var anInt = 2
anInt = 3
```

如何定义变量

```
val anInt = 2    →    final int anInt = 2;

var anInt = 2    →    int anInt = 2;
```

Kotlin　　　　　　　　Java

如何定义函数

```
void hello(int anInt){
    System.out.println(anInt);
}
```

如何定义函数

```
fun hello(anInt: Int): Unit{
    println(anInt)
}
```

如何定义函数

```
fun hello(anInt: Int): Unit{
    println(anInt)
}
```

如何定义函数

```
fun hello(anInt: Int) {
    println(anInt)
}
```

如何定义函数

```
fun hello(anInt: Int) {
    println(anInt)
}
```

如何定义函数

```
fun hello(anInt: Int) {
    println(anInt)
}
```

如何定义函数

```
fun hello(anInt: Int) {
    println(anInt)
}
```

如何定义函数

```
fun hello(anInt: Int) {
    println(anInt)
}
```

如何定义函数

```
fun hello(anInt: Int): Unit{
    println(anInt)
}
```

如何定义数组

```
int[] intArray = new int[5];

int[] anotherIntArray = {1,2, 3,4,5};
```

如何定义数组

```
val intArray = IntArray(5)
val anotherIntArray = intArrayOf(1, 2, 3, 4, 5)
```

如何定义数组

```
int[] intArray = new int[5];
```

如何定义数组

```java
int[] intArray = new int[5];

for (int i = 0; i < intArray.length; i++) {
    intArray[i] = i;
}
```

如何定义数组

```
val intArray = IntArray(5){ it }
```

如何定义数组

```
val intArray = IntArray(5){ it * 2 }
```

如何操作数组

```
intArray[1] = 2;
int element3 = intArray[3] ;
```

如何操作数组

```
intArray[1] = 2
val element3 = intArray[3]
```

运算符

intArray[1] = 2 ⟷ intArray.set(1, 2)

val element3 = intArray[3] ⟷ **val** element3 = intArray.get(3)

如果是 List、Set 呢？

运算符

```
class 逗你玩{
    operator fun set(int: Int, value: Any){}

    operator fun get(int: Int) = Unit
}
```

运算符

```
class 逗你玩{
    operator fun set(int: Int, value: Any){}

    operator fun get(int: Int) = Unit
}
```

```
val 逗你玩的实例 = 逗你玩()
逗你玩的实例[2] = 3
val x = 逗你玩的实例[1]
```

WARNING

请不要用中文编程，

不然你可能找不到工作。

数组的类型

- 基本类型的数组
  - ShortArray/IntArray/LongArray/FloatArray/DoubleArray/CharArray/BooleanArray
- 其他类型数组
  - **val** stringArray = Array(5)**{ it**.toString()**}** *// Array<String>*
  - **val** anotherStringArray = *arrayOf*(**"hello"**, **"world"**)
  - **val** strings = *arrayOfNulls*<String>(5)

数组的类型

- 基本类型的数组
  - ShortArray/IntArray/LongArray/FloatArray/DoubleArray/CharArray/BooleanArray
- 其他类型数组
  - **val** stringArray = Array(5){ **it**.toString()} *// Array<String>*
  - **val** anotherStringArray = *arrayOf*(**"hello"**, **"world"**)
  - **val** strings = *arrayOfNulls*<String>(5)

数组的类型

- 基本类型的数组
  - ShortArray/IntArray/LongArray/FloatArray/DoubleArray/CharArray/BooleanArray
- 其他类型数组
  - `val stringArray = Array(5){ it.toString()} // Array<String>`
  - `val anotherStringArray = arrayOf("hello", "world")`
  - `val strings = arrayOfNulls<String>(5)`

如何定义类

```
interface AnInterface{}

class SuperClass{
    public SuperClass() {
    }
}

class SubClass extends SuperClass implements AnInterface{
    public SubClass() {
        super();
    }
}
```

如何定义类

```
interface AnInterface

class SuperClass(){}

class SubClass : SuperClass(), AnInterface{}
```

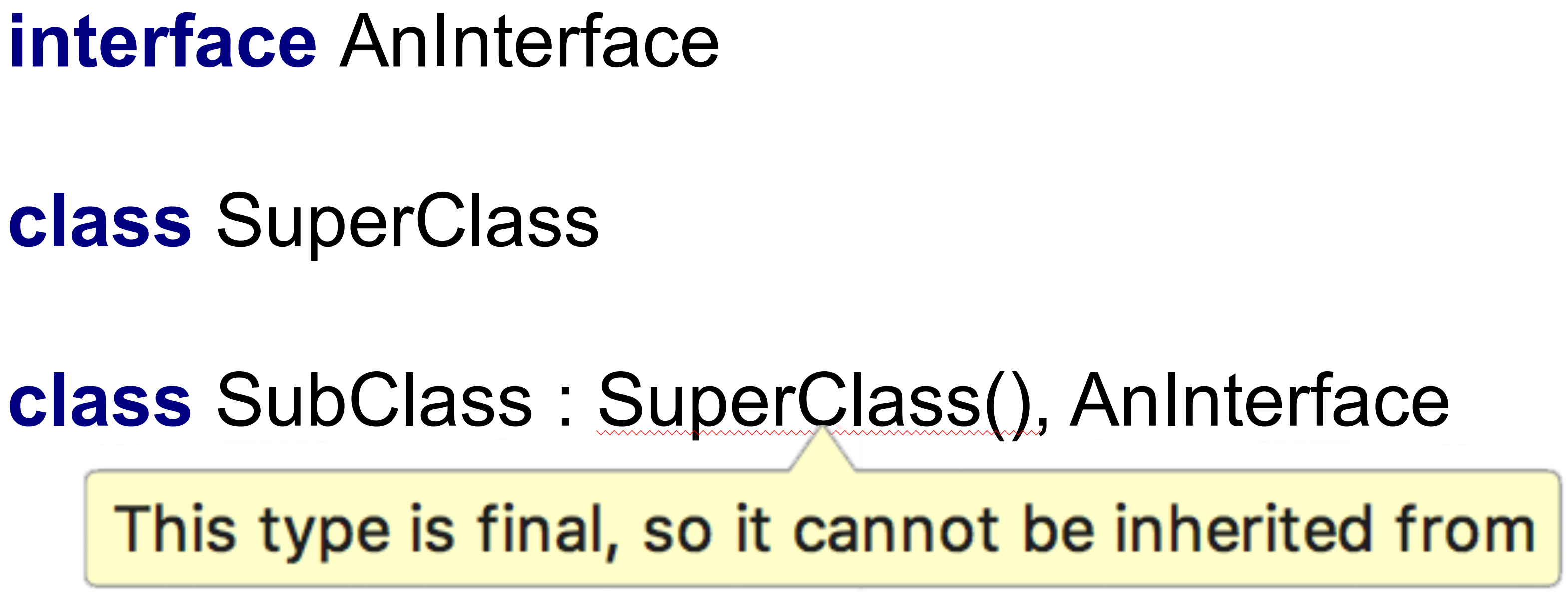
如何定义类

```
interface AnInterface

class SuperClass{}

class SubClass : SuperClass(), AnInterface{}
```

如何定义类

```
interface AnInterface

class SuperClass

class SubClass : SuperClass(), AnInterface
```

如何定义类



```
interface AnInterface

class SuperClass

class SubClass : SuperClass(), AnInterface
```

This type is final, so it cannot be inherited from

如何定义类

```
interface AnInterface

open class SuperClass

class SubClass : SuperClass(), AnInterface
```

如何定义类

```
interface AnInterface

open class SuperClass

class SubClass : SuperClass(), AnInterface
```

如何实例化类

```
SubClass subClass = new SubClass();
```

如何实例化类

```
val subClass = new SubClass()
```

如何实例化类

```
val subClass = new SubClass()
```

如何实例化类

```
val subClass = SubClass()
```

如何实例化类

```
var subClass = SubClass()
```

构造方法

```
interface AnInterface

open class SuperClass

class SubClass : SuperClass(), AnInterface
```

构造方法

```
interface AnInterface

open class SuperClass( aParam: Int )

class SubClass : SuperClass( 10 ), AnInterface
```

构造方法

```
interface AnInterface

open class SuperClass(aParam: Int) {
    init {
        println(aParam)
    }

    val aProperty = aParam
}

class SubClass : SuperClass(10), AnInterface
```

构造方法

```
interface AnInterface

open class SuperClass(aParam: Int) {
    init {
        println(aParam)
    }

    val aProperty = aParam
}


class SubClass ( aParamForSuper: Int )
             : SuperClass( aParamForSuper ), AnInterface
```

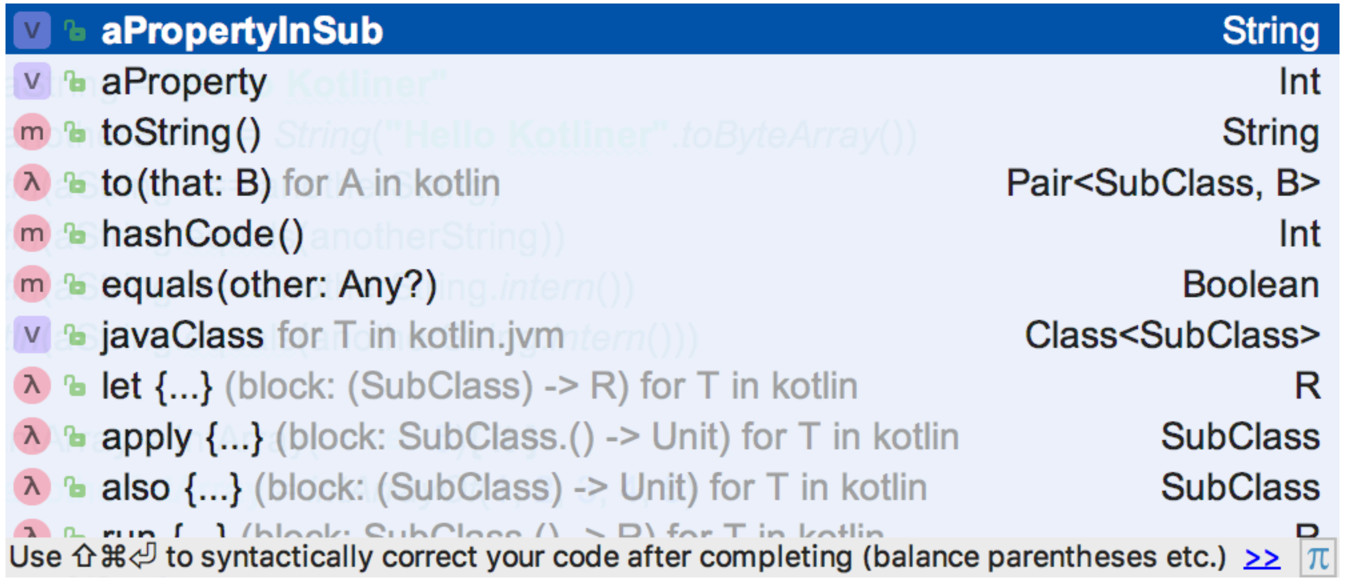
构造方法

```
class SubClass( val aPropertyInSub: String, aParamForSuper: Int )
            : SuperClass( aParamForSuper ), AnInterface
```

构造方法

```kotlin
class SubClass( val aPropertyInSub: String, aParamForSuper: Int )
          : SuperClass( aParamForSuper ), AnInterface
```

```kotlin
val subClass = SubClass("Hello", 10)

subClass.
```



```
v aPropertyInSub                                              String
v aProperty                                                   Int
m toString()                                                  String
λ to(that: B) for A in kotlin                                 Pair<SubClass, B>
m hashCode()                                                  Int
m equals(other: Any?)                                         Boolean
v javaClass for T in kotlin.jvm                               Class<SubClass>
λ let {...} (block: (SubClass) -> R) for T in kotlin          R
λ apply {...} (block: SubClass.() -> Unit) for T in kotlin    SubClass
λ also {...} (block: (SubClass) -> Unit) for T in kotlin      SubClass
Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.) >> π
```

构造方法

```
public class Call {

    private String callId;

}
```

构造方法

```
public class Call {

    private String callId;

    public Call(String callId) {
        this.callId = callId;
    }


}
```

构造方法

```java
public class Call {

    private String callId;

    public Call(){
        // 稍后再初始化 callId
    }

    public Call(String callId) {
        this.callId = callId;
    }

}
```

构造方法

```
public class Call {

    private final String callId;

    public Call(String callId) {
        this.callId = callId;
    }

}
```

构造方法

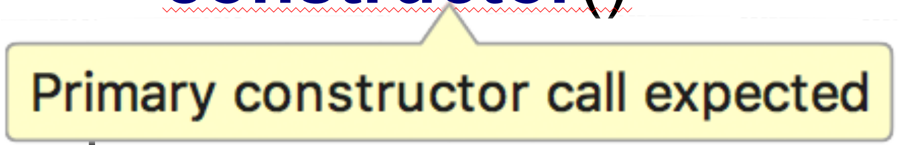
```
class Call(val callId: String)
```

构造方法

```
class Call(val callId: String){

    constructor()

}
```

构造方法

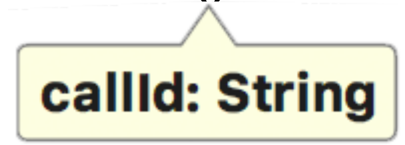
```
class Call(val callId: String){

    constructor()
}
```



Primary constructor call expected

构造方法

```
class Call(val callId: String){

    constructor(): this()

}
```


callId: String

构造方法

```
public ViewGroup(Context context) {
    this(context, null);
}

public ViewGroup(Context context, AttributeSet attrs) {
    this(context, attrs, 0);
}

public ViewGroup(Context context, AttributeSet attrs, int defStyleAttr) {
    this(context, attrs, defStyleAttr, 0);
}

public ViewGroup(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
    super(context, attrs, defStyleAttr, defStyleRes);
    ...
}
```

失败
DEFEAT

构造方法

```
public SomeViewGroup(Context context) {
    super(context);
    init();
}

public SomeViewGroup(Context context, AttributeSet attrs) {
    super(context, attrs);
    init();
}
```

构造方法

```
public SomeViewGroup(Context context) {
    super(context);
    init();
    initData();
}

public SomeViewGroup(Context context, AttributeSet attrs) {
    super(context, attrs);
    init();
}
```

构造方法

```
public SomeViewGroup(Context context) {
    super(context);
    init();
    initData();
}

public SomeViewGroup(Context context, AttributeSet attrs) {
    super(context, attrs);
    init();
    initData();
}
```

构造方法

```
public SomeViewGroup(Context context) {
    super(context);
    init();
    initData();
}
```

```
public SomeViewGroup(Context context, AttributeSet attrs) {
    super(context, attrs);
    init();
    initData();
}
```

方法重载

```
public interface List<E> extends Collection<E> {
    ...
    public E remove(int location);

    public boolean remove(Object object);
    ...
}
```

方法重载

```
public interface List<E> extends Collection<E> {
        ...
        public E remove(int location);

        public boolean remove(Object object);
        ...
}
```

```
List<Integer> integers = new ArrayList<>();

...

integers.remove(2);
```

方法重载

```
public interface List<E> extends Collection<E> {
        ...
        public E remove(int location);

        public boolean remove(Object object);
        ...
}
```

```
val integers = ArrayList<Int>()

...

integers.removeAt(2)
```

方法重载

```
public interface List<E> extends Collection<E> {
        ...
        public E remove(int location);

        public boolean remove(Object object);
        ...
}
```

```
val integers = ArrayList<Int>()

...

integers.removeAt(2)
```

默认参数

```
class SomeViewGroup(context: Context,
                    attrs: AttributeSet?,
                    defStyleAttr: Int,
                    defStyleRes: Int)
    : ViewGroup(context, attrs, defStyleAttr, defStyleRes) {

    init {
        ...
    }
}
```

默认参数

```
class SomeViewGroup(context: Context,
                    attrs: AttributeSet? = null,
                    defStyleAttr: Int = 0,
                    defStyleRes: Int = 0)
  : ViewGroup(context, attrs, defStyleAttr, defStyleRes) {

  init {
        ...
  }
}
```

覆写父类成员

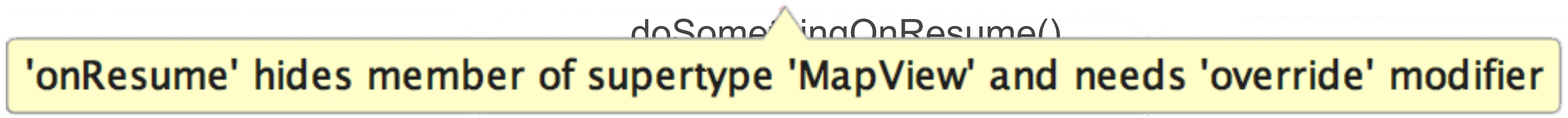
```
@Override
protected void onResume() {
    super.onResume();
    tencentMapView.onResume();
}

@Override
protected void onPause() {
    super.onPause();
    tencentMapView.onPause();
}
```

覆写父类成员

```
public class MyMapView extends MapView {
      ...
   public void onResume() {
      doSomethingOnResume();
   }

   @Override
   public void onPause() {
      super.onPause();
     doSomethingOnPause();
   }
      ...
}
```

覆写父类成员

```
class MyMapView(context: Context) : MapView(context) {

    fun onResume() {
      doSome   ingOnResume()
```



'onResume' hides member of supertype 'MapView' and needs 'override' modifier

```
    override fun onPause() {
        super.onPause()
      doSomethingOnPause()
    }
}
```

覆写父类成员

```
class MyMapView(context: Context) : MapView(context) {

    override fun onResume() {
      doSomethingOnResume()
    }

    override fun onPause() {
       super.onPause()
      doSomethingOnPause()
    }
}
```

属性代理

```
public class Link {
    private double length = -1;

    private List<LatLng> points;

    ...
}
```

属性代理

```
public void update(LatLng latLng){
    ...
    if(length == -1){
        length = resolveLength();
    }
    ...
}

private double resolveLength(){
    return ...;
}
```

属性代理

```
public void update(LatLng latLng){
    ...
    double length = getLength();
    ...
}

public double getLength(){
    if(length == -1){
        length = resolveLength();
    }
    return length;
}

private double resolveLength(){
    return ...;
}
```

属性代理

```
private val length by lazy {
    resolveLength()
}

private fun resolveLength(): Double {
    return ...
}
```

属性代理

```
private val length by lazy {
    ...
}
```

属性代理

```
private val length by lazy( SYNCHRONIZED ) {
    ...
}
```

属性代理

```
enum class LazyThreadSafetyMode {
    SYNCHRONIZED,
    PUBLICATION,
    NONE,
}
```

属性代理

```
inline operator fun <T> Lazy<T>.getValue(
        thisRef: Any?,
        property: KProperty<*>
): T = value
```

属性代理

```
private var _value: Any? = UNINITIALIZED_VALUE
override val value: T
    get() {
        if (_value === UNINITIALIZED_VALUE) {
            _value = initializer!!()
            initializer = null
        }
        return _value as T
    }
```

属性代理

```
private var _value: Any? = UNINITIALIZED_VALUE
override val value: T
    get() {
        if (_value === UNINITIALIZED_VALUE) {
            _value = initializer!!()
            initializer = null
        }
        return _value as T
    }
```

```
private val length by lazy {
    ...
}
```

属性代理

```
public interface ReadWriteProperty<in R, T> {

    public operator fun getValue(thisRef: R, property: KProperty<*>): T

    public operator fun setValue(thisRef: R, property: KProperty<*>, value: T)
}
```

属性代理

```
class PropertiesDelegate <T> (val path: String): ReadWriteProperty <Any, T> {

    val properties: Properties by lazy {
        ... // 读属性
    }

    override operator fun getValue(thisRef: Any, property: KProperty<*>): T {
        val value = properties[property.name]
        val classOfT = property.returnType.classifier as KClass<*>
        return if (Number::class.isSuperclassOf(classOfT)) {
                ... // 如果是数值需要做一些转换
        } else {
            value
        } as T
    }

    override operator fun setValue(thisRef: Any, property: KProperty<*>, value: T) {
        properties[property.name] = value
        ... // 存属性
    }
}
```

属性代理

```
class PropertiesDelegate (val path: String) {

    val properties: Properties by lazy {
        ... // 读属性
    }

    operator fun <T> getValue(thisRef: Any, property: KProperty<*>): T {
        val value = properties[property.name]
        val classOfT = property.returnType.classifier as KClass<*>
        return if (Number::class.isSuperclassOf(classOfT)) {
                ... // 如果是数值需要做一些转换
        } else {
            value
        } as T
    }

    operator fun <T> setValue(thisRef: Any, property: KProperty<*>, value: T) {
        properties[property.name] = value
        ... // 存属性
    }
}
```

属性代理

```
abstract class AbsProperties(path: String) {
    protected val prop = PropertiesDelegate(path)
}
```

属性代理

```
abstract class AbsProperties(path: String) {
    protected val prop = PropertiesDelegate(path)
}

object MetaInfo: AbsProperties("/meta.properties"){
    val version: String by prop
    val author: String by prop
    val name: String by prop
    val desc: String by prop
}
```

属性代理

```
object MetaInfo: AbsProperties("/meta.properties"){
    val version: String by prop
    val author: String by prop
    val name: String by prop
    val desc: String by prop
}
```

```
version=1.0-SNAPSHOT
author=bennyhuo@kotliner.cn
name=QCloudImageUploader
desc=方便地批量上传指定目录的图片到腾讯云的免费 50G 图床
```

属性代理

link: <https://api.github.com/user/repos?page=2>; rel="next",

<https://api.github.com/user/repos?page=5>; rel="last"

属性代理

link: <https://api.github.com/user/repos?page=2>; rel="next", <https://api.github.com/user/repos?page=5>; rel="last"

属性代理

```
map["next"] = "https://api.github.com/user/repos?page=2"
map["last"] = "https://api.github.com/user/repos?page=5"
```

属性代理

```
map["next"] = "https://api.github.com/user/repos?page=2"
map["last"] = "https://api.github.com/user/repos?page=5"
map["first"] = "https://api.github.com/user/repos?page=1"
map["prev"] = "https://api.github.com/user/repos?page=1"
```

属性代理

```
class GitHubPaging{
    val isLast: Boolean = ?
    val isFirst: Boolean = ?
    val hasPrev: Boolean = ?
    val hasNext: Boolean = ?
    operator fun get(key: String): String?{
        return ?
    }
}
```

属性代理

```
class GitHubPaging{
    val first: String? = ?
    val last: String? = ?
    val next: String? = ?
    val prev: String? = ?
    val isLast: Boolean = ?
    val isFirst: Boolean = ?
    val hasPrev: Boolean = ?
    val hasNext: Boolean = ?
}
```

属性代理

```
class GitHubPaging{
    val first by map
    ...
    val isFirst
        get() = first == null
    ...
}
```

参考 Kotlin 公众号文章：用 Map 为你的属性做代理

属性代理

```
object Settings {

    var lastPage by pref(0)

    var dayNightMode by pref(false)

    var enablePush by pref(false)
}
```

参考 Kotlin 公众号文章：用 Map 为你的属性做代理

接口代理

```kotlin
interface PageManager {
    fun showPage(clazz: KClass<out Page>)

    fun goBack()
}
```

接口代理

```
class PageManagerImpl: PageManager{
    override fun showPage(clazz: KClass<out Page>) {
        ...
    }

    override fun goBack() {
        ...
    }

    override fun dismiss() {
        ..
    }
}
```

接口代理

```kotlin
abstract class AbstractPage(val pageManager: PageManager)
    : Page, PageManager {
    override fun showPage(clazz: KClass<out Page>) {
        pageManager.showPage(clazz)
    }

    override fun goBack() {
        pageManager.goBack()
    }

    override fun dismiss() {
        pageManager.dismiss()
    }
}
```

接口代理

```
abstract class AbstractPage(val pageManager: PageManager)
    : Page, PageManager by pageManager{

}
```

接口代理

```
abstract class AbstractPage(val pageManager: PageManager)
    : Page, PageManager by pageManager
```

扩展成员

```
inline operator fun <T> Lazy<T>.getValue(
        thisRef: Any?,
        property: KProperty<*>
): T = value
```

扩展成员

```
inline operator fun <T> Lazy<T>.getValue(
        thisRef: Any?,
        property: KProperty<*>
): T = value
```

扩展成员

```
inline operator fun <T> Lazy<T>.getValue(
        thisRef: Any?,
        property: KProperty<*>
): T = value
```

扩展成员

```
inline operator fun <T> Lazy<T>.getValue(
        thisRef: Any?,
        property: KProperty<*>
): T = value
```

扩展成员

```
public class DensityUtil {

    /**
     * 根据手机的分辨率从 dp 的单位 转成为 px(像素)
     */
    public static int dip2px(float dpValue) {
        final float scale = ContextHolder.getInstance().getContext().getResources().getDisplayMetrics().density;
        return (int) (dpValue * scale + 0.5f);
    }

    /**
     * 根据手机的分辨率从 px(像素) 的单位 转成为 dp
     */
    public static int px2dip(float pxValue) {
        final float scale = ContextHolder.getInstance().getContext().getResources().getDisplayMetrics().density;
        return (int) (pxValue / scale + 0.5f);
    }
}
```

扩展成员

```kotlin
//returns dip(dp) dimension value in pixels
fun Context.dip(value: Int): Int = (value * resources.displayMetrics.density).toInt()
fun Context.dip(value: Float): Int = (value * resources.displayMetrics.density).toInt()

//return sp dimension value in pixels
fun Context.sp(value: Int): Int = (value * resources.displayMetrics.scaledDensity).toInt()
fun Context.sp(value: Float): Int = (value * resources.displayMetrics.scaledDensity).toInt()

//converts px value into dip or sp
fun Context.px2dip(px: Int): Float = px.toFloat() / resources.displayMetrics.density
fun Context.px2sp(px: Int): Float = px.toFloat() / resources.displayMetrics.scaledDensity
```

扩展成员

```
listView.scrollX = dip(2)
paint.strokeWidth = dip(2).toFloat()
paint.textSize = sp(12).toFloat()
```

扩展成员

```
logger.error("*************************")
logger.error("++++++++++++++++")
```

扩展成员

```kotlin
fun String.times(other: Int): String{
    return (1..other).fold(StringBuilder()){ acc, i ->
        acc.append(this)
        acc
    }.toString()
}
```

扩展成员

```
logger.error("****************************")
logger.error("++++++++++++++++")
```

扩展成员

```
logger.error("*" .times(8) )
logger.error("+" .times(8) )
```

扩展成员

```
fun String.times(other: Int): String{
    return (1..other).fold(StringBuilder()){ acc, i ->
        acc.append(this)
        acc
    }.toString()
}
```

扩展成员

```
operator fun String.times(other: Int): String{
    return (1..other).fold(StringBuilder()){ acc, i ->
        acc.append(this)
        acc
    }.toString()
}
```

扩展成员

```
logger.error("*" .times(8) )
logger.error("+" .times(8) )
```

扩展成员

```
logger.error("*" * 8 )
logger.error("+" * 8 )
```

运算符

a + b	↔	a.plus(b)
a - b	↔	a.minus(b)
a * b	↔	a.times(b)
a / b	↔	a.div(b)

扩展成员

```
String formattedDate
     = new SimpleDateFormat("yyyy-MM-dd")
       .format(new Date());
```

Java

扩展成员

```
new Date().format("yyyy-MM-dd")
```

Groovy

扩展成员

```
val formattedDate
    = SimpleDateFormat("yyyy-MM-dd").format(Date())
```

Kotlin

扩展成员

```
fun Date.format(format: String) : String{
    return SimpleDateFormat(format).format(this)
}
```

扩展成员

```
fun Date.format(format: String) : String
        = SimpleDateFormat(format).format(this)
```

扩展成员

```
fun Date.format(format: String)
        = SimpleDateFormat(format).format(this)
```

扩展成员

```
val formattedDate
    = SimpleDateFormat("yyyy-MM-dd").format(Date())
```

Kotlin

扩展成员

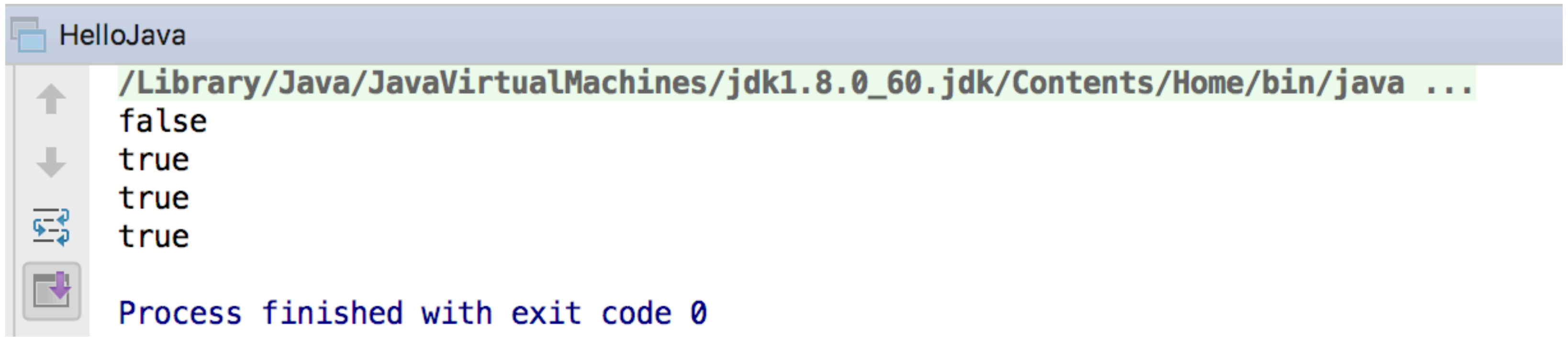
```
val formattedDate = Date().format("yyyy-MM-dd")
```

如何比较对象

```
String aString = "Hello Kotliner";
String anotherString = new String("Hello Kotliner");
System.out.println(aString == anotherString);
System.out.println(aString.equals(anotherString));
System.out.println(aString == anotherString.intern());
System.out.println(aString.equals(anotherString.intern()));
```

如何比较对象

```java
String aString = "Hello Kotliner";
String anotherString = new String("Hello Kotliner");
System.out.println(aString == anotherString);
System.out.println(aString.equals(anotherString));
System.out.println(aString == anotherString.intern());
System.out.println(aString.equals(anotherString.intern()));
```


HelloJava

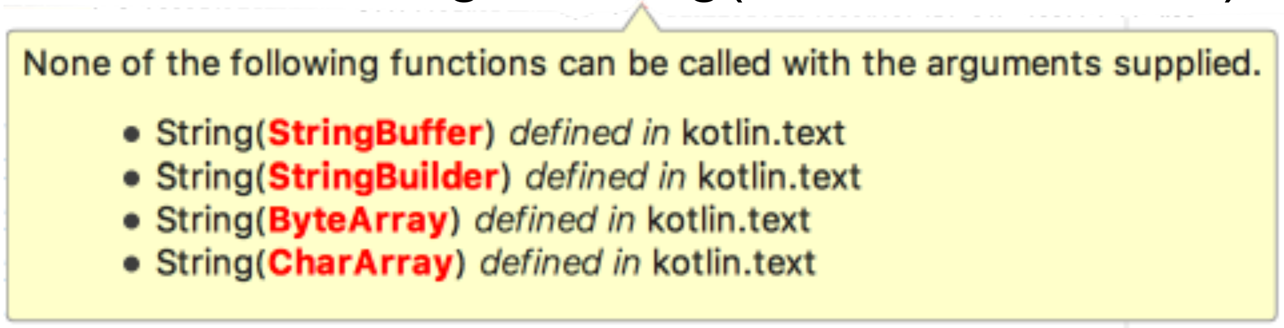
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java ...
false
true
true
true

Process finished with exit code 0
```

如何比较对象

```kotlin
val aString = "Hello Kotliner"
val anotherString = String("Hello Kotliner")
```

如何比较对象

```
val aString = "Hello Kotliner"

val anotherString = String("Hello Kotliner")
```


None of the following functions can be called with the arguments supplied.

- String(StringBuffer) *defined in* kotlin.text
- String(StringBuilder) *defined in* kotlin.text
- String(ByteArray) *defined in* kotlin.text
- String(CharArray) *defined in* kotlin.text

如何比较对象

```
val aString = "Hello Kotliner"
val anotherString = String("Hello Kotliner".toByteArray())
println(aString === anotherString)
println(aString == anotherString)
println(aString === anotherString.intern())
println(aString == anotherString.intern())
```

如何比较对象

```
val aString = "Hello Kotliner"
val anotherString = String("Hello Kotliner".toByteArray())
println(aString === anotherString)
println(aString == anotherString)
println(aString === anotherString.intern())
println(aString == anotherString.intern())
```

如何比较对象

```
val aString = "Hello Kotliner"
val anotherString = String("Hello Kotliner".toByteArray())
println(aString === anotherString)
println(aString == anotherString)
println(aString === anotherString.intern())
println(aString == anotherString.intern())
```

如何比较对象

```
val aString = "Hello Kotliner"
val anotherString = String("Hello Kotliner".toByteArray())
println(aString === anotherString)
println(aString == anotherString)
println(aString === anotherString.intern())
println(aString == anotherString.intern())
```

如何比较对象

```
val aString = "Hello Kotliner"
val anotherString = String("Hello Kotliner".toByteArray())
println(aString === anotherString)
println(aString.equals(anotherString))
println(aString === anotherString.intern())
println(aString.equals(anotherString.intern()))
```

如何比较对象

Kotlin		Java
`a == b`	➡	`a.equals(b)`
`a === b`	➡	`a == b`

如何比较对象

```
var anInt = ...
var anotherInt = ...
if(anInt < anotherInt){
    println("anInt < anotherInt")
} else {
    println("anInt > anotherInt")
}
```

如何比较对象

```
var anInt = ...
var anotherInt = ...
if(anInt .compareTo( anotherInt ) < 0){
    println("anInt < anotherInt")
} else {
    println("anInt > anotherInt")
}
```

运算符

a == b ⟺ a.equals(b)

a < b ⟺ a.compareTo(b) < 0

a > b ⟺ a.compareTo(b) > 0

a >= b 和 a <= b 呢？

数据类

```
public class Link {
    private String id;
    private double length;
    private String name;
    private String attributes;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    ... // Getters/Setters
}
```

数据类

```
class Link(var id: String,
           var length: Double,
           var name: String,
           var attributes: String)
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")

val anotherInstanceOfThisLink = link.copy()
```

数据类

```kotlin
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

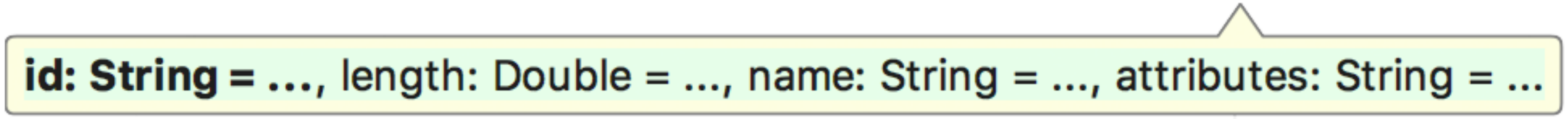
```kotlin
val link = Link("1", 326.0, "苏州街", "01|13")

val anotherInstanceOfThisLink = link.copy()
```

```java
public final Link copy(@NotNull String id,
              double length, @NotNull String name, @NotNull String attributes) {
  return new Link(id, length, name, attributes);
}
```

数据类

```kotlin
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```kotlin
val link = Link("1", 326.0, "苏州街", "01|13")

val anotherInstanceOfThisLink = link.copy()
```


```
id: String = ..., length: Double = ..., name: String = ..., attributes: String = ...
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")

val anotherInstanceOfThisLink = link.copy(name = "彩和坊路")
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")

val anotherInstanceOfThisLink = link.copy(name = "彩和坊路")

val (id, length, name, attriutes) = link
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")

val anotherInstanceOfThisLink = link.copy(name = "彩和坊路")

val (id, length, name, attributes) = link
```

数据类

```
data class Pair<out A, out B>(
        public val first: A,
        public val second: B) : Serializable
```

数据类

```
data class Triple<out A, out B, out C>(
        public val first: A,
        public val second: B,
        public val third: C) : Serializable
```

数据类

```
fun returnPair(): Pair<Double, Double>{
    return 2.0 to 3.0
}
```

数据类

```
fun returnPair(): Pair<Double, Double>{
    return 2.0.to(3.0)
}
```

数据类

```
fun returnPair(): Pair<Double, Double>{
    return 2.0.to(3.0)
}
```

```
val (first, second) = returnPair()
```

类型别名

```
public class LatLng implements Parcelable {
    public final double latitude;
    public final double longitude;
     ...
}
```

类型别名

```
data class Point(val latitude: Double, val longitude: Double)
```

类型别名

```
//data class Point(val latitude: Double, val longitude: Double)

typealias Point = LatLng
```

定义“静态成员”

```
class Statics{


}
```

定义“静态成员”

```
class Statics{
    companion object {


    }
}
```

定义 “静态成员”

```
class Statics{
    companion object {
        fun notStaticFun(){
            println("I'm not a static method! I mean it!")
        }
    }
}
```

定义“静态成员”

```
class Statics{
    companion object {
        fun notStaticFun(){
            println("I'm not a static method! I mean it!")
        }
        val `me2!!`: String = "Me either!"
    }
}
```

定义“静态成员”

```
Statics.notStaticFun()
println(Statics.`me2!!`)
```

定义“静态成员”

```
fun main(args: Array<String>) {
    Statics.notStaticFun()
    println(Statics.`me2!!`)
}
```

函数是“一等公民”

```
@Override
public void onSucess() {
    listView.post(new Runnable() {
        @Override
        public void run() {
            adapter.notifyDataSetChanged();
        }
    });
}
```

函数是 “一等公民”

```
override fun onSucess() {
    listView.post(object: Runnable{
        override fun run() {
            adapter.notifyDataSetChanged()
        }
    } )
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post( Runnable { adapter.notifyDataSetChanged() } )
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post( { adapter.notifyDataSetChanged() } )
}
```

函数是 “一等公民”

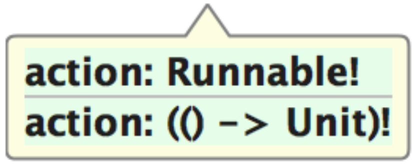
```
override fun onSucess() {
    listView.post{ adapter . notifyDataSetChanged() }
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post( adapter :: notifyDataSetChanged )
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post( adapter :: notifyDataSetChanged )
}
```



```
action: Runnable!
action: (() -> Unit)!
```

函数是“一等公民”

```
public void notifyDataSetChanged()
```

函数是 “一等公民”

```
public void notifyDataSetChanged()
```

函数是“一等公民”

```
public void notifyDataSetChanged()
```

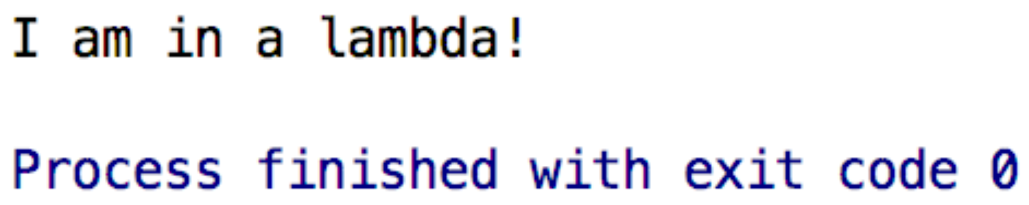
Lambda

```
val aLambda = {
    println("I am in a lambda!")
}
```

Lambda

```
val aLambda = {
    println("I am in a lambda!")
}
```

```
aLambda()
```

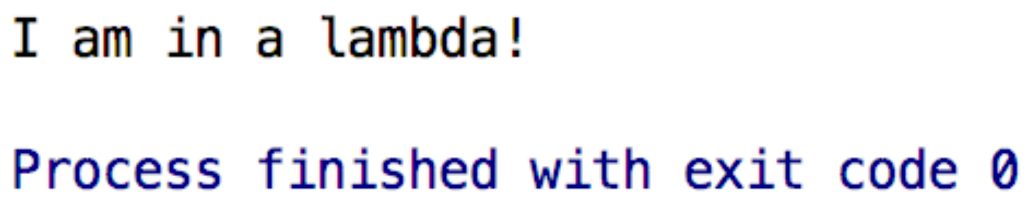


```
I am in a lambda!

Process finished with exit code 0
```

Lambda

```
val aLambda = {
    println("I am in a lambda!")
}
```

```
aLambda.invoke()
```

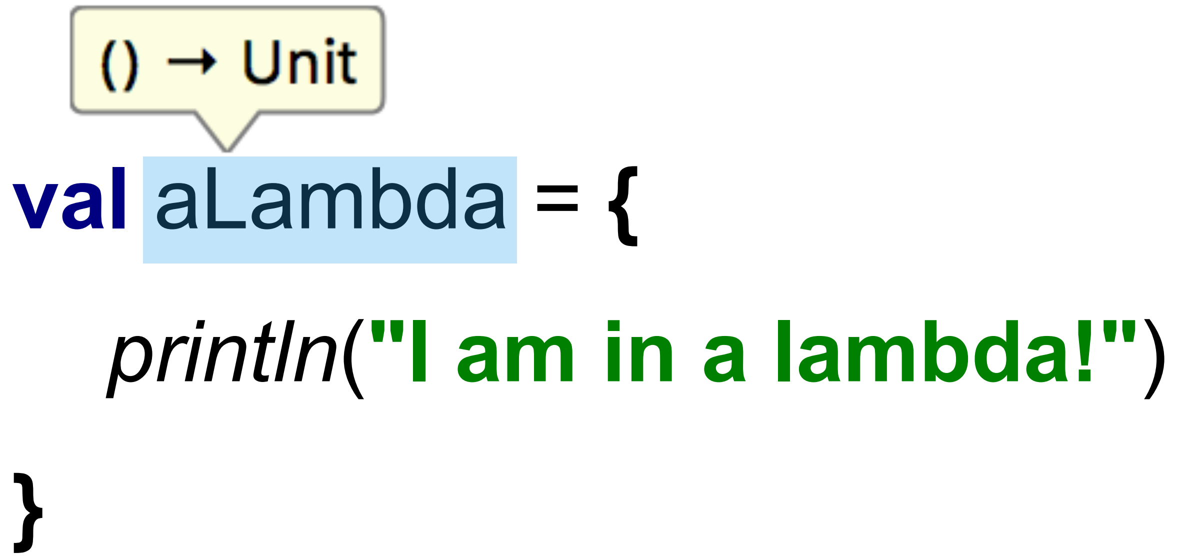


```
I am in a lambda!

Process finished with exit code 0
```

Lambda

```
val aLambda = {
    println("I am in a lambda!")
}
```

Lambda



```
() → Unit
```

```kotlin
val aLambda = {
    println("I am in a lambda!")
}
```

Lambda

(Int, Int) → Int

```
val aLambda = { left: Int, right: Int ->
    left * right
}
```

Lambda

```
val aLambda = { left: Int, right: Int ->
    left * right
}
```

```
aLambda(2, 3)
```

Lambda

```
val aLambda = { left: Int, right: Int ->
    left * right
}
```

```
aLambda(2, 3).let(::println)
```

```
6

Process finished with exit code 0
```

高阶函数

```
inline fun <T, R> T.let(block: (T) -> R): R = block(this)
```

高阶函数

```
inline fun <T, R> T.let(block: (T) -> R): R = block(this)
```

```
"Hello World".let(::println)
```

高阶函数

```
inline fun <T, R> T.let(block: (T) -> R): R = block(this)
```

```
"Hello World".let(::println)
```

高阶函数

```
inline fun <T, R> Array<out T>.map(transform: (T) -> R): List<R>
inline fun <T, R> Iterable<T>.map(transform: (T) -> R): List<R>
```

高阶函数

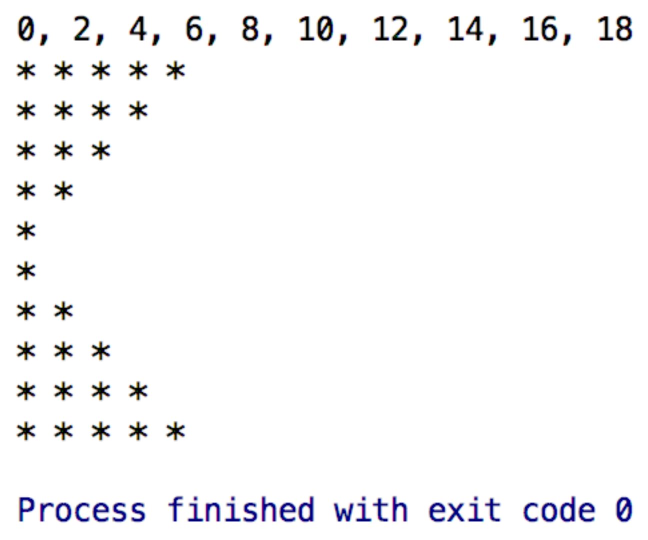
```
inline fun <T, R> Array<out T>.map(transform: (T) -> R): List<R>
inline fun <T, R> Iterable<T>.map(transform: (T) -> R): List<R>
```

```
Array(10){ it }.map { it * 2 }.joinToString().let(::println)
List(11){ it }
        .map { Math.abs(it - 5) }
        .filter { it != 0 }
        .map { "* " * it }.joinToString("\n").let(::println)
```

高阶函数

```
inline fun <T, R> Array<out T>.map(transform: (T) -> R): List<R>
inline fun <T, R> Iterable<T>.map(transform: (T) -> R): List<R>
```

```
Array(10){ it }.map { it * 2 }.joinToString().let(::println)
List(11){ it }
        .map { Math.abs(it - 5) }
        .filter { it != 0 }
        .map { "* " * it }.joinToString("\n").let(::println)
```



```
0, 2, 4, 6, 8, 10, 12, 14, 16, 18
* * * * *
* * * *
* * *
* *
*
*
* *
* * *
* * * *
* * * * *

Process finished with exit code 0
```

高阶函数

```
operator fun String.times(other: Int): String{
  return (1..other).fold(StringBuilder()){ acc, i ->
    acc.append(this)
    acc
  }.toString()
}
```

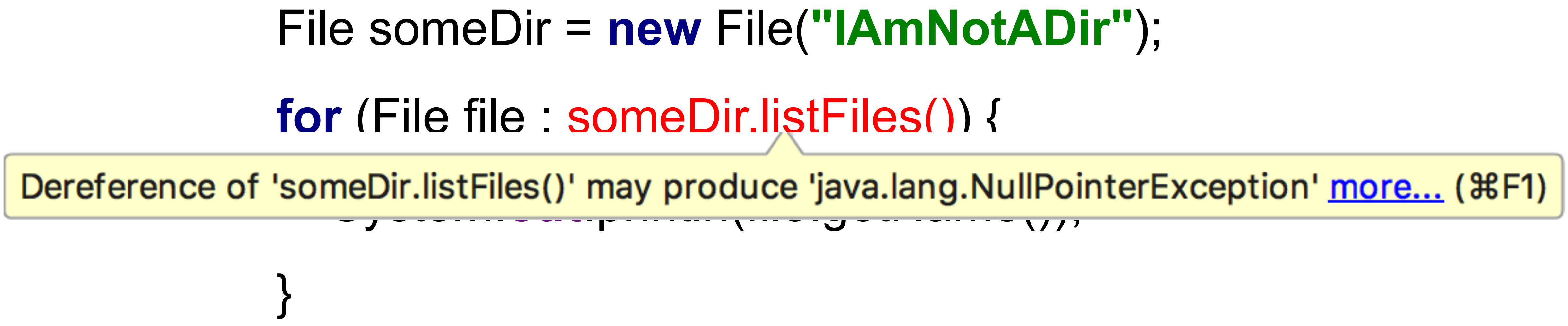
空类型安全

```
File someDir = ...;
for (File file : someDir.listFiles()) {
    System.out.println(file.getName());
}
```

空类型安全

```
File someDir = new File("IAmNotADir");
for (File file : someDir.listFiles()) {
    System.out.println(file.getName());
}
```

空类型安全



```
File someDir = new File("IAmNotADir");
for (File file : someDir.listFiles()) {
    System.out.println(file.getName());
}
```

Dereference of 'someDir.listFiles()' may produce 'java.lang.NullPointerException' more... (⌘F1)

空类型安全

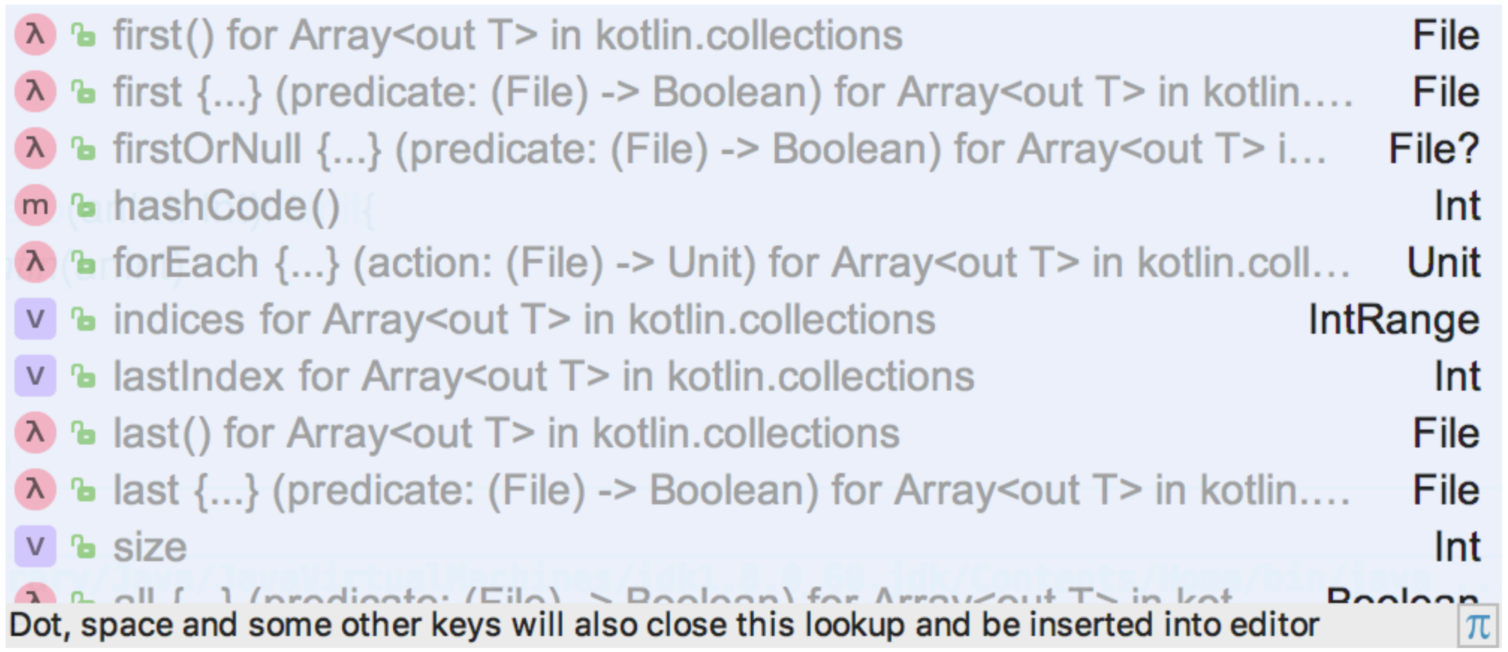
```
val someDir = File("IAmNotADir")
```

空类型安全

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()
```

空类型安全

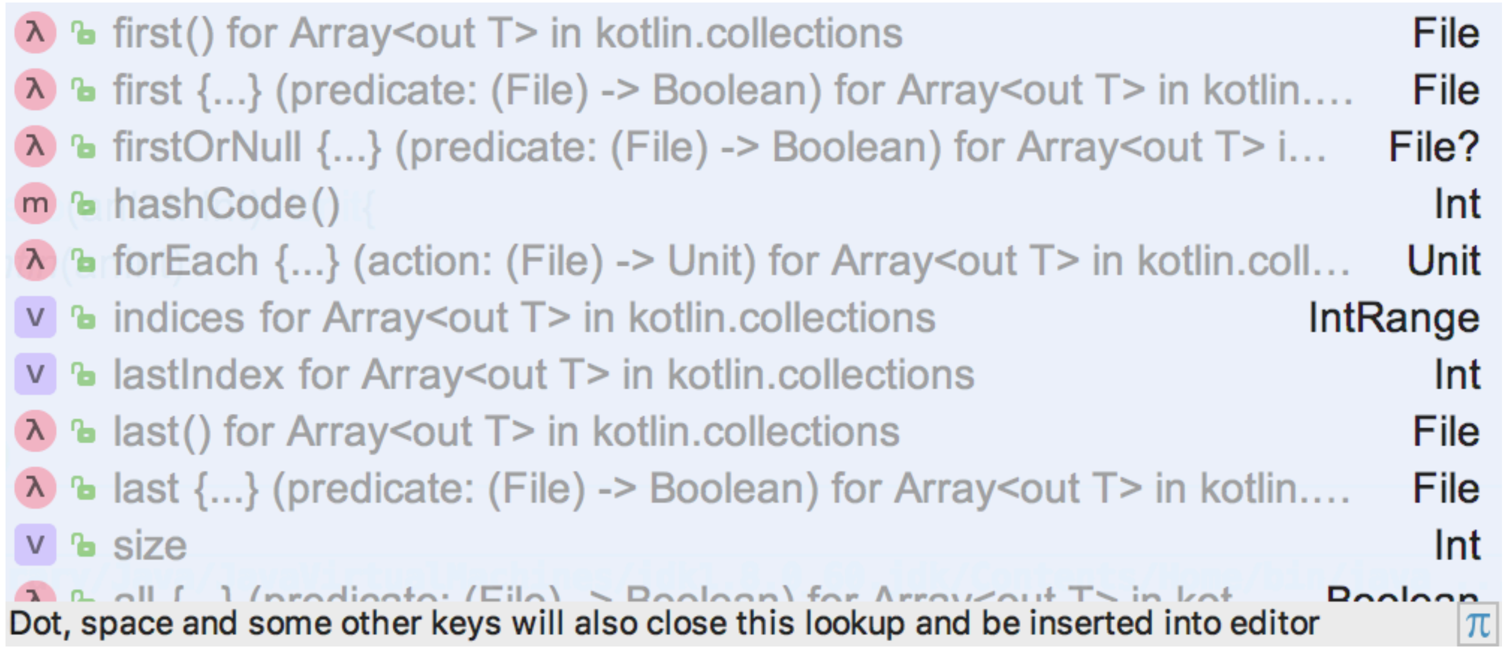
```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

subFiles.
```



```
λ first() for Array<out T> in kotlin.collections                                  File
λ first {...} (predicate: (File) -> Boolean) for Array<out T> in kotlin....       File
λ firstOrNull {...} (predicate: (File) -> Boolean) for Array<out T> i...          File?
m hashCode()                                                                       Int
λ forEach {...} (action: (File) -> Unit) for Array<out T> in kotlin.coll...       Unit
v indices for Array<out T> in kotlin.collections                              IntRange
v lastIndex for Array<out T> in kotlin.collections                                 Int
λ last() for Array<out T> in kotlin.collections                                   File
λ last {...} (predicate: (File) -> Boolean) for Array<out T> in kotlin....        File
v size                                                                             Int
λ all {...} (predicate: (File) -> Boolean) for Array<out T> in kot...          Boolean
Dot, space and some other keys will also close this lookup and be inserted into editor  π
```

空类型安全

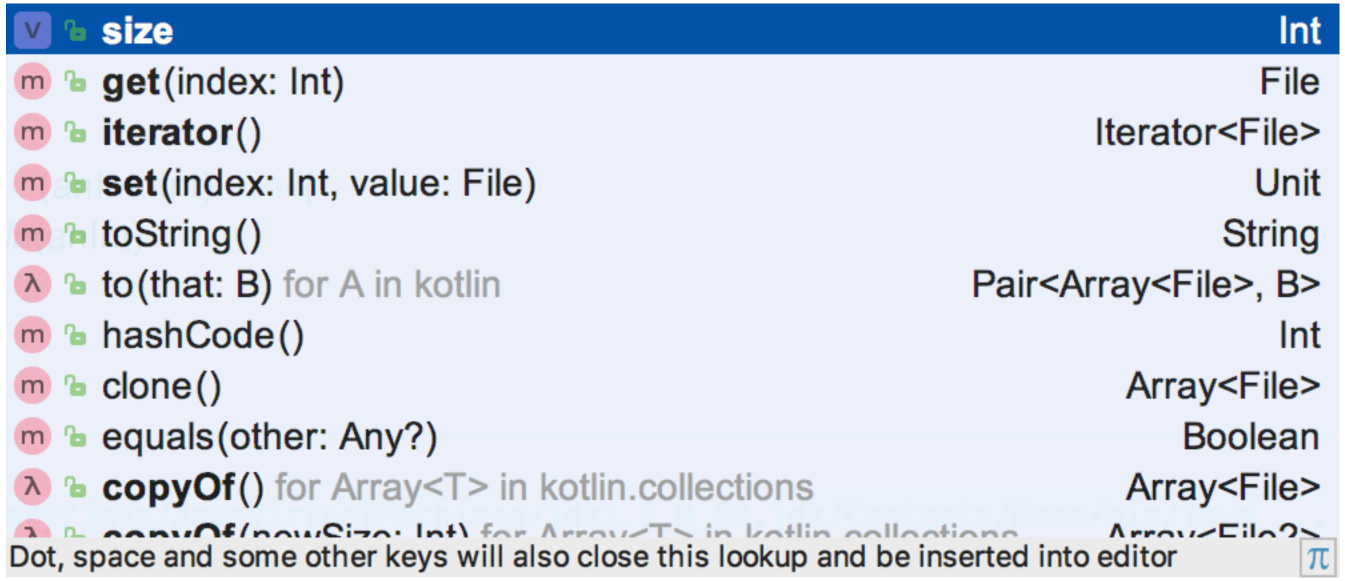
```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

subFiles.
```



```
λ first() for Array<out T> in kotlin.collections                          File
λ first {...} (predicate: (File) -> Boolean) for Array<out T> in kotlin....   File
λ firstOrNull {...} (predicate: (File) -> Boolean) for Array<out T> i...    File?
m hashCode()                                                              Int
λ forEach {...} (action: (File) -> Unit) for Array<out T> in kotlin.coll...  Unit
v indices for Array<out T> in kotlin.collections                      IntRange
v lastIndex for Array<out T> in kotlin.collections                        Int
λ last() for Array<out T> in kotlin.collections                           File
λ last {...} (predicate: (File) -> Boolean) for Array<out T> in kotlin....    File
v size                                                                    Int
Dot, space and some other keys will also close this lookup and be inserted into editor  π
```

空类型安全

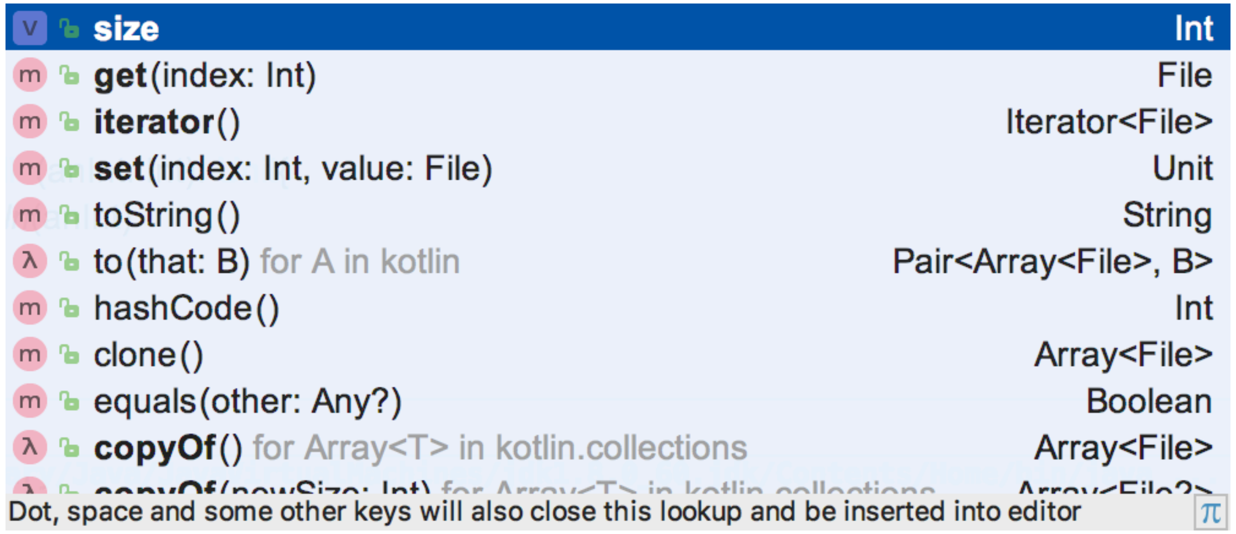
```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

subFiles?.
```



```
v size                                                    Int
m get(index: Int)                                         File
m iterator()                                              Iterator<File>
m set(index: Int, value: File)                            Unit
m toString()                                              String
λ to(that: B) for A in kotlin                             Pair<Array<File>, B>
m hashCode()                                              Int
m clone()                                                 Array<File>
m equals(other: Any?)                                     Boolean
λ copyOf() for Array<T> in kotlin.collections             Array<File>
Dot, space and some other keys will also close this lookup and be inserted into editor  π
```

类型智能转换

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

if(subFiles != null){
    subFiles.
}
```

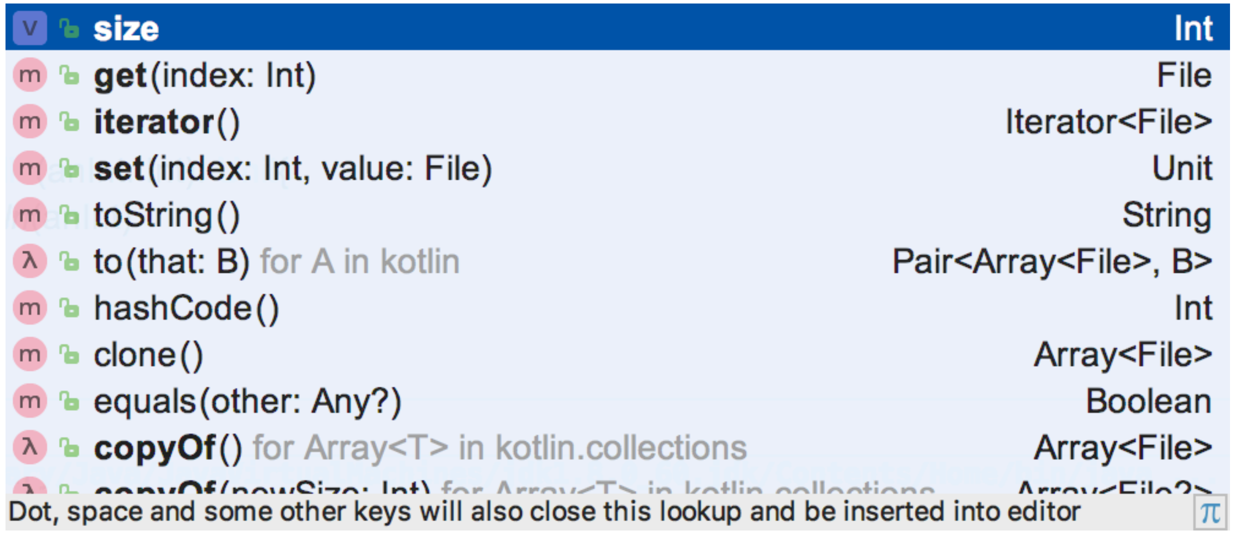


```
size                                              Int
get(index: Int)                                   File
iterator()                                        Iterator<File>
set(index: Int, value: File)                      Unit
toString()                                        String
to(that: B) for A in kotlin                       Pair<Array<File>, B>
hashCode()                                        Int
clone()                                           Array<File>
equals(other: Any?)                               Boolean
copyOf() for Array<T> in kotlin.collections       Array<File>
copyOf(newSize: Int) for Array<T> in kotlin.collections   Array<File?>
Dot, space and some other keys will also close this lookup and be inserted into editor
```

类型智能转换

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

if(subFiles != null){
    subFiles.
}
```

智能转换为
Array<File>类型



```
v size                                                   Int
m get(index: Int)                                        File
m iterator()                                             Iterator<File>
m set(index: Int, value: File)                           Unit
m toString()                                             String
λ to(that: B) for A in kotlin                            Pair<Array<File>, B>
m hashCode()                                             Int
m clone()                                                Array<File>
m equals(other: Any?)                                    Boolean
λ copyOf() for Array<T> in kotlin.collections            Array<File>
λ copyOf(newSize: Int) for Array<T> in kotlin.collections Array<File?>
Dot, space and some other keys will also close this lookup and be inserted into editor  π
```

类型智能转换

```
View view = ...;
TextView textView = ...;
if(view instanceof ViewGroup){
    ((ViewGroup) view).addView(textView);
}
```

类型智能转换

```
val view: View = ...
val textView = ...
if(view is ViewGroup) {
    view.addView(textView)
}
```

类型安全转换

```
val view: View = ...
val textView = ...
 (view as? ViewGroup)?.addView(textView)
```

尝试转换，失败就返回null

类型安全转换

```
val view: View = ...
ViewGroup? tView = ...
(view as? ViewGroup)?.addView(textView)
```

类型安全转换

```
val view: View = ...
ViewGroup? tView = ...
(view as? ViewGroup)?.addView(textView)
```

类型转换

```
val view: View = RelativeLayout(this)
val textView = ...
 (view as? ViewGroup)?.addView(textView)
```

类型转换

```
val view: View = RelativeLayout(this)
val textView = ...
 (view as ViewGroup).addView(textView)
```

Elvis 运算符

```
val view: View = ...
ViewGroup? tView = ...
(view as? ViewGroup)?.addView(textView)
```

Elvis 运算符

```
val view: View = ...
Unit? textView = ...
(view as? ViewGroup)?.addView(textView)
```

Elvis 运算符

```
val view: View = ...
val textView = ...
if(view is ViewGroup) {
    view.addView(textView)
} else {
     ...
}
```

Elvis 运算符

```
val view: View = ...
val textView = ...
 (view as? ViewGroup)?.addView(textView)
```

Elvis 运算符

```
val view: View = ...
val textView = ...
(view as? ViewGroup)?.addView(textView) ?: toast("对不起，我们尽力了")
```

let ... "else" ...

```
(view as? ViewGroup)?.let {
    it.addView(view)
}?: run {
    toast("对不起，我们尽力了")
}
```

let ... "else" ...

```
(view as? ViewGroup)?.let {
    it.addView(view)
}?: run {
    toast("对不起，我们尽力了")
}
```

let … ~~"else" …~~

```
(view as? ViewGroup)?.let {
    it.addView(view)
}?. run {
    toast("对不起，我们尽力了")
}
```

let ... ~~"else" ...~~

```
(view as? ViewGroup)?.let {
    it.addView(view)
}?. run {
    toast("很好，我们添加了一个 View")
}
```

泛型

```
public interface List<out E> : Collection<E> {
    ...
}
```

泛型

```
public interface List<out E> : Collection<E> {
    ...
}
```

泛型

```
public interface Comparable<in T> {
    ...
}
```

Reified

```
String json = ...;
Gson gson = ...;
Link link = gson.fromJson(json, Link.class);
```

Reified

```
String json = ...;
Gson gson = ...;
Link link = gson.fromJson(json, Link.class);
```

Reified

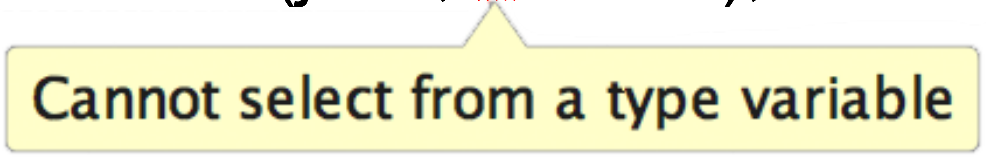
```
String json = ...;

Gson gson = ...;

Link link = gson.fromJson(json, Link.class);
```

Reified

```
public <T> T fromJson(String json){
    return gson.fromJson(json, T.class);
}
```

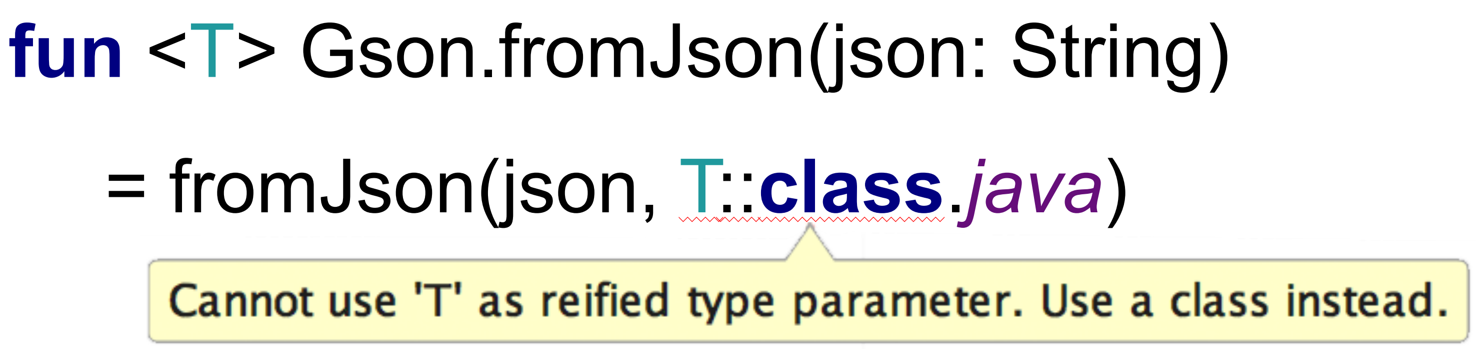
Reified

```
public <T> T fromJson(String json){
    return gson.fromJson(json, T.class);
}
```



Cannot select from a type variable

Reified



```kotlin
fun <T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

Cannot use 'T' as reified type parameter. Use a class instead.

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
        = fromJson(json, T::class.java)
```

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

```
val link: Link = gson.fromJson(json)
```

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
        = fromJson(json, T::class.java)
```

```
useLink( gson.fromJson(json) )

fun useLink(link: Link){
    ...
}
```

受检异常

```
FileWriter fw = new FileWriter(file);
fw.write(string, 0, string.length());
fw.close();
```

受检异常

```
try {
    FileWriter fw = new FileWriter(file);
    fw.write(string, 0, string.length());
    fw.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

受检异常

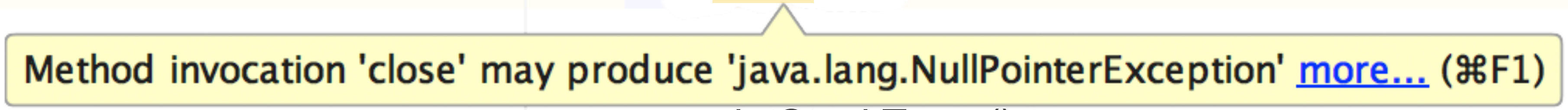
```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    fw.close();
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
```



Method invocation 'close' may produce 'java.lang.NullPointerException' more... (⌘F1)

```
        e.printStackTrace();
    }
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

受检异常

```
FileWriter fw = new FileWriter(file);
fw.write(string, 0, string.length());
fw.close();
```

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

受检异常

```
file.writer().use {
    it.write(string)
}
```

DSL

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

DSL

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

DSL

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

DSL

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

DSL

```
buildscript {
    extra["kotlinVersion"] = "1.1.51"
    repositories {
        jcenter()
    }

    val kotlinVersion: String by extra
    dependencies {
        classpath("com.android.tools.build:gradle:2.3.0")
        classpath("org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlinVersion")
    }
}
```

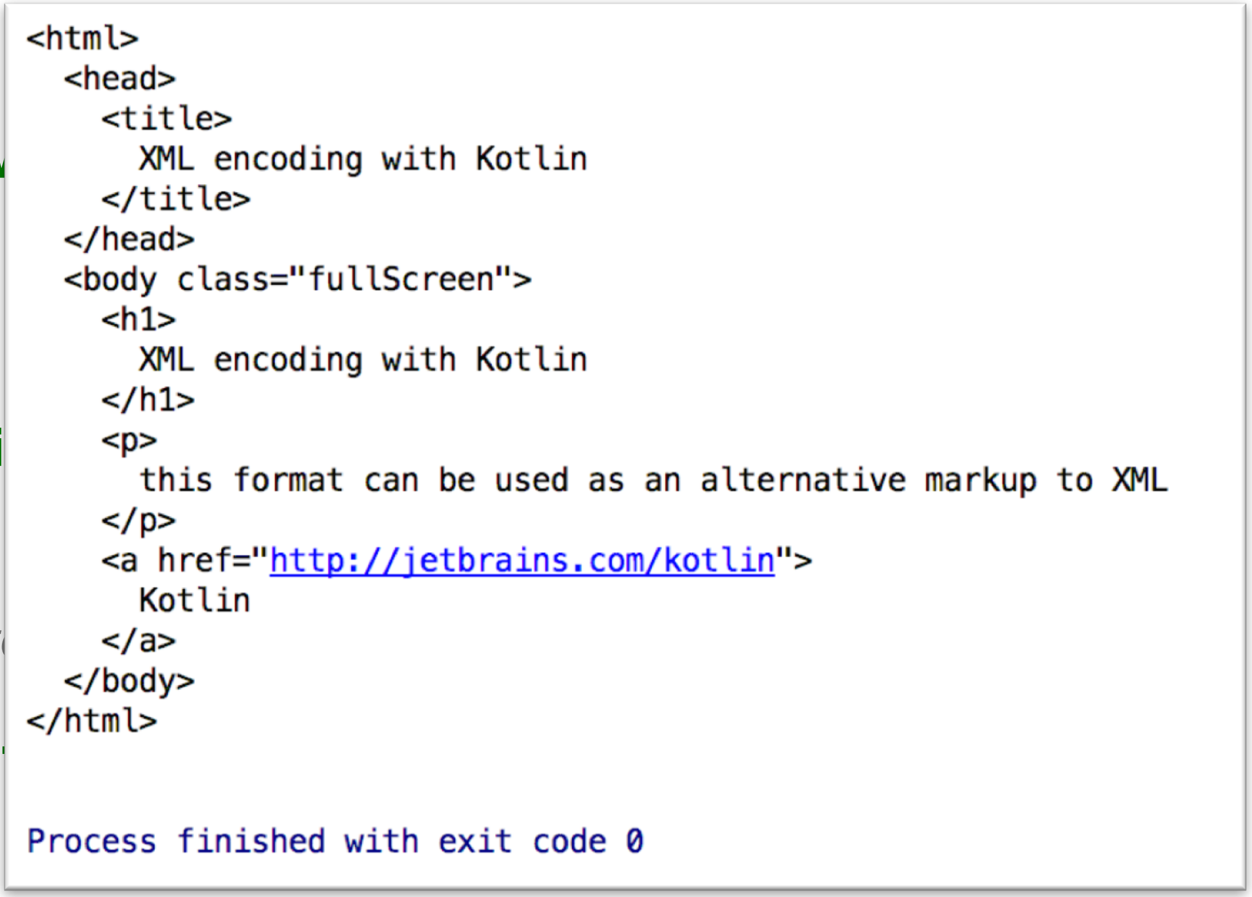
DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

See: https://try.kotl.in/#/Examples/Longer%20examples/HTML%20Builder/HTML%20Builder.kt

DSL

```
html {
    head {
        title { +"XML encoding w
    }
    body {
        h1 { +"XML encoding wi
        p { +"this format can be
        // an element with attribut
        a(href = "http://jetbrains.
    }
}
```



```
<html>
  <head>
    <title>
      XML encoding with Kotlin
    </title>
  </head>
  <body class="fullScreen">
    <h1>
      XML encoding with Kotlin
    </h1>
    <p>
      this format can be used as an alternative markup to XML
    </p>
    <a href="http://jetbrains.com/kotlin">
      Kotlin
    </a>
  </body>
</html>


Process finished with exit code 0
```

See: https://try.kotl.in/#/Examples/Longer%20examples/HTML%20Builder/HTML%20Builder.kt

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

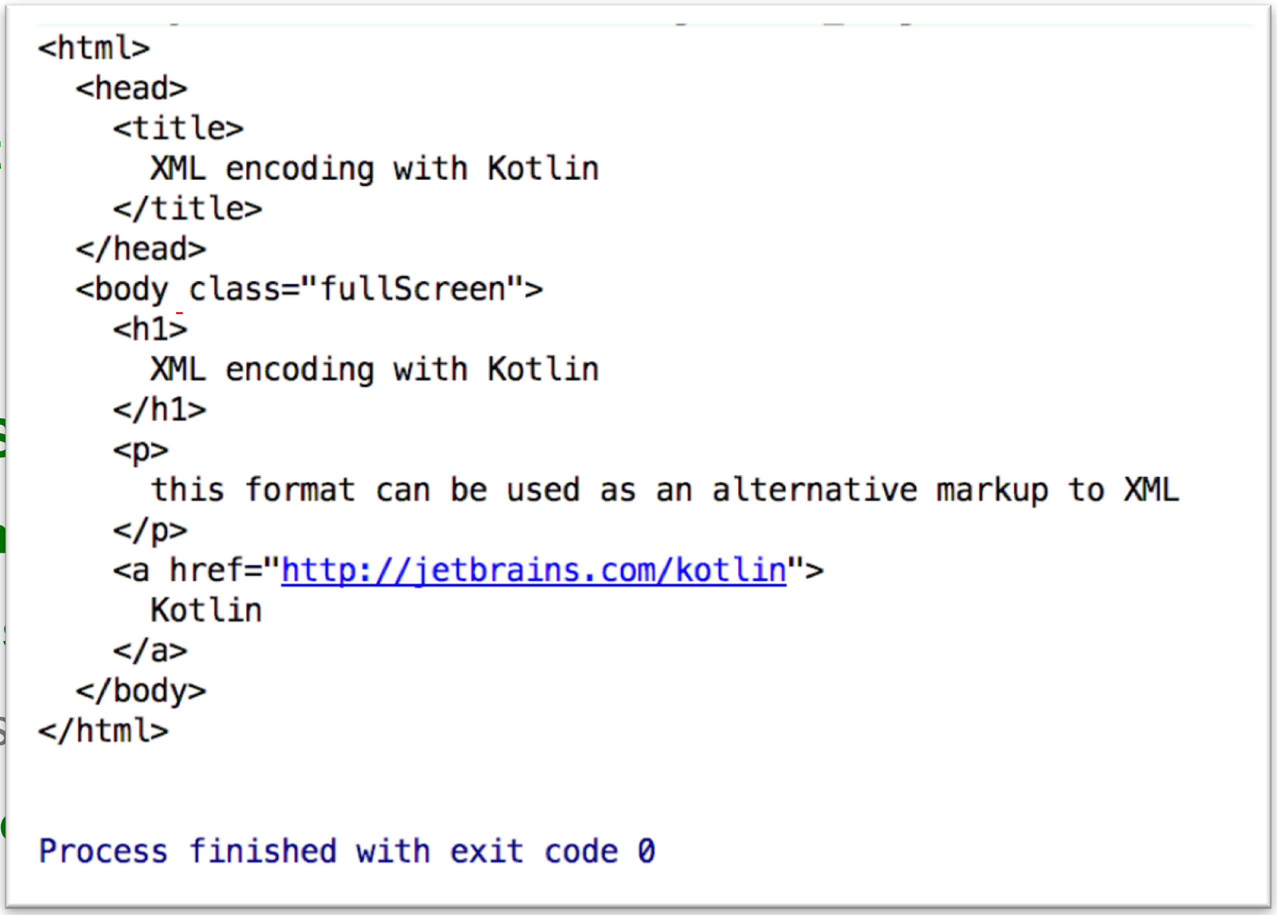
See: https://try.kotl.in/#/Examples/Longer%20examples/HTML%20Builder/HTML%20Builder.kt

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        attributes["class"] = "fullScreen"
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding wit
    }
    body {
        attributes["class"] = "fullS
        h1 { +"XML encoding with
        p { +"this format can be u
        // an element with attributes
        a(href = "http://jetbrains.c
    }
}
```



```
<html>
  <head>
    <title>
      XML encoding with Kotlin
    </title>
  </head>
  <body class="fullScreen">
    <h1>
      XML encoding with Kotlin
    </h1>
    <p>
      this format can be used as an alternative markup to XML
    </p>
    <a href="http://jetbrains.com/kotlin">
      Kotlin
    </a>
  </body>
</html>


Process finished with exit code 0
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        attributes["class"] = "fullScreen"
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

$\longrightarrow$

```
operator fun String.invoke(value: String){
    attributes[this] = value
}
```

DSL

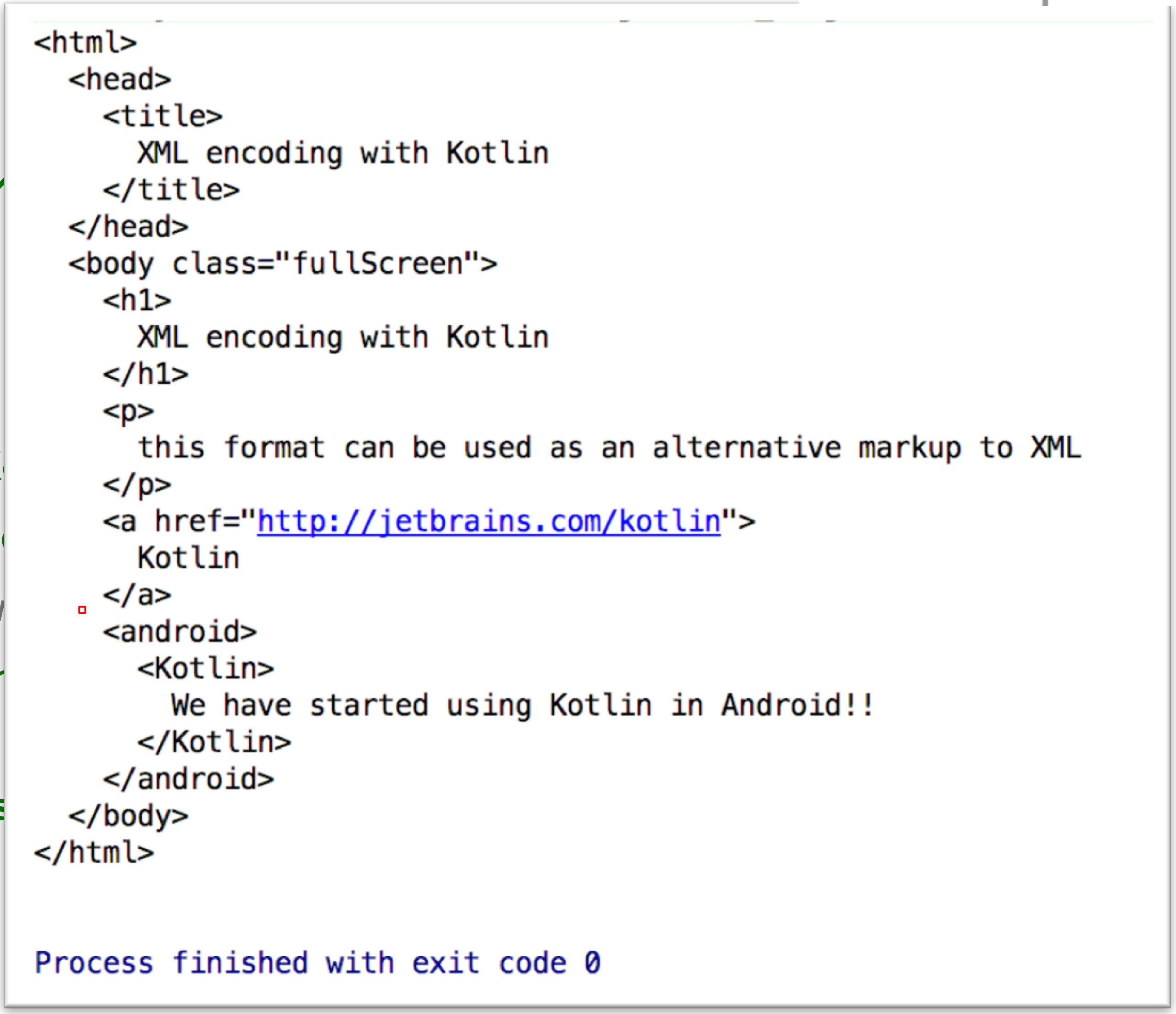
```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }

         "android"{
              "Kotlin"{    +"We have started using Kot lin in Android!!" }
        }

    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with K
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Ko
        p { +"this format can be use
        // an element with attributes ar
        a(href = "http://jetbrains.com

         "android"{
              "Kotlin"{    +"We have s
        }

    }
}
```



```
<html>
  <head>
    <title>
      XML encoding with Kotlin
    </title>
  </head>
  <body class="fullScreen">
    <h1>
      XML encoding with Kotlin
    </h1>
    <p>
      this format can be used as an alternative markup to XML
    </p>
    <a href="http://jetbrains.com/kotlin">
      Kotlin
    </a>
    <android>
      <Kotlin>
        We have started using Kotlin in Android!!
      </Kotlin>
    </android>
  </body>
</html>


Process finished with exit code 0
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }

         "android"{
               "Kotlin"{    +"We have started using Kot lin in Android!!" }
        }

    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }

         "android"{
              "Kotlin"{    +"We have started using Kot lin in Android!!" }
        }

    }
}
```

```
operator fun String.invoke(init: BodyTag.() -> Unit)
        = object: BodyTag(this@invoke){}
        .apply { this@Tag.initTag(this, init) }
```

反射

- 一个 2.5M 大小的 jar 包　　compile "org.jetbrains.kotlin:kotlin-reflect:$kotlin_version"
- 不支持的 built-in Kotlin types
- 还没来得及优化的性能

	构造对象	访问属性	修改属性	调用方法
Java 反射	12.7	25.2	12.2	18.8
Kotlin 反射	14938.0	85247.5	1316.7	326.3

Kotlin 公众号文章：Kotlin 反射你敢用吗？

Kotlin-Android-Extensions

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
```

Kotlin-Android-Extensions

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

Kotlin-Android-Extensions

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Hello World!"/>

</RelativeLayout>
```

Kotlin-Android-Extensions

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Hello World!"/>

</RelativeLayout>
```

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        textView.text = "Hello"
    }
}
```

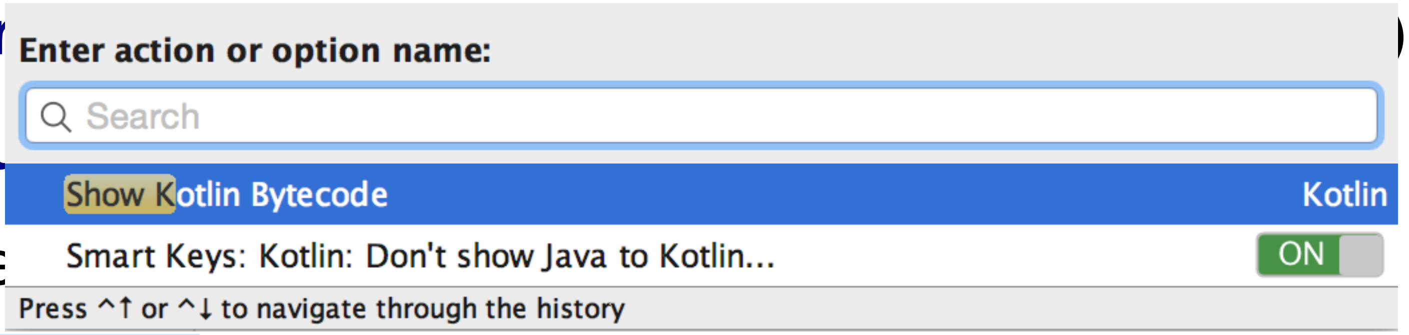
Kotlin-Android-Extensions

```xml
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Hello World!"/>

</RelativeLayout>
```

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        textView.text = "Hello"
    }
}
```

Kotlin-Android-Extensions

```
class MainActivity : AppCompatActivity() {
    over                                                    ) {
        su
        se
        textView.text = "Hello"
    }
}
```



Enter action or option name:

Search

Show Kotlin Bytecode — Kotlin

Smart Keys: Kotlin: Don't show Java to Kotlin... — ON

Press ^↑ or ^↓ to navigate through the history

Kotlin-Android-Extensions

```
LINENUMBER 11 L2
ALOAD 0
GETSTATIC com/bennyhuo/testrelection/R$id.textView : I
INVOKEVIRTUAL com/bennyhuo/testrelection/MainActivity._$_findCachedViewById (I)Landroid/view/View;
CHECKCAST android/widget/TextView
LDC "Hello"
CHECKCAST java/lang/CharSequence
INVOKEVIRTUAL android/widget/TextView.setText (Ljava/lang/CharSequence;)V
```

Kotlin-Android-Extensions

```
LINENUMBER 11 L2
ALOAD 0
GETSTATIC com/bennyhuo/testrelection/R$id.textView : I
INVOKEVIRTUAL com/bennyhuo/testrelection/MainActivity._$_findCachedViewById (I)Landroid/view/View;
CHECKCAST android/widget/TextView
LDC "Hello"
CHECKCAST java/lang/CharSequence
INVOKEVIRTUAL android/widget/TextView.setText (Ljava/lang/CharSequence;)V
```

Kotlin-Android-Extensions

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp">

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="10dp"
        android:textColor="#000"
        android:textSize="16sp"/>

</FrameLayout>
```

Kotlin-Android-Extensions

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp">

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="10dp"
        android:textColor="#000"
        android:textSize="16sp"/>

</FrameLayout>
```

```
override fun getView(position: Int,
        convertView: View?,
        parent: ViewGroup
): View {
    val view = convertView
            ?: inflater.inflate(R.layout.item, parent, false)
    view.name.text = "benny"
    return view
}
```

Kotlin-Android-Extensions

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp">

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="10dp"
        android:textColor="#000"
        android:textSize="16sp"/>

</FrameLayout>
```

```kotlin
override fun getView(position: Int,
        convertView: View?,
        parent: ViewGroup
): View {
    val view = convertView
            ?: inflater.inflate(R.layout.item, parent, false)
    view.name.text = "benny"
    return view
}
```

Anko 扩展

```
compile "org.jetbrains.anko:anko:$anko_version"
```

Anko 扩展

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(TestActivity.this, "I'm clicked!", Toast.LENGTH_SHORT);
    }
});
```

Anko 扩展

```
button.setOnClickListener(View.OnClickListener {
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT)
})
```

Anko 扩展

```
button.setOnClickListener({
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT)
})
```

Anko 扩展

```
button.setOnClickListener{
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT)
}
```

Anko 扩展

```
button.onClick {
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT)
}
```

Anko 扩展

```
button.onClick {
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT).show()
}
```

Anko 扩展

```
button.onClick {
    toast("I'm clicked!")
}
```

Anko 扩展

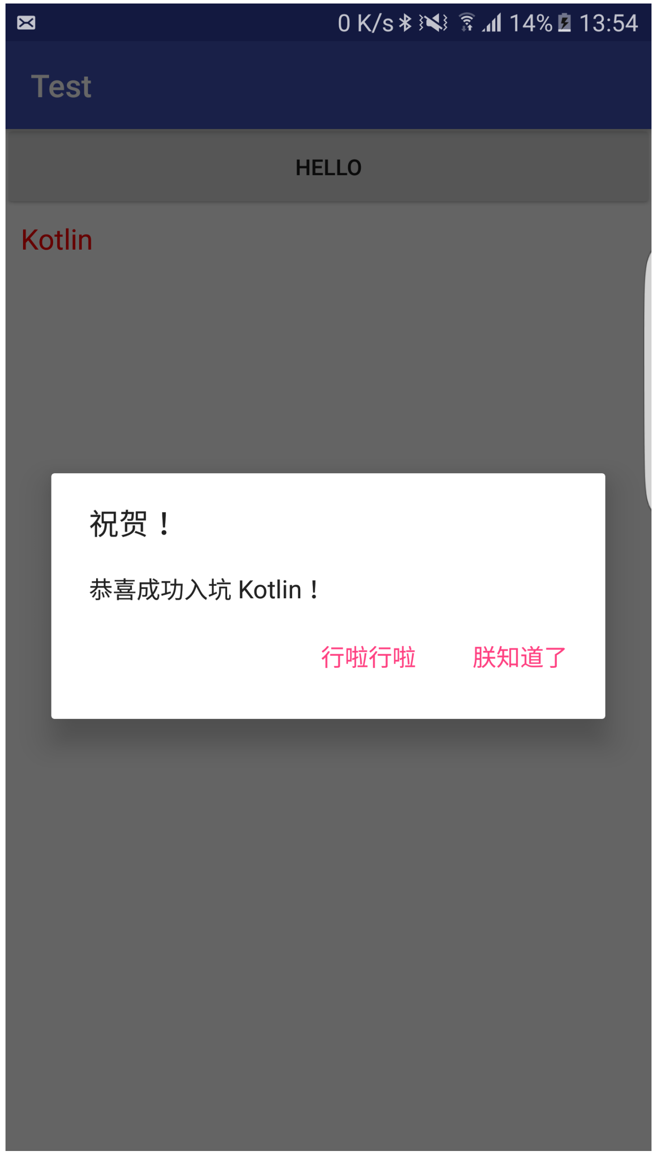
```
AlertDialog alertDialog = new AlertDialog.Builder(this)
        .setTitle("警告！ ")
        .setMessage("95后都在玩，再不学 Kotlin 就说明你老啦！ ")
        .setPositiveButton("朕知道了", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        })
        .setNegativeButton("朕就不学", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(TestActivity.this, "反了你们了", Toast.LENGTH_SHORT).show();
                dialog.dismiss();
            }
        }).create();
alertDialog.show();
```

Anko 扩展

```
alert {
    title = "祝贺！ "
    message = "恭喜成功入坑 Kotlin！ "
    positiveButton("朕知道了"){
        toast("多大点儿事儿")
    }
    negativeButton("行啦行啦"){
        toast("退下吧")
    }
}.show()
```

Anko 扩展

```
alert {
    title = "祝贺！ "
    message = "恭喜成功入坑 Kotlin！ "
    positiveButton("朕知道了"){
        toast("多大点儿事儿")
    }
    negativeButton("行啦行啦"){
        toast("退下吧")
    }
}.show()
```

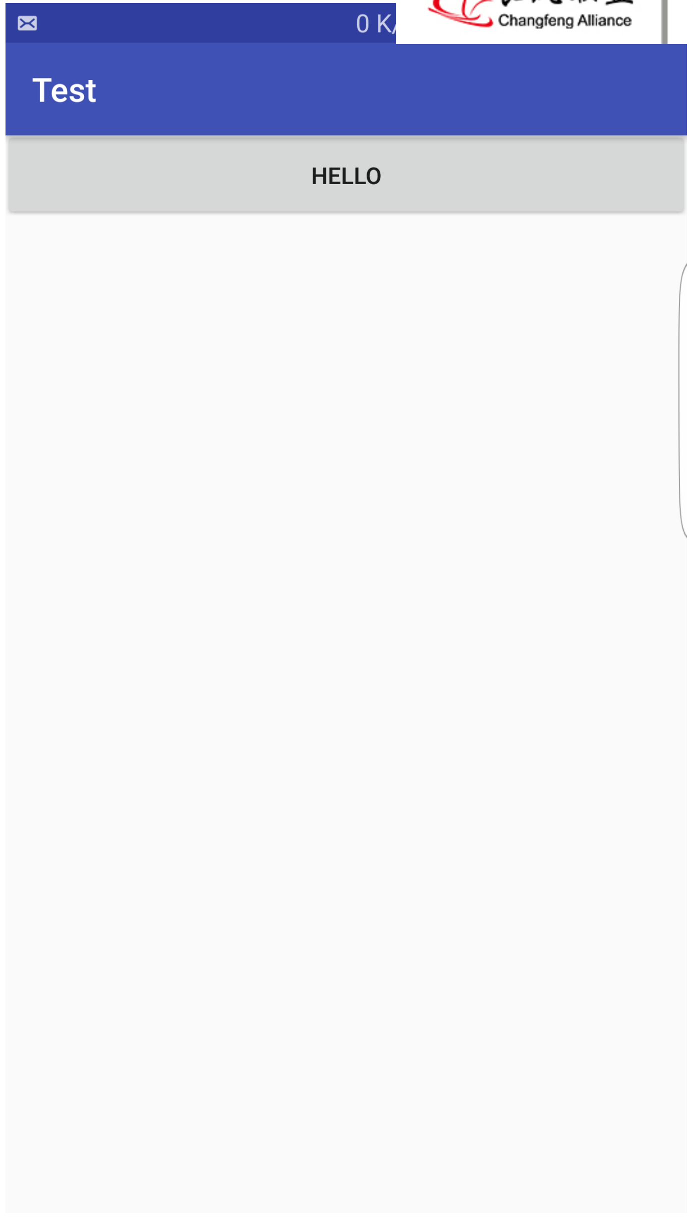


祝贺！

恭喜成功入坑 Kotlin！

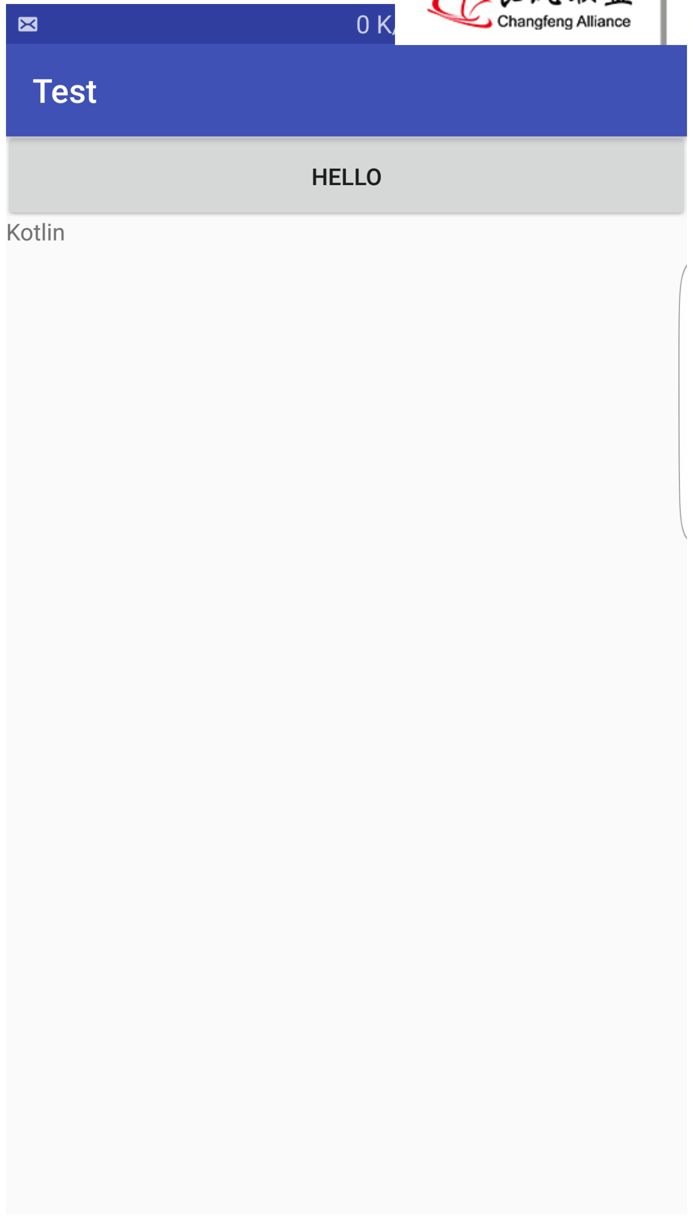
Anko 布局

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        verticalLayout {
            button("Hello")
        }
    }
}
```



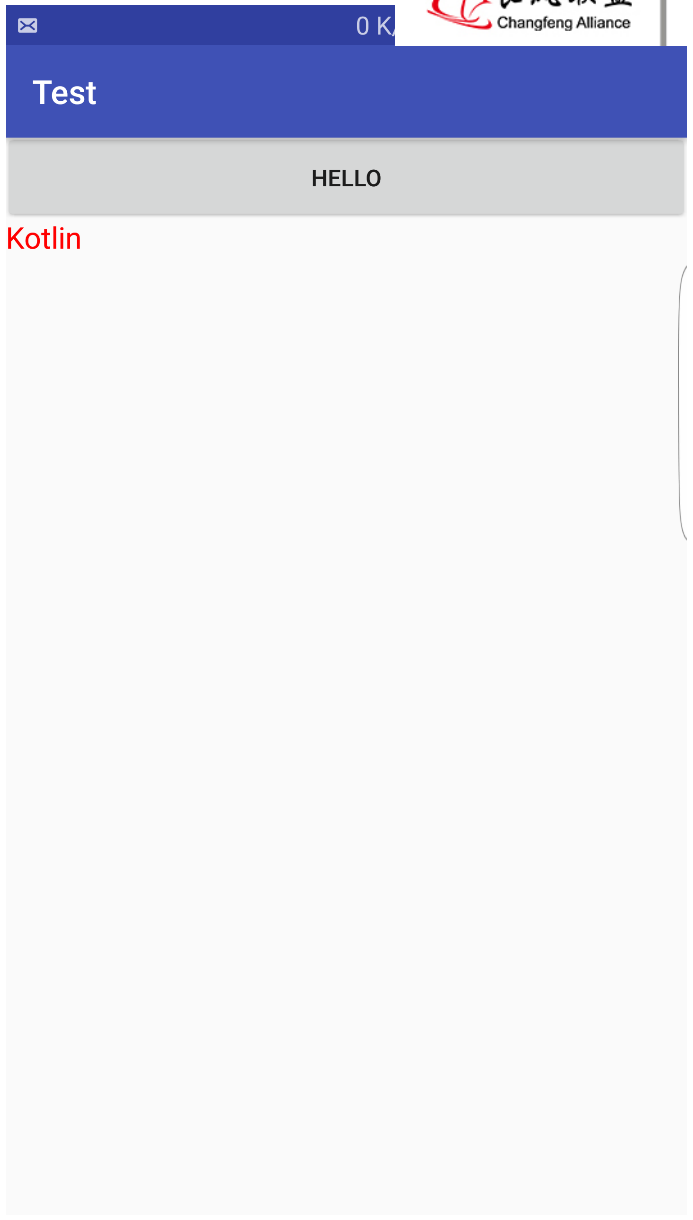
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin")
}
```



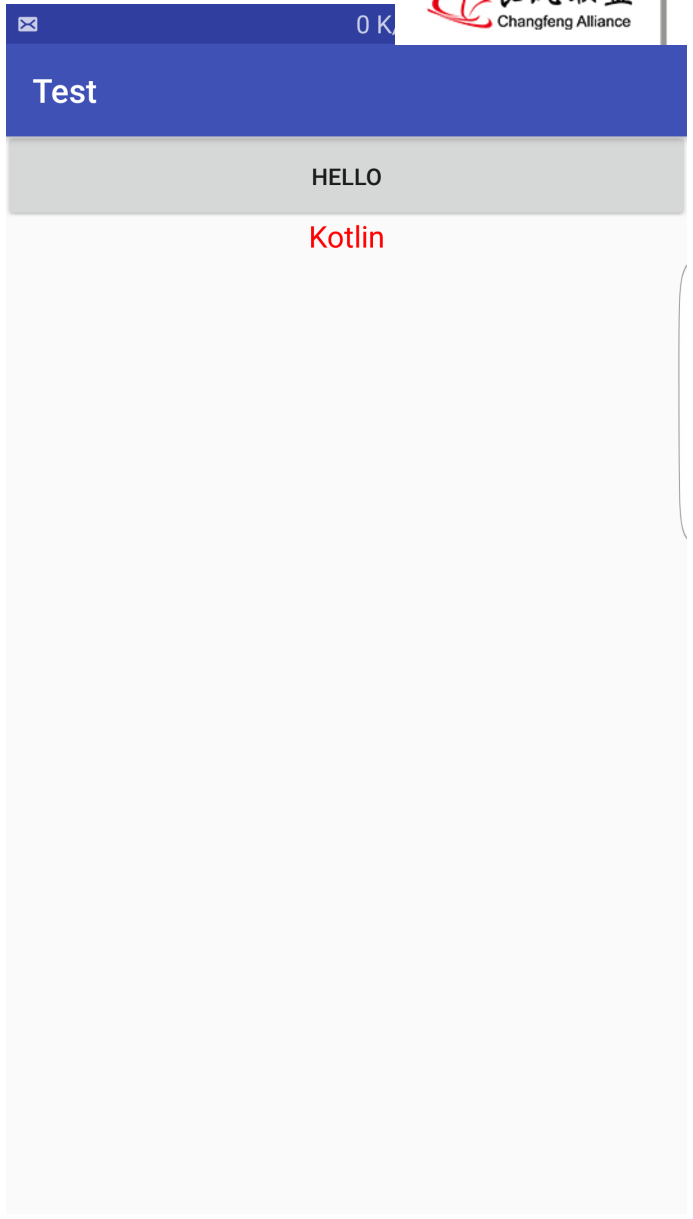
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
    }
}
```


0 K
Test
HELLO
Kotlin

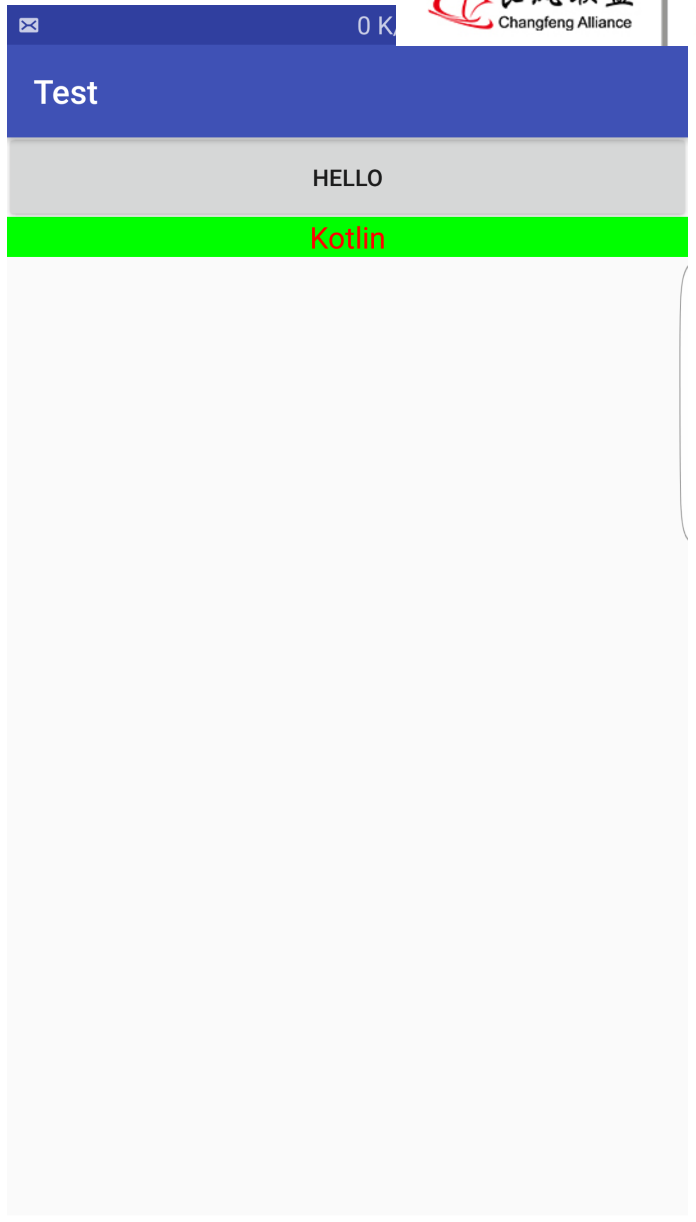
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
        gravity = Gravity.CENTER
    }
}
```



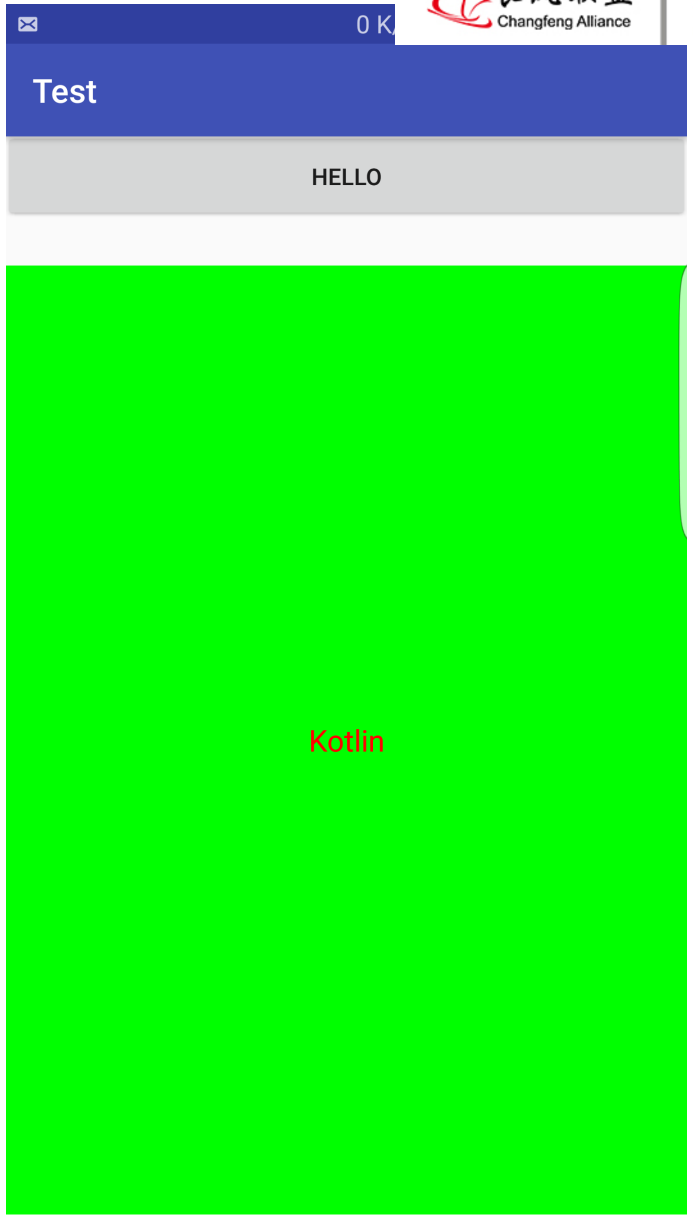
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
        gravity = Gravity.CENTER
        backgroundColor = Color.GREEN
    }
}
```



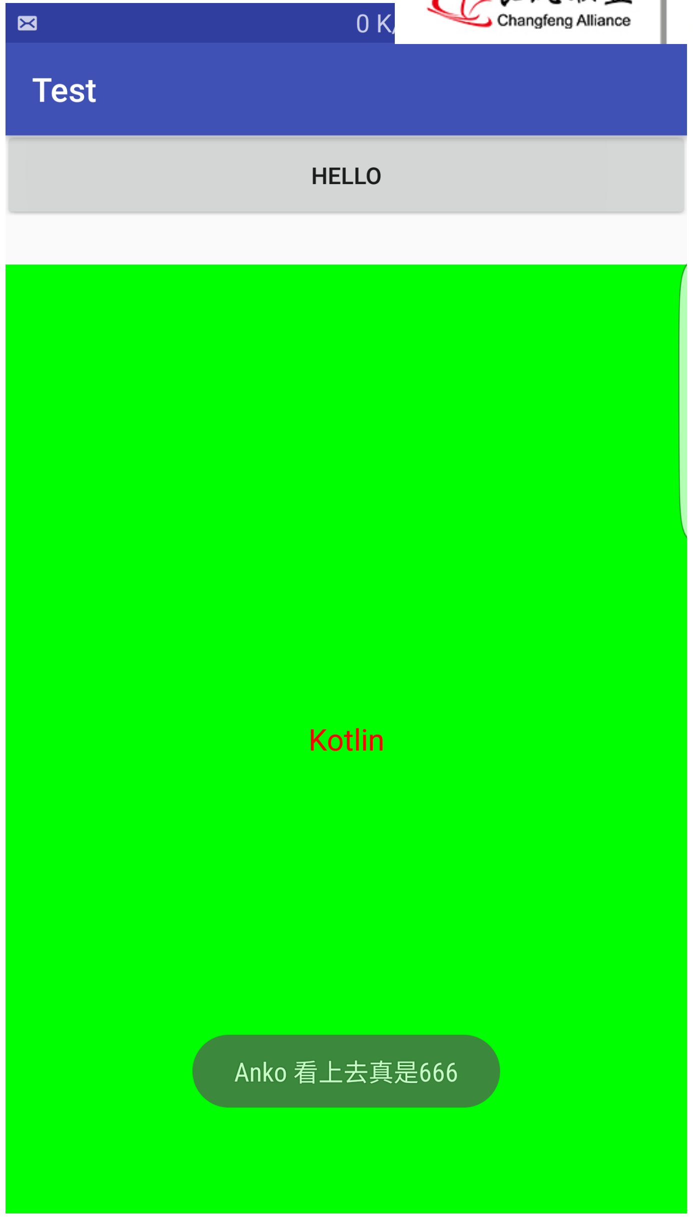
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
        gravity = Gravity.CENTER
        backgroundColor = Color.GREEN
    }.lparams(matchParent, matchParent){
        topMargin = dip(30)
    }
}
```



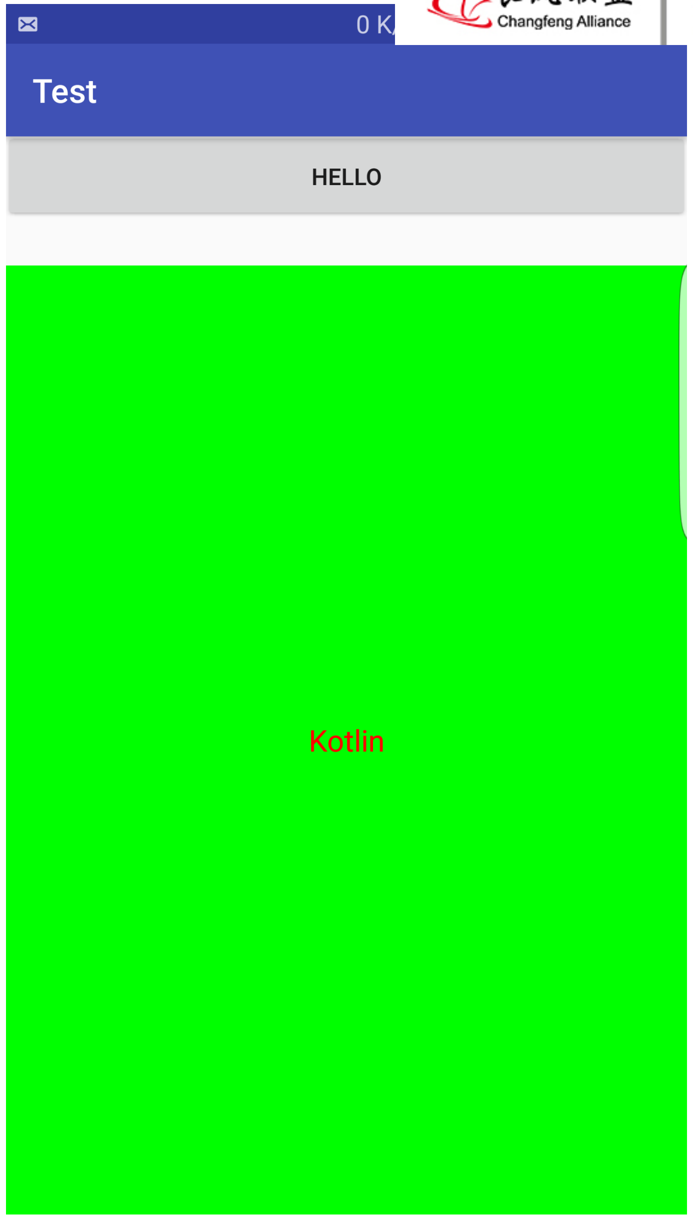
Anko 布局

```
verticalLayout {
    button("Hello") {
        onClick { toast("Anko 看上去真是666") }
    }
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
        gravity = Gravity.CENTER
        backgroundColor = Color.GREEN
    }.lparams(matchParent, matchParent){
        topMargin = dip(30)
    }
}
```



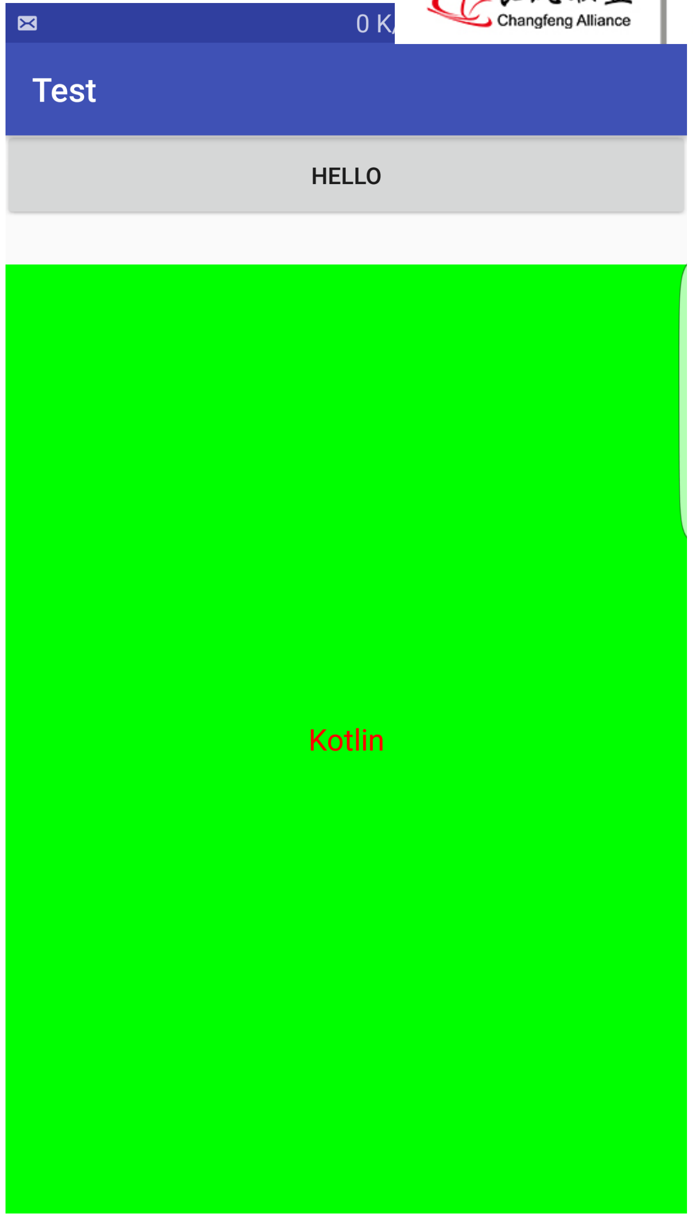
Anko 布局

```
verticalLayout {
    button("Hello") {
        onClick { toast("Anko 看上去真是666") }
    }
    themedTextView(
            "Kotlin",
            R.style.commonText
    ).lparams(matchParent, matchParent) {
            topMargin = dip(30)
            }
}
```



Anko 布局

```
<color name="red">#FF0000</color>
<color name="green">#00FF00</color>
<style name="commonText">
    <item name="android:textSize">18sp</item>
    <item name="android:textColor">@color/red</item>
    <item name="android:gravity">center</item>
    <item name="android:background">@color/green</item>
</style>
```



Anko 布局

- 无运行时开销，类型安全
- 代码更容易复用
- 预览功能受限制，必须编译才可预览
- 使用体验一般，Kotlin-Android-extensions 无效

- 除非硬编码布局可考虑使用，XML 布局仍是最佳的布局方案。

更多关于 Anko

https://github.com/Kotlin/anko

Coroutine

- 轻量级调度执行
- 异步代码写起来看上去如同同步代码一般
- 异常处理更轻松

Coroutine 认知三步走

- 应用
- 标准库
- 字节码

Coroutine 应用

```
interface ImageRequestCallback{
    fun onSuccess(bitmap: Bitmap)
    fun onError(e: Throwable)
}
fun fetchImageWithCallback(url: String, callback: ImageRequestCallback){
    try {
        callback.onSuccess(fetchImage(url))
    }catch (e: Exception){
        callback.onError(e)
    }
}
```

Coroutine 应用

```
interface ImageRequestCallback{
    fun onSuccess(bitmap: Bitmap)
    fun onError(e: Throwable)
}
fun fetchImageWithCallback(url: String, callback: ImageRequestCallback){
    try {
        callback.onSuccess(fetchImage(url))
    }catch (e: Exception){
        callback.onError(e)
    }
}
```

```
fetchImageWithCallback(urlA, object : ImageRequestCallback {
    override fun onSuccess(bitmap: Bitmap) {
        handler.post { imageViewA.setImageBitmap(bitmap) }
    }

    override fun onError(e: Throwable) {
        e.printStackTrace()
        handler.post { showErrorOnUi(e.message) }
    }
})
```

Coroutine 应用

使用 Kotlinx.coroutines Android 库

```
compile "org.jetbrains.kotlinx:kotlinx-coroutines-android:$version"
```

Coroutine 应用

```
launch(UI) {
    val imageA = async { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

Coroutine 应用

```
launch(UI) {
    val imageA = async { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
compile "org.jetbrains.anko:anko-coroutines:$anko_version"
```

Coroutine 应用 – Anko 扩展

```
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
internal var POOL = newFixedThreadPoolContext(2 * Runtime.getRuntime().availableProcessors(), "bg")

inline fun <T> bg(crossinline block: () -> T): Deferred<T> = async(POOL) {
    block()
}
```

Coroutine 应用 – Anko 扩展

```
internal var POOL = newFixedThreadPoolContext(2 * Runtime.getRuntime().availableProcessors(), "bg")

inline fun <T> bg(crossinline block: () -> T): Deferred<T> = async(POOL) {
    block()
}
```

Coroutine 应用 – Anko 扩展

```
launch(UI) {
    val imageA = async { fetchImage(urlA)
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(im geA.await())
    imageViewB.setIma                        ait())
}
```

很耗时

Activity 内存泄露

Coroutine 应用 – Anko 扩展

```
val imageViewARef = imageViewA.asReference()
val imageViewBRef = imageViewB.asReference()
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA.await())
    imageViewBRef().setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
val imageViewARef = imageViewA.asReference()
val imageViewBRef = imageViewB.asReference()
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    im ImageView! ().setImageBitmap(imageA.await())
    imageViewBRef().setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
class Ref<out T : Any> internal constructor(obj: T) {
    private val weakRef = WeakReference(obj)
    suspend operator fun invoke(): T {
        return suspendCoroutineOrReturn {
            val ref = weakRef.get() ?: throw CancellationException()
            ref
        }
    }
}
fun <T : Any> T.asReference() = Ref(this)
```

Coroutine 应用 – Anko 扩展

```
class Ref<out T : Any> internal constructor(obj: T) {
    private val weakRef = WeakReference(obj)
    suspend operator fun invoke(): T {
        return suspendCoroutineOrReturn {
            val ref = weakRef.get() ?: throw CancellationException()
            ref
        }
    }
}
fun <T : Any> T.asReference() = Ref(this)
```

No expression found

```
throw CancellationException()
```

```
weakRef.get()
```

```
throw CancellationException()

weakRef.get()
```

$$\begin{tikzcd}
\text{Nothing} \arrow[dr] & \\
& T\ (\text{ImageView}) \\
T\ (\text{ImageView}) \arrow[ur] &
\end{tikzcd}$$

Coroutine 应用 – Anko 扩展

```
val imageViewARef = imageViewA.asReference()
val imageViewBRef = imageViewB.asReference()
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA.await())
    imageViewBRef().setImageBitmap(imageB.await())
}
```

Coroutine 应用 – 处理异常

```
launch(UI) {
    try {
        ...
    } catch (e: Exception) {
        showErrorOnUi(e.message)
    }
}
```

Coroutine 应用 – 处理异常

```
launch(UI) {
    ...
}.invokeOnCompletion {
   it?.printStackTrace()
}
```

无异常时为null

Coroutine 应用 – AsyncAwait 扩展

```
compile 'co.metalab.asyncawait:asyncawait:1.0.0'
```

Coroutine 应用 – AsyncAwait 扩展

```
async {
    val imageA = await { fetchImage(urlA) }
    val imageB = await { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA)
    imageViewBRef().setImageBitmap(imageB)
}.onError {
    showErrorOnUi(it.message)
} .finally{
    releaseRefs()
}
```

Coroutine 应用对比

```
async {
    val imageA = await { fetchImage(urlA) }
    val imageB = await { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA)
    imageViewBRef().setImageBitmap(imageB)
}.onError {
    showErrorOnUi(it.message)
} .finally{
     releaseRefs()
}
```

```
launch(UI) {
    val imageA = async { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA.await())
    imageViewBRef().setImageBitmap(imageB.await())
}.invokeOnCompletion {
    it?.printStackTrace()
}
```

有何不同？

Coroutine 认知三步走

- 应用
- 标准库：较为底层，主要提供给应用层框架开发者使用
- 字节码：主要提供给编译器用

可参考Kotlin 公众号文章：

- 深入理解 Kotlin Coroutine
- 深入理解 Kotlin Coroutine (2)
- 深入理解 Kotlin Coroutine（3）

案例参考

案例参考 – 腾讯云图床上传工具

- 地址：QCloudImageUploaderForMarkDown
- 简介：一个完整的 JVM 程序，涉及文件操作、属性读写、正则表达式等内容。可以一键上传文件夹下所有图片到腾讯云并替换对应的 Markdown 文件中的图片地址。

案例参考 – 腾讯云图床上传工具

- 腾讯云图床上传工具使用方法
  - 开通腾讯云对象存储服务
  - 在腾讯云上创建对象存储的 Bucket
  - 获取 AppId、SecretId、SecretKey、BucketName 以及区域（例如华北就是tj）
  - 配置好 conf 目录下面的 settings.properties 文件

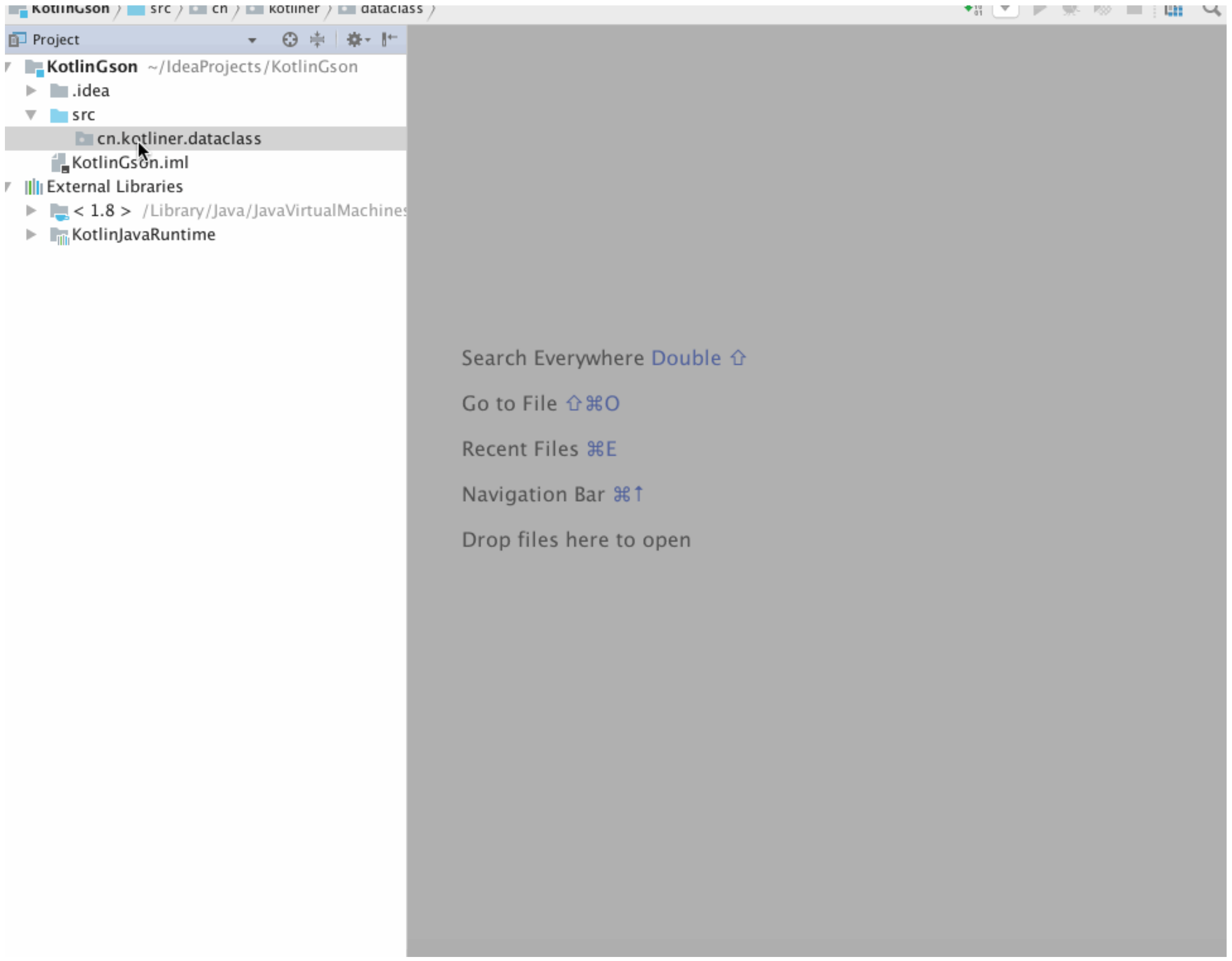
番外篇：博客/公众号文章编辑器

- MWEB：Markdown 本地编辑器
  - 支持复制图片直接粘贴到 Markdown
  - 支持博客目录文件管理（例如 Hexo 搭建的博客等）
  - 支持部分图床的上传（七牛、WordPress 等）
- 微信公众号排版工具：http://md.barretlee.com/
- 腾讯云图床上传工具
  - 支持以目录为单位上传和单文件上传，同一目录图片不重复上传
  - 支持直接替换原文档图片地址，可选择图片传后即删

案例参考 – Kotlin 版 GsonFormat

- 地址：NewDataClassAction
- 简介：将 Json 数据转换为 Kotlin data class 的 IntelliJ 插件。

案例参考 – Kotlin 版 GsonFormat



案例参考 – Kotlin 版 GsonFormat

```
{
  "name": "Bennyhuo",
  "company": "@Tencent",
  "site": "www.kotliner.cn",
  "location": "Beijing",
  "email": "enbandari@qq.com",
  "bio": "微信公众号 Kotlin"
}
```

案例参考 – Kotlin 版 GsonFormat

```
{
  "name": "Bennyhuo",
  "company": "@Tencent",
  "site": "www.kotliner.cn",
  "location": "Beijing",
  "email": "enbandari@qq.com",
  "bio": "微信公众号 Kotlin"
}
```

```
data class User(var name: String,
                var company: String,
                var site: String,
                var location: String,
                var email: String,
                var bio: String)
```

案例参考 – 谷歌官方 Android 案例

- 地址：Android Samples
- 简介：官方给出了较多案例，可供大家参考如何将 Kotlin 应用到 Android 开发当中。

案例参考 – Kotlin 版设计模式

- 地址：Design-Patterns-In-Kotlin
- 简介：使用 Kotlin 编写的设计模式案例

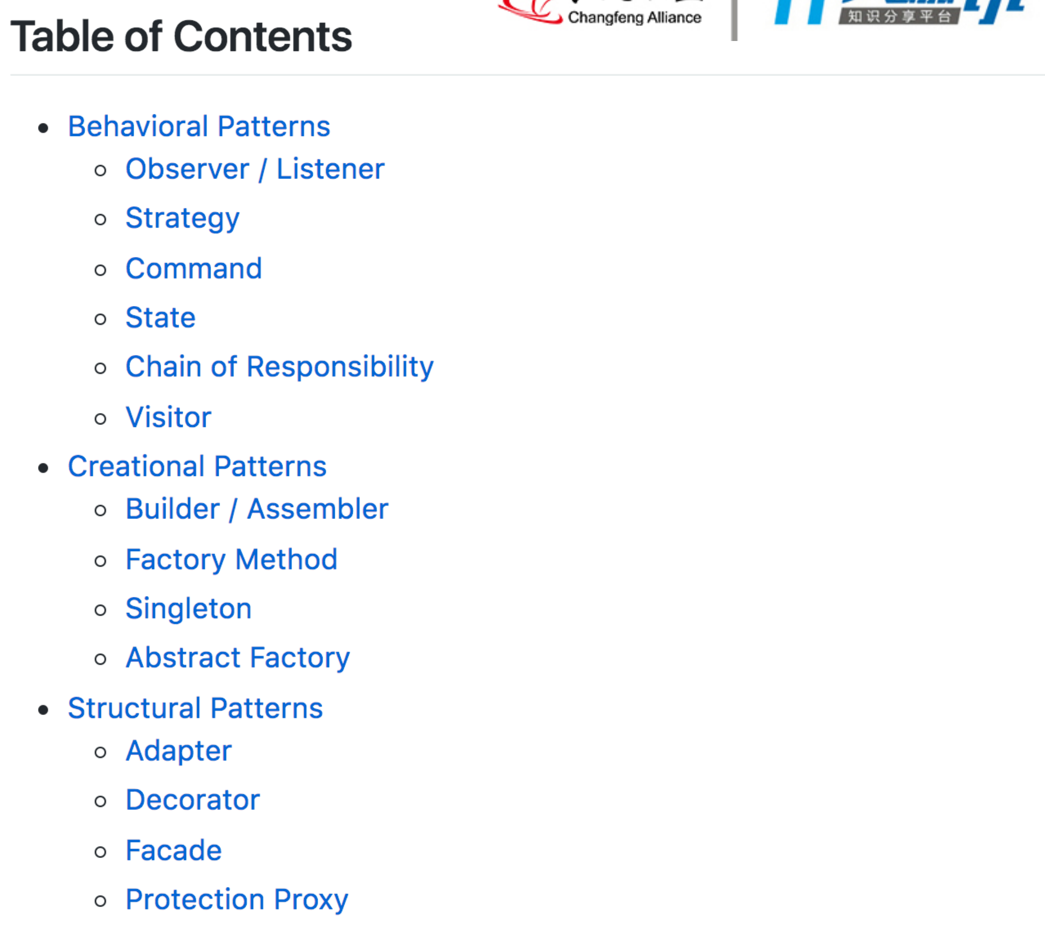


Table of Contents

- Behavioral Patterns
  - Observer / Listener
  - Strategy
  - Command
  - State
  - Chain of Responsibility
  - Visitor
- Creational Patterns
  - Builder / Assembler
  - Factory Method
  - Singleton
  - Abstract Factory
- Structural Patterns
  - Adapter
  - Decorator
  - Facade
  - Protection Proxy

案例参考 – Kotlin 版设计模式

```
class Printer(val stringFormatterStrategy: (String) -> String) {
    fun printString(string: String)
        = println(stringFormatterStrategy.invoke(string))
}

val lowerCaseFormatter: (String) -> String = { it.toLowerCase() }

val upperCaseFormatter = { it: String -> it.toUpperCase() }
```

案例参考 – Kotlin 版 ButterKnife

- 地址：kotterknife
- 简介：出自 Jake Wharton 之手，是属性代理的很好的学习案例。

案例参考 – Kotlin 版 ButterKnife

```
public class PersonView(context: Context, attrs: AttributeSet?)
        : LinearLayout(context, attrs) {
    val firstName: TextView by bindView(R.id.first_name)
    val lastName: TextView by bindView(R.id.last_name)
}
```

案例参考 – 协程框架 AsyncAwait

- 地址：AsyncAwait
- 简介：使用标准库 API 实现的协程封装，麻雀虽小五脏俱全，如果你想了解标准库中的协程 API，又无意于深入研究和扩展协程框架，推荐阅读它的源码。

案例参考 – 协程框架 Kotlinx.coroutines

- 地址：kotlinx.coroutines
- 简介：官方提供的协程框架，提供了较为完善和丰富的协程框架体系。如果你想更进一步深入学习研究 Kotlin 协程 API 的原理和使用，建议你仔细阅读这个框架。

“

纸上得来终觉浅，绝知此事要躬行

”

谢谢大家！