Introduction to Kotlin

Using Kotlin for Android

Using Kotlin for Android

- 兼容性
- 性能
- 互通性
- 占用
- 编译时长
- 学习曲线

Hello Kotlin

MainActivity.java

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

MainActivity.java

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

MainActivity.kt

```
class MainActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
}
```

TextView.java

```
public float getAlpha() {
    // Retrieve value...
}

public void setAlpha(float alpha) {
    // Set value...
}
```

MainActivity.java

```
TextView tv = // ...
Log.d("MainActivity", "Alpha: " + tv.getAlpha());
tv.setAlpha(0f);
```

MainActivity.kt

```
val tv = // ...
Log.d("MainActivity", "Alpha: " + tv.alpha)
tv.alpha = 0f
```

MainActivity.java

```
LinearLayout views = // ...
for (int i = 0; i < views.getChildCount(); i++) {
    View view = views.getChildAt(i);
    // TODO do something with view
}
```

MainActivity.kt

```
val views = // ...
for (index in 0 until views.childCount) {
    val view = views.getChildAt(index)
    // TODO do something with view
}
```

MainActivity.kt

```
val views = // ...
views.forEach { view ->
    // TODO do something with view
}
```

ViewGroups.kt

```
fun ViewGroup.forEach(action: (View) -> Unit) {
    for (index in 0 until childCount) {
        action(getChildAt(index))
    }
}
```

MainActivity.kt

```
val views = // ...
val first = views[0]
views -= first
views += first
if (first in views) doSomething()
Log.d("MainActivity", "View count: ${views.size}")
```

ViewGroups.kt

```
operator fun ViewGroup.get(index: Int): View? = getChildAt(index)
operator fun ViewGroup.minusAssign(child: View) = removeView(child)
operator fun ViewGroup.plusAssign(child: View) = addView(child)
operator fun ViewGroup.contains(child: View) = indexOfChild(child) != -1

val ViewGroup.size: Int
    get() = childCount
```

MainActivity.kt

```
val views = // ...
for (view in views.children()) {
    // TODO do something with view
}
val visibleHeight = views.children()
    .filter { it.visibility == View.VISIBLE }
    .sumBy { it.measuredHeight }
```

ViewGroups.kt

```
fun ViewGroup.children() = object : Iterable<View> {
    override fun iterator() = object : Iterator<View> {
        var index = 0
        override fun hasNext() = index < childCount
        override fun next() = getChildAt(index++)
    }
}
```

Person.java

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

MainActivity.java

```
Trace.beginSection(sectionName);
expensiveCalculation();
Trace.endSection();
```

Traces.kt

```
inline fun <T> trace(sectionName: String, body: () -> T): T {
    Trace.beginSection(sectionName)
    try {
        return body()
    } finally {
        Trace.endSection()
    }
}
```

MainActivity.kt

```
val result = trace("foo") {
    expensiveCalculation()
}
```

MainActivity.java

```
SQLiteDatabase db = // ..
db.beginTransaction();
try {
    db.delete("users", "first_name = ?", new String[]{"jake"});
    db.setTransactionSuccessful();
} finally {
    db.endTransaction();
}
```

Databases.kt

```
inline fun SQLiteDatabase.transaction(body: () -> Unit) {
    beginTransaction()
    try {
        body()
        setTransactionSuccessful()
    } finally {
        endTransaction()
    }
}
```

MainActivity.kt

```
val db = // ..
db.transaction {
    db.delete("users", "first_name = ?", arrayOf("jake"))
}
```

Databases.kt

```
inline fun SQLiteDatabase.transaction(body: (SQLiteDatabase) -> Unit) {
    beginTransaction()
    try {
        body(this)
        setTransactionSuccessful()
    } finally {
        endTransaction()
    }
}
```

MainActivity.kt

```
val db = // ..
db.transaction {
    it.delete("users", "first_name = ?", arrayOf("jake"))
}
```

Databases.kt

```
inline fun SQLiteDatabase.transaction(body: SQLiteDatabase.() -> Unit) {
    beginTransaction()
    try {
        body()
        setTransactionSuccessful()
    } finally {
        endTransaction()
    }
}
```

MainActivity.kt

```
val db = // ..
db.transaction {
    delete("users", "first_name = ?", arrayOf("jake"))
}
```

Delegates

```
private val name by Delegates.observable("<no name>") { old, new, prop ->
  println("Name changed from $old to $new")
}

private val address by Delegates.notNull<String>()
```

MyListener.kt

```
class MyListener : TransitionListener {
    override fun onTransitionEnd(transition: Transition) {...}
    override fun onTransitionResume(transition: Transition) {...}
    override fun onTransitionPause(transition: Transition) {...}
    override fun onTransitionCancel(transition: Transition) {...}
    override fun onTransitionStart(transition: Transition) {...}
}
```

MyListener.kt

```
class MyListener : TransitionListener {
    override fun onTransitionStart(transition: Transition) {
        ...
    }
}
```

EmptyTransitionListener.kt

```
object EmptyTransitionListener : TransitionListener {
    override fun onTransitionEnd(transition: Transition) {}
    override fun onTransitionResume(transition: Transition) {}
    override fun onTransitionPause(transition: Transition) {}
    override fun onTransitionCancel(transition: Transition) {}
    override fun onTransitionStart(transition: Transition) {}
}
```

MyListener.kt

```
class MyListener : TransitionListener by EmptyTransitionListener{
    override fun onTransitionStart(transition: Transition) {
        ...
    }
}
```

Thanks