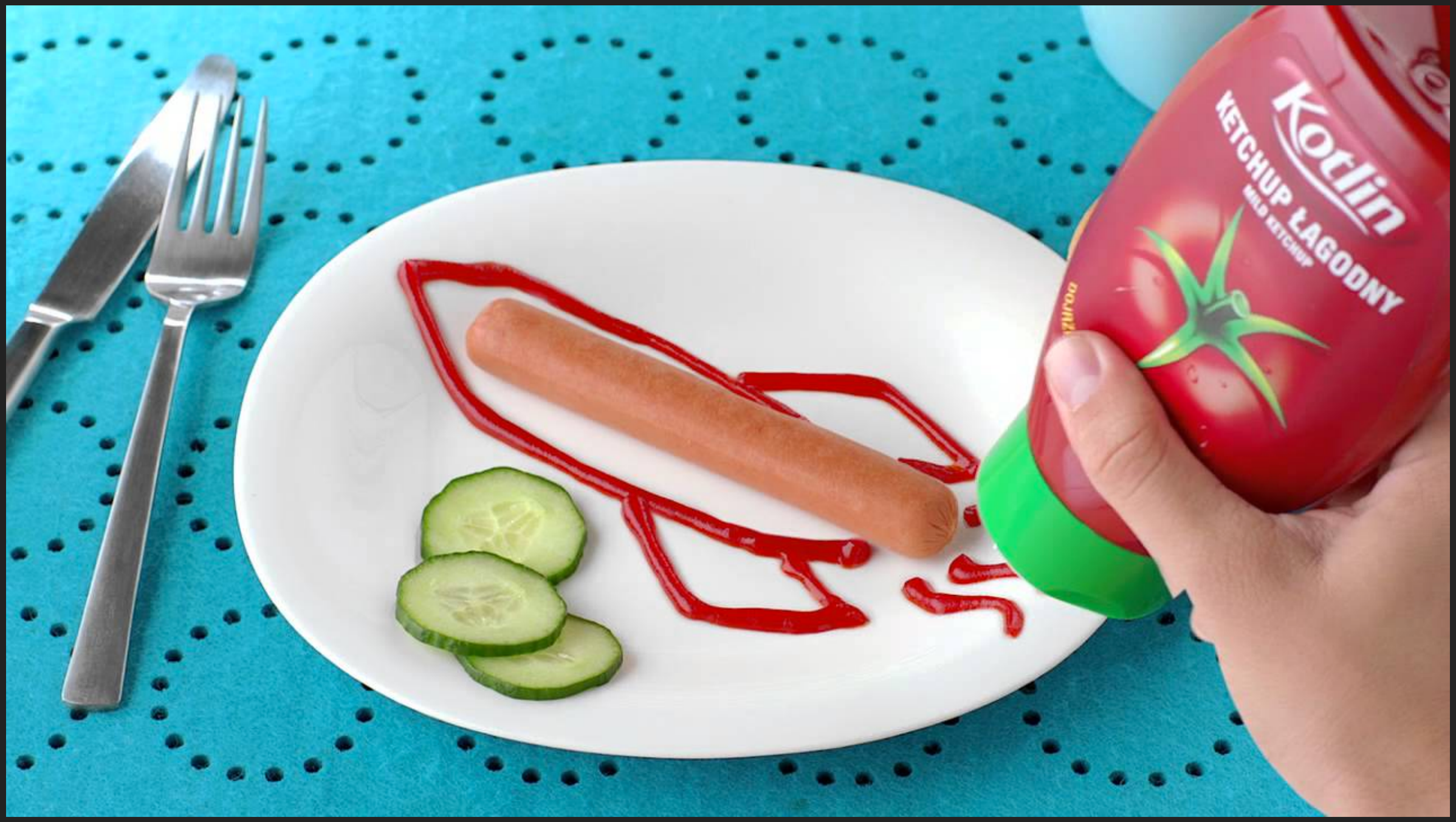
PROGRAMMING ANDROID WITH KOTLIN

By Nico

INTRODUCTION

- What's Kotlin
  - Talk is cheap, show me the code
  - Kotlin VS Swift
- Why use Kotlin to develop a new Android application
  - Powerful language features (Kotlin code brief)
  - Kotlin Android Extensions
- Short code demonstration to show the magic of Kotlin
  - Delegated properties
  - Function extension, property extension even more
  - Anko Library
- Conclusions
- Q&A

WHAT'S KOTLIN



What is Kotlin

- Kotlin is an Open Source language, statically typed
- Made by JetBrains (folks behind IntelliJ IDEA , AppCode…)
- Leverage existing frameworks and libraries of the JVM with 100% Java Interoperability.
- Named after an island near Saint Petersburg (the location of the development office behind the project)
- Kotlin 1.0.2 is out

This is Kotlin

```
class Student(var name: String){

    var gender: Int = 0 //I'm property, not a member or field
    var email: String? = null
    var age: Int = 0
    lateinit var firstName: String
    val isBoy by lazy { gender == 1 }

    constructor(name: String, gender: Int): this(name){
        this.email = email
        this.gender = gender
        firstName = name
    }

    var bodyHeight = 0f
        set(value) {
            field = value
        }
        get() {
            return field
        }

    override fun toString() = mapOf("name" to name,
    "email" to email, "age" to age, "bodyHeight" to bodyHeight).toString()

}
```

This is Swift

```
class Student {
    var name: String
    var email = ""
    var age = 0
    var gender = 0
    lazy var isBoy: Bool = { return self.gender==1 }()
    var bodyHeight: Float{
        set{
            bodyHeight = newValue
        }
        get{
            return self.bodyHeight
        }
    }
    init(name: String){
        self.name = name
    }
    convenience init(name: String, gender: Int){
        self.init(name: name)
        self.gender = gender
    }

    var description: String{
```

This is Java...

```
public class Student {

    public Student(String name) {
        this.name = name;
    }

    public Student(String name, int gender) {
        this(name);
        this.gender = gender;
    }

    private String name;
    private int gender;
    private String email;
    private int age;
    private float bodyHeight;


    @Override
    public String toString() {
        HashMap<String, String> map = new HashMap<>();
        map.put("name", name);
        map.put("email", email);
        map.put("age", String.valueOf(age));
        map.put("bodyHeight", String.valueOf(bodyHeight));
        return map.toString();
    }
```


Kotlin
KETCHUP ŁAGODNY
MILD KETCHUP
TERAZ JESZCZE WIĘCEJ
DOJRZEWAJĄCYCH W SŁOŃCU POMIDORÓW
KETCHUP
POD KONTROLĄ

VS


SWIFT

WHY USE KOTLIN TO DEVELOP A NEW ANDROID APPLICATION

Powerful language features

- Named and optional arguments
- Lambdas
- Null and type safety ()
- Data class (POJO)
- Function Extension
- Higher-Order Functions: filter{}, map{} ...
- balabala...when, generic...

NAMED AND OPTIONAL ARGUMENTS

```
package similar

/**
 * Created by demon on 5/14/16.
 */
class Student(var name: String = "",
              val gender: Int = 0,
              var email: String? = null,
              var age: Int = 0){
    var bodyHeight = 0f
        set(value) {
            field = value
        }
        get() {
            return field
        }

    fun information(): String {
        return toString()
    }
}
```

```
//val student = Student(“David”, 1, “hello@gmail.com”, 90)
or
//val student = Student(email = “David")
```

LAMBDA

without it:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

    }
});
```

from now on:

```
button.setOnClickListener { //view ->

}
 //Equal to: button.onClick({view-> println(view)})
```

LAMBDA

```
button.setOnClickListener(object: View.OnClickListener{
    override fun onClick(v: View?) {
        toast("helloworld")
    }
})



//Java8 SAM liked
button.setOnClickListener ({ view -> toast("helloworld") })

button.setOnClickListener { view -> toast("helloworld") }

button.setOnClickListener { toast("helloworld") }

button.onClick { toast("helloworld") } //We need Anko Lib
```

LAMBDA

```
val ints = listOf<Int>()
var sum = 0
ints.filter { it > 0 }.forEach {
    sum += it
}
print(sum)
```

INLINE

```
fun <T> lock(lock: Lock, body: () -> T): T {
    lock.lock()
    try {
        return body()
    }
    finally {
        lock.unlock()
    }
}
```

lock(l) { foo() }

How about inline?

```
inline fun <T> lock(lock: Lock, body: () -> T): T {xxxx}
```

lock(l) { foo() } →

```
l.lock()
try {
    foo()
}
finally {
    l.unlock()
}
```

DATA CLASS

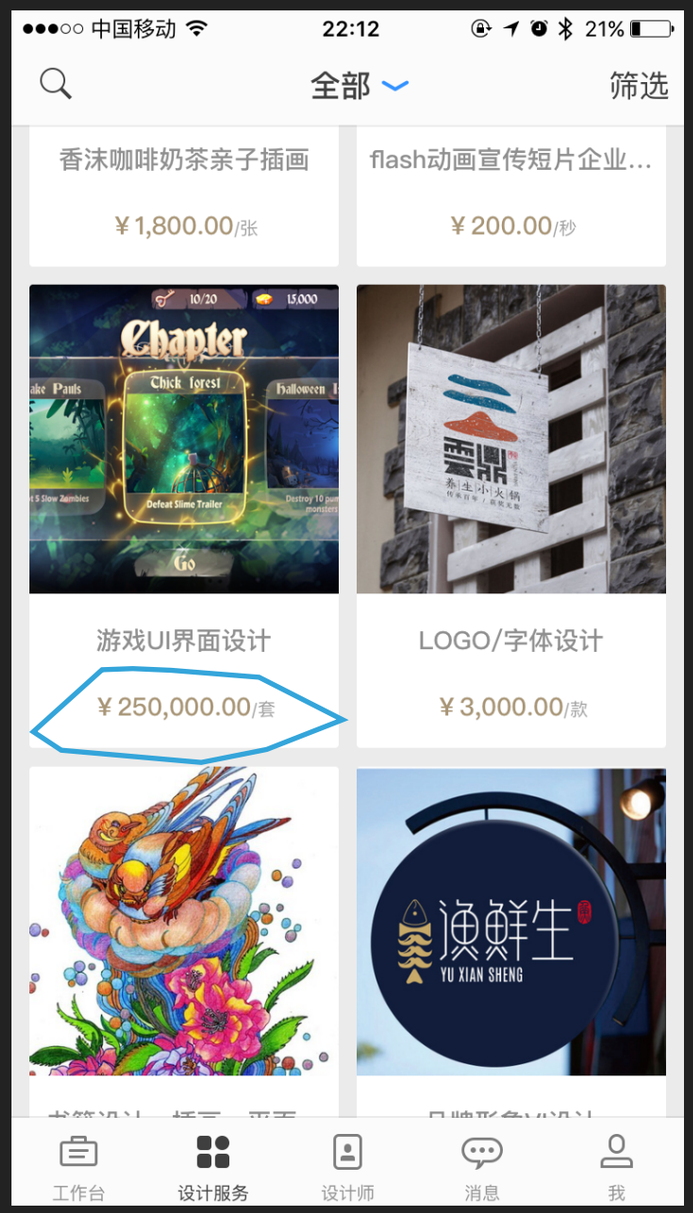
```
data class Order(val orderId: String = "",
                 val sellerId: Long = 0,
                 val buyerId: Long = 0,
                 val name: String = "",
                 val serviceId: String = "",
                 val workflowId: String? = null,
                 val seller: XX = XX(),
                 val buyer: XX= XX(),
                 val lastStatusFromBuyer: Boolean = false){
}
```

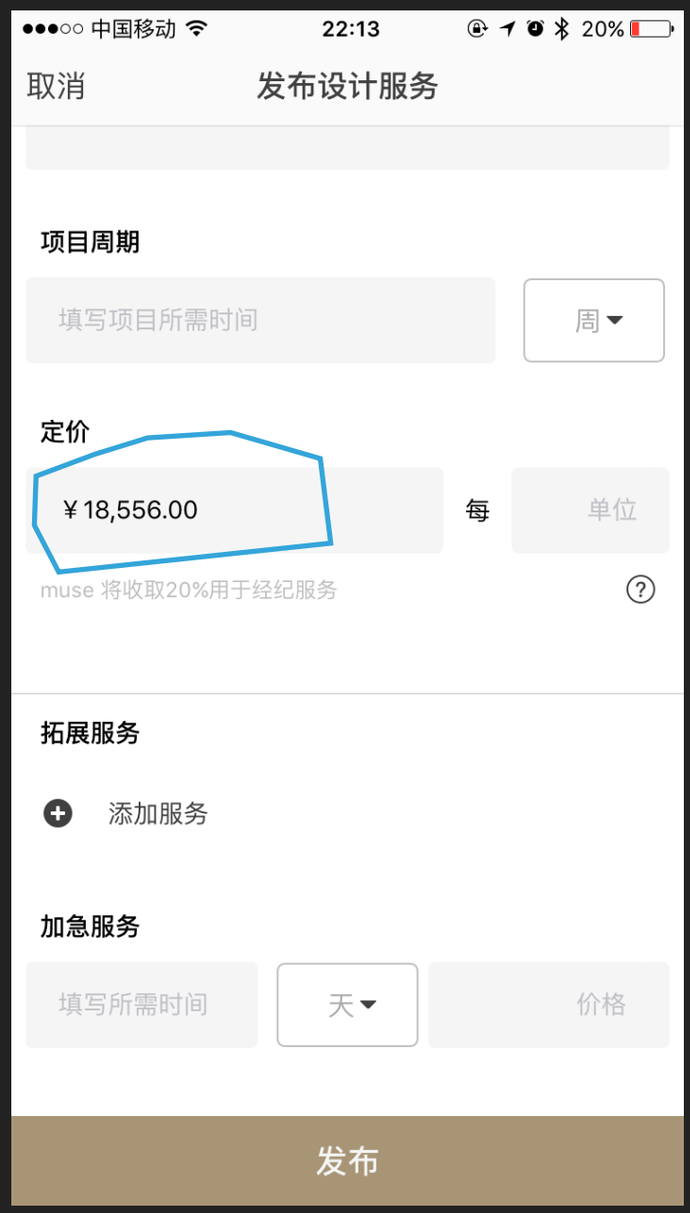
```
val order = Order(orderId = “orderId”)

val changedOrder  = order.copy(name=“orderName”,  workflowId = “0xjfk12df”)

 println(changedOrder)
 //Order(orderId=xxoxxx, extra=(xxxx), sellerId=xxxx, buyerId=xxxx)
```

DATA CLASS

- The primary constructor needs to have at least one parameter;
- All primary constructor parameters need to be marked as val or var;
- Data classes cannot be abstract, open, sealed or inner;
- Data classes may not extend other classes (but may implement interfaces).

EXTENSION FUNCTIONS->SCENE 0






```
fun Double.Yuan(): String{
    val formatter = DecimalFormat("0.00")
    formatter.roundingMode = RoundingMode.DOWN
    return "¥"+formatter.format(this)
}
```

```
final UserAPI userAPI = null;//some api interface here
userAPI.fetchUserInfoByUserId(123).enqueue(new Callback<HBUser>() {
    @Override
    public void onResponse(Response<HBUser> response, Retrofit
retrofit) {

    }

    @Override
    public void onFailure(Throwable t) {

    }
});
```

EXTENSION FUNCTIONS -> SCENE 1

```
fun <T> Call<T>.enqueueWithBlock(block: ((Throwable?, Response<T>?, Retrofit?)
->Unit)){
    this.enqueue(object : Callback<T>{
        override fun onFailure(t: Throwable?) {
            block(t, null, null)
        }

        override fun onResponse(response: Response<T>?, retrofit: Retrofit?) {
            block(null, response, retrofit)
        }
    })
}
```

With function extension

```
val userAPI = xxx
userAPI.fetchUserDetail(123).enqueueWithBlock { throwable, response,
retrofit ->
    response?.let {//do some logic here

    }
}
```

Farewell to findViewById

Kotlin Android Extensions

KOTLIN ANDROID EXTENSIONS

- apply plugin: ‘kotlin-android-extensions'
- import kotlinx.android.synthetic.main.<layout>.*
  // Don’t make mistake here. If you had more than one views with the same name, make sure import the correct synthetic file. *please don’t forget me*
- Let’s rock and roll!

MAGIC OF KOTLIN

DELEGATED PROPERTY

```
data class User(val name: String? = null,
                val age: Int = 0,
                val gender:Int =0)

var User.email: String by PropertyAttachedDelegation()

class PropertyAttachedDelegation<T>{
    operator fun getValue(thisRef: Any?, property: KProperty<*>): T {
        return “”
    }
    operator fun setValue(thisRef: Any?, property: KProperty<*>, value:
String){

    }
}
```

PROPERTY EXTENSION

```
data class User(val name: String? = null,
                val age: Int = 0,
                val gender:Int =0)

var User.email: String by PropertyAttachedDelegation()

class PropertyAttachedDelegation<T>{
    operator fun getValue(thisRef: Any?, property: KProperty<*>): T {
        return thisRef?.variablesMap?.get(property.toString()) as T
    }

    operator fun setValue(thisRef: Any?, property: KProperty<*>, value:
String){
        thisRef?.variablesMap?.set(property.toString(), value)
    }
}

private val Any.variablesMap by lazy {
    val mutableMap: MutableMap <String, Any> = mutableMapOf()
    mutableMap
}
```

- A DSL (Domain Specific Language) to makes the same logic easy to read, easy to write and there is no runtime overhead

```
val act = this
verticalLayout {
    val name = editText()
    button("Say Hello") {
        onClick { toast("Hello, ${name.text}!") }
    }
}
```

```
val act = this
val layout = LinearLayout(act)
layout.orientation = LinearLayout.VERTICAL
val name = EditText(act)
val button = Button(act)
button.text = "Say Hello"
button.setOnClickListener {
    Toast.makeText(act, "Hello, ${name.text}!", Toast.LENGTH_SHORT).show()
}
layout.addView(name)
layout.addView(button)
```

ANKO LIBRARY

```
val act = this
val layout = LinearLayout(act)
layout.orientation = LinearLayout.VERTICAL
val name = EditText(act)
val button = Button(act)
button.text = "Say Hello"
button.setOnClickListener {
    Toast.makeText(act, "Hello, ${name.text}!",Toast.LENGTH_SHORT).show()
}
layout.addView(name)
layout.addView(button)
```

```
verticalLayout {
    val name = editText()
    button("Say Hello") {
        onClick { toast("Hello, ${name.text}!") }
    }
}
```

- extensible

```
public inline fun ViewManager.mapView() = mapView {}
public inline fun ViewManager.mapView(init: MapView.() -> Unit) =
ankoView({ MapView(it) }, init)
```

DSL LAYOUT DEMO

- Partially defined listeners

```
purchaseStepPager.addOnPageChangeListener(object :
ViewPager.OnPageChangeListener {
    override fun onPageSelected(position: Int) {
        toast(“position:${position}")
    }

    override fun onPageScrollStateChanged(state: Int) {

    }

    override fun onPageScrolled(position: Int, positionOffset: Float,
positionOffsetPixels: Int) {

    }
})//Not cool, not style of Anko


purchaseStepPager.onPageChangeListener {
    onPageSelected { position ->
        toast(“position:${position}")
    }
}//Have a cup of coffee
```

CONCLUSIONS

- Easy to try and not hard to learn
- Code clean, complain less
- Android or iOS?

To be improved:

- Compile time is longer compared to Java

What if encounter a problem ???

TRAP

- Don’t trust Anko DSL review
- Realm
- Crash report track

THANKS

Q&A