MyDrive

Kotlin Extensions And Beyond

By Pascal How
Android Developer at MyDrive Solutions

About me

MyDrive

https://github.com/pascalhow

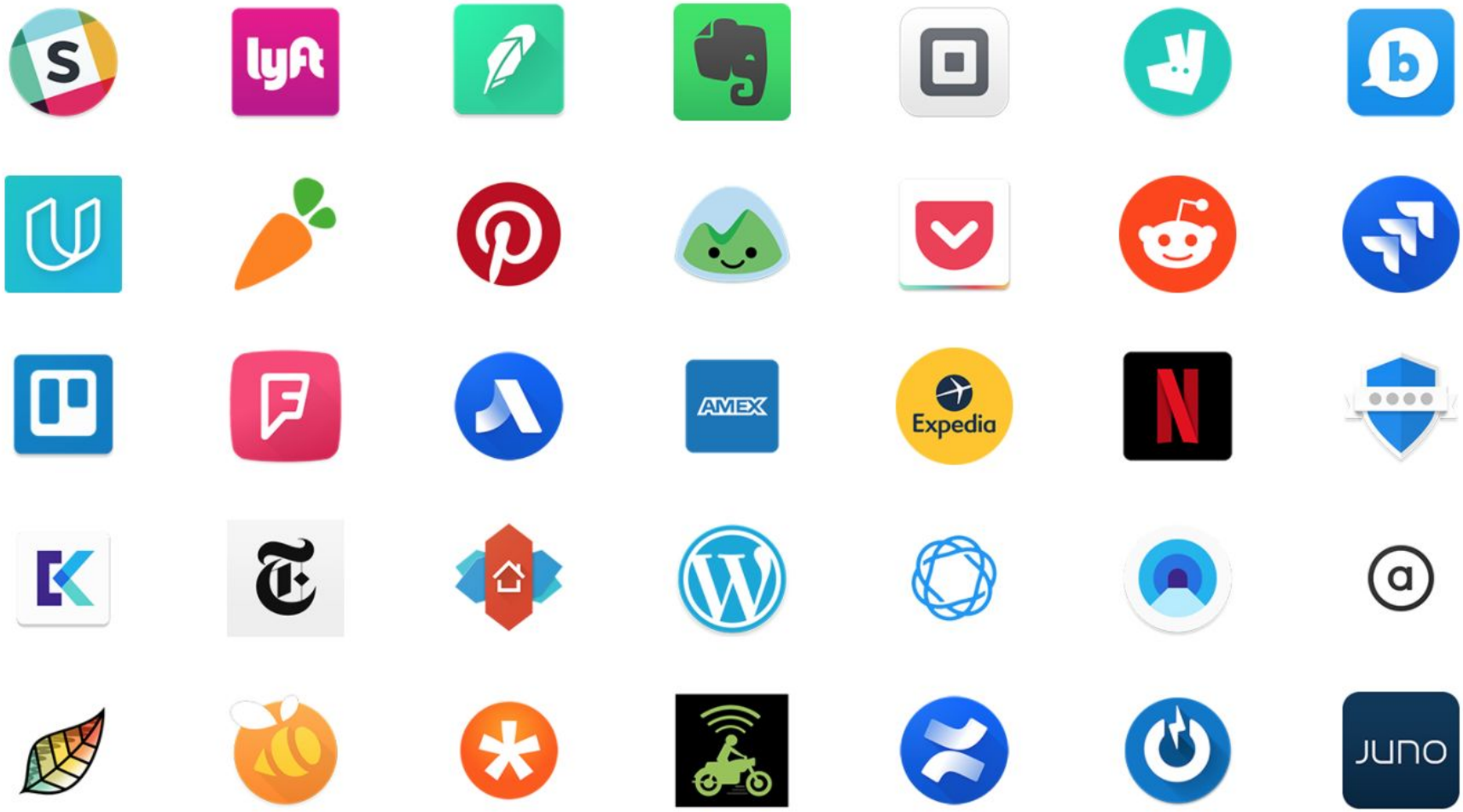
https://www.linkedin.com/in/pascal-how-35b3133b/

@pascal_how

MyDrive

Apps Built With Kotlin

MyDrive



Kotlin Extensions

Kotlin Extensions

MyDrive

Class Dog
bark( )
wagTail( )
run( )

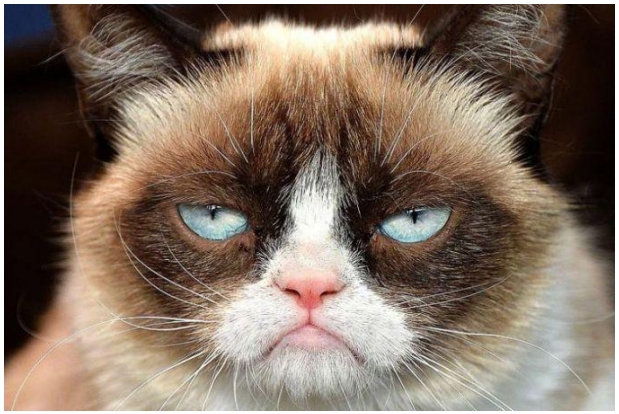
Class Cat
purr( )
scratch( )
jump( )

Kotlin Extensions

MyDrive

Class Dog
bark( ) wagTail( ) run( )
doTricks() follow(anotherDog)

Class Cat
purr( ) scratch( ) jump( )
break(preferablyGlass) tear(anyObject)



Why Extension Functions?

MyDrive

Add functionality to a class

Reducing boilerplate code

Less code = Less bugs

Why Extension Functions?

MyDrive

Add functionality to a class

Reducing boilerplate code

Less code = Less bugs

Why Extension Functions?

MyDrive

Add functionality to a class

Reducing boilerplate code

Less code = Less bugs

Implementation

Implementation And Usage

MyDrive

```
class Dog(var name: String) {
    fun bark() { }
    fun wagTail() { }
    fun run() { }
}


//  Elsewhere
val ninja = Dog("Nin Ja")
ninja.bark()
```

Implementation And Usage

MyDrive

```
class Dog(var name: String) {
    fun bark() { }
    fun wagTail() { }
    fun run() { }
}


//  Elsewhere
val ninja = Dog("Nin Ja")
ninja.bark()
val snoopy = Dog("Snoopy")
ninja.follow(snoopy)
```

Implementation And Usage

MyDrive

```
//  Usage
val ninja = Dog("Nin Ja")
ninja.bark()
val snoopy = Dog("Snoopy")
ninja.follow(snoopy)
```

Receiver Class

The Other Dog

```
fun Dog.follow(dog: Dog) {
    Timber.d("${this.name} follows ${dog.name} to the park")
}
```

Receiver Object

Implementation And Usage

MyDrive

```
//  Usage
val ninja = Dog("Nin Ja")
ninja.bark()
val snoopy = Dog("Snoopy")
ninja.follow(snoopy)
```

```
D/MainActivity: Nin Ja barks
D/MainActivity: Nin Ja follows Snoopy to the park
```

Under The Hood

MyDrive

DO NOT ACTUALLY ADD new members to a class

Resolved **STATICALLY**

Under The Hood

MyDrive

DO NOT ACTUALLY ADD new members to a class

Resolved **STATICALLY**

Under The Hood

MyDrive

Resolved **STATICALLY**

KotlinExtensions [~/Documents/Android/Learnings/KotlinExtensions] - .../app/src/main/java/com/pascalhow/kotlinextensions/extension/dog/DogExtensions.kt [app]

```kotlin
package com.pascalhow.kotlinextensions.extension.dog

import com.pascalhow.kotlinextensions.dog.Dog
import com.pascalhow.kotlinextensions.extension.dog.Breed.*
import com.pascalhow.kotlinextensions.staticdispatching.Pug
import timber.log.Timber

fun Dog.doTricks() = "${this.name} does a trick!"

fun Pug.doTricks() = "${this.name} fetches the ball"

fun Dog.follow(dog: Dog) {
    Timber.d("${this.name} follows ${dog.name} to the park")
}

enum class Breed {
    PUG, DOBERMAN,
    POODLE, CORGI,
    ROTTWEILER, GERMAN_SHEPHERD
}

fun Dog.Companion.isGentle(breed: Breed): Boolean {
    return when (breed) {
        PUG, POODLE, CORGI, GERMAN_SHEPHERD -> true
        DOBERMAN, ROTTWEILER -> false
    }
}
```

9: Version Control | Terminal | Build | 6: Logcat | TODO | Event Log

Gradle build finished in 2s 231ms (3 minutes ago)

28:1 LF UTF-8 Git: master Context: <no context>

Under The Hood

MyDrive

Resolved **STATICALLY**

Android Studio File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

KotlinExtensions [~/Documents/Android/Learnings/KotlinExtensions] - ...xtensions/extension/dog/DogExtensions.kt [app]

```
package com.pascalhow.kotlinextensions.extension.dog

import com.pascalhow.kotlinextensions.dog.Dog
import com.pascalhow.kotlinextensions.extension.dog.Breed.*
import com.pascalhow.kotlinextensions.staticdispatching.Pug
import timber.log.Timber

fun Dog.doTricks() = "${this.name} does a trick!"

fun Pug.doTricks() = "${this.name} fetches the ball"

fun Dog.follow(dog: Dog) {
    Timber.d("${this.name} follows ${dog.name} to the park")
}

enum class Breed {
    PUG, DOBERMAN,
    POODLE, CORGI,
    ROTTWEILER, GERMAN_SHEPHERD
}

fun Dog.Companion.isGentle(breed: Breed): Boolean {
    return when (breed) {
        PUG, POODLE, CORGI, GERMAN_SHEPHERD -> true
        DOBERMAN, ROTTWEILER -> false
    }
}
```

- AVD Manager
- SDK Manager
- Layout Inspector
- Theme Editor
- Firebase
- App Links Assistant
- Tasks & Contexts ▶
- Generate JavaDoc...
- IDE Scripting Console
- Create Command-line Launcher...
- JShell Console...
- Groovy Console...
- Kotlin ▶
  - Configure Kotlin Plugin Updates
  - Decompile Kotlin To Java
  - Kotlin Internal Mode
  - Kotlin REPL
  - Configure Kotlin in Project
  - Show Kotlin Bytecode

9: Version Control | Terminal | Build | 6: Logcat | TODO | Event Log

Gradle build finished in 2s 231ms (4 minutes ago)

28:1 LF UTF-8 Git: master Context: <no context>

Under The Hood

MyDrive

Resolved **STATICALLY**

KotlinExtensions [~/Documents/Android/Learnings/KotlinExtensions] - .../app/src/main/java/com/pascalhow/kotlinextensions/extension/dog/DogExtensions.kt [app]

```kotlin
package com.pascalhow.kotlinextensions.extension.dog

import com.pascalhow.kotlinextensions.dog.Dog
import com.pascalhow.kotlinextensions.extension.dog.Breed.*
import com.pascalhow.kotlinextensions.staticdispatching.Pug
import timber.log.Timber

fun Dog.doTricks() = "${this.name} does a trick!"

fun Pug.doTricks() = "${this.name} fetches the ball"

fun Dog.follow(dog: Dog) {
    Timber.d("${this.name} follows ${dog.name} to the park")
}

enum class Breed {
    PUG, DOBERMAN,
    POODLE, CORGI,
    ROTTWEILER, GERMAN_SHEPHERD
}

fun Dog.Companion.isGentle(breed: Breed): Boolean {
    return when (breed) {
        PUG, POODLE, CORGI, GERMAN_SHEPHERD -> true
        DOBERMAN, ROTTWEILER -> false
    }
}
```

Kotlin Bytecode

Decompile ☑ Inline ☑ Optimization ☑ Assertions ☐ IR

```
// ===============com/pascalhow/kotlinextensions/extensi
// class version 50.0 (50)
// access flags 0x4031
// signature Ljava/lang/Enum<Lcom/pascalhow/kotlinextensi
// declaration: com/pascalhow/kotlinextensions/extension/
public final enum com/pascalhow/kotlinextensions/extensio


  // access flags 0x4019
  public final static enum Lcom/pascalhow/kotlinextension

  // access flags 0x4019
  public final static enum Lcom/pascalhow/kotlinextension

  // access flags 0x4019
  public final static enum Lcom/pascalhow/kotlinextension

  // access flags 0x4019
  public final static enum Lcom/pascalhow/kotlinextension

  // access flags 0x4019
  public final static enum Lcom/pascalhow/kotlinextension

  // access flags 0x4019
  public final static enum Lcom/pascalhow/kotlinextension

  // access flags 0x8
  static <clinit>()V
    BIPUSH 6
    ANEWARRAY com/pascalhow/kotlinextensions/extension/do
    DUP
    DUP
    ICONST_0
    NEW com/pascalhow/kotlinextensions/extension/dog/Bree
    DUP
    LDC "PUG"
    ICONST_0
    INVOKESPECIAL com/pascalhow/kotlinextensions/extensio
    DUP
    PUTSTATIC com/pascalhow/kotlinextensions/extension/do
    AASTORE
    DUP
    ICONST_1
```

9: Version Control | Terminal | Build | 6: Logcat | TODO | Event Log

Gradle build finished in 2s 231ms (5 minutes ago)

28:1 LF UTF-8 Git: master Context: <no context>

Under The Hood

MyDrive

Resolved **STATICALLY**

KotlinExtensions [~/Documents/Android/Learnings/KotlinExtensions] - kotlinDecompiled/DogExtensions.decompiled.java

```
37      d1 = {"\u0000(\n\u0000\n\u0002\u0010\u000e\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0000\n\u0002\u0010\u0002\n\u00
38      d2 = {"doTricks", "", "Lcom/pascalhow/kotlinextensions/dog/Dog;", "Lcom/pascalhow/kotlinextensions/staticdispatching
39  )
40  public final class DogExtensionsKt {
41      @NotNull
42  @   public static final String doTricks(@NotNull Dog $receiver) {
43          Intrinsics.checkParameterIsNotNull($receiver, "$receiver");
44          return "" + $receiver.getName() + " does a trick!";
45      }
46
47      @NotNull
48  @   public static final String doTricks(@NotNull Pug $receiver) {
49          Intrinsics.checkParameterIsNotNull($receiver, "$receiver");
50          return "" + $receiver.getName() + " fetches the ball";
51      }
52
53      public static final void follow(@NotNull Dog $receiver, @NotNull Dog dog) {
54          Intrinsics.checkParameterIsNotNull($receiver, "$receiver");
55          Intrinsics.checkParameterIsNotNull(dog, "dog");
56          Timber.d("" + $receiver.getName() + " follows " + dog.getName() + " to the park", new Object[0]);
57      }
58
59      public static final boolean isGentle(@NotNull Companion $receiver, @NotNull Breed breed) {
60          Intrinsics.checkParameterIsNotNull($receiver, "$receiver");
61          Intrinsics.checkParameterIsNotNull(breed, "breed");
62          boolean var10000;
63          switch(DogExtensionsKt$WhenMappings.$EnumSwitchMapping$0[breed.ordinal()]) {
64          case 1:
65          case 2:
66          case 3:
67          case 4:
68              var10000 = true;
```

DogExtensionsKt › follow()

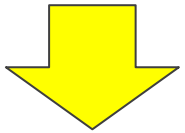
9: Version Control Terminal Build 6: Logcat TODO Event Log

Cannot resolve symbol 'Dog' 53:46 LF UTF-8 Git: master Context: <no context>

Under The Hood

MyDrive

```kotlin
//  Kotlin code
fun Dog.doTricks() {
    Timber.d("${this.name} does a trick!")
}
```



```java
//  Java Code
public static final void doTricks(Dog dog) {
  Timber.d(dog.getName() + " does a trick!");
}
```

Limitations

MyDrive

Member function **ALWAYS** wins

CANNOT OVERRIDE member functions

Adding Static Functions

MyDrive

```
class Dog(var name: String) {
    fun bark() { }
    fun wagTail() { }
    fun run() { }
}
```

Adding Static Functions

MyDrive

```
class Dog(var name: String) {
    fun bark() { }
    fun wagTail() { }
    fun run() { }
    companion object
}
```

Adding Static Functions

MyDrive

```
enum class Breed {
    PUG, DOBERMAN,
    POODLE, CORGI,
    ROTTWEILER, GERMAN_SHEPHERD
}

fun Dog.Companion.isGentle(breed: Breed): Boolean {
    return when (breed) {
        PUG, POODLE, CORGI, GERMAN_SHEPHERD -> true
        DOBERMAN, ROTTWEILER -> false
    }
}
```

Adding Static Functions

MyDrive

```
val isGoodDog = Dog.isGentle(Breed.CORGI)
```

Limitations

MyDrive

MUST have a companion object

Static Dispatching

MyDrive

```
open class Dog(val name: String) {
    open fun run() {
        Timber.d("$name runs around the house")
    }
}

class Pug(name: String) : Dog(name) {
    override fun run() {
        Timber.d("Pugs prefer to walk instead")
    }
}
```

Static Dispatching

MyDrive

```
fun Dog.doTricks() =
Timber.d("${this.name} does a trick!")
```

```
fun Pug.doTricks() =
Timber.d("${this.name} fetches the ball")
```

Static Dispatching

MyDrive

```
val pug: Dog = Pug("Goofy")
val otherPug = Pug("Dingo")

pug.run()
otherPug.run()

pug.doTricks()
otherPug.doTricks()
```

Static Dispatching

MyDrive

```
D/MainActivity: Pugs prefer to walk instead
D/MainActivity: Pugs prefer to walk instead

D/MainActivity: Goofy does a trick!
D/MainActivity: Dingo fetches the ball
```

Static Dispatching

MyDrive

```
open class Dog(val name: String) {
    open fun run() {
        Timber.d("$name runs around the house")
    }
}

class Pug(name: String) : Dog(name) {
    override fun run() {
        Timber.d("Pugs prefer to walk instead")
    }
}
```

```
D/MainActivity: Pugs prefer to walk instead
D/MainActivity: Pugs prefer to walk instead
```

Static Dispatching

MyDrive

```
fun Dog.doTricks() =
Timber.d("${this.name} does a trick!")

fun Pug.doTricks() =
Timber.d("${this.name} fetches the ball")
```

```
D/MainActivity: Goofy does a trick!
D/MainActivity: Dingo fetches the ball
```

Do You Test?

MyDrive

```
fun String?.reverseOrderOfWords(): String {
    return this?.let {
        split(" ").reversed().joinToString(" ")
    } ?: "Cannot reverse this :("
}
```

Do You Test?

MyDrive

```
@Test
fun `givenStringNotNull_whenReverseOrder_thenOutputWordsAreReversed`() {
    val message = "This is just a random string"
    val expectedString = "string random a just is This"
    val actualString = message.reverseOrderOfWords()
    assertEquals(actualString, expectedString)
}

@Test
fun `givenNullString_whenReverseOrder_thenOutputErrorString`() {
    val message: String? = null
    val expectedString = "Cannot reverse this :("
    val actualString = message.reverseOrderOfWords()
    assertEquals(actualString, expectedString)
}
```

Interoping With Java

MyDrive

💯 compatible with Java

Interoping With Java

MyDrive

StringExt.kt

```
fun String?.reverseOrderOfWords(): String {
   //   Reverse order of words
}
```

Kotlin

```
val message = "Kotlin extensions are cool!"
val reversedMessage = message.reverseOrderOfWords()
```

Interoping With Java

MyDrive

Remember?

Resolved **STATICALLY**

Interoping With Java

MyDrive

StringExt.kt

```
fun String?.reverseOrderOfWords(): String {
   //   Reverse order of words
}
```

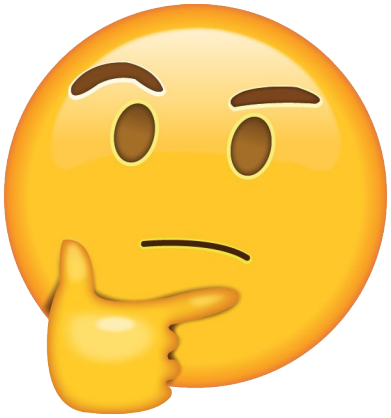
Java

```
String message = "Kotlin extensions are cool!";
String reversedMessage = StringExtKt.reverseOrderOfWords(message);
```

Interoping With Java

MyDrive

Do I really need all my classes to be named after Kt?



Interoping With Java

MyDrive

StringExt.kt

```kotlin
@file:JvmName("StringHelper")

fun String?.reverseOrderOfWords(): String {
    //   Reverse order of words
}
```

Java

```java
String message = "Kotlin extensions are cool!";
String reversedMessage = StringHelper.reverseOrderOfWords(message);
```

Interoping With Java

MyDrive

StringExt.kt

```kotlin
@file:JvmName("StringHelper")

fun String?.reverseOrderOfWords(): String {
    //   Reverse order of words
}
```

Java

```java
String message = "Kotlin extensions are cool!";
String reversedMessage = StringHelper.reverseOrderOfWords(message);
```

Interoping With Java

MyDrive

Refactoring Existing Java Code

```
//  Java code somewhere somewhere
String regex = "#{date}";
String title = "Travelling from London on #{date}";
String format = "dd MM YYYY"
Long timestamp = 123456789L;



String formattedTitle = StringHelper.replaceDate(
        "#{date}", // regex
        "Travelling from London on #{date}", // title
        "dd MM YYYY", // format
        123456789L, // timestamp
);
```

Interoping With Java

MyDrive

```
@file:JvmName("StringHelper")
fun String.replaceDate(regex: String, format: String, timestamp:
Long): String {
    return this.replace(
            regex,
            DateTimeFormat.forPattern(dateFormat)
                    .print(DateTime(timestamp))
    )
}
```

```
org.junit.ComparisonFailure:
Expected :Travelling from London on 02 01 1970
Actual   :{#date}
```

Interoping With Java

MyDrive

```
String formattedTitle = StringHelper.replaceDate(
        regex,
        title,
        format,
        timestamp
);
```

```
String formattedTitle = StringHelper.replaceDate(
        title,
        regex,
        format,
        timestamp
);
```

Higher Order Functions

Higher Order Functions

MyDrive

```
private fun performAtDogShow() {
    val trainer = Trainer()
    val dog = Dog(name)

    trainer.pats(dog)

    dog.wagTail()
    dog.fetchesBall()
    dog.roll()
    dog.doBackFlip()

    trainer.treats(dog)
}
```

Higher Order Functions

MyDrive

```
private fun performAtDogShow() {
    val trainer = Trainer()
    val dog = Dog(name)

    trainer.pats(dog)

    dog.wagTail()
    dog.fetchesBall()
    dog.roll()
    dog.doBackFlip()

    trainer.treats(dog)
}
```

Higher Order Functions

MyDrive

```
private fun unleash(name: String, doTricks: (Dog) -> Unit) {
    val trainer = Trainer()
    val dog = Dog(name)
    trainer.pats(dog)
    doTricks(dog)
    trainer.treats(dog)
}


private val doTricks: (Dog) -> Unit = { dog ->
    dog.wagTail()
    dog.fetchesBall()
    dog.roll()
    dog.doBackFlip()
}
```

Higher Order Functions

MyDrive

```
private fun unleash(name: String, doTricks: (Dog) -> Unit) {
    val trainer = Trainer()
    val dog = Dog(name)
    trainer.pats(dog)
    doTricks(dog)
    trainer.treats(dog)
}


private val doTricks: (Dog) -> Unit = { dog ->
    dog.wagTail()
    dog.fetchesBall()
    dog.roll()
    dog.doBackFlip()
}
```

Higher Order Functions

MyDrive

```
private val doTricks: (Dog) -> Unit = { dog ->
    dog.wagTail()
    dog.fetchesBall()
    dog.roll()
    dog.doBackFlip()
}


private fun performAtDogShow() {
    unleash("Nin Ja", doTricks)
}
```

Higher Order Functions

MyDrive

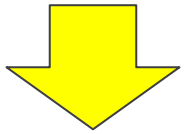
```
private val doTricks: (Dog) -> Unit = { dog ->
    dog.wagTail()
    dog.fetchesBall()
    dog.roll()
    dog.doBackFlip()
}

private val doTricks: (Dog) -> Unit = {
    it.wagTail()
    it.fetchesBall()
    it.roll()
    it.doBackFlip()
}
```

Lambdas With Receivers

Lambdas With Receivers

MyDrive

```
private val doTricks: (Dog) -> Unit = { dog ->
    dog.wagTail()
    dog.fetchesBall()
    dog.roll()
    dog.doBackFlip()
}
```

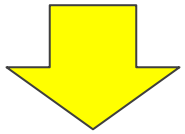


```
private val doTricks: Dog.() -> Unit = {
    wagTail()
    fetchesBall()
    roll()
    doBackFlip()
}
```

Lambdas With Receivers

MyDrive

```
private fun unleash(name: String, doTricks: (Dog) -> Unit) {
    val trainer = Trainer()
    val dog = Dog(name)
    trainer.pats(dog)
    doTricks(dog)
    trainer.treats(dog)
}
```

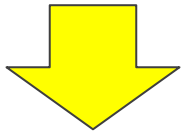


```
private fun unleash(name: String, doTricks: Dog.() -> Unit) {
    val trainer = Trainer()
    val dog = Dog(name)
    trainer.pats(dog)
    doTricks(dog)
    trainer.treats(dog)
}
```

Lambdas With Receivers

MyDrive

```
private fun unleash(name: String, doTricks: (Dog) -> Unit) {
    val trainer = Trainer()
    val dog = Dog(name)
    trainer.pats(dog)
    doTricks(dog)
    trainer.treats(dog)
}
```



```
private fun unleash(name: String, doTricks: Dog.() -> Unit) {
    val trainer = Trainer()
    val dog = Dog(name)
    trainer.pats(dog)
    dog.doTricks()
    trainer.treats(dog)
}
```

Lambdas With Receivers

MyDrive

```
private inline fun unleash(name: String, doTricks: Dog.() -> Unit) {
    val trainer = Trainer()
    val dog = Dog(name)
    trainer.pats(dog)
    dog.doTricks()
    trainer.treats(dog)
}
```

Lambdas With Receivers

MyDrive

Inline

```
private final void unleash(String name, Function1 doTricks) {
    Trainer trainer = new Trainer();
    Dog dog = new Dog(name);
    trainer.pats(dog);
    doTricks.invoke(dog);
    trainer.treats(dog);
}

private final void performAtDogShow() {
   String name$iv = "Ninja";
   Trainer trainer$iv = new Trainer();
   Dog dog$iv = new Dog(name$iv);
   trainer$iv.pats(dog$iv);
   dog$iv.wagTail();
   dog$iv.fetchesBall();
   dog$iv.roll();
   dog$iv.doBackFlip();
   trainer$iv.treats(dog$iv);
}
```

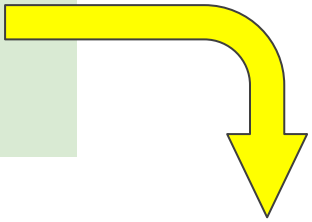
NO Inline

```
private final void unleash(String name, Function1 doTricks) {
   Trainer trainer = new Trainer();
   Dog dog = new Dog(name);
   trainer.pats(dog);
   doTricks.invoke(dog);
   trainer.treats(dog);
}

private final void performAtDogShow() {
    this.unleash("Ninja", (Function1)null.INSTANCE);
}
```

Lambdas With Receivers

MyDrive

```
private val doTricks: Dog.() -> Unit = {
    wagTail()
    fetchesBall()
    roll()
    doBackFlip()
}
```



```
private final void unleash(String name, Function1 doTricks) {
    Trainer trainer = new Trainer();
    Dog dog = new Dog(name);
    trainer.pats(dog);
    doTricks.invoke(dog);
    trainer.treats(dog);
}
```

No Inline

MyDrive

```
private final void performAtDogShow() {
    this.unleash("Ninja", (Function1)null.INSTANCE);
}
```

No Inline

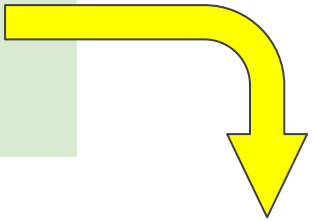
MyDrive

```
private final void performAtDogShow() {
    unleash("Ninja", new Function() {
        @Override
        public void invoke() {
            wagTail();
            fetchesBall();
            roll();
            doBackFlip();
        }
    });
}
```

With Inline

MyDrive

```
private val doTricks: Dog.() -> Unit = {
    wagTail()
    fetchesBall()
    roll()
    doBackFlip()
}
```



```
private final void unleash(String name, Function1 doTricks) {
    Trainer trainer = new Trainer();
    Dog dog = new Dog(name);
    trainer.pats(dog);
    doTricks.invoke(dog);
    trainer.treats(dog);
}
```

With Inline

MyDrive

```
private val doTricks: Dog.() -> Unit = {
    wagTail()
    fetchesBall()
    roll()
    doBackFlip()
}
```

```
private final void unleash(String name, Function1 doTricks) {
    Trainer trainer = new Trainer();
    Dog dog = new Dog(name);
    trainer.pats(dog);
    doTricks.invoke(dog);
    trainer.treats(dog);
}
```

With Inline

MyDrive

```
private final void performAtDogShow() {
    String name$iv = "Ninja";
    Trainer trainer$iv = new Trainer();
    Dog dog$iv = new Dog(name$iv);
    trainer$iv.pats(dog$iv);
    dog$iv.wagTail();
    dog$iv.fetchesBall();
    dog$iv.roll();
    dog$iv.doBackFlip();
    trainer$iv.treats(dog$iv);
}
```

Lambdas With Receivers

MyDrive

Inline

```
private final void performAtDogShow() {
    String name$iv = "Ninja";
    Trainer trainer$iv = new Trainer();
    Dog dog$iv = new Dog(name$iv);
    trainer$iv.pats(dog$iv);
    dog$iv.wagTail();
    dog$iv.fetchesBall();
    dog$iv.roll();
    dog$iv.doBackFlip();
    trainer$iv.treats(dog$iv);
}
```

NO Inline

```
private final void performAtDogShow() {
    this.unleash(
            "Ninja",
            (Function1) null.INSTANCE
    );
}
```

Lambdas With Receivers

MyDrive

```
private inline fun unleash(name: String, doTricks: Dog.() -> Unit) {
    val trainer = Trainer()
    val dog = Dog(name)
    trainer.pats(dog)
    dog.doTricks()
    trainer.treats(dog)
}
```

```
private fun performAtDogShow() {
    unleash("Nin Ja", doTricks)
}
```

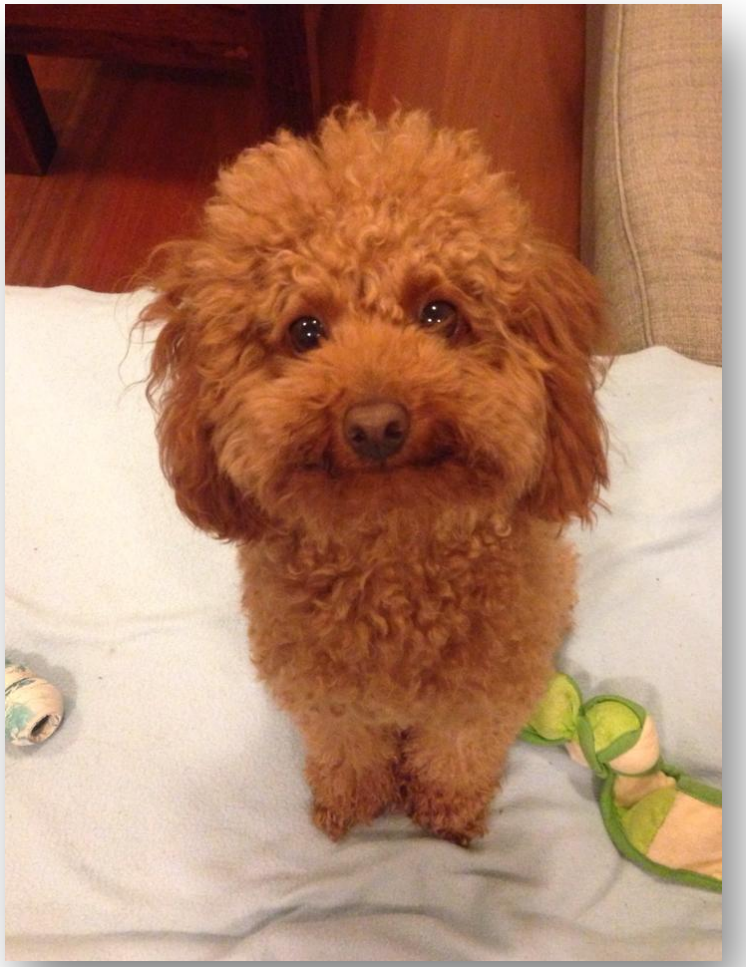
```
private val doTricks: Dog.() -> Unit = {
    wagTail()
    fetchesBall()
    roll()
    doBackFlip()
}
```

Lambdas With Receivers

MyDrive

```
private fun performAtDogShow() {
    unleash("Nin Ja", doTricks)
}
```

```
D/Dog: Nin Ja wags tail
D/Dog: Nin Ja fetches ball
D/Dog: Nin Ja rolls
D/Dog: Nin Ja does a backflip
```



Now What?

Extend All The Things

Extend All The Things

MyDrive

Android Views

```
fun View.setVisible(visible: Boolean) {
    this.visibility = if (visible) View.VISIBLE else View.GONE
}
```

```
//  Elsewhere
myButtonView.setVisible(true)
myImageView.setVisible(false)
```

Extend All The Things

MyDrive

SharedPreferences

```
//  Classic usage of SharedPreferences
```

```
fun saveDog() {
    val editor = prefs.edit()
    editor.putString("name", "Nin Ja")
    editor.putInt("age", 7)
    editor.apply()
}
```

```
fun saveDog() {
    prefs.edit {
        putString("name", "Nin Ja")
        putInt("age", 7)
    }
}
```

High Order Functions

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    val editor = prefs.edit()
    editor.putString("name", "Nin Ja")
    editor.putInt("age", 7)
    editor.apply()
}
```

```
fun SharedPreferences.edit(editor: SharedPreferences.Editor, action: () -> Unit)
{
    action()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    val editor = prefs.edit()
    prefs.edit(editor) {
        editor.putString("name", "Nin Ja")
        editor.putInt("age", 7)
    }
    editor.apply()
}
```

```
fun SharedPreferences.edit(editor: SharedPreferences.Editor, action: () -> Unit)
{
    action()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    val editor = prefs.edit()
    prefs.edit(editor) {
        editor.putString("name", "Nin Ja")
        editor.putInt("age", 7)
    }
    editor.apply()
}
```

```
fun SharedPreferences.edit(editor: SharedPreferences.Editor, action: () -> Unit)
{
    action()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
   prefs.edit(editor) {
           editor.putString("name", "Nin Ja")
           editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(editor: SharedPreferences.Editor, action: () -> Unit)
{
     val editor = edit()
     action()
     editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
   prefs.edit(editor) {
           editor.putString("name", "Nin Ja")
           editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(editor: SharedPreferences.Editor, action: () ->
Unit) {
     val editor = edit()
     action()
     editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
   prefs.edit() {
           editor.putString("name", "Nin Ja")
           editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: () -> Unit) {
     val editor = edit()
     action()
     editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
   prefs.edit() {
           editor.putString("name", "Nin Ja")
           editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: () -> Unit) {
     val editor = edit()
     action()
     editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    prefs.edit { editor ->
        editor.putString("name", "Nin Ja")
        editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: (SharedPreferences.Editor) -> Unit) {
    val editor = edit()
    action(editor)
    editor.apply()
}
```

Lambdas With Receivers

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
     prefs.edit { editor ->
         editor.putString("name", "Nin Ja")
         editor.putInt("age", 7)
     }
}
```

```
fun SharedPreferences.edit(action: (SharedPreferences.Editor) -> Unit) {
    val editor = edit()
    action(editor)
    editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    prefs.edit { editor ->
        editor.putString("name", "Nin Ja")
        editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: (SharedPreferences.Editor) -> Unit) {
    val editor = edit()
    action(editor)
    editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    prefs.edit { editor ->
        editor.putString("name", "Nin Ja")
        editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: (SharedPreferences.Editor) -> Unit) {
    val editor = edit()
    action(editor)
    editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    prefs.edit { editor ->
        editor.putString("name", "Nin Ja")
        editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: SharedPreferences.Editor.() -> Unit) {
    val editor = edit()
    action(editor)
    editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    prefs.edit { editor ->
        editor.putString("name", "Nin Ja")
        editor.putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: SharedPreferences.Editor.() -> Unit) {
    val editor = edit()
    action(editor)
    editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
fun saveDog() {
    prefs.edit {
        putString("name", "Nin Ja")
        putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: SharedPreferences.Editor.() -> Unit) {
    val editor = edit()
    editor.action()
    editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

```
//  More Kotlinesque code
```

```
fun saveDog() {
    prefs.edit {
        putString("name", "Nin Ja")
        putInt("age", 7)
    }
}
```

```
fun SharedPreferences.edit(action: SharedPreferences.Editor.() -> Unit) {
    val editor = edit()
    editor.action()
    editor.apply()
}
```

Extend All The Things

MyDrive

SharedPreferences

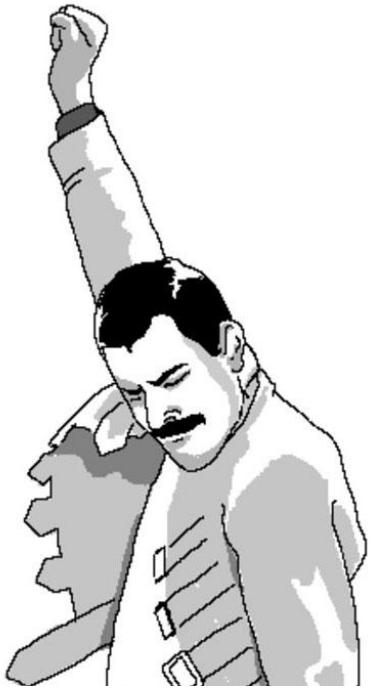
```
//  More Kotlinesque code
```

```
fun saveDog() {
    prefs.edit {
        putString("name", "Nin Ja")
        putInt("age", 7)
    }
}
```

```
inline fun SharedPreferences.edit(action: SharedPreferences.Editor.() -> Unit) {
    val editor = edit()
    editor.action()
    editor.apply()
}
```

Extend All The Things

MyDrive



Extend All The Things

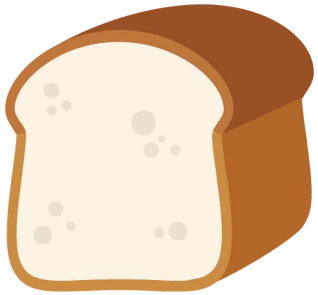
MyDrive

```
fun Context.loadImageCenterCrop(url: String, imageView: ImageView) {
  Picasso.withContext(this)
    .load(url)
    .centerCrop()
    .fit()
    .into(imageView)
}


// Elsewhere
context.loadImageCenterCrop("http://example.com/image.jpg", myImageView)
```

Extend All The Things

MyDrive

```
fun Context.toast(message: CharSequence,
                  duration: Int = Toast.LENGTH_SHORT): Toast {
    return Toast.makeText(this, message, duration).apply { show() }
}
```



```
// Elsewhere
toast("Message sent!")
```

Extend All The Things

MyDrive

```kotlin
fun View.showDialog(activity: Activity) {
    val builder = AlertDialog.Builder(activity)
    builder.setView(this)
    builder.setIcon(R.drawable.ic_launcher_background)
    builder.setTitle("Alert Dialog")
    builder.setMessage("This is bad use of Kotlin extensions")
    builder.setPositiveButton("OK", null)
    builder.setNegativeButton("Cancel", null)
    builder.create()
    builder.show()
}

// Elsewhere
myCustomView.showDialog(this)
```

MyDrive

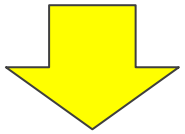
Stop! What Do You Think You Are Doing!?

Abusing Extension Functions

Abusing Extension Functions

MyDrive

```
fun Context.loadImageCenterCrop(url: String, imageView: ImageView) {
  Picasso.withContext(this)
    .load(url)
    .centerCrop()
    .fit()
    .into(imageView)
}
```



```
fun ImageView.loadImageCenterCrop(url: String) {
  Picasso.withContext(this.getContext())
    .load(url)
    .centerCrop()
    .fit()
    .into(this)
}
```

Abusing Extension Functions

MyDrive

```
fun Context.toast(message: CharSequence,
                  duration: Int = Toast.LENGTH_SHORT): Toast {
    return Toast.makeText(this, message, duration).apply { show() }
}
```

```
// Elsewhere
toast("Message sent!")
```

Abusing Extension Functions

MyDrive

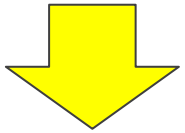
```
fun Any.showToast(context: Context,
              duration: Int = Toast.LENGTH_SHORT): Toast {
    return Toast.makeText(context, this.toString(), duration).apply {
show() }
}
```

```
// Elsewhere
"Good morning Pisa!".showToast(context)
"Good morning ${conference.name}!".showToast(context)
```

Abusing Extension Functions

MyDrive

```kotlin
fun View.showDialog(activity: Activity) {
    val builder = AlertDialog.Builder(activity)
    builder.setView(this)
    builder.setIcon(R.drawable.ic_launcher_background)
    builder.setMessage("Abusing Kotlin extensions")
    builder.setPositiveButton("OK", null)
    builder.create()
    builder.show()
}
```



```kotlin
fun showDialog(activity: Activity, customView: View) {
    //  Customise dialog here
}
```

Abusing Extension Functions

MyDrive

With Great Power Comes Great Responsibilities

Some Tips

Tips

MyDrive

Naming Conventions For Extension Files

Decide On Extension Files Locations

Avoid Repetition

Don’t Get Carried Away

Consider Creating A Centralised Resource

Don’t Use Same Signature As Member Functions

Android KTX

Tips

MyDrive

Naming Conventions For Extension Files

Decide On Extension Files Locations

Avoid Repetition

Don’t Get Carried Away

Consider Creating A Centralised Resource

Don’t Use Same Signature As Member Functions

Android KTX

Tips

MyDrive

Naming Conventions For Extension Files

Decide On Extension Files Locations

Avoid Repetition

Don’t Get Carried Away

Consider Creating A Centralised Resource

Don’t Use Same Signature As Member Functions

Android KTX

Tips

MyDrive

Naming Conventions For Extension Files

Decide On Extension Files Locations

Avoid Repetition

Don’t Get Carried Away

Consider Creating A Centralised Resource

Don’t Use Same Signature As Member Functions

Android KTX

Tips

MyDrive

Naming Conventions For Extension Files

Decide On Extension Files Locations

Avoid Repetition

Don’t Get Carried Away

Consider Creating A Centralised Resource

Don’t Use Same Signature As Member Functions

Android KTX

Tips

MyDrive

Naming Conventions For Extension Files

Decide On Extension Files Locations

Avoid Repetition

Don’t Get Carried Away

Consider Creating A Centralised Resource

Don’t Use Same Signature As Member Functions

Android KTX

Tips

MyDrive

Naming Conventions For Extension Files

Decide On Extension Files Locations

Avoid Repetition

Don’t Get Carried Away

Consider Creating A Centralised Resource

Don’t Use Same Signature As Member Fu

Android KTX


MyDrive
Any Questions?