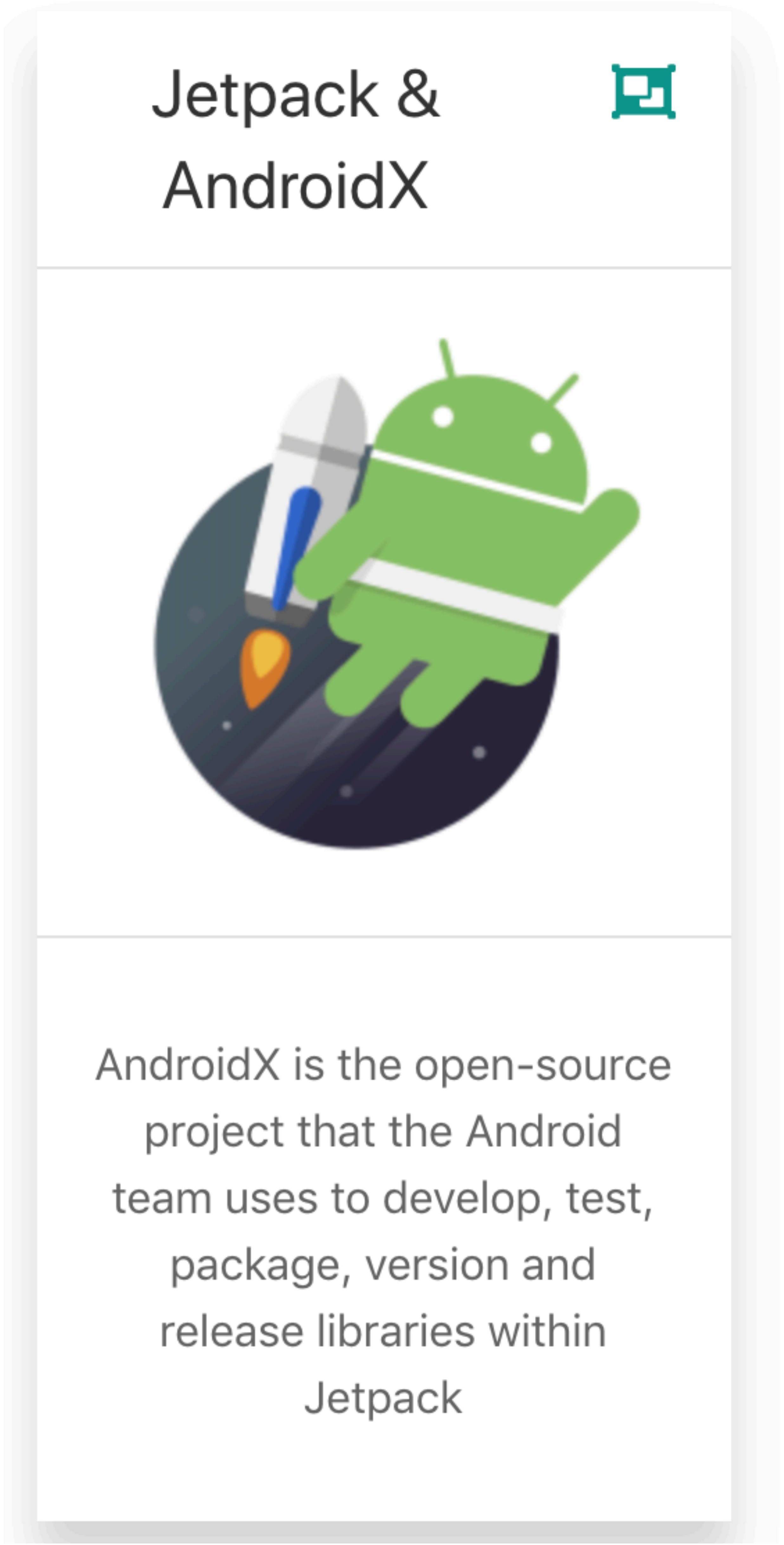
Jetpack & AndroidX



Jetpack & AndroidX

AndroidX is the open-source project that the Android team uses to develop, test, package, version and release libraries within Jetpack

Android Jetpack

Jetpack is a collection of Android software components to make it easier for you to develop great Android apps. These components help you follow best practices, free you from writing boilerplate code, and simplify complex tasks, so you can focus on the code you care about.

Jetpack comprises the androidx.* package libraries, unbundled from the platform APIs. This means that it offers backward compatibility and is updated more frequently than the Android platform, making sure you always have access to the latest and greatest versions of the Jetpack components.

https://developer.android.com/jetpack/

Jetpack

Introduced in 2017 to repackage existing libraries + introduce a range of new / modernised libraries

These are ‘unbundled’ from the SDK, supporting older releases of android

AndroidX

The namespace + versioning for the Jetpack libraries

Android Jetpack Components

Android Jetpack components are a collection of libraries that are individually adoptable and built to work together while taking advantage of Kotlin language features that make you more productive. Use them all or mix and match!

Foundation

Foundation components provide cross-cutting functionality like backwards compatibility, testing and Kotlin language support.

Architecture

Architecture components help you design robust, testable and maintainable apps.

Behavior

Behavior components help your app integrate with standard Android services like notifications, permissions, sharing and the Assistant.

UI

UI components provide widgets and helpers to make your app not only easy, but delightful to use.

Accelerate development

Components are individually adoptable but built to work together while taking advantage of Kotlin language features that make you more productive.

Eliminate boilerplate code

Android Jetpack manages tedious activities like background tasks, navigation, and lifecycle management, so you can focus on what makes your app great.

Build high quality, robust apps

Built around modern design practices, Android Jetpack components enable fewer crashes and less memory leaked with backwards-compatibility baked in.

Foundation

Foundation components provide cross-cutting functionality like backwards compatibility, testing and Kotlin language support.

Android KTX

Write more concise, idiomatic Kotlin code

AppCompat

Degrade gracefully on older versions of Android

Car

Components to help develop Android apps for cars

Benchmark

Quickly benchmark your Kotlin-based or Java-based code from within Android Studio

Multidex

Provide support for apps with multiple DEX files

Security

Read and write encrypted files and shared preferences by following security best practices.

Test

An Android testing framework for unit and runtime UI tests

TV

Components to help develop apps for Android TV

Wear OS by Google

Components to help develop apps for Wear

Architecture

Architecture components help you design robust, testable and maintainable apps.

Data Binding

Declaratively bind observable data to UI elements

Lifecycles

Manage your activity and fragment lifecycles

LiveData

Notify views when underlying database changes

Navigation

Handle everything needed for in-app navigation

Paging

Gradually load information on demand from your data source

Room

Fluent SQLite database access

ViewModel

Manage UI-related data in a lifecycle-conscious way

WorkManager

Manage your Android background jobs

Behavior

Behavior components help your app integrate with standard Android services like notifications, permissions, sharing and the Assistant.

CameraX

Easily add camera capabilities to your apps

Media & playback

Backwards-compatible APIs for media playback and routing (including Google Cast)

Notifications

Provides a backwards-compatible notification API with support for Wear and Auto

Permissions

Compatibility APIs for checking and requesting app permissions

Preferences

Create interactive settings screens

Sharing

Provides a share action suitable for an app’s action bar

Slices

Create flexible UI elements that can display app data outside the app

UI

UI components provide widgets and helpers to make your app not only easy, but delightful to use.

Animation & transitions

Move widgets and transition between screens

Emoji

Enable an up-to-date emoji font on older platforms

Fragment

A basic unit of composable UI

Layout

Lay out widgets using different algorithms

Palette

Pull useful information out of color palettes

AndroidX Overview

AndroidX is the open-source project that the Android team uses to develop, test, package, version and release libraries within Jetpack.

AndroidX is a major improvement to the original Android Support Library. Like the Support Library, AndroidX ships separately from the Android OS and provides backwards-compatibility across Android releases. AndroidX fully replaces the Support Library by providing feature parity and new libraries. In addition AndroidX includes the following features:

- All packages in AndroidX live in a consistent namespace starting with the string `androidx`. The Support Library packages have been mapped into corresponding `androidx.*` packages. For a full mapping of all the old classes and build artifacts to the new ones, see the Package Refactoring page.
- Unlike the Support Library, AndroidX packages are separately maintained and updated. The `androidx` packages use strict Semantic Versioning starting with version 1.0.0. You can update AndroidX libraries in your project independently.
- All new Support Library development will occur in the AndroidX library. This includes maintenance of the original Support Library artifacts and introduction of new Jetpack components.

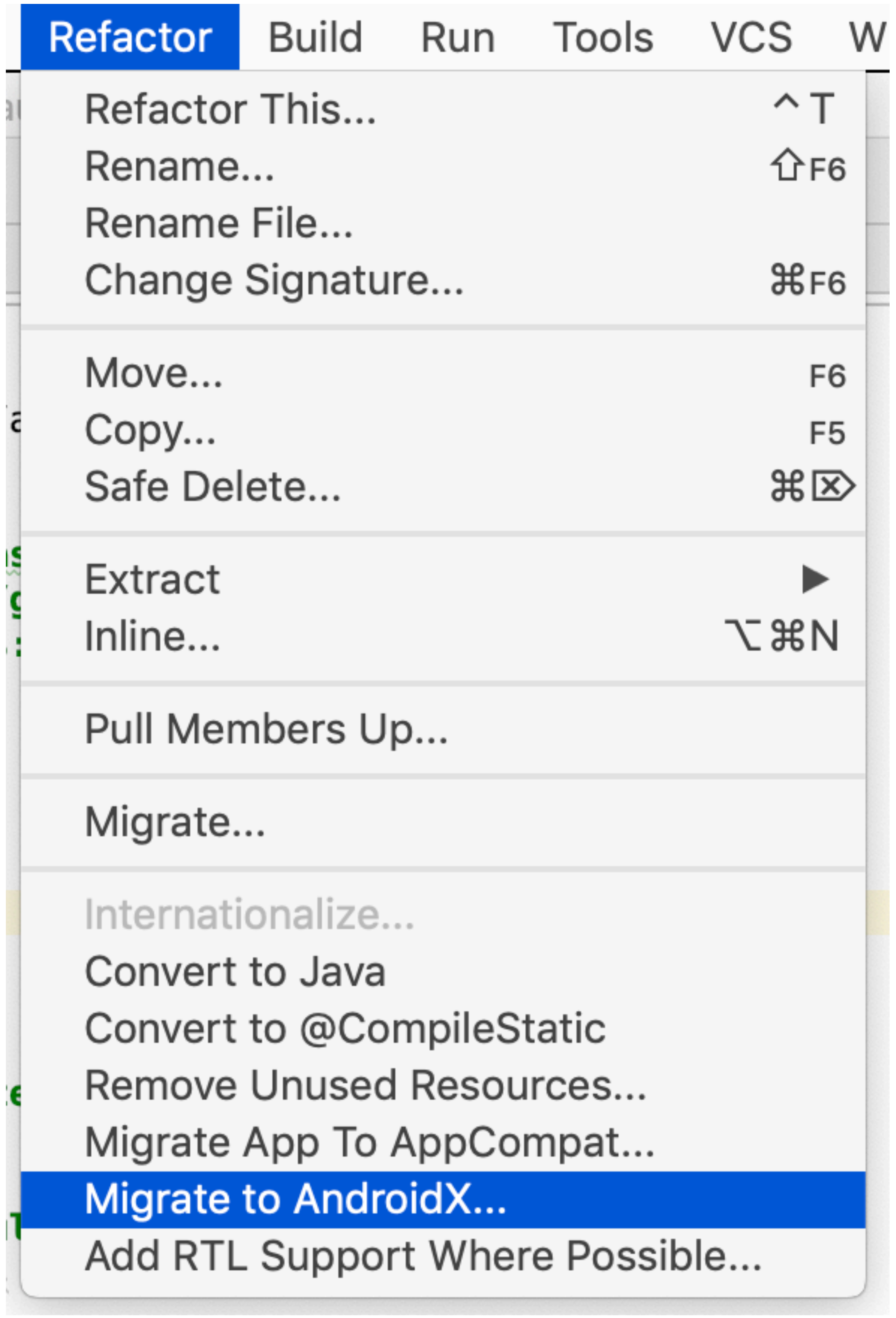
Migrate an existing project using Android Studio

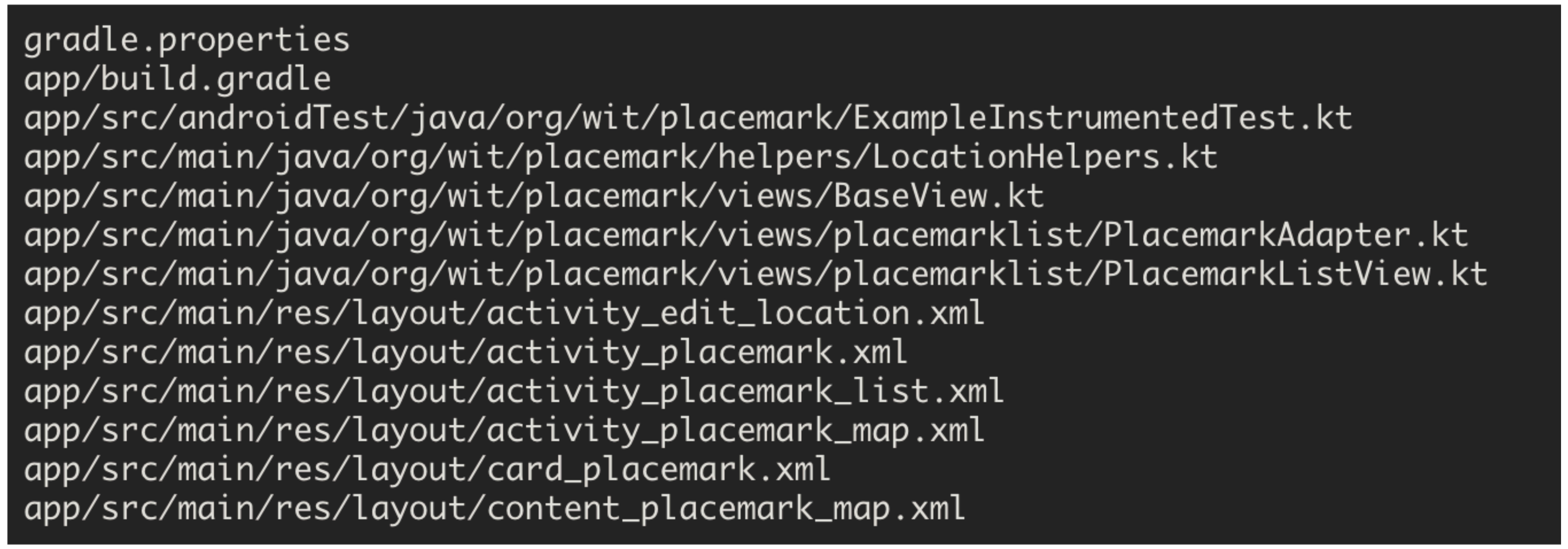
With Android Studio 3.2 and higher, you can quickly migrate an existing project to use AndroidX by selecting **Refactor > Migrate to AndroidX** from the menu bar.

If you have any Maven dependencies that have not been migrated to the AndroidX namespace, the Android Studio build system also migrates those dependencies for you when you set the following two flags to `true` in your `gradle.properties` file:

```
android.useAndroidX=true
android.enableJetifier=true
```

To migrate an existing project that does not use any third-party libraries with dependencies that need converting, you can set the `android.useAndroidX` flag to `true` and the `android.enableJetifier` flag to `false`.





```
gradle.properties
app/build.gradle
app/src/androidTest/java/org/wit/placemark/ExampleInstrumentedTest.kt
app/src/main/java/org/wit/placemark/helpers/LocationHelpers.kt
app/src/main/java/org/wit/placemark/views/BaseView.kt
app/src/main/java/org/wit/placemark/views/placemarklist/PlacemarkAdapter.kt
app/src/main/java/org/wit/placemark/views/placemarklist/PlacemarkListView.kt
app/src/main/res/layout/activity_edit_location.xml
app/src/main/res/layout/activity_placemark.xml
app/src/main/res/layout/activity_placemark_list.xml
app/src/main/res/layout/activity_placemark_map.xml
app/src/main/res/layout/card_placemark.xml
app/src/main/res/layout/content_placemark_map.xml
```

Artifact mappings

The following table lists the current mappings from old artifacts to new ones. You can also download these mappings in CSV format.

Old build artifact	AndroidX build artifact
`android.arch.core:common`	`androidx.arch.core:core-common:2.0.0-rc01`
`android.arch.core:core`	`androidx.arch.core:core:2.0.0-rc01`
`android.arch.core:core-testing`	`androidx.arch.core:core-testing:2.0.0-rc01`
`android.arch.core:runtime`	`androidx.arch.core:core-runtime:2.0.0-rc01`
`android.arch.lifecycle:common`	`androidx.lifecycle:lifecycle-common:2.0.0-rc01`
`android.arch.lifecycle:common-java8`	`androidx.lifecycle:lifecycle-common-java8:2.0.0-rc01`
`android.arch.lifecycle:compiler`	`androidx.lifecycle:lifecycle-compiler:2.0.0-rc01`
`android.arch.lifecycle:extensions`	`androidx.lifecycle:lifecycle-extensions:2.0.0-rc01`
`android.arch.lifecycle:livedata`	`androidx.lifecycle:lifecycle-livedata:2.0.0-rc01`
`android.arch.lifecycle:livedata-core`	`androidx.lifecycle:lifecycle-livedata-core:2.0.0-rc01`
`android.arch.lifecycle:reactivestreams`	`androidx.lifecycle:lifecycle-reactivestreams:2.0.0-rc01`
`android.arch.lifecycle:runtime`	`androidx.lifecycle:lifecycle-runtime:2.0.0-rc01`
`android.arch.lifecycle:viewmodel`	`androidx.lifecycle:lifecycle-viewmodel:2.0.0-rc01`
`android.arch.paging:common`	`androidx.paging:paging-common:2.0.0-rc01`
`android.arch.paging:runtime`	`androidx.paging:paging-runtime:2.0.0-rc01`
`android.arch.paging:rxjava2`	`androidx.paging:paging-rxjava2:2.0.0-rc01`
`android.arch.persistence.room:common`	`androidx.room:room-common:2.0.0-rc01`
`android.arch.persistence.room:compiler`	`androidx.room:room-compiler:2.0.0-rc01`
`android.arch.persistence.room:guava`	`androidx.room:room-guava:2.0.0-rc01`
`android.arch.persistence.room:migration`	`androidx.room:room-migration:2.0.0-rc01`
`android.arch.persistence.room:runtime`	`androidx.room:room-runtime:2.0.0-rc01`
`android.arch.persistence.room:rxjava2`	`androidx.room:room-rxjava2:2.0.0-rc01`
`android.arch.persistence.room:testing`	`androidx.room:room-testing:2.0.0-rc01`
`android.arch.persistence:db`	`androidx.sqlite:sqlite:2.0.0-rc01`

Class mappings

The following table lists the current mappings from the old namespace to the new `androidx` packages. You can also download these mappings in CSV format.

Support Library class	AndroidX class
`android.arch.core.executor.ArchTaskExecutor`	`androidx.arch.core.executor.ArchTaskExecutor`
`android.arch.core.executor.DefaultTaskExecutor`	`androidx.arch.core.executor.DefaultTaskExecuto`
`android.arch.core.executor.JunitTaskExecutorRule`	`androidx.arch.core.executor.JunitTaskExecutorR`
`android.arch.core.executor.TaskExecutor`	`androidx.arch.core.executor.TaskExecutor`
`android.arch.core.executor.TaskExecutorWithFakeMainThread`	`androidx.arch.core.executor.TaskExecutorWithFakeMainThread`
`android.arch.core.executor.testing.CountingTaskExecutorRule`	`androidx.arch.core.executor.testing.CountingTaskExecutorRule`
`android.arch.core.executor.testing.InstantTaskExecutorRule`	`androidx.arch.core.executor.testing.InstantTaskExecutorRule`
`android.arch.core.internal.FastSafeIterableMap`	`androidx.arch.core.internal.FastSafeIterableMa`
`android.arch.core.internal.SafeIterableMap`	`androidx.arch.core.internal.SafeIterableMap`
`android.arch.core.util.Function`	`androidx.arch.core.util.Function`
`android.arch.lifecycle.AndroidViewModel`	`androidx.lifecycle.AndroidViewModel`
`android.arch.lifecycle.ClassesInfoCache`	`androidx.lifecycle.ClassesInfoCache`
`android.arch.lifecycle.CompositeGeneratedAdaptersObserver`	`androidx.lifecycle.CompositeGeneratedAdaptersO`
`android.arch.lifecycle.ComputableLiveData`	`androidx.lifecycle.ComputableLiveData`
`android.arch.lifecycle.DefaultLifecycleObserver`	`androidx.lifecycle.DefaultLifecycleObserver`
`android.arch.lifecycle.Elements_extKt`	`androidx.lifecycle.Elements_extKt`
`android.arch.lifecycle.EmptyActivityLifecycleCallbacks`	`androidx.lifecycle.EmptyActivityLifecycleCallb`
`android.arch.lifecycle.ErrorMessages`	`androidx.lifecycle.ErrorMessages`
`android.arch.lifecycle.FullLifecycleObserver`	`androidx.lifecycle.FullLifecycleObserver`
`android.arch.lifecycle.FullLifecycleObserverAdapter`	`androidx.lifecycle.FullLifecycleObserverAdapte`
`android.arch.lifecycle.GeneratedAdapter`	`androidx.lifecycle.GeneratedAdapter`
`android.arch.lifecycle.GenericLifecycleObserver`	`androidx.lifecycle.GenericLifecycleObserver`
`android.arch.lifecycle.HolderFragment`	`androidx.lifecycle.HolderFragment`
`android.arch.lifecycle.Input_collectorKt`	`androidx.lifecycle.Input_collectorKt`

Android KTX | Part of Android Jetpack.

Android KTX is a set of Kotlin extensions that are included with Android Jetpack and other Android libraries. KTX extensions provide concise, idiomatic Kotlin to Jetpack, Android platform, and other APIs. To do so, these extensions leverage several Kotlin language features, including the following:

- Extension functions
- Extension properties
- Lambdas
- Named parameters
- Parameter default values
- Coroutines