Kotlin Multiplatform Conversions at Android Jetpack Scale

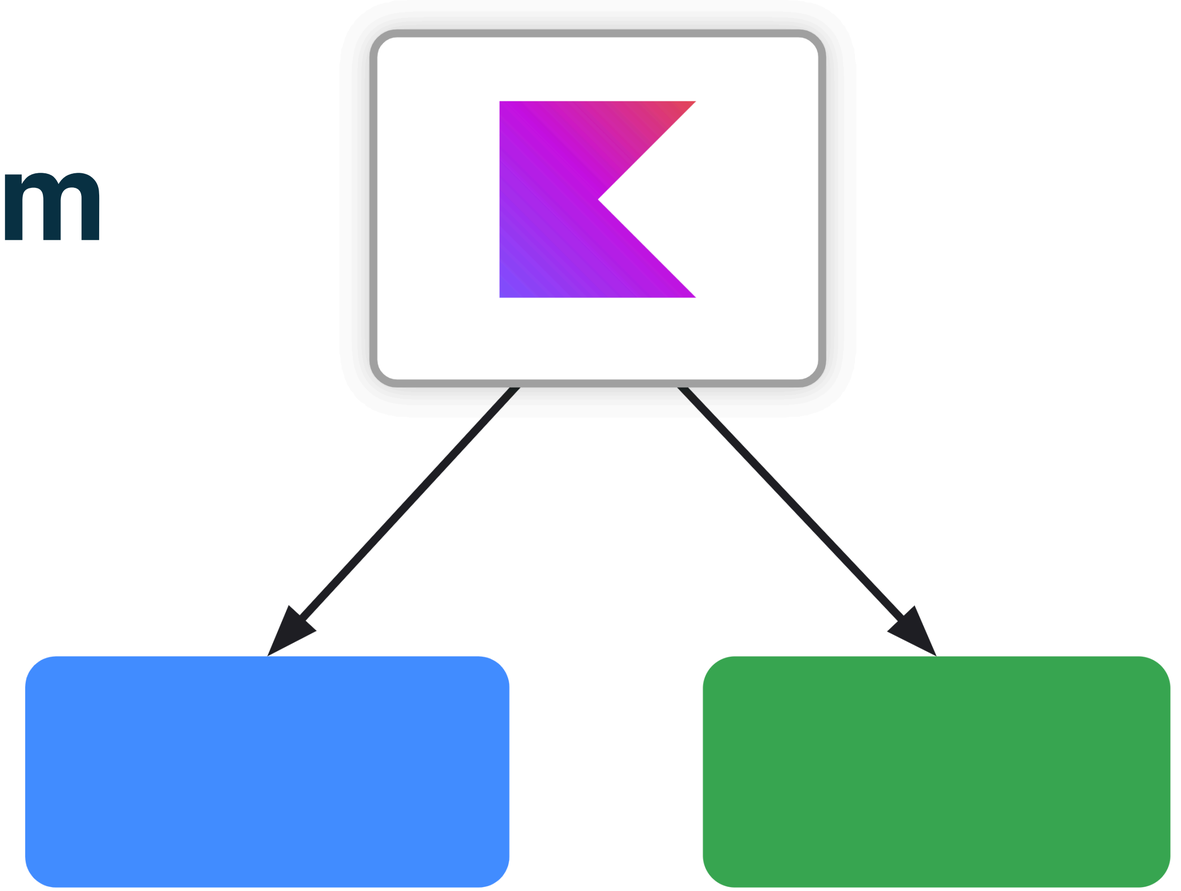
Dustin Lam
@itsdustinlam

James Ward
@_JamesWard

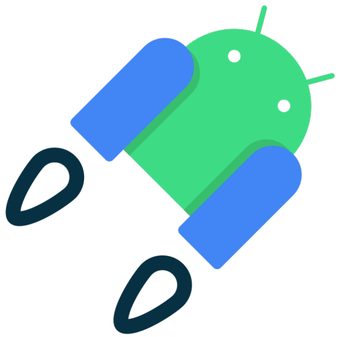
KotlinConf’23
Amsterdam

Kotlin Multiplatform

Share business logic written in Kotlin
between platforms



Google’s Contributions to Kotlin Multiplatform



Jetpack

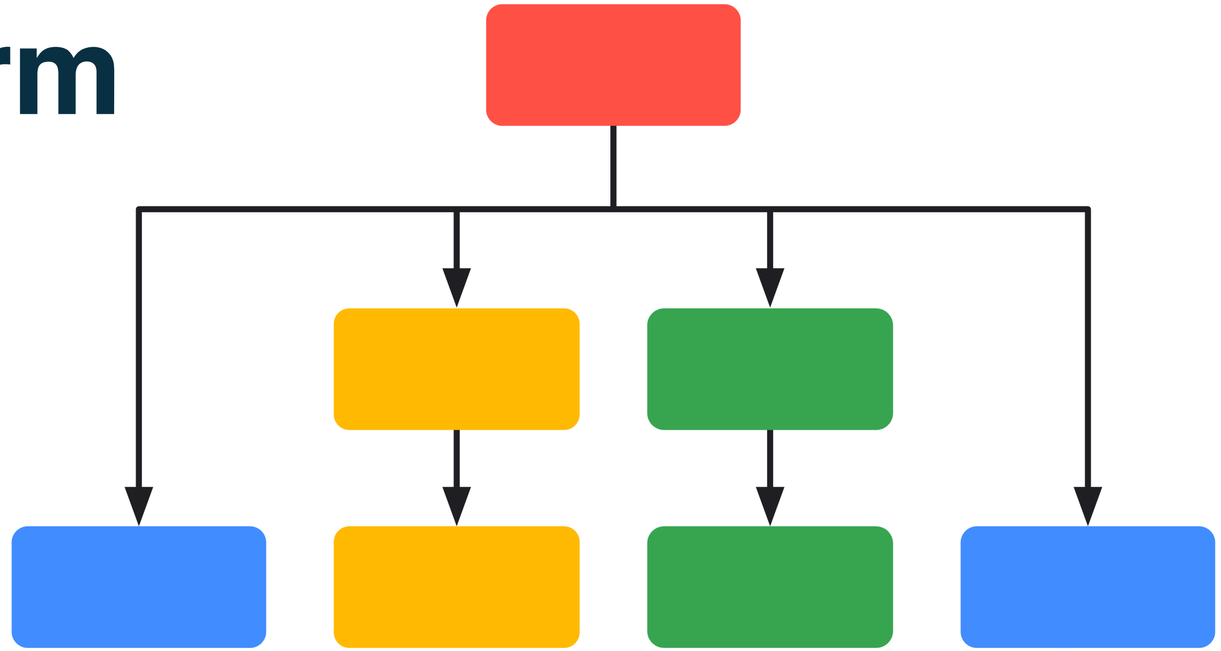
Gradle

Compiler

Kotlin Multiplatform

in Google Workspace

Experiments in code sharing with Kotlin
across Android, iOS, and web



Jetpack Alphas

goo.gle/kotlin-multiplatform

Annotations

Common utilities for richer APIs

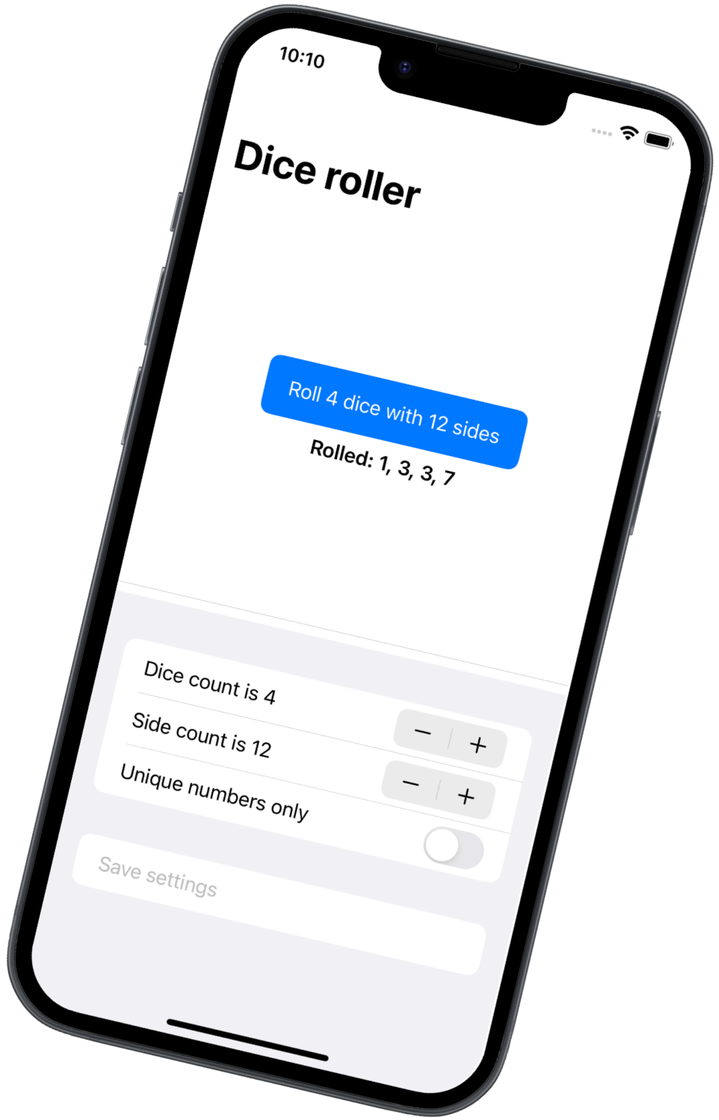
Collections

Mobile-optimized Collections

DataStore

Stream API for persistent data storage (K/V or Proto)

goo.gle/kmm-samples



Supporting our existing users

Compatibility in Android

- Compatibility is an important consideration for Android
- The ability to deploy an Android app on almost any device takes a lot of work!
- Android Support Libraries were born as a part of this effort

Compatibility in Jetpack

- Jetpack was founded on top of Android Support Libraries
- New APIs require four release stages before becoming stable
- Strict semantic versioning, with major versions often lasting years

Challenges with compatibility

- Android's stance on "Kotlin first" does not mean Kotlin only
- We care about binary + source compatibility from both Kotlin and Java
- Kotlin's Java interop story poses some very interesting problems

```
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```java
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```java
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```java
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```kotlin
abstract class ArraySet<E>() : MutableSet<E> {
    override val size: Int
        get() = ...

    override fun remove(element: E): Boolean { ... }
}
```

```java
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```kotlin
abstract class ArraySet<E>() : MutableSet<E> { // Kotlin has its own set of stdlib collections.
    override val size: Int
        get() = ...

    override fun remove(element: E): Boolean { ... }
}
```

```java
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```kotlin
abstract class ArraySet<E>() : MutableSet<E> { // Kotlin has its own set of stdlib collections.
    override val size: Int // size() replaced with a property.
        get() = ...

    override fun remove(element: E): Boolean { ... }
}
```

```java
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```kotlin
abstract class ArraySet<E>() : MutableSet<E> { // Kotlin has its own set of stdlib collections.
    override val size: Int // size() replaced with a property.
        get() = ...

    override fun remove(element: E): Boolean { ... } // The argument `element` is now typed.
}
```

```java
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }

    @Override
    public E remove(Object o) { ... }
}
```

```kotlin
abstract class ArraySet<E>() : MutableSet<E> { // Kotlin has its own set of stdlib collections.
    override val size: Int // size() replaced with a property.
        get() = ...

    override fun remove(element: E): Boolean { ... } // The argument `element` is now typed.
}
```

How does any of this even work ???

```
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size();
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size();
}

abstract class ArraySet<E>()  :  MutableSet<E>{
    override val size:  Int
        get() = ...
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size();
}

public abstract class androidx.collection.ArraySet<E> implements java.util.Set<E>,
  kotlin.jvm.internal.markers.KMutableSet {

  public androidx.collection.ArraySet();
  public int getSize();

  public final int size();
    Code:
       0: aload_0
       1: invokevirtual #20                 // Method getSize:()I
       4: ireturn
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size();
}

public abstract class androidx.collection.ArraySet<E> implements java.util.Set<E>,
  kotlin.jvm.internal.markers.KMutableSet {

  public androidx.collection.ArraySet();
  public int getSize();

  public final int size();
    Code:
       0: aload_0
       1: invokevirtual #20                 // Method getSize:()I
       4: ireturn
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size();
}

public abstract class androidx.collection.ArraySet<E> implements java.util.Set<E>,
  kotlin.jvm.internal.markers.KMutableSet {

  public androidx.collection.ArraySet();
  public int getSize();

  public final int size();
    Code:
       0: aload_0
       1: invokevirtual #20                 // Method getSize:()I
       4: ireturn
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size();
}

public abstract class androidx.collection.ArraySet<E> implements java.util.Set<E>,
  kotlin.jvm.internal.markers.KMutableSet {

  public androidx.collection.ArraySet();
  public int getSize();

  public final int size();
    Code:
       0: aload_0
       1: invokevirtual #20                 // Method getSize:()I
       4: ireturn
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size();
}

public abstract class androidx.collection.ArraySet<E> implements java.util.Set<E>,
  kotlin.jvm.internal.markers.KMutableSet {

  public androidx.collection.ArraySet();
  public int getSize();

  public final int size();
    Code:
       0: aload_0
       1: invokevirtual #20                 // Method getSize:()I
       4: ireturn
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size();
}

public abstract class androidx.collection.ArraySet<E> implements java.util.Set<E>,
  kotlin.jvm.internal.markers.KMutableSet {

  public androidx.collection.ArraySet();
  public int getSize();

  public final int size(); // final to ensure size(), getSize() have expected behavior.
    Code:
       0: aload_0
       1: invokevirtual #20                 // Method getSize:()I
       4: ireturn
}
```

```
public abstract class androidx.collection.ArraySet<E>  implements  java.util.Set<E> {
    public androidx.collection.ArraySet();
    public int size(); // Non-final, which is binary incompatible!
}

public abstract class androidx.collection.ArraySet<E> implements java.util.Set<E>,
  kotlin.jvm.internal.markers.KMutableSet {

  public androidx.collection.ArraySet();
  public int getSize();

  public final int size(); // final to ensure size(), getSize() have expected behavior.
    Code:
      0: aload_0
      1: invokevirtual #20                 // Method getSize:()I
      4: ireturn
}
```

```
// jvmMain
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }
}
```

```
// jvmMain
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }
}

// commonMain
expect abstract class ArraySet<E>() : MutableSet<E> {
    override val size: Int
}
```

```
// jvmMain
public abstract class ArraySet<E> implements Set<E> {
    @Override
    public int size() { ... }
}

// commonMain
expect abstract class ArraySet<E>() : MutableSet<E> {
    override val size: Int
}

// nonJvmMain
actual abstract class ArraySet<E> : MutableSet<E> {
    actual override val size: Int
        get() = ...
}
```

kotlin/libraries/stdlib/jvm/src/kotlin/collections/TypeAliases.kt

```
@SinceKotlin("1.1") public actual typealias ArrayList<E> = java.util.ArrayList<E>
@SinceKotlin("1.1") public actual typealias LinkedHashMap<K, V> = java.util.LinkedHashMap<K, V>
@SinceKotlin("1.1") public actual typealias HashMap<K, V> = java.util.HashMap<K, V>
@SinceKotlin("1.1") public actual typealias LinkedHashSet<E> = java.util.LinkedHashSet<E>
@SinceKotlin("1.1") public actual typealias HashSet<E> = java.util.HashSet<E>
```

What about @JvmName?

- Renames the function itself, not the generated bridges
- Not fully implemented and throws INAPPLICABLE_JVM_NAME
- Some hope in @BinarySignatureName (KEEP-302)

Validating compatibility

Existing tools

- japicmp
- binary-compatibility-validator
- Metalava

japicmp

github.com/siom79/japicmp

- Compares .class files in jar archives to determine public API changes
- Built for Java, not Kotlin
- Doesn't account for Kotlin-specific constructs

```
typealias PublicAlias = Int
```

```
typealias PublicAlias = Int

$ javap -p SampleKt.class

Compiled from "Sample.kt"
public final class androidx.collection.SampleKt {
}
```

```
typealias PublicAlias = Int

val property: Int = 0

$ javap -p SampleKt.class

Compiled from "Sample.kt"
public final class androidx.collection.SampleKt {
    private static final int property;
    public  static  final int getProperty();
    static {};
}
```

```
typealias PublicAlias = Int

val property: Int = 0

$ javap -p SampleKt.class

Compiled from "Sample.kt"
public final class androidx.collection.SampleKt {
    private static final int property;
    public  static  final int getProperty();
    static {};
}
```

```
typealias PublicAlias = Int

val property: Int = 0

fun jvmApi(argument: Int) {}
```

```
$ javap -p SampleKt.class

Compiled from "Sample.kt"
public final class androidx.collection.SampleKt {
    private static final int property;
    public  static  final int getProperty();
    public  static  final void jvmApi(int);
    static {};
}
```

```kotlin
typealias PublicAlias = Int

val property: Int = 0

fun jvmApi(argument: Int) {}
```

```
$ javap -p SampleKt.class

Compiled from "Sample.kt"
public final class androidx.collection.SampleKt {
    private static final int property;
    public  static  final int getProperty();
    public  static  final void jvmApi(int);
    static {};
}
```

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```
$ javap -p SampleKt.class

Compiled from "Sample.kt"
public final class androidx.collection.SampleKt {
    private static final int property;
    public  static  final int getProperty();
    public  static  final void jvmApi(int);
    static {};
}
```

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```
$ javap -p SampleKt.class

Compiled from "Sample.kt"
public final class androidx.collection.SampleKt {
    private static final int property;
    public  static  final int getProperty();
    public  static  final void jvmApi(int);
    static {};
}
```

binary-compatibility-validator

github.com/Kotlin/binary-compatibility-validator

- A Kotlin tool, so it should consider Kotlin constructs
- Supports multiple JVM targets
- No support for source compatibility

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```
$ ./gradlew lib:apiDump && cat lib/api/lib.api

public final class androidx/collection/SampleKt {
    public static final fun getProperty ()I
    public static final fun jvmApi (I)V
}
```

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```
$ ./gradlew lib:apiDump && cat lib/api/lib.api

public final class androidx/collection/SampleKt {
    public static final fun getProperty ()I
    public static final fun jvmApi (I)V
}
```

Metalava

android.googlesource.com/platform/tools/metalava

- Android's current solution for binary + source compatibility
- Parses UAST, so it technically has more information to work with
- Ongoing work for full support, but Kotlin Multiplatform is very new

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```
$ ./gradlew lib:updateApi && cat lib/api/current.txt

public final class SampleKt {
    method public static int getProperty();
    method public static void jvmApi(int argument);
    property public static final int property;
}
```

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```
$ ./gradlew lib:updateApi && cat lib/api/current.txt
```

```
public final class SampleKt {
    method public static int getProperty();
    method public static void jvmApi(int argument);
    property public static final int property;
}
```

```kotlin
typealias PublicAlias = Int

val property: Int = 0

@JvmName("jvmApi")
fun kotlinApi(argument: Int) {}
```

```
$ ./gradlew lib:updateApi && cat lib/api/current.txt

public final class SampleKt {
    method public static int getProperty();
    method public static void jvmApi(int argument);
    property public static final int property;
}
```

Tracking source compatibility

- No perfect solution currently exists
- Jetpack solved this manually using integration tests
- We believe this is automatable with a focused tool, but it doesn't exist yet

```
fun arraySetSourceCompatibility(): Boolean {
    var arraySet: ArraySet<Int> = ArraySet()
    arraySet = ArraySet<Int>(5)
    arraySet = ArraySet(arraySet)
    arraySet = ArraySet(setOf())
    arraySet = ArraySet(arrayOf())
    ...
}
```

```
fun arraySetSourceCompatibility(): Boolean {
    var arraySet: ArraySet<Int> = ArraySet()
    arraySet = ArraySet<Int>(5)
    arraySet = ArraySet(arraySet)
    arraySet = ArraySet(setOf())
    arraySet = ArraySet(arrayOf())
    ...
}

public static boolean arraySetSourceCompatibility() {
    ArraySet<Integer> arraySet = new ArraySet<>();
    arraySet = new ArraySet<>(5);
    arraySet = new ArraySet<>(arraySet);
    arraySet = new ArraySet<>(new HashSet<>());
    arraySet = new ArraySet<>(new Integer[5]);
    ...
}
```

JVM Dependencies

DataStore

- Storage solution for key-value pairs and protobufs
- Leaf library lowers the overall risk of experimentation
- Offers a file storage API which means a dependency on java.io

Package kotlin.io

IO API for working with files and streams.

Types

FileTreeWalk

This class is intended to implement different file traversal methods. It allows to iterate through all files inside a given directory.

```
class FileTreeWalk : Sequence<File>
```

FileWalkDirection

An enumeration to describe possible walk directions. There are two of them: beginning from parents, ending with children, and beginning from children

Reads a line of input from the standard input stream and returns it, or return `null` if EOF has already been reached when readlnOrNull is called.

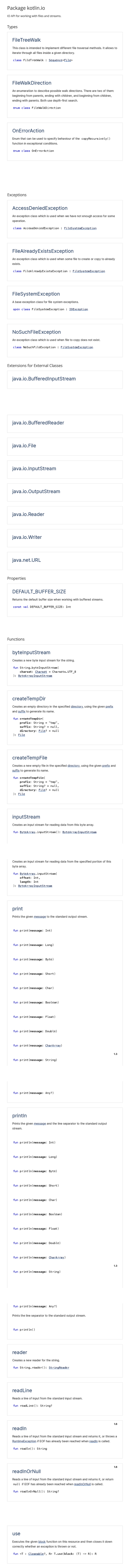
```
fun readlnOrNull(): String?
```

use

Executes the given block function on this resource and then closes it down correctly whether an exception is thrown or not.

```
fun <T : Closeable?, R> T.use(block: (T) -> R): R
```

kotlin.io



Package kotlin.io

IO API for working with files and streams.

Types

FileTreeWalk

This class is intended to implement different file traversal methods. It allows to iterate through all files inside a given directory.

```
class FileTreeWalk : Sequence<File>
```

FileWalkDirection

An enumeration to describe possible walk directions. There are two of them: beginning from parents, ending with children, and beginning from children, ending with parents. Both use depth-first search.

```
enum class FileWalkDirection
```

OnErrorAction

Enum that can be used to specify behaviour of the `copyRecursively()` function in exceptional conditions.

```
enum class OnErrorAction
```

Exceptions

AccessDeniedException

An exception class which is used when we have not enough access for some operation.

```
class AccessDeniedException : FileSystemException
```

FileAlreadyExistsException

An exception class which is used when some file to create or copy to already exists.

```
class FileAlreadyExistsException : FileSystemException
```

FileSystemException

A base exception class for file system exceptions.

```
open class FileSystemException : IOException
```

NoSuchFileException

An exception class which is used when file to copy does not exist.

```
class NoSuchFileException : FileSystemException
```

Extensions for External Classes

java.io.BufferedInputStream

java.io.BufferedReader

java.io.File

java.io.InputStream

java.io.OutputStream

java.io.Reader

java.io.Writer

java.net.URL

Properties

DEFAULT_BUFFER_SIZE

Returns the default buffer size when working with buffered streams.

```
const val DEFAULT_BUFFER_SIZE: Int
```

Functions

byteInputStream

Creates a new byte input stream for the string.

```
fun String.byteInputStream(
    charset: Charset = Charsets.UTF_8
): ByteArrayInputStream
```

createTempDir

Creates an empty directory in the specified directory, using the given prefix and suffix to generate its name.

```
fun createTempDir(
    prefix: String = "tmp",
    suffix: String? = null,
    directory: File? = null
): File
```

createTempFile

Creates a new empty file in the specified directory, using the given prefix and suffix to generate its name.

```
fun createTempFile(
    prefix: String = "tmp",
    suffix: String? = null,
    directory: File? = null
): File
```

inputStream

Creates an input stream for reading data from this byte array.

```
fun ByteArray.inputStream(): ByteArrayInputStream
```

Creates an input stream for reading data from the specified portion of this byte array.

```
fun ByteArray.inputStream(
    offset: Int,
    length: Int
): ByteArrayInputStream
```

print

Prints the given message to the standard output stream.

```
fun print(message: Int)
fun print(message: Long)
fun print(message: Byte)
fun print(message: Short)
fun print(message: Char)
fun print(message: Boolean)
fun print(message: Float)
fun print(message: Double)
fun print(message: CharArray)
```

1.3

```
fun print(message: String)
fun print(message: Any?)
```

println

Prints the given message and the line separator to the standard output stream.

```
fun println(message: Int)
fun println(message: Long)
fun println(message: Byte)
fun println(message: Short)
fun println(message: Char)
fun println(message: Boolean)
fun println(message: Float)
fun println(message: Double)
fun println(message: CharArray)
```

1.3

```
fun println(message: String)
fun println(message: Any?)
```

Prints the line separator to the standard output stream.

```
fun println()
```

reader

Creates a new reader for the string.

```
fun String.reader(): StringReader
```

readLine

Reads a line of input from the standard input stream.

```
fun readLine(): String?
```

1.6

readln

Reads a line of input from the standard input stream and returns it, or throws a RuntimeException if EOF has already been reached when readln is called.

```
fun readln(): String
```

1.6

readlnOrNull

Reads a line of input from the standard input stream and returns it, or return `null` if EOF has already been reached when readlnOrNull is called.

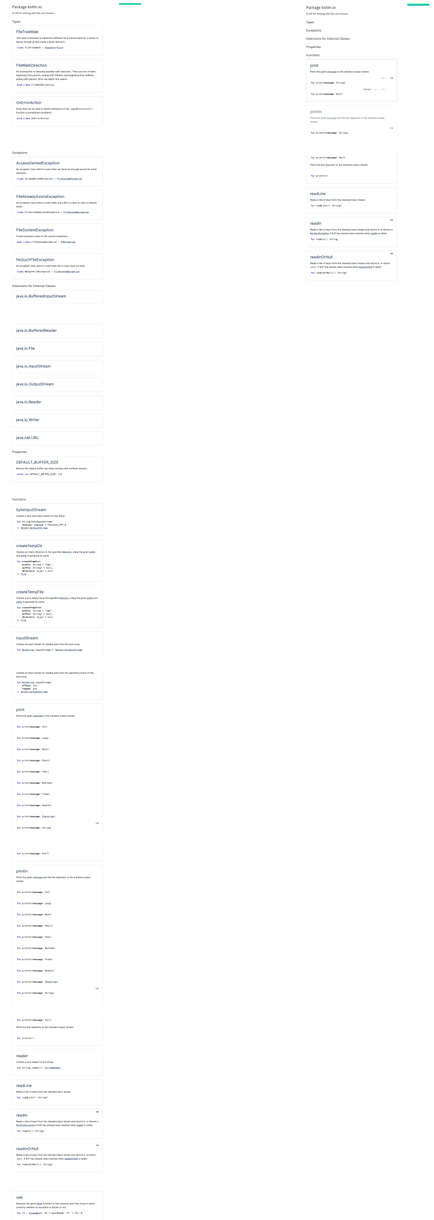
```
fun readlnOrNull(): String?
```

use

Executes the given block function on this resource and then closes it down correctly whether an exception is thrown or not.

```
fun <T : Closeable?, R> T.use(block: (T) -> R): R
```

kotlin.io


Package kotlin.io
Types
FileTreeWalk
FileWalkDirection
OnErrorAction
Exceptions
AccessDeniedException
FileAlreadyExistsException
FileSystemException
NoSuchFileException
Extensions for External Classes
java.io.BufferedInputStream
java.io.BufferedReader
java.io.File
java.io.InputStream
java.io.OutputStream
java.io.Reader
java.io.Writer
java.net.URL
Properties
DEFAULT_BUFFER_SIZE
Functions
byteInputStream
createTempDir
createTempFile
inputStream
print
println
reader
readLine
readln
readlnOrNull
use
Package kotlin.io
Types
Exceptions
Extensions for External Classes
Properties
Functions
print
println
readLine
readln
readlnOrNull

kotlin.io

- Most of the API surface is JVM only
- Need to support File I/O
- Multiplatform API from stdlib is insufficient

Okio

square.github.io/okio

- Popular I/O library on top of java.io by Block
- Supports for Kotlin Multiplatform
- Cannot force the dependency on all of our users

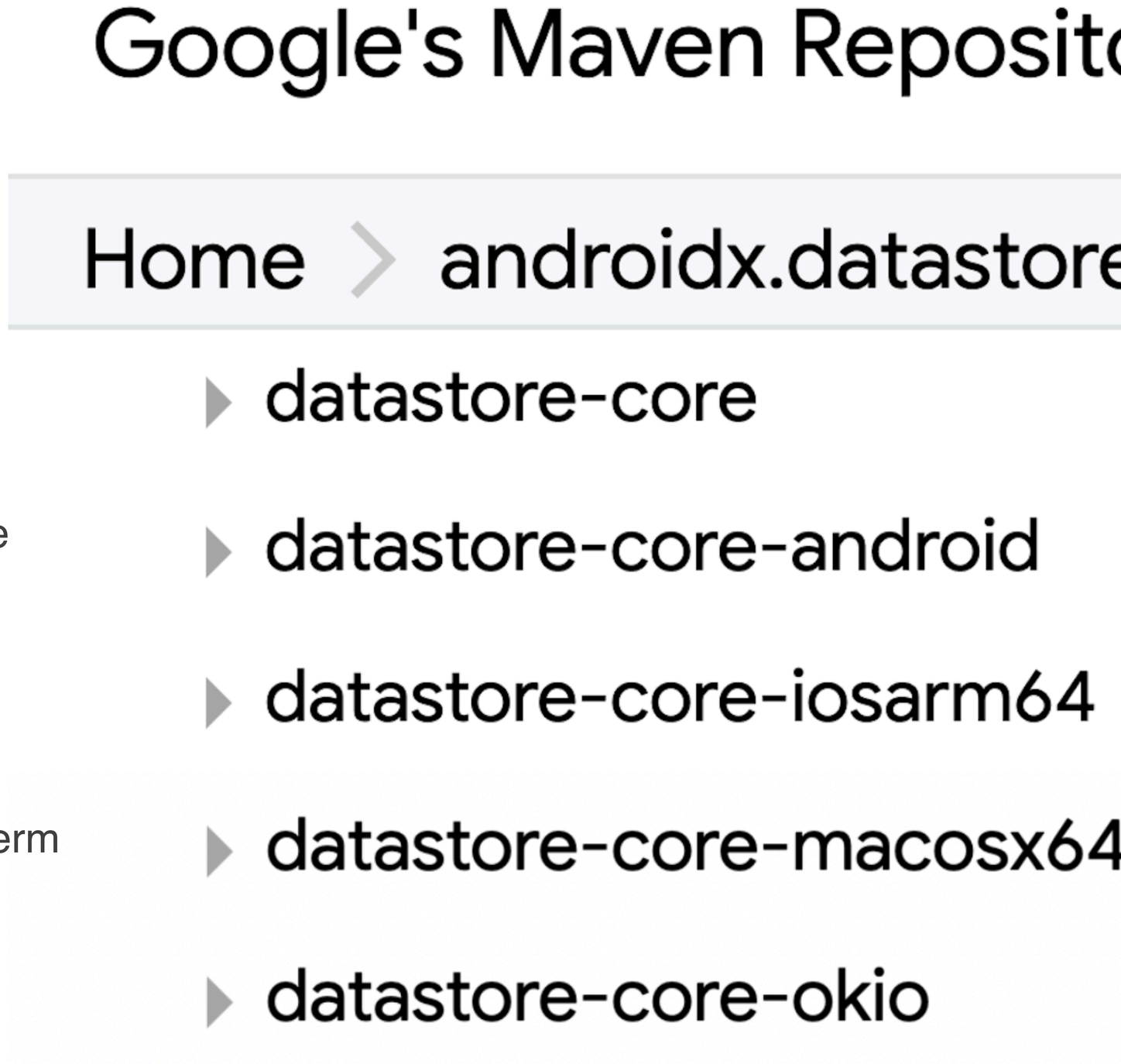
Google's Maven Repository

Home > androidx.datastore

- datastore-core
- datastore-core-android
- datastore-core-iosarm64
- datastore-core-macosx64
- datastore-core-okio

Solution

- Abstraction allowing a pluggable implementation
- Optional artifact providing Okio
- Kickstart kotlinx.io for the long term



```
interface ReadScope<T> : Closeable {
  suspend fun readData(): T
}
```

```
interface ReadScope<T> : Closeable {
    suspend fun readData(): T
}

interface WriteScope<T> : ReadScope<T> {
    suspend fun writeData(value: T)
}
```

```
internal open class OkioReadScope<T>(...) : ReadScope<T> {
    override suspend fun readData(): T { ... }

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

```
internal open class OkioReadScope<T>(...) : ReadScope<T> {
    override suspend fun readData(): T { ... }

    override fun close() { ... }
}

internal class OkioWriteScope<T>(...) : OkioReadScope<T>(...), WriteScope<T> {
    override suspend fun writeData(value: T) { ... }
}
```

Testing

JVM test conversions

- Two main parts to consider:
- Test framework - configuration of test execution
- Assertion library - API for writing tests

Requirements for conversions

- Achievable, practical amount of work
- Independently verifiable
- Solution must also be suitable for non-Kotlin Multiplatform projects

State of testing in Jetpack

- 200+ libraries
- 45,000+ tests
- 50,000+ usages of Google Truth

```
@Test
fun sampleTest() {
    assertThat(value).isEqualTo(expectedValue)
    assertThat(collection).containsExactly(0, 1, 2)
    assertThat(flag).isTrue()
}
```

```
@Test
fun sampleTest() {
    assertThat(value).isEqualTo(expectedValue)
    assertThat(collection).containsExactly(0, 1, 2)
    assertThat(flag).isTrue()
}
```

```
@Test
fun sampleTest() {
    assertThat(value).isEqualTo(expectedValue)
    assertThat(collection).containsExactly(0, 1, 2)
    assertThat(flag).isTrue()
}
```

```
@Test
fun sampleTest() {
    assertThat(value).isEqualTo(expectedValue)
    assertThat(collection).containsExactly(0, 1, 2)
    assertThat(flag).isTrue()
}
```

Kotlin Multiplatform "Kruth"

- Internal conversion of Google Truth
- Fully behaviorally compatible
- Collaborating with Truth on a longer term solution

M room/room-common/build.gradle

```
25  25 dependencies {
26  26     api("androidx.annotation:annotation:1.3.0")
27  27     api(libs.kotlinStdlibJdk8)
28  28     testImplementation(libs.junit)
29  29     testImplementation(libs.mockitoCore4)
30  30     testImplementation(libs.guava)
31      -  testImplementation(libs.truth)
    31 +  testImplementation(project(":internal-testutils-kmp"))
32  32 }
```

M room/room-common/src/test/java/androidx/room/AmbiguousColumnResolverTest.kt

```
File  File
17  17 package androidx.room
18  18
19      - import com.google.common.truth.Truth.assertThat
    19 + import androidx.kruth.assertThat
20  20 import java.util.Locale
21  21 import org.junit.Test
22  22 import org.junit.Ignore
```

"Kruth"

- Clearly definable amount of work
- Allows independent test conversions
- No difference to non-multiplatform consumers

M room/room-common/build.gradle

```
25  25  dependencies {
26  26      api("androidx.annotatio
27  27      api(libs.kotlinStdlibJ
28  28      testImplementation(libs
29  29      testImplementation(libs
30  30      testImplementation(libs
31      -   testImplementation(libs
    31  +   testImplementation(proj
32  32  }
```

M room/room-common/src/test/java/andr

```
File  File
17  17  package androidx.room
18  18
19      -   import com.google.common.tr
    19  +   import androidx.kruth.asser
20  20  import java.util.Locale
21  21  import org.junit.Test
22  22  import org.junit.Ignore
```

Idiomatic Kotlin in "Kruth"

- Visibility through actual value for Kotlin extensions
- Generic types propagated correctly to public API where possible
- Contracts for null-marked types that used to compile, but immediately throw

Performance

Performance is important

- Jetpack's offerings are widely used in Android's ecosystem
- Kotlin Multiplatform is awesome, but first priority are our existing users
- Critical for Jetpack to have performance testing in CI

Benchmarking in Jetpack

androidx-perf.skia.org

- Measuring performance using Jetpack Benchmark
- Run on real devices with automatic alerts + manual triage
- Large number of internal clients that we can leverage to vet our work

Jetpack Benchmark

- Built for Android with platform integrations
- Internal extensions to support iOS
- Check out Rahul's talk in Berlage Zaal @4:15pm

What about kotlinx.benchmark?

github.com/Kotlin/kotlinx-benchmark

- Multiplatform API that delegates to well established frameworks
- Delegates to JMH for JVM
- Doesn't support Jetpack Benchmark, but it could...

A final word

Smoothing Future Evolutions

- Intent on doing this in a way that is faithful to our existing clients
- Building for the long-term; this isn't about just making it work
- Not just about us - working with JetBrains to make the technology better

We could use your help!

- Needs: Metalava Kotlin Support, Source Compat Tooling, etc
- github.com/androidx/androidx
- Questions?

Thanks, and don’t forget to vote

Dustin Lam
@itsdustinlam

James Ward
@_JamesWard

KotlinConf’23
Amsterdam

Kotlin Multiplatform Conversions at Android Jetpack Scale

Dustin Lam
@itsdustinlam

James Ward
@_JamesWard

KotlinConf’23
Amsterdam