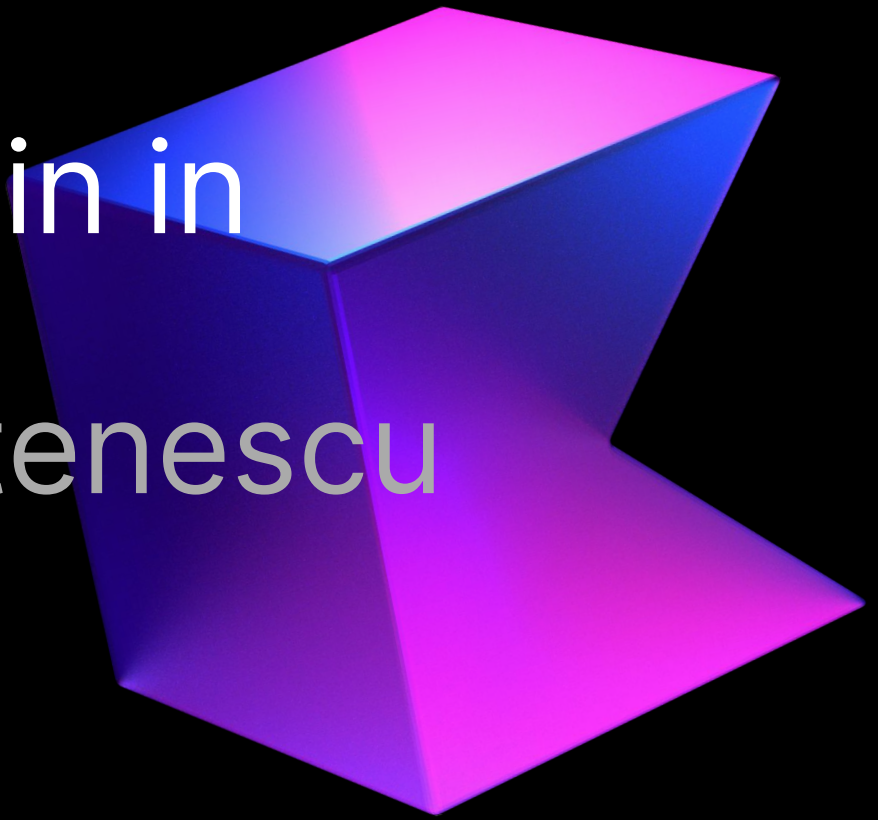
Kotlin 1.4 Online Event

# State of Kotlin in Android

Florina Muntenescu



@FMuntenescu

October, 2020

# Why Android ❤️ Kotlin

Expressiveness

# Why Android ❤️ Kotlin

Expressiveness

Safety

# Kotlin on Android in 2020

**20%**

**Less likely to crash**

*Source: Google Internal Data, July 2020*

# Why Android ❤️ Kotlin

Expressiveness

Safety

Interoperability

# Why Android ❤️ Kotlin

Expressiveness

Safety

Interoperability

Structured concurrency

# Kotlin on Android in 2020

**50%**

**More likely to be very satisfied**

*Source: Google Internal Data, May 2020*

# Kotlin on Android in 2020



**60%**

**Pro Android developers use Kotlin**

*Source: Google Internal Data, May 2020*

# Kotlin on Android in 2020

**70%+**

**Top 1k apps contain Kotlin code**

*Source: Google Internal Data, May 2020*

# Google’s contributions to Kotlin

Co-leading Kotlin Foundation

Contributing to the Kotlin compiler

Developing Kotlin related tooling and libraries

# Google’s contributions to teaching Kotlin

- Docs - d.android.com/kotlin
- Samples - github.com/android
- Codelabs - goo.gle/kotlin-codelabs
- Articles - goo.gle/kotlin-posts
- Videos - goo.gle/kotlin-videos

# Google’s contributions to teaching Kotlin

- Docs - d.android.com/kotlin
- Samples - github.com/android
- Codelabs - goo.gle/kotlin-codelabs
- Articles - goo.gle/kotlin-posts
- Videos - goo.gle/kotlin-videos
- Faculty training

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

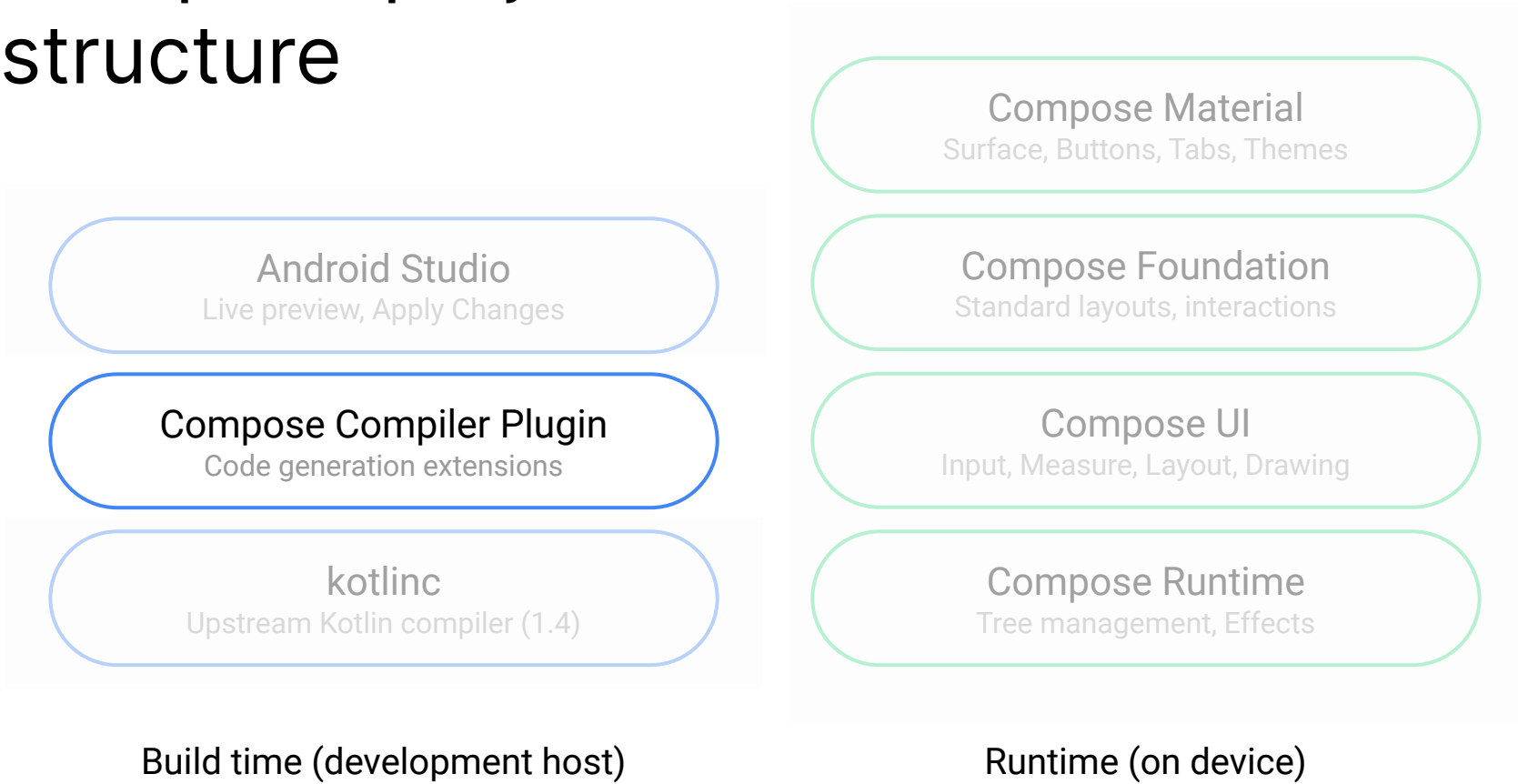
# Since last year's Google I/O...

| Libraries | Tools | Learning |
| --- | --- | --- |
|  Kotlin-first libraries |  Kotlin live templates |  Vocabulary series |
|  New KTX releases |  New lint checks |  Samples and docs |
|  Nullness annotations |  R8 Optimizations |  Kotlin/Everywhere |

# Jetpack Compose

- Modern, declarative UI toolkit
- Built on Kotlin
- Unbundled

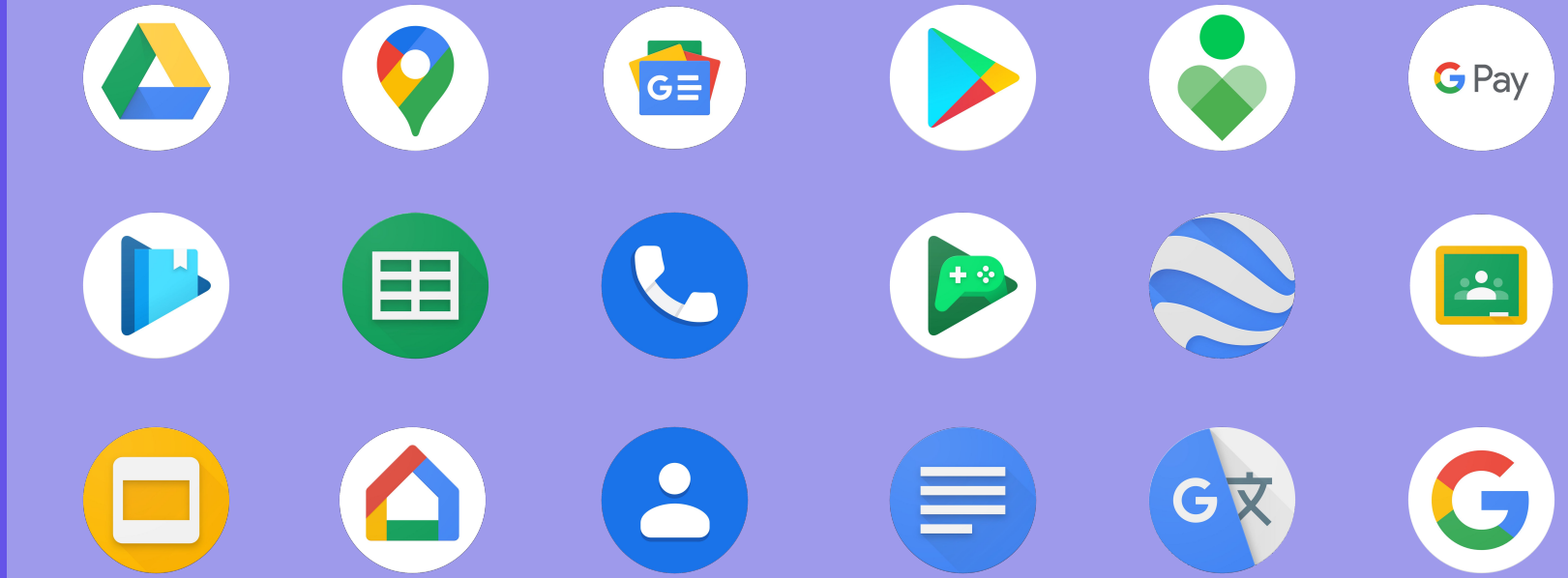
# Compose project structure



# Jetpack Compose API

```
Button(
    onClick = { /* do an action */ },
    modifier = Modifier.padding(vertical = 20.dp) // default: Modifier
) {
    Text(text = "Click me")
}
```

# Kotlin at Google



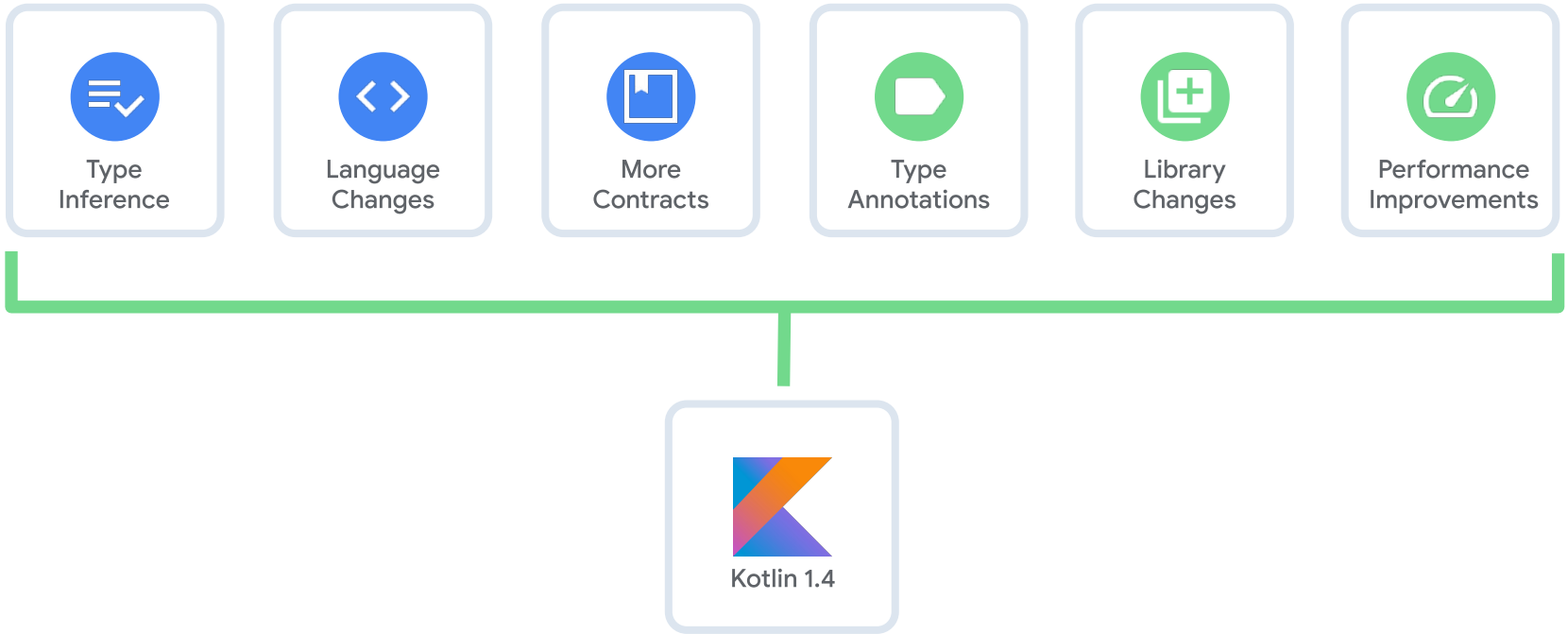

*Source: Google Internal Data, May 2020*

# Kotlin at Google

- > 2M lines of Kotlin code
- Used on server-side
- Kotlin in OSS projects:
  - gRPC
  - ProtoBuf

    (coming soon)

# Kotlin 1.4 preview



# Kotlin Tooling Performance

**“Faster Please!”**

# Kotlin Tooling Performance

- Incremental Annotation Processors

# Kotlin Tooling Performance

- Incremental Annotation Processors
- IDE Performance

# Kotlin Tooling Performance

- Incremental Annotation Processors
- IDE Performance
- Gradle improvements

# Kotlin for library authors

- Explicit API mode in Kotlin 1.4
- Kotlin Symbol Processing
- R8 Kotlin Metadata support

# Explicit API mode

- Be intentional about your public API
- Explicitly declare:
  - Visibility modifiers
  - Type

# Explicit API mode

**build.gradle**

```
kotlin {
    // for strict mode
    explicitApi()
    // or
    explicitApi = 'strict'

    // for warning mode
    explicitApiWarning()
    // or
    explicitApi = 'warning'
}
```

# Kotlin for library authors

- Explicit API mode in Kotlin 1.4
- **Kotlin Symbol Processing**
- R8 Kotlin Metadata support

# Kotlin Symbol Processing

## Developer Preview

- Alternative to KAPT
- Incremental resolution
- 30-40% faster
- Multiplatform ready
- Kotlin constructs
- Rich set of APIs

**We need your feedback!**

**goo.gle/ksp**

# R8

- Support for

  **Kotlin metadata rewriting**
- Android Gradle Plugin

  4.1.0-beta03

# AsyncTask

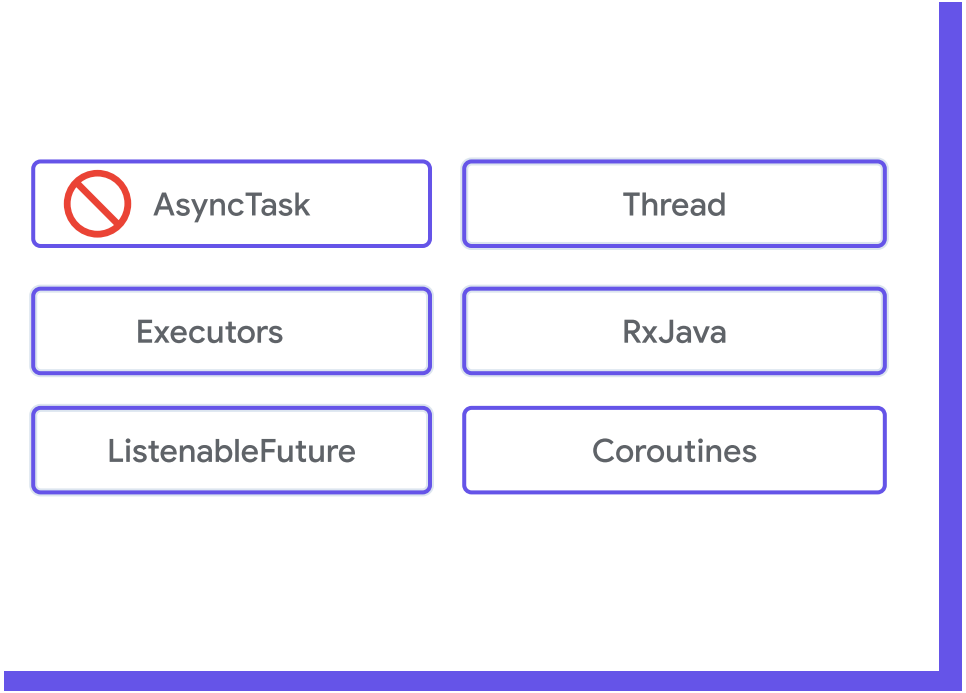
## Deprecated in Android 11

**This class was deprecated in API level R (Android 11)**

Use the standard **`java.util.concurrent`** or Kotlin concurrency utilities instead.

# Async programming

## What should we use?



# Coroutines

## Brief overview

Coroutines are the Kotlin way
to do **asynchronous programming**.

Compiler support is stable since
Kotlin 1.3,
together with a robust
*kotlinx.coroutines* library.

# Coroutines

Professional developers who use **Coroutines** have reported seeing **increased productivity**

*Source: Google Internal Data, May 2020*

# Coroutines

```
viewModelScope.launch {
    val status = withContext(Dispatchers.IO) {
      val office = officeDetailsService.findOffice("US-BVE")
        db.officeDetails().insert(office)
        office.status
    }
    val isOpenText = if (status == OfficeStatus.OPEN) {
      "open :)" } else { "closed :(" }
    officeStatusLiveData.value = "Your office is $isOpenText"
}
```

# Coroutines

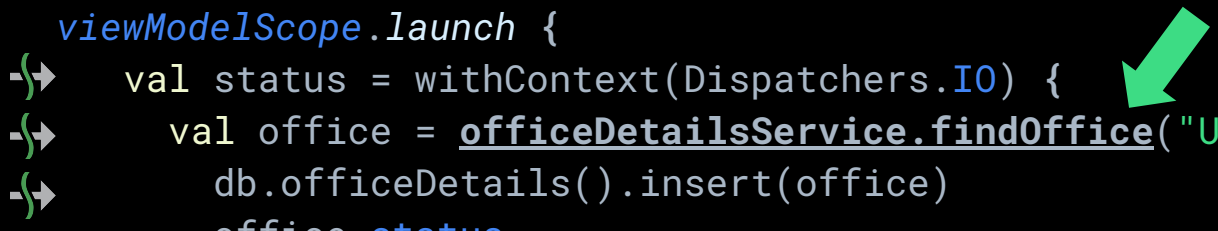
```
viewModelScope.launch {
    val status = withContext(Dispatchers.IO) {
      val office = officeDetailsService.findOffice("US-BVE")
        db.officeDetails().insert(office)
        office.status
    }
    val isOpenText = if (status == OfficeStatus.OPEN) {
      "open :)" } else { "closed :(" }
    officeStatusLiveData.value = "Your office is $isOpenText"
}
```

# Coroutines



```
viewModelScope.launch {
    val status = withContext(Dispatchers.IO) {
      val office = officeDetailsService.findOffice("US-BVE")
        db.officeDetails().insert(office)
        office.status
    }
    val isOpenText = if (status == OfficeStatus.OPEN) {
      "open :)" } else { "closed :(" }
    officeStatusLiveData.value = "Your office is $isOpenText"
}
```

# Coroutines



```
viewModelScope.launch {
    val status = withContext(Dispatchers.IO) {
      val office = officeDetailsService.findOffice("US-BVE")
        db.officeDetails().insert(office)
        office.status
    }
    val isOpenText = if (status == OfficeStatus.OPEN) {
      "open :)" } else { "closed :(" }
    officeStatusLiveData.value = "Your office is $isOpenText"
}
```

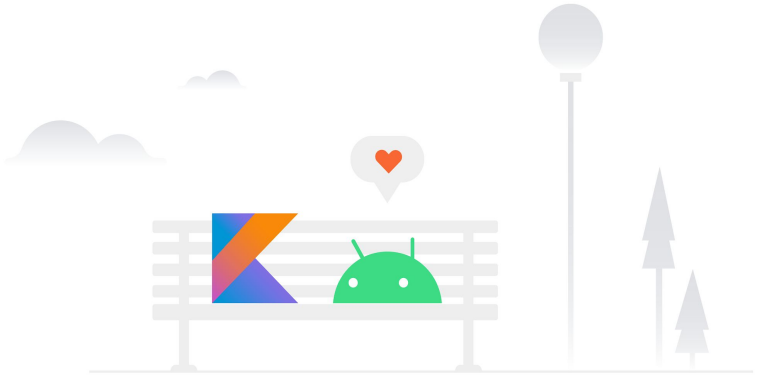
# Coroutines

```
viewModelScope.launch {
    val status = withContext(Dispatchers.IO) {
      val office = officeDetailsService.findOffice("US-BVE")
        db.officeDetails().insert(office)
        office.status
    }
    val isOpenText = if (status == OfficeStatus.OPEN) {
      "open :)" } else { "closed :(" }
    officeStatusLiveData.value = "Your office is $isOpenText"
}
```

# Coroutines

```
viewModelScope.launch {
    val status = withContext(Dispatchers.IO) {
      val office = officeDetailsService.findOffice("US-BVE")
        db.officeDetails().insert(office)
        office.status
    }
    val isOpenText = if (status == OfficeStatus.OPEN) {
      "open :)" } else { "closed :(" }
    officeStatusLiveData.value = "Your office is $isOpenText"
}
```

Kotlin coroutines are the recommended solution for async code



# Android 💚 coroutines

- Structured concurrency
- Non-blocking, sequential code
- Cancellation propagation
- Natural exception handling

# Coroutines & Jetpack

# Room

## Database queries

```
@Dao
interface UsersDao {

    @Insert
    suspend fun insertUsers(vararg users: User)

    @Update
    suspend fun updateUsers(vararg users: User)

    @Delete
    suspend fun deleteUsers(vararg users: User)

    @Query("SELECT * FROM users")
    suspend fun getUsers(): List<User>

}
```

# Room

## Database queries

```
@Dao
interface UsersDao {

    @Query("SELECT * FROM users")
    suspend fun getUsers(): List<User>

    @Query("SELECT * FROM users")
    fun getUsers(): Flow<List<User>>

}
```

# WorkManager

## CoroutineWorker

```
class PeriodicSyncWorker(
  appContext: Context,
  workerParams: WorkerParameters
) : CoroutineWorker(appContext, workerParams) {

    override suspend fun doWork(): Result {
        val data = readFromDb()
        val serverData = uploadToServer(data)
        writeToDb(serverData)

        return Result.success()
    }
}
```

# Paging

## PagingSource

```
class UsersPagingSource(val backend: UsersService)
: PagingSource<Int, User>() {

  override suspend fun load(
    params: LoadParams<Int>
  ): LoadResult<Int, User> {

      ...
      val response = backend.getUsers(page)
      return ...
  }
}
```

# Paging

## Pager

```
val users: Flow<PagingData<User>> = Pager(
    config = PagingConfig(pageSize = 10)
) {
  UsersPagingSource(usersService)
}.flow
```

# DataStore

- Data storage solution
- SharedPreferences replacement
- **Proto DataStore** - typed objects
- **Preferences DataStore** - key-value pairs

# DataStore

## Preferences DataStore and Proto DataStore

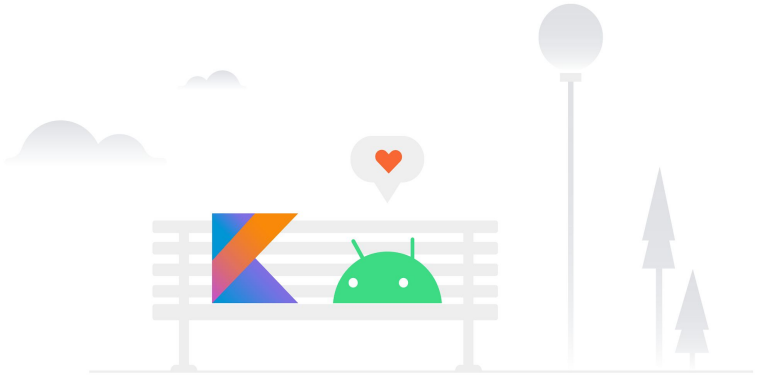
```
// Using Proto DataStore

val myCounterFlow: Flow<Int> = settingsDataStore.data
    .map { settings ->
        settings.myCounter
    }
```

# Jetpack Compose

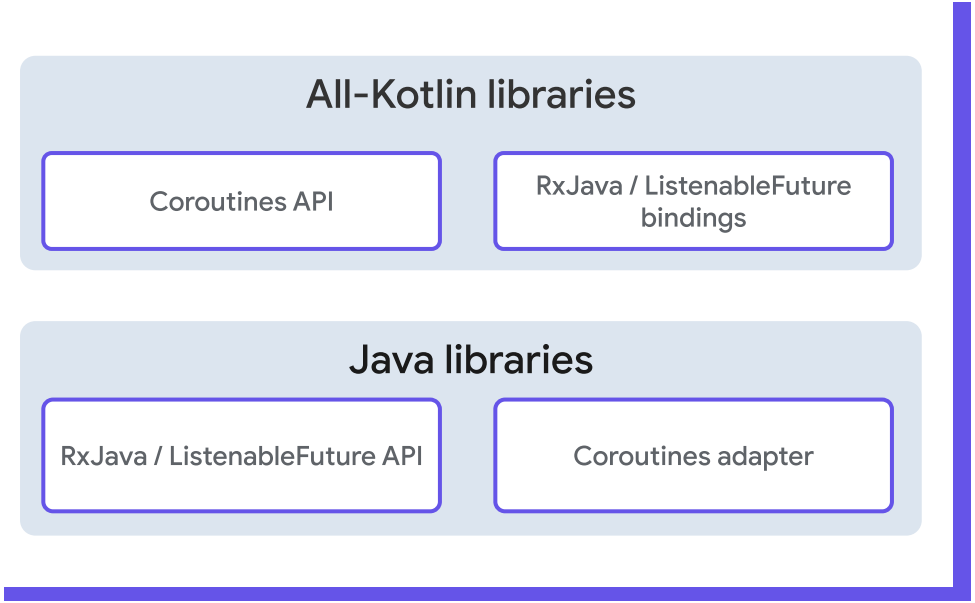
```
@Composable
fun Search(
// ...
) {
    launchInComposition(state.query.text) {
        state.searching = true
        state.searchResults = SearchRepo.search(state.query.text)
        state.searching = false
    }
}
```

Kotlin coroutines are the recommended solution for async code



# Coroutines interop

## For Java ↔ Kotlin users

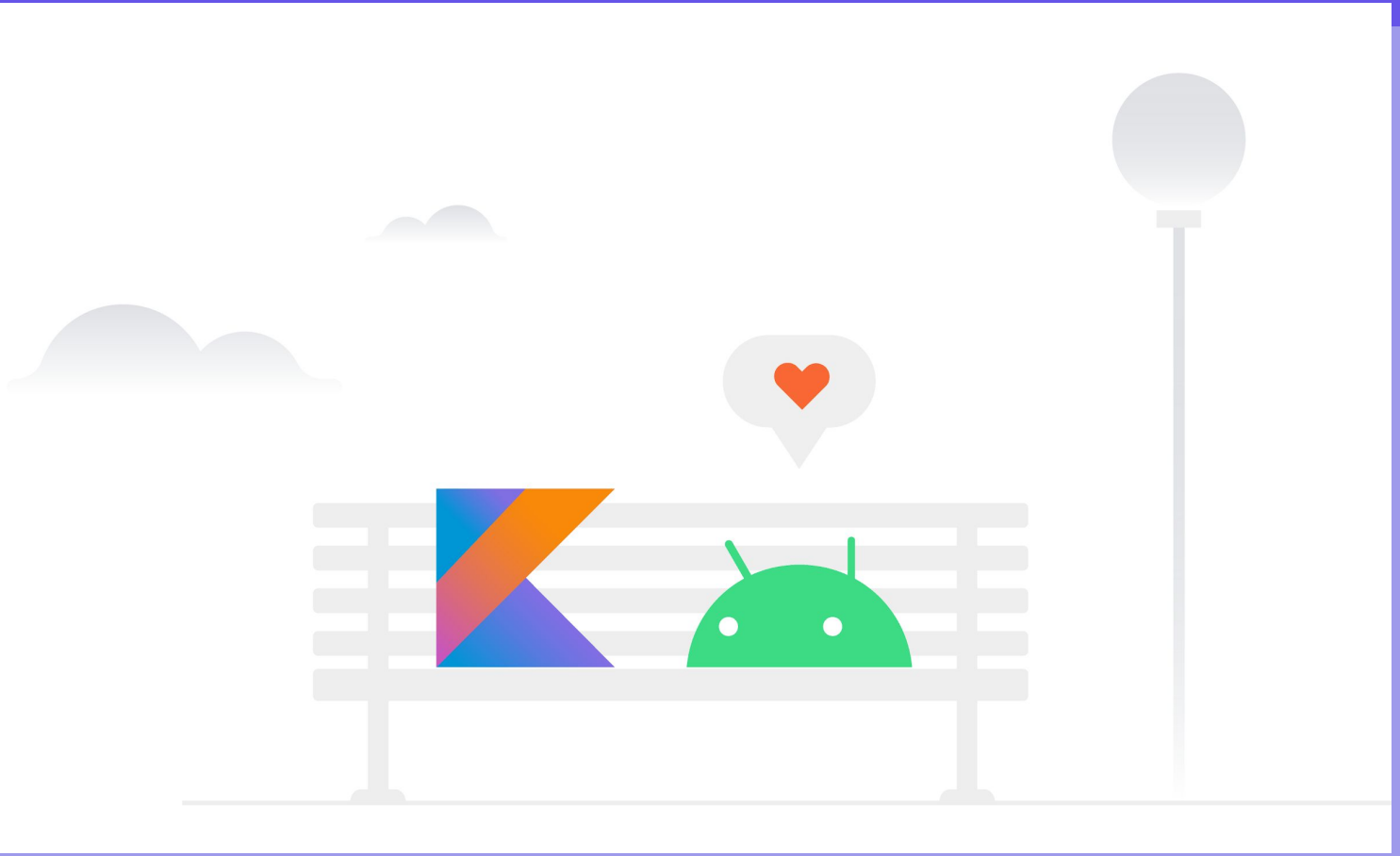


# Coroutines learning

goo.gle/coroutines-posts
- Coroutines: First things first
- Cancellation in coroutines
- Exceptions in Coroutines
- The suspend modifier — Under the hood

goo.gle/coroutines-videos
- Coroutines 101
- Android Coroutines: How to manage async tasks in Kotlin

goo.gle/kotlin-codelabs
- Use Kotlin Coroutines in your Android App
- Learn advanced coroutines with Kotlin Flow and LiveData
- Building a Kotlin extensions library



# Android 💚 Kotlin

- Productive Language
- Growing support
- New releases & tooling
- Coroutines recommended for async programming
- Expanding docs, codelabs & videos

# Thanks!
# Have a nice Kotlin

@FMuntenescu