

Beyond the UI

Compose as a Foundation for Multiplatform Apps

mDevCamp - June '25 🇨🇿

Ash Davies | ashdavies.dev

Android GDE Berlin

Jetpack Compose UI

**[github.com/androidx/androidx/tree/
androidx-main/compose/ui](https://github.com/androidx/androidx/tree/androidx-main/compose/ui)**



```
@Composable
fun JetpackCompose() {
    Card {
        var expanded by remember { mutableStateOf(false) }
        Column(Modifier.clickable { expanded = !expanded }) {
            Image(painterResource(R.drawable.jetpack_compose))
            AnimatedVisibility(expanded) {
                Text(
                    text = "Jetpack Compose",
                    style = MaterialTheme.typography.bodyLarge,
                )
            }
        }
    }
}
```

Jetpack Compose UI

**[github.com/androidx/androidx/tree/
androidx-main/compose/ui](https://github.com/androidx/androidx/tree/androidx-main/compose/ui)**



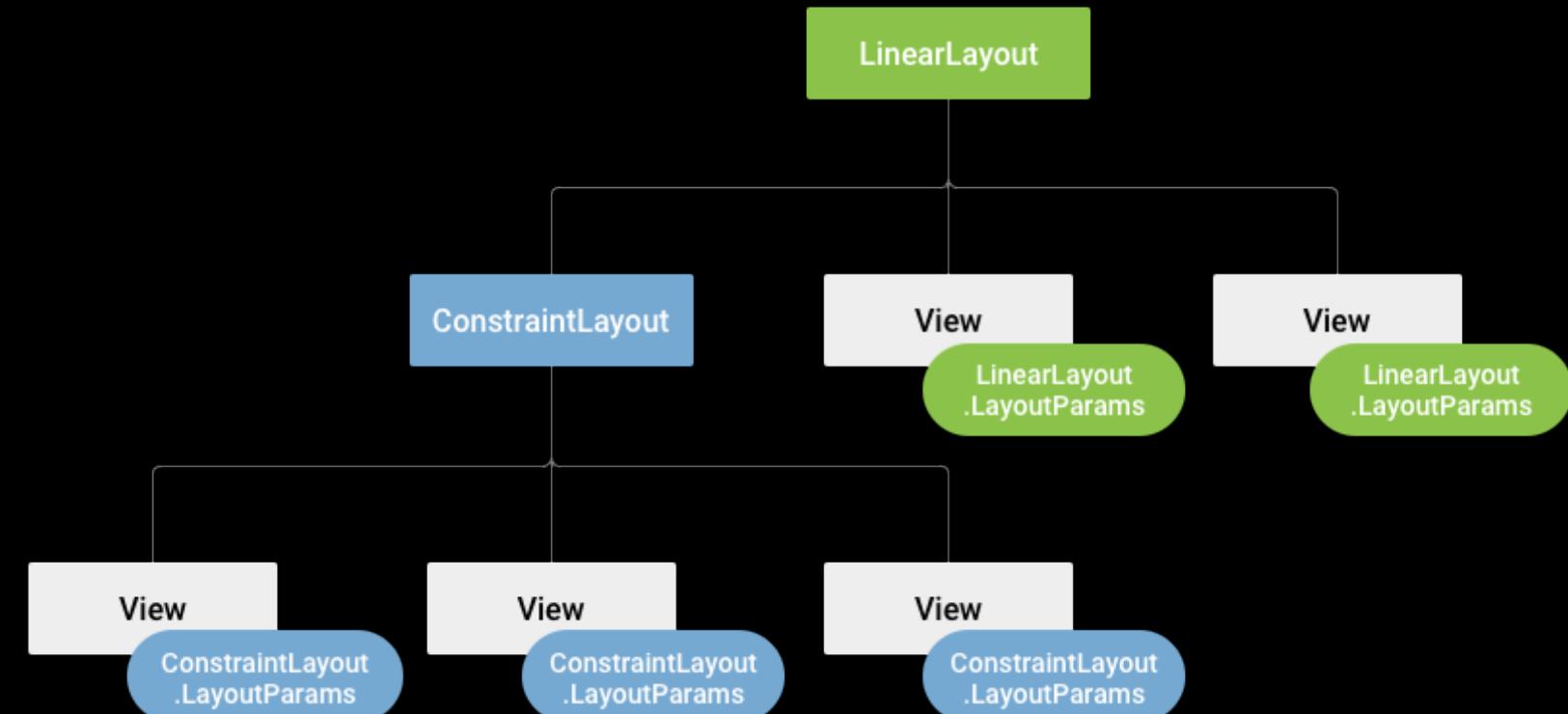
Android Layouts

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />

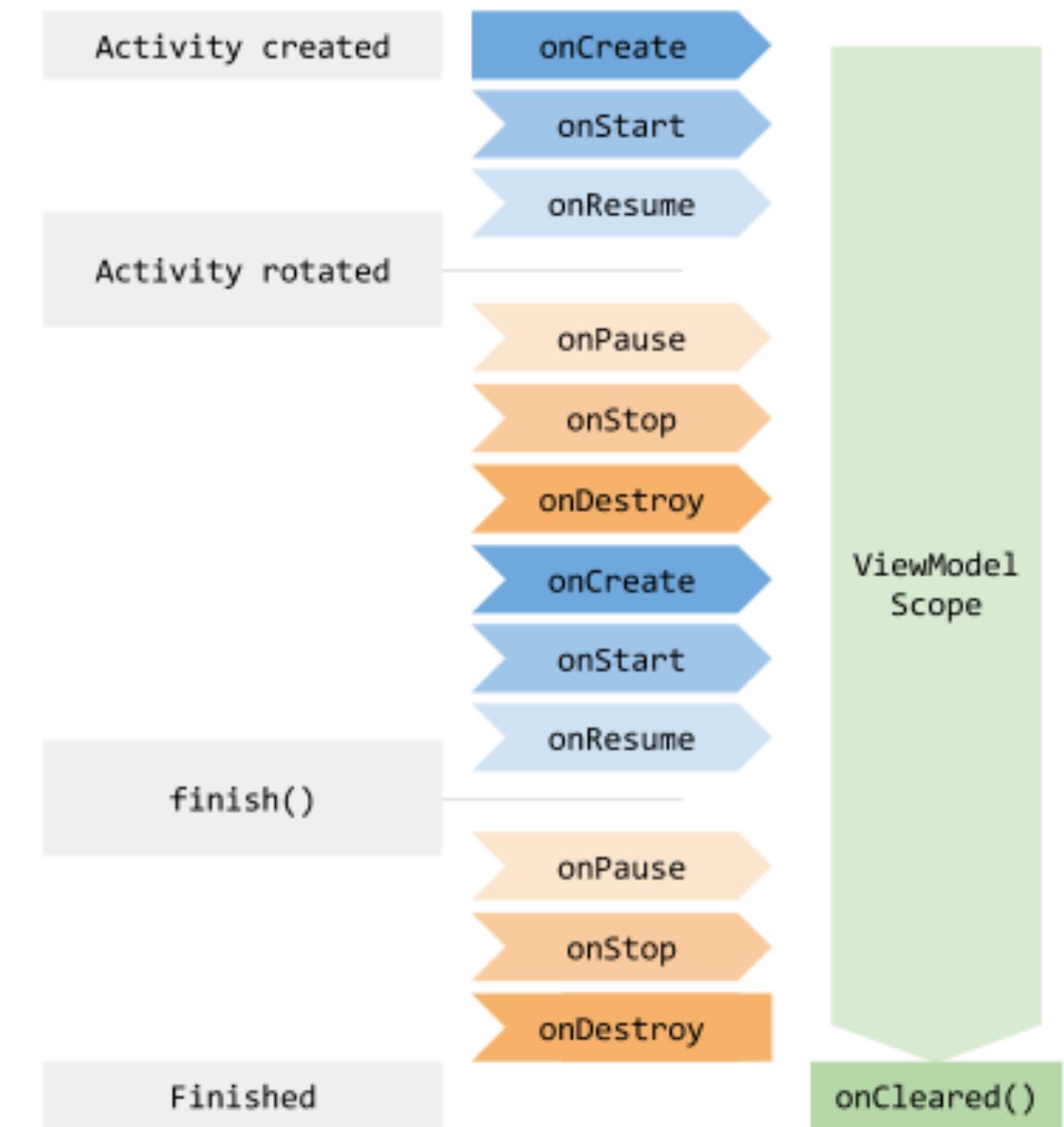
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />

</LinearLayout>
```



Android Layouts

Lifecycle

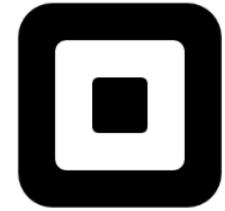


Android Layouts

Lifecycle Abstraction

square/mortar

A simple library that makes it easy to pair thin views with dedicated controllers, isolated from most of the vagaries...



20

Contributors

32

Issues

2k

Stars

154

Forks

square/flow

Name UI states, navigate between them, remember where you've been.



25

Contributors

33

Issues

3k

Stars

238

Forks

Android Layouts

Lifecycle Abstraction

square/workflow-kotlin

A Swift and Kotlin library for making composable state machines, and UIs driven by those state machines.

66
Contributors

139
Issues

25
Discussions

1k
Stars

104
Forks



uber/RIBs

Uber's cross-platform mobile architecture framework - Android Repository

63
Contributors

19
Used by

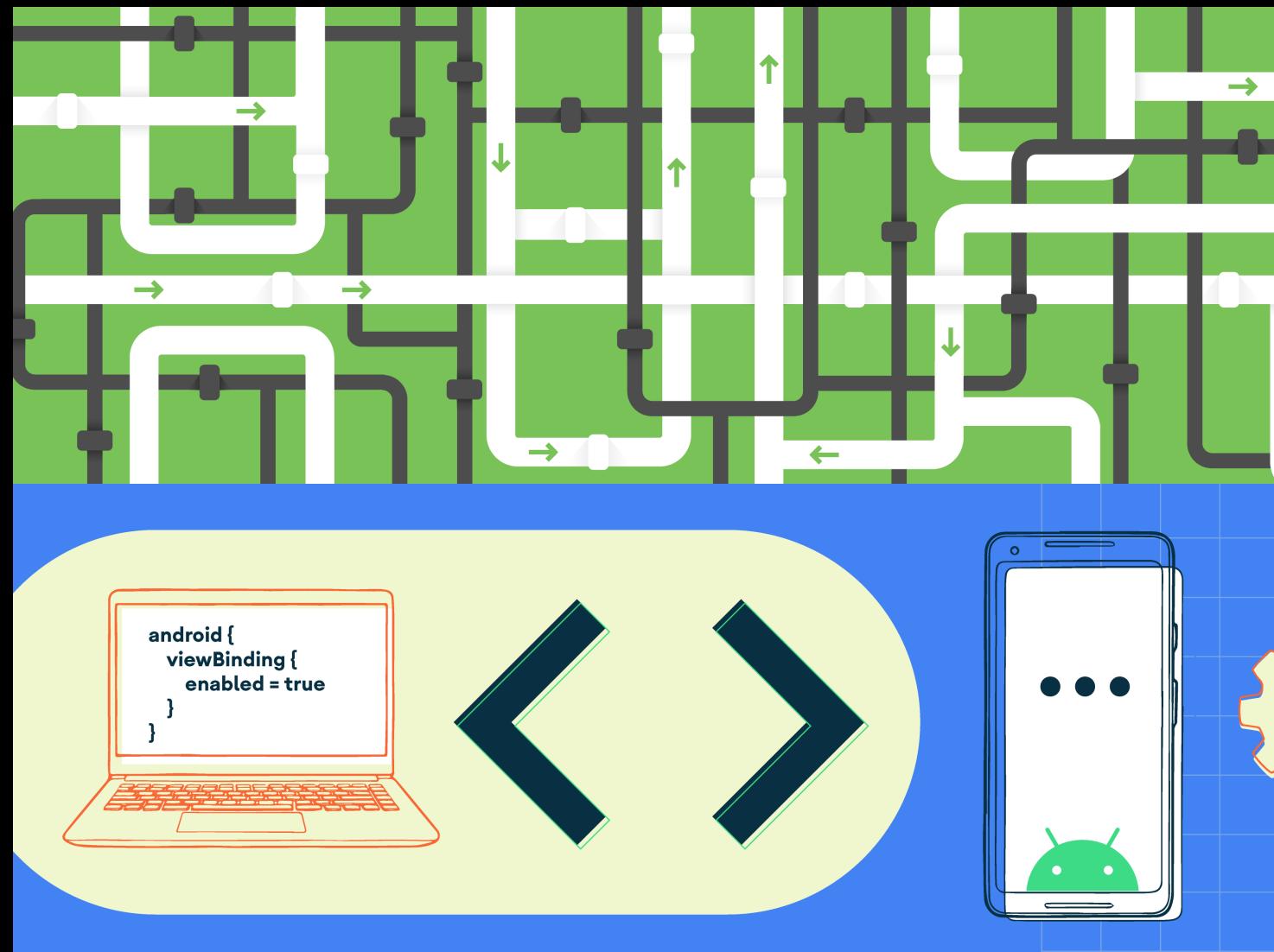
8k
Stars

906
Forks



Android Layouts

Kotlin Interop



Anko^K

LouisCAD/Splitties

A collection of hand-crafted extensions for your Kotlin projects.



11

Contributors

51

Issues

3

Discussions

3k

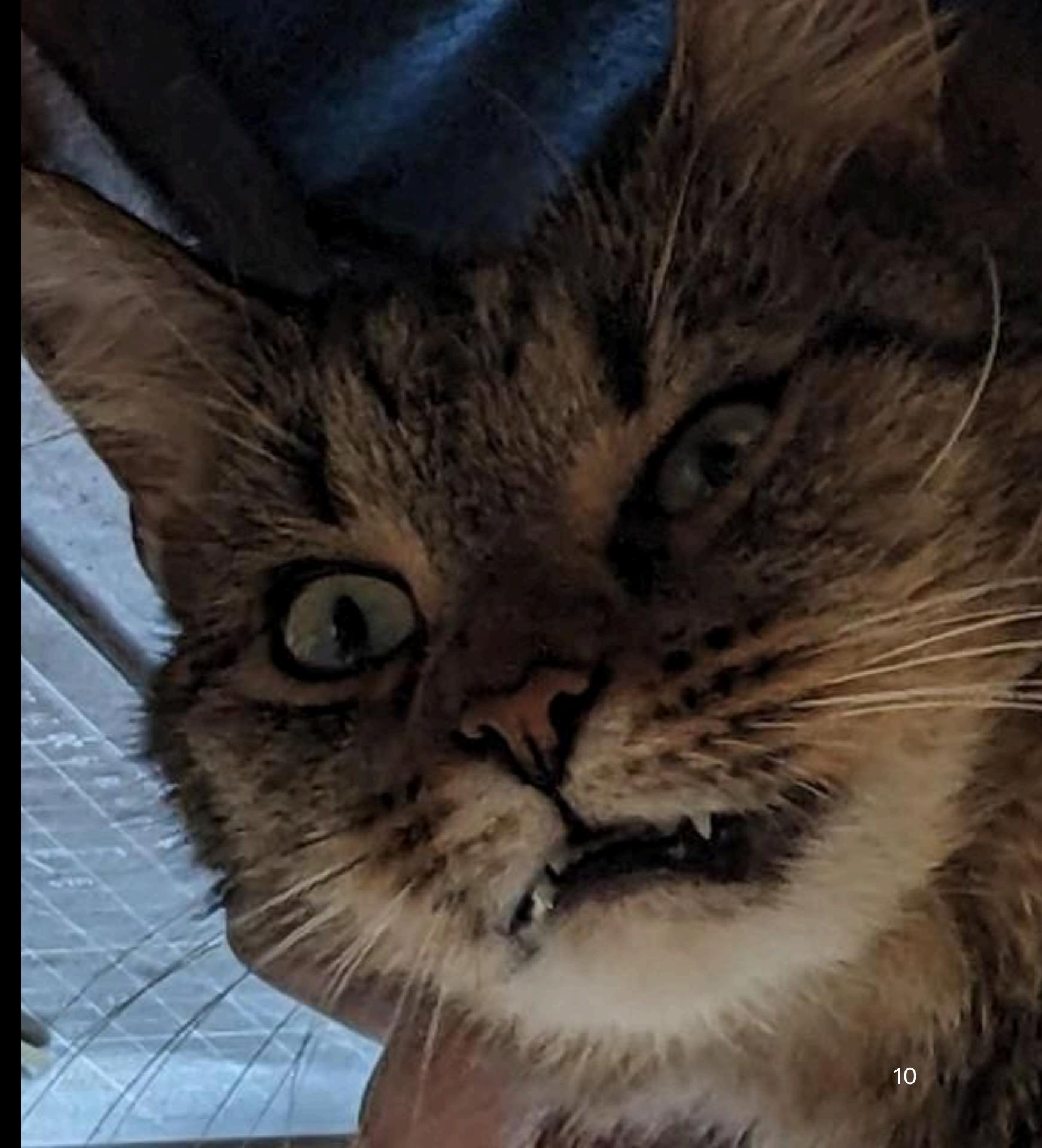
Stars

162

Forks



XML?!



Jetpack Compose UI



1.0 (July 2021)



Declarative UI Framework

Compose UI

Intelligent Recomposition



Compose UI

```
@Composable
fun ClickCounter(clicks: Int, onClick: () -> Unit) {
    Button(onClick = onClick) {
        Text("I've been clicked $clicks times")
    }
}
```

Kotlin Language Features 💪

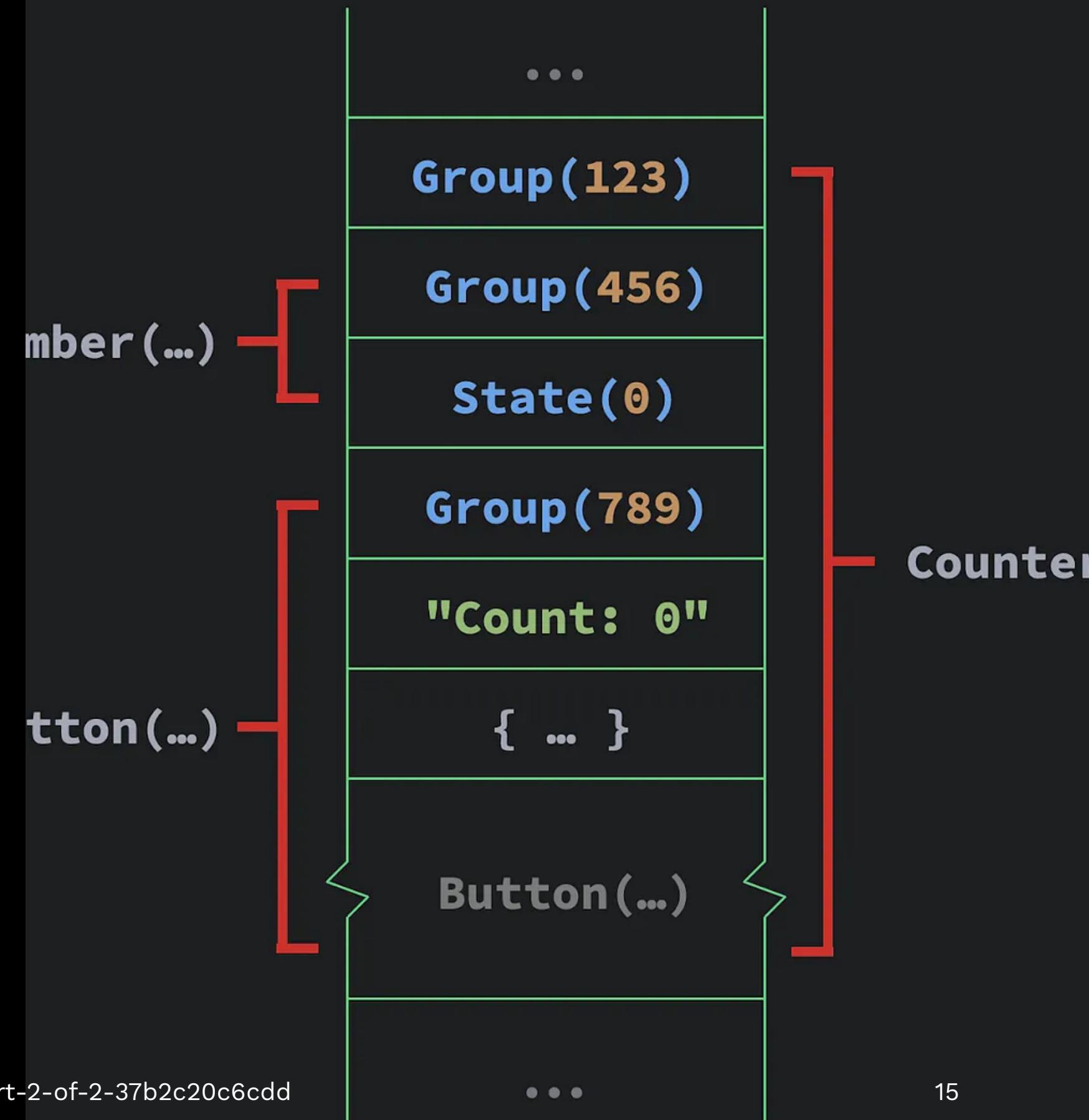
Compose UI

- Default parameters
- Higher order functions
- Trailing lambdas
- Scopes / Receivers
- Delegated properties
- ...

Jetpack Compose UI

Under-the-hood

- Kotlin compiler plugin
- Gap buffer data structure

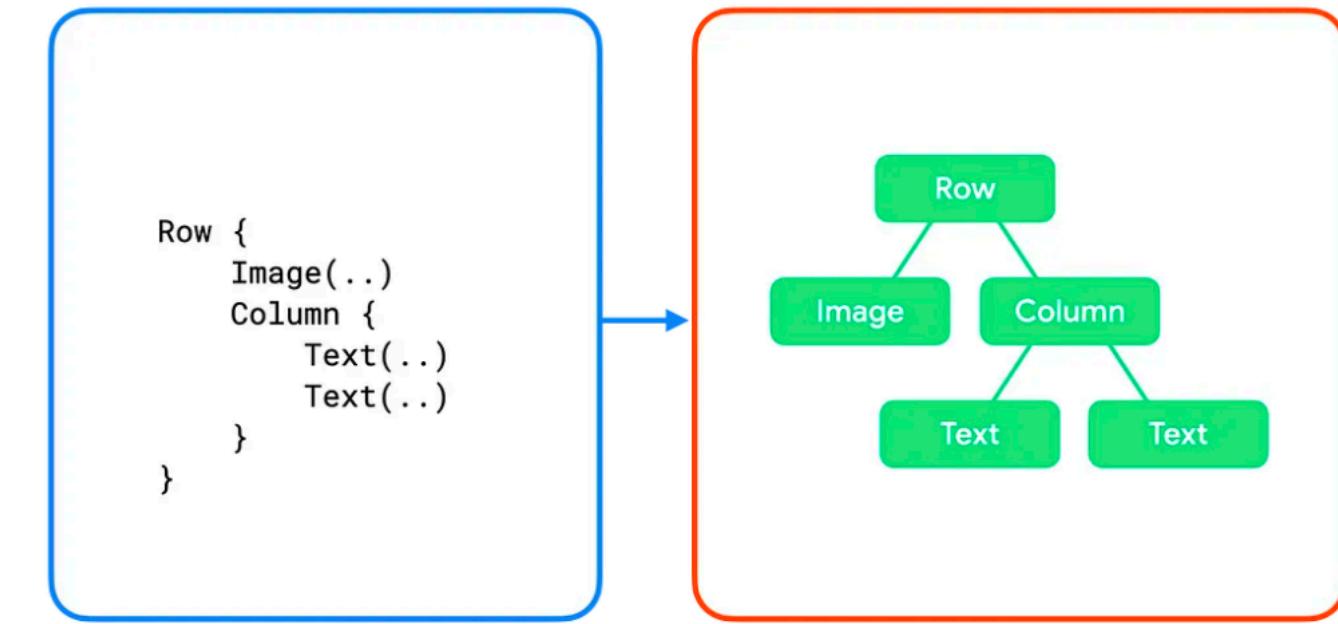


Jetpack Compose UI

Talk is Cheap

```
@Composable
fun Counter() {
    var count by remember { mutableStateOf(0) }

    Button(onClick = { count += 1 }) {
        Text("Count: $count")
    }
}
```



**Compose is, at its core, a general-purpose
tool for managing a tree of nodes of any
type ... a “tree of nodes” describes just
about anything, and as a result Compose
can target just about anything.**

— Jake Wharton

Compose != Compose UI

Kotlin Multiplatform

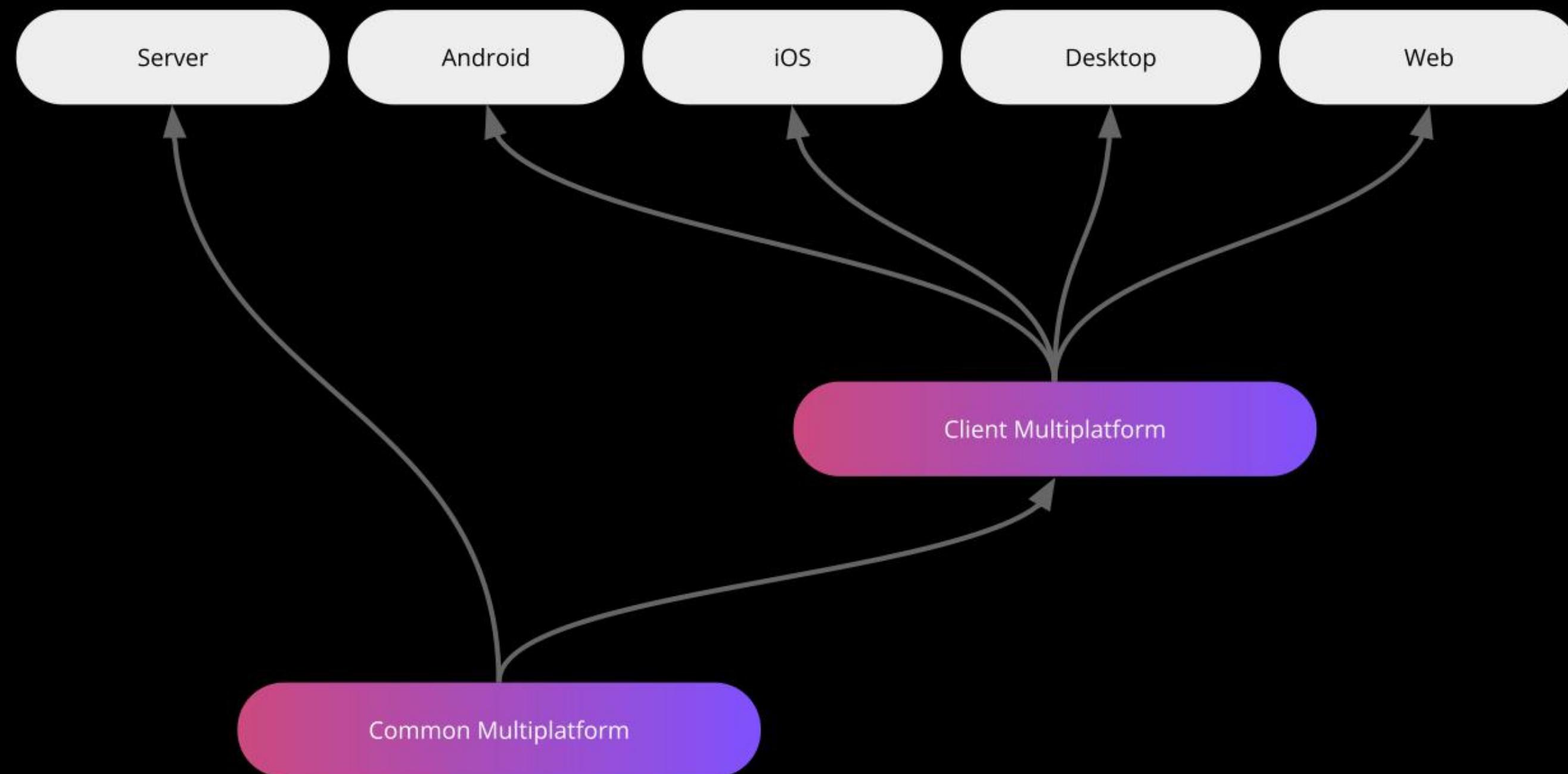
Stable (1.9.20)

The Before Times



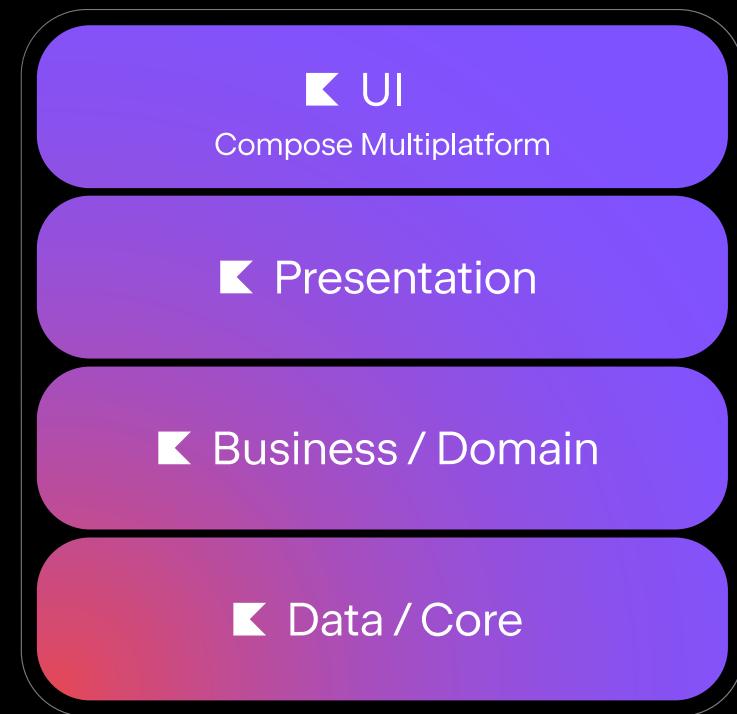
The Before Times





Compose Multiplatform

v1.0 | 2021



JetBrains/compose- multiplatform-core



Development environment for Android Jetpack extension libraries under the androidx namespace.

Synchronized with Android Jetpack's primary development branch on AOSP.

0

Contributors

0

Issues

541

Stars

87

Forks



Compose Multiplatform Migration

- Change artifact coordinates
- Do nothing
- Profit

```
// Compiled Compose code

fun Counter($composer: Composer) {
    $composer.startRestartGroup(-1913267612)

    /* ... */

    $composer.endRestartGroup()
}
```

// Compiled Coroutines code

```
fun counter($completion: Continuation) {
    /* ... */
}
```

KotlinX Coroutines

```
@SuppressLint("DEPRECATION")
class CallbackLoginPresenter(
    private val service: SessionService,
    private val goTo: (Screen) -> Unit,
) {
    /* ... */

    inner class LoginAsyncTask : AsyncTask<Submit,Void,LoginResult>() {
        private var username: String = ""

        override fun doInBackground(vararg events: Submit?): LoginResult {
            val event = events[0]!!
            username = event.username
            return runBlocking { service.login(event.username, event.password) }
        }

        override fun onPostExecute(result: LoginResult?) {
            when (result) {
                is Success -> goTo(LoggedInScreen(username))
                is Failure -> goTo(ErrorScreen(result.throwable?.message ?: ""))
                else -> {}
            }
        }
    }
}
```

```
Observable.just("Hey")
    .subscribeOn(Schedulers.io())
    .map(String::length)
    .subscribeOn(Schedulers.computation())
    .observeOn(AndroidSchedulers.mainThread())
    .doOnSubscribe { doAction() }
    .flatMap {
        doAction()

        Observable.timer(1, TimeUnit.SECONDS)
            .subscribeOn(Schedulers.single())
            .doOnSubscribe { doAction() }
    }
    .subscribe { doAction() }
```

KotlinX Coroutines

- Lightweight memory usage
- Structured concurrency
- Cancellation propagation
- Lifecycle aware

KotlinX Coroutines

- Native library
- Imperative syntax
- suspend fun

Reactive Architecture

- Push (not pull)
- Unidirectional Data Flow
- Declarative
- Idempotent

```
downloadManager
    .downloadFile("https://.../")
    .addOnCompletionListener { result ->
        fileManager
            .saveFile("storage/file", result)
            .addOnCompletionListener { success ->
                if (success) {
                    println("Downloaded file successfully")
                }
            }
    }
}
```

```
downloadManager
    .downloadFile("https://.../")
    .flatMap { result ->
        fileManager.saveFile("storage/file", result)
    }
    .observe { success ->
        if (success) {
            println("Downloaded file successfully")
        }
    }
}
```

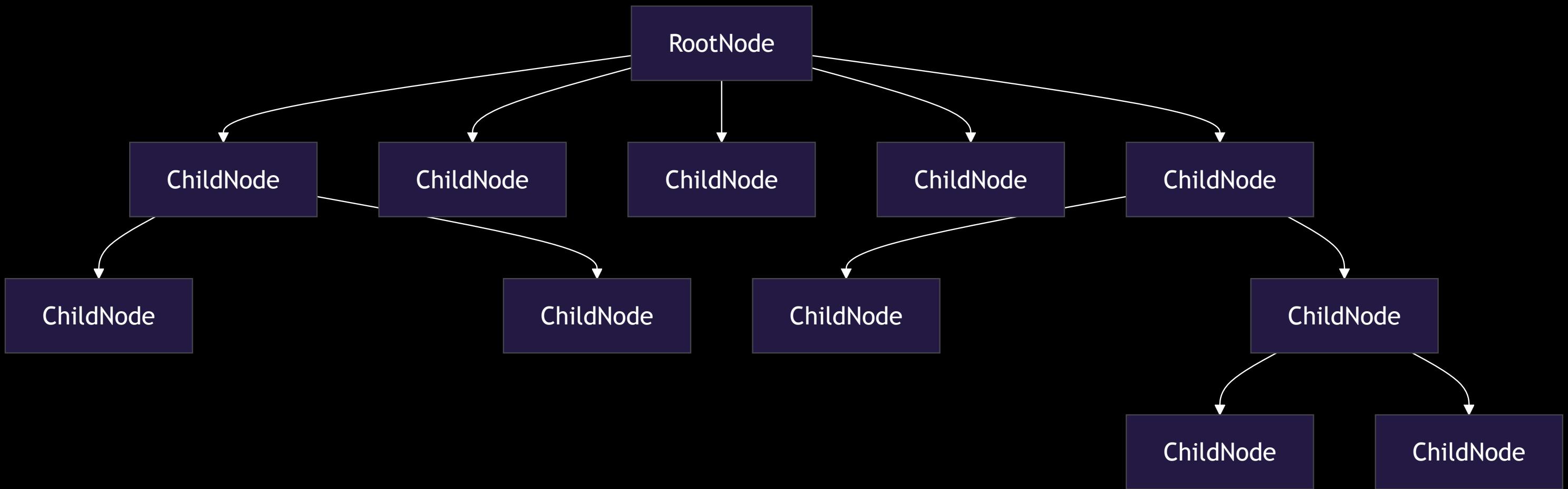
```
val file = downloadFile("https://.../")
val success = fileManager.saveFile("storage/file", file)

if (success) {
    println("Downloaded file successfully")
}
```



```
- downloadManager
-   .downloadFile("https://.../")
-   .flatMap { result ->
-     fileManager.writeFile("storage/file", result)
-   }
-   .observe { success ->
-     if (success) {
-       println("Downloaded file successfully")
-     }
-   }
- }

+ downloadManager.
+   downloadFile("https://.../")
+   .flatMapLatest { state ->
+     when (state) {
+       is State.Loaded ->
+         stateFileManager
+           .writeFile("storage/file", state.value)
+ 
+       else -> state
+     }
+   }
+   .collect { state ->
+     when (state) {
+       is State.Saved ->
+         println("Downloaded file successfully")
+ 
+       is State.Loading ->
+         /* ... */
+     }
+ }
```



```
val downloadState = downloadManager
    .downloadFile("https://.../")
    .collectAsState(State.Downloading)

val fileState = when(downloadState) {
    is State.Loaded ->
        stateFileManager
            .saveFile("storage/file", downloadState.value)
    else -> downloadState
}

when (fileState) {
    is State.Loading -> /* ... */

    is State.Saved -> LaunchedEffect(fileState) {
        println("Downloaded file successfully")
    }
}
```



cashapp/molecule

Build a StateFlow stream using Jetpack Compose



25

Contributors

24

Issues

18

Discussions

2k

Stars

91

Forks



Molecule

```
fun CoroutineScope.launchCounter(): StateFlow<Int> {
    return launchMolecule(mode = ContextClock) {
        var count by remember { mutableStateOf(0) }

        LaunchedEffect(Unit) {
            while (true) {
                delay(1_000)
                count++
            }
        }
        count
    }
}
```

Testing

```
@Test
fun counter() = runTest {
    moleculeFlow(RecompositionMode.Immediate) {
        Counter()
    }.test {
        assertEquals(0, awaitItem())
        assertEquals(1, awaitItem())
        assertEquals(2, awaitItem())
        cancel()
    }
}
```

cashapp/turbine

A testing library for kotlinx.coroutines Flow



33

Contributors

11

Issues

14

Discussions

3k

Stars

113

Forks



Turbine

```
flowOf("one", "two").test {  
    assertEquals("one", awaitItem())  
    assertEquals("two", awaitItem())  
    awaitComplete()  
}
```

Architecture



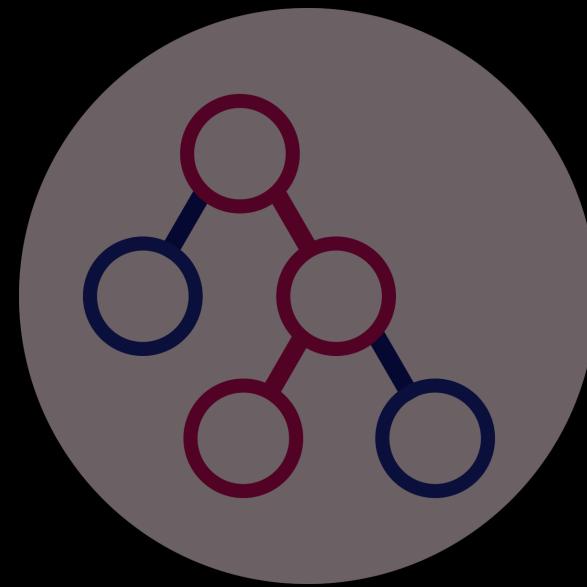
Navigation



slackhq/circuit

⚡ A Compose-driven architecture for Kotlin and Android applications.

33 Contributors 14 Issues 86 Discussions 2k Stars 91 Forks

A GitHub repository card for the 'circuit' repository. It shows the repository name 'slackhq/circuit', a description '⚡ A Compose-driven architecture for Kotlin and Android applications.', and various statistics: 33 contributors, 14 issues, 86 discussions, 2k stars, and 91 forks. The card also features the Slack logo and a GitHub icon.

Jetpack Navigation 3

Pre-Compose Era



Tooling in Compose Multiplatform

Decompose & Esenty

arkivanov.github.io/Decompose

```
import com.arkivanov.decompose.ComponentContext

class DefaultRootComponent(
    componentContext: ComponentContext,
) : RootComponent, ComponentContext by componentContext {

    init {
        lifecycle... // Access the Lifecycle
        stateKeeper... // Access the StateKeeper
        instanceKeeper... // Access the InstanceKeeper
        backHandler... // Access the BackHandler
    }
}
```

adrielcafe/voyager

voyager.adriel.cafe



Voyager 🚀

```
class PostListScreen : Screen {

    @Composable
    override fun Content() {
        // ...
    }

    @Composable
    private fun PostCard(post: Post) {
        val navigator = LocalNavigator.currentOrThrow

        Card(
            modifier = Modifier.clickable {
                navigator.push(PostDetailsScreen(post.id))
            }
        ) {
            // ...
        }
    }
}
```

appyx

bumble-tech.github.io/appyx

PreCompose

github.com/Tlaster/PreCompose

cashapp/molecule

Build a StateFlow stream using Jetpack Compose



25

Contributors

24

Issues

18

Discussions

2k

Stars

91

Forks



slackhq/circuit

⚡ A Compose-driven architecture for Kotlin and
Android applications.



33

Contributors

14

Issues

86

Discussions

2k

Stars

91

Forks



```
@Parcelize
data object CounterScreen : Screen {
    data class CounterState(
        val count: Int,
        val eventSink: (CounterEvent) -> Unit,
    ) : CircuitUiState

    sealed interface CounterEvent : CircuitUiEvent {
        data object Increment : CounterEvent
    }
}

@Composable
fun CounterPresenter(): CounterState {
    var count by rememberSaveable { mutableStateOf(0) }

    return CounterState(count) { event ->
        when (event) {
            CounterEvent.Increment -> count++
        }
    }
}

@Composable
fun Counter(state: CounterState) {
    Column(Modifier.align(Alignment.Center)) {
        Text(text = "Count: ${state.count}")

        Button(onClick = { state.eventSink(CounterEvent.Increment) }) {
            Icon(rememberVectorPainter(Icons.Filled.Add), "Increment")
        }
    }
}
```

But Why?



Prototyping





no restart.



Recomposition



Remember



Remember

```
var path by remember { mutableStateOf("https://...") }
val file = remember(path) {
    downloadManager.downloadFile(path)
}
```

Remember

```
var path by remember { mutableStateOf("https://...") }
val file = rememberSaveable(path) {
    // Must be Parcelable on Android!
    downloadManager.downloadFile(path)
}
```

Circuit

rememberRetained()

```
rememberRetainedCoroutineScope(  
)
```

Retaining Beyond ViewModels

Consistent State Patterns

- Hoisting, unidirectional data flow
- Shared reactive state handling

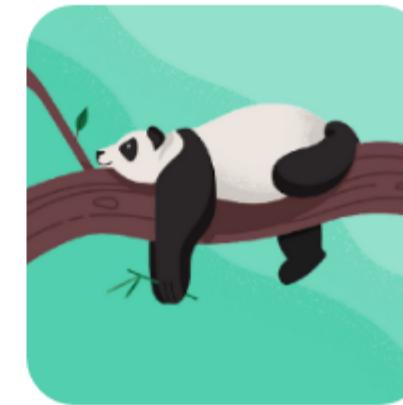
Compose Multiplatform

Material Theming

```
MaterialTheme(  
    colorScheme = /* ... */,  
    typography = /* ... */,  
    shapes = /* ... */,  
) {  
    // M3 app content  
}
```

alexzhirkevich/compose-cupertino

Compose Multiplatform UI components for iOS
(Cupertino Widgets)



11

Contributors

0

Issues

1k

Stars

52

Forks



```
@Composable  
fun AppTheme(  
    theme: Theme,  
    content: @Composable () -> Unit  
) {  
    AdaptiveTheme(  
        material = {  
            // Tweak this for your Material design  
            MaterialTheme(content = it)  
        },  
        cupertino = {  
            // Tweak this for your iOS design  
            CupertinoTheme(content = it)  
        },  
        target = theme,  
        content = content  
    )  
}
```

Search

Toggle layout direction



 SF Symbols [One >](#)

 Sections [Two >](#)

 Adaptive Widgets [Three >](#)

 Bottom Sheet [Four >](#)

IDE-First Experience

- JetBrains tools tightly integrated
- Live previews and navigation supported

Beyond the UI

Wrap-Up

- ✓ Compose is more than a UI toolkit
- ✓ Enables scalable, shared architecture
- ✓ Designed for Kotlin-first developers
- ✓ Multiplatform not just business logic

Thank You!

Ash Davies | ashdavies.dev

Android GDE Berlin

