

Navigation in a Multiplatform World

Choosing the Right Framework for your App

Droidcon NYC - September '24 

Ash Davies - SumUp

Android & Kotlin GDE Berlin

ashdavies.dev



Project timeline



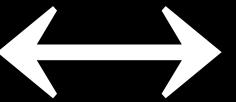


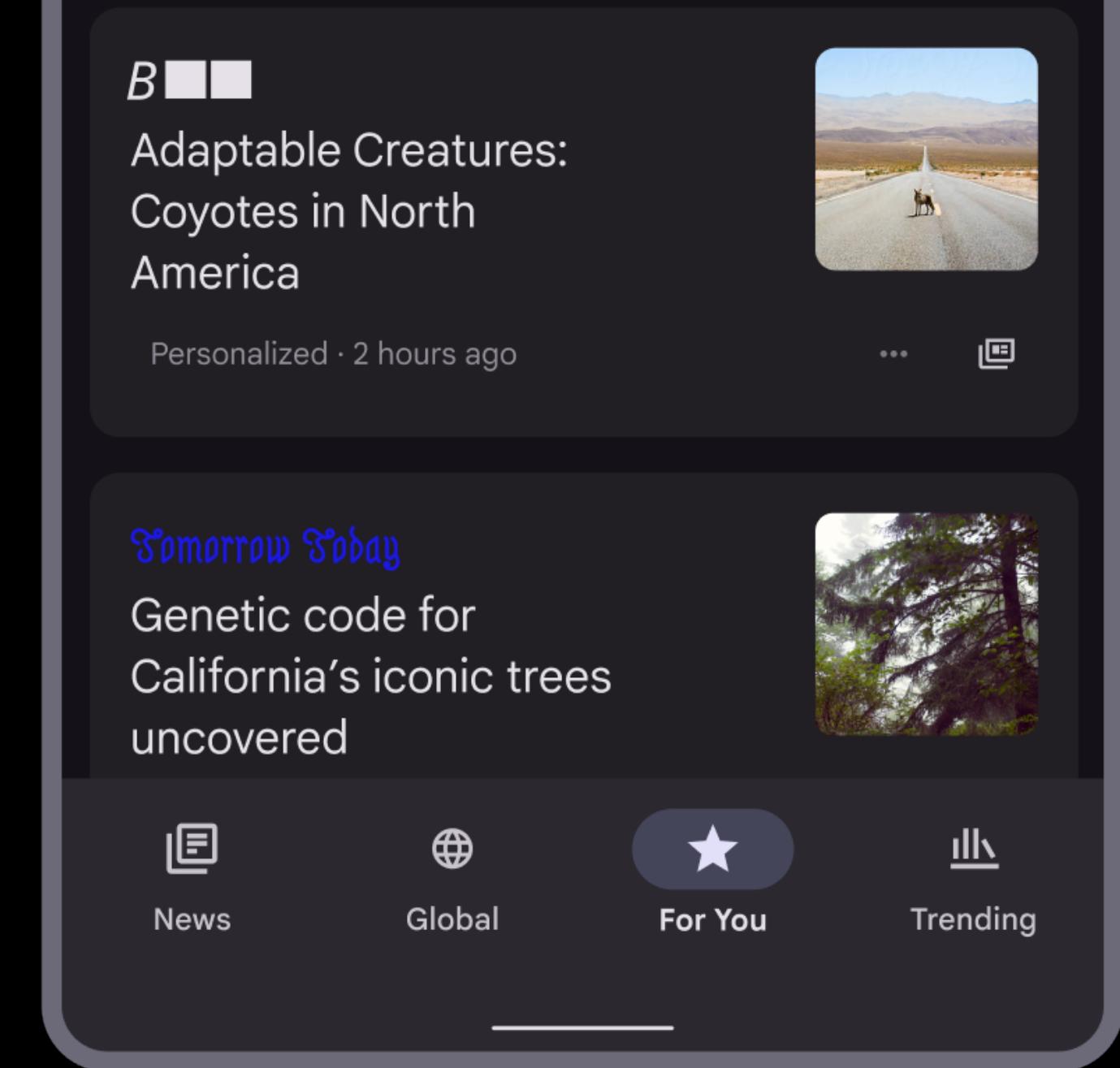
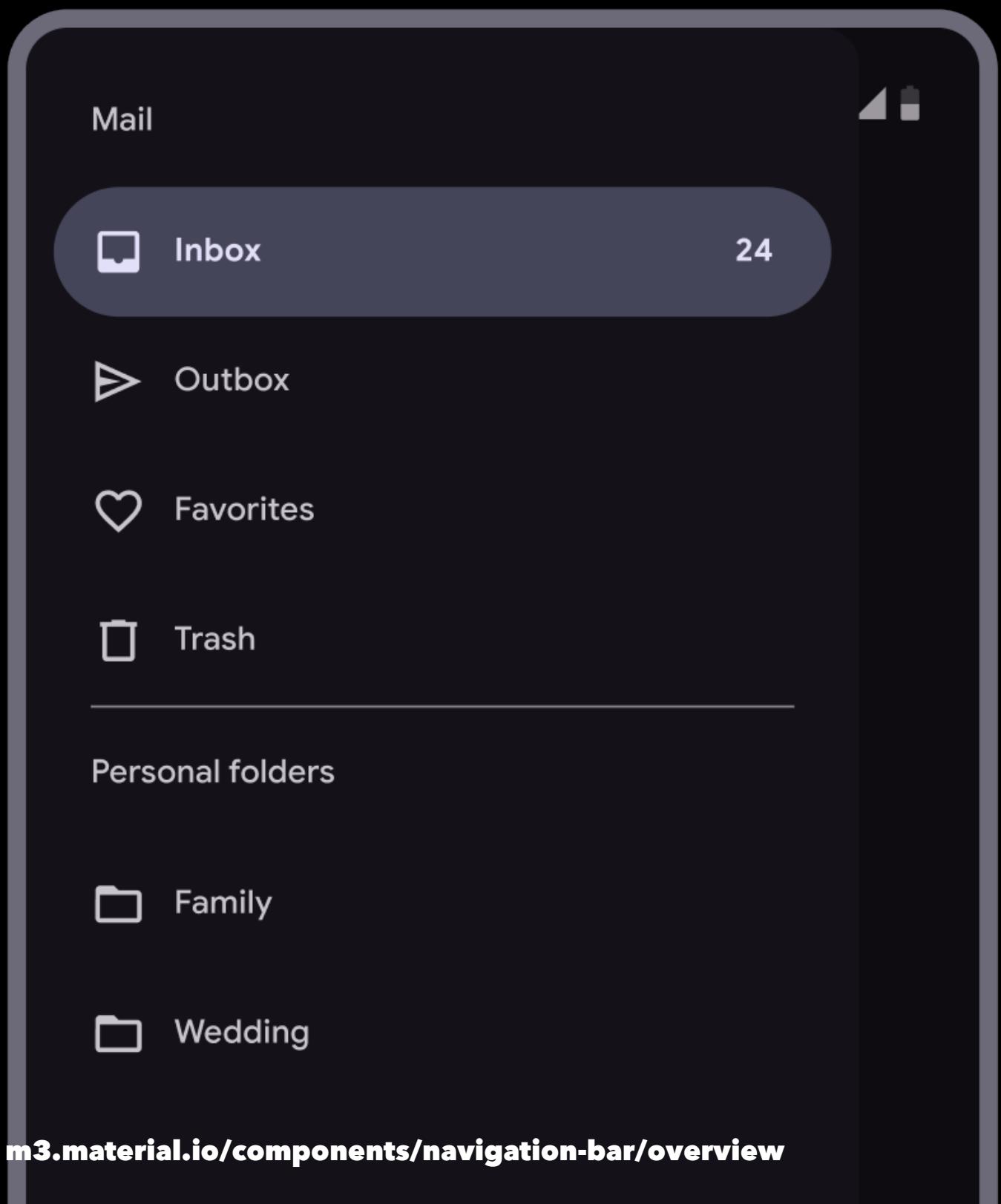
```
val history = ArrayDeque<Screen>()
```

```
history.addLast(ForwardScreen)
```

```
history.removeLast()
```

It's Complicated



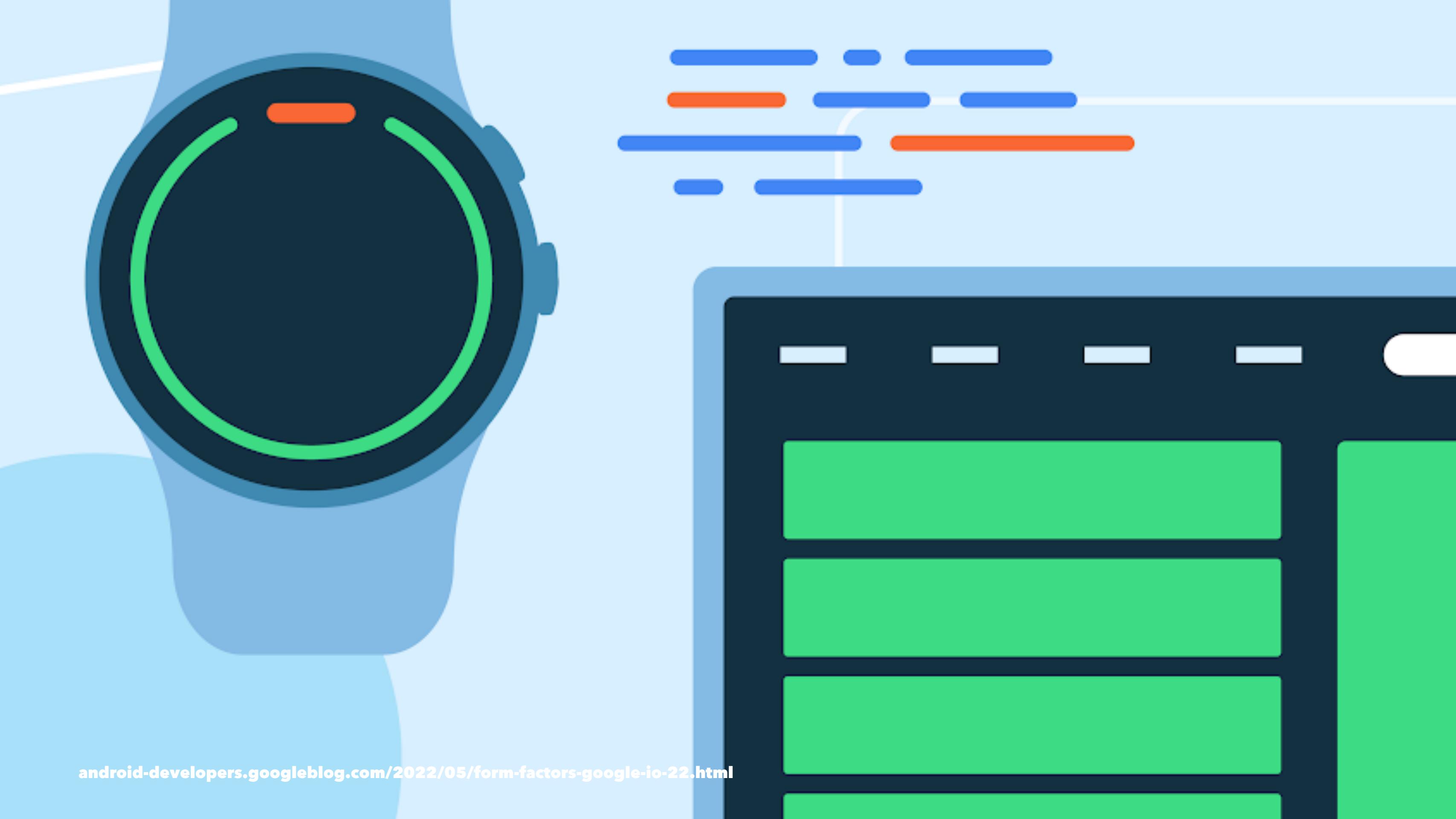




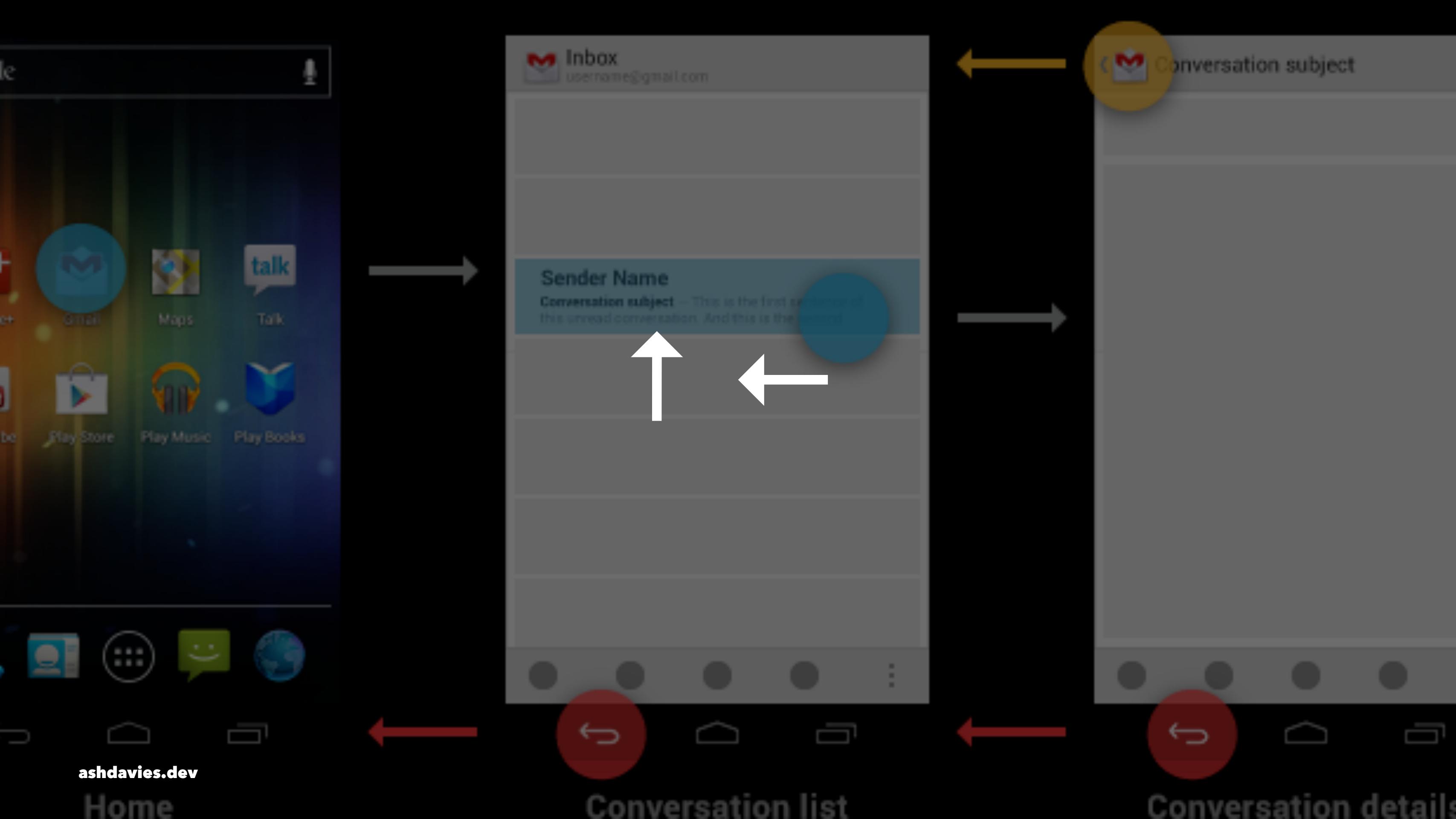
Hanan toiminta - Tap operation









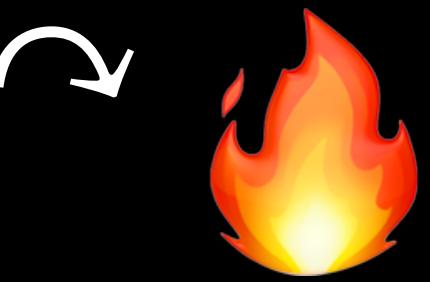


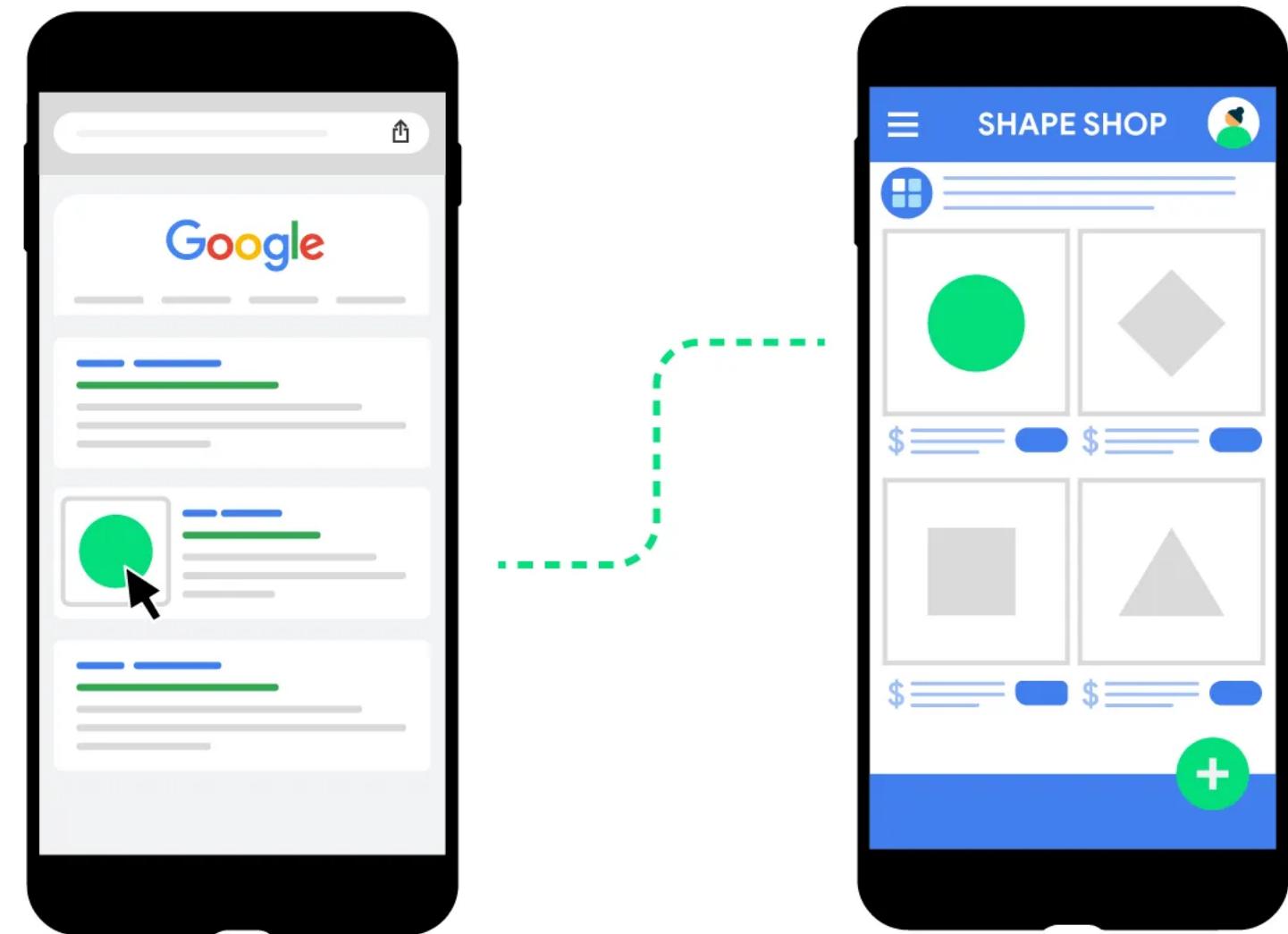
ashdavies.dev

Home

Conversation list

Conversation details







✖ Migrating

Move existing activity logic to fragment

- FragmentBindings -> ActivityBindings
- onCreate() -> onCreateView()
- onViewCreated()
- getViewLifecycleOwner()

@ashdavies

***"Once we have gotten in to this entry-point to your UI,
we really don't care how you organise the flow inside."***

– Dianne Hackborn, Android Framework team, 2016

Jetpack Navigation



Fragments

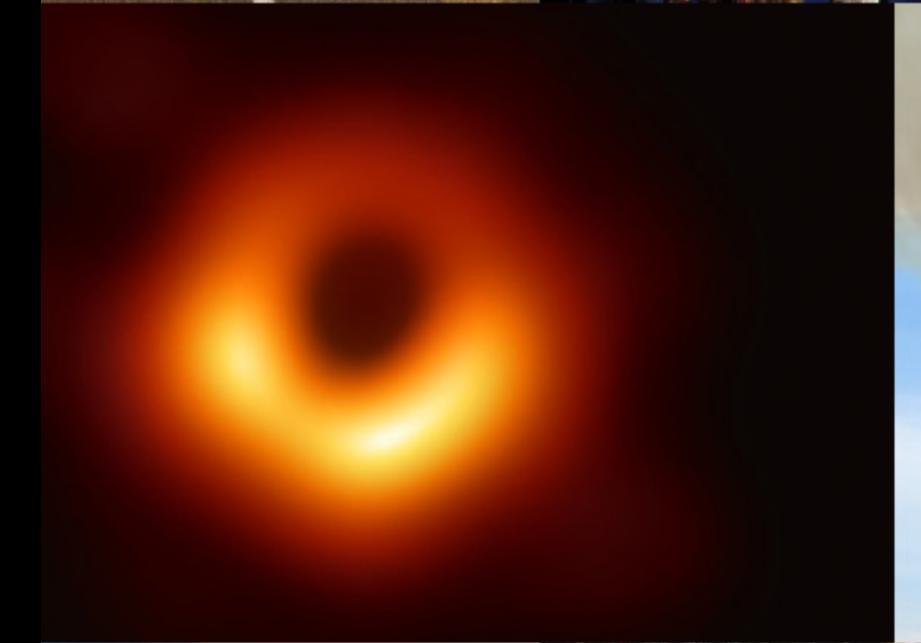
flow_step_one_dest

Step One

NAVIGATE NEXT STEP

Fragment Destinations

2019

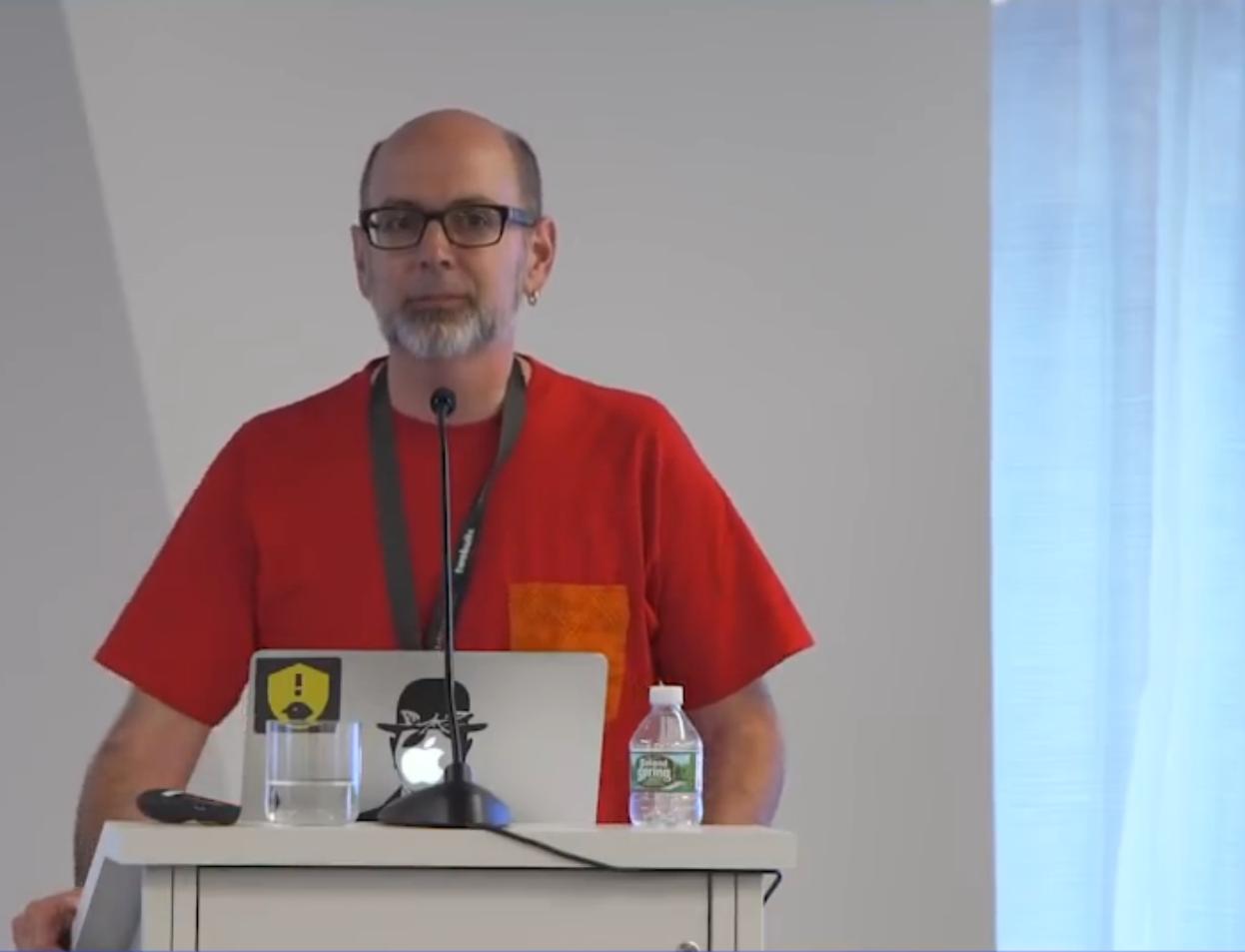


2014: mortar & flow

github.com/square/mortar

.droidconNYC

September 25-26 2017



reacting to code sprawl

Reactive Workflows

Ray Ryan



square/workflow

square.github.io/workflow

Why should I use this?

You work at Square ^_(ツ)_/^-

2016: uber/ribs

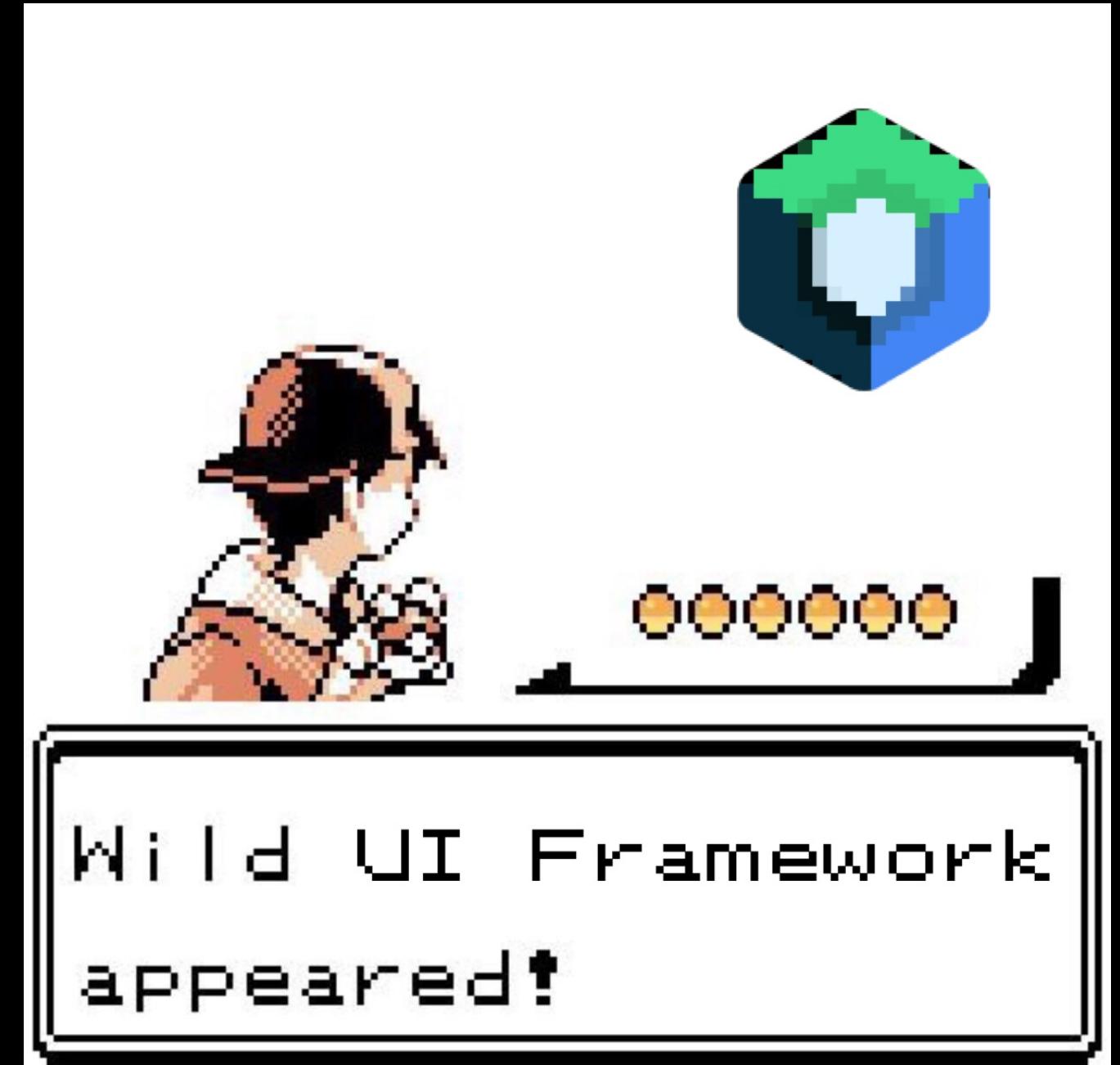
github.com/uber/RIBs



Moving On...

Compose UI

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



Compose UI

- Declarative UI Framework
- Open Source Kotlin
- Accelerate UI development
- Intuitive Idiomatic API

Obligatory Notice

Compose != Compose UI

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



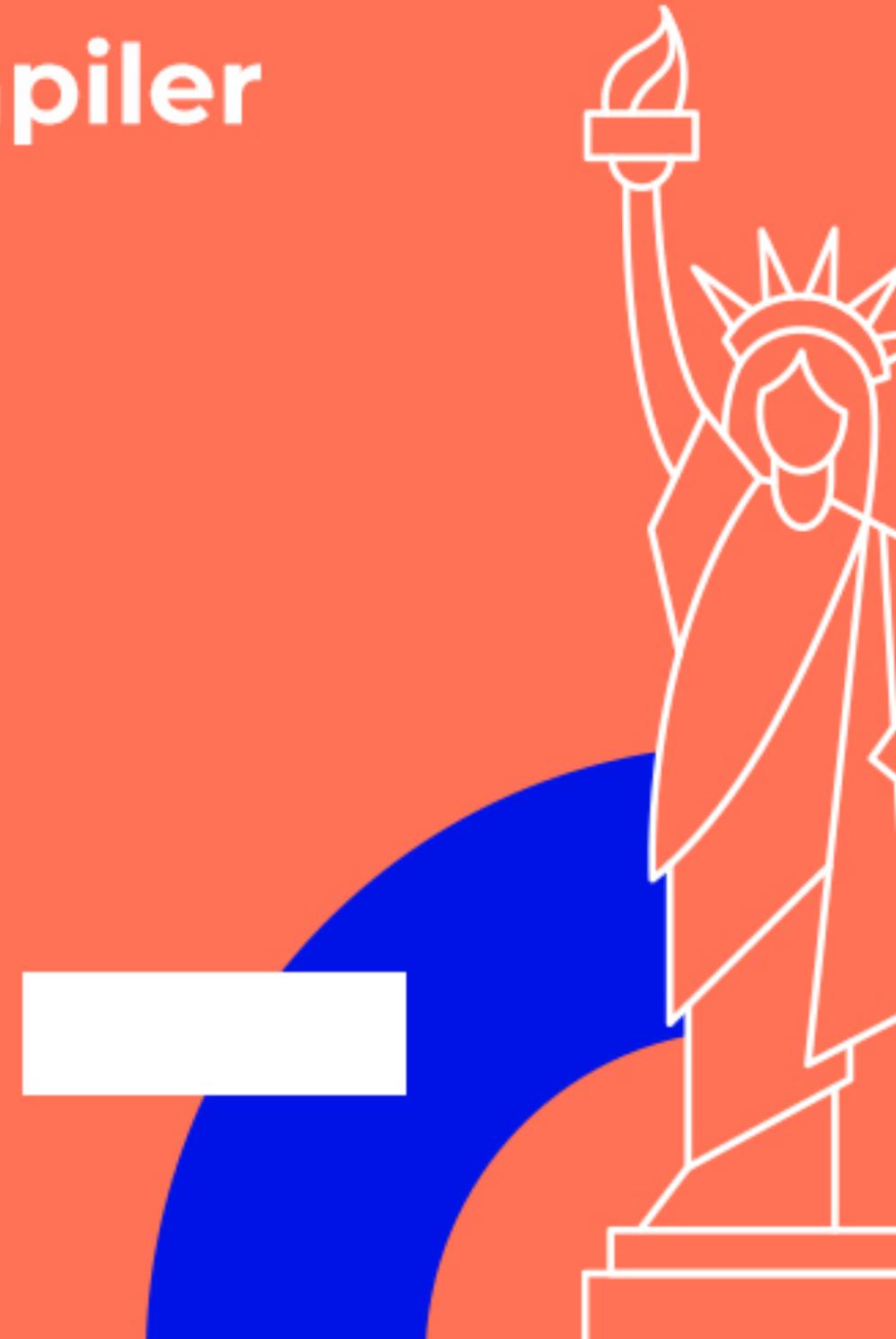
X Deep Dive into the Compose Compiler



**Mohit
Sarveiya**

Google Developers Expert
Kotlin & Android

SEPTEMBER 19th - 20th, 2024



Jetpack Navigation Compose

v2.4.0 (2021)

- Build a navigation graph with a @Composable Kotlin DSL
- Compose viewModel() scoped to navigation destination
- Destination level scope for rememberSaveable()
- Automatic back handling support

Jetpack Navigation Compose < v2.8.0

```
private const val HOME_ROUTE = "home"

NavHost(
    navController = navController,
    startDestination = HOME_ROUTE,
) {
    composable(route = HOME_ROUTE) {
        HomeScreen(
            onBackClick = navController::popBackStack,
            /* ... */
        )
    }
}
```

Jetpack Navigation Compose < v2.8.0

```
private const val DETAIL_ID_KEY = "detailId"
private const val DETAIL_ROUTE = "detail"

NavHost(
    navController = navController,
    startDestination = DETAIL_ROUTE,
) {
    composable(
        route = DETAIL_ROUTE,
        arguments = listOf(
            navArgument(DETAIL_ID_KEY) {
                type = NavType.StringType
                defaultValue = null
                nullable = true
            }
        )
    ) {
        DetailScreen(/* ... */)
    }
}
```

Jetpack Navigation Compose < v2.8.0

```
private const val DETAIL_ID_KEY = "detailId"

fun NavController.navigateToDelete(detailId: String) {
    navigate("detail?${DETAIL_ID_KEY}=$detailId")
}

savedStateHandle.getStateFlow(DETAIL_ID_KEY, null)
```

Jetpack Navigation Compose v2.8.0 (04.09.2024)

```
@Serializable  
data class DetailRoute(val id: String)  
  
NavHost(  
    navController = navController,  
    startDestination = "detail",  
) {  
    composable<DetailRoute> {  
        DetailScreen(/* ... */)  
    }  
}  
  
val route = savedStateHandle.toRoute<DetailRoute>()
```



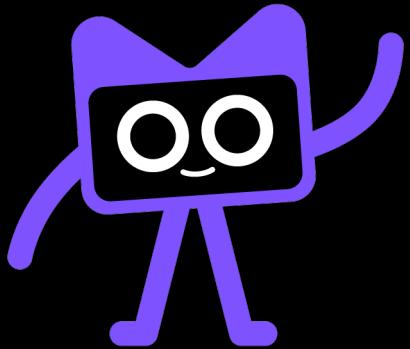
Navigation in a Multiplatform World

Choosing the Right Framework for your App

Android Navigation für N00bs
by Some Dude

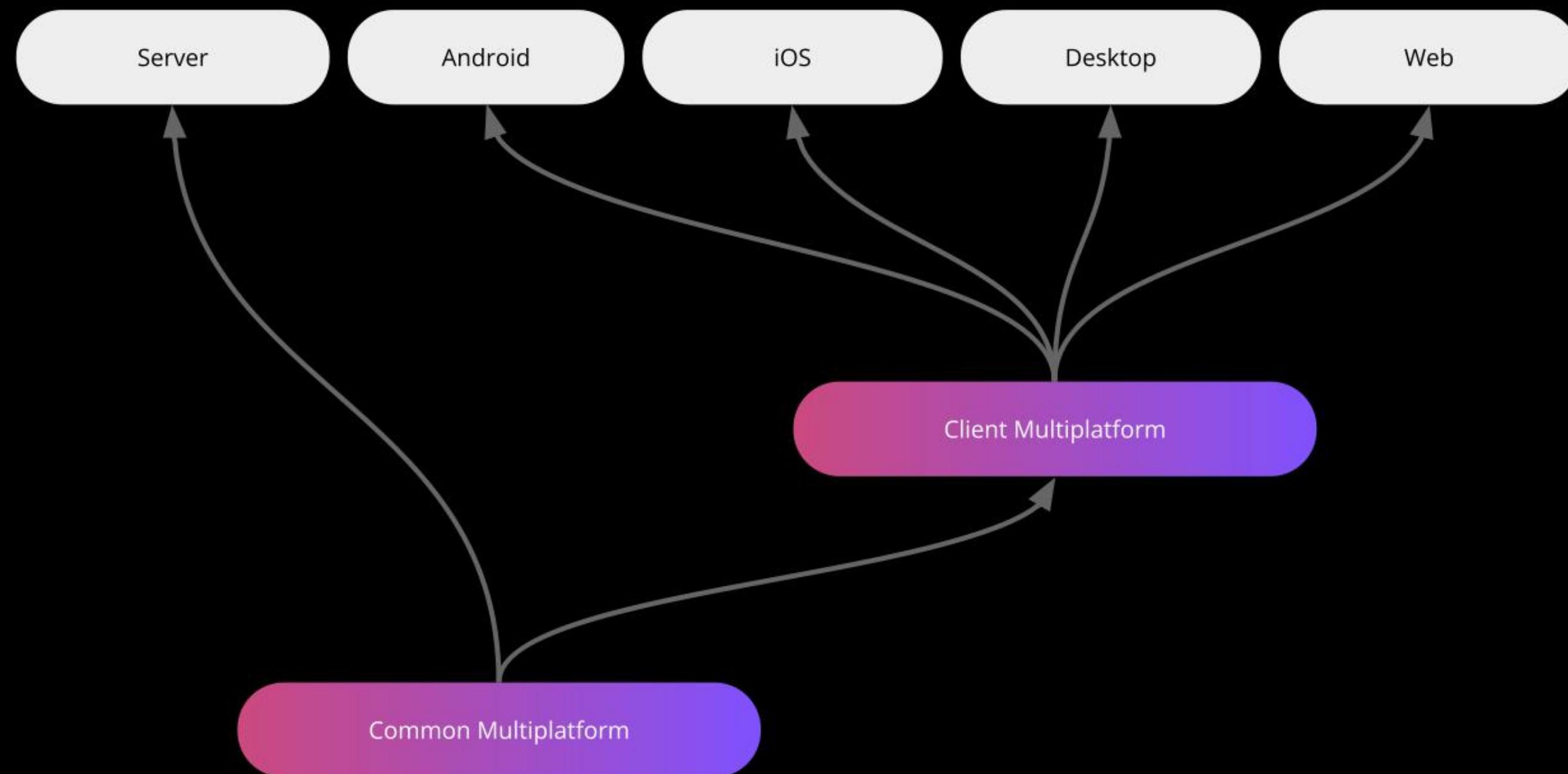
Kotlin Multiplatform

Stable (1.9.20)

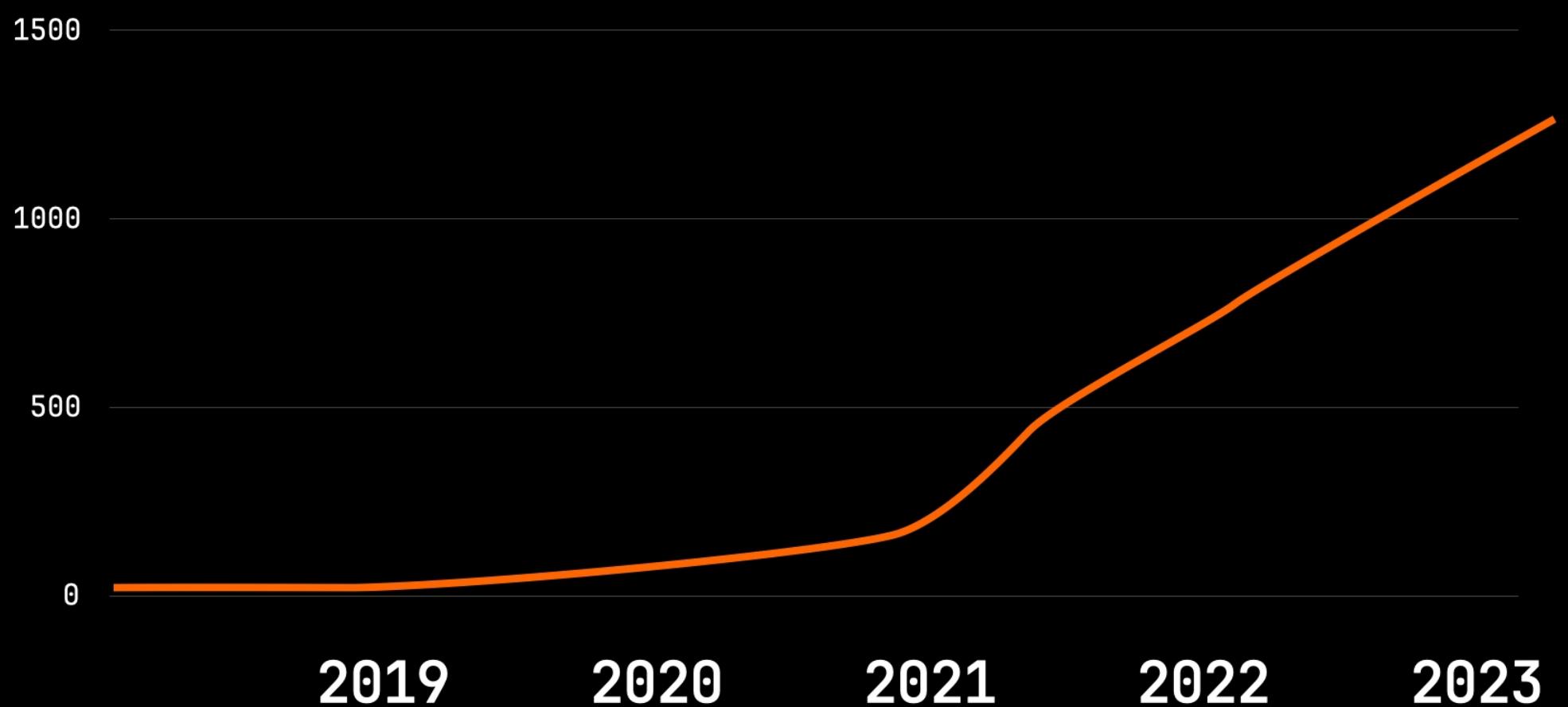


The Before-Times

The Before Times



Multiplatform Libraries by Year

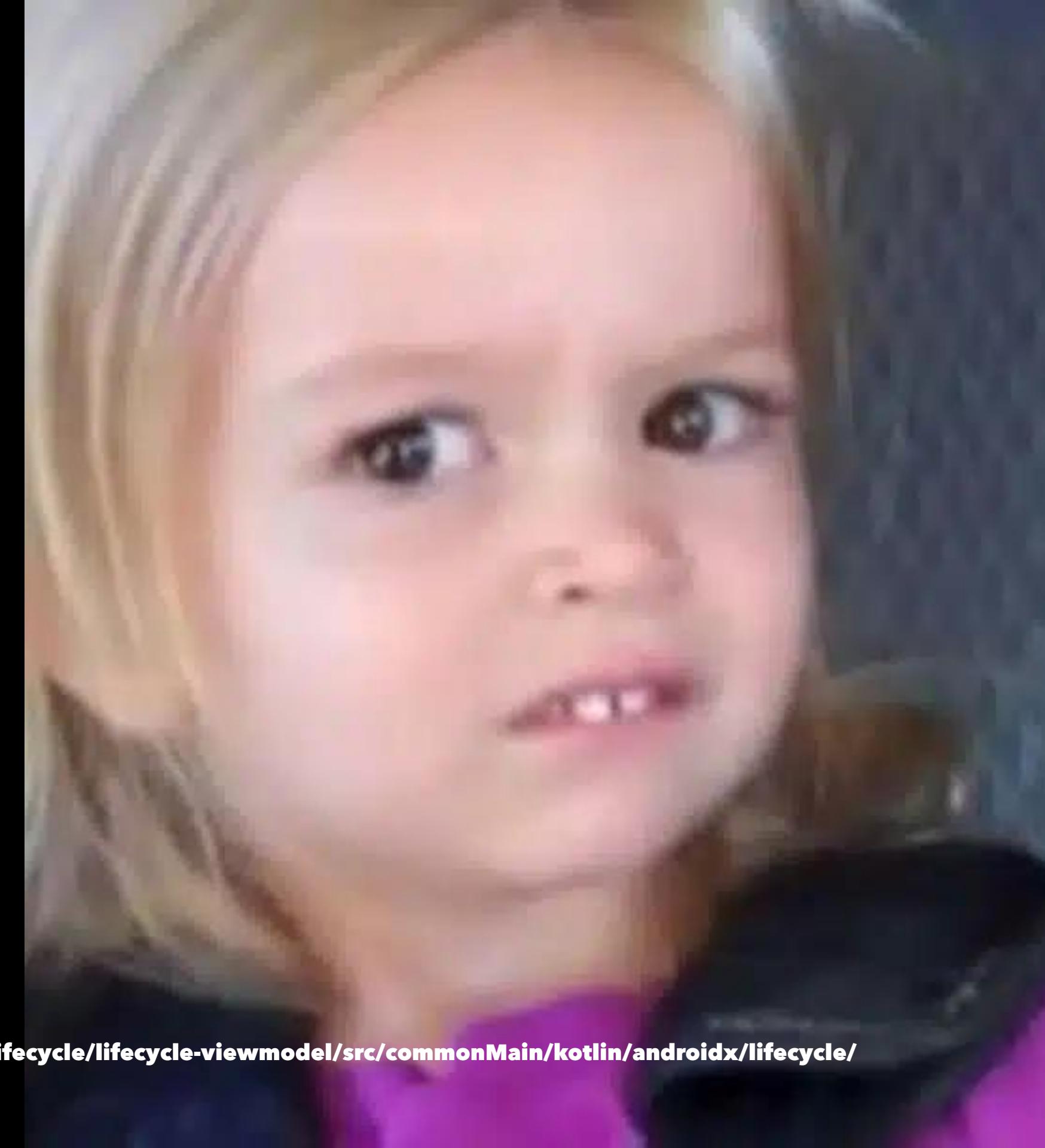




Maven Group ID	Latest Update	Stable Release	Alpha Release
annotation	04.09.2024	1.8.2	1.9.0-alpha03
collection	04.09.2024	1.4.3	1.5.0-alpha01
datastore	01.05.2024	1.1.1	-
lifecycle	04.09.2024	2.8.5	2.9.0-alpha02
paging	07.08.2024	3.3.2	-
room	21.08.2024	2.6.1	2.7.0-alpha07
sqlite	21.08.2024	2.4.0	2.5.0-alpha07

```
kotlin {  
    sourceSets.commonMain.dependencies {  
        implementation(  
            "androidx.lifecycle:" +  
            "lifecycle-viewmodel-ktx:" +  
            "2.8.5"  
        )  
    }  
}
```

```
// Backed by ViewModelImpl  
public expect abstract class ViewModel
```



Multiplatform Architecture

Decompose & Essenty

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
    }
}
```

```
class RootComponent(context: ComponentContext) : Root, ComponentContext {
    private val navigation = StackNavigation<Config>()
    override val childStack = childStack(/* ... */)

    fun createChild(config: Config, context: ComponentContext): Child = when (config) {
        is Config.List -> Child.List(ItemListComponent(context)) {
            navigation.push(Config.Details(itemId = it)) }
        )
        is Config.Details -> /* ... */
    }
}

private sealed class Config : Parcelable {
    @Parcelize object List : Config()
    @Parcelize data class Details(val itemId: Long) : Config()
}
```

Decompose

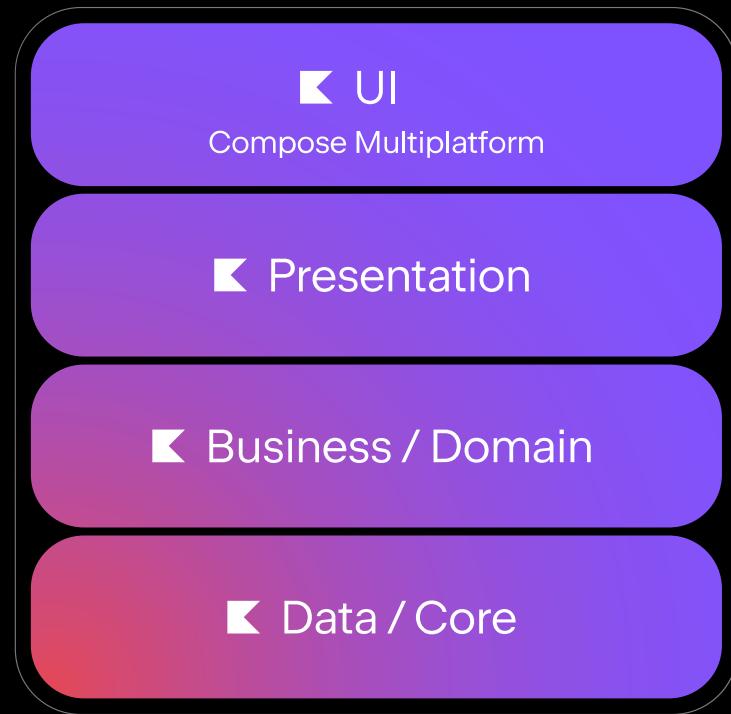
- com.arkivanov.decompose:extensions-compose
- com.arkivanov.decompose:extensions-android
- com.arkivanov.essenty:state-keeper

Compose Multiplatform



Compose Multiplatform

v1.0 | 2021



Platform	Stability level
Android	Stable
iOS	Stable
Desktop (JVM)	Stable
Server-side (JVM)	Stable
Web based on Kotlin/Wasm	Alpha
Web based on Kotlin/JS	Stable
watchOS	Best effort
tvOS	Best effort

Compose Multiplatform

1.6.11

1.6.10

1.6.2

...

Jetpack Compose

1.6.7

1.6.7

1.6.4

GitHub - JetBrains/compose- x +

github.com/JetBrains/compose-multiplatform-core

Product Solutions Resources Open Source Enterprise Pricing Sign in Sign up

JetBrains / compose-multiplatform-core Public forked from androidx/androidx

Notifications Fork 72 Star 427

Code Pull requests 34 Actions Projects Security Insights

jb-main ▾ 659 Branches 663 Tags Go to file Code ▾

This branch is 3993 commits ahead of, 13404 commits behind androidx/androidx:androidx-main .

ASalavei Fix keyboard closing while scrollin... 8619c0a · 2 days ago 182,738 Commits

.github Disable iOS PR checks with GitHub Acti... 2 months ago

.idea CodeStyle. Use single name import (#6... last year

.run Fix running tests via IDEA configuration ... 3 months ago

activity Fix wrongly committed files because of t... 8 months ago

annotation Snap annotation-1.8.0-alpha02 6 months ago

appactions Fix wrongly committed files because of t... 8 months ago

appcompat Fix wrongly committed files because of t... 8 months ago

About

Development environment for Android Jetpack extension libraries under the androidx namespace. Synchronized with Android Jetpack's primary development branch on AOSP.

[android.googlesource.com/platform/fr...](#)

Readme Apache-2.0 license Activity Custom properties 427 stars 17 watching 72 forks Report repository

```
kotlin {
    sourceSets.commonMain.dependencies {
        implementation("org.jetbrains.androidx.navigation:navigation-compose:2.8.0-alpha10")
    }
}

@Serializable
data object HomeRoute

NavHost(navController, HomeRoute) {
    composable<HomeRoute> {
        HomeScreen()
    }
}

val route = savedStateHandle.toRoute<HomeRoute>()
```

The screenshot shows a web browser window with the following details:

- Title Bar:** Navigation and routing | Kotlin
- Address Bar:** jetbrains.com/help/kotlin-multiplatform-dev/compose-navigation-routing.html#setup
- Page Header:** JET BRAINS Kotlin Multiplatform Development
- Left Sidebar (Navigation):**
 - Get started
 - Kotlin Multiplatform overview
 - What's new in Kotlin Multiplatform
 - Set up an environment
 - Create an app with shared logic and native UI
 - Create an app with shared logic and UI
 - Make your app multiplatform
 - Develop with Kotlin Multiplatform
 - Test your multiplatform app
 - Publish your application
 - Samples
 - Compose Multiplatform UI framework
 - Why Compose Multiplatform ↗
 - Multiplatform resources
 - Lifecycle
 - Common ViewModel
 - Navigation and routing
 - Drag-and-drop operations
 - Testing Compose Multiplatform UI
 - iOS-specific features
 - Android-specific components
 - Desktop-specific components
 - Compose Multiplatform for web ↗
 - Compose compiler
 - Compatibility and versions
 - Releases ↗
- Page Content:**
 - Section Header:** Navigation and routing
 - Text:** Compose Multiplatform UI framework / Navigation and routing
 - Right Sidebar (Navigation):**
 - Navigation and routing
 - Setup
 - Sample project
 - Limitations
 - Third-party alternatives
 - Alert Box:** The navigation library is currently Experimental. You're welcome to try it in your Compose Multiplatform projects. We would appreciate your feedback in YouTrack ↗.
 - Text:** Navigation is a key part of UI applications that allows users to move between different application screens. Compose Multiplatform adopts the Jetpack Compose approach to navigation ↗.
 - Section Header:** Setup
 - Text:** To use the navigation library, add the following dependency to your commonMain source set:
 - Code Block:**

```
kotlin {  
    // ...  
    sourceSets {  
        // ...  
        commonMain.dependencies {  
            // ...  
            implementation("org.jetbrains.androidx.navigation:navigation-  
        }  
        // ...  
    }  
}
```

Compose Multiplatform Migration

- Change artifact coordinates
- Do nothing
- Profit

*The early bird gets the
worm ... but the second
mouse gets the cheese*

Reactive Architecture

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

Architecture

Callbacks

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

Architecture Observables

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

Architecture

Coroutines

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

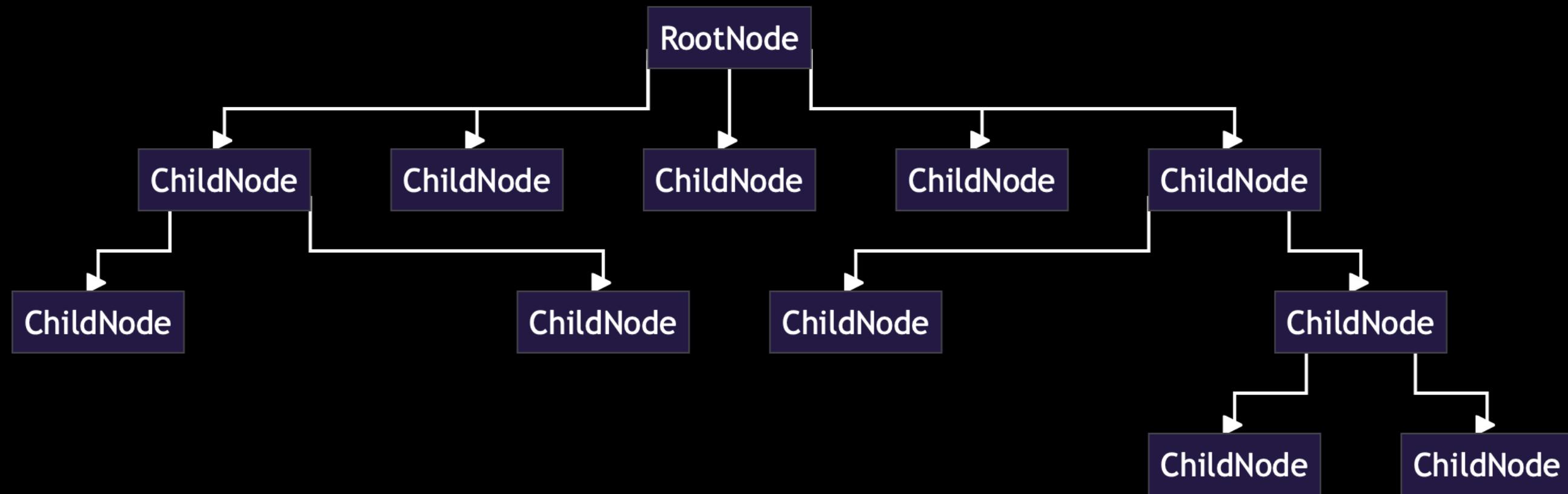
Architecture

Coroutines (Again)

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

Architecture

Compose



Architecture

Compose

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

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

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

cashapp/molecule



Build a StateFlow stream using Jetpack Compose

cashapp/molecule

24
Contributors

22
Issues

15
Discussions

2k
Stars

80
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
    }
}
```

Demystifying Molecule

Droidcon NYC 2022

ashdavies.dev/talks/demystifying-molecule-nyc

```
@SuppressLint("DEPRECATION")
class CallbackLoginPresenter(
    private val service: SessionService,
    private val goTo: (Screen) -> Unit,
    var onModel: (LoginUiModel) -> Unit = {},
    var task: AsyncTask<Submit, Void, LoginResult>? = null
) {
    ...
    fun stop() {
        task?.cancel(true)
    }
    fun onEvent(event: LoginUiEvent) {
        when (event) {
            is Submit -> task = LoginAsyncTask()
            .also { it.execute(event) }
        }
    }
    ...
}
```

Role of Architecture

Pre-Compose Era

slackhq/circuit

github.com/slackhq/circuit

Circuit

- Supports most supported KMP platforms
- Compose first architecture
- Presenter & UI separation
- Unidirectional Data Flow

Circuit State

```
@Parcelize
data object HomeScreen : Screen {
    data class State(
        val title: String,
    ): CircuitUiState
}
```

Circuit

Presenter

```
class HomePresenter : Presenter<HomeScreen.State> {  
    @Composable  
    override fun present(): HomeScreen.State {  
        return HomeScreen.State("Hello World")  
    }  
}
```

Circuit

UI

```
@Composable
fun HomeScreen(
    state: HomeScreen.State,
    modifier: Modifier = Modifier,
) {
    Text(
        text = state.title,
        modifier = modifier,
    )
}
```

Circuit

```
val circuit = Circuit.Builder()
    .addPresenter<HomeScreen, HomeScreen.State>(HomePresenter())
    .addUi<LauncherScreen, LauncherScreen.State> { _, _ -> HomeScreen(state, modifier) }
    .build()

CircuitCompositionLocals(circuit) {
    val backStack = rememberSaveableBackStack(HomeScreen)

    NavigableCircuitContent(
        navigator = rememberCircuitNavigator(backStack),
        backStack = backStack,
    )
}
```

Circuit

Navigation

```
@Parcelize
data object HomeScreen : Screen {
    data class State(
        val title: String,
        val eventSink: (Event) -> Unit
    ): CircuitUiState

    sealed interface Event {
        data class DetailClicked(
            val id: String,
        ): Event
    }
}
```

Circuit

Navigation

```
class HomePresenter(private val navigator: Navigator) : Presenter<HomeScreen.State> {  
  
    @Composable  
    override fun present(): HomeScreen.State {  
        return HomeScreen.State("Hello World") { event ->  
            when (event) {  
                is HomeScreen.Event.DetailClicked -> navigator.goTo(DetailScreen(event.id))  
            }  
        }  
    }  
}
```



KotlinConf '23

MODERN COMPOSE ARCHITECTURE WITH CIRCUIT

Zac Sweers

Kieran Elliott



CircuitX

- com.slack.circuit:circuitx-android
- com.slack.circuit:circuitx-effects
- com.slack.circuit:circuitx-gesture-navigation
- com.slack.circuit:circuitx-overlays

rememberRetained()

Circuit

Examples

- **Chris Banes: Tivi**

github.com/chrisbanes/tivi

- **Zac Sweers: CatchUp**

github.com/ZacSweers/CatchUp

- **Zac Sweers: FieldSpottr**

github.com/zacsweers/fieldspottr

- **Ash Davies: Playground**

github.com/ashdavies/playground.ashdavies.dev

Full Disclosure

Bias

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))
            }
        ) {
            // ...
        }
    }
}
```

Voyager



```
interface ParcelableScreen : Screen, Parcelable

// Compile
@Parcelize
data class Post(/*...*/) : Parcelable

@Parcelize
data class ValidScreen(
    val post: Post
) : ParcelableScreen {
    // ...
}

// Not compile
data class Post(/*...*/)

@Parcelize
data class ValidScreen(
    val post: Post
) : ParcelableScreen {
    // ...
}
```

appyx

bumble-tech.github.io/appyx

PreCompose

github.com/Tlaster/PreCompose

Comparison

	androidx	circuit	decompose	voyager	workflow
Multiplatform	✓	✓	✓	✓	✓
Compose 1st	✗	✓	✗	✓	✗
Documented*	✗	✓	✓	✓	✗
Ease-of-Use**	✗	✓	/	✓	✗
Opinionated***	✗	✓	✗	✗	✓

* Documentation exists, but is outdated or hard to find

** Subjective, how quick to get started

*** Additional API surface

Y Tho?

Good Code == Removable Code
Code your own obsolescence

Thank You!

Ash Davies - SumUp

Android / Kotlin GDE Berlin

ashdavies.dev

Don't Forget to Vote! 🇺🇸 🐴