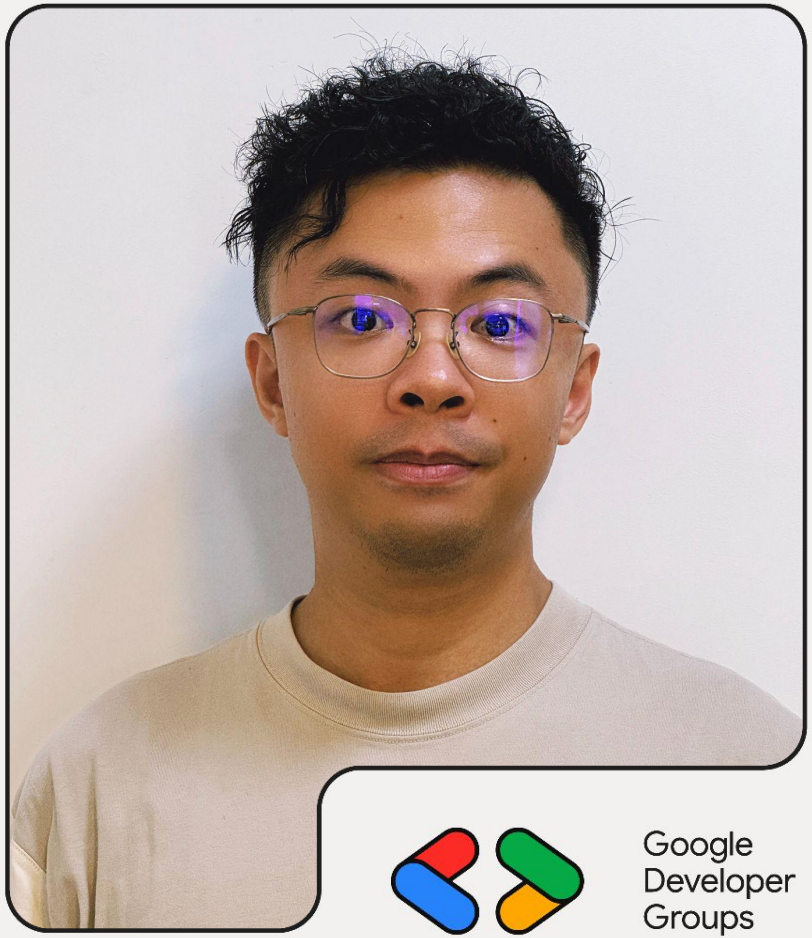
{ DevFest }

Taipei, 30th Nov

Adapting the MediaPipe Demos for Kotlin Multiplatform

EI

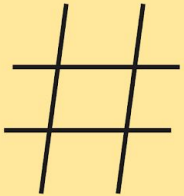


Google Developer Groups

1. On Device Model + KMP
2. 將 Android 專案遷移到 KMP
3. 換個角度看 KMP

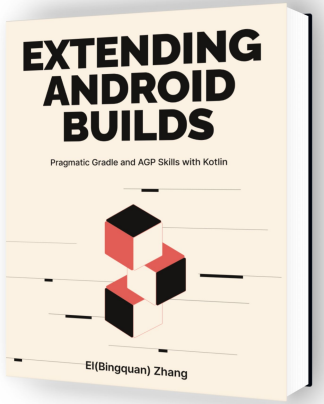
Google Developer Groups

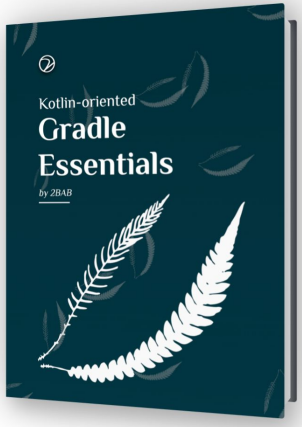
Adapting the MediaPipe Demos for Kotlin Multiplatform



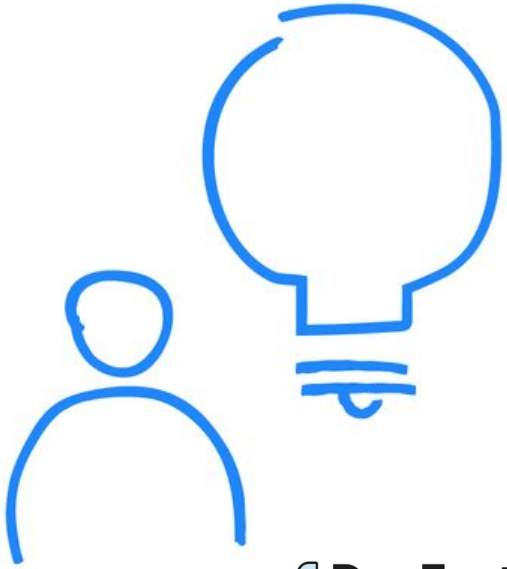
Mobile @DevFest

Experts
Android





Scan To Read

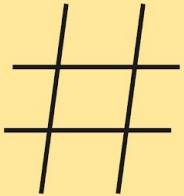


DevFest
Taipei

1. On Device Model + KMP
2. 將 Android 專案移植到 KMP
3. 換個角度看 KMP

Google Developer Groups

Adapting the MediaPipe Demos for Kotlin Multiplatform



Mobile @DevFest

Google x On-Device Model

（只考慮行動設備）

MediaPipe Solutions provides a suite of libraries to **quickly** apply **AI** and **machine learning (ML)** techniques in your applications.

https://mediapipe-studio.webapps.google.com/home

Solution	Android	Web	Python	iOS	Customize model
LLM Inference API	●	●		●	●
Object detection	●	●	●	●	●
Image classification	●	●	●	●	●
Image segmentation	●	●	●		
Interactive segmentation	●	●	●		
Hand landmark detection	●	●	●	●	
Gesture recognition	●	●	●	●	●
Image embedding	●	●	●		
Face detection	●	●	●	●	
Face landmark detection	●	●	●		
Face stylization	●	●	●		●
Pose landmark detection	●	●	●		
Image generation	●				●
Text classification	●	●	●	●	●
Text embedding	●	●	●		
Language detector	●	●	●		
Audio classification	●	●	●		

Google Developer Groups

Google x On-Device Model

（只考慮行動設備）

Model	支援框架	硬體要求	開放原始碼	實作穩定性
Gemma (1&2)	**MediaPipe**	較簡單 (但不是低)	Y	推出時間久，記憶體足夠時還算穩定
Gemini Nano	**AICore**	較複雜	N	才出 0.0.1-exp01，一個特殊符號可能就會導致無法輸出

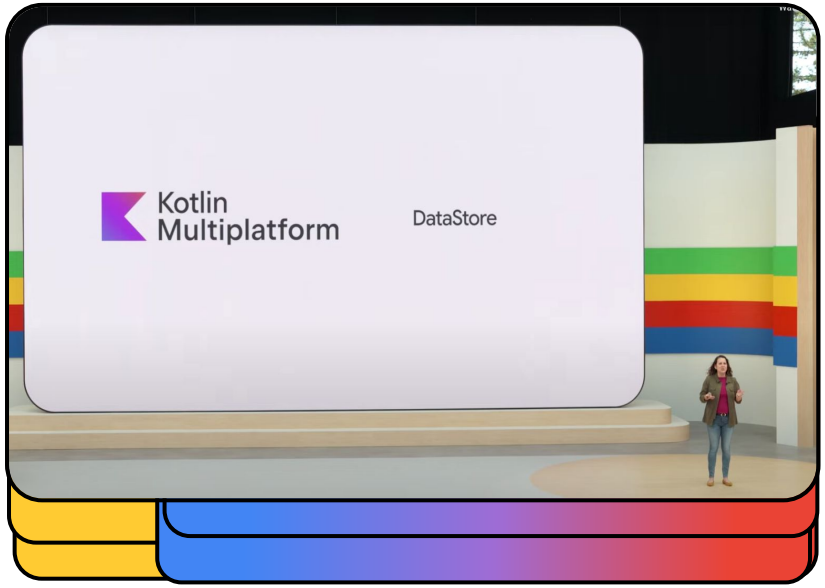
近似效能

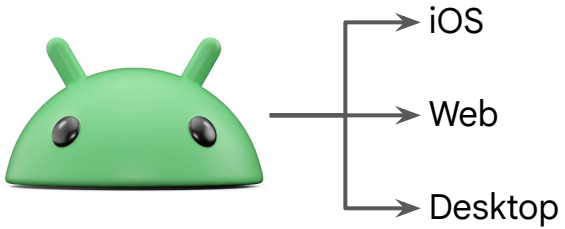
Kotlin Multiplatform on Android

First-class tooling and **library support** for Kotlin Multiplatform on Android.

Now expanded to your favourite libraries like **Room / DataStore**.

Leveraging KMP for business logic across Android, iOS and web. Strongly **enhancing** the **ecosystem** of KMP.





Google Developer Groups

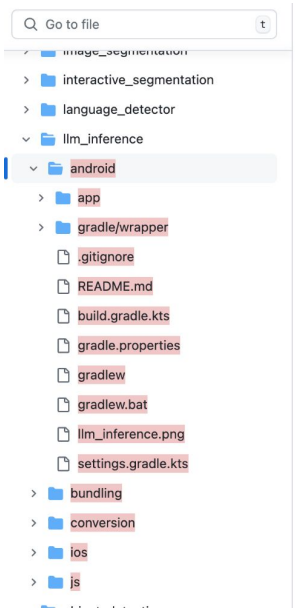
KMP Case Studies

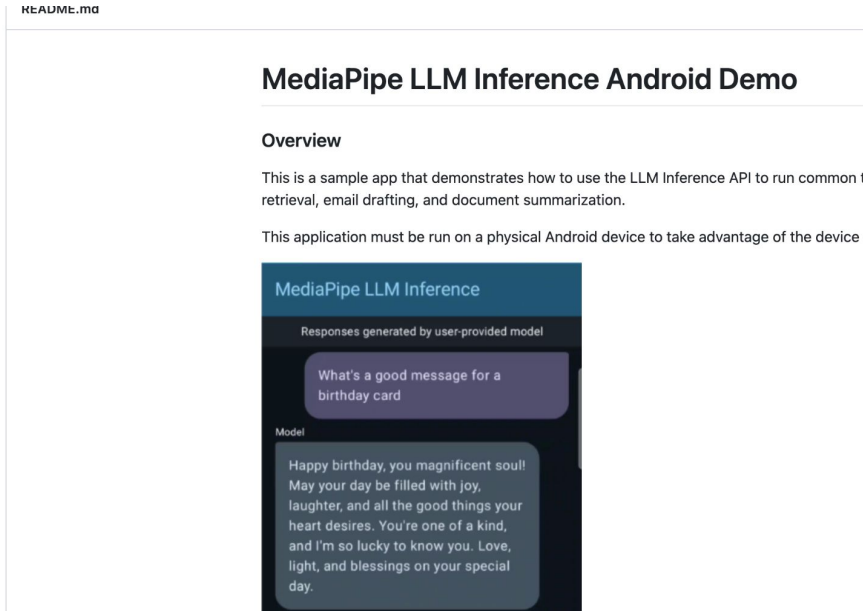
MediaPipe Demo

（以 LLM 為例）

1. **純 Kotlin**
2. UI 完全採用 **Jetpack Compose**
3. 所依賴的 LLM 任務 SDK **高度封裝**，僅暴露三個核心方法

這不已經達到
KMP 的標準?



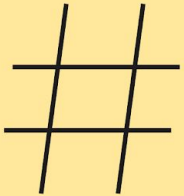


Google Developer Groups

1. On Device Model + KMP
2. 將 Android 專案移植到 KMP
3. 換個角度看 KMP

Google Developer Groups

Adapting the MediaPipe Demos for Kotlin Multiplatform

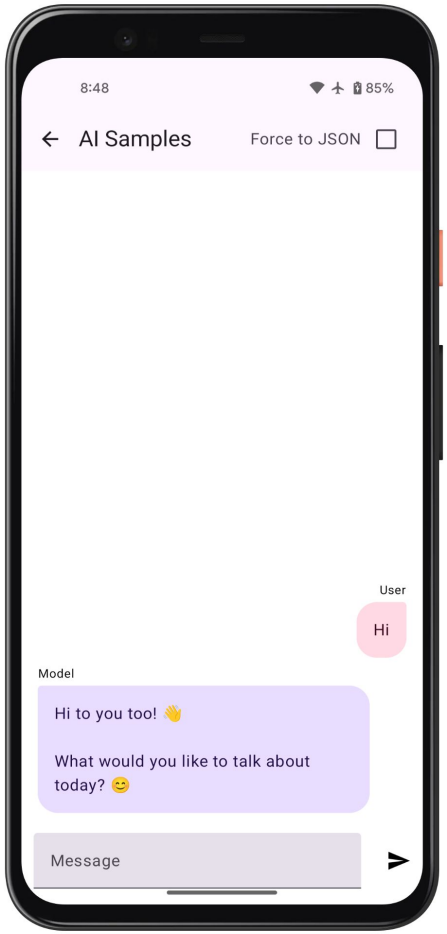


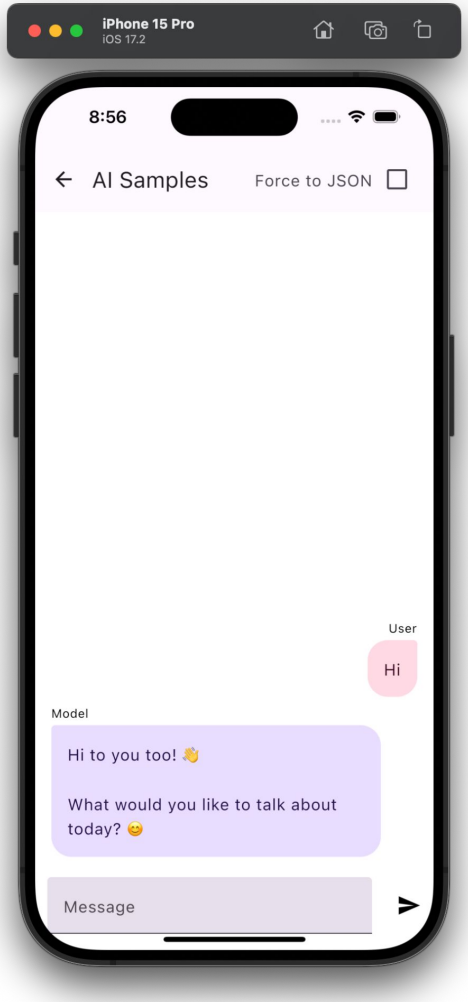
Mobile @DevFest

LLM Inference

Showing up a conversational chat interface with gemma2-2b models. The UI design and original codebase come from Google.

https://github.com/google-ai-edge/mediapipe-samples/tree/main/examples/llm_inference/android



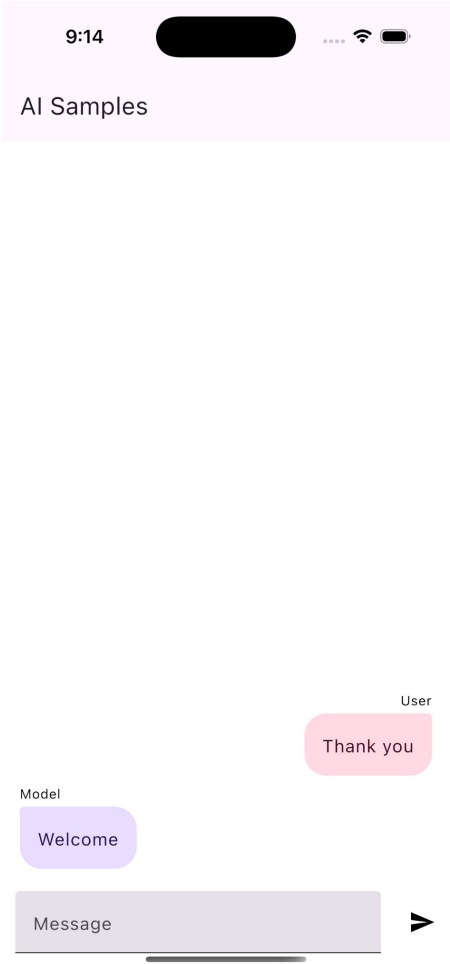


Google Developer Groups

UI Migration
Live Demo

遷移技巧第壹式

1. 大膽嘗試
2. 替換 Android 組件的 API 為 Voyage（當然如果用了新的ViewModel 和 Navigation，可能更簡單）
3. 替換 R 為 Res、以及 Kotlin 的 Log Message 等等



Google Developer Groups

Logic Migration

```kotlin
class LLMInferenceAndroidImpl(private val ctx: Context): LLMOperator {
    ...
    override suspend fun initModel(): String? {
        ...
        return try {
            ...
            loadModel(modelPath)
        } catch (e: Exception) {}
    }
    private fun loadModel(modelPath: String) {
        val options = LlmInference.LlmInferenceOptions.builder()
            ...
            .setResultListener { partialResult, done ->
                partialResultsFlow.tryEmit(partialResult to done)
            }
            .build()

        llmInference = LlmInference.createFromOptions(ctx, options)
    }
    override fun sizeInTokens(text: String): Int = llmInference.sizeInTokens(text)
    override suspend fun generateResponse(inputText: String): String {
        ...
        return llmInference.generateResponse(inputText)
    }
    override suspend fun generateResponseAsync(inputText: String): Flow<Pair<String, Boolean>> {
        ...
        llmInference.generateResponseAsync(inputText)
        return partialResultsFlow.asSharedFlow()
    }
}
```

LLM Task Engine for Android

```
plugins {
    ...
    alias(libs.plugins.cocoapods)
}
cocoapods {
    ...
    ios.deploymentTarget = "15"

    pod("MediaPipeTasksGenAIC") {
        version = "0.10.14"
        extraOpts += listOf("-compiler-option", "-fmodules")
    }
    pod("MediaPipeTasksGenAI") {
        version = "0.10.14"
        extraOpts += listOf("-compiler-option", "-fmodules")
    }
}

```

LLM Task Engine for iOS (Option 1)

LLM Task Engine for iOS (Option 1)

```
// Note that below imports are starting with cocoapods
import cocoapods.MediaPipeTasksGenAI.MPPLLMInference
import cocoapods.MediaPipeTasksGenAI.MPPLLMInferenceOptions
import platform.Foundation.NSBundle
...
class LLMOperatorIOSImpl: LLMOperator {

    private val inference: MPPLLMInference

    init {
        val modelPath = NSBundle.mainBundle.pathForResource(..., "bin")

        val options = MPPLLMInferenceOptions(modelPath!!)
        options.setModelPath(modelPath!!)
        options.setMaxTokens(2048)
        ...
        // NPE was thrown here right after it printed the success initialization message internally.
        inference = MPPLLMInference(options, null)
    }

    override fun generateResponse(inputText: String): String {...}
    override fun generateResponseAsync(inputText: String, ...) :... {
        ...
    }
    ...
}
```

可能是因為 Interop 目前的支援仍有限，對於 Swift 程式碼中帶有 throws 關鍵字的 init 方法转换有些问题 (KT 1.9/2.0)。

```kotlin
// Add one more interface for Swift
// app/src/iosMain/.../llm/LLMOperator.kt
interface LLMOperatorSwift {
    suspend fun loadModel(modelName: String)
    fun sizeInTokens(text: String): Int
    suspend fun generateResponse(inputText: String): String
    suspend fun generateResponseAsync(
        inputText: String,
        progress: (partialResponse: String) → Unit,
        completion: (completeResponse: String) → Unit
    )
}
```

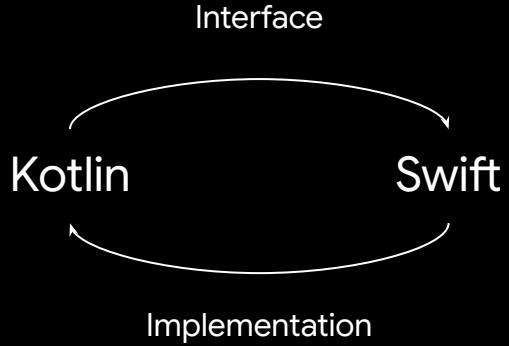
LLM Task Engine for iOS (Option 2)

LLM Task Engine for iOS (Option 2)

```swift
// iosApp/iosApp/LLMInferenceDelegate.swift
class LLMOperatorSwiftImpl: LLMOperatorSwift {
    ...
    var llmInference: LlmInference?
    func loadModel(modelName: String) async throws {
        let path = Bundle.main.path(forResource: modelName, ofType: "bin")!
        let llmOptions =  LlmInference.Options(modelPath: path)
        ...// options parameters setup
        llmInference = try LlmInference(options: llmOptions)
    }
    func generateResponse(inputText: String) async throws -> String {
        return try llmInference!.generateResponse(inputText: inputText)
    }
    func generateResponseAsync(inputText: String, progress: @escaping (String) -> Void, completion: @escaping (String) -
> Void) async throws {
        try llmInference!.generateResponseAsync(inputText: inputText) { partialResponse, error in
            // progress
            if let e = error {
                completion(e.localizedDescription)
                return
            }
            if let partial = partialResponse {
                progress(partial)
            }
        } completion: {completion("")}
    }
}
```

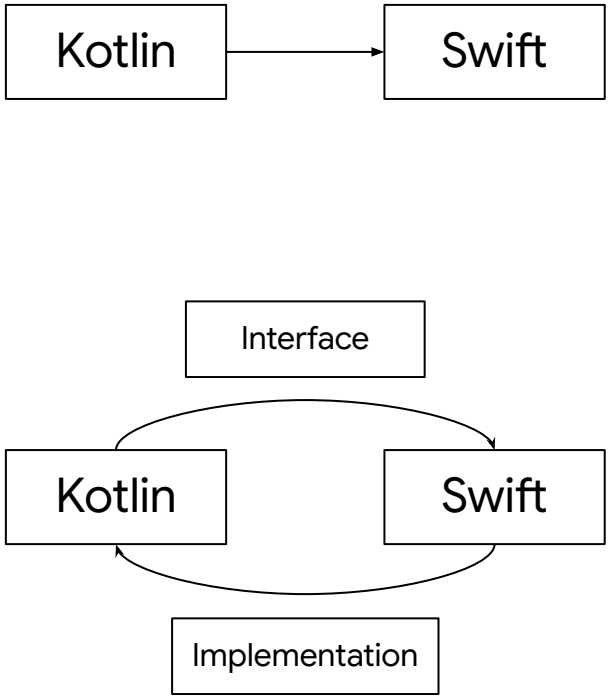
```swift
// iosApp/iosApp/iosApp.swift
class AppDelegate: UIResponder, UIApplicationDelegate {
    ...
    func application() {
        ...
        let delegate = try LLMOperatorSwiftImpl()
        MainKt.onStartup(llmInferenceDelegate: delegate)
    }
}

// LLMOperatorIOSImpl.kt
class LLMOperatorIOSImpl(
   private val delegate: LLMOperatorSwift) : LLMOperator {
    ...
}
```

LLM Task Engine for iOS (Option 2)



遷移技巧第貳式

1. Kotlin 可以直接呼叫透過CocoaPods 引入的第三方函式庫。
2. Kotlin 也能透過 iOS 專案呼叫第三方函式庫。
3. ObjC Export 有升級空間, Swift Export 更是值得期待。

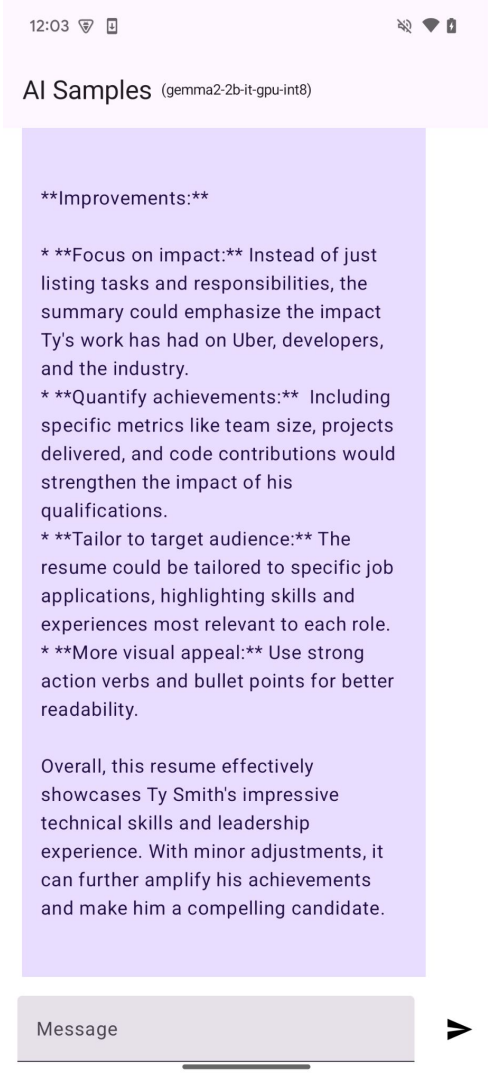


遷移技巧第貳式

1. Kotlin 可以直接呼叫透過CocoaPods 引入的第三方函式庫。
2. Kotlin 也能透過 iOS 專案呼叫第三方函式庫。
3. ObjC Export 有升級空間, Swift Export 更是值得期待。

小彩蛋:

Mediapipe 已支援 Gemma2-2B, 你可以讓它寫封郵件, 改個簡報備忘, 或者分析一份履歷。



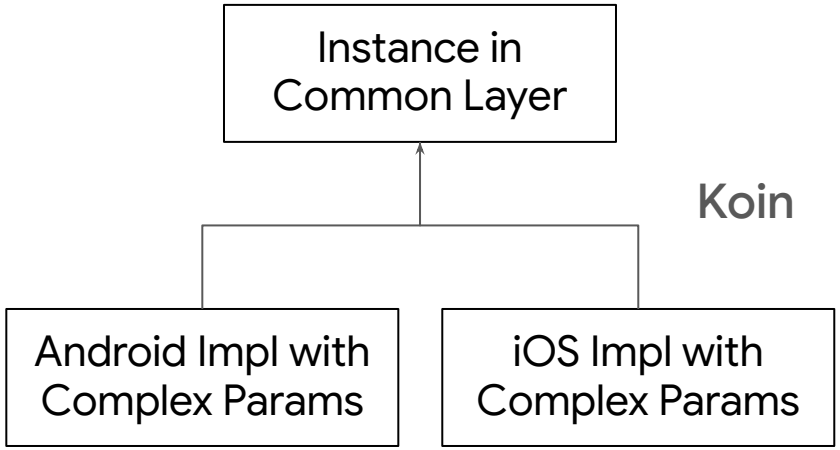
What about the passing procedure?

Multiplatform DI With Koin

```kotlin
// Common
expect class LLMOperatorFactory {
    fun create(): LLMOperator
}
val sharedModule = module {
   // Creating LLMOperator for Common Layer from different LLMOperatorFactory
  single<LLMOperator> { get<LLMOperatorFactory>().create() }
}

// Android
actual class LLMOperatorFactory(private val context: Context){
    actual fun create(): LLMOperator = LLMInferenceAndroidImpl(context)
}
val androidModule = module {
    // Inject Application Context for Android
    single { LLMOperatorFactory(androidContext()) }
}

// iOS
actual class LLMOperatorFactory(private val llmInferenceDelegate: LLMOperatorSwift) {
    actual fun create(): LLMOperator = LLMOperatorIOSImpl(llmInferenceDelegate)
}
module {
    // Inject the delegate from `onStartup` method for iOS
    single { LLMOperatorFactory(llmInferenceDelegate) }
}
```

遷移技巧第叁式

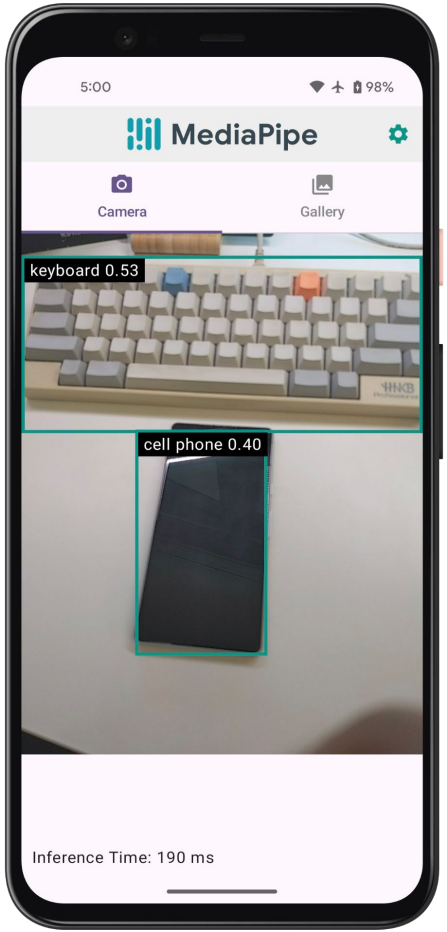
1. 當兩端的 Impl 實例需要不同的建構子參數時，通常會使用 KMP 的 expect 與 actual 關鍵字來解決此類需求。
2. 利用 expect 類別不需要聲明建構子參數的特性，增加了一層封裝。
3. 透過 Koin，為各平台注入所需的參數(很多時候還要增加額外的計算邏輯)，並統一將建立好的介面實作注入到Common 層的使用場景中。



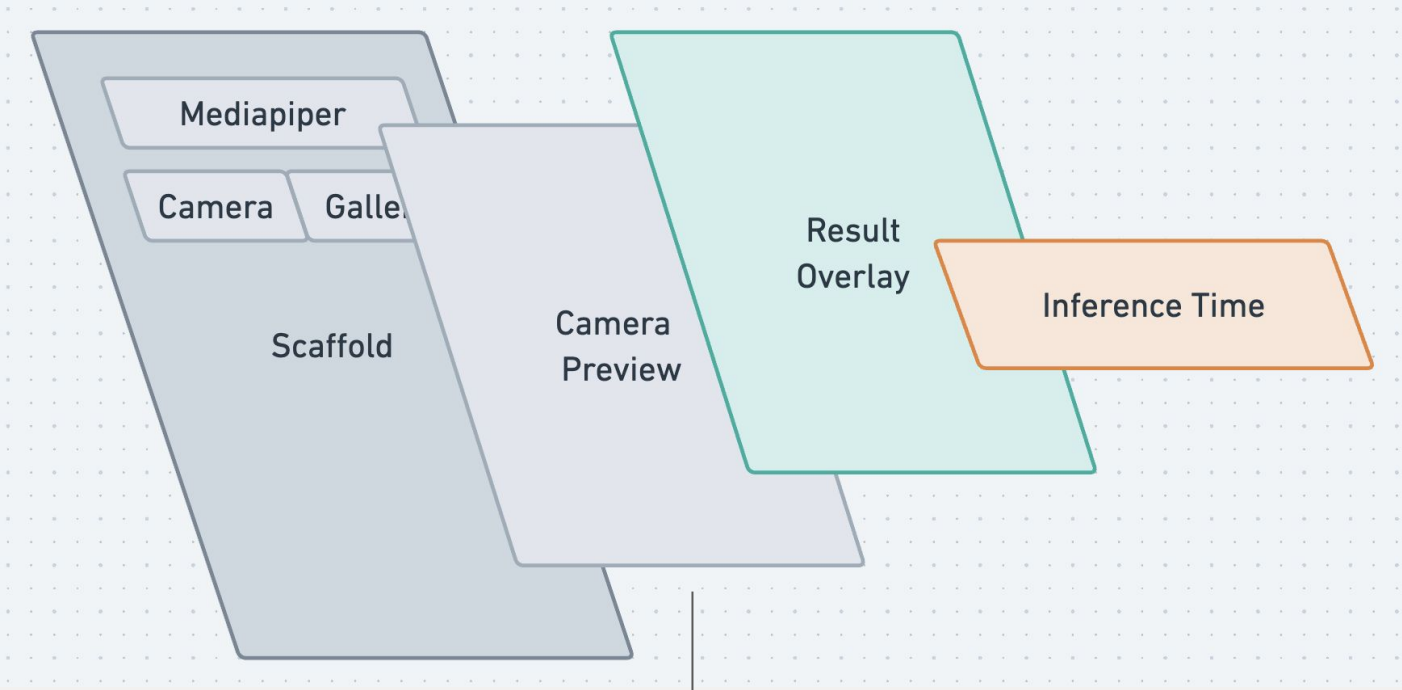
Object Detection

(物件偵測) Showing up a live object detection with camera preview. The UI design and original codebase come from Google.

https://github.com/google-ai-edge/mediapipe-samples/tree/main/examples/object_detection/android-jetpack-compose

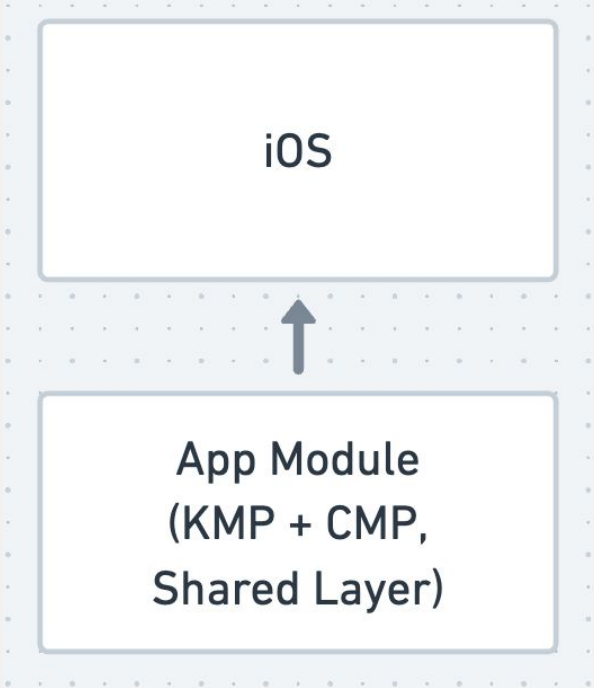


Google Developer Groups



Detection SDK(ML)

1. 原生功能的多端實現 + CMP



2. Kotlin 如何呼叫上層的 iOS 專案？

Compose Multiplatform + Native View

```
@Composable
fun CameraView(
    threshold: Float,
    maxResults: Int,
    delegate: Int,
    mlModel: Int,
    setInferenceTime: (newInferenceTime: Int) → Unit,
) {
    CameraPermissionControl {
        CameraPreview(
            threshold,
            maxResults,
            delegate,
            mlModel,
            setInferenceTime,
            onDetectionResultUpdate = { detectionResults →
                ...
            })
    }
}

@Composable
expect fun CameraPermissionControl(PermissionGrantedContent:  @Composable @UiComposable () → Unit)

@Composable
expect fun CameraPreview(...)
```

```kotlin
import platform.AVFoundation.*

@Composable
actual fun CameraPermissionControl(PermissionGrantedContent:  @Composable @UiComposable () -> Unit) {
    var hasCameraPermission by remember { mutableStateOf<Boolean?>(null) }
    LaunchedEffect(Unit) {
        hasCameraPermission = requestCameraAccess()
    }
    when (hasCameraPermission) {
        true -> { PermissionGrantedContent() }
        false -> { Text("...")}
        null -> { Text("Requesting camera permission...") }
    }
}

private suspend fun requestCameraAccess(): Boolean = suspendCoroutine { continuation ->
    val authorizationStatus = AVCaptureDevice.authorizationStatusForMediaType(AVMediaTypeVideo)
    when (authorizationStatus) {
        AVAuthorizationStatusNotDetermined -> {
            AVCaptureDevice.requestAccessForMediaType(AVMediaTypeVideo) { granted ->
                continuation.resume(granted)
            }
        }
        AVAuthorizationStatusRestricted, AVAuthorizationStatusDenied -> {
            continuation.resume(false)
        }
        AVAuthorizationStatusAuthorized -> {
            continuation.resume(true)
        }
        else -> {
            continuation.resume(false)
        }
    }
}
```

iOS Permission Control in Kotlin

```
typealias IOSCameraPreviewCreator = (
    threshold: Float,
    maxResults: Int,
    delegate: Int,
    mlModel: Int,
    setInferenceTime: (newInferenceTime: Int) -> Unit,
    callback: IOSCameraPreviewCallback
) -> UIView

typealias IOSCameraPreviewCallback = (result: ObjectDetectionResult) -> Unit

fun onStartup(iosCameraPreviewCreator: IOSCameraPreviewCreator) {
    Startup.run { koinApp ->
        koinApp.apply {
            modules(module {
                single { LLMOperatorFactory() }
                single<IOSCameraPreviewCreator> { iosCameraPreviewCreator }
            })
        }
    }
}
```

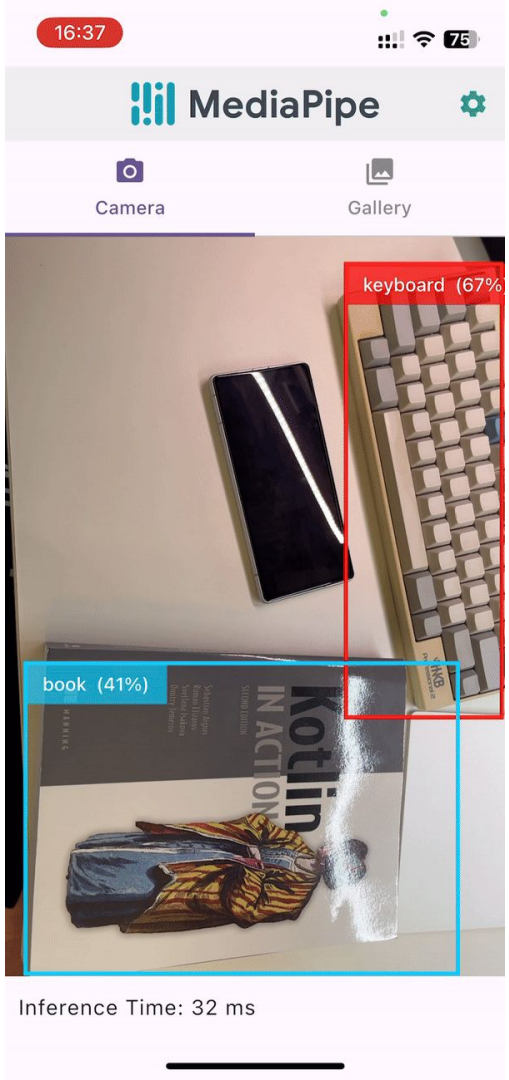
Pass Back

```kotlin
import androidx.compose.ui.viewinterop.UIKitView
import platform.UIKit.UIView

@Composable
actual fun CameraPreview(
    ...
) {
    val iOSCameraPreviewCreator = koinInject<IOSCameraPreviewCreator>()
    UIKitView(
        factory = {
            val iosCameraPreview: UIView = iOSCameraPreviewCreator(
                threshold,
                maxResults,
                delegate,
                mlModel,
                setInferenceTime,
                onDetectionResultUpdate)
            iosCameraPreview
        },
        modifier = Modifier.fillMaxSize(),
        update = { _ -> }
    )
}
```

Pass Back

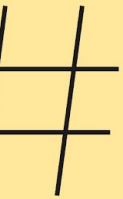
遷移技巧第肆式

1. 使用 Compose Multiplatform 嵌入 UIKit View，可以解決許多**高效能**需求，包括需要直接使用**複雜原生 API** 和**硬體資源**的情況。
2. 實際上，彩色方框(Overlay)若是在 CMP 編寫，可以擴展出更多可能。例如視訊通話時，結合人臉識別 Model，提供 Emoji 選擇與渲染。由於**邏輯**的高度**複雜性**，**重用**同一個 **StickerOverlay** 的**價值**變得非常顯著。



Google Developer Groups

1. On Device Model + KMP
2. 將 Android 專案移植到 KMP
3. 換個角度看 KMP



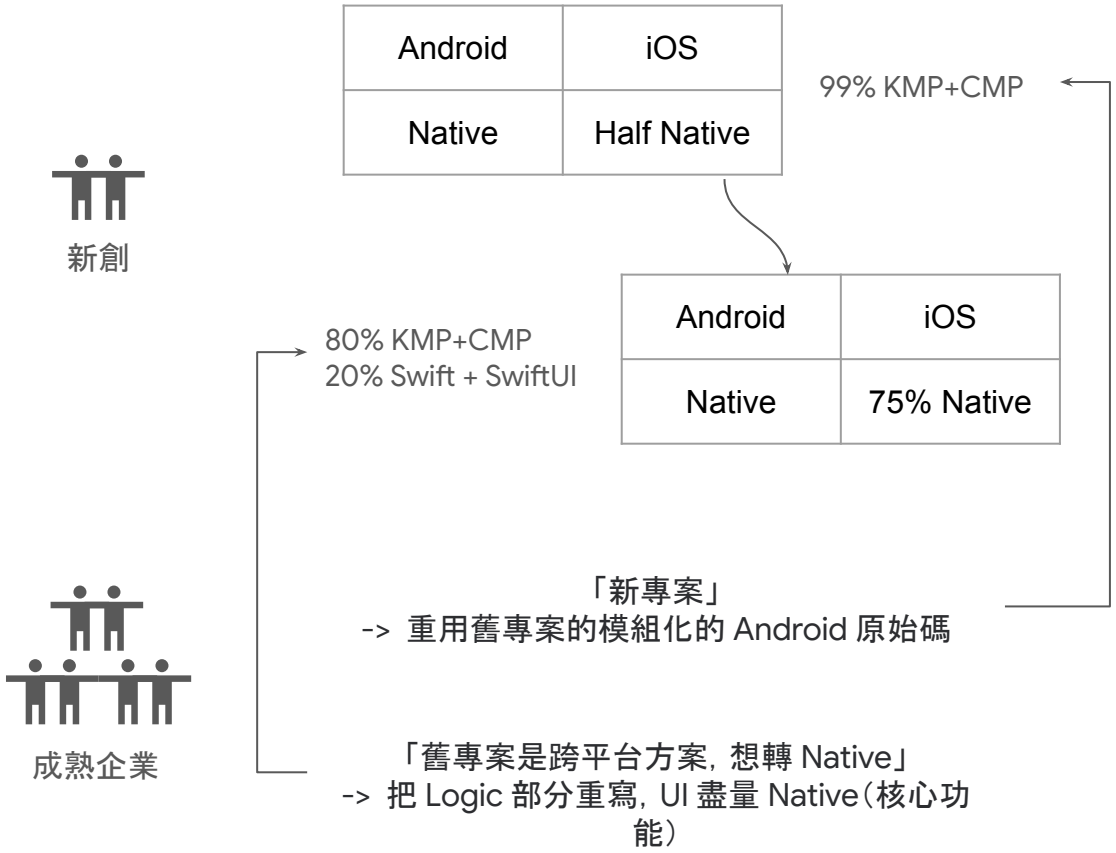
Google Developer Groups

Adapting the MediaPipe Demos for Kotlin Multiplatform

Mobile @DevFest

換個角度看 KMP

1. **可"遷移"**是個值得利用的條件。
2. 企業：重視**成本**。
3. Kotlin/Native **被视为 Native**，但 CMP 還未成熟。



Google Developer Groups

我只是在寫 Android 專案時稍加
改動，結果，iOS 也寫好了。

Why KMP?

Progressive Adoption

你永遠可以選擇只使用它來實現部分功能，並充分運用現有的 Android 程式碼。以 iOS 為例，基於 Kotlin 的 Android 應用程式只需簡單修改，即可輕鬆遷移到 iOS，無需重寫整個應用。

Native-level Perf

以 iOS 為例，非 UI 的常見場景都能達到接近 Swift 的效能水準。

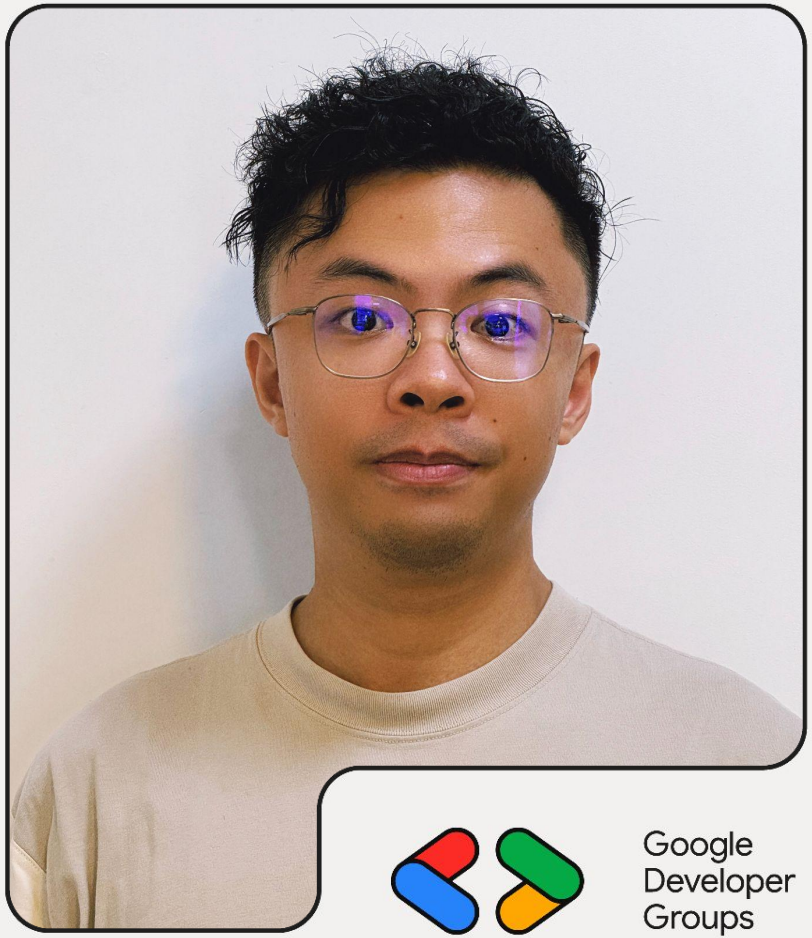
Fluent Interop

以 iOS 為例，ObjC-Interop 賦予 Kotlin 直接呼叫 iOS 相依項目的能力；反過來，Kotlin 編譯生成的 .framework 也能讓 Swift/ObjC 無縫地呼叫 Kotlin 所撰寫的功能。

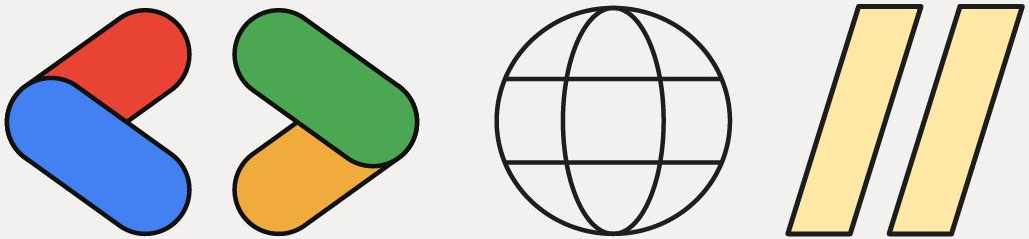
{ DevFest }

Taipei, 30th Nov

Thank You



Google Developer Groups



DevFest

Taipei

This is an example of a section title slide.

Google Developer Groups

Google Blue 500
#4285f4

Google Green 500
#34a853

Google Yellow 500
#fbbc04

Google Red 500
#ea4335

Halftone Blue
#57caff

Halftone Green
#5cdb6d

Halftone Yellow
#ffd427

Halftone Red
#ff7daf

Off-White
#f0f0f0

Pastel Blue
##c3ecf6

Pastel Green
#ccf6c5

Pastel Yellow
#ffe7a5

Pastel Red
#f8d8d8

Black 02
#1e1e1e

Headlines

Google Sans Bold

Subheads

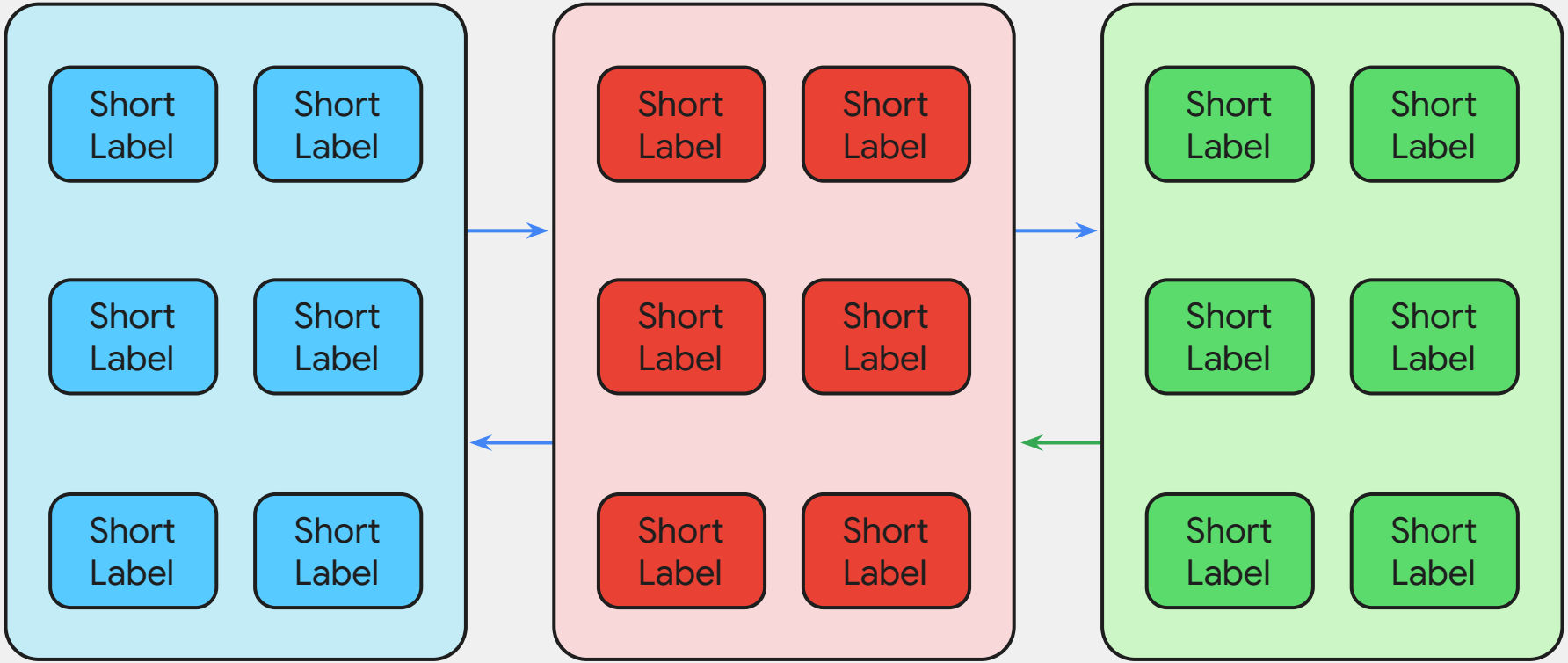
Roboto Mono Light

Body

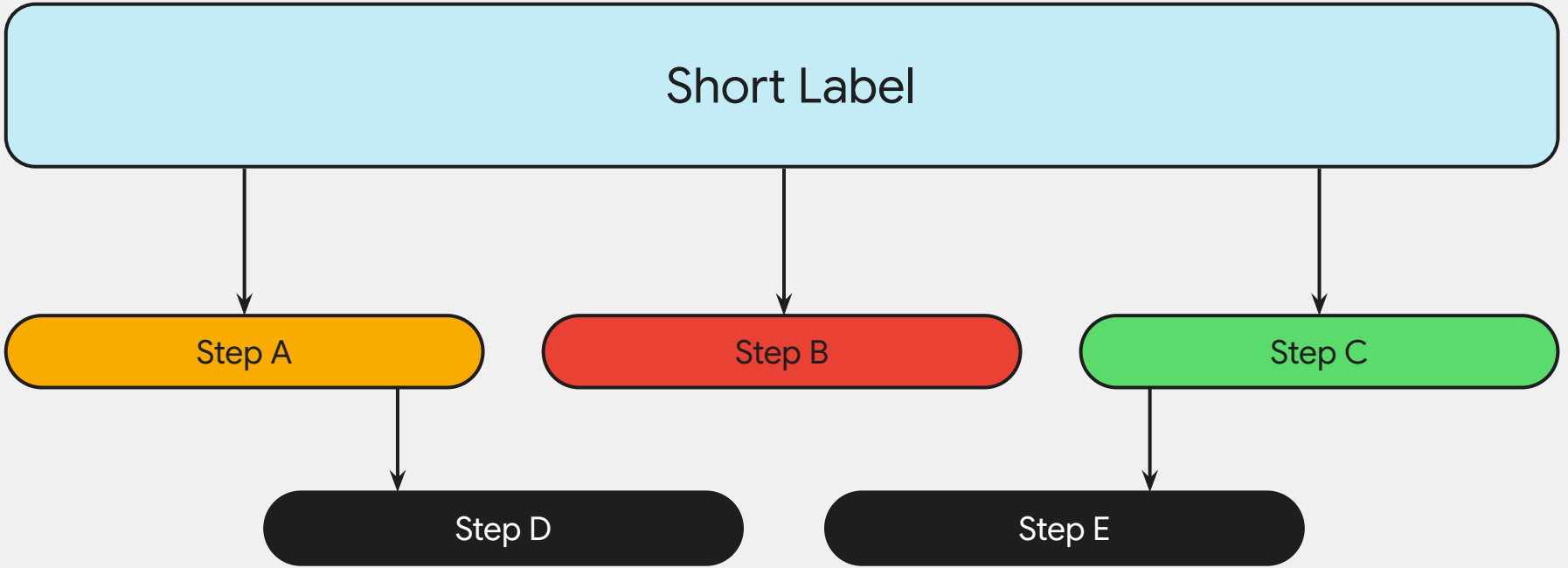
Google Sans Normal

Google Developer Groups

Simple Chart



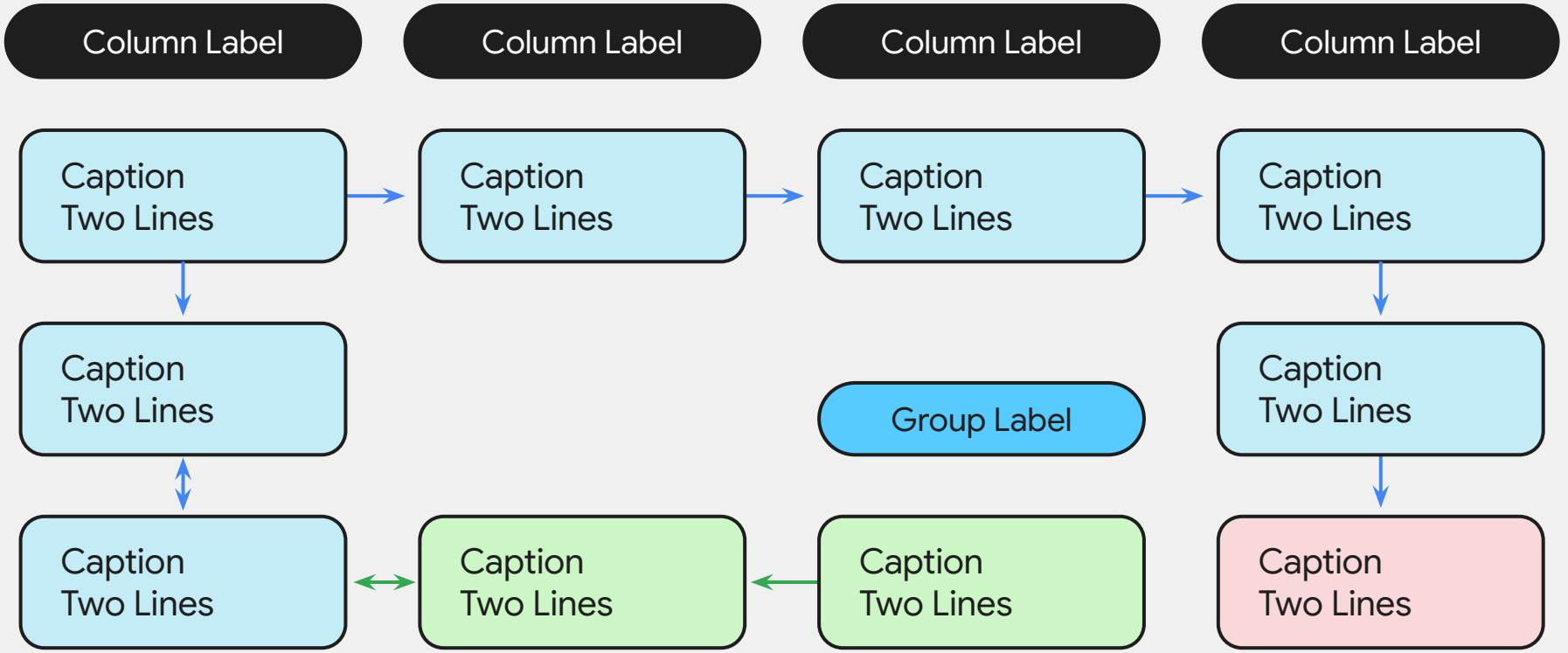
All-Purpose Chart



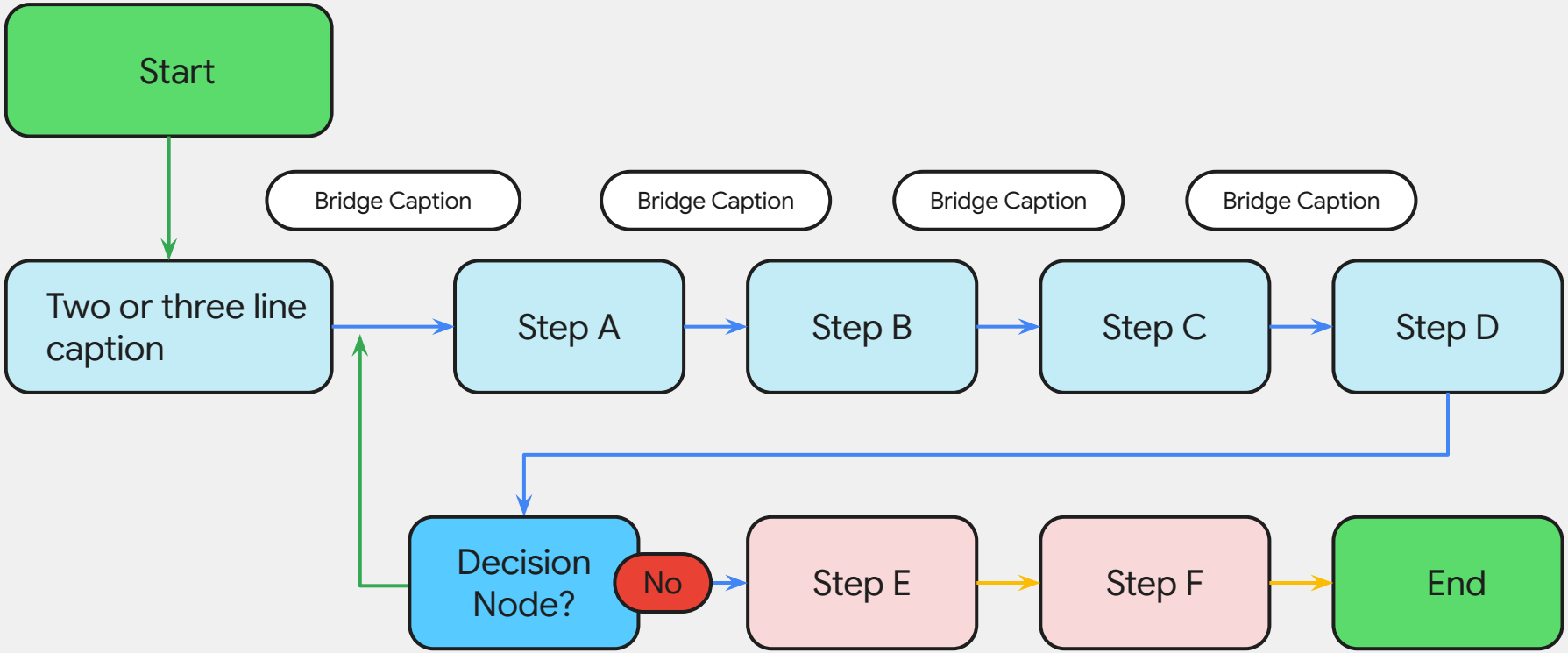
Big Portfolio Chart

Label One	Label Two	Label Three	Label Four
Short Label	Short Label	Short Label	Short Label
Short Label	Short Label	Short Label	Short Label
Short Label	Short Label	Short Label	Short Label

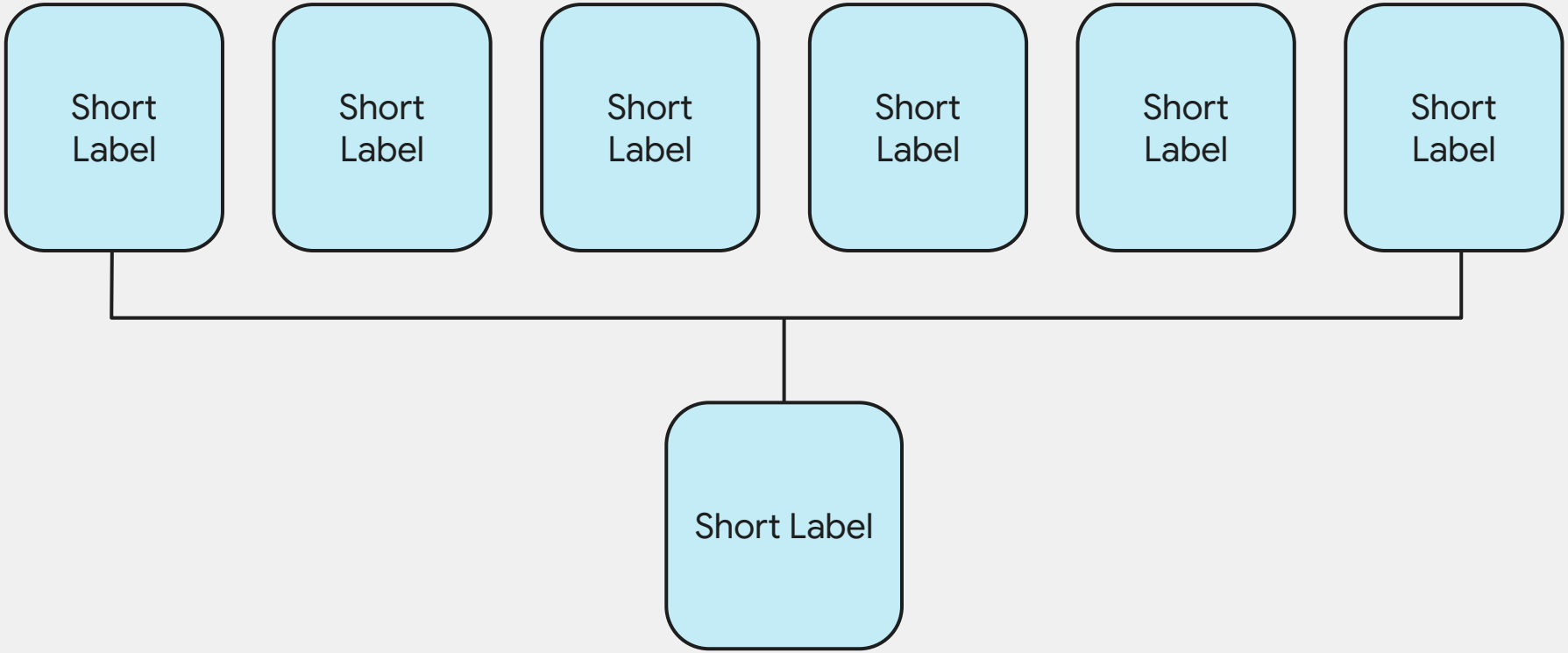
Process Chart



Flow-Style Chart



Group Chart


Short Label
Short Label
Short Label
Short Label
Short Label
Short Label
Short Label

Process Chart

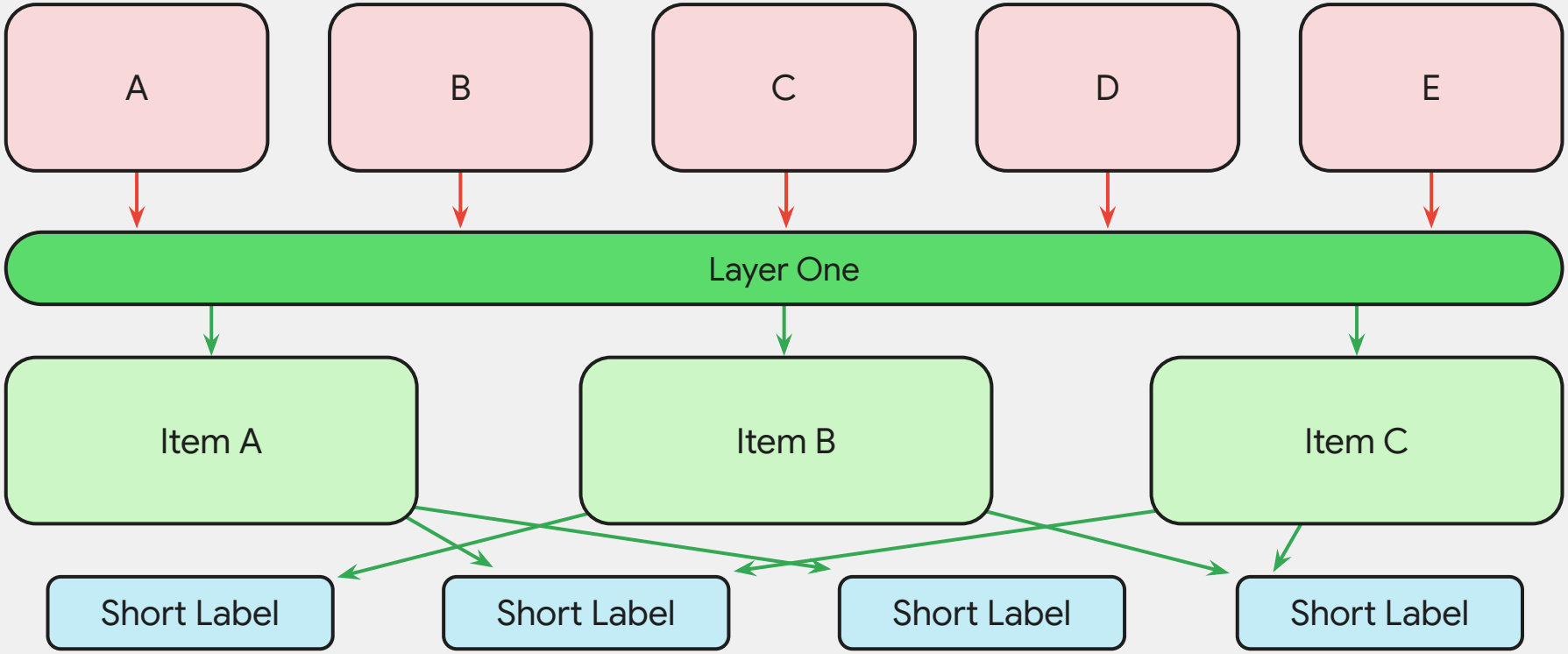
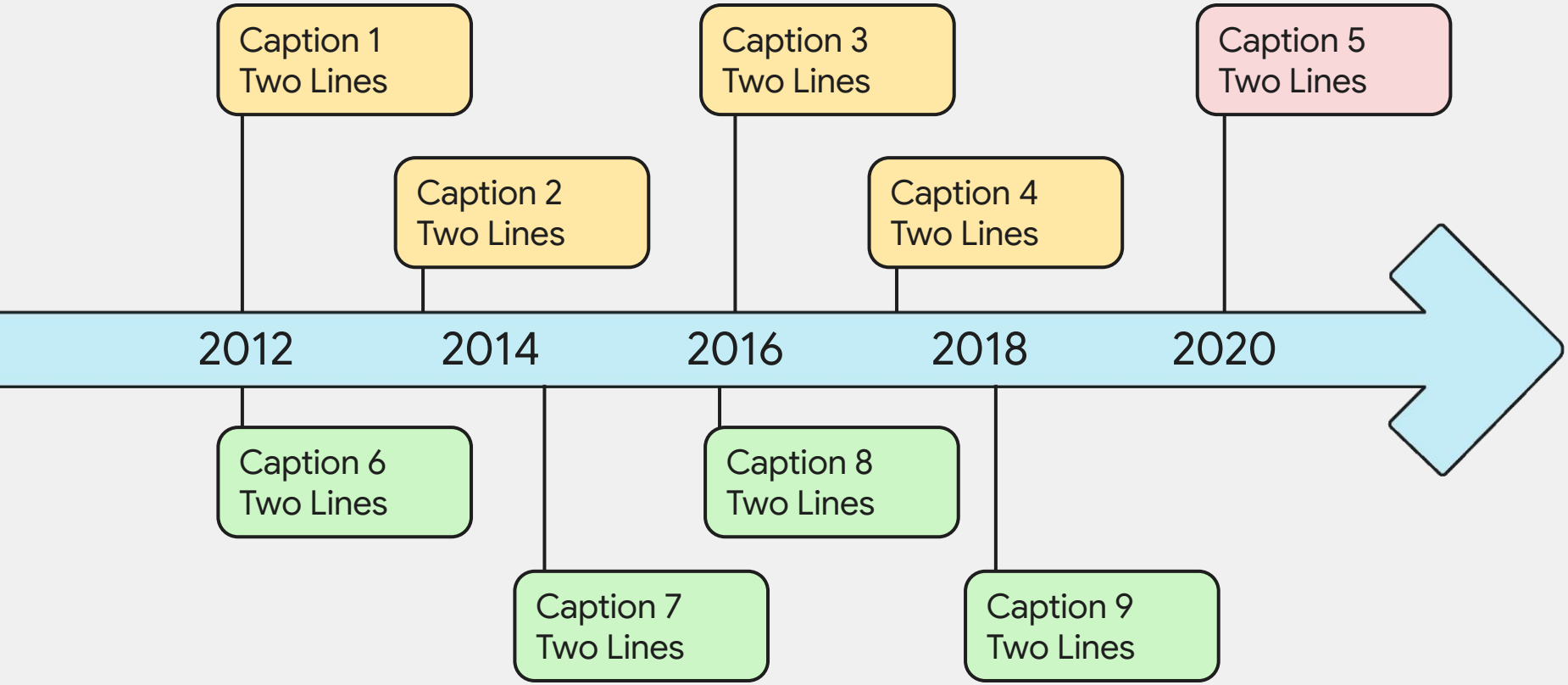


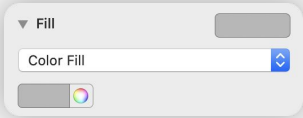
Chart Title

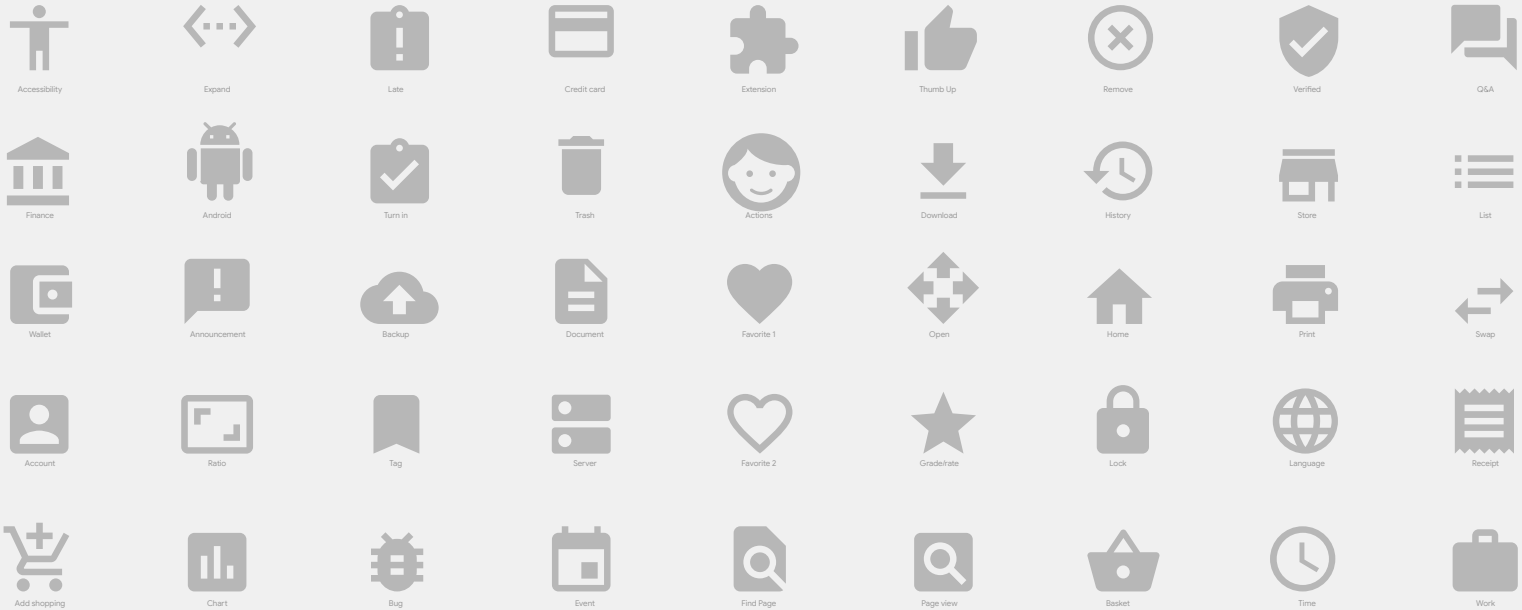

Caption 1
Two Lines
Caption 2
Two Lines
Caption 3
Two Lines
Caption 4
Two Lines
Caption 5
Two Lines
2012
2014
2016
2018
2020
Caption 6
Two Lines
Caption 7
Two Lines
Caption 8
Two Lines
Caption 9
Two Lines

Icons

Icons

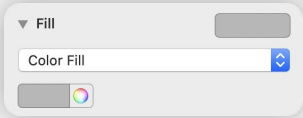
All icons are vector objects and can be recolored using the fill menu.

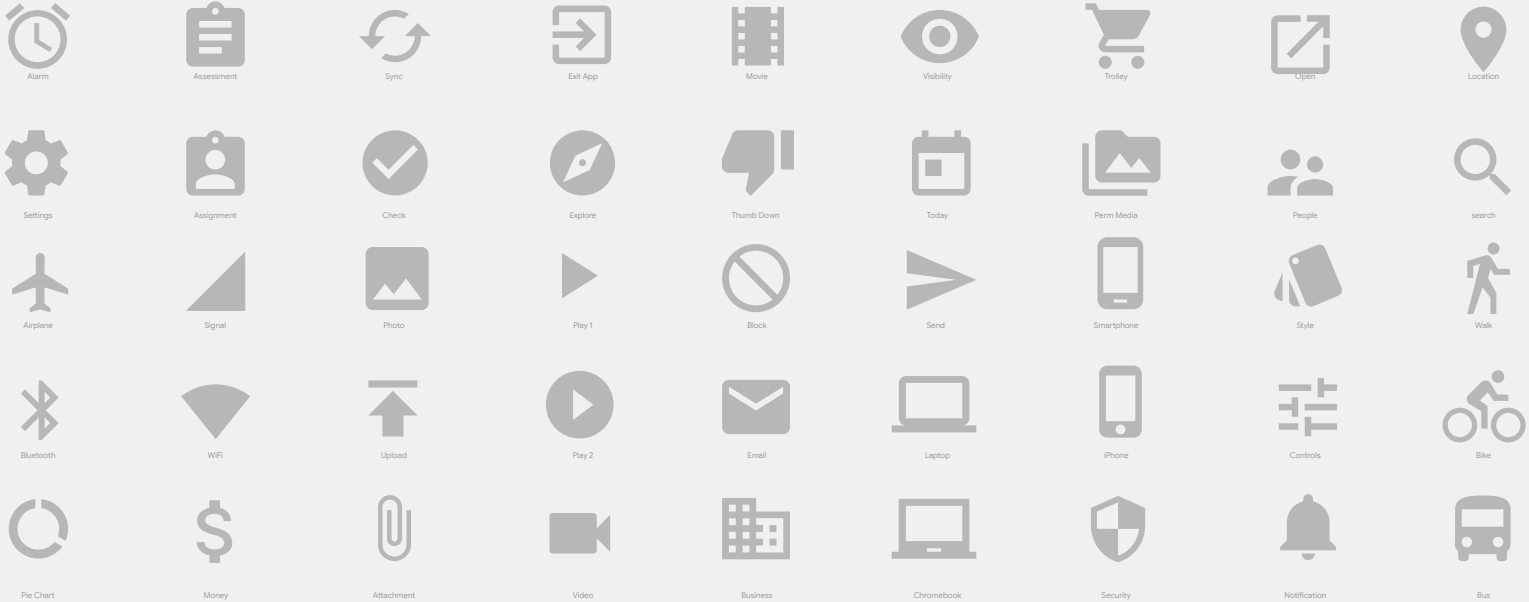




Icons

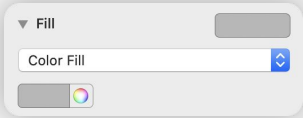
All icons are vector objects and can be recolored using the fill menu.

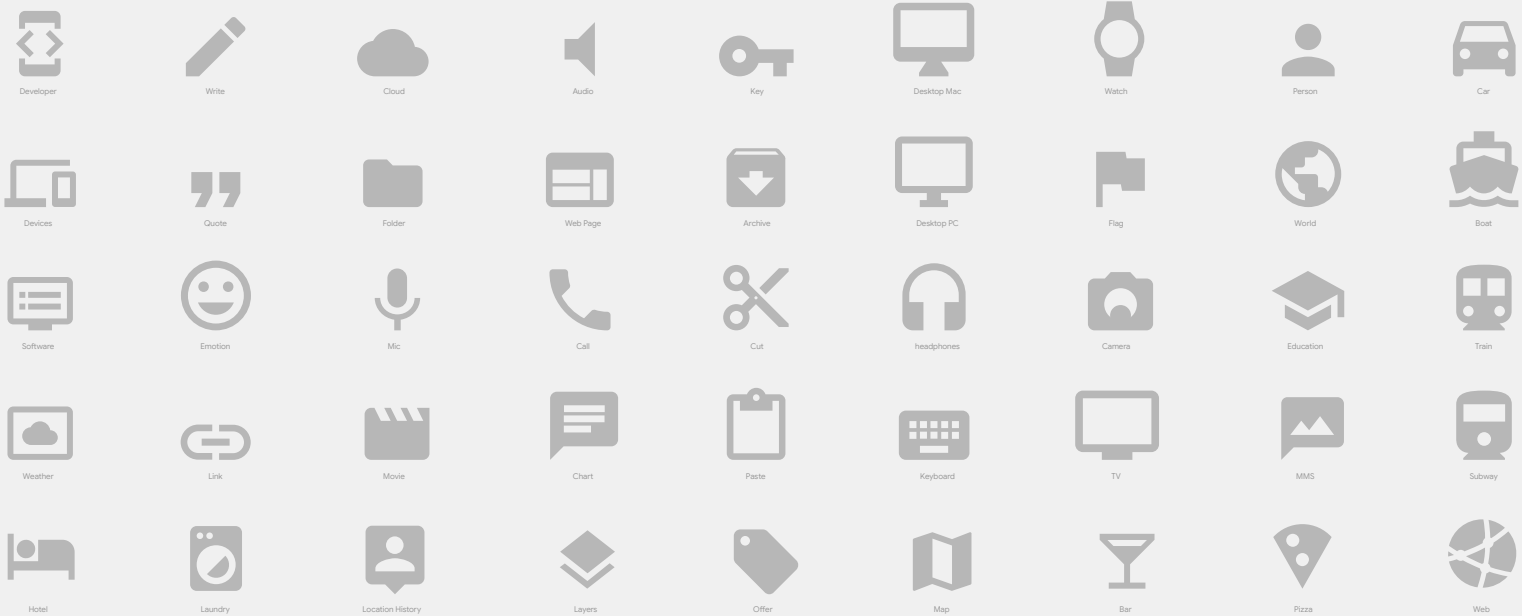




Icons

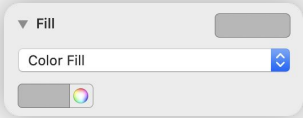
All icons are vector objects and can be recolored using the fill menu.





Icons

All icons are vector objects and can be recolored using the fill menu.




Cafe


Theatre


Gaming


Florist


Restaurant


Gas


Delivery


Hospital


Taxi


Print

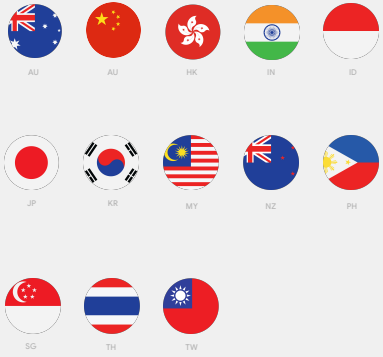

Radio


Stream

Flags

Americas

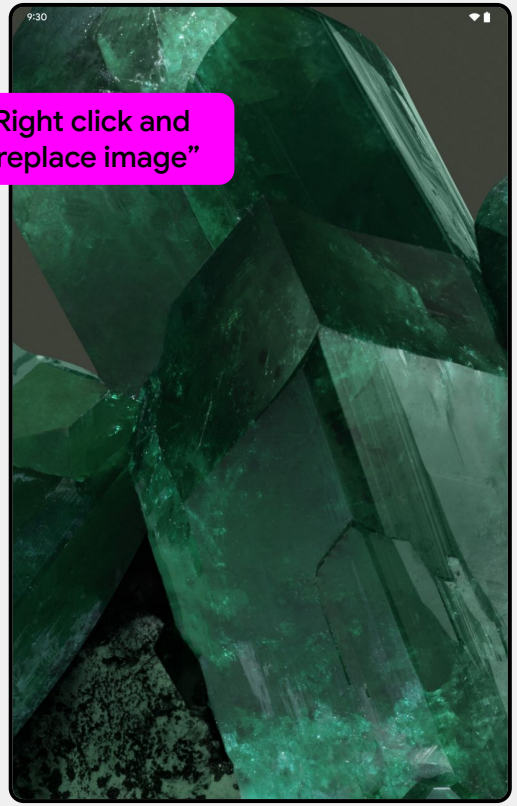
APAC

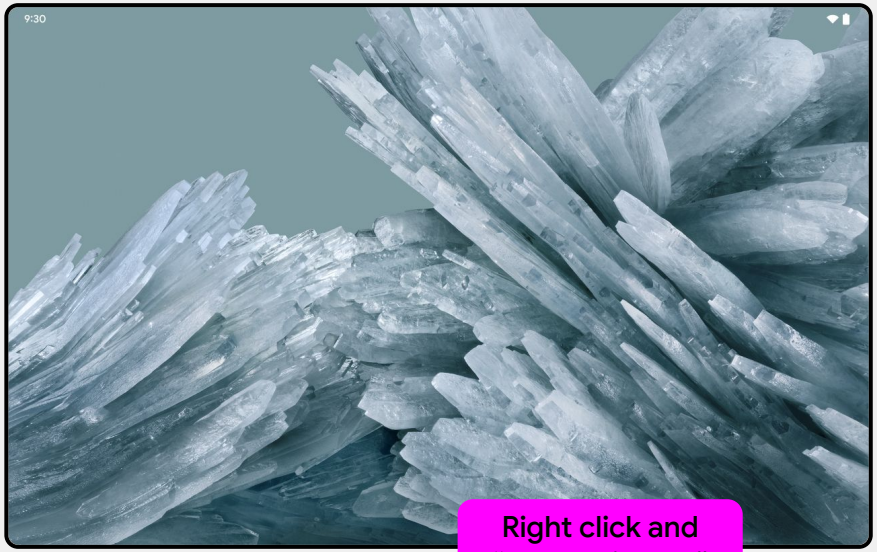


EMEA

WIREFRAME - MOBILE

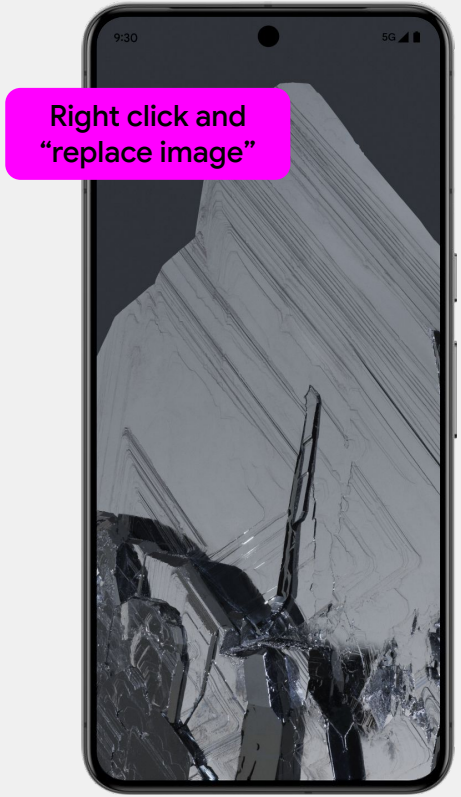
WIREFRAME - TABLET



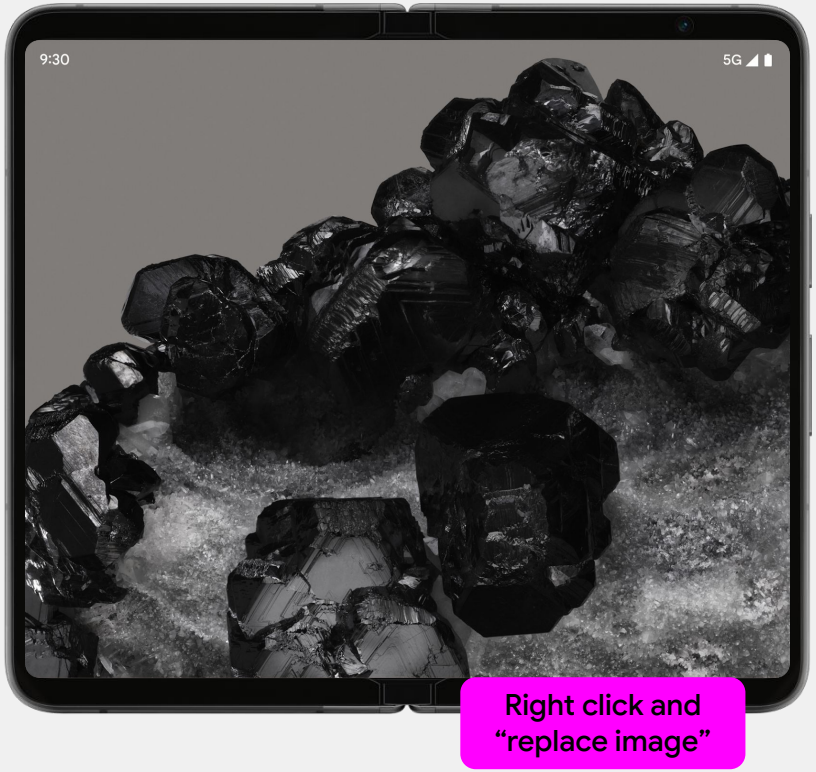


WIREFRAME - WEAR OS

DEVICE FRAME - PIXEL 8



DEVICE FRAME - PIXEL FOLD

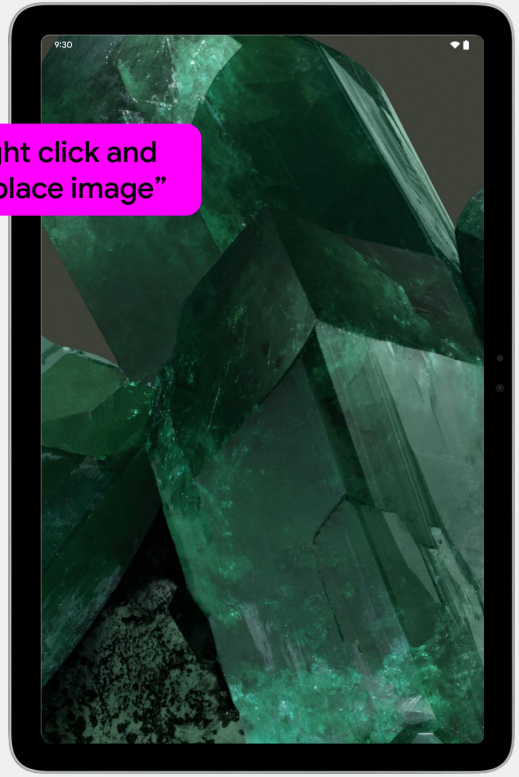


DEVICE FRAME - IPHONE 14

Right click and “replace image”

Right click and “replace image”

DEVICE FRAME - TABLET



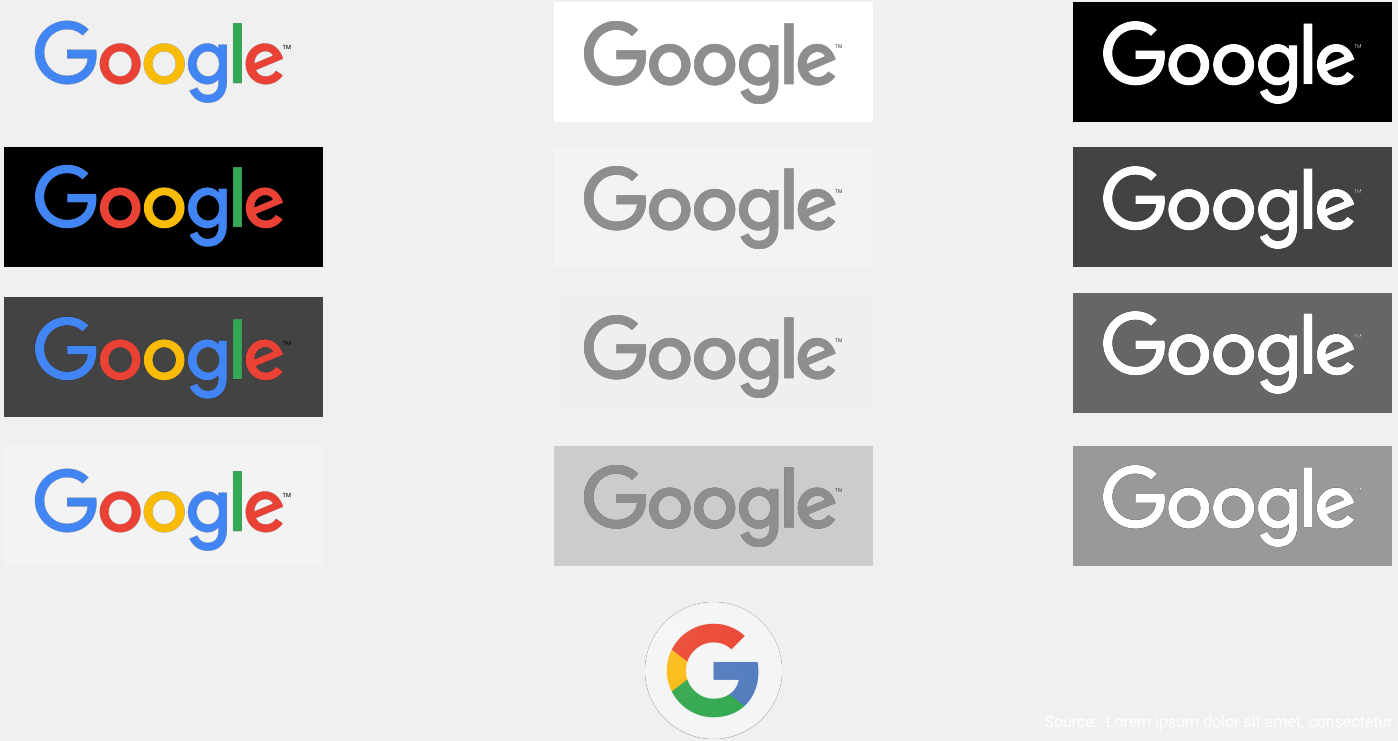
DEVICE FRAME - WEAR OS

DEVICE FRAME - CHROMEBOOK PLUS

DEVICE FRAME - PIXELBOOK GO

Logo Library

Logos can be scaled to any size



Source: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis non erat sem