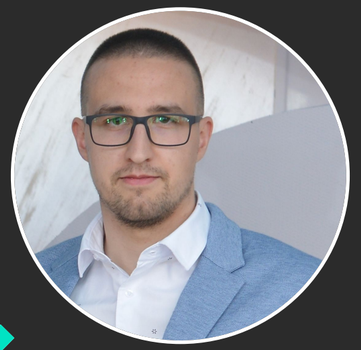
droidcon academy

# Understanding the Concept of KMP




Stefan Jovanović

@StevdzaS

in/stevdza-san

# Overview

- Quick Kotlin Overview
- KMP Overview
- The Concept of Sharing
- Multiplatform libraries
- KMP vs Compose Multiplatform
- KMP vs Flutter
- Basic KMP Project Structure
- Targets and Source Sets
- Expect/Actual Mechanism
- Summary

droidcon academy

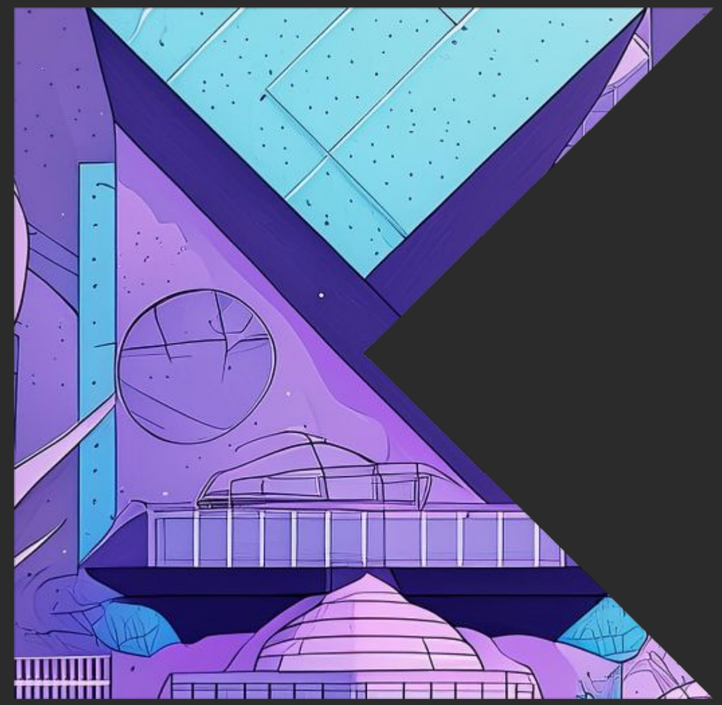
# Quick Kotlin Overview

## Up Next

droidcon academy

# Quick Kotlin Overview

- Concise
- Far better then Java
- Keeping up with technology
- Jetpack Compose



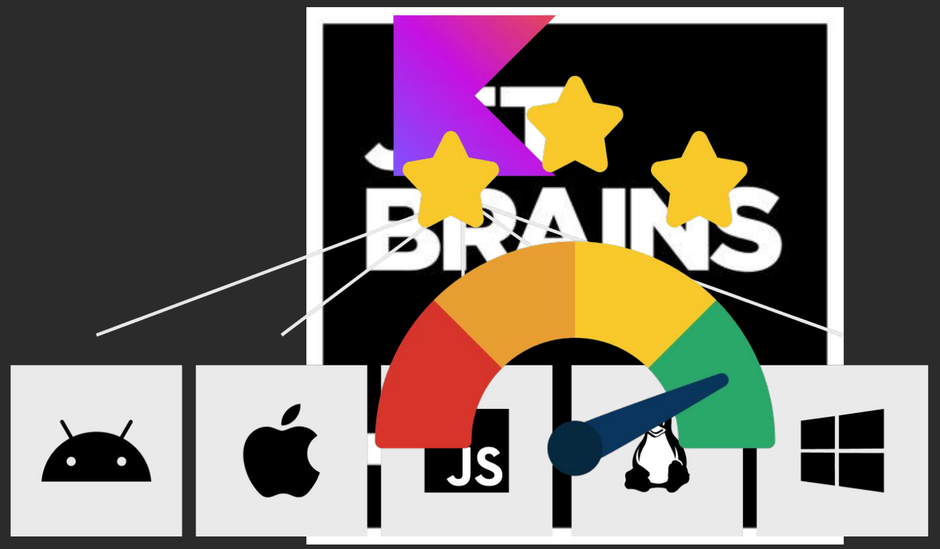
# KMP Overview

## Up Next

droidcon academy

# KMP Overview

- About KMP
- Developed by Jetbrains
- Build apps for various platforms
- Single codebase
- Native performance



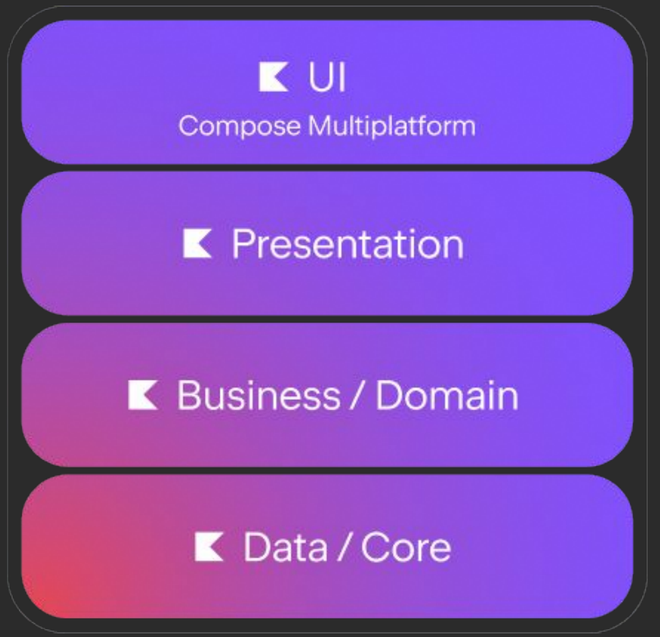

# The Concept of Sharing

## Up Next

droidcon academy

# The Concept of Sharing

- Partial sharing
- Share Business Logic, keep Native UI
- Share Business Logic and UI




droidcon academy

# Multiplatform libraries

## Up Next

droidcon academy

# Multiplatform libraries

- Platform-specific implementation
- Responsible developers
- KMP - Growing ecosystem
- Good examples:
  - KotlinX Serialization
  - Ktor
  - MongoDB Realm
  - KotlinX Date/Time
  - KMP Settings
  - KMP Auth

droidcon academy

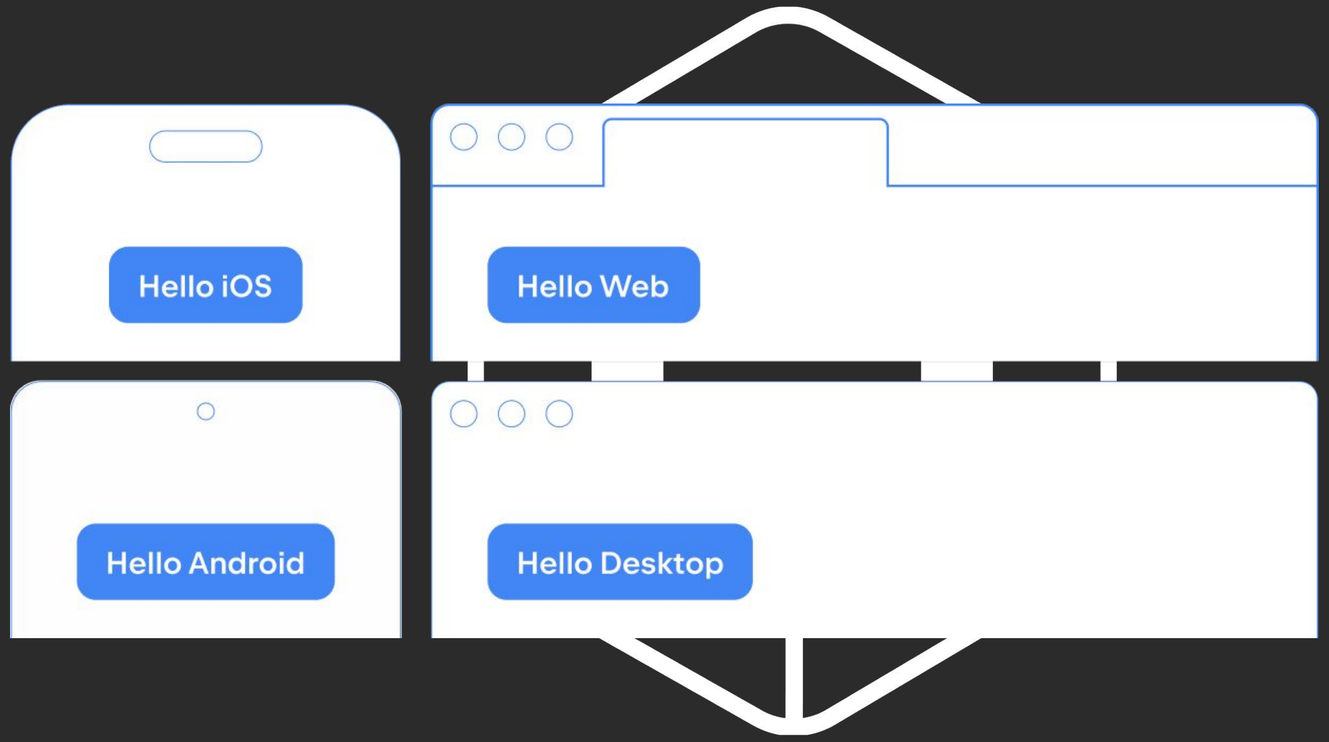
# KMP vs Compose Multiplatform

Up Next

droidcon academy

# KMP vs Compose Multiplatform

- Jetpack Compose
- Bright future
- Share UI




droidcon academy

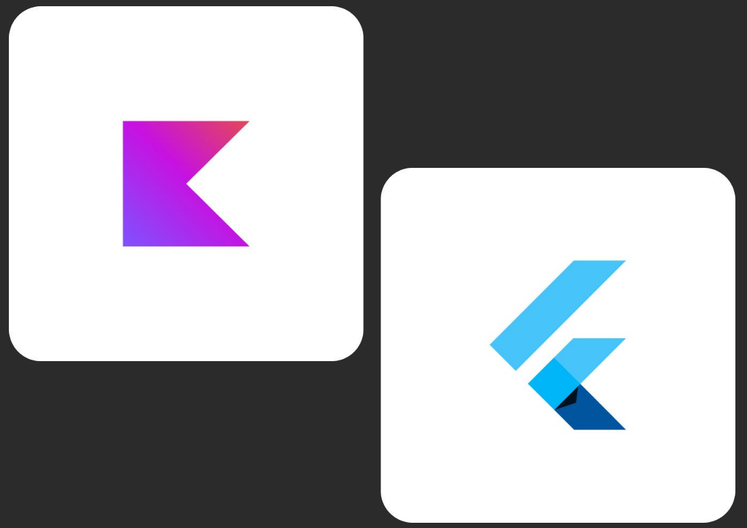
# KMP vs Flutter

Up Next

droidcon academy

# KMP vs Flutter

- Another cross-platform technology?
- Dart vs Kotlin
- Already made widgets
- Development costs
- Kotlin is quite flexible

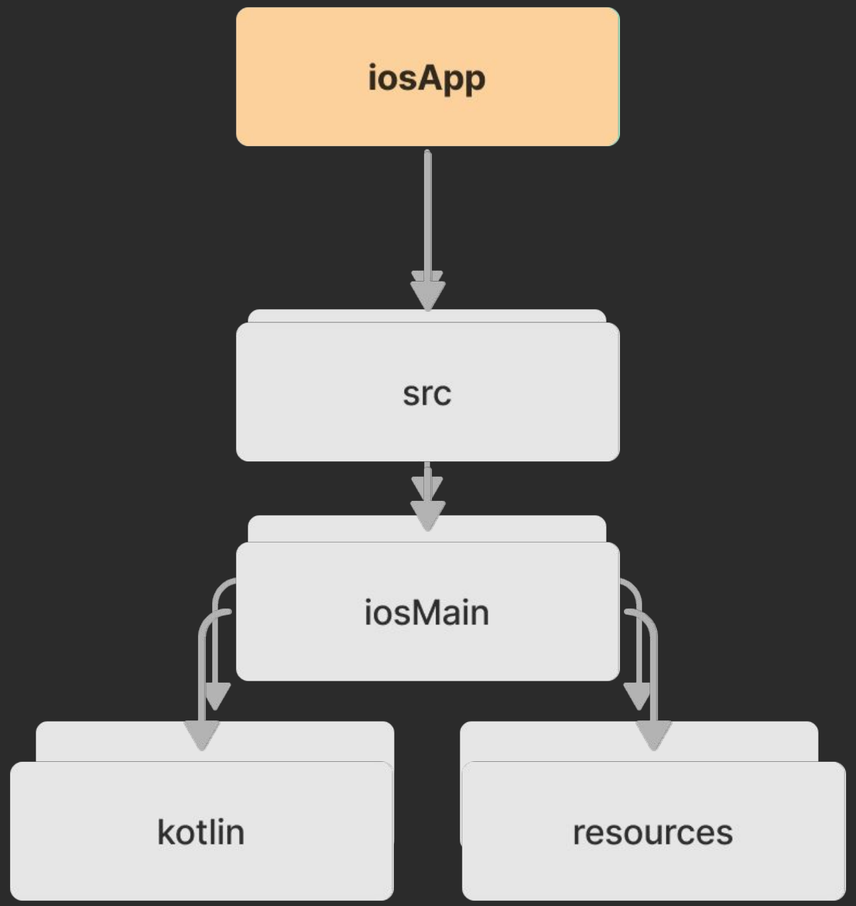


# Basic KMP Project Structure

Up Next

# Basic KMP Project Structure

- Keep things organized
- Sharing is caring
- Platform-specific code




droidcon academy

# Targets and Source Sets

Up Next

droidcon academy

# Targets and Source Sets

- Target - platform
- Source Set - directory
- commonMain

```
// commonMain/kotlin/common.kt
// Compile-time error in common code
fun common() {
    val versionCode: Int = BuildConfig.VERSION_CODE
}
```

# Targets and Source Sets

- If you want Kotlin to compile your code to a specific platform, declare a corresponding target.

- To choose a directory or source file to store the code, first decide among which targets you want to share your code:
  - If the code is shared among all targets, it should be declared in commonMain.
  - If the code is used for only one target, it should be defined in a platform-specific source set for that target (for example, *jvmMain* for the *JVM*).

- Code written in platform-specific source sets can access declarations from the common source set. For example, the code in *jvmMain* can use code from *commonMain*. However, the opposite isn't true: *commonMain* can't use code from *jvmMain*.

# Expect/Actual Mechanism

Up Next

droidcon academy

# Expect/Actual Mechanism

- Expected behavior/Platform implementation
- commonMain (expect)
- Platform - androidMain, iosMain... (actual)
- expect declaration - no implementation
- Same package path
- Kotlin interface
- Enum classes

# Expect/Actual Mechanism

```
// An else clause is required:
fun matchCity(city: City) {
    when (city) {
        City.Belgrade -> println("Serbia")
        City.Sarajevo -> println("Bosnia")
        City.Podgorica -> println("Montenegro")
        else -> println("Some other country")
    }
}
```

droidcon academy

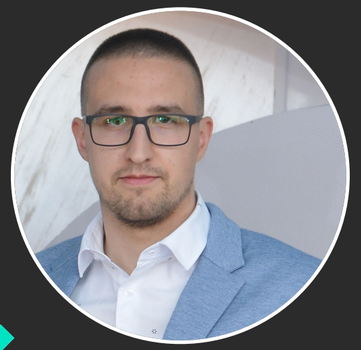
# Summary

- Transition from Java to Kotlin.
- KMP and cross platform development.
- Multiplatform libraries.
- Compose Multiplatform and sharing UI.
- Flutter vs Kotlin
- Targets and Source Sets
- Expect/Actual mechanism

droidcon academy

droidcon academy

# Thank you!




Stefan Jovanović

𝕏 @StevdzaS

in/stevdza-san