Continue

Kotlin library

Want to master Kotlin and build stunning apps? Whether you're into Android development, web dev, or just want to pick up a new skill, Kotlin is an awesome choice! It's simple, powerful, and getting more popular by the day. This Kotlin roadmap will guide you through the basics and advanced concepts so you can code confidently. Kotlin Roadmap This article covers everything you need to know about Kotlin, from working with variables and control structures to complex topics like coroutines and null safety. We'll also dive into what sets Kotlin apart from Java, how it's used for Android dev, and explore frameworks like Spring Boot and Ktor. What is Kotlin? Kotlin is a modern programming language that runs on the Java Virtual Machine (JVM) and can be used alongside Java. It was created by JetBrains to be easy to use, less prone to errors, and simple to read and write. Kotlin is mainly used for building Android apps but can also be used for web, server-side, and desktop applications. Prerequisites to Learn Before Diving into Kotlin Before you start learning Kotlin, make sure you have a solid grasp of the following: * Basic Programming Concepts * Version Control (Git) * Object-Oriented Programming Kotlin provides various tools for developers to efficiently manage their app's data. It covers essential topics such as using arrays and lists in Kotlin, understanding how to create classes and objects, inheritance, interfaces, null safety, extensions, coroutines, error handling, testing frameworks, and deployment to the cloud. This comprehensive guide includes: * Managing app data with Kotlin tools * Organizing code into classes and objects for better program management * Creating new classes based on existing ones using inheritance and defining common rules with interfaces * Handling missing or empty values safely through null safety features * Extending existing classes with new functions, making it easier to work with UI components * Using coroutines to handle multiple tasks without slowing down the app * Properly catching and handling errors with try-catch blocks and custom exceptions * Writing unit tests in Kotlin to ensure code correctness * Utilizing popular frameworks and libraries for Android development and web applications * Deploying Kotlin apps to the cloud, including Google Cloud and AWS, using Docker packaging Kotlin is a versatile language for developing apps, websites, and backend systems. By learning its basics and syntax, you can start coding right away. As you explore more features, you'll find Kotlin to be a flexible and rewarding language that offers many exciting development opportunities. Some notable libraries in the Kotlin ecosystem include: * `kotlinx.coroutines` for multiplatform support of coroutines * `Ktor` for building microservices, web applications, and more * `Koin` for lightweight dependency injection * `okio` for a modern I/O library * `SQLDelight` for generating typesafe APIs from SQL For Kotlin/Native, you can use the `-produce library` or `-p library` flag to generate a library with the Kotlin/Native compiler. To link to a library, use the `-library ` or `-l ` flag. The `cinterop` tool produces .klib wrappers for native libraries. You can inspect and install libraries using the `klib` utility, which provides commands for listing contents, inspecting details, installing, and removing libraries. Here's an example of creating a library and using it in a program: First, create a tiny library source code in `kotlinizer.kt`: ```kotlin package kotlinizer val String.kotlinized get() = "Kotlin $this" ``` Compile the library with the Kotlin/Native compiler: `$ kotlinc-native kotlinizer.kt -p library -o kotlinizer` Check out the contents of the library: `$ klib contents kotlinizer` Install the library to the default repository: `$ klib install kotlinizer` Create a short program in `use.kt`: ```kotlin import kotlinizer.* fun main(args: Array) { println("Hello, ${"world".kotlinized}!") } ``` Compile the program linking with the library: `$ kotlinc-native use.kt -l kotlinizer -o kohello` Run the program: `$ ./kohello.kexe Hello, Kotlin world!` Have fun exploring the Kotlin ecosystem and building exciting projects! To compile kotlin libraries, specify either the current directory or an absolute path with the -repo flag. Libraries in the default repository, located at ~/.konan, can be accessed using the kotlin.data.dir Gradle property or the -Xkonan-data-dir compiler option. For Kotlin/Native libraries, the zip file structure includes predefined directories for ir, targets, kotlin, native, linkdata, resources, and manifest files. The layout is as follows: component_name/ contains ir/Serialized Kotlin IR, targets/$platform/Kotlin compiled to LLVM bitcode, and native/bitcode files of additional native objects. Relative paths in klibs are available since Kotlin 1.6.20, allowing for a serialized IR representation of source files within the library. By default, stored paths are absolute, but the -Xklib-relative-path-base compiler option can change this to use only relative paths. To make relative path usage work, pass one or multiple base paths of source files as an argument: tasks.named("compileKotlin").configure { compilerOptions.freeCompilerArgs.add("-Xklib-relative-path-base=$base") }. Last modified on September 25, 2024. The career of app development has seen a significant surge in recent years, with Kotlin and Swift emerging as the dominant programming languages for Android and iOS applications respectively. With its introduction in 2010, Java paved the way for Android app development due to its simplicity and ease of use. However, in recent times, Kotlin has replaced Java as the primary language for Android app development. Kotlin has become the go-to language for Android app development, backed by Google and gaining popularity among developers since its introduction in 2017. It overtook Java and C++ to claim the top spot and is now considered the primary language for creating Android mobile applications. The pandemic and increased use of technology have made it essential for private sector institutions and governments to develop mobile apps to connect with people and share information. Kotlin's features make it an attractive choice for developers. It is an open-source language, compatible with both Java and JavaScript, allowing for easy modification and building. This interoperability with Java is a significant advantage, as developers can switch between the two languages seamlessly. Kotlin also boasts null safety, reducing runtime crashes, and allows developers to add extensions to the existing language, making it more efficient. Kotlin libraries provide extensive support to developers, including Kotless, which simplifies serverless deployment, Kotest for testing, Ktor for building connected systems, and Koin, a dependency injection library. These features make Kotlin an ideal choice for app development, offering speed, better performance, and ease of use. Despite its growing popularity, some may still favor Java due to familiarity, but developers find Kotlin more productive and efficient. With its open-source nature, interoperability with Java, and numerous libraries, Kotlin has solidified its position as the top language for Android app development. Kotlin is a cross-platform programming language that's gaining popularity for developing Android apps, backed by Google and introduced in 2017. It offers benefits similar to other modern languages like Java and Kotlin. As a developer, working on Kotlin can lead to high career growth prospects. A DI framework called Koin helps manage controllers and services, specifically designed for Kotlin developers. Another library, Xodus, is written in both Java and Kotlin, providing transactional embedded functionality. Kotlin's 2.1.20 release includes several updates such as performance improvements, bug fixes, and new features like custom formatters for development builds and improved UUID support. This version also introduces common atomic types and time-tracking functionality. The Kotlin plugin is now bundled with IntelliJ IDEA 2023.3 and Android Studio Iguana (2023.2.1) Canary 15, eliminating the need to install it from JetBrains Marketplace. To update to the new version, change your build scripts to use Kotlin 2.1.20. If needed, download the command-line compiler from GitHub. For more information on updates and features, refer to the release notes on GitHub or What's new in Kotlin 2.1.20. Stay updated with the latest Kotlin developments by subscribing to receive notifications via the form at the bottom of this post.

Kotlin library github. Kotlin library version. Kotlin library in java. Kotlin library in java project. Kotlin library mod. Kotlin library 0 was compiled with a newer. Kotlin library mode. Kotlin library repository. Kotlin library functions. Kotlin-library-conventions. Kotlin library gradle. Kotlin library template. Kotlin library search. Kotlin library android. Kotlin library development.