**Getting started with Kotlin for android developer**

If you're a [mobile developer](https://www.netguru.co/services/mobile) you are probably using java language for building android apps. But did you know that new languages that might challenge Java's in the [Android](https://www.netguru.co/services/android-mobile-development) are popping up all over the place? One of them is [Kotlin](https://www.netguru.co/services/kotlin" \t "_blank), This year, [Google acknowledged Kotlin and made it the second official language of Android](https://clearbridgemobile.com/what-you-missed-at-google-io/).

In May 2017 , Kotlin tools are included with Android Studio 3.0 by default. [Kotlin](https://www.netguru.co/blog/kotlin-faq-what-you-need-to-know-about-this-official-language-for-android-development) is built by[JetBrains](https://www.jetbrains.com/) which is powerful, simple and versatile. There are other JVM languages can be tried to use on Android, but Kotlin offers integrations with Android Studio that is the Google's primary IDE for Android.   
  
**Should java developers learn Kotlin or remains stick to java?**For many of us kotlin is new but when google announced it as official language for android, many people loved it but the fact is same things happened when swift was released back in 2014.Now we can see swift’s success, I believe the same story goes for kotlin too.It may take some time but it will, many developers will start using kotlin soon, there will be lots of resources regarding kotlin in blogs, stackoverflow or documentation, there’s no doubt.

**What is Kotlin?**Kotlin is statically typed programming language for the JVM and JavaScript. It is both an object-oriented (OO) and functional programming (FP) language. It is compatible with OO and FP styles allowing developers to incorporate elements from each form.  
Kotlin can be used for Android development, server-side development, web development, desktop development and Kotlin/Native is currently in production.

Kotlin is free and will remain free. It is developed under the Apache 2.0 license and the source code is available [on GitHub](https://github.com/jetbrains/kotlin).

**Kotlin can build applications for :**

* JVM
* Android
* Browser
* Native

### Set-up Kotlin with android studio

### Step - 1 Install the Kotlin Plugin for your Android Studio.

### Android Studio → Preferences →Plugins → Browse Repository → type “Kotlin” in search box → install

### C:\Users\nisha\Desktop\setup.png

### Step - 2 Add Kotlin classpath to project Build.Gradle

buildscript {  
 **ext.kotlin\_version = "1.2.70"**  
 ext.supportLibVersion = "25.3.0"  
 repositories {  
 jcenter()  
 }  
 dependencies {  
 **classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin\_version"  
 classpath "org.jetbrains.kotlin:kotlin-android-extensions:$kotlin\_version"**  
 }  
}

### Step 3: Add Kotlin library and apply Kotlin Plugins in your module Build.gradle.

apply plugin: 'com.android.application'  
**apply plugin: 'kotlin-android'  
apply plugin: 'kotlin-android-extensions'**android {  
 // ... various gradle setup  
}  
  
dependencies {  
 compile fileTree(dir: 'libs', include: ['\*.jar'])  
 testCompile 'junit:junit:4.12'  
 compile "com.android.support:appcompat-v7:$supportLibVersion"  
 compile "com.android.support:recyclerview-v7:$supportLibVersion"  
 **compile "org.jetbrains.kotlin:kotlin-stdlib:$kotlin\_version"**}

### Step 4: Ready to go

Now Kotlin is setup for your app development, you could start writing Kotlin code (in .kt extension). Another way is convert your Java file to Kotlin, using **Shift-Alt-Cmd-K** or **Shift-Shift**+ *search*Convert Java File to Kotlin File.

**Some java Issues addressed in kotlin**

* **Null references are**[**controlled by the type system**](https://kotlinlang.org/docs/reference/null-safety.html)**:** One of the most common exception is null pointer exception , which is eliminated in kotlin using !! operator.
* [**No raw types**](https://kotlinlang.org/docs/reference/java-interop.html)**:** Kotlin is designed with Java Interoperability in mind. Existing Java code can be called from Kotlin in a natural way, and Kotlin code can be used from Java rather smoothly as well.
* **Arrays in Kotlin are**[**invariant**](https://kotlinlang.org/docs/reference/basic-types.html#arrays) **:** Kotlin does not allow us to assign an Array<String> to an Array<Any>, which prevents a possible runtime failure (but you can use Array<out Any>.Kotlin also has specialized classes to represent arrays of primitive types without boxing overhead: ByteArray, ShortArray, IntArray and so on. These classes have no inheritance relation to the Array class, but they have the same set of methods and properties. Each of them also has a corresponding factory function:
* **Kotlin has proper**[**function types**](https://kotlinlang.org/docs/reference/lambdas.html#function-types)**, as opposed to Java's SAM-conversions :** Kotlin uses a family of function types like (Int) -> String for declarations that deal with functions: val onClick: () -> Unit
* **Kotlin does not have checked**[**exceptions**](https://kotlinlang.org/docs/reference/exceptions.html) **:** Checked exceptions are those where the compiler forces the caller of a function to catch or (re-throw) an exception. These are often unnecessary and cause empty catch blocks. Non-existent checked exceptions are an annoyance for developers because they’re forced to weed through code to identify a possible exception that never actually occurred. As a solution, Kotlin removed them entirely, which again minimizes verbosity and improves type-safety.

## **Why java developers should learn kotlin?** Kotlin has introduced new function that are not supported by java. We have defined them below.

* [**Extension functions**](https://kotlinlang.org/docs/reference/extensions.html) **:** It is used to extend a class with new functionality. Basically, an extension function is a member function of a class that is defined outside the class.  
  For example, you need to use a method to the [String class](https://www.programiz.com/kotlin-programming/string) that returns a new string with first and last character removed; this method is not already available in String class. You can use extension function to accomplish this task.

fun String.removeFirstLastChar(): String = this.substring(1, this.length - 1)

fun main(args: Array<String>) {

val myString= "Hello Everyone"

val result = myString.removeFirstLastChar()

println("First character is: $result")  
}  
Output : First character is: ello Everyone

* [**Null-safety**](https://kotlinlang.org/docs/reference/null-safety.html) **:** Kotlin compiler by default doesn’t allow any types to have a value of null at compile-time.
* [**Smart casts**](https://kotlinlang.org/docs/reference/typecasts.html) **:** Suppose we have a function : **fun** someMethod(obj: Any?) {}  
  Here the variable **obj** is both optional and derived from the Kotlin root class (**Any**). This means that **obj** could be basically any kind of class if it isn’t null.
* [**String templates**](https://kotlinlang.org/docs/reference/basic-types.html#strings) **:** They are represented by the type String. Strings are immutable, can concatenate string using + operator.
* [**Properties**](https://kotlinlang.org/docs/reference/properties.html) **:** Classes in Kotlin have properties. These can be declared as mutable, using the var keyword or read-only using the val keyword.
* [**Primary constructors**](https://kotlinlang.org/docs/reference/classes.html)**:** The classes in kotlin has primary and secondary constructor

class Person constructor(firstName: String) { ... } = > Primary constructor  
  
class Person { => Secondary constuctor

constructor(parent: Person) {

parent.children.add(this)

}

}

* **Declaration-site variance:** It supports **“delegation”** design pattern by introducing a new keyword **“by”**. Using this keyword or delegation methodology, Kotlin allows the derived class to access all the implemented public methods of an interface through a specific object. The following example demonstrates how this happens in Kotlin.

interface Base {

fun printMe() //abstract method

}

class BaseImpl(val x: Int) : Base {

override fun printMe() { println(x) } //implementation of the method

}

class Derived(b: Base) : Base by b // delegating the public method on the object b

fun main(args: Array<String>) {

val b = BaseImpl(10)

Derived(b).printMe() // prints 10 :: accessing the printMe() method

}

* [**Lambda expressions**](https://kotlinlang.org/docs/reference/lambdas.html)**:** To print hello world in lambda expression use below code :

**Basic Syntax**val printHelloWorld = {

println("Hello, world!")

}  
  
printHelloWorld()

// or

printHelloWorld.invoke()

**Benefits of kotlin**

### Automatic type casting

### Make android development much easier

### Safer than Java

### Easily setup with android studio

### Language and environment fully tested before release

### Can create functions inside class

### Everything in Kotlin is closed by default

### No need to use findViewById

### Saves lengthy line of code

### Conclusion: We can conclude by saying that as it is evolving as a favorite language for an android developer as it is safe and expressive. It is getting enhanced day by day and. If you are already an expert in java , I guess its high time you should start learning Kotlin and working on it.