# android kotlin tutorials

There was an article on apmba a while back about 7 reasons why we should start studying Kotlin for Android in 2018. The reasons why we should start learning Kotlin for Android in 2018 are now clear, yet learning materials for beginners to create Android apps using Kotlin are still scarce. As a result, the author will help readers who are looking for resources to learn how to construct Android apps using the Kotlin programming language.

**For Whom Is This Tutorial Intended?**

This lesson is for anyone who wants to learn how to make Android apps with Kotlin, whether they've previously learnt to code or have never attempted to do so. Readers should ideally have studied programming and be familiar with basic concepts such as branching (if), looping (for), and functions.

**What Will We Discover?**

Android programming is a big topic, a very big issue. It will take a long time to cover the entire topic of Android programming. As a result, we'll start with the mandatory topics before moving on to the optional ones.

The author refers to the themes that are required to construct a minimalist application as obligatory topics. To put it another way, the topic we are studying is everything that can assist us in realising the application we want to create in the easiest way possible (even though it has minimal features). We will cover the following topics as a requirement:

RecyclerView Viewing Data Sets with Activity and Layout XML Intent  
Getting data from an API (JSON) and saving it locally (SQLite)

Following the mandatory topics, we will attempt to explore additional areas that can help us improve the quality of our application, such as:

AlarmManager Job Scheduler Firebase (Authentication, Realtime Database, Firestore, and other topics) and other topics (please request)

This tutorial's main goal is to assist Codepolitan readers who wish to learn how to construct Android apps using the Kotlin programming language. The author wants readers who have never coded before to be able to construct their own simple Android applications, and readers who have previously been able to create Android applications using Java to be able to do so with Kotlin.

This tutorial's writing style is influenced by the author's own learning experiences. The author employs the strategies that he or she deems to be the most practical (according to the author's definition, of course). The order in which the topics are addressed is likewise the author's discretion and does not follow Google's criteria.

Enough with the introductory phrases. Let's get started with a more solid method by installing Android Studio first.

**Getting Android Studio to Work**

We'll be utilising Android Studio as our primary tool for creating Kotlin-based Android apps. Go to https://developer.android.com/studio/index.html to download Android Studio. Click the DOWNLOAD ANDROID STUDIO button on the next page, as seen in the image.

Please continue the Android Studio installation process with one of the YouTube video tutorials after the download is complete:

* Mac OS X Windows Linux (Ubuntu)
* Android Studio 3.0.1 or later is required.
* With Kotlin in Android Studio, you can say hello to the world.

When the installation is finished, launch the Android Studio application and wait for the Welcome to Android Studio window to display.

Start a new Android studio project by clicking the Start a new Android studio project button (top option). Fill in the application name with the name of the project you want to work on. The name "Hello World" is given by the author. Replace the reader's personal or company domain in the Company domain field. Specify the project location in the Project location field; avoid saving it in a location with spaces, as this can create issues (click the ... button on the right). Last but not least, don't forget to enable Kotlin support. Then press the Next button.

We'll be prompted to choose which devices can utilise our software in the next window. Keep in mind that Android encompasses not only smartphones and tablets, but also wearables (Wear/Smartwatch), automobiles (Android Auto), televisions, and Internet of Things (IoT) (Android Things). Put a check box next to Phones and Tablets because we'll only be learning how to develop Android apps for smartphones and tablets.

In the Phone and Tablet choices, the API selector just does what Android Studio says (unless the reader already knows what API level to use). The API level chosen will be the minimal Android version that can be used to install our app afterwards. Android devices with versions lower than the bare minimum determined by us will be unable to install it.

We'll be asked to choose an initial template in the next box. Simply choose Basic Activity and press the Next button.

The Activity Name, Layout, and Activity Title will be specified in the next window. For the time being, disregard everything and press the Finish button. Allow the gradle process to complete.

**XML for Activity and Layout**

If nothing changes, when you create a new project in Android Studio, it will appear like this:

Three key files are visible from here: MainActivity, activity main.xml, and content main.xml. Because they are in the layout folder, the activity main.xml and content main.xml files are layout files. The layout folder contains files that are responsible for showing anything to the user. However, we need another component, namely an Activity, before the layout file can display what the user sees.

An Activity is a page or a screen that displays anything to the user in Android programming. Isn't the layout in charge of presenting anything to the user? Take a look at the following MainActivity file's contents:

Notice the statement setContentView(R.layout.activity main) on the fifth line. This line associates the layout activity main.xml with MainActivity, causing the activity main.xml layout to appear when MainActivity is executed.

Although the author claims that the layout file is in charge of showing something to the user, it does so only with the help of an Activity. Even if we have planned the layout in such a way, if there is no Actvitiy that sets it in the setContentView, the layout will not display. If we build an Activity but don't use setContentView to connect it with a layout, the resulting page won't be able to display anything.

On Android, Getting to Know XML Layouts The beginning tag's name will always begin with a capital letter. The tag is expressed as Opening>/Opening> if it has content (such as an Opener and a ContentGroup). If it doesn't have any content (for example, Content), it's written as Content/>. Take note of how the / sign is written.

Let's look at the activity main.xml file now:

Isn't that a lot? But don't worry, we don't need to know everything in activity main.xml just yet. Lines 23 and 25-32 are the ones we need to pay attention to.

Lines 25-32 define the FloatingActionButton component, which is the pink button in the bottom right corner. Lines 25-32 can be removed because they will not be used for the time being.

A command to read another layout file from a layout file is found on line 23. If the reader is aware, there is no component in activity main.xml that says Hello World, as displayed in the Preview image. Because the Hello World text is stored in the content main.xml file, this occurs.  
Concerning the activity main.xml and content main.xml files

We produced an Activity and a layout consisting of two files when we picked Basic Activity during project creation. The contents of content main.xml remain part of activity main.xml despite the fact that the file is independent. The line include layout="@layout/content main" /> can be read as "I want to include the contents of content main.xml in this file (activity main.xml) starting from this line."

Let's have a look at the content main.xml file's contents:

Lines 11-18 should be visible to the reader. The component that shows the text "Hello World" on the preview screen is defined in this section (line 14).  
Learn more about View and ViewGroup are two types of views.

We can still see two components in the content main.xml file, namely android.support.constraint and android.support.requirement.

TextView and ConstraintLayout The TextView View is an example of a View that displays text on the screen. Other views beyond TextView include EditText, which is used to enter data (input), Button, which is used to make buttons, and ImageView, which is used to display images.

A single point of view is insufficient. It needs a parent to which it can bond. ConstraintLayout is an example of a ViewGroup, which is the parent of a View.

**View and ViewGroup attributes**

To set its height and width, each View and ViewGroup component must have the android:height and andorid:width properties. We have three options for filling in the blanks:

Wrap content denotes that the height or width of the component follows the content of the component (not fixed)  
match parent denotes that the height or width is determined by the parent.  
xxdp indicates that we wish to manually set the height or width by writing the size (change xx with a number) and using dp units, such as 100dp, 200dp, and 300dp.

**LinearLayout and RelativeLayout are two different types of layouts.**

We won't be using ConstraintLayout in this lesson series because I haven't mastered it yet. Instead, we'll use LinearLayout and RelativeLayout, the two most commonly used ViewGroups.  
LinearLayout

Now, in lines 2 and 20, change the ViewGroup ConstraintLayout to a LinearLayout and add another TextView component to observe what happens in the Preview.

LinearLayout is a ViewGroup that arranges its components horizontally (left to right) or vertically (top to bottom) (vertical). LinearLayout will offer horizontal orientation if it is not set. We must manually modify the orientation of the components if we want them to be stacked vertically. In the opening tag, add the android:orientation="vertical" attribute:

**RelativeLayout**

RelativeLayout provides more free alternatives than LinearLayout, which only has two options for displaying its components. The following RelativeLayout code has the same layout:

All components will stack in the top left corner when using a RelativeLayout. We give a TextView an id and then indicate the TextView we want to display at the top. Line 13 demonstrates how to write id.

With the property android:layout below="@id/helloFirst," we can tell the other TextView to be beneath the TextView with the id helloFirst.

It's difficult to adequately describe how RelativeLayout works in this context. As a result, the author encourages readers to experiment with RelativeLayout to learn more about how to design diverse layouts.

**Developing Easy-to-Use Interactive Apps**

Change the following to the contents of content main.xml:

We have an EditText as a form for entering a name and a Button as a button that, when pressed, displays the contents of the EditText in the TextView.

These two lines are required since the components in activity main.xml and content main.xml will be used. The ids of the Button, TextView, and EditText components are btnEnter, textName, and editName, respectively.

**The App's Operation**

We'll start by detecting when the user presses the YES button. A setOnClickListener must be sent to a component to do this detection. We write btnLog in because our button's id is btnEnter. setOnClickListener -> -> -> -> -> -> -> -> -> -> -> -> -> -> -> Any component can be supplied the setOnClickListener method, so just write the component name before the method name. We write view -> inside the curly braces, then what actions we execute when the button is pressed to the right of the arrow.

We want the contents of the EditText with the id editName to be received and then filled into the TextView with the id textName when the button is pressed. The contents of the text in the component can be retrieved by writing namaKomponen.text. This method will fetch all components that have the "android:text" attribute in their XML.  
Run!

How will we know the results if it hasn't been implemented? We need an emulator or a native Android device to run programmes created in Android Studio. To run the programme on an emulator or a genuine Android smartphone, please follow the steps below.

* *Oktrik | Aneka Tips, Trik Dan Tutorial Gratis | 2022*. [https://www.oktrik.com](https://oktrik.com). Accessed 24 Mar. 2022.
* *AraKpt |*. [https://www.arakomputer.com](https://arakomputer.com). Accessed 24 Mar. 2022.
* *Tekno Arsip | Anton-Nb*. [https://www.anton-nb.com/tekno](https://anton-nb.com/tekno). Accessed 27 Mar. 2022.
* *Harian Regional -*. [https://www.harianregional.com](https://www.harianregional.com/). Accessed 27 Mar. 2022.

**Closing**

This is an introduction course for novices who want to learn how to make Android apps with Kotlin. Following that, we'll look at Intents. I hope this is clear to all readers. Please use the comments section to express any criticisms, suggestions, questions, or requests.