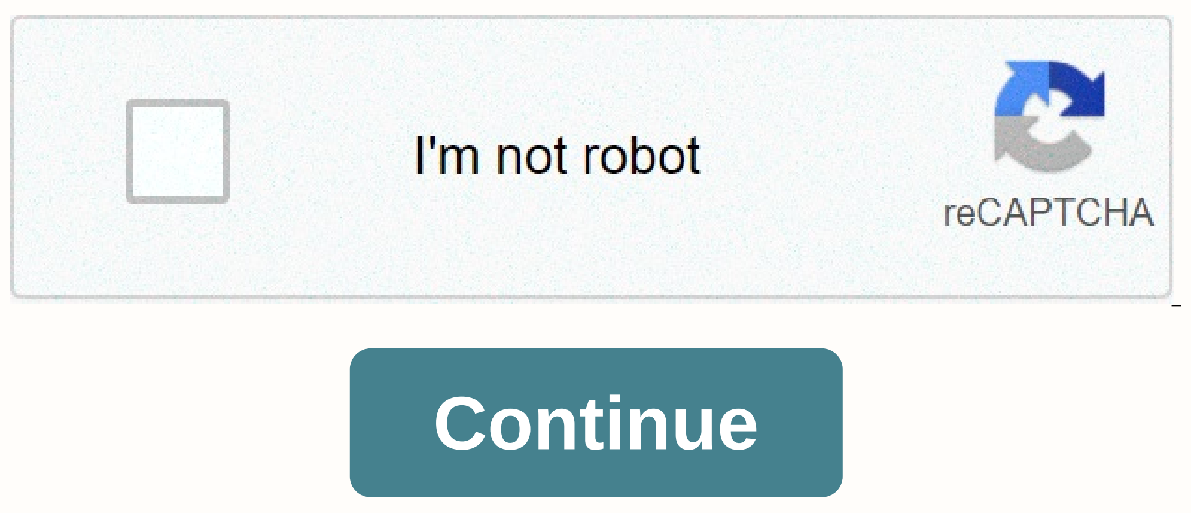
I'm not robot
reCAPTCHA
Continue

**Android application development tutorial for beginners pdf**

code Codelab-based In this free, pace android basics in Kotlin training for students new to programming, you learn the beginning Android programming concepts using Kotlin programming language and you build a number of apps using Android Studio. directions_run Training level: Beginner Start course code Codelab-based Take your Android coding skills to the next level in our free, pace advanced Android in Kotlin training. The course uses Kotlin programming languages and teaches you about notifications, graphics, and animations on Android, how to log users, add maps to your apps, and how to properly test your apps. Each lesson includes a tutorial with solution code in GitHub. directions_run Training level: Advanced Start course code Codelab-based In our free, pace Android Kotlin Fundamentals training, you learn basic Android programming concepts using Kotlin programming language and you build a variety of apps. Each code lab provides a tutorial with solution code in GitHub. directions_run Training level: Intermediate Start course code Codelab-based In our free, pace Android Developer Fundamentals training, you learn basic Android programming concepts using Java programming languages. You build a variety of apps, starting with Hello World, and working your way up to apps that plan jobs, update settings, and use Android Architecture Components. Each lesson includes a code lab with solution code in GitHub, concept documentation, and a slide deck. The Google Developers Training team designed this course. directions_run Training Level: Intermediate Get More Codelab-based Take Your Android Coding Skills to the Next Level in our free, self-filled Advanced Android Development training. The course uses the Java programming language and teaches you how to expand the user experience, improve app performance, and add features like custom views, animations, and location awareness. Each lesson includes a tutorial with solution code in GitHub, concept documentation, and a slide deck. The Google Developers Training team designed this course. directions_run: Advanced Read More If you've been encoded for a while, we've got courses to help you build your Android skills and learn best practices. Know object-oriented programming and want to learn Kotlin for Android programming and other purposes? Try one of these courses. Google is committed to promoting racial equality for black communities. See how. This section describes how to build a simple Android app. First you learn how to create a Hello, World! project with Android Studio and run it. Then you create a new interface for the app that takes user input and switches to a new screen in the app to view it. Before you start, there are two basic you need to understand about Android apps: how they provide multiple entry points and how they different devices. Android apps are built as a combination of components that can be activated individually. For example, an activity is a task that is not in the same way as an activity. The main activity starts when the user taps the app's icon. You can also direct the user to an activity from other places, such as a web site. Broadcast recipients and services allow your app to perform background tasks without a user interface. After you build your first app, learn more about the other application fundamentals app components. Android allows you to provide different resources for different devices. For example, you can create different layouts for different screen sizes. The system determines which layout to use based on the screen size of the current device. If any of the app's features need specific hardware, such as hardware, you can use the app's features. You can specify that your app requires specific hardware so that Google Play doesn't allow the app to be installed on devices without them. Once you've built your first app, learn more about device configurations in the device compatibility overview. With these two basic concepts in mind, move on to the next lesson to build your first app! Content and code samples on this page are subject to the licenses described in the content license. Java is a registered trademark of Oracle and/or its affiliates. Last Updated 2020-11-18 UTC. Android Programming Tutorials The small examples The small examples A full packaged Android App Development Tutorials App 👉 with examples and explanations for you to take a step forward in your career. The app is designed in a way that even a layman who has no prior knowledge of Android but has some basic knowledge of Java can learn Android App development and become a professional developer with advance level concepts and tutorials. Professional Android developers can use this Android Tutorials app to remember basic concepts when needed for coding. The best Android App Development CourseApp consists of tutorials, code samples, demo and theoretical explanations. You can learn basic concepts in Android App development, beginner level Android development concepts and examples with code and demo, advance level Android features with code and demo, professional Android app codes with explanation and useful info segments with important information about becoming a professional Android developer and knowledge of various important things related to Android App Development. Further, we have provided 9 professional apps fully developed with advance android development coding techniques that you can purchase and improve your skills further. The basics: Introduction to Android- Architecture &amp; Software Stack-Studio- Structure-Application Fundamentals-Intent-Views, Layouts &amp; Resources-Fragments- UI Widgets- Containers-Menu-Data Storage- JSON Parsing- FirebaseBeginner Level-UI Widgets-Menu-Intent-Fragments Intermediate Level-Advance UI-Containers-Material Design-Notifications-Storage-SQLiteAdvance Android - Use of Android Download Manager-Flashlight Torch Application Using using Camera2 API-QR Code Scanner Application-Convert Speech to Text-Convert Text to Speech-Bitcoin Price Index Application using JSON-Firebase User Authentication App-Youtube Player Application-Convert Website to Application-PDF Creator ApplicationHelpful Info- General Tips- Useful Resources- Useful Plugins-Important Libraries-Android Studio Keyboard Shortcuts Play-Store Optimization (ASO) - App MonetizationFull App CodesProfessionally developed Android Apps that you can buy to learn professional level programming or re-skin them like your own apps.- Grocery store Super Store- Fitness Workout App-Material Design-VPN App-Daily Time Tracker-Memory Game-Movies and Live TV App-Document Reminder-Health CalculatorAll these incredible tutorials with demo and many more to come to you. Become a professional Android App Developer using this amazing app by Bluestream.io. The more you learn, the more you groom and the more you practice, the more you gain skill. Then learn and practice Android App Development to become a skilled Android Developer and change the world for the better. Optimization.Improved stability. Learning Android app development may seem like a daunting task, but it can open up a huge world of possibilities. You can create the next hit app that changes the way we work or interact with each other. You may want to develop a tool that you can use to improve your workflow. Or maybe you'll just get a new skill that lands you a great job! Read also: How to create an app with no programming experience: What are your options? Either way, learning Android app development might not be as tough as you think, as long as you understand what all the different moving parts are for and have a roadmap to guide you through. This post is that roadmap! Step 1: Download the tools you need to develop Android appsFirst, create your development environment so your desktop is ready to support your Android development goals. For that you will need Android Studio and Android SDK. Fortunately, these both come packed together in a single download that you can find here. Android Studio is an IDE. It stands for integrated development environment, which is essentially an interface where you can enter your code (primarily Java or Kotlin) and access all the different tools needed for development. Android Studio gives you access to libraries and APIs from android SDK, giving you access native features of i System. You will also be able to build your app into an APK using Gradle, test it via a virtual device (emulator), and debug your code while it is running. With all that said, keep in mind that there are other options for your Android app development. For example, Unity is a very powerful tool for game development across platforms that also supports Android. Similarly, Visual Studio with Xamarin is an excellent combination for creating cross-platform apps in C#. We have practical guides to get started with each of these options: Android Studio is the best place for most people to start (with Android game development being an exception), especially since it provides all these extra tools and resources in a single place. Fortunately, the setup is very simple and you only need to follow the instructions on the screen. Get set up with Android Studio by following our handy guides: Android Studio tutorial for beginnersHow to install Android SDKStep 2: Start a new projectOn your machine, the next step is to start a new project. This is a simple process, but you will need to make a few decisions that will affect your Android app development going forward. Go to &gt; New &gt; project. You are now prompted to select a Project Template. This defines the code and UI elements that are included in your new app when it loads. The word Activity refers to a screen in your app. Thus, a project with No Activity will be completely empty, except for the basic file structure. A Basic activity on the other hand will create a home screen for your app and will add a button at the bottom and a hamburger menu at the top. These are common elements in many Android apps, so this can save you some time. That said, it can also risk making things more complicated once you get a handle on the development. That's why we choose Tom activity. This will create an activity and some files for us, but it won't add a lot of extra code. Select a name and package name for your new app. The name is what your audience will see when the app is installed on their device. The package name is an internal reference used by Android to distinguish it from other apps. This should be put together using your top-level domain (e.g. . com), domain name and app name. For example: com.androidauthority.sampleapp.If you don't have a domain or a business, just use com followed by something that appeals to you! You also need to decide where you want the files stored and what language you want to code in: Java or Kotlin.Java vs. Kotlin for Android app developmentA of the biggest decisions you need to make as an Android developer is whether you want to learn Kotlin or Java. Both languages are officially supported by Google and Studio, but they have some different differences. Java has been supported by Google the longest and is what developers have used to design Android apps for years. Java is also one of the most requested programming languages in the world, making it a great choice for those who want to begin a career in development. As the oldest Android programming language, there is also a little more support for Java vs. Kotlin, although it's not by much. Kotlin on the other hand has become Google's preferred choice for Android development. This is the standard when you launch a new app, and it will probably become more common going forward. Kotlin is also considerably easier to get a handle on if you're a beginner. For these reasons, Kotlin is probably the language of choice for Android developers who are learning for fun or who have no aspirations to develop for other platforms. But Java makes more sense if you are interested in becoming a professional developer. You can learn more about the two options here: Kotlin vs Java for Android: important differencesMinimum SDKFinally, you should also consider your Minimum SDK. This is the lowest version of Android that your app should support. The lower you make this number, the wider your potential audience becomes. Keep in mind that there is a relatively low adoption rate for the latest versions of Android, so sticking with the latest update will prevent a lot of users from trying your creation. If we leave the version by default (Android 10), then we only support 8.2% of devices! Google: do better. However, you will only be able to access the latest features of Android if you target a newer version. If you like the sound of support chat bubbles, then you will want to stick to the latest version. Step 3: Familiarize yourself with the filesI remember the first time I tried Android app development. I loaded up Android Studio and was immediately amazed by what I saw. There are just so many different files, multiple types of code, folders and more! These were worlds away from the single blank file I was used to working with in Python or even QBasic (anyone remember QBasic??). This can be pretty scary, but here's what you need to know. The file that is open is MainActivity.java or MainActivity.kt. This is the most important logic file for the activity that should define how your app behaves. Look left and you'll see that this file is found in: MyApplication &gt; app &gt; src &gt; main &gt; java &gt; com &gt; company name &gt; myapplication. The folders used are important for Android app development as they help Android Studio and Gradle find everything and build it correctly (more on Gradle in an instant). Suffice to say, you can't just rename these as you please! You will notice that there is already some code on the front page. This is what we call the default code, which means that it For example, code that's almost identical across different app projects and needed to make basic features work. Standard plate code is what you will find yourself writing out over and over again! One of the advantages of Kotlin is that it requires less standard plate, which means you have less code on the screen if that's what you choose. Introducing layout files The role of this code is to tell Android where the associated layout file is. A layout file is slightly different from a Kotlin/Java file. This defines the way an activity looks and allows you to add things like buttons, text, and browser windows. You can find this file in: MyApplication &gt; app &gt; src &gt; res &gt; layout. It will be called activity_main.xml. Note that files stored in the resource folder cannot use uppercase letters. they must use the underline symbol to distinguish between different words. Double-click this file and it will open in the main window where you are editing your code. Note that you can switch between the open files using tabs along the top. You can view this file through code view, Design view, or a split view that shows these windows side-by-side. There are buttons to change mode at the top right. In Design view, you can actually drag and drop various widgets on the screen. The code view shows you a load of the XML script. When you add new widgets through the Design view, this script is updated. Likewise, you can adjust the properties of widgets (called views) in here and see them reflected in real time via the code view. In the vast majority of apps, you'll need to create a new Java/Kotlin file and a corresponding XML file each time you want a new activity. And for those who were wondering: yes, that means you have to learn either Kotlin or Java and XML. This is a bit of a headache, but it actually simplifies the process in the long run. For an introduction to using XML, check out this guide: To get a handle on the different views and what they do: Building your Android UI: Everything you need to know about ViewsThe other files and foldersThere are many more files and folders here though, so what do they all do? In truth, you don't need to know what all this is. But some things that are useful to know about:Android Manifest: This is an XML file in the res folder that defines important features of your app. It includes the orientation of the app, the activities you want to be included in it, the version, etc. For more, read: xml: everything you need to knowDrawable: This folder is available in res. This is where you want to put things like pictures that you want to refer to later. Values: This resource folder is a useful place to store values that are used globally across your app. For example, you can include color codes (making it easy for you to change the look of your entire app) or strict (words). You must these values in individual XML files XML files XML files Colors.xml.Gradle: Gradle is the tool that takes all your files and bundles them into a usable APK for testing. It is also useful for generating previews etc. You don't have to worry about the files in here often, but if you want to add an addiction, this is where you want to do it. Dependencies are external libraries that give you access to additional features from your own code. Learn more about Gradle and how it works here: Introducing Gradle to new Android developers - Master builderStep 4: Test your appThe first thing you need to do when you familiarise yourself with a new programming language is to create an app that says Hello World Fortunately this is very easy in this case to see as that's what the code that's already here does! If you look at XML, it contains a small label that just says: Hello World! If you look at the controls along the top, you'll see that there's a little green game arrow. To the left of this is a drop-down menu with a phone name in it. Once you installed Android Studio, this should also have installed an Android system image along with Virtual Device Manager. In other words, you should already have an Android emulator created and ready to go! By clicking on this green arrow, you will be able to start it and test your app! Note that this will also let you use the imitated phone as if it were a real device. You can change the settings for your virtual device - such as screen size, Android version, space &gt; etc. You can also download new imagery here. Make sure your virtual device meets or exceeds the smallest SDK you put at the beginning. You can also try connecting a physical device to your computer and using it to test your new app. You'll need to turn on developer settings yourself, and activate USB Debugging.How to enable Developer Settings on your Android DeviceStep 5: Do one thing! The best way to learn Android app development is by doing! This means that you need to have a plug on editing the code in front of you, to see if you can do it do something new. Changing the message that appears is as simple as going into your XML and changing the line that says Hello World! in Howdy World! But what if you want to add some kind of interactive element to your creation? In this case, you can decide to let the user click the button to change the text. First, place this line inside the TextView tag in your activity_main.xml:android:id =@+id/helloButton android:onClick =onHelloButtonClickThis will give the text label the name helloButton and will indicate that the method onHelloButtonClick will refer to this view. We will add that to our code in a moment. Now you can add the following code to your MainActivity. If the text appears in red as you type it, means that you need to import this code from the Android SDK. Click on the red text and then press Alt + Enter and Android Studio will do this for you automatically. In short, this tells Android that you are referring to a library that is part of the Android SDK. (The following example is written in Java.) public class MainActivity expands AppCompatActivity { TextView helloButton;     @Override invalid onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);         setContentView(R.layout.activity_main);         helloButton = (TextView) findViewById(R.id.helloButton);         } public void onHelloButtonClick (View v) { helloButton.setText(Howdy World!);     } }In this code sample, we first create an object in the helloButton code. Inside our onCreate method (all within curly brackets) we then tell Android that this object represents the button in our layout file. The code you place here is what will run first when you launch an app. Next, we create the method that runs when someone clicks the button. When this happens, we can then change the text of that button. Note that once again, Kotlin requires significantly fewer lines to achieve the same! Run the app and you should now see that when you click the button, the text changes! This is a very simple app, but it shows the basics of how Android app development works. In general, you want to create new on-screen elements in the layout file, and then define how they behave in the associated Java or Kotlin file. As you become more advanced, start manipulating and storing data. To do this, use variables that contain numbers and strings (words). We have Java tutorials that will help you get started: Java tutorial for beginners: write a simple app without previous experienceDown you have read through it, you will have a basic idea of how Java works and all that is left is to learn how you can apply these skills to Android app development. To that end, a great strategy is to choose a project and then work on it. And wouldn't you know it: we have a ton of great projects to try out! Here are just a few: The key is not to try to learn all Android app development, but to set your sights on a realistic first project. You keep learning when you add new features and want to do new things, and when you have a goal, your learning becomes fun and structured. Before you know it, you'll be a pro! Pro!