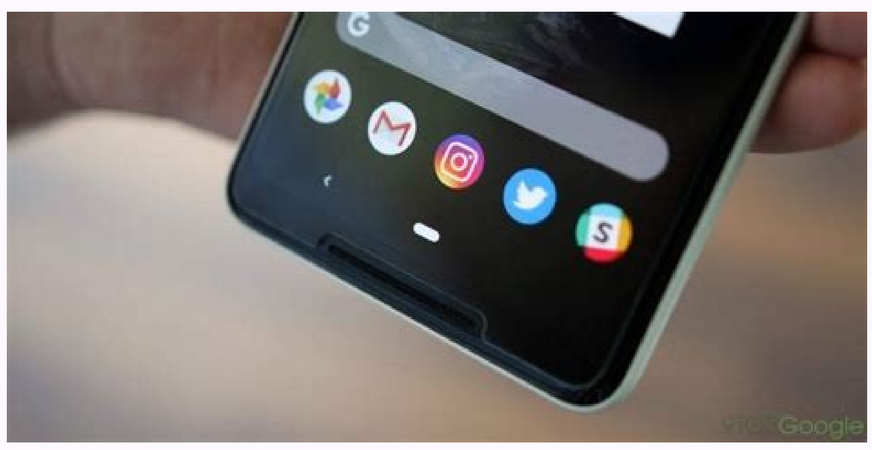
How to add back button in android studio

Continue



How to add back button in action bar android studio. How to use back button in android studio. How to add back button in toolbar android studio. How to add back button in actionbar in android studio. How to add action button in android studio. How to add border to button in android studio.

A drop-down or drop-down menu, also known as a spinner, is one of the most important elements of an application's user interface. In this tutorial, I'll show you how to add it to your Android application using Java. Drop-down menus organize the application and enhance the user experience. Almost every app has a drop-down menu built into its interface. Although adding a dropdown menu to your Android app is as easy as dragging and dropping, it can definitely be difficult, especially if you're new to Android Studio. If you're a beginner developer building your first apps in Android Studio, this guide is for you. Setting up the environment After creating the project in Android Studio, open the following files: res/layout/activity_main.xml res/values/strings.xml app/java/your.project.name/MainActivity.java (MainActivity. java is opened by default when you create project.) Once all the files are open, your IDE should look like this: Adding the Dropdown Menu Layout Now it's time to add the dropdown menu layout. Android Studio adds layouts to layout XML files. To do this, call the activity_main.xml file. In the activity_main.xml file, open the Design tab. It can be found in the upper right corner of the IDE. Android dropdown menus in Android Studio are added using spinners. If your app screen has standard text, go back to the code section and remove all TextViews. Now select Container from the design palette. There you will find spinners. Please note that we are using Android Studio 4.2.2. In older versions of Android Studio, the spinner may be located in the widgets section. If you can't find it, just click on the search icon and find the cutter. Once you find a cutter, drag it around the mobile app. Android Studio will run the appropriate code for you, and you can check it later by returning to the code screen. Depending on where you drop the spinner,the code should look something like this: Spinner Application android:id="@+id/spinner_languages" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_marginBottom="338dp" android:spinnerMode="dropdown": layout_constraintBottom_toBottomOf = "parent" app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent" /> However, Android Studio allows you to customize the dropdown by changing its height, width, and margins in the attributes panel without having to code everything from scratch. Once you've done that, make sure your spinnerMode is set to dropdown in order to create the dropdown menu. You can find this setting in the attributes panel. Placing a drop-down menu on the application screen is quite simple. The Android Studio GUI will give you all the constraints to determine where the drop down menu is. When you're happy with the style of the dropdown menu, switch to code view and edit the spinner ID. This will be needed later when we integrate the Spinner with the Java file. The Spinner ID is in the first line of the tag. Enter an identifier that you will remember for use elsewhere in your application code. Finally, go back to the design section and click the Infer Constraints button - which I personally call the "magic button" - at the top to take care of all the missing constraints in our code: Adding Items to the Dropdown Now that you've added the dropdown to the screen, time is full a set of options from which the user can choose. To do this, you need to open the strings.xml file. This file will initially be empty and should look something like this: To add items to an Android dropdown menu, you must declare an array of strings and name it. Declare an array of strings below the already declared string using the following syntax: dropDownMenuExample C++ Java JavaScript VisualBasic An array of strings must be declared within a resource tag . Otherwise, a syntax error will be displayed. You can also declare a string array in the main Java file, but putting it in a separate XML file increases code reuse and improves your application. There is no limit to the number of items you can have in the drop-down menu. Calling the counter in a Java file Before we start coding, import the following classes into your code: import androidx.appcompat.app.AppCompatActivity; import android.os.Bundle; import android.view.View; import android.widget.AdapterView; import android.widget.ArrayAdapter; import android.widget.Spinner; import android.widget.Toast; It's best to import these classes early to avoid syntax errors later. However, if you still get a syntax error, you can always hover over it and then press Alt-Enter to import the appropriate class for your code. In order to pass the Android drop down menu to Java, you need to define a Spinner object. . Use the class name Spinner and name the object accordingly. Then create an instance of the spinner by finding it with the same ID you provided in the activity_main.xml file: Spinner spinnerLanguages=findViewById(R.id.spinner_languages); In the next step, you need to create an ArrayAdapter. The ArrayAdapter will be responsible for displaying each item in the array of language strings on the screen when accessing the Java dropdown menu. ArrayAdapteradapter=ArrayAdapter.createFromResource(to, R.array.languages, android.R.layout.simple_spinner_item); adapter.setDropDownViewResource(android.R.layout.simple_spinner_item); createFromResources() is a built-in method of the ArrayAdapter class that takes three input parameters: application environment - inside the activity just useThe StringArray layout type that you specified in the strings.xml file. For this particular example, we'll use a basic rotation layout. The above adapter is useless unless it is added to our spinner.). So set the spinner to use this adapter. spinnerLangauges.setAdapter(adapter); Once the ArrayAdapter is declared and successfully bound to the spinner, you've successfully integrated your first Android dropdown into your application. You can now run the application in the emulator. It looks something like this: With the Java dropdown set up, you can now play around with it a bit in the activity_main.xml file. Try changing the height attribute and see how the dropdown moves around the screen. Summary You can add a dropdown menu to your Android app in a few simple steps. Beginners need to edit XML files. Integrate a drop-down menu into them using the drag-and-drop feature in Android Studio. You then need to create an array of strings to add all the relevant items to the dropdown list. Then you need to get spinner instance along with ArrayAdapter in main java. Finally, set up the cutter to use this adapter. Because all three code files run simultaneously, the first Android app will have a fully functional Android dropdown menu. The maximum number of comparable products has been reached. Products cannot be compared. To compare different types of products, click Clear All. Adding an external library in Android Studio is a very common thing, but it is difficult for most beginners or novices. Some of you have had to go through this. Whenever you want to add an external library and try to do it with Maven (File > Project Structure > Dependencies) and you try to add your dependencies but when you click on gradle sync it fails. So now you want to quickly build an app and you get an error message: Error: There was a problem configuring.":app" Many external libraries are available to simplify many specific tasks with APIs and classes. So including all these libraries in our project will really help you. Many developers often don't know how to add these libraries to their projects. This article will be useful to all these people. In this article, we will see different ways to add library projects to our project. But before we get to that, let's take a look: we can use Gradle to add library projects with a single line of code. The Android Studio project is modular and we have a main module called "Application". A project can have several modules. We can connect them together using Gradle, this connection is called dependency. Now let's go back to our main program. Almost every known Android library is available in the Maven repository, and installing it requires only one line of code in app/build.gradle:dependencies {comile â com.jakewharton:butterknife:6.0.0â}Let's make an external library. Add the library to our project: Step 1. Create a new project with Android Studio and name it whatever you want (GFG in this example) and hit the "Finish" button. Step 2. The initial structure of the project created by Android Studio looks like this: Step 3: In the root directory (GFG/) create a new folder: /libs, where we will put external libraries (this step is not necessary - just keep the project structure cleaner). Step 4. Paste your library into the newly created /libs folder. In this example, we will use the PagerSlidingTabStrip library (just download the ZIP from GitHub, rename the library directory to "PagerSlidingTabStrip" and copy). Our new project structure should look like this: Step 5. Edit the settings.gradle file to add the library to include. If you're using a custom path like me, you'll also need to define our library project directory. The entire settings.gradle file should look like thisbelow: enable â:appâ, â:PagerSlidingTabStripâproject(â:PagerSlidingTabStripâ).projectDir = new File(âlibs/PagerSlidingTabStripâ)Step 5.1: If you encounter default configuration error, try this instead of step 5, enable â:appâ, enable â : libs :PagerSlidingTabStrip. Step 6: In app/build.gradle, add our library project as a dependency: â:PagerSlidingTabStripâ)} Step 6.1: If you followed step 5.1, do this instead of step 6, Dependencies { compile filetree(dir: âlibsâ, include: [â* .jarâ]) Compile âcom.android.support:appcompat-v7:21.0 .3 Compile the project (â:libs:PagerSlidingTabStripâ) Step 7: If your library project does not have a build.gradle file, you need to create it manually. Step 8: That's it. Just click... Sync project with Gradle. Your library must be available for your project. Method 2: Step 1: Select File > New Module. Click Import Existing Project. Step 2: Select the required library and module. Then click Done. Android Studio imports the library into your project and syncs the gradle files. Step 3: The next step is to add the imported module as a dependency to your project. Right-click on the application folder > "Open module settings". Step 4: Go to “Dependencies” tab > click “+” button -> click “Module Dependency”. The library module is then added to the project dependencies. Step 1 Go to File > Project Structure. Step 2: Then click on “Modules”. In order to import a library with Gradle, you may need to add it to the dependencies section of your build.gradle (module). Method 4: Simply go to Project Structure > Modules (see images below), click the plus button and choose Import Existing Project and import. Then sync your gradleThere is a possibility that an error will occur. Error: SDK build tools version too low (xx.x.x). Required minimum: yy.y.y. Then open the build.gradle file in your library's project directory and update the buildToolsVersion to the recommended version. When importing a library module, select the project module, add a dependency, and then select the imported module. Module.