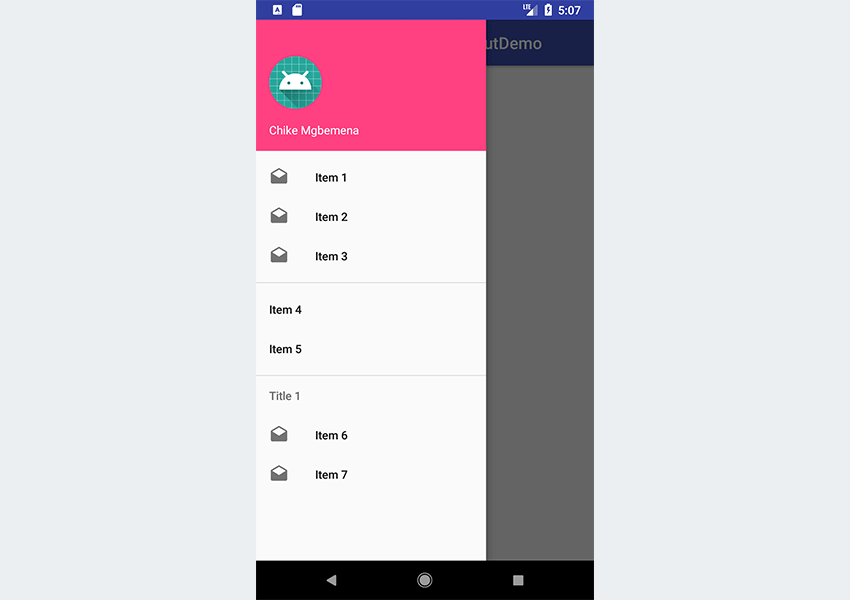
How to Code a Navigation Drawer for an Android App

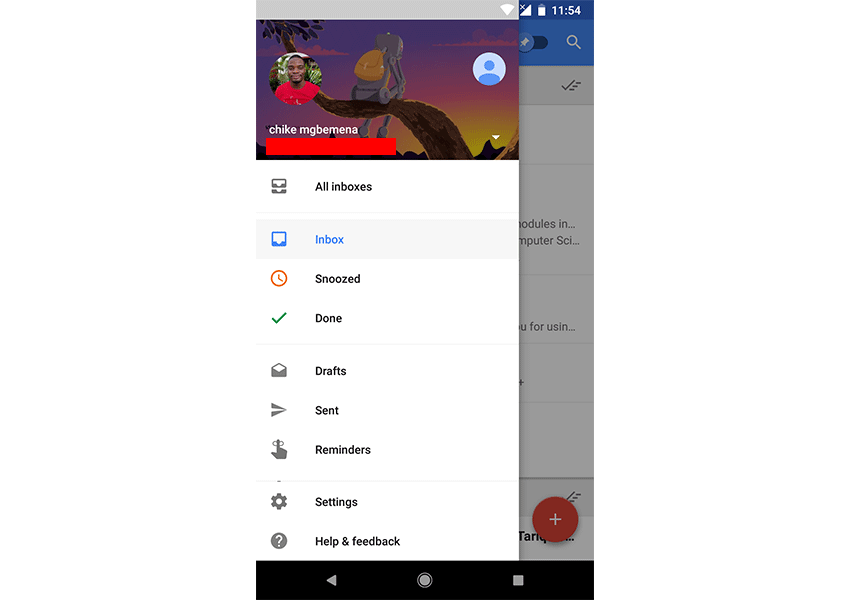
[Android](https://code.tutsplus.com/categories/android)

What You'll Be Creating

The [material design team at Google](https://material.io/guidelines/patterns/navigation-drawer.html) defines the functionality of a navigation drawer in Android as follows:

*The navigation drawer slides in from the left and contains the navigation destinations for your app.*

An example of a popular Android app that implements the navigation drawer is the Inbox app from Google, which uses a navigation drawer to navigate to different sections of the application. You can check it yourself by downloading the Inbox app from the Google Play store, if you don't already have it on your device. The screenshot below shows Inbox with the navigation drawer pulled open.



The user can view the navigation drawer when they swipe a finger from the left edge of the activity. They can also find it from the home activity (the top level of the app), by tapping the app icon (also known as the "hamburger" menu) in the action bar.

Note that if you have many different destinations (more than six, say) in your app, it's recommended that you use a navigation drawer.

In this post, you'll learn how to display navigation items inside a navigation drawer in Android. We'll cover how to use the DrawerLayout and NavigationView API to perform this task. For a bonus, you'll also learn how to use the Android Studio templates feature to quickly bootstrap your project with a navigation drawer.

A sample project (in Kotlin) for this tutorial can be found on our GitHub repo so you can easily follow along.

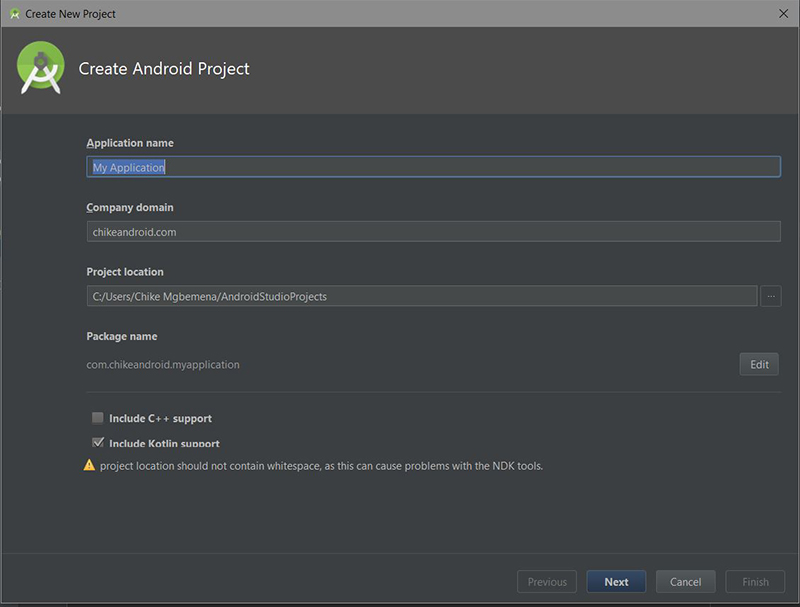
**Prerequisites**

To be able to follow this tutorial, you'll need:

* [Android Studio](https://developer.android.com/studio/index.html) 3.0 or higher
* [Kotlin plugin](https://plugins.jetbrains.com/plugin/6954-kotlin) 1.1.51 or higher

**1. Create an Android Studio Project**

Fire up Android Studio and create a new project (you can name it NavigationDrawerDemo) with an empty activity called MainActivity. Make sure to also check the **Include Kotlin support** check box.



Advertisement

**2. Adding the DrawerLayout and NavigationView**

To begin using DrawerLayout and NavigationView in your project, you'll need to import the design support and also the Android support artifact. So add these to your module's **build.gradle** file to import them.

|  |  |
| --- | --- |
| 1  2  3  4 | dependencies {      implementation 'com.android.support:design:27.0.2'      implementation 'com.android.support:support-v4:27.0.2'  } |

Also, include both the DrawerLayout widget and also the NavigationView widget in your **res/layout/activlty\_main.xml** file.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <?xml version="1.0" encoding="utf-8"?>  <android.support.v4.widget.DrawerLayout          xmlns:android="<https://schemas.android.com/apk/res/android>"          xmlns:app="<http://schemas.android.com/apk/res-auto>"          xmlns:tools="<http://schemas.android.com/tools>"          android:id="@+id/drawer\_layout"          android:layout\_width="match\_parent"          android:layout\_height="match\_parent"          tools:openDrawer="start">        <include              layout="@layout/app\_bar\_main"              android:layout\_width="match\_parent"              android:layout\_height="match\_parent" />        <android.support.design.widget.NavigationView              android:id="@+id/nav\_view"              android:layout\_width="wrap\_content"              android:layout\_height="match\_parent"              android:layout\_gravity="start"              app:headerLayout="@layout/nav\_header\_main"              app:menu="@menu/activity\_main\_drawer" />    </android.support.v4.widget.DrawerLayout> |

Here we created a DrawerLayout widget with the id drawer\_layout. The tools:openDrawer property is used to display the navigation drawer when the XML layout is open in Android Studio design view.

The [official documentation](https://developer.android.com/reference/android/support/v4/widget/DrawerLayout.html) says the following about DrawerLayout:

*DrawerLayout acts as a top-level container for window content that allows for interactive "drawer" views to be pulled out from one or both vertical edges of the window.*

After adding the DrawerLayout widget, we included a child layout which points to @layout/app\_bar\_main.

Here is my **app\_bar\_main.xml** resource file. This file simply has a CoordinatorLayout, an AppBarLayout, and a Toolbar widget.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | <?xml version="1.0" encoding="utf-8"?>  <android.support.design.widget.CoordinatorLayout          xmlns:android="<http://schemas.android.com/apk/res/android>"          xmlns:app="<http://schemas.android.com/apk/res-auto>"          xmlns:tools="<http://schemas.android.com/tools>"          android:layout\_width="match\_parent"          android:layout\_height="match\_parent"          tools:context=".MainActivity">        <android.support.design.widget.AppBarLayout              android:layout\_width="match\_parent"              android:layout\_height="wrap\_content"              android:fitsSystemWindows="true"              android:theme="@style/AppTheme.AppBarOverlay">            <android.support.v7.widget.Toolbar                  android:id="@+id/toolbar\_main"                  android:layout\_width="match\_parent"                  android:layout\_height="?attr/actionBarSize"                  android:background="?attr/colorPrimary"                  app:layout\_scrollFlags="scroll|enterAlways"                  app:popupTheme="@style/AppTheme.PopupOverlay" />        </android.support.design.widget.AppBarLayout>  </android.support.design.widget.CoordinatorLayout> |

Finally, we created a NavigationView widget. The [official documentation](https://developer.android.com/reference/android/support/design/widget/NavigationView.html) says the following about NavigationView:

*NavigationView represents a standard navigation menu for application. The menu contents can be populated by a menu resource file.*

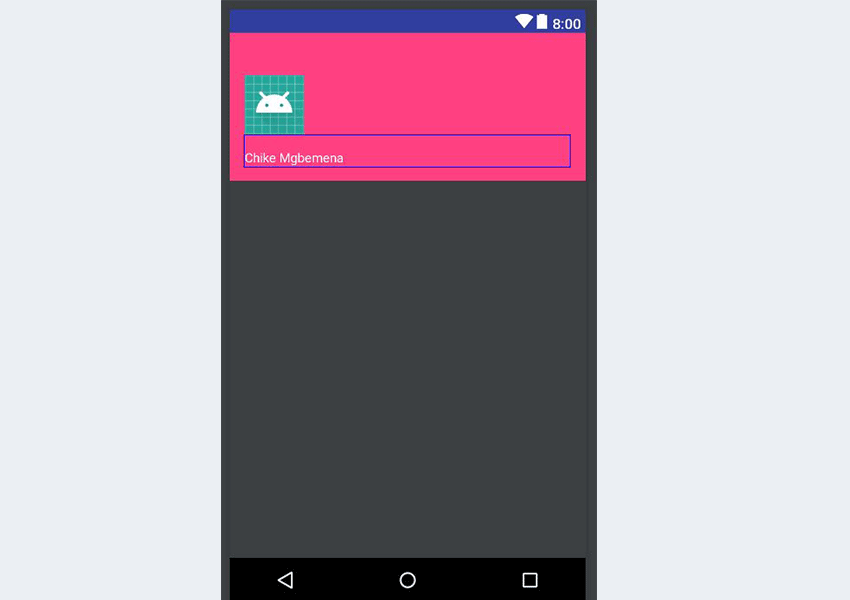
In the NavigationView XML widget, you can see that we added an android:layout\_gravity attribute with value start. This is used to position the drawer—you want the drawer to come out from left or right (the start or end on platform versions that support layout direction). In our own case, the drawer will come out from the left.

We also included an app:headerLayout attribute which points to @layout/nav\_header\_main. This will add a View as a header of the navigation menu.

Here is my **nav\_header\_main.xml** layout resource file:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"                android:id="@+id/nav\_header"                android:layout\_width="match\_parent"                android:layout\_height="160dp"                android:background="@color/colorAccent"                android:clickable="true"                android:focusable="true"                android:foreground="?attr/selectableItemBackgroundBorderless"                android:gravity="bottom"                android:orientation="vertical"                android:padding="16dp"                android:theme="@style/ThemeOverlay.AppCompat.Dark">        <ImageView              android:id="@+id/nav\_header\_imageView"              android:layout\_width="64dp"              android:layout\_height="64dp"              android:src="@mipmap/ic\_launcher" />        <TextView              android:id="@+id/nav\_header\_textView"              android:layout\_width="match\_parent"              android:layout\_height="wrap\_content"              android:paddingTop="16dp"              android:text="Chike Mgbemena"              android:textAppearance="@style/TextAppearance.AppCompat.Body1" />  </LinearLayout> |

This layout file simply has a LinearLayout, an ImageView, and a TextView.



To include the menu items for the navigation drawer, we can use the attribute app:menu with a value that points to a menu resource file.

|  |  |
| --- | --- |
| 1  2 | <android.support.design.widget.NavigationView          app:menu="@menu/activity\_main\_drawer" /> |

Here is the **res/menu/activity\_main\_drawer.xml** menu resource file:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | <?xml version="1.0" encoding="utf-8"?>  <menu xmlns:android="<http://schemas.android.com/apk/res/android>">      <group>          <item android:id="@+id/nav\_item\_one"                android:icon="@drawable/ic\_drafts\_black\_24dp"                android:title="Item 1" />          <item android:id="@+id/nav\_item\_two"                android:icon="@drawable/ic\_drafts\_black\_24dp"                android:title="Item 2" />          <item android:id="@+id/nav\_item\_three"                android:icon="@drawable/ic\_drafts\_black\_24dp"                android:title="Item 3" />      </group>        <group android:id="@+id/group\_menu">          <item android:id="@+id/nav\_item\_four"                android:title="Item 4" />          <item android:id="@+id/nav\_item\_five"                android:title="Item 5" />      </group>        <item android:title="Title 1">          <menu>              <item android:id="@+id/nav\_item\_six"                    android:icon="@drawable/ic\_drafts\_black\_24dp"                    android:title="Item 6" />              <item android:id="@+id/nav\_item\_seven"                    android:icon="@drawable/ic\_drafts\_black\_24dp"                    android:title="Item 7" />          </menu>      </item>  </menu> |

Here we have defined a Menu using the <menu> which serves as a container for menu items. An <item> creates a MenuItem, which represents a single item in a menu.

We then defined our first menu group using the <group>. A <group> serves as an invisible container for <item> elements—menu items in our case. Each of the <item> elements has an id, an icon, and a title. Note that a horizontal line will be drawn at the end of each <group> for us when shown in the navigation drawer.

A <item> can also contain a nested <menu> element in order to create a submenu—we did just this in our last <item>. Notice that this last <item> has a title property.

Note that when showing the navigation list items from a menu resource, we could use a ListView instead. But, by configuring the navigation drawer with a menu resource, we get the material design styling on the navigation drawer for free! If you used a ListView, you would have to maintain the list and also style it to meet the recommended [material design specs for the navigation drawer](https://material.io/guidelines/patterns/navigation-drawer.html#navigation-drawer-specs).

**3. Initialization of Components**

Next, we are going to initialize instances of our DrawerLayout and ActionBarDrawerToggle. Initialization is going to happen inside onCreate() in **MainActivity.kt**.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | import android.content.res.Configuration  import android.os.Bundle  import android.support.v4.widget.DrawerLayout  import android.support.v7.app.ActionBarDrawerToggle  import android.support.v7.app.AppCompatActivity  import android.support.v7.widget.Toolbar  import android.view.MenuItem    class MainActivity : AppCompatActivity() {        private lateinit var drawer: DrawerLayout      private lateinit var toggle: ActionBarDrawerToggle        override fun onCreate(savedInstanceState: Bundle?) {          super.onCreate(savedInstanceState)          setContentView(R.layout.activity\_main)            val toolbar: Toolbar = findViewById(R.id.toolbar\_main)          setSupportActionBar(toolbar)            drawer = findViewById(R.id.drawer\_layout)            toggle = ActionBarDrawerToggle(this, drawer, toolbar, R.string.navigation\_drawer\_open, R.string.navigation\_drawer\_close)          drawer.addDrawerListener(toggle)          supportActionBar?.setDisplayHomeAsUpEnabled(true)          supportActionBar?.setHomeButtonEnabled(true)      }        // ...  } |

The [ActionBarDrawerToggle](https://developer.android.com/reference/android/support/v4/app/ActionBarDrawerToggle.html" \t "_blank) sets up the app icon located on the left of the action bar or toolbar to open and close the navigation drawer. To be able to create an instance of ActionBarDrawerToggle, we have to provide the following parameters:

* a parent context—for example, in an Activity you use this, while in a Fragment you call getActivity()
* an instance of the DrawerLayout widget to link to the activity's ActionBar
* the icon to place on top of the app icon to indicate that there is a toggle
* the string resources for the open and close operations respectively (for accessibility)

We invoked the method addDrawerListener() on a DrawerLayout so as to connect an ActionBarDrawerToggle with a DrawerLayout.

Note that we also enable the app icon via setHomeButtonEnabled() and enable it for “up” navigation via setDisplayHomeAsUpEnabled().

We then forward the onPostCreate(), onConfigurationChanged(), and onOptionsItemSelected() activity callback methods on to the toggle:

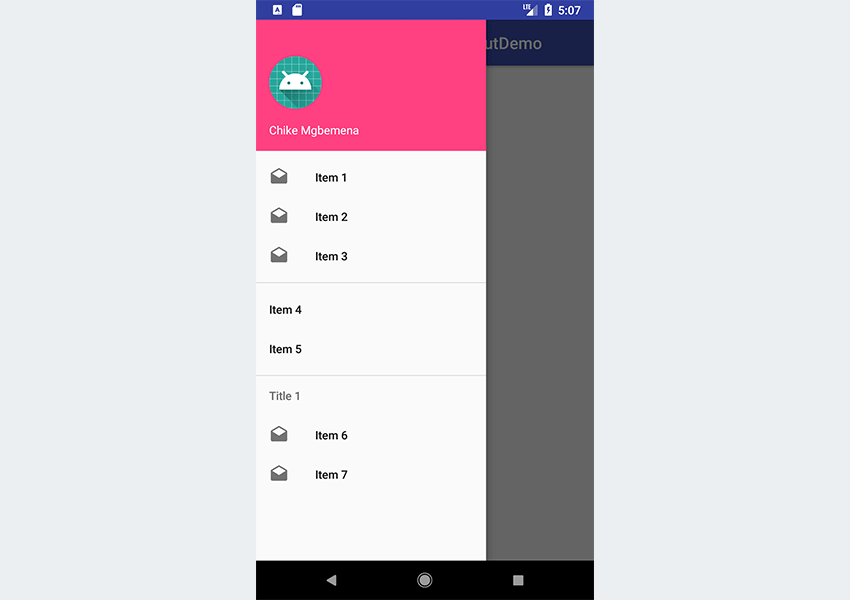
|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | class MainActivity : AppCompatActivity() {        // ...        override fun onPostCreate(savedInstanceState: Bundle?) {          super.onPostCreate(savedInstanceState)          toggle.syncState()      }        override fun onConfigurationChanged(newConfig: Configuration?) {          super.onConfigurationChanged(newConfig)          toggle.onConfigurationChanged(newConfig)      }        override fun onOptionsItemSelected(item: MenuItem?): Boolean {          if (toggle.onOptionsItemSelected(item)) {              return true          }          return super.onOptionsItemSelected(item)      }  } |

Here is what the syncState() does, according to the [official documentation](https://developer.android.com/reference/android/support/v4/app/ActionBarDrawerToggle.html#syncState()):

*Synchronizes the state of the drawer indicator/affordance with the linked DrawerLayout... This should be called from your Activity's [onPostCreate](https://developer.android.com/reference/android/app/Activity.html" \l "onPostCreate(android.os.Bundle)) method to synchronize after the DrawerLayout's instance state has been restored, and any other time when the state may have diverged in such a way that the ActionBarDrawerToggle was not notified. (For example, if you stop forwarding appropriate drawer events for a period of time.)*

**4. Testing the App**

At this point, we can run the app!



As you can see, launching the app will show the “hamburger” navigation drawer icon in the action bar. Try tapping this app icon to open the drawer. Also, clicking on the navigation drawer items won't do anything—we're going to handle that part in the next section.

**5. Handling Click Events**

Now, let's see how to handle click events for each of the items in the navigation drawer. Note that clicking on any item is supposed to take you to a new Activity or Fragment—that's why it's called a navigation drawer!

First, your activity needs to implement the NavigationView.OnNavigationItemSelectedListener.

|  |  |
| --- | --- |
| 1  2  3 | class MainActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener {      // ...  } |

By implementing this contract or interface, we must now override the only method: onNavigationItemSelected().

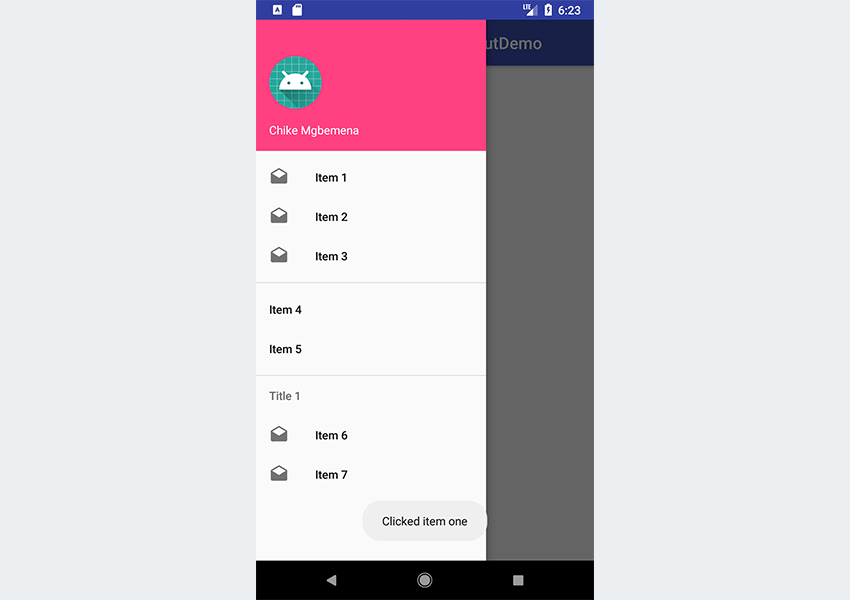
|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | class MainActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener {        // ...        override fun onNavigationItemSelected(item: MenuItem): Boolean {            when (item.itemId) {              R.id.nav\_item\_one -> Toast.makeText(this, "Clicked item one", Toast.LENGTH\_SHORT).show()              R.id.nav\_item\_two -> Toast.makeText(this, "Clicked item two", Toast.LENGTH\_SHORT).show()              R.id.nav\_item\_three -> Toast.makeText(this, "Clicked item three", Toast.LENGTH\_SHORT).show()              R.id.nav\_item\_four -> Toast.makeText(this, "Clicked item four", Toast.LENGTH\_SHORT).show()          }          return true      }  } |

This method is invoked when an item in the navigation menu is selected. We used the when expression to perform different actions based on the menu item that was clicked—the menu item ids serve as constants for the when expression.

Next, we have to initialize our NavigationView and set this listener inside onCreate() of our activity.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class MainActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener {      // ...      override fun onCreate(savedInstanceState: Bundle?) {          // ...          val navigationView: NavigationView = findViewById(R.id.nav\_view)          navigationView.setNavigationItemSelectedListener(this)          // ...      }  // ... |

Run the project again!



When you click on some items, a toast message is displayed—just what we expected. But remember that clicking on an item should take the user to a new Activity or Fragment (we ignored this here for brevity's sake).

You will notice that when you click on an item, the drawer still remains. It would be better if it closed automatically anytime an item was clicked. Let's see how to do that.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | override fun onNavigationItemSelected(item: MenuItem): Boolean {        when (item.itemId) {          R.id.nav\_item\_one -> Toast.makeText(this, "Clicked item one", Toast.LENGTH\_SHORT).show()          R.id.nav\_item\_two -> Toast.makeText(this, "Clicked item two", Toast.LENGTH\_SHORT).show()          R.id.nav\_item\_three -> Toast.makeText(this, "Clicked item three", Toast.LENGTH\_SHORT).show()          R.id.nav\_item\_four -> Toast.makeText(this, "Clicked item four", Toast.LENGTH\_SHORT).show()      }      drawer.closeDrawer(GravityCompat.START)      return true  } |

To close the drawer after a link has been clicked, simply invoke closeDrawer() on an instance of DrawerLayout and pass GravityCompat.START to the method.

Run the project one more time and see the result!

**6. Handling the Back Button Being Pressed**

When the drawer is open, it would be a better user experience not to close the home activity if the **Back** button is pressed. This is the way popular apps like Google's Inbox app work.

So, when the drawer is open and the **Back**button is pressed, only close the drawer instead of the current home activity. Then, if the user presses the **Back** button again, the home activity should be closed.

Here's how we can achieve this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | override fun onBackPressed() {      if (drawer.isDrawerOpen(GravityCompat.START)) {          drawer.closeDrawer(GravityCompat.START)      } else {          super.onBackPressed()      }  } |

Run the project again and test it out!