# How do I use DrawerLayout to display over the ActionBar/Toolbar and under the status bar?



I've seen in the new material design Side Nav spec that you can display the drawer over the action bar and behind the status bar. How can I implement this?

android navigation-drawer appcompat

edited Oct 18 '14 at 14:52
a.bertucci
5,860 ● 1 ● 17 ● 29

asked Oct 18 '14 at 14:35
Chris Banes
14.4k ● 10 ● 38 ● 42

## 9 Answers

New functionality in the framework and support libs allow exactly this. There are three 'pieces of the puzzle':

1. Using Toolbar so that you can embed your action bar into your view hierarchy.
2. Making DrawerLayout `fitsSystemWindows` so that it is layed out behind the system bars.
3. Disabling `Theme.Material` 's normal status bar coloring so that DrawerLayout can draw there instead.

I'll assume that you will use the new appcompat.

First, your layout should look like this:

```
<!-- The important thing to note here is the added fitSystemWindows -->
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/my_drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">

    <!-- Your normal content view -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <!-- We use a Toolbar so that our drawer can be displayed
             in front of the action bar -->
        <android.support.v7.widget.Toolbar
            android:id="@+id/my_awesome_toolbar"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:minHeight="?attr/actionBarSize"
            android:background="?attr/colorPrimary" />

        <!-- The rest of your content view -->

    </LinearLayout>

    <!-- Your drawer view. This can be any view, LinearLayout
         is just an example. As we have set fitSystemWindows=true
         this will be displayed under the status bar. -->
    <LinearLayout
        android:layout_width="304dp"
        android:layout_height="match_parent"
        android:layout_gravity="left|start"
        android:fitsSystemWindows="true">

        <!-- Your drawer content -->

    </LinearLayout>

</android.support.v4.widget.DrawerLayout>
```

Then in your Activity/Fragment:

```
public void onCreate(Bundled savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
    // Your normal setup. Blah blah ...

    // As we're using a Toolbar, we should retrieve it and set it
    // to be our ActionBar
    Toolbar toolbar = (...) findViewById(R.id.my_awesome_toolbar);
    setSupportActionBar(toolbar);

    // Now retrieve the DrawerLayout so that we can set the status bar color.
    // This only takes effect on Lollipop, or when using translucentStatusBar
    // on KitKat.
    DrawerLayout drawerLayout = (...) findViewById(R.id.my_drawer_layout);
    drawerLayout.setStatusBarBackgroundColor(yourChosenColor);
}
```

Then you need to make sure that the DrawerLayout is visible behind the status bar. You do that by changing your values-v21 theme:

**values-v21/themes.xml**

```
<style name="Theme.MyApp" parent="Theme.AppCompat.Light.NoActionBar">
    <item name="android:windowDrawsSystemBarBackgrounds">true</item>
    <item name="android:statusBarColor">@android:color/transparent</item>
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

Note: If a `<fragment android:name="fragments.NavigationDrawerFragment">` is used instead of

```
<LinearLayout
    android:layout_width="304dp"
    android:layout_height="match_parent"
    android:layout_gravity="left|start"
    android:fitsSystemWindows="true">

    <!-- Your drawer content -->

</LinearLayout>
```

the actual layout, the desired effect will be achieved if you call `fitsSystemWindows(boolean)` on a view that you return from `onCreateView` method.

```
@Override
public View onCreateView(LayoutInflater inflater,
                         ViewGroup container,
                         Bundle savedInstanceState) {
    View rootView = inflater.inflate(
        R.layout.fragment_navigation_drawer, container, false);
    mDrawerListView.setFitsSystemWindows(true);
    return mDrawerListView;
}
```

edited Apr 5 at 19:20

JJD
11.4k ● 16 ● 75 ● 147

answered Oct 18 '14 at 14:35

Chris Banes
14.4k ● 10 ● 38 ● 42

3 Yes, the `DrawerLayout` needs it to to be layed out behind the system bars. You then need to set it on the drawer view so that `DrawerLayout` lays it out within the window insets. – Chris Banes Oct 18 '14 at 14:53

8 Can't get this to work at all. With the code above in a new demo application the result looks similar to what @ScootrNova describes. See this Screenshot: imgur.com/QLk3syt – Michael Schmidt Oct 30 '14 at 10:32

6 Looks like you actually need to use a negative `layout_marginTop` in your drawer content's layout's root. Otherwise, your drawer content's layout's background will be drawn on top of the status bar, and everything else will be pushed down. This however seems like a kinda gross solution, as far as I know there is no ?attr/statusBarSize or anything like that supplied by Android. – Charles Madere Oct 31 '14 at 7:33

17 To get the effect where the drawer is visible through the status bar, I ended up setting android:fitsSystemWindows to false on the DrawerLayout, android:fitsSystemWindows to true on my main content layout (the layout containing the Toolbar), and adding <item name="android:windowTranslucentStatus">true</item> to my theme - on Lollipop that is – Jakob Nov 6 '14 at 0:20

7 This answer is incomplete. You also need to wrap your navigation drawer inside a ScrimLayout. See stackoverflow.com/a/26926998/174149 – markus-hi Nov 14 '14 at 9:45



## EDIT: The new Design Support Library supports this and the previous method is no longer required.

This can now be achieved using the new Android Design Support Library.

You can see the Cheesesquare sample app by Chris Banes which demos all the new features.

---

### Previous method:

Since there is no complete solution posted, here is the way I achieved the desired result.

First include a ScrimInsetsFrameLayout in your project.

```
/*
* Copyright 2014 Google Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

/**
* A layout that draws something in the insets passed to
* {@link #fitSystemWindows(Rect)}, i.e. the area above UI chrome
* (status and navigation bars, overlay action bars).
*/
public class ScrimInsetsFrameLayout extends FrameLayout {
    private Drawable mInsetForeground;

    private Rect mInsets;
    private Rect mTempRect = new Rect();
    private OnInsetsCallback mOnInsetsCallback;

    public ScrimInsetsFrameLayout(Context context) {
        super(context);
        init(context, null, 0);
    }

    public ScrimInsetsFrameLayout(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context, attrs, 0);
    }

    public ScrimInsetsFrameLayout(
        Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(context, attrs, defStyle);
    }

    private void init(Context context, AttributeSet attrs, int defStyle) {
        final TypedArray a = context.obtainStyledAttributes(attrs,
                R.styleable.ScrimInsetsView, defStyle, 0);
        if (a == null) {
            return;
        }
        mInsetForeground = a.getDrawable(
            R.styleable.ScrimInsetsView_insetForeground);
        a.recycle();

        setWillNotDraw(true);
    }

    @Override
    protected boolean fitSystemWindows(Rect insets) {
        mInsets = new Rect(insets);
        setWillNotDraw(mInsetForeground == null);
        ViewCompat.postInvalidateOnAnimation(this);
        if (mOnInsetsCallback != null) {
            mOnInsetsCallback.onInsetsChanged(insets);
        }
        return true; // consume insets
    }

    @Override
    public void draw(Canvas canvas) {
        super.draw(canvas);

        int width = getWidth();
        int height = getHeight();
        if (mInsets != null && mInsetForeground != null) {
            int sc = canvas.save();
            canvas.translate(getScrollX(), getScrollY());

            // Top
            mTempRect.set(0, 0, width, mInsets.top);
            mInsetForeground.setBounds(mTempRect);
            mInsetForeground.draw(canvas);

            // Bottom
            mTempRect.set(0, height - mInsets.bottom, width, height);
            mInsetForeground.setBounds(mTempRect);
            mInsetForeground.draw(canvas);
```

```
            // Left
            mTempRect.set(
                0,
                mInsets.top,
                mInsets.left,
                height - mInsets.bottom);
            mInsetForeground.setBounds(mTempRect);
            mInsetForeground.draw(canvas);

            // Right
            mTempRect.set(
                width - mInsets.right,
                mInsets.top, width,
                height - mInsets.bottom);
            mInsetForeground.setBounds(mTempRect);
            mInsetForeground.draw(canvas);

            canvas.restoreToCount(sc);
        }
    }

    @Override
    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        if (mInsetForeground != null) {
            mInsetForeground.setCallback(this);
        }
    }

    @Override
    protected void onDetachedFromWindow() {
        super.onDetachedFromWindow();
        if (mInsetForeground != null) {
            mInsetForeground.setCallback(null);
        }
    }

    /**
     * Allows the calling container to specify a callback for custom
     * processing when insets change (i.e. when {@link #fitSystemWindows(Rect)}
     * is called. This is useful for setting padding on UI elements
     * based on UI chrome insets (e.g. a Google Map or a ListView).
     * When using with ListView or GridView, remember to set
     * clipToPadding to false.
     */
    public void setOnInsetsCallback(OnInsetsCallback onInsetsCallback) {
        mOnInsetsCallback = onInsetsCallback;
    }

    public static interface OnInsetsCallback {
        public void onInsetsChanged(Rect insets);
    }
}
```

Then create a styleable so that the `insetForeground` can be set.

**values/attrs.xml**

```
<declare-styleable name="ScrimInsetsView">
    <attr name="insetForeground" format="reference|color" />
</declare-styleable>
```

Update your activity's xml file and make sure `android:fitsSystemWindows` is set to true on both the `DrawerLayout` as well as the `ScrimInsetsFrameLayout`.

**layout/activity_main.xml**

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawerLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".MainActivity">

    <!-- The main content view -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <!-- Your main content -->

    </LinearLayout>

    <!-- The navigation drawer -->
    <com.example.app.util.ScrimInsetsFrameLayout
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:id="@+id/scrimInsetsFrameLayout"
        android:layout_width="320dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="@color/white"
        android:elevation="10dp"
        android:fitsSystemWindows="true"
        app:insetForeground="#4000">
```

```
        <!-- Your drawer content -->

    </com.example.app.util.ScrimInsetsFrameLayout>

</android.support.v4.widget.DrawerLayout>
```

Inside the onCreate method of your activity set the status bar background color on the drawer layout.

**MainActivity.java**

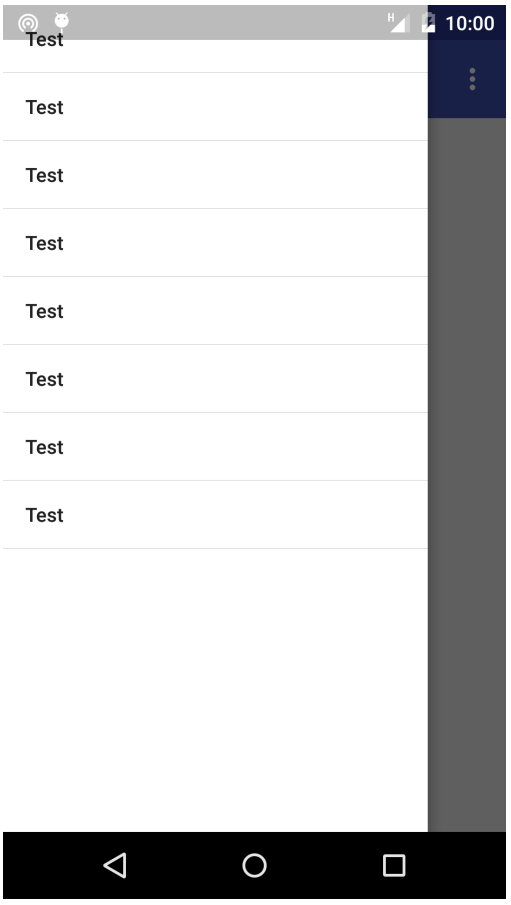
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // ...

    mDrawerLayout = (DrawerLayout) findViewById(R.id.drawerLayout);
    mDrawerLayout.setStatusBarBackgroundColor(
        getResources().getColor(R.color.primary_dark));
}
```

Finally update your app's theme so that the `DrawerLayout` is behind the status bar.

**values-v21/styles.xml**

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <item name="android:windowDrawsSystemBarBackgrounds">true</item>
    <item name="android:statusBarColor">@android:color/transparent</item>
</style>
```

**Result:**




edited Jun 24 at 20:17

answered Nov 26 '14 at 15:46

Suyash
974 ● 3 ● 11

7 This worked the best for me. I spent 2 days to get this thing work and this is the one. This should be marked as Answer. Thank you so much. – kalehv Dec 7 '14 at 21:36

2 I am not using appcompat and this didn't work. :( Is there any guide one getting this working without appcompat? – Jared Rummler Jan 1 at 22:29

2 This is the most perfect answer. I have implemented and tested it in various Android devices from 4.X to 5.X and can confirm that it works perfectly. Thanks a lot. – Aritra Roy Jan 8 at 20:17

1 As I said, it works perfectly, but I have a little problem. The activity with the Navigation Drawer has the transparent status bar, but all other activities have lost the "primaryDark" color from the status bar. How to get that back? – Aritra Roy Jan 11 at 7:09

4 It took me three days to accept that there is currently no other way than using this additional layout wrapped around the drawer fragment. Otherwise the status bar will either be gray (first child's background color) or the drawer will be displayed below the status bar. – Denis Loh Jan 11 at 14:06

With the release of the latest Android Support Library (rev 22.2.0) we've got a Design Support Library and as part of this a new view called NavigationView. So instead of doing everything on our own with the `ScrimInsetsFrameLayout` and all the other stuff we simply use this view and everything is done for us.

## Example

### Step 1

Add the `Design Support Library` to your `build.gradle` file

```
dependencies {
    // Other dependencies like appcompat
    compile 'com.android.support:design:22.2.0'
}
```

### Step 2

Add the `NavigationView` to your `DrawerLayout` :

```
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:app="http://schemas.android.com/apk/res-auto"
     android:id="@+id/drawer_layout"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:fitsSystemWindows="true"> <!-- this is important -->

     <!-- Your contents -->

     <android.support.design.widget.NavigationView
         android:id="@+id/navigation"
         android:layout_width="wrap_content"
         android:layout_height="match_parent"
         android:layout_gravity="start"
         app:menu="@menu/navigation_items" /> <!-- The items to display -->
 </android.support.v4.widget.DrawerLayout>
```

### Step 3

Create a new menu-resource in `/res/menu` and add the items and icons you wanna display:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_home"
            android:icon="@drawable/ic_action_home"
            android:title="Home" />
        <item
            android:id="@+id/nav_example_item_1"
            android:icon="@drawable/ic_action_dashboard"
            android:title="Example Item #1" />
    </group>

    <item android:title="Sub items">
        <menu>
            <item
                android:id="@+id/nav_example_sub_item_1"
                android:title="Example Sub Item #1" />
        </menu>
    </item>

</menu>
```

### Step 4

Init the NavigationView and handle click events:

```
public class MainActivity extends AppCompatActivity {

    NavigationView mNavigationView;
    DrawerLayout mDrawerLayout;

    // Other stuff

    private void init() {
        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        mNavigationView = (NavigationView) findViewById(R.id.navigation_view);
        mNavigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(MenuItem menuItem) {
                mDrawerLayout.closeDrawers();
                menuItem.setChecked(true);
```

```
                switch (menuItem.getItemId()) {
                    case R.id.nav_home:
                        // TODO - Do something
                        break;
                    // TODO - Handle other items
                }
                return true;
            }
        });
    }
}
```

### Step 5

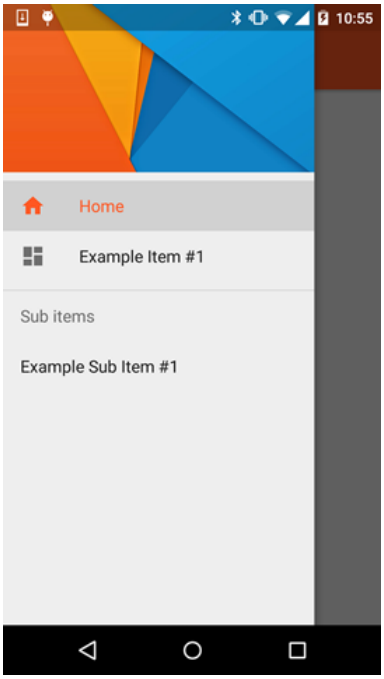
Be sure to set `android:windowDrawsSystemBarBackgrounds` and `android:statusBarColor` in `values-v21` otherwise your Drawer won`t be displayed "under" the StatusBar

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- Other attributes like colorPrimary, colorAccent etc. -->
    <item name="android:windowDrawsSystemBarBackgrounds">true</item>
    <item name="android:statusBarColor">@android:color/transparent</item>
</style>
```

### Optional Step

Add a Header to the NavigationView. For this simply create a new layout and add `app:headerLayout="@layout/my_header_layout"` to the NavigationView.

### Result




### Notes

- The **highlighted color** uses the color defined via the `colorPrimary` attribute
- The **List Items** use the **color** defined via the `textColorPrimary` attribute
- The **Icons** use the **color** defined via the `textColorSecondary` attribute

You can also check the example app by *Chris Banes* which highlights the NavigationView along with the other new views that are part of the Design Support Library (like the *FloatingActionButton*, *TextInputLayout*, *Snackbar*, *TabLayout* etc.)

answered May 29 at 8:58

reVerse
10k ● 12 ● 31 ● 43

1 thanks @reVerse!!. > If you want to update every item and icon in the list from the style files, you can do it using: `<item name="itemTextColor">@color/YOUR_COLOR</item>` `<item name="itemIconTint">@color/YOUR_COLOR</item>` – juancho May 30 at 19:56

1 This should be the accepted answer, great work – Tim Rae Jun 10 at 14:01

The above all approaches are correct and may be working . I have created a working demo following the above guide and tested on 2.x to 5.x

You can clone from Github

The important thing to play around is in Main Activity

```
toolbar = (Toolbar) findViewById(R.id.toolbar);
res = this.getResources();

this.setSupportActionBar(toolbar);
ActionBar actionBar = getSupportActionBar();
actionBar.setDisplayHomeAsUpEnabled(true);
actionBar.setHomeButtonEnabled(true);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
    ScrimInsetsFrameLayout scrimInsetsFrameLayout = (ScrimInsetsFrameLayout)
            findViewById(R.id.linearLayout);
    scrimInsetsFrameLayout.setOnInsetsCallback(this);
}
```

and the call back

```
@Override
public void onInsetsChanged(Rect insets) {
    Toolbar toolbar = this.toolbar;
    ViewGroup.MarginLayoutParams lp = (ViewGroup.MarginLayoutParams)
        toolbar.getLayoutParams();
    lp.topMargin = insets.top;
    int top = insets.top;
    insets.top += toolbar.getHeight();
    toolbar.setLayoutParams(lp);
    insets.top = top; // revert
}
```

Absolutely the Theme for V21 does the magic

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- API 21 theme customizations can go here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/accent_material_light</item>
    <item name="windowActionModeOverlay">true</item>
    <item name="android:windowDrawsSystemBarBackgrounds">true</item>
    <item name="android:statusBarColor">@android:color/transparent</item>
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

and the ScrimInsetsFrameLayout

Now this come more easy with new Design Support library

```
compile 'com.android.support:design:22.2.0'
```

clone from @Chris Banes https://github.com/chrisbanes/cheesesquare

edited Jun 5 at 10:04

answered Dec 30 '14 at 12:01

Vipinhelloindia

959 ● 9 ● 23

---

Make it work, in values-v21 styles or theme xml needs to use this attribute:

```
<item name="android:windowTranslucentStatus">true</item>
```

That make the magic!

answered Nov 4 '14 at 22:03

Nicolas Jafelle

703 ● 8 ● 13

3 This makes my actionbar appear behind the statusbar as well. – Mixx Nov 5 '14 at 9:01

But that is the effect that we are trying to achieve, look at the Gmail 5.0 app? It shows the Side Bar behind the status bar. – Nicolas Jafelle Nov 5 '14 at 13:03

That's not what i meant to say. Eventhough the navigation drawer is behind the statusbar with this line of code, so is the toolbar/actionbar. – Mixx Nov 5 '14 at 15:55

2 The poster and answerer is a Google employee who designed AppCompat, so I'm pretty sure he knew what he was doing when he wrote the content. – afollestad Nov 17 '14 at 0:55

@Mixx, You can set android:fitsSystemWindows="true" for your normal content view (if taken the accepted answer layout for example). It can work, even for Android 4.4. But there is a slight difference here: In Android 5.0 the transparency level on the status bar is somewhat higher (higher alpha value) than that in GMail, making the status bar even darker. – Qianqian Dec 14 '14 at 12:00

---

This is the most simple, and it worked for me:

In the values-21:

```
<resources>
    <style name="AppTheme" parent="AppTheme.Base">
```

```
        ...
        <item name="android:windowTranslucentStatus">true</item>
    </style>
    <dimen name="topMargin">25dp</dimen>
</resources>
```

In the values:

```
<resources>
    <dimen name="topMargin">0dp</dimen>
</resources>
```

And set to your toolbar

```
android:layout_marginTop="@dimen/topMargin"
```

answered Nov 21 '14 at 19:59

Nicolás López

109 • 3

I also did this, and it worked like a charm on Lollipop. But I used `24dp` for top margin. – Rafa Apr 19 at 21:52

Try with this:

```
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/drawer_layout"
android:fitsSystemWindows="true">


<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!--Main layout and ads-->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <FrameLayout
            android:id="@+id/ll_main_hero"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1">

        </FrameLayout>

        <FrameLayout
            android:id="@+id/ll_ads"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <View
                android:layout_width="320dp"
                android:layout_height="50dp"
                android:layout_gravity="center"
                android:background="#ff00ff" />
        </FrameLayout>


    </LinearLayout>

    <!--Toolbar-->
    <android.support.v7.widget.Toolbar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/toolbar"
        android:elevation="4dp" />
</FrameLayout>


<!--left-->
<ListView
    android:layout_width="240dp"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:choiceMode="singleChoice"
    android:divider="@null"
    android:background="@mipmap/layer_image"
    android:id="@+id/left_drawer"></ListView>

<!--right-->
<FrameLayout
    android:layout_width="240dp"
    android:layout_height="match_parent"
    android:layout_gravity="right"
    android:background="@mipmap/layer_image">
```

```
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@mipmap/ken2"
        android:scaleType="centerCrop" />
</FrameLayout>
```

style :

```
<style name="ts_theme_overlay" parent="Theme.AppCompat.Light.NoActionBar">
    <item name="colorPrimary">@color/red_A700</item>
    <item name="colorPrimaryDark">@color/red1</item>
    <item name="android:windowBackground">@color/blue_A400</item>
</style>
```

Main Activity extends ActionBarActivity

```
toolBar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolBar);
```

Now you can `onCreateOptionsMenu` like as normal ActionBar with ToolBar.

This is my Layout

- TOP: Left Drawer - Right Drawer
  - MID: ToolBar (ActionBar)
  - BOTTOM: ListFragment

Hope you understand !have fun !

answered Mar 20 at 7:58

Son Nguyen Thanh

78 ● 6

---

Instead of using the `ScrimInsetsFrameLayout` ... Isn't it easier to just add a view with a fixed height of `24dp` and a background of `primaryColor` ?

I understand that this involves adding a dummy view in the hierarchy, but it seems cleaner to me.

I already tried it and it's working well.

```
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/activity_base_drawer_layout"
android:layout_width="match_parent"
android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <!-- THIS IS THE VIEW I'M TALKING ABOUT... -->
        <View
            android:layout_width="match_parent"
            android:layout_height="24dp"
            android:background="?attr/colorPrimary" />

        <android.support.v7.widget.Toolbar
            android:id="@+id/activity_base_toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            android:elevation="2dp"
            android:theme="@style/ThemeOverlay.AppCompat.Dark" />

        <FrameLayout
            android:id="@+id/activity_base_content_frame_layout"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </LinearLayout>

    <fragment
        android:id="@+id/activity_base_drawer_fragment"
        android:name="com.myapp.drawer.ui.DrawerFragment"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:elevation="4dp"
        tools:layout="@layout/fragment_drawer" />

</android.support.v4.widget.DrawerLayout>
```

answered May 25 at 22:15

mato

402 ● 2 ● 9

people why COMPLEXITY when the solution is SIMPLER than that !!!!!!!!!

when you set :

```
<item name="android:windowTranslucentStatus">true</item>
```

in your Style.xml (V21) the app bar will go behind the status bar all you have to do :

is set a view Button or TextView above the tool bar and give it a color.

with height equal to status bar height witch is 24dp (as the documentation said)

and match_parent width then you solved the problem :D :D :D

look this photos its my result :

http://www.4shared.com/photo/jbSaG883ce/device-2015-02-11-135545__1_.html
http://www.4shared.com/photo/qgYhbYoqce/device-2015-02-11-135656.html

edited Feb 11 at 13:55

answered Feb 11 at 13:49

MFQ
1 ● 1 ● 1

**protected** by Community ♦ Mar 10 at 0:06

Thank you for your interest in this question. Because it has attracted low-quality answers, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?