Creating a Navigation Drawer

The navigation drawer is a panel that displays the app's main navigation options on the left edge of the screen. It is hidden most of the time, but is revealed when the user swipes a finger from the left edge of the screen or, while at the top level of the app, the user touches the app icon in the action bar.

This lesson describes how to implement a navigation drawer using the `DrawerLayout` `(/reference/android/support/v4/widget/DrawerLayout.html)` APIs available in the Support Library (/tools/ support-library.html).

THIS LESSON TEACHES YOU TO:

1. Create a Drawer Layout
2. Initialize the Drawer List
3. Handle Navigation Click Events
4. Listen for Open and Close Events
5. Open and Close with the App Icon

TRY IT OUT

Download the sample app

NavigationDrawer.zip

Download the nav drawer icons

Android_Navigation_Drawer_Icon_20130516.zip

> **Navigation Drawer Design**
>
> Before you decide to use a navigation drawer in your app, you should understand the use cases and desi nciples defined in the Navigation Drawer (/design/p /navigation-drawer.html) design guide.

Create a Drawer Layout

To add a navigation drawer, declare your user inte rface with a `DrawerLayout` `(/reference/a ayout.html)` object as the root view of your layout. Inside the `DrawerLayout` `/widget/DrawerLayout.html)`, add one view that contains the main content f /hen the drawer is hidden) and another view that contains the contents of the

For example, tl `ayout` `(/reference/a ayout.html)` with two child views: a `FrameLayout` `(/reference/a` to contain the main content (populated by a `Fragment` `(/reference/a` ime), and a `ListView` `(/reference/a` the navigation drawer.

```
<android.s                                    ut
    xmlns:                                    roid.com/apk/res/android"
    androi
    androi                                    t"
    androi                                    nt">
    <!-- 1
    <Frame
        ar                                    e"
        ar                                    arent"
        ar                                    parent" />
    <!-- 1
    <ListV                                    rawer"
        ar
        ar                                    parent"
        ar                                    "
        ar                                    ice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp"
        android:background="#111"/>
</android.support.v4.widget.DrawerLayout>
```

This layout demonstrates some important layout characteristics:

- The main content view (the FrameLayout above) **must be the first child** in the DrawerLayout because the XML order implies z-ordering and the drawer must be on top of the content.
- The main content view is set to match the parent view's width and height, because it represents the entire UI when the navigation drawer is hidden.
- The drawer view (the ListView) **must specify its horizontal gravity** with the `android:layout_gravity` attribute. To support right-to-left (RTL) languages, specify the value with `"start"` instead of `"left"` (so the drawer appears on the right when the layout is RTL).
- The drawer view specifies its width in `dp` units and the height matches the parent view. The drawer width should be no more than 320dp so the user can always see a portion of the main content.

Initialize the Drawer List

In your activity, one of the first things to do is initialize the navigation drawer's list of items. How you do so depends on the content of your app, but a navigation drawer often consists of a ListView (/reference/android/widget/ListView.html), so the list should be populated by an Adapter (/reference/android/widget/Adapter.html) (such as ArrayAdapter (/reference/android/widget/ArrayAdapter.html) or SimpleCursorAdapter (/reference/android/widget/SimpleCursorAdapter.html)).

For example, here's how you can initialize the navigation list with a string array (/guide/topics/resources/string-resource.html#StringArray):

```
public class MainActivity extends Activity {
    private String[] mPlanetTitles;
    private ListView mDrawerList;
    ...

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

        mPlanetTitles = getResources().getStringArray(R.array.planets_array);
        mDrawerList = (ListView) findViewById(R.id.left_drawer);

        // Set the adapter for the list view
        mDrawerList.setAdapter(new ArrayAdapter<String>(this,
                R.layout.drawer_list_item, mPlanetTitles));
        // Set the list's click listener
        mDrawerList.setOnItemClickListener(new DrawerItemClickListener());

        ...
    }
}
```

This code also calls setOnItemClickListener() (/reference/android/widget/AdapterView.html#setOnItemClickListener(android.widget.AdapterView.OnItemClickListener)) to receive click events in the navigation drawer's list. The next section shows how to implement this interface and change the content view when the user selects an item.

Handle Navigation Click Events

When the user selects an item in the drawer's list, the system calls onItemClick() (/reference/android/widget/AdapterView.OnItemClickListener.html#onItemClick(android.widget.AdapterView<?>, android.view.View, int, long)) on the OnItemClickListener (/reference/android/widget/AdapterView.OnItemClickListener.html) given to setOnItemClickListener()

(/reference/android/widget/AdapterView.html#setOnItemClickListener(android.widget.AdapterView.OnItemClickListener)).

What you do in the onItemClick() (/reference/android/widget/AdapterView.OnItemClickListener.html#onItemClick(android.widget.AdapterView<?>, android.view.View, int, long)) method depends on how you've implemented your app structure (/design/patterns/app-structure.html). In the following example, selecting each item in the list inserts a different Fragment (/reference/android/app/Fragment.html) into the main content view (the FrameLayout (/reference/android/widget/FrameLayout.html) element identified by the R.id.content_frame ID):

```
private class DrawerItemClickListener implements ListView.OnItemClickListener {
    @Override
    public void onItemClick(AdapterView parent, View view, int position, long id) {
        selectItem(position);
    }
}

/** Swaps fragments in the main content view */
private void selectItem(int position) {
    // Create a new fragment and specify the planet to show based on position
    Fragment fragment = new PlanetFragment();
    Bundle args = new Bundle();
    args.putInt(PlanetFragment.ARG_PLANET_NUMBER, position);
    fragment.setArguments(args);

    // Insert the fragment by replacing any existing fragment
    FragmentManager fragmentManager = getFragmentManager();
    fragmentManager.beginTransaction()
                   .replace(R.id.content_frame, fragment)
                   .commit();

    // Highlight the selected item, update the title, and close the drawer
    mDrawer.setItemChecked(position, true);
    setTitle(mPlanetTitles[position]);
    mDrawerLayout.closeDrawer(mDrawer);
}

@Override
public void setTitle(CharSequence title) {
    mTitle = title;
    getActionBar().setTitle(mTitle);
}
```

Listen for Open and Close Events

To listen for drawer open and close events, call setDrawerListener() (/reference/android/support/v4/widget/DrawerLayout.html#setDrawerListener(android.support.v4.widget.DrawerLayout.DrawerListener)) on your DrawerLayout (/reference/android/support/v4/widget/DrawerLayout.html) and pass it an implementation of DrawerLayout.DrawerListener (/reference/android/support/v4/widget/DrawerLayout.DrawerListener.html). This interface provides callbacks for drawer events such as onDrawerOpened() (/reference/android/support/v4/widget/DrawerLayout.DrawerListener.html#onDrawerOpened(android.view.View)) and onDrawerClosed() (/reference/android/support/v4/widget/DrawerLayout.DrawerListener.html#onDrawerClosed(android.view.View)).

However, rather than implementing the DrawerLayout.DrawerListener (/reference/android/support/v4/widget/DrawerLayout.DrawerListener.html), if your activity includes the action bar (/guide/topics/ui/actionbar.html), you can instead extend the ActionBarDrawerToggle (/reference/android/support/v4/app/ActionBarDrawerToggle.html) class. The ActionBarDrawerToggle (/reference/android/support/v4/app/ActionBarDrawerToggle.html) implements DrawerLayout.DrawerListener (/reference/android/support/v4/widget/DrawerLayout.DrawerListener.html) so you can still override those callbacks, but it also facilitates the proper interaction behavior between the action bar icon and the navigation drawer (discussed further in the next section).

As discussed in the Navigation Drawer (/design/patterns/navigation-drawer.html) design guide, you should modify the contents of the action bar when the drawer is visible, such as to change the title and remove action items that are contextual to the main content. The following code shows how you can do so by overriding DrawerLayout.DrawerListener (/reference/android/support/v4/widget/DrawerLayout.DrawerListener.html) callback methods with an instance of the ActionBarDrawerToggle (/reference/android/support/v4/app/ActionBarDrawerToggle.html) class:

```
public class MainActivity extends Activity {
    private DrawerLayout mDrawerLayout;
    private ActionBarDrawerToggle mDrawerToggle;
    private CharSequence mDrawerTitle;
    private CharSequence mTitle;
    ...

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

        mTitle = mDrawerTitle = getTitle();
        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,
                R.drawable.ic_drawer, R.string.drawer_open, R.string.drawer_close) {

            /** Called when a drawer has settled in a completely closed state. */
            public void onDrawerClosed(View view) {
                getActionBar().setTitle(mTitle);
                invalidateOptionsMenu(); // creates call to onPrepareOptionsMenu()
            }

            /** Called when a drawer has settled in a completely open state. */
            public void onDrawerOpened(View drawerView) {
                getActionBar().setTitle(mDrawerTitle);
                invalidateOptionsMenu(); // creates call to onPrepareOptionsMenu()
            }
        };

        // Set the drawer toggle as the DrawerListener
        mDrawerLayout.setDrawerListener(mDrawerToggle);
    }

    /* Called whenever we call invalidateOptionsMenu() */
    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        // If the nav drawer is open, hide action items related to the content view
        boolean drawerOpen = mDrawerLayout.isDrawerOpen(mDrawerList);
        menu.findItem(R.id.action_websearch).setVisible(!drawerOpen);
        return super.onPrepareOptionsMenu(menu);
    }
}
```

The next section describes the ActionBarDrawerToggle (/reference/android/support/v4/app/ActionBarDrawerToggle.html) constructor arguments and the other steps required to set it up to handle interaction with the action bar icon.

Open and Close with the App Icon

Users can open and close the navigation drawer with a swipe gesture from or towards the left edge of the screen, but if you're using the action bar (/guide/topics/ui/actionbar.html), you should also allow users to open and close it by touching the app icon. And the app icon should also indicate the presence of the navigation drawer with a special icon. You can implement all this behavior by using the ActionBarDrawerToggle (/reference/android/support/v4/app/ActionBarDrawerToggle.html) shown in the previous section.

To make ActionBarDrawerToggle (/reference/android/support/v4/app/ActionBarDrawerToggle.html) work, create an instance of it with its constructor, which requires the following arguments:

- The Activity hosting the drawer.
- The DrawerLayout.
- A drawable resource to use as the drawer indicator.

  Download the standard navigation icons (http://developer.android.com/downloads/design/Android_Navigation_Drawer_Icon_20130516.zip) (available for both dark and light themes).

- A String resource to describe the "open drawer" action (for accessibility).
- A String resource to describe the "close drawer" action (for accessibility).

Then, whether or not you've created a subclass of ActionBarDrawerToggle (/reference/android/support/v4/app/ActionBarDrawerToggle.html) as your drawer listener, you need to call upon your ActionBarDrawerToggle (/reference/android/support/v4/app/ActionBarDrawerToggle.html) in a few places throughout your activity lifecycle:

```
public class MainActivity extends Activity {
    private DrawerLayout mDrawerLayout;
    private ActionBarDrawerToggle mDrawerToggle;
    ...

    public void onCreate(Bundle savedInstanceState) {
        ...

        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        mDrawerToggle = new ActionBarDrawerToggle(
                this,                  /* host Activity */
                mDrawerLayout,         /* DrawerLayout object */
                R.drawable.ic_drawer,  /* nav drawer icon to replace 'Up' caret */
                R.string.drawer_open,  /* "open drawer" description */
                R.string.drawer_close  /* "close drawer" description */
                ) {

            /** Called when a drawer has settled in a completely closed state. */
            public void onDrawerClosed(View view) {
                getActionBar().setTitle(mTitle);
            }

            /** Called when a drawer has settled in a completely open state. */
            public void onDrawerOpened(View drawerView) {
                getActionBar().setTitle(mDrawerTitle);
            }
        };

        // Set the drawer toggle as the DrawerListener
        mDrawerLayout.setDrawerListener(mDrawerToggle);

        getActionBar().setDisplayHomeAsUpEnabled(true);
        getActionBar().setHomeButtonEnabled(true);
    }

    @Override
    protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has occurred.
        mDrawerToggle.syncState();
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        mDrawerToggle.onConfigurationChanged(newConfig);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Pass the event to ActionBarDrawerToggle, if it returns
        // true, then it has handled the app icon touch event
        if (mDrawerToggle.onOptionsItemSelected(item)) {
          return true;
        }
        // Handle your other action bar items...

        return super.onOptionsItemSelected(item);
    }

    ...
}
```

For a complete example of a navigation drawer, download the sample available at the top of the page (#top).