



More widgets

Common Controls

- SeekBar
- ToggleButton / Switch
- TimePicker
- DatePicker
- Swipe-to-Refresh and ListView
- Action bar

SeekBar



XML Part

```
<SeekBar  
    android:id="@+id/tipSeekBar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:max="100"  
    android:progress="50"  
    android:layout_marginTop="20dp"  
    android:layout_gravity="fill_horizontal"/>
```

SeekBar

- oJava Part (adding listener to notify changes)

```
seekBarInstance.setOnSeekBarChangeListener(new  
OnSeekBarChangeListener() {...}
```

- oWe need to implement three abstract methods here;

```
public void onProgressChanged(SearchBar seekBar, int  
progressValue, boolean fromUser) {...}
```

- oThis listener method will be invoked if any change is made in the SeekBar

```
public void onStartTrackingTouch(SearchBar seekBar) {...}
```

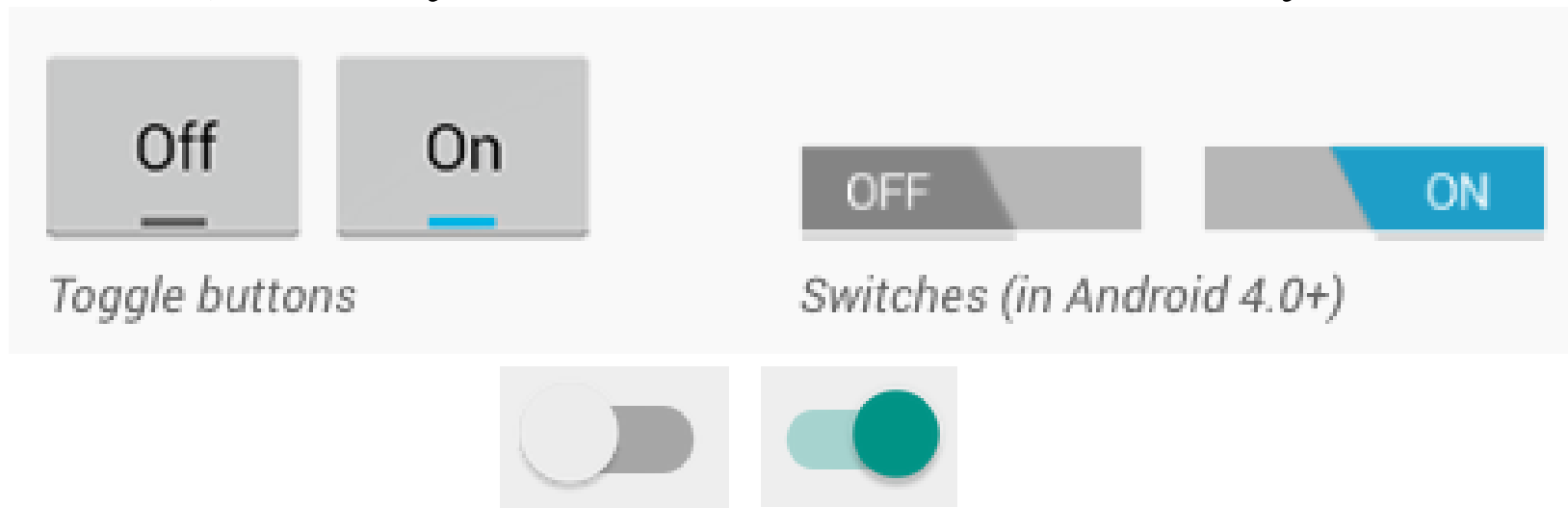
- oThis listener method will be invoked at the start of user's touch event.

```
public void onStopTrackingTouch(SearchBar seekBar) {...}
```

- oThis listener method will be invoked at the end of user touch event.

ToggleButton or Switch

- A toggle button allows the user to change a setting between two states.
- You can add a basic toggle button to your layout with the **ToggleButton** object.
- Android 4.0 (API level 14) introduces another kind of toggle button called a switch that provides a slider control, which you can add with a **Switch** object



ToggleButton or Switch

○XML Part

<Switch

```
    android:id="@+id/switchButton"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textOn="Vibrate on"  
    android:textOff="Vibrate off"  
    android:showText="true"  
    android:onClick="onSwitchClicked"/>
```

ToggleButtons or Switch

oJava Part

```
public void onSwitchClicked(View view) {  
    // Is the switch on?  
    boolean on = ((Switch) view).isChecked();  
  
    if (on) {  
        // Enable vibrate  
    } else {  
        // Disable vibrate  
    }  
}
```

SwitchCompat

- SwitchCompat is a version of the Switch widget which runs on devices back to API 7.

```
<android.support.v7.widget.SwitchCompat
    android:id="@+id/switchButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="on"
    android:textOff="off"
    android:showText="true"
    android:onClick="onSwitchClicked"/>
```


ImageView and Asset folder

- You can load an unknown number of images from assets folder (app\src\main\assets\...)

```
AssetManager assets = this.getAssets();
try {
    String [] images = assets.list("png");

    for(int i = 0; i < images.length; i++)
    {
        ImageView imageView = new ImageView(this);
        String imageName = "png/" + images[i];
        Drawable image =
        Drawable.createFromStream(assets.open(imageName), imageName);
        imageView.setImageDrawable(image);

        //addView to add the image
    }

} catch (IOException e) {
    e.printStackTrace();
}
```

ScrollView

- A ScrollView is a simple scrolling container you can use to scroll whatever you put inside it, which might be a list of items, or it might not

<ScrollView

```
    android:id="@+id/scrollView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
        <LinearLayout
            android:id="@+id/linearLayout1"
            android:orientation="vertical"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
```

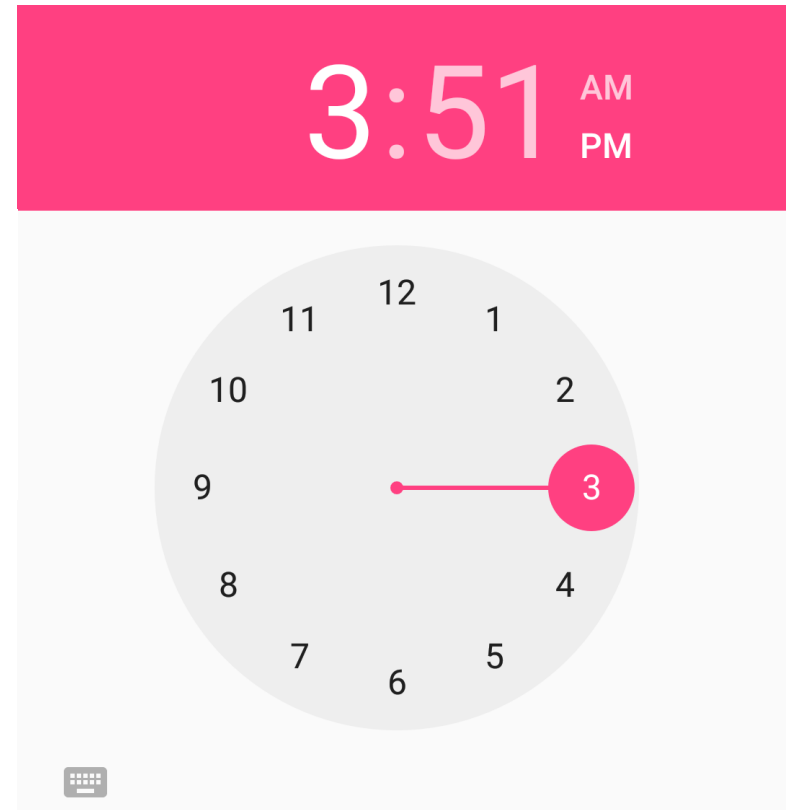
</ScrollView>

TimePicker

- Android TimePicker allows you to select the time of day in either 24 hour or AM/PM mode

- Default form (clock)

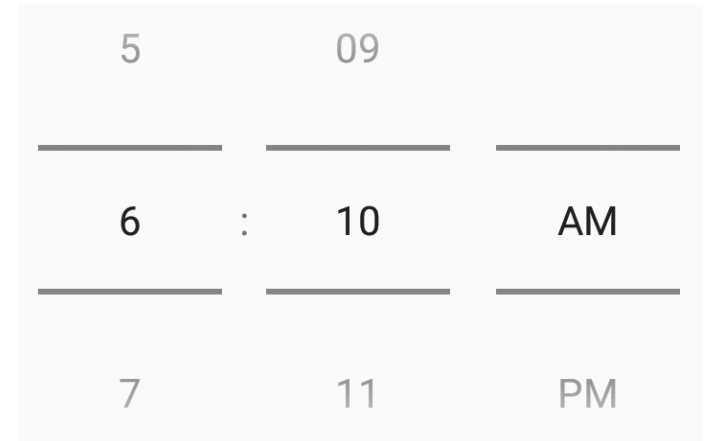
```
<TimePicker
    android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="clock">
</TimePicker>
```



TimePicker

○ Spinner mode

```
<TimePicker
    android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="spinner">
</TimePicker>
```



```
TimePicker timePicker = (TimePicker)findViewById(R.id.timerPicker1);
// to get the time selected by the user on the screen
int min = timePicker.getCurrentMinute(); // This method was deprecated in API
level 23. Use getMinute()

int hour = timePicker.getCurrentHour(); // This method was deprecated in API
level 23. Use getHour()
```

TimePicker

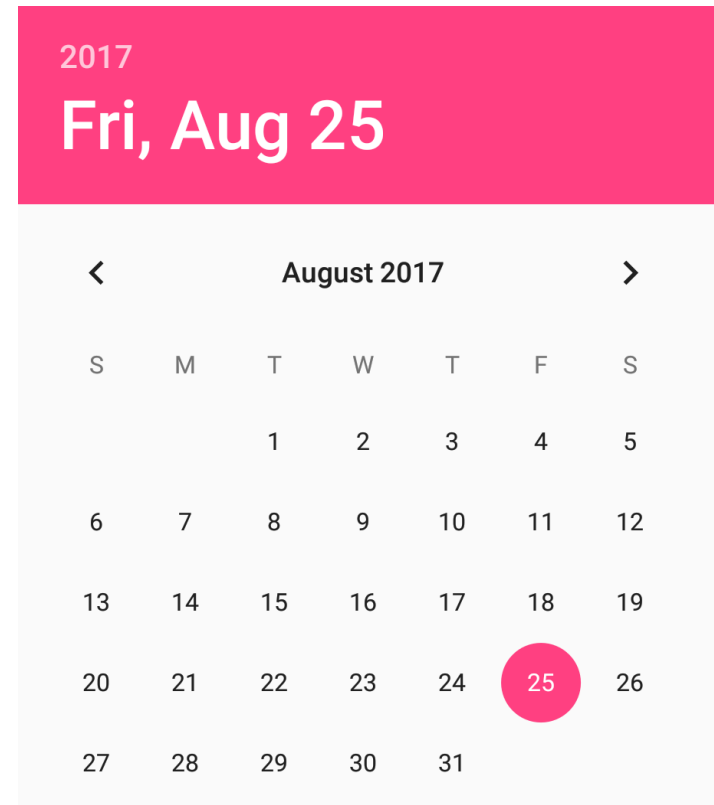
- `setOnTimeChangeListener(TimePicker.OnTimeChangeListener onTimeChangeListener)`
 - This method set the callback that indicates the time has been adjusted by the user

```
timePicker.setOnTimeChangeListener(new TimePicker.OnTimeChangeListener() {  
    @Override  
    public void onTimeChanged(TimePicker timePicker, int hourOfDay, int minute)  
    {  
  
    }  
});
```

DatePicker

- Provides a widget for selecting a date.
- Default form (calendar)

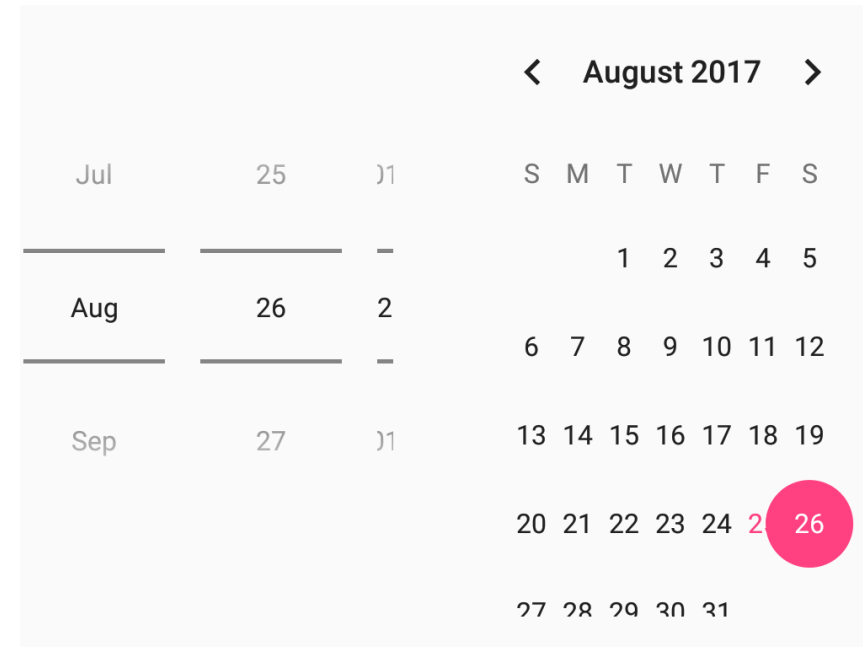
```
<DatePicker  
    android:id="@+id/datePicker1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:datePickerMode="calendar">  
</DatePicker>
```



DatePicker

- spinner form

```
<DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:datePickerMode="spinner">
</DatePicker>
```



To update the selected date:

```
DatePicker datePicker= (DatePicker)findViewById(R.id.datePicker1);
datePicker.updateDate(year, month, dayOfMonth);
```

Get the selected date:

```
int day = datePicker.getDayOfMonth();
int month = datePicker.getMonth() + 1;
int year = datePicker.getYear();
```

DatePicker

- callback used to indicate the user changed the date
 - Use the init method

```
DatePicker datePicker= (DatePicker)findViewById(R.id.datePicker1);
```

```
Calendar gregorianCalendar = new GregorianCalendar();
```

```
int day = gregorianCalendar.get(Calendar.DAY_OF_MONTH);
```

```
int month = gregorianCalendar.get(Calendar.MONTH);
```

```
int year = gregorianCalendar.get(Calendar.YEAR);
```

```
datePicker.init(year, month, year, new DatePicker.OnDateChangedListener() {  
    @Override  
    public void onDateChanged(DatePicker datePicker, int year, int monthOfYear,  
int dayOfMonth) {  
  
    }  
});
```

The DatePicker public method setOnDateChangedListener is added in [API level 26](#)

➔ Min API level should be 26 in order to use it.

Swipe-to-Refresh and ListView

To add the swipe to refresh widget to an existing app, add **SwipeRefreshLayout** as the parent of a **single ListView** or **GridView**

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

```
<ListView  
    android:id="@android:id/list"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

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

Swipe-to-Refresh - Responding to a Refresh Request

Implement the `SwipeRefreshLayout.OnRefreshListener` interface and its `onRefresh()` method.

```
mySwipeRefreshLayout.setOnRefreshListener(  
    new SwipeRefreshLayout.OnRefreshListener() {  
        @Override  
        public void onRefresh() {  
  
            /* This method performs the actual data-refresh operation.  
               The method calls setRefreshing(false) when it's finished */  
  
            MyUpdateOperation();  
        }  
    }  
);
```

Swipe-to-Refresh – Update the ListView

```
private void MyUpdateOperation()
{
    for(int j = 0; j < 10; j++, i++)
        objects.add("text " +i);    // objects is an ArrayList

    ArrayAdapter<String> arrayAdapter= new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, objects);

    listView.setAdapter(arrayAdapter);

    Toast.makeText(getApplicationContext(), "refreshed",
    Toast.LENGTH_LONG).show();

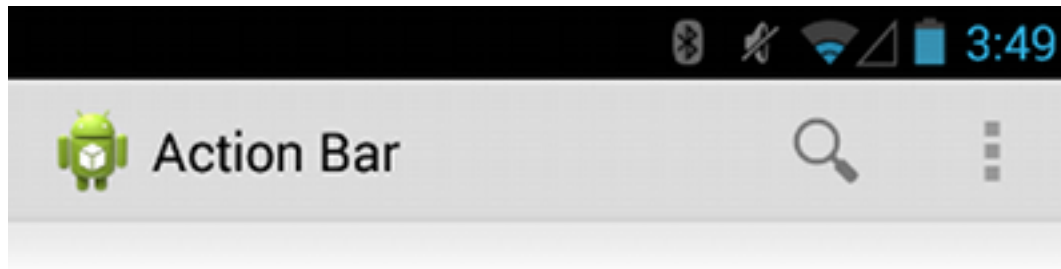
    mySwipeRefreshLayout.setRefreshing(false);
}
```

What's the action bar?

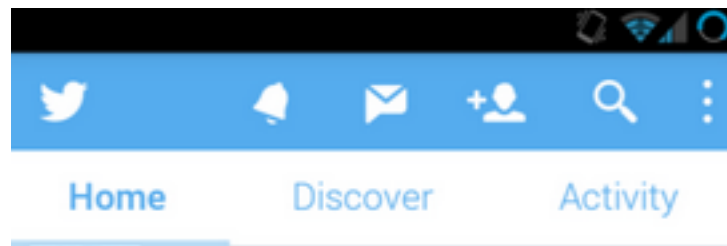
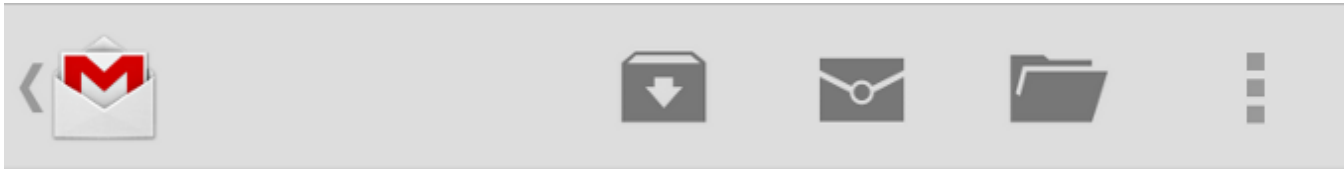
One of the most important design elements you can implement for your app's activities.

Key functions include:

- A dedicated space for giving your app an identity and indicating the user's location in the app.
- Access to important actions in a predictable way (such as Search).
- Support for navigation and view switching (with tabs or drop-down lists).

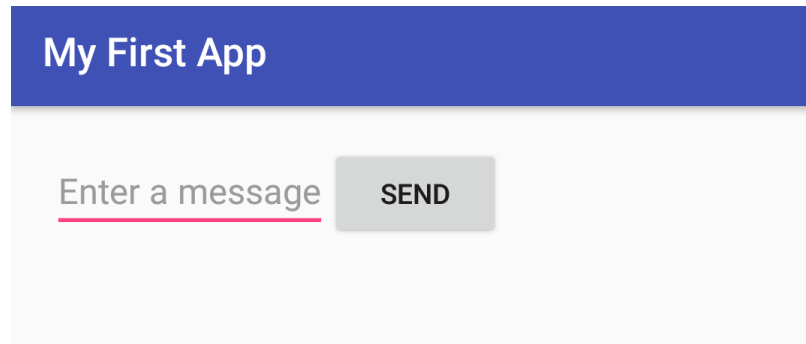


Some real life apps Action bars



Basic form of Action Bar

- In its most basic form, the action bar displays the title for the activity on the left.



Setting up the Action Bar

- 2 ways to create an Activity with App Bar
 1. Some themes set up an **ActionBar** object as an app bar by default (when using Holo theme) and extending ActionBarActivity.
 2. Using **Toolbar** widget : This is a generalization of the ActionBar that gives you much more control and flexibility. Toolbar is a view in your hierarchy just like any other

Add a Toolbar to an Activity

1. Make sure the activity extends AppCompatActivity
2. In the app Manifest, set the <activity> element to use one of appcompat's **NoActionBar** themes. Using one of these themes prevents the app from using the native ActionBar class to provide the app bar

Example :

```
android:theme="@style/Theme.AppCompat.Light.NoActionBar"
```

1. Add Toolbar widget to the activity's layout

Add a Toolbar to an Activity

```
<android.support.design.widget.AppBarLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:theme="@style/AppTheme.AppBarOverlay">
```

```
<android.support.v7.widget.Toolbar  
    android:id="@+id/toolbar"  
    android:layout_width="match_parent"  
    android:layout_height="?attr/actionBarSize"  
    android:background="?attr/colorPrimary"  
    app:popupTheme="@style/AppTheme.PopupOverlay" />
```

```
</android.support.design.widget.AppBarLayout>
```

Add a Toolbar to an Activity

In the Activity onCreate() method :

```
Toolbar myToolbar = (Toolbar) findViewById(R.id.my_toolbar);  
setSupportActionBar(myToolbar);
```

Adding Action buttons

○ 3 main steps are needed to add the action buttons:

1. Specify the actions in XML
2. Add the actions to the action bar
3. Respond to Action buttons

Adding Action buttons – Step 1

- To add actions to the action bar, create a new XML file in your project's **res/menu/** directory
- Add an **<item>** element for each item you want to include in the action bar.

Adding Action buttons – Step 1

○Example:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"
        android:showAsAction="ifRoom" />

    <!-- Settings, should always be in the overflow -->
    <item
        android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:showAsAction="never" />
</menu>
```

Adding Action buttons – Step 1

<item

```
android:id="@+id/action_search"  
android:icon="@drawable/ic_action_search"  
android:title="@string/action_search"  
android:showAsAction="ifRoom" />
```

ID similar
to any view
before

Android:icon
Is the icon that
will appear in
the action bar

Android:showAsAction

Is used to specify if you want the icon
to appear in the action bar or in the
overflow

ifRoom means show it in the action bar
if there is room, if not then overflow

Adding Action buttons – Step 1

- You can download action bar icon packs from Android website.

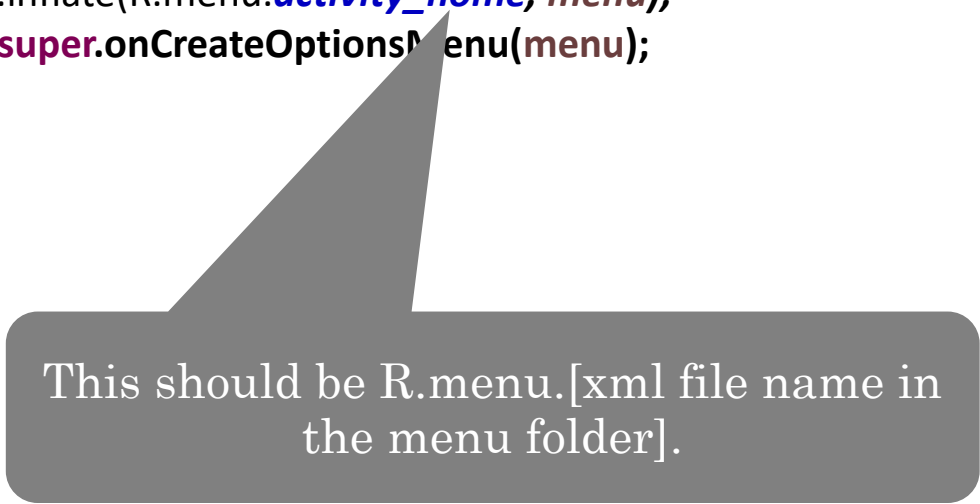
<http://developer.android.com/design/style/iconography.html#action-bar>

- You can create new icons in android studio
 - Right click on Res folder
 - New → Image Asset

Adding Action buttons – Step 2

- To place the menu items into the action bar, implement the `onCreateOptionsMenu()` callback method in your activity to inflate the menu resource into the given **Menu** object.

```
public boolean onCreateOptionsMenu(Menu menu)
{
    // Inflate the menu items for use in the action bar
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.activity_home, menu);
    return super.onCreateOptionsMenu(menu);
}
```



This should be R.menu.[xml file name in the menu folder].

Adding Action buttons – Step 3

- When the user presses one of the action buttons or another item in the action overflow, the system calls your activity's [`onOptionsItemSelected\(\)`](#) callback method.
- In your implementation of this method, call [`getItemId\(\)`](#) on the given [`MenuItem`](#) to determine which item was pressed—the returned ID matches the value you declared in the corresponding `<item>` element's **`android:id`** attribute

Adding Action buttons – Step 3

```
public boolean onOptionsItemSelected(MenuItem item)
{
    switch(item.getItemId())
    {
        case R.id.action_search:
            // Open Search Activity
            Intent intent = new Intent("com.example.testingactionbar2.SearchActivity");
            startActivity(intent);
            return true;

        case R.id.action_settings:
            // do something

        return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```