**Menu Resource**

A menu resource defines an application menu (Options Menu, Context Menu, or submenu) that can be inflated with [MenuInflater](http://developer.android.com/reference/android/view/MenuInflater.html).

**FILE LOCATION:**

res/menu/*filename*.xml  
The filename will be used as the resource ID.

**RESOURCE REFERENCE:**

In Java: R.menu.*filename*  
In XML: @[*package*:]menu.*filename*

**SYNTAX:**

<[menu](http://developer.android.com/guide/topics/resources/menu-resource.html#menu-element) >  
    <[item](http://developer.android.com/guide/topics/resources/menu-resource.html#item-element)

android:id="@[+][*package*:]id/*resource\_name*"  
          android:title="*string*"

          android:icon="@[package:]drawable/*drawable\_resource\_name*"  
          android:onClick="*method name*"  
          app:showAsAction=["ifRoom" | "never" | "withText" | "always" |

"collapseActionView"]

          android:visible=["true" | "false"]/>

</menu>

**SYNTAX (Menu Group):**

<[menu](http://developer.android.com/guide/topics/resources/menu-resource.html#menu-element) >  
    <[group](http://developer.android.com/guide/topics/resources/menu-resource.html#group-element)

android:id="@[+][*package*:]id/*resource name*"  
           android:checkableBehavior=["none" | "all" | "single"]  
           android:visible=["true" | "false"]  
           android:enabled=["true" | "false"]>  
        <[item](http://developer.android.com/guide/topics/resources/menu-resource.html#item-element) />

<[item](http://developer.android.com/guide/topics/resources/menu-resource.html#item-element) />

<[item](http://developer.android.com/guide/topics/resources/menu-resource.html#item-element) />  
    </group>

</menu>

**SYNTAX (Sub-Menu):**

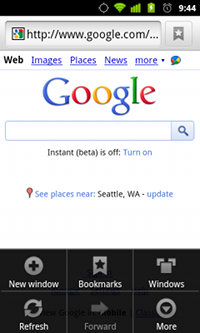
<[menu](http://developer.android.com/guide/topics/resources/menu-resource.html#menu-element) >  
    <[item](http://developer.android.com/guide/topics/resources/menu-resource.html#item-element) >  
        <[menu](http://developer.android.com/guide/topics/resources/menu-resource.html#menu-element)>  
          <[item](http://developer.android.com/guide/topics/resources/menu-resource.html#item-element) />  
        </menu>  
    </item>  
</menu>

**Options menu and app bar**

The [options menu](http://developer.android.com/guide/topics/ui/menus.html#options-menu) is the primary collection of menu items for an activity. It's where you should place actions that have a global impact on the app, such as "Search," "Compose email," and "Settings."

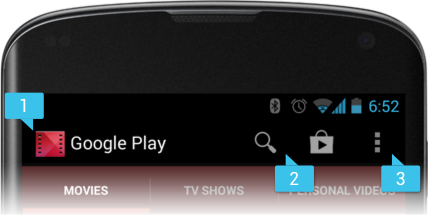
If you've developed your application for **Android 2.3.x (API level 10) or lower**, the contents of your options menu appear at the bottom of the screen

**For Example**



If you've developed your application for **Android 3.0 (API level 11) and higher**, items from the options menu are available in the app bar. By default, the system places all items in the action overflow, which the user can reveal with the action overflow icon on the right side of the app bar (or by pressing the device *Menu* button, if available).

**For Example**



**Adding OptionsMenu To Activity**

1. Your class Must extends **"AppCompactActivity"** class else if u r class extends **“Activity”** class we can see Option's Menu at bottom.

2. To Inflate OptionMenu we need to override **onCreateOptionsMenu** menthod

3. inside the method inflate Option's Menu by using “**MenuInflater”**

MenuInflater mi = getMenuInflater();

mi.inflate(R.menu.*options1*,menu);

**Handling click events**

When the user selects an item from the options menu (including action items in the app bar), the system calls your activity's  [onOptionsItemSelected()](http://developer.android.com/reference/android/app/Activity.html#onOptionsItemSelected(android.view.MenuItem)) method.

This method passes the [MenuItem](http://developer.android.com/reference/android/view/MenuItem.html) selected. You can identify the item by calling [getItemId()](http://developer.android.com/reference/android/view/MenuItem.html#getItemId()), which returns the unique ID for the menu item (defined by the **android:id**attribute in the menu resource or with an integer given to the [add()](http://developer.android.com/reference/android/view/Menu.html#add(int, int, int, int)) method).

**Context menu and contextual action mode**

A context menu is a [floating menu](http://developer.android.com/guide/topics/ui/menus.html#FloatingContextMenu) that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame.

The [contextual action mode](http://developer.android.com/guide/topics/ui/menus.html#CAB) displays action items that affect the selected content in a bar at the top of the screen and allows the user to select multiple items.

**Creating a floating context menu**

1. Register the [View](http://developer.android.com/reference/android/view/View.html) to which the context menu should be associated by calling [registerForContextMenu()](http://developer.android.com/reference/android/app/Activity.html#registerForContextMenu(android.view.View)) and pass it the [View](http://developer.android.com/reference/android/view/View.html).

If your activity uses a [ListView](http://developer.android.com/reference/android/widget/ListView.html) or [GridView](http://developer.android.com/reference/android/widget/GridView.html) and you want each item to provide the same context menu, register all items for a context menu by passing the [ListView](http://developer.android.com/reference/android/widget/ListView.html) or [GridView](http://developer.android.com/reference/android/widget/GridView.html) to [registerForContextMenu()](http://developer.android.com/reference/android/app/Activity.html#registerForContextMenu(android.view.View)).

1. Implement the [onCreateContextMenu()](http://developer.android.com/reference/android/view/View.OnCreateContextMenuListener.html#onCreateContextMenu(android.view.ContextMenu, android.view.View, android.view.ContextMenu.ContextMenuInfo)) method in your [Activity](http://developer.android.com/reference/android/app/Activity.html) or [Fragment](http://developer.android.com/reference/android/app/Fragment.html).

When the registered view receives a long-click event, the system calls your [onCreateContextMenu()](http://developer.android.com/reference/android/view/View.OnCreateContextMenuListener.html#onCreateContextMenu(android.view.ContextMenu, android.view.View, android.view.ContextMenu.ContextMenuInfo)) method. This is where you define the menu items, usually by inflating a menu resource.

1. Implement [onContextItemSelected()](http://developer.android.com/reference/android/app/Activity.html#onContextItemSelected(android.view.MenuItem)).

When the user selects a menu item, the system calls this method so you can perform the appropriate action.

**Using the contextual action mode**

The contextual action mode is a system implementation of [ActionMode](http://developer.android.com/reference/android/view/ActionMode.html) that focuses user interaction toward performing contextual actions.

**Enabling the contextual action mode for individual views**

If you want to invoke the contextual action mode only when the user selects specific views, you should

1. Implement the [ActionMode.Callback](http://developer.android.com/reference/android/view/ActionMode.Callback.html) interface. In its callback methods, you can specify the actions for the contextual action bar, respond to click events on action items, and handle other lifecycle events for the action mode.
2. Call [startActionMode()](http://developer.android.com/reference/android/app/Activity.html#startActionMode(android.view.ActionMode.Callback)) when you want to show the bar (such as when the user long-clicks the view).

**ActionMode.Callback interface abstract methods**

* [onCreateActionMode(ActionMode, Menu)](https://developer.android.com/reference/android/view/ActionMode.Callback.html#onCreateActionMode(android.view.ActionMode, android.view.Menu)) once on initial creation
* [onPrepareActionMode(ActionMode, Menu)](https://developer.android.com/reference/android/view/ActionMode.Callback.html#onPrepareActionMode(android.view.ActionMode, android.view.Menu)) after creation and any time the [ActionMode](https://developer.android.com/reference/android/view/ActionMode.html) is invalidated
* [onActionItemClicked(ActionMode, MenuItem)](https://developer.android.com/reference/android/view/ActionMode.Callback.html#onActionItemClicked(android.view.ActionMode, android.view.MenuItem)) any time a contextual action button is clicked
* [onDestroyActionMode(ActionMode)](https://developer.android.com/reference/android/view/ActionMode.Callback.html#onDestroyActionMode(android.view.ActionMode)) when the action mode is closed

**Enabling batch contextual actions in a ListView or GridView**

If you have a collection of items in a [ListView](http://developer.android.com/reference/android/widget/ListView.html) or [GridView](http://developer.android.com/reference/android/widget/GridView.html) (or another extension of [AbsListView](http://developer.android.com/reference/android/widget/AbsListView.html)) and want to allow users to perform batch actions, you should.

* Implement the [AbsListView.MultiChoiceModeListener](http://developer.android.com/reference/android/widget/AbsListView.MultiChoiceModeListener.html) interface and set it for the view group with [setMultiChoiceModeListener()](http://developer.android.com/reference/android/widget/AbsListView.html#setMultiChoiceModeListener(android.widget.AbsListView.MultiChoiceModeListener)).
* In the listener's callback methods, you can specify the actions for the contextual action bar, respond to click events on action items, and handle other callbacks inherited from the [ActionMode.Callback](http://developer.android.com/reference/android/view/ActionMode.Callback.html) interface.
* Call [setChoiceMode()](http://developer.android.com/reference/android/widget/AbsListView.html#setChoiceMode(int)) with the [CHOICE\_MODE\_MULTIPLE\_MODAL](http://developer.android.com/reference/android/widget/AbsListView.html#CHOICE_MODE_MULTIPLE_MODAL) argument.

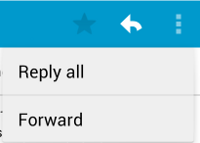
**Java Snippet**

ListView listView = getListView();  
listView.setChoiceMode(ListView.CHOICE\_MODE\_MULTIPLE\_MODAL);  
listView.setMultiChoiceModeListener(new MultiChoiceModeListener() {...});

**Popup menu**

A popup menu displays a list of items in a vertical list that's anchored to the view that invoked the menu.

**Creating a Popup Menu**



A popup menu in the Gmail app, anchored to the overflow button at the top-right.

A [PopupMenu](http://developer.android.com/reference/android/widget/PopupMenu.html) is a modal menu anchored to a [View](http://developer.android.com/reference/android/view/View.html). It appears below the anchor view if there is room, or above the view otherwise.

If you [define your menu in XML](http://developer.android.com/guide/topics/ui/menus.html#xml), here's how you can show the popup menu:

1. Instantiate a [PopupMenu](http://developer.android.com/reference/android/widget/PopupMenu.html) with its constructor, which takes the current application [Context](http://developer.android.com/reference/android/content/Context.html) and the [View](http://developer.android.com/reference/android/view/View.html) to which the menu should be anchored.
2. Use [MenuInflater](http://developer.android.com/reference/android/view/MenuInflater.html) to inflate your menu resource into the [Menu](http://developer.android.com/reference/android/view/Menu.html) object returned by [PopupMenu.getMenu()](http://developer.android.com/reference/android/widget/PopupMenu.html#getMenu()).
3. Call [PopupMenu.show()](http://developer.android.com/reference/android/widget/PopupMenu.html#show()).

**Handling click events**

To perform an action when the user selects a menu item, you must implement the [PopupMenu.OnMenuItemClickListener](http://developer.android.com/reference/android/widget/PopupMenu.OnMenuItemClickListener.html) interface and register it with your [PopupMenu](http://developer.android.com/reference/android/widget/PopupMenu.html) by calling [setOnMenuItemclickListener()](http://developer.android.com/reference/android/widget/PopupMenu.html#setOnMenuItemClickListener(android.widget.PopupMenu.OnMenuItemClickListener)).

When the user selects an item, the system calls the [onMenuItemClick()](http://developer.android.com/reference/android/widget/PopupMenu.OnMenuItemClickListener.html#onMenuItemClick(android.view.MenuItem))

callback in your interface.