

Android Developer Fundamentals V2

User Interaction

Lesson 4



4.3 Menus and pickers

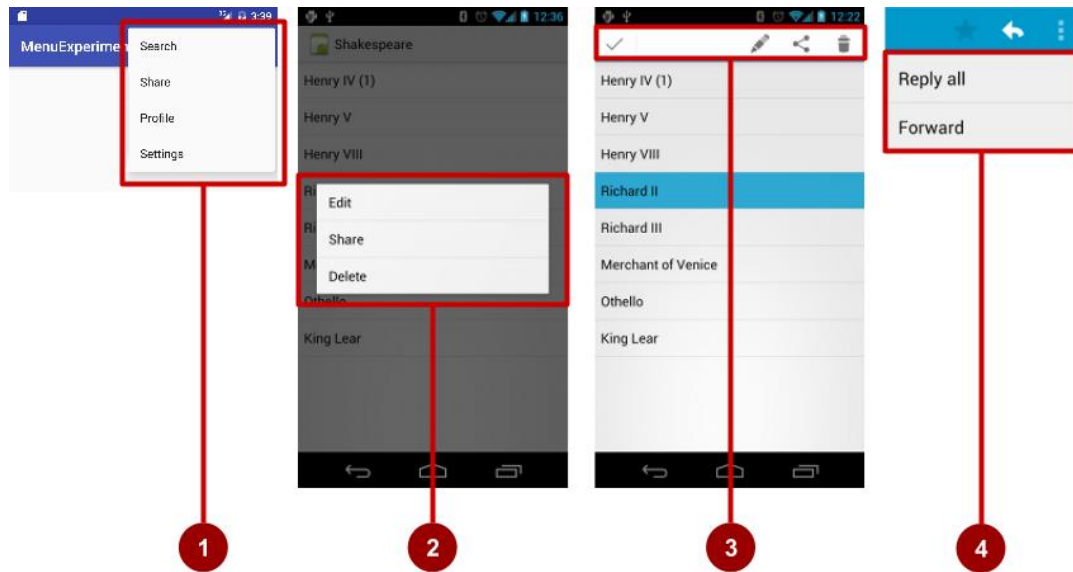
Contents

- Overview
- App Bar with Options Menu
- Contextual menus
- Popup menus
- Dialogs
- Pickers

Overview

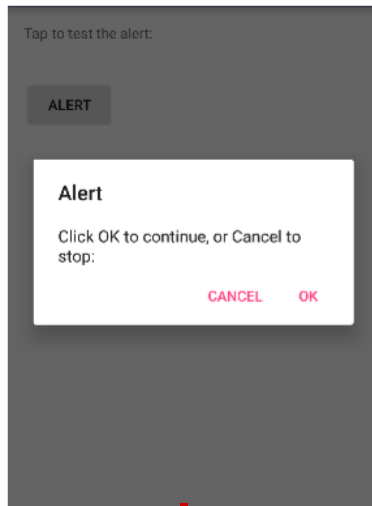
Types of Menu

1. App bar with options menu
2. Context menu
3. Contextual action bar
4. Popup menu

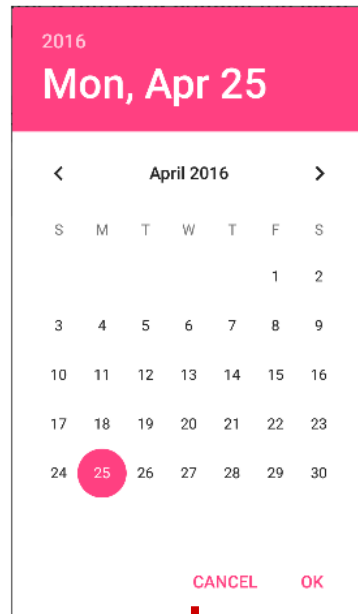


Dialogs and pickers

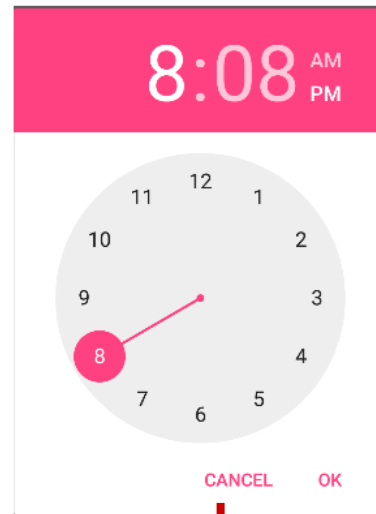
1. Alert dialog
2. Date picker
3. Time picker



1



2



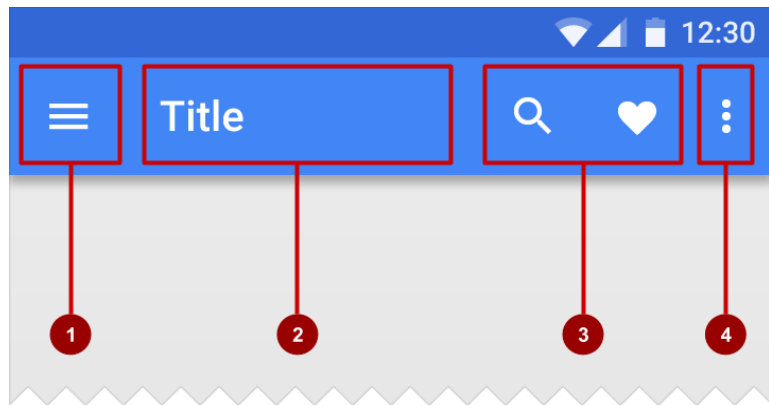
3

App Bar with Options Menu

What is the App Bar?

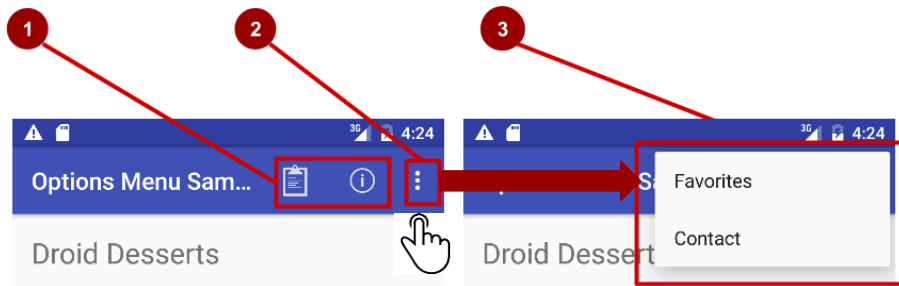
Bar at top of each screen—same for all devices (usually)

1. Nav icon to open navigation drawer
2. Title of current Activity
3. Icons for options menu items
4. Action overflow button for the rest of the options menu



What is the options menu?

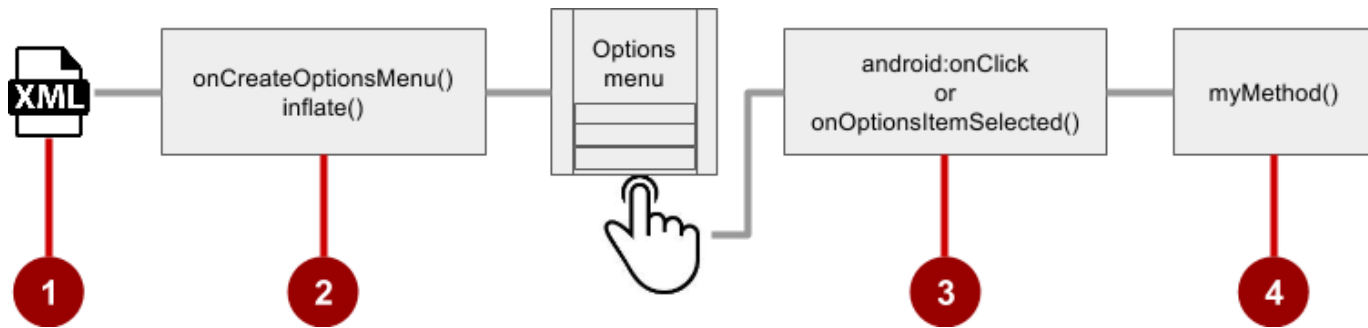
- Action icons in the app bar for important items (1)
- Tap the three dots, the "action overflow button" to see the options menu (2)
- Appears in the right corner of the app bar (3)
- For navigating to other activities and editing app settings



Adding Options Menu

Steps to implement options menu

1. XML menu resource (menu_main.xml)
2. onCreateOptionsMenu() to inflate the menu
3. onClick attribute or onOptionsItemSelected()
4. Method to handle item click



Create menu resource

1. Create menu resource directory
2. Create XML menu resource (menu_main.xml)
3. Add entry for each menu item (**Settings** and **Favorites**):

```
<item    android:id="@+id/option_settings"  
        android:title="Settings" />
```

```
<item    android:id="@+id/option_favorites"  
        android:title="Favorites" />
```

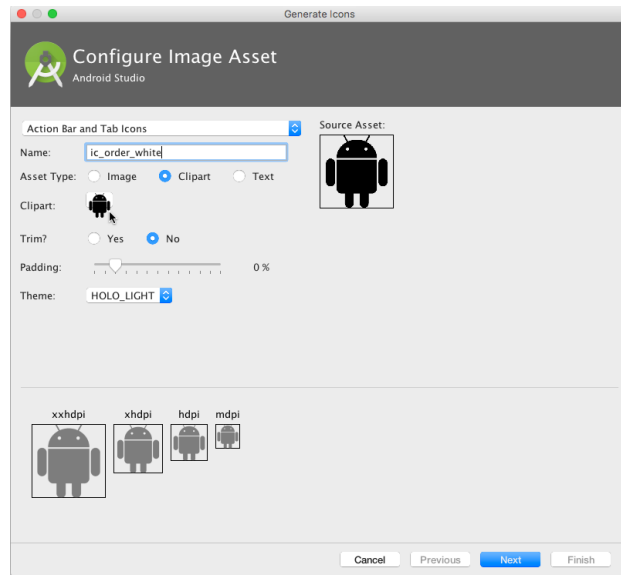
Inflate options menu

Override onCreateOptionsMenu() in Activity

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

Add icons for menu items

1. Right-click **drawable**
2. Choose **New > Image Asset**
3. Choose **Action Bar and Tab Items**
4. Edit the icon name
5. Click clipart image, and click icon
6. Click **Next**, then **Finish**



Add menu item attributes

```
<item
```

```
    android:id="@+id/action_favorites"  
    android:icon="@drawable/ic_favorite"  
    android:orderInCategory="30"  
    android:title="@string/action_favorites"  
    app:showAsAction="ifRoom" />
```

Override onOptionsItemSelected()

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_settings:
            showSettings();
            return true;
        case R.id.action_favorites:
            showFavorites();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

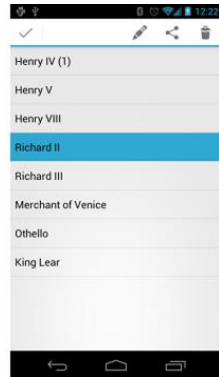
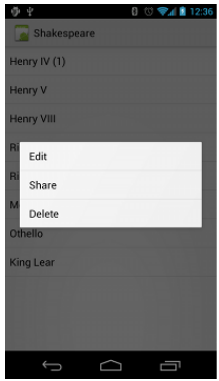

Contextual Menus

What are contextual menus?

- Allows users to perform action on selected View
- Can be deployed on any View
- Most often used for items in RecyclerView, GridView, or other View collection

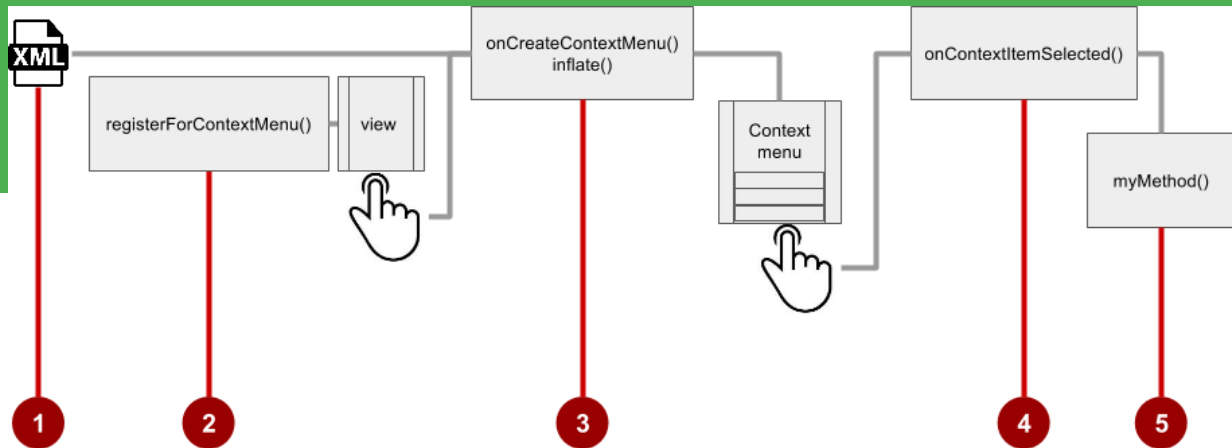
Types of contextual menus

- Floating context menu—long-press on a View
 - User can modify View or use it in some fashion
 - User performs action on one View at a time
- Contextual action mode—temporary action bar in place of or underneath app bar
 - Action items affect the selected View element(s)
 - User can perform action on multiple View elements at once



Floating Context Menu

Steps



1. Create XML menu resource file and assign appearance and position attributes
2. Register View using `registerForContextMenu()`
3. Implement `onCreateContextMenu()` in Activity to inflate menu
4. Implement `onContextItemSelected()` to handle menu item clicks
5. Create method to perform action for each context menu item

Create menu resource

1. Create XML menu resource (menu_context.xml)

```
<item  
    android:id="@+id/context_edit"  
    android:title="Edit"  
    android:orderInCategory="10"/>
```

```
<item  
    android:id="@+id/context_share"  
    android:title="Share"  
    android:orderInCategory="20"/>
```

Register a view to a context menu

In onCreate() of the Activity:

2. Register [View.OnCreateContextMenuListener](#) to View:

```
TextView article_text = findViewById(R.id.article);  
registerForContextMenu(article_text);
```

Implement onCreateContextMenu()

3. Specify which context menu

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                               ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_context, menu);
}
```


Implement onOptionsItemSelected()

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.context_edit:
            editNote();
            return true;
        case R.id.context_share:
            shareNote();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Contextual Action Bar

What is Action Mode?

- UI mode that lets you replace parts of normal UI interactions temporarily
- For example: Selecting a section of text or long-pressing an item could trigger action mode

Action mode has a lifecycle

- Start it with [`startActionMode\(\)`](#), for example, in the listener
- [`ActionMode.Callback`](#) interface provides lifecycle methods you override:
 - [`onCreateActionMode\(ActionMode, Menu\)`](#) once on initial creation
 - [`onPrepareActionMode\(ActionMode, Menu\)`](#) after creation and any time [`ActionMode`](#) is invalidated
 - [`onActionItemClicked\(ActionMode, MenuItem\)`](#) any time contextual action button is clicked
 - [`onDestroyActionMode\(ActionMode\)`](#) when action mode is closed

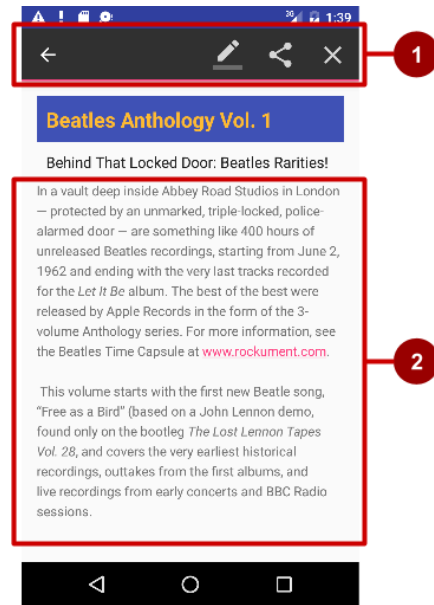
What is a contextual action bar?

Long-press on View shows contextual action bar

1. Contextual action bar with actions

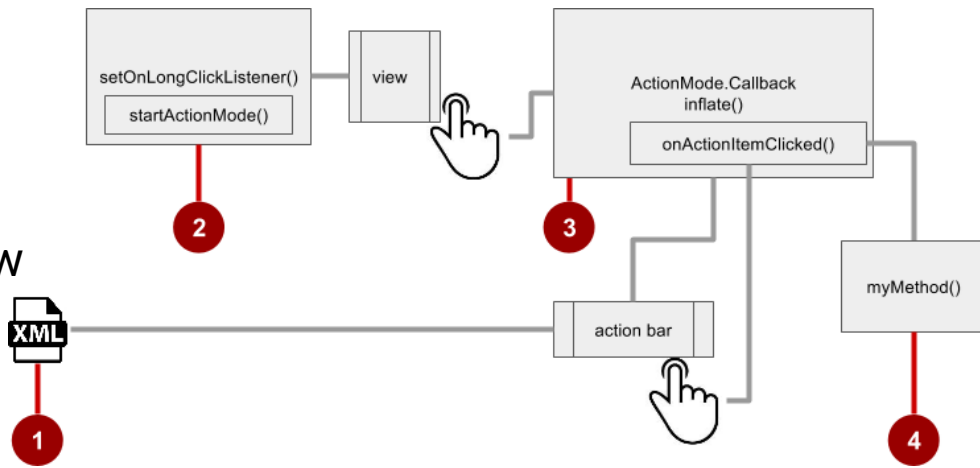
- **Edit, Share, and Delete**
- Done (left arrow icon) on left side
- Action bar is available until user taps Done

2. View on which long press triggers contextual action bar



Steps for contextual action bar

1. Create XML menu resource file and assign icons for items
2. `setOnLongClickListener()` on View that triggers contextual action bar and call `startActionMode()` to handle click



3. Implement `ActionMode.Callback` interface to handle `ActionMode` lifecycle; include action for menu item click in `onActionItemClicked()` callback
4. Create method to perform action for each context menu item

Use setOnLongClickListener

```
private ActionMode mActionMode;
```

In onCreate():

```
View view = findViewById(article);
view.setOnLongClickListener(new View.OnLongClickListener() {
    public boolean onLongClick(View view) {
        if (mActionMode != null) return false;
        mActionMode =
            MainActivity.this.startActionMode(mActionModeCallback);
        view.setSelected(true);
        return true;
    }
});
```

Implement mActionModeCallback

```
public ActionMode.Callback mActionModeCallback =  
    new ActionMode.Callback() {  
        // Implement action mode callbacks here.  
    };
```


Implement onCreateActionMode

```
@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
    MenuInflater inflater = mode.getMenuInflater();
    inflater.inflate(R.menu.menu_context, menu);
    return true;
}
```

Implement onPrepareActionMode

- Called each time action mode is shown
- Always called after onCreateActionMode, but may be called multiple times if action mode is invalidated

```
@Override  
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {  
    return false; // Return false if nothing is done.  
}
```

Implement onOptionsItemSelected

- Called when users selects an action
- Handle clicks in this method

@Override

```
public boolean onOptionsItemSelected(ActionMode mode, MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action_share:  
            // Perform action for the Share menu item.  
            mode.finish(); // Action picked, so close the action bar.  
            return true;  
        default:  
            return false;  
    }  
}
```

Implement onDestroyActionMode

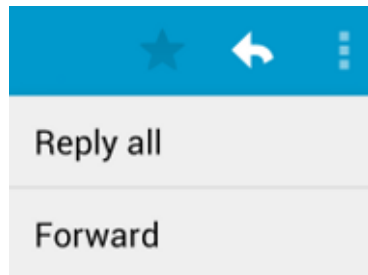
- Called when user exits the action mode

```
@Override  
public void onDestroyActionMode(ActionMode mode) {  
    mActionMode = null;  
}
```

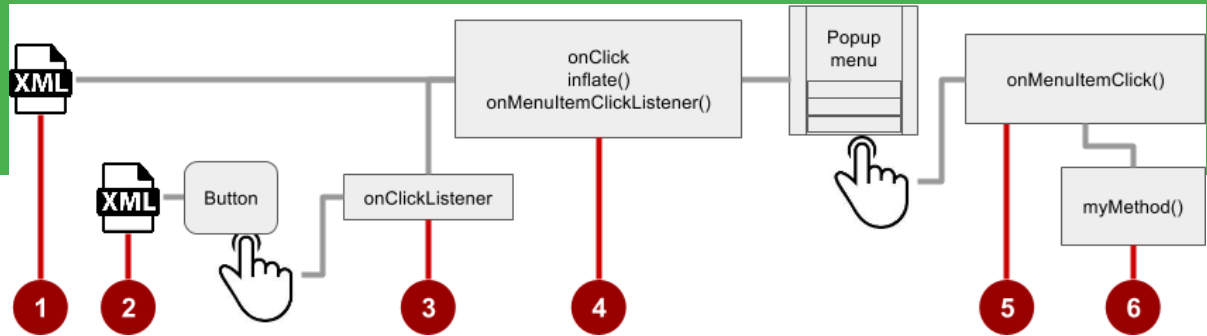
Popup Menu

What is a popup menu?

- Vertical list of items anchored to a view
- Typically anchored to a visible icon
- Actions should not directly affect view content
 - Options menu overflow icon that opens options menu
 - In email app, **Reply All** and **Forward** relate to email message but don't affect or act on message



Steps



1. Create XML menu resource file and assign appearance and position attributes
2. Add ImageButton for the popup menu icon in the XML activity layout file
3. Assign `onClick` listener to ImageButton
4. Override `onClick()` to inflate the popup and register it with `onMenuItemClickListener()`
5. Implement `onMenuItemClick()`
6. Create a method to perform an action for each popup menu item

Add ImageButton



```
<ImageButton  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/button_popup"  
    android:src="@drawable/@drawable/ic_action_popup"/>
```


Assign onClickListener to button

```
private ImageButton mButton =  
    (ImageButton) findViewById(R.id.button_popup);
```

In onCreate():

```
mButton.setOnClickListener(new View.OnClickListener() {  
    // define onClick  
});
```

Implement onClick

```
@Override
public void onClick(View v) {
    PopupMenu popup = new PopupMenu(MainActivity.this, mButton);
    popup.getMenuInflater().inflate(
        R.menu.menu_popup, popup.getMenu());
    popup.setOnMenuItemClickListener(
        new PopupMenu.OnMenuItemClickListener() {
            // implement click listener.
        });
    popup.show();
}
```

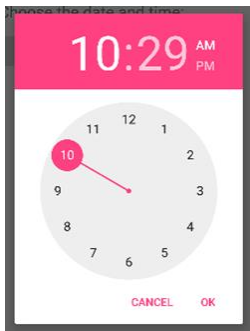
Implement onOptionsItemSelected

```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.option_forward:  
            // Implement code for Forward button.  
            return true;  
        default:  
            return false;  
    }  
}
```

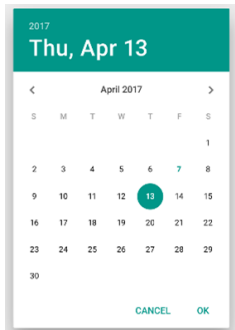
Dialogs

Dialogs

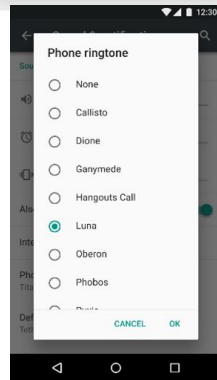
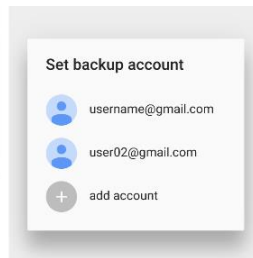
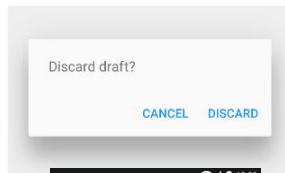
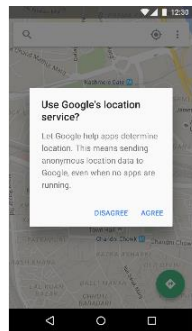
- Dialog appears on top, interrupting flow of Activity
- Requires user action to dismiss



TimePickerDialog



DatePickerDialog

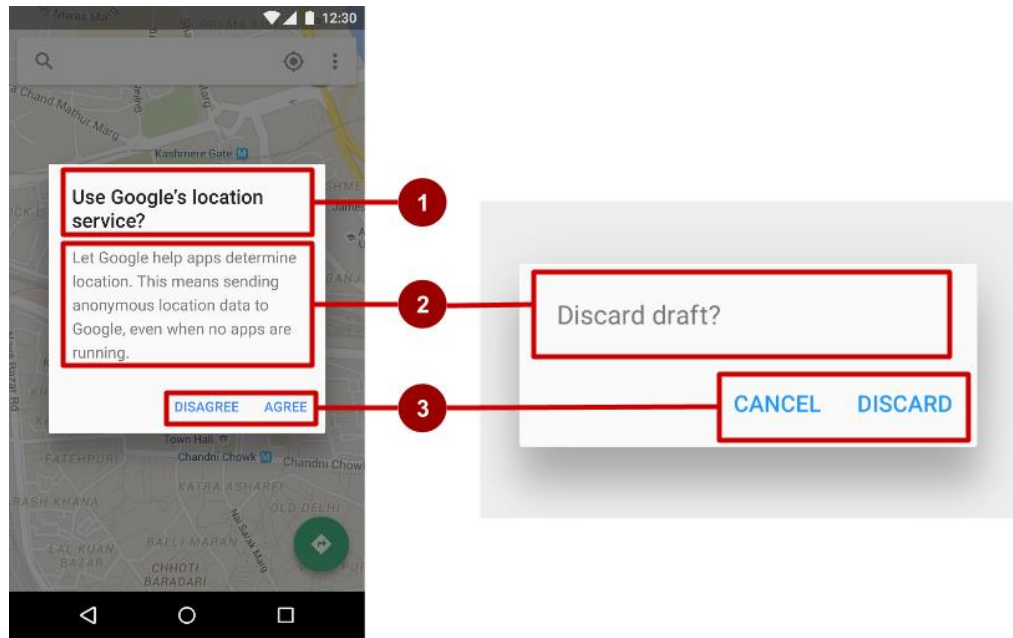


AlertDialog

AlertDialog

AlertDialog can show:

1. Title (optional)
2. Content area
3. Action buttons



Build the AlertDialog

Use `AlertDialog.Builder` to build alert dialog and set attributes:

```
public void onClickShowAlert(View view) {  
    AlertDialog.Builder alertDialog = new  
        AlertDialog.Builder(MainActivity.this);  
    alertDialog.setTitle("Connect to Provider");  
    alertDialog.setMessage(R.string.alert_message);  
    // ... Code to set buttons goes here.
```

Set the button actions

- `alertDialog.setPositiveButton()`
- `alertDialog.setNeutralButton()`
- `alertDialog.setNegativeButton()`

alertDialog code example

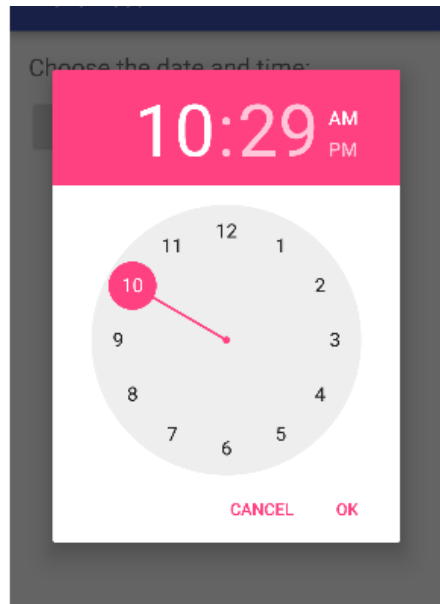
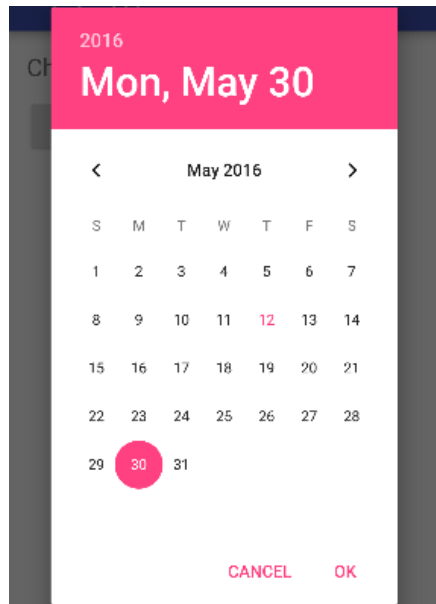
```
AlertDialog.setPositiveButton(  
    "OK", new DialogInterface.OnClickListener() {  
        public void onClick(DialogInterface dialog, int which) {  
            // User clicked OK button.  
        }  
    });
```

Same pattern for `setNegativeButton()` and `setNeutralButton()`

Pickers

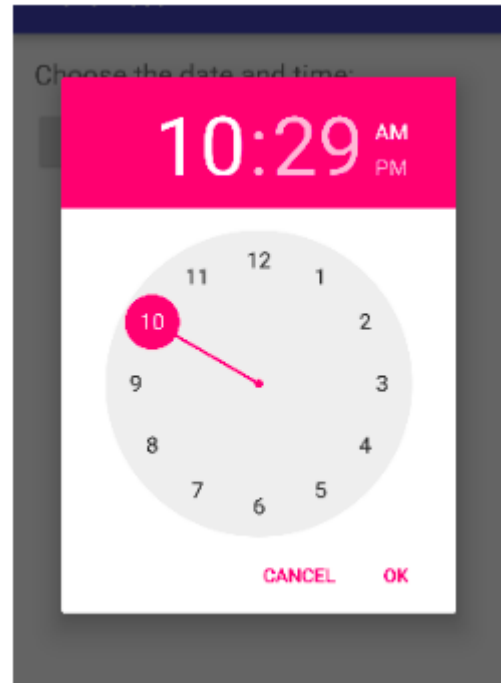
Pickers

- [DatePickerDialog](#)
- [TimePickerDialog](#)



Pickers use fragments

- Use [DialogFragment](#) to show a picker
- DialogFragment is a window that floats on top of Activity window



Introduction to fragments

- A [Fragment](#) is like a mini-Activity within an Activity
 - Manages its own own lifecycle
 - Receives its own input events
- Can be added or removed while parent Activity is running
- Multiple fragments can be combined in a single Activity
- Can be reused in more than one Activity

Creating a date picker dialog

1. Add a blank Fragment that extends `DialogFragment` and implements `DatePickerDialog.OnDateSetListener`
2. In `onCreateDialog()` initialize the date and return the dialog
3. In `onDateSet()` handle the date
4. In Activity show the picker and add method to use date

Creating a time picker dialog

1. Add a blank Fragment that extends `DialogFragment` and implements `TimePickerDialog.OnTimeSetListener`
2. In `onCreateDialog()` initialize the time and return the dialog
3. In `onTimeSet()` handle the time
4. In Activity, show the picker and add method to use time

Learn more

- [Adding the App Bar](#)
- [Menus](#)
- [Menu Resource](#)
- [Fragments](#)
- [Dialogs](#)
- [Pickers](#)
- [Drawable Resources](#)

What's Next?

- Concept Chapter: [4.3 Menus and pickers](#)
- Practical: [4.3 Menus and pickers](#)

END