MultiSceen, Localization and Menus

EE5415

Mobile Apps Design and Development

Content

- Learn how to make your app support different screen form factors and languages and
- Learn how to create options and context menus
- Lab05: Localization and Menus
  - Android localization Method and localize BMI App
  - Option Menu and Context Menu
  - Create Option Menu and Context Menu for the BMI App
  - Review questions

Multiscreen Concepts

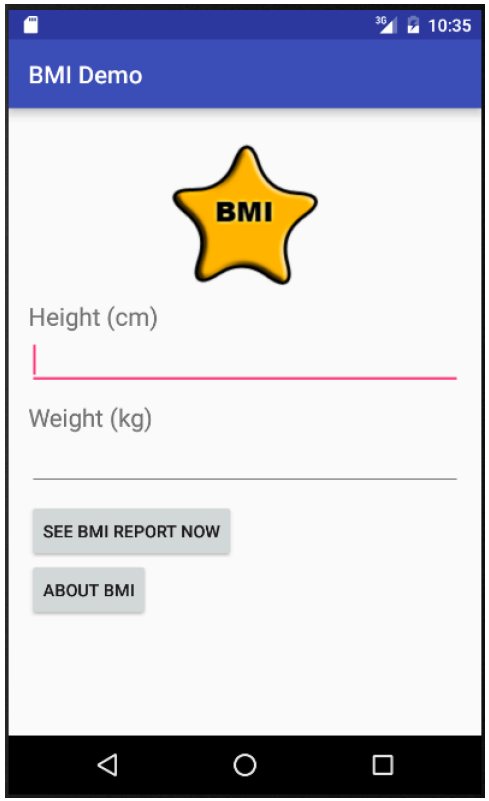
- Orientation : Portrait and landscape
- Screen Size (inches)
- Resolution or Screen Density (dpi or ppi)
- Density-independent pixel (dp)

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	**Total**
Small	2.4%						2.4%
Normal		5.1%	0.1%	41.5%	22.9%	14.8%	84.4%
Large	0.3%	5.0%	2.3%	0.6%	0.5%		8.7%
Xlarge		3.5%		0.3%	0.7%		4.5%
Total	2.7%	13.6%	2.4%	42.4%	24.1%	14.8%	

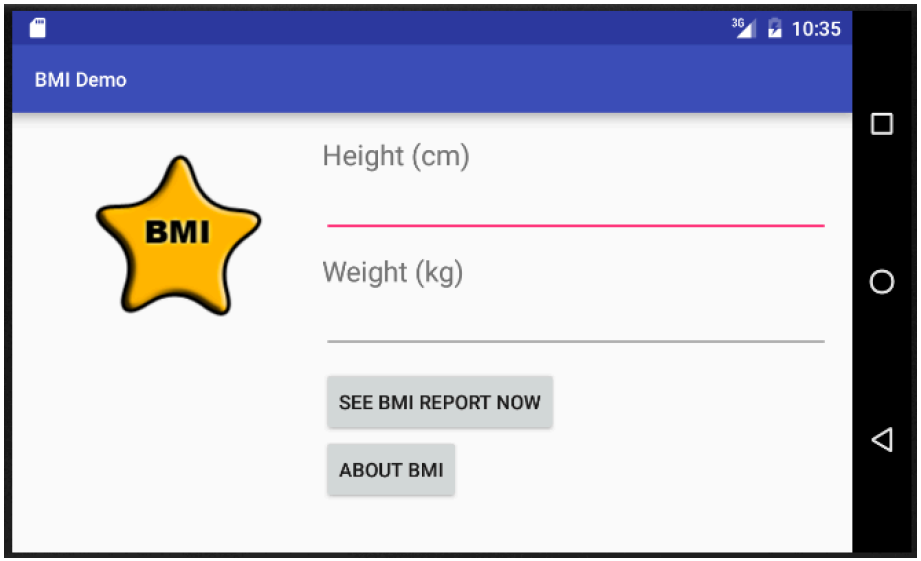
Orientations

- The orientation of the screen from the user's point of view.

Portrait

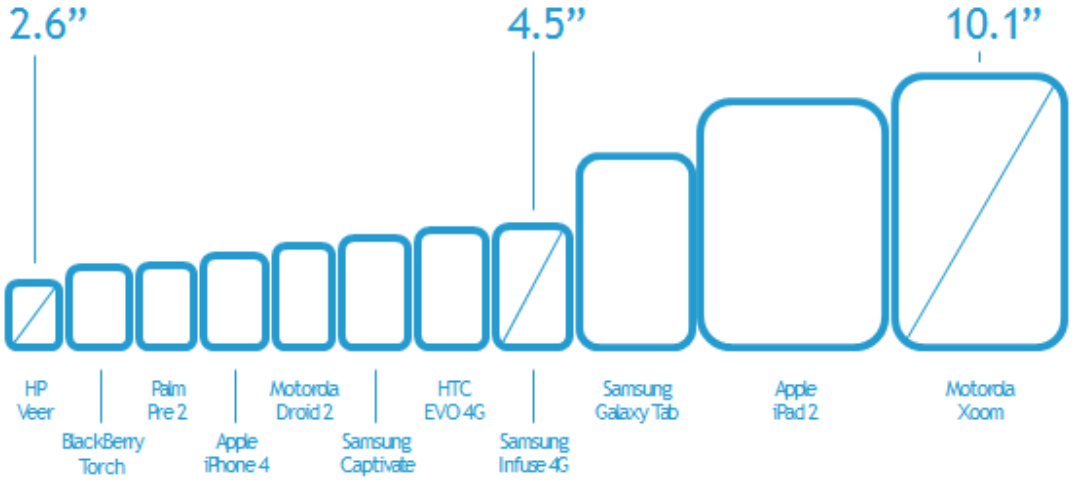


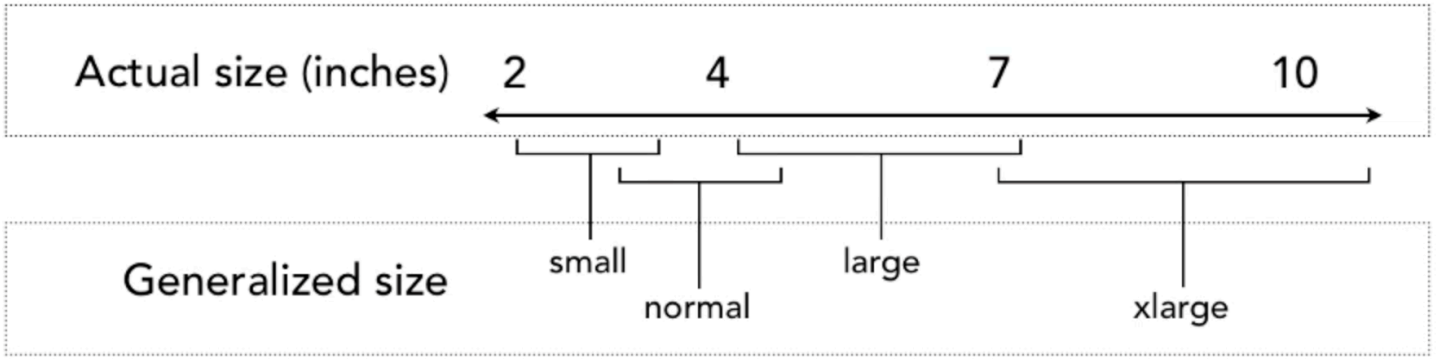
Landscape



Screen Size

- Actual physical size, measured as the screen's diagonal.
- A set of 4 generalized sizes:
  - small
  - normal
  - large
  - xlarge (Extra-Large)



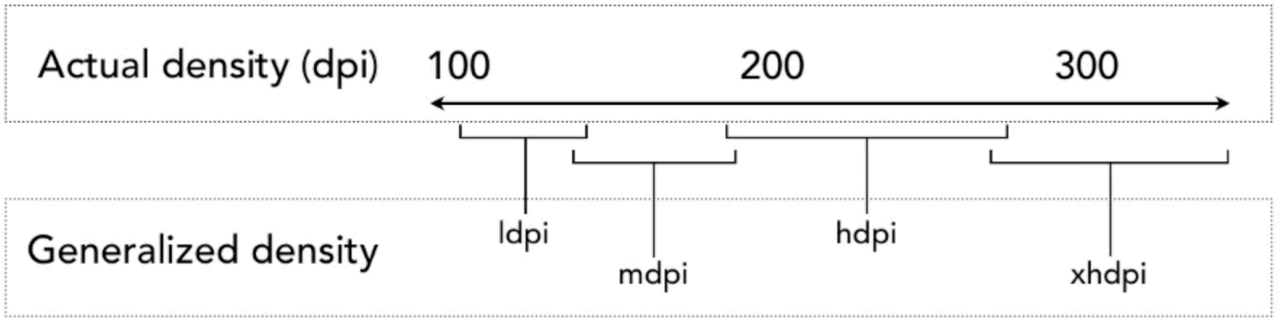


Resolution

- The total number of physical pixels on a screen.
  - How densely those pixels are packed
- Usually referred as Dots Per Inch (dpi) or Pixels Per Inch (ppi).
- When adding support for multiple screens, apps do not work directly with resolution; apps should be concerned only with screen size and density.

Screen Density in dpi

- In Android, resolution is referred as Screen Density with unit of dpi.
- A set of six generalized densities:
  - ldpi (low) ~120dpi
  - mdpi (medium) ~160dpi
  - hdpi (high) ~240dpi
  - xhdpi (extra-high) ~320dpi
  - xxhdpi (extra-extra-high) ~480dpi
  - xxxhdpi (extra-extra-extra-high) ~640dpi



Android Screen Configuration Qualifiers

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	2.4%						2.4%
Normal		5.1%	0.1%	41.5%	22.9%	14.8%	84.4%
Large	0.3%	5.0%	2.3%	0.6%	0.5%		8.7%
Xlarge		3.5%		0.3%	0.7%		4.5%
Total	2.7%	13.6%	2.4%	42.4%	24.1%	14.8%	

Screen characteristic	Qualifier
Size	`small`
	`normal`
	`large`
	`xlarge`
Density	`ldpi`
	`mdpi`
	`hdpi`
	`xhdpi`
	`nodpi`
Orientation	`land`
	`port`
Aspect ratio	`long`
	`notlong`

Examples of Configuration Qualifiers

```
res/layout/my_layout.xml              // layout for normal screen size ("default")
res/layout-small/my_layout.xml        // layout for small screen size
res/layout-large/my_layout.xml        // layout for large screen size
res/layout-xlarge/my_layout.xml       // layout for extra large screen size
res/layout-xlarge-land/my_layout.xml  // layout for extra large in landscape orientation

res/drawable-mdpi/my_icon.png         // bitmap for medium density
res/drawable-hdpi/my_icon.png         // bitmap for high density
res/drawable-xhdpi/my_icon.png        // bitmap for extra high density
```

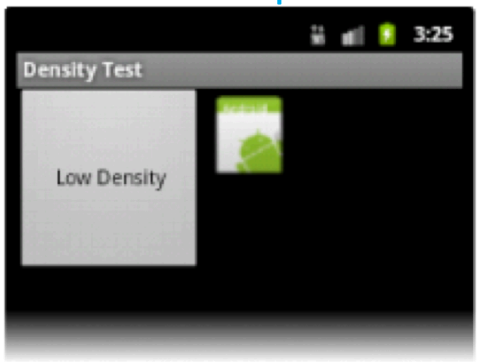
Density-independent pixel (dp)

- A virtual pixel unit to express layout dimensions or position in a density-independent way.
- The density-independent pixel is equivalent to one physical pixel on a 160 dpi screen
- The conversion of dp units to screen pixels is simple: px = dp * (dpi / 160).
- For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels.

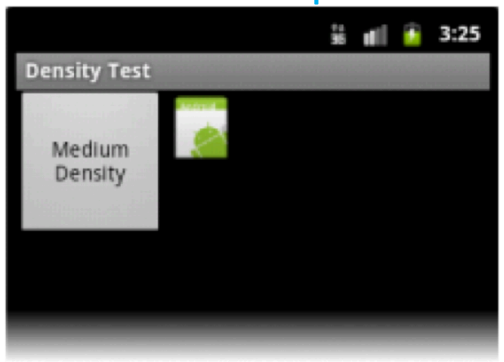
px and dp on Multiscreen

- UI elements using px to specify the size

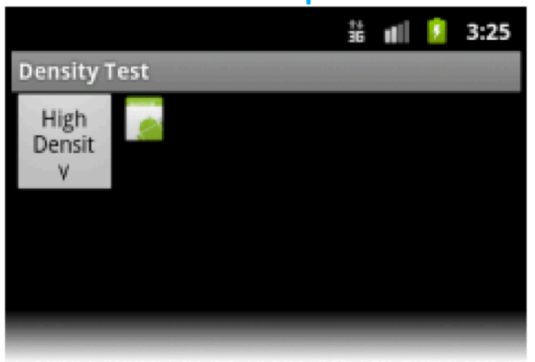
160 dpi



240 dpi

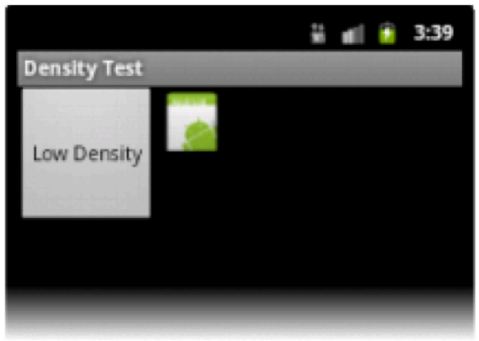


320 dpi

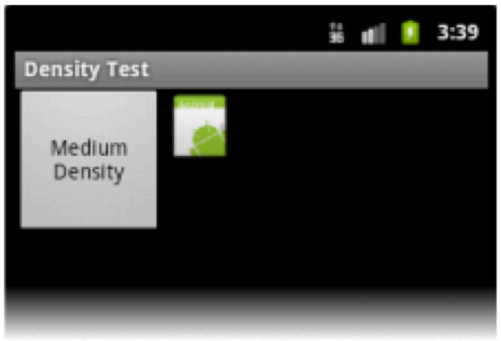


- UI elements using dp to specify the size

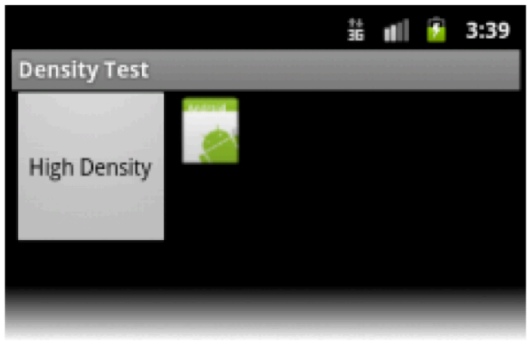
160 dpi



240 dpi



320 dpi



Best Practices for Screen Independence

- Use wrap_content, match_parent, or dp units when specifying in an XML layout file
- Do not use hard coded pixel values in your application code
- Supply alternative bitmap drawables for different screen densities

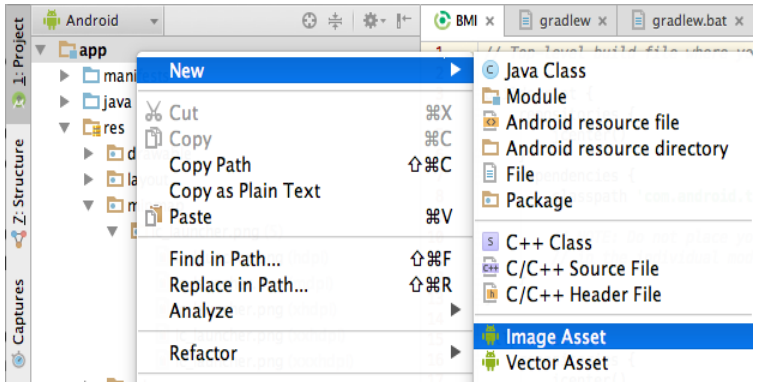
Creating Launcher Icons

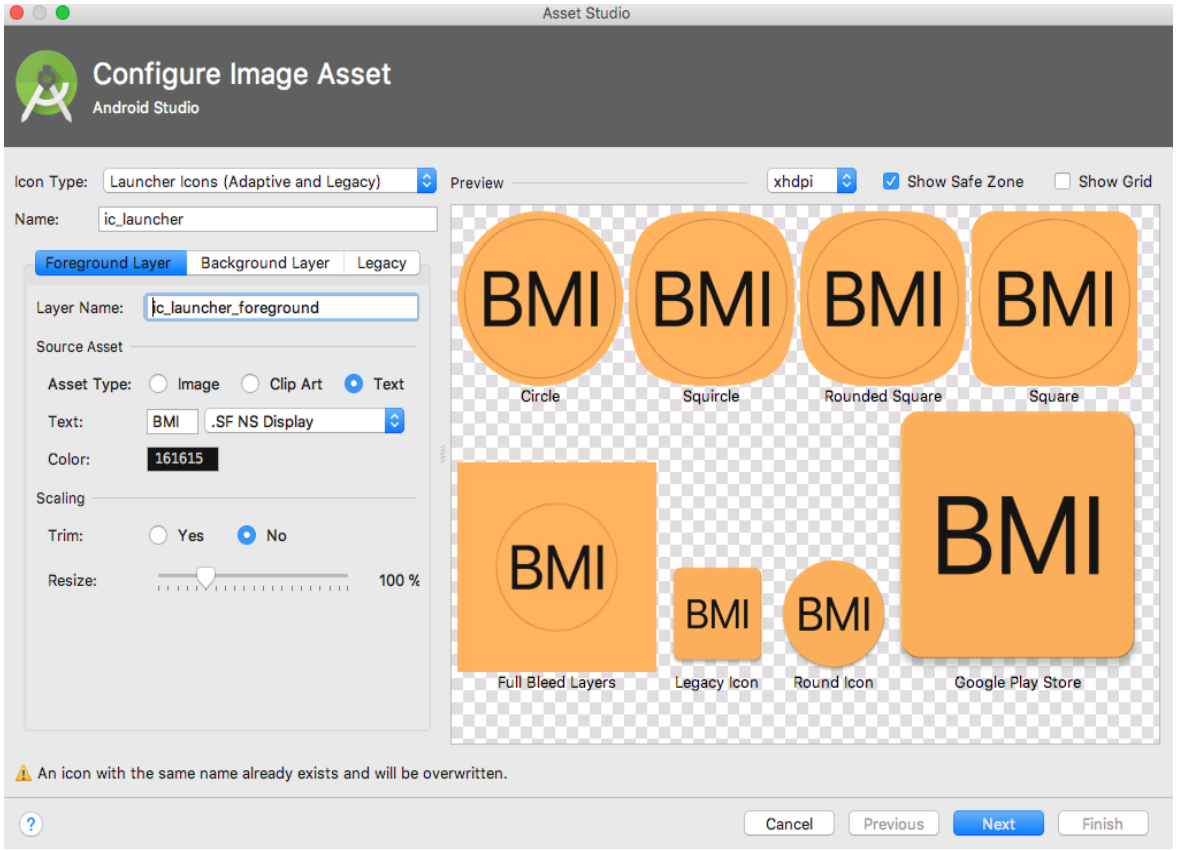
- The default Launcher icons with multiple resolutions are stored in
  - res/mipmap-mdpi/ic_launcher.png
  - res/mipmap-hdpi/ic_launcher.png
  - res/mipmap-xhdpi/ic_launcher.png
  - res/mipmap-xxhdpi/ic_launcher.png
  - res/mipmap-xxxhdpi/ic_launcher.png

- Android Studio provides a very useful tools to create launcher icons with multiple resolutions
- This is “Android Asset Studio”

Android Asset Studio (1)

- File -> New -> Image Asset

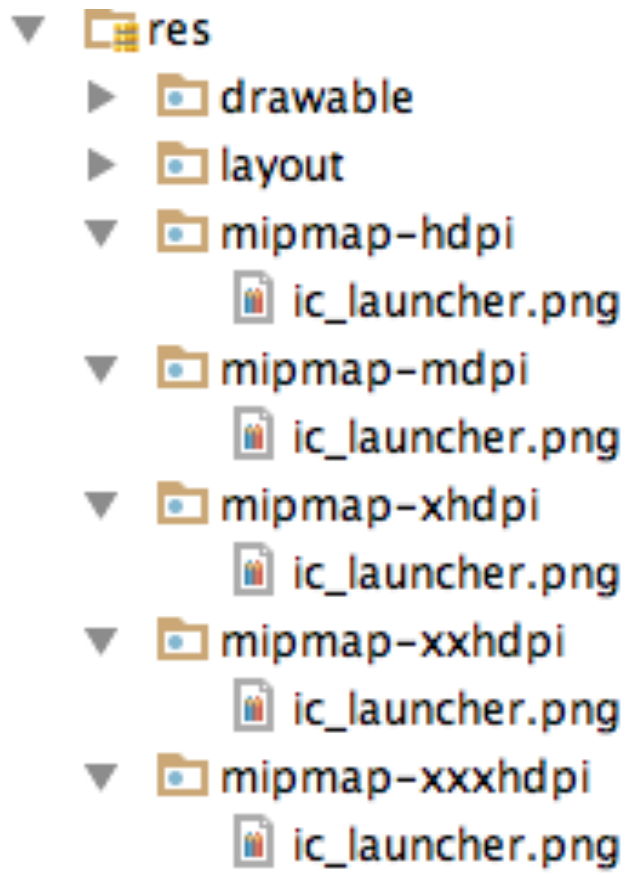




Android Asset Studio (2)

These image files are specified for different screen density with sizes of

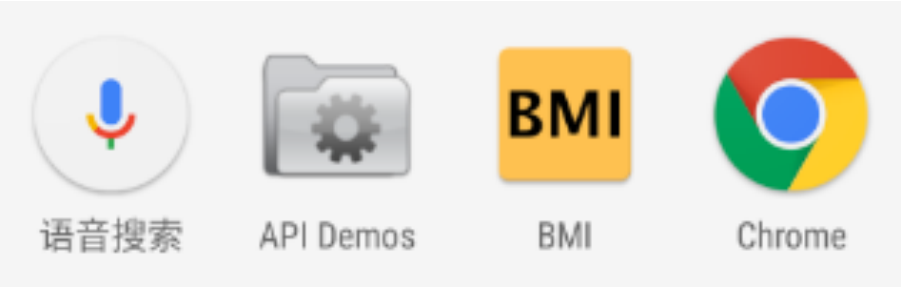
- 48x48 (mdip)
- 72x72 (hdip)
- 96x96 (xhdip)
- 144x144 (xxhdip)
- 192x192 (xxxhdip).



AndroidManifest.xml

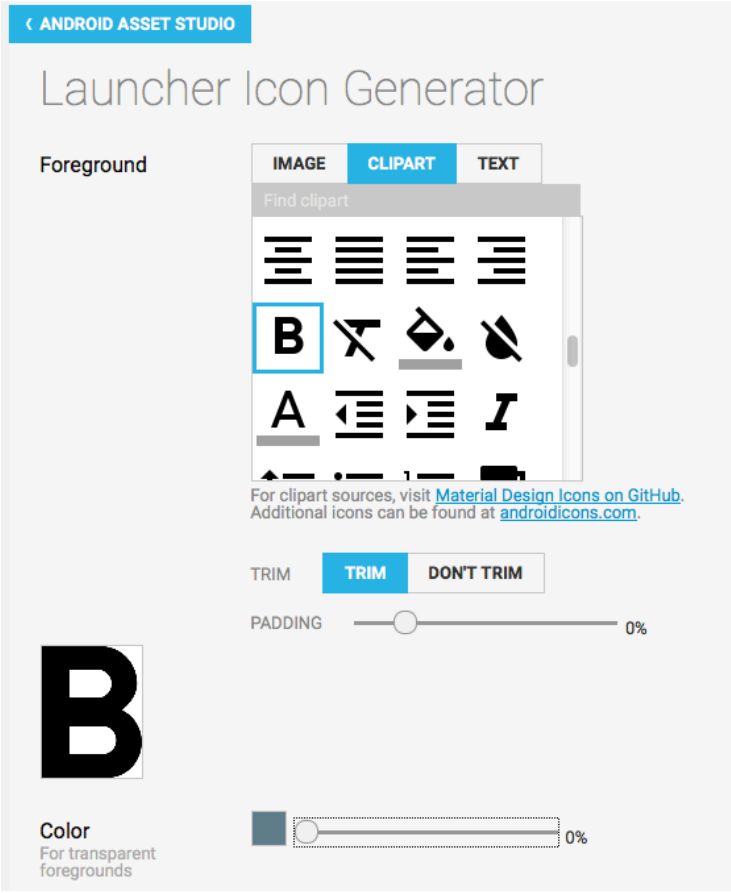
- To use these image files as launcher icon for your app, you have to specify this resource in the AndroidManifest.xml

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="BMI"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".ReportActivity"></activity>
```



Android Asset Studio

- Icon generators allow you to quickly and easily generate icons from existing source images, clipart, or text.
- https://romannurik.github.io/AndroidAssetStudio/



Android Localization

Android Localization

- To build multilingual Android apps you need to collect the texts into resource files and translate them.
- Once you provide the translation, Android OS will choose the resources that match user's locale.
- If your app is available in several languages, Android will select the language that the device uses.

Use the String Resources

- In Java code, we can refer a string resource:
  - **R.string.<string_name>**

```
// Get a string resource from your app's Resources
      String hello = getResources().getString(R.string.hello_world);

// Or supply a string resource to a method that requires a string
      TextView textView = new TextView(this);
      textView.setText(R.string.hello_world);
```

- In XML, we can refer a string resource:
  - **@string/<string_name>**

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

Strings.xml files

- In order to have a multilingual Android app you need to provide Android with the localized resource files.
- If the locale is *'en-US'*, Android will look for a value of *"R.string.title"* by searching the files in the following order:
  - res/values-en-rUS/strings.xml
  - res/values-en/strings.xml
  - res/values/strings.xml
- This means that the default language acts as a fallback option and when translation exists, it's used instead.

Supporting Different Languages

- It’s always a good practice to extract UI strings from your app code and keep them in an external file.
- Android makes this easy with a resources directory in each Android project

English (default locale), /values/strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
        <string name="title" >My Application</string>
        <string name="hello_world" >Hello World!</string>
</resources>
```

Simplified Chinese character, /values-zh/strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
        <string name="title" >我的应用程序</string>
        <string name="hello_world" >你好世界！（中国）</string>
</resources>
```

Localization Qualifiers

Locale Code	Language/Country	Location of strings.xml	Location of image files
Default	English/ UK	/res/values	/res/drawable
zh-rCN	Simplified Chinese/Mainland China	/res/values-zh-rCN	/res/drawable-zh-rCH
zh-rTW	Traditional Chinese/Taiwan (China)	/res/values-zh-rTW	/res/drawable-zh-rTW
fr-rFR	French/France	/res/values-fr	/res/drawable-fr-rFR
en-rUS	English/US	/res/values	/res/values/en-rUS
es-rVE	Spanish/Venezuela	/res/values-es	/res/values/es-rVE

Localize the BMI App

Options and Context Menus

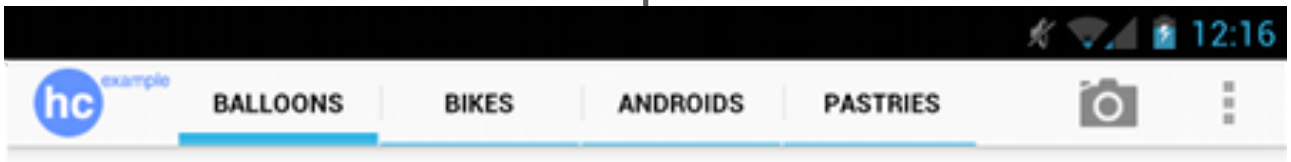
Two types of Android Menus

- The Android menus are quite different from those in PC applications. There are mainly two types of menus in Android.

- **Options Menu:**
  - It is invoked by pressing the **Menu Button**, and contains the primary menu items for an activity. When invoked, it is often displayed at the bottom of the screen.
- **Context Menu:**
  - It is invoked by pressing and holding (also called long-pressing) a view. It appears as a floating list of menu items in the middle of the screen.

Options Menu and Action Bar

- On Android 3.0 and higher, items from the options menu are presented by the action bar as a combination of on-screen action items and overflow options.
- Beginning with Android 3.0, the *Menu* button is deprecated (some devices don't have one), so you should migrate toward using the action bar to provide access to actions and other options.

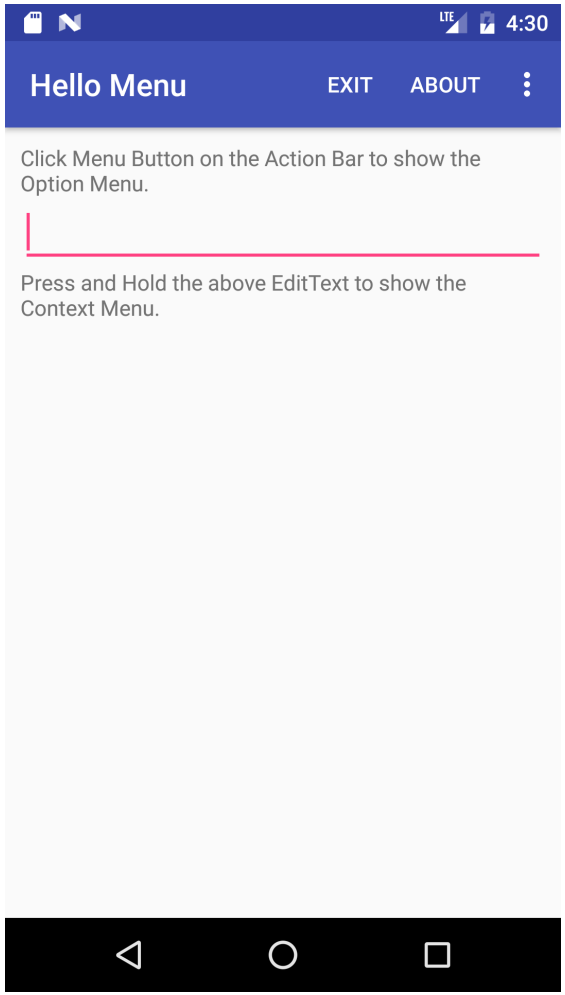


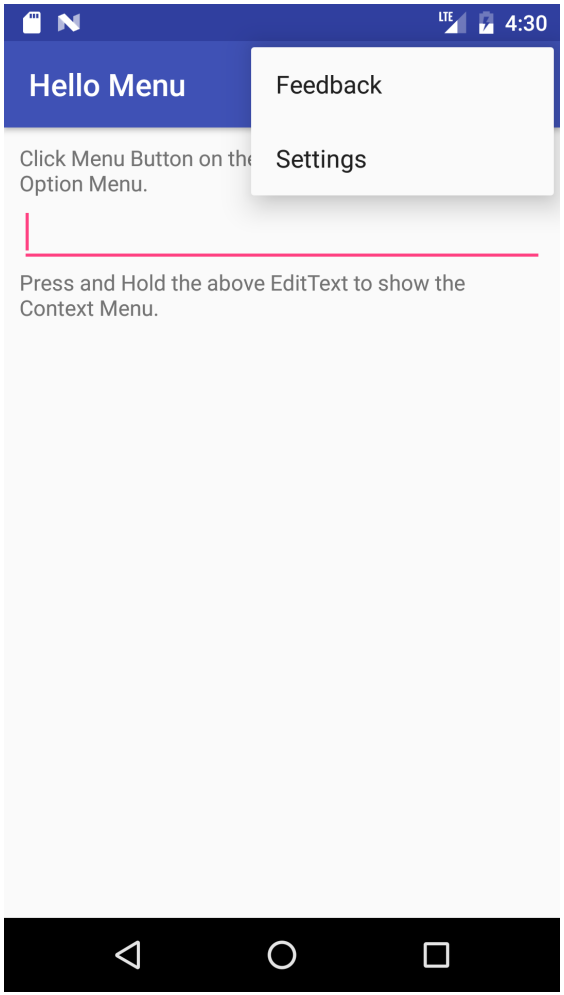
Action bar from the Honeycomb Gallery app, showing navigation tabs and a camera action item (plus the action overflow button).

Options menu in the Browser, on Android 2.3.

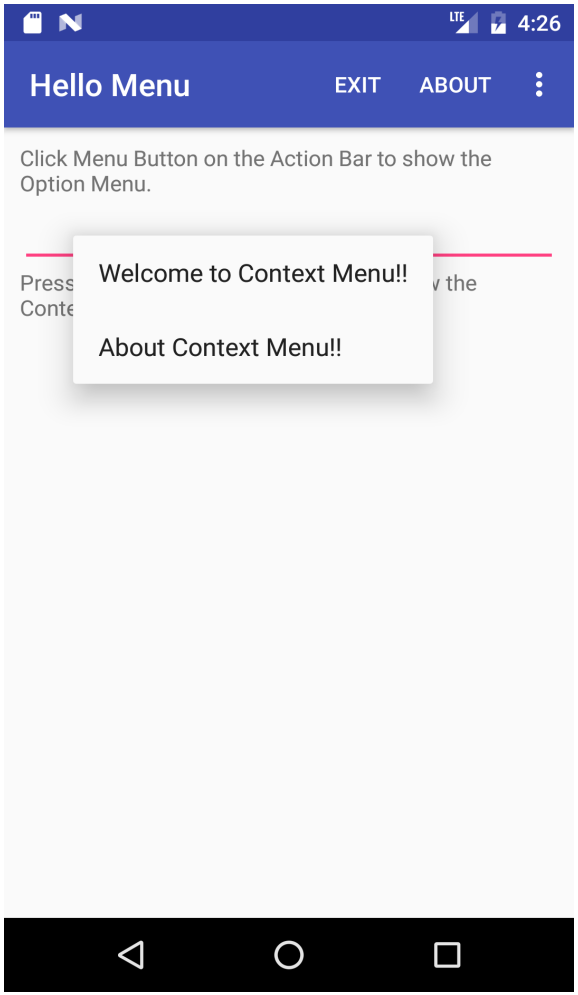
Example Project for Options and Context Menus

Options Menu



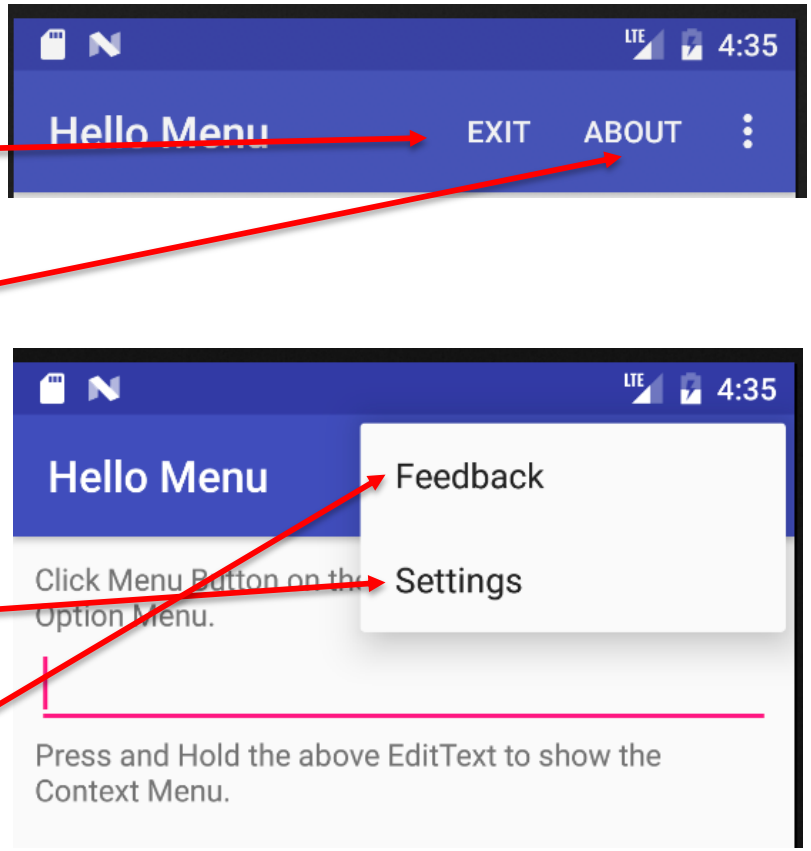


Context Menu



res/menu/menu_options.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">
    <item
        android:id="@+id/menu_exit"
        android:title="@string/exit"
        app:showAsAction="ifRoom" />

    <item
        android:id="@+id/menu_about"
        android:title="@string/about"
        android:orderInCategory="1"
        app:showAsAction="always" />

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/menu_settings"
        app:showAsAction="never" />

    <item
        android:id="@+id/menu_feedback"
        android:title="@string/feedback" />
</menu>
```



Inflating the option menu resource

- Inflating a menu resource (`res/menu/`**`menu_options.xml`**) by adding **`onCreateOptionsMenu(Menu menu)`** in the main Activity.
- Menu items in menu_options.xml will appear when the user touches the MENU button

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the aaction bar if it is present
    getMenuInflater().inflate(R.menu.menu_options, menu);
    return true;
}
```

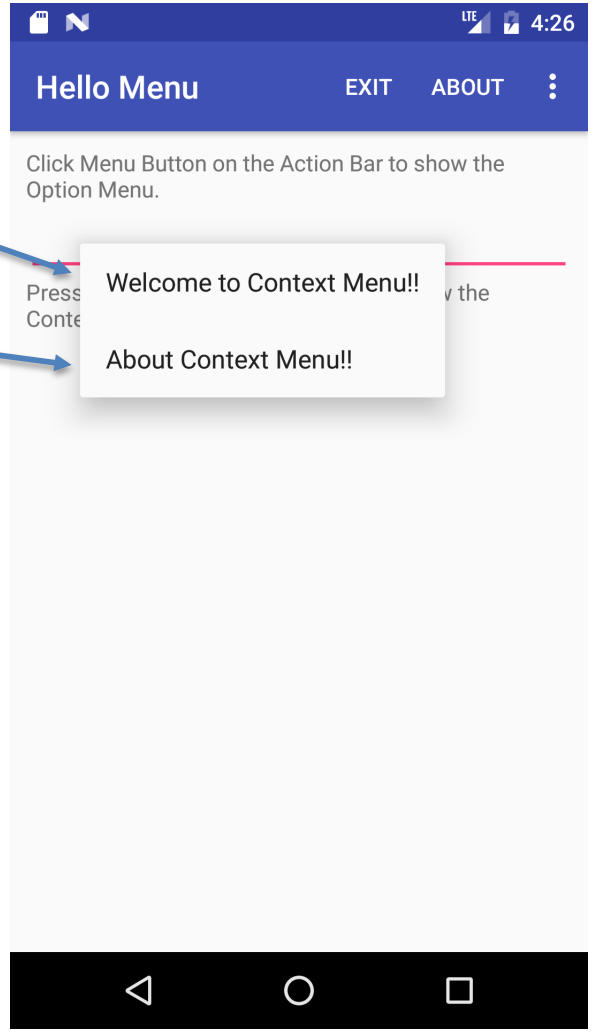
Response to user action

- Response to menu click events by overriding **`onOptionsItemSelected(Menu menu)`** in the main Activity.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    int id = item.getItemId();

    switch (id) {
        case R.id.action_settings:
            Toast.makeText(this, "Settings Button Clicked !",
                    Toast.LENGTH_LONG).show();
            return true;
        case R.id.menu_about:
            Toast.makeText(this, "About Button Clicked !",
                    Toast.LENGTH_LONG).show();
            return true;
        case R.id.menu_feedback:
            Toast.makeText(this, "Send Feedback Button Clicked !",
                    Toast.LENGTH_LONG).show();
            return true;
        case R.id.menu_exit:
            finish();
            return true;
    }
    return false;
}
```

res/menu/menu_context.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/menuItemWelcome"
        android:title="Welcome to Context Menu!!" />
    <item
        android:id="@+id/menuItemAbout"
        android:title="About Context Menu!!" />
</menu>
```



- This context menu is registered for the EditText
- Long press of the EditText, the context menu will appear

Register View for a context menu

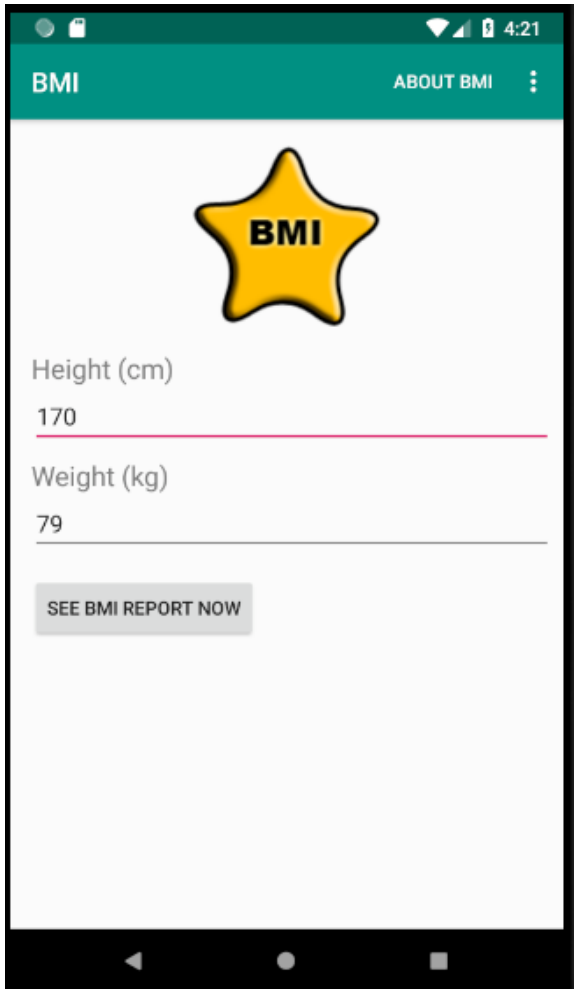
- By calling **`registerForContextMenu()`** and passing it a View (an EditText in this example) you assign it a context menu.
- When this View (EditText) receives a long-press, it displays a context menu.

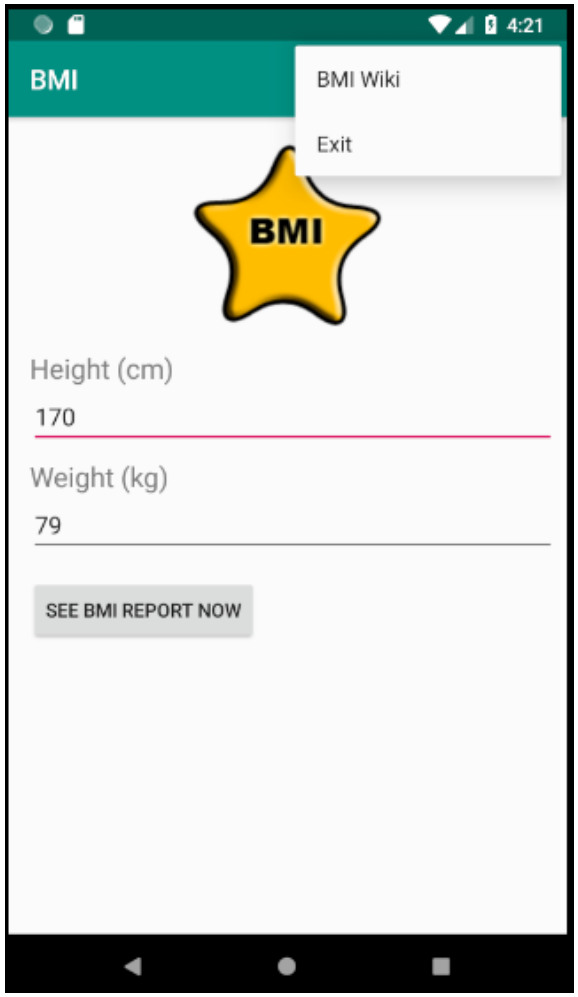
```
public class MainActivity extends AppCompatActivity {

    private EditText mOutEditText;

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

        mOutEditText = (EditText) findViewById(R.id.editText);
        registerForContextMenu(mOutEditText);
    }
```

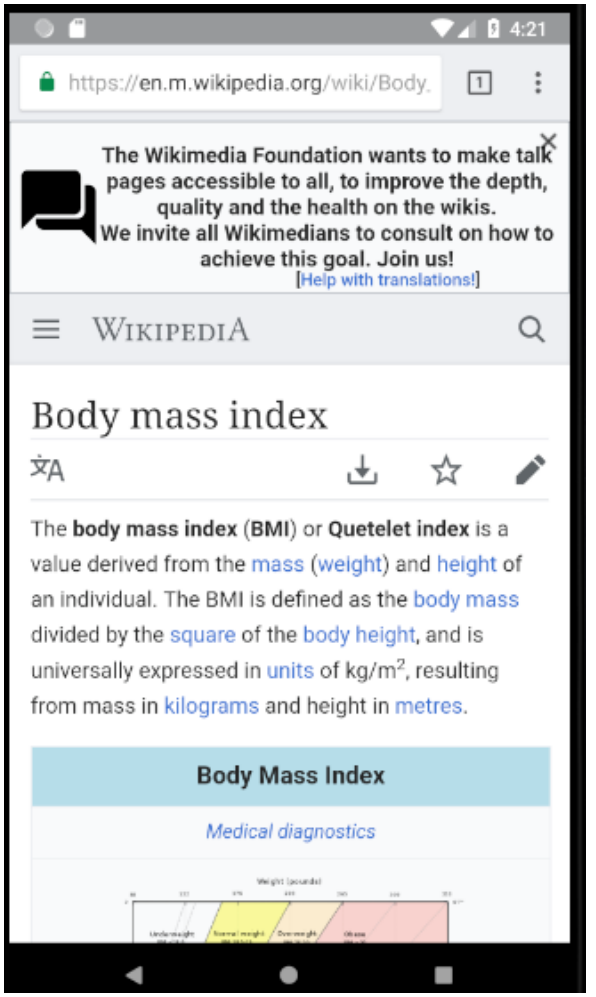
Inflate and define the behavior of the context menu

```
@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);
}

@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menuItemWelcome:
            mOutEditText.setText( "Welcome to Context Menu!!" );
            return true;
        case R.id.menuItemAbout:
            mOutEditText.setText( "About Context Menu!!" );
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

Create Options Menu for BMI App

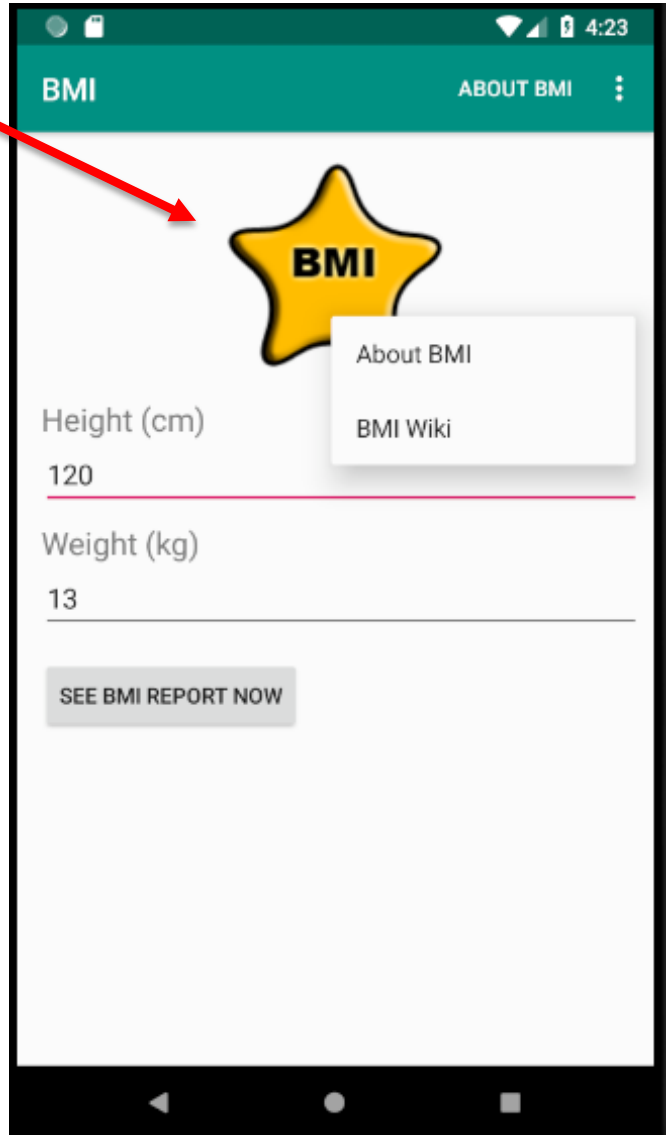






Create Context Menu for BMI App

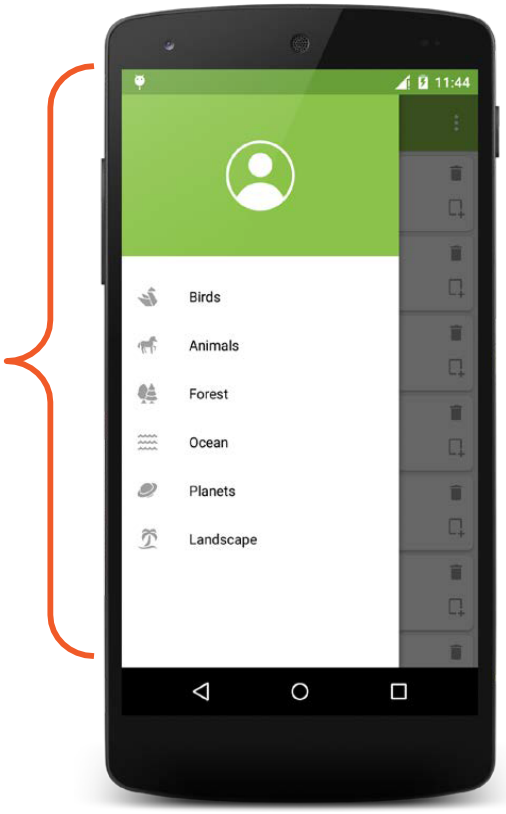
- Long press the BMI Icon Image to display the context menu

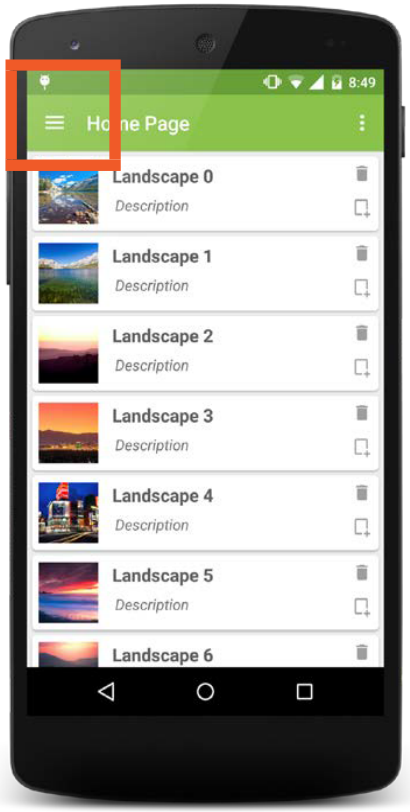


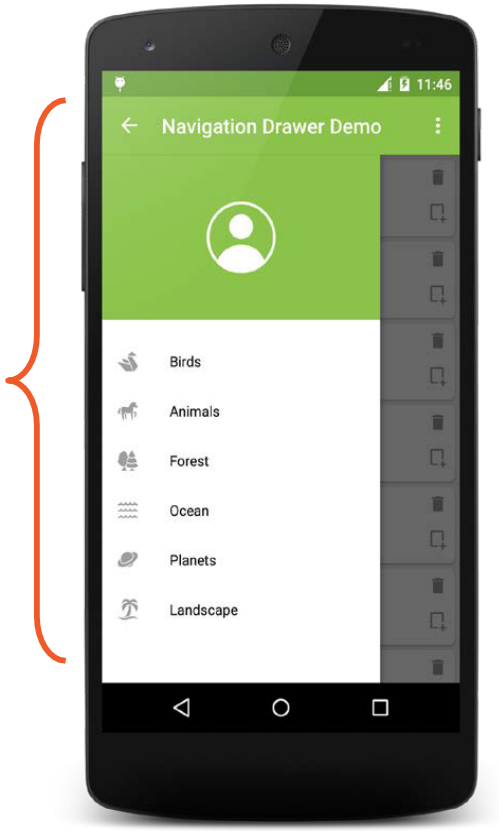
Chinese Menus

Sidebar Navigation Drawer

- Two Ways to Implement Navigation Drawer







Full Height

Clipped

https://www.youtube.com/watch?v=ofu1IqiBNCY