Basics

Apps provide multiple entry points

- Android apps are built as a combination of components that can be invoked individually. + an activity is a type of app component that provides a user interface (UI).
- The "main" activity starts when the user taps your app's icon
- You can direct the user to an activity from elsewhere (eg. notification, different app)
- Other components, such as broadcast receivers and services, allow your app to perform background tasks without a UI.

Apps adapt to different devices

- different resources for different devices. For example, you can create different layouts for different screen sizes.
- If any of your app's features need specific hardware, such as a camera, you can query at runtime whether the device has that hardware or not, and then disable the corresponding features if it doesn't.

Project files

- app > java > com.example.myfirstapp > MainActivity + entry point for your app. When you build and run your app, the system launches an instance of this Activity and loads its layout
- app > res > layout > activity_main.xml + XML file that defines the layout for the activity's user interface (UI)
- app > manifests > AndroidManifest.xml + The manifest file describes the fundamental characteristics of the app and defines each of its components.
- Gradle Scripts > build.gradle + build configurations - plain text files that use Domain Specific Language (DSL) to describe and manipulate the build logic using Groovy

User Interface

The user interface (UI) for an Android app is built as a hierarchy of layouts and widgets. The layouts are `ViewGroup` objects, containers that control how their child views are positioned on the screen. Widgets are `View` objects, UI components such as buttons and text boxes.

[viewgroup_2x.png] | *https://developer.android.com/images/viewgroup_2x.png*

Android provides an XML vocabulary for `ViewGroup` and `View` classes, so most of your UI is defined in XML files.

Topics

App Manifest

Every app project must have an AndroidManifest.xml file (with precisely that name) at the root of

the project source set. The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.

App components

For each app component that you create in your app, you must declare a corresponding XML element in the manifest file:

- <activity> for each subclass of Activity.
- <service> for each subclass of Service.
- <receiver> for each subclass of BroadcastReceiver.
- <provider> for each subclass of ContentProvider.

Reference

Actvity

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(View)`. While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows, Multi-Window mode or embedded into other windows. There are two methods almost all subclasses of Activity will implement:

- `onCreate(Bundle) is where you initialize your activity. Most importantly, here you will usually call `setContentView(int) with a layout resource defining your UI, and using `findViewById(int)`` to retrieve the widgets in that UI that you need to interact with programmatically.
- `onPause()` is where you deal with the user pausing active interaction with the activity. Any changes made by the user should at this point be committed (usually to the `ContentProvider` holding the data). In this state the activity is still visible on screen.

To be of use with `Context.startActivity()`, all activity classes must have a corresponding `<activity>` declaration in their package's `AndroidManifest.xml`.

TextView

A user interface element that displays text to the user. To provide user-editable text, see EditText.

The following code sample shows a typical use, with an XML layout and code to modify the contents of the text view:

```
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="match_parent"
     android:layout_height="match_parent">
   <TextView
       android:id="@+id/text_view_id"
       android:layout_height="wrap_content"
       android:layout_width="wrap_content"
       android:text="@string/hello" />
</LinearLayout>
```

This code sample demonstrates how to modify the contents of the text view defined in the previous XML layout:

```
public class MainActivity extends Activity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final TextView helloTextView = (TextView) findViewById(R.id.text_view_id);
        helloTextView.setText(R.string.user_greeting);
    }
}
```

To customize the appearance of TextView, see Styles and Themes.