**Introduction to Android**

Android provides a rich application framework that allows you to build innovative apps and games for mobile devices in a Java language environment. The documents listed in the left navigation provide details about how to build apps using Android's various APIs.

If you're new to Android development, it's important that you understand the following fundamental concepts about the Android app framework:

**Apps provide multiple entry points**

Android apps are built as a combination of distinct components that can be invoked individually. For instance, an individual *activity* provides a single screen for a user interface, and a *service* independently performs work in the background.

From one component you can start another component using an *intent*. You can even start a component in a different app, such an activity in a maps app to show an address. This model provides multiple entry points for a single app and allows any app to behave as a user's "default" for an action that other apps may invoke.

#### Apps adapt to different devices

Android provides an adaptive app framework that allows you to provide unique resources for different device configurations. For example, you can create different XML layout files for different screen sizes and the system determines which layout to apply based on the current device's screen size.

You can query the availability of device features at runtime if any app features require specific hardware such as a camera. If necessary, you can also declare features your app requires so app markets such as Google Play Store do not allow installation on devices that do not support that feature.

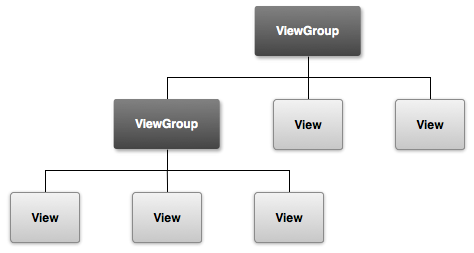
# **UI Overview**

All user interface elements in an Android app are built using [View](https://developer.android.com/reference/android/view/View.html) and [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) objects. A [View](https://developer.android.com/reference/android/view/View.html) is an object that draws something on the screen that the user can interact with. A [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) is an object that holds other [View](https://developer.android.com/reference/android/view/View.html) (and [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html)) objects in order to define the layout of the interface.

Android provides a collection of both [View](https://developer.android.com/reference/android/view/View.html) and [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) subclasses that offer you common input controls (such as buttons and text fields) and various layout models (such as a linear or relative layout).

## **User Interface Layout**

The user interface for each component of your app is defined using a hierarchy of [View](https://developer.android.com/reference/android/view/View.html) and [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) objects, as shown in figure 1. Each view group is an invisible container that organizes child views, while the child views may be input controls or other widgets that draw some part of the UI. This hierarchy tree can be as simple or complex as you need it to be (but simplicity is best for performance).



**Figure 1.** Illustration of a view hierarchy, which defines a UI layout.

To declare your layout, you can instantiate [View](https://developer.android.com/reference/android/view/View.html) objects in code and start building a tree, but the easiest and most effective way to define your layout is with an XML file. XML offers a human-readable structure for the layout, similar to HTML.

The name of an XML element for a view is respective to the Android class it represents. So a <TextView> element creates a [TextView](https://developer.android.com/reference/android/widget/TextView.html) widget in your UI, and a <LinearLayout> element creates a [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) view group.

For example, a simple vertical layout with a text view and a button looks like this:

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
              android:layout\_width="fill\_parent"   
              android:layout\_height="fill\_parent"  
              android:orientation="vertical" >  
    <TextView android:id="@+id/text"  
              android:layout\_width="wrap\_content"  
              android:layout\_height="wrap\_content"  
              android:text="I am a TextView" />  
    <Button android:id="@+id/button"  
            android:layout\_width="wrap\_content"  
            android:layout\_height="wrap\_content"  
            android:text="I am a Button" />  
</LinearLayout>

When you load a layout resource in your app, Android initializes each node of the layout into a runtime object you can use to define additional behaviors, query the object state, or modify the layout.

For a complete guide to creating a UI layout, see [XML Layouts](https://developer.android.com/guide/topics/ui/declaring-layout.html).

## **User Interface Components**

You don't have to build all of your UI using [View](https://developer.android.com/reference/android/view/View.html) and [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) objects. Android provides several app components that offer a standard UI layout for which you simply need to define the content. These UI components each have a unique set of APIs that are described in their respective documents, such as [Action Bar](https://developer.android.com/guide/topics/ui/actionbar.html), [Dialogs](https://developer.android.com/guide/topics/ui/dialogs.html), and [Status Notifications](https://developer.android.com/guide/topics/ui/notifiers/notifications.html).

## **Mobile application testing**

Mobile applications are first tested within the development environment using emulators and later subjected to [field testing](http://en.wikipedia.org/wiki/Field_testing). [Emulators](http://en.wikipedia.org/wiki/Emulator) provide an inexpensive way to test applications on mobile phones to which developers may not have physical access. The following are examples of tools used for testing application across the most popular [mobile operating systems](http://en.wikipedia.org/wiki/Mobile_operating_system).

* Google Android Emulator

Google Android Emulator is an [Android](http://en.wikipedia.org/wiki/Android_(operating_system)) emulator that is patched to run on a Windows PC as a standalone app, without having to download and install the complete and complex [Android SDK](http://en.wikipedia.org/wiki/Android_SDK). It can be installed and Android compatible apps can be tested on it.

* The official Android SDK Emulator

The official Android SDK Emulator includes a mobile device emulator which mimics all of the hardware and [software](http://en.wikipedia.org/wiki/Software) features of a typical mobile device (without the calls).

* MobiOne

MobiOne Developer is a [mobile Web](http://en.wikipedia.org/wiki/Mobile_Web) [IDE](http://en.wikipedia.org/wiki/Integrated_development_environment) for [Windows](http://en.wikipedia.org/wiki/Windows) that helps developers to code, test, debug, package and deploy mobile [Web applications](http://en.wikipedia.org/wiki/Web_application) to devices such as[iPhone](http://en.wikipedia.org/wiki/IPhone), [BlackBerry](http://en.wikipedia.org/wiki/BlackBerry), [Android](http://en.wikipedia.org/wiki/Android_(operating_system)), and the [Palm Pre](http://en.wikipedia.org/wiki/Palm_Pre).

* TestiPhone

TestiPhone is a [web browser](http://en.wikipedia.org/wiki/Web_browser)-based [simulator](http://en.wikipedia.org/wiki/Simulator) for quickly testing [iPhone](http://en.wikipedia.org/wiki/IPhone) [web applications](http://en.wikipedia.org/wiki/Web_applications). This tool has been tested and works using [Internet Explorer 7](http://en.wikipedia.org/wiki/Internet_Explorer_7), [Firefox 2](http://en.wikipedia.org/wiki/Firefox_2)and [Safari 3](http://en.wikipedia.org/wiki/Safari_3).

* iPhoney

iPhoney gives a [pixel](http://en.wikipedia.org/wiki/Pixel)-accurate web browsing environment and it is powered by [Safari](http://en.wikipedia.org/wiki/Safari_(web_browser)). It can be used while developing [web sites](http://en.wikipedia.org/wiki/Web_sites) for the [iPhone](http://en.wikipedia.org/wiki/IPhone). It is not an iPhone simulator but instead is designed for web developers who want to create 320 by 480 (or 480 by 320) websites for use with iPhone.iPhoney will only run on [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X) 10.4.7 or later.

* BlackBerry Simulator

There are a variety of official BlackBerry simulators available to emulate the functionality of actual BlackBerry products and test how the BlackBerry device software, screen, keyboard and [trackwheel](http://en.wikipedia.org/wiki/Trackwheel" \o "Trackwheel) will work with application.

* Windows UI Automation

To test applications that use the Microsoft UI Automation technology, it requires Windows Automation API 3.0. It is pre-installed on Windows 7, Windows Server 2008 R2 and later versions of Windows. On other operating systems, you can install using Windows Update or download it from the [Microsoft](http://en.wikipedia.org/wiki/Microsoft) Web site.

### Tools**[[edit](http://en.wikipedia.org/w/index.php?title=Mobile_application_development&action=edit&section=3" \o "Edit section: Tools)]**

* [eggPlant](http://en.wikipedia.org/wiki/Eggplant_(GUI_testing_tool)): A GUI-based automated test tool for mobile application across all operating systems and devices.
* [Ranorex](http://en.wikipedia.org/wiki/Ranorex): Test automation tools for mobile, web and desktop apps.
* [Testdroid](http://en.wikipedia.org/wiki/Testdroid): Real mobile devices and test automation tools for testing mobile and web apps.

## **Application stores[[edit](http://en.wikipedia.org/w/index.php?title=Mobile_application_development&action=edit&section=4" \o "Edit section: Application stores)]**

[Several initiatives](http://en.wikipedia.org/wiki/List_of_digital_distribution_platforms_for_mobile_devices) exist both from mobile vendor and mobile operators around the world. Application developers can propose and publish their applications on the[stores](http://en.wikipedia.org/wiki/List_of_digital_distribution_platforms_for_mobile_devices), being rewarded by a revenue sharing of the selling price. An example is Apple's [App Store](http://en.wikipedia.org/wiki/App_Store_(iOS)), where only approved applications may be distributed and run on iOS devices (otherwise known as a [walled garden](http://en.wikipedia.org/wiki/Walled_garden_(technology))). There are approximately 700,000 [iOS](http://en.wikipedia.org/wiki/IOS) Applications. Google's [Android Market](http://en.wikipedia.org/wiki/Android_Market) (now known as the "[Play Store](http://en.wikipedia.org/wiki/Play_Store)") has a large number of apps running on devices with Android OS. [HP](http://en.wikipedia.org/wiki/HP) / [Palm, Inc](http://en.wikipedia.org/wiki/Palm,_Inc) have also created the [Palm App Catalog](http://en.wikipedia.org/wiki/Palm_App_Catalog) where [HP](http://en.wikipedia.org/wiki/HP) / [Palm, Inc](http://en.wikipedia.org/wiki/Palm,_Inc) [webOS](http://en.wikipedia.org/wiki/WebOS" \o "WebOS) device users can download applications directly from the device or send a link to the application via a web distribution method. Mobile operators such as [Telefonica Group](http://en.wikipedia.org/wiki/Telefonica_Group) and[Telecom Italia](http://en.wikipedia.org/wiki/Telecom_Italia) have launched [cross-platform](http://en.wikipedia.org/wiki/Cross-platform) application stores for their subscribers. Additionally, mobile phone manufacturers such as [Nokia](http://en.wikipedia.org/wiki/Nokia) has launched [Ovi](http://en.wikipedia.org/wiki/Ovi_(Nokia)" \o "Ovi (Nokia)) app store for Nokia smartphones. Some independent companies, namely Amazon Appstore, [Aptoide](http://en.wikipedia.org/wiki/Aptoide" \o "Aptoide) and GetJar, have created their own third-party platforms to reach more users in different locations. The [Windows Phone Marketplace](http://en.wikipedia.org/wiki/Windows_Phone_Marketplace) had more than 100,000+ apps available as of 7-11-2012 .