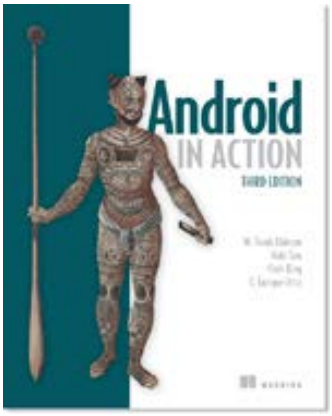
MANNING PUBLICATIONS



Android in Action, Third Edition
By W. Frank Ableson, Robi Sen, and Chris King

Android has become a market-moving technology platform—not just because of the functionality available in the platform but because of how the platform has come to market. In this article based on chapter 5 of Android in Action, Third Edition, the authors introduce the platform and provide context to help you understand Android and where it fits in the global cell phone scene.

Android—the Big Picture

Android has become a market-moving technology platform—not just because of the functionality available in the platform but because of how the platform has come to market.

You've heard about Android. You've read about Android. Now it's time to begin unlocking Android. Android is a software platform that's revolutionizing the global cell phone market. It's the first open-source mobile application platform that's moved the needle in major mobile markets around the globe. When you're examining Android, there are a number of technical and market-related dimensions to consider. In this green paper, we introduce the platform and provide context to help you understand Android and where it fits in the global cell phone scene. Android has eclipsed the cell phone market, and with the release of Android 3.X, and now with the recent release of Android 4 (also known as Ice Cream Sandwich), it has begun making inroads into the tablet market as well.

Android is primarily a Google effort, in collaboration with the Open Handset Alliance. Open Handset Alliance is an alliance of dozens of organizations committed to bringing a "better" and more "open" mobile phone to market. Considered a novelty at first by some, Android has grown to become a market-changing player in a few short years, earning both respect and derision from peers in the industry.

After reading this green paper, you'll understand how Android is constructed, how it compares with other offerings in the market, and what its foundational technologies are, plus you'll get a preview of Android application architecture. More specifically, this green paper takes a look at the Android platform and its relationship to the popular Linux operating system, the Java programming language, and the runtime environment known as the Dalvik virtual machine (VM).

The Android platform

Android is a software environment built for mobile devices. It's not a hardware platform. Android includes a Linux kernel-based OS, a rich UI, end-user applications, code libraries, application frameworks, multimedia support, and much more. And, yes, even telephone functionality is included! Whereas components of the underlying OS are written in C or C++, user applications are built for Android in Java. Even the built-in applications are written in Java.

One feature of the Android platform is that there's no difference between the built-in applications and applications that you create with the SDK. This means that you can write powerful applications to tap into the resources available on the device. Figure 1 shows the relationship between Android and the hardware it runs on.

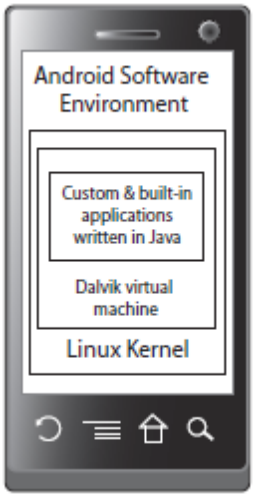


Figure 1 Android is software only. By leveraging its Linux kernel to interface with the hardware, Android runs on many different devices from multiple cell phone manufacturers. Developers write applications in Java.

The most notable feature of Android might be that it's open source; missing elements can and will be provided by the global developer community. Android's Linux kernel-based OS doesn't come with a sophisticated shell environment, but because the platform is open, you can write and install shells on a device. Likewise, multimedia codecs can be supplied by third-party developers and don't need to rely on Google or anyone else to provide new functionality. That's the power of an open-source platform brought to the mobile market.

> **PLATFORM VS DEVICE** With all of that as a backdrop, creating a successful mobile platform is clearly a nontrivial task involving numerous players. Android is an ambitious undertaking, even for Google, a company of seemingly boundless resources and moxie—and they're getting the job done. Within a span of three years, Android has seen numerous major software releases, the release of multiple handsets across most major mobile carriers in the global market, and most recently the introduction of Android-powered tablets.

Now that you've got an introduction to what Android is, let's look at the why and where of Android to provide some context and set the perspective for Android's introduction to the marketplace. After that, it's onto exploring the platform itself!

Understanding the Android market

Android promises to have something for everyone. It aims to support a variety of hardware devices, not just high-end ones typically associated with expensive smartphones. Of course, Android users will enjoy improved performance on a more powerful device, considering that it sports a comprehensive set of computing features. But how well can Android scale up and down to a variety of markets and gain market and mind share? How quickly can the smartphone market become the standard? Some folks are still clinging to phone-only devices, even though smartphones are growing rapidly in virtually every demographic. Let's look at Android from the perspective of a few existing players in the marketplace. When you're talking about the cellular market, the place to start is at the top, with the carriers, or as they're sometimes referred to, the *mobile operators*.

Mobile operators

Mobile operators (the cell phone companies such as AT&T and Verizon) are in the business, first and foremost, of selling subscriptions to their services. Shareholders want a return on their investment, and it's hard to imagine an industry where there's a larger investment than in a network that spans such broad geographic territory. To the mobile operator, cell phones are simultaneously a conduit for services, a drug to entice subscribers, and an annoyance to support and lock down.

Some mobile operators are embracing Android as a platform to drive new data services across the excess capacity operators have built into their networks. Data services represent high-premium services and high-margin revenues for the operator. If Android can help drive those revenues for the mobile operator, all the better.

Other mobile operators feel threatened by Google and the potential of "free wireless," driven by advertising revenues and an upheaval of the market. Another challenge for mobile operators is that they want the final say on what services are enabled across their networks. Historically, handset manufacturers complain that their devices are handicapped and don't exercise all the features designed into them because mobile operators lack the capability or willingness to support those features. An encouraging sign is that there are mobile operators involved in the Open Handset Alliance.

Let's move on to a comparison of Android and existing cell phones on the market today.

Android vs. the feature phones

The majority of cell phones on the market continue to be consumer flip phones and *feature phones*—phones that aren't smartphones.[1] These phones are the ones consumers get when they walk into the retailer and ask what can be had for free. These consumers are the "I just want a phone" customers. Their primary interest is a phone for voice communications, an address book, and increasingly, texting. They might even want a camera. Many of these phones have additional capabilities such as mobile web browsing, but because of relatively poor user experience, these features aren't employed heavily. The one exception is text messaging, which is a dominant application no matter the classification of device.

Another increasingly in-demand category is location-based services, which typically use the *Global Positioning System (GPS)*. Android's challenge is to scale down to this market. Some of the bells and whistles in Android can be left out to fit into lower-end hardware. One of the big functionality gaps on these lower-end phones is the web experience the user gets. Part of the problem is screen size, but equally challenging is the browser technology itself, which often struggles to match the rich web experience of desktop computers. Android features the market-leading WebKit browser engine, which brings desktop-compatible browsing to the mobile arena. Figure 2 shows WebKit in action on Android. If a rich web experience can be effectively scaled down to feature phone class hardware, it would go a long way toward penetrating this end of the market.

Figure 2 Android's built-in browser technology is based on WebKit's browser engine.

[1] About 25% of phones sold in the second quarter of 2011 were smartphones: http://www.gartner.com/it/page.jsp?id=1764714.

WEBKIT The WebKit (www.webkit.org) browser engine is an open source project that powers the browser found in Macs (Safari) and is the engine behind Mobile Safari, which is the browser on the iPhone. It's not a stretch to say that the browser experience is one of a few features that made the iPhone popular out of the gate, so its inclusion in Android is a strong plus for Android's architecture.

Software at the lower end of the market generally falls into one of two camps:

- *Qualcomm's BREW environment*—BREW stands for Binary Runtime Environment for Wireless. For a high-volume example of BREW technology, consider Verizon's Get It Now-capable devices, which run on this platform. The challenge for software developers who want to gain access to this market is that the bar to get an application on this platform is high, because everything is managed by the mobile operator, with expensive testing and revenue-sharing fee structures. The upside to this platform is that the mobile operator collects the money and disburses it to the developer after the sale, and often these sales recur monthly. Just about everything else is a challenge to the software developer. Android's open application environment is more accessible than BREW.
- *Java ME*, or *Java Platform, Micro Edition*—A popular platform for this class of device. The barrier to entry is much lower for software developers. Java ME developers will find a same-but-different environment in Android. Android isn't strictly a Java ME-compatible platform, but the Java programming environment found in Android is a plus for Java ME developers. There are some projects underway to create a bridge environment, with the aim of enabling Java ME applications to be compiled and run for Android. Gaming, a better browser, and anything to do with texting or social applications present fertile territory for Android at this end of the market.

Although the majority of cell phones sold worldwide are not considered smartphones, the popularity of Android (and other capable platforms) has increased demand for higher-function devices. That's what we're going to discuss next.

Android vs. the smartphones

Let's start by naming the major smartphone players: Symbian (big outside North America), BlackBerry from Research in Motion, iPhone from Apple, Windows (Mobile, SmartPhone, and now Phone 7), and of course, the increasingly popular Android platform.

One of the major concerns of the smartphone market is whether a platform can synchronize data and access Enterprise Information Systems for corporate users.

Device-management tools are also an important factor in the enterprise market. The browser experience is better than with the lower-end phones, mainly because of larger displays and more intuitive input methods, such as a touchscreen, touch pad, slideout keyboard, or jog dial.

Android's opportunity in this market is to provide a device and software that people want. For all the applications available for the iPhone, working with Apple can be a challenge; if the core device doesn't suit your needs, there's little room to maneuver because of the limited models available and historical carrier exclusivity. Now that email, calendaring, and contacts can sync with Microsoft Exchange, the corporate environment is more accessible, but Android will continue to fight the battle of scaling the Enterprise walls. Later Android releases have added improved support for the Microsoft Exchange platform, though third-party solutions still outperform the built-in offerings. BlackBerry is dominant because of its intuitive email capabilities, and the Microsoft platforms are compelling because of tight integration to the desktop experience and overall familiarity for Windows users. iPhone has surprisingly good integration with Microsoft Exchange—for Android to compete in this arena, it must maintain parity with iPhone on Enterprise support.

You've seen how Android stacks up next to feature phones and smartphones. Next, we'll see whether Android, the open source mobile platform, can succeed as an open source project.

Android vs. itself

Android will likely always be an open source project, but to succeed in the mobile market, it must sell millions of units and stay fresh. Even though Google briefly entered the device fray with its Nexus One and Nexus S phones, it's not a hardware company. Historically, Android-powered devices have been brought to market by others such as

HTC, Samsung, and Motorola, to name the larger players. Starting in mid-2011, Google began to further flex its muscles with the acquisition of Motorola's mobile business division. Speculation has it that Google's primary interest is in Motorola's patent portfolio, because the intellectual property scene has heated up considerably. A secondary reason may be to acquire the Motorola Xoom platform as Android continues to reach beyond cell phones into tablets and beyond.

When a manufacturer creates an Android-powered device, they start with the Android Open Source Platform (AOSP) and then extend it to meet their need to differentiate their offerings. Android isn't the first open-source phone, but it's the first from a player with the market-moving weight of Google leading the charge. This market leadership position has translated to impressive unit sales across multiple manufacturers and markets around the globe. With a multitude of devices on the market, can Android keep the long-anticipated fragmentation from eroding consumer and investor confidence?

Open source is a double-edged sword. On one hand, the power of many talented people and companies working around the globe and around the clock to deliver desirable features is a force to be reckoned with, particularly in comparison with a traditional, commercial approach to software development. Depending on your perspective, the variety of Android offerings is a welcome alternative to a more monolithic iPhone device platform where consumers have few choices available.

Another challenge for Android is that the licensing model of open source code used in commercial offerings can be sticky. Some software licenses are more restrictive than others, and some of those restrictions pose a challenge to the open source label. At the same time, Android licensees need to protect their investment, so licensing is an important topic for the commercialization of Android.

Licensing Android

Android is released under two different open source licenses. The Linux kernel is released under the *GNU General Public License* (*GPL*) as is required for anyone licensing the open source OS kernel. The Android platform, excluding the kernel, is licensed under the *Apache Software License* (*ASL*). Although both licensing models are open source–oriented, the major difference is that the Apache license is considered friendlier toward commercial use. Some open-source purists might find fault with anything but complete openness, source-code sharing, and noncommercialization; the ASL attempts to balance the goals of open source with commercial market forces. So far there has been only one notable licensing hiccup impacting the Android mod community, and that had more to do with the gray area of full system images than with a manufacturer's use of Android on a mainstream product release. Currently, Android is facing intellectual property challenges; both Microsoft and Apple are bringing litigation against Motorola and HTC for the manufacturer's Android-based handsets.

Any technical discussion of a software environment must include a review of the layers that compose the environment, sometimes referred to as a *stack* because of the layer-upon-layer construction. Next up is a high-level breakdown of the components of the Android stack.

Selling applications

A mobile platform is ultimately valuable only if there are applications to use and enjoy on that platform. To that end, the topic of buying and selling applications for Android is important and gives us an opportunity to highlight a key difference between Android and the iPhone. The Apple App Store contains software titles for the iPhone—lots of them. But Apple's somewhat draconian grip on the iPhone software market requires that all applications be sold through its venue. Although Apple's digital rights management (DRM) is the envy of the market, this approach can pose a challenging environment for software developers who might prefer to make their application available through multiple distribution channels.

Contrast Apple's approach to application distribution with the freedom Android developers enjoy to ship applications via traditional venues such as freeware and shareware and commercially through various marketplaces, including their own website!

For software publishers who want the focus of an on-device shopping experience, Google has launched and continues to mature the Android Market. For software developers who already have titles for other platforms such as Windows Mobile, Palm, and BlackBerry, traditional software markets such as Handango (www.Handango.com)

also support selling Android applications. Handango and its ilk are important outlets; consumers new to Android will likely visit sites such as Handango because that might be where they first purchased one of their favorite applications for their prior device.

The layers of Android

The Android stack includes an impressive array of features for mobile applications. In fact, looking at the architecture alone, without the context of Android as a platform designed for mobile environments, it would be easy to confuse Android with a general computing environment. All the major components of a computing platform are there. Here's a quick rundown of prominent components of the Android stack:

- A *Linux kernel* that provides a foundational hardware abstraction layer as well as core services such as process, memory, and filesystem management. The kernel is where hardware-specific drivers are implemented—capabilities such as Wi-Fi and Bluetooth are here. The Android stack is designed to be flexible, with many optional components that largely rely on the availability of specific hardware on a given device. These components include features such as touch screens, cameras, GPS receivers, and accelerometers.
- *Prominent code libraries*, including the following:
  - Browser technology from WebKit, the same open source engine powering Mac's Safari and the iPhone's Mobile Safari browser. WebKit has become the de facto standard for most mobile platforms.
  - Database support via SQLite, an easy-to-use SQL database.
  - Advanced graphics support, including 2D, 3D, animation from Scalable Games Language (SGL), and OpenGL ES.
  - Audio and video media support from PacketVideo's OpenCORE, and Google's own Stagefright media framework.
  - Secure Sockets Layer (SSL) capabilities from the Apache project.
- *An array of managers* that provide services for
  - Activities and views
  - Windows
  - Location-based services
  - Telephony
  - Resources
- The Android runtime, which provides
  - Core Java packages for a nearly full-featured Java programming environment. Note that this isn't a Java ME environment.
  - The Dalvik VM, which employs services of the Linux-based kernel to provide an environment to host Android applications.

Both core applications and third-party applications (such as the ones you'll build in this book) run in the Dalvik VM, atop the components we just listed. You can see the relationship among these layers in figure 3.

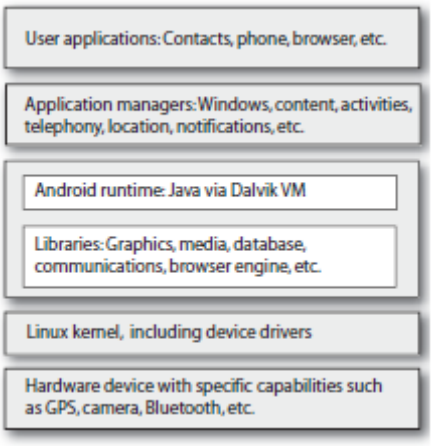


Figure 3 The Android stack offers an impressive array of technologies and capabilities.

> **TIP** Without question, Android development requires Java programming skills. Be sure to brush up on your Java programming knowledge. There are many Java references on the Internet, and no shortage of Java books on the market. An excellent source of Java titles can be found at www.manning.com/catalog/java.

Now that we've shown you the obligatory stack diagram and introduced all the layers, let's look more in depth at the runtime technology that underpins Android.

Building on the Linux kernel

Android is built on a Linux kernel and on an advanced, optimized VM for its Java applications. Both technologies are crucial to Android. The Linux kernel component of the Android stack promises agility and portability to take advantage of numerous hardware options for future Android-equipped phones. Android's Java environment is key: it makes Android accessible to programmers because of both the number of Java software developers and the rich environment that Java programming has to offer.

Why use Linux for a phone? Using a full-featured platform such as the Linux kernel provides tremendous power and capabilities for Android. Using an open source foundation unleashes the capabilities of talented individuals and companies to move the platform forward. Such an arrangement is particularly important in the world of mobile devices, where products change so rapidly. The rate of change in the mobile market makes the general computer market look slow and plodding. And, of course, the Linux kernel is a proven core platform. Reliability is more important than performance when it comes to a mobile phone, because voice communication is the primary use of a phone. All mobile phone users, whether buying for personal use or for a business, demand voice reliability, but they still want cool data features and will purchase a device based on those features. Linux can help meet this requirement.

Speaking to the rapid rate of phone turnover and accessories hitting the market, another advantage of using Linux as the foundation of the Android platform stack is that it provides a hardware abstraction layer; the upper levels remain unchanged despite changes in the underlying hardware. Of course, good coding practices demand that user applications fail gracefully in the event a resource isn't available, such as a camera not being present in a particular handset model. As new accessories appear on the market, drivers can be written at the Linux level to provide support, just as on other Linux platforms. This architecture is already demonstrating its value; Android devices are already available on distinct hardware platforms. HTC, Motorola, and others have released Android-based devices built on their respective hardware platforms. User applications, as well as core Android applications, are written in Java and are compiled into *byte codes*. Byte codes are interpreted at runtime by an interpreter known as a *virtual machine* (*VM*).

Running in the Dalvik VM

The Dalvik VM is an example of the need for efficiency, the desire for a rich programming environment, and even some intellectual property constraints, colliding, with innovation as the result. Android's Java environment provides a rich application platform and is accessible because of the popularity of Java itself. Also, application performance, particularly in a low-memory setting such as you find in a mobile phone, is paramount for the mobile market. But this isn't the only issue at hand.

Android isn't a Java ME platform. Without commenting on whether this is ultimately good or bad for Android, there are other forces at play here. There's the matter of Java VM licensing from Oracle. From a high level, Android's code environment is Java.

Applications are written in Java, which is compiled to Java byte codes and subsequently translated to a similar but different representation called *dex files*. These files are logically equivalent to Java byte codes, but they permit Android to run its applications in its own VM that's both (arguably) free from Oracle's licensing clutches and an open platform upon which Google, and potentially the open source community, can improve as necessary. Android is facing litigation challenges from Oracle about the use of Java.

> **NOTE** From the mobile application developer's perspective, Android is a Java environment, but the runtime isn't strictly a Java VM. This accounts for the incompatibilities between Android and proper Java environments and libraries.

If you have a code library that you want to reuse, your best bet is to assume that your code is nearly *source compatible*, attempt to compile it into an Android project, and then determine how close you are to having usable code.

The important things to know about the Dalvik VM are that Android applications run inside it and that it relies on the Linux kernel for services such as process, memory, and filesystem management.

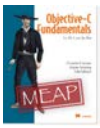
Summary

This green paper introduced the Android platform and briefly touched on market positioning, including what Android is up against in the rapidly changing and highly competitive mobile marketplace. In a few years, the Android SDK has been announced, released, and updated numerous times. And that's just the software. Major device manufacturers have now signed on to the Android platform and have brought capable devices to market, including a privately labeled device from Google itself. Add to that the patent wars unfolding between the major mobile players, and the stakes continue to rise—and Android's future continues to brighten.

We examined the Android stack and discussed its relationship with Linux and Java. With Linux at its core, Android is a formidable platform, especially for the mobile space where it's initially targeted. Although Android development is done in the Java programming language, the runtime is executed in the Dalvik VM, as an alternative to the Java VM from Oracle. Regardless of the VM, Java coding skills are an important aspect of Android development.

Here are some other Manning titles you might be interested in:

Android in Practice
Charlie Collins, Michael D. Galpin, and Matthias Kaeppler



Objective-C Fundamentals
For iOS4 and iPad
Christopher K. Fairbairn, Johannes Fahrenkrug, and Collin Ruffenach

Windows Phone 7 in Action
Massimo Perga, Timothy Binkley-Jones and Michael Sync

Last updated: December 1, 2011