1

Hello, Android

WHAT'S IN THIS CHAPTER?

- A background of mobile application development
- What is Android?
- Which devices Android runs on
- Why you should develop for mobile and Android
- An introduction to the Android SDK and development framework

ANDROID APPLICATION DEVELOPMENT

Whether you're an experienced mobile engineer, a desktop or web developer, or a complete programming novice, Android represents an exciting opportunity to write applications for an audience of over two billion Android device users.

You're probably already familiar with Android, the most common software powering mobile phones. If not, and you purchased this book in the hope that Android development would help you create an unstoppable army of emotionless robot warriors on a relentless quest to cleanse the earth of the scourge of humanity, you should reconsider this book purchase (and your life choices.)

When announcing Android at its launch in 2007, Andy Rubin described it as follows:

> *The first truly open and comprehensive platform for mobile devices. It includes an operating system, user-interface and applications—all of the*

> *software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation.*
>
> —Where's My Gphone?
> *(http://googleblog.blogspot.com/2007/11/wheres-my-gphone.html)*

Since then, Android has expanded beyond mobile phones to provide a development platform for an increasingly wide range of hardware, including tablets, televisions, watches, cars, and Internet-of-Things (IoT) devices.

Android is an open source software stack that includes an operating system, middleware, and key applications for mobile and embedded devices.

Critically, for us as developers, it also includes a rich set of API libraries that make it possible to write applications that can shape the look, feel, and function of the Android devices on which they run.

In Android, system, bundled, and all third-party applications are written with the same APIs and executed on the same run time. These APIs feature hardware access, video recording, location-based services, support for background services, maps, notifications, sensors, relational databases, inter-application communication, Bluetooth, NFC, and 2D and 3D graphics.

This book describes how to use these APIs to create your own Android applications. In this chapter you learn some guidelines for mobile and embedded hardware development, and are introduced to some of the platform features available to Android developers.

Android has powerful APIs, a huge and diverse ecosystem of users, excellent documentation, a thriving developer community, and has no required costs for development or distribution. As the Android device ecosystem continues to grow, you have the opportunity to create innovative applications for users, no matter what your development experience.

A LITTLE BACKGROUND

In the days before Instagram, Snapchat, and Pokémon Go, when Google was still a twinkle in its founders' eyes and dinosaurs roamed the earth, mobile phones were just that—portable phones small enough to fit inside a briefcase, featuring batteries that could last up to several hours. They did, however, offer the freedom to make calls without being physically connected to a landline.

In the 10 years since the first Android device was launched, smart phones have become ubiquitous and indispensable. Hardware advancements have made devices more powerful, featuring bigger, brighter screens and featuring advanced hardware including accelerometers, fingerprint scanners, and ultra-high-resolution cameras.

These same advances have more recently resulted in a proliferation of additional form factors for Android devices, including a large variety of smart-phones, tablets, watches, and televisions.

These hardware innovations offer fertile ground for software development, providing many opportunities to create innovative new applications.

The Not-So-Distant Past

In the early days of native phone application development, developers, generally coding in low-level C or C++, needed to understand the specific hardware they were coding for, typically a single device or possibly a range of devices from a single manufacturer. The complexity inherent in this approach meant the applications written for these devices often lagged behind their hardware counterparts. As hardware technology and mobile Internet access have advanced, this closed approach has become outmoded.

The next significant advancement in mobile phone application development was the introduction of Java-hosted MIDlets. MIDlets were executed on a Java virtual machine (JVM), a process that abstracted the underlying hardware and let developers create applications that ran on many devices that supported the Java run time.

Unfortunately, this convenience came at the price of more heavily restricted access to the device hardware. Similarly, it was considered normal for third-party applications to receive different hardware access and execution rights from those given to native applications written by the phone manufacturers, with MIDlets often receiving few of either.

The introduction of Java MIDlets expanded developers' audiences, but the lack of low-level hardware access and sandboxed execution meant that most mobile applications were regular desktop programs or websites designed to render on a smaller screen, and didn't take advantage of the inherent mobility of the handheld platform.

Living in the Future

At its introduction, Android was part of a new wave of modern mobile operating systems designed specifically to support application development on increasingly powerful mobile hardware.

Android offers an open development platform built on an open source Linux kernel. Hardware access is available to all applications through a series of API libraries, and application interaction, while carefully controlled, is fully supported.

In Android, all applications have equal standing. Third-party and native Android applications are written with the same APIs and are executed on the same run time. Users can replace most system application with a third-party developer's alternative; indeed, even the dialer and home screens can be replaced.

THE ANDROID ECOSYSTEM

The Android ecosystem is made up of a combination of three components:

- A free, open source operating system for embedded devices
- An open source development platform for creating applications
- Devices that run the Android operating system (and the applications created for it)

More specifically, Android is made up of several necessary and dependent parts, including the following:

- A Compatibility Definition Document (CDD) and Compatibility Test Suite (CTS) that describe the capabilities required for a device to support the Android software stack.
- A Linux operating system kernel that provides a low-level interface with the hardware, memory management, and process control, all optimized for mobile and embedded devices.
- Open source libraries for application development, including SQLite, WebKit, OpenGL, and a media manager.
- A run time used to execute and host Android applications, including the Android Run Time (ART) and the core libraries that provide Android-specific functionality. The run time is designed to be small and efficient for use on embedded devices.
- An application framework that agnostically exposes system services to the application layer, including the Window Manager and Location Manager, databases, telephony, and sensors.
- A user interface framework used to host and launch applications.
- A set of core preinstalled applications.
- A software development kit (SDK) used to create applications, including the related tools, IDE, sample code, and documentation.

What really makes Android compelling is its open philosophy, which ensures that you can fix any deficiencies in user interface or native application design by writing an extension or replacement. Android provides you, as a developer, with the opportunity to create applications designed to look, feel, and function exactly as you imagine them.

With more than 2 billion monthly active users of devices running the Android operating system, installing over 82 billion apps and games in from Google Play in 2016 alone, the Android ecosystem represents an unparalleled chance to create apps that can affect and improve billions of people's lives.

PRE-INSTALLED ANDROID APPLICATIONS

Android devices typically come with a suite of preinstalled applications that users expect. On smart phones these typically include:

- A phone dialer
- An SMS management application
- A web browser
- An e-mail client
- A calendar
- A contacts list

- A music player and picture gallery
- A camera and video recording application
- A calculator
- A home screen
- An alarm clock

In many cases Android devices also ship with the following proprietary Google mobile applications:

- The Google Play Store for downloading third-party Android applications
- The Google Maps application, including StreetView, driving directions, and turn-by-turn navigation, satellite views, and traffic conditions
- The Gmail email client
- The YouTube video player
- The Google Chrome browser
- The Google home screen and Google Assistant

The data stored and used by many of these native applications—such as contact details—are also available to third-party applications.

The exact makeup of the applications available on new Android devices is likely to vary based on the hardware manufacturer, the carrier or distributor, and the type of device.

The open source nature of Android means that carriers and OEMs can customize the user interface and the applications bundled with each Android device.

It's important to note that for compatible devices, the underlying platform and SDK remains consistent across OEM and carrier variations. The look and feel of the user interface may vary, but your applications will function in the same way across all compatible Android devices.

ANDROID SDK FEATURES

For us developers, the true appeal of Android lies in its APIs.

As an application-neutral platform, Android gives you the opportunity to create applications that are as much a part of the phone as anything provided out-of-the-box. The following list highlights some of the most noteworthy Android features:

- Transparent access to telephony and Internet resources through GSM, EDGE, 3G, 4G, LTE, and Wi-Fi network support, enabling your app to send and retrieve data across mobile and Wi-Fi networks
- Comprehensive APIs for location-based services such as GPS and network-based location detection
- Full support for integrating maps within the user interface

- Full multimedia hardware control, including playback and recording with the camera and microphone
- Media libraries for playing and recording a variety of audio/video or still-image formats
- APIs for using sensor hardware, including accelerometers, compasses, barometers, and fingerprint sensors
- Libraries for using Wi-Fi, Bluetooth, and NFC hardware
- Shared data stores and APIs for contacts, calendar, and multi-media
- Background services and an advanced notification system
- An integrated web browser
- Mobile-optimized, hardware-accelerated graphics, including a path-based 2D graphics library and support for 3D graphics using OpenGL ES 2.0
- Localization through a dynamic resource framework

WHAT DOES ANDROID RUN ON?

The first Android mobile handset, the T-Mobile G1, was released in the United States in October 2008. By the end of 2017 there are more than 2 billion monthly active Android devices globally, making it the most common smart phone operating system in use world-wide.

Rather than being a mobile OS created for a single hardware implementation, Android is designed to support a large variety of hardware platforms, from smart phones to tablets, televisions, watches, and IoT devices.

With no licensing fees or proprietary software, the cost to handset manufacturers for providing Android devices is comparatively low, which, combined with a massive ecosystem of powerful applications, has encouraged device manufacturers to produce increasingly diverse and tailored hardware.

As a result, hundreds of manufacturers, including Samsung, LG, HTC, and Motorola, are creating Android devices. These devices are distributed to users via hundreds of carriers world-wide.

WHY DEVELOP FOR MOBILE?

Smart phones have become so advanced and personal to us that for many people they've become an extension of themselves. Studies have shown that many mobile phone users become anxious if they misplace their device, lose connectivity, or their battery runs out.

The ubiquity of mobile phones, and our attachment to them, makes them a fundamentally different platform for development from PCs. With a microphone, camera, touchscreen, location detection, and environmental sensors, a phone can effectively become an extra-sensory perception device.

Smart phone ownership easily surpasses computer ownership in many countries, with more than 3 billion mobile phone users worldwide. 2009 marked the year that more people accessed the Internet for the first time from a mobile phone rather than a PC.

The increasing popularity of smart phones, combined with the increasing availability of high-speed mobile data and Wi-Fi hotspots, has created a huge opportunity for advanced mobile applications.

Smartphone applications have changed the way people use their phones. This gives you, the application developer, a unique opportunity to create dynamic, compelling new applications that become a vital part of people's lives.

WHY DEVELOP FOR ANDROID?

In addition to providing access to the largest ecosystem of smart phone users, Android represents a dynamic framework for app development based on the reality of modern mobile devices designed by developers, for developers.

With a simple, powerful, and open SDK, no licensing fees, excellent documentation, a diverse range of devices and form-factors, and a thriving developer community, Android represents an opportunity to create software that can change people's lives.

The barrier to entry for new Android developers is minimal:

- No certification is required to become an Android developer.
- The Google Play Store provides free, up-front purchase, in-app billing, and subscription options for distribution and monetization of your applications.
- There is no approval process for application distribution.
- Developers have total control over their brands.

From a commercial perspective, Android represents the most common smart phone operating system, and provides access to over 2 billion monthly active Android devices globally, offering unparalleled reach to make your applications available to users around the world.

INTRODUCING THE DEVELOPMENT FRAMEWORK

Android applications normally are written using the Java or Kotlin programming languages, and are executed by means of the Android Run Time (ART).

> **NOTE** *Historically, Android apps were written primarily using Java language syntax. More recently, Android Studio 3.0 introduced full support for Kotlin as an official first class language for application development. Kotlin is a JVM language, which is interoperable with existing Android languages and the Android Run Time, allowing you to use both Java and Kotlin syntax within the same applications.*

Each Android application runs in a separate process, relinquishing all responsibility for memory and process management to the Android Run Time, which stops and kills processes as necessary to manage resources.

ART sits on top of a Linux kernel that handles low-level hardware interaction, including drivers and memory management, while a set of APIs provides access to all the underlying services, features, and hardware.

What Comes in the Box

The Android SDK includes everything you need to start developing, testing, and debugging Android applications:

- **The Android API Libraries**—The core of the SDK is the Android API libraries that provide developer access to the Android stack. These are the same libraries that Google uses to create native Android applications.
- **Development tools**—The SDK includes the Android Studio IDE and several other development tools that let you compile and debug your applications to turn Android source code into executable applications. You learn more about the developer tools in Chapter 2, "Getting Started."
- **The Android Virtual Device Manager and Emulator**—The Android Emulator is a fully interactive mobile device emulator featuring several alternative skins. The Emulator runs within an Android Virtual Device (AVD) that simulates a device hardware configuration. Using the Emulator you can see how your applications will look and behave on a real Android device. All Android applications run within ART, so the software emulator is an excellent development environment—in fact, because it's hardware-neutral, it provides a better independent test environment than any single hardware implementation.
- **Full documentation**—The SDK includes extensive code-level reference information detailing exactly what's included in each package and class and how to use them. In addition to the code documentation, Android's reference documentation and developer guides explain how to get started, give detailed explanations of the fundamentals behind Android development, highlight best practices, and provide deep-dives into framework topics.
- **Sample code**—The Android SDK includes a selection of sample applications that demonstrate some of the possibilities available with Android, as well as simple programs that highlight how to use individual API features.
- **Online support**—Android has vibrant developer communities on most online social networks, Slack, and many developer forums. Stack Overflow (`www.stackoverflow.com/questions/tagged/android`) is a hugely popular destination for Android questions and a great place to find answers to beginner questions. Many Android engineers from Google are active on Stack Overflow and Twitter.

Understanding the Android Software Stack

The Android software stack is a Linux kernel and a collection of C/C++ libraries exposed through an application framework that provides services for, and management of, the run time and applications, as shown in Figure 1-1.

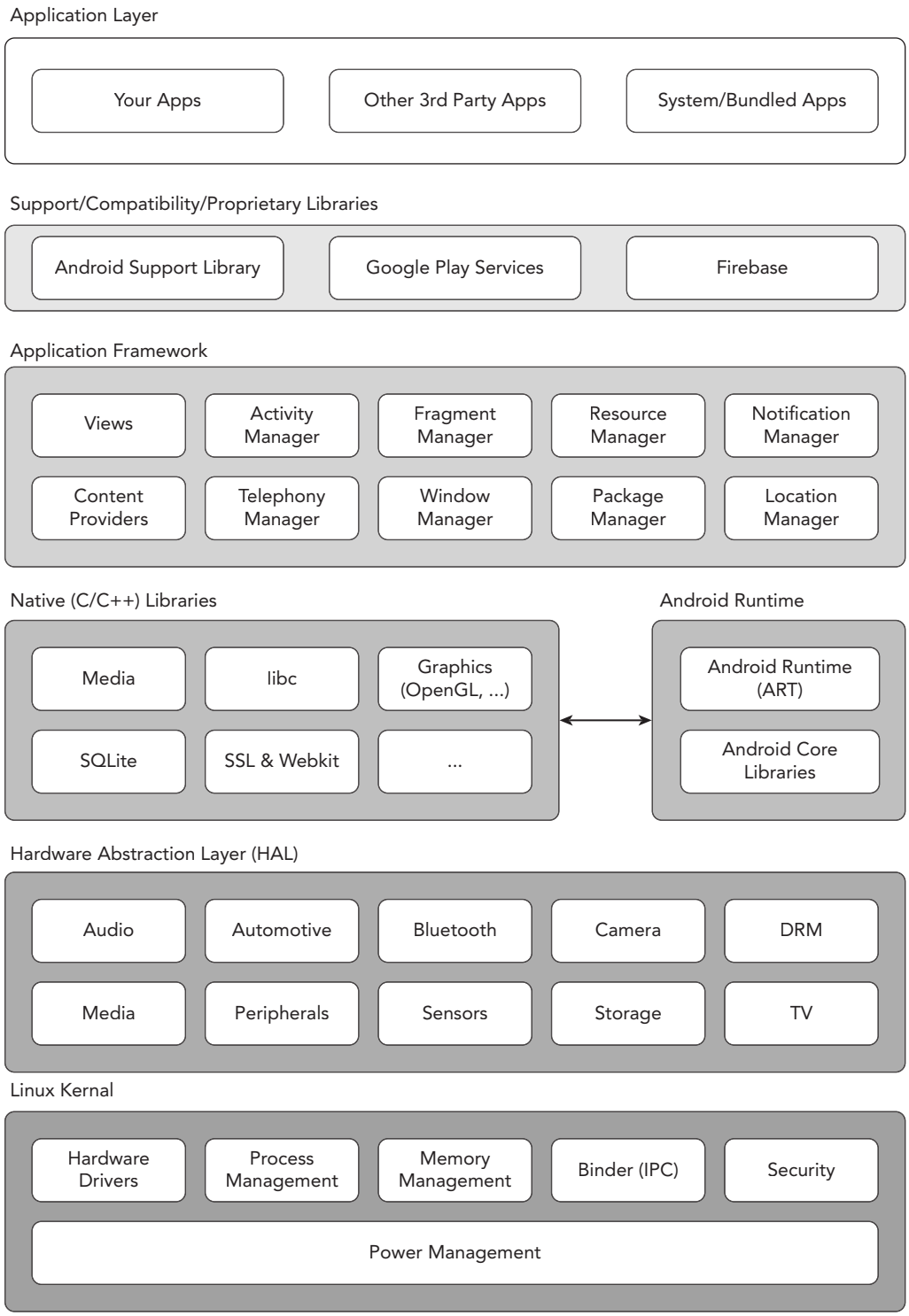
Application Layer
Your Apps
Other 3rd Party Apps
System/Bundled Apps
Support/Compatibility/Proprietary Libraries
Android Support Library
Google Play Services
Firebase
Application Framework
Views
Activity Manager
Fragment Manager
Resource Manager
Notification Manager
Content Providers
Telephony Manager
Window Manager
Package Manager
Location Manager
Native (C/C++) Libraries
Android Runtime
Media
libc
Graphics (OpenGL, ...)
SQLite
SSL & Webkit
...
Android Runtime (ART)
Android Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Automotive
Bluetooth
Camera
DRM
Media
Peripherals
Sensors
Storage
TV
Linux Kernal
Hardware Drivers
Process Management
Memory Management
Binder (IPC)
Security
Power Management

FIGURE 1-1

- **Linux kernel**—Core services (including hardware drivers, process and memory management, security, network, and power management) are handled by a Linux kernel (the specific kernel version depends on the Android platform version and hardware platform).
- **Hardware Application Layer (HAL)**—The HAL provides an abstraction layer between the underlying physical device hardware and the remainder of the stack.
- **Libraries**—Running on top of the kernel and HAL, Android includes various C/C++ core libraries such as libc and SSL, as well as the following:
  - A media library for playback of audio and video media
  - A surface manager to provide display management
  - Graphics libraries that include SGL and OpenGL for 2D and 3D graphics
  - SQLite for native database support
  - SSL and WebKit for integrated web browser and Internet security
- **Android Run Time**—The run time is what makes an Android phone an Android phone rather than a mobile Linux implementation. Including the core libraries, the Android Run Time is the engine that powers your applications and forms the basis for the application framework.
- **Core libraries**—Although most Android application development is written using the Java or Kotlin JVM languages, ART is not a Java VM. The core Android libraries provide most of the functionality available in the core Java libraries, as well as the Android-specific libraries.
- **Application framework**—The application framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.
- **Application layer**—All applications, both native and third-party, are built on the application layer by means of the same API libraries. The application layer runs within the Android Run Time, using the classes and services made available from the application framework.

The Android Run Time

One of the key elements of Android is the Android Run Time (ART). Rather than using a traditional Java VM such as Java ME, Android uses its own custom run time designed to ensure that multiple instances run efficiently on a single device.

ART uses the device's underlying Linux kernel to handle low-level functionality, including security, threading, and process and memory management. It's also possible to write C/C++ applications that run closer to the underlying Linux OS. Although you *can* do this, in most cases there's no reason you should need to.

If the speed and efficiency of C/C++ is required for your application, Android provides a native development kit (NDK). The NDK is designed to enable you to create C++ libraries using the libc and libm libraries, along with native access to OpenGL.

> **NOTE** *This book focuses exclusively on writing applications that run within ART using the SDK; NDK development is not within the scope of this book. If your inclinations run toward NDK development, exploring the Linux kernel and C/C++ underbelly of Android, modifying ART, or otherwise tinkering with things under the hood, check out the Android Open Source Project at* `source.android.com`.

All Android hardware and system service access is managed using ART as a middle tier. By using this run time to host application execution, developers have an abstraction layer that ensures they should never have to worry about a particular hardware implementation.

ART executes Dalvik executable files (`.dex`)—named after an earlier virtual machine implementation named "Dalvik"—a format optimized to ensure minimal memory footprint. You create `.dex` executables by transforming Java or Kotlin language compiled classes using the tools supplied within the SDK.

> **NOTE** *You learn more about how to create Dalvik executables in Chapter 2.*

Android Application Architecture

Android's architecture encourages component reuse, enabling you to publish and share Activities, Services, and data with other applications, with access managed by the security restrictions you define.

The same mechanism that enables you to produce a replacement contact manager or phone dialer can let you expose your application's components in order to let other developers build on them by creating new UI front ends or functionality extensions.

The following application services are the architectural cornerstones of all Android applications, providing the framework you'll be using for your own software:

- **Activity Manager and Fragment Manager**—Activities and Fragments are used to define the user interface of your apps. The Activity and Fragment Managers control the life cycle of your Activities and Fragments, respectively, including management of the Activity stack (described in Chapters 3 and 5).
- **Views**—Used to construct the user interfaces controls within your Activities and Fragments, as described in Chapter 5.
- **Notification Manager**—Provides a consistent and nonintrusive mechanism for signaling your users, as described in Chapter 11.
- **Content Providers**—Lets your applications share data, as described in Chapter 10.
- **Resource Manager**—Enables non-code resources, such as strings and graphics, to be externalized, as shown in Chapter 4.
- **Intents**—Provides a mechanism for transferring data between applications and their components, as described in Chapter 6.

Android Libraries

Android offers a number of APIs for developing your applications. Rather than list them all here, check out the documentation at `developer.android.com/reference/packages.html`, which gives a complete list of packages included in the Android SDK.

Android is intended to target a wide range of mobile hardware, so be aware that the suitability and implementation of some of the advanced or optional APIs may vary depending on the host device.