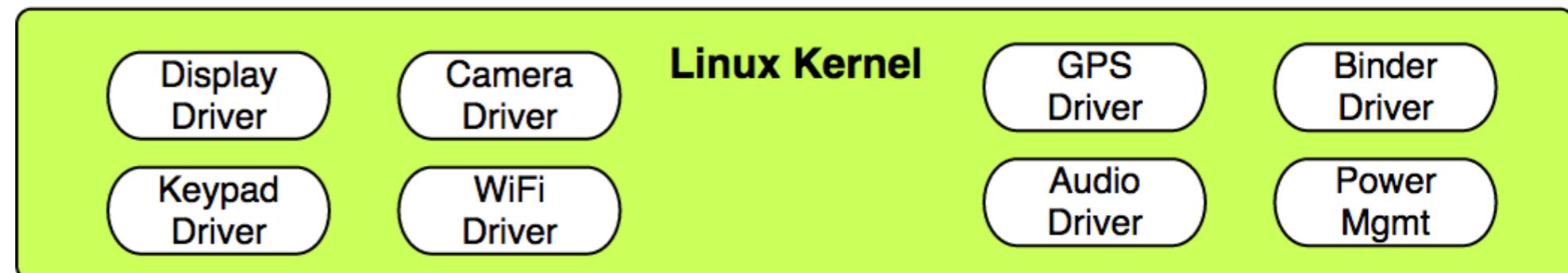
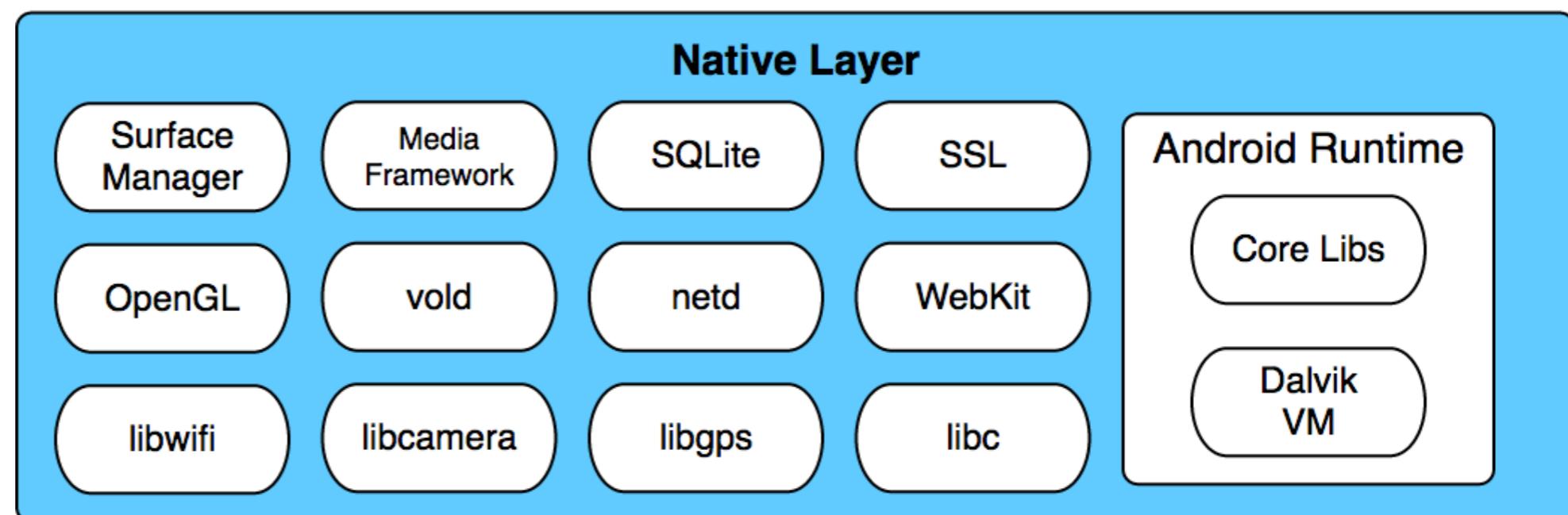
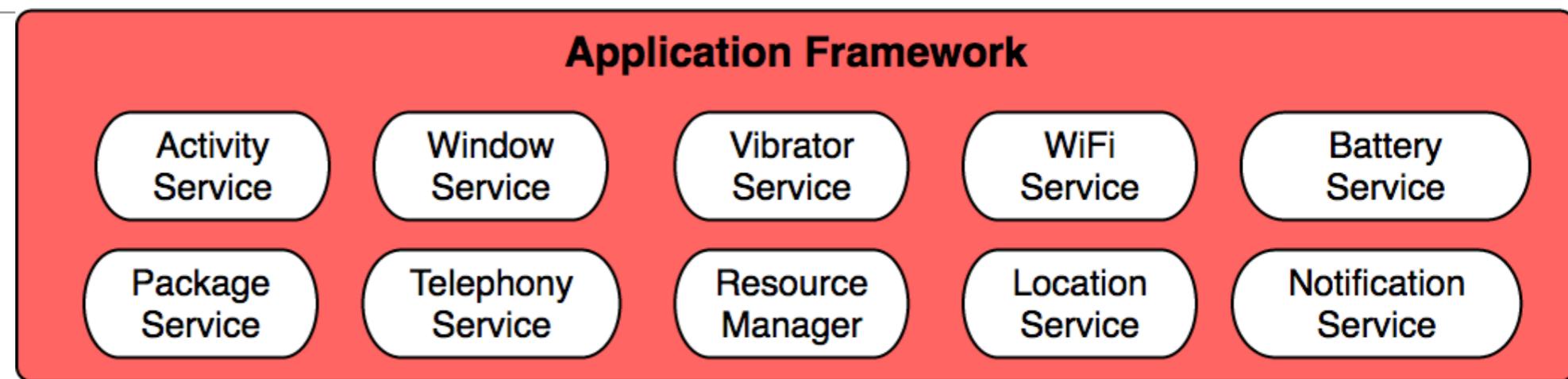


The Android Stack

Overview

- The Android operating system is like a cake consisting of various layers.
- Each layer has its own characteristics and purpose—but the layers are not always cleanly separated and often seep into one another.



- Although Android is based on linux, it is not just another flavour of Linux, in the way that Ubuntu, Fedora, or Red Hat are.
- Many things you'd expect from a typical Linux distribution aren't available in Android, such as the X11 window manager, the ability to add a person as a Linux user or even the glibc standard C library.
- On the other hand, Android adds quite a bit to the Linux kernel, such as
 - an improved power management that is well-suited for mobile battery-powered devices,
 - a very fast interprocess communication mechanisms
 - mechanisms for sand- boxing applications so they are isolated from one another.

Linux Kernel

- **Portable**

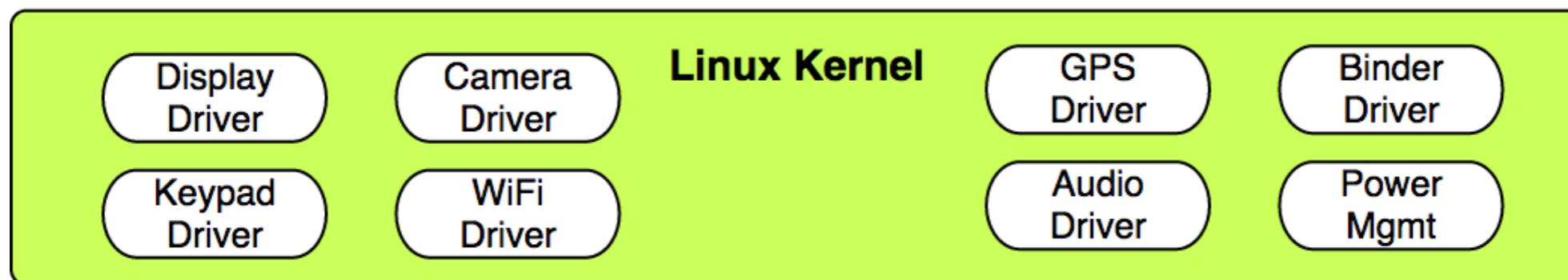
- Most low-level parts of Linux have been written in fairly portable C code, which allows for third parties to port Android to a variety of devices.

- **Secure**

- Linux is a highly secure system, having been tried and tested through some very harsh environments over the decades.
- Android relies heavily on Linux for security, and all Android applications run as separate Linux processes with permissions set by the Linux system, passing many security concerns to the underlying Linux system.
- The kernel is the sole enforcer of Android permissions, providing a simple, powerful, security mechanism. It also allows Android apps access to native code, such as fast C implementations of various libraries via the Java Native Interface.

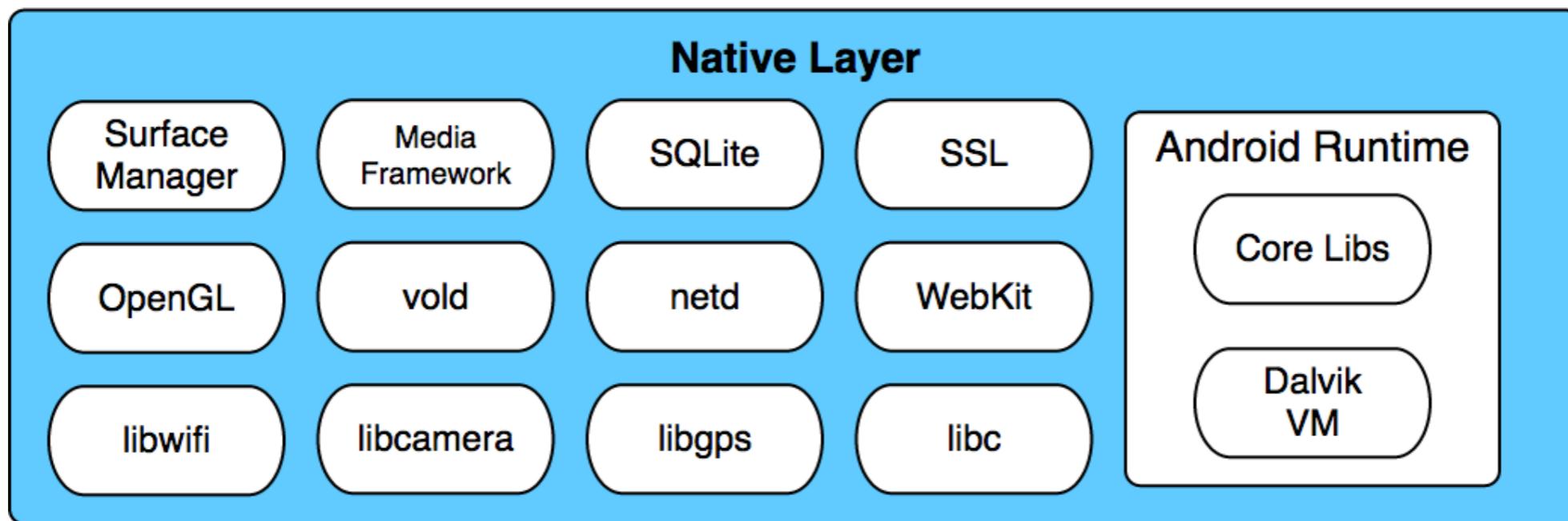
- Features

- The Linux kernel comes with a range of features. Android leverages many of them, e.g. support for memory and power management, networking and radio functionality.



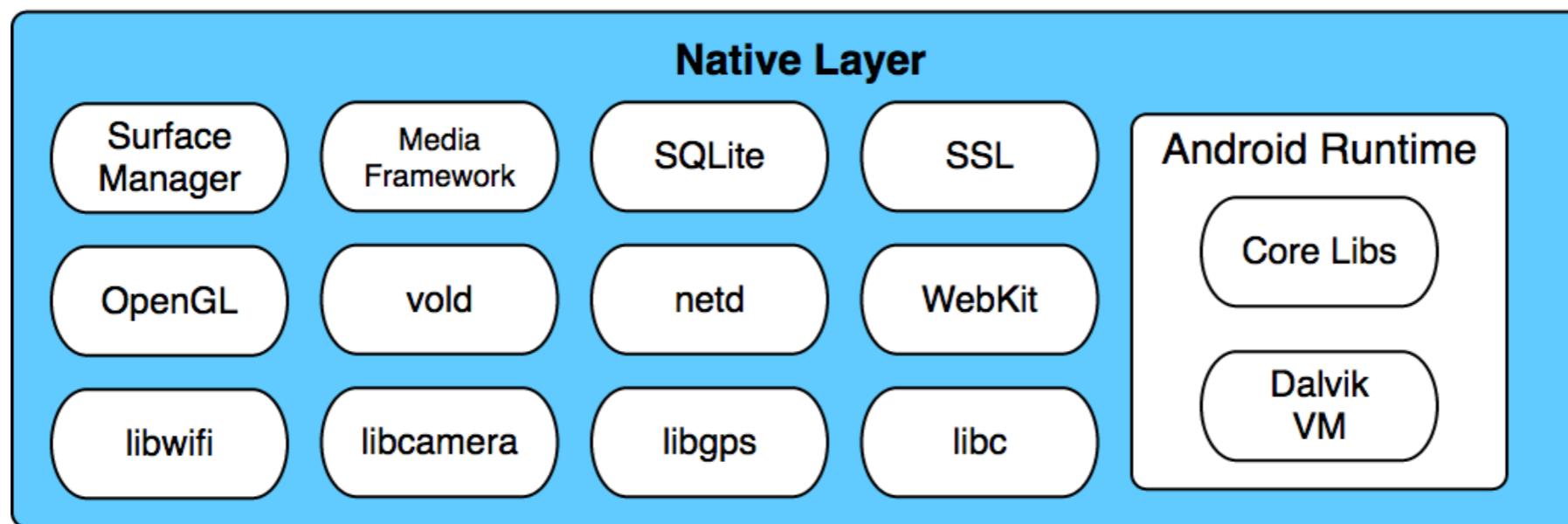
Native Layer

- The native libraries are C/C++ libraries. Their primary job is to support the Android Application Framework layer



- Some of these libraries are purpose-built for the Android OS, whereas others are often taken from the open source community in order to complete the operating system.

- **Binder:** A very fast inter-process communication mechanism that allows for one Android app to talk to another.
- **Framework libraries:** Various libraries designed to support system services, such as location, media, package installer, telephony, WiFi, voip, and so on.
- **Webkit:** A fast web-rendering engine used by Safari, Chrome, and other browsers.
- **SQLite:** A full-featured SQL database that the Android app framework exposes to applications.

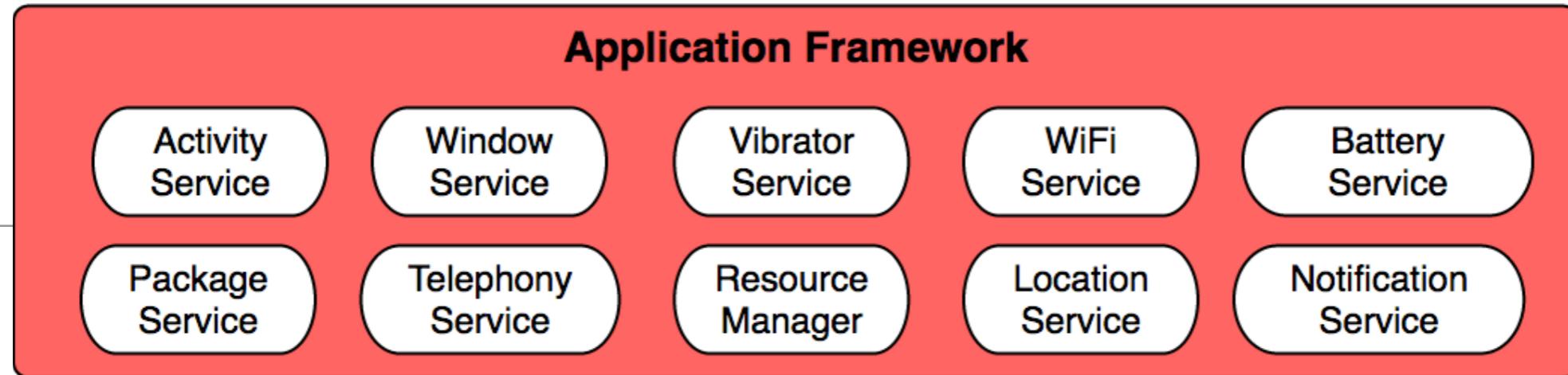


- **Apache Harmony:** An open source implementation of Java libraries.
- **OpenGL:** 3D graphics libraries.
- **OpenSSL:** The secure socket layer, allowing for secure point-to-point connectivity.

Native Daemons

- Native daemons are executable code that usually runs to support some kind of system service. Prominent examples:
 - **Service Manager (`servicemanager`)**: The umbrella process running all other framework services. It is the most critical native daemon.
 - **Radio interface layer daemon (`rild`)**: Responsible for supporting the telephony functionality via GSP or CDMA, usually.
 - **Installation daemon (`installd`)**: Supports management of apps, including installation, upgrades, as well as granting of permissions.
 - **Media server (`mediaserver`)**: Supports camera, audio, and other media services.
 - **Android Debug Bridge (`adb`)**: Supports developer connectivity from your PC to the device (including the emulator) so that you can develop apps for Android.

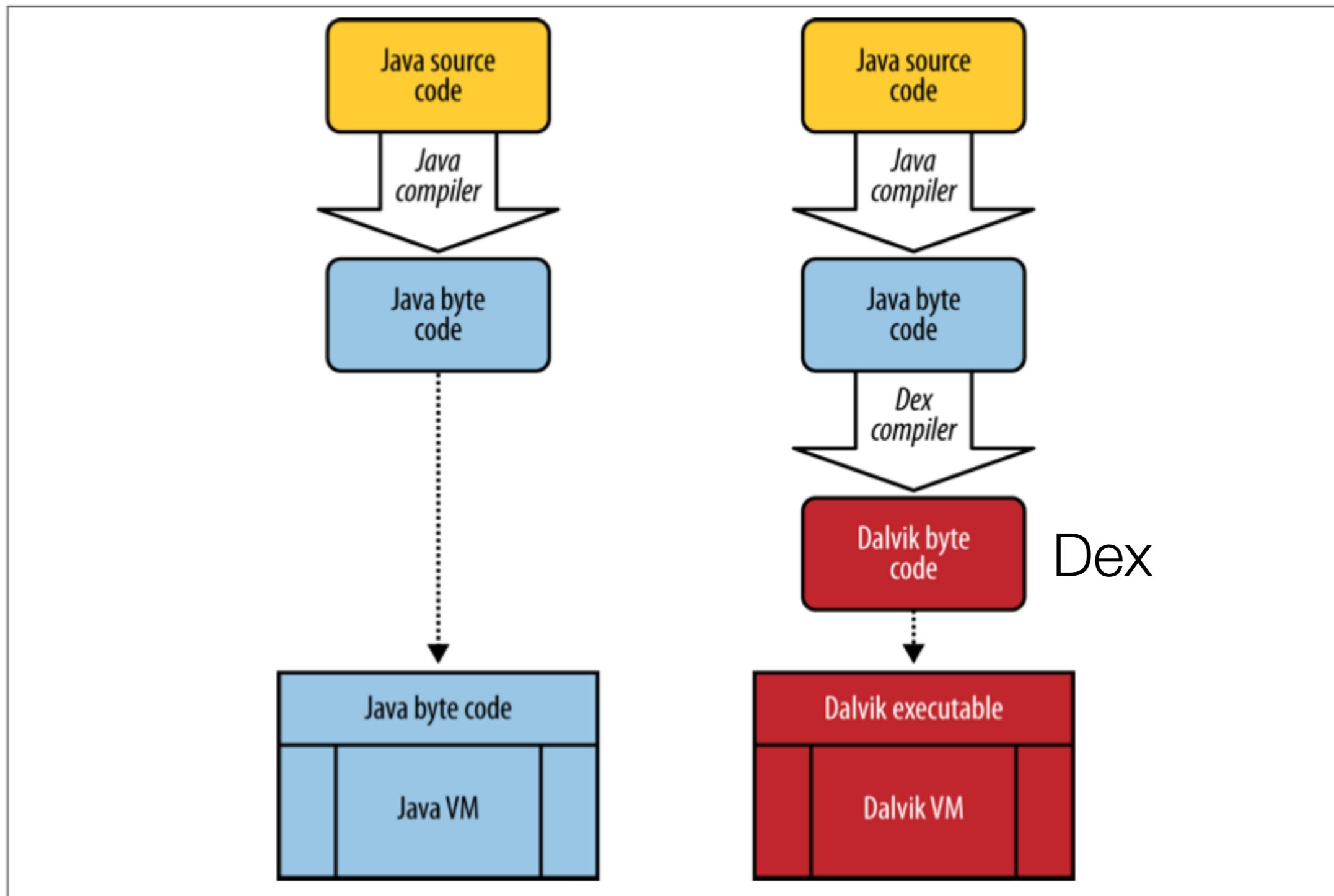
Application Frameworks



- The application framework is a rich environment that provides numerous libraries and services to help the app developer
- This is the best-documented and most extensively covered part of the platform because it is this layer that empowers developers to get applications to the market.
- In the application framework layer, there are numerous Java libraries specifically built for Android. These purpose-built Android classes live in android.* packages.
- There are also most of the standard Java libraries, such as java.lang.* , java.utils.* , java.io.* , java.net.* , etc, which behave as documented in the oracle documentation
- You will also find many services (or managers) that provide the ecosystem of capabilities your application can tap into, such as location, sensors, WiFi, telephony, etc...

JVM vs Dalvik

- In Java, you write your Java source file, compile it into Java **byte code** using the Java compiler, and then run this byte code on the Java VM.
- In Android you write the Java source file, and you still compile it to Java byte code using the same Java compiler.
- But at that point, you recompile it once again to Dalvik byte code using the Dalvik compiler - producing a **DEX** file
- It is this Dalvik byte code - **DEX Code** - that is then executed on the Dalvik

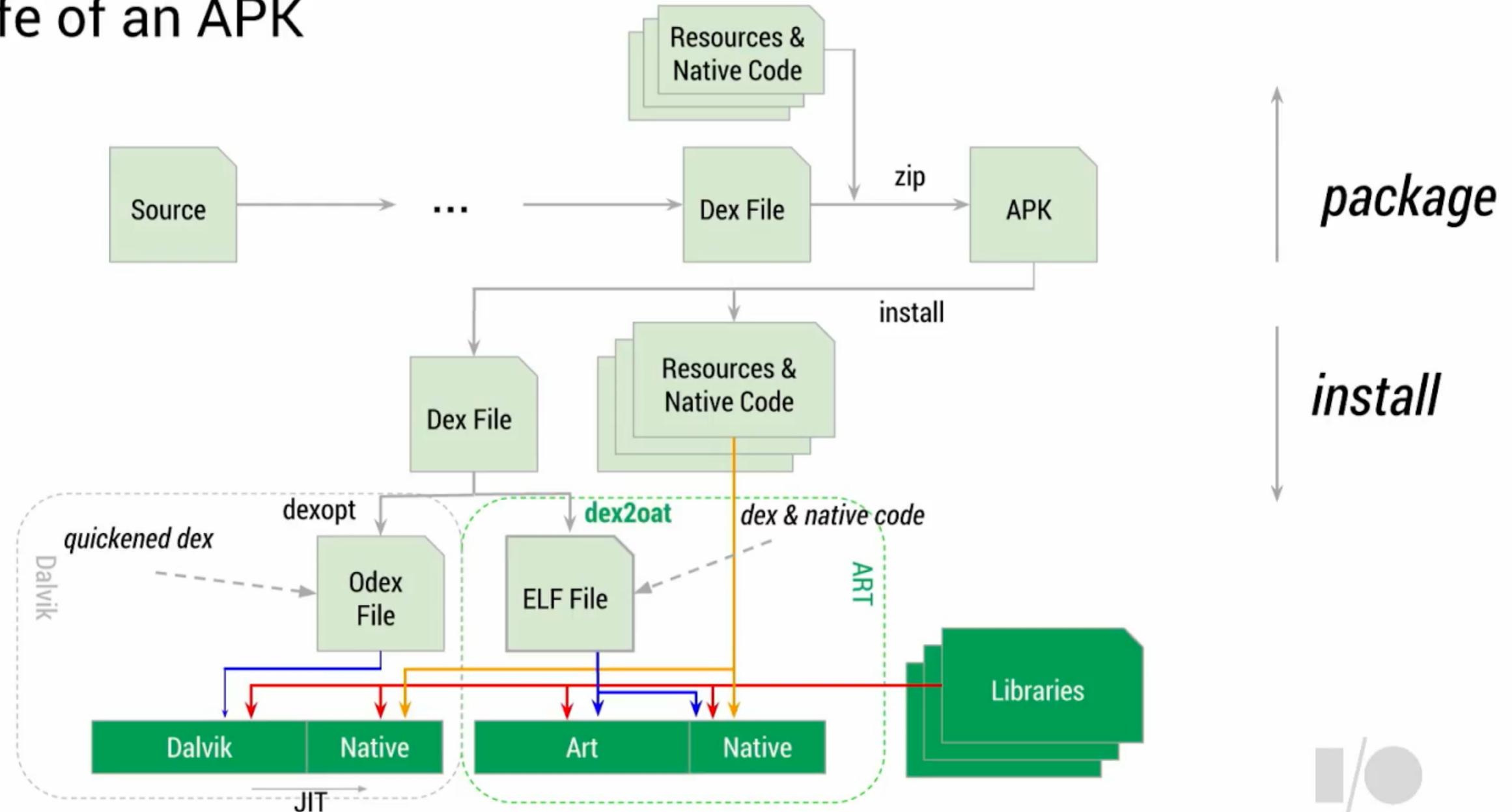




Android 4.4 and **ART**

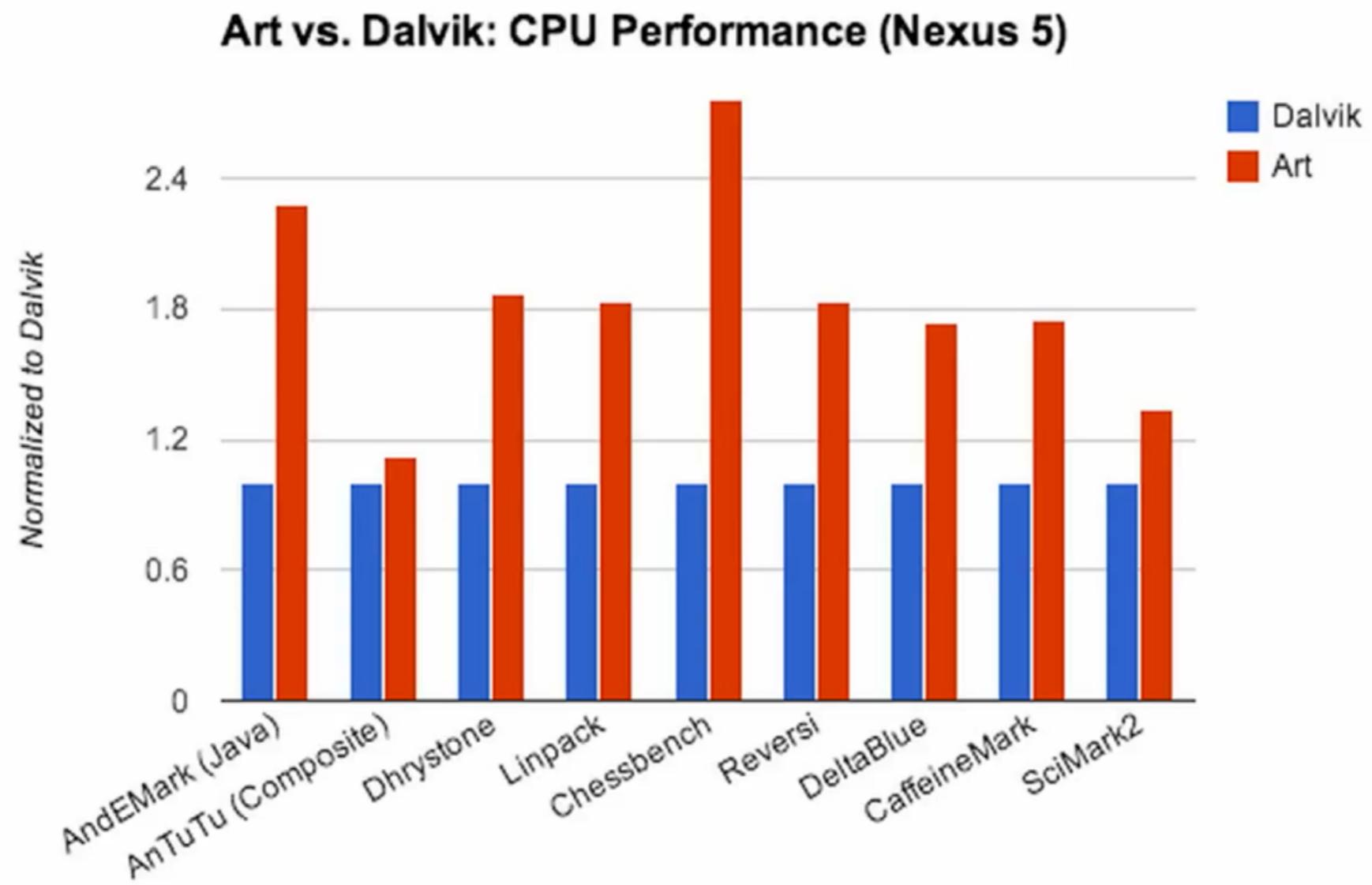
- ART, which stands for Android Runtime, handles app execution in a fundamentally different way from Dalvik.
- The big shift that ART brings, is that instead of being a Just-in-Time (JIT) compiler, it now compiles application code Ahead-of-Time (AOT).
- The runtime goes from having to compile from bytecode to native code each time you run an application, to having it to do it only once, and any subsequent execution from that point forward is done from the existing compiled native code.

The life of an APK



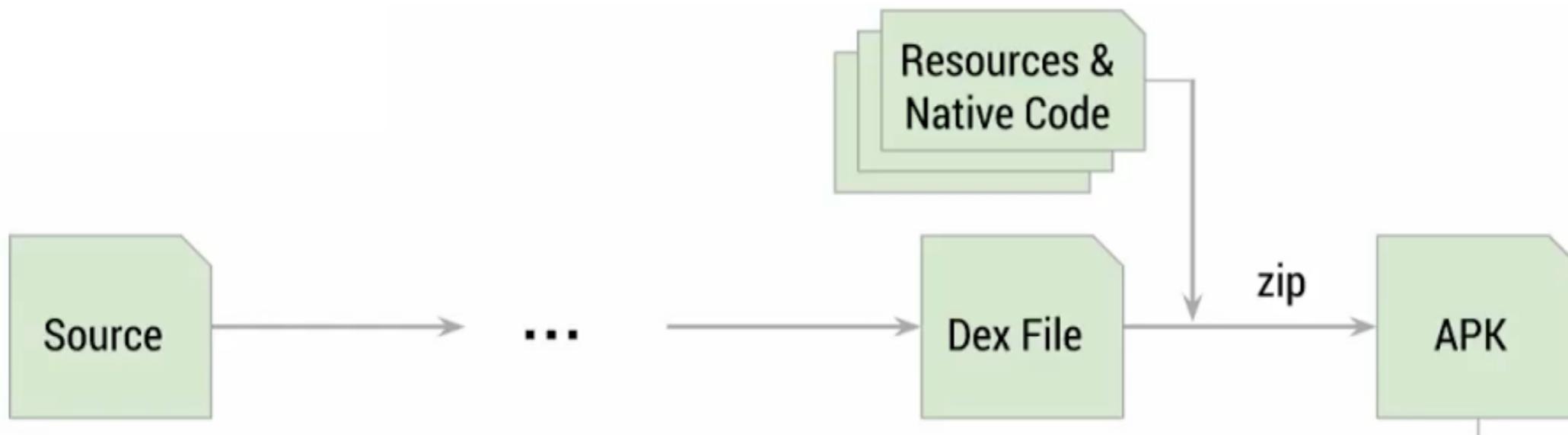
- ART is compatible with Dalvik's existing byte-code format ("dex").
- From a developer's perspective, there are no changes at all in terms of having to write applications for one or the other runtime and no need to worry about compatibilities.

Performance Boosting Thing, realized



Applications

- An application is a single file. We call it an Android application package, or APK for short.
- It is a ZIP file that you can unzip and look inside using any archiving tool



Components of an APK (1)

- **Android Manifest file**

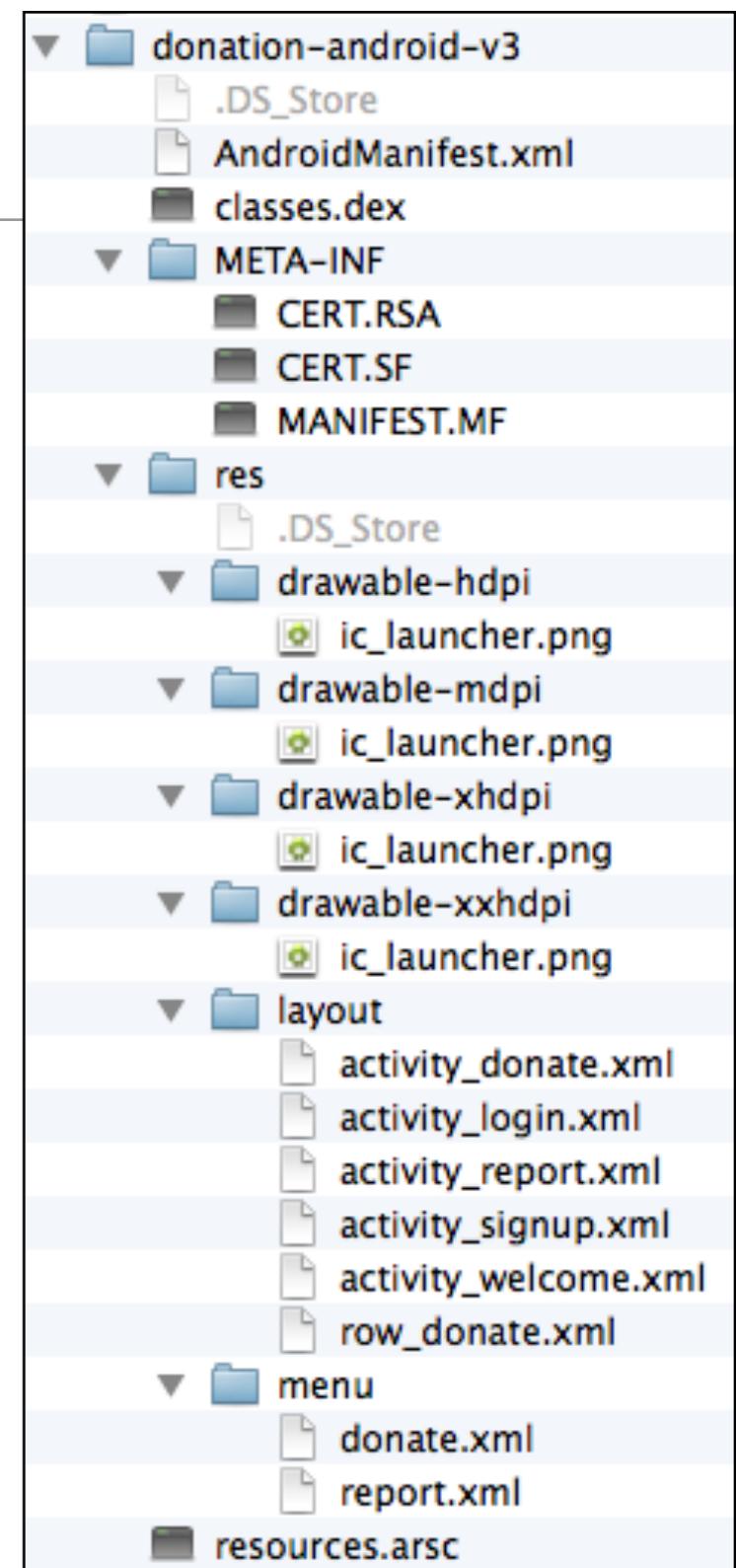
- This is the main file that provides the big picture about your app—all of its components, permissions, version, and minimum API level needed to run it.

- **Dalvik executable**

- This is all your Java source code compiled down to a Dalvik executable. The Dalvik executable is the code that runs your application. It is located in a file called classes.dex.

- **Resources**

- Resources are everything that is not code. Your application may contain a number of images and audio/video clips, as well as numerous XML files describing layouts, language packs, and so on. Collectively, these items are the resources. They are in a file called resources.arsc inside the APK archive as well as in subdirectories such as drawable for images.



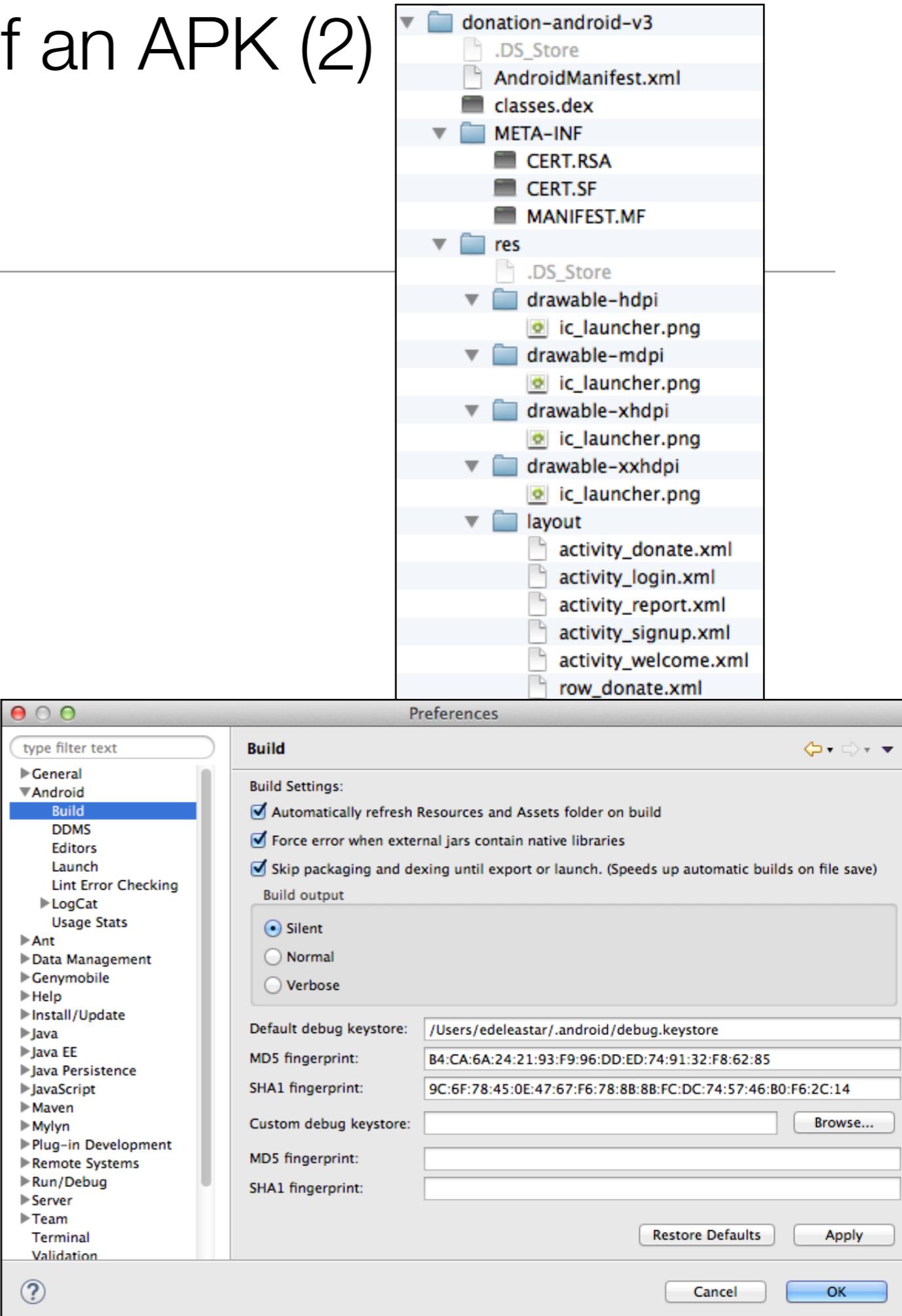
Components of an APK (2)

Native libraries

- Optionally, your application may include some native code, such as C/C++ libraries. These libraries could be packaged together with your APK file.

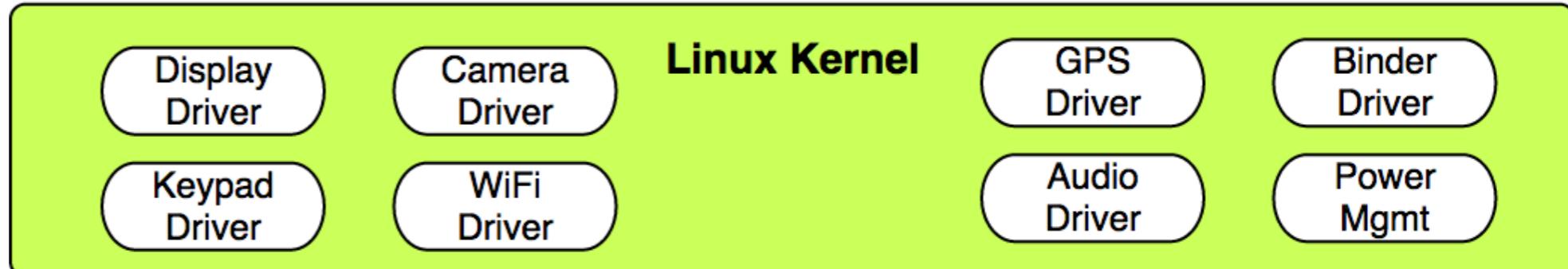
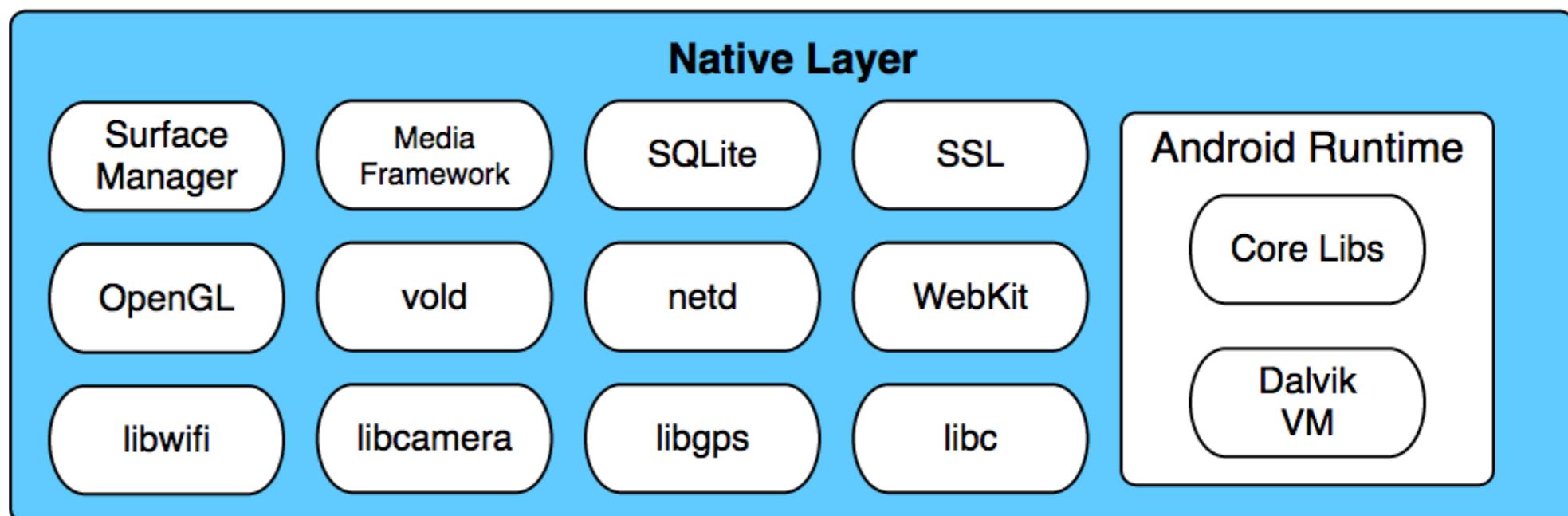
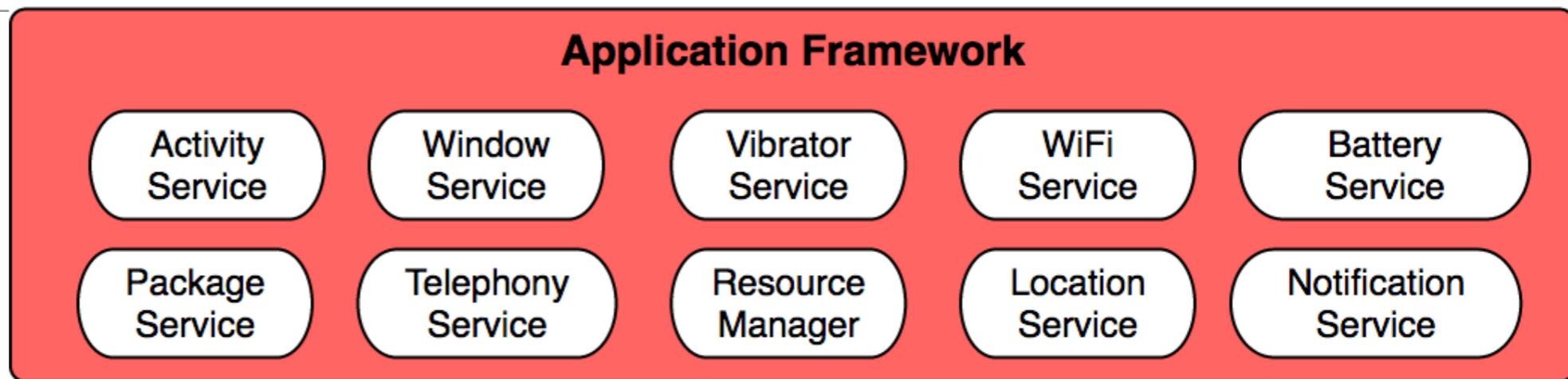
Signatures

- Your APK file also contains a digital signature certifying that you are the author of this application. Signatures are in the META-INF folder.
- Android applications must be signed before they can be installed on a device. For development purposes, we sign applications with a debug key—a key that you already have on your development platform. However, when you distribute your application commercially, you must sign it with your own key.

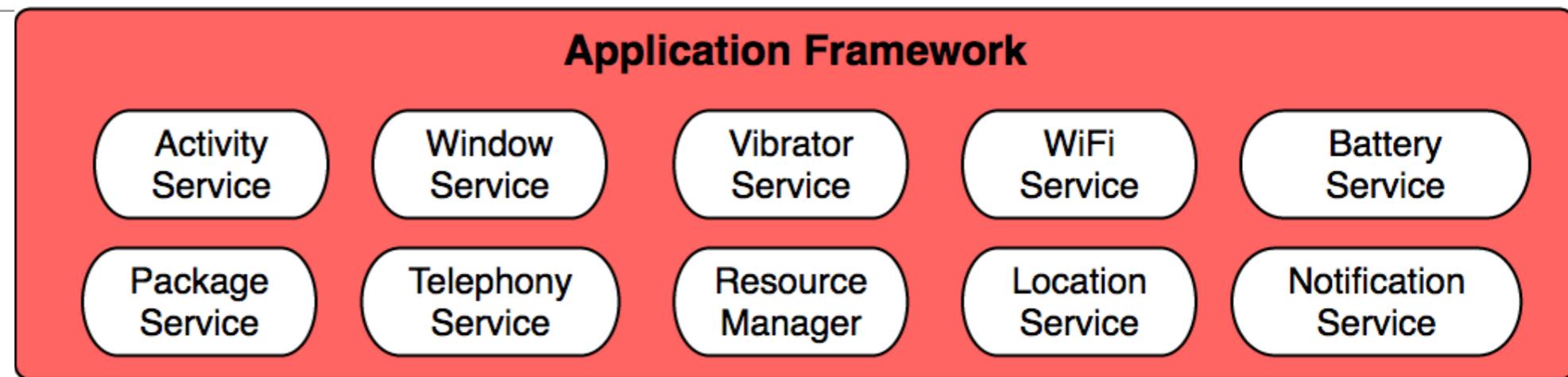


Android Stack

- Which part should we learn?



Android Stack



- Almost exclusively - the Application Framework
- Learning Resources?

The screenshot shows a web browser window displaying the official Android Developers website at developer.android.com/index.html. The page features a navigation bar with tabs for "Developers" (selected), "Design", "Develop", and "Distribute". A search bar is located in the top right corner. The main content area includes a large image of two devices (a smartphone and a tablet) running the Android L Developer Preview. To the right of the devices, there is a section titled "L Developer Preview" with a description and a red "Learn More" button. Below this, there are three cards: "Building Apps for Wearables" (with a thumbnail of a person wearing a smartwatch), "Material Design" (with a thumbnail of overlapping colored squares), and "Android Studio" (with a thumbnail of the Android robot holding a gear). At the bottom of the page is a green footer bar with links: "Get the SDK >", "Browse Samples >", "Watch Videos >", and "Manage Your Apps >".

Android Developers

developer.android.com/index.html

Developers | Design | Develop | Distribute

L Developer Preview

The L Developer Preview lets you design and develop against the next major release of Android. Take the time to test and build your app before the platform officially launches.

Learn More

6.32 Building Apps for Wearables Learn how to build notifications, send and sync data, and use voice actions.

Material Design Learn how to apply material design to your apps.

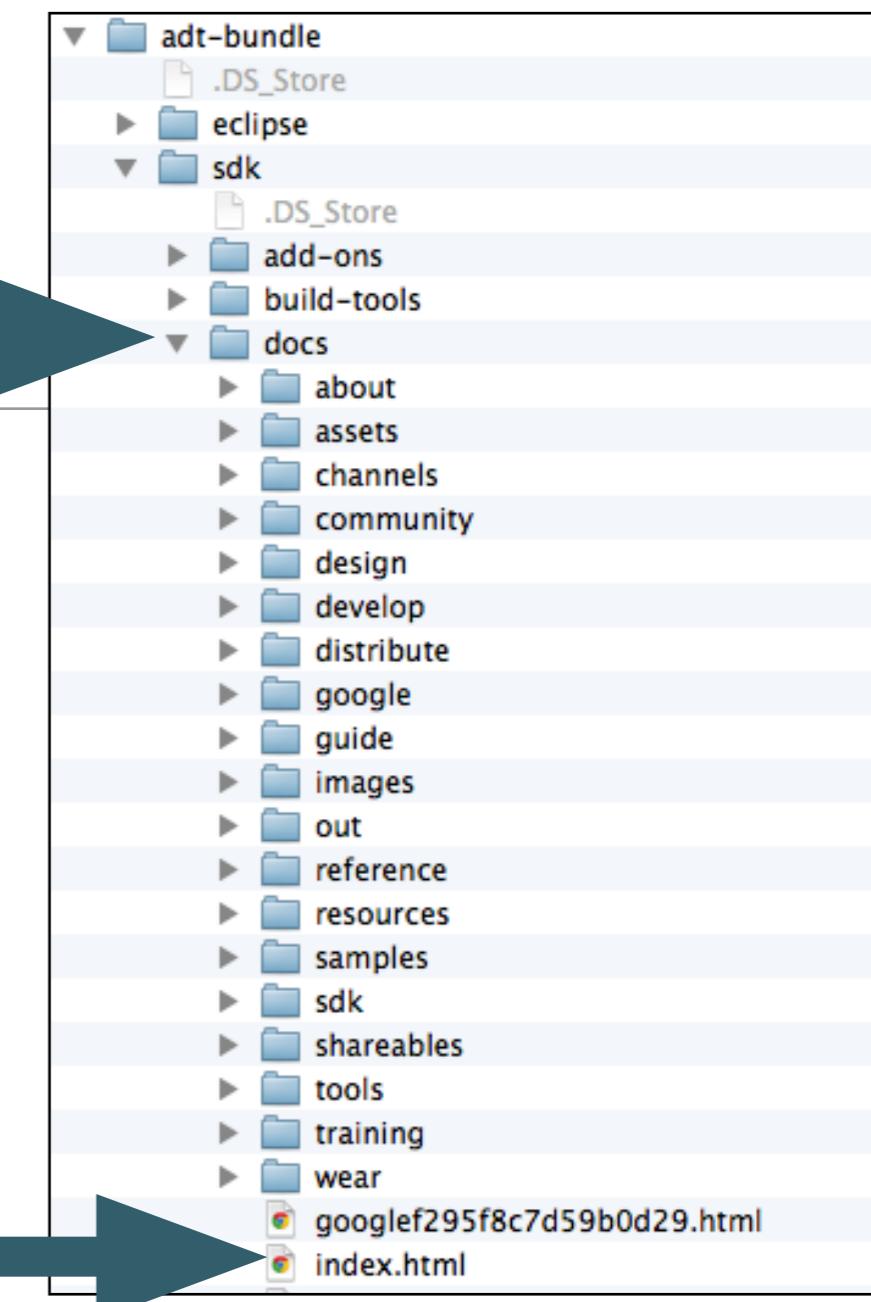
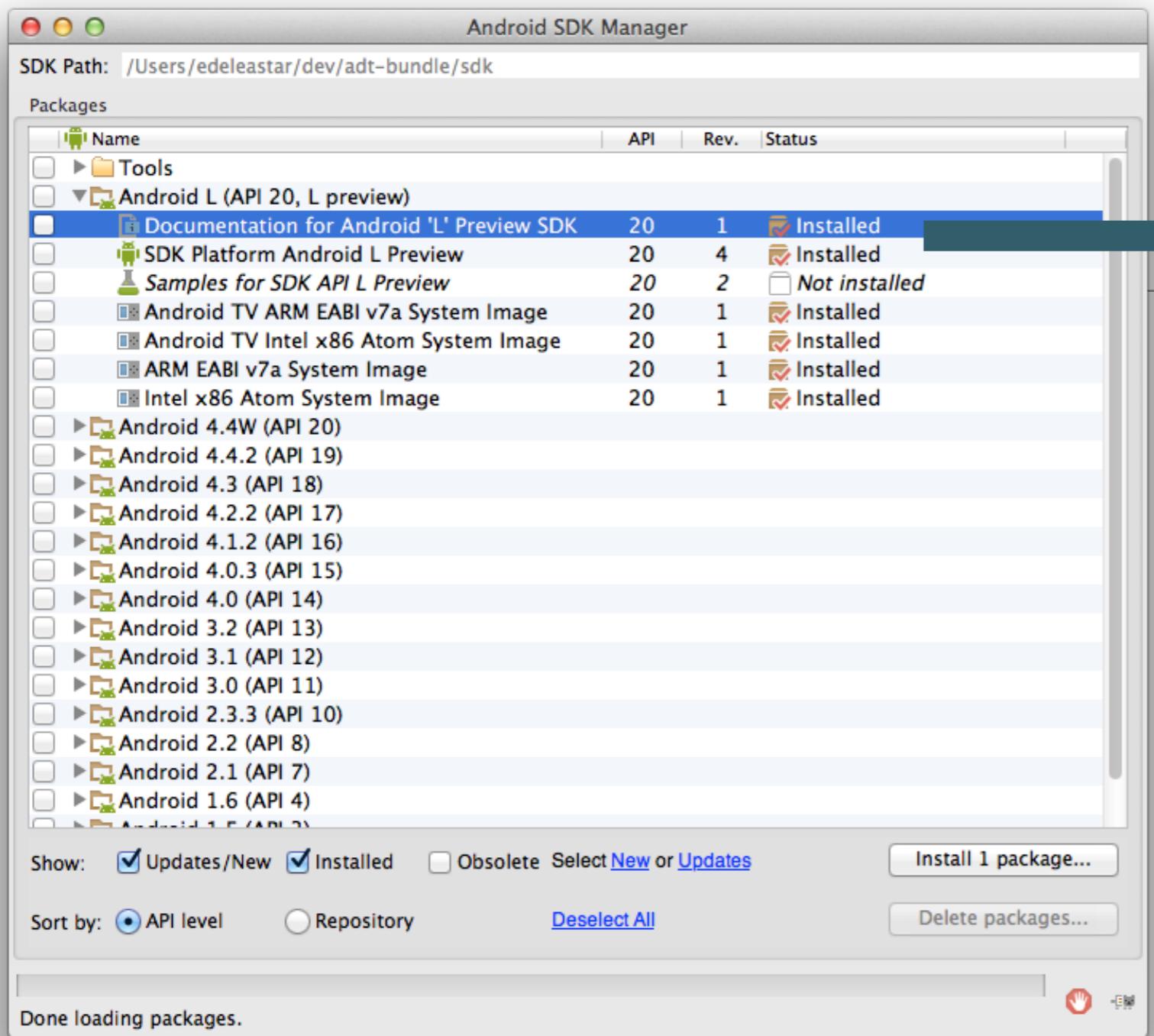
Android Studio Learn about the new features in the beta release of our new IDE.

Get the SDK >

Browse Samples >

Watch Videos >

Manage Your Apps >



- Latest version of documentation can be downloaded locally via the SDK Manager
- Can then be browsed as a static web site

Design

Design | Android Developers

developer.android.com/design/index.html

Developers | Design | Develop | Distribute

Get Started

Devices

Style

Patterns

Building Blocks

Creative Vision >

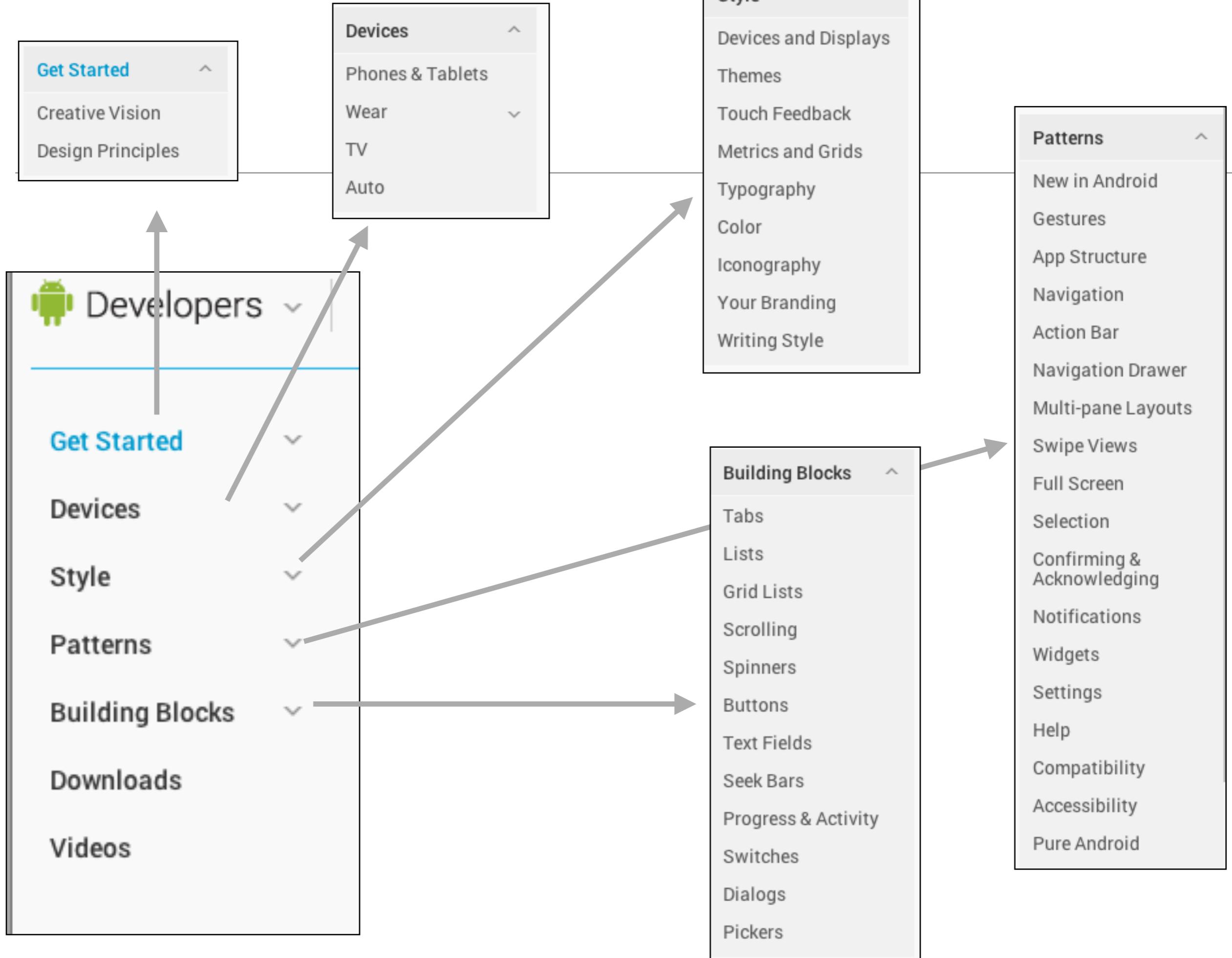
Downloads

Videos

L Developer Preview

The next version of Android uses a design metaphor inspired by paper and ink that provides a reassuring sense of tactility. Before it arrives for users, you can get an early look at the new Material design.

[Learn more about Material](#)



Develop

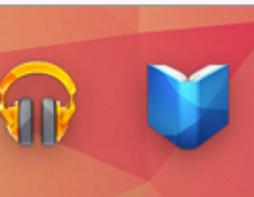
The screenshot shows a web browser displaying the developer.android.com/develop/index.html page. The page has a header with tabs for Developers (selected), Design, Develop (highlighted in orange), and Distribute. Below the tabs are links for Training, API Guides, Reference, Tools, Google Services, and Samples. The main content area features a large image of three devices (laptop, tablet, smartphone) showing a "PHOTOHUNT" app interface. To the right of the image is a section titled "New Cross-Platform Single Sign On" with a "Read more" button. Below this are sections for Developer News, Featured Docs, Developers Live, and Video Playlists.

New Cross-Platform Single Sign On

Google+ Sign-In is an easy, trusted way to sign a user into your app. Now it's even more seamless. A user can sign in to your app on one device and pick it up on another—without signing in again. Best of all, it's built into Google+ Sign-in, so there's no change needed in your app.

[Read more](#)

DEVELOPER NEWS

 [Making Beautiful Android App Icons](#)
As higher density screens gain popularity, it's important to make sure your launcher icon is crisp and high quality...

 [The Beautiful Design Summer 2013 Collection](#)
See the apps chosen by the Android Design team for their masterfully crafted design details...

FEATURED DOCS

 [DevBytes: Google Play Services 4.4](#)
Another release of Google Play services is out - version 4.4. This release includes a blockbuster announcement: Street View ...

 [Media Router Framework - Part 2 - MediaRouteProvider](#)
Learn why and how to implement a MediaRouteProvider in this second part of our series on the Media Router Framework. To learn...

DEVELOPERS LIVE

 [Media Router Framework - Part 2 - MediaRouteProvider](#)
Learn why and how to implement a MediaRouteProvider in this second part of our series on the Media Router Framework. To learn...

VIDEO PLAYLISTS

Develop/Training

The screenshot shows a web browser displaying the [Android Developers Training site](http://developer.android.com/training/index.html). The page is titled "Getting Started". The navigation bar at the top includes links for "Design", "Develop" (which is highlighted in orange), and "Distribute". Below the navigation bar, there are tabs for "Training" (also highlighted in orange), "API Guides", "Reference", "Tools", "Google Services", and "Samples".

The main content area features a sidebar on the left with a hierarchical menu of training topics:

- Getting Started
 - Building Apps with Content Sharing
 - Building Apps with Multimedia
 - Building Apps with Graphics & Animation
 - Building Apps with Connectivity & the Cloud
 - Building Apps with User Info & Location
 - Building Apps for Wearables
 - Best Practices for Interaction & Engagement
 - Best Practices for User Interface
 - Best Practices for User Input
 - Best Practices for Background Jobs
 - Best Practices for Performance
 - Best Practices for Security & Privacy
 - Best Practices for Testing
 - Using Google Play to Distribute & Monetize

The main content area contains several sections with numbered icons (1, 2, 3) and descriptive text:

- ## Getting Started

Welcome to Training for Android developers. Here you'll find sets of lessons within classes that describe how to accomplish a specific task with code samples you can re-use in your app. Classes are organized into several groups you can see at the top-level of the left navigation.

This first group, *Getting Started*, teaches you the bare essentials for Android app development. If you're a new Android app developer, you should complete each of these classes in order:
- ## Building Your First App

After you've installed the Android SDK, start with this class to learn the basics about Android app development.

[Creating an Android Project](#)
[Running Your Application](#)
[Building a Simple User Interface](#)
[Starting Another Activity](#)
- ## Adding the Action Bar

The action bar is one of the most important design elements you can implement for your app's activities. Although first introduced with API level 11, you can use the Support Library to include the action bar on devices running Android 2.1 or higher.

[Setting Up the Action Bar](#)
[Adding Action Buttons](#)
[Styling the Action Bar](#)
[Overlaying the Action Bar](#)
- ## Supporting Different Devices

How to build your app with alternative resources that provide an optimized user experience on multiple device form factors using a single APK.

[Supporting Different Languages](#)
[Supporting Different Screens](#)
[Supporting Different Platform Versions](#)
- ## Managing the Activity Lifecycle

How Android activities live and die and how to create a seamless user experience by implementing lifecycle callback methods.

[Starting an Activity](#)
[Pausing and Resuming an Activity](#)
[Stopping and Restarting an Activity](#)
[Recreating an Activity](#)

Develop/API Guides

The screenshot shows a web browser window displaying the Android Developers website at developer.android.com/training/index.html. The page is titled "Getting Started" under the "Training" category. The left sidebar lists various topics under "Getting Started", each with a small icon and a brief description. The main content area contains several sections: "Getting Started", "Building Your First App", "Adding the Action Bar", "Supporting Different Devices", and "Managing the Activity Lifecycle". Each section includes a summary and a list of related articles.

Getting Started

- Building Apps with Content Sharing
- Building Apps with Multimedia
- Building Apps with Graphics & Animation
- Building Apps with Connectivity & the Cloud
- Building Apps with User Info & Location
- Building Apps for Wearables
- Best Practices for Interaction & Engagement
- Best Practices for User Interface
- Best Practices for User Input
- Best Practices for Background Jobs
- Best Practices for Performance
- Best Practices for Security & Privacy
- Best Practices for Testing
- Using Google Play to Distribute & Monetize

Getting Started

Welcome to Training for Android developers. Here you'll find sets of lessons within classes that describe how to accomplish a specific task with code samples you can re-use in your app. Classes are organized into several groups you can see at the top-level of the left navigation.

This first group, *Getting Started*, teaches you the bare essentials for Android app development. If you're a new Android app developer, you should complete each of these classes in order:

Building Your First App

After you've installed the Android SDK, start with this class to learn the basics about Android app development.

Creating an Android Project
Running Your Application
Building a Simple User Interface
Starting Another Activity

Adding the Action Bar

The action bar is one of the most important design elements you can implement for your app's activities. Although first introduced with API level 11, you can use the Support Library to include the action bar on devices running Android 2.1 or higher.

Setting Up the Action Bar
Adding Action Buttons
Styling the Action Bar
Overlaying the Action Bar

Supporting Different Devices

How to build your app with alternative resources that provide an optimized user experience on multiple device form factors using a single APK.

Supporting Different Languages
Supporting Different Screens
Supporting Different Platform Versions

Managing the Activity Lifecycle

How Android activities live and die and how to create a seamless user experience by implementing lifecycle callback methods.

Starting an Activity
Pausing and Resuming an Activity
Stopping and Restarting an Activity
Recreating an Activity

Develop/Reference

The screenshot shows a web browser displaying the Android Package Index at developer.android.com/reference/packages.html. The page is part of the 'Develop' section of the Android Developers website. The navigation bar includes links for Developers, Design, Develop, Distribute, Training, API Guides, Reference (which is currently selected), Tools, Google Services, and Samples. A search bar is also present.

The main content area is titled 'Package Index' and displays the 'Android APIs API level: 18'. On the left, there is a sidebar with a list of packages starting with 'android' and including 'android.accessibilityservice', 'android.accounts', 'android.animation', 'android.app', 'android.app.admin', 'android.app.backup', 'android.appwidget', 'android.bluetooth', 'android.content', 'android.content.pm', 'android.content.res', 'android.database', and 'android.database.sqlite'. Below this list is a placeholder text: 'Select a package to view its members'.

The main content area contains a table listing various Android packages:

Package	Description
android	Contains resource classes used by applications included in the platform and defines application permissions for system features.
android.accessibilityservice	The classes in this package are used for development of accessibility service that provide alternative or augmented feedback to the user.
android.accounts	
android.animation	These classes provide functionality for the property animation system, which allows you to animate object properties of any type. <code>int</code> , <code>float</code> , and hexadecimal color values are supported by default. You can animate any other type by telling the system how to calculate the values for that given type with a custom TypeEvaluator . For more information, see the Animation guide.
android.app	Contains high-level classes encapsulating the overall Android application model.
android.app.admin	Provides device administration features at the system level, allowing you to create security-aware applications that are useful in enterprise settings, in which IT professionals require rich control over employee devices. For more information, see the Device Administration guide.
android.app.backup	Contains the backup and restore functionality available to applications. If a user wipes the data on their device or upgrades to a new Android-powered device, all applications that have enabled backup can restore the user's previous data when the application is reinstalled. For more information, see the Data Backup guide.
android.appwidget	Contains the components necessary to create "app widgets", which users can embed in other applications (such as the home screen) to quickly access application data and services without launching a new activity. For more information, see the App Widgets guide.
android.bluetooth	Provides classes that manage Bluetooth functionality, such as scanning for devices, connecting with devices, and managing data exchange between devices. The Bluetooth API supports both "Classic"

The bottom of the page shows the URL developer.android.com/reference/android/accessibilityservice/package-summary.html.

Develop/tools

Android SDK | Android Dev X developer.android.com/sdk/index.html

Developers | Design | Develop | Distribute | [Search](#)

Training API Guides Reference Tools Google Services Samples

Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in ADT (Android Developer Tools) to streamline your Android app development.

[Download Eclipse ADT with the Android SDK for Mac](#)

With a single download, the Eclipse ADT bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- A version of the Android platform
- A version of the Android system image for the emulator

If you prefer to use an existing version of Eclipse or another IDE, you can instead download the stand-alone Android SDK Tools:

- GET THE SDK FOR AN EXISTING IDE
- SYSTEM REQUIREMENTS
- VIEW ALL DOWNLOADS AND SIZES

Except as noted, this content is licensed under [Creative Commons Attribution 2.5](#). For details and restrictions, see the [Content License](#).

[About Android](#) | [Legal](#) | [Support](#)



Get Android Studio Beta

Android Studio is a new IDE powered by IntelliJ that provides new features and improvements over ADT. It's currently in beta but will be the official Android IDE once it's ready.

If you're a new Android developer, you should consider starting with Android Studio, because the ADT plugin for Eclipse is no longer in active development.

[Learn more about Android Studio](#)

Develop/Google Services

The screenshot shows a web browser window displaying the developer.android.com/google/index.html page. The page is titled "Google Services" and features a navigation bar with tabs for Developers, Design, Develop (which is highlighted), Distribute, Training, API Guides, Reference, Tools, Google Services (which is also highlighted), and Samples. A search bar and a menu icon are also present.

The main content area is titled "Google Services" and contains a brief introduction: "Google offers a variety of services that help you build new revenue streams, manage app distribution, track app usage, and enhance your app with features such as maps, sign-in, and cloud messaging." Below this, there are several service cards:

- Google Maps**: Includes a description of how to embed Google Maps in an app, mentioning markers, overlays, and custom perspectives.
- Google Play In-App Billing**: Describes building a steady revenue stream through Google Play.
- Google Wallet Instant Buy**: Provides fast checkout for physical goods and services.
- Google Cloud Platform**: Details building a backend at Google scale with managed infrastructure.
- Google Analytics**: Measures user engagement and provides insights into app content.
- Google Cloud Messaging**: Focuses on integrating Google Cloud Messaging for app notifications.
- Google Mobile Ads**: Details managing app distribution and advertising.

On the left sidebar, there's a "Overview" section with links to Games, Location, Google+, Maps, Drive, Cast, Ads, Wallet, Google Play Services, Google Play In-app Billing, Google Cloud Messaging, Google Cloud Save, and Google Play Distribution.

Recommended Texts

