

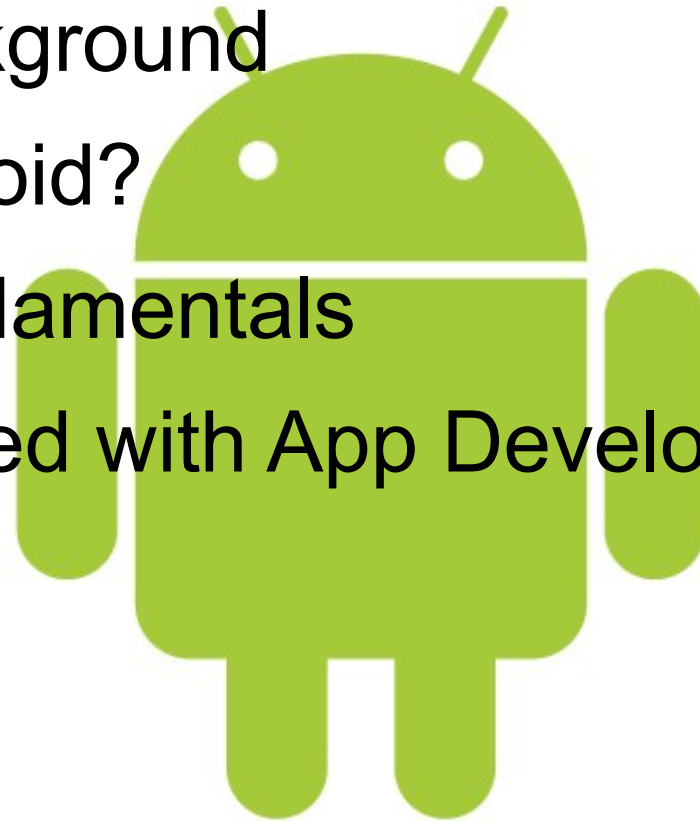
# **Introduction to Android Application Development**

**Mike Kvintus  
Principal Engineer  
JDSU**






# Agenda

- Android Background
- What is Android?
- Android Fundamentals
- Getting Started with App Development
- Demo
- Tips/Links





# Android Background

- Open Source Mobile Operating System from Android Inc, which was purchased by Google  
<http://source.android.com/>
- Part of the Open Handset Alliance
- Releases named after desserts
  - 1.5 – Cupcake April 30, 2009 
  - 1.6 – Donut Sept 15, 2009 
  - 2.0/2.0.1 – Eclair Oct 26, 2009 
  - 2.1 – Eclair Jan 12, 2010



# Why S hould You Care

- Android is gaining momentum
  - Share of the smartphone market grew from 2.8% to 3.5% in Q3 2009
  - October, 2009, Gartner Inc. predicted that by 2012, Android would become the world's second most popular smartphone platform, behind only Symbian OS
  - Estimated 250,000 Motorola Droid phones were sold in the US during the phone's first week in stores
- Open Source OS and free SDK/IDE
- Runs a Linux kernel
- App development uses the Java language syntax



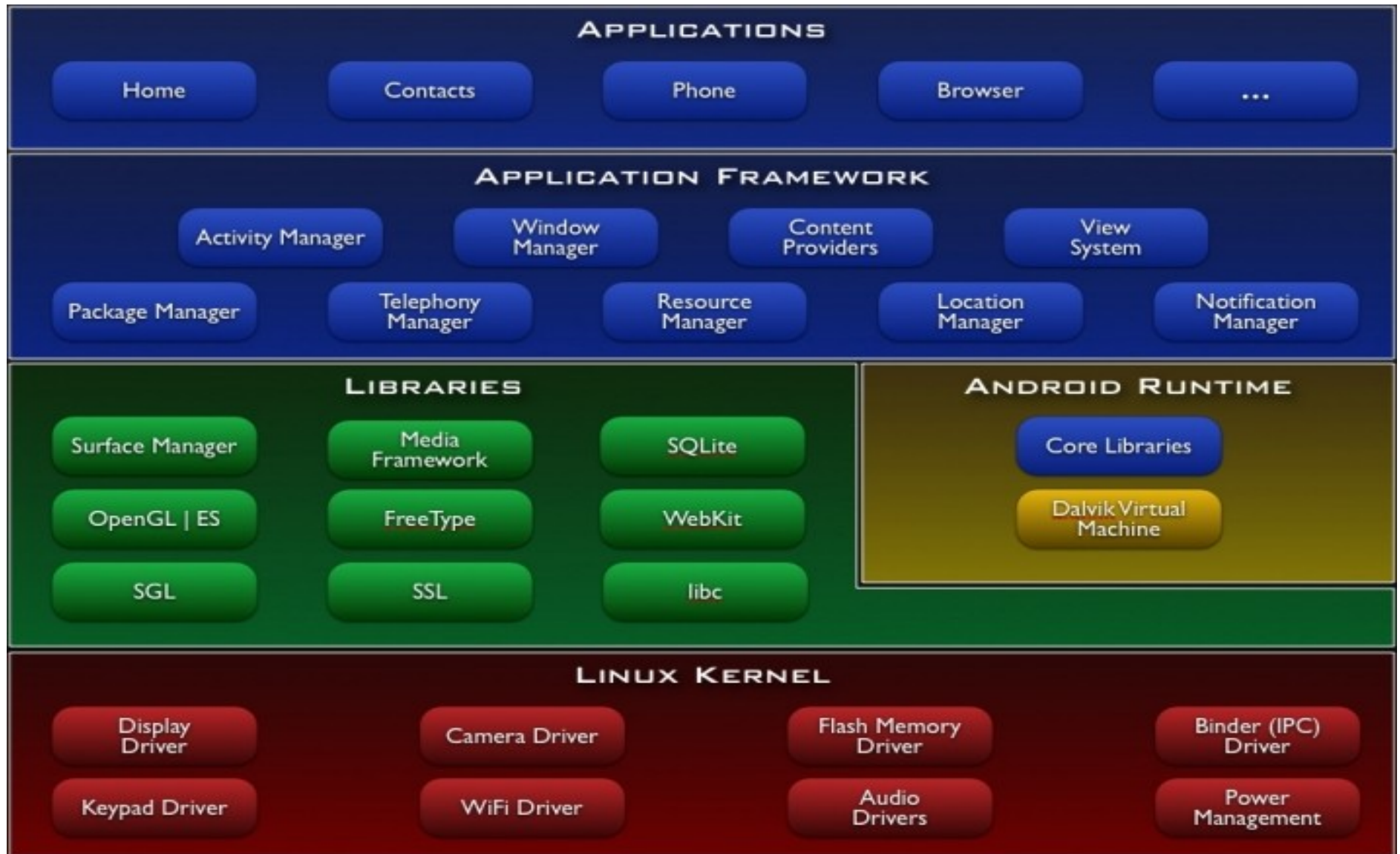


# Android Features

- Application framework enabling reuse and replacement of components
- Dalvik virtual machine optimized for mobile devices
- Integrated browser based on the open source WebKit engine
- Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, and WiFi (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE



# Android Architecture



# Application Framework

- Underlying all applications is a set of services and systems, including:
  - A rich and extensible set of views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
  - Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data
  - A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
  - A Notification Manager that enables all applications to display custom alerts in the status bar
  - An Activity Manager that manages the lifecycle of applications and provides a common navigation backstack



# Libraries

- System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- Media Libraries - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view
- SGL - the underlying 2D graphics engine
- 3D libraries - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- FreeType - bitmap and vector font rendering
- SQLite - a powerful and lightweight relational database engine available to all applications



# Runtime Environment

- Dalvik VM
  - Uses the Java language syntax, but does not provide the full-class libraries and APIs bundled with Java SE or ME
  - The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint
  - The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool
- Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model
  - The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.
- It can be difficult to maintain applications working on different versions of Android, because of various compatibility issues between versions 1.5 and 1.6, specifically concerning the different resolution ratios of the various Android phones





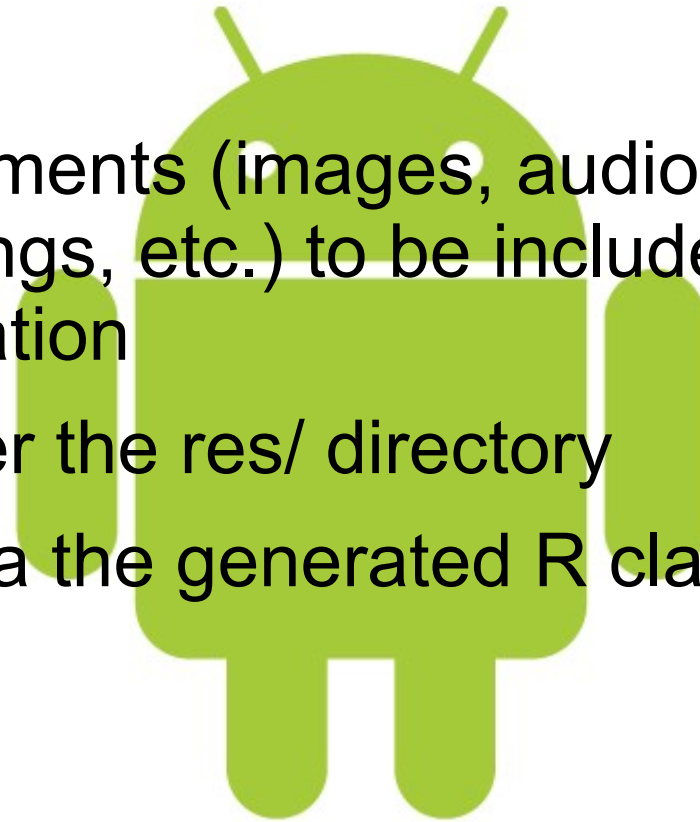
# Applications

- Compiled Java code and resources are bundled into an .apk file
  - Each .apk file is considered an application.
- By default, every application runs in its own Linux process with its own JVM so code runs in isolation from other applications
- By default each application is assigned a unique Linux user ID with security set so other user IDs can't read the application's files
  - It's possible for 2 applications to share a single Linux user id.



# Application Components

- Resources
  - External elements (images, audio, video, layouts, themes, strings, etc.) to be included and referenced in an application
  - Reside under the res/ directory
  - Accessed via the generated R class



# Application Components

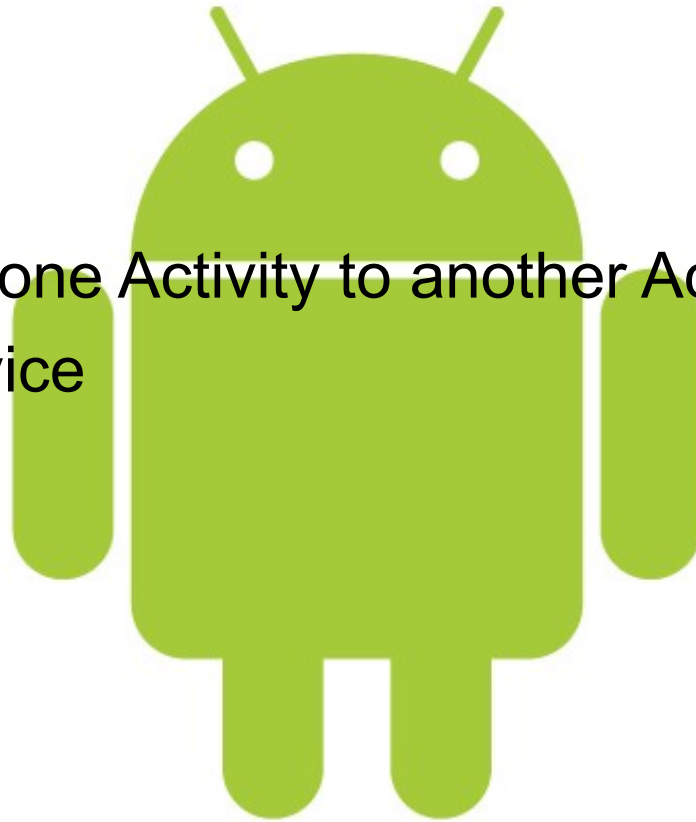
- Activities

- Presents a visual user interface focused on one task
- Apps can have 1 or more activities
- Each activity is given a default window to draw in
- Visual content of the window is provided by a hierarchy of views
- One activity is designated the first one shown
- Move from one activity to another by having the current activity start the next activity
- Launched by passing an Intent object to `Context.startActivity()` or `Activity.startActivityForResult()`
- Stopped by calling `finish()` or `finishActivity()` of another Activity



# Application Components

- Intents
  - “Verbs”
    - Move from one Activity to another Activity
    - Start a service



# Application Components

- Services

- No UI and run in the background for an indefinite period of time
- Code can “bind” to a service and communicate with the service via the interface the service exposes
- Runs on the app's main thread, so time consuming tasks should spawn another thread to avoid blocking the UI or other components
- Launched by passing an Intent object to `Context.startService()`
- Stopped by calling `stopSelf()` or `Context.stopService()`

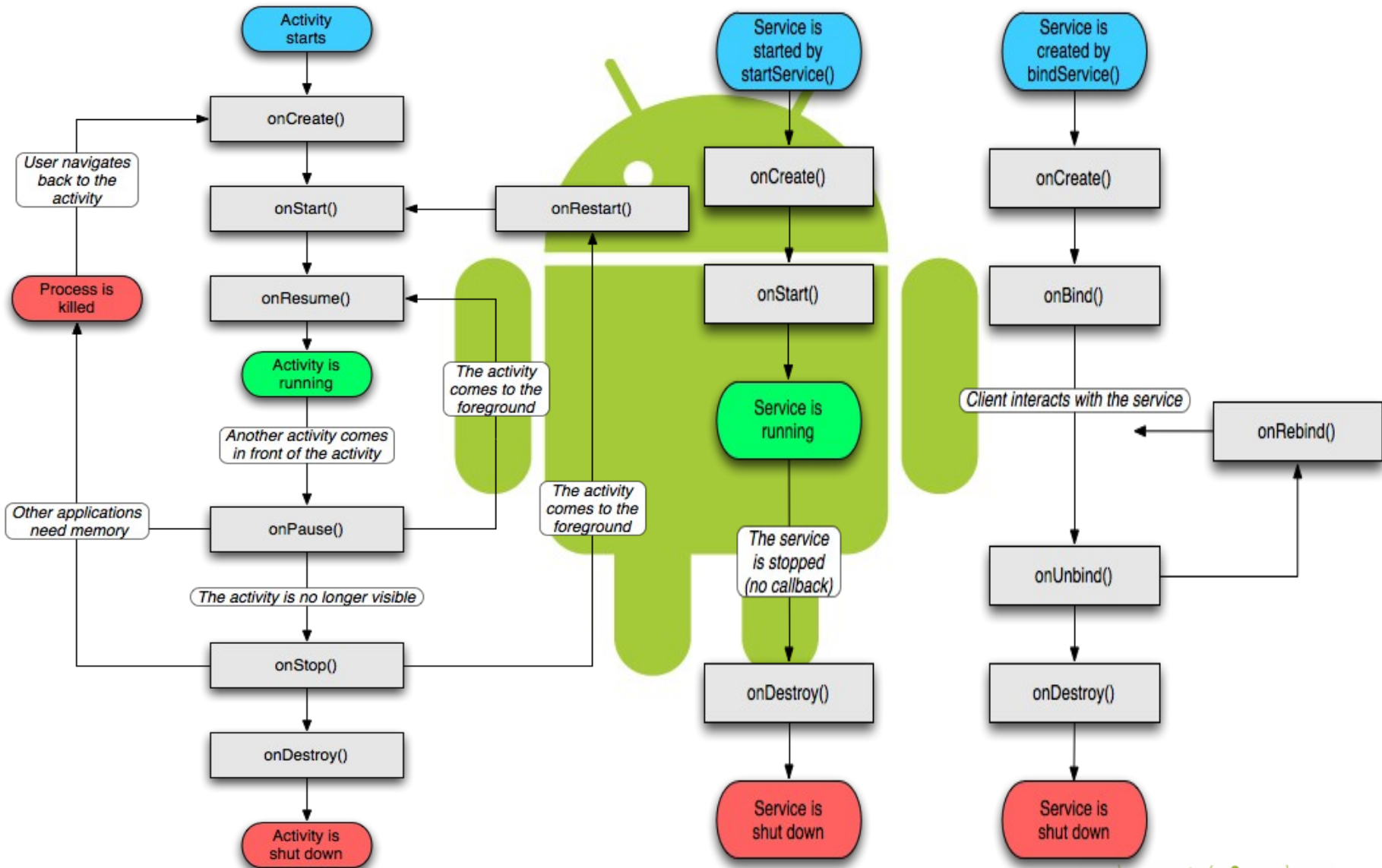


# Application Components

- Broadcast receivers
  - Receives and reacts to broadcast announcements
  - Many broadcasts come from system code but apps can also initiate broadcasts
  - No UI, but can start activities and use NotificationManager to alert the user
- Broadcasts initiated by passing an Intent object to `Context.sendBroadcast()`, `Context.sendOrderedBroadcast()`, or `Context.sendStickyBroadcast()`
- Content providers
  - Makes a set of app's data available to other apps
  - Activated when targeted by a request from a `ContentResolver`



# Component Lifecycles



# The AndroidManifest.xml File

- Names the Java package for the application (the package name serves as a unique identifier for the application)
- Describes the components of the app (activities, services, broadcast receivers, and content providers) \*\*
- Declares which permissions the app must have in order to access protected parts of the API and interact with other apps
- Declares the permissions that others are required to have in order to interact with the app's components
- Declares the minimum level of the Android API required
- Lists the libraries that the app must be linked against
- Determines which processes will host app components



# Location API

- You can build location-based and maps-based capabilities into your applications using the classes of the `android.location` package and the Google Maps external library
- Main component of the location framework is the LocationManager system service
- Provide mock location data when testing your app with the emulator
  - DDMS tool
    - Manually send individual longitude/latitude coordinates to the device or use a GPX file describing a route for playback to the device or use a KML file describing individual placemarks for sequenced playback to the device
  - “geo” command in the console
    - `geo fix -121.45356 46.51119 4392`
    - `geo nmea $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62`



# Maps External Library

- Classes of the `com.google.android.maps` package offer built-in downloading, rendering, and caching of Maps tiles, as well as a variety of display options and controls
- Main class in the Maps package is `com.google.android.maps.MapView`, a subclass of `ViewGroup`
- Wrapper around Google Maps API
- Not part of the standard Android library
- To display Google Maps data in a `MapView`, you must register with the Google Maps service and obtain a Maps API Key

<http://code.google.com/android/add-ons/google-apis/mapkey.html>



# Data Storage

- Preferences
  - Lightweight mechanism to store and retrieve key-value pairs of primitive data types
- Files
  - `Context.openFileInput()` `Context.openFileOutput()`  
`Resources.openRawResource (R.raw.myDataFile)`
- Databases – SQLite – Portable, used by iPhone, Mac, et al
- Network – `java.net.*` and `android.net.*` classes



# Getting Started with App Development



# What Do You Need to Dev Apps?

- x86-architecture computer running Linux (any modern desktop Linux distribution), Mac OS X 10.4.8 or later, Windows XP, Vista, 7
- Java 5 or 6 JDK (JRE alone is not sufficient)
- Apache Ant 1.6.5 or later for Linux and Mac, 1.7 or later for Windows
- Android SDK (<http://developer.android.com/sdk/index.html>)
- Eclipse IDE 3.4 (Ganymede) or 3.5 (Galileo) - optional
  - Eclipse JDT plugin (included in most Eclipse IDE packages)
  - Android Development Tools plugin - <http://developer.android.com/sdk/eclipse-adt.html>
- MOTODEV Studio
  - <http://developer.motorola.com>



# Dev Environment Setup

- Install SDK
  - Install versions you want to support
  - Create Android Virtual Devices (specify release, memory, screen resolution, etc) for emulator
- Install Android SDK plugin in Eclipse
  - Set the location of the Android SDK under Preferences



# Types of Apps

- Native Android apps (SDK, Eclipse, Netbeans, MOTODEV Studio)
- Web apps

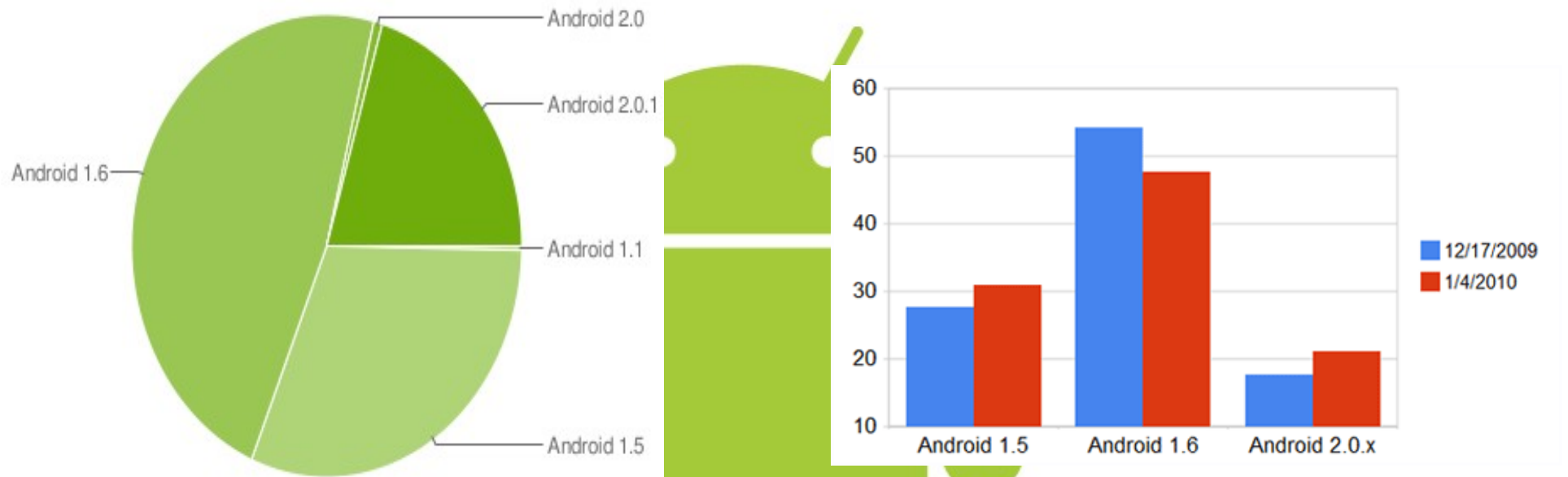


# Steps to Dev Apps Using Eclipse

- Get code signing certificate (optional)
- Get Google Maps API key (required if using Google Maps API)
- Decide on the lowest release of Android you want to target
- Create Android Project
- Create components, layouts, classes, etc
- Test using emulator
- Export application package (.apk)
- Load/test on Android device
- Publish to Android Market (app must be signed, \$25 signup fee) or make app available for download



# Android Fragmentation



Source:  
<http://developer.android.com/resources/dashboard/platform-versions.html>

Source:  
<http://androidandme.com/2010/01/news/google-updates-android-fragmentation-numbers/>



# Setting Up A Phone for Testing Apps

- Add `android:debuggable="true"` to the `<application>` element in `AndroidManifest.xml`
- Turn on "USB Debugging" on your device
- Setup your system to detect your phone
  - Install Windows USB driver
  - See online docs for Linux setup
- Connect phone via USB and verify using "adb devices" command





# Tips

- Make sure to add new components to `AndroidManifest.xml`
- For `ListView`, use `android:cacheColorHint="#00000000"` when using a background image
- Test with different screen resolutions and different devices if possible
- Target the appropriate release of Android



# Other App Dev Tools

- Titanium Mobile  
<http://www.appcelerator.com/products/titanium-mobile/>
- Ruboto - JRuby on Android  
<http://github.com/headius/ruboto-irb>
- ASE – Android Scripting Environment  
<http://code.google.com/p/android-scripting/>
- Many others



# Useful Links

- <http://developer.android.com/>
  - SDK, Developer Guide – Sample apps, Tutorials/Videos
- <http://www.android.com/>
  - General Android info
- <http://www.eclipse.org/>
  - Eclipse IDE
- <http://developer.motorola.com>
  - MOTODEV Studio - @motodev
- <http://developer.android.com/guide/developing/tools/emulator.html#controlling>
  - Commands to control the Android Emulator
- <http://training.oreilly.com/androidapps-java/>
  - Online Course: Developing Android Applications with Java



