Android is a software stack based on a custom Linux OS that includes some standard/new libraries, custom VM and java application frameworks and services.

- **Latest release**
  - Tablet: 3.2.1 honeycomb
  - Phones: 2.3.7 gingerbread

- **Future release**: 4.3
  - Sweeter version of Jelly Bean

- **SDK / App**
  - Eclipse, ADT + emulator
  - Now also an NDK (performance critical part of app C/C++)
  - STK: SIM toolkit
  - Android play store
OHA (open handset alliance)
- More than 30 members in 2007
- More than 70 members in 2010
- 84 members in 2013
- Members: mobile operators, handset manufactures, semiconductor companies, software companies, commercialization companies

Led by Google

Open source, but an application can be modified by the manufactures
- Android : Apache
- Linux & Webkit : LGPLv2
- SQLite: public domain

Source files are mixed: free, open-source, and proprietary

Fragmentation across various manufactures
- Solution : Android standard + manufactures layer + operators layer
Evolution

2009
- 2 smartphones HTC (G1 Dream/G2 magic)
- Display 320x480
- Proc. 528Mhz
- Ram 256MB
- 6000 application (1500/month)
- 4 version: 1.1, 1.5, 1.6, 2.0

2010
- 30 smartphones, 20 tablet
- Display 480x854
- Proc. 1Ghz
- Ram 512MB
- Multitouch
- 100,000 apps (14000/month)
- 4 version: 2.0.1, 2.1, 2.2, 2.3, 3.0
- 1st smartphone OS in US
- Performance and 3D: Open GL ES 2, JIT (2.2), gaming
Chronology and Usage

- **Problem:** Too fast evolution manufacturers & operators

---

**Android Version History**
- **Cupcake**
  - Android 1.5
- **Donut**
  - Android 1.6
- **Eclair**
  - Android 2.0/2.1
- **Froyo**
  - Android 2.2
- **Gingerbread**
  - Android 2.3
- **Honeycomb**
  - Android 3.0
- **Ice Cream Sandwich**
  - Android 4.0

---

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>9.4%</td>
</tr>
<tr>
<td>Normal</td>
<td>79.4%</td>
</tr>
<tr>
<td>Large</td>
<td>6.4%</td>
</tr>
<tr>
<td>Xlarge</td>
<td>4.8%</td>
</tr>
</tbody>
</table>

**Total:** 99.9%
Android Jelly Bean

- Android 4.3 API level 18
- Faster, smoother and more responsive UI
- OpenGL ES 3.0 for high performance graphics
- Enhanced Bluetooth connectivity, single API
  - Both client and server mode
  - Audio/video remote control profile (AVRCP)
- Support of fine-grained restricted profile
- Optimized location and sensor capabilities
  - Keep wifi scanning on for location accuracy
- Support for international users
- Enterprise security
- Analyze and profile the app performance
Development process for Android applications

**Setup**
- Set up your development environment
- Install the Android SDK, Android Development Tools, and Android platforms.
- Set up AVDs and devices for testing
- Create Android Virtual Devices and connect hardware devices that will be used for testing.

**Development**
- Create your application
  - Create an Android project with your source code, resource files, and Android manifest file.

**Debugging and Testing**
- Build and run your application
  - Build and run your application in debug mode.
  - Debug your application using the Android debugging and logging tools
  - Test your application using the Android testing and instrumentation framework.

**Publishing**
- Prepare your application for release
  - Configure, build, and test your application in release mode.
- Release your application
  - Publicize, sell, and distribute your application to users.
Essential Command line utility

- **android** [global options] action [action options]
  - Create and update Android projects and create, move, and delete AVDs
  - `Android -- help`

- **emulator -avd <avd_name> [options]**
  - Run your applications on an emulated Android platform
  - `telnet localhost 5554, and write help`
    - `geo <fix|nmea>, event <send|types|codes|text>, power <display|ac|status|present|health|capacity>, network status, network delay gprs, network speed 14.4 80, gsm <call|accept|busy|cancel|data|hold|list|voice|status>, sms send <senderPhoneNumber> <textmessage>, vm <start|stop|status>`
Essential Command line utility

- **adb [-d|-e|-s <serialNumber>] <command>**
  - Android Debug Bridge is the interface with your emulator or connected device (install apps, shell the device, issue commands, etc.)
  - adb – help, adb devices, adb -s emulator-5556 install helloWorld.apk, adb forward tcp:6100 tcp:7100, adb pull <remote> <local>, adb push <local> <remote>, adb push foo.txt /sdcard/foo.txt

- **ant**
  - To compile and build your Android project into an installable .apk file.

- **Keytool**
  - To generate a keystore and private key, used to sign your .apk file. Keytool is part of the JDK.

- **Jarsigner (or similar signing tool)**
  - To sign your .apk file with a private key generated by Keytool. Jarsigner is part of the JDK.

Creating a project with cmd

1. Add the platform-tools/ and tools/ directories to your PATH environment variable
2. cd tool/
3. android list targets
4. android create project --target <target-id> --name MyFirstApp --path <path-to-workspace>/MyFirstApp --activity MainActivity --package com.example.myfirstapp

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>cd &lt;path-to-workspace&gt;/MyFirstApp</td>
</tr>
<tr>
<td>2.</td>
<td>ant debug, see ant.properties</td>
</tr>
<tr>
<td>3.</td>
<td>Optionally: Android avd,</td>
</tr>
<tr>
<td>4.</td>
<td>adb install bin/MyFirstApp-debug.apk</td>
</tr>
<tr>
<td>5.</td>
<td>android avd</td>
</tr>
</tbody>
</table>
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 2.0 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)
- **Telephony** (hardware dependent)
- **Connectivity** for NFC, Bluetooth, cellular, and WiFi (hardware dependent)
- **Peripherals** for Camera, GPS, compass, and accelerometer (hardware dependent)
- **Development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE
### Android API

<table>
<thead>
<tr>
<th>Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>android</td>
<td>Resource classes used by std app</td>
</tr>
<tr>
<td>android.app.*</td>
<td>Android app model</td>
</tr>
<tr>
<td>android.appwidget</td>
<td>Publish Embedded view</td>
</tr>
<tr>
<td>android.bluetooth</td>
<td>Manage bluetooth functionalities</td>
</tr>
<tr>
<td>android.content.*</td>
<td>Accessing/publishing data</td>
</tr>
<tr>
<td>android.database</td>
<td>Explore data via a content provider</td>
</tr>
<tr>
<td>android.database.sqlite</td>
<td>SQLite database management</td>
</tr>
<tr>
<td>android.gesture</td>
<td>Create/reorganize/load/save gestures</td>
</tr>
<tr>
<td>android.graphics.*</td>
<td>Low-level graphics tools</td>
</tr>
<tr>
<td>android.hardware</td>
<td>Support for hardware device</td>
</tr>
<tr>
<td>android.location</td>
<td>Location-based related services</td>
</tr>
<tr>
<td>android.media</td>
<td>Manage various media interface (A/V)</td>
</tr>
<tr>
<td>android.net.*</td>
<td>Help with network access</td>
</tr>
<tr>
<td>android.opengl</td>
<td>OpenGL utilities</td>
</tr>
<tr>
<td>android.os</td>
<td>Basic OS services</td>
</tr>
<tr>
<td>android.preference</td>
<td>Manage app/UI preferences</td>
</tr>
<tr>
<td>android.provider</td>
<td>Access content provider</td>
</tr>
<tr>
<td>android.telephony.*</td>
<td>Access basic phone information</td>
</tr>
<tr>
<td>android.test.*</td>
<td>Writing test cases and suites</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>android.text.*</td>
<td>Render or track or span on the screen text</td>
</tr>
<tr>
<td>android.util</td>
<td>Common utility methods (date/time)</td>
</tr>
<tr>
<td>android.view.*</td>
<td>Expose basic user interface</td>
</tr>
<tr>
<td>android.webkit</td>
<td>Tools for browsing the web</td>
</tr>
<tr>
<td>android.widget</td>
<td>UI elements to use on the app screen</td>
</tr>
<tr>
<td>android(bytecode)</td>
<td>Classes surrounding the Dalvik bytecode</td>
</tr>
<tr>
<td>java.io</td>
<td>IO facilities (streaming, file access)</td>
</tr>
<tr>
<td>java.lang.*</td>
<td>Core classes for Android environment</td>
</tr>
<tr>
<td>java.math</td>
<td>Arbitrary-precision integers and decimals</td>
</tr>
<tr>
<td>java.net</td>
<td>Networking related functionalities (sock)</td>
</tr>
<tr>
<td>java.nio.*</td>
<td>Buffers that help handling data</td>
</tr>
<tr>
<td>java.security.*</td>
<td>Security framework</td>
</tr>
<tr>
<td>java.sql</td>
<td>Compatibility interface for SQL-based DB</td>
</tr>
<tr>
<td>java.text</td>
<td>Uncouple app’s text from natural languages</td>
</tr>
<tr>
<td>java.util</td>
<td>Extensive set of utility classes</td>
</tr>
<tr>
<td>javax.*</td>
<td>Crypto, ssl, net, xml, sql,</td>
</tr>
<tr>
<td>org.apache.http.*</td>
<td>Core interface of HTTP components</td>
</tr>
<tr>
<td>org.json</td>
<td></td>
</tr>
<tr>
<td>org.w3c.dom</td>
<td></td>
</tr>
</tbody>
</table>
## Android API

<table>
<thead>
<tr>
<th>Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>android.accessibilityservice</td>
<td></td>
</tr>
<tr>
<td>android.animation</td>
<td>Animate objects properties</td>
</tr>
<tr>
<td>android.drm</td>
<td>Digital right manager</td>
</tr>
<tr>
<td>android.mpt</td>
<td>Media transfer protocol and picture transfer protocol (PTP)</td>
</tr>
<tr>
<td>android.nfc</td>
<td>NFC functionalities, read tag</td>
</tr>
<tr>
<td>android.renderscripts</td>
<td>Low level rendering and computational API, for math, graphics, and 3D</td>
</tr>
<tr>
<td>android.speech</td>
<td></td>
</tr>
<tr>
<td>dalvik.system</td>
<td>Utilities and system information class specific to dalvik</td>
</tr>
<tr>
<td>org.xml.sax</td>
<td>Core sax API</td>
</tr>
</tbody>
</table>
Android includes a set of core (native) applications
  - Email client, SMS program, calendar, maps, browser, contacts

Applications are written in Java and compiles into an android packages (archive file with .apk suffix)
  - Native lib + resources + dalvik

Each Android application lives in its own security sandbox (principle of least privilege)
  - Each application is assigned with a unique Linux user ID and permissions for file access
  - Each application runs its own Linux process
  - Each application/process has its own VM (isolate the applications)

Applications can share data and access system services
  - Assign same user ID and VM
  - Request a permission to access device data at the install time
Application Framework

- Underlying all applications is a set of services and systems
  - View System 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
  - Resource Manager providing access to non-code resources such as localized strings, graphics, and layout files
  - Notification Manager enables all applications to display custom alerts in the status bar
  - Activity Manager that manages the life cycle of applications and provides a common navigation backstack

- Any application can publish its capabilities and any other application may then make use of those capabilities (principle of reusability subject to security constraints)
Libraries

- **C/C++ libraries used by various components**
  - **System C Libs → Bionic under BSD**
    - Standard system C library for Android embedded Linux-based OS
  - **Media Framework** (based on PacketVideo's OpenCORE, open SL)
    - the libraries support playback and recording for audio and video formats, and static image files (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG)
  - **Surface Manager**
    - manages access to the display subsystem
    - seamlessly composites 2D/3D graphic layers from multiple applications
  - **LibWebCore (webkit)**
    - a modern web browser engine which powers both the Android browser and an embeddable web view
  - **SGL**
    - the underlying 2D graphics engine
  - **3D Libs (OpenGL ES 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

- **These capabilities are exposed to developers through application frameworks**
Example: Audio API in Android

Applications
- Recorder App
- Media Player App

Applications Framework
- Media Recorder
- Media Player
  - JNI

Libraries (user space)
- Media Recorder
- Media Player
  - JNI
- Audio Flinger
- Audio Hardware Interface
  - Proprietary Libs
  - ALSA
  - Other Audio Drivers
- ALSA Kernel Driver

Linux Kernel
Android Runtime

- Every Android application runs in its own process, with its own instance of the Dalvik (.dex) virtual machine
  - .dex = .class post-processed by “dx” utility
  - a device may run multiple VMs

- Davlik is a non-standard Java VM based on some of Java SE packages
  - Apache commons
  - HttpClient
  - Junit

- Android’s programs are written in Java, using Java-oriented IDEs
  - It just doesn’t compile the java code into java bytecode but instead Dalvik bytecode (.dex)
  - optimized for minimal footprint
    - Uncompressed .dex = .5 * uncompressed .jar
Core Libraries

- Core libraries basically inherited from java core libraries
  - java.io.*
  - java.util.*
  - Java.net.*
  - Java.math.*

- Relies on the Linux kernel for underlying functionality
  - Threading
  - Low-level memory management
Linux Kernel

- Android relies on Linux version 2.6.x/3.0.x for core system services
  - Does not include the full set of Linux utilities
  - New Android specific components have been added
    - Memory management, runtime power management
    - Alarm, “Android Shared Memory”
    - “Kernel Memory Killer”, Kernel Debugger, Logger

- Abstraction layer between the hardware and the rest of the software stack

- Linux manages variety of services
  - Security
  - Memory management
  - Process management
  - Networking
  - Handing the physical hardware: drivers for a variety of devices
Memory management

- Asynchronous Shared Memory (ASHMEM)
  - Used to provide shared memory
  - Advantage: can be freed by kernel

- Physical Memory (PMEM)
  - Allows allocating contiguous memory to drivers and memories

- Low Memory Killer
  - Terminated processes with low importance
Android Internals: Linux Kernel

- IPC Binder – based on lightweight OpenBinder
  - Message passing between processes and threads using shared memory
- Logger – main, system, event and radio
- Alarm – timer to wake the device up from sleep
- RAM_CONSOLE – saves console output to reserved RAM area
- Timed output/gpio
- USB driver for adb
Three main mount points:

- system
- sdcard: for large files, everyone can access it
- apps: dedicated sandbox in /data/data/
Android Internals: Platform Initialization

A: Bootloader

B: daemons for handling low level Hw interfaces (usb, adb, debugger) & Dalvik VM process (Zygote)

C: runtime process to start Dalvik for running system server

D: start process to handle graphics and audio

E: Start all the other Android components
Android Internals: Platform Initialization

Source: marakana
Android Internals: Runtime Overview

System Startup

Daemon Processes

Runtime

Zygote

Zygote

App

New app is launched in its own process
Android Internals: Layer Interaction

App- Runtime service

App – runtime native service
Android Internals: Java Native Interface (JNI)

- Enables Java code running in a JVM to call, and to be called by native applications and libraries written in other languages such as C, C++ and assembly

Call gate for other languages
Android Application

- Android application is a collection of components, through which the system can enter your application
  - Activities
  - Services
  - Content Providers
  - Broadcast Receiver
Activities

- An object that has a life cycle and does some work
  - Entire, visible, and foreground lifetime

- Activity provides a screen for user interaction (entry point)
  - Displaying a UI to the user
  - Floating window, or just return value

- An application may consist of multiple activities
  - Back stack is LIFO

- Adding Activity
  - The new Java class must extend the framework “Activity” class
  - Created “Activity” must be defined into the application’s Manifest

```xml
<manifest ...
  <application ...
    <activity android:name=".ExampleActivity" />
  </application ...
</manifest >
```
Activity Lifecycle: revisited with demo

- Does not crash if the user receives a phone call or switches to another app while using your app.
- Does not consume valuable system resources when the user is not actively using it.
- Does not lose the user's progress if they leave your app and return to it at a later time.
- Does not crash or lose the user's progress when the screen rotates between landscape and portrait orientation.
Activities: Task and Backstack

- Each application contains multiple activities
  - Activity = user actions

- Task is a collection of activities that user interact with when performing a certain job
  - Starting point: home screen
  - Operates as a last in, first out (LIFO)
Services

- A service is a component without UI in the background and can call activities
  - It can run in its own process, or in the context of another application's process
  - Other components "bind" to a Service and invoke methods on it via remote procedure calls
- Example: media player
- Invoking a service (similar to Activity)
  - Extend the Framework service class
  - Define the new service in application’s manifest

```xml
<manifest ... >
  <application ... >
    <service android:name=".ExampleService" />
  </application>
</manifest>
```
Content Provider

- Manage access to data on the device and share data between applications
  - File system, SQLite database, any other persistent storage location

- Share data between applications
  - Connects data in one process with code running in another process

- Example: list of contacts, user dictionary

- Interact with content provider through content resolver providing the basic CRUD (for Security reasons)
  - getContentResolver().query(uri, projection, selection, selectionsArgs, sortOrder)

- A “Content Providers” exposes a unique URI to identify data in a provider
  - scheme://path/resources, e.g. content://fr.example.eurecom/cm/faculty/10
  - The scheme part is standard and fully-qualified to ensure its uniqueness
  - The path to determine what kind of data is being requested
  - A specific record being requested, if any

- Request permission
  - e.g. android.permission.READ_USER_DICTIONARY
Content Providers

- **content://media/internal/images**
  - URI string that would return a list of all the internal images on the device

- **content://media/external/images**
  - URI string that would return a list of all the images on the "primary" external storage (e.g., the SD card).

- **content://contacts/people/**
  - URI that would return a list of all contact names on the device.

- **content://contacts/people/23**
  - URI string that would return a single result row, the contact with ID = 23.
Broadcast Receivers

- A component that responds to system-wide broadcast (event handler) initiated by the system or an application
  - Receive intents sent by sendBroadcast() through dynamic or static registration,
  - No UI, but could create a status bar notification to alert user
  - Example: screen has turned off, battery low, data downloaded

- Application can start another application’s component through Android System
  - intent to system $\rightarrow$ system activate the component (access restriction)
  - Multiple entry points to start an application

- How to receive these specific “Intents”?  
  - extend the framework “BroadcastReceiver” class
  - implement “onReceive()” method
  - Dynamic registration: register filters dynamically by calling Context.registerReceiver()
  - Static registration: describe “receiver” element into the application’s Manifest

```xml
<receiver android:name=".Alerter">
  <intent-filter>
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
  </intent-filter>
</receiver>
```
How to Activate components?

- Activities, services, and broadcast receivers are activated by an asynchronous message
  - Intent: bind individual components to each other at run time
  - Explicit intent and implicit intent

- The content provider, on the other hand, are not activated by intents but rather by the content resolver
  - Security concerns
Intents

- Simple message object that represents an "intention" to do something
  - can also be used to broadcast system events and notifications

- It is a passive data structure holding an abstract description of an operation to be performed — or

- Often in the case of broadcasts, a description of something that has happened and is being announced

Example
- to display a web page, send an "Intent" to view the URI
- the most suitable application/component (the browser) is called to handle the request
- If multiple application/component can perform the action described by the intent, then user selects

An “intent” is a bundle of information describing the action or the service
- “Name” represents the name of the component that should handle the intent
- “action” attribute is typically a verb (VIEW, EDIT, DIAL, MAIN, ...)
- “data” to operate on is expressed in the form of an Universal Resource Identifier (URI)
- “category” attribute gives additional information about the action to execute
- “extras” Key-value pairs for additional information
- “flag” how to launch an activity and how the app should be treated
Intent Resolution and Intent Filters

- **Explicit intents**: designate the target component by its name
- **Implicit intents**: do not name a target
- **Intent filters** declares which implicit intents they can handle
  -Expose components capabilities used to decide if the components can respond to the received intent or not
  -Delimit the intents the application can handle
- **Action**: the standard MAIN action is a main entry point
- **Category**: the LAUNCHER category says that this entry point should be listed in the application launcher

```
<intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
    <data android:mimeType="audio/mpeg" android:scheme="http" ... />
</intent-filter>
```
# Intent filters

<table>
<thead>
<tr>
<th>Constant</th>
<th>Target component</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION_CALL</td>
<td>activity</td>
<td>Initiate a phone call.</td>
</tr>
<tr>
<td>ACTION_EDIT</td>
<td>activity</td>
<td>Display data for the user to edit.</td>
</tr>
<tr>
<td>ACTION_MAIN</td>
<td>activity</td>
<td>Start up as the initial activity of a task, with no data input and no returned output.</td>
</tr>
<tr>
<td>ACTION_SYNC</td>
<td>activity</td>
<td>Synchronize data on a server with data on the mobile device.</td>
</tr>
<tr>
<td>ACTION_BATTERY_LOW</td>
<td>broadcast receiver</td>
<td>A warning that the battery is low.</td>
</tr>
<tr>
<td>ACTION_HEADSET_PLUG</td>
<td>broadcast receiver</td>
<td>A headset has been plugged into the device, or unplugged from it.</td>
</tr>
<tr>
<td>ACTION_SCREEN_ON</td>
<td>broadcast receiver</td>
<td>The screen has been turned on.</td>
</tr>
<tr>
<td>ACTION_TIMEZONE_CHANGED</td>
<td>broadcast receiver</td>
<td>The setting for the time zone has changed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constant</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORY_BROWSABLE</td>
<td>The target activity can be safely invoked by the browser to display data referenced by a link — for example, an image or an e-mail message.</td>
</tr>
<tr>
<td>CATEGORY_GADGET</td>
<td>The activity can be embedded inside of another activity that hosts gadgets.</td>
</tr>
<tr>
<td>CATEGORY_HOME</td>
<td>The activity displays the home screen, the first screen the user sees when the device is turned on or when the Home button is pressed.</td>
</tr>
<tr>
<td>CATEGORY_LAUNCHER</td>
<td>The activity can be the initial activity of a task and is listed in the top-level application launcher.</td>
</tr>
<tr>
<td>CATEGORY_PREFERENCE</td>
<td>The target activity is a preference panel.</td>
</tr>
</tbody>
</table>
AndroidManifest.xml

- Declare the information about application to the Android system
  - Name of java package, components (the activities, services, broadcast receivers, and content providers), processes, permissions, API level, and HW/SW features

- Identify user permissions required by the application
- API Libs the applications needs to be linked against
- Market filter based on the application compatibility with the device

```
<?xml version="1.0" encoding="utf-8"?>
<manifest

xmlns:android="http://schemas.android.com/apk/res/android"

package="com.my_domain.app.helloactivity">

  <application android:label="@string/app_name">
    <activity android:name=".HelloActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>

  <uses-sdk android:minSdkVersion="10"/>

</manifest>
```
<xml version="1.0" encoding="utf-8"/>

<manifest>

<uses-permission />
<permission />
<permission-tree />
<permission-group />
<instrumentation />
<uses-sdk />
<uses-configuration />
<uses-feature />
<supports-screens />
<compatible-screens />
<supports-gl-texture />

<application>

<activity>
  <intent-filter>
    <action />
    <category />
    <data />
  </intent-filter>
  <meta-data />
</activity>

</application>

<activity-alias>
  <intent-filter> . . . </intent-filter>
  <meta-data />
</activity-alias>

<service>
  <intent-filter> . . . </intent-filter>
  <meta-data/>
</service>

<receiver>
  <intent-filter> . . . </intent-filter>
  <meta-data />
</receiver>

<provider>
  <grant-uri-permission />
  <meta-data />
</provider>

<uses-library />

</manifest>
Application Resources

- Each resources are identified by a unique integer ID
  - Used to reference the resource from the code and possibly use it in UI
  - E.g. eurecom_logo.png in res/drawable, the resource ID named R.drawable.eurecom_logo

- Alternative resources for different device configuration
  - E.g. language, layout
User Interface

- UI is defined as a hierarchy of View (subclass of Widget) and ViewGroup Object (subclass of layout)
  - Declare UI in XML layout file (most common way)
  - Instantiate UI element in runtime (Expressed directly in the code using addView(View))

- Apply Style and themes

- Check out the Hello View tutorial

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />

    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />

</LinearLayout>
```
Views

- An object that knows how to draw itself on the screen
  - Standard applications may use existing tree of views
  - Rich graphical applications, such as games, may require a custom view

- Load
  - `setContentView(R.layout.main_layout)` on `onCreate` callback

- “id: of a view: a unique integer ID associated with the view in `R.Java` file
  ```xml
  <Button id="@+id/my_button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/my_button_text"/>
  ```

- Creating an instance of the view
  ```java
  Button myButton = (Button) findViewById(R.id.my_button);
  ```
Common Layouts

- Linear layout
  - List View
- Relative layout
  - Grid View
- Web view
  ```html
  <!-- web page -->
  </html>
  ```
Example Layouts

Hello LinearLayout
- red
- green
- blue
- yellow

Hello RelativeLayout
- row one
- row two
- row three
- row four

Hello TableLayout
- Open...
- Save...
- Save As...
- X Import...
- X Export...
- Quit

Hello GridView
- This is the Artists tab

Hello ListView
- American Samoa
- El Salvador
- Saint Helena
- Saint Kitts and Nevis
- Saint Lucia
- Saint Pierre and Miquelon
- Saint Vincent and the Grenadines
- Samoa
- San Marino
- Saudi Arabia
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label"/>
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancel" />
</RelativeLayout>
Example Widgets and Other View
Optimizing your UI

- Hierarchy view
  - Display, debug, and help to optimize the View objects

- lint (static code scanning tool):
  - analyze the XML files that define your application's UI to find inefficiencies in the view hierarchy

## Summary of Important Layouts

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrameLayout</td>
<td>Layout that acts as a view frame to display a single object.</td>
</tr>
<tr>
<td>Gallery</td>
<td>A horizontal scrolling display of images, from a bound list.</td>
</tr>
<tr>
<td>GridView</td>
<td>Displays a scrolling grid of m columns and n rows.</td>
</tr>
<tr>
<td>LinearLayout</td>
<td>A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.</td>
</tr>
<tr>
<td>ListView</td>
<td>Displays a scrolling single column list.</td>
</tr>
<tr>
<td>RelativeLayout</td>
<td>Enables you to specify the location of child objects relative to each other</td>
</tr>
<tr>
<td>ScrollView</td>
<td>A vertically scrolling column of elements.</td>
</tr>
<tr>
<td>Spinner</td>
<td>Displays a single item at a time from a bound list, inside a one-row textbox.</td>
</tr>
<tr>
<td>SurfaceView</td>
<td>Provides direct access to a dedicated drawing surface.</td>
</tr>
<tr>
<td>TabHost</td>
<td>Provides a tab selection list that monitors clicks and enables change the screen</td>
</tr>
<tr>
<td>TableLayout</td>
<td>A tabular layout with an arbitrary number of rows and columns,</td>
</tr>
<tr>
<td>ViewFlipper</td>
<td>A list that displays one item at a time, inside a one-row textbox.</td>
</tr>
<tr>
<td>ViewSwitcher</td>
<td>Same as ViewFlipper.</td>
</tr>
</tbody>
</table>
**Dialog**

- Small window in front of current activity
  - Used for notifications that require user interruption
  - Underlying activity loses focus and dialog accepts user interaction

- AlertDialog
- ProgressDialog
- DatapickerDialog
- TimepickerDialog
Notification

- Notify user about an event that occurs in your app
  - Toast notification: small message from background
  - Status bar notification: requesting user attention
  - Dialog notification: user must wait until the end of the activity

Example
- SMS
- Voicemail or voice call
- Applications can create their own

Mechanism for alerting the user of something that needs their attention
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
Android Development Flow

Build Process

- The yellow rounded rectangles are Android SDK tools
- The orange boxes are files created by the programmer
- The remaining white boxes are generated by the tools

Source Code

Java Byte Code

.dex byte code

.apk Package

- The yellow rounded rectangles are Android SDK tools
- The orange boxes are files created by the programmer
- The remaining white boxes are generated by the tools
Android Development Flow

Build Process

Android Project → Compilation and Packaging → Android Package (.apk)
  - .dex files
  - resources
  - .arsc
  - uncompiled resources
  - AndroidManifest.xml

Signing → ADB → Device or Emulator

ADB

Application Resources
- aspt
- R.java
- Application Source Code
- Java Interfaces
- Java Compiler
- .class Files
- dsex
- dex files
- Compiled Resources
- apckbuilder
- Android Package (.apk)
- Jarsigner
- Signed .apk
- zipalign (release mode)
- Signed and Aligned .apk

.aidl Files

3rd Party Libraries and .class Files

Other Resources

Debug or Release Keystore

©Navid Nikaein 2011
Hello, Android
Developement Environment

- Eclipse IDE version 4.3 Kepler
- JDK 5, 6, 7
- Android SDK 4.3
- Tools
  - ADT (Android Development Tools)
  - Android Emulator,
  - AVD (Android Virtual Device)
  - Hierarch viewer
  - Layoutopt
  - Draw 9-patch
  - DDMS (Davlik Debug Monitor Service)
  - ADB, AIDL, Logcat, TraceView, mksdcard, sqlite, dx, Application exerciser monkey, zipalign
Java and eclipse

- The language is based in Java SM
- Objected oriented language
  - An infinity of documentation, examples, tutorials, libraries, etc...
- Eclipse is one IDE to program
  - Independent of Java
  - Free and open source
- 3rd party development tool
  - IntelliJ IDEA (http://www.jetbrains.com/idea/)
To Start

- Open Eclipse
  - File>New>Project
    - Android application project
  - Set Project name, target, application and package name
  - You can finish the project
Create a new device

- To run your widget you need a device

You can set many different aspects of the device, including the version of the API in use.
Java Programming (1)

- Packages are used to organize the classes
  - In java it is represented into distinct directories
- The basic unit are classes
- The instantiation of a class is called object

```java
package android.tests;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class AndroidTests extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView view=new TextView(this);
        view.setText("My first test");
        setContentView(view);
    }
}
```
Public methods/variables may be accessed from any other class

Private, protected and package not
- Package: any class from the same package has access
- Protected: is specific of the class, but the subclasses, children, have access
- Private: not even the subclass classes may access

Inheritance is an important concept, enables reuse
EURECOM MEMBERS

Academia

Aalto University
School of Science and Technology

EPFL
École Polytechnique Fédérale de Lausanne

Technische Universität München

NTNU
Norwegian University of Science and Technology

Industry

Swisscom

Thales

SFR

Orange

ST Ericsson

BMW Group Forschung und Technik

Cisco

Symantec

SAP