Chapter 2:
Getting Started with Android Development

Objectives

- Understand Android and its key features
- Understand Android architecture
- Understand the fundamentals of Android app development
  - The core components that define Android application
  - The manifest file to declare components and required device features
  - Resources that are separate from the application code
- Understand the anatomy of an Android app
- Setup and configuration of Android development environment

What is Android?

- Mobile Software Stack
  - operating system, middleware and key applications, tools to allow developers to quickly create applications.
- Built on the open Linux Kernel (open source mobile OS based on modified version of Linux)
- Open Source Apache License
- Open Handset Alliance (OHA)
  - 60+ companies from all over the world
  - biggest Telecom vendors, handset makers, and component manufacturers
- Free to license: No cost to developers to get tools, or publish apps

Android

- Vendors / hardware manufacturers can add proprietary extensions
- Unified approach to app development – for different devices
- Competitors:
  - Apple's iOS, Windows Mobile / Phone, OS' of phone manufacturers such as Mimo, Symbian etc.

Why Android is special?

- A truly **open**, **free** development platform based on **Linux** and open source
- A component-based architecture inspired by Internet mashups
- Tons of built-in services out of the box
- Automatic management of the application life cycle
- High-quality graphics and sound
- Portability across a wide range of current and future hardware

Source: Hello, Android

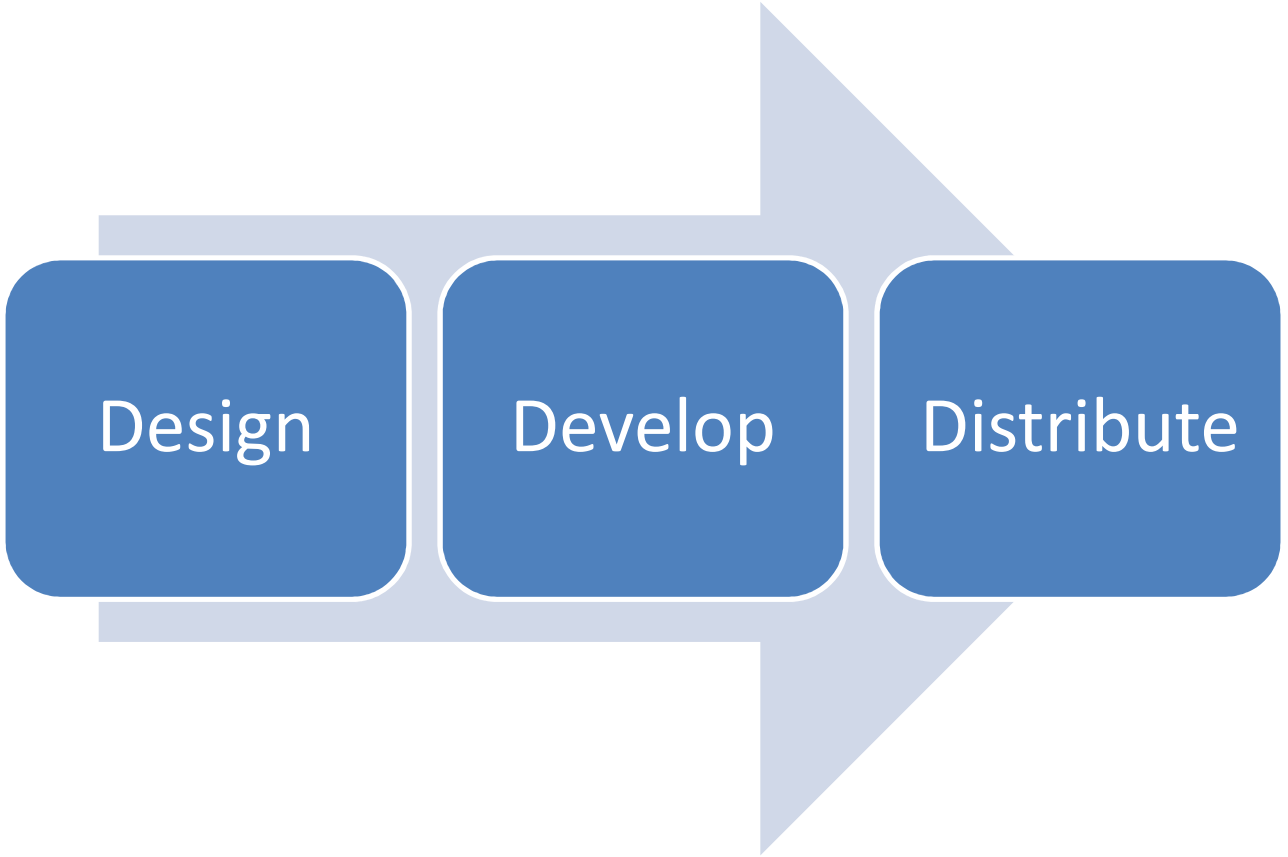
Features

- **Data Storage** –SQLite
- **Connectivity** – support GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), Wi-Fi, LTE, and WiMAX
- **Messaging** – support both SMS and MMS
- **Web browser** – based on the open source WebKit + Chrome's V8 Javascript engine
- **Media support** – H.263, H.264 (in 3GP or MP4 container), MPEG-4, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF and BMP

Features

- **Hardware support** – Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, GPS
- **Multi-touch screens**
- **Multi-tasking applications**
- **Flash support** – Android 2.3 supports Flash 10.1
- **Tethering** – support sharing of Internet connections as a wired/wireless hotspot

Android Apps Development Cycle



- Source: http://developer.android.com/

Design

Android Design Principles

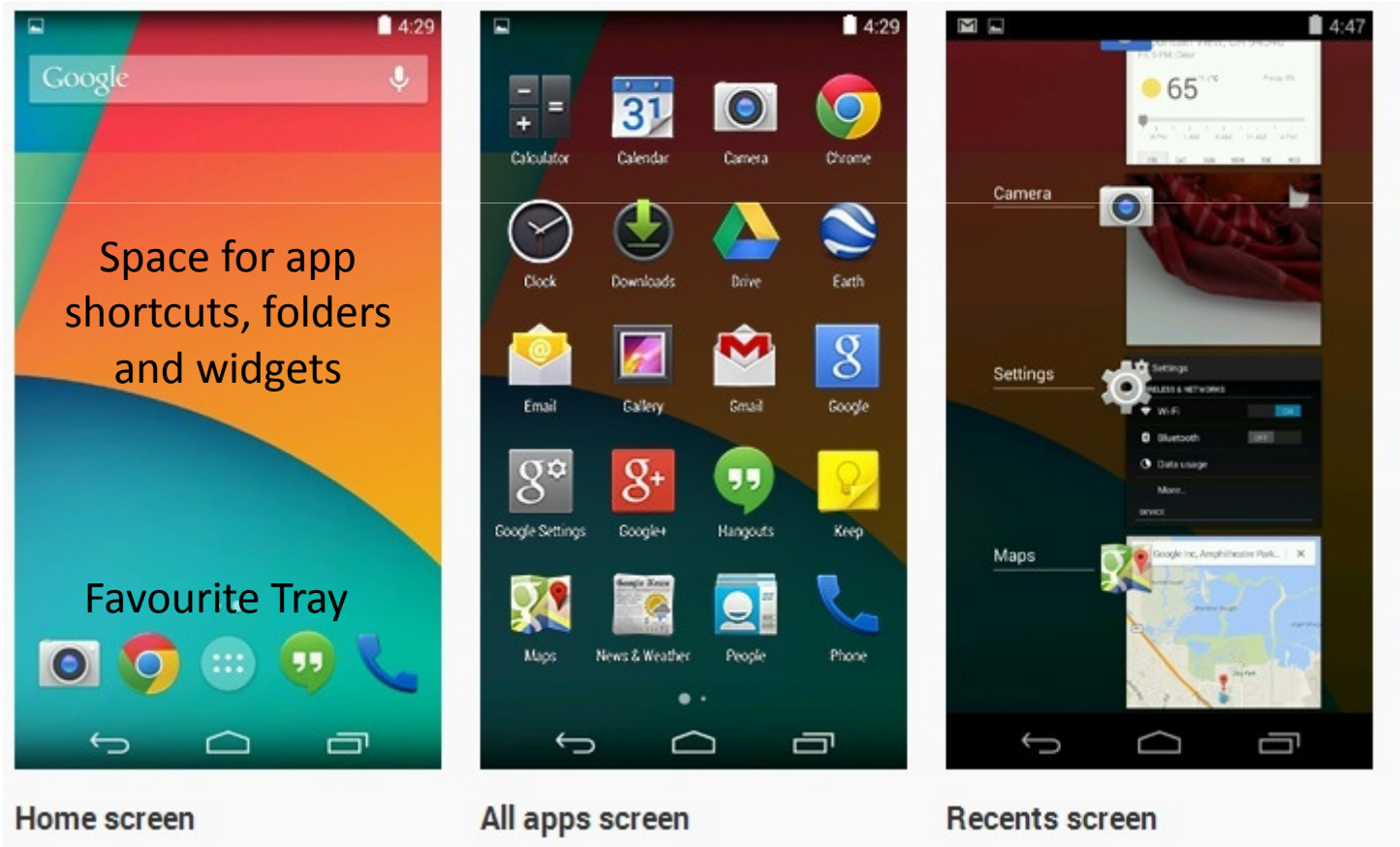
- Keep users' best interests in mind. Consider them as you apply your own creativity and design thinking. Deviate with purpose.

1. **Enchant** Me
2. **Simplify** My Life
3. Make Me **Amazing**

- http://developer.android.com/design/get-started/principles.html

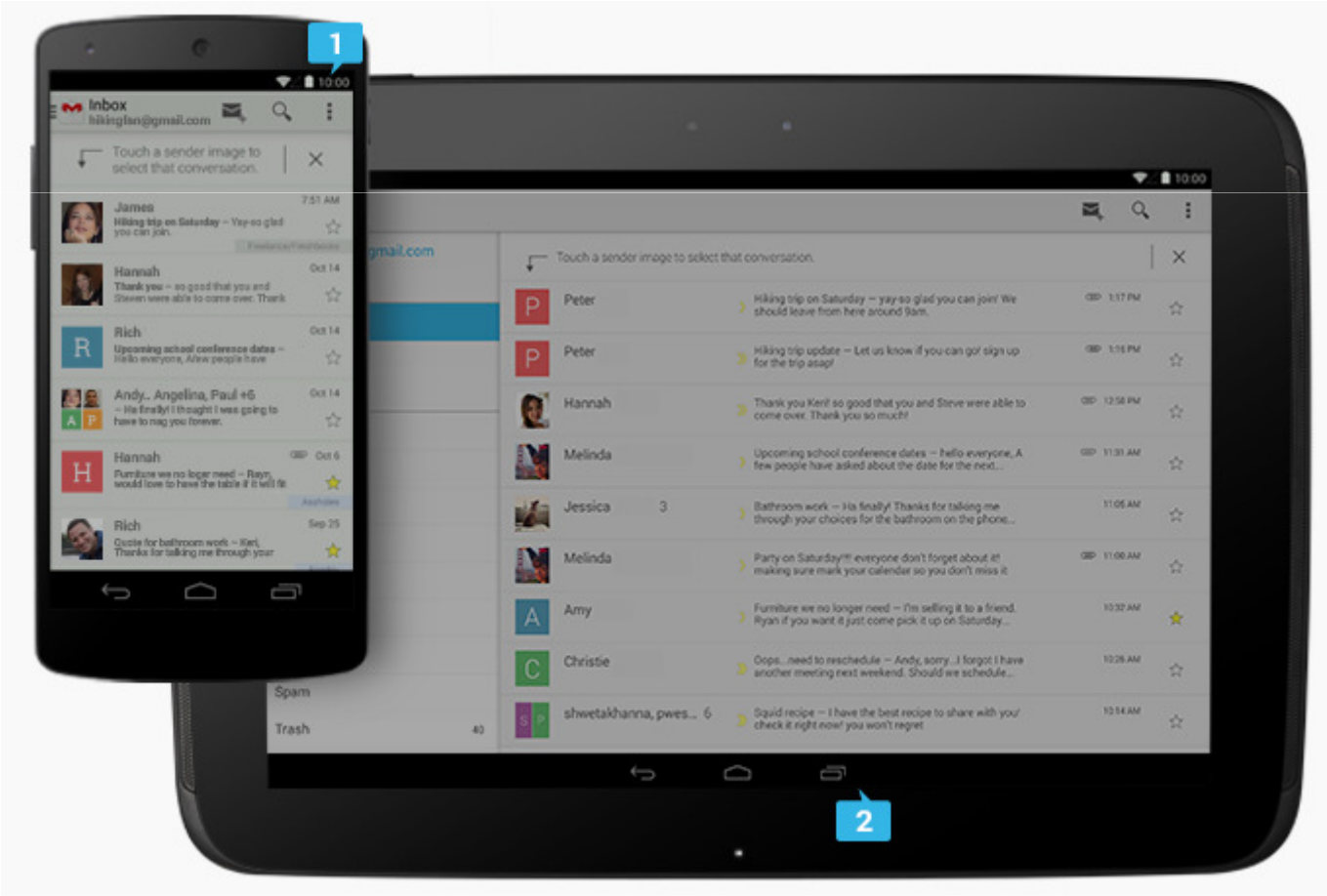
UI Overview

- Android's system UI provides the framework on top of which you build your app.
- http://developer.android.com/design/get-started/ui-overview.html



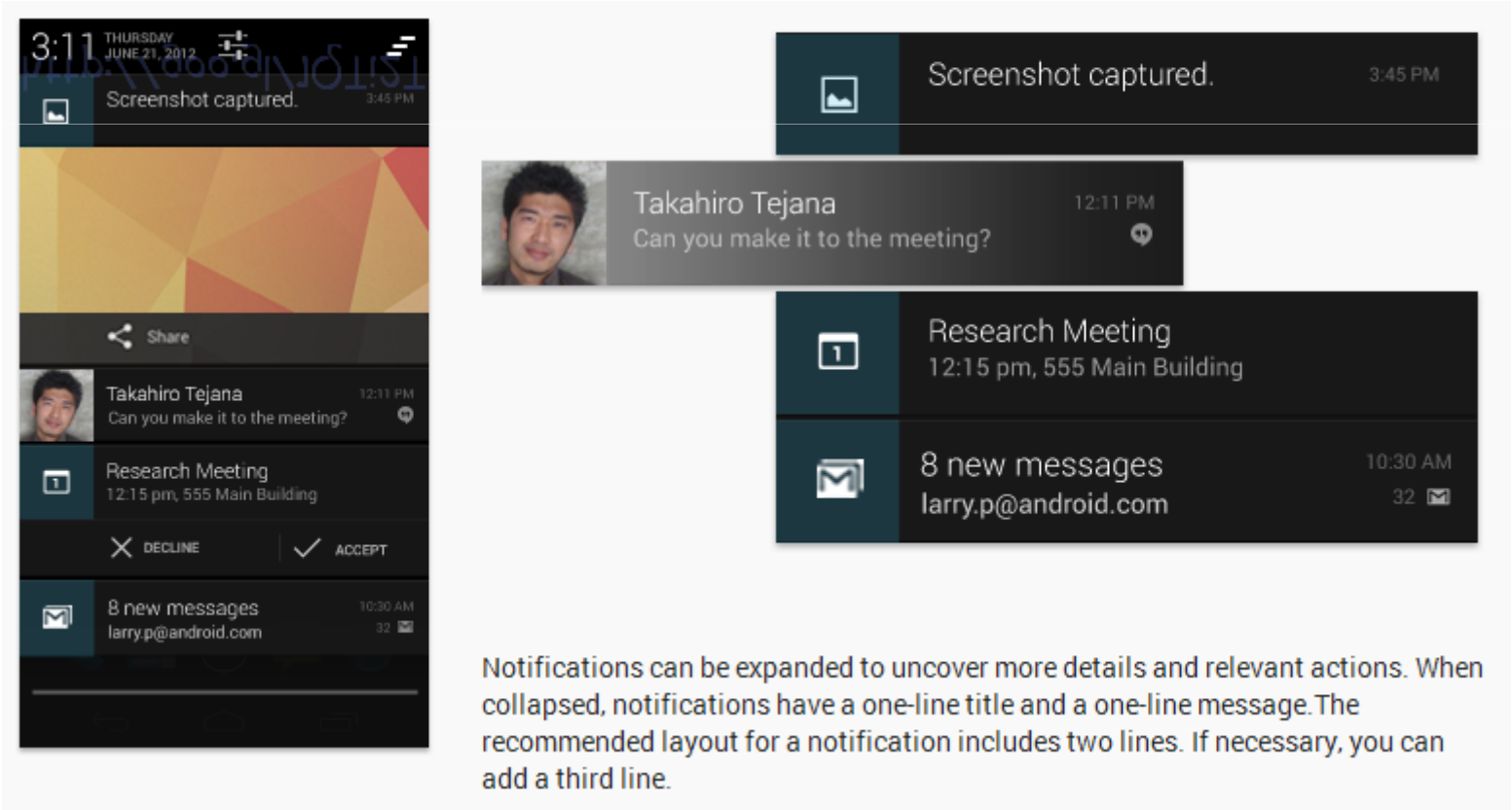
System Bar

1. Status Bar
2. Navigation Bar



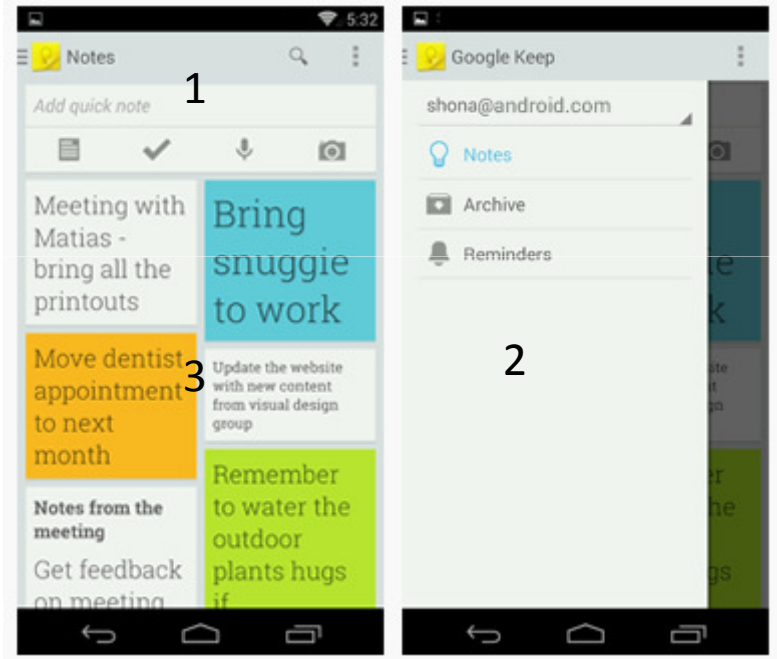
Notification

- Notifications are brief messages that users can access at any time from the status bar.



Notifications can be expanded to uncover more details and relevant actions. When collapsed, notifications have a one-line title and a one-line message. The recommended layout for a notification includes two lines. If necessary, you can add a third line.

Common App UI



1. Action Bar (http://developer.android.com/design/patterns/actionbar.html)
2. Navigation Drawer (http://developer.android.com/design/patterns/navigation-drawer.html)
3. Content Area

Style

- Metrics and Grids
- Devices and displays
- Theme
- Color
- Palette
- Typography
- Iconography
- Writing Style
- Consistency has its place in Android, but you also have the flexibility to customize the look of your app to reinforce your brand.
- http://developer.android.com/design/style/index.html

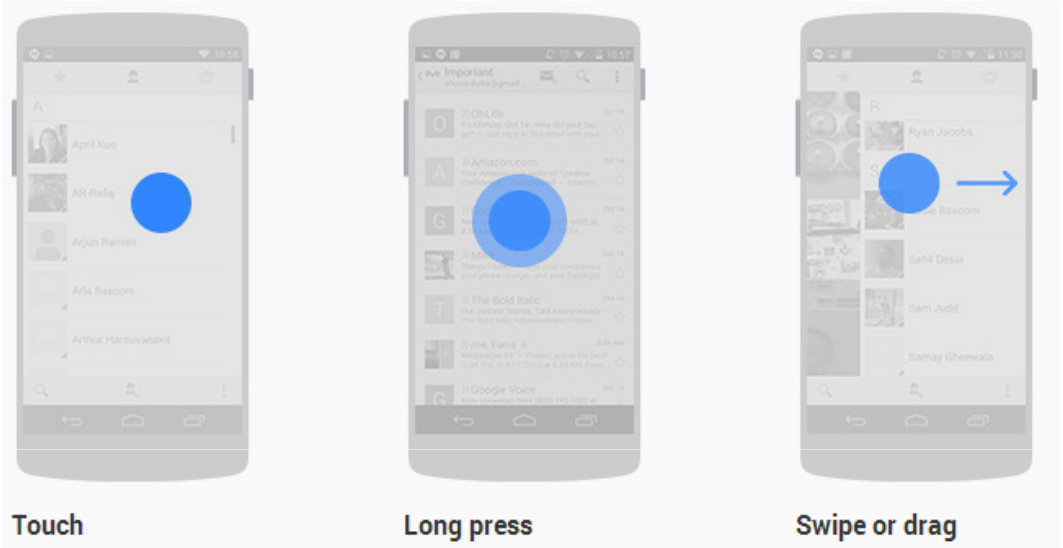
Patterns

- Different platforms play by different rules and conventions. Design decisions that make perfect sense on one platform will look and feel misplaced in the context of a different platform.

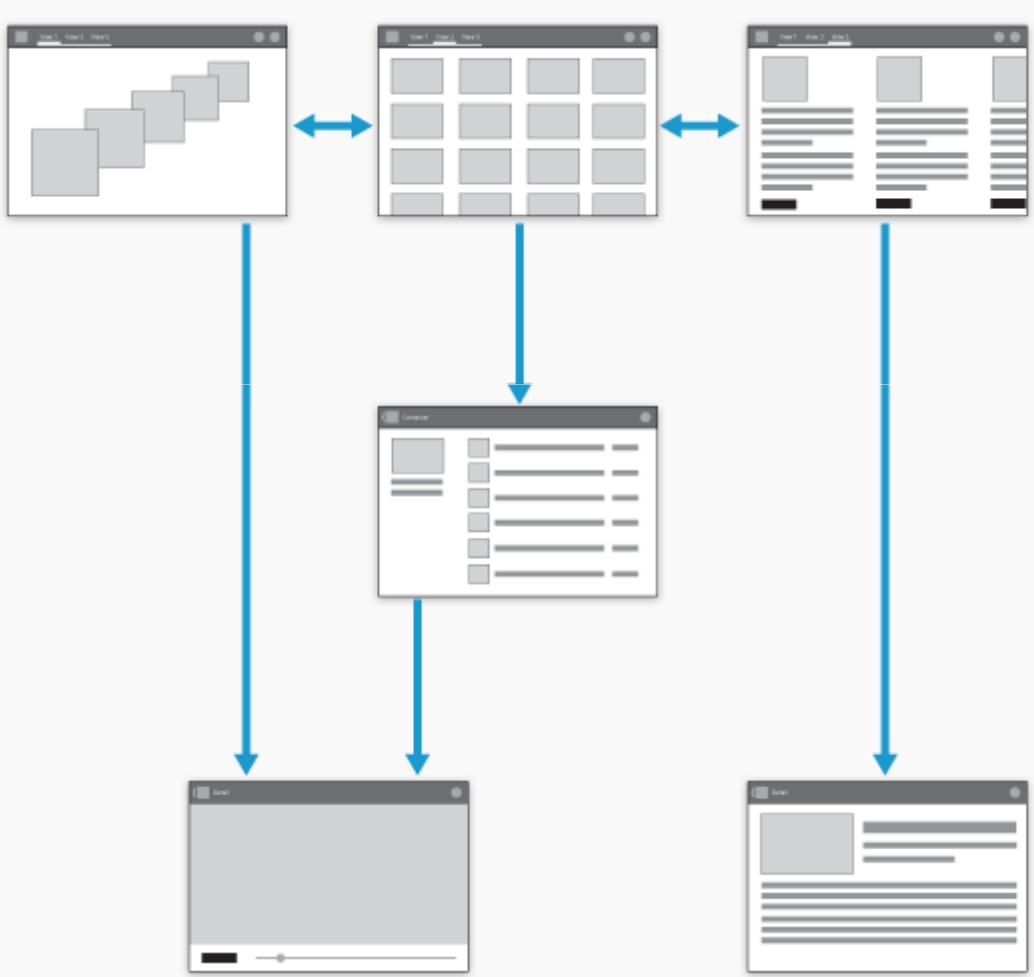
- http://developer.android.com/design/patterns/index.html

- Gestures
- App structure
- Navigation
- Action Bar
- Navigation drawer
- Multi-pane layouts
- Swipe views
- Full screen
- Selection
- Confirming & Acknowledging
- Notifications
- Widgets
- Settings
- Help
- Backward Compatibility
- Accessibility

Gestures


Touch
Long press
Swipe or drag
Long press drag
Double touch
Double touch drag

App structure



Top level views

The top level of the app typically consists of the different views that your app supports. The views either show different representations of the same data or expose an altogether different functional facet of your app.

Category views

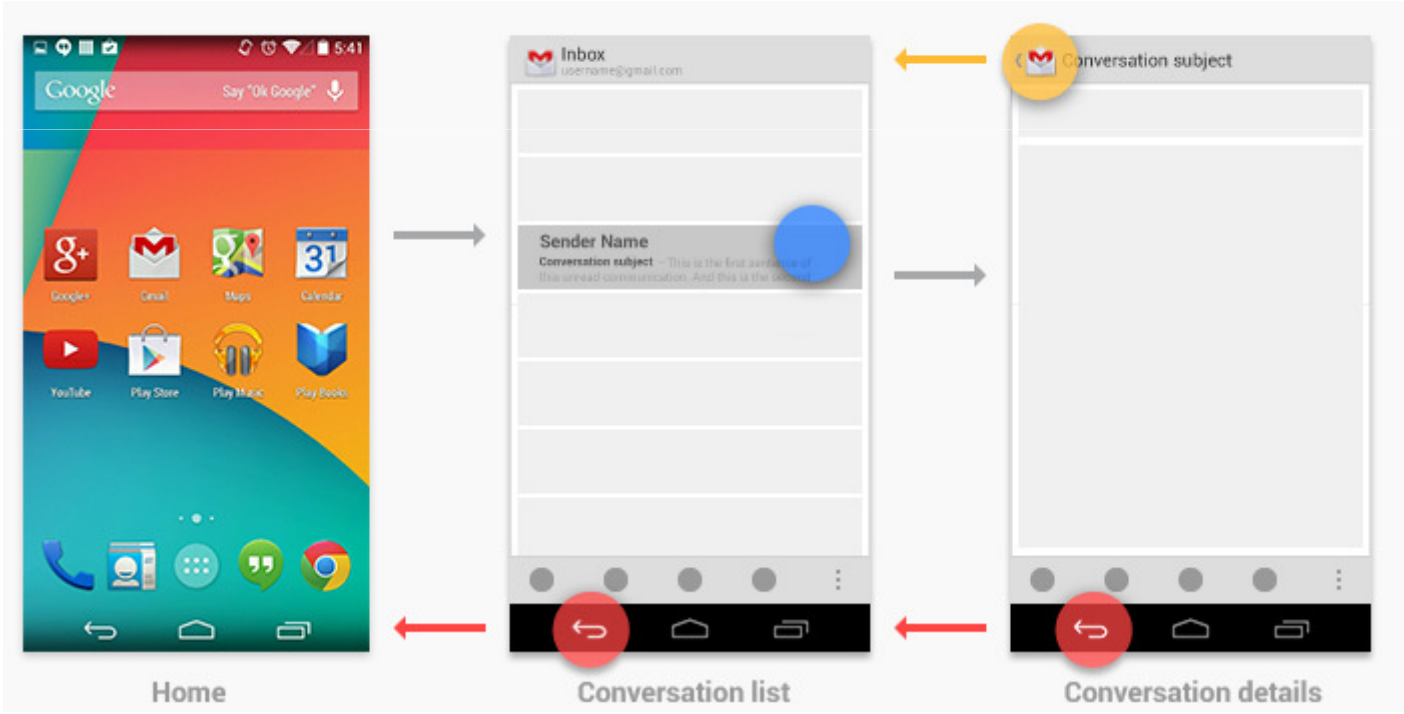
Category views allow you to drill deeper into your data.

Detail/edit view

The detail/edit view is where you consume or create data.

Navigation

- Back vs. Up



Action Bar

1. **App icon**

   The app icon establishes your app's identity. It can be replaced with a different logo or branding if you wish. Important: If the app is currently not displaying the top-level screen, be sure to display the Up caret to the left of the app icon, so the user can navigate up the hierarchy. For more discussion of Up navigation, see the Navigation pattern.

App icon with and without "up" affordance.

2. **View control**

   If your app displays data in different views, this segment of the action bar allows users to switch views. Examples of view-switching controls are drop-down menus or tab controls. For more information on view-switching, see the App Structure pattern.

   If your app doesn't support different views, you can also use this space to display non-interactive content, such as an app title or longer branding information.

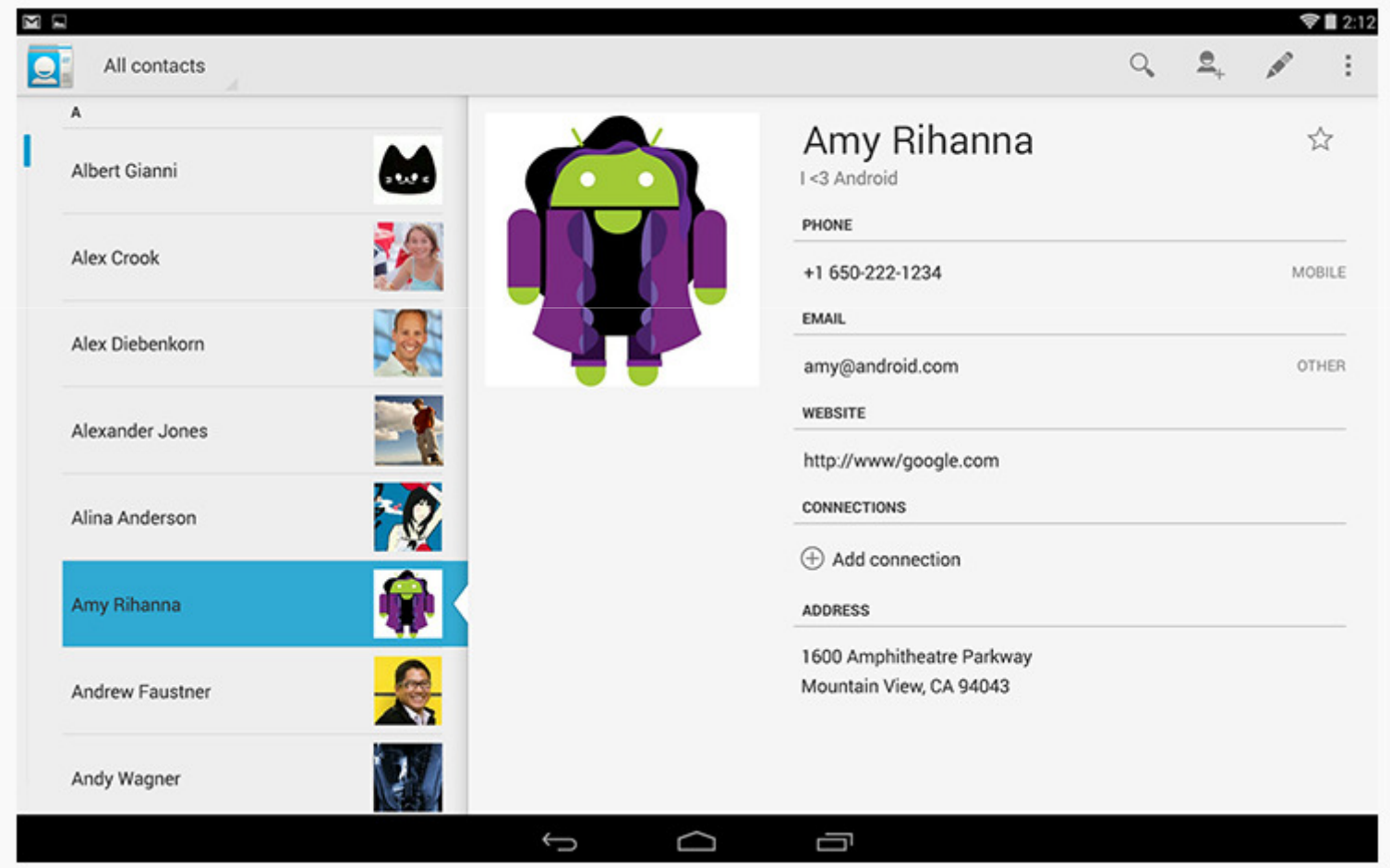
3. **Action buttons**

   Show the most important actions of your app in the actions section. Actions that don't fit in the action bar are moved automatically to the action overflow. Long-press on an icon to view the action's name.

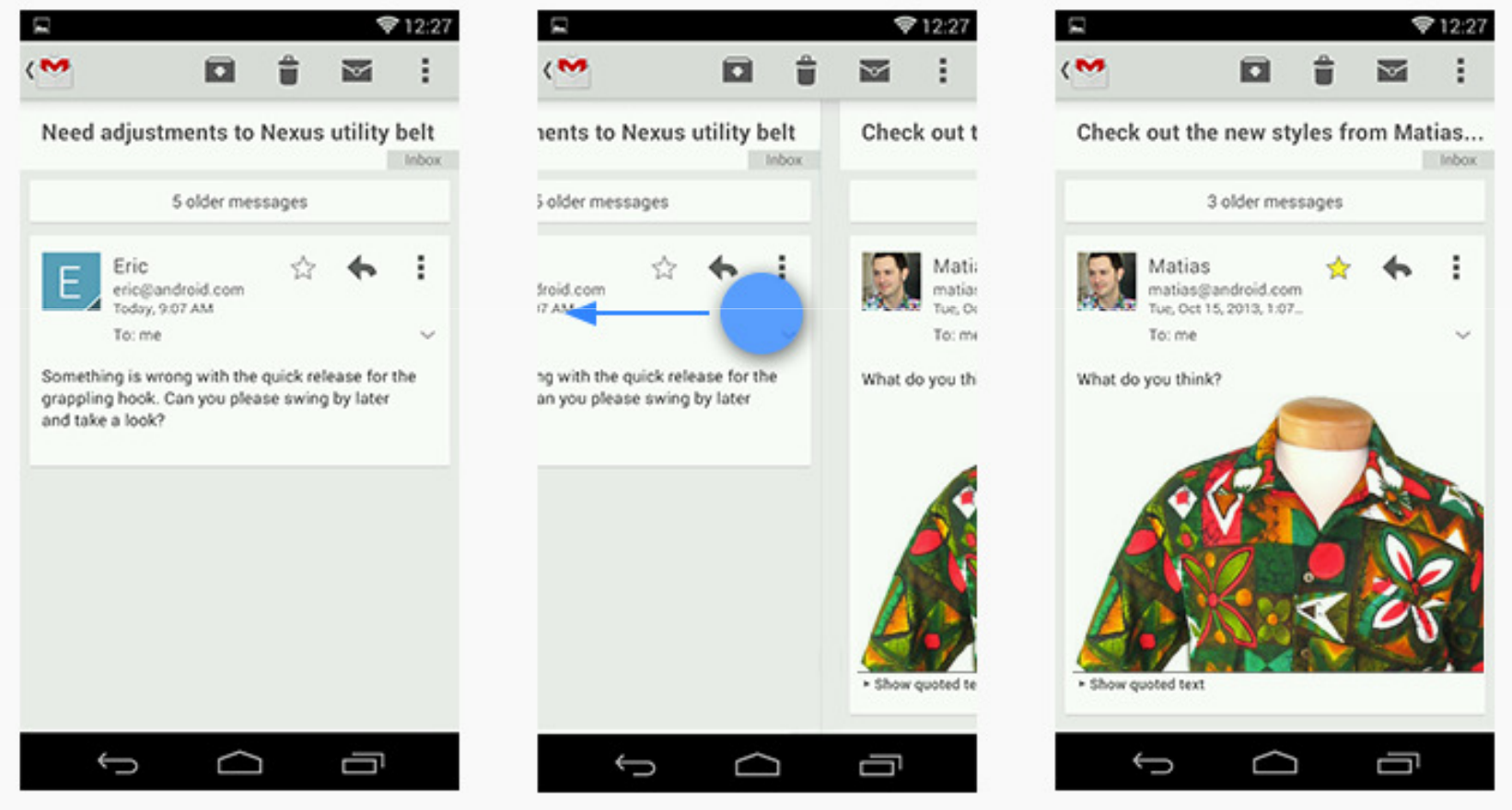
4. **Action overflow**

   Move less often used actions to the action overflow.

Multi-pane Layouts



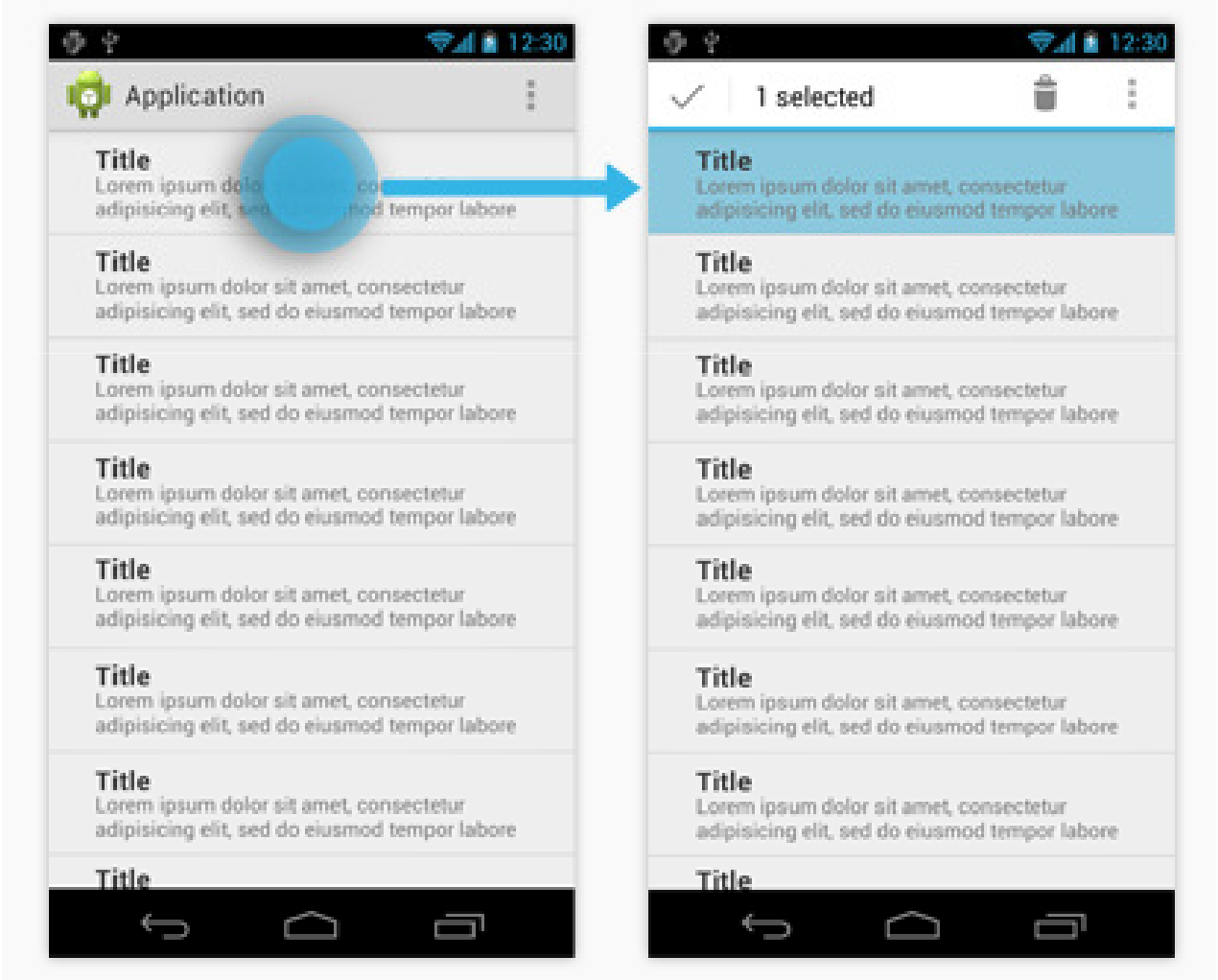
Swipe views


12:27
Need adjustments to Nexus utility belt
Inbox
5 older messages
Eric
eric@android.com
Today, 9:07 AM
To: me
Something is wrong with the quick release for the grappling hook. Can you please swing by later and take a look?
12:27
Check out t
older messages
What do you th
Show quoted te
12:27
Check out the new styles from Matias...
Inbox
3 older messages
Matias
matias@android.com
Tue, Oct 15, 2013, 1:07...
To: me
What do you think?
Show quoted text

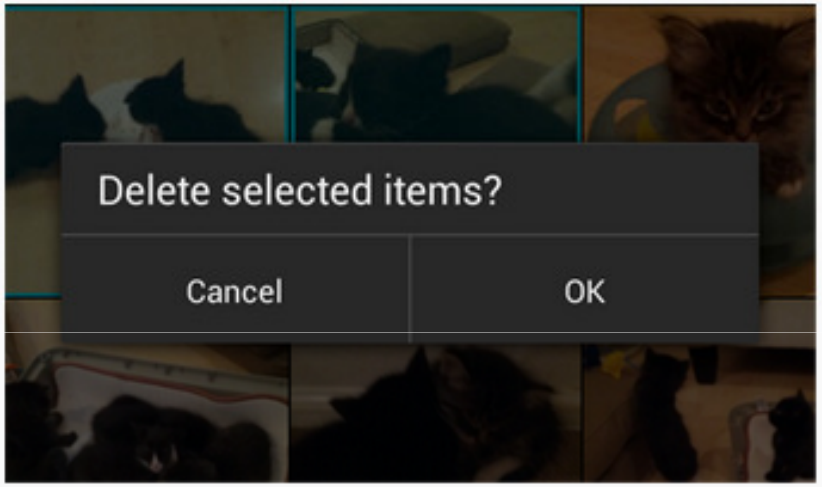
Full screen


said Alice.
with her head!" the Queen
at the top of her voice. Nobody
Who cares for you!" said Alice (she
grown to her full size by this time).
nothing but a pack of cards!"

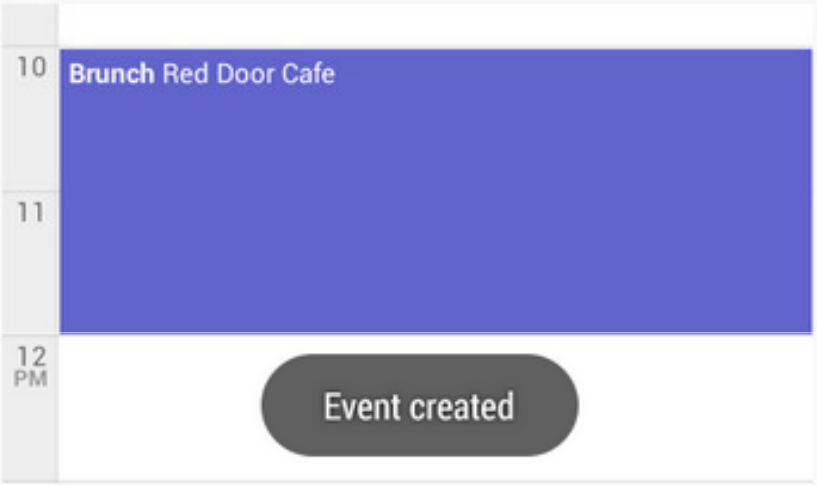
Selection


12:30
Application
Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore
1 selected

Confirming & Acknowledging



Confirming is asking the user to verify that they truly want to proceed with an action they just invoked. In some cases, the confirmation is presented along with a warning or critical information related to the action that they need to consider.



Acknowledging is displaying text to let the user know that the action they just invoked has been completed. This removes uncertainty about implicit operations that the system is taking. In some cases, the acknowledgment is presented along with an option to undo the action.

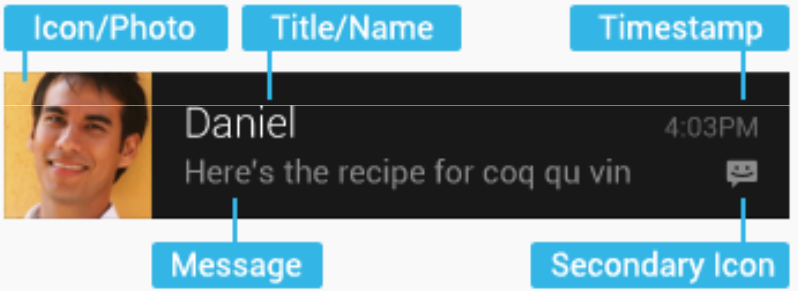
Notifications

Anatomy of a notification

Base Layout

At a minimum, all notifications consist of a base layout, including:

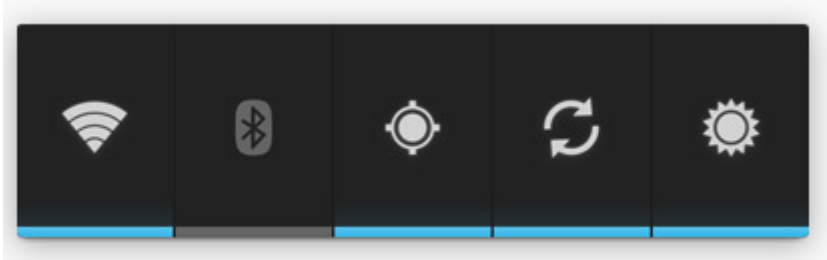
- the sending application's notification icon or the sender's photo
- a notification title and message
- a timestamp
- a secondary icon to identify the sending application when the senders image is shown for the main icon



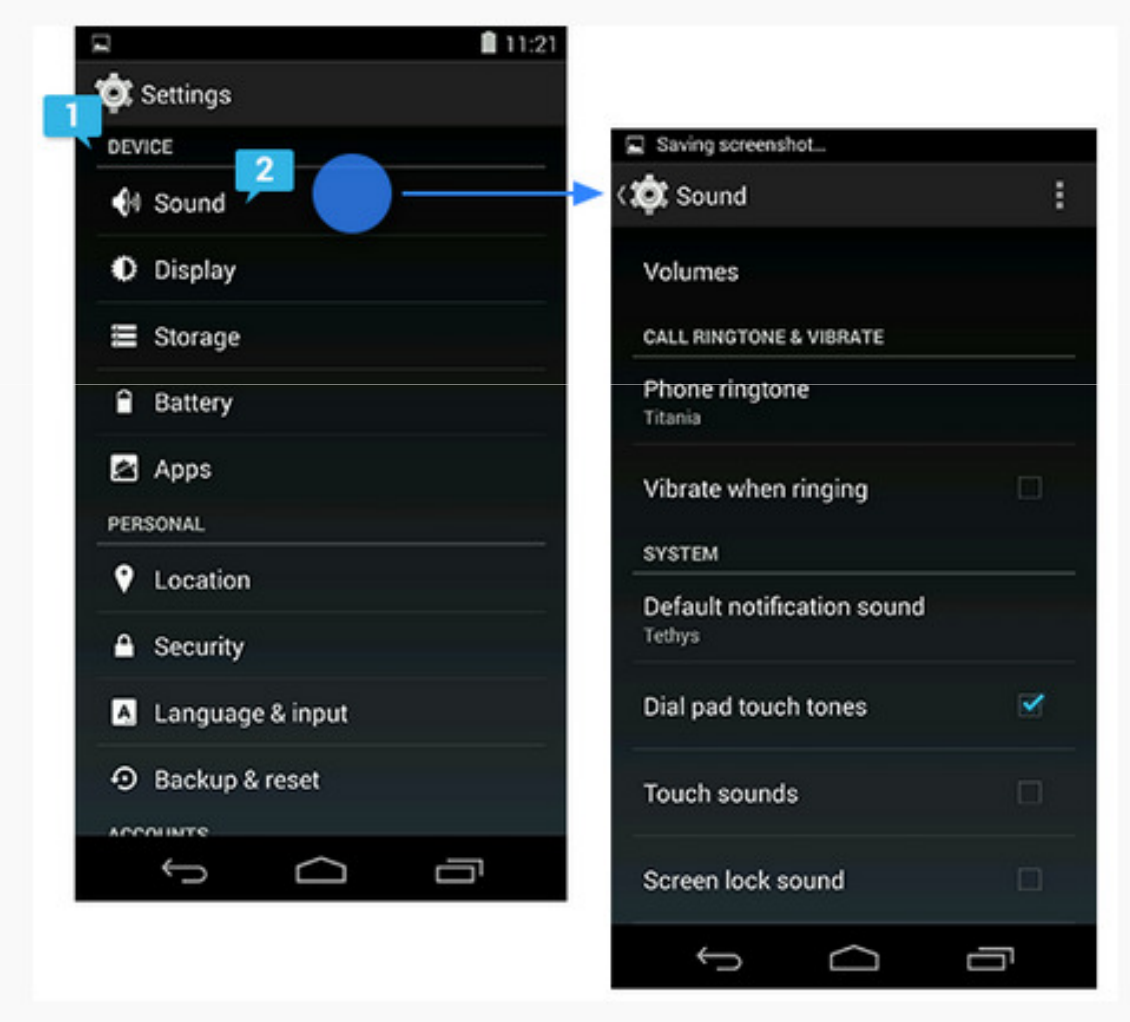
Base layout of a notification

Widgets

- "at-a-glance" views of an app's most important data and functionality that is accessible right from the user's home screen



Settings


11:21
Settings
1
DEVICE
2
Sound
Display
Storage
Battery
Apps
PERSONAL
Location
Security
Language & input
Backup & reset
Saving screenshot...
Sound
Volumes
CALL RINGTONE & VIBRATE
Phone ringtone
Titania
Vibrate when ringing
SYSTEM
Default notification sound
Tethys
Dial pad touch tones
Touch sounds
Screen lock sound

Help

- Principles for Writing On-Screen Help Content: http://developer.android.com/design/patterns/help.html

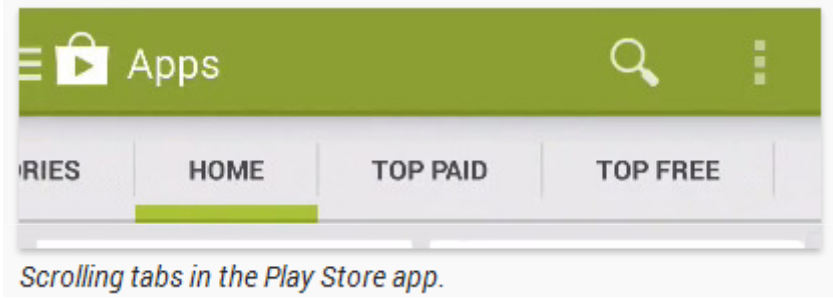
Android App UI Building Blocks

- Tabs
- Lists
- Grid Lists
- Spinners
- Buttons
- Text Fields
- Seek Bars
- Progress & Activity
- Switches
- Dialogs
- Pickers

http://developer.android.com/design/building-blocks/index.html

Tabs

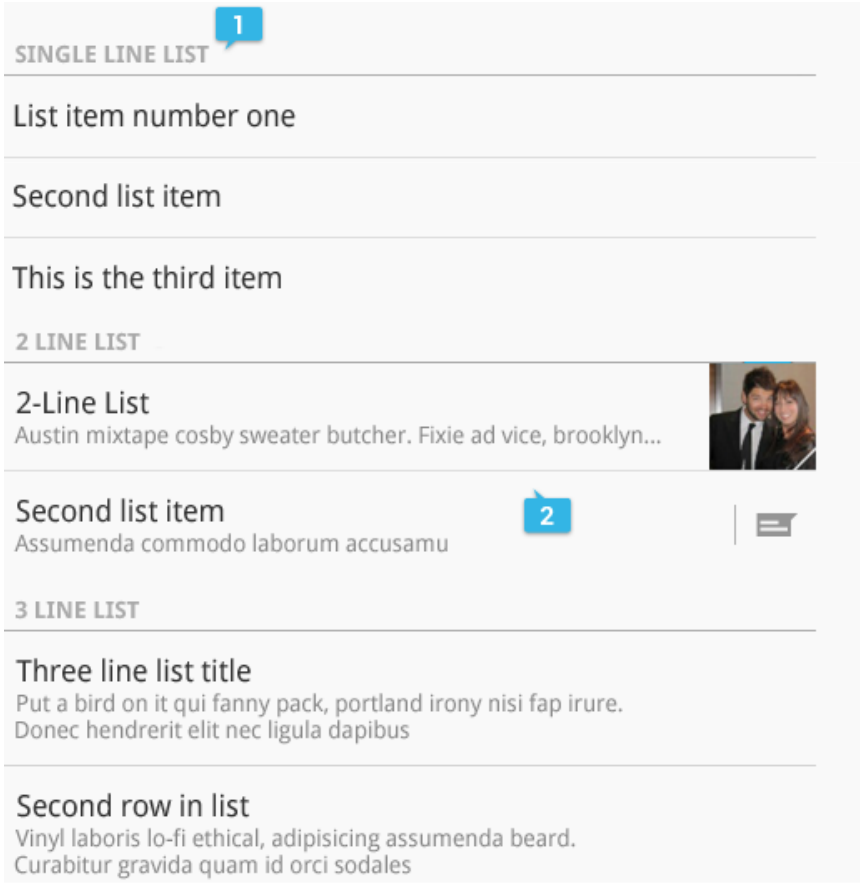
- Scrollable and Fixed tabs



Scrolling tabs in the Play Store app.

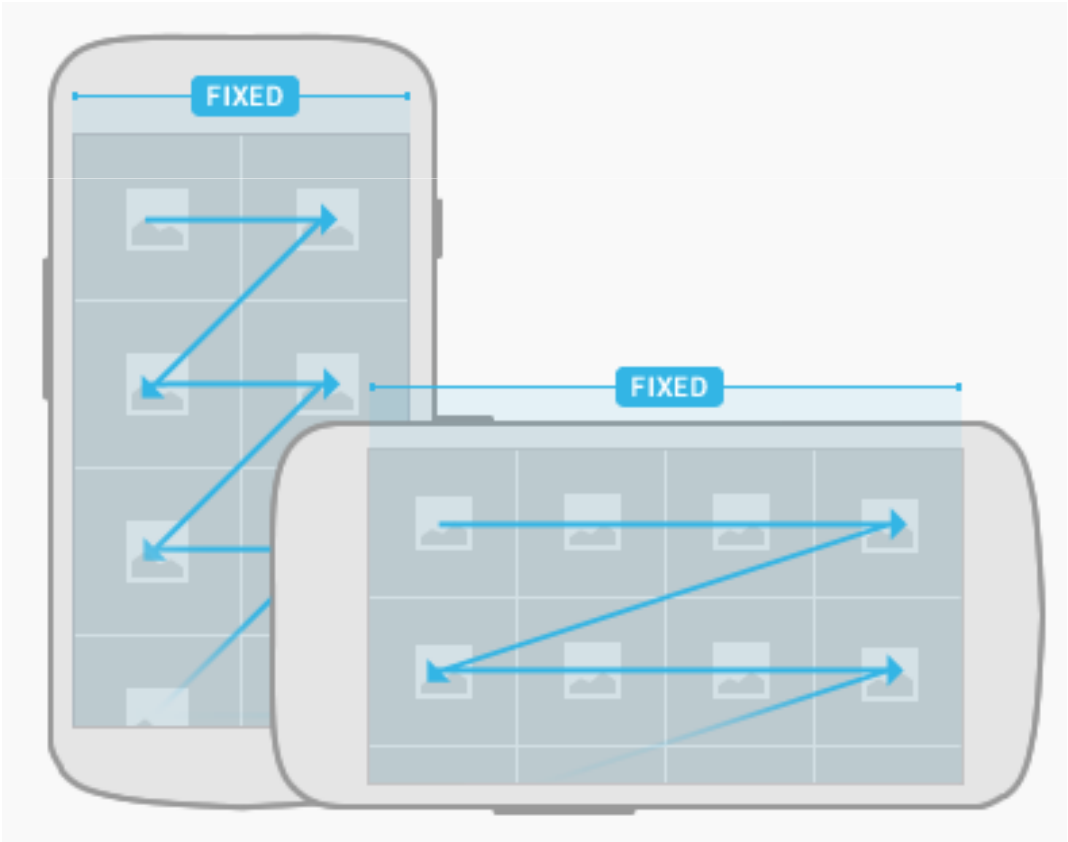
Lists

- Lists present multiple line items in a vertical arrangement. They can be used for data selection as well as drilldown navigation.



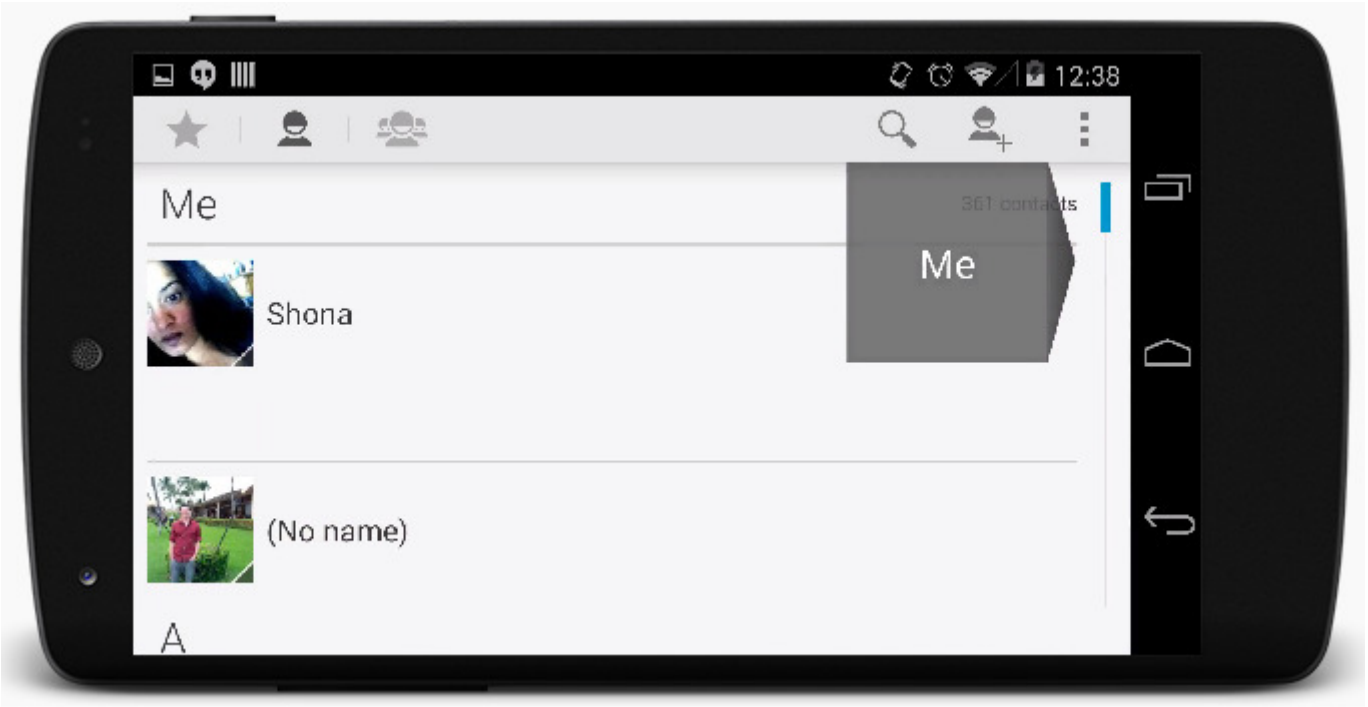
Grid Lists

- Vertical vs. horizontal scrolling

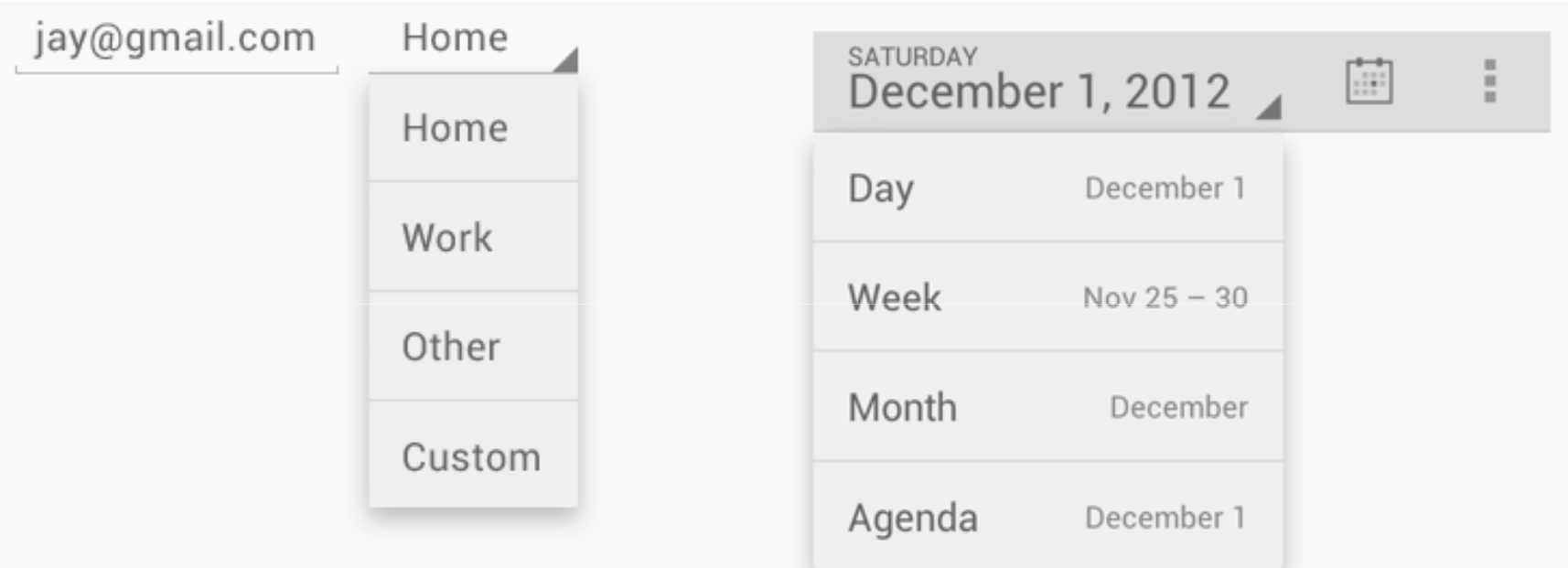


Scrolling

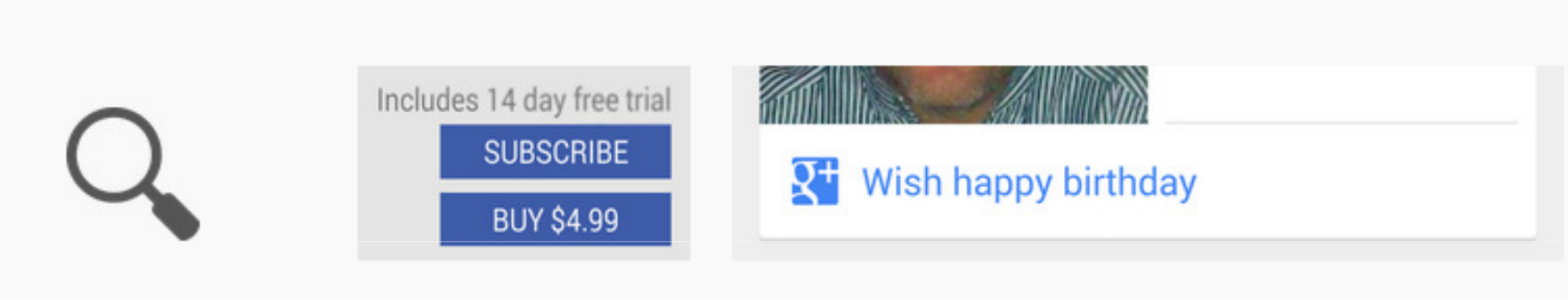
- Scrolling allows the user to navigate to content in the overflow using a swipe gesture. The scrolling speed is proportional to the speed of the gesture.



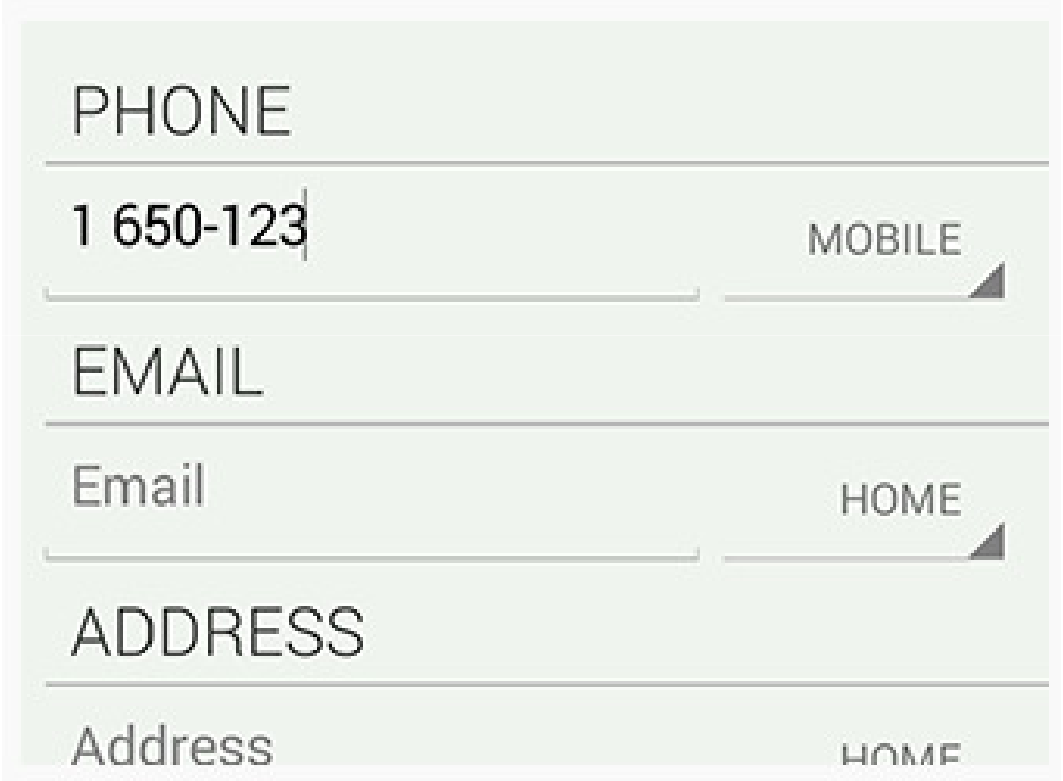
Spinners


jay@gmail.com
Home
Home
Work
Other
Custom
SATURDAY
December 1, 2012
Day
December 1
Week
Nov 25 – 30
Month
December
Agenda
December 1

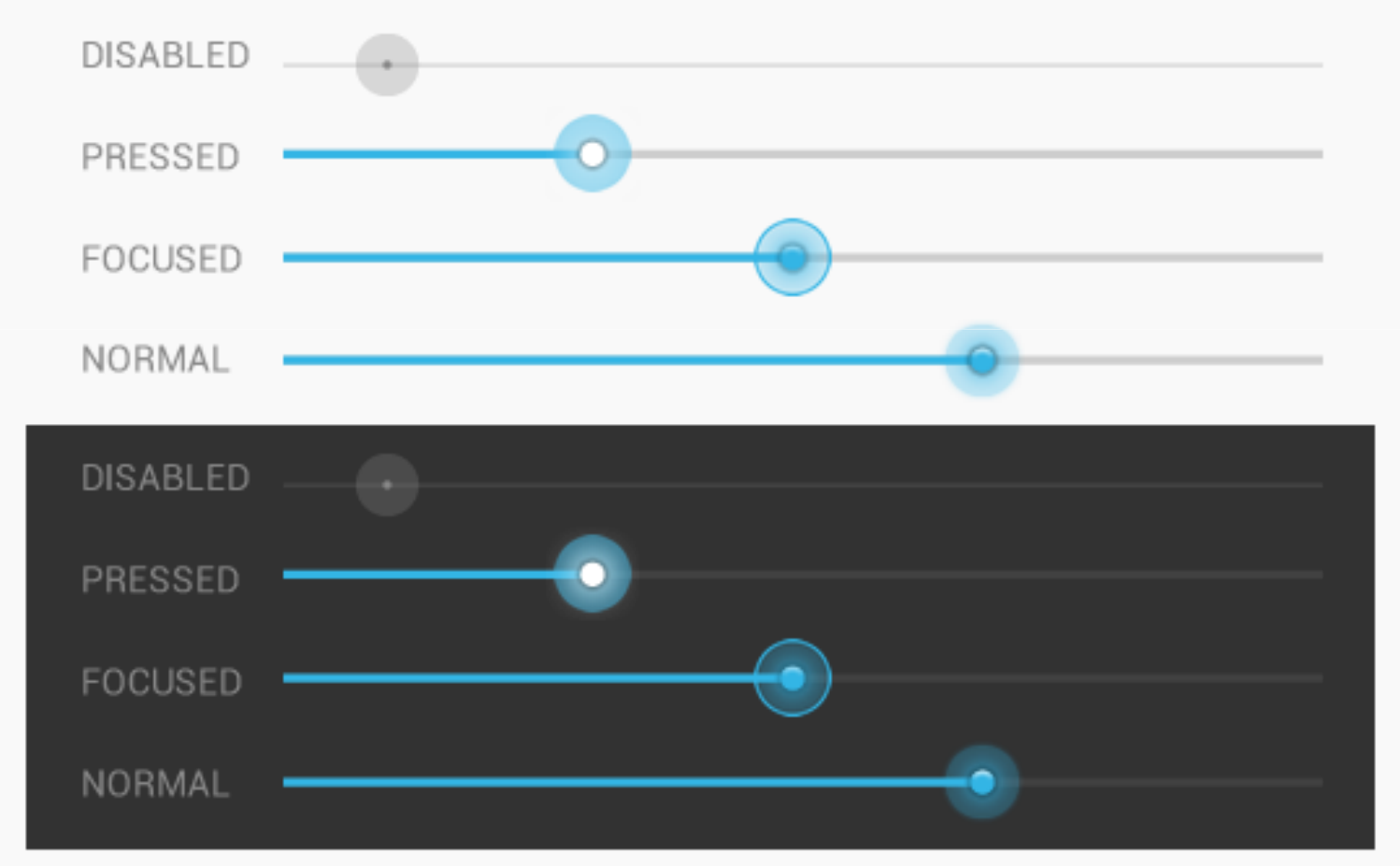
Buttons


Includes 14 day free trial
SUBSCRIBE
BUY $4.99
Wish happy birthday

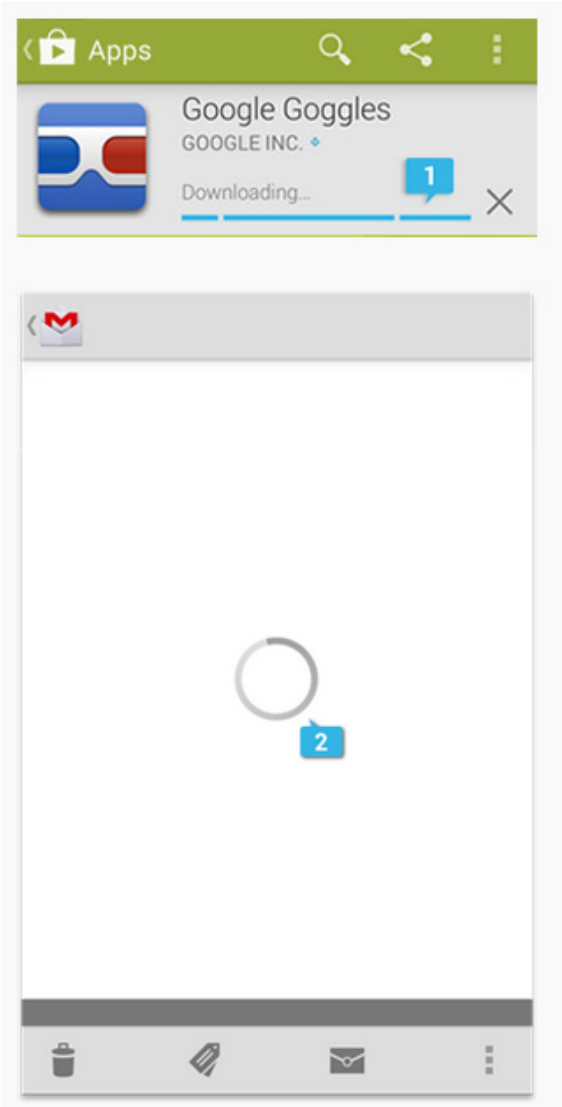
Text Fields


PHONE
1 650-123
MOBILE
EMAIL
Email
HOME
ADDRESS
Address
HOME

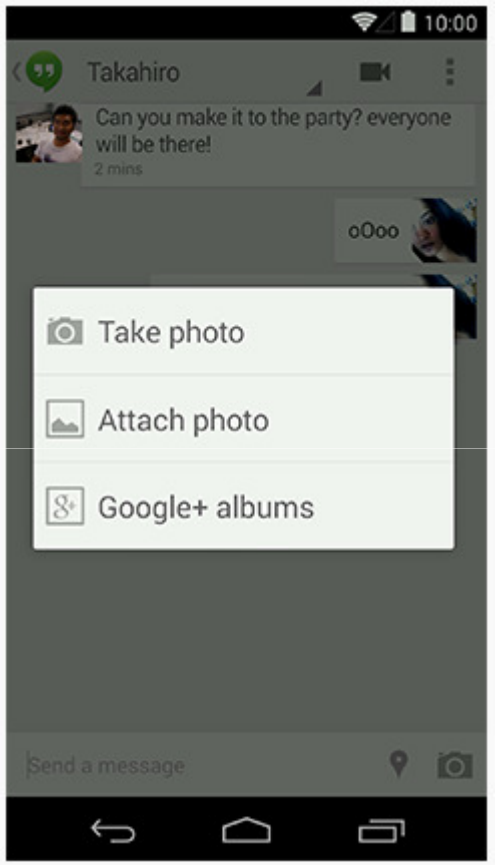
Seek Bars or Sliders

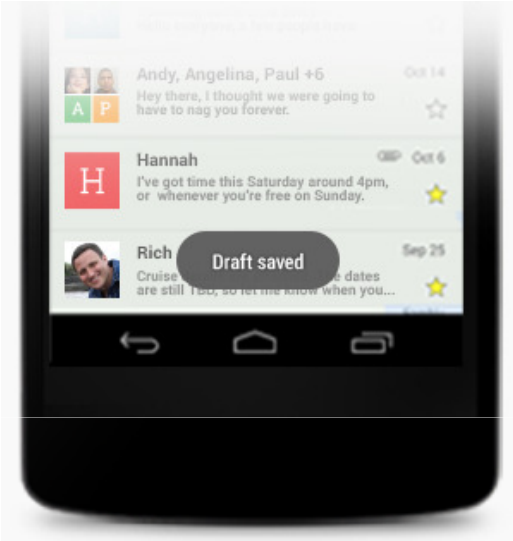


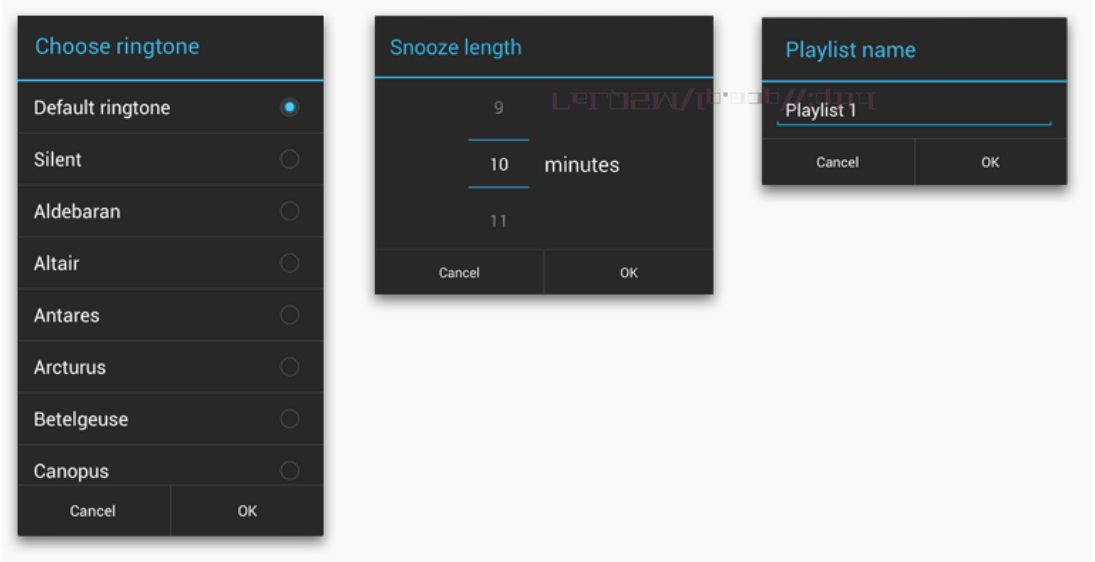
Progress & Activity



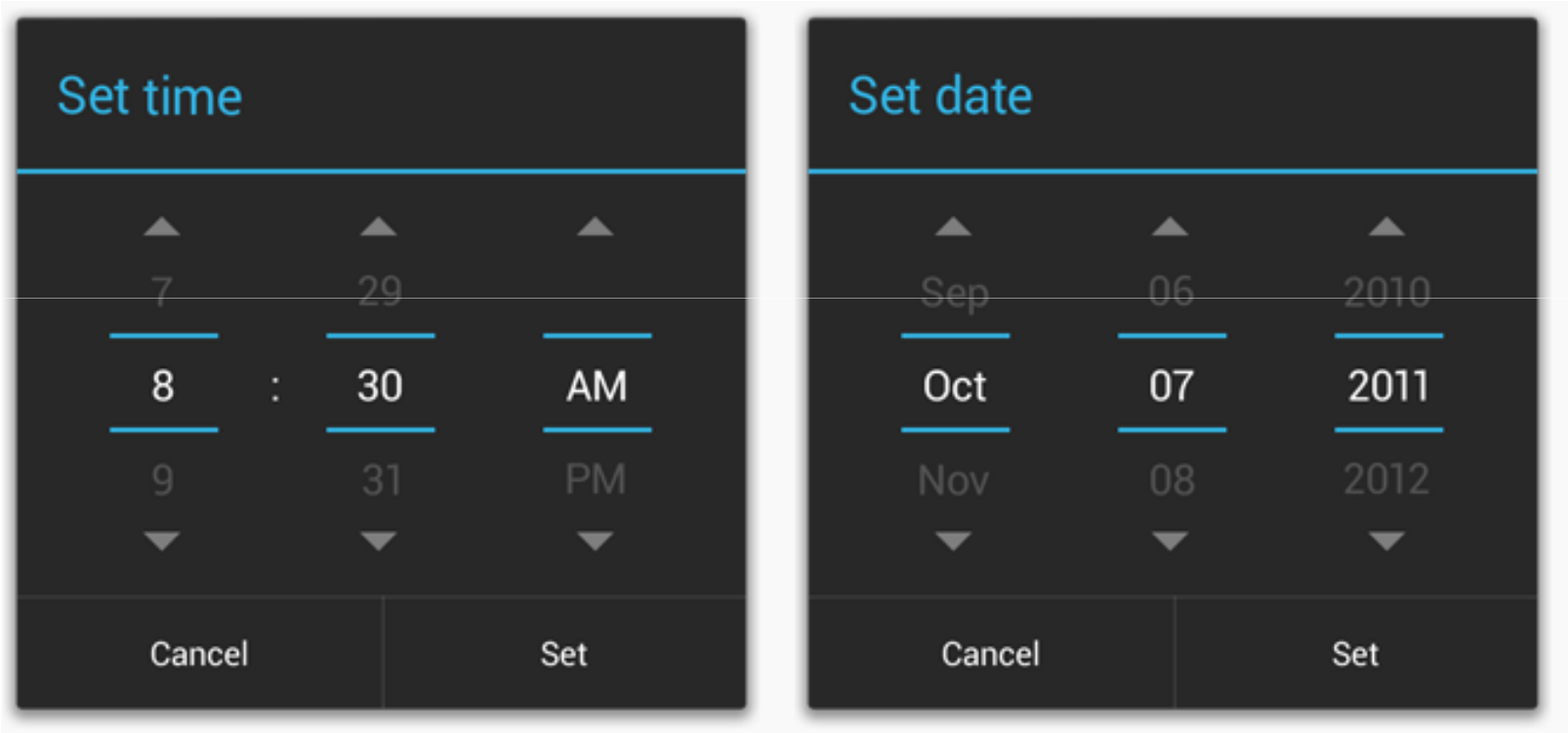
Dialogs


10:00
Takahiro
Can you make it to the party? everyone will be there!
2 mins
oOoo
Take photo
Attach photo
Google+ albums
Send a message

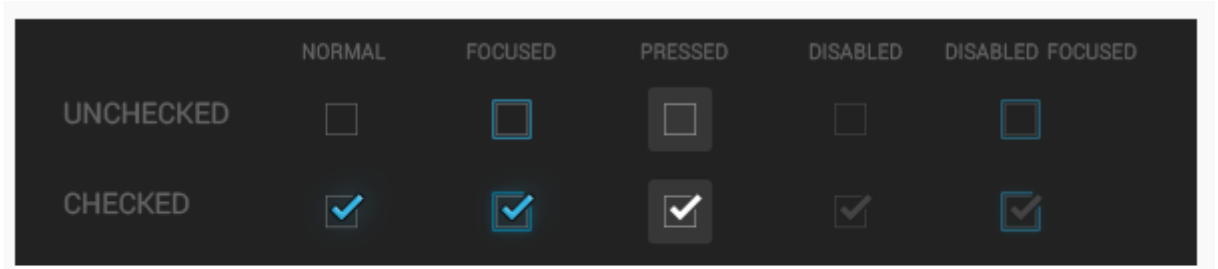

Andy, Angelina, Paul +6
Hey there, I thought we were going to have to nag you forever.
Hannah
I've got time this Saturday around 4pm, or whenever you're free on Sunday.
Rich
Draft saved


Choose ringtone
Default ringtone
Silent
Aldebaran
Altair
Antares
Arcturus
Betelgeuse
Canopus
Cancel
OK
Snooze length
9
10
minutes
11
Cancel
OK
Playlist name
Playlist 1
Cancel
OK

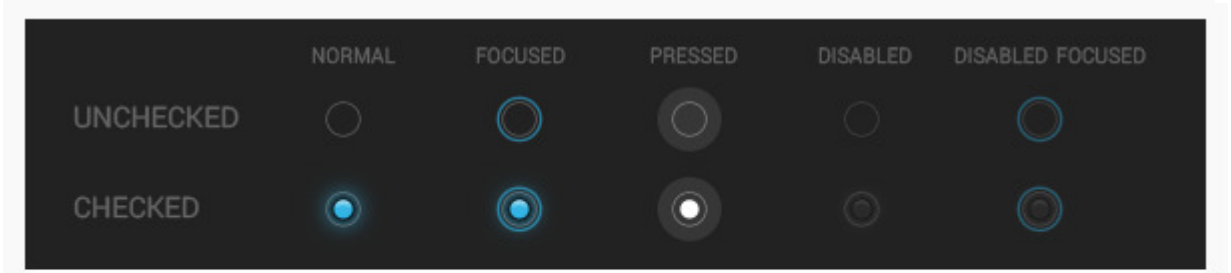
Pickers


Set time
7
29
8
:
30
AM
9
31
PM
Cancel
Set
Set date
Sep
06
2010
Oct
07
2011
Nov
08
2012
Cancel
Set

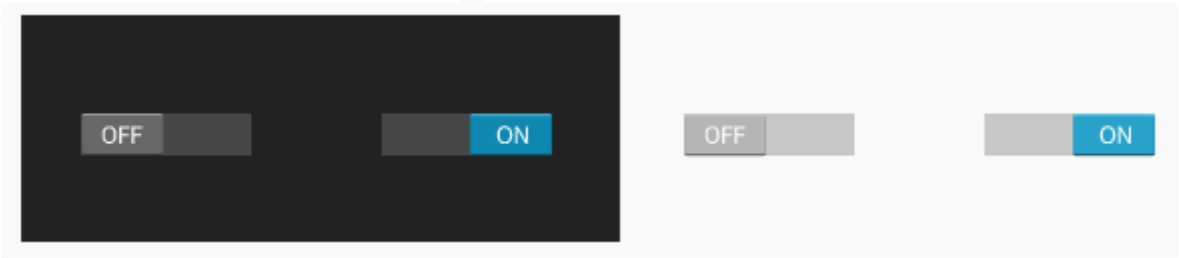
Switches


NORMAL
FOCUSED
PRESSED
DISABLED
DISABLED FOCUSED
UNCHECKED
CHECKED

UNCHECKED
CHECKED


NORMAL
FOCUSED
PRESSED
DISABLED
DISABLED FOCUSED
UNCHECKED
CHECKED

UNCHECKED
CHECKED


OFF
ON
OFF
ON

Accessibility

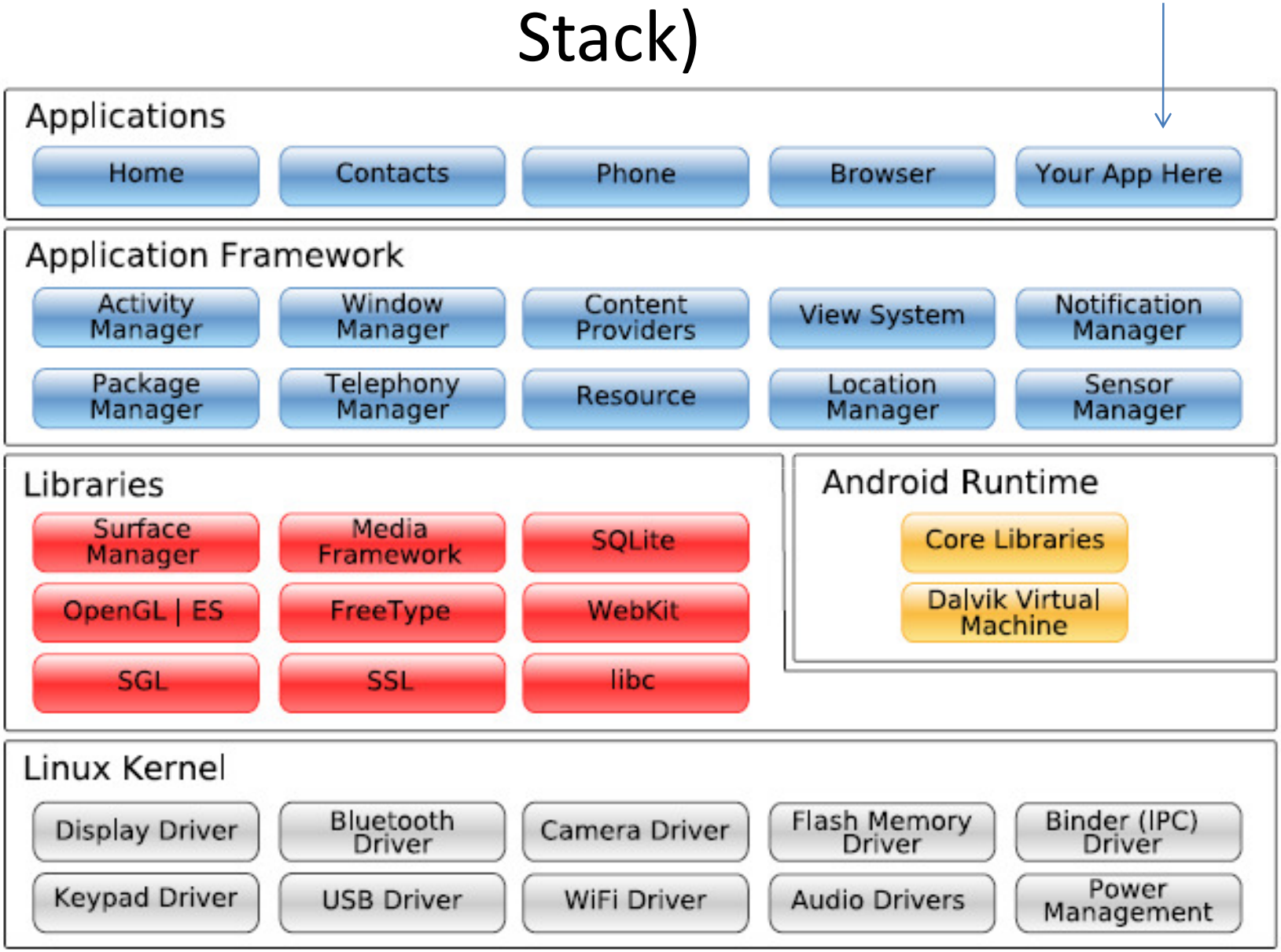
- Accessibility is the measure of how successfully a product can be used by people with varying abilities.
  - Make navigation intuitive
  - Use recommended touch target sizes
  - Label visual UI elements meaningfully
  - Provide alternatives to affordances that time out
  - Use standard framework controls or enable TalkBack for custom controls
  - Try it out yourself

Develop

Android Architecture

- Applications (highest layer)
- Application framework
- Android runtime + Libraries
- Linux kernel (lowest layer)

Android Architecture (Android Software Stack)



Android open source software stack

- All Activities and Services in an application run in a single process by default.
- By default, all of the application code in a single process runs in the main UI thread. This is the same thread that also handles UI events.

Linux Kernel

- Linux provides the **hardware abstraction layer** for Android which allow Android to be ported to a wide variety of platforms
- Android uses Linux for its memory management, process management, networking, and other operating system services.
- Android developers will NOT make Linux calls directly.

Native Libraries

- Shared libraries are all written in C or C++, compiled for the particular hardware architecture used by the phone, and preinstalled by the phone vendor.
- Surface Manager:
  - compositing window manager (Vista, Compiz)
  - drawing commands go into offscreen bitmaps that are then combined with other bitmaps to form the display
  - allow interesting effects such as see-through windows and fancy transitions

Native Libraries

- 2D and 3D graphics:
  - 2- and 3-D elements can be combined in a single user interface
- Media codecs:
  - Android can play video and record and play back audio in a variety of formats including AAC, AVC (H.264), H.263, MP3, and MPEG-4.
- *SQL database:*
  - lightweight SQLite database engine (Firefox, iPhone)
  - for persistent storage
- Browser engine:
  - WebKit library (Google Chrome, Apple's Safari)

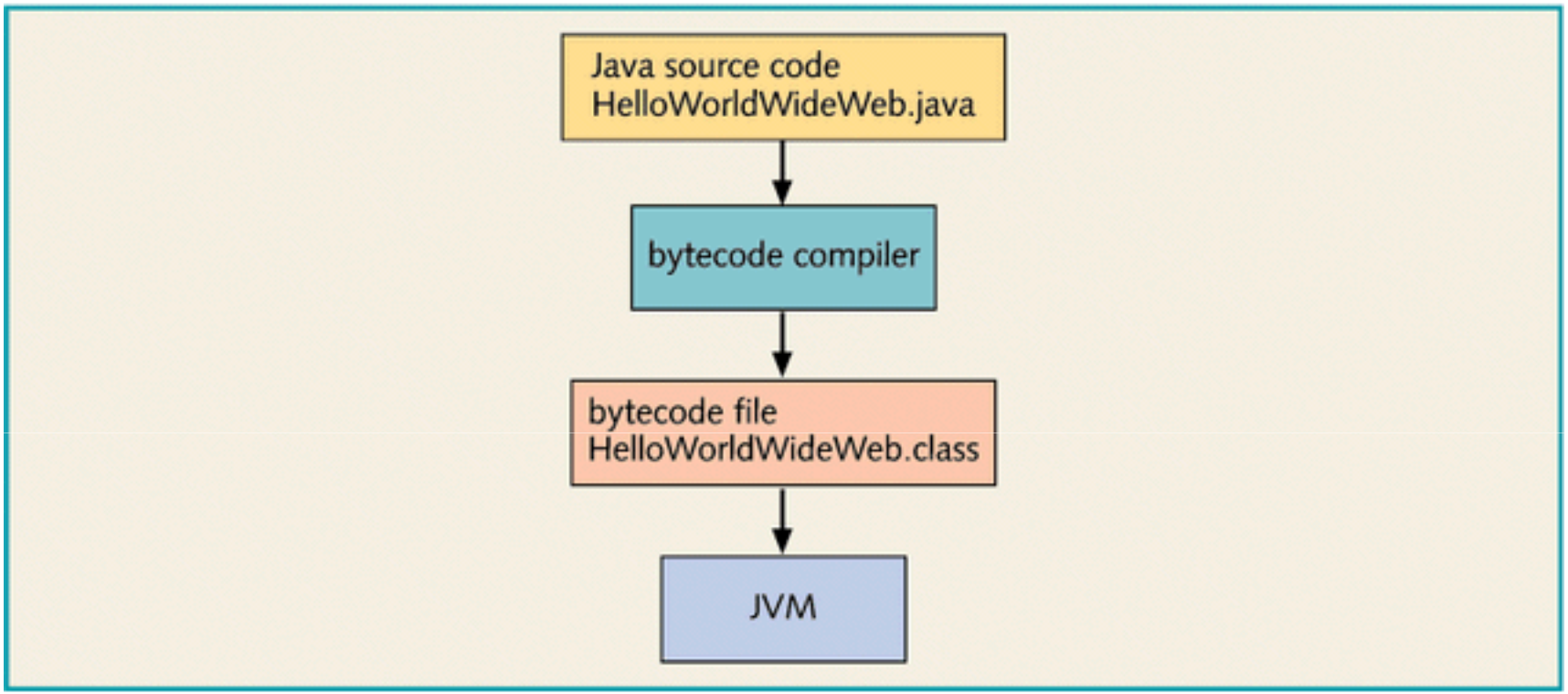


Figure 2-1 Compiling Java

Android Runtime

- Dalvik virtual machine (VM):
  - Google's implementation of Java, optimized for mobile devices
  - All the code you write for Android in Java will run within the VM
  - The VM runs .dex files, which are converted at compile time from standard .class and .jar files.
  - .dex files are more compact and efficient than class files, an important consideration for the limited memory and battery-powered devices that Android targets.
- core Java libraries
  - different from Java SE / Java ME libraries

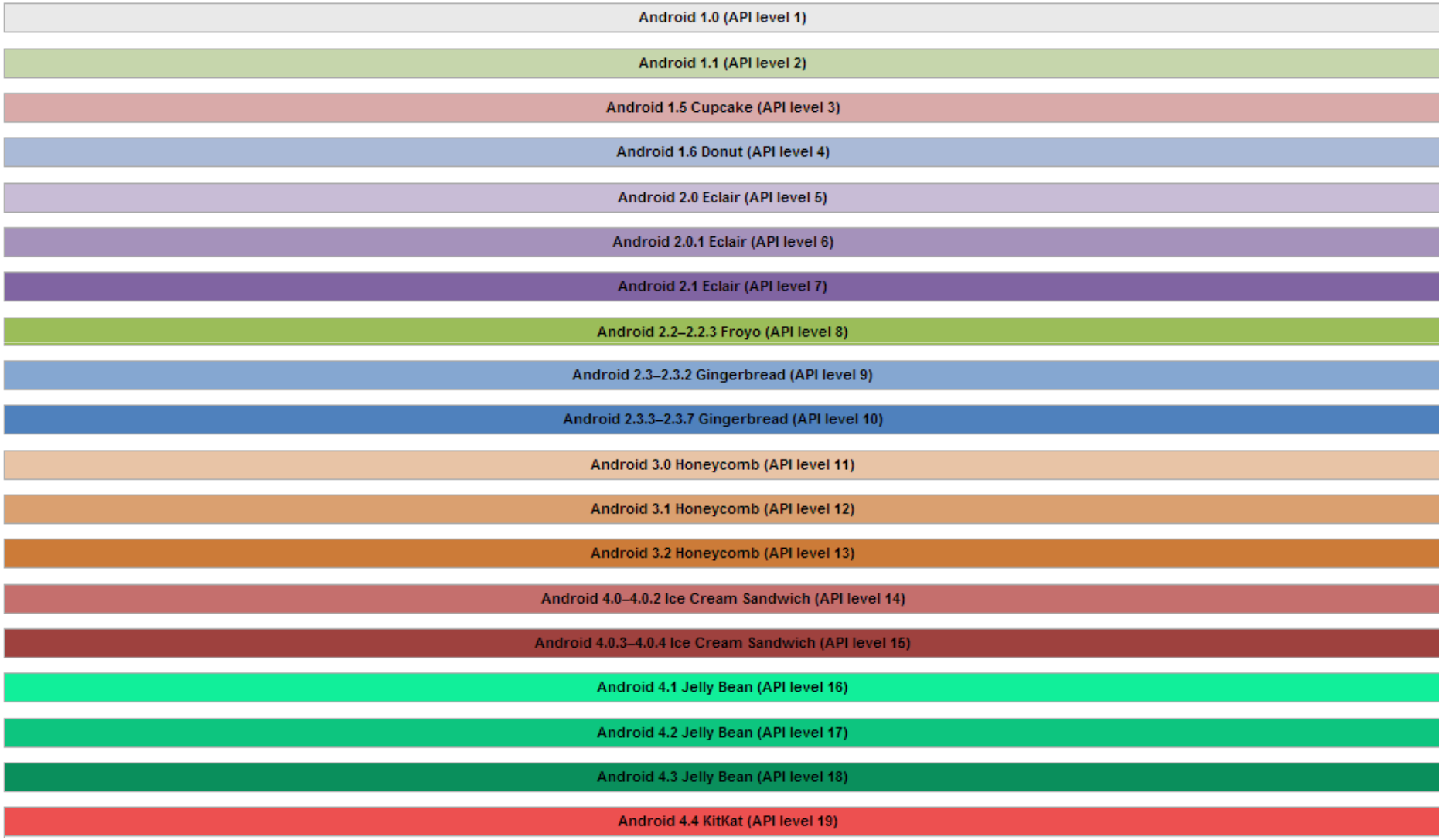
Application Framework

- high-level building blocks to create Android applications
- preinstalled with Android and extensible:
  - **Activity manager**: controls the life cycle of applications
  - **Content providers**: encapsulate data that needs to be shared between applications, such as contacts.
  - **Resource manager**: Resources are anything that goes with your program that is not code.
  - **Location manager**: An Android phone always knows where it is (location).
  - **Notification manager**: Events such as arriving messages, appointments, proximity alerts, alien invasions, and more can be presented in an unobtrusive fashion to the user.

What is an API Level?

- The API Level identifier serves a key role in ensuring the best possible experience for users and application developers:
  - It lets the Android platform describe the maximum framework API revision that it supports
  - It lets applications describe the framework API revision that they require
  - It lets the system negotiate the installation of applications on the user's device, such that version-incompatible applications are not installed.

Android versions & API Levels



http://en.wikipedia.org/wiki/Android_version_history

What is an API Level?

- The Android platform provides a **framework API** that applications can use to interact with the underlying Android system.
- **API Level** is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.
- The framework API consists of:
  - A core set of **packages** and **classes**
  - A set of XML elements and attributes for declaring a **manifest** file
  - A set of XML elements and attributes for declaring and accessing **resources**
  - A set of **Intents**
  - A set of permissions that applications can request, as well as permission enforcements included in the system
- Read more: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels

Applications

- **End users** will see only these applications, blissfully unaware of all the action going on below the waterline.
- Some standard system applications:
  - Phone dialer
  - Email
  - Contacts
  - Web browser
  - Android Market / Google Play
  - Etc.

Android Application Fundamentals

- Written in **Java**
- Android SDK compiles source codes into **Android package**, an archive file with an ***.apk*** suffix (with all data and resource files)
- 1 Android App = 1 single ***.apk*** file (file that Android-powered devices use to install the application)
- Once installed on a device, each Android application lives in its own security sandbox: **principle of least privilege** - each application, by default, has access only to the components that it requires to do its work and no more - very secure environment

Android Application Fundamentals

- Android operating system = **multi-user Linux system** (each application is a different user)
- By default, the system assigns each application a **unique Linux user ID** (used only by the system and is unknown to the application).
- The system sets permissions for all the files in an application so that only the user ID assigned to that application can access them.

Android Application Fundamentals

- Each process has its **own virtual machine (VM)**, so an application's code runs in isolation from other applications.
- By default, every application **runs in its own Linux process or VM**.
- Android *starts* the process when any of the application's components need to be executed, then *shuts down* the process when it is no longer needed or when the system must recover memory for other applications.

Can applications share data and access system services? Yes

- Share data/files:
  - Apps share the **same Linux user ID**, run in the same Linux process and share the same VM
- Access system services / device data:
  - An app can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, Bluetooth, and more (All application permissions must be granted at install time.)

How Android Apps work?

- One foreground application, which typically takes over the whole display except for the status line
- When the user turns on their phone, the first application: **Home application**

How Android Apps work?

- When the user runs an application, Android starts it and brings it to the foreground.
- From that application, the user might invoke another application, or another screen in the same application, and then another and another.
- All these programs and screens are recorded on the *application stack* by the system's Activity Manager.
- At any time, the user can press the Back button to return to the previous screen on the stack.
- From the user's point of view, it works a lot like the history in a web browser. Pressing Back returns them to the previous page.

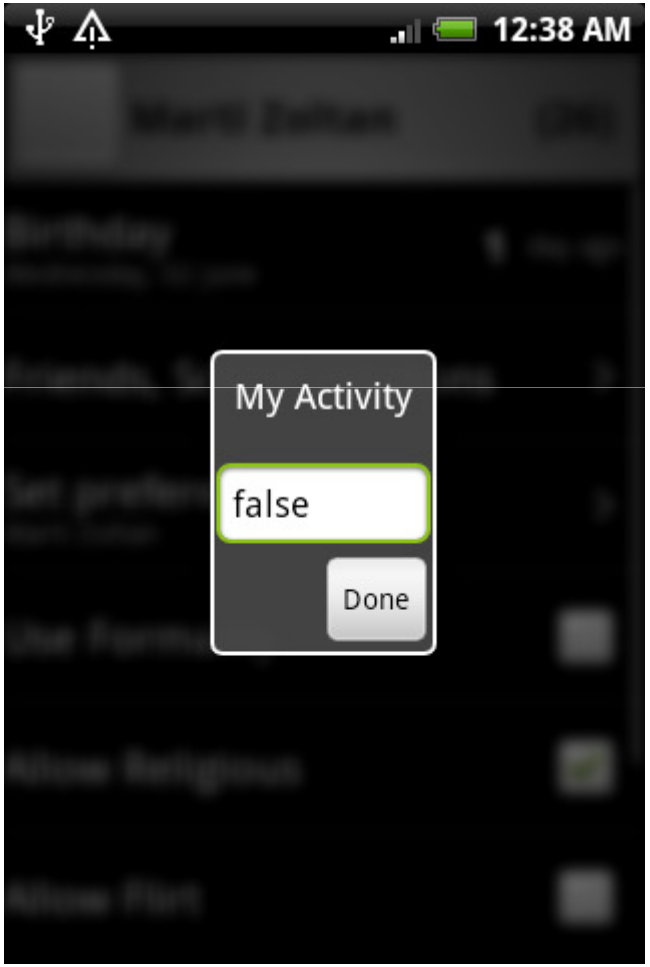
Anatomy of Android Components

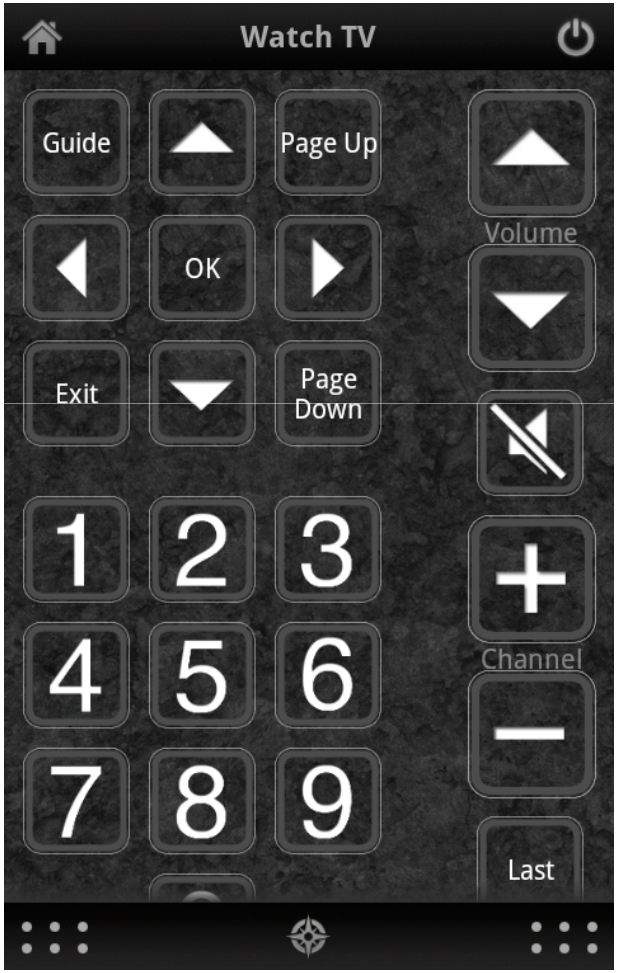
- **Activities**
- Services
- Content providers
- Broadcast receivers
- Widgets (Homescreen)

Activity

- An *activity* represents a **single screen** with a user interface.
- For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails.
- Each activity is **independent** of the others.
- A different application can start any one of these activities (if the application allows it during install time)
- An activity is implemented as a subclass of class *Activity*
- More details on *Activity* in Chapter 3

Activity

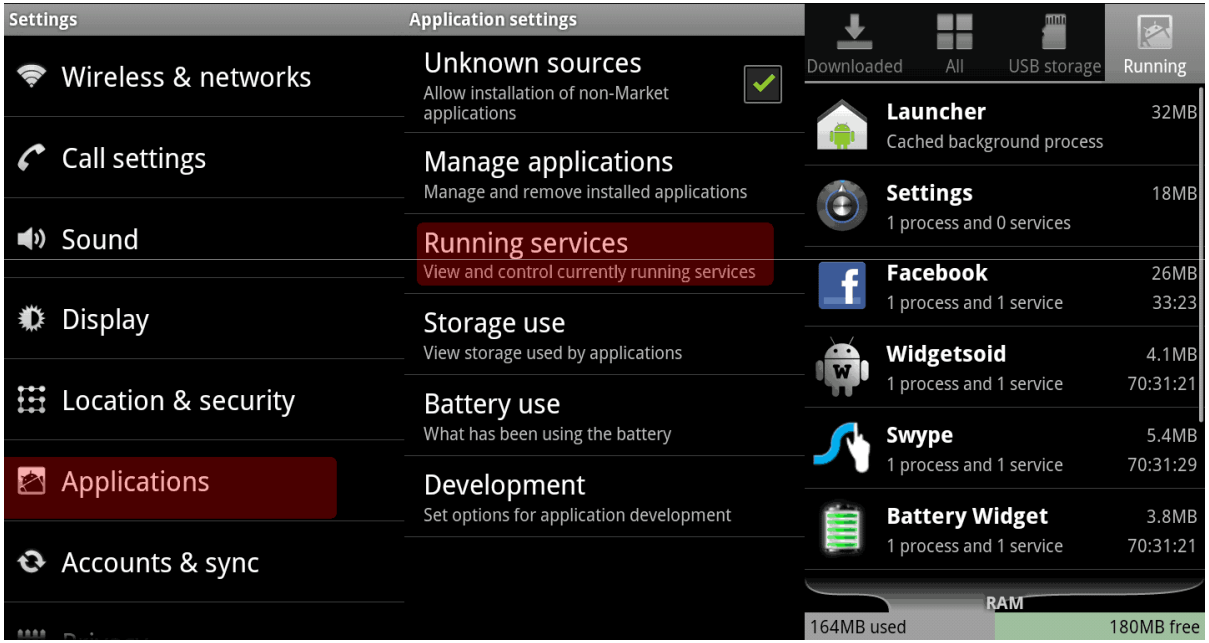

12:38 AM
My Activity
false
Done


Watch TV
Guide
Page Up
OK
Exit
Page Down
Volume
1
2
3
4
5
6
7
8
9
Channel
Last

Services

- A *service* is a component that runs **in the background** to perform long-running operations or to perform work for remote processes.
- A service **does not provide a user interface**. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.
- An activity can start the service and let it run or bind to it in order to interact with it.
- A service is implemented as a subclass of class *Service*

Services

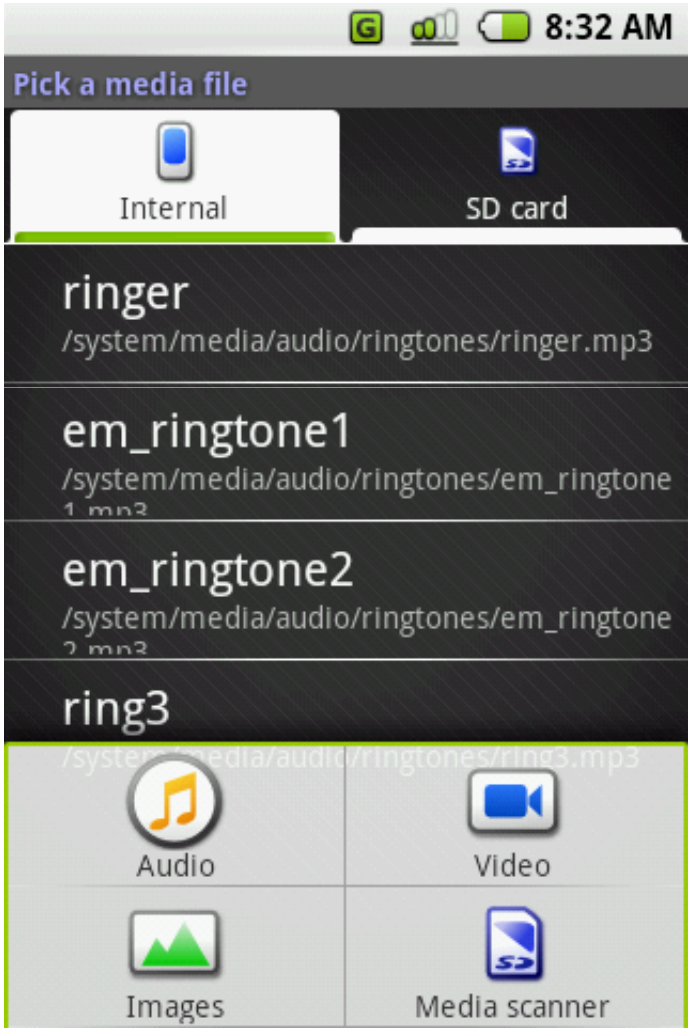

Settings
Wireless & networks
Call settings
Sound
Display
Location & security
Applications
Accounts & sync
Application settings
Unknown sources
Allow installation of non-Market applications
Manage applications
Manage and remove installed applications
Running services
View and control currently running services
Storage use
View storage used by applications
Battery use
What has been using the battery
Development
Set options for application development
Downloaded
All
USB storage
Running
Launcher
Cached background process
32MB
Settings
1 process and 0 services
18MB
Facebook
1 process and 1 service
26MB
33:23
Widgetsoid
1 process and 1 service
4.1MB
70:31:21
Swype
1 process and 1 service
5.4MB
70:31:29
Battery Widget
1 process and 1 service
3.8MB
70:31:21
RAM
164MB used
180MB free


Downloaded
All
On SD card
Running
Samsung home
Cached background process
9.8MB
Android Core Apps
Cached background process
5.3MB
Easy Battery Saver
Cached background process
3.8MB
Program monitor
Cached background process
2.6MB
Media
5.2MB
RAM
108MB used
134MB free

Content providers

- A *content provider* manages a shared set of application data in the **file system**, an SQLite database, on the Web, or any other **persistent storage** location an app can access or private **data** of an application
- Through the content provider, other applications can **query** or **modify** the data (if the content provider allows it). For example, user's contact information.

Content providers



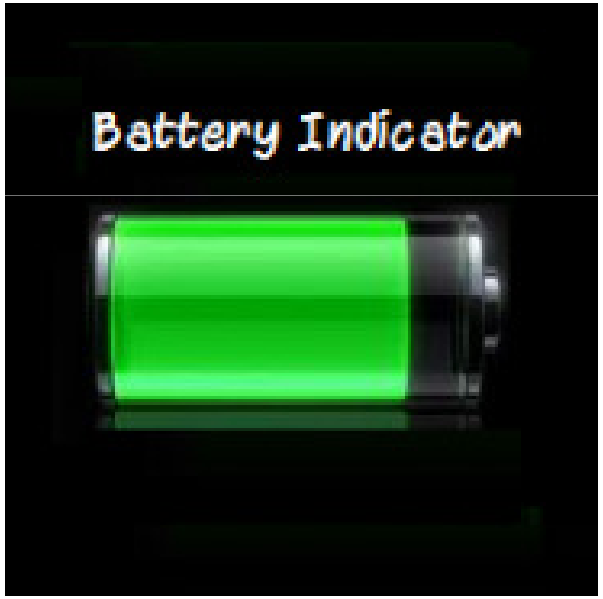
Broadcast Receivers

- A *broadcast receiver* is a component that responds to system-wide broadcast announcements (broadcasts originate **from the system**—announcing that the screen has turned off, the battery is low, or a picture was captured etc.)
- Applications can initiate broadcasts—for example, to let other applications know that some data has been downloaded to the device and is available for them to use.

Broadcast receivers

- Although *broadcast receivers* do not display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs.
- A broadcast receiver is implemented as a subclass of class BroadcastReceiver and each broadcast is delivered as an Intent object.

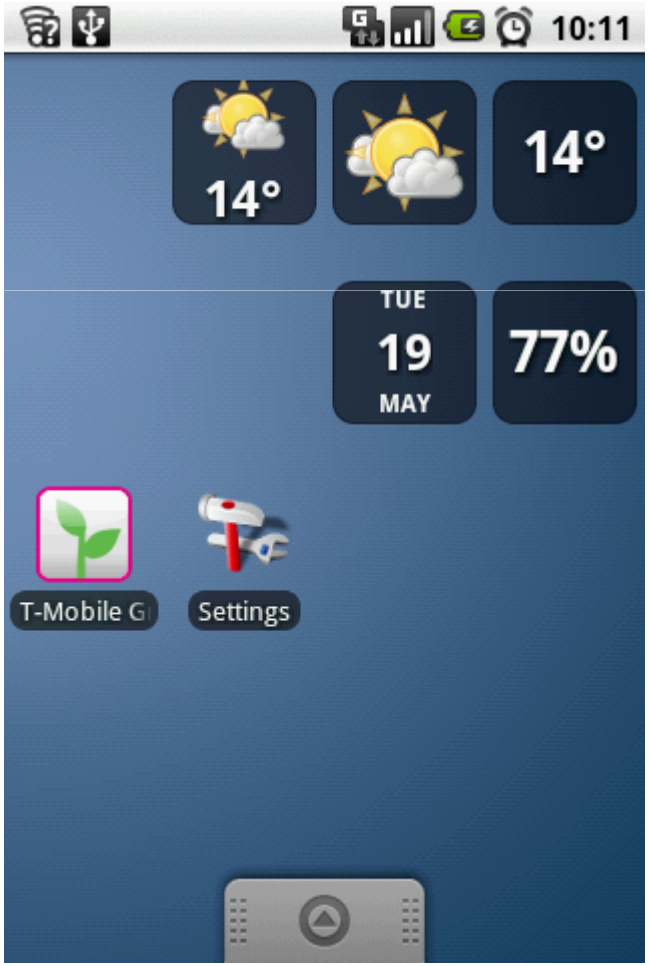
Broadcast receiver



Widgets

- *Widgets* are **interactive** components which are primarily used on the Android **homescreen**.
- Typically display some kind of data and allow the user to perform actions via them.
- For example a *Widget* could display a short summary of new emails and if the user selects an email, it could start the email application with the selected email.

Widgets


10:11
14°
14°
TUE
19
MAY
77%
T-Mobile G
Settings


10:22
Baseball
7
Shopping
14
days left
8
days left
295
Dec 24, 2011
Christmas
days left
347
Feb 14, 2012
Valentines
day
days left
93
Jun 5, 2011
steves
birthday
days left
93
Jun 5, 2011
Moms
birthday

Android Development

- Android is free to develop for any platform: Linux, Windows, Mac
- Lots of info on the Web, lots of books written
- Documentation is good
- SDK is packed with tools
- Covered in terms of UI, resources, form-factors, security, debugging, testing and integration
- Enforces strict application model - you have to keep a lot of rules in mind
- Framework is good, so you need to learn things

Anatomy of an Android Application

TABLE 1.1 Android Project Folder Structure

ITEM	EXPLANATION
`src/`	This folder contains your app's Java source code. It follows the standard Java package conventions. For example, a `com.example.Foo` class would be located in the folder `src/com/example/Foo.java`.
`res/`	This folder contains all the resources of your app and is where you declare the layout using XML. This folder contains all layout files, images, themes, and strings.
`gen/`	This folder is auto-generated when you compile the XML layouts in `res/`. It usually contains only a single file, `R.java`. This file contains the constants you need to reference the resources in the `res/` folder. Do not edit anything in this folder.
`assets/`	This folder contains miscellaneous files needed by your application. If your app needs a binary asset to function, place it here. To use these files, you need to open them using the Java File application programming interfaces (APIs).
`AndroidManifest.xml`	The manifest contains essential information about your app that the Android system needs. This includes the activites and services your app uses, the permissions it requires, any intents it responds to, and basic info like the app name.
`default.properties`	Lists the Android API build target.

Anatomy of an Android Application

- **AndroidManifest.xml** describes the fundamental characteristics of the app and defines each of its components - various declarations
- **src/**
  - Directory for app's main source .java files. By default, it includes an Activity class that runs when your app is launched using the app icon.
- **bin/** contains files built by the ADT during the build process, generate **.apk** files (Android Package – application binary of an app)
- **assets/** contains assets used by the app such as HTML, text files, databases etc.

Anatomy of an Android Application

- **res/** contains contains all the resources used in the app to support devices with different screen resolutions and densities such as:
  - **layout**/ (e.g. main.xml)
    - Directory for files that define your app's user interface.
  - **values**/ (e.g. strings.xml)
    - Directory for other various XML files that contain a collection of resources, such as string and color definitions.

Common Android packages

- android.os.Bundle: http://developer.android.com/reference/android/os/Bundle.html
- android.app.Activity (activity base class): http://developer.android.com/reference/android/app/Activity.html

AndroidManifest.xml structure

- <uses-sdk>
  - attributes:
    - android:minSdkVersion
    - android:targetSdkVersion
- <application>
  - attributes:
    - android:label
    - android:theme
  - <activity>
    - attributes:
      - android:label
      - android:theme

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.Activity101"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".Activity101Activity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

For details, please refer to
http://developer.android.com/guide/topics/manifest/manifest-intro.html

The Manifest File

`AndroidManifest.xml`

- Before the Android system can start an application component, the system must know that the component exists by reading the application's `AndroidManifest.xml` *file* (declaration file).
- The file must be at the **root** of the application project directory
- 2 purposes:
  - Declaring components
  - Declaring component capabilities

The Manifest File

`AndroidManifest.xml`

- Declare app's structure and functionality.
  - activities the app uses,
  - services it provides,
  - database content via a content provider, and
  - intents it processes
- declare the physical hardware features an app needs to run – enable Google Play to filter applications based on a user's hardware configuration.
- declare the permissions required by an app
- declare the icons and labels used by an app

The Manifest File

`AndroidManifest.xml`

- declare your supported Android API versions

ITEM	EXPLANATION
`android:minSDKVersion`	Declares the minimum API level required by your application. Devices running Android versions lower than this will not be able to install your application.
`android:targetSDKVersion`	Declares the version of your application you are building against. This is what determines the features available to your app. If this differs from the `minSDKVersion`, you may need to use Java reflection to access APIs that are unavailable on the lower version.
`android:maxSDKVersion`	Declares the maximum SDK your application supports. Use this to prevent installation on newer devices that you may not be ready to support.

`AndroidManifest.xml`

Declaring application requirements

- define a profile for the types of **devices your application supports** by declaring device and software requirements
- informational only and the system does not read them
- external services such as Google Play do read them in order to provide filtering for users when they search for applications for their devices.

AndroidManifest.xml

Declaring components

- inform the system about the application's components

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

`AndroidManifest.xml`
Declaring components

- You must declare all application components this way:
  - **<activity> elements for activities**
  - <service> elements for services
  - <receiver> elements for broadcast receivers
  - <provider> elements for content providers
- Activities, services, and content providers that you include in your source files (.java) but do not declare in the manifest (`AndroidManifest.xml`) are not visible to the system and, consequently, can never run.

`AndroidManifest.xml`

Declaring component capabilities

- Explicitly naming the target component (using the component class name) in the intent
- The system identifies the components that can respond to an intent is by comparing the intent received to the *intent filters* provided in the manifest file of other applications on the device.
- We can optionally include intent filters that declare the capabilities of the component so it can respond to intents from other applications.
- We can declare an intent filter for component by adding an <intent-filter> element as a child of the component's declaration element.

Resources

- res/ folder
- Anything that isn't Java code such as images, layout files, app strings, localized strings, themes, animations and anything relating to the **visual presentation** of the application
- Define animations, menus, styles, colors, and the layout of activity user interfaces with **XML files** (structured values which are known to the Android platform)

Resources

- uses the directory structure to separate resources for use in different device configurations. For example
  - drawable-ldpi: low-density resources
  - drawable-mdpi: medium-density resources
  - drawable-hdpi: high-density resources
- At runtime, the Android system will select the proper resource based on the device hardware.
- If no resource matches, it will select the most closely matching resource.

res/values & strings.xml

- Place constant values used in layout: colors, dimensions, styles, and strings. For example user-visible strings in strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ExampleActivity!</string>
    <string name="app_name">Example</string>
</resources>
```

- Never use string literals in Java code or XML layouts.
- Always declare user-visible strings in the strings.xml file. This makes it easier to localize resources later. When using these strings in an app, reference them by the **name** attribute of the string element.
- A single string that can be referenced from the application or from other resource files (such as an XML layout). Resource reference:
  - In Java code: R.string.*string_name*
  - In XML:@string/*string_name*

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string
        name="string_name"
        >text_string</string>
</resources>
```

strings.xml

example:

XML file saved at **res/values/strings.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello!</string>
</resources>
```

main.xml →

This layout XML applies a string to a View:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

*.java →

This application code retrieves a string:

```
String string = getString(R.string.hello);
```

res/layout

- contains the XML files that declare application layout
- Android UI can be created using either XML or Java code.
- It is recommended to use XML for layouts, because it provides a good separation between UI and application logic (MVC)
- Folder names are used to separate layouts for different device configurations.

res/layout/**main.xml**

- define your UI using an XML file (located in the res/layout folder)

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

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
→ android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

Tools needed

- Android SDK (compiler, debugger, libraries, emulator, documentation, sample code, tutorials): http://developer.android.com/sdk/index.html
- Eclipse IDE: support coding, compiling and debugging of Android apps http://www.eclipse.org/downloads/
- Android Developer Tool plugin: http://developer.android.com/sdk/installing/bundle.html
- Note: Avoid installing Android SDK in Program Files as you may run into folder permission denied issues. Install it outside of that folder.

Tools

ANDROID SDK

The Android SDK is required to build and deploy Android applications. The SDK contains the tools you'll use to test and debug your application. It also contains tools for creating flexible layouts. You can download the Android SDK at http://developer.android.com/.

ECLIPSE

Eclipse is the recommended IDE for Android development and is the only IDE officially supported by Google. Google publishes a plugin called Android Developer Tools that integrates with Eclipse and provides features like a drag-and-drop interface builder. You are not required to use Eclipse, as the Android SDK fully supports command-line development. Throughout this book, however, it is assumed you are using Eclipse. You can download Eclipse at www.eclipse.org.

ANDROID SDK MANAGER

The Android SDK Manager is used to download and install the Android SDK. You will also use the SDK Manager to install add-on features like sample code, third-party emulators, and the compatibility library. The Android SDK Manager can also be used to create and launch emulated Android devices, called Android Virtual Devices. The Android SDK Manager can be found in the SDK `tools/` directory as `android`.

HIERARCHY VIEWER

This tool displays a graphical representation of your layout hierarchy and can help you debug layout performance issues by spotting overly complex layouts. It's also a good tool for understanding how Android layout works. You can find this tool in the SDK `tools/` directory as `hierarchyviewer`.

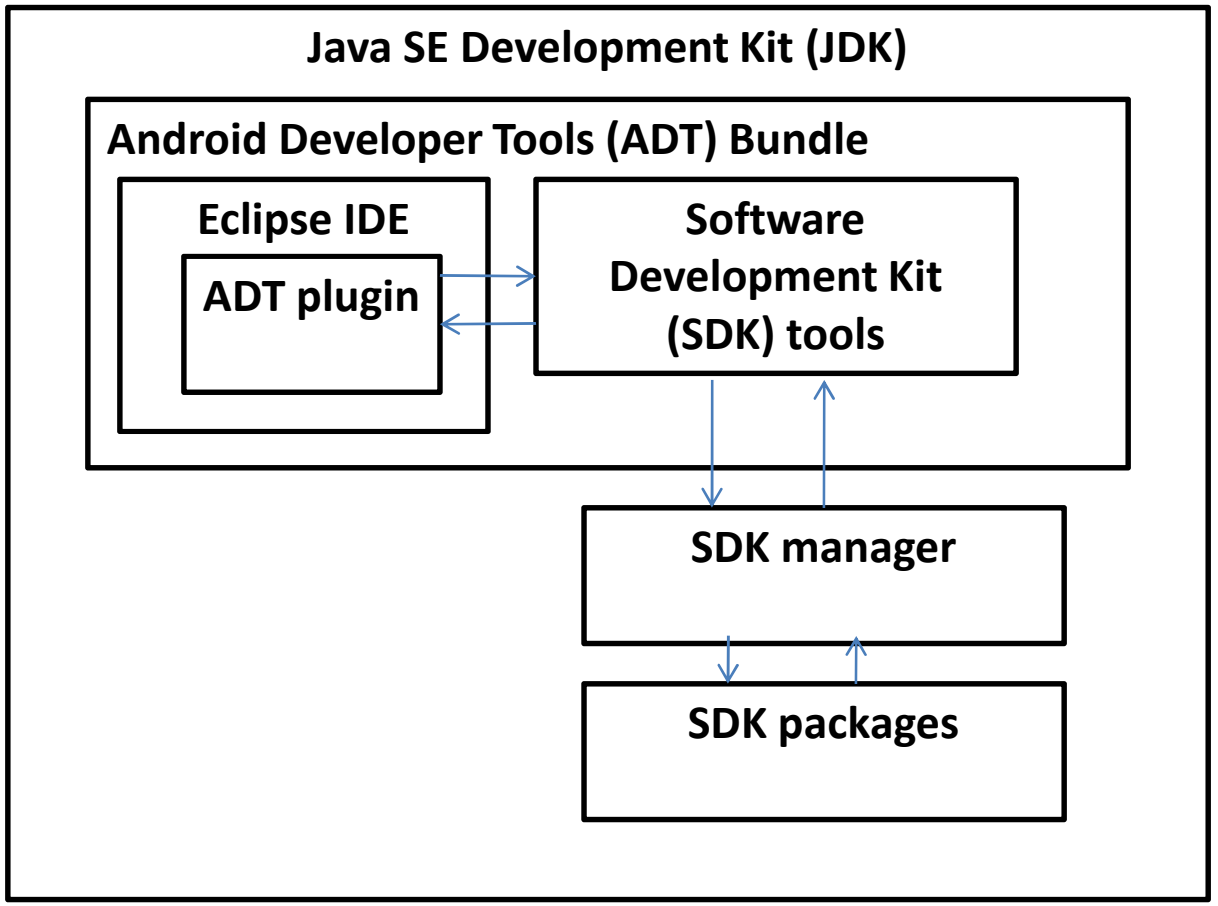
DDMS

The Dalvik Debug Monitor Server (DDMS) is used to debug application performance issues. It provides Java heap usage, running thread counts, and object allocation tracking. You also use it to take screen shots. The DDMS tool is built into Eclipse through the ADT or can be run standalone from the `tools/` directory of the SDK.

Setup and Installation

- Follow the directions provided on the developer website to set up the Eclipse development environment.
- Setting up Android SDK:
  - http://developer.android.com/sdk/installing/bundle.html
- Setting up Eclipse IDE:
  - http://developer.android.com/sdk/installing/installing-adt.html
- Setting up Android ADT plugin:
  - https://developer.android.com/sdk/installing/installing-adt.html

Development Environment



What to do first?

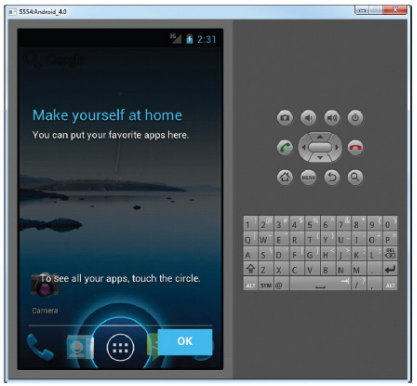
1. Download and install Android SDK (ADT bundle) and configure them

- download Android SDK:
  - https://developer.android.com/sdk/index.html#ExistingIDE
- install and set up Android SDK:
  - https://developer.android.com/sdk/installing/bundle.html
- Install Eclipse Plugin for Android:
  - https://developer.android.com/sdk/installing/installing-adt.html
- Add packages: Just add the general tools needed + tools for Android 4 and above - API Level 17 and above and configure Graphic Acceleration for the emulator:
  - http://developer.android.com/tools/devices/emulator.html#acceleration

2. Create a HelloWorld project and runs it on an emulator
   - https://developer.android.com/training/basics/firstapp/creating-project.html

Android Virtual Devices (AVDs)

- Emulator instance that enables you to model an actual device.
- Test your applications with several different configurations.
- Each AVD consists of a hardware profi le; a mapping to a system image; as well as emulated storage, such as a secure digital (SD) card.
- To confirm the behavior of your application when it is run on different devices with varying capabilities.



Developing First Android App

- Example in textbook pages 20-33
- http://developer.android.com/training/basics/firstapp/index.html
- More examples will be given in the lab

Development Considerations

- Application forward compatibility
- Application backward compatibility
- Selecting a platform version and API Level
- Declaring a minimum API Level

- Read more: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#considerations

Debugging from Eclipse with ADT

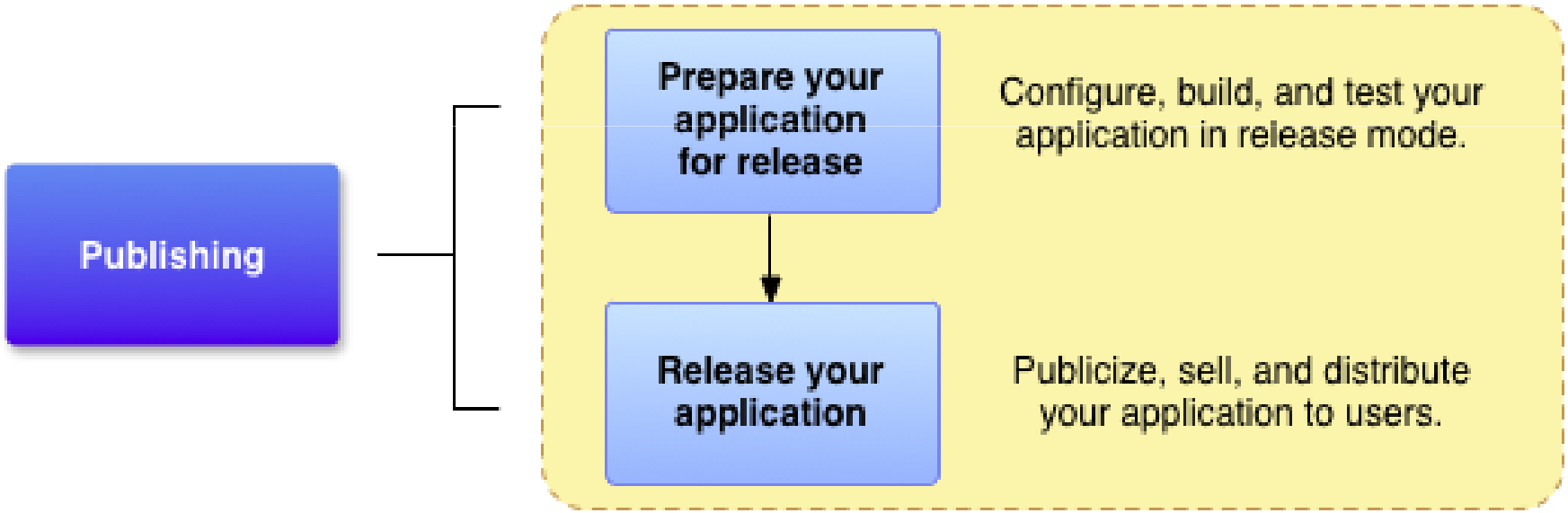
- **Debug Perspective: Window > Open Perspective > Debug**
  - Debug - Displays previously and currently debugged Android applications and its currently running threads
  - Variables - When breakpoints are set, displays variable values during code execution
  - Breakpoints - Displays a list of the set breakpoints in your application code
  - LogCat - Allows you to view system log messages in real time. The LogCat tab is also available in the DDMS perspective.
- http://developer.android.com/tools/debugging/debugging-projects.html

Debugging from Eclipse with ADT

- **DDMS (**Dalvik Debug Monitor Server) **Perspective: Window > Open Perspective > DDMS**
  - Devices - Shows the list of devices and AVDs that are connected to ADB.
  - Emulator Control - Lets you carry out device functions.
  - LogCat - Lets you view system log messages in real time.
  - Threads - Shows currently running threads within a VM.
  - Heap - Shows heap usage for a VM.
  - Allocation Tracker - Shows the memory allocation of objects.
  - File Explorer - Lets you explore the device's file system.
- http://developer.android.com/tools/debugging/debugging-projects.html

Distribute

Prepare for release



Google Play

- Get users
- Engage and retain
- Monetize

- More in Chapter 10

▶ WHAT YOU LEARNED IN THIS CHAPTER

TOPIC	KEY CONCEPTS
Android OS	Android is an open source mobile operating system based on the Linux operating system. It is available to anyone who wants to adapt it to run on their own devices.
Languages used for Android application development	You use the Java programming language to develop Android applications. Written applications are compiled into Dalvik executables, which are then run on top of the Dalvik virtual machine.
Android Market	The Android Market hosts all the various Android applications written by third-party developers.
Tools for Android application development	Eclipse IDE, Android SDK, and the ADT
Activities	An activity is represented by a screen in your Android application. Each application can have zero or more activities.
The Android manifest file	The `AndroidManifest.xml` file contains detailed configuration information for your application. As your example application becomes more sophisticated, you will modify this file, and you will see the different information you can add to it as you progress through the chapters.

Where to seek help online?

- Google Android Training: http://developer.android.com/training/index.html
- Stack Overflow: http://www.stackoverflow.com
- Android Discuss: https://groups.google.com/forum/?fromgroups#!forum/android-discuss

References

- Textbook: Wei-Meng Lee “Beginning Android 4 Application Development”
- Jason Ostrander (2012). *Android UI Fundamentals: Develop and Design.* Peachpit Press.
- Ed Burnette (2008). *Hello, Android. Introducing Google’s Mobile Development Platform*. The Pragmatic Bookshelf, Raleigh, North Carolina Dallas, Texas
- Zigurd Mednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura (2011). *Programming Android - Java Programming for the New Generation of Mobile Devices*. O'Reilly Media
- http://developer.android.com/index.html
- http://www.vogella.com/articles/Android/article.html

Tutorials

- Android - A beginner's guide: http://www.codeproject.com/Articles/102065/Android-A-beginner-s-guide
- Android Development Tutorial: http://www.vogella.com/articles/Android/article.html#overview
- http://androiddevelopement.blogspot.com/2011/12/android-40-development-tutorial.html