Android

Steven R. Bagley

Android

- Today, overview of the Android software stack
- Tomorrow, how we can use the SDK to write and test apps…

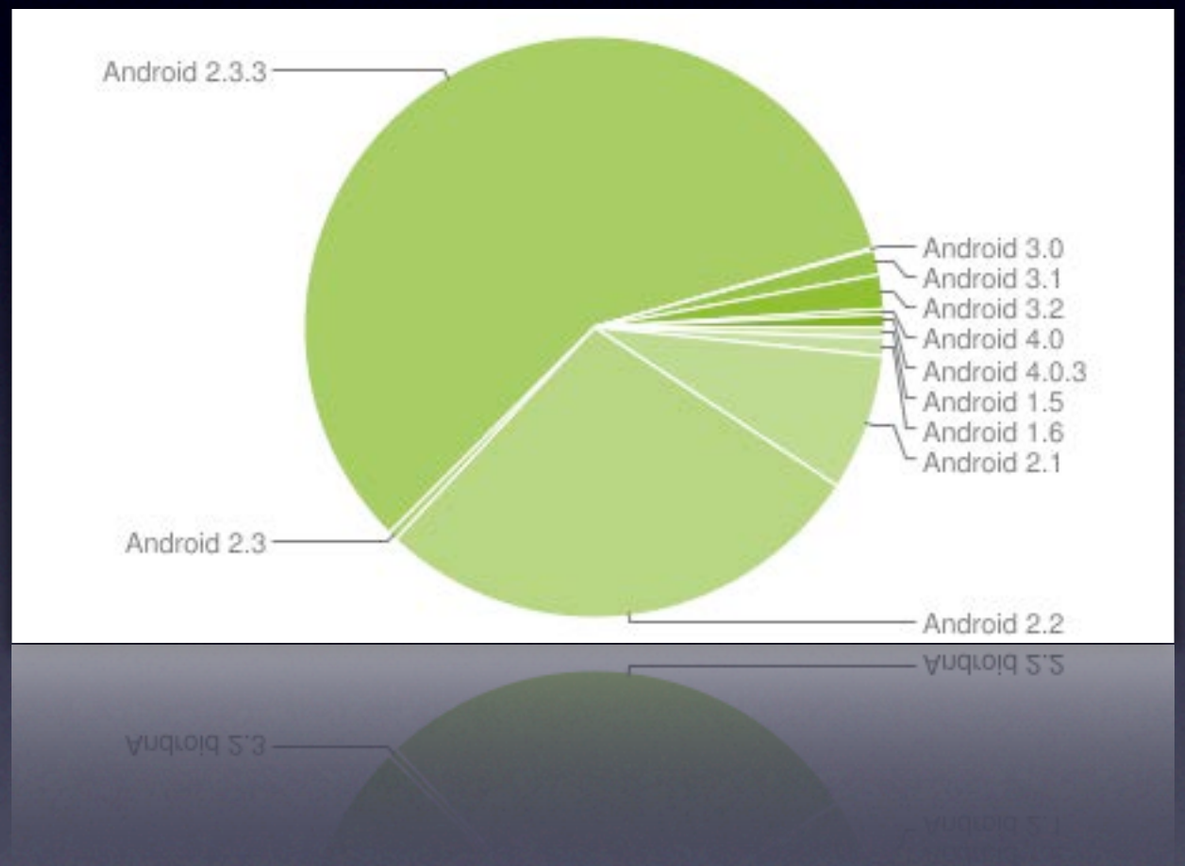
Android

- Android Inc purchased by Google in 2005
- Notionally open-source
- Leverages existing technology
  - Linux kernel — but not Linux
  - Java language — but not really Java
- Very different programming model

Android

- Several versions in current use
  - Android 2.2/2.3 - phones/cheap tablets
  - Android 3.x (Honeycomb) - Tablets only
  - Android 4.0 (Ice cream sandwich)
- Forks (e.g. Amazon Kindle Fire)

Android Versions



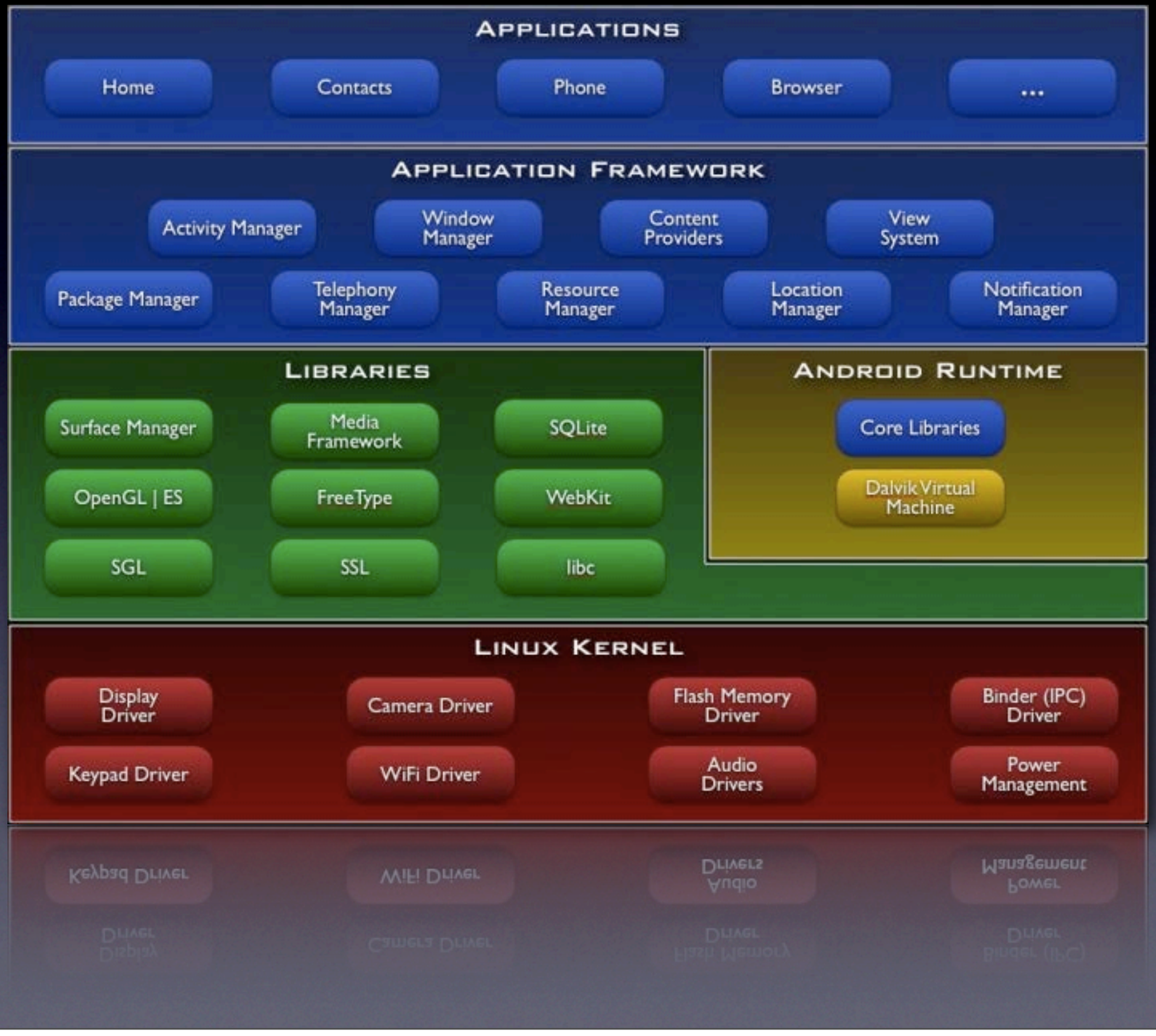
Platform	Codename	API Level	Distribution
Android 1.5	Cupcake	3	0.6%
Android 1.6	Donut	4	1.0%
Android 2.1	Eclair	7	7.6%
Android 2.2	Froyo	8	27.8%
Android 2.3- 2.3.2	Gingerbread	9	0.5%
Android 2.3.3-2.3.7		10	58.1%
Android 3.0	Honeycomb	11	0.1%
Android 3.1		12	1.4%
Android 3.2		13	1.9%
Android 4.0 - 4.0.2	Ice Cream Sandwich	14	0.3%
Android 4.0.3		15	0.7%

Compare this with iOS where 40% users were on iOS 5 within a month of its release...

Use of new APIs...

Android

- Software Stack for Mobile Devices
  - OS
  - Middleware
  - Key applications


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Dalvik

- Applications are written using Java
- But run on Google's own VM — Dalvik
  - Uses its own bytecode (DEX) format
  - Before v2.2 had no JIT
- Code compiled using standard Java tools then convert to DEX format

Android Apps

- Written in Java
- Code, data and resource files packed into a `.apk` file using `aapt`
- Apps are sandboxed
  - Own process, own VM, own UID
  - Can't access another apps files (directly)

Mobile OSes seem to have been designed very much with single user operation in mind...

Android Apps

- Component-based Programming model
- App is built up from separate components
- No `main()` method — specify the first component
- Can reuse components from other apps

Android Components

- **Activities** — UI component
- **Services** — do something in the background
- **Broadcast Receivers** — respond to broadcast messages
- **Content Providers** — make data available to other apps

Activities

- Sub-classes of `android.app.Activity`
- Presents a visual UI
- Each Activity has its own window
- UI usual specified in a separate XML file
- Need to `setContentView()` to display it
- Apps can have several Activities

Activities

- Activities can start other activities
- Forms a stack of Activities — current activity is on the top
- Form a Task
- Activities in a task move as a unit from foreground to background and vice versa

Intents

- Activities are started by sending an Intent
- Represented by an `Intent` object
- Contains the name of the action requested
- And the URI of the data to act on

Launching Activities

- Activity is launched by passing an Intent to `Context.startActivity()` or `Activity.startActivityForResult()`
- Result is returned in an Intent object that passed to your `onActivityResult()`

The Manifest

- Android needs to know about the components in your app
- .apk contains a manifest file (`AndroidManifest.xml`) that defines them
- XML syntax
- Defines each component

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.android.demo.notepad3">

    <application android:icon="@drawable/icon">

        <activity android:name=".Notepadv3"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name
                                        ="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".NoteEdit" />

    </application>
</manifest>
```

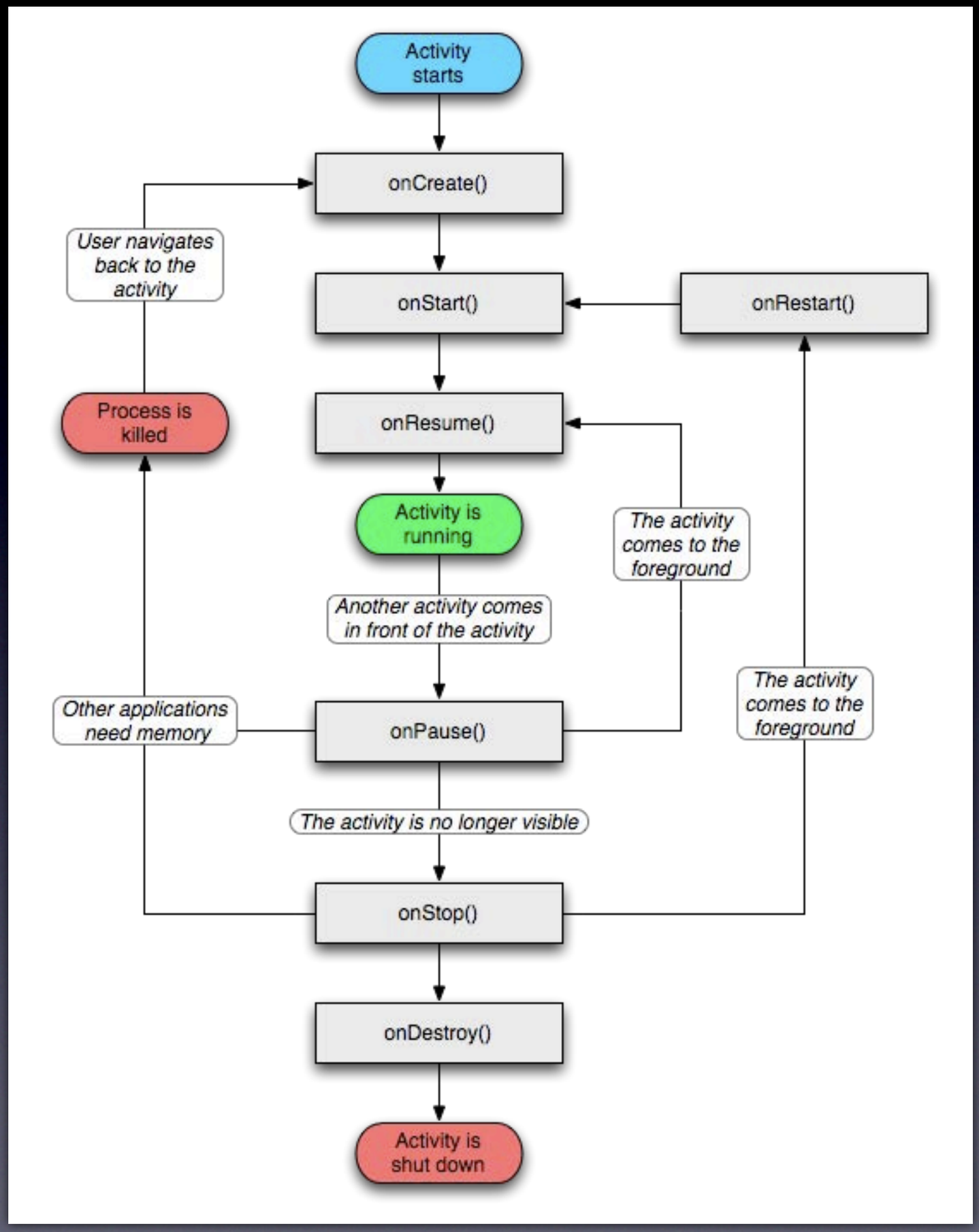
Use the @drawable @string notation to refer to resources

Intent filters

- `Intent`s can either explicitly name a target
- Or android can try and find the best match
- These are specified by Intent filters in the manifest
- These specify the category, action and data that the Activity can handle
- Root Activity specified using an intent filter

Activity Lifecycle

- Essentially three states
  - Active — in the foreground
  - Paused — still visible, but not top
  - Stopped — obscured by another activity
- If paused or stopped, the system can drop the `Activity` from memory


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Activity Objects

- Memory is limited on mobile devices
- OS needs to manage its memory differently to a computer
- Java Object representing an Activity can be destroyed, while the app is notionally running
- Need to support this in our program

Saving State

- Shouldn't rely on an `Activity` storing state
- If you need to keep it, save it
- Before `onPause()` is called, Android will call `onSaveInstanceState()`
- This allows you to save any state into a `Bundle` object

Restoring State

- When the Activity is recreated, the Bundle is passed to `onCreate()`
- And `onRestoreInstanceState()`
- Giving the Activity chance to restore its state

Services

- Subclass of `android.app.Service`
- No UI
- Run in background for an indefinite period of time
- Still runs on the main thread of the app
- So might want to start a separate thread to avoid slowing UI

Broadcast Receiver

- Responds to broadcast announcements
- Either from System or other apps
- No UI, but can start a new Activity
- Or alert the user using a Notification

Content Providers

- Subclasses `android.content.ContentProvider`
- Makes part of the application's data available to other apps
- Data can be stored in the FS, or in a SQLite DB etc.
- Not accessed directly, apps use a `ContentResolver` object