Android 5 Lollipop
Development Tools

Norman McEntire
norman.mcentire@servin.com
Opening Remarks

• Welcome!

• Thank you for attending!

• My promise to you

  • Provide a solid introduction to Android 5 Lollipop Development Tools

  • Android Studio
About Myself

• Norman McEntire
  • norman.mcentire@servin.com
• BS/MS Computer Engineering
  • USC - University of South Carolina
• 30+ Years Computer Engineering Experience
  • Hardware Engineering (chips, boards, systems)
  • Software Engineering (drivers, systems software, mobile apps)
• Current Software Focus: Android/Java, iOS 8/ObjC/Swift, IoT, Linux
How To Take This Course

• Option 1 - Corporate Training
  
  • Contact me at norman.mcentire@servin.com to schedule this Android course at your corporation

• Option 2 - UCSD Extension Course
  
My Assumptions About You

• You are in one of two broadly defined groups

  • Group 1. **Experienced** Android Software Developer
    • You have been using **Eclipse ADT**

  • Group 2. **New** to Android Software Development
    • No experience with Eclipse ADT

• **Common** to both groups: **Learn Android Studio**!
Agenda

• Android 5 Development Tool Options

• Migrating from Eclipse ADT to Android Studio

• Using Android Studio

• Using Android Studio Tools
Android 5 Development Tool Options
### Android 5 Development Tools Options

- You have three major options

- **Option 1. Start Using** Android Studio
  - The official Android 5 IDE

- **Option 2. Continue to use** Eclipse ADT for now
  - Migrate to Android Studio when you can

- **Option 3. Use the** command-line (no IDE)
Why Android Studio
System Requirements

- **Linux** (e.g. Ubuntu 12.04)
  - GNOME or KDE, glibc 2.11 or higher
  - JDK 7

- **Mac** OS X
  - 10.8.5 or higher
  - JDK 7
  - NOTE: Run Android Studio with JDK 1.6 for optimized rendering, but use JDK 7 for building Android code

- **Windows**
  - Windows 7, 8
  - JDK 7
## Android Studio Downloads

![All Android Studio Packages](image)

### All Android Studio Packages

Select a specific Android Studio package for your platform. Also see the [Android Studio release notes](#).

<table>
<thead>
<tr>
<th>Platform</th>
<th>Package</th>
<th>Size</th>
<th>SHA-1 Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>android-studio-bundle-135.1641136.exe</td>
<td>868344232 bytes</td>
<td>9c1c8ea6aa17fb74e0593c62fd48ee62a8950be7</td>
</tr>
<tr>
<td></td>
<td>(Recommended)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>android-studio-ide-135.1641136.exe (No SDK tools included)</td>
<td>260272840 bytes</td>
<td>464d1c5497ab3d1bdef441365791ab36c89cd5ae</td>
</tr>
<tr>
<td></td>
<td>android-studio-ide-135.1641136-windows.zip</td>
<td>246249059 bytes</td>
<td>6d6856aca83f6ff747ca40b10f70edfbbcccd91c</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>android-studio-ide-1641136.dmg</td>
<td>245729073 bytes</td>
<td>49506ba2cf6b56be4f7d07e6a00c4ec3ba2249d5</td>
</tr>
<tr>
<td>Linux</td>
<td>android-studio-ide-135.1641136-linux.zip</td>
<td>243917559 bytes</td>
<td>7c8f2d0cec21b98984cda45ab5a25f26d67f23a</td>
</tr>
</tbody>
</table>
Eclipse IDE and Android Studio IDE

• The **Good News** About **Android Studio**
  
  • It is **highly integrated** with the **Android Java** build environment
    
    • Example 1: **GUI layout much easier** (see multiple layouts at same time)
    
    • Example 2: **Build Variants** (build “free” and “paid” version of app at same time)

• The **Bad News for Eclipse Users**: You may be using Eclipse for other software development projects (Android, C/C++, PHP, etc) and now you have “yet another IDE” to learn
Exporting Your Eclipse Project for Import to Android Studio
Steps To Migrate Your Eclipse Project To Android Studio

• Step 1 [In Eclipse]. Export your project

• Step 2 [In Studio]. Close all open projects

• Step 3 [In Studio]. Select Import Non-Android Studio Project
To Create an Archive File In Eclipse
Archive File In Eclipse

- Archive file
  - Export resources to an archive file on the local file system.
  - To archive file: /Users/nmcentre/Downloads/HelloAndroid.zip

- Filter Types
- Select All
- Deselect All

- Options
  - Save in zip format
  - Create directory structure for files
  - Compress the contents of the file

- Finish
Migrating From Eclipse ADT to Android Studio
Migrating To Android Studio

Migrating to Android Studio

If you have been using Eclipse with ADT, be aware that Android Studio is now the official IDE for Android, so you should migrate to Android Studio to receive all the latest IDE updates.
Steps To Migrate Your Eclipse Project To Android Studio

• Step 1 [In **Eclipse**]. **Export** your project

• Step 2 [In **Studio**]. Close all open projects

• Step 3 [In **Studio**]. Select **Import Non-Android Studio Project**
Demo
Migrating from Eclipse ADT Project to Android Studio
Android Studio 1.1 Startup

```
nextView.setText("Count = " + count);
return true;

@Override
public void onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    // Add items to the action bar if it is
    // not null
    
    Android Studio 1.1 Preview 1
    Build AI-35.1681273, built on January 14, 2015
    
    JRE: 1.6.0_65-b14-466.1-11M4716 x86_64
    JVM: Java HotSpot(TM) 64-Bit Server VM by Apple Inc.
```
Android Studio Startup
Import Non-Android Studio Project
Select Project To Import
Cannot Directly Import From Zip File
Unzip File, Then Import
Import Destination Directory

Importing a project creates a full copy of the project and does not alter the original Eclipse project.

Import Destination Directory:
/Users/nmcentire/AndroidStudioProjects/HelloAndroid5

Next
Import Project from Eclipse ADT

The ADT project importer can identify some .jar files and even whole source copies of libraries, and replace them with Gradle dependencies. However, it cannot figure out which exact version of the library to use, so it will use the latest. If your project needs to be adjusted to compile with the latest library, you can either import the project again and disable the following options, or better yet, update your project.

- Replace jars with dependencies, when possible
- Replace library sources with dependencies, when possible

Other Import Options:
- Create Gradle-style (camelCase) module names

Finish
Import In Progress

Note: Notice use of Gradle to build project.
Results of Import

Ignored Files:
The following files were *not* copied into the new Gradle project; you should evaluate whether these are still needed in your project and if so manually move them:
* ic_launcher-web.png
* proguard-project.txt

Replaced Jars with Dependencies:

Moved Files:
Android Gradle projects use a different directory structure than ADT Eclipse projects. Here's how the projects were restructured:
* AndroidManifest.xml => app/src/main/AndroidManifest.xml
* assets/ => app/src/main/assets/
* res/ => app/src/main/res/
* src/ => app/src/main/java/

Next Steps:
You can now build the project. The Gradle project needs network connectivity to download dependencies.
Build and Run In Android Studio
Device Chooser
Android DDMS
Screen Capture
Captured Screen

Count = 5
Summary

• Built and Ran Android 5 App using Eclipse

• Exported from Eclipse to Android Studio

• Built and Ran Android 5 App using Android Studio
Demo

Import Eclipse ADT Project that has .jar files in the project
Eclipse ADT Project with .jar file (HelloJar.jar)
Export Project from Eclipse (Select Archive File)
Archive File

(HelloAndroid5WithJar)
Import Into Android Studio
Select Directory

![Select Directory window](image)
Destination Directory

![Import Project from ADT (Eclipse Android) window]

Importing a project creates a full copy of the project and does not alter the original Eclipse project.

Import Destination Directory:

/Users/nmcentire/AndroidStudioProjects/HelloAndroid5WithJar
“ADT project importer can identify some .jar files”
Gradle Configuration
build.gradle(Module: app)

```groovy
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

defaultConfig {
    applicationId "com.example.helloandroid5withjar"
    minSdkVersion 19
    targetSdkVersion 21
}

buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'),
    }
}

dependencies {
    compile 'com.android.support:support-v4:21.0.3'
    compile files('libs/HelloJar.jar')
}
```
Moved Files During Import

Moved Files:

Android Gradle projects use a different directory structure than ADT Eclipse projects. Here's how the projects were restructured:

* AndroidManifest.xml => app/src/main/AndroidManifest.xml
* assets/ => app/src/main/assets/
* libs/HelloJar.jar => app/libs/HelloJar.jar
* res/ => app/src/main/res/
* src/ => app/src/main/java/
Results Shown In Android Studio
Using Android Studio
Android Studio Startup

Welcome to Android Studio

- Start a new Android Studio project
- Open an existing Android Studio project
- Import an Android code sample
- Check out project from Version Control
- Import Non-Android Studio project
- Configure
- Docs and How-Tos
New Project

Configure your new project

- **Application name:** HelloAndroidSDemo2
- **Company Domain:** servin.com
- **Package name:** com.servin.helloandroidSDemo2

**Project location:** /Users/nmcentre/AndroidStudioProjects/HelloAndroidSDemo2

Next
Select Form Factors

![Select form factors screen from Android Studio](image)

- **Phone and Tablet**
  - Minimum SDK: API 19: Android 4.4 (KitKat)
  - Lower API levels target more devices, but have fewer features available. By targeting API 19 and later, your app will run on approximately 24.5% of the devices that are active on the Google Play Store. Help me choose.

- **TV**
  - Minimum SDK: API 21: Android 5.0 (Lollipop)

- **Wear**
  - Minimum SDK: API 21: Android 5.0 (Lollipop)

- **Glass (Not Installed)**
  - Minimum SDK: 

  ![Next button](image)
Add Activity
Options for New File (Use Defaults)
Getting Project Ready
(Takes a few moments)
Project Ready for Editing
Editing XML Layout

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello world!" />
</RelativeLayout>
```
Editing MainActivity.java

```java
public class MainActivity extends ActionBarActivity {

    TextView textView;
    int count;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textView = (TextView) findViewById(R.id.textView1);
        textView.setText("Count = 0");
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            count++;
            textView.setText("Count = " + count);
        }
        return super.onTouchEvent(event);
    }
}
```
Tracking status of the build
Device Chooser

Choose a running device

<table>
<thead>
<tr>
<th>Device</th>
<th>Serial Number</th>
<th>State</th>
<th>C...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung SAMSUNG-SGH-I747 Android 4 094d52be</td>
<td></td>
<td>Online</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Launch emulator

Android virtual device: Nexus 5 API 21 x86

Use same device for future launches

[OK]
Screen Capture

HelloAndroid5Demo2

Count = 6
Android Studio Skills
Android Studio Skill
Code Completion

• Just start typing and Android Studio will show you choices

• Just press **ENTER** to complete a given choice

• Also press **TAB** go to next parameter

• Example

  • Toast.makeText(this,"hi",Toast.LENGTH_LONG).show()
Android Studio Skill

SmartType Code Completion

• When you get to certain locations in your code, press Shift+Control+Spacebar to see choices based on context

• Example

  • StringBuffer sb = new Shift+Control+SpaceBar

  • Calendar now = new GregorianCalendar(Shift+Control+SpaceBar)
Android Studio Skill

Navigate in Current File

• To navigate in the current file

  • Option 1 [Menu]. **Navigate, File Structure**
    • Then select member you want to navigate to

  • Option 2 [Kbd]. **Command+F12**
    • Then select member you want to navigate to
Android Studio Skill

Quickly View Definition

• To view a quick definition of a class

• Step 1. Highlight a given class name

• Step 2 [Menu]. View, Quick Definition
Android Studio Skill
Quickly View Documentation

• To quickly view documentation of a class
  • Step 1. Highlight a given class name
  • Step 2.
    • Option 1 [Menu]. View, Quick Documentation
    • Option 2 [Kbd]. F1
Android Studio Skill

Navigate to Declaration

• To navigate to the declaration of a class, variable, or method
  
  • Step 1. Position cursor on name
  
  • Step 2.
    
    • Option 1 [Menu]. Navigate, Declaration
    
    • Option 2 [Kbd]. Command+b
Android Studio Skill

Refactor a Name

• To refactor a name of a class, method, or variable

• Step 1. Position cursor on name

• Step 2.
  • Option 1 [Menu]. Refactor, Rename
  • Option 2 [Kbd]. Shift+F6
Android Studio Skill

Display Override Methods

• To display list of methods you can override in the base class
  • Option 1 [Menu]. Code, Override Methods
  • Option 2 [Kbd]. Control+o
  • NOTE: You can also “just start typing the name”
Android Studio Skill
Display Methods of Interface

• To display list of methods you can implement in an interface

• Option 1 [Menu]. Code, Implement Methods

• Option 2 [Kbd]. Control+i
Android Studio Skill
Add New File To Project

• To add a new file to your project

• Step 1. Click on “folder” where you want to create the file (e.g. java, res, etc)

• Option 1 [Menu]. File, New

• Option 2 [MacKbd]. Command+n

• Option 2 [WinKbd]. Alt+Insert
Results of Adding Demo Class

```
package com.servin.helloandroid5demo2;

/**
 * Created by norman.mcentire@servin.com on
 *
 * public class Demo {
 * }
```
Android Studio Skill
Generate Code

• To generate code for a class (e.g. constructor, getter/setter, etc.)

• Step 1. Click on location where you want the code generated

• Option 1 [Menu]. Code, Generate

• Option 2 [MacKbd]. Command+n

• Option 2 [WinKbd]. Alt+Insert
Results of Code Generation

```java
/**
 * Created by norman.mcentire@servin.com on 1/16/15.
 */
public class Demo {

    private String name;
    private int number;

    public Demo() {
    }

    public Demo(int number, String name) {
        this.number = number;
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```
Using Android Studio Templates
No Activity
No Activity
New Project
Select Form Factor

Select the form factors your app will run on

Different platforms require separate SDKs

- **Phone and Tablet**
  - Minimum SDK: API 19: Android 4.4 (KitKat)
  - Lower API levels target more devices, but have fewer features available. By targeting API 19 and later, your app will run on approximately 24.5% of the devices that are active on the Google Play Store. Help me choose.

- **TV**
  - Minimum SDK: API 21: Android 5.0 (Lollipop)

- **Wear**
  - Minimum SDK: API 21: Android 5.0 (Lollipop)

- **Glass (Not Installed)**
  - Minimum SDK: 

[Next button highlighted]
(Optional)
Add Activity
Result
AndroidManifest.xml

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.servin.helloandroid5noactivity">

    <application android:allowBackup="true" android:label="HelloAndroid5NoActivity"
        android:icon="@drawable/ic_launcher" android:theme="@style/AppTheme">

    </application>

</manifest>
```
Blank Activity
Blank Activity
Blank Activity
AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.servin.helloandroid5blankactivity">
    <application>
        <activity
            android:name="MainActivity"
            android:label="HelloAndroid5BlankActivity"
            android:theme="@style/AppTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```
package com.servin.helloandroid5blankactivity;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <TextView android:text="@string/hello_world" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
Blank Activity with Fragment
Blank Activity
AndroidManifest.xml
public class MainActivity extends ActionBarActivity {

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    if (savedInstanceState == null) {
        getSupportFragmentManager().beginTransaction().
            add(R.id.container, new PlaceholderFragment()).
            commit();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here.
    // The action bar Howe button is automatically handled.
    // long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}
/**
 * A placeholder fragment containing a simple view.
 */
public static class PlaceholderFragment extends Fragment {
    
    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_main, container, false);
        return rootView;
    }
}
activity_main.xml
fragment_activity_main.xml

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity$PlaceholderFragment">

    <TextView android:text="@string/hello_world" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```
Fullscreen Activity
Fullscreen Activity
Result
Google Maps Activity
Google Maps Activity
Google Play Services Activity
Google Play Services Activity
Login Activity
Login Activity
Master/Detail Flow
Master/Detail Flow
Navigation Draw Activity
Navigation Draw Activity
Settings Activity
Settings Activity
Tabbed Activity
Tabbed Activity
Gradle Build Scripts
Gradle Build Scripts
`build.grade` (Project)

```groovy
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.0.0'
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
    }
}
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "com.servin.helloandroid5noactivity"
        minSdkVersion 19
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }

    dependencies {
        compile fileTree(dir: 'libs', include: ['*.jar'])
        compile 'com.android.support:appcompat-v7:21.0.3'
    }
}
gradle-wrapper.properties

#Wed Apr 10 15:27:10 PDT 2013

distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https://services.gradle.org/distributions/gradle-2.2.1-all.zip
gradle-wrapper.properties

```properties
#Wed Apr 10 15:27:10 PDT 2013
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https://services.gradle.org/distributions/gradle-2.2.1-all.zip
```
Android Studio
DDMS Options
Screen Capture
Screen Recorder

Screen Recorder can record the device's display for a maximum of 3 minutes. By default, it records at the device's native resolution or at 720p at a 4 Mbps bitrate. You can customize these options below. Leave empty to use defaults.

- Bit Rate (Mpbs): [4]
- Resolution (width x height, in px): [ ]

Options: Cancel, Start Recording
System Information
System Information
Activity Manager State
System Information

Package Information

Verifiers:
- Required: com.android.vending (uid=10067)

Libraries:
- sws -> (jar) /system/framework/sws.jar
- sechardware -> (jar) /system/framework/sechardware.jar
- com.android.future.usb.accessory -> (jar) /system/framework/com.android.future.usb.accessory.jar
- allshare -> (jar) /system/framework/allshare.jar
- com.sec.android.mdm -> (jar) /system/framework/sec_edm.jar
- android.test.runner -> (jar) /system/framework/android.test.runner.jar
- com.sec.android.visualeffect -> (jar) /system/framework/com.sec.android.visualeffect.jar
- seccamera -> (jar) /system/framework/seccamera.jar
- videowall -> (jar) /system/framework/videowall.jar
- secvision -> (jar) /system/framework/secvision.jar
- libvmanagerjar -> (jar) /system/framework/libvmanagerjar.jar
- sec_platform_library -> (jar) /system/framework/sec_platform_library.jar
- com.google.android.media.effects -> (jar) /system/framework/com.google.android.media.effects.jar
- secmediarecorder -> (jar) /system/framework/secmediarecorder.jar
- com.broadcom.bt -> (jar) /system/framework/com.broadcom.bt.jar
- com.policydm.features -> (jar) /system/framework/com.policydm.features.jar
- com.android.location.provider -> (jar) /system/framework/com.android.location.provider.jar
- com.sec.smartcard.auth -> (jar) /system/framework/secsmartcard.jar
- org.simalliance.openmobileapi -> (jar) /system/framework/org.simalliance.openmobileapi.jar
- gcom.digitalpen.util -> (jar) /system/framework/gcom.digitalpen.util.jar
- javax.obex -> (jar) /system/framework/javax.obex.jar
System Information
Memory Usage

Applications Memory Usage (kB):
Uptime: 3239834 Realtime: 4021446

Total PSS by process:
- 76793 kB: com.sec.android.app.launcher (pid 1116 / activities)
- 48778 kB: system (pid 692)
- 47899 kB: com.android.systemui (pid 971)
- 33022 kB: com.sec.android.inputmethod (pid 1919)
- 27438 kB: com.google.android.apps.plus (pid 3643)
- 20363 kB: com.google.android.gms (pid 1619)
- 20200 kB: com.google.android.googlequicksearchbox:search (pid 1805)
- 19322 kB: com.android.vending (pid 3423)
- 15996 kB: com.android.phone (pid 1091)
- 15779 kB: com.google.process.location (pid 1406)
- 15144 kB: com.google.process.gapps (pid 1593)
- 14948 kB: com.sec.android.gallery3d (pid 3916)
- 14635 kB: zygote (pid 174)
- 13075 kB: com.google.android.apps.magazines (pid 8301)
- 12427 kB: com.sec.spp.push (pid 3539)
- 11589 kB: com.android.contacts (pid 2096)
- 9640 kB: com.sec.android.inputmethod:ACService (pid 1977)
- 9441 kB: android.process.media (pid 3526)
- 9225 kB: com.samsung.android.MtpApplication (pid 7590)
- 8865 kB: media,server (pid 176)
- 8862 kB: com.google.android.talk (pid 8752)
- 8520 kB: com.android.email (pid 7875)
System Information
Memory Usage Over Time

AGGRADED OVER LAST 24 HOURS:
* system / 1000:
  TOTAL: 100% (44MB-47MB-53MB/40MB-43MB-48MB over 28)
  Persistent: 100% (44MB-47MB-53MB/40MB-43MB-48MB over 28)
* com.android.systemui / 00179:
  TOTAL: 100% (41MB-45MB-59MB/35MB-39MB-52MB over 28)
  Persistent: 100% (41MB-45MB-59MB/35MB-39MB-52MB over 28)
  Imp Fg: 0.03%
* com.sec.knox.eventmanager / 1000:
  TOTAL: 100% (2.3MB-2.6MB-2.8MB/1.6MB-1.7MB-1.7MB over 28)
  Persistent: 100% (2.3MB-2.6MB-2.8MB/1.6MB-1.7MB-1.7MB over 28)
  Service: 0.16%
* com.android.phone / 1001:
  TOTAL: 100% (16MB-16MB-16MB/14MB-14MB-15MB over 28)
  Persistent: 100% (16MB-16MB-16MB/14MB-14MB-15MB over 28)
* com.android.nfc / 1027:
  TOTAL: 100% (4.8MB-6.2MB-7.5MB/3.9MB-5.1MB-6.2MB over 28)
  Persistent: 100% (4.8MB-6.2MB-7.5MB/3.9MB-5.1MB-6.2MB over 28)
* org.simalliance.openmobileapi.service:remote / 1101:
  TOTAL: 100% (2.4MB-2.6MB-2.7MB/1.8MB-1.9MB-1.9MB over 31)
  Imp Fg: 100% (2.4MB-2.6MB-2.7MB/1.8MB-1.9MB-1.9MB over 31)
* android.process.acore / 009:
  TOTAL: 100% (8.1MB-8.7MB-9.4MB/7.5MB-8.0MB-8.6MB over 28)
  Persistent: 100% (8.1MB-8.7MB-9.4MB/7.5MB-8.0MB-8.6MB over 28)
  Imp Fg: 0.07%
  [Cached]: 0.03%
System Information

Graphics State

Applications Graphics Acceleration Info:
Uptime: 3443082 Realtime: 4224693

** Graphics info for pid 8262 [com.servin.helloandroid5fullscreenactivity] **

Recent DisplayList operations
  DrawText
  RestoreToCount
  RestoreToCount
  DrawRect
  DrawRect
  multiDraw
    DrawText
    DrawBitmap
  DrawDisplayList
  DrawDisplayList
    DrawDisplayList
    DrawRect
    DrawDisplayList
    Save
    ClipRect
    Translate
    DrawText
    DrawText
Memory Monitor
Android Studio
Dynamic Layout
Dynamic Layout

- Three Options
- Design View
  - Drag from Palette to Preview or Component Tree
- Text View
- Preview
Dynamic Design Preview
Choices

• You can preview all of the following
  • Different Device Images
  • Different Display Densities
  • Different UI Modes
  • Different Android Versions
Motivation For Dynamic Layout

Platform Versions

This section provides data about the relative number of devices running a given version of the Android platform.

For information about how to target your application to devices based on platform version, read Supporting Different Platform Versions.

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Froyo</td>
<td>8</td>
<td>0.4%</td>
</tr>
<tr>
<td>2.3.3-2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>7.8%</td>
</tr>
<tr>
<td>4.0.3-4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>6.7%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>19.2%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>20.3%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>6.5%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>39.1%</td>
</tr>
</tbody>
</table>

Data collected during a 7-day period ending on January 5, 2015.
Any versions with less than 0.1% distribution are not shown.
Design View
Text View
Dynamic Preview based on API Version
Dynamic Preview based on Screen Size

![Dynamic Preview of various smartphone screen sizes](image)
Keeping Android Studio Up-To-Date
Four Update Channels

Update channels

Android Studio provides four update channels to keep Android Studio up-to-date based on your code-level preference:

- **Canary channel**: Canary builds provide bleeding edge releases, updated about weekly. While these builds do get tested, they are still subject to bugs, as we want people to see what’s new as soon as possible. This is not recommended for production.
- **Dev channel**: Dev builds are hand-picked older canary builds that survived the test of time. They are updated roughly bi-weekly or monthly.
- **Beta channel**: Beta builds are used for beta-quality releases before a production release.
- **Stable channel**: Used for stable, production-ready versions.

By default, Android Studio uses the Stable channel. Use **File > Settings > Updates** to change your channel setting.
Easy Access to Android Code Samples via GitHub
Open Project From GitHub
Code Samples
Code Samples from GitHub Description
Code Samples from GitHub
Preview
Code Samples from GitHub

Import Sample

Sample Setup
Provide information about your project

Application name: Camera2Basic
GitHub URL: https://github.com/googlesamples/android-Camera2Basic/
Project location: /Users/nmcentire/AndroidStudioProjects/Camera2Basic

Finish
Possible Error When Importing Sample Projects
Installing Build Tools
License Agreement
Installing Build Tools
Installing Requested Components
Android Device Monitor
Starting Android Device Monitor From Android Studio
Android Device Monitor
Android SDK Manager
Starting **Android SDK Manager** From Android Studio
Android SDK Manager
Android Virtual Device (AVD) Manager
Starting **Android Virtual Device (AVD) Manager** From Android Studio
Android Virtual Device (AVD) Manager - Create Device
Android Virtual Device (AVD) Manager - Choose Device
Android Virtual Device (AVD) Manager - System Image
Android Virtual Device (AVD) Manager - Verify Configuration
Android Virtual Device (AVD) Manager - Your Virtual Device
Android Virtual Device (AVD) Manager - Startup

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Resolution</th>
<th>API</th>
<th>Target</th>
<th>CPU/ABI</th>
<th>Size on Disk</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nexus One API 21</td>
<td>480 × 800: hdpi</td>
<td>21</td>
<td>Google APIs</td>
<td>x86</td>
<td>650 MB</td>
<td></td>
</tr>
</tbody>
</table>
Android Virtual Device (AVD) Manager - Startup
Code Running on Emulator
Android Build System

• Previous Android Build System
  • Eclipse ADT with ANT

• New Build System
  • Android Studio with Gradle
Two Options for the Build System

• Option 1.
  • Use features of Android Studio without any focus on the underlying build system
  • Create single APK

• Option 2.
  • Customize the build system
  • Create multiple APKs using same project
  • Reuse code and resources across project sets
Gradle Build Script

- The build files are called build.gradle
- They are plain text files
- They use the Groovy syntax to configure the build
- With elements provided by the Android plugin for Gradle
- In most cases, you only need to edit build files at the module level (see next slide)
Normally only edit build.gradle at Module Level
Key Point:
The Android Build System enables you to customize build WITHOUT modification to app source files!
The "apply" line includes Android-Specific Elements

```groovy
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "com.servin.helloandroid5"
        minSdkVersion 19
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }

    dependencies {
        compile fileTree(dir: 'libs', include: ['*.jar'])
        compile 'com.android.support:appcompat-v7:21.0.3'
    }
}
The android Element configures all Android-specific build options

```groovy
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "com.servin.helloandroid5"
        minSdkVersion 19
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }

    dependencies {
        compile fileTree(dir: 'libs', include: ['*.jar'])
        compile 'com.android.support:appcompat-v7:21.0.3'
    }
}
compileSdkVersion 21
buildToolsVersion
(always higher than compileSdkVersion)
defaultConfig
(override settings in AndroidManifest.xml)
(apply to all build variants)
buildTypes
(how to build and package app)

Note: By default, build system defines two types: **debug** and **release**
dependencies

```java
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "com.servin.helloandroid5"
        minSdkVersion 19
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }

    dependencies {
        compile fileTree(dir: 'libs', include: ['*.jar'])
        compile 'com.android.support:appcompat-v7:21.0.3'
    }
```
Important!
Making changes to build system requires a sync!
(see next slide)
Sync Now

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```java
apply plugin: 'com.android.application'

android {
  compileSdkVersion 21
  buildToolsVersion "21.1.2"

  defaultConfig {
    applicationId "com.servin.helloandroid5"
    minSdkVersion 19
    targetSdkVersion 21
    versionCode 1
    versionName "1.0"
  }

  buildTypes {
    release {
      proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
  }
}
```
applicationId

- Use applicationId to uniquely identify application packages for publishing
Key Point: `applicationId` specified in `build.grade`, and NOT in `AndroidManifest.xml`
Build Variants Demo

• This demo will show how to create to build variants from a single project
  
  • A Free (Limited Featured) Version
  
  • A Paid (Fully Featured) Version
  
  • Both versions will have the same MainActivity
  
  • Each version will have a different SecondActivity
Step 1. Define Two Product Flavors

```json
productFlavors {
  free {
    applicationId "com.servin.buildsystemexample.free"
    versionName "1.0-free"
  }
  paid {
    applicationId "com.servin.buildsystemexample.paid"
    versionName "1.0-paid"
  }
}
```
Step 2. Make sure grade performs a sync.
Step 3. Select the “freeDebug”
Step 4. Add Directories for “free” Product Flavor
Step 5a. Right-Click on java, select New, Activity, Blank Activity
Step 5b. Add SecondActivity for “free” Product Flavor
Directory Structure for “free” Product Flavor
Selecting Different Product Flavors
- freeDebug and paidDebug
Running In Demo Mode
Adding a .jar file to your Android Studio Project
Step 1. Select **Project View**
Step 2. Drag/Drop .jar file into **libs** directory
Step 3. Right-click on .jar file, select **Add as Library**
Step 4. Confirm Results
Step 5. Confirm gradle

```groovy
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile files('libs/HelloJar.jar')
}
```
Step 5. Confirm In MainActivity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.i("DEMO", "onCreate");

    Demo demo = new Demo();
    Log.i("DEMO", demo.sayHello());
}
```
Thank You!