Gradle (for Android DSL) Cheat Sheet

This is for the build.gradle file for Android projects.

Level 0

Welcome, Dev. This level assumes that you know nothing about "Gradle" or you don't know why it's useful. Here's a few reasons you may want to learn about Gradle (explained in simple terms).

- It's now the main system used to help create Android apps, i.e. the part that helps with changing code to installable app.
- TODO: Add more reasons why Dev should learn "Gradle".
- TODO: Integrate some of this: http://tools.android.com/tech-docs/new-build-system/user-guide#TOC-Goals-of-the-new-Build-System

Level 1

Gradle is a build tool like Make and Rake and much more than Ant and Maven. It is used to help build code for use.

Level 2

How to publish with Gradle

- Awesome: http://meedamian.com/post/publishing-with-gradle (using the "Perfect" flow)

Third-Party Example

- Good: https://gist.github.com/cyrilmottier/8234960

How To Add Custom Fields To The Auto-Generated BuildConfig.java

```
// Access these fields in Java files by using `BuildConfig.IS_PRO`.
productFlavors {
    free {
        buildConfigField "boolean", "IS_PRO", "false"
    }
    pro {
        buildConfigField "boolean", "IS_PRO", "true"
    }
}
```

How To Use A Single Signing Keystore For All Flavors

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

android {
    compileSdkVersion 19
    buildToolsVersion "19.1"

    defaultConfig { ... }
```

```
    productFlavors { ... }

    signingConfigs {
        release {
            storeFile     file(STORE_FILE)
            storePassword STORE_PASSWORD
            keyAlias      KEY_ALIAS
            keyPassword   KEY_PASSWORD
        }
    }

    buildTypes {
        release {
            signingConfig signingConfigs.release
            ...
        }
        debug {
            applicationIdSuffix ".debug"
        }
    }
}
```

How To Use A Different Signing Keystore For Each Flavor

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

android {
    compileSdkVersion 19
    buildToolsVersion "19.1"

    defaultConfig { ... }

    productFlavors {
        free { ... }
        pro { ... }
    }

    signingConfigs {
        free {
            storeFile     file(FREE_STORE_FILE)
            storePassword FREE_STORE_PASSWORD
            keyAlias      FREE_KEY_ALIAS
            keyPassword   FREE_KEY_PASSWORD
        }
        pro {
            storeFile     file(PRO_STORE_FILE)
            storePassword PRO_STORE_PASSWORD
            keyAlias      PRO_KEY_ALIAS
            keyPassword   PRO_KEY_PASSWORD
        }

    buildTypes {
        release {
            productFlavors.free.signingConfig signingConfigs.free
            productFlavors.pro.signingConfig signingConfigs.pro
            runProguard true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-project.txt'

            // This is used to alter output directory and file name. If you don't need it
            // you can safely comment it out.
            applicationVariants.all { variant ->
                def file = variant.outputFile

                String parent = file.parent
                if (project.hasProperty('OUTPUT_DIR') && new File((String) OUTPUT_DIR).exists())
                    parent = OUTPUT_DIR

                variant.outputFile = new File(parent, (String) file.name.replace(".apk",

                        // Alter this string to change output file name.
                        "-" + defaultConfig.versionName + "-build" + defaultConfig.versionCode + ".apk"
                ))
            }
            // End your comment here.
        }
```

```
        debug {
            applicationIdSuffix ".debug"
        }
    }
}
```

Level 3

- Official Android Gradle docs, but not kept up-to-date while Android Studio is still in beta: http://tools.android.com/tech-docs/new-build-system/user-guide
- Attributes to customize: http://tools.android.com/tech-docs/new-build-system/user-guide#TOC-Basic-Build-Customization
- TODO: Possibly add more command line stuff. Or, on another level?

Level 4

- More info: http://tools.android.com/tech-docs/new-build-system/user-guide/manifest-merger

Sample AndroidManifest.xml

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

General layout of /app/build.gradle

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

android {
    compileSdkVersion 19
    buildToolsVersion "19.1"

    defaultConfig { ... } // This is the base properties for all flavors.
    signingConfigs { ... } // Must go before buildTypes so that these properties can be accessed by buildTypes.
    buildTypes { ... }
    productFlavors { ... } // This supports the same properties as in defaultConfig, and overrides them.
}
```

Sample app/build.gradle (single package)

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

android {
    compileSdkVersion 19
    buildToolsVersion "19.1"

    defaultConfig {
        applicationId "com.example.my.app"
        minSdkVersion 15
        targetSdkVersion 19
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            runProguard true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-project.txt'
        }
    }
}
```

```
dependencies {
    compile 'com.android.support:support-v4:20.+'
    compile files('libs/GoogleAdMobAdsSdk-6.4.1.jar')
    compile files('libs/libGoogleAnalyticsServices.jar')
}
```

Sample app/build.gradle (multiple packages)

```
android {
    ...
    productFlavors {
        pro {
            applicationId = "com.example.my.pkg.pro"
        }
        free {
            applicationId = "com.example.my.pkg.free"
        }
    }

    buildTypes {
        debug {
            applicationIdSuffix ".debug"
        }
    }
    ...
}
```

Sample custom variables

- There is already an implicit placeholder `${applicationId}` that holds the build.gradle application id value, which can be used something like `android:attribute="com.example.${applicationId}.foo"` in the AndroidManifest.

```
// In build.gradle
productFlavors {
    pro {
        applicationId = "com.example.my.pkg.pro"
        manifestPlaceholders = [
            backupApiKey: "myPaidBackupApiKeyValue"
        ]
    }
    free {
        applicationId = "com.example.my.pkg.free"
        manifestPlaceholders = [
            backupApiKey: "myFreeBackupApiKeyValue"
        ]
    }
}

// In AndroidManifest.xml
<meta-data
    android:name="com.google.android.backup.api_key"
    android:value="${backupApiKey}" />
```

Gradle Snippets

How to remove unused resources

```
android {
    ...
    buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
            ...
        }
    }
}
```

Android Manifest Notes

- More info: http://tools.android.com/tech-docs/new-build-system/applicationid-vs-packagename
- Three types of manifest files, in priority order: (These get merged into a single resulting app manifest)
    i. Product flavors (like pro/free) and build types (like debug/alpha/beta) specific manifest files
    ii. Main manifest file for the application
    iii. Library manifest files (4. Injectable values for placeholders and XML generation)
- Some elements, like "Activity", must have a key (typically the `android:name` attribute) since there can be multiple present in a single AndroidManifest.xml. Other elements, like "Application", do not require a key since there can be only one.
- Each flavor/build of app can have different package names, but the auto-generated R class should have the same package name across the different builds. It is decoupled the following way:
    - The "application id" in build.gradle is used for building builds and for Google Play.
    - The "package" attribute in the AndroidManifest is the package name used within code and R class.
- There is a GUI way to change build.gradle as needed, via the Project Structure dialog.

Sources Read

- Awesome: http://meedamian.com/post/publishing-with-gradle
- Awesome: http://blog.robustastudio.com/mobile-development/android/building-multiple-editions-of-android-app-gradle/
- Good: http://tools.android.com/tech-docs/new-build-system/resource-merging
- Good: http://developer.android.com/sdk/installing/studio-build.html
- Good: http://tools.android.com/tech-docs/new-build-system/user-guide/manifest-merger
- Good: http://tools.android.com/tech-docs/new-build-system/tips
- Main: http://tools.android.com/tech-docs/new-build-system
- Good: http://engineering.meetme.com/2014/07/android-and-gradle-migrating-to-the-new-project-structure/
- Good real/full example: https://plus.google.com/+CyrilMottier/posts/WRgB2shaeuu (which leads to https://gist.github.com/cyrilmottier/8234960)
- Okay: http://developer.android.com/google/play/publishing/multiple-apks.html
- Meh: http://tools.android.com/tech-docs/new-build-system/migrating-from-eclipse-projects
- Good, but not Gradle: http://developer.android.com/guide/topics/manifest/manifest-intro.html