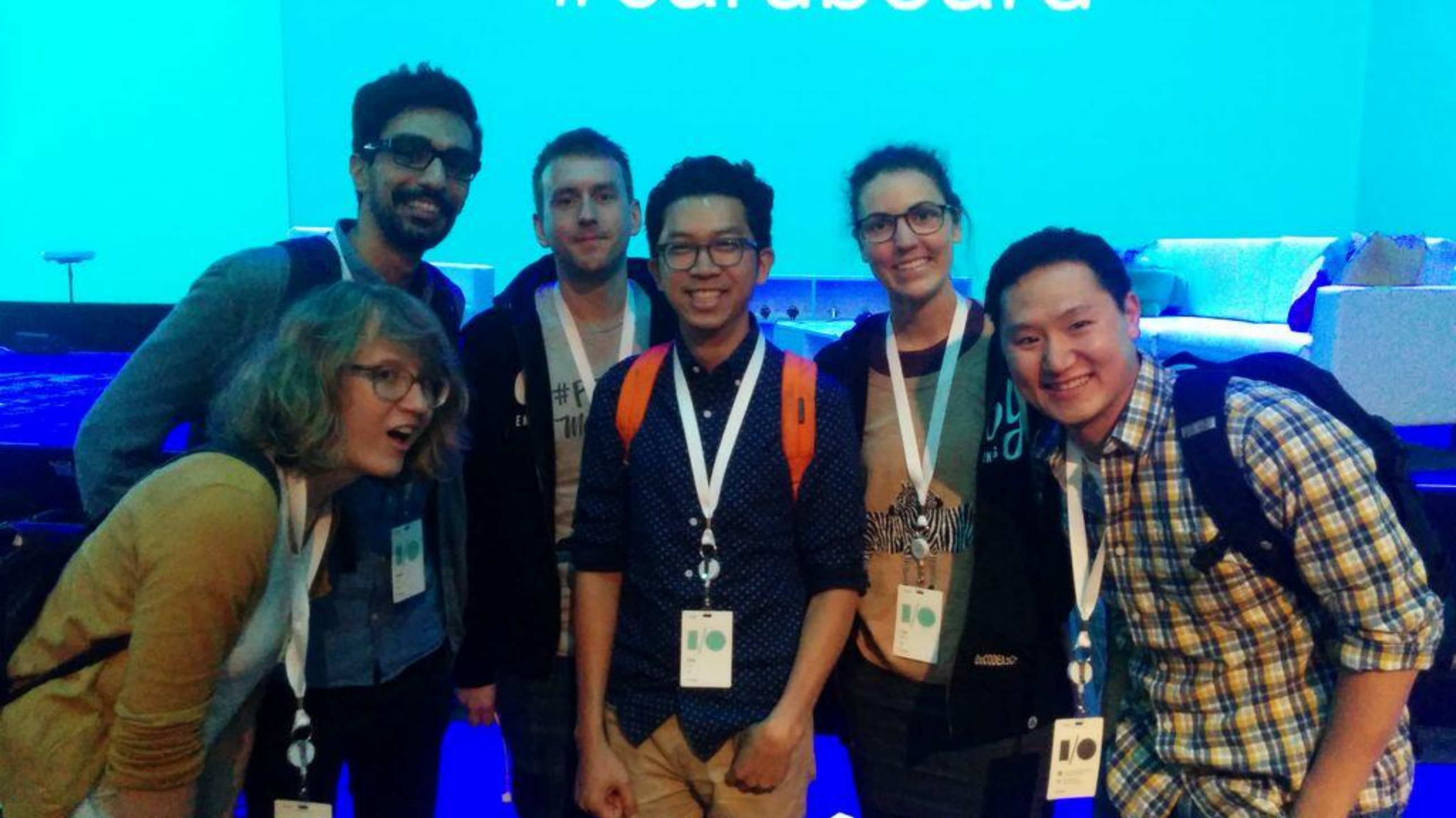
GRADLIN'
NUTS AND BOLTS

LISA NEIGUT DROIDCON NYC 21 SEPT 2014

HELLO, I'M LISA

I WORK ON THE ANDROID TEAM AT ETSY



WHY GRADLE?

WHY GRADLE?


ate Code Analyze Refactor Build Run Tools VCS Window Help
(C:\Users\Paresh\AndroidStudioProject
java
com.technotalkative.test
MainActivity
res
drawable
Studio_logo.png
drawable-hdpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
layout
menu
values
values-sw600dp
values-sw720dp-land
values-v11
values-v14
AndroidManifest.xml
ic_launcher-web.png
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/an
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:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello world!"
android:id="@+id/textView" />
<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/imageView"
android:src="@drawable/Studio_logo"
android:layout_below="@+id/textView"
android:layout_alignParentLeft="true" />
</RelativeLayout>
Preview
Nexus 4
AppTheme
MainActivity
Nexus One (3.7")
Galaxy Nexus (4.7")
TechnoTalkative.com

NUTS AND BOLTS

- Groovy
- Gradle
  - Tasks
  - Build Phases
  - Projects

THINGS NOT COVERED IN THIS TALK:

» Task dependencies

» Manipulating Files

» Build Graph Hooks

» The Android Plugin

» Multiproject Build Logic

» Plugins

GROOVY:

A dynamic, static, strongly, duck typed language that is imperative, object-oriented, functional, and scripting.
It runs on the Java Virtual Machine.

ArdentlyCrafted

in United States

GROOVY:

A dynamic, static, strongly, duck typed language that is imperative, object-oriented, functional, and scripting.
It runs on the Java Virtual Machine.

Java:

```
for (String it : new String[] { "Erin", "Jan", "Taylor" }) {
    if (it.length() >= 4) {
        System.out.println(it);
    }
}
```

Groovy:

```
["Erin", "Jan", "Taylor"].findAll{ it.size() >= 4 }.each{ println it }
```

Groovy:

```
["Erin", "Jan", "Taylor"]
```

Groovy:

```
        .findAll{ it.size() >= 4 }
```

Groovy:

```
    .each{ println it }
```

Groovy:

```
["Erin", "Jan", "Taylor"].findAll{ it.size() >= 4 }.each{ println it }
```

Java:

```
for (String it : new String[] { "Erin", "Jan", "Taylor" }) {
    if (it.length() >= 4) {
        System.out.println(it);
    }
}
```

Groovy:

```
["Erin", "Jan", "Taylor"].findAll{ it.size() >= 4 }.each{ println it }
```

GROOVY

FANCY GROOVY FEATURES

» Syntax

» Null Checks

» Dynamic Typing

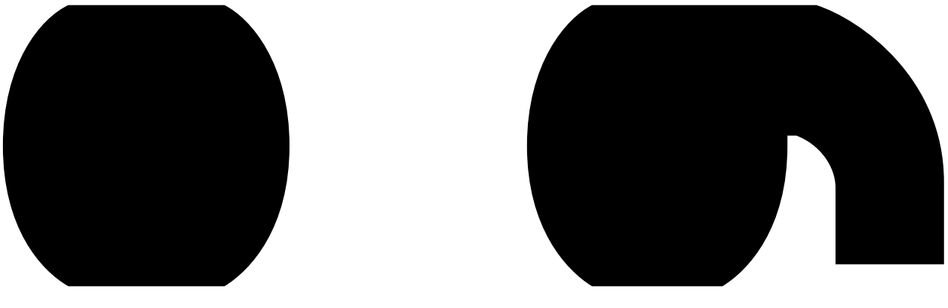
» GStrings

» Arrays

» Closures

» Objects

LET'S START WITH THE GOOD PARTS!



SYNTAX

Groovy

```
take(coffee).with(sugar, milk).and(liquor)
```

SYNTAX

Groovy

```
take(coffee).with(sugar, milk).and(liquor)

take coffee with sugar, milk and liquor
```

SYNTAX

Groovy

```
    .findAll { it.size() >= 4 }
```

SYNTAX

Groovy

```
    .findAll { it.size() >= 4 }

    .findAll({ it.size() >= 4 })
```

NULL CHECKS

Java

```
if (activity != null) {
    activity.finish();
}
```

NULL CHECKS

Java

```
if (activity != null) {
    activity.finish();
}
```

Groovy

```
activity?.finish()
```

DYNAMIC TYPING

Use the def keyword!

Groovy:

```
def variable
```

DYNAMIC TYPING

Use the def keyword!

Groovy:

```
def variable
String variable
```

GSTRINGS

Java:

```
int amount = 4563;
String yourBill = "You owe me " + amount;
```

GSTRINGS

Java:

```
int amount = 4563;
String yourBill = "You owe me " + amount;

String.format("You owe me %d", amount);
```

GSTRINGS

Java:

```
    String.format("On %s the %s of %s %s paid $%d to %s",
        dayOfWeek, day, month, payee, amountPaid, richMan);
```

GSTRINGS

Groovy:

```
def amount = 4563
def yourBill = "You owe me $amount"
```

GSTRINGS

Groovy:

```
def amount = 4563
def yourBill = "You owe me $amount"

def fancyAmount = [4,5,6,3]
def yourBill = "You owe me ${ fancyAmount.join() }"
```

GSTRINGS

Groovy:

```
def amount = 4563
def yourBill = "You owe me $amount"

def fancyAmount = [4,5,6,3]
def yourBill = "You owe me ${ fancyAmount.join() }"

def yourBill = "You owe me \${ fancyAmount.join() }"
```

GSTRINGS

Groovy:

```
String string =
    “On $dayOfWeek the $day of $month” +
    “ $payee paid \$$amountPaid to $richMan”
```

ARRAYS

Groovy:

```
["Erin", "Jan", "Taylor" ]
```

ARRAYS

```
["Erin", "Jan", "Taylor" ].size()
["Erin", "Jan", "Taylor" ].get(1)
["Erin", "Jan", "Taylor" ][1]
```

ARRAYS

```
["Erin", "Jan", "Taylor" ].size()
["Erin", "Jan", "Taylor" ].get(1)
["Erin", "Jan", "Taylor" ][1]

for (String string : [ "Erin", "Jan", "Taylor" ]) { println string.length() }
```

ARRAYS

```
["Erin", "Jan", "Taylor" ].size()
["Erin", "Jan", "Taylor" ].get(1)
["Erin", "Jan", "Taylor" ][1]

for (String string : [ "Erin", "Jan", "Taylor" ]) { println string.length() }
["Erin", "Jan", "Taylor" ].each { println it.length() }
```

CLOSURES

CLOSURES

Java:

```
int[] ints = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9};
int[] odds = new OddFinder().findOdds(ints);
```

CLOSURES

Groovy:

```
list = [1, 2, 3, 4, 5, 6, 7, 8, 9]

def odds = list.findAll { it % 2 }

assert odds == [1, 3, 5, 7, 9]
```

CLOSURES

Groovy:

```
{ it % 2 }
```

CLOSURES

```
Groovy:

\\ Abbreviated with it

 { it % 2 }

\\ Long form

  { number -> number % 2 }
```

CLOSURES

Let’s run this!

GROOVY OBJECTS

Groovy:

```
class AGroovyObject {
    String color
}

def myGroovyObject = new AGroovyObject()

myGroovyObject.setColor('blue')
assert myGroovyObject.getColor() == 'blue'

myGroovyObject.color = 'fuschia'
assert myGroovyObject.color == 'fuschia'
```

GROOVY OBJECTS

Groovy:

Let’s make a dynamic properties object!

AIN'T IT GROOVY?

» Syntax

» Null Checks

» Dynamic Typing

» GStrings

» Arrays

» Closures

» Objects

GRADLE:

Gradle is a project automation tool that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the more traditional XML form of declaring the project configuration.

GRADLE:

Java builds without XML.

STRUCTURE OF A GRADLE PROJECT

» build.gradle

» build.gradle

» settings.gradle

» build.gradle

» settings.gradle

» gradle.properties

GRADLE.PROPERTIES

```
org.gradle.daemon=true
org.gradle.java.home
# org.gradle.jvmargs
org.gradle.parallel
org.gradle.configureondemand
```

GRADLE CORE COMPONENTS

» Tasks

» Build Phases

» Projects

TASKS

The task is the core building block of a build action.

The task is the core building block of a build action.

» Build an Android Project

```
$ ./gradlew  assemble
```

The task is the core building block of a build action.

» Build an Android Project

```
$ ./gradlew  assemble
```

» Show me all the Tasks in a project

```
$ ./gradlew tasks
```

```
:tasks

------------------------------------------------------------
All tasks runnable from root project
------------------------------------------------------------

Android tasks
-------------
androidDependencies - Displays the Android dependencies of the project
signingReport - Displays the signing info for each variant

Build tasks
-----------
assemble - Assembles all variants of all applications and secondary packages.
assembleDebug - Assembles all Debug builds
assembleDebugTest - Assembles the Test build for the Debug build
assembleRelease - Assembles all Release builds
build - Assembles and tests this project.
...
```

Groovy / Gradle:

```
task hello {
    doLast {
        println "Hello DroidConNYC!"
    }
}
```

Groovy / Gradle:

```
task hello {
    doLast {
        println "Hello DroidConNYC!"
    }
}
```

Run:

```
$ ./gradlew hello
```

And we get...

```
♤ ./gradlew hello
:app:hello
Hello DroidConNYC!

BUILD SUCCESSFUL

Total time: 11.075 secs
```

Gradle:

```
♤ ./gradlew brokenHello
Hello -- Too Soon!!
:app:brokenHello UP-TO-DATE

BUILD SUCCESSFUL

Total time: 10.184 secs
```

BUILD PHASES

» Initialization

» Initialization

» Configuration

» Initialization

» Configuration*

gradle.properties

```
org.gradle.configureondemand=true
```

- Initialization
- Configuration
  - Resolves Dependency Graph

- Initialization
- Configuration
  - Resolves Dependency Graph
  - Task Execution Plan

- Initialization
- Configuration
  - Resolves Dependency Graph
  - Task Execution Plan
- Execution

```
♤ ./gradlew hello
> Configuring > 2/2 projects

:app:newHello
Hello Again!

BUILD SUCCESSFUL
```

Groovy:

```
task hello2 << {
    println "Hello Again!”
}

task brokenHello {
    println "Hello -- Too Soon!!"
}
```

CLOSURE MAGIC!

TASK ACTIONS

» Tasks are just a set of “actions”, which are closures

TASK ACTIONS

- Tasks are just a set of “actions”, which are closures
- Three ways to add an Action (closure) to a task:
  - doFirst {}
  - doLast {}
  - leftShift {} / <<

DEBUGGING TASKS

```
./gradlew <command> --stacktrace --debug --info
```

THE PROJECT

PROJECT COMPONENTS

» Tasks

» Dependencies

» Configurations

» Properties

» Plugins

PROJECT COMPONENTS

» Tasks

» Dependencies

» Configurations

» Properties

» Plugins*

DEPENDENCIES

```
dependencies {
   compile 'commons-lang:commons-lang:2.6'
   testCompile 'org.mockito:mockito:1.9.0-rc1'
   compile 'com.actionbarsherlock:actionbarsherlock:4.4.0@aar'

   compile files('hibernate.jar', 'libs/spring.jar')

   compile fileTree('libs')
   compile project(:app2)
}
```

CONFIGURATIONS

» Groups of dependencies

» Can extend each other

PROPERTIES

Groovy:

```
class ExpandoClass {

    def propertyMissing(String name) {
        println "Missing property $name"
    }

    def backingMap = [:]

    Object getProperty( String property ) {
      if( backingMap[ property ] == null ) {
        propertyMissing( property )
      }
      else {
        backingMap[ property ]
      }
    }

    void setProperty( String property, Object value ) {
      backingMap[ property ] = value
    }
}
```

PROPERTIES

```
project.ext.prop1 = "foo"
task doStuff {
    ext.prop2 = "bar"
}
```

PROPERTIES

```
project.ext.prop1 = "foo"
task doStuff {
    ext.prop2 = "bar"
}

if (project.hasProperty['prop1']) {
    // Do something
}
```

PROPERTIES

```
./gradlew :app:properties
```

PROPERTIES

```
/**
 * Adds release signing properties to build if
 * proper properties are present
 */
if (project.hasProperty('alias') && project.hasProperty('keystore') &&
        project.hasProperty('keyPassword')
        && project.hasProperty('storePassword')) {
    project.android.signingConfigs.release.storeFile project.file(project.keystore)
    project.android.signingConfigs.release.keyAlias project.alias
    project.android.signingConfigs.release.storePassword = project.storePassword
    project.android.signingConfigs.release.keyPassword = project.keyPassword
} else {
    project.android.buildTypes.release.signingConfig = null
}
```

PROPERTIES

```
$ ./gradlew assembleRelease \
    -Palias=ALIAS  \
    -Pkeystore=file.keystore \
    -PkeyPassword=PASSWORD
```

NUTS AND BOLTS

- Groovy
- Gradle
  - Tasks
  - Build Phases
  - Projects

FURTHER READING:

» Gradle Beyond the Basics
http://chimera.labs.oreilly.com/books/1234000001741/ch01.html

» Extensive Gradle API documentation at gradle.org

» Gradle Plugin User Guide - Android Tools Project Site
http://tools.android.com/tech-docs/new-build-system/user-guide

DROIDCON LONDON
OCTOBER 31
GRADLIN': PLUGGING IT IN FOR BUILD SUCCESS

GRADLIN'
NUTS AND BOLTS

LISA NEIGUT DROIDCON NYC SEPT 21