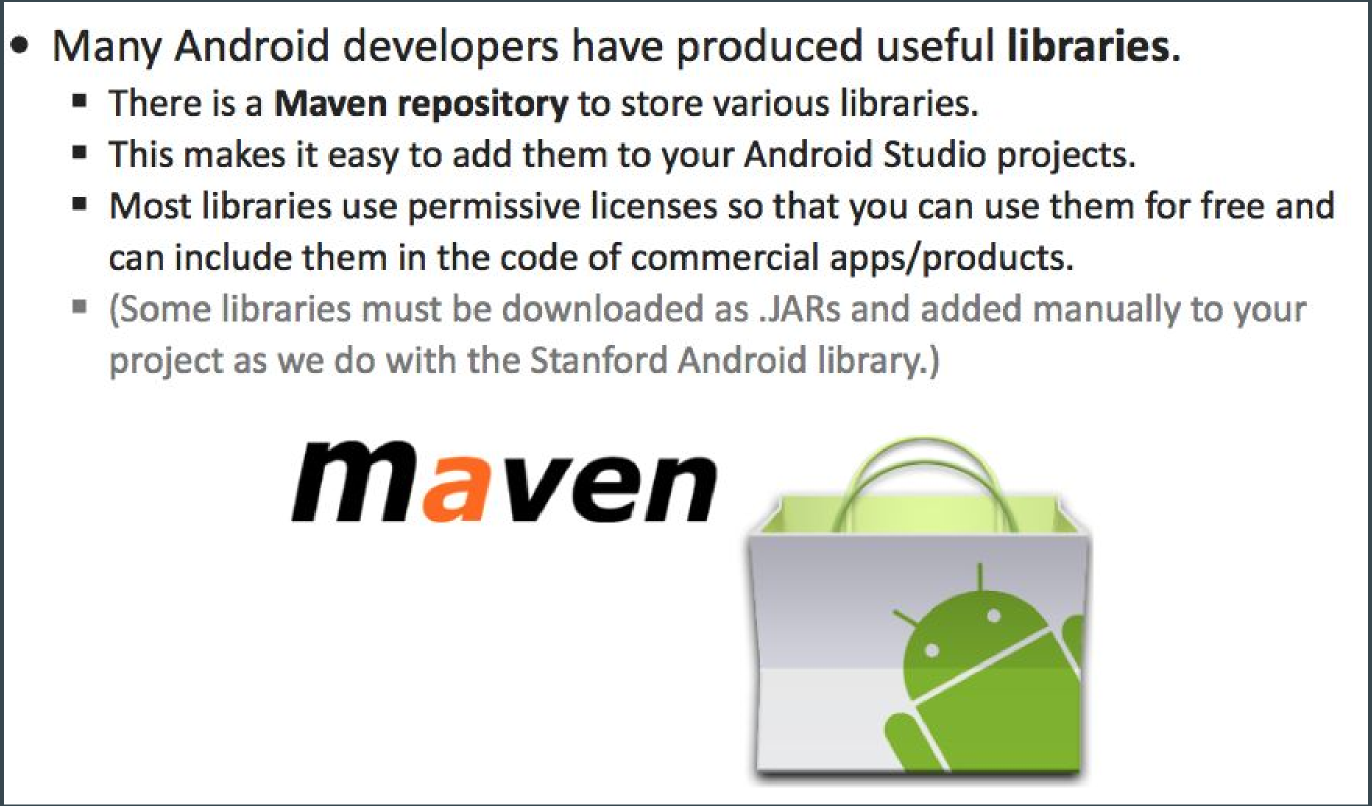
Libraries

•••

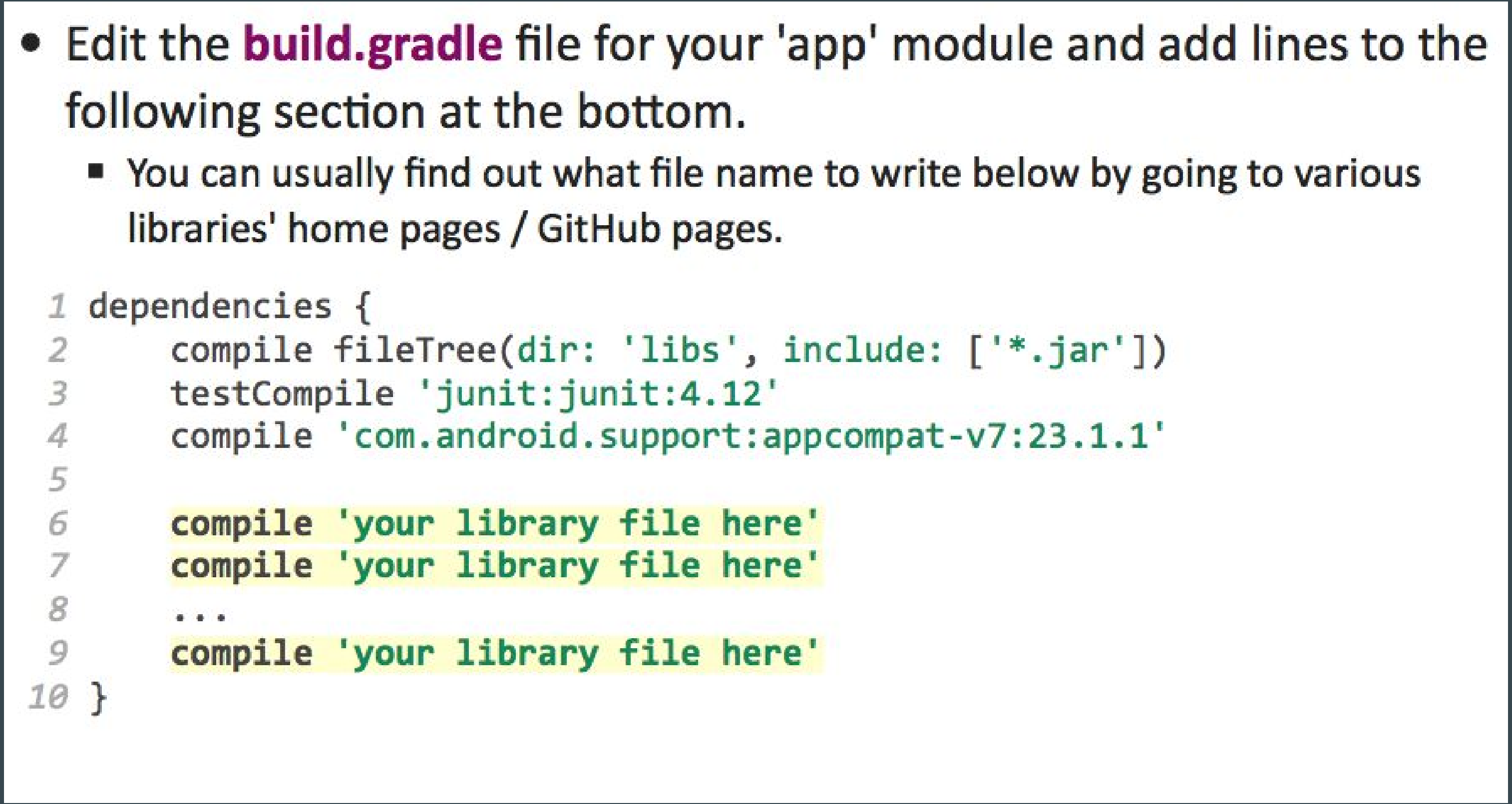
Lecture 10

Libraries



- Many Android developers have produced useful **libraries**.
  - There is a **Maven repository** to store various libraries.
  - This makes it easy to add them to your Android Studio projects.
  - Most libraries use permissive licenses so that you can use them for free and can include them in the code of commercial apps/products.
  - (Some libraries must be downloaded as .JARs and added manually to your project as we do with the Stanford Android library.)

Adding a library to your project

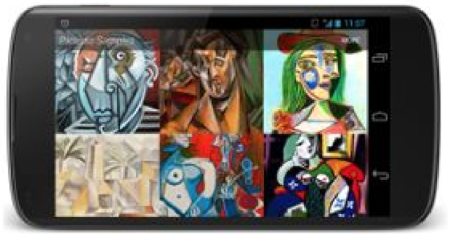


- Edit the **build.gradle** file for your 'app' module and add lines to the following section at the bottom.
  - You can usually find out what file name to write below by going to various libraries' home pages / GitHub pages.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'

    compile 'your library file here'
    compile 'your library file here'
    ...
    compile 'your library file here'
}
```

Picasso

- **Picasso** is a powerful library for manipulating images.
  - written by Square, inc.
  - http://square.github.io/picasso/

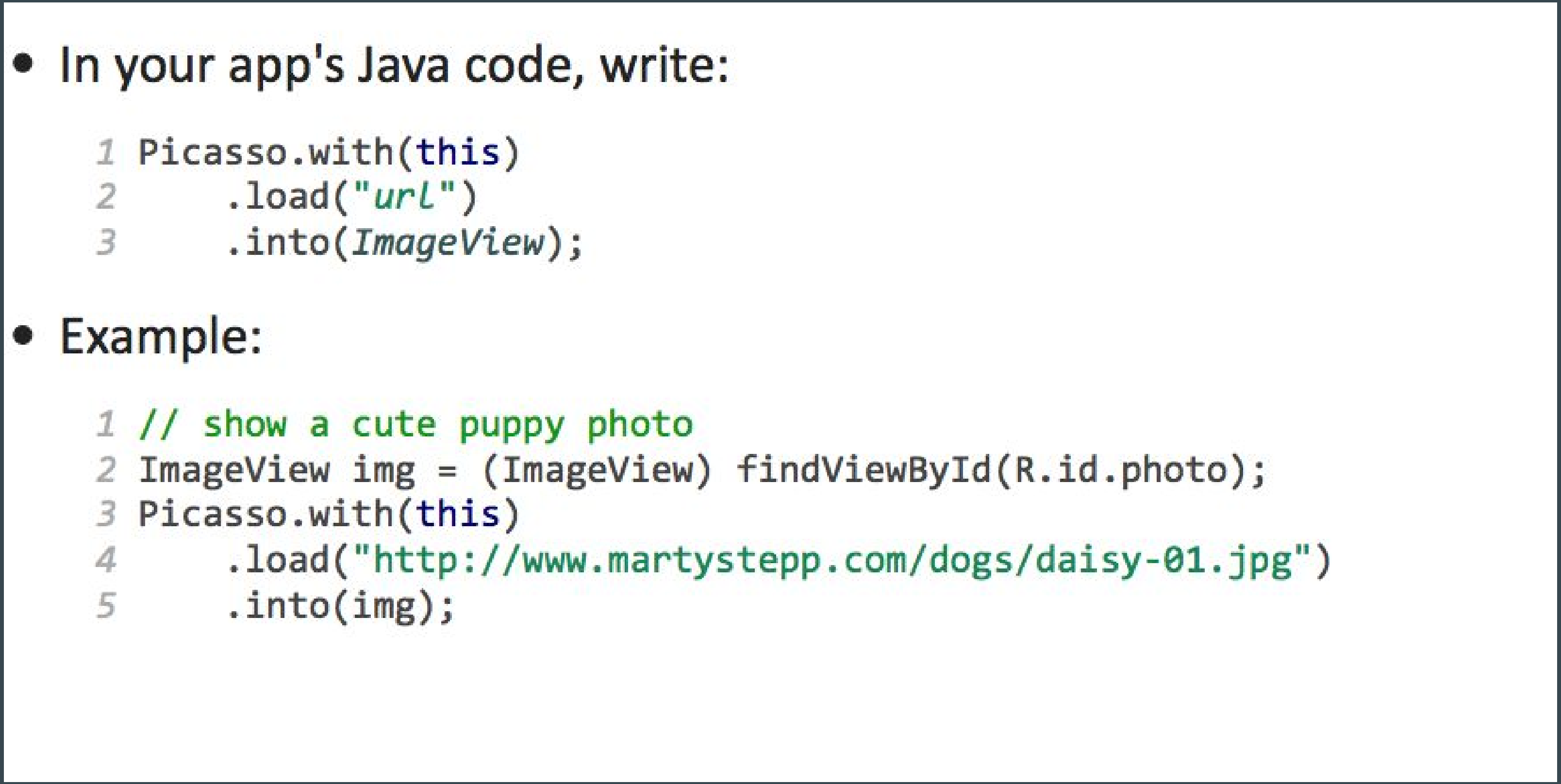


- To add Picasso to your project:

```
// in build.gradle
dependencies {
    ...
    compile 'com.squareup.picasso:picasso:2.5.2'
}
```

```
<!-- in AndroidManifest.xml -->
<uses-permission android:name="android.permission.INTERNET" />
```

Displaying a web photo



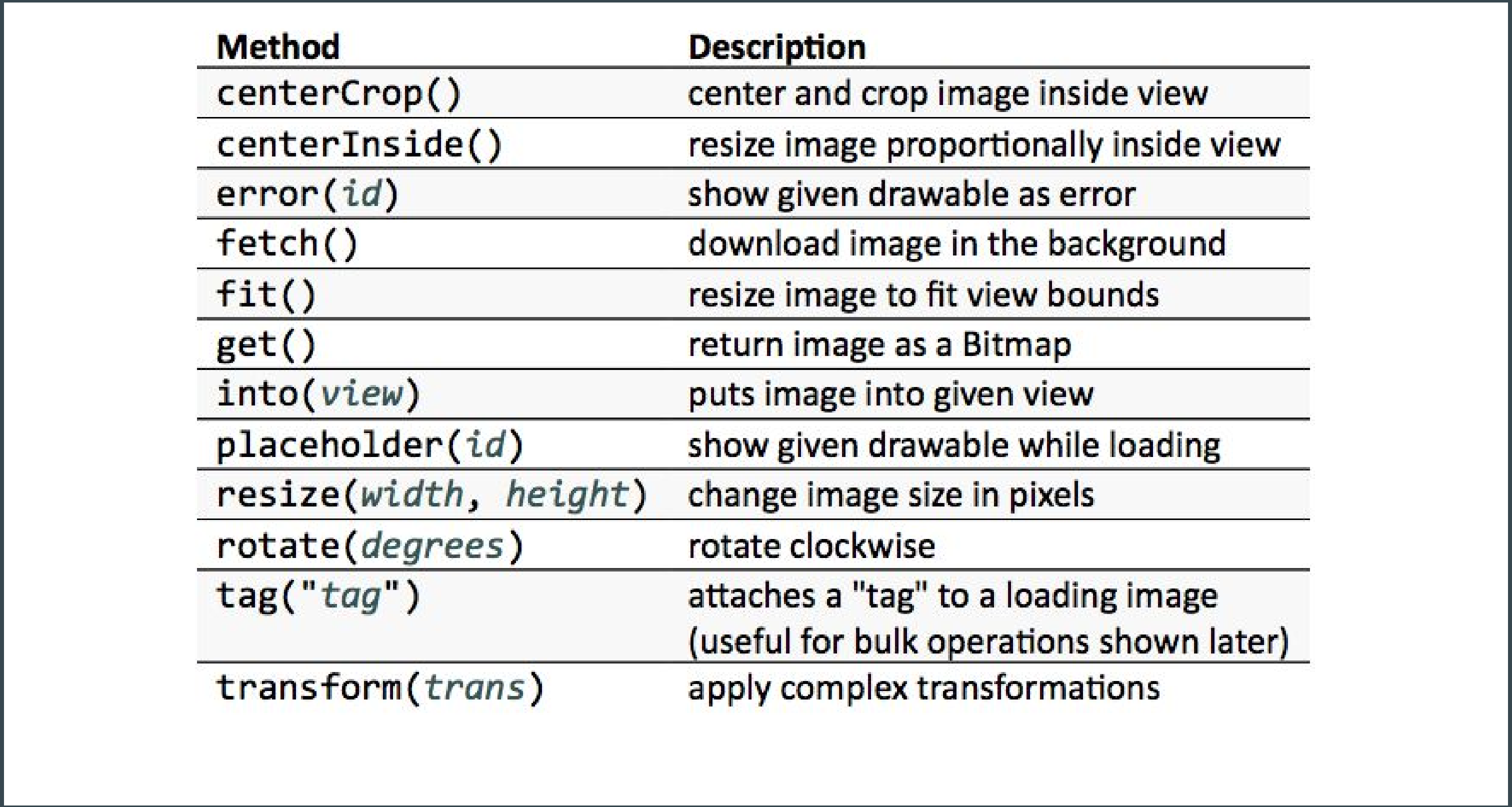
- In your app's Java code, write:

```
Picasso.with(this)
    .load("url")
    .into(ImageView);
```

- Example:

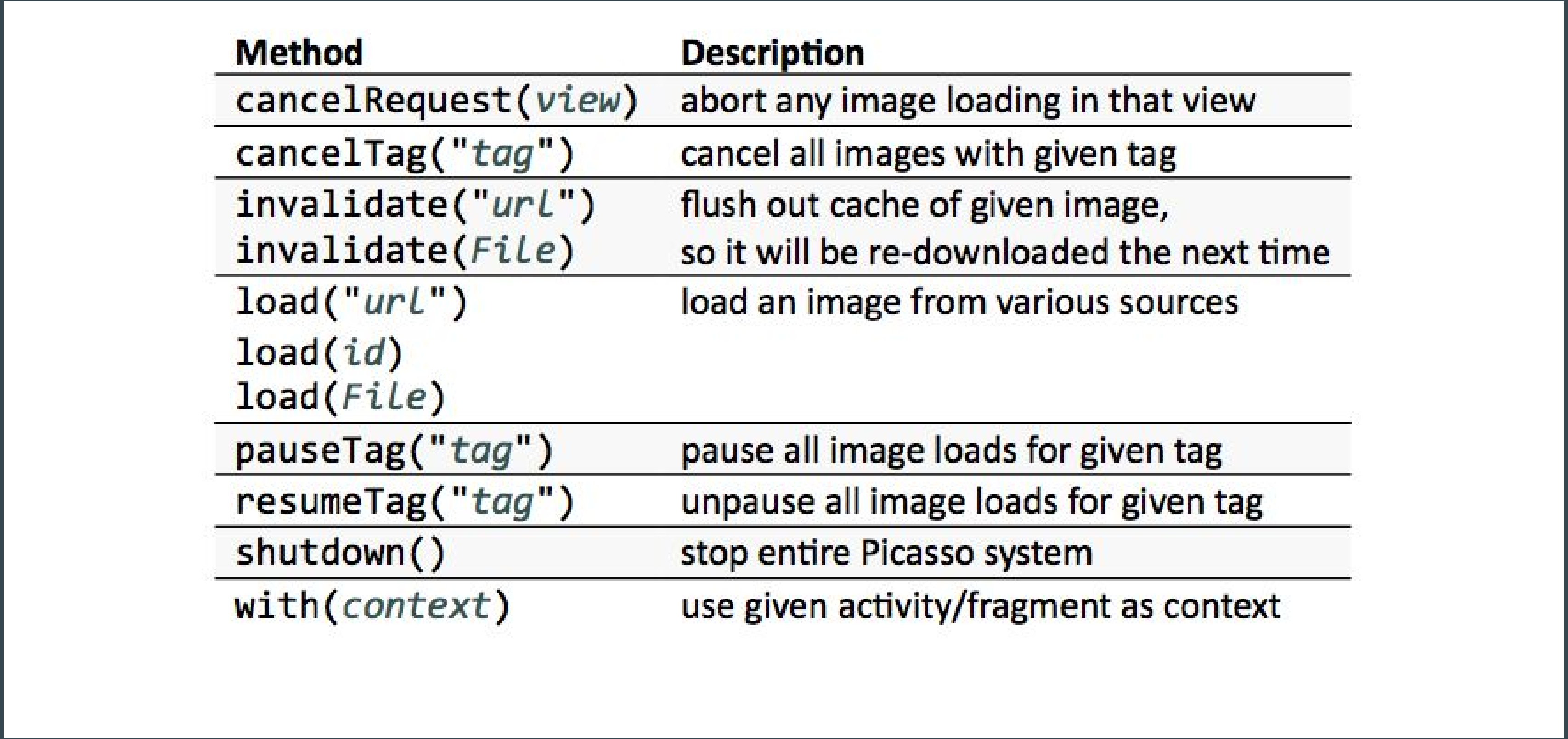
```
// show a cute puppy photo
ImageView img = (ImageView) findViewById(R.id.photo);
Picasso.with(this)
    .load("http://www.martystepp.com/dogs/daisy-01.jpg")
    .into(img);
```

Picasso Image methods



Method	Description
`centerCrop()`	center and crop image inside view
`centerInside()`	resize image proportionally inside view
`error(`*`id`*`)`	show given drawable as error
`fetch()`	download image in the background
`fit()`	resize image to fit view bounds
`get()`	return image as a Bitmap
`into(`*`view`*`)`	puts image into given view
`placeholder(`*`id`*`)`	show given drawable while loading
`resize(`*`width`*`, `*`height`*`)`	change image size in pixels
`rotate(`*`degrees`*`)`	rotate clockwise
`tag("`*`tag`*`")`	attaches a "tag" to a loading image (useful for bulk operations shown later)
`transform(`*`trans`*`)`	apply complex transformations

Picasso methods



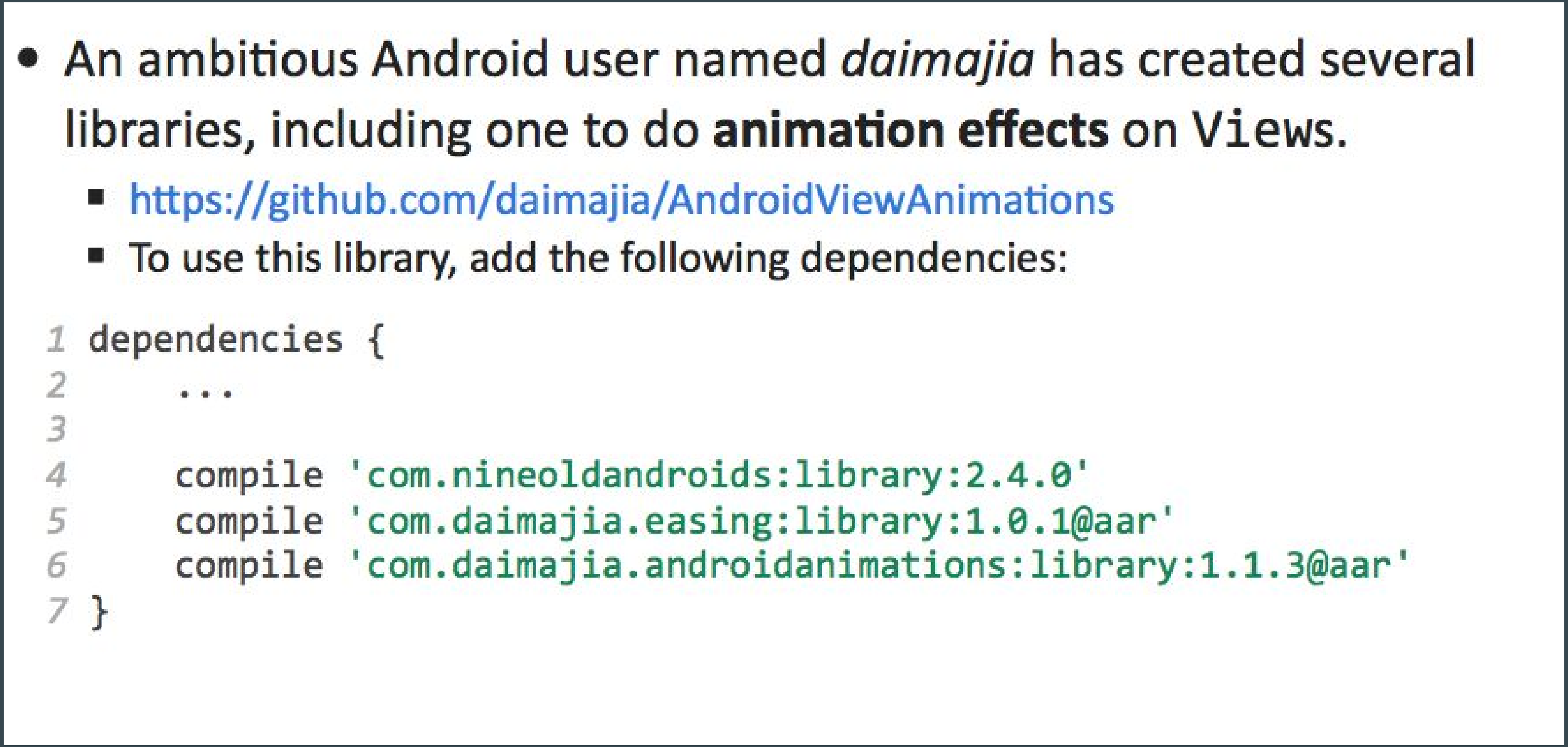
Method	Description
`cancelRequest(`*view*`)`	abort any image loading in that view
`cancelTag("`*tag*`")`	cancel all images with given tag

`invalidate("`*url*`")`
`invalidate(`*File*`)`
	flush out cache of given image,
	so it will be re-downloaded the next time

`load("`*url*`")`
`load(`*id*`)`
`load(`*File*`)`
	load an image from various sources

`pauseTag("`*tag*`")`	pause all image loads for given tag
`resumeTag("`*tag*`")`	unpause all image loads for given tag
`shutdown()`	stop entire Picasso system
`with(`*context*`)`	use given activity/fragment as context

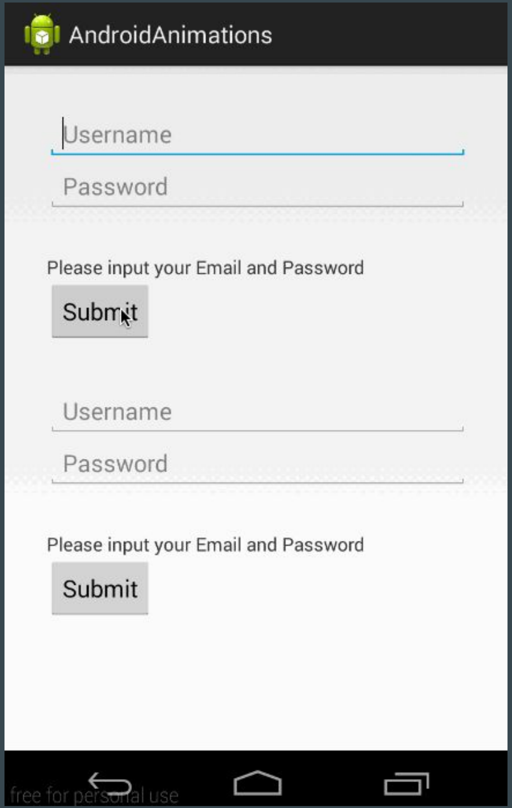
Android Animations



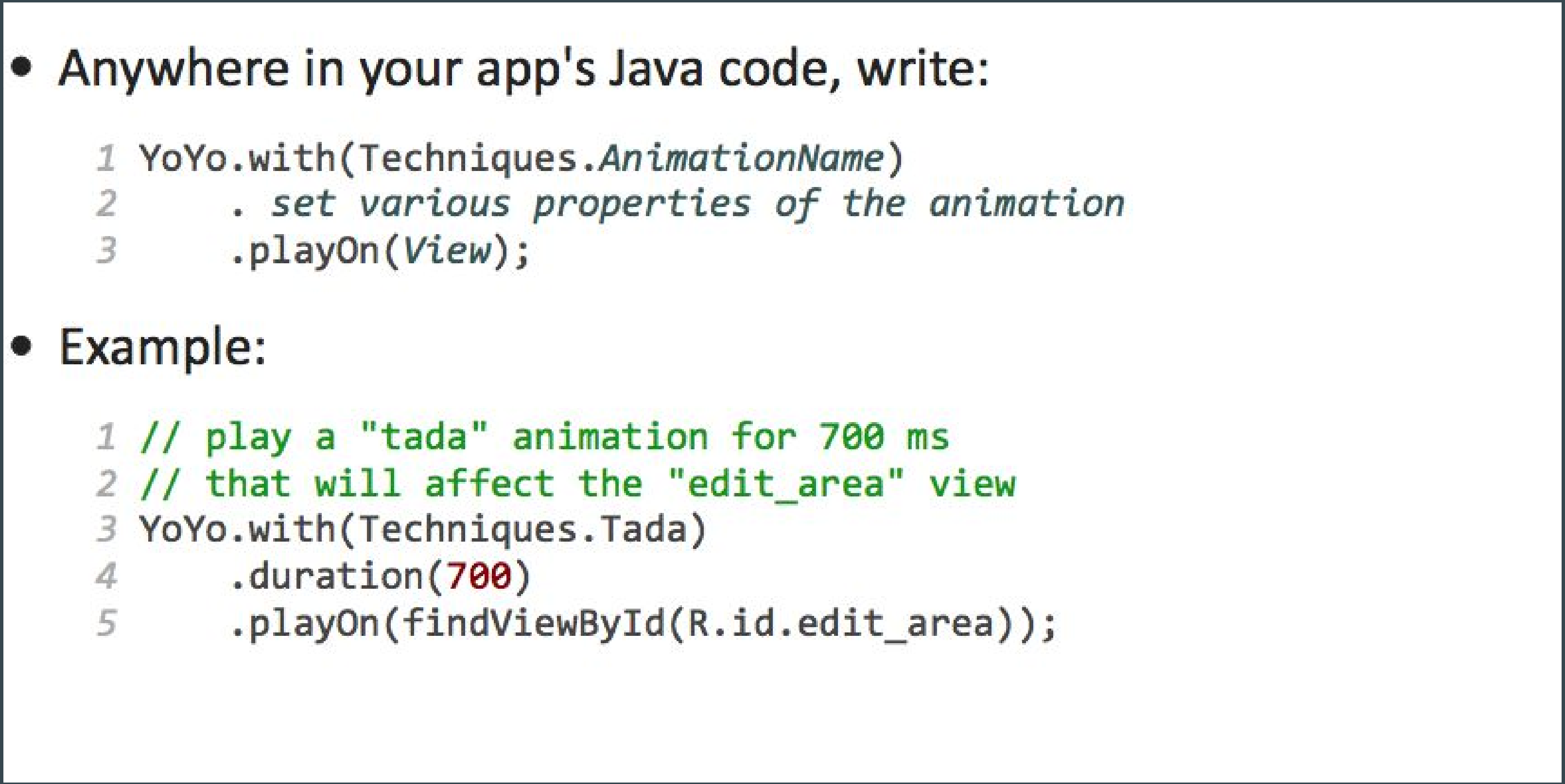
- An ambitious Android user named *daimajia* has created several libraries, including one to do **animation effects** on `Views`.
  - https://github.com/daimajia/AndroidViewAnimations
  - To use this library, add the following dependencies:

```
dependencies {
    ...

    compile 'com.nineoldandroids:library:2.4.0'
    compile 'com.daimajia.easing:library:1.0.1@aar'
    compile 'com.daimajia.androidanimations:library:1.1.3@aar'
}
```

Animations demo



Using an animation



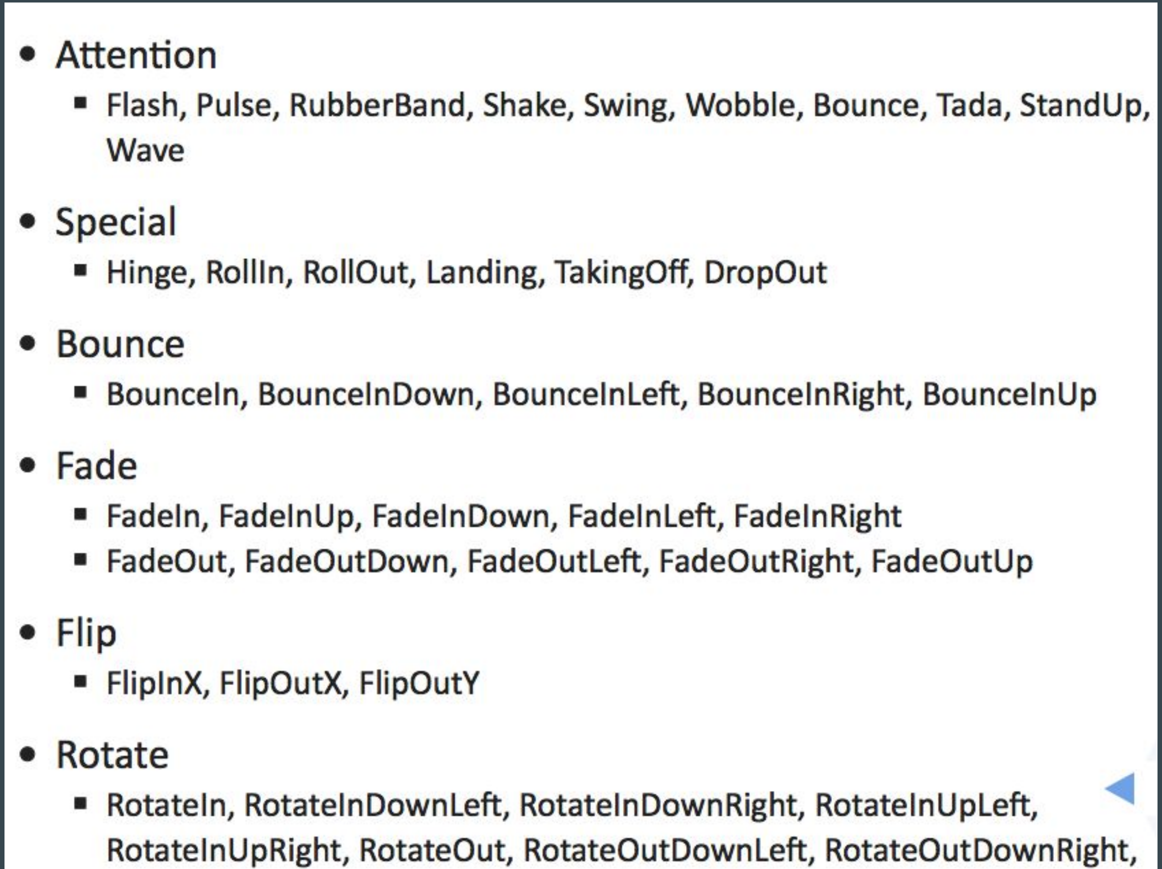
- Anywhere in your app's Java code, write:

```
YoYo.with(Techniques.AnimationName)
    . set various properties of the animation
    .playOn(View);
```

- Example:

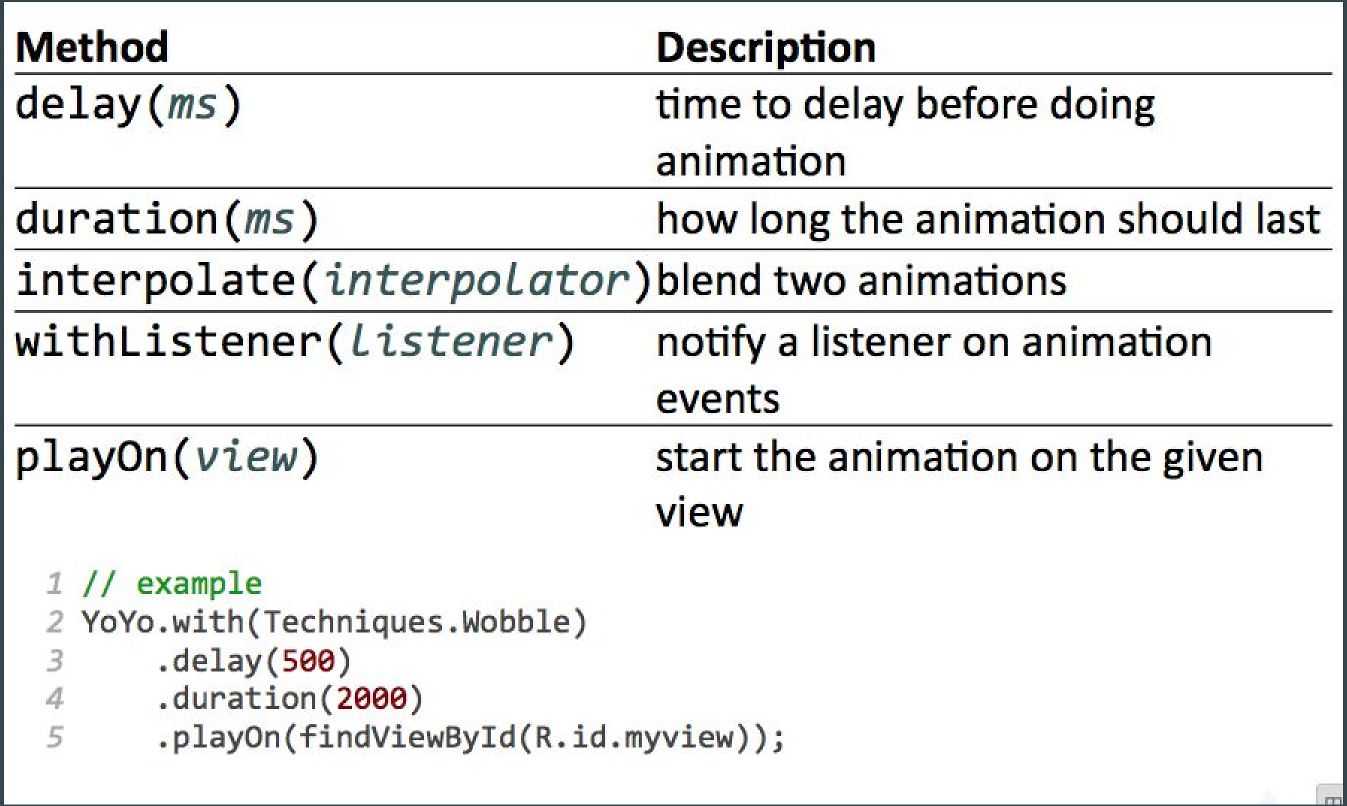
```
// play a "tada" animation for 700 ms
// that will affect the "edit_area" view
YoYo.with(Techniques.Tada)
    .duration(700)
    .playOn(findViewById(R.id.edit_area));
```

Animations



- Attention
  - Flash, Pulse, RubberBand, Shake, Swing, Wobble, Bounce, Tada, StandUp, Wave
- Special
  - Hinge, RollIn, RollOut, Landing, TakingOff, DropOut
- Bounce
  - BounceIn, BounceInDown, BounceInLeft, BounceInRight, BounceInUp
- Fade
  - FadeIn, FadeInUp, FadeInDown, FadeInLeft, FadeInRight
  - FadeOut, FadeOutDown, FadeOutLeft, FadeOutRight, FadeOutUp
- Flip
  - FlipInX, FlipOutX, FlipOutY
- Rotate
  - RotateIn, RotateInDownLeft, RotateInDownRight, RotateInUpLeft, RotateInUpRight, RotateOut, RotateOutDownLeft, RotateOutDownRight,

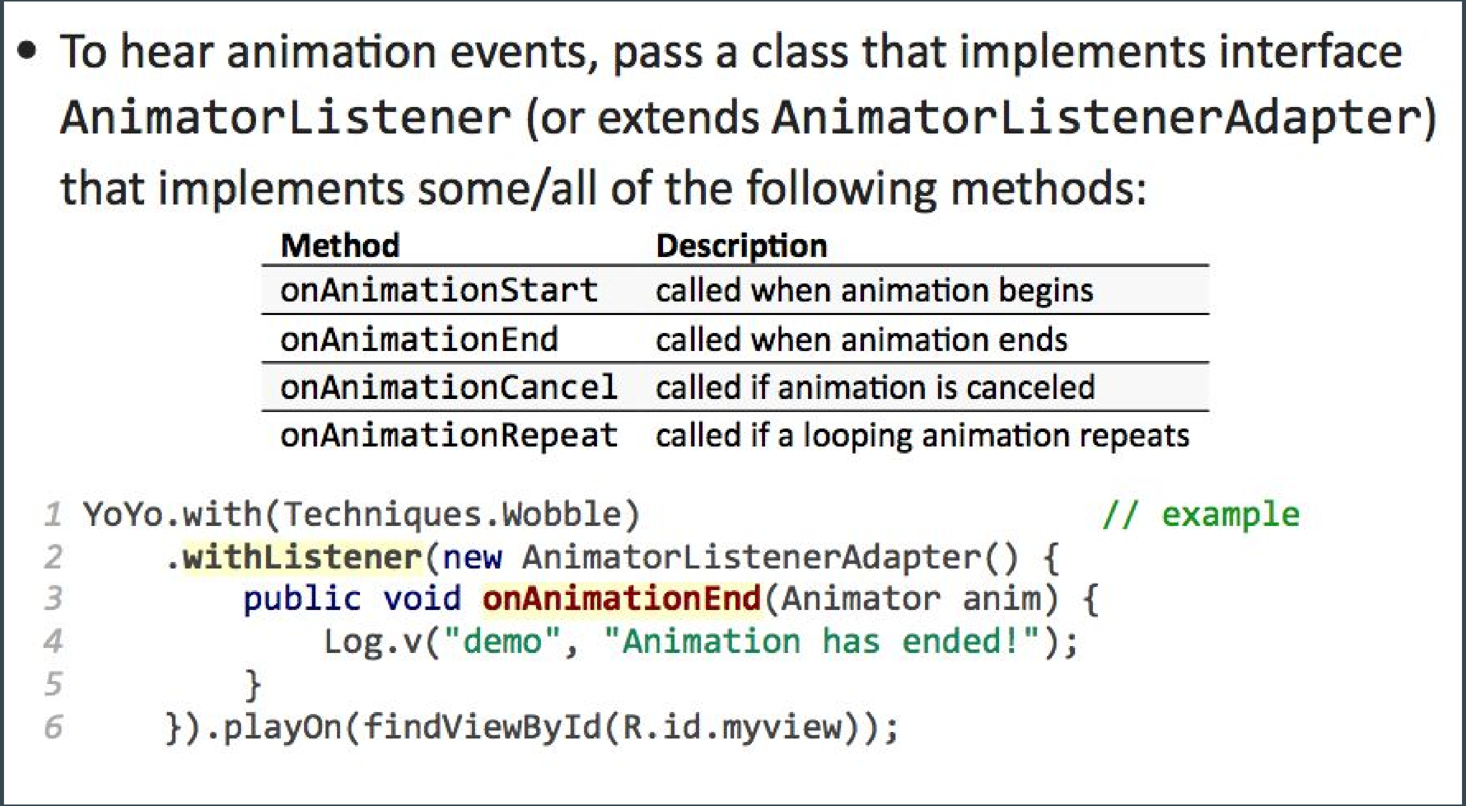
Example (Yoyo animation properties)



Method	Description
delay(*ms*)	time to delay before doing animation
duration(*ms*)	how long the animation should last
interpolate(*interpolator*)	blend two animations
withListener(*listener*)	notify a listener on animation events
playOn(*view*)	start the animation on the given view

```
// example
YoYo.with(Techniques.Wobble)
    .delay(500)
    .duration(2000)
    .playOn(findViewById(R.id.myview));
```

YoYo animation events

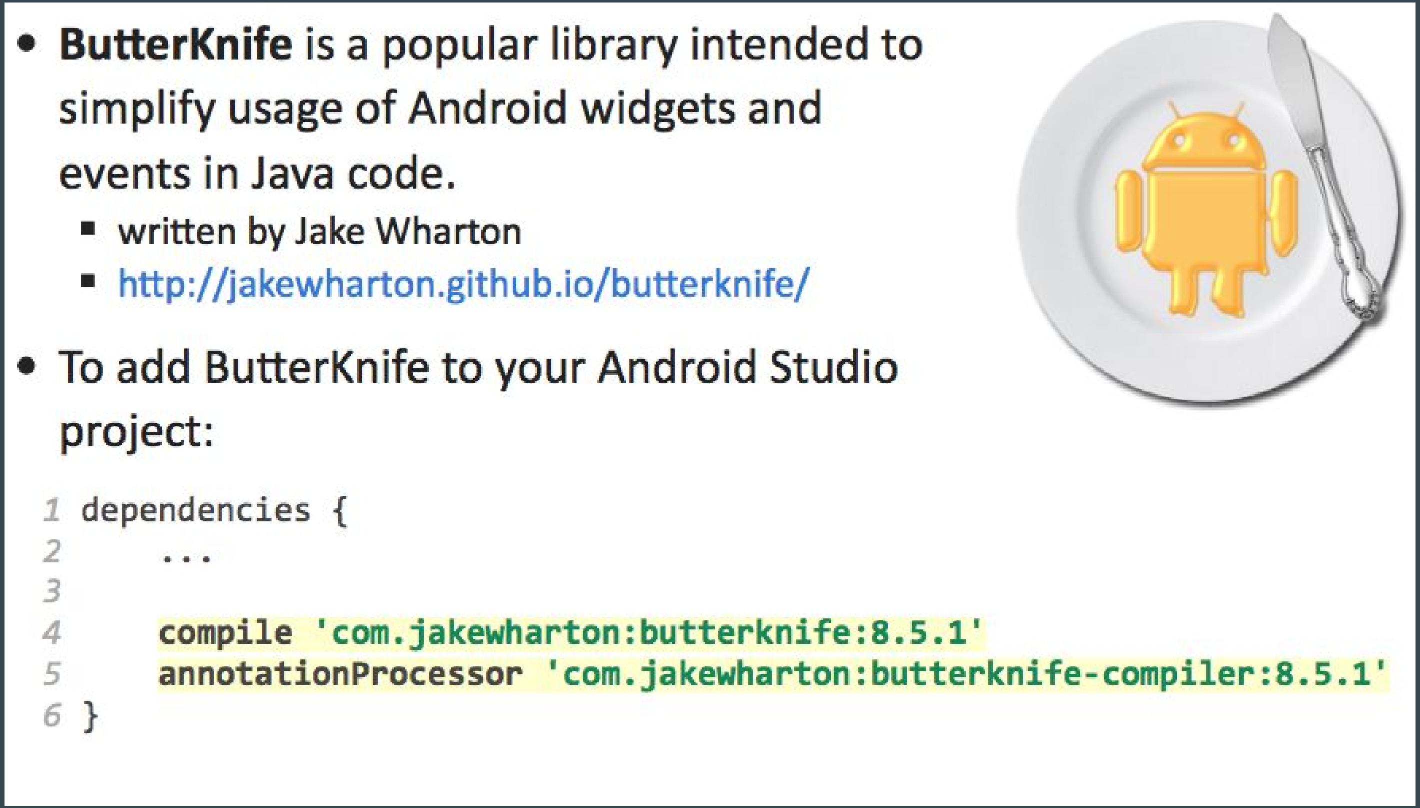


- To hear animation events, pass a class that implements interface `AnimatorListener` (or extends `AnimatorListenerAdapter`) that implements some/all of the following methods:

Method	Description
`onAnimationStart`	called when animation begins
`onAnimationEnd`	called when animation ends
`onAnimationCancel`	called if animation is canceled
`onAnimationRepeat`	called if a looping animation repeats

```
YoYo.with(Techniques.Wobble)                        // example
    .withListener(new AnimatorListenerAdapter() {
        public void onAnimationEnd(Animator anim) {
            Log.v("demo", "Animation has ended!");
        }
    }).playOn(findViewById(R.id.myview));
```

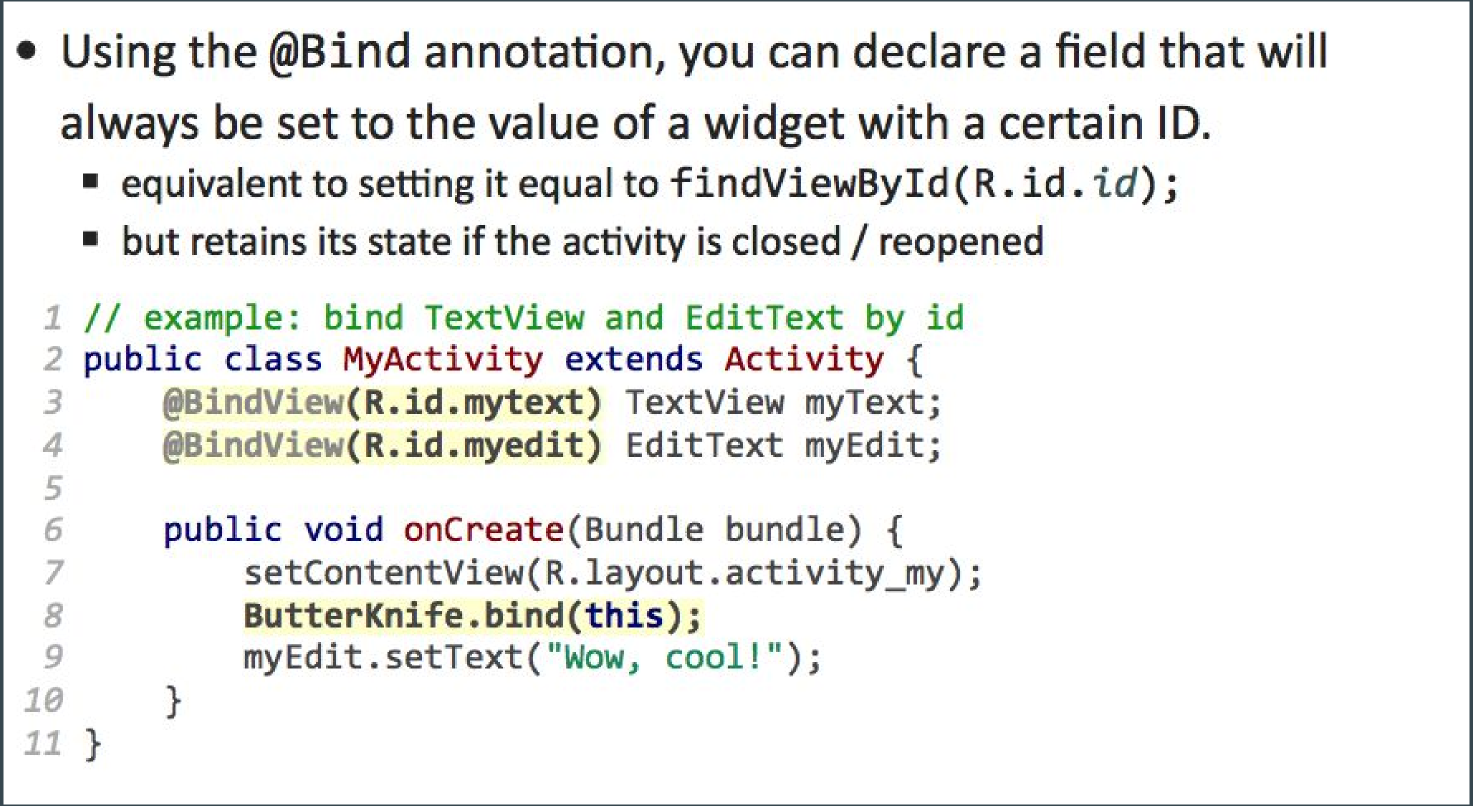
ButterKnife library



- **ButterKnife** is a popular library intended to simplify usage of Android widgets and events in Java code.
  - written by Jake Wharton
  - http://jakewharton.github.io/butterknife/
- To add ButterKnife to your Android Studio project:

```
dependencies {
    ...

    compile 'com.jakewharton:butterknife:8.5.1'
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}
```

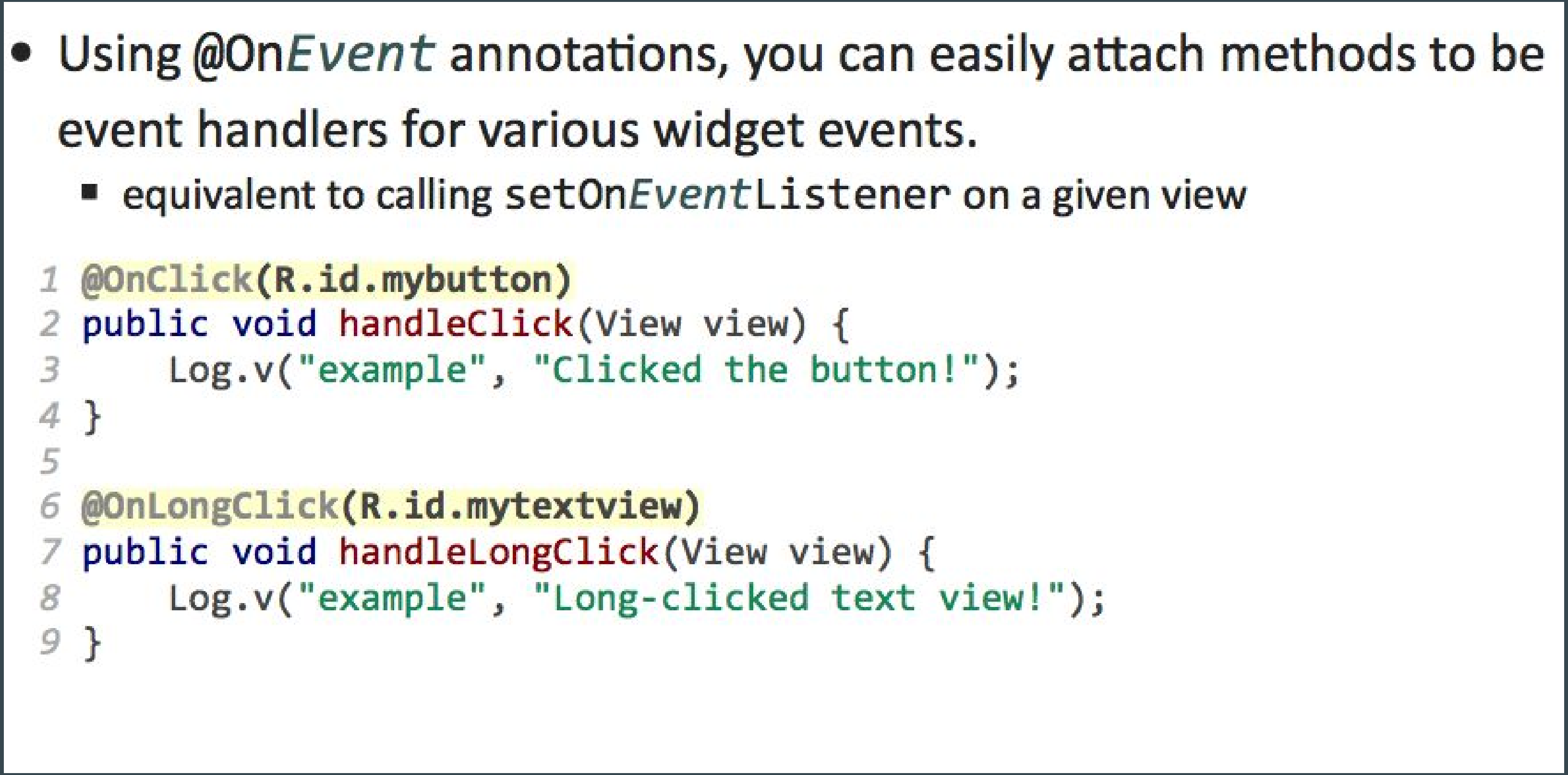
ButterKnife field bindings



- Using the @Bind annotation, you can declare a field that will always be set to the value of a widget with a certain ID.
  - equivalent to setting it equal to `findViewById(R.id.id);`
  - but retains its state if the activity is closed / reopened

```java
// example: bind TextView and EditText by id
public class MyActivity extends Activity {
    @BindView(R.id.mytext) TextView myText;
    @BindView(R.id.myedit) EditText myEdit;

    public void onCreate(Bundle bundle) {
        setContentView(R.layout.activity_my);
        ButterKnife.bind(this);
        myEdit.setText("Wow, cool!");
    }
}
```

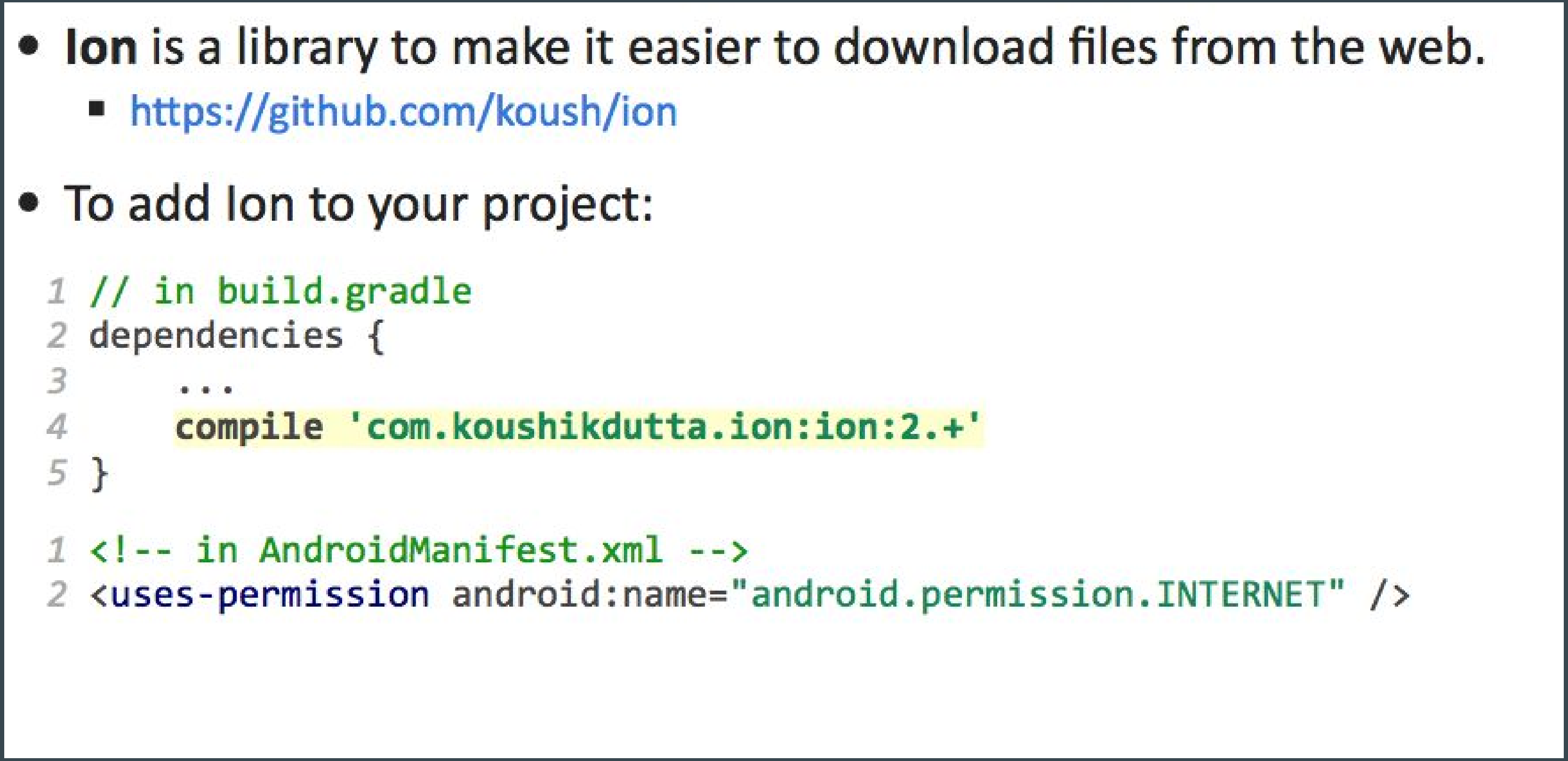
ButterKnife event bindings



- Using @On*Event* annotations, you can easily attach methods to be event handlers for various widget events.
  - equivalent to calling `setOn`*`Event`*`Listener` on a given view

```java
@OnClick(R.id.mybutton)
public void handleClick(View view) {
    Log.v("example", "Clicked the button!");
}

@OnLongClick(R.id.mytextview)
public void handleLongClick(View view) {
    Log.v("example", "Long-clicked text view!");
}
```

Ion Library

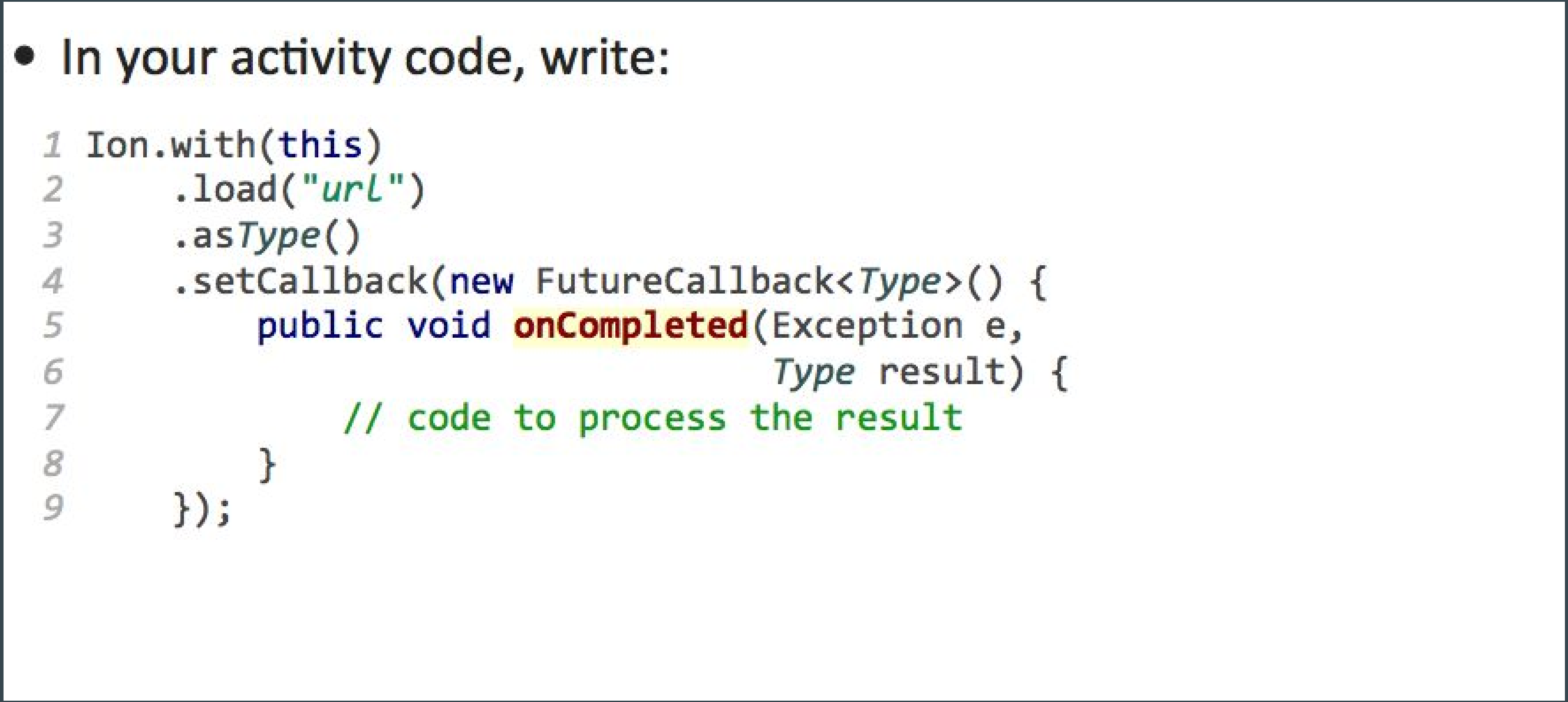


- **Ion** is a library to make it easier to download files from the web.
  - https://github.com/koush/ion
- To add Ion to your project:

```
// in build.gradle
dependencies {
    ...
    compile 'com.koushikdutta.ion:ion:2.+'
}
```

```
<!-- in AndroidManifest.xml -->
<uses-permission android:name="android.permission.INTERNET" />
```

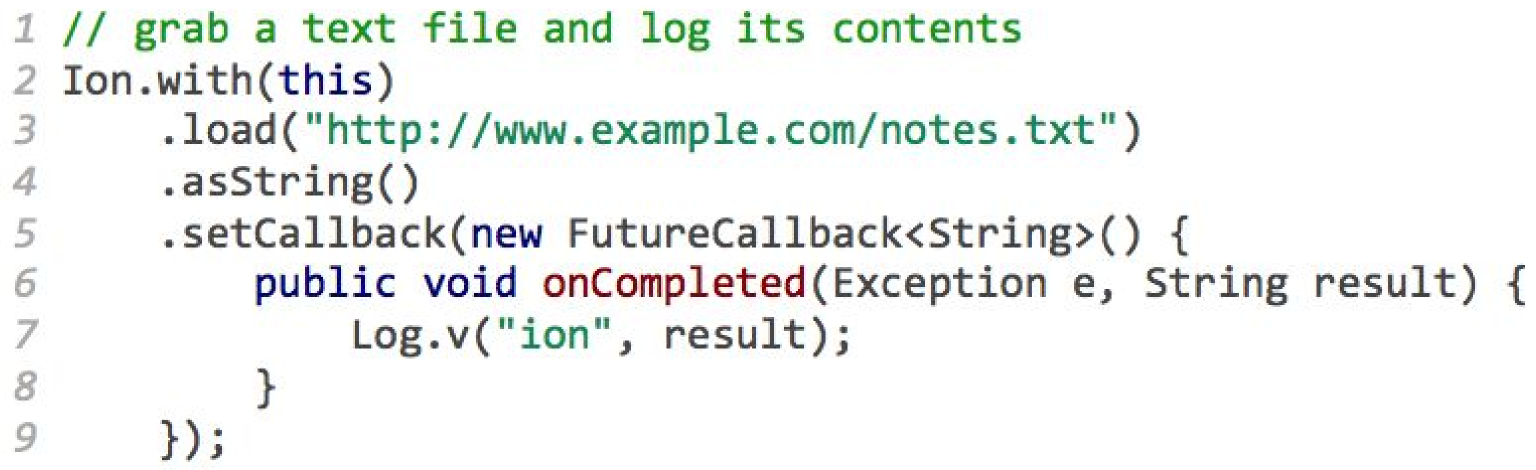
Downloading a web file



- In your activity code, write:

```
Ion.with(this)
    .load("url")
    .asType()
    .setCallback(new FutureCallback<Type>() {
        public void onCompleted(Exception e,
                                Type result) {
            // code to process the result
        }
    });
```

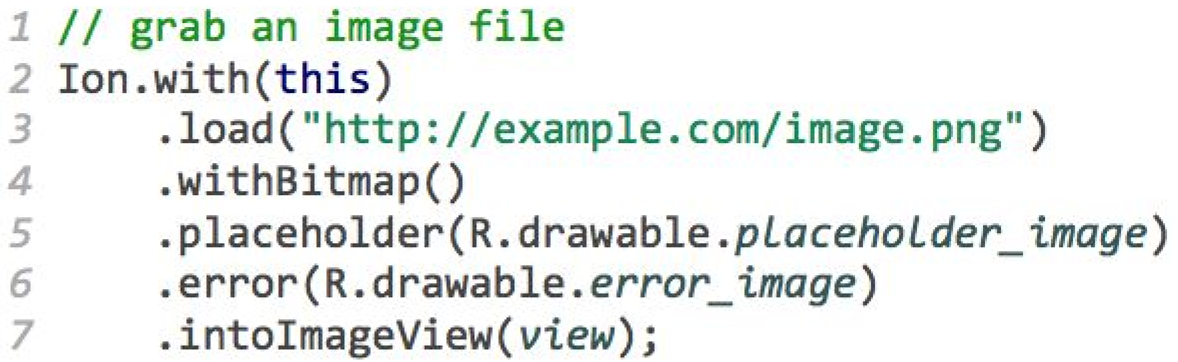
Ion download example



```java
// grab a text file and log its contents
Ion.with(this)
    .load("http://www.example.com/notes.txt")
    .asString()
    .setCallback(new FutureCallback<String>() {
        public void onCompleted(Exception e, String result) {
            Log.v("ion", result);
        }
    });
```

- other types: `asJsonObject`, `asByteArray`

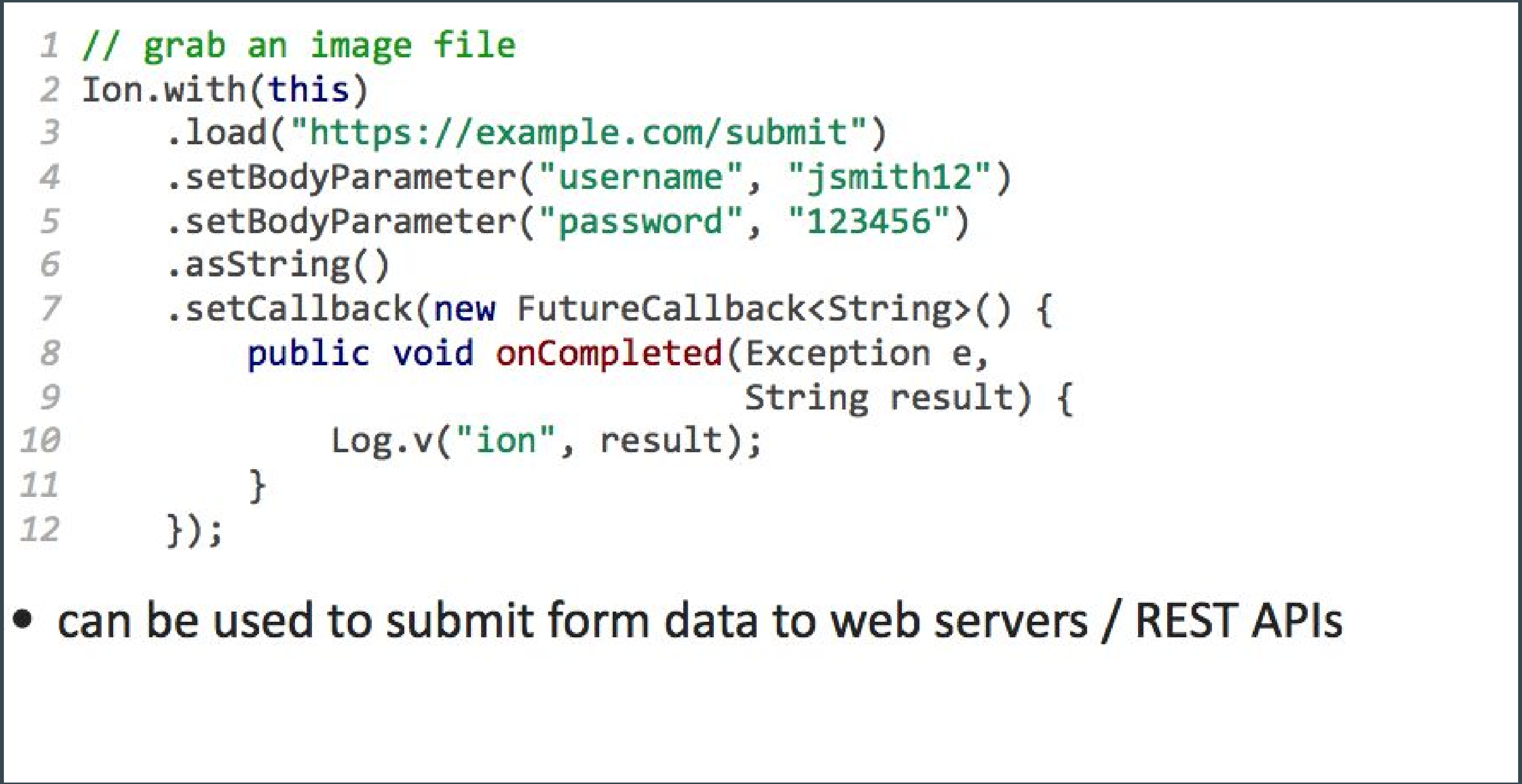
Ion to fetch an image



```java
// grab an image file
Ion.with(this)
    .load("http://example.com/image.png")
    .withBitmap()
    .placeholder(R.drawable.placeholder_image)
    .error(R.drawable.error_image)
    .intoImageView(view);
```

- similar to functionality of Picasso library, without as many image processing features (fit, resize, crop)

Ion to post data to a web server



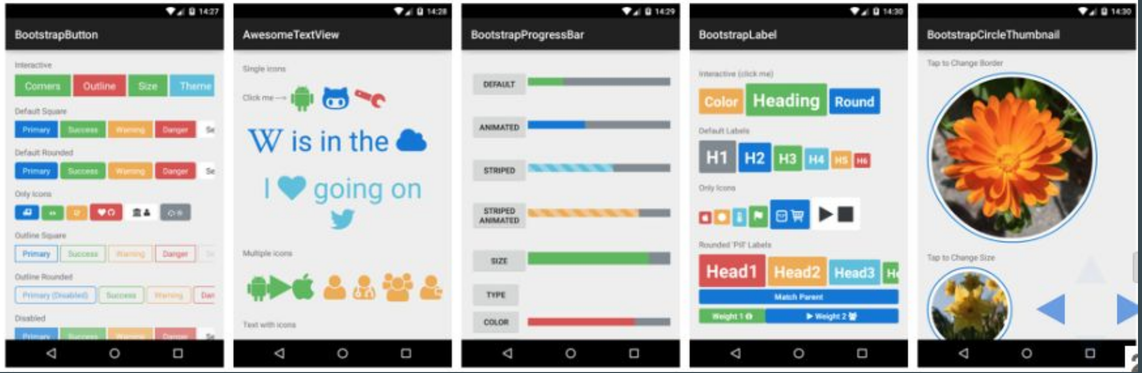
```java
// grab an image file
Ion.with(this)
    .load("https://example.com/submit")
    .setBodyParameter("username", "jsmith12")
    .setBodyParameter("password", "123456")
    .asString()
    .setCallback(new FutureCallback<String>() {
        public void onCompleted(Exception e,
                                String result) {
            Log.v("ion", result);
        }
    });
```

- can be used to submit form data to web servers / REST APIs

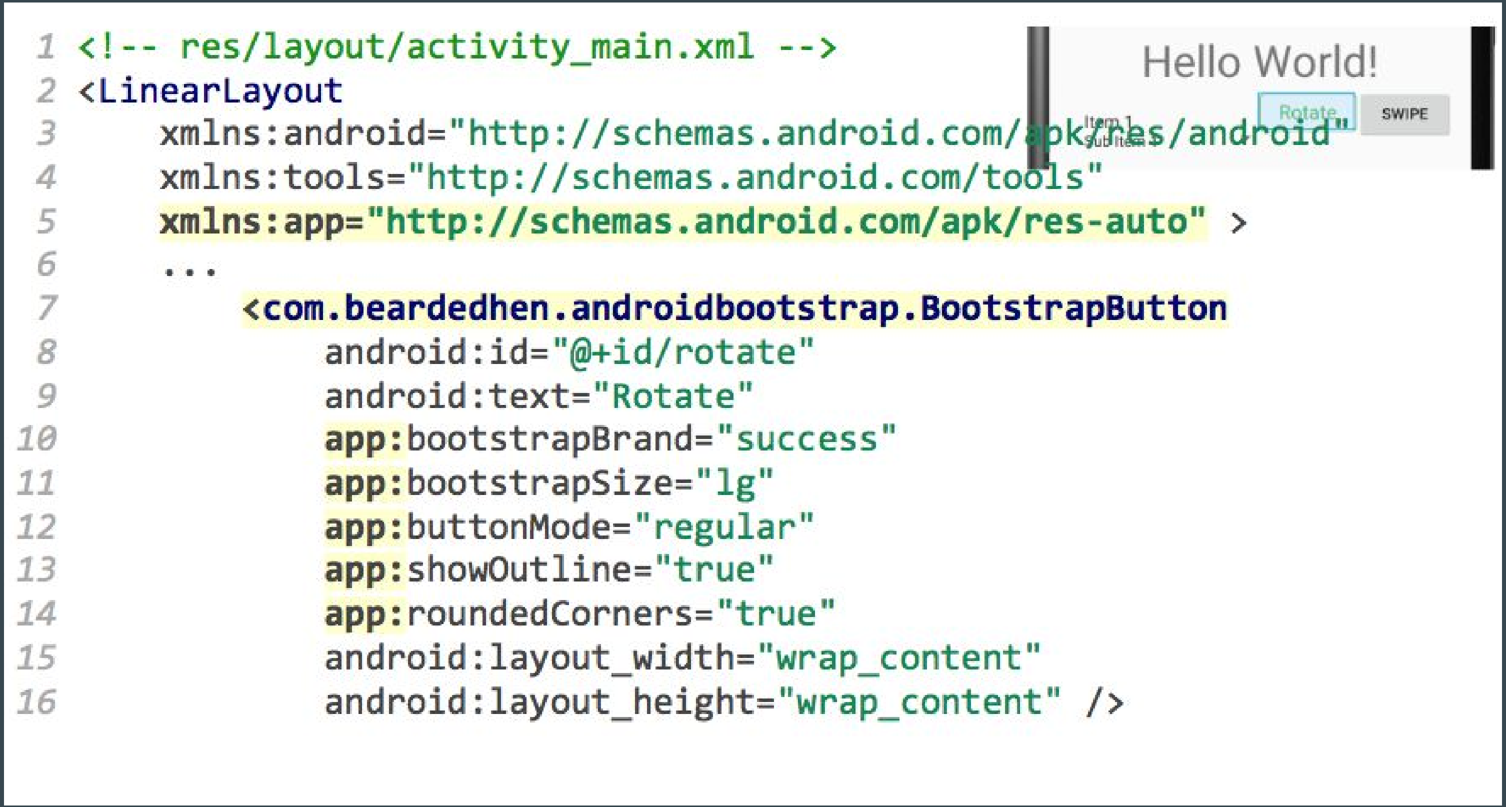
Android-bootstrap library

- **Android-Bootstrap** is a library that provides some good-looking customizable widgets not normally available in Android
  - https://github.com/Bearded-Hen/Android-Bootstrap
- To add it to your project:

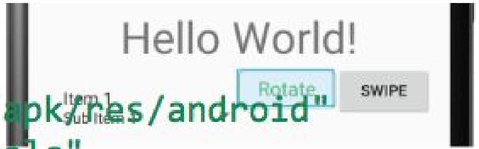
```
// in build.gradle
dependencies {
    compile 'com.beardedhen:androidbootstrap:2.3.1'
}
```



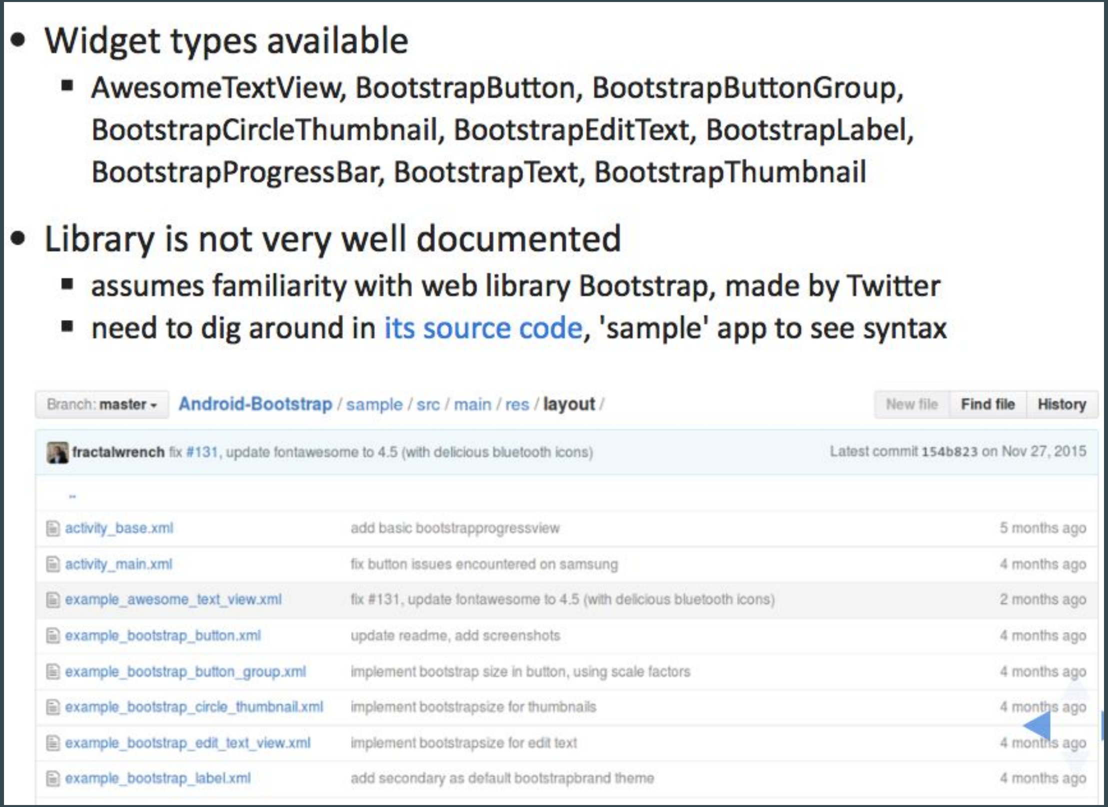
Using Android-Bootstrap widgets



```
<!-- res/layout/activity_main.xml -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" >
    ...
        <com.beardedhen.androidbootstrap.BootstrapButton
            android:id="@+id/rotate"
            android:text="Rotate"
            app:bootstrapBrand="success"
            app:bootstrapSize="lg"
            app:buttonMode="regular"
            app:showOutline="true"
            app:roundedCorners="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
```



More about Android-Bootstrap



- Widget types available
  - AwesomeTextView, BootstrapButton, BootstrapButtonGroup, BootstrapCircleThumbnail, BootstrapEditText, BootstrapLabel, BootstrapProgressBar, BootstrapText, BootstrapThumbnail
- Library is not very well documented
  - assumes familiarity with web library Bootstrap, made by Twitter
  - need to dig around in its source code, 'sample' app to see syntax

Branch: master ▾ **Android-Bootstrap** / sample / src / main / res / **layout** /

New file | Find file | History

fractalwrench fix #131, update fontawesome to 4.5 (with delicious bluetooth icons) — Latest commit 154b823 on Nov 27, 2015

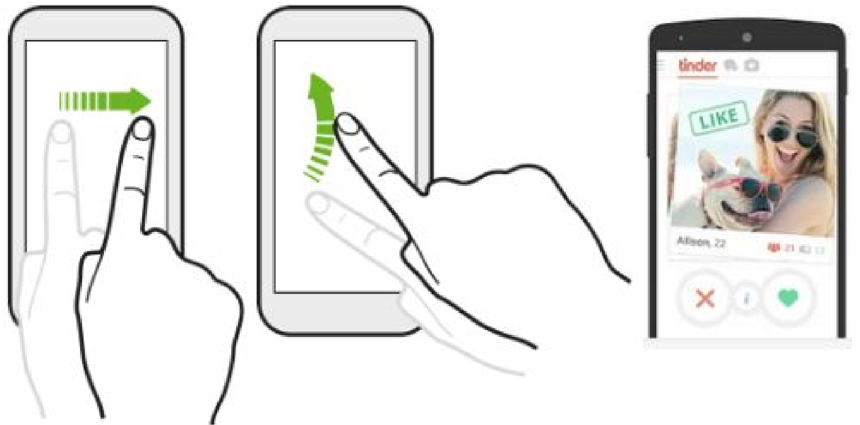
..		
activity_base.xml	add basic bootstrapprogressview	5 months ago
activity_main.xml	fix button issues encountered on samsung	4 months ago
example_awesome_text_view.xml	fix #131, update fontawesome to 4.5 (with delicious bluetooth icons)	2 months ago
example_bootstrap_button.xml	update readme, add screenshots	4 months ago
example_bootstrap_button_group.xml	implement bootstrap size in button, using scale factors	4 months ago
example_bootstrap_circle_thumbnail.xml	implement bootstrapsize for thumbnails	4 months ago
example_bootstrap_edit_text_view.xml	implement bootstrapsize for edit text	4 months ago
example_bootstrap_label.xml	add secondary as default bootstrapbrand theme	4 months ago

Swiping

- **swipe**: Sliding the finger in a given direction.
  - Commonly used in mobile apps to accept/reject, delete, dismiss
  - Most common use case:

    swipe left (no, negative, delete), or

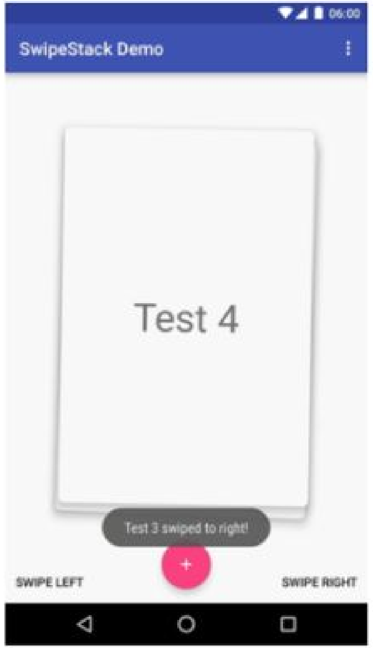
    swipe right (yes, positive, approve)



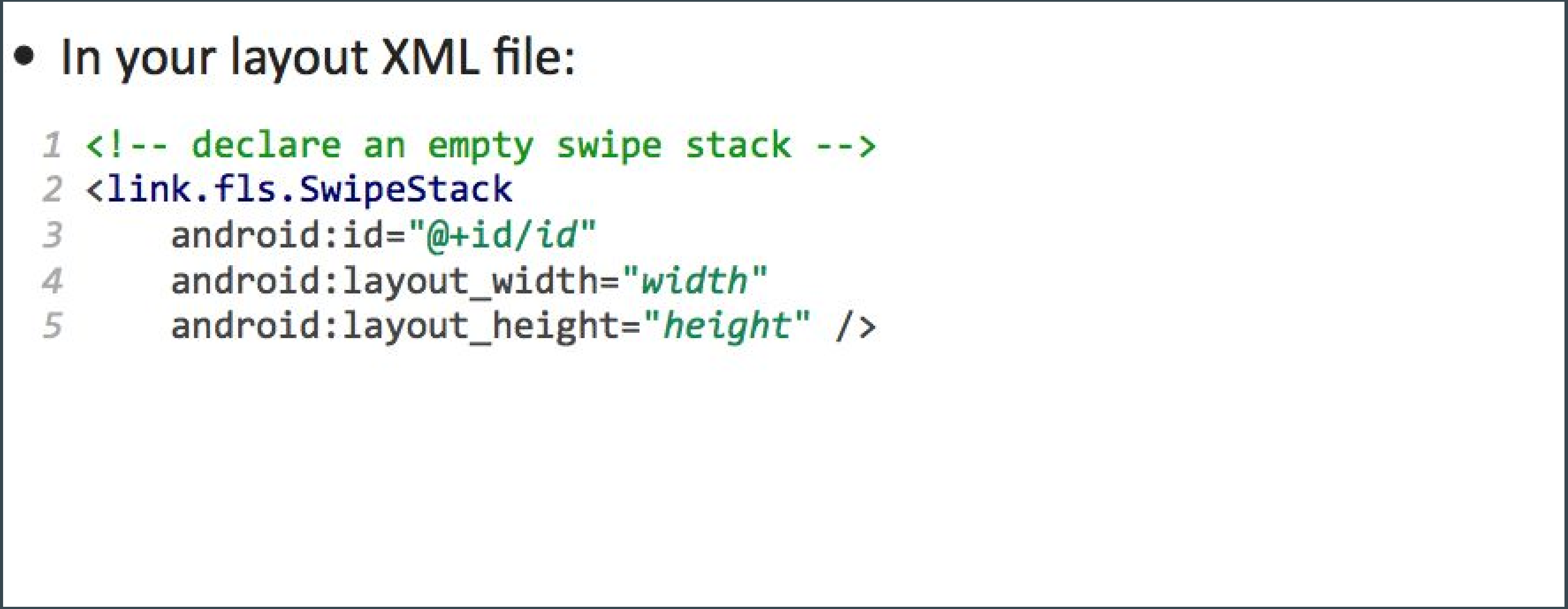
SwipeStack library

- **SwipeStack** is a library that helps you make a stack of views that look like cards that you can "swipe" left or right.
  - https://github.com/flschweiger/SwipeStack
- To add SwipeStack to your project:

```
// in build.gradle
dependencies {
      ...
      compile 'link.fls:swipestack:0.3.0'
}
```



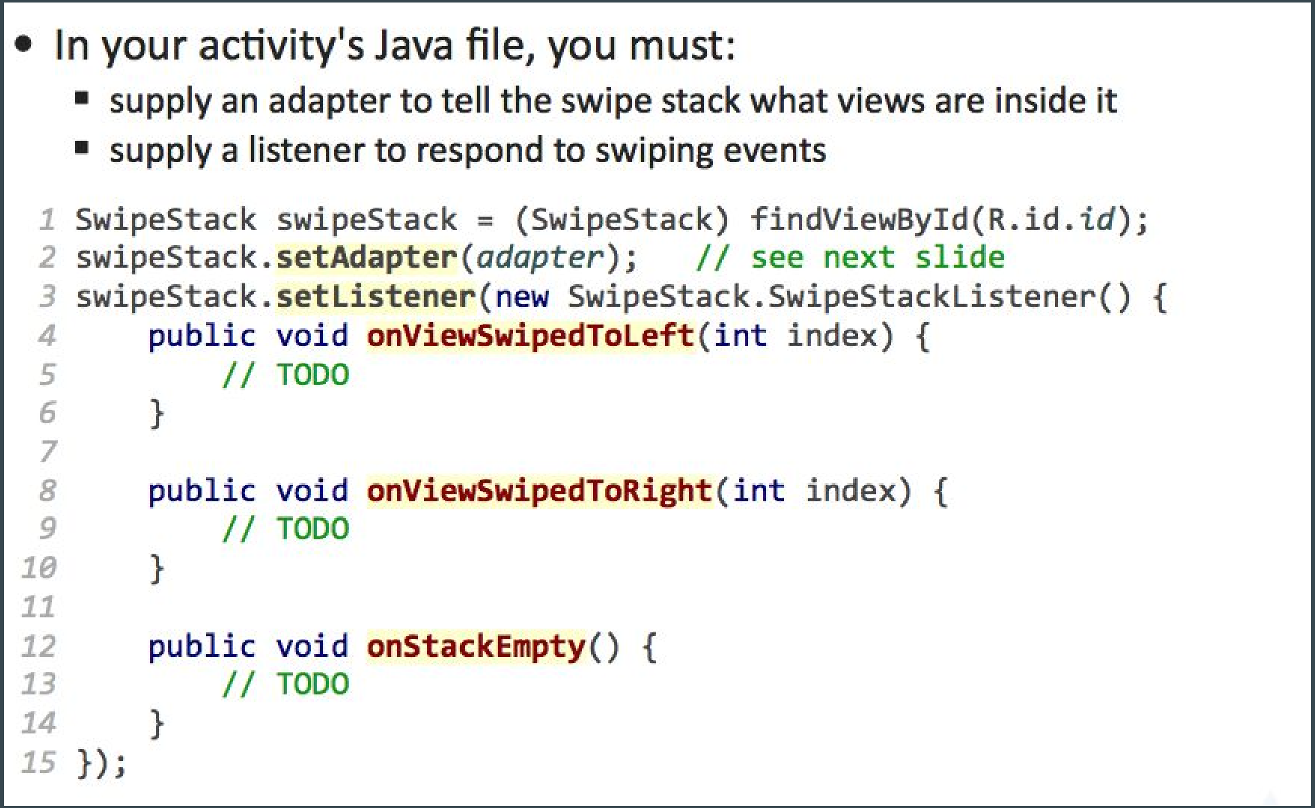
Using a SwipeStack (XML)



- In your layout XML file:

```xml
<!-- declare an empty swipe stack -->
<link.fls.SwipeStack
    android:id="@+id/id"
    android:layout_width="width"
    android:layout_height="height" />
```

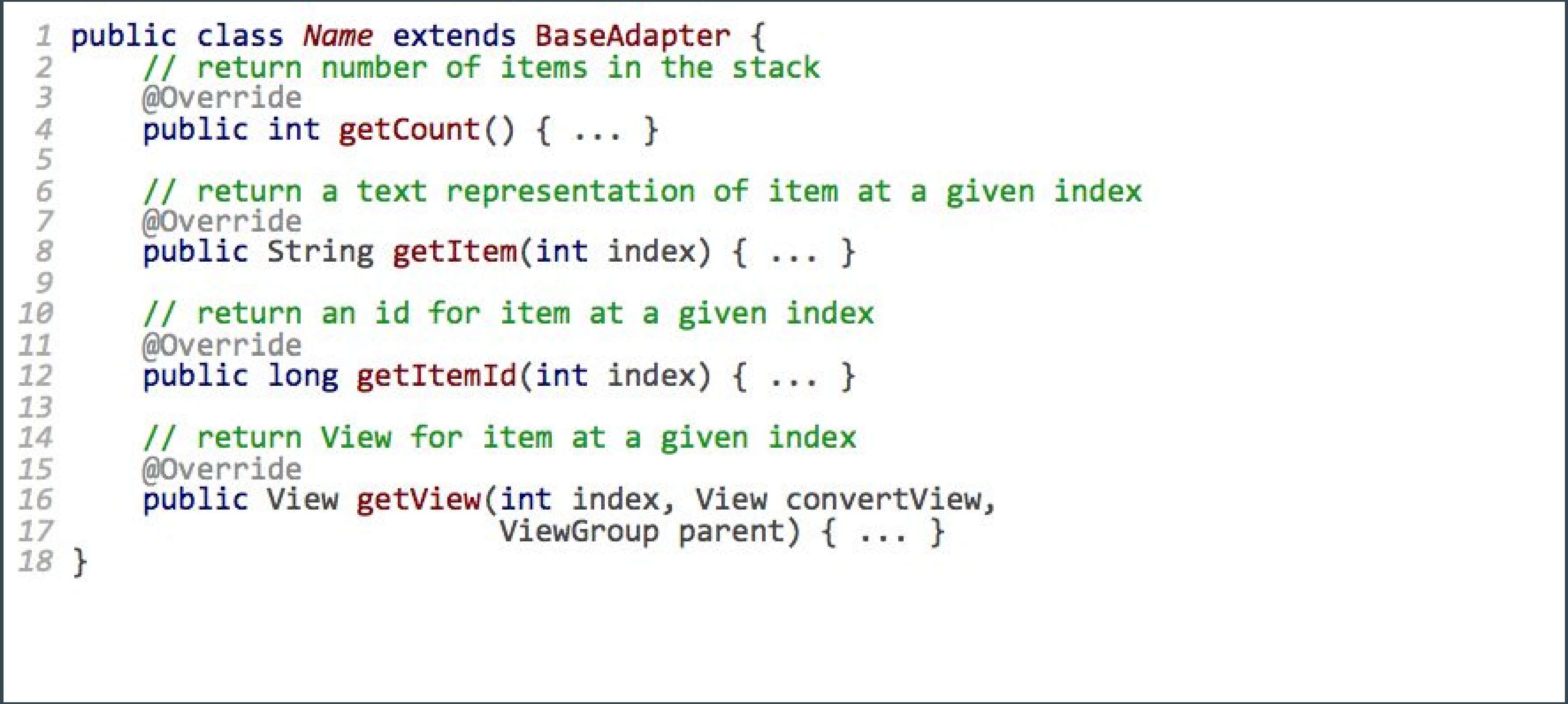
Using a SwipeStack (Java)



- In your activity's Java file, you must:
  - supply an adapter to tell the swipe stack what views are inside it
  - supply a listener to respond to swiping events

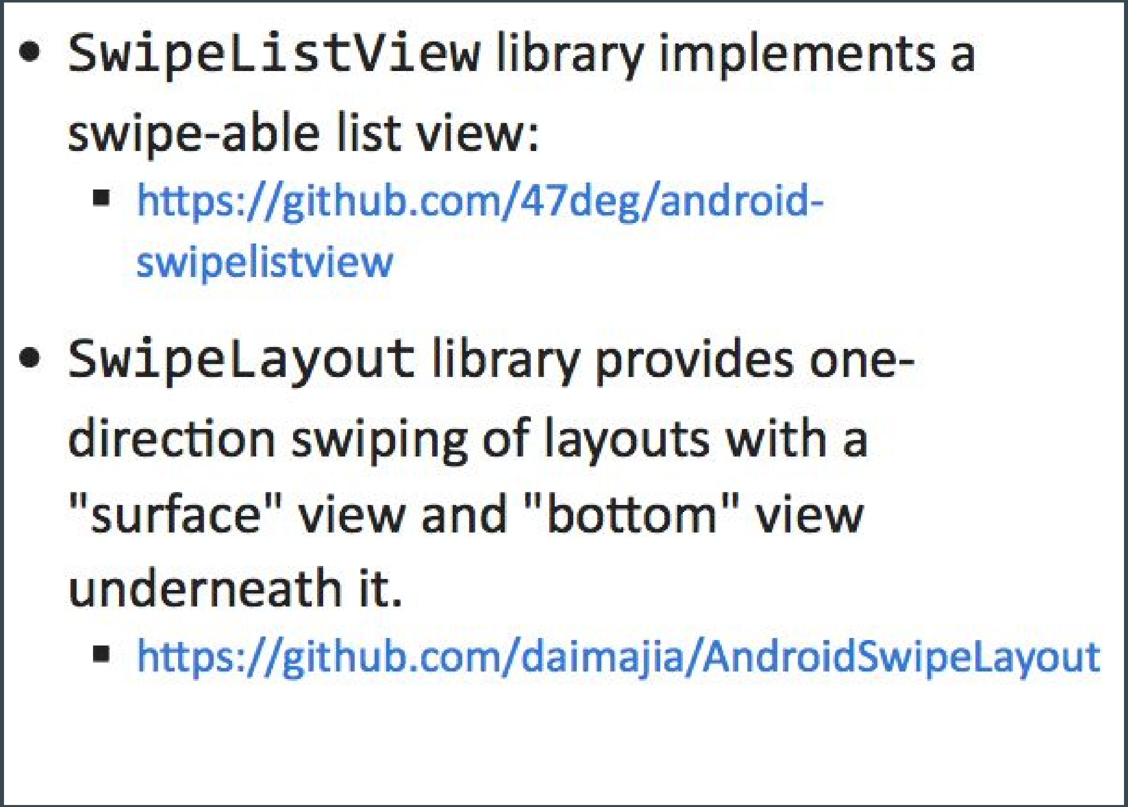
```java
SwipeStack swipeStack = (SwipeStack) findViewById(R.id.id);
swipeStack.setAdapter(adapter);   // see next slide
swipeStack.setListener(new SwipeStack.SwipeStackListener() {
    public void onViewSwipedToLeft(int index) {
        // TODO
    }

    public void onViewSwipedToRight(int index) {
        // TODO
    }

    public void onStackEmpty() {
        // TODO
    }
});
```

Writing an adapter class



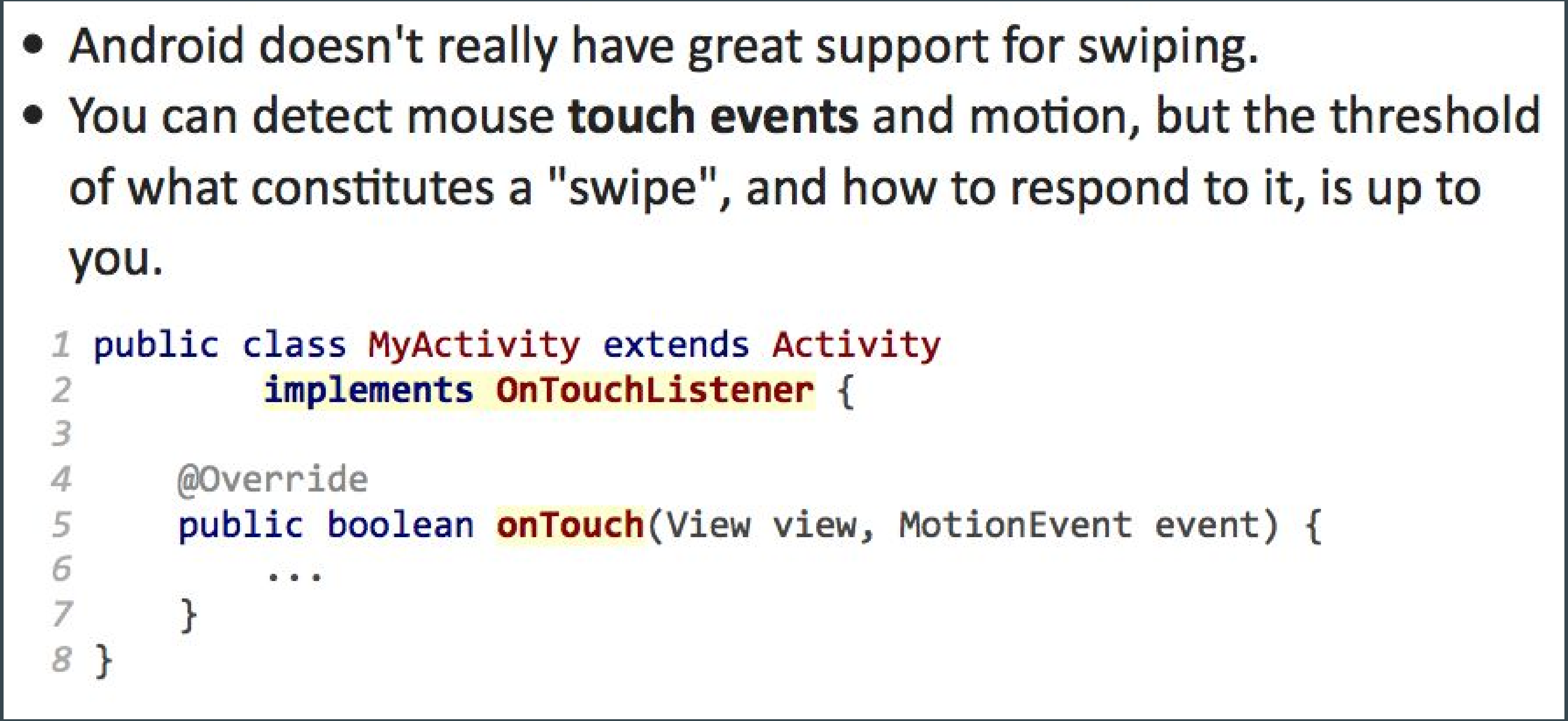
```java
public class Name extends BaseAdapter {
    // return number of items in the stack
    @Override
    public int getCount() { ... }

    // return a text representation of item at a given index
    @Override
    public String getItem(int index) { ... }

    // return an id for item at a given index
    @Override
    public long getItemId(int index) { ... }

    // return View for item at a given index
    @Override
    public View getView(int index, View convertView,
                        ViewGroup parent) { ... }
}
```

Other swiping libraries



- `SwipeListView` library implements a swipe-able list view:
  - https://github.com/47deg/android-swipelistview
- `SwipeLayout` library provides one-direction swiping of layouts with a "surface" view and "bottom" view underneath it.
  - https://github.com/daimajia/AndroidSwipeLayout

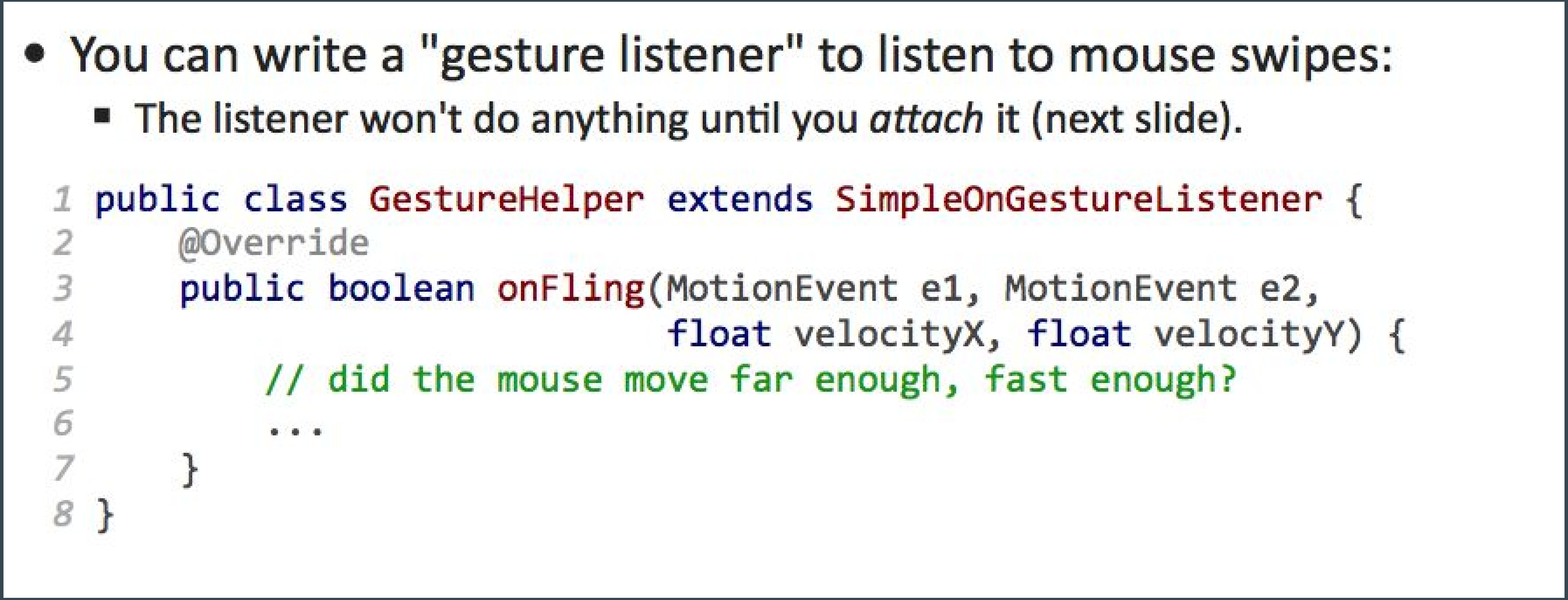
Swipe Support in Android



- Android doesn't really have great support for swiping.
- You can detect mouse **touch events** and motion, but the threshold of what constitutes a "swipe", and how to respond to it, is up to you.

```java
public class MyActivity extends Activity
          implements OnTouchListener {

    @Override
    public boolean onTouch(View view, MotionEvent event) {
        ...
    }
}
```

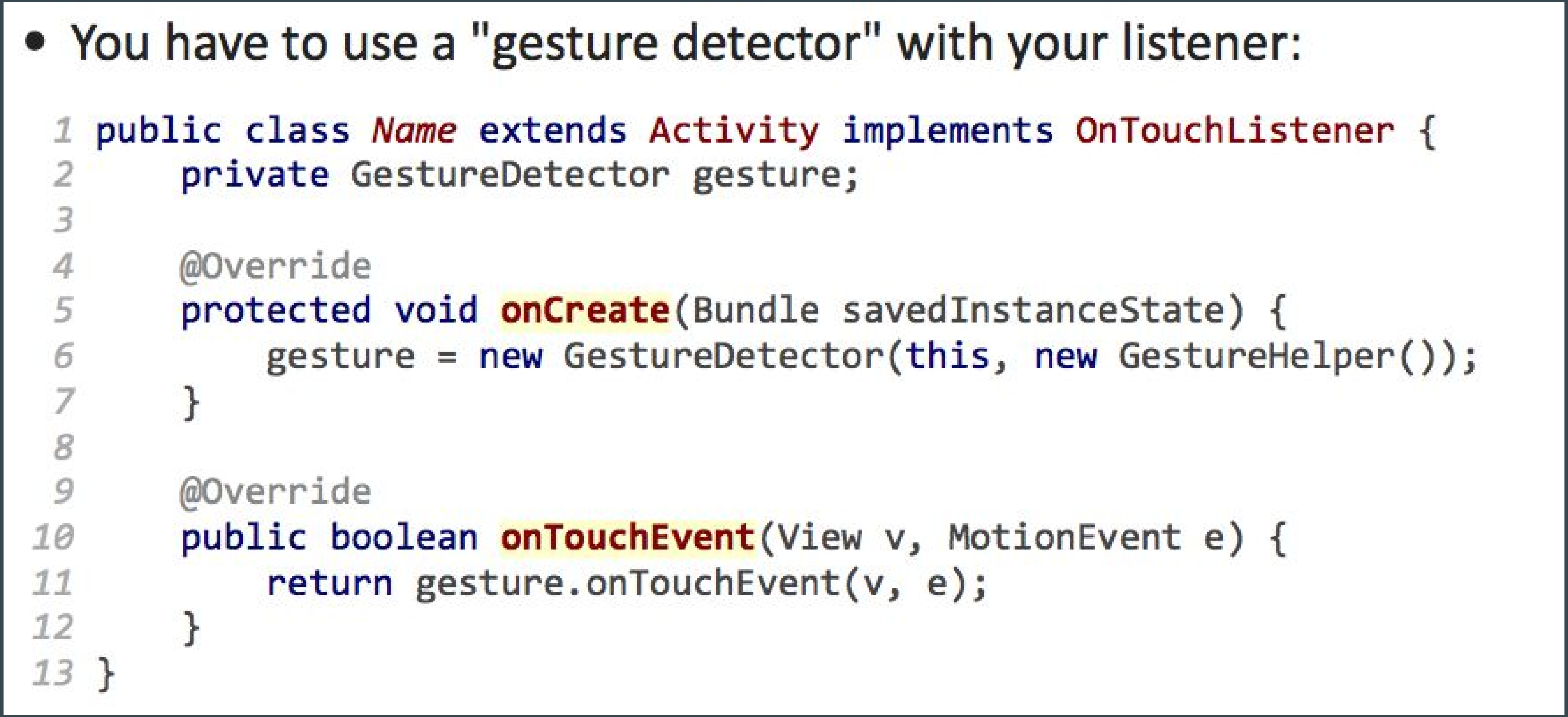
A gesture listener



- You can write a "gesture listener" to listen to mouse swipes:
  - The listener won't do anything until you *attach* it (next slide).

```java
public class GestureHelper extends SimpleOnGestureListener {
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2,
                           float velocityX, float velocityY) {
        // did the mouse move far enough, fast enough?
        ...
    }
}
```

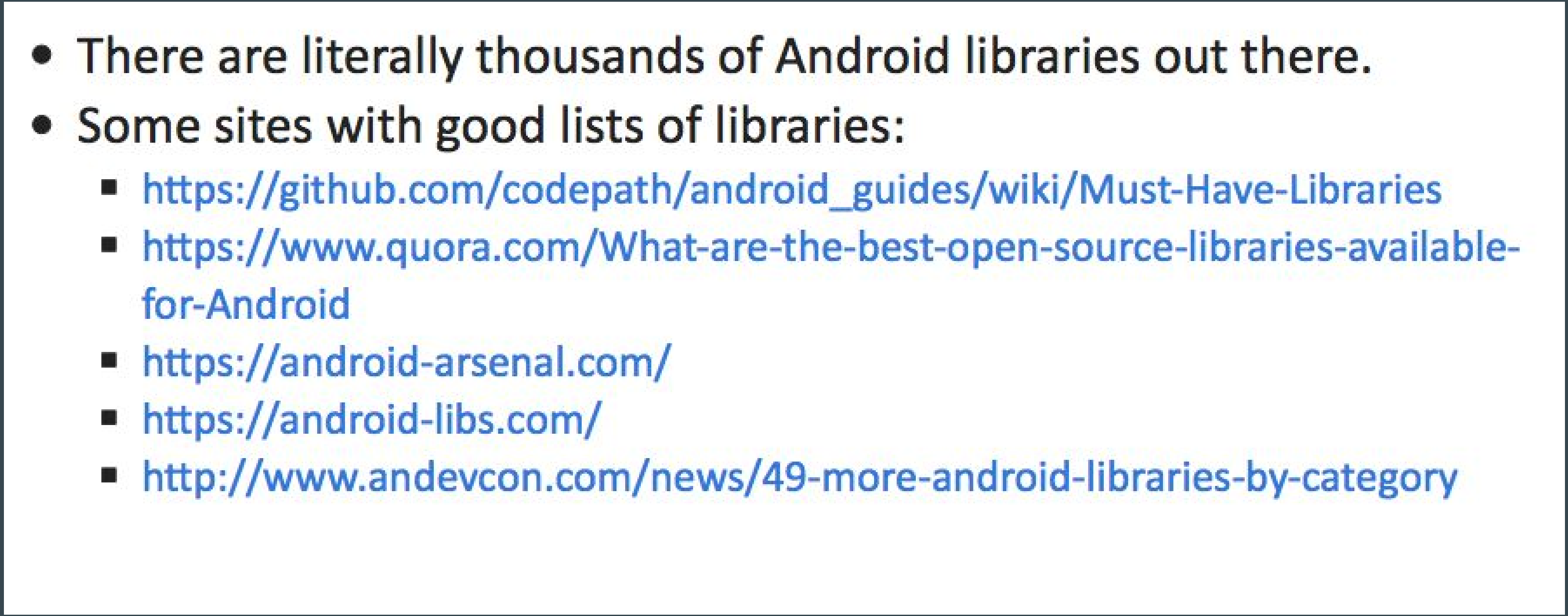
Listening for swipe gesture



- You have to use a "gesture detector" with your listener:

```java
public class Name extends Activity implements OnTouchListener {
    private GestureDetector gesture;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        gesture = new GestureDetector(this, new GestureHelper());
    }

    @Override
    public boolean onTouchEvent(View v, MotionEvent e) {
        return gesture.onTouchEvent(v, e);
    }
}
```

Other useful libraries



- There are literally thousands of Android libraries out there.
- Some sites with good lists of libraries:
  - https://github.com/codepath/android_guides/wiki/Must-Have-Libraries
  - https://www.quora.com/What-are-the-best-open-source-libraries-available-for-Android
  - https://android-arsenal.com/
  - https://android-libs.com/
  - http://www.andevcon.com/news/49-more-android-libraries-by-category