CSc 317
Resources and IDs

Benjamin Dicken

Announcements

- PA 1 due this evening
- PA 2
- Quiz 2

Four Main Building blocks of apps

- **Activities** - The entry point for interacting with the user. It represents a single screen with a user interface.
- **Services** - A general-purpose entry point for keeping an app running in the background for all kinds of reasons. For instance, a download or music.
- **Broadcast Receivers** - A component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements.
- **Content providers** - Manages a shared set of app data.

Loosely taken from the android dev docs

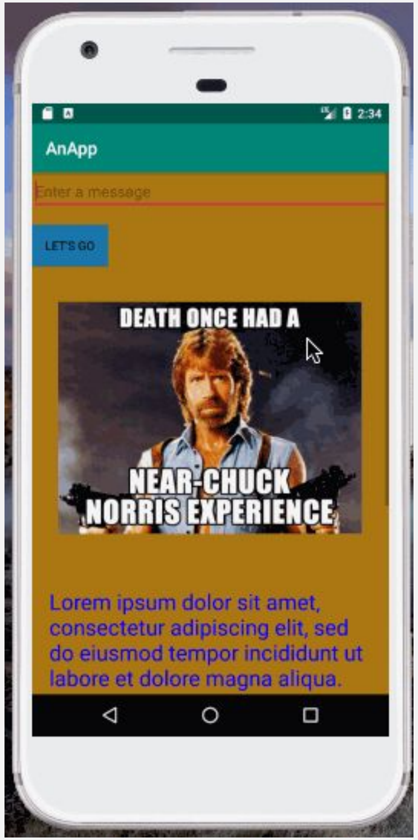
Activities

- Activities are one of the **building blocks** of android applications
- From the reading:

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle.

Activities

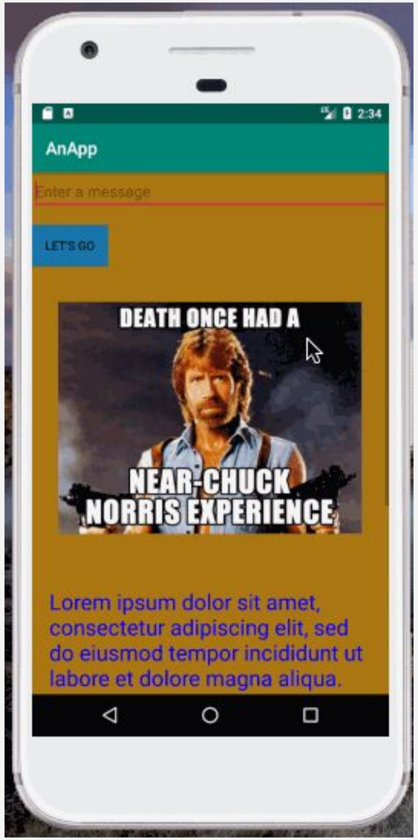
- For now, each main interface screen of your application will be built using an **Activity**
- A simple, one-page app might only need one activity
- Some apps may have many
- Each activity that has a UI should have an associated layout xml file



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Activities

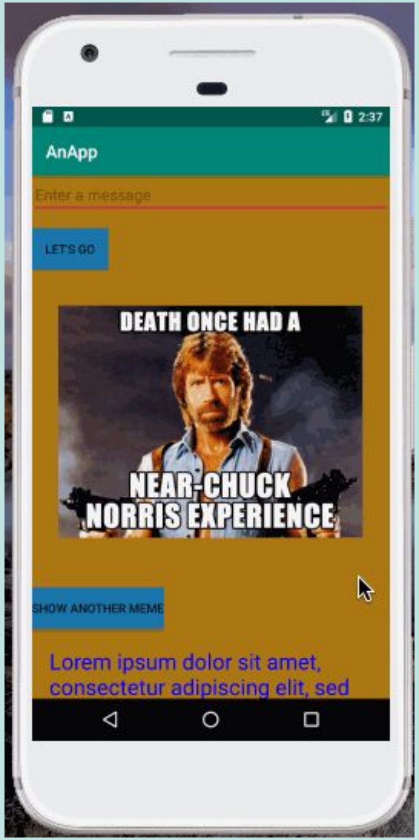
- How does one create an activity?
- How does one change from one active activity to another?



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

A new activity

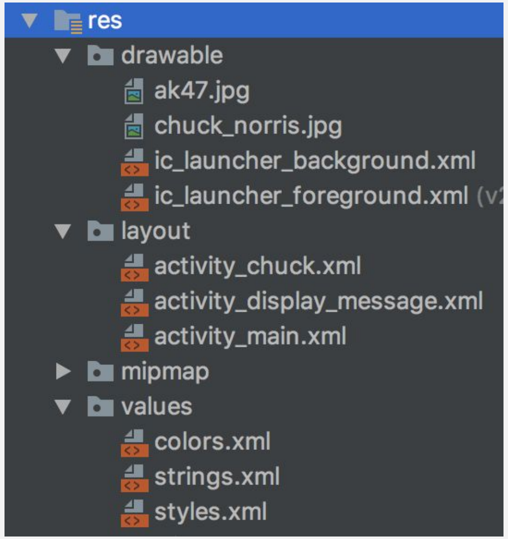
- Create a new activity, and call it **MemeActivity**
  - Two new files should be created: **`MemeActivity.java`** and **`activity_meme.xml`**
- The new view should display one meme of your choice
- Do this in code/xml, not with the GUI editor



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed

Resources

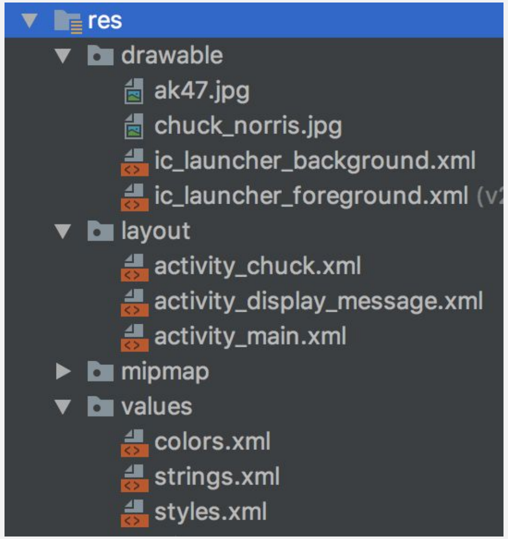
- Utilizing resources is key to creating a well-engineered application
- Can break up your application into two main categories:
  - ***Code:*** for logic
  - ***Resources:*** Use for string, images, colors, animations, UI, layout, etc.
    - Don’t try to do those things in code, unless necessary



```
▼ res
  ▼ drawable
      ak47.jpg
      chuck_norris.jpg
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v
  ▼ layout
      activity_chuck.xml
      activity_display_message.xml
      activity_main.xml
  ▶ mipmap
  ▼ values
      colors.xml
      strings.xml
      styles.xml
```

Resources

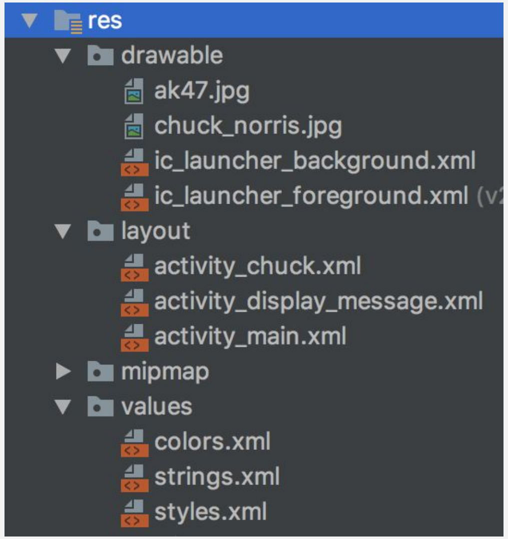
- Have already come across some examples of resources:
  - Images
  - Strings
  - Colors



```
▼ res
    ▼ drawable
          ak47.jpg
          chuck_norris.jpg
          ic_launcher_background.xml
          ic_launcher_foreground.xml (v
    ▼ layout
          activity_chuck.xml
          activity_display_message.xml
          activity_main.xml
    ▶ mipmap
    ▼ values
          colors.xml
          strings.xml
          styles.xml
```

Alternative Resources

- Can provide alternate versions of the "same" resource for differences in:
  - Screen density
  - language/region
  - Layout direction
  - UI mode (car, watch, TV, etc)
  - Mode (day or night)
- Specified via qualifiers at the end of the file or directory



```
res
  drawable
    ak47.jpg
    chuck_norris.jpg
    ic_launcher_background.xml
    ic_launcher_foreground.xml (v
  layout
    activity_chuck.xml
    activity_display_message.xml
    activity_main.xml
  mipmap
  values
    colors.xml
    strings.xml
    styles.xml
```

For instance, screen orientation

- Can have different resources files for landscape and portrait modes

ICA

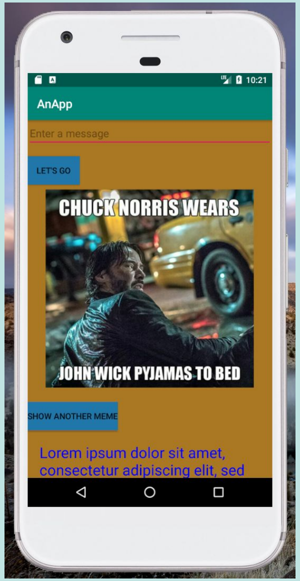
Multiple resources

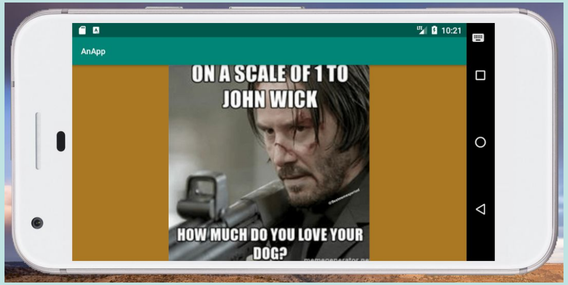
- First step: search online for two images of roughly the same proportion
- Save to your Desktop

ICA

Multiple resources - portrait/land

- Change the app so it displays another image in the main activity, except:
  - The image should *change* depending on the screen orientation
  - No need to write *code* to do this
  - When dragging the files to the drawable directory, add either **-port** or **-land** to the end





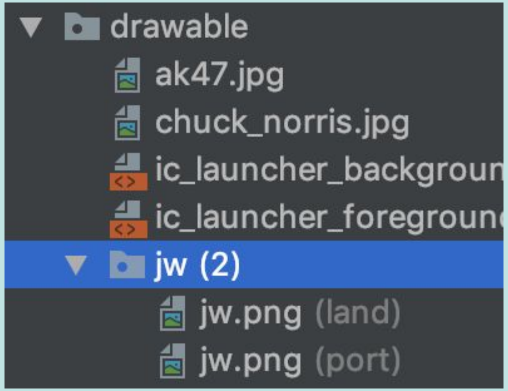
ICA

Multiple resources - portrait/land

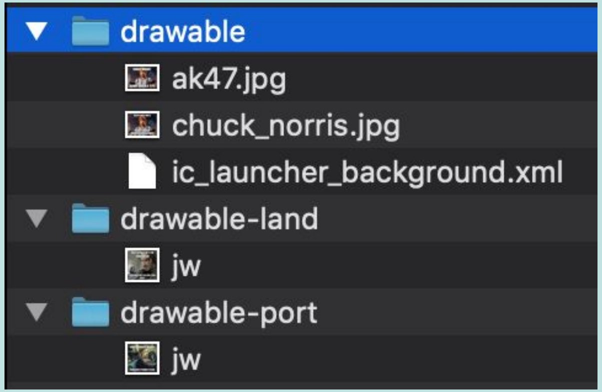
Take a look at the directory structure in

- Android Studio
- Finder / Windows explorer

What's the difference?



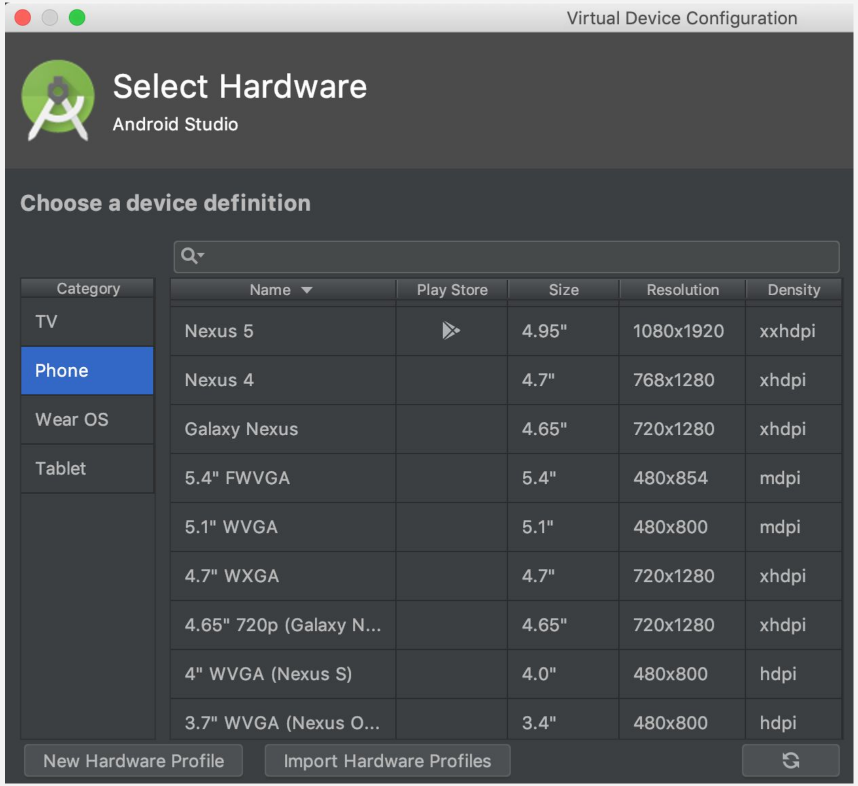
```
▼ drawable
    ak47.jpg
    chuck_norris.jpg
    ic_launcher_backgroun
    ic_launcher_foregroun
  ▼ jw (2)
      jw.png (land)
      jw.png (port)
```



```
▼ drawable
    ak47.jpg
    chuck_norris.jpg
    ic_launcher_background.xml
▼ drawable-land
    jw
▼ drawable-port
    jw
```

For instance, Resolution

- Various densities
  - xxhdpi
  - xhdpi
  - mdpi
  - hdpi
  - ldpi



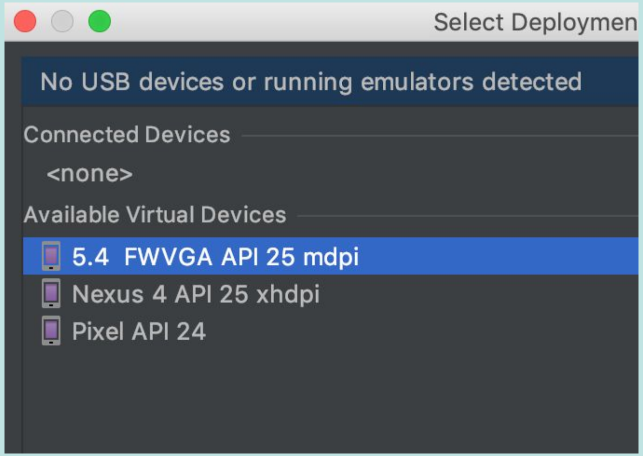
Choose a device definition

Name	Play Store	Size	Resolution	Density
Nexus 5	▶	4.95"	1080x1920	xxhdpi
Nexus 4		4.7"	768x1280	xhdpi
Galaxy Nexus		4.65"	720x1280	xhdpi
5.4" FWVGA		5.4"	480x854	mdpi
5.1" WVGA		5.1"	480x800	mdpi
4.7" WXGA		4.7"	720x1280	xhdpi
4.65" 720p (Galaxy N...		4.65"	720x1280	xhdpi
4" WVGA (Nexus S)		4.0"	480x800	hdpi
3.7" WVGA (Nexus O...		3.4"	480x800	hdpi

ICA

Multiple resources (might have to watch)

- Create two new virtual devices
  - One with xhdpi, another with mdpi
- Create two of the “same” image resources, one for each resolution type
- Add or update code so that image displays
- Try running the application on both virtual devices
  - https://stackoverflow.com/questions/5099550/how-to-check-an-android-device-is-hdpi-screen-or-mdpi-screen


Select Deploymen

No USB devices or running emulators detected

Connected Devices

<none>

Available Virtual Devices

5.4 FWVGA API 25 mdpi

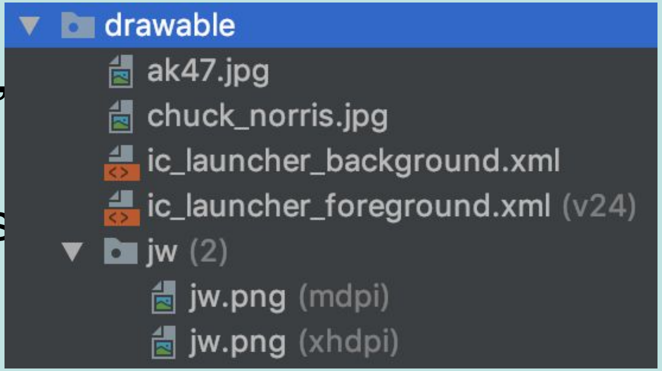
Nexus 4 API 25 xhdpi

Pixel API 24

ICA

Multiple resources (might have to watch)

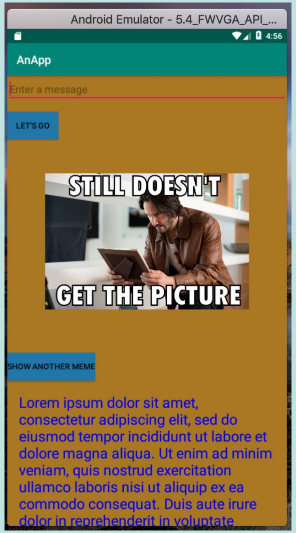
- Create two new virtual devices
  - One with xhdpi, another with mdpi
- Create two of the “same” image resources, one for each resolution type
- Add or update code so that image displays
- Try running the application on both virtual devices
  - https://stackoverflow.com/questions/5099550/how-to-check-an-android-device-is-hdpi-screen-or-mdpi-screen



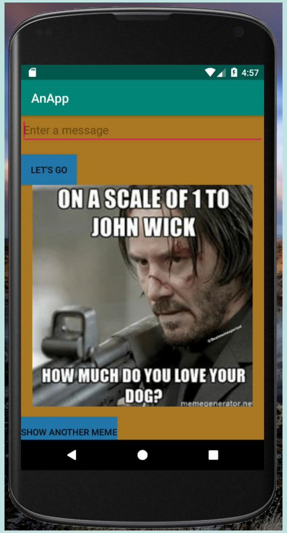
```
drawable
    ak47.jpg
    chuck_norris.jpg
    ic_launcher_background.xml
    ic_launcher_foreground.xml (v24)
  jw (2)
      jw.png (mdpi)
      jw.png (xhdpi)
```

Multiple resources (might have to watch)

- Create two new virtual devices
  - One with xhdpi, another with mdpi
- Create two of the “same” image resources, one for each resolution type
- Add or update code so that image displays
- Try running the application on both virtual devices
  - https://stackoverflow.com/questions/5099550/how-to-check-an-android-device-is-hdpi-screen-or-mdpi-screen



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate



Identifiers

- Every UI element **should have an ID**

```
<ImageView
    android:id="@+id/team_mascot_image" . . .
```

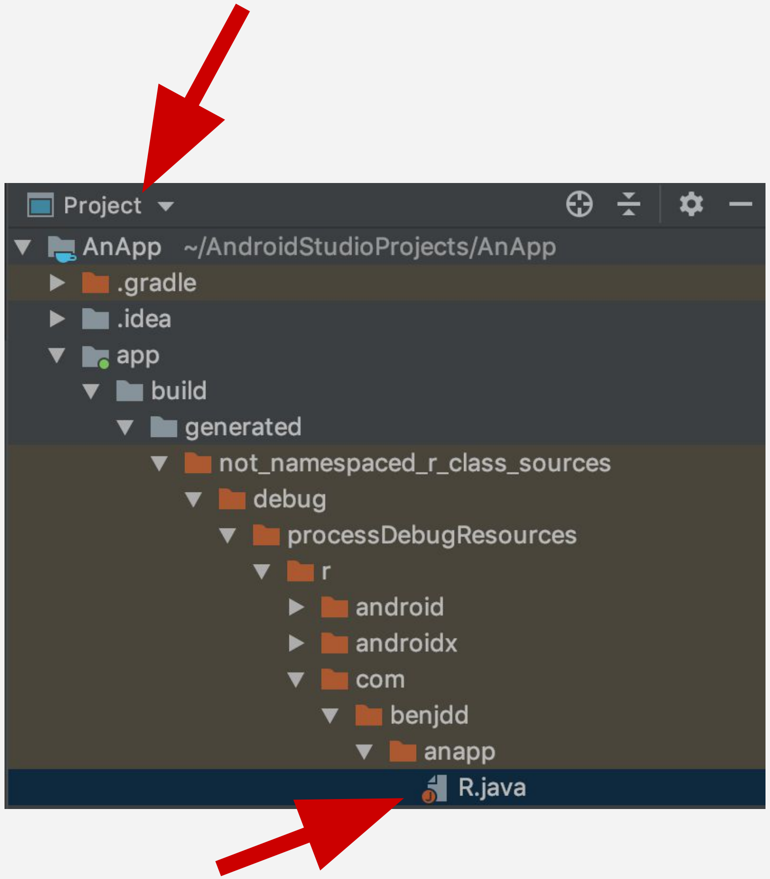
- Even if you don’t know if you’re going to need it, always give it one

Why IDs?

- All types of resources (UI elements, strings, colors, images, etc) should have a unique identifier
    - For values, use the **name attribute**
    - For drawables, the **file name**
    - For UI elements, the **android:id** attribute
- The ID can be used to reference the resource when you want to use or update it

The **R** class

- The build process automatically generates a file called **R.java**
- This files has a bunch of **`public static final`** values
- Identifiers for the various resources and UI elements are placed here



```
Project ▾
▼ AnApp  ~/AndroidStudioProjects/AnApp
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▼ build
      ▼ generated
        ▼ not_namespaced_r_class_sources
          ▼ debug
            ▼ processDebugResources
              ▼ r
                ▶ android
                ▶ androidx
                ▼ com
                  ▼ benjdd
                    ▼ anapp
                        R.java
```

R use 1 - MainActivity.java

```java
Intent intent = new Intent(this, DisplayMessageActivity.class);
EditText editText = (EditText) findViewById(R.id.editText);
String message = editText.getText().toString();
intent.putExtra(EXTRA_MESSAGE, message);
startActivity(intent);
```

R use 2 - DisplayMessageActivity.java

```java
// Get the Intent that started this activity and extract the string
Intent intent = getIntent();
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

// Capture the layout's TextView and set the string as its text
TextView textView = findViewById(R.id.textView);
textView.setText(message);
```

If time permits

- Try experimenting with a set of alternative resources of your own
    - Not including the ones already shown