

5COSC005W MOBILE APPLICATION DEVELOPMENT

Lecture 1: Introduction to Android

Dr Dimitris C. Dracopoulos

Module Web page:

<https://dracopd.users.ecs.westminster.ac.uk/DOCUM/courses/5cosc005w/5cosc005w.html>

This material is based on **Android Developer Fundamentals 2** by Google used under
a Creative Commons Attribution 4.0 International license.

Introduction to the Module

- Syllabus
- Lectures (slides + theory concepts)
- Tutorials
- Software
- Assessment
- Schedule
- What is expected from you?
 - Lecture Attendance
 - Tutorial Attendance (actual not just logging in BB)
 - Completion of ALL Tutorial Exercises within the week (if not possible within the tutorial session then on your own time).
 - Code of Conduct

Code of Conduct

- Do not cheat on assignments (this is *INDIVIDUAL* work and **NOT** the product of collaboration!):
 - Discuss only general approaches not specific details of implementation
 - Do not take written notes on other's work and do not exchange code
- Cheating is reported to university and then it is out of the module lecturers hands (independent committee decision without the participation of the module tutors)
- Possible consequences:
 - A mark of 0 for assignment
 - A mark of 0 for the course
 - A permanent note on student record
 - Suspension/Expulsion from university

Code of Conduct (cont'ed)

- Any code found in the web or textbook and used in your work should be properly referenced in comments within your code.

Academic Integrity

- The University of Westminster is committed to the highest standards of academic integrity and honesty. Students are expected to be familiar with these standards regarding academic honesty and to uphold the policies of the University in this respect. Students are particularly urged to familiarize themselves with the provisions of the Academic Regulations and in this case with Academic Misconduct Regulations (<https://www.westminster.ac.uk/sites/default/public-files/general-documents/Section-10-Academic-Misconduct-v2.pdf>) and avoid any behavior which could potentially result in suspicions of cheating, plagiarism, misrepresentation of facts and/or participation in an offence. Academic dishonesty is a serious offence and can result in suspension or expulsion from the University.

Android Developer Fundamentals V2

Build your first app

Lesson 1

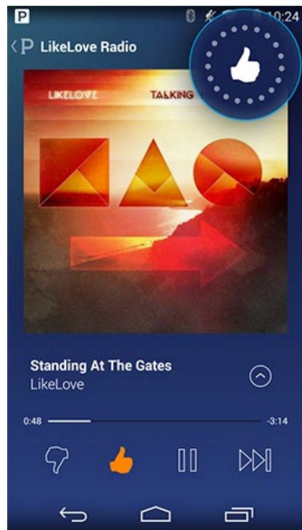


1.0 Introduction to Android

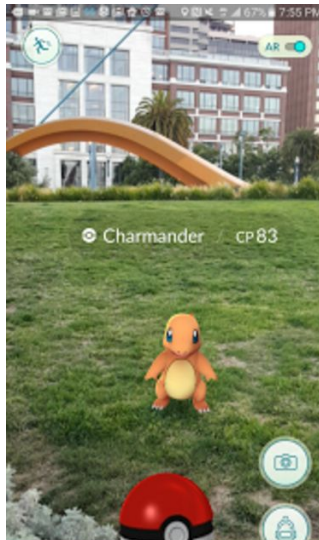
What is Android?

- Mobile operating system based on [Linux kernel](#)
- User Interface for touch screens
- Used on [over 80%](#) of all smartphones
- Powers devices such as watches, TVs, and cars
- Over 2 Million Android apps in Google Play store
- Highly customizable for devices / by vendors
- Open source

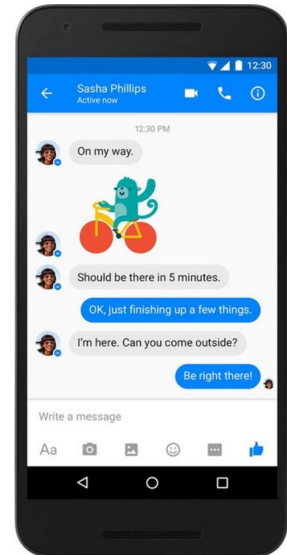
Android app examples



Pandora



Pokemon GO

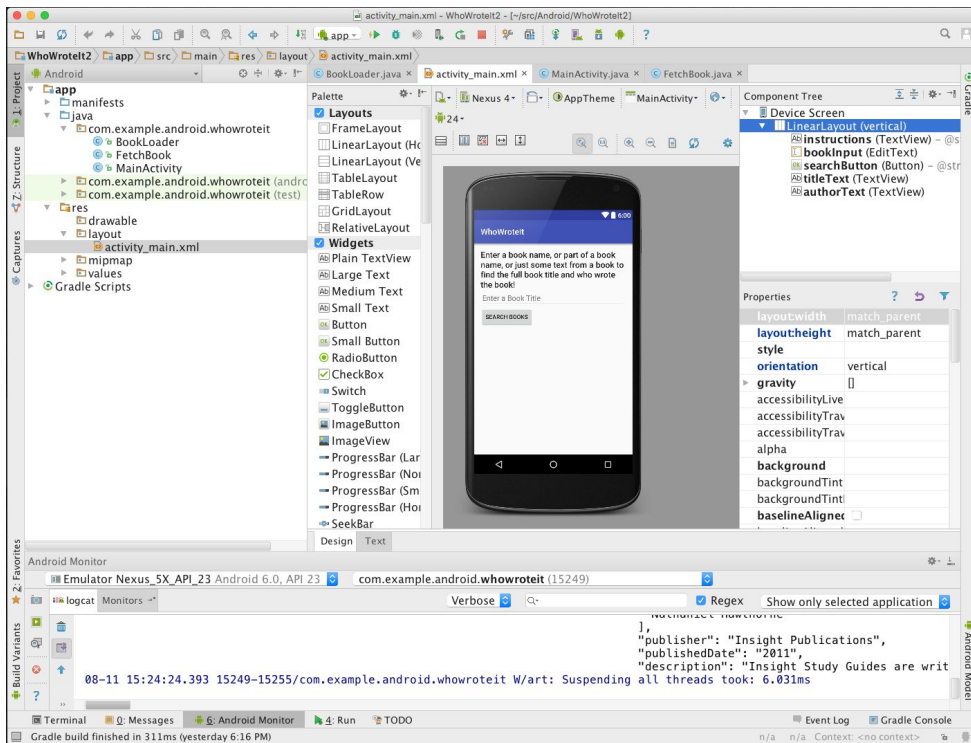


Facebook
Messenger

Android Software Developer Kit (SDK)

- Development tools (debugger, monitors, editors)
- Libraries (maps, wearables)
- Virtual devices (emulators)
- Documentation (developer.android.com)
- Sample code

Android Studio

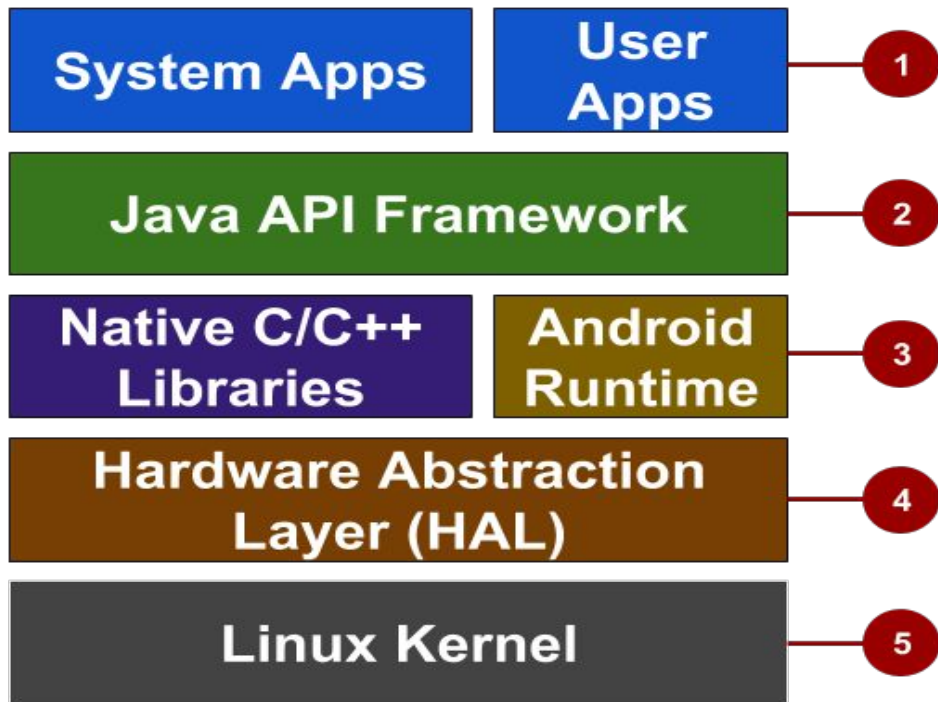


- [Official Android IDE](#)
- Develop, run, debug, test, and package apps
- Monitors and performance tools
- Virtual devices
- Project views
- Visual layout editor

Android Platform Architecture

Android stack

1. System and user apps
2. Android OS API in Java framework
3. Expose native APIs; run apps
4. Expose device hardware capabilities
5. Linux Kernel



System and user apps



- System apps have no special status
- System apps provide key capabilities to app developers

Example:

Your app can use a system app to deliver a SMS message.

Java API Framework

The entire feature-set of the Android OS is available to you through APIs written in the Java language.

- View class hierarchy to create UI screens
- Notification manager
- Activity manager for life cycles and navigation

Android runtime

Each app runs in its own process with its own instance of the Android Runtime.

C/C++ libraries

- Core C/C++ Libraries give access to core native Android system components and services.

Hardware Abstraction Layer (HAL)

- Standard interfaces that expose device hardware capabilities as libraries

Examples: Camera, bluetooth module

Linux Kernel

- Threading and low-level memory management
- Security features
- Drivers

Older Android versions



Codename	Version	Released	API Level
<i>Honeycomb</i>	3.0 - 3.2.6	Feb 2011	11 - 13
<i>Ice Cream Sandwich</i>	4.0 - 4.0.4	Oct 2011	14 - 15
<i>Jelly Bean</i>	4.1 - 4.3.1	July 2012	16 - 18
<i>KitKat</i>	4.4 - 4.4.4	Oct 2013	19 - 20
<i>Lollipop</i>	5.0 - 5.1.1	Nov 2014	21 - 22

[Android History](#) and [Platform Versions](#) for more and earlier versions before 2011

Newer Android versions



Codename	Version	Released	API Level
<i>Marshmallow</i>	6.0 - 6.0.1	Oct 2015	23
<i>Nougat</i>	7.0 - 7.1	Sept 2016	24 - 25
<i>Oreo</i>	8.0 - 8.1	Sept 2017	26 - 27
<i>Pie</i>	9.0	Aug 2018	28

App Development

What is an Android app?

- One or more interactive screens
- Written using [Java Programming Language](#) and [XML](#)
- Uses the Android Software Development Kit (SDK)
- Uses Android libraries and Android Application Framework
- Executed by Android Runtime Virtual machine (ART)

Challenges of Android development

- Multiple screen sizes and resolutions
- Performance: make your apps responsive and smooth
- Security: keep source code and user data safe
- Compatibility: run well on older platform versions
- Marketing: understand the market and your users
(Hint: It doesn't have to be expensive, but it can be.)

App building blocks

- Resources: layouts, images, strings, colors as XML and media files
- Components: activities, services, and helper classes as Java code
- Manifest: information about app for the runtime
- Build configuration: APK versions in Gradle config files

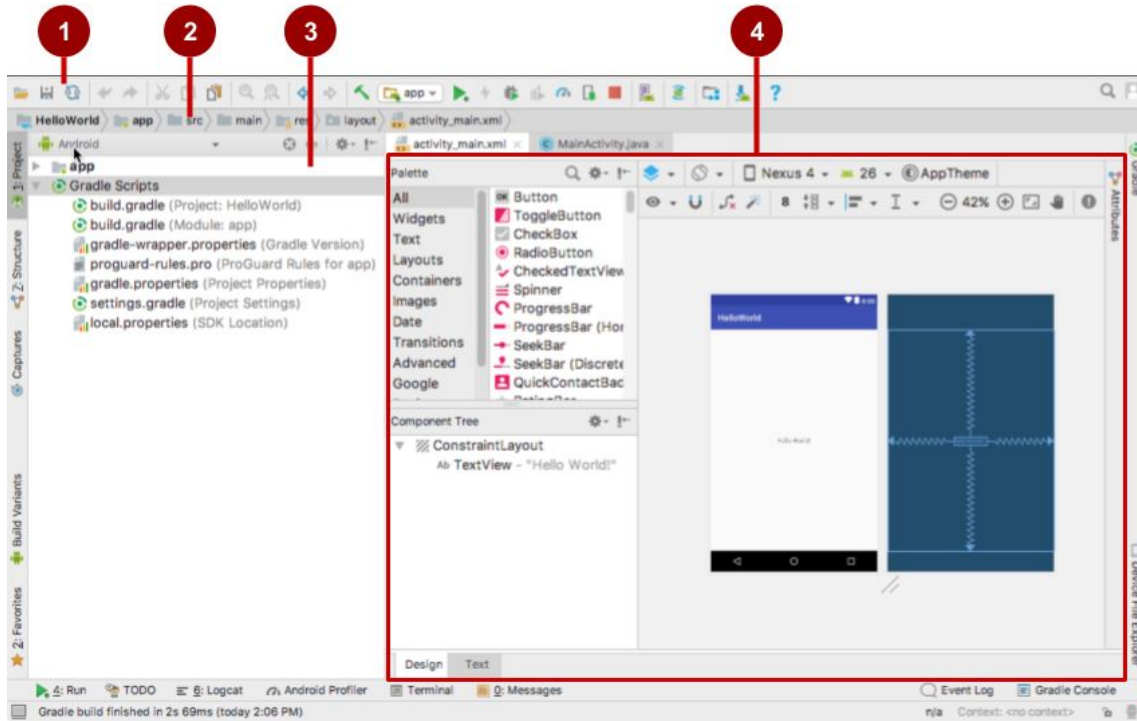
1.1 Your first Android app

Android Studio

What is Android Studio?

- Android integrated development environment (IDE)
- Project and Activity templates
- Layout editor
- Testing tools
- Gradle-based build
- Log console and debugger
- Emulators

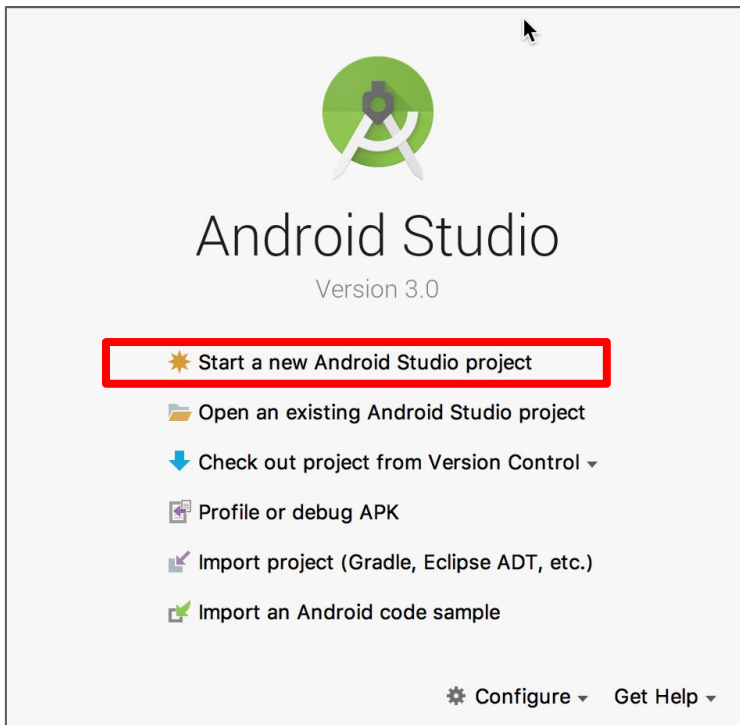
Android Studio interface



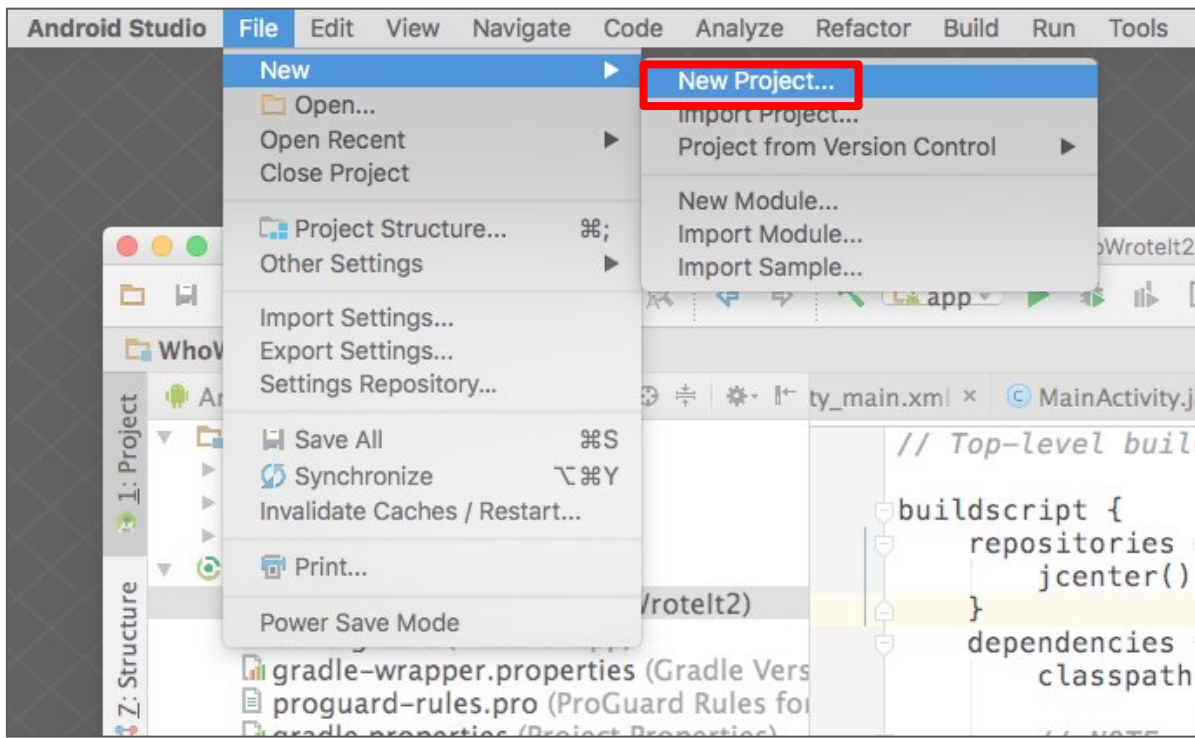
1. Toolbar
2. Navigation bar
3. Project pane
4. Editor
5. Tabs for other panes

Creating your first Android app

Start Android Studio



Create a project inside Android Studio



Name your app

Create New Project

Create Android Project

Application name
Hello World

Company domain
android.example.com

Project location
/Users/tbove/AndroidStudioProjects/HelloWorld

Package name
com.example.android.helloworld Edit

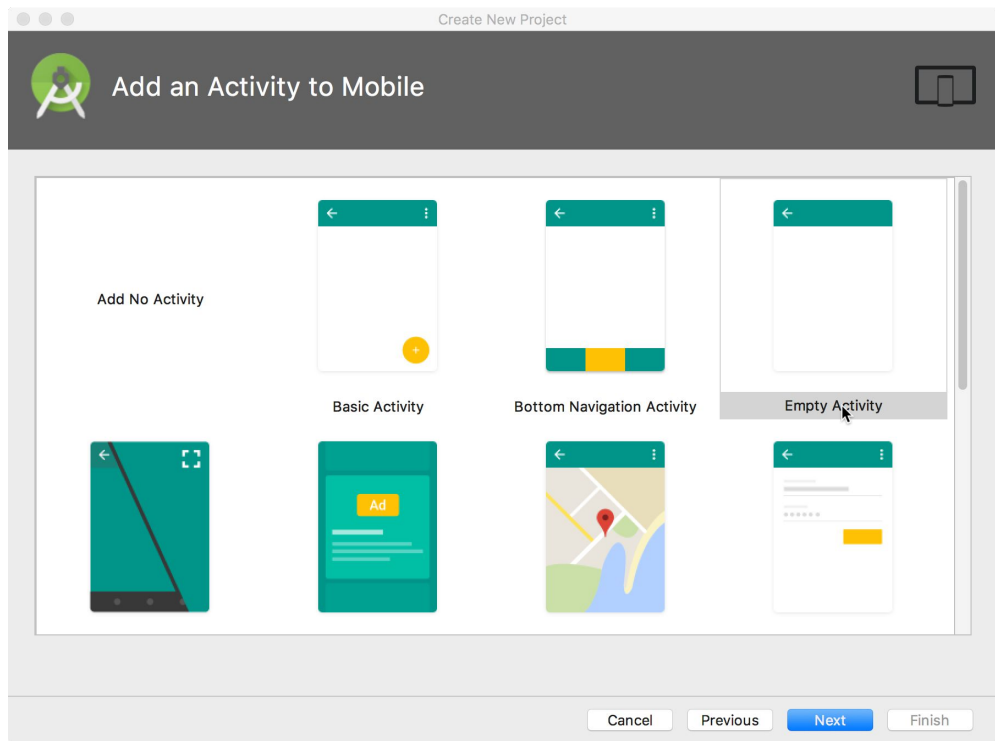
Include C++ support
 Include Kotlin support

Cancel Previous Next Finish

Pick activity template

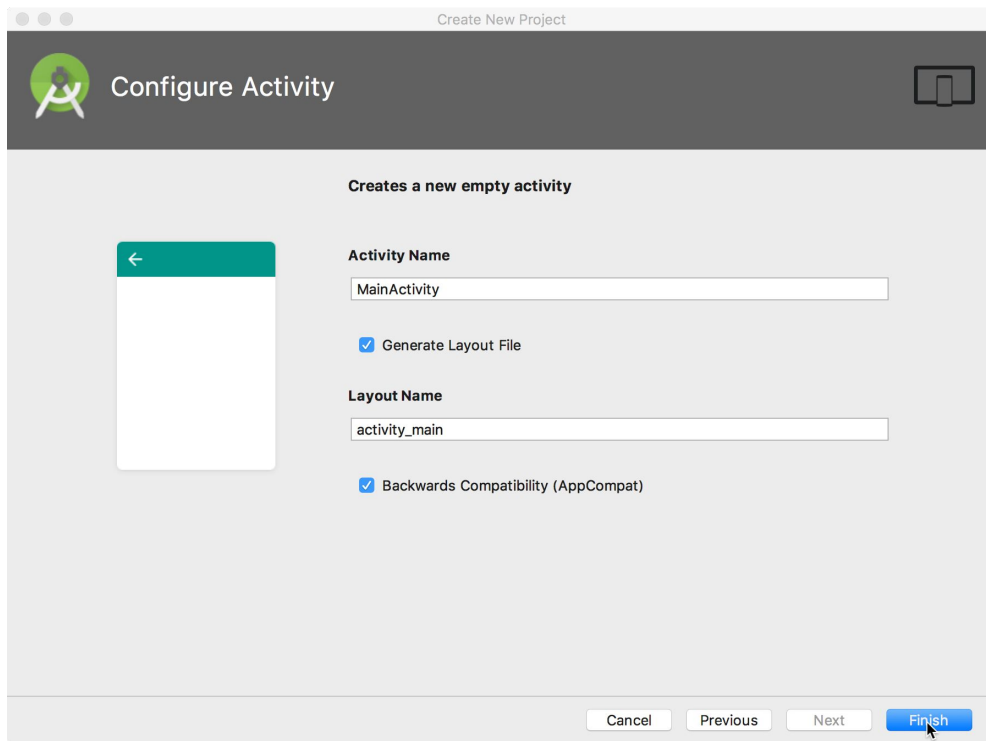
Choose templates for common activities, such as maps or navigation drawers.

Pick Empty Activity or Basic Activity for simple and custom activities.



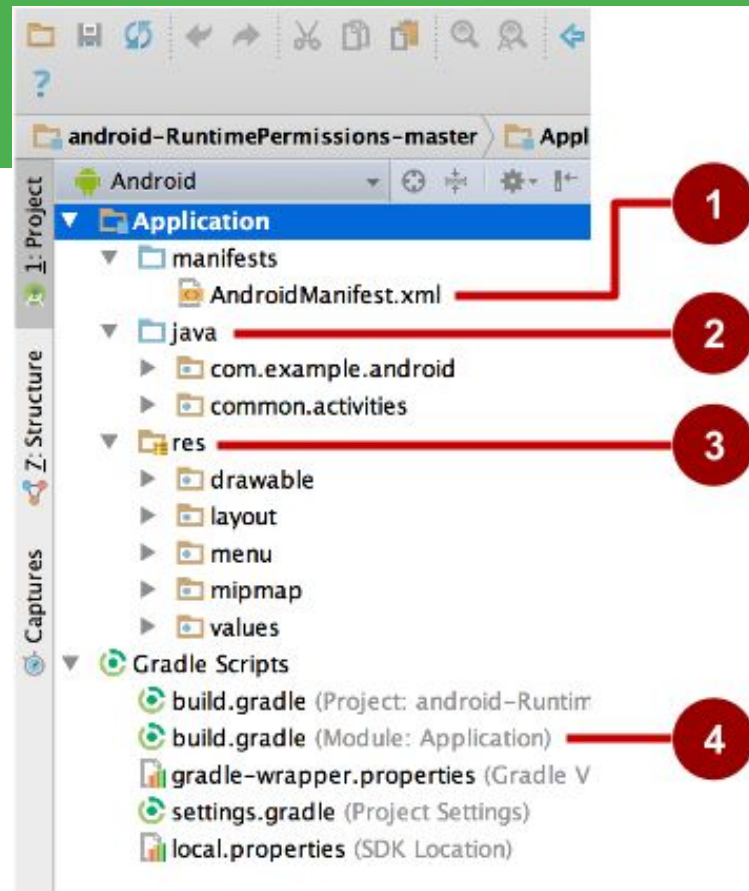
Name your activity

- Good practice:
 - Name main activity
MainActivity
 - Name layout
activity_main
- Use AppCompatActivity
- Generating layout file is convenient



Project folders

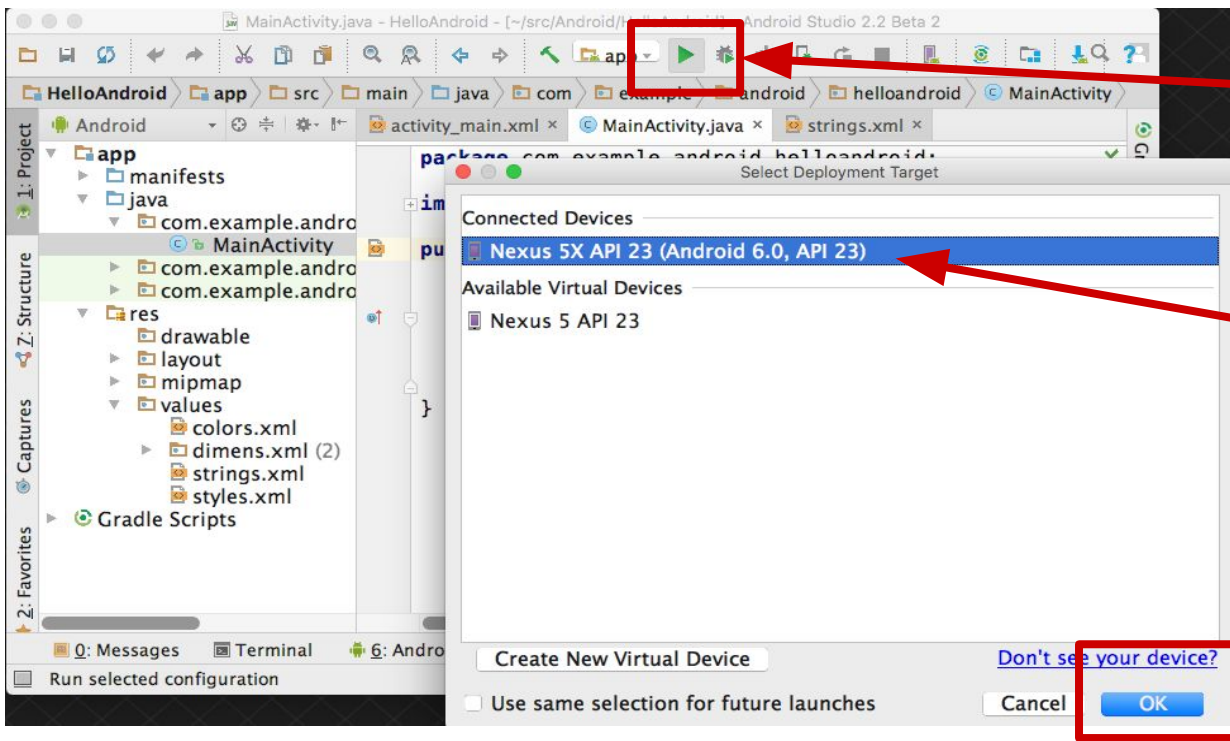
1. **manifests**—Android Manifest file - description of app read by the Android runtime
2. **java**—Java source code packages
3. **res**—Resources (XML) - layout, strings, images, dimensions, colors...
4. **build.gradle**—Gradle build files



Gradle build system

- Modern build subsystem in Android Studio
- Three build.gradle:
 - project
 - module
 - settings
- Typically not necessary to know low-level Gradle details
- Learn more about gradle at <https://gradle.org/>

Run your app



1. Run

2. Select virtual or physical device

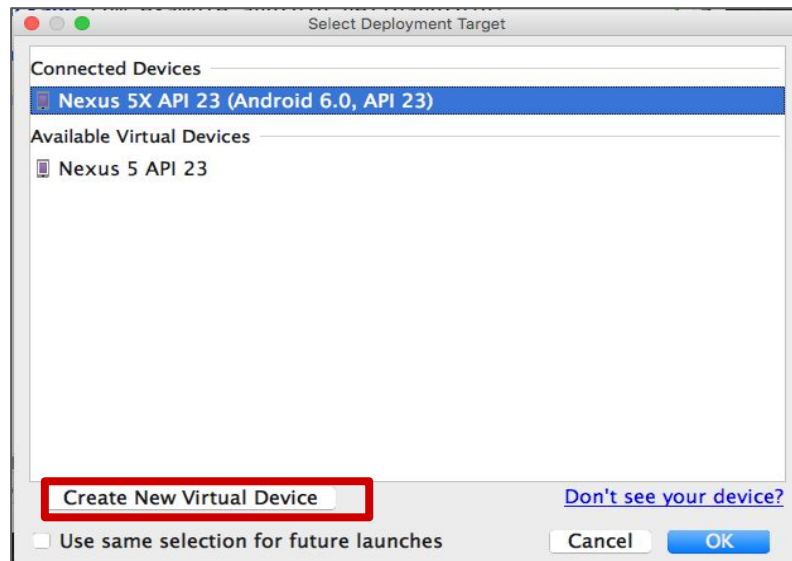
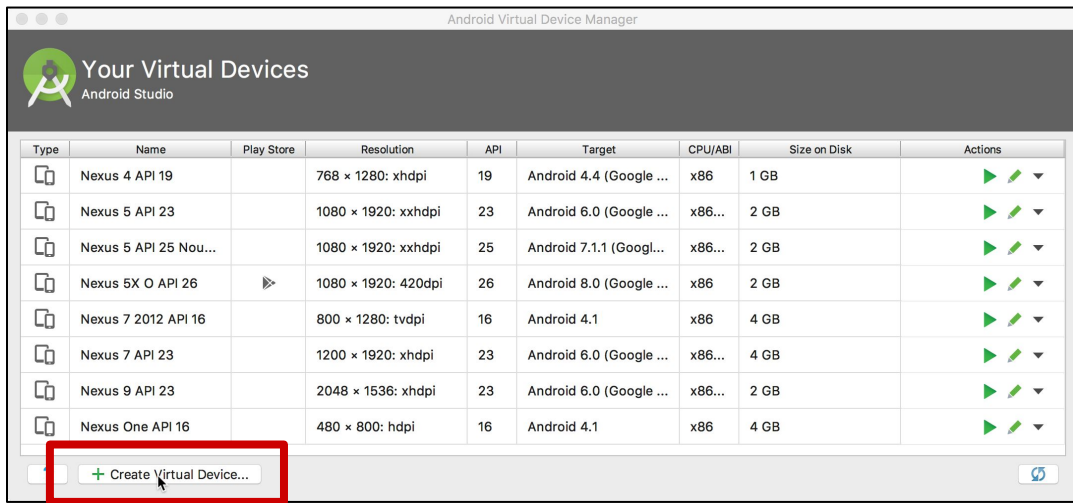
3. OK

Create a virtual device

Use emulators to test app on different versions of Android and form factors.

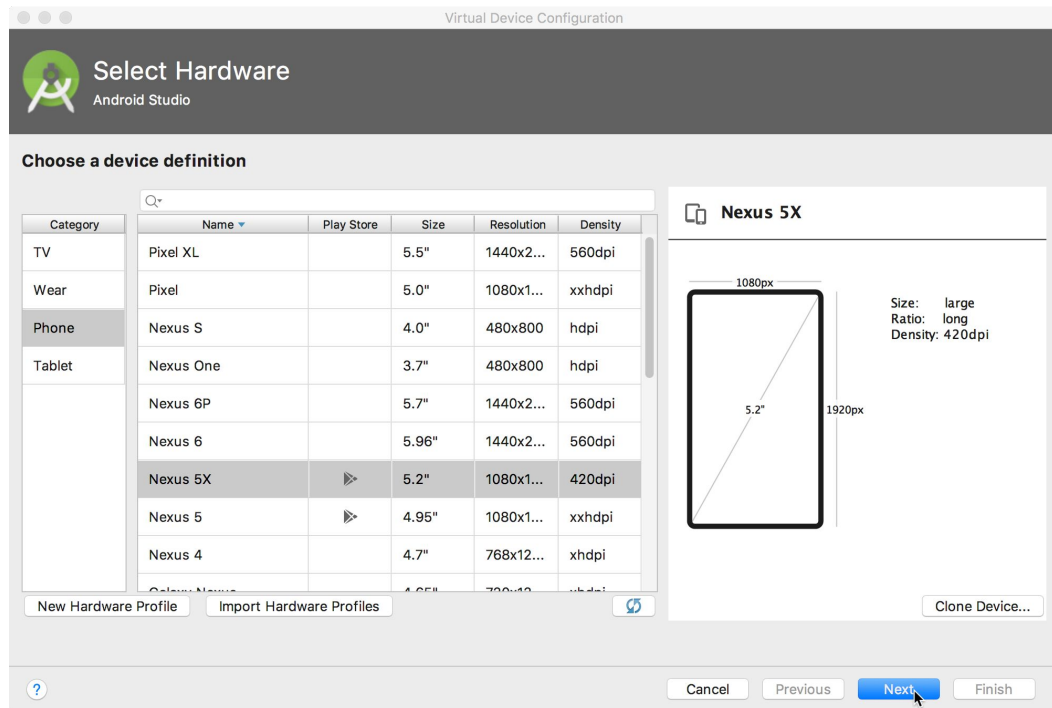
Tools > Android > AVD Manager

or:

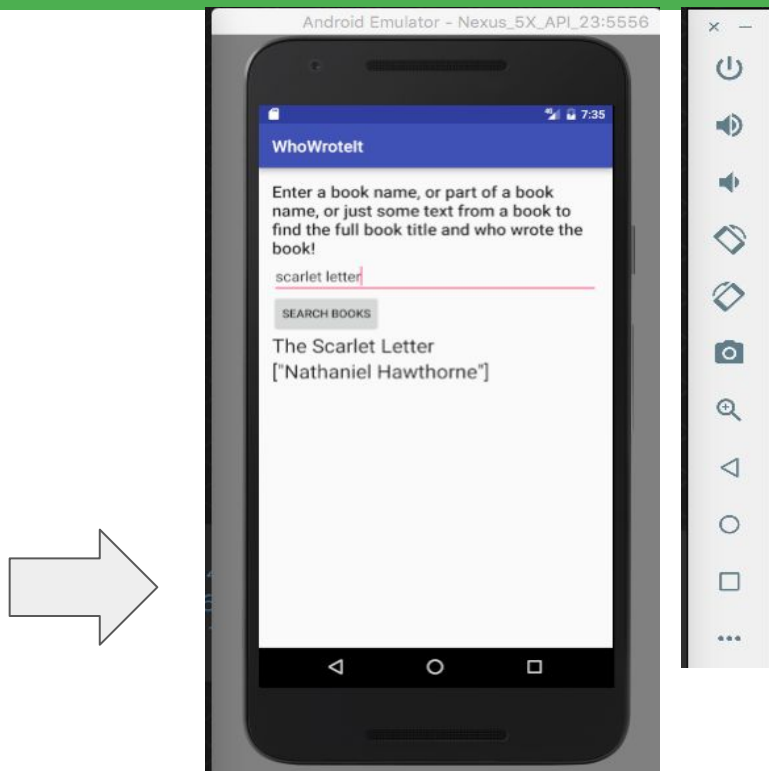


Configure virtual device

1. Choose hardware
2. Select Android version
3. Finalize



Run on a virtual device



Run on a physical device

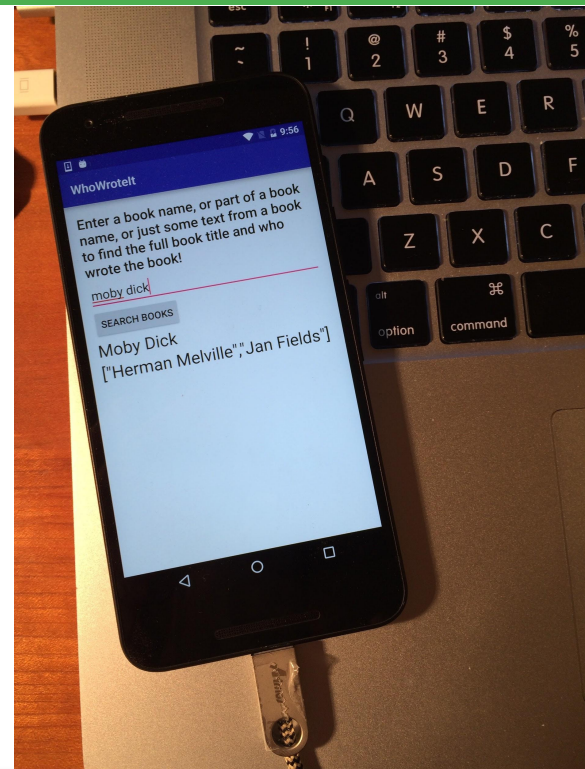
1. Turn on Developer Options:
 - a. **Settings > About phone**
 - b. Tap **Build number** seven times
2. Turn on USB Debugging
 - a. **Settings > Developer Options > USB Debugging**
3. Connect phone to computer with cable

Windows/Linux additional setup:

- [Using Hardware Devices](#)

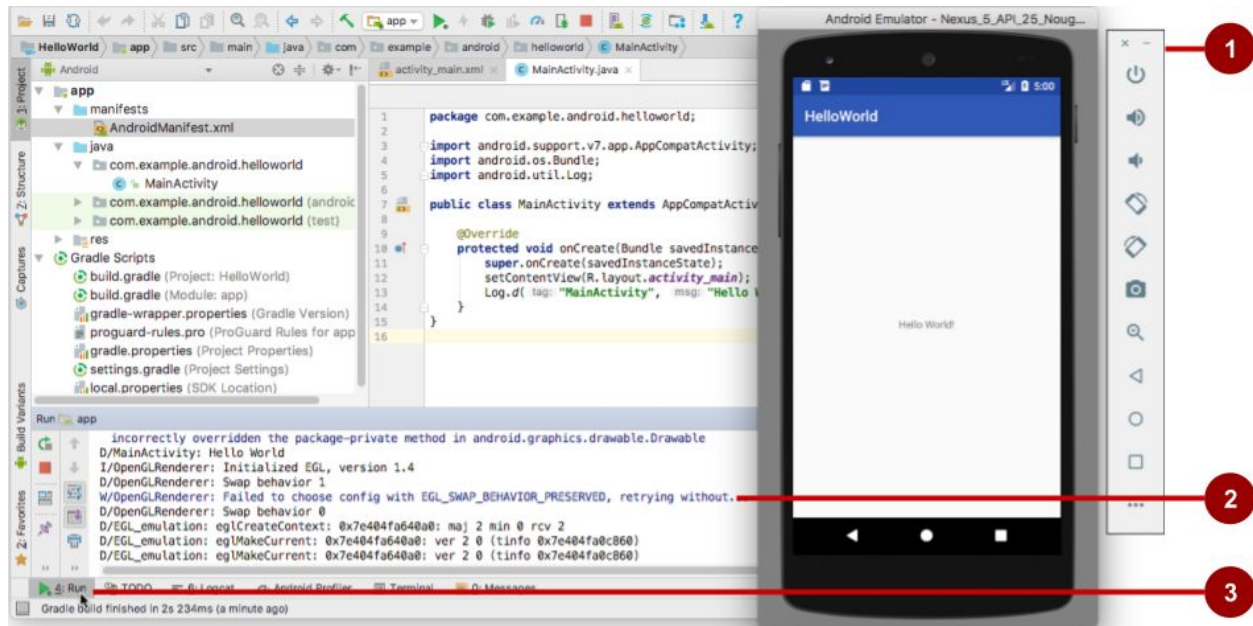
Windows drivers:

- [OEM USB Drivers](#)



Get feedback as your app runs

1. Emulator running the app
2. Run pane
3. **Run** tab to open or close the Run pane

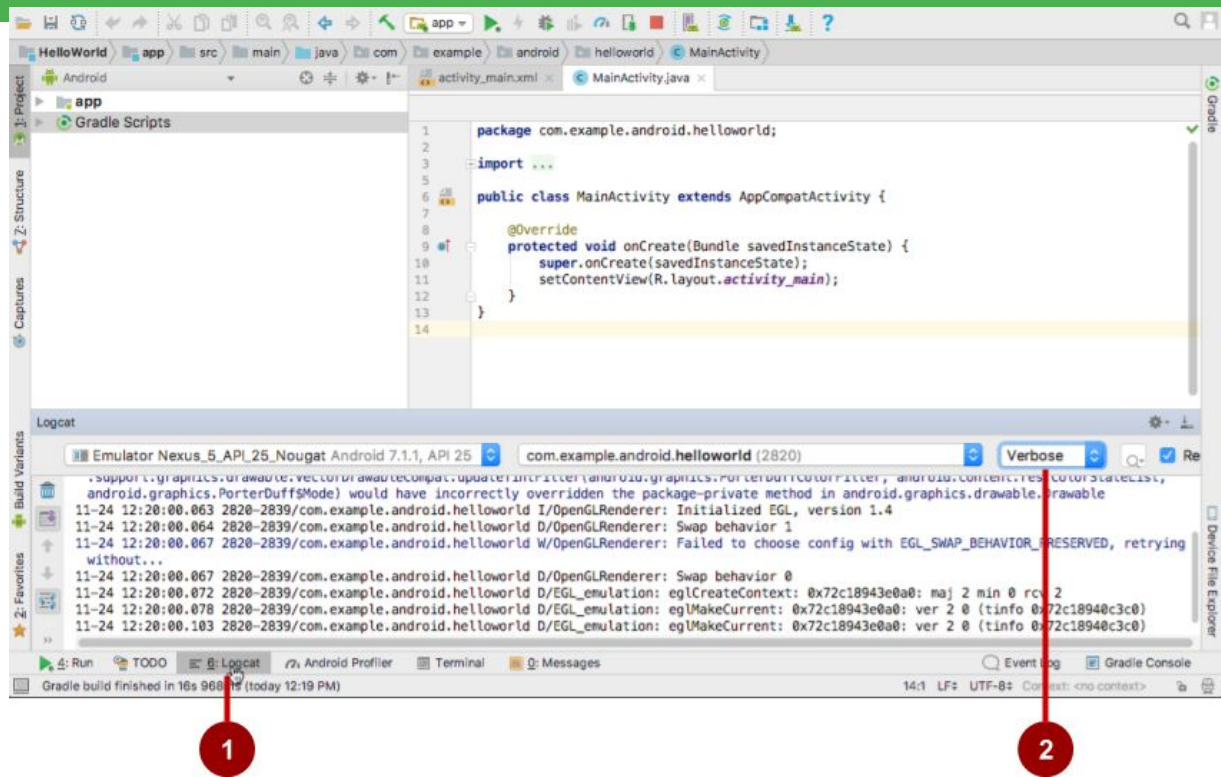


Adding logging to your app

- As the app runs, the **Logcat** pane shows information
- Add logging statements to your app that will show up in the Logcat pane
- Set filters in **Logcat** pane to see what's important to you
- Search using tags

The Logcat pane

1. **Logcat** tab to show Logcat pane
2. Log level menu



Logging statement

```
import android.util.Log;

// Use class name as tag
private static final String TAG =
    MainActivity.class.getSimpleName();

// Show message in Android Monitor, logcat pane
// Log.<log-level>(TAG, "Message");
Log.d(TAG, "Creating the URI...");
```

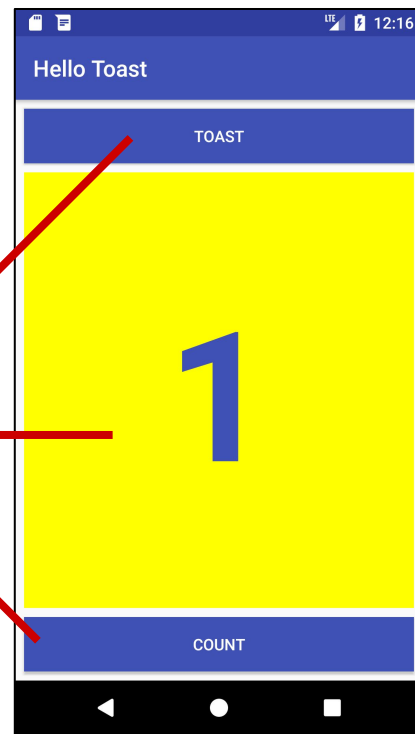
1.2 Layouts and resources for the UI

Views

Everything you see is a view

If you look at your mobile device, every user interface element that you see is a **View**.

Views



What is a view?

View subclasses are basic user interface building blocks

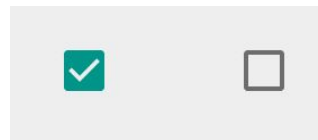
- Display text (TextView class), edit text (EditText class)
- Buttons (Button class), menus, other controls
- Scrollable (ScrollView, RecyclerView)
- Show images (ImageView)
- Group views (ConstraintLayout and LinearLayout)

Examples of view subclasses

Button



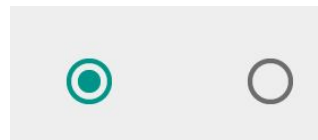
CheckBox



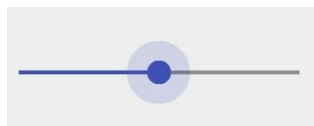
EditText



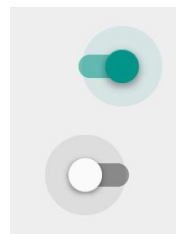
RadioButton



Slider



Switch



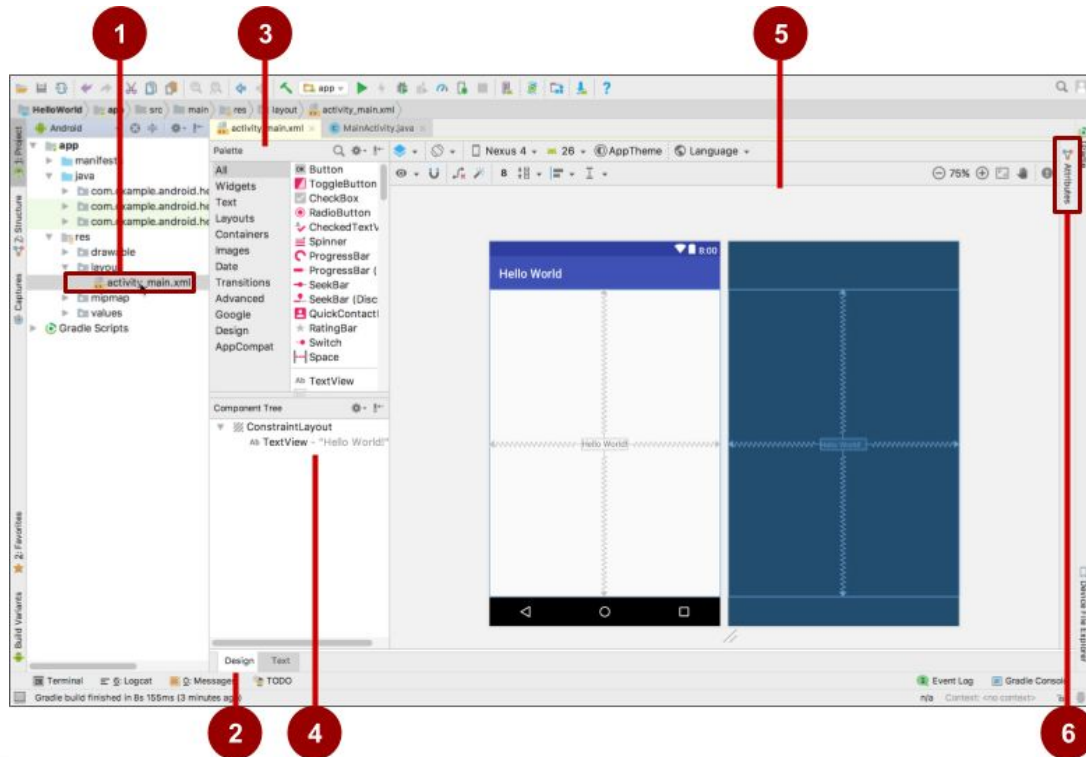
View attributes

- Color, dimensions, positioning
- May have focus (e.g., selected to receive user input)
- May be interactive (respond to user clicks)
- May be visible or not
- Relationships to other views

Create views and layouts

- Android Studio layout editor: visual representation of XML
- XML editor
- Java code

Android Studio layout editor



1. XML layout file
2. **Design and Text** tabs
3. **Palette** pane
4. **Component Tree**
5. Design and blueprint panes
6. **Attributes** tab

View defined in XML

<TextView

```
    android:id="@+id/show_count"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="@color/myBackgroundColor"  
    android:text="@string/count_initial_value"  
    android:textColor="@color/colorPrimary"  
    android:textSize="@dimen/count_text_size"  
    android:textStyle="bold"
```

```
/>
```

View attributes in XML

android:<property_name>=<property_value>

Example: android:layout_width="match_parent"

android:<property_name>="@<resource_type>/resource_id"

Example: android:text="@string/button_label_next"

android:<property_name>="@+id/view_id"

Example: android:id="@+id/show_count"

Create View in Java code

In an Activity:

context



```
TextView myText = new TextView(this);  
myText.setText("Display this text!");
```

What is the context?

- [Context](#) is an interface to global information about an application environment
- Get the context:

```
Context context = getApplicationContext();
```
- An Activity is its own context:

```
TextView myText = new TextView(this);
```

ViewGroup and View hierarchy

ViewGroup contains "child" views

- [ConstraintLayout](#): Positions UI elements using constraint connections to other elements and to the layout edges
- [ScrollView](#): Contains one element and enables scrolling
- [RecyclerView](#): Contains a list of elements and enables scrolling by adding and removing elements dynamically

ViewGroups for layouts

Layouts

- are specific types of ViewGroups (subclasses of [ViewGroup](#))
- contain child views
- can be in a row, column, grid, table, absolute

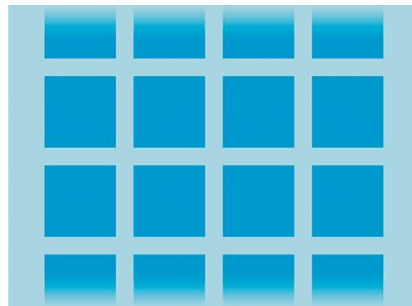
Common Layout Classes



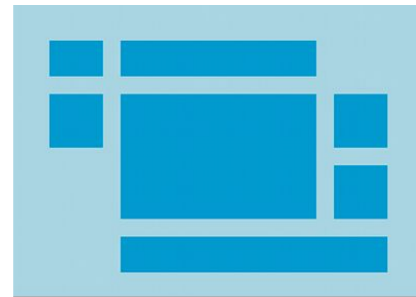
LinearLayout



ConstraintLayout



GridLayout

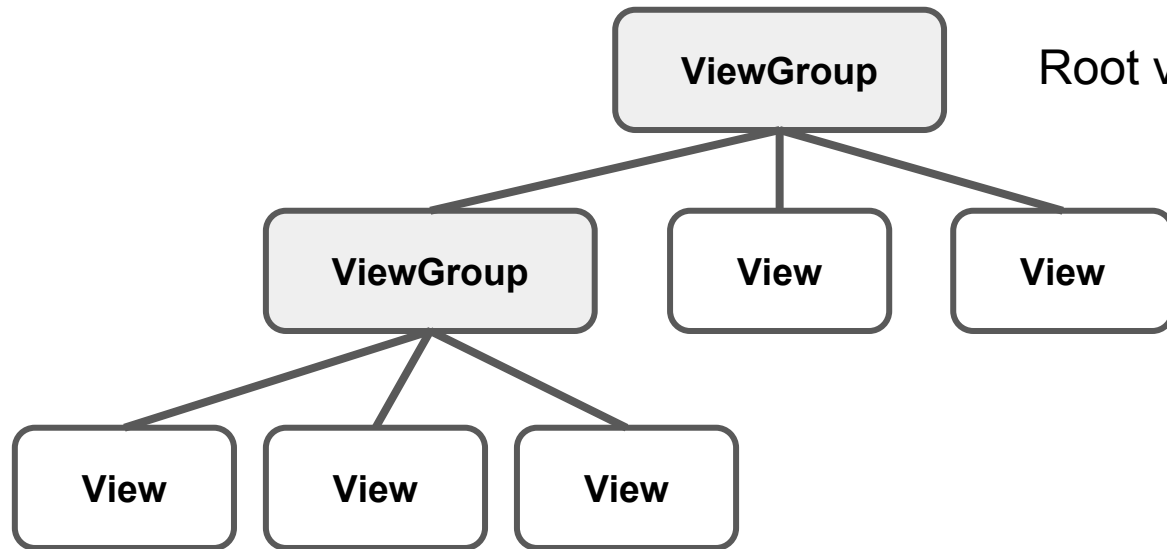


TableLayout

Common Layout Classes

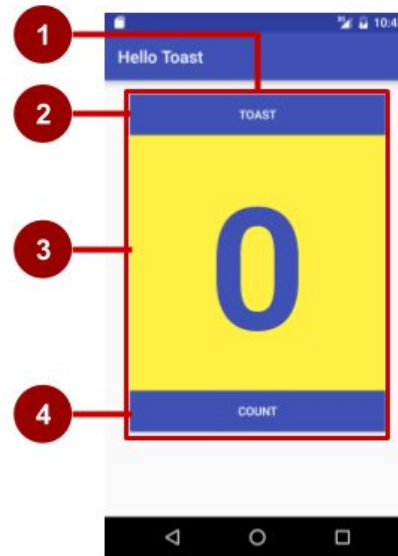
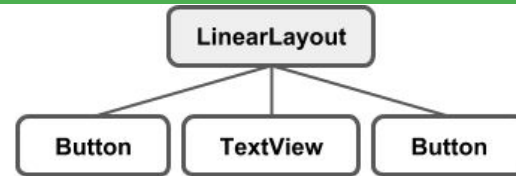
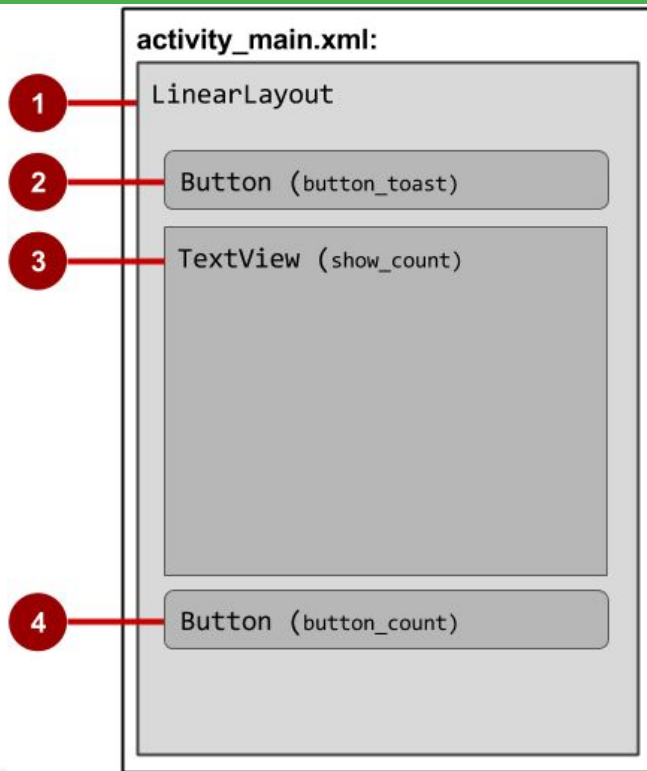
- `ConstraintLayout`: Connect views with constraints
- `LinearLayout`: Horizontal or vertical row
- `RelativeLayout`: Child views relative to each other
- `TableLayout`: Rows and columns
- `FrameLayout`: Shows one child of a stack of children

Hierarchy of viewgroups and views

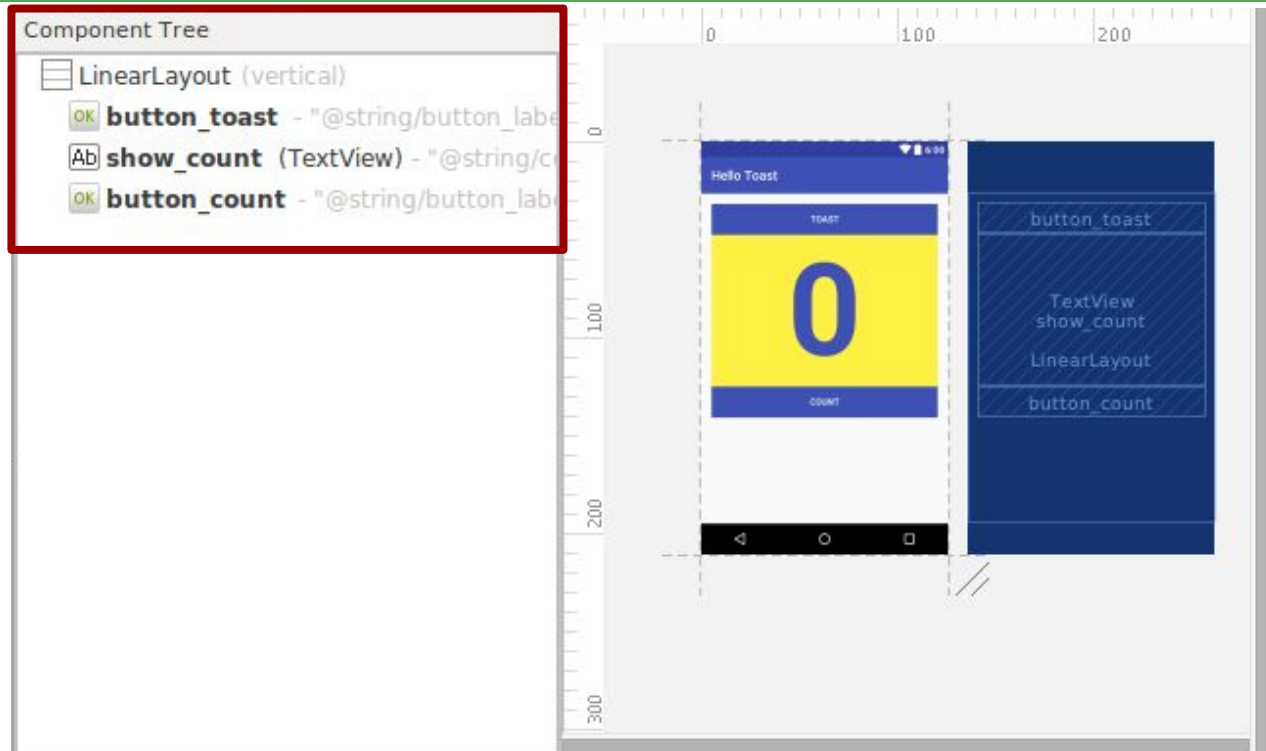


Root view is always a ViewGroup

View hierarchy and screen layout



View hierarchy in the layout editor



Layout created in XML

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        ... />
    <TextView
        ... />
    <Button
        ... />
</LinearLayout>
```

Layout created in Java Activity code

```
LinearLayout linearL = new LinearLayout(this);  
linearL.setOrientation(LinearLayout.VERTICAL);  
  
TextView myText = new TextView(this);  
myText.setText("Display this text!");  
  
linearL.addView(myText);  
setContentView(linearL);
```

Set width and height in Java code

Set the width and height of a view:

```
LinearLayout.LayoutParams layoutParams =  
    new LinearLayout.LayoutParams(  
        layoutParams.MATCH_PARENT,  
        layoutParams.MATCH_CONTENT);  
myView.setLayoutParams(layoutParams);
```

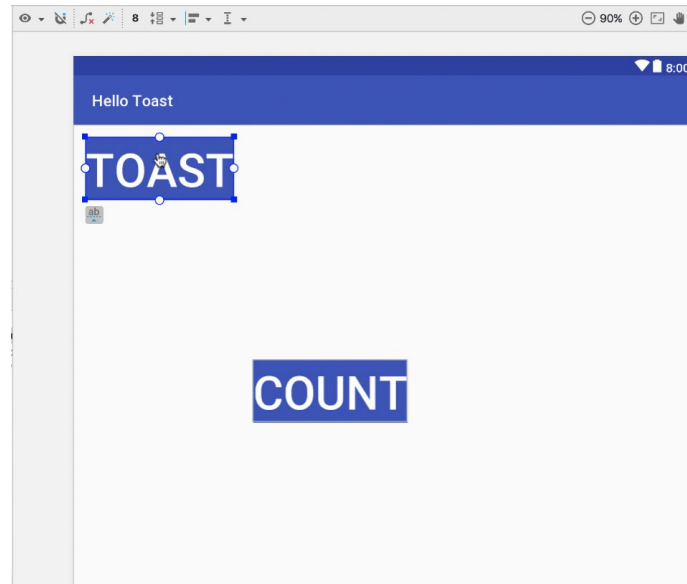
Best practices for view hierarchies

- Arrangement of view hierarchy affects app performance
- Use smallest number of simplest views possible
- Keep the hierarchy flat—limit nesting of views and view groups

The layout editor and Constraint Layout

The layout editor with ConstraintLayout

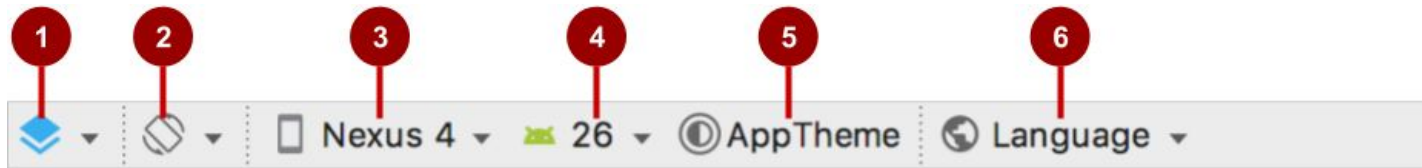
- Connect UI elements to parent layout
- Resize and position elements
- Align elements to others
- Adjust margins and dimensions
- Change attributes



What is ConstraintLayout?

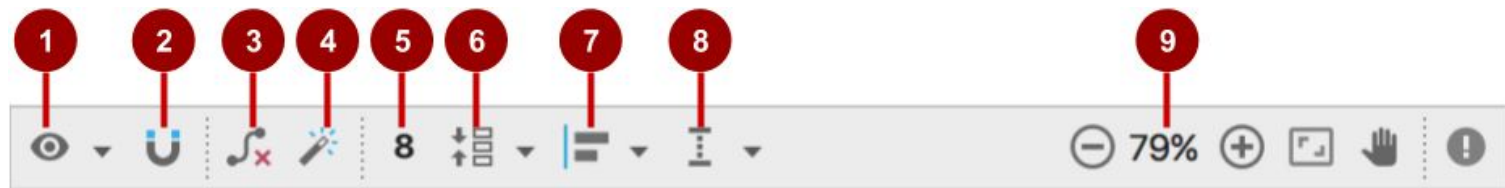
- Default layout for new Android Studio project
- ViewGroup that offers flexibility for layout design
- Provides constraints to determine positions and alignment of UI elements
- Constraint is a connection to another view, parent layout, or invisible guideline

Layout editor main toolbar




1. Select Design Surface: Design and Blueprint panes
2. Orientation in Editor: Portrait and Landscape
3. Device in Editor: Choose device for preview
4. API Version in Editor: Choose API for preview
5. Theme in Editor: Choose theme for preview
6. Locale in Editor: Choose language/locale for preview

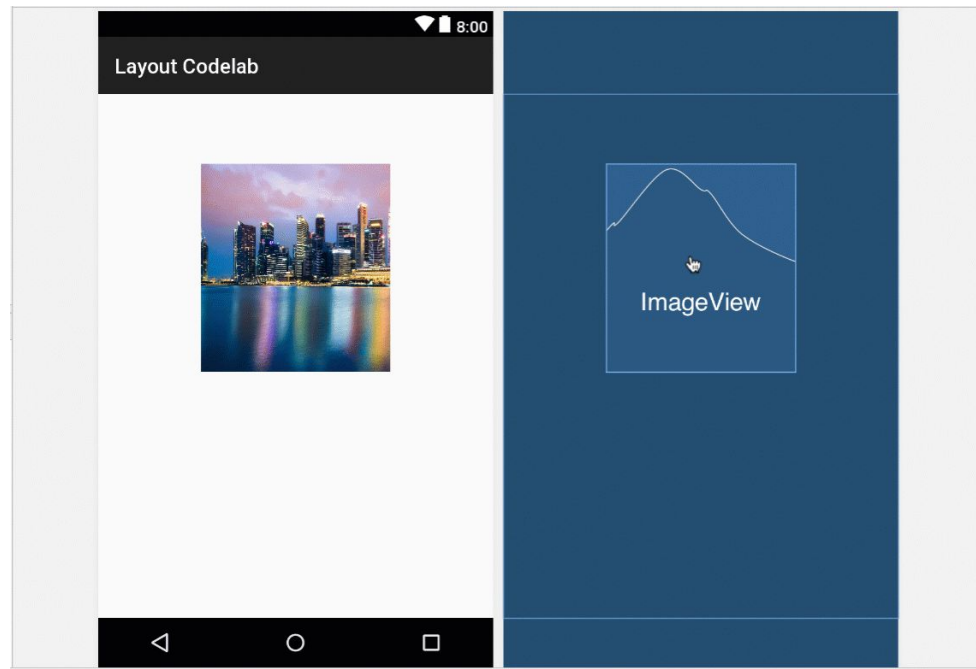
ConstraintLayout toolbar in layout editor



1. Show: Show Constraints and Show Margins
2. Autoconnect: Enable or disable
3. Clear All Constraints: Clear all constraints in layout
4. Infer Constraints: Create constraints by inference
5. Default Margins: Set default margins
6. Pack: Pack or expand selected elements
7. Align: Align selected elements
8. Guidelines: Add vertical or horizontal guidelines
9. Zoom controls: Zoom in or out

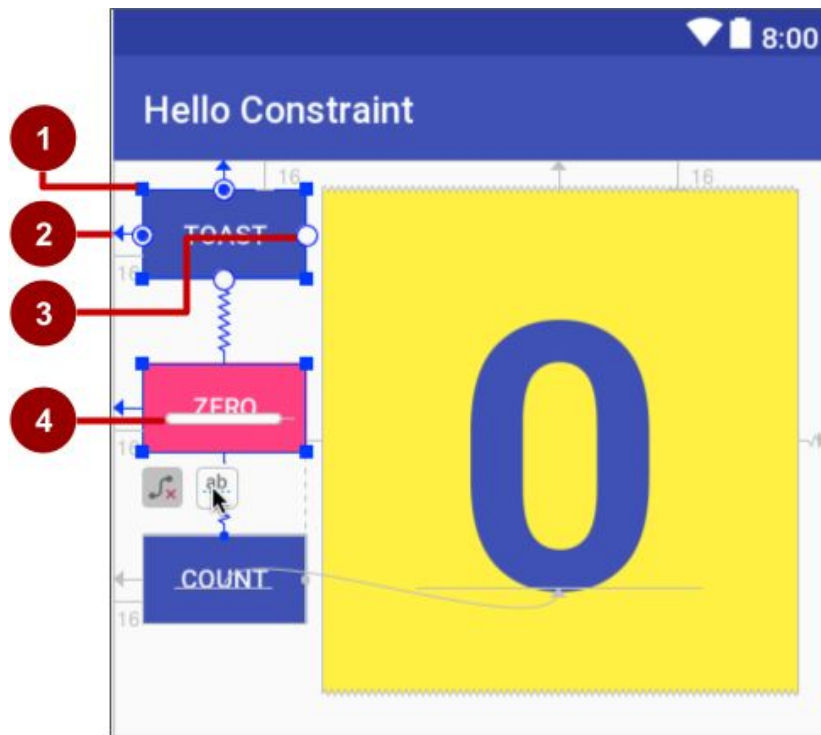
Autoconnect

- Enable Autoconnect  in toolbar if disabled
- Drag element to any part of a layout
- Autoconnect generates constraints against parent layout



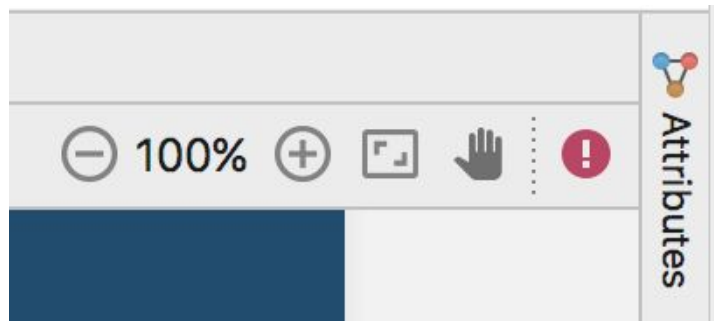
ConstraintLayout handles

1. Resizing handle
2. Constraint line and handle
3. Constraint handle
4. Baseline handle



Attributes pane

- Click the Attributes tab
- Attributes pane includes:
 - Margin controls for positioning
 - Attributes such as `layout_width`



Event Handling

Events

Something that happens

- In UI: Click, tap, drag
- Device: [DetectedActivity](#) such as walking, driving, tilting
- Events are "noticed" by the Android system

Event Handlers

Methods that do something in response to a click

- A method, called an **event handler**, is triggered by a specific event and does something in response to the event

Attach in XML and implement in Java

Attach handler to view in XML layout:

```
android:onClick="showToast"
```

Implement handler in Java activity:

```
public void showToast(View view) {  
    String msg = "Hello Toast!";  
    Toast toast = Toast.makeText(  
        this, msg, duration);  
    toast.show();  
}  
}
```

Alternative: Set click handler in Java

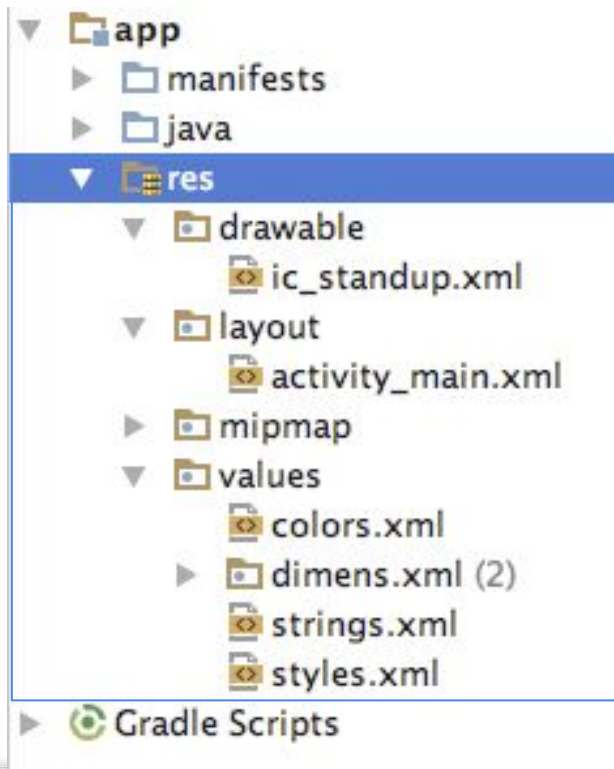
```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String msg = "Hello Toast!";
        Toast toast = Toast.makeText(this, msg, duration);
        toast.show();
    }
});
```

Resources and measurements

Resources

- Separate static data from code in your layouts.
- Strings, dimensions, images, menu text, colors, styles
- Useful for localization

Where are the resources in your project?



← resources and resource files stored in **res** folder

Refer to resources in code

- **Layout:**

```
R.layout.activity_main
```

```
setContentView(R.layout.activity_main);
```

- **View:**

```
R.id.recyclerview
```

```
rv = (RecyclerView) findViewById(R.id.recyclerview);
```

- **String:**

```
In Java: R.string.title
```

```
In XML: android:text="@string/title"
```

Measurements

- Density-independent Pixels (dp): for Views
- Scale-independent Pixels (sp): for text

Don't use device-dependent or density-dependent units:

- ~~● Actual Pixels (px)~~
- ~~● Actual Measurement (in, mm)~~
- ~~● Points - typography 1/72 inch (pt)~~