Android Developer Fundamentals

Hello World

Lesson 1

1.1 Create Your First Android App

Contents

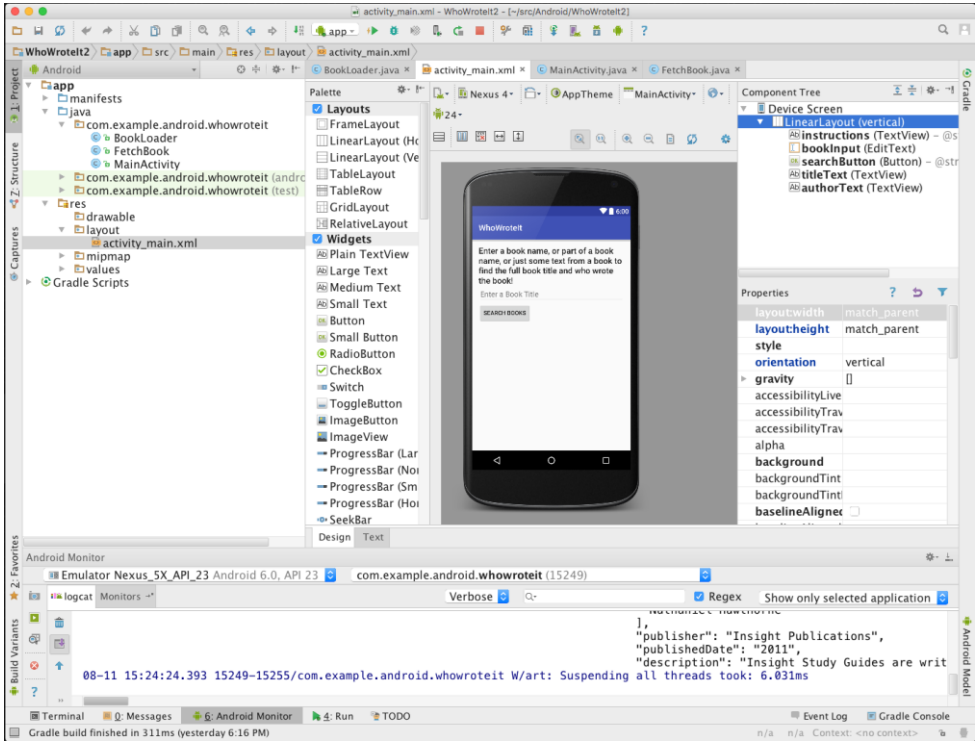
- Android Studio
- Creating "Hello World" app in Android Studio
- Basic app development workflow with Android Studio
- Running apps on virtual and physical devices

Prerequisites

- Java Programming Language
- Object-oriented programming
- XML - properties / attributes
- Using an IDE for development and debugging

Android Studio

What is Android Studio?



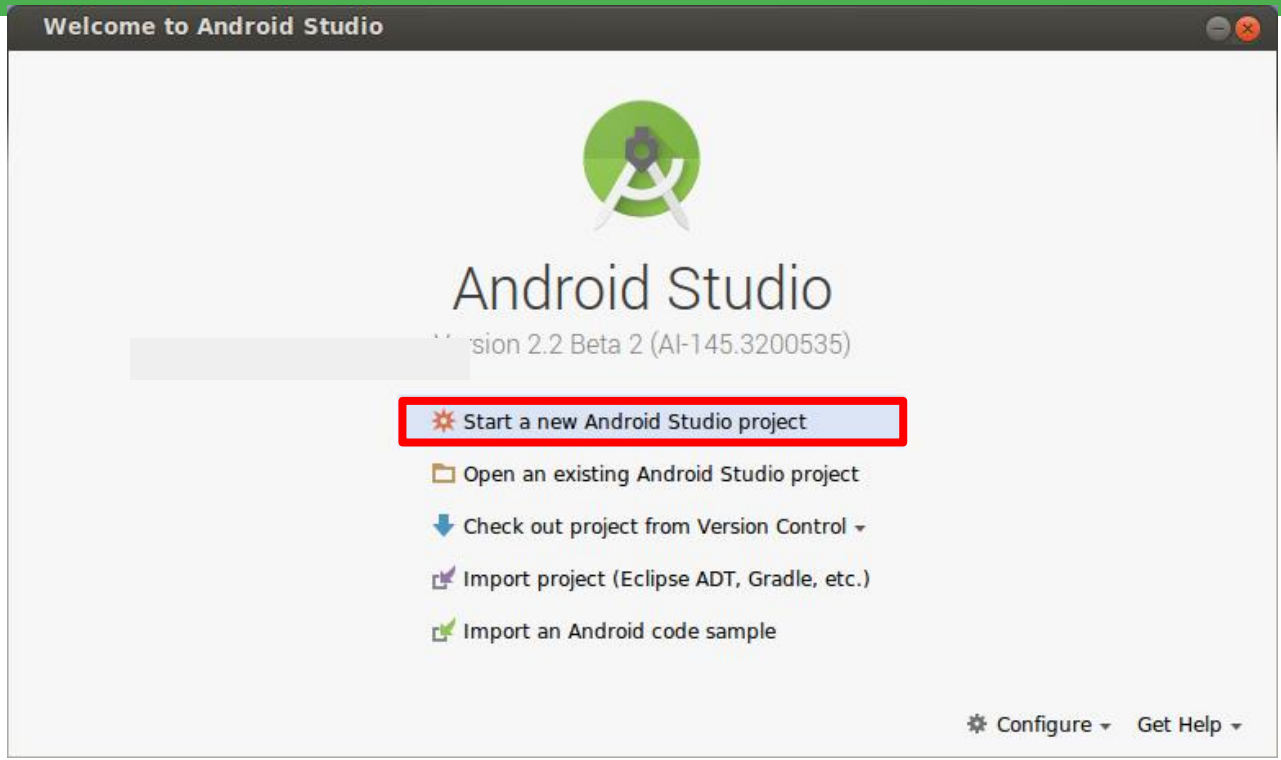
- Android IDE
- Project structure
- Templates
- Layout Editor
- Testing tools
- Gradle-based build
- Log Console
- Debugger
- Monitors
- Emulators

Installation Overview

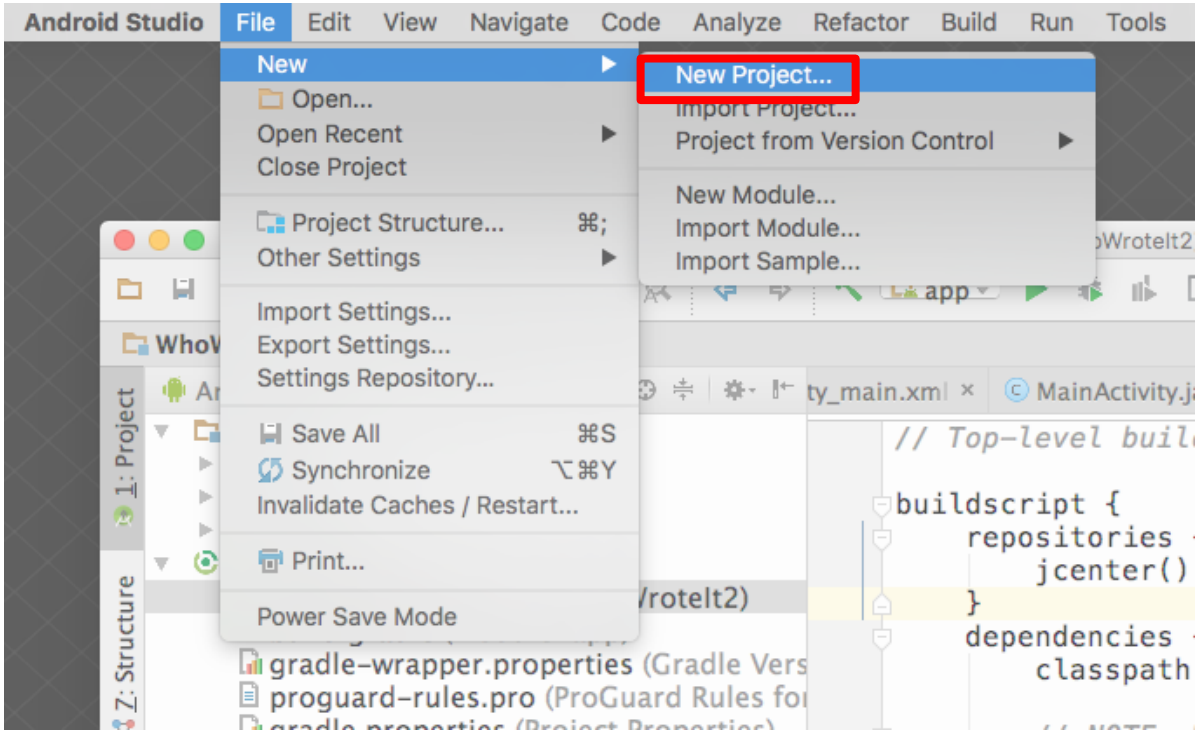
- Mac, Windows, or Linux
- Requires Java Development Kit (JDK) 1.7 or better from Oracle Java SE downloads page
- Set JAVA_HOME to JDK installation location
- Download and install Android Studio from http://developer.android.com/sdk/index.html
- See 1.1 P Install Android Studio for details

Creating Your First Android App

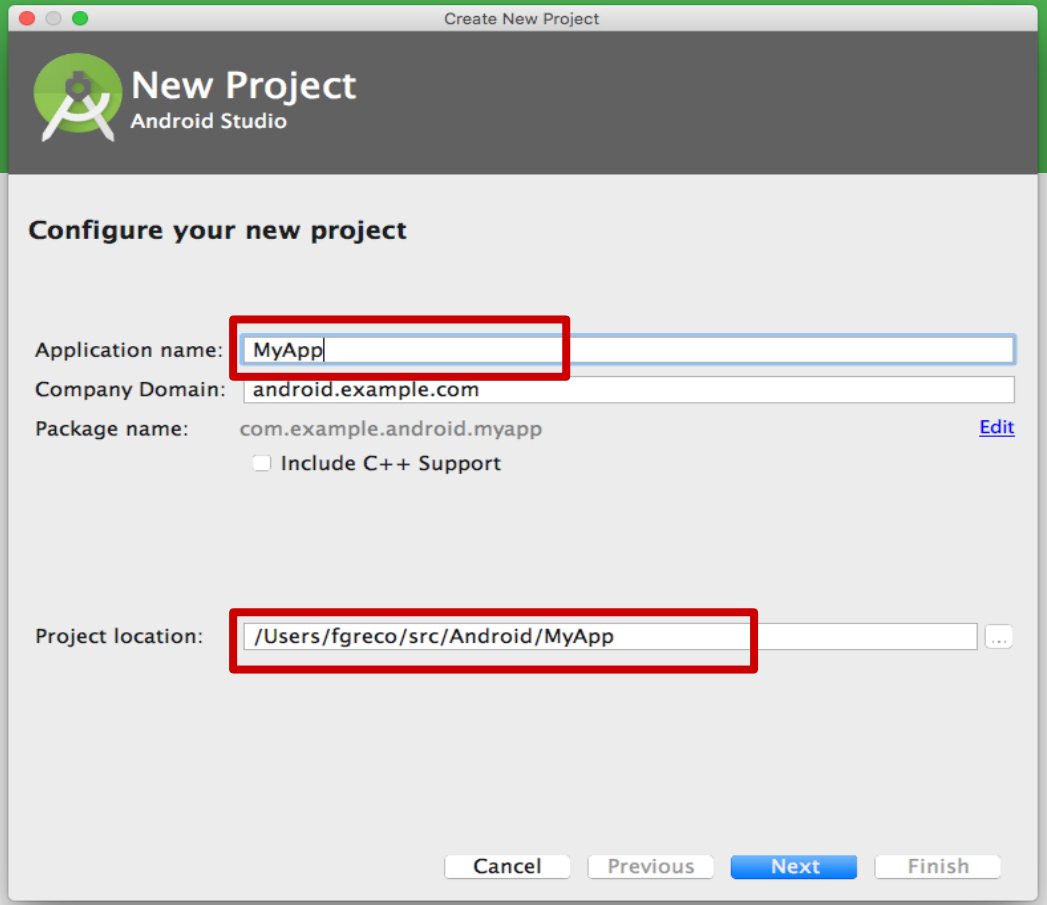
Start Android Studio



Create a project inside Android Studio



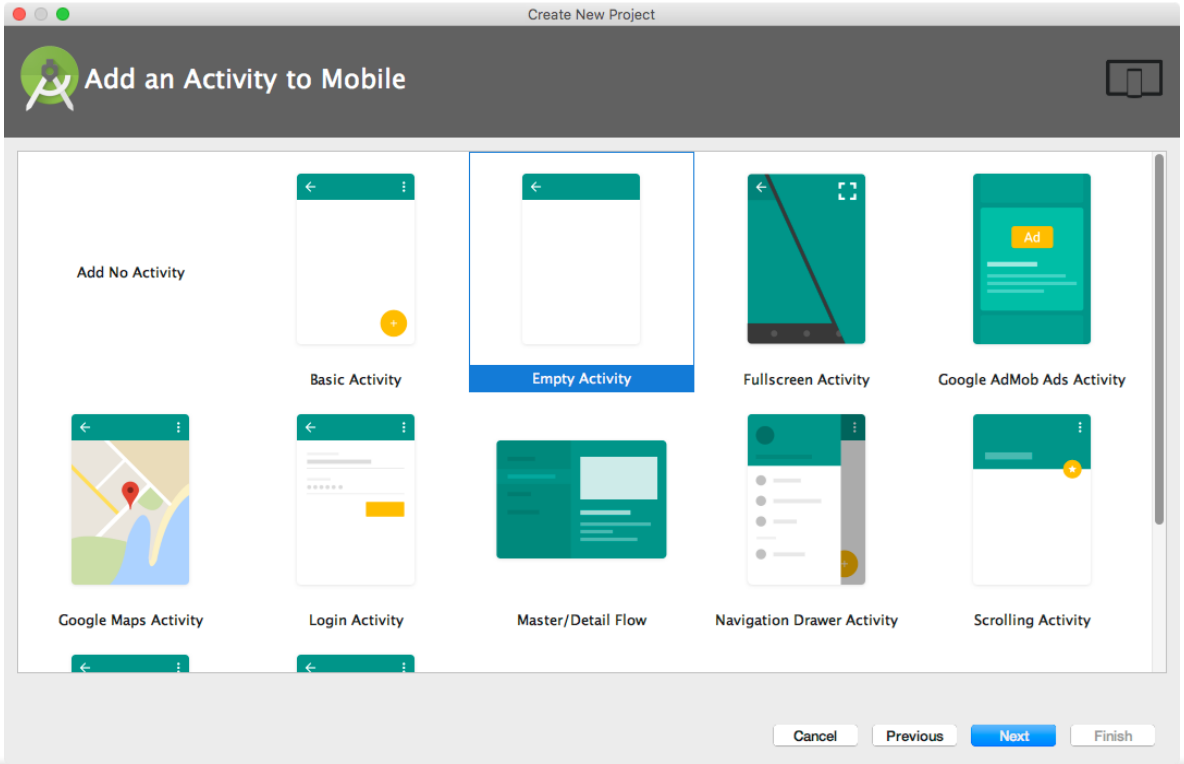
Name your app



Pick activity template

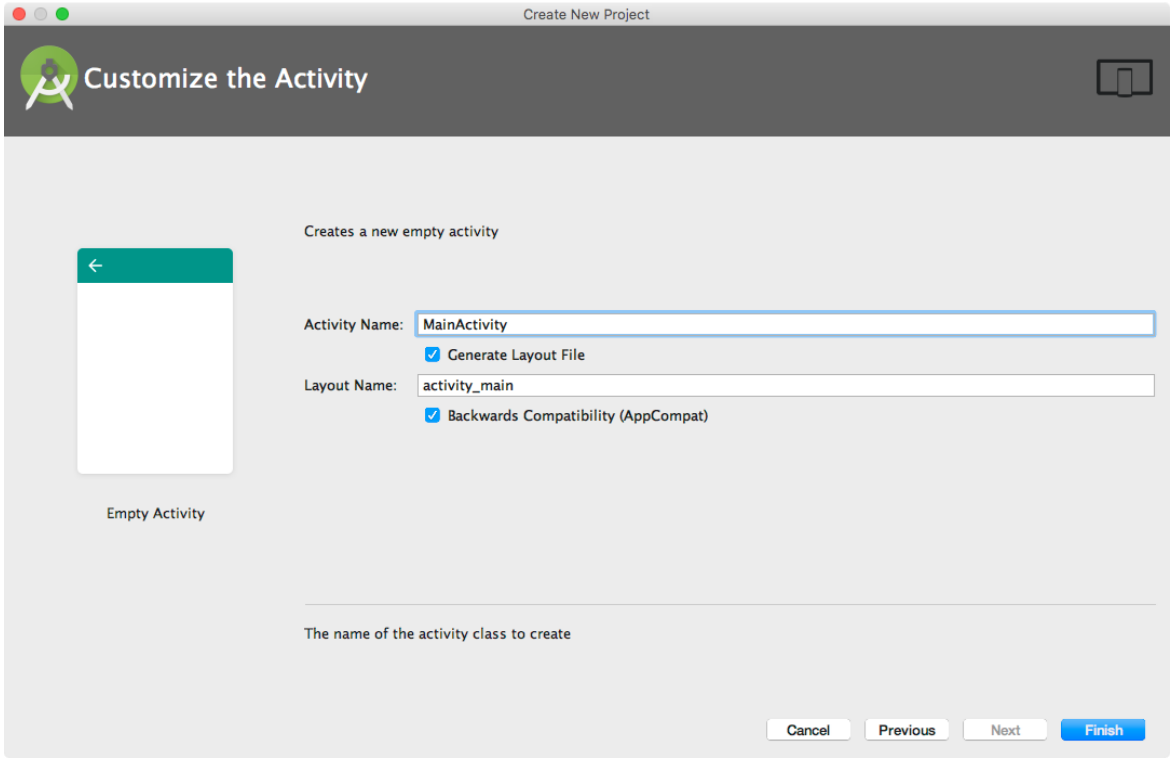
Choose templates for common activities, such as maps or navigation drawers.

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

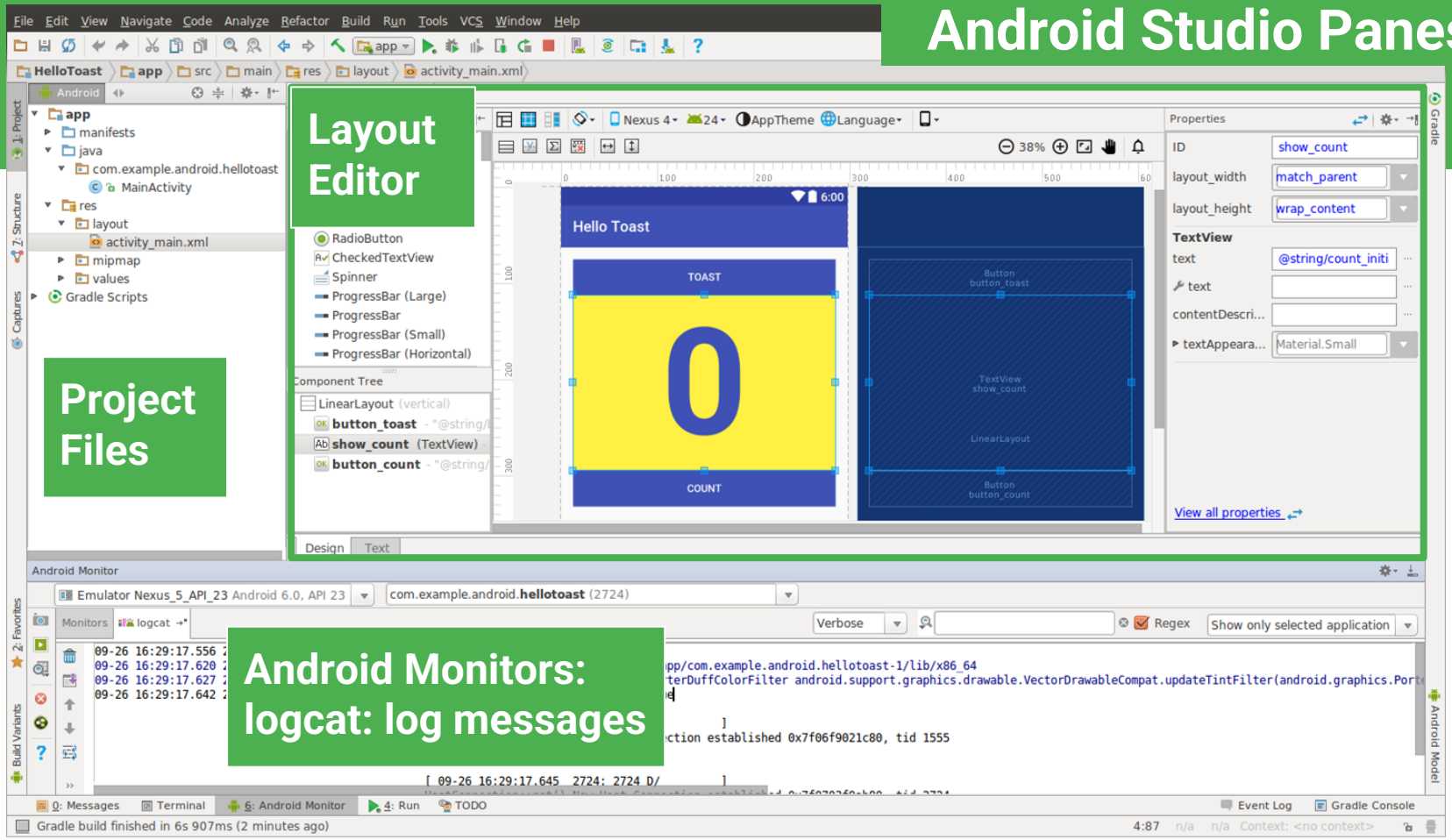


Name your activity

- Good practice to name main activity MainActivity and activity_main layout
- Use AppCompat
- Generating layout file is convenient

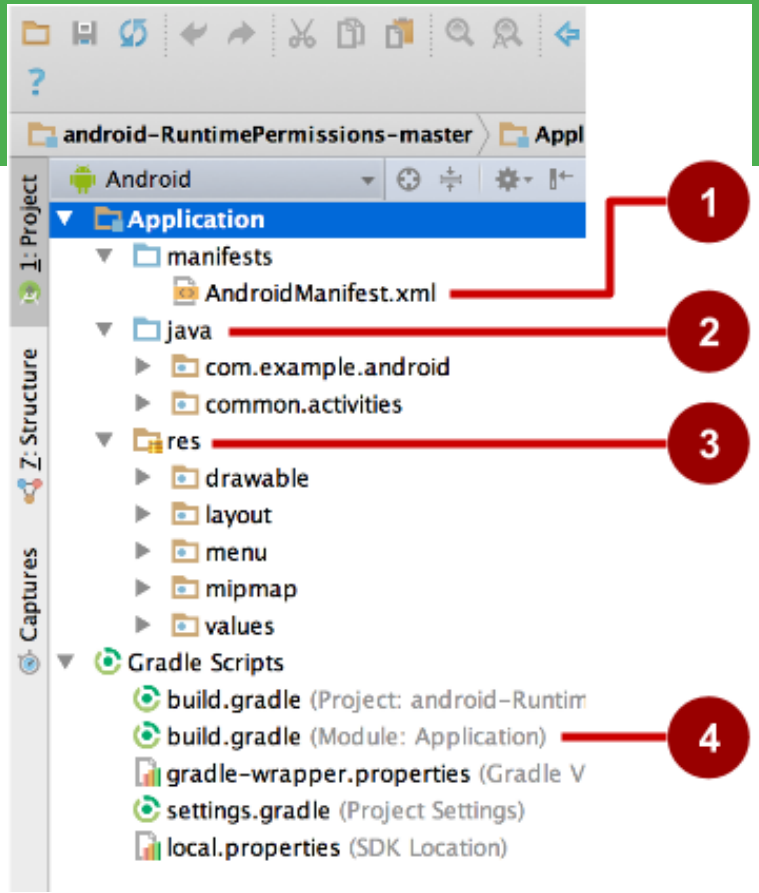


Android Studio Panes



Project folders

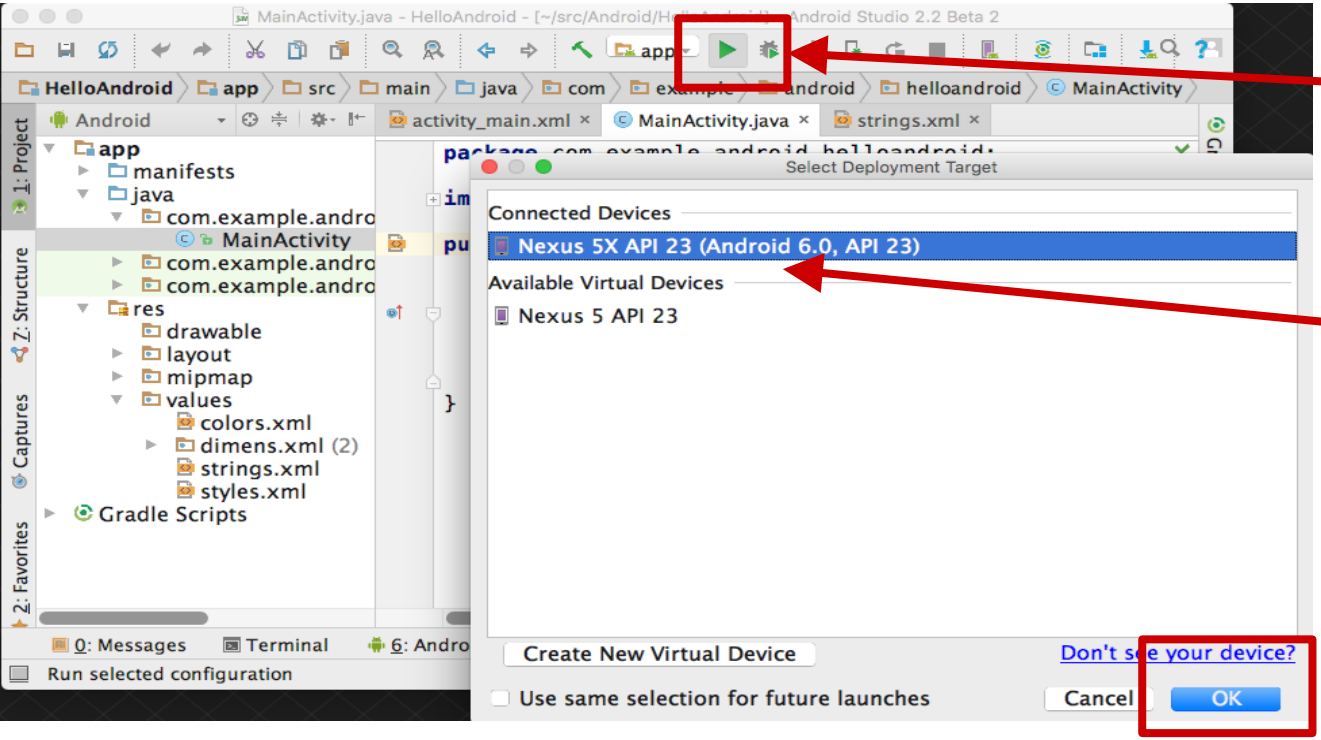
1. **manifests**—Android Manifest file - description of app read by the Android runtime
2. **java**—Java source code packages
3. **res**—Resources (XML) - layout, strings, images, dimensions, colors...
4. **build.gradle**—Gradle build files



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

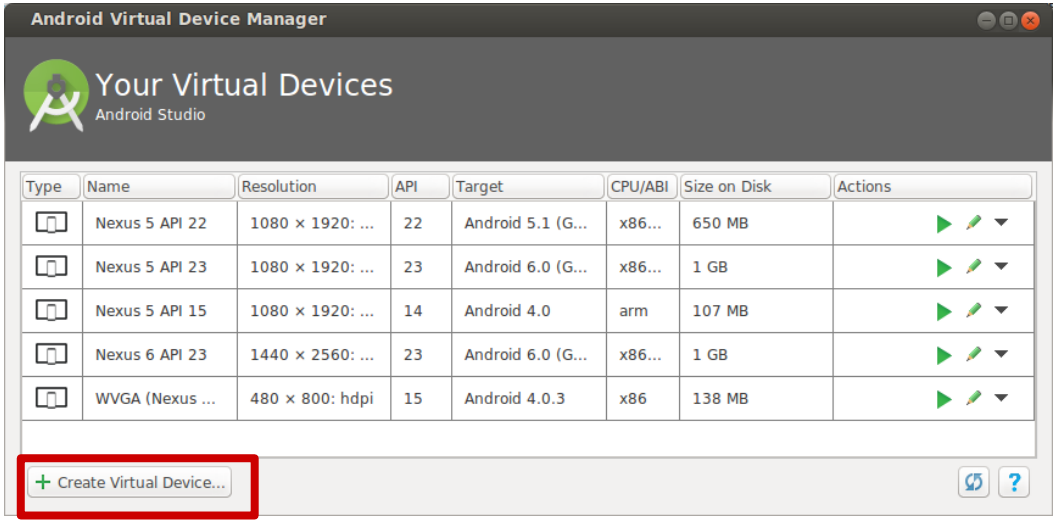
2. Select virtual or physical device

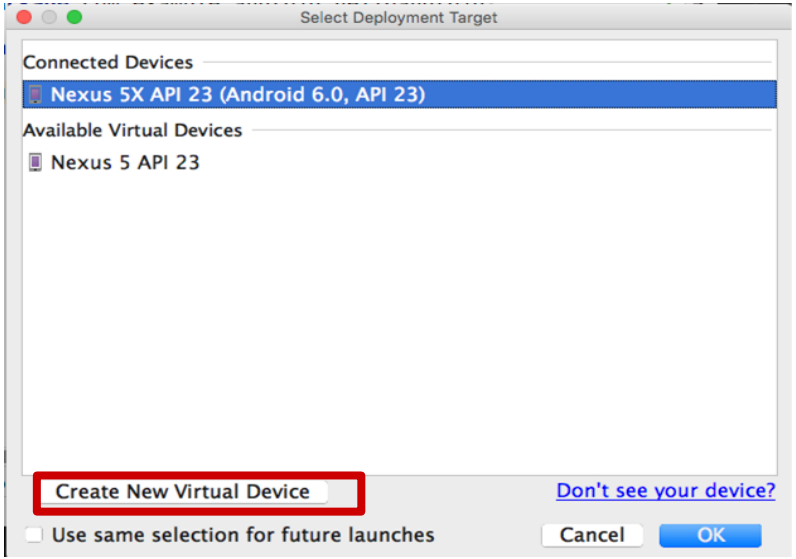
3. OK

Create a virtual device

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

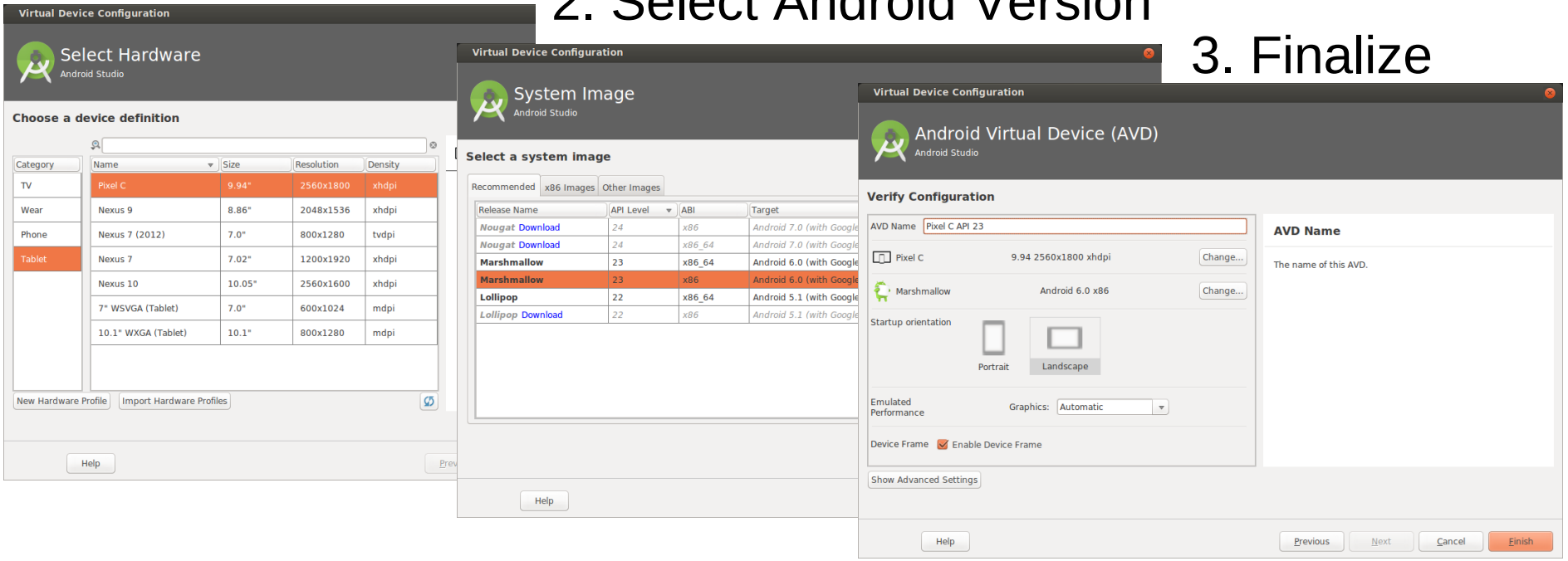
Tools > Android > AVD Manager or:



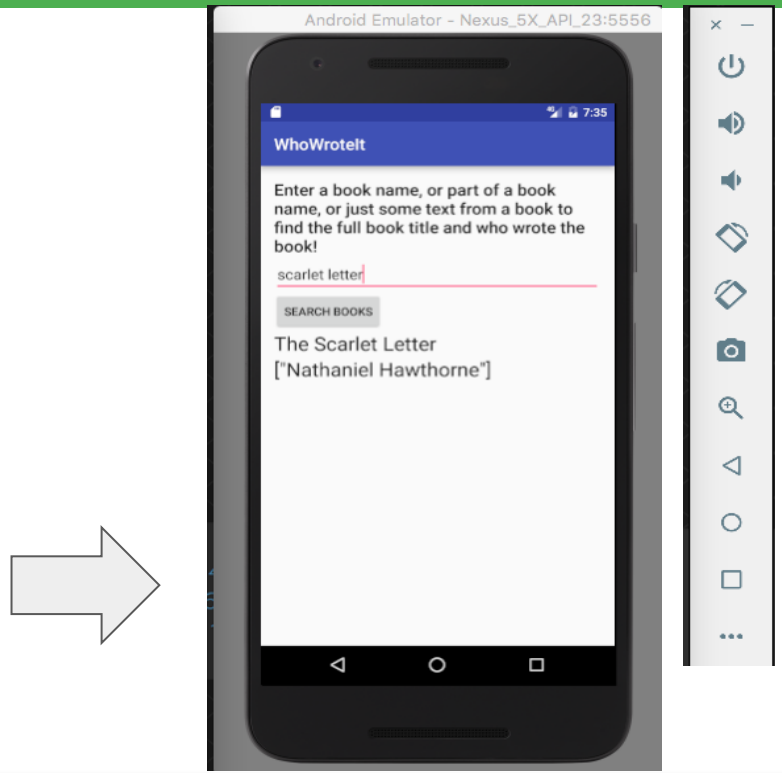


Configure virtual device

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



Run on a virtual device



Run on a physical device

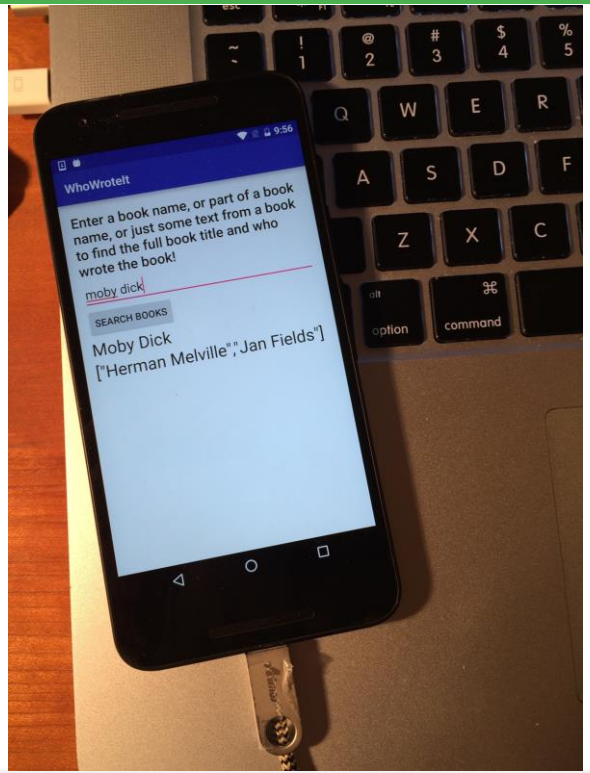
1. Turn on Developer Options:
   a. **Settings > About phone**
   b. Tap **Build number** seven times
2. Turn on USB Debugging
   a. **Settings > Developer Options > USB Debugging**
3. Connect phone to computer with cable

Windows/Linux additional setup:

- Using Hardware Devices

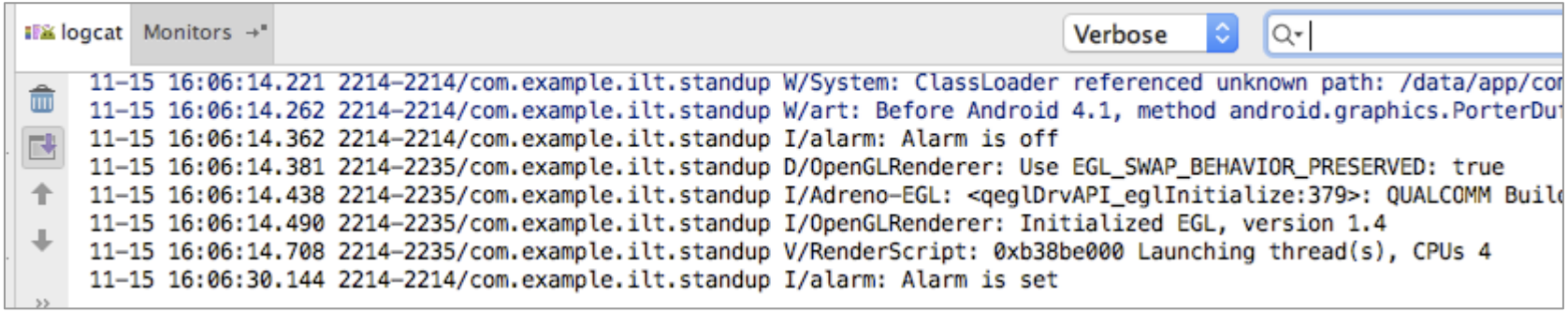
Windows drivers:

- OEM USB Drivers



Get feedback as your app runs

- As the app runs, Android Monitor logcat shows information
- You can add logging statements to your app that will show up in logcat.



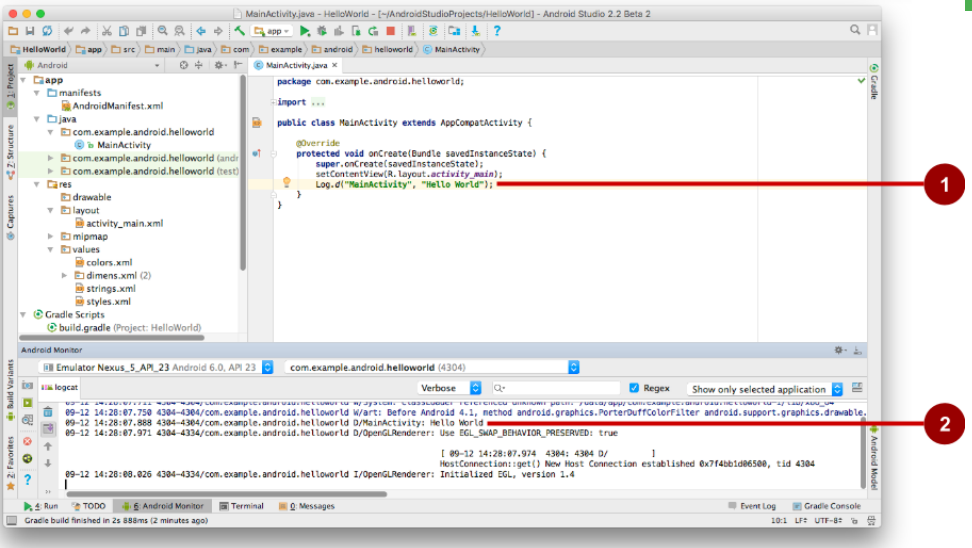
Logging

```
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…”);
```

Android Monitor > logcat pane



1. Log statements in code.
2. logcat pane shows system and logging messages

- Set filters to see what's important to you
- Search using tags

Learn more

- Meet Android Studio
- Official Android documentation at developer.android.com
- Create and Manage Virtual Devices
- Supporting Different Platform Versions
- Supporting Multiple Screens

Learn even more

- Gradle Wikipedia page
- Google Java Programming Language style guide
- Find answers at Stackoverflow.com

What's Next?

- Concept Chapter: 1.1 C Create Your First Android App
- Practical: 1.1 P Install Android Studio and Run Hello World

END

Android Developer Fundamentals

Hello World

Lesson 1

1.2 Views, Layouts, and Resources

Contents

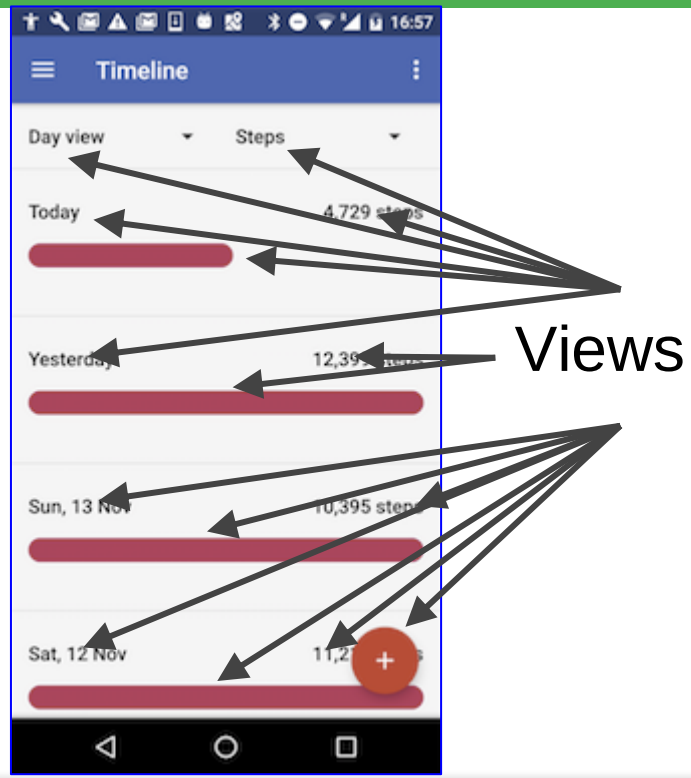
- Views, view groups, and view hierarchy
- Layouts in XML and Java code
- Event Handling
- Resources
- Screen Measurements

Views

An Android application is incomplete without the proper appearance of the screen

Everything you see is a view

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



What is a view

Views are building blocks of user interface.

- display text (TextView class), edit text (EditText class)
- buttons (Button class), menus, other controls
- scrollable (ScrollView, RecyclerView)
- show images (ImageView)
- subclass of View class

Views have properties

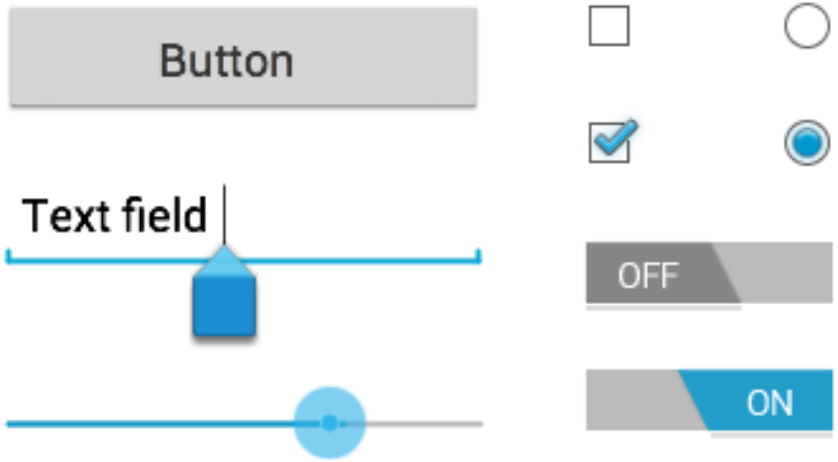
- Have properties (e.g., color, dimensions, positioning)
- May have focus (e.g., selected to receive user input)
- May be interactive (respond to user clicks)
- May be visible or not
- Have relationships to other views

Examples of views

Button

EditText

SeekBar



CheckBox

RadioButton

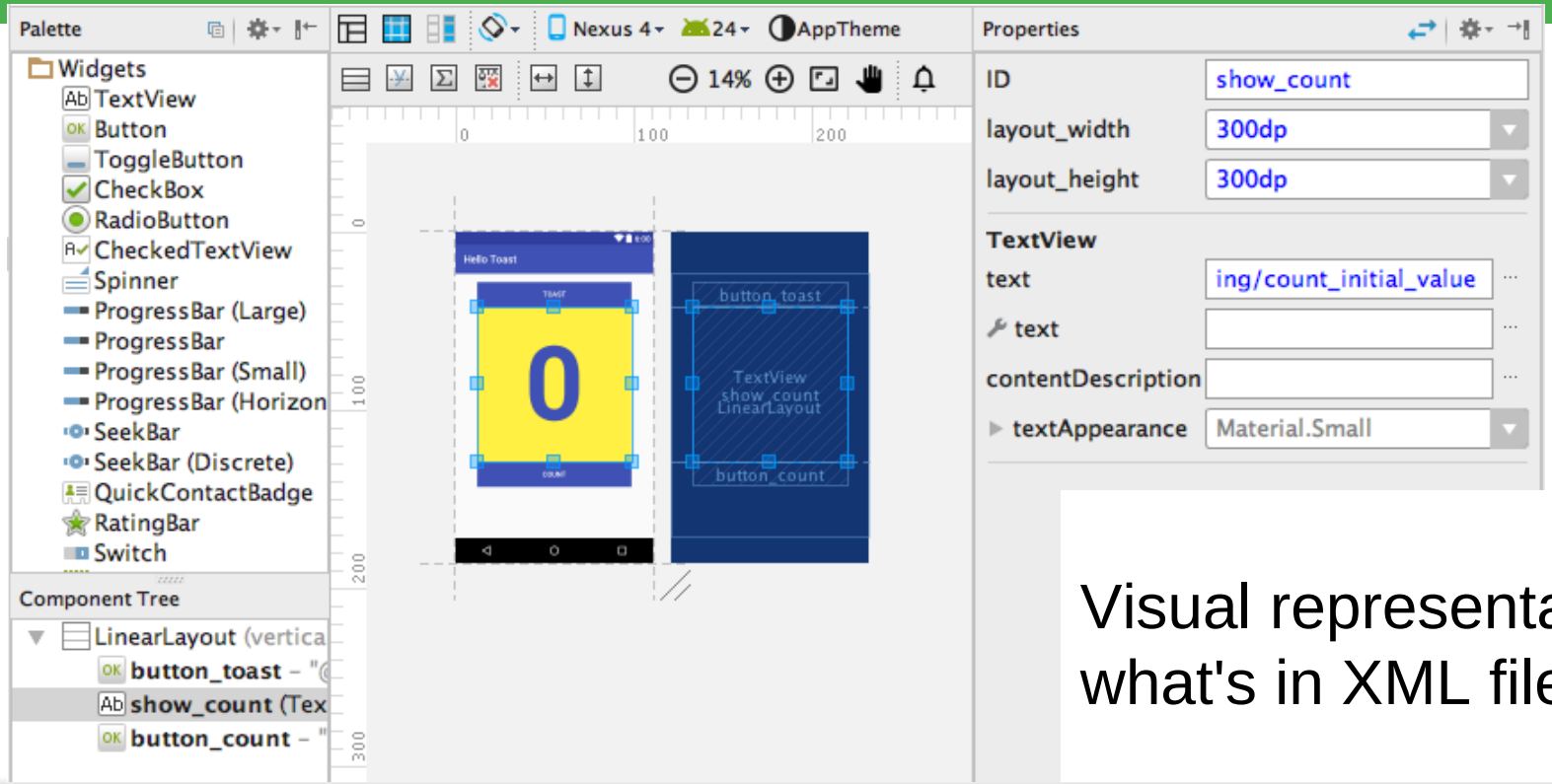
Switch

Creating and laying out views

3 ways to define views

- Graphically within Android Studio
- XML Files
- Programmatically

Views defined in Layout Editor

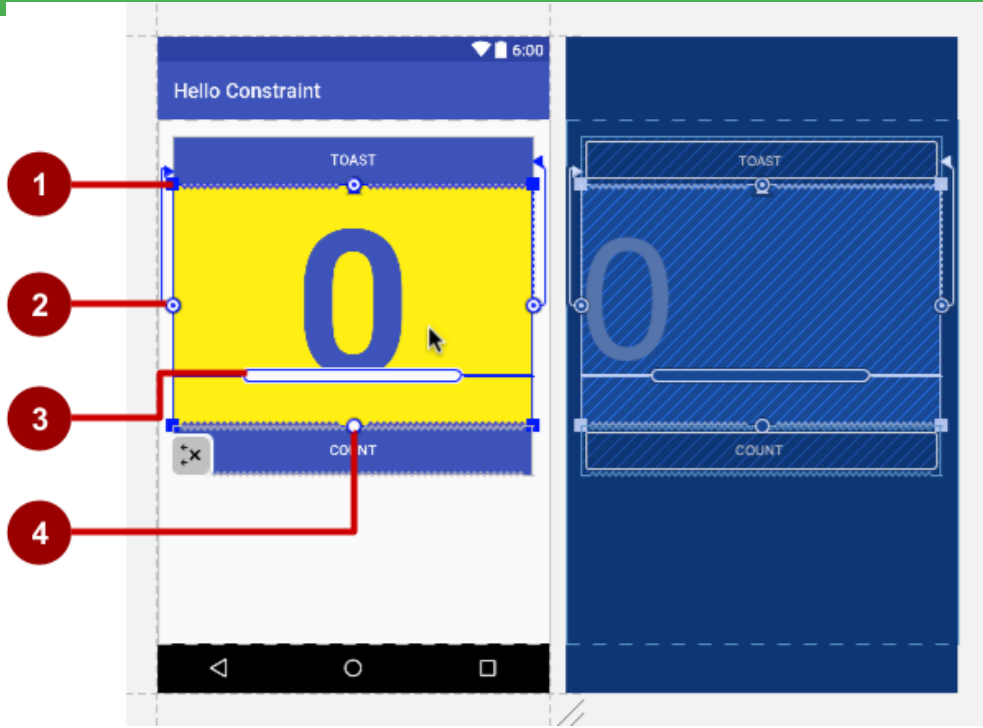


Visual representation of what's in XML file.

Using the Layout Editor

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

A constraint represents a connection or alignment to another view



Views 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"
/>
```

Using "match_parent" for the width means the UI element will take up all the available width within its parent container.

For example, if you have a LinearLayout that contains a button with android:layout_width="match_parent", the button will stretch horizontally to fill the entire width of the LinearLayout.

"wrap_content": This value indicates that the UI element's height should be as tall as necessary to accommodate its content. In other words, the UI element will adjust its height to fit the content it holds.

For example, if you have a TextView with a long text, setting its height to "wrap_content" will ensure that the TextView is tall enough to display the entire text without cutting it off.

View properties in XML

android:<property_name>="<property_value>"

Example: android:layout_width="match_parent“

match_parent will take the width from the parent tag

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);
```

Custom views

- Over 100 (!) different types of views available from the Android system, all children of the View class
- If necessary, create custom views by subclassing existing views or the View class

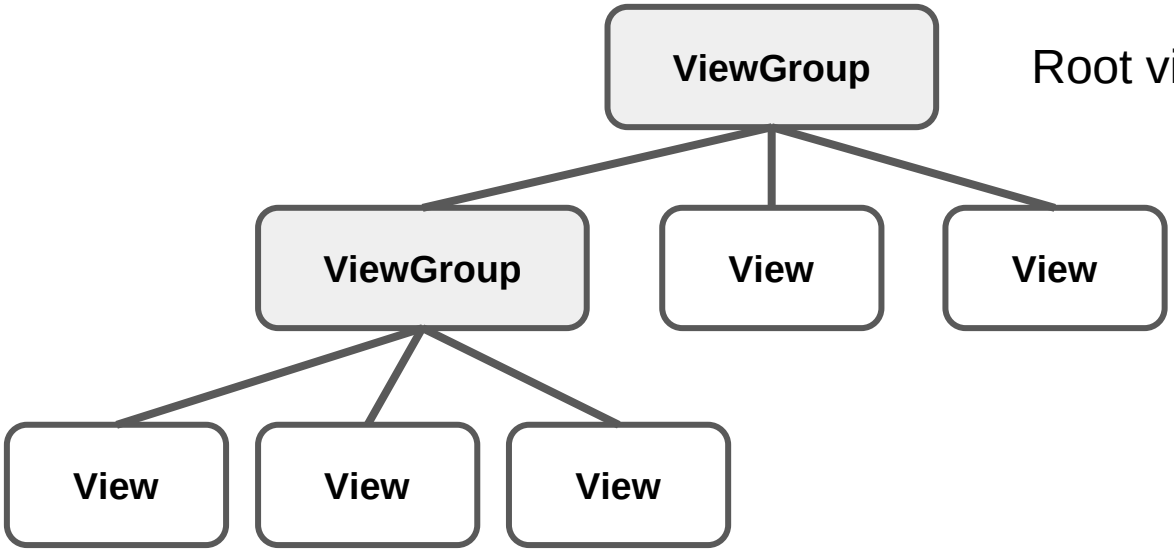
ViewGroup & View Hierarchy

ViewGroup views

A ViewGroup (parent) is a type of view that can contain other views (children). It acts as the container to contain the views. ViewGroup is the base class for layouts and view components

- ScrollView—scrollable view that contains one child view
- LinearLayout—arrange views in horizontal/vertical row
- RecyclerView—scrollable "list" of views or view groups

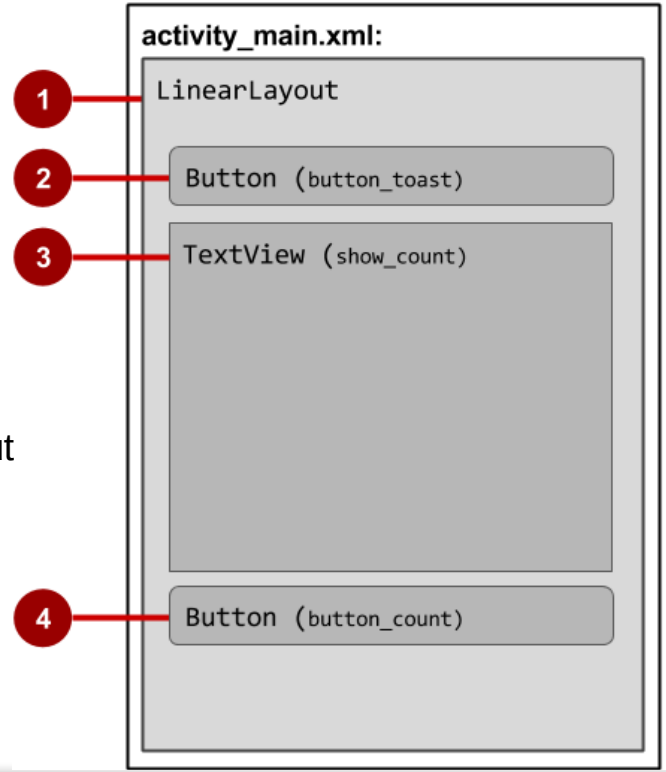
Hierarchy of view groups and views

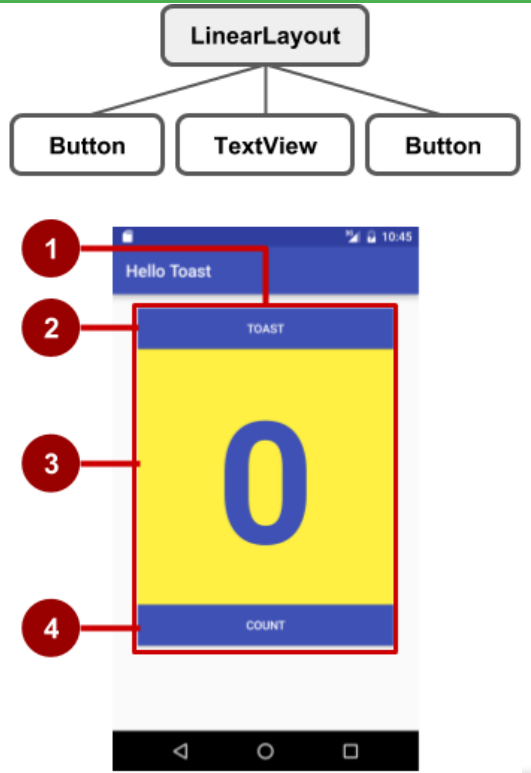


Root view is always a view group

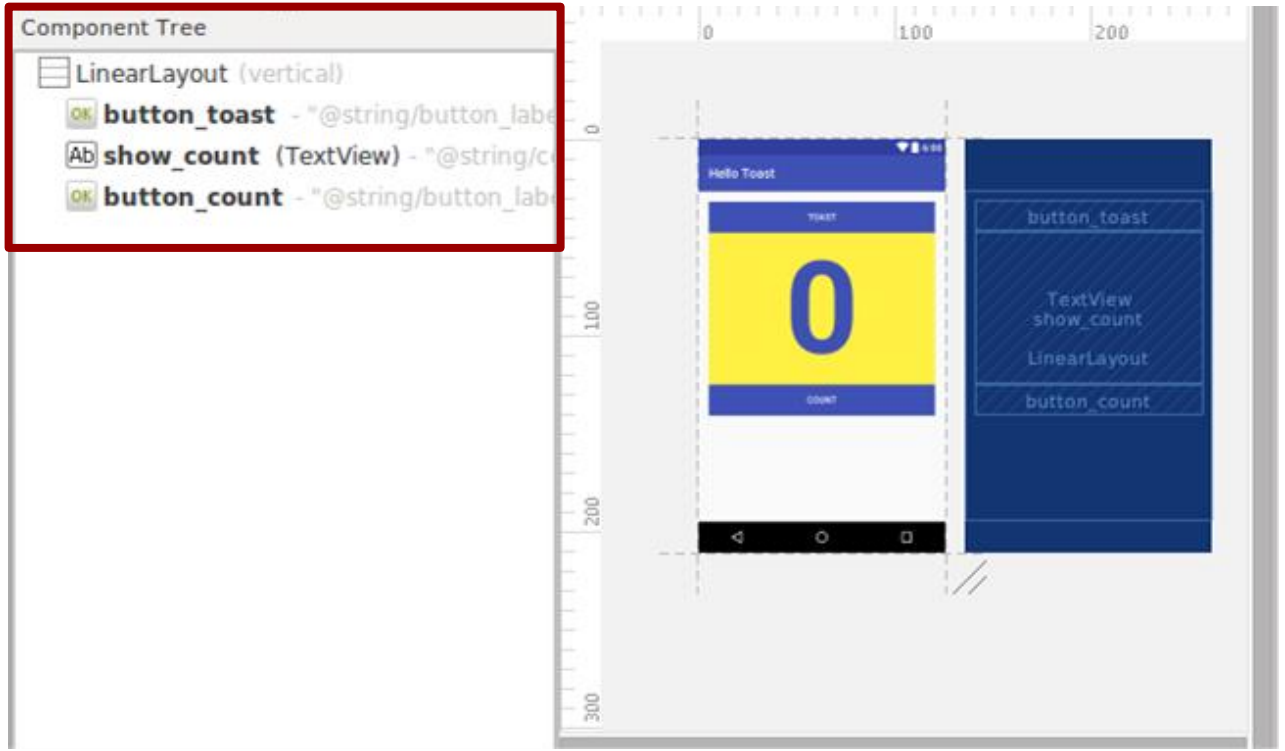
View hierarchy and screen layout

LinearLayout is the ViewGroup





View hierarchy in the component tree



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

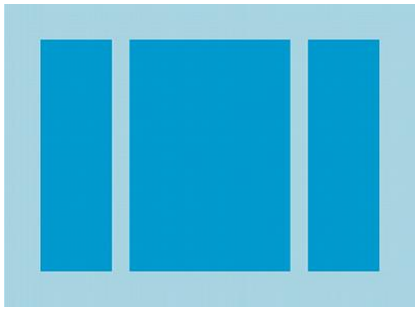
Layouts

Layout Views represent ViewGroups

Layouts

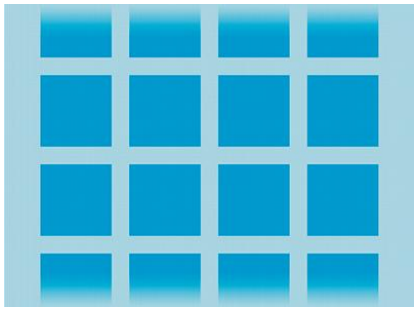
- are specific types of view groups(ScrollView, LinearLayout, RecyclerView)
- are subclasses of ViewGroup
- contain child views
- can be in a row, column, grid, table, absolute

Common Layout Classes

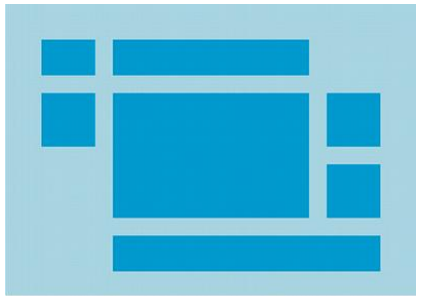


LinearLayout

RelativeLayout



GridLayout



TableLayout

All UI components are related to each other

Common Layout Classes

- **ConstraintLayout (put a constraint on view) -** 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
- **GridView** - 2D scrollable grid

Class Hierarchy vs. Layout Hierarchy

- View class-hierarchy is standard object-oriented class inheritance
  - For example, Button is-a TextView is-a View is-a Object
  - Superclass-subclass relationship

- Layout hierarchy is how Views are visually arranged
  - For example, LinearLayout can contain Buttons arranged in a row
  - Parent-child relationship

Layout created in XML

```
<LinearLayout
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
      <EditText
          ... />
      <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); //activity code method
```

Setting width and height in Java code

Set the width and height of a view:

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

Event Handling

Events

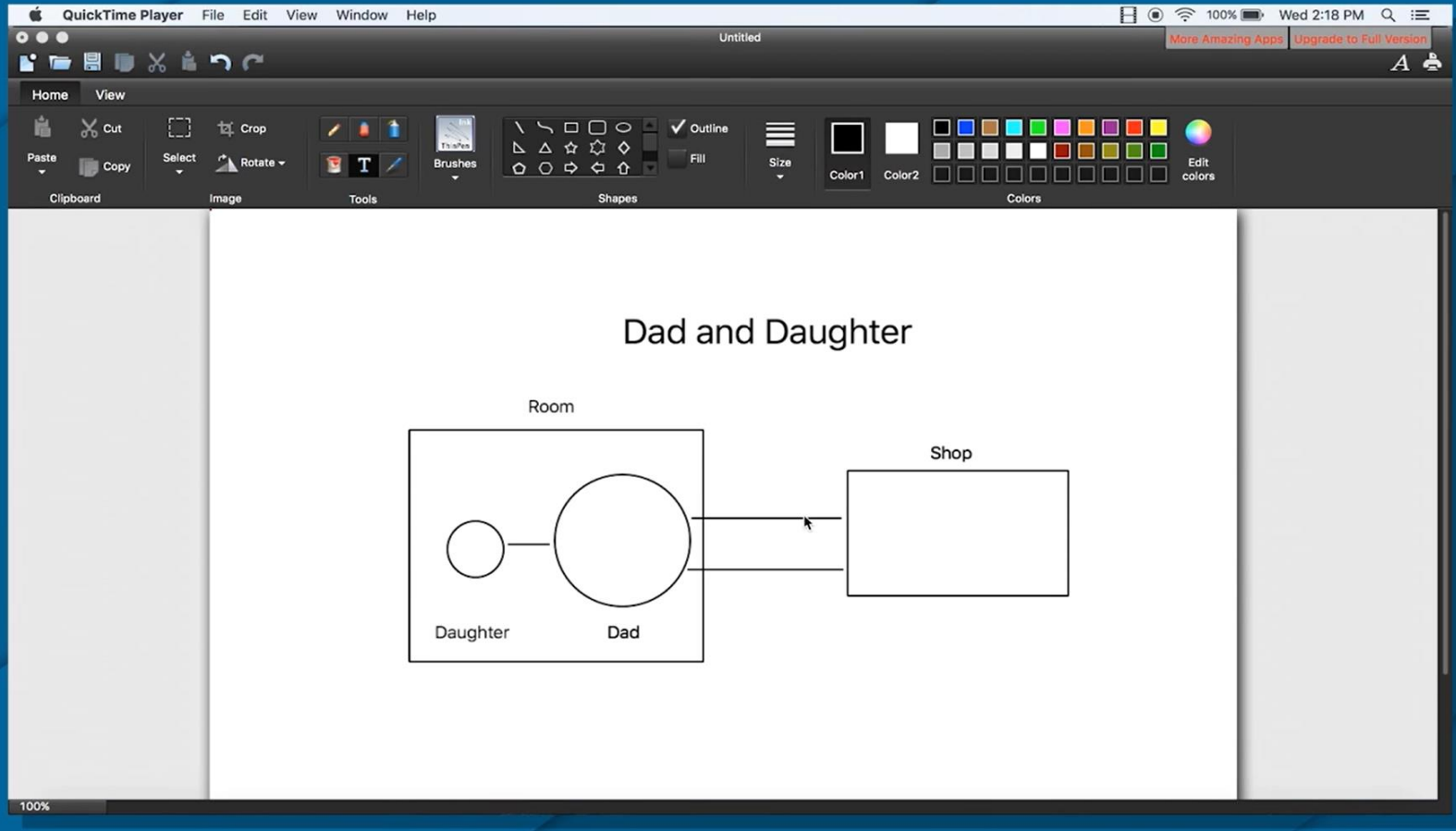
Something that happens

- In UI: Click, tap, drag
- Device: DetectedActivity such as walking, driving, tilting
- Events are "noticed" by the Android system
- 3 things to handle events: event source (daughter), event listener (Dad) and Event handler (Shopkeeper)

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



Handling clicks in XML & Java

Attach handler to view in layout:

```
android:onClick="showToast“
```

Click event can be generated by any view

Implement handler in activity:

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

Toast is the interactive dialog box to notify the user

Setting click handlers 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();
        }
    });
```

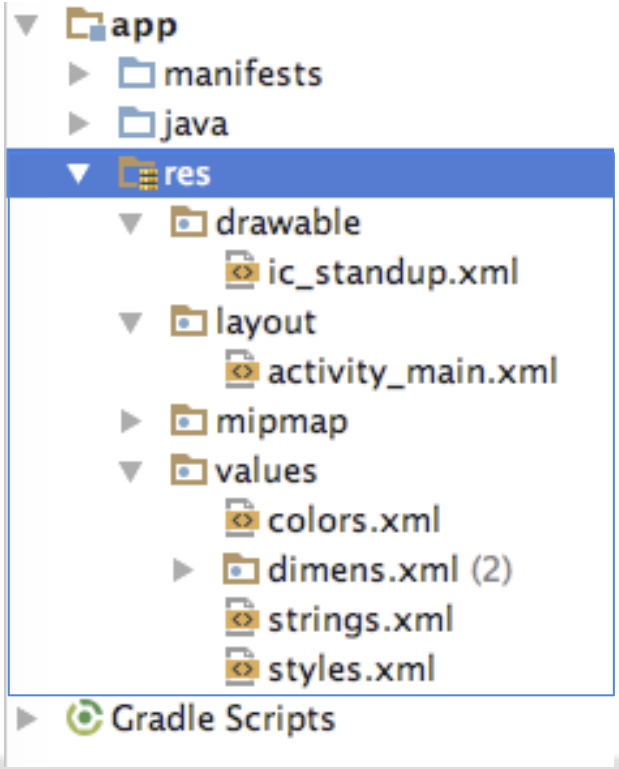
Resources

Resources

Why do we define resources separately?

- Separate static data from code in your layouts.
- Strings, dimensions, images, menu text, colors, styles
- Useful for localization (no need to define the resource repeatedly)

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"`

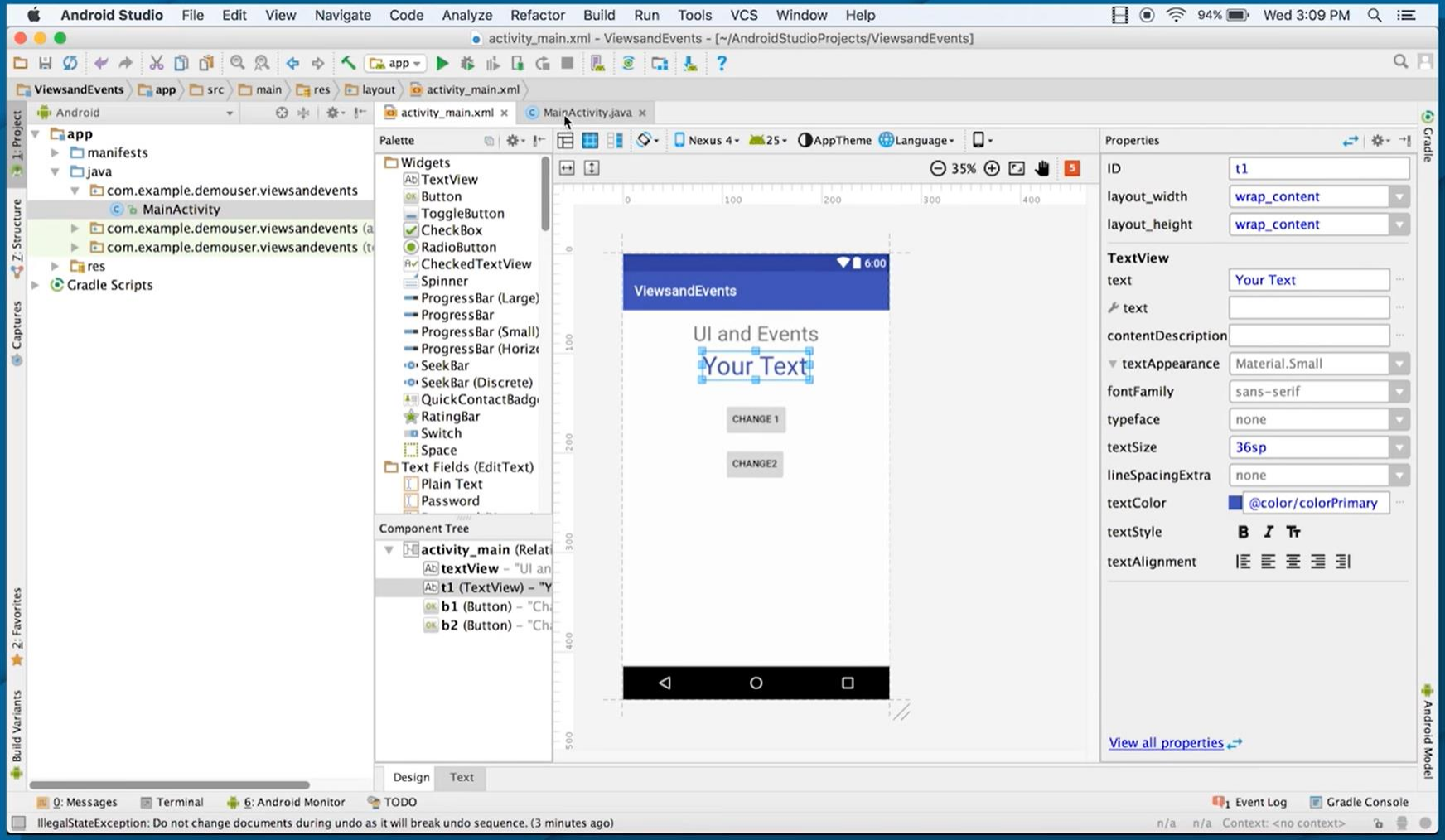
Measurements

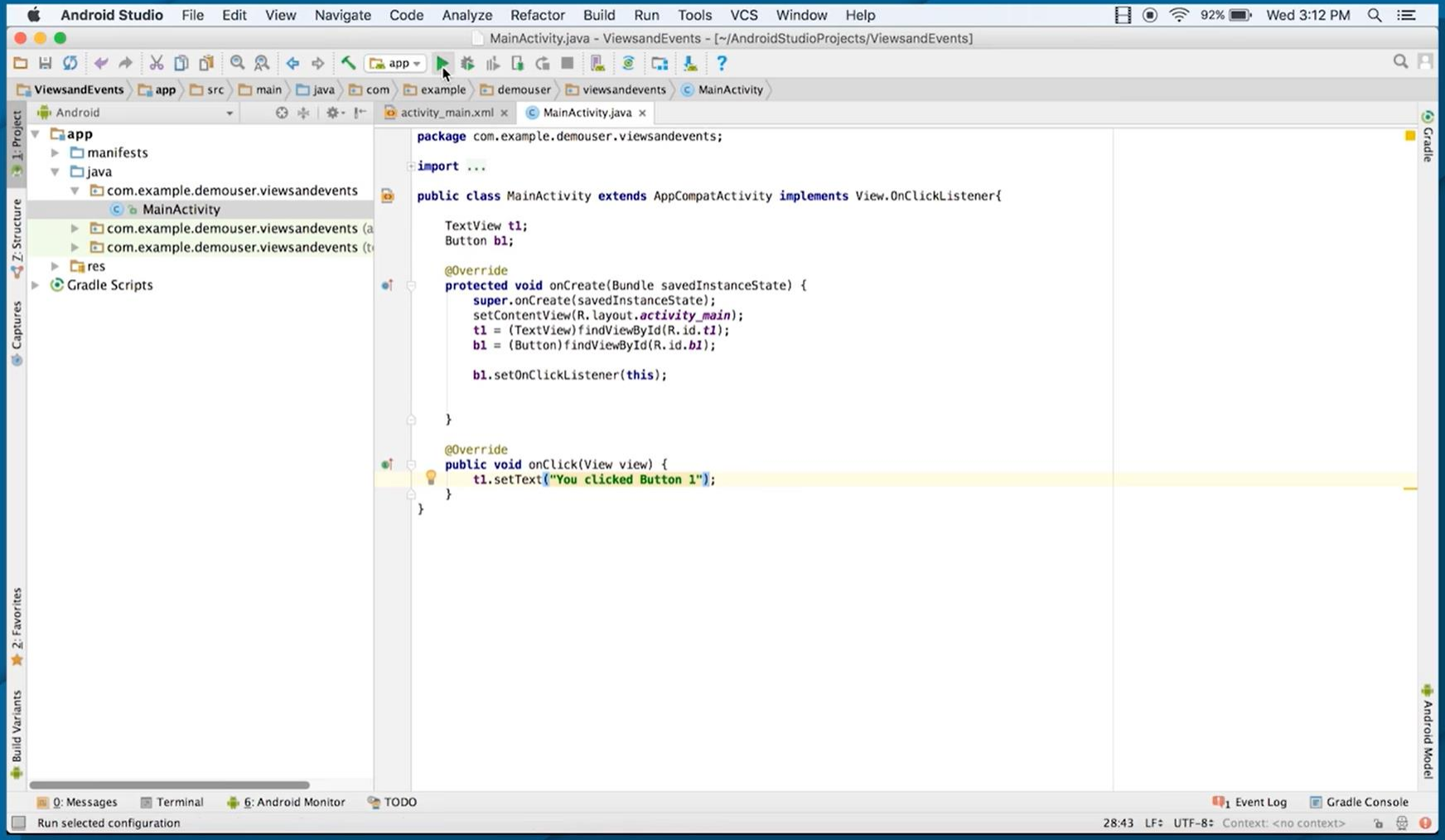
- Device Independent Pixels (dp) - for Views
- Scale Independent Pixels (sp) - for text

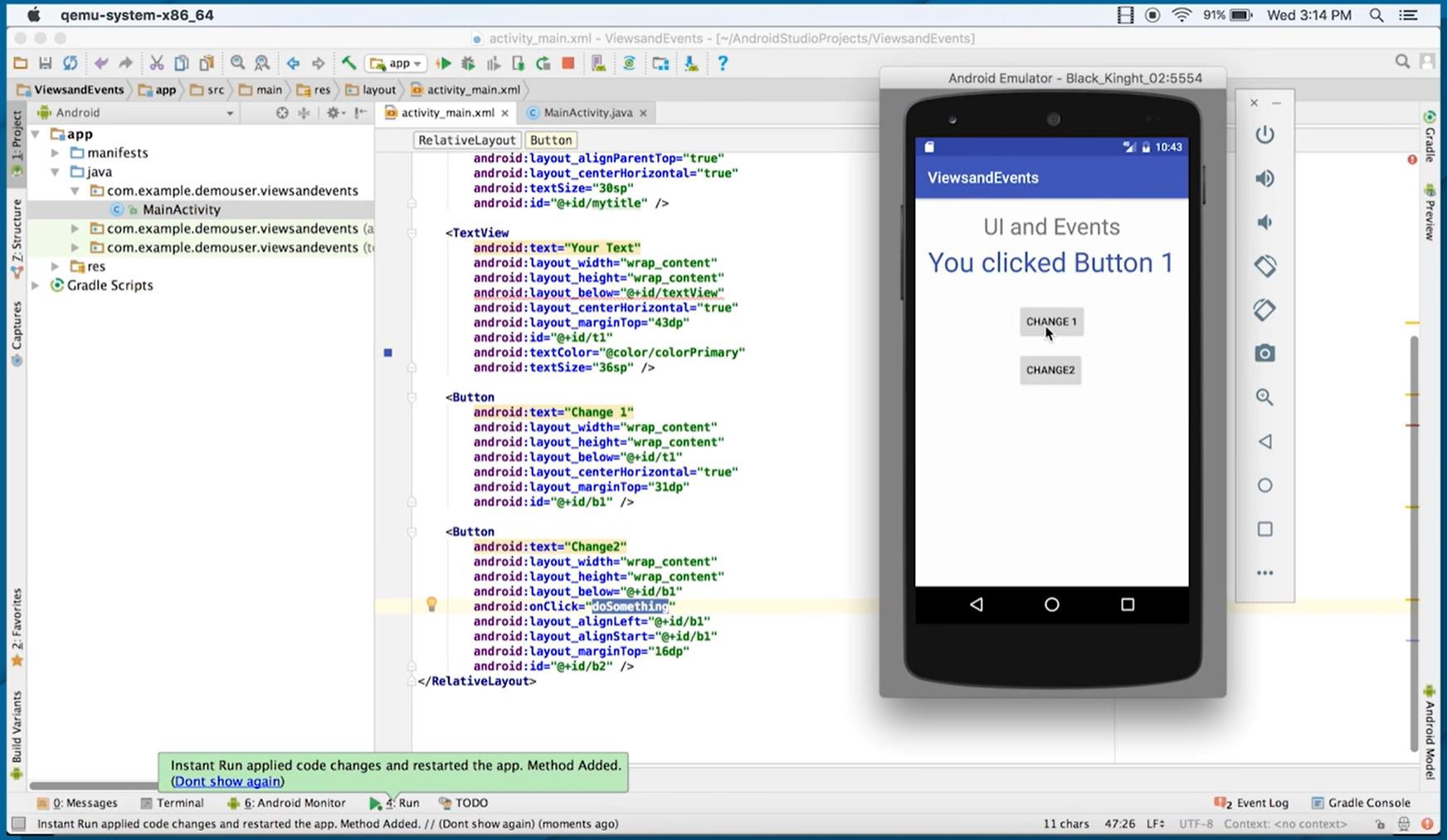
Don't use device-dependent units:

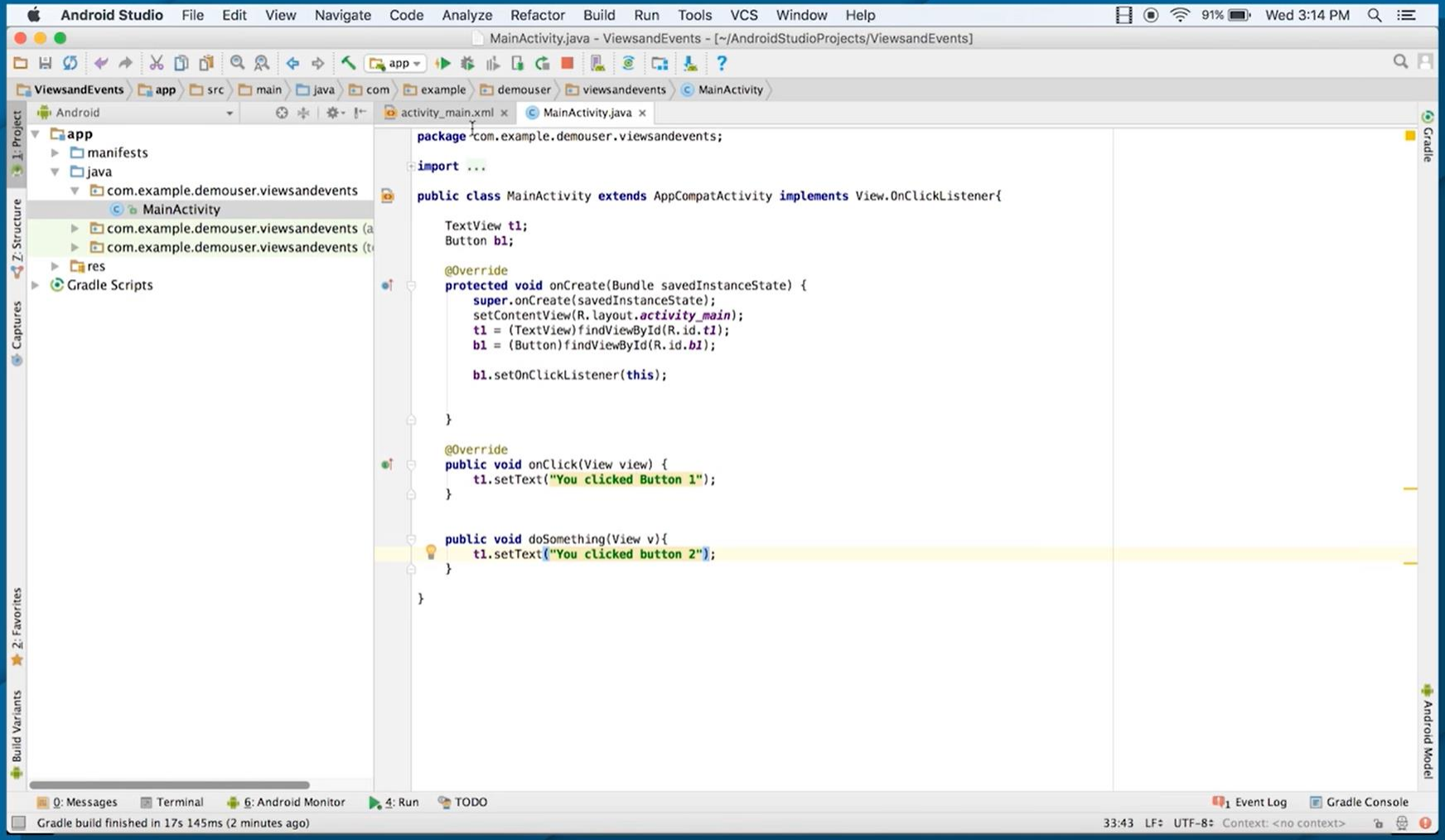
- Actual Pixels (px)
- Actual Measurement (in, mm)
- Points - typography 1/72 inch (pt)

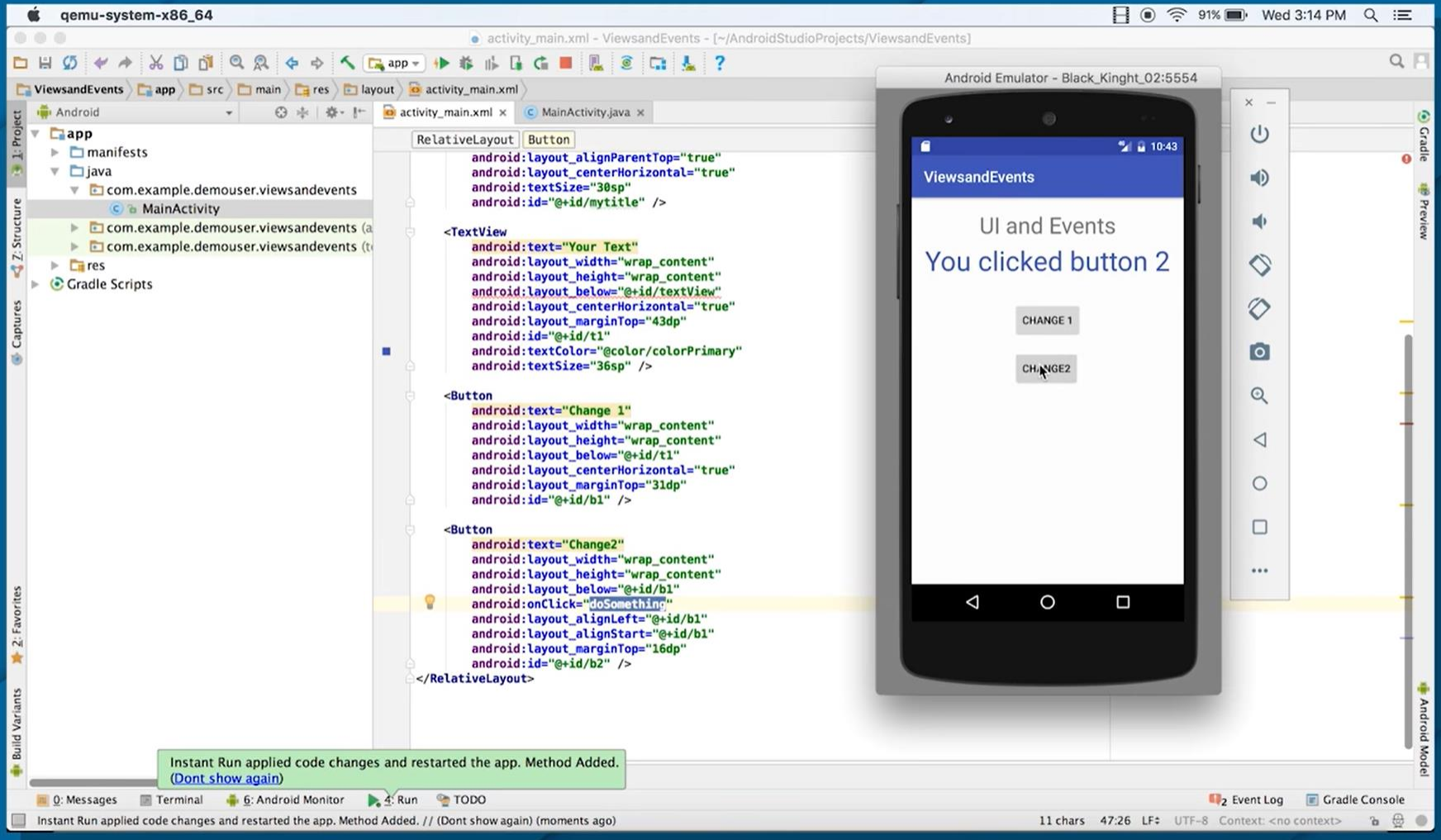
Demo for Layout, View and Event handling


Android Studio File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
activity_main.xml - ViewsandEvents - [~/AndroidStudioProjects/ViewsandEvents]
ViewsandEvents app src main res layout activity_main.xml
Android
app
manifests
java
com.example.demouser.viewsandevents
MainActivity
com.example.demouser.viewsandevents (a
com.example.demouser.viewsandevents (t
res
Gradle Scripts
activity_main.xml
MainActivity.java
Palette
Widgets
TextView
Button
ToggleButton
CheckBox
RadioButton
CheckedTextView
Spinner
ProgressBar (Large)
ProgressBar
ProgressBar (Small)
SeekBar
SeekBar (Discrete)
RatingBar
Switch
Space
Text Fields (EditText)
Plain Text
Password
Component Tree
activity_main (Relat
textView - "UI an
t1 (TextView) - "Y
b1 (Button) - "Ch
b2 (Button) - "Ch
Nexus 4 25 AppTheme Language
35%
6:00
ViewsandEvents
UI and Events
Your Text
CHANGE 1
CHANGE2
Properties
ID t1
layout_width wrap_content
layout_height wrap_content
TextView
text Your Text
text
contentDescription
textAppearance Material.Small
fontFamily sans-serif
typeface none
textSize 36sp
lineSpacingExtra none
textColor @color/colorPrimary
textStyle
textAlignment
View all properties
Design Text
0: Messages Terminal 6: Android Monitor TODO
1 Event Log Gradle Console
IllegalStateException: Do not change documents during undo as it will break undo sequence. (3 minutes ago)
n/a n/a Context: <no context>
1: Project 7: Structure Captures 2: Favorites Build Variants Gradle Android Model


Android Studio File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
92% Wed 3:12 PM
MainActivity.java - ViewsandEvents - [~/AndroidStudioProjects/ViewsandEvents]
app
ViewsandEvents app src main java com example demouser viewsandevents MainActivity
Android
app
manifests
java
com.example.demouser.viewsandevents
MainActivity
com.example.demouser.viewsandevents (a
com.example.demouser.viewsandevents (t
res
Gradle Scripts
activity_main.xml
MainActivity.java
package com.example.demouser.viewsandevents;
import ...
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
TextView t1;
Button b1;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
t1 = (TextView)findViewById(R.id.t1);
b1 = (Button)findViewById(R.id.b1);
b1.setOnClickListener(this);
}
@Override
public void onClick(View view) {
t1.setText("You clicked Button 1");
}
}
1: Project
2: Favorites
7: Structure
Captures
Build Variants
Gradle
Android Model
0: Messages
Terminal
6: Android Monitor
TODO
1 Event Log
Gradle Console
Run selected configuration
28:43 LF UTF-8 Context: <no context>


qemu-system-x86_64
activity_main.xml - ViewsandEvents - [~/AndroidStudioProjects/ViewsandEvents]
ViewsandEvents
app
src
main
res
layout
activity_main.xml
Android
app
manifests
java
com.example.demouser.viewsandevents
MainActivity
com.example.demouser.viewsandevents (a
com.example.demouser.viewsandevents (t
res
Gradle Scripts
activity_main.xml
MainActivity.java
RelativeLayout
Button
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:textSize="30sp"
android:id="@+id/mytitle" />
<TextView
android:text="Your Text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/textView"
android:layout_centerHorizontal="true"
android:layout_marginTop="43dp"
android:id="@+id/t1"
android:textColor="@color/colorPrimary"
android:textSize="36sp" />
<Button
android:text="Change 1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/t1"
android:layout_centerHorizontal="true"
android:layout_marginTop="31dp"
android:id="@+id/b1" />
<Button
android:text="Change2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/b1"
android:onClick="doSomething"
android:layout_alignLeft="@+id/b1"
android:layout_alignStart="@+id/b1"
android:layout_marginTop="16dp"
android:id="@+id/b2" />
</RelativeLayout>
Android Emulator - Black_Kinght_02:5554
10:43
ViewsandEvents
UI and Events
You clicked Button 1
CHANGE 1
CHANGE2
Instant Run applied code changes and restarted the app. Method Added.
(Dont show again)
0: Messages
Terminal
6: Android Monitor
4: Run
TODO
Event Log
Gradle Console
Instant Run applied code changes and restarted the app. Method Added. // (Dont show again) (moments ago)
11 chars
47:26
LF
UTF-8
Context: <no context>
Gradle
Preview
Android Model
Captures
Structure
Favorites
Build Variants


Android Studio File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MainActivity.java - ViewsandEvents - [~/AndroidStudioProjects/ViewsandEvents]
ViewsandEvents app src main java com example demouser viewsandevents MainActivity
activity_main.xml
MainActivity.java
app
manifests
java
com.example.demouser.viewsandevents
MainActivity
com.example.demouser.viewsandevents (a
com.example.demouser.viewsandevents (t
res
Gradle Scripts
package com.example.demouser.viewsandevents;
import ...
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
TextView t1;
Button b1;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
t1 = (TextView)findViewById(R.id.t1);
b1 = (Button)findViewById(R.id.b1);
b1.setOnClickListener(this);
}
@Override
public void onClick(View view) {
t1.setText("You clicked Button 1");
}
public void doSomething(View v){
t1.setText("You clicked button 2");
}
}
0: Messages
Terminal
6: Android Monitor
4: Run
TODO
Event Log
Gradle Console
Gradle build finished in 17s 145ms (2 minutes ago)
33:43 LF UTF-8 Context: <no context>
Gradle
Android Model
Build Variants
2: Favorites
Captures
Z: Structure
1: Project


qemu-system-x86_64
activity_main.xml - ViewsandEvents - [~/AndroidStudioProjects/ViewsandEvents]
ViewsandEvents
app
src
main
res
layout
activity_main.xml
Android
app
manifests
java
com.example.demouser.viewsandevents
MainActivity
com.example.demouser.viewsandevents (a
com.example.demouser.viewsandevents (t
res
Gradle Scripts
activity_main.xml
MainActivity.java
RelativeLayout
Button
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:textSize="30sp"
android:id="@+id/mytitle" />
<TextView
android:text="Your Text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/textView"
android:layout_centerHorizontal="true"
android:layout_marginTop="43dp"
android:id="@+id/t1"
android:textColor="@color/colorPrimary"
android:textSize="36sp" />
<Button
android:text="Change 1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/t1"
android:layout_centerHorizontal="true"
android:layout_marginTop="31dp"
android:id="@+id/b1" />
<Button
android:text="Change2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/b1"
android:onClick="doSomething"
android:layout_alignLeft="@+id/b1"
android:layout_alignStart="@+id/b1"
android:layout_marginTop="16dp"
android:id="@+id/b2" />
</RelativeLayout>
Android Emulator - Black_Kinght_02:5554
10:43
ViewsandEvents
UI and Events
You clicked button 2
CHANGE 1
CHANGE2
Instant Run applied code changes and restarted the app. Method Added.
(Dont show again)
0: Messages
Terminal
6: Android Monitor
4: Run
TODO
Event Log
Gradle Console
Instant Run applied code changes and restarted the app. Method Added. // (Dont show again) (moments ago)
11 chars
47:26
LF
UTF-8
Context: <no context>
Gradle
Preview
Android Model
Captures
Structure
Favorites
Build Variants

What's Next?

- Concept Chapter: 1.2 C Layouts, Views, and Resources
- Practicals:
  - 1.2A P Make Your First Interactive UI
  - 1.2B P Using Layouts

END

Android Developer Fundamentals

Hello World

Lesson 1

1.3 Text and Scrolling Views

Contents

Creating text views, putting those text views into ScrollView and create beautiful forms.

- TextView
- ScrollView

TextView

TextView for text

- TextView is a view for displaying single and multi-line text
- EditText is a subclass of TextView with editable text
- Controlled with layout attributes
- Set text statically from a string resource in XML, or dynamically from Java code and any source

Formatting text in string resource

- Use <b> and <i> HTML tags for bold and italics
- All other HTML tags are ignored
- String resources: one unbroken line = one paragraph
- \n starts a new a line or paragraph
- Escape apostrophes and quotes with backslash (\", \')
- Escape any non-ASCII characters with backslash (\)

Creating TextView in XML

```
<TextView android:id="@+id/textview"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/my_story"/>
```

Common TextView attributes

android:**text**—text to display

android:**textColor**—color of text

android:**textAppearance**—predefined style or theme

android:**textSize**—text size in `sp`

android:**textStyle**—`normal`, `bold`, `italic`, or `bold|italic`

android:**typeface**—`normal`, `sans`, `serif`, or `monospace`

android:**lineSpacingExtra**—extra space between lines in `sp`

Formatting active web links

```
<string name="article_text">... www.rockument.com ...</string>
```

```
<TextView
    android:id="@+id/article"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:autoLink="web"
    android:text="@string/article_text"/>
```

android:autoLink="web" to make your textview active for websites. Similarly, you can make your textview active for email or phone or map

Don't use HTML for a web link in free-form text

autoLink values:"web", "email", "phone", "map", "all"

Creating TextView in Java code

```
TextView myTextview = new TextView(this);
myTextView.setWidth(LayoutParams.MATCH_PARENT);
myTextView.setHeight(LayoutParams.WRAP_CONTENT);
myTextView.setMinLines(3);
myTextView.setText(R.string.my_story);
myTextView.append(userComment);
```

ScrollView

What about large amounts of text?

- The user may need to scroll.
  - News stories, articles, ...
- To allow users to scroll a TextView, embed it in a `ScrollView`.
- ScrollView is categorized in ViewGroup. You can put other views in it. But only one view at a time. You use it to show large text that goes beyond your screen. If you want to design a form with so many text views, buttons, EditText view and scroll the form, you need to define ScrollView at the root and inside that we defind layout and layout can contain as many views as you want.
- Other Views can be embedded in a `ScrollView`.
  - `LinearLayout`, `TextView`, `Button`, ...

ScrollView for scrolling content

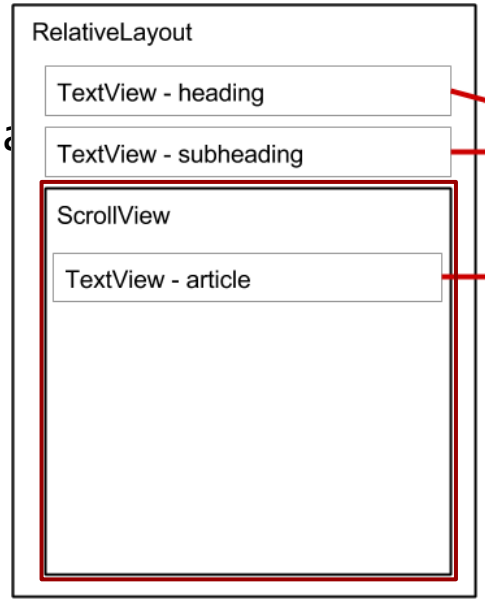
- ScrollView is a subclass of FrameLayout
- Can only hold **one** view (which can be a ViewGroup)
- Holds all content in memory
- Not good for long texts, complex layouts
- Do not nest multiple scrolling views
- Use HorizontalScrollView for horizontal scrolling
- Use a RecyclerView for lists

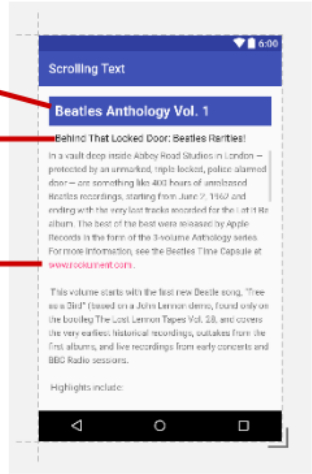
ScrollView layout with one TextView

```
<ScrollView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_below="@id/article_subhea

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    .../>

</ScrollView>
```



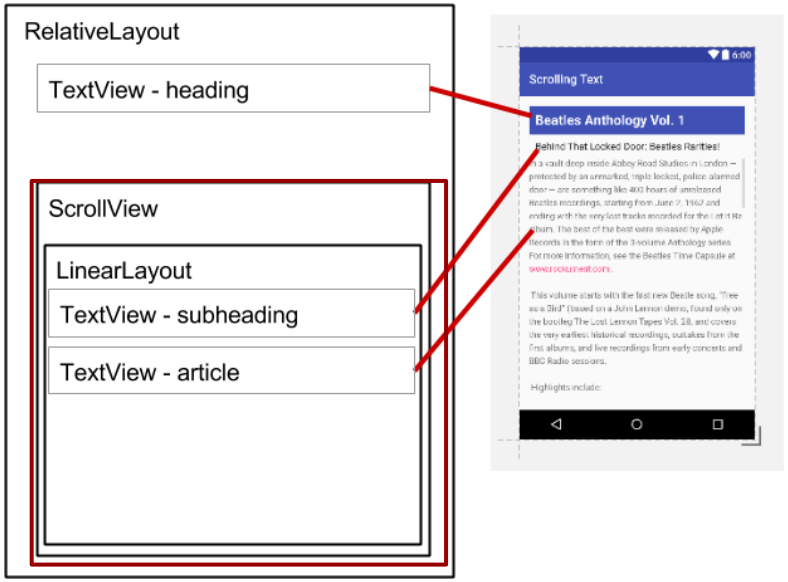


ScrollView layout with a view group

```
<ScrollView ...
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/article_subheading"
            .../>

        <TextView
            android:id="@+id/article" ... />
    </LinearLayout>
</ScrollView>
```

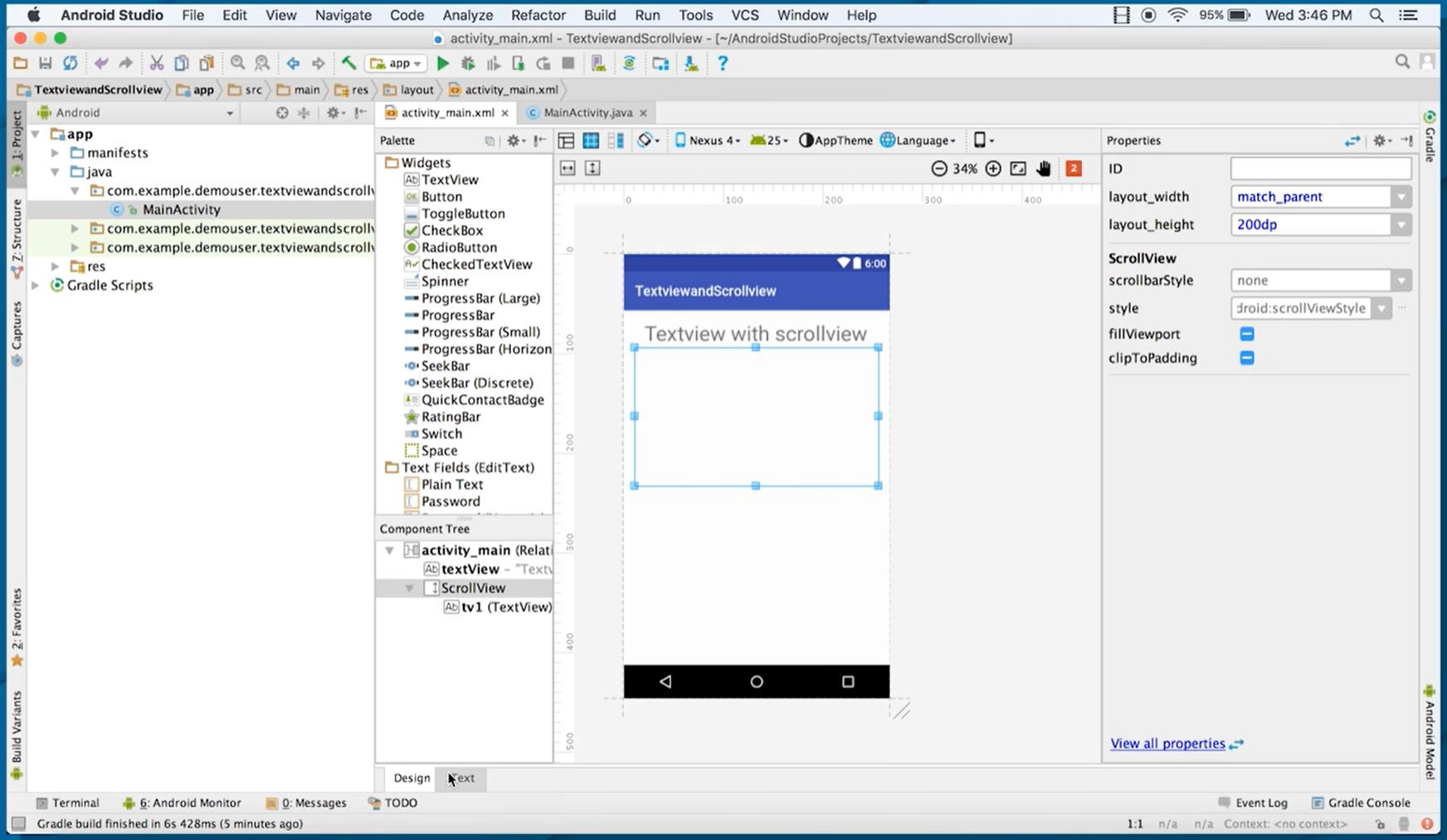


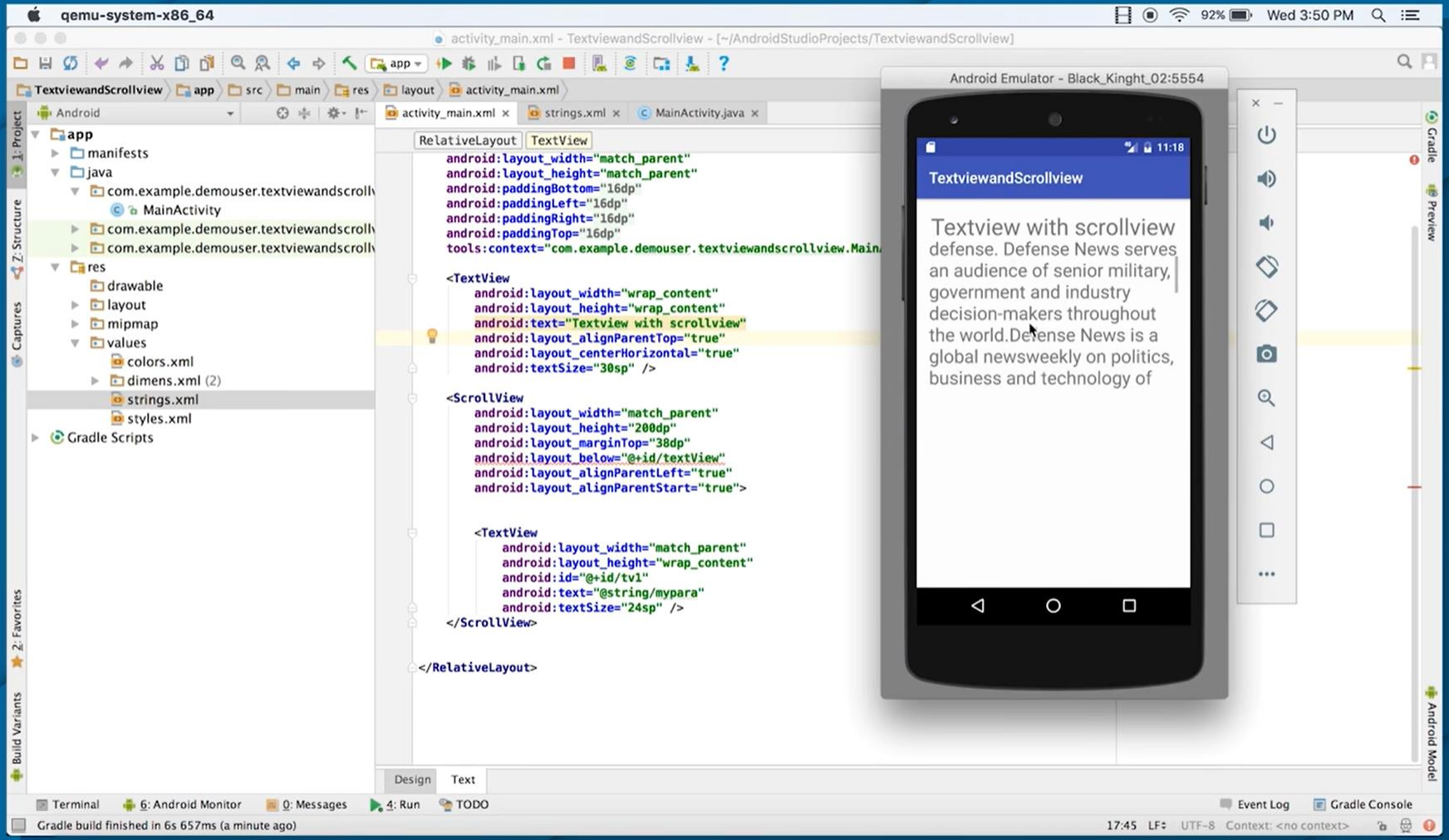
ScrollView with image and button

```
<ScrollView...>
    <LinearLayout...>
        <ImageView.../>
        <Button.../>
        <TextView.../>
    </LinearLayout>
</ScrollView>
```

One child of ScrollView which can be a layout

Children of the layout


Android Studio File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
95% Wed 3:46 PM
activity_main.xml - TextviewandScrollview - [~/AndroidStudioProjects/TextviewandScrollview]
TextviewandScrollview app src main res layout activity_main.xml
Android
app
manifests
java
com.example.demouser.textviewandscrollv
MainActivity
com.example.demouser.textviewandscrollv
com.example.demouser.textviewandscrollv
res
Gradle Scripts
activity_main.xml
MainActivity.java
Palette
Widgets
TextView
Button
ToggleButton
CheckBox
RadioButton
CheckedTextView
Spinner
ProgressBar (Large)
ProgressBar
ProgressBar (Small)
ProgressBar (Horizon
SeekBar
SeekBar (Discrete)
QuickContactBadge
RatingBar
Switch
Space
Text Fields (EditText)
Plain Text
Password
Component Tree
activity_main (Relat
textView - "Textv
ScrollView
tv1 (TextView)
Nexus 4 25 AppTheme Language
34%
6:00
TextviewandScrollview
Textview with scrollview
Properties
ID
layout_width match_parent
layout_height 200dp
ScrollView
scrollbarStyle none
style droid:scrollViewStyle
fillViewport
clipToPadding
View all properties
Design Text
Terminal 6: Android Monitor 0: Messages TODO
Event Log Gradle Console
Gradle build finished in 6s 428ms (5 minutes ago)
1:1 n/a n/a Context: <no context>
1: Project 7: Structure Captures 2: Favorites Build Variants
Gradle Android Model


qemu-system-x86_64
activity_main.xml - TextviewandScrollview - [~/AndroidStudioProjects/TextviewandScrollview]
TextviewandScrollview
app
src
main
res
layout
activity_main.xml
Android
app
manifests
java
com.example.demouser.textviewandscrollv
MainActivity
com.example.demouser.textviewandscrollv
com.example.demouser.textviewandscrollv
res
drawable
layout
mipmap
values
colors.xml
dimens.xml (2)
strings.xml
styles.xml
Gradle Scripts
activity_main.xml
strings.xml
MainActivity.java
RelativeLayout
TextView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="16dp"
android:paddingLeft="16dp"
android:paddingRight="16dp"
android:paddingTop="16dp"
tools:context="com.example.demouser.textviewandscrollview.Main
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Textview with scrollview"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:textSize="30sp" />
<ScrollView
android:layout_width="match_parent"
android:layout_height="200dp"
android:layout_marginTop="38dp"
android:layout_below="@+id/textView"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true">
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/tv1"
android:text="@string/mypara"
android:textSize="24sp" />
</ScrollView>
</RelativeLayout>
Design
Text
Android Emulator - Black_Kinght_02:5554
11:18
TextviewandScrollview
Textview with scrollview
defense. Defense News serves an audience of senior military, government and industry decision-makers throughout the world.Defense News is a global newsweekly on politics, business and technology of
Terminal
6: Android Monitor
0: Messages
4: Run
TODO
Event Log
Gradle Console
Gradle build finished in 6s 657ms (a minute ago)
17:45
LF
UTF-8
Context: <no context>
92%
Wed 3:50 PM

Learn more

Developer Documentation:

- TextView
- ScrollView and HorizontalScrollView
- String Resources

Other:

- Android Developers Blog: Linkify your Text!
- Codepath: Working with a TextView

What's Next?

- Concept Chapter: 1.3 C Text and Scrolling Views
- Practical: 1.3 P Working with TextView Elements

END

Android Developer Fundamentals

Activities and Intents

Lesson 2

2.1 Activities
The whole screen what you see on yourdevice

Contents

- Activities
- Defining an activity
- Starting a new activity with an intent
- Passing data between activities with extras
- Navigating between activities

Application Components

1. Activity
2. Service
3. Broadcast receiver
4. Content provider

Activities (high-level view)

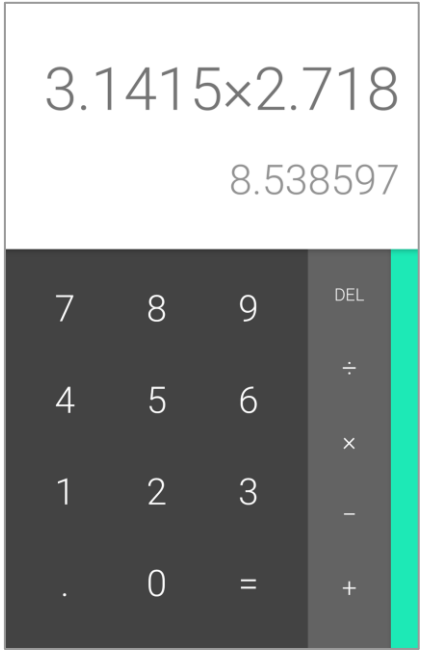
What is an Activity?

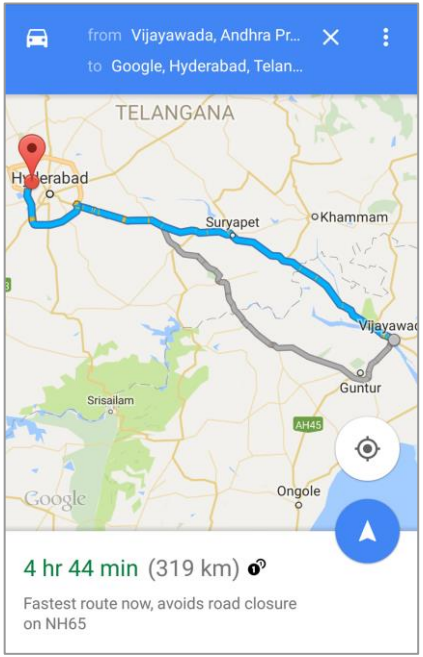
- An `Activity` is an application component
- Represents one window, one hierarchy of views
- Typically fills the screen, but can be embedded in other activity or a appear as floating window (When you drag your notification bar, there is an activity already running in the background and notification bar is floating actvity
- Java class, typically one activity in one file
- An activity comprises of two things: A Java class and an UI layout

What does an Activity do?

- Represents an activity, such as ordering groceries, sending email, or getting directions
- Handles user interactions, such as button clicks, text entry, or login verification
- Can start other activities in the same or other apps
- Has a life cycle—is created, started, runs, is paused, resumed, stopped, and destroyed

Examples of activities





Apps and activities

- Activities are loosely tied together to make up an app
- First activity user sees is typically called "main activity“/launcher activity
- Activities can be organized in parent-child relationships in the Android manifest to aid navigation

Layouts and Activities

- An activity typically has a UI layout
- Layout is usually defined in one or more XML files
- Activity "inflates" layout as part of being created
- We need to register all the activities in manifest file
- Activity is not just a Java file. You also need a UI file
- If you need to show any UI on screen then first you have to create UI in your XML file and then connect XML file to Java file

Implementing Activities

Implement new activities

1. Define layout in XML
2. Define Activity Java class
    - extends AppCompatActivity
3. Connect Activity with Layout
    - Set content view in onCreate()
4. Declare Activity in the Android manifest

1. Define layout in XML

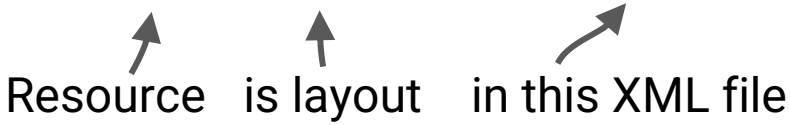
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Let's Shop for Food!" />
</RelativeLayout>
```

2. Define Activity Java class

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

3. Connect activity with layout

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```



4. Declare activity in Android manifest

```
<activity android:name=".MainActivity">
```

4. Declare main activity in manifest

Main Activity needs to include intent to start from launcher icon

```
<activity android:name=".MainActivity">
   <intent-filter>
       <action android:name="android.intent.action.MAIN" />
       <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
```

If you want to provide a behaviour to a particular activity then use intent-filter. Which activity you want to be a launcher activity. This you can do with intent filter. Intent-filter says that this is your main activity. If there are 10 activities in your application then which activity to be shown when user opens your application?

Intents

Under the hood all screens you see are launched by intents.
Intent is used to navigate the screens of your app

What is an intent?

An intent is a description of an operation to be performed.
An Intent is an object used to request an action from another app component via the Android system. Using intent, we invoke activity, broadcast receiver and service.
Intent is used to invoke all Android's fundamental components like activity, service and broadcast receiver except content provider.
Whenever an app is opened, There is an intent which is been fired. Android system now knows that it has to launch the intent who is the main and the launcher.

Whenever you open your app, an intent is fired. Android system knows that it has to open the intent which is responsible to launch the main activity.

What can intents do?

- Start activities
  - A button click starts a new activity
  - Clicking Share opens an app that allows you to post a photo
- Start services
  - Initiate downloading a file in the background
- Deliver broadcasts
  - The system informs everybody that the phone is now charging

Explicit and implicit intents

Explicit Intent (source and destination are known)

- Starts a specific activity
- This type of intent is used inside an application. That is to transfer from one activity to another activity within an application (from 2nd activity to 7th activity)

Implicit Intent (used to invoke activity from different app)

- Asks system to find an activity that can handle this request
  - Clicking Share opens a chooser with a list of apps

Starting Activities

Start an Activity with an explicit intent

To start a specific activity, use an explicit intent

1. Create an intent
   - `Intent intent = new Intent(this, ActivityName.class);`
2. Use the intent to start the activity
   - `startActivity(intent);`

Start an Activity with implicit intent

To ask Android to find an Activity to handle your request, use an implicit intent

1. Create an intent
   - `Intent intent = new Intent(action, uri);`
2. Use the intent to start the activity
   - `startActivity(intent);`

Implicit Intents - Examples

Show a web page

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

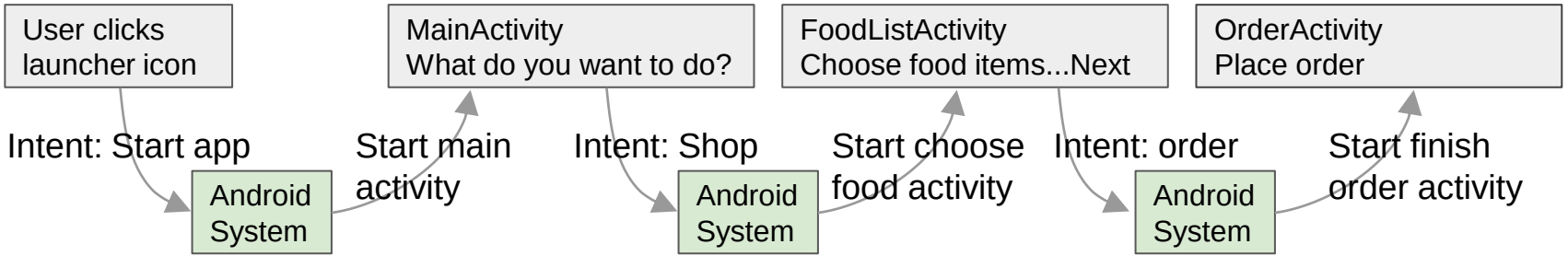
Action_VIEW is a general action. Depending on data, a particular app will open.

Dial a phone number

```
Uri uri = Uri.parse("tel:8005551234");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
startActivity(it);
```

How Activities Run

- All activities are managed by the Android runtime
- Started by an "intent", a message to the Android runtime to run an activity

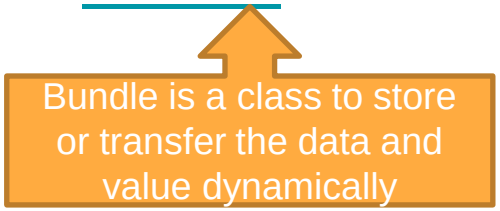


Sending and Receiving Data

Two types of sending data with intents

- Data—one piece of information whose data location can be represented by an URI

- Extras—one or more pieces of information as a collection of key-value pairs in a Bundle



A screen with a name field where you will enter the name and a button. When you click this button you want the name to be transferred to the next activity. We can use extras.

Sending and retrieving data

In the first (sending) activity:

1. Create the Intent object
2. Put data or extras into that intent
3. Start the new activity with `startActivity()`

In the second (receiving) activity,:

1. Get the intent object the activity was started with
2. Retrieve the data or extras from the Intent object

Putting a URI as intent data

```
// A web page URL
intent.setData(
    Uri.parse("http://www.google.com"));

// a Sample file URI
intent.setData(
    Uri.fromFile(new File("/sdcard/sample.jpg")));
```

Put information into intent extras

- `putExtra(String name, int value)`
  ⇒ `intent.putExtra("level", 406);`
- `putExtra(String name, String[] value)`
  ⇒
```
String[] foodList = {"Rice", "Beans", "Fruit"};
intent.putExtra("food", foodList);
```
- `putExtras(bundle);`
  ⇒ if lots of data, first create a bundle and pass the bundle.
- See documentation for all

Sending data to an activity with extras

```
public static final String EXTRA_MESSAGE_KEY =
    "com.example.android.twoactivities.extra.MESSAGE";

Intent intent = new Intent(this, SecondActivity.class);
String message = "Hello Activity!";
intent.putExtra(EXTRA_MESSAGE_KEY, message);
startActivity(intent);
```

Get data from intents

- `getData();`
  `⇒ Uri locationUri = intent.getData();`
- `int getIntExtra (String name, int defaultValue)`
  `⇒ int level = intent.getIntExtra("level", 0);`
- `Bundle bundle = intent.getExtras();`
  ⇒ Get all the data at once as a bundle.
- See documentation for all

Returning data to the starting activity

1. Use `startActivityForResult()` to start the second activity
2. To return data from the second Activity:
   - Create a ***new*** Intent
   - Put the response data in the Intent using `putExtra()`
   - Set the result to `Activity.RESULT_OK` or `RESULT_CANCELED`, if the user cancelled out
   - call `finish()` to close the activity
1. Implement `onActivityResult()` in first activity

startActivityForResult()

`startActivityForResult(intent, requestCode);`

- Starts activity (`intent`), assigns it identifier (`requestCode`)
- Returns data via intent extras
- When done, pop stack, return to previous activity, and execute `onActivityResult()` callback to process returned data
- Use `requestCode` to identify which activity has "returned"

1. startActivityForResult() Example

```
public static final int CHOOSE_FOOD_REQUEST = 1;

Intent intent = new Intent(this, ChooseFoodItemsActivity.class);
startActivityForResult(intent, CHOOSE_FOOD_REQUEST);
```

2. Return data and finish second activity

```
// Create an intent
Intent replyIntent = new Intent();

// Put the data to return into the extra
replyIntent.putExtra(EXTRA_REPLY, reply);

// Set the activity's result to RESULT_OK
setResult(RESULT_OK, replyIntent);

// Finish the current activity
finish();
```

3. Implement onActivityResult()

```
public void onActivityResult(int requestCode,
                             int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  if (requestCode == TEXT_REQUEST) { // Identify activity
    if (resultCode == RESULT_OK) { // Activity succeeded
      String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
      // … do something with the data
 }}}
```

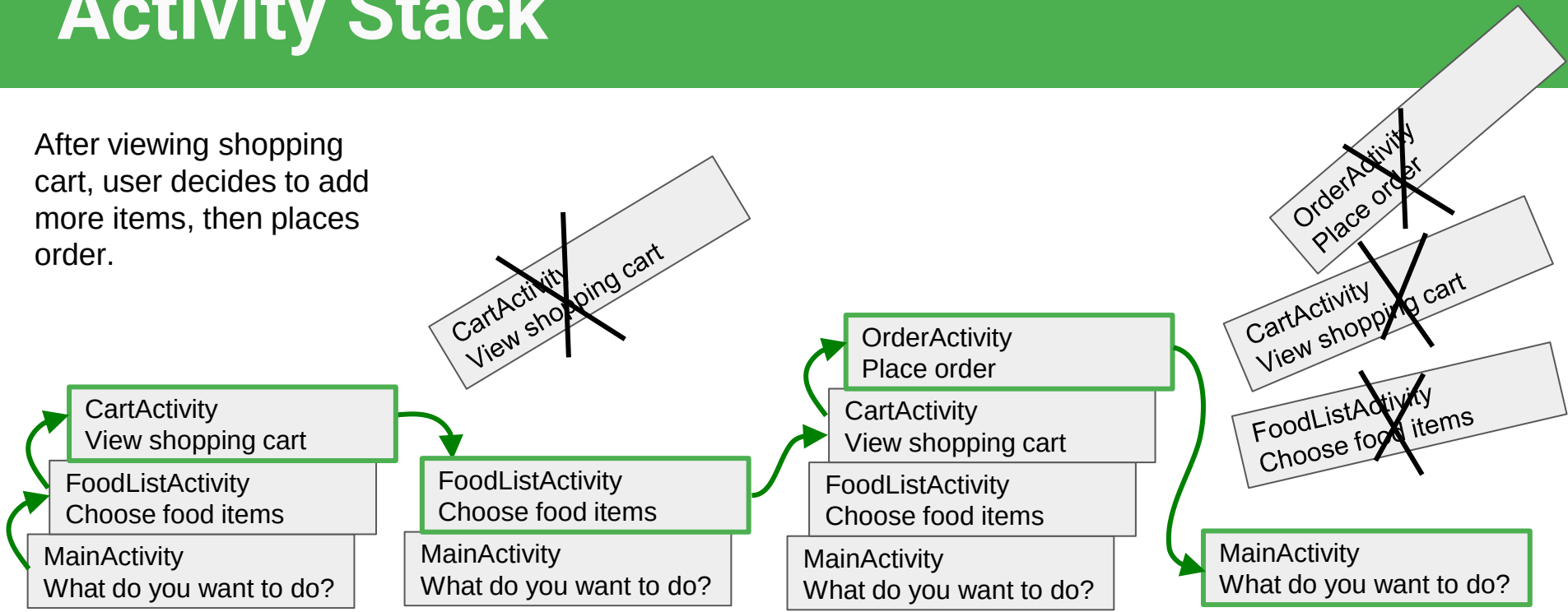
Navigation

Activity stack

- When a new activity is started, the previous activity is stopped and pushed on the activity back stack
- Last-in-first-out-stack—when the current activity ends, or the user presses the Back  button, it is popped from the stack and the previous activity resumes

Activity Stack

After viewing shopping cart, user decides to add more items, then places order.



Two forms of navigation

Temporal or back navigation

- provided by the device's back button
- controlled by the Android system's back stack

Ancestral or up navigation

- provided by the app's action bar
- controlled by defining parent-child relationships between activities in the Android manifest

Back navigation

- Back stack preserves history of recently viewed screens
- Back stack contains all the activities that have been launched by the user in reverse order *for the current task*
- Each task has its own back stack
- Switching between tasks activates that task's back stack
- Launching an activity from the home screen  starts a new task
- Navigate between tasks  with the overview or recent tasks screen

Up navigation

- Goes to parent of current activity
- Define an activity's parent in Android manifest
- Set parentActivityName

```
<activity
    android:name=".ShowDinnerActivity"
    android:parentActivityName=".MainActivity" >
</activity>
```

Learn more

Learn more

- Android Application Fundamentals
- Starting Another Activity
- Activity (API Guide)
- Activity (API Reference)
- Intents and Intent Filters (API Guide)
- Intent (API Reference)
- Navigation

What's Next?

- Concept Chapter: 2.1 C Understanding Activities and Intents
- Practical: 2.1 P Create and Start Activities

END