Android UI Tour
Home Screen

- First screen, includes **favorites** tray (e.g. phone, mail, messaging, web, etc)
All Apps Screen

- Accessed by touching **all apps button** in favorites tray
- Can swipe through multiple app screens, customizable

---

Android 5.0

**all apps button**
Recent Apps Screen

- Accessed by touching recent apps button
- Shows recently used apps, touch app to switch to it

Android 5.0
Status Bar and Notification Screen

- **Status**: time, battery, cell signal strength, bluetooth enabled, etc
- **Notification**: wifi, mail, bewell, voicemail, usb active, music, etc
Android Apps: Big Picture
UI Design using XML

- UI design code (XML) separate from the program (Java)

- Why? Can modify UI without changing Java program

- **Example**: Shapes, colors can be changed in XML file without changing Java program

- UI designed using either:
  - Drag-and drop graphical (WYSIWYG) tool or
  - Programming Extensible Markup Language (XML)

- **XML**: Markup language, both human-readable and machine-readable"
Android App Compilation

- Android Studio compiles code, data and resource files into **Android PacKage** (filename.apk).
  - .apk is similar to .exe on Windows

- Apps download from Google Play, or copied to device as filename.apk

- Installation = installing apk file
Activities

- Activity? 1 Android screen or dialog box
- Apps
  - Have at least 1 activity that deals with UI
  - Entry point, similar to `main()` in C
  - Typically have multiple activities

- Example: A camera app
  - **Activity 1**: to focus, take photo, launch activity 2
  - **Activity 2**: to view photo, save it

- Activities
  - independent of each other
  - E.g. Activity 1 can write data, read by activity 2
  - App Activities derived from Android’s **Activity** class
Our First Android App
3 Files in “Hello World” Android Project

- **Activity_my.xml**: XML file specifying screen layout
- **MainActivity.Java**: Java code to define behavior, actions taken when button clicked (intelligence)
- **AndroidManifest.xml**: Lists all screens, components of app
  - Analogous to a table of contents for a book
  - E.g. Hello world program has 1 screen, so AndroidManifest.xml has 1 item listed
  - App starts running here (like main( ) in C)

- **Note**: Android Studio creates these 3 files for you
Execution Order

Start in AndroidManifest.xml
Read list of activities (screens)
Start execution from Activity tagged Launcher

Create/execute activities (declared in java files)
E.g. MainActivity.Java

Format each activity using layout
In XML file (e.g. Activity_my.xml)

Next: Samples of AndroidManifest.xml
Hello World program
Inside “Hello World” AndroidManifest.xml

This file is written using XML namespace and tags and rules for android.

```xml
<?xml version="1.0"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.commonsware.android.skeleton"
    android:versionCode="1"
    android:versionName="1.0">

    <application>
        <activity
            android:name="Now"
            android:label="Now">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
        </activity>
    </application>

</manifest>
```

Your package name

Android version

List of activities (screens) in your app

One activity (screen) designated LAUNCHER. The app starts running here.
Execution Order

Start in `AndroidManifest.xml`
Read list of activities (screens)
Start execution from Activity tagged Launcher

Create/execute activities (declared in java files)
E.g. `MainActivity.Java`

Format each activity using layout
In XML file (e.g. `Activity_my.xml`)
**Example Activity Java file**  
(E.g. MainActivity.java)

- **Package declaration**
  ```java
  package com.commonsware.empublite;
  ```

- **Import needed classes**
  ```java
  import android.app.Activity;
  import android.os.Bundle;
  ```

- **My class inherits from Android activity class**
  ```java
  public class EmPubLiteActivity extends Activity {
  ```

- **Initialize by calling onCreate() method of base Activity class**
  ```java
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
  ```

**Note:** Android calls your Activity’s `onCreate` method once it is created.
Execution Order

Start in AndroidManifest.xml
Read list of activities (screens)
Start execution from Activity tagged Launcher

Create/execute activities
(declared in java files)
E.g. MainActivity.Java

Format each activity using layout
In XML file (e.g. Activity_my.xml)
Simple XML file Designing UI

- After choosing the layout, then widgets added to design UI
- XML Layout files consist of:
  - UI components (boxes) called **Views**
  - Different types of views. E.g
    - **TextView**: contains text,
    - **ImageView**: picture,
    - **WebView**: web page
- **Views** arranged into layouts or **ViewGroups**

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".EmPubLiteActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world"/>

</RelativeLayout>
```
Android Files
The root folder has the same name as your project.

The `build/` folder contains files that Android Studio creates for you. You don't usually edit anything in this folder.

Every Android project needs a file called `R.java`, which is created for you and it lives in the `generated` folder. Android uses it to help it keep track of resources in the app.

The `src/` folder contains source code you write and edit.

The `java/` folder contains any Java code you write. Any activities you create live here.

You can find system resources in the `res/` folder. The `layout/` folder contains layouts, and the `values/` folder contains resource files for values such as strings. You can get other types of resources too.

Every Android app must include a file called `AndroidManifest.xml` at its root. The manifest file contains essential information about the app, such as what components it contains, required libraries, and other declarations.

`MainActivity.java` defines an activity. An activity tells Android how the app should interact with the user.

`activity_main.xml` defines a layout. A layout tells Android how your app should look.

`strings.xml` contains string id/value pairs. It includes strings such as the application name and any default text values. Other files such as layouts and activities can look up text values from here.
Every Android project needs a file called `R.java`, which is created for you and it lives in the generated folder. Android uses it to help keep track of resources in the app.

Java code for app. E.g. What happens on user input, etc

XML files for look or layout of Android screens

Every Android app must include a file called `AndroidManifest.xml` at its root. The manifest file contains essential information about the app, such as what components it contains, required libraries, and other declarations.

`strings.xml` contains string id/value pairs. It includes strings such as the application name and any default text values. Other files such as layouts and activities can look up text values from here.

You can find system resources in the `res/` folder. The `layout/` folder contains layouts, and the `values/` folder contains resource files for values such as strings. You can get other types of resources too.
Every Android project needs a file called `R.java`, which is created for you and it lives in the `generated` folder. Android uses it to help keep track of resources in the app.

The `build/` folder contains files that Android Studio creates for you. You don’t usually edit anything in this folder.

The root folder has the same name as your project.

The `src/` folder contains source code you write and edit.

The `java/` folder contains any Java code you write. Any activities you create live here.

You can find system resources in the `res/` folder. The `layout/` folder contains layouts, and the `values/` folder contains resource files for values such as strings. You can get other types of resources too.

Every Android app must include a file called `AndroidManifest.xml` at its root. The manifest file contains essential information about the app, such as what components it contains, required libraries, and other declarations.

MainActivity.java defines an activity. An activity tells Android how the app should interact with the user.

activity_main.xml defines a layout. A layout tells Android how your app should look.

strings.xml contains string id/value pairs. It includes strings such as the application name and any default text values. Other files such as layouts and activities can look up text values from here.

3 Main Files to Write Android app

Android Project File Structure
Files in an Android Project

- **res/** folder contains static resources you can embed in Android screen (e.g. pictures, string declarations, etc)

- **res/menu/**: XML files for menu specs

- **res/drawable-xyz/**: images (PNG, JPEG, etc) at various resolutions

- **res/raw**: general-purpose files (e.g. audio clips, CSV files)

- **res/values/**: strings, dimensions, etc
Concrete Example: Files in an Android Project

- **res/layout:** layout, dimensions (width, height) of screen cells are specified in XML file here

- **res/drawable-xyz/**: The images stored in jpg or other format here

- **java/**: App’s behavior when user clicks on a selection in java file here

- **AndroidManifest.XML:** Contains app name (Pinterest), list of app screens, etc
Basic Overview of an App

- Tutorial 8: Basic Overview of an App [11:36 mins]
  - https://www.youtube.com/watch?v=9l1lWAiHPg

- Main topics
  - Introduces main files of Android App
    - Activity_main.xml
    - MainActivity.java
    - AndroidManifest.xml
  - How to work with these files within Android Studio
  - Editing files using either drag-and-drop interface or XML
  - Flow of basic app
Editting Android

- Activity_my.xml is XML file specifying screen layout, widgets
- Can edit XML directly or drag and drop
Activity_main.xml

- **Widgets**: elements that can be dragged onto activity (screen)
- **Design View**: Design app screen using Drag-and-drop widgets
Activity_main.xml: Text View

- **Text view:** Design screen by editing XML file directly
- **Note:** dragging and dropping widgets auto-generates corresponding XML
MainActivity.java

- Java code, defines actions, handles interaction/put taken (intelligence)
  - E.g. What app will do when button/screen clicked
AndroidManifest.xml

- App’s starting point (a bit like main( ) in C)
Resources
Declaring Strings in Strings.xml

- Can declare all strings in strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">EmPubLite</string>
  <string name="hello_world">Hello world!</string>
</resources>
```

- Then reference in any of your app’s xml files

```xml
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="@string/hello_world"/>
</RelativeLayout>
```
Strings in AndroidManifest.xml

- Strings declared in strings.xml can be referenced by all other XML files (activity_my.xml, AndroidManifest.xml)

**String declaration in strings.xml**

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">EmPubLite</string>
  <string name="hello_world">Hello world!</string>
</resources>
```

**String usage in AndroidManifest.xml**

```xml
<application
    android:allowBackup="false"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">
    <activity
        android:name="EmPubLiteActivity"
        android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
</application>
</manifest>
```
Where is strings.xml in Android Studio?

Editing any string in strings.xml changes it wherever it is displayed.
**Styled Text**

- In HTML, tags can be used for italics, bold, etc
  - E.g. `<i>Hello</i>` makes text *Hello*
  - `<b>Hello</b>` makes text **Hello**

- Can use the same HTML tags to add style (italics, bold, etc) to Android strings

```xml
<resources>
  <string name="b">This has <b>bold</b> in it.</string>
  <string name="i">Whereas this has <i>italics</i>!</string>
</resources>
```
Phone Dimensions Used in Android UI

- Physical dimensions measured diagonally
  - E.g. Nexus 4 is 4.7 inches diagonally
- Resolution in pixels
  - E.g. Nexus 4 resolution 768 x 1280 pixels
- Pixels per inch (PPI) =
  - $\text{Sqrt}[(768 \times 768) + (1280 \times 1280)] / 4.7 = 318$
- Dots per inch (DPI) is number of pixels in a physical area
  - Low density (ldpi) = 120 dpi
  - Medium density (mdpi) = 160 dpi
  - High density (hdpi) = 240 dpi
  - Extra High Density (xhdpi) = 320 dpi
Adding Pictures

- Android supports images in PNG, JPEG and GIF formats
- Default directory for images (drawables) is `res/drawable-xyz`
- Images in `res/drawable-xyz` can be referenced by XML and java files
  - `res/drawable-ldpi`: low dpi images (~ 120 dpi of dots per inch)
  - `res/drawable-mdpi`: medium dpi images (~ 160 dpi)
  - `res/drawable-hdpi`: high dpi images (~ 240 dpi)
  - `res/drawable-xhdpi`: extra high dpi images (~ 320 dpi)
  - `res/drawable-xxhdpi`: extra extra high dpi images (~ 480 dpi)
  - `res/drawable-xxxhdpi`: high dpi images (~ 640 dpi)
- Images in these directories are **different resolutions, same size**
Adding Pictures

- Just the generic picture name is used (no format e.g. png)
  - No specification of what resolution to use
  - E.g. to reference an image `ic_launcher.png`

```xml
<application
    android:allowBackup="false"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">
```

- Android chooses which directory (e.g. –mdpi) at run-time based on actual device resolution
- Android studio tools for generating icons
  - **Icon wizard or Android asset studio**: generates icons in various densities from starter image
  - Cannot edit images (e.g. dimensions) with these tools
Styles

- Styles specify rules for look of Android screen
- Similar to Cascaded Style Sheets (CSS) in HTML
- E.g CSS enables setting look of certain types of tags.
  - E.g. font and size of all <h1> and <h2> elements
- Android widgets have properties
  - E.g. Foreground color = red
- **Styles in Android:** collection of values for properties
- Styles can be specified one by one or themes (e.g. Theme, Theme.holo and Theme.material) can be used
Default Themes

- Android chooses a default theme if you specify none
- Many stock themes to choose from

**Theme.Holo:** default theme in Android 3.0

**Theme.Material:** default theme in Android 5.0
Examples of Themes in Use

GMAIL in Holo Light

Settings screen in Holo Dark
References

- Busy Coder’s guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014
- Android App Development for Beginners videos by Bucky Roberts (thenewboston)