Week 2 Lecture

Activities, Lifecycles, Interactive UI

01

Android Framework Components

What are the building blocks of an Android app?

02

Reviewing Our New Project

What is generated when you create a new project?

03

Working With Activities

How does an Activity control our screen?

04

Creating Interactive User Interfaces

How to define and interact with UI elements?

05

Testing On A Real Device

How to run your app on a real Android device?

Android Framework Components

What are the building blocks on an Android app?

What is Android?

Multi-user, Linux-based operating system for mobile devices

    phones, tablets, Chromebooks, watches, tv, iot, auto

Apps run in their own Linux process

    each process has a unique virtual machine to keep isolated from other apps

A process is created any time a core Android component needs to be run

Core Android App Components

Activity

Represents what is on the screen
Entry point for user interaction

Service

Enables long running tasks to work in the background

Doesn't include any user interface

Broadcast Receiver

Allows the operating system to deliver event messages

Doesn't include any user interface

Content Provider

Manages app data shared with other apps

Doesn't include any user interface

Communicating Between Components

An Intent enables communication between Activities, Services, and Broadcast Receivers

Intents can specify a specific component, a general action, any extra required data, …

Android Manifest

An app's manifest registers app components with the operating system

AndroidManifest.xml

The manifest serves several other purposes:

- identifies user permissions
- declares specific required hardware features
- defines any url deeplinks that an app can handle

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >

        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>

        <service ...> </service>
        <receiver ...> </receiver>
        <provider ...> </provider>

        ...
    </application>
</manifest>
```

Android Apps

Include a combination of app components that receive events from the system, can communicate with one another in a secure fashion, and include some type of interactive user elements

Android Apps

Activities are responsible for what you see on the screen and what the user interacts with.

Activities

An Activity defines a layout which can include any number of layout containers and views; buttons, text, toggle, etc

Activities

An app can have any number of Activities and transition between Activities as needed.

Reviewing Our New Project

What is generated when you create a new project?

Reviewing Our New Project

- MainActivity
- activity_main.xml
- Review project structure
- AndroidManifest.xml
- strings.xml
- colors.xml
- styles.xml
- build.gradle

MainActivity & activity_main.xml

- MainActivity is what we are seeing on our screen
- activity_main.xml defines the layout
- setContentView(R.layout.activity_main) sets the layout to the Activity
- MainActivity is declared in AndroidManifest.xml

AndroidManifest.xml

- MainActivity is declared
- an Intent Filter is included to indicate which Activity to start on app launch

- app title, icon, theme are all defined in AndroidManifest.xml

Review Project Structure

- Android vs Project view
- Source sets

- Code vs Resource directories

- Res directory types

Android vs Project View

Both represent your project's files; source code, build files, resources

Project View displays exactly as files appear on file system

Android View consolidates and reorganizes to match how Android Studio thinks about the project

Source Sets

main - where your app code should go

test - where unit tests (not Android specific) go

androidTest - where Android instrumentation tests go

Code vs Resource Directories

java - all Java & Kotlin files are organized here

res - all Android resource files will go here

Resource Directory Types

drawable - images, colors, compound drawables

layout - layout files for Activities, Fragments, custom Views

values - other value types; strings, colors, dimensions, styles, etc

strings.xml

Define any user-visible strings as string resources

Supports localization & configuration changes

dimens.xml

Define any view/text dimensions

Supports different configuration values

colors.xml & styles.xml

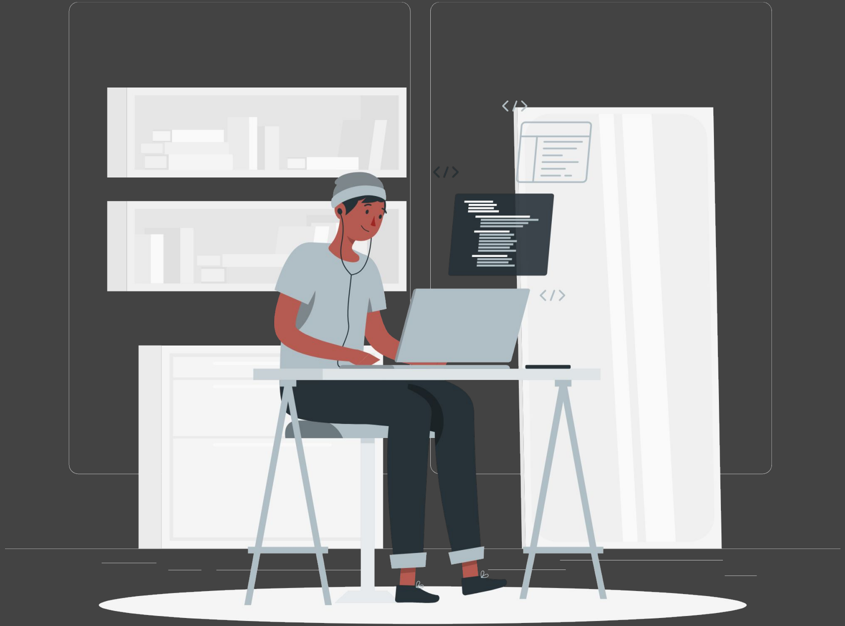
Define reusable colors and styles

Supports different locales and configurations by placing in xml

Build Files

- 2 build.gradle files present in our simple app

- root level controls configuration for entire project
- app level controls config and dependencies for our app module

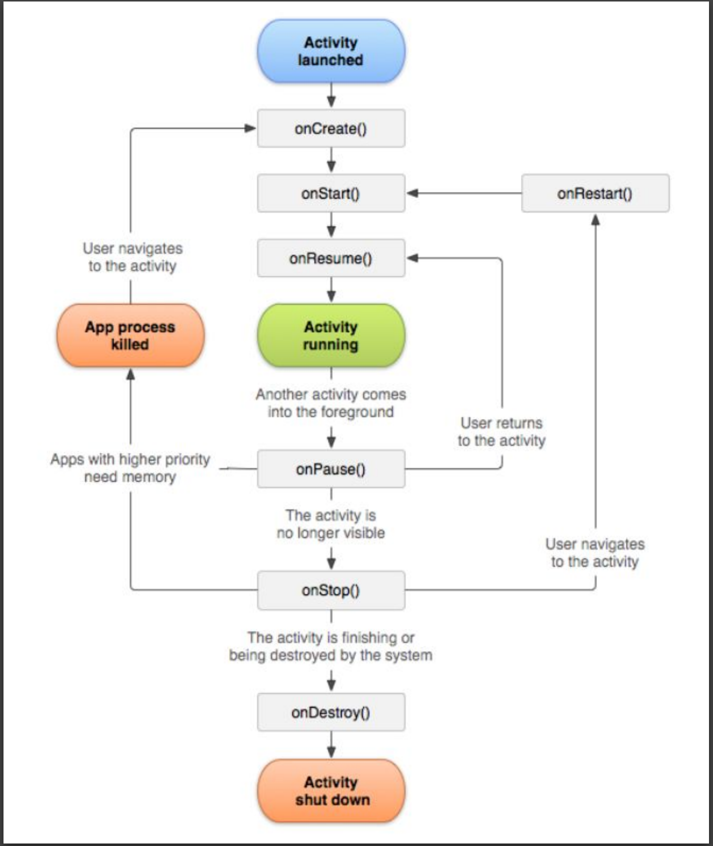


Working With Activities

How does an Activity control our screen?

Working With Activities

- how is an Activity defined?
- how do we specify the UI for an Activity?
- what is the Activity lifecycle?


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity Lifecycle

How an Activity is created and destroyed

Activity Lifecycle

onCreate()
Created but not yet active on the screen

onStart()
Activity is visible but not quite ready to receive focus or interaction

onResume()
Activity is visible and active in the app foreground

onPause()
Activity is visible, but something has taken foreground priority

onStop()
Activity is no longer visible on screen

onDestroy()
Activity is about to be destroyed because user navigated away for OS needs resources

Debugger

Helps you locate and diagnose problems in your application.

Allows you to suspend the running of your app at specific points, and then walk through the execution of your code step by step

Breakpoint

A line of code at which the debugger will suspend execution of your code

We will set a breakpoint in different Activity lifecycle methods

Creating Interactive User Interfaces

How to define and interact with UI elements?

UI Element Types

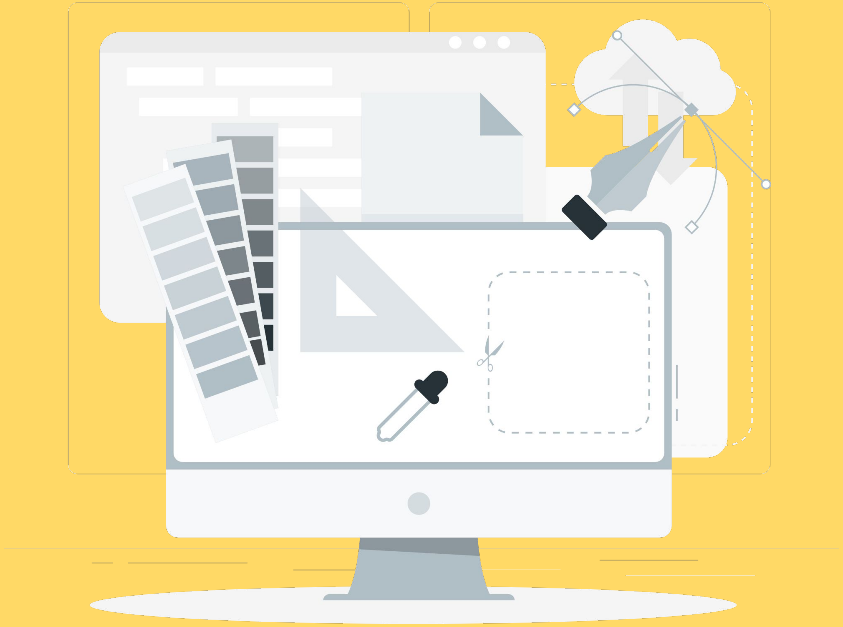
- ViewGroups
- Views
- Custom Views

ViewGroups

- FrameLayout
- LinearLayout
- RelativeLayout
- ConstraintLayout
- CoordinatorLayout

FrameLayout

No special ordering of views is applied. Child views are laid out starting from the top left. Great when a child view will fill the entire parent and no layout logic is needed.



LinearLayout

Child views are laid out one after the other in either vertical or horizontal orientation. Great for simple UI.

RelativeLayout

Child views are laid out relative to either the parent ViewGroup or other sibling views.

ConstraintLayout

Child views are laid out based on a complex series of equations. The layout design view is built to work best with ConstraintLayout. Great for flat view hierarchies, and complex animation.

Common Views

- TextView
- Button
- EditText
- ImageView

TextView

Displays text on the screen. Think of it as a simple label. Can be customized to display complex formatting if needed.

Button

A clickable view that can contain text and supports a click listener for responding to user interaction.

EditText

A view for entering & receiving user input. Can be customized for specific input types such as email, phone number, etc

ImageView

Displays some type of image or drawable. Can be loaded from an asset, a URL, or from a resource file.

Responding to user interaction

Adding a click listener and retreiving user input

Working With Value Resources

- text sizes using sp
- view size using dp
- colors

Text Size Using sp

SP stands for scale independent pixels. Defining text sizes using sp allows the operating system to update the size of text elements based on the scale settings of a device. This is done to support users with poor vision.

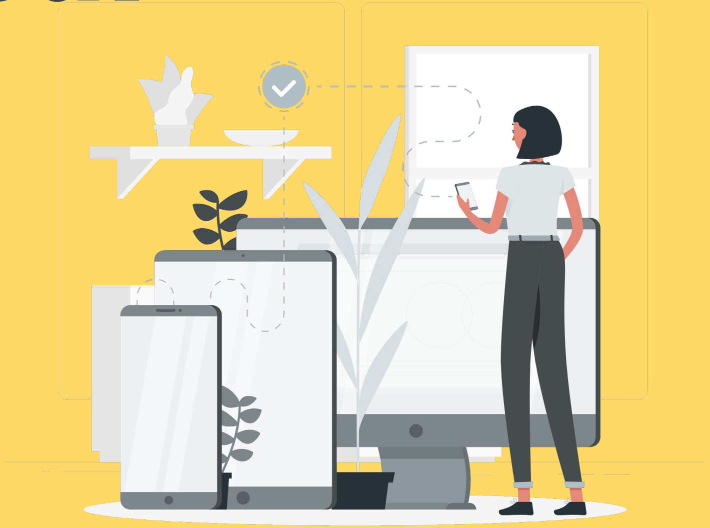
View Size Using dp

DP stands for density independent pixels. This means that a value defined using dp will be the same size whether on a low pixel density screen or a very high resolution screen. DP should be used when defining sizes for viewgroups or views.

Using colors

We can define reusable color resources in colors.xml. These resources can be used with views for things like background color. Color resources can also be used to define styles and themes in your app to customize the overall design aesthetic of your app.

Testing On A Real Device



Developer Options

Developer options provide specific settings and features to make developing and testing Android apps easier. Enable developer options and enable USB Debugging to enable the deployment of your apps from Android Studio to a real device.

CREDITS