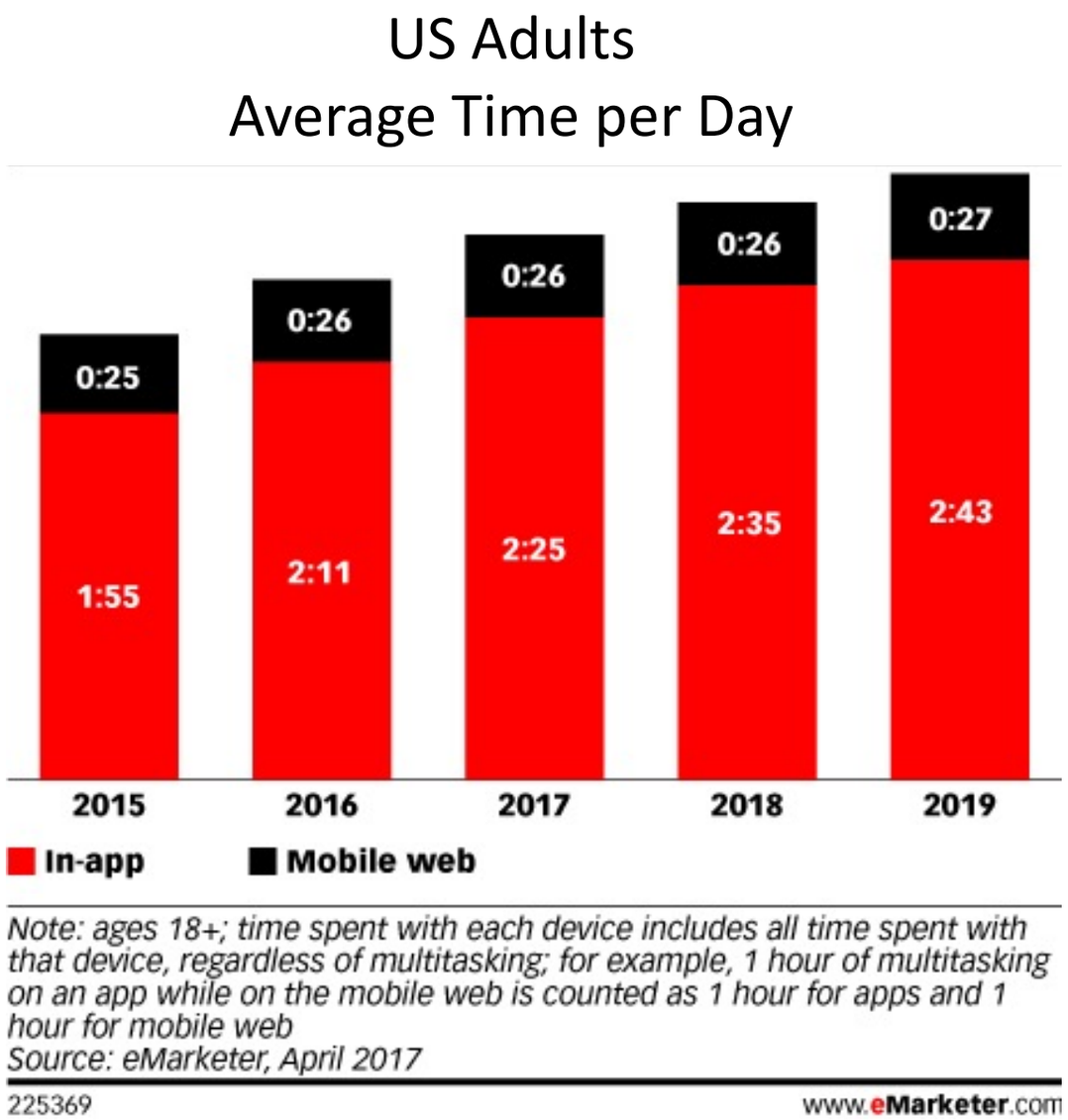
EEP 523:
MOBILE APPLICATIONS FOR SENSING AND CONTROL

SUMMER 2023
COURSE OVERVIEW

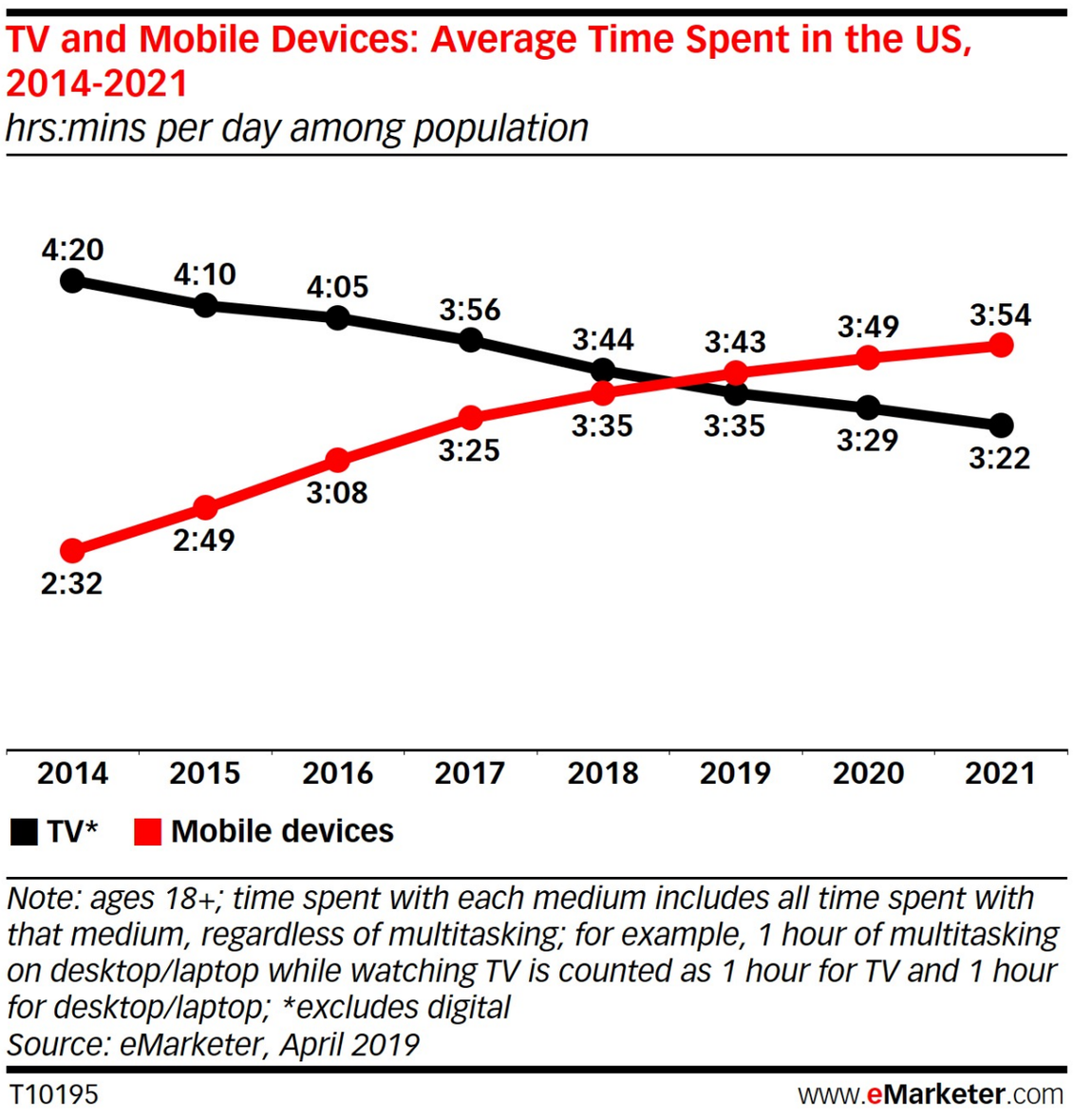
Tamara Bonaci

tbonaci@uw.edu

It’s an App World !!!



It’s an App World !!!



It’s an App World !!!

It’s an App World !!!

Pokemon Go trainers on the hunt at Green Lake in Seattle. (GeekWire Photo / John Cook). December 2018

Fill Bubble

It’s an App World !!!

Phantom-Vibration Syndrome

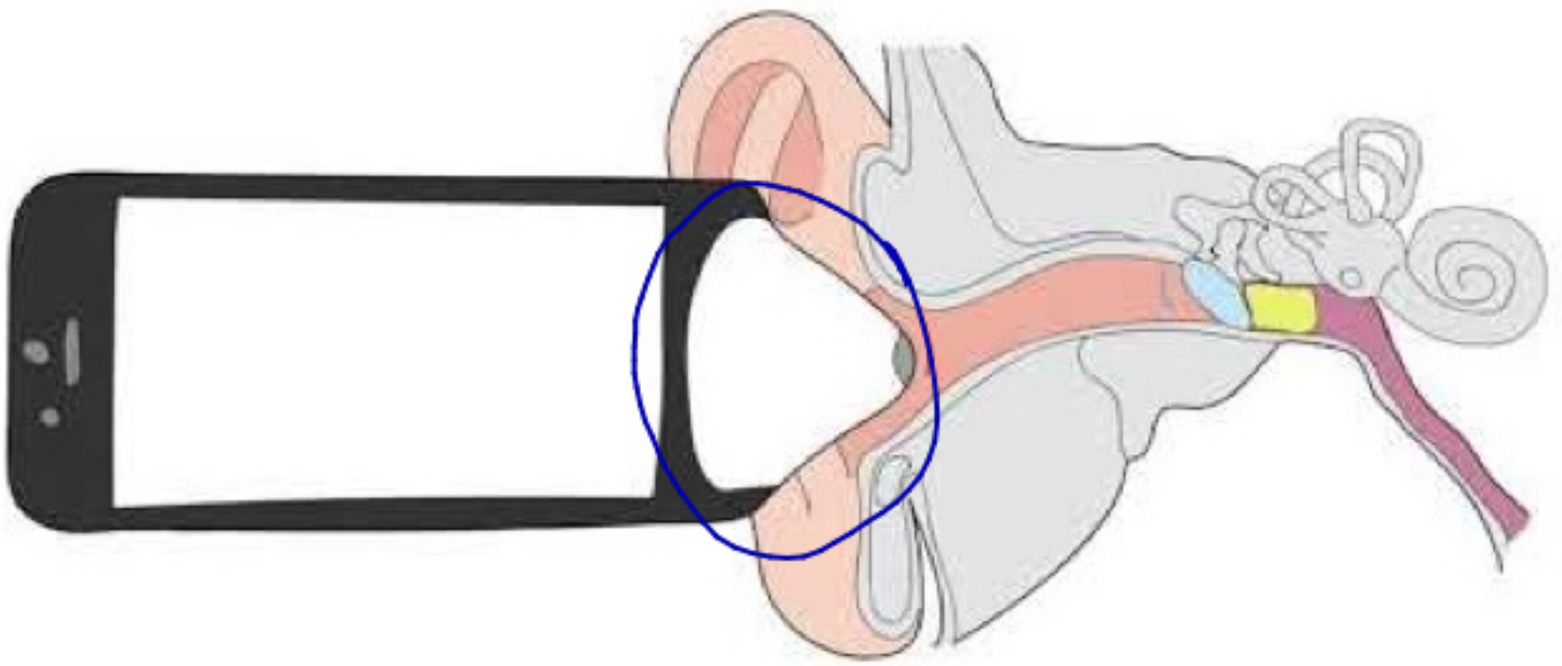
Study shows that 2/3 of users report

“feeling their phone vibrate when in fact nothing is happening”*

** Source: Newsweek, July 16, 2012*

Smartphones Beyond Recreation

App for ear infection detection



DIGITAL PHRENOLOGY

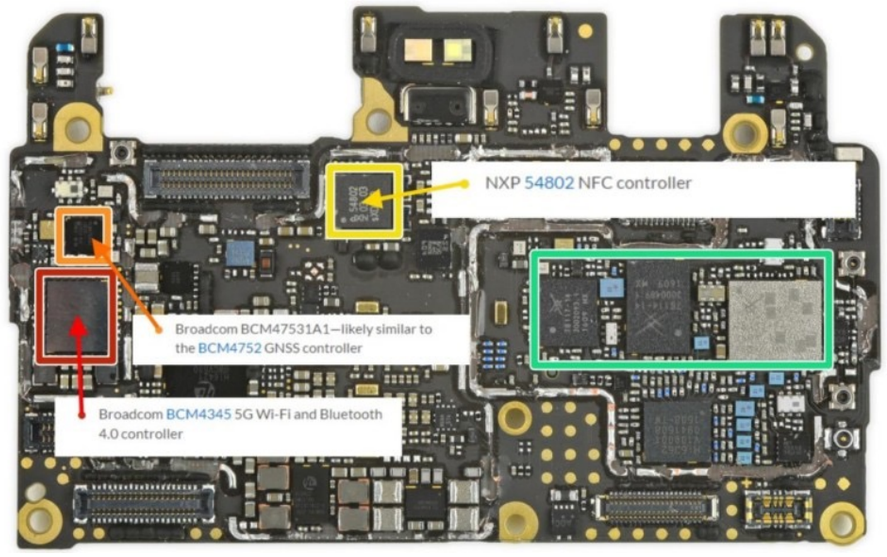
Smartphones Beyond Recreation

App to help improving ergonomics at work

What’s inside a smartphone?

Sensors: Before and After

Sensor A converter that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument …



Smartphone Sensors

- Sensors have been used in cellphones since they were invented … *Microphone, number keys, etc*
- **What made smartphones smart**? *Touchscreens, accelerometers, gyroscopes, GPS, cameras, etc …*
  Allowed cellphones explode into different markets R.I.P. Garmin, Tomtom, Kodak …. Intel?
- Instead of carrying around 10 separate devices, now you just need 1

Android supports:

Accelerometer, Ambient Temperature, Gravity, Gyroscope, Light, Linear Acceleration, Magnetic Field, Orientation, Pressure, Proximity, Relative Humidity, Rotation Vector

Lot's of Hardware, So What?

Now that you can collect all this data,
how do you use it?

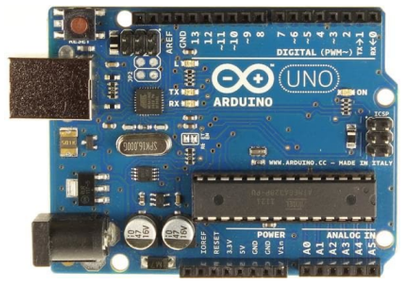
- Application Programming Interfaces (APIs) expose a smartphone's sensors and sensor data to the smartphone programmer (you!)
- The Android Accessory Development Kit (ADK) even provides the ability to add external sensors in a standard way

Smartphone Limitations

- Not all sensors are present in all phones
- What about *LEDs / NFC /ultrasound /moisture/ other*?
- What about *actuators, motors*?
- May want a remote sensor separate from your phone

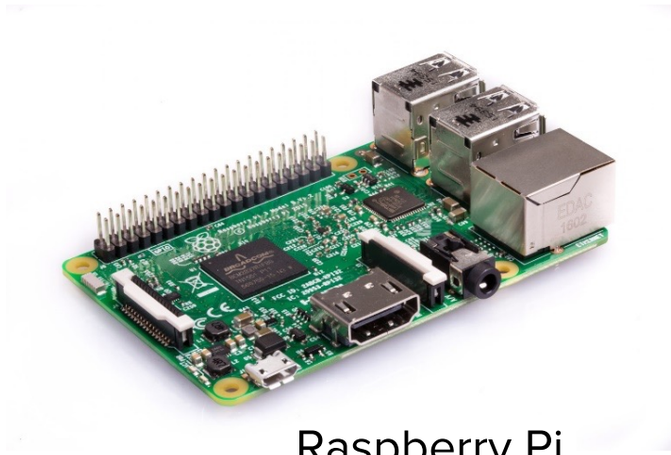
Using External Sensors with Android

- Single-board microcontroller
- Extend smartphone functionalities (wireless) : motors, sensors, actuators, etc

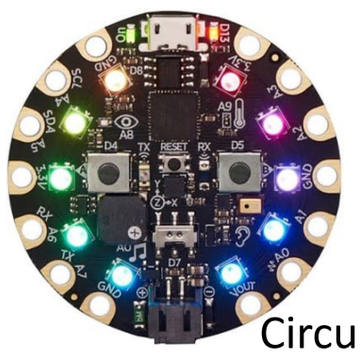


Arduino uno
$17

HUZZAH32 –
ESP32 Feather
$20

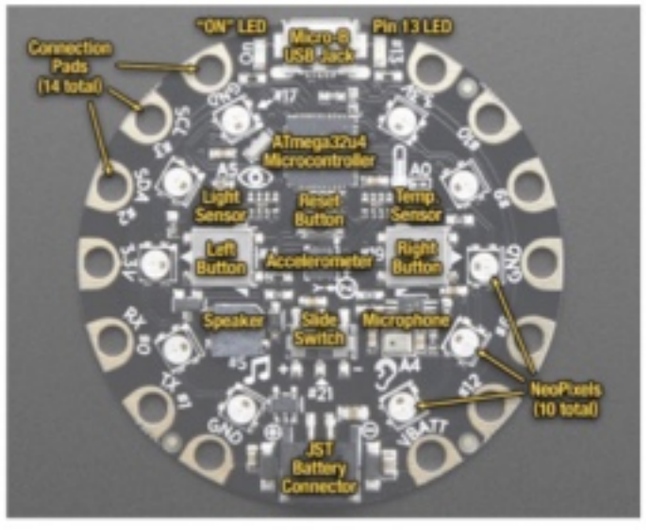


Raspberry Pi
$35



Circuit Playground
$25 + $18 BLE

Connect Your Phone to Arduino



Connect Your Phone to Arduino

Why Machine Learning ?

Why Machine Learning ?

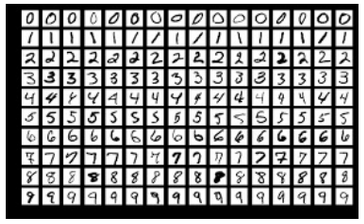
Instructor speech

DL generated speech

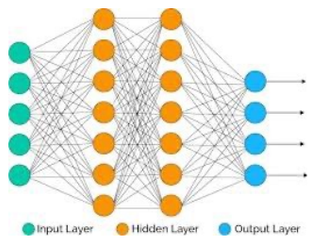
Why Deep Learning ?

- Classification/Regression

1. Data set (ground truth)



2. Train NN with true data



3. Generate a "Lite" version of the NN

model.tflite

- Generate new data

What is Android

EE P 523, Lecture 1

What is Android ?

- Mobile OS maintained by Google
- Android Inc. purchased by Google in 2005 for at least $50 million
- Runs on phones, tables, watches, TVs, etc
- Based on Kotlin (Java)
- #1 mobile OS worldwide
- > 1 million apps in Play Store
- Open-source code (easier to customize than iOS)

Google I/O

- Annual developer conference held by Google in Mountain View, California.

- "I/O" stands for input/output, as well as the slogan "Innovation in the Open".

- I/O 2019 https://events.google.com/io2019/schedule/events/

Why develop for Android ?

- Why not just write a web page?
  - Android has a browser…
- ✓ Better UI and user experience
- ✓ More direct access to device hardware (camera, gps, etc)

Users prefer apps over mobile web browsing

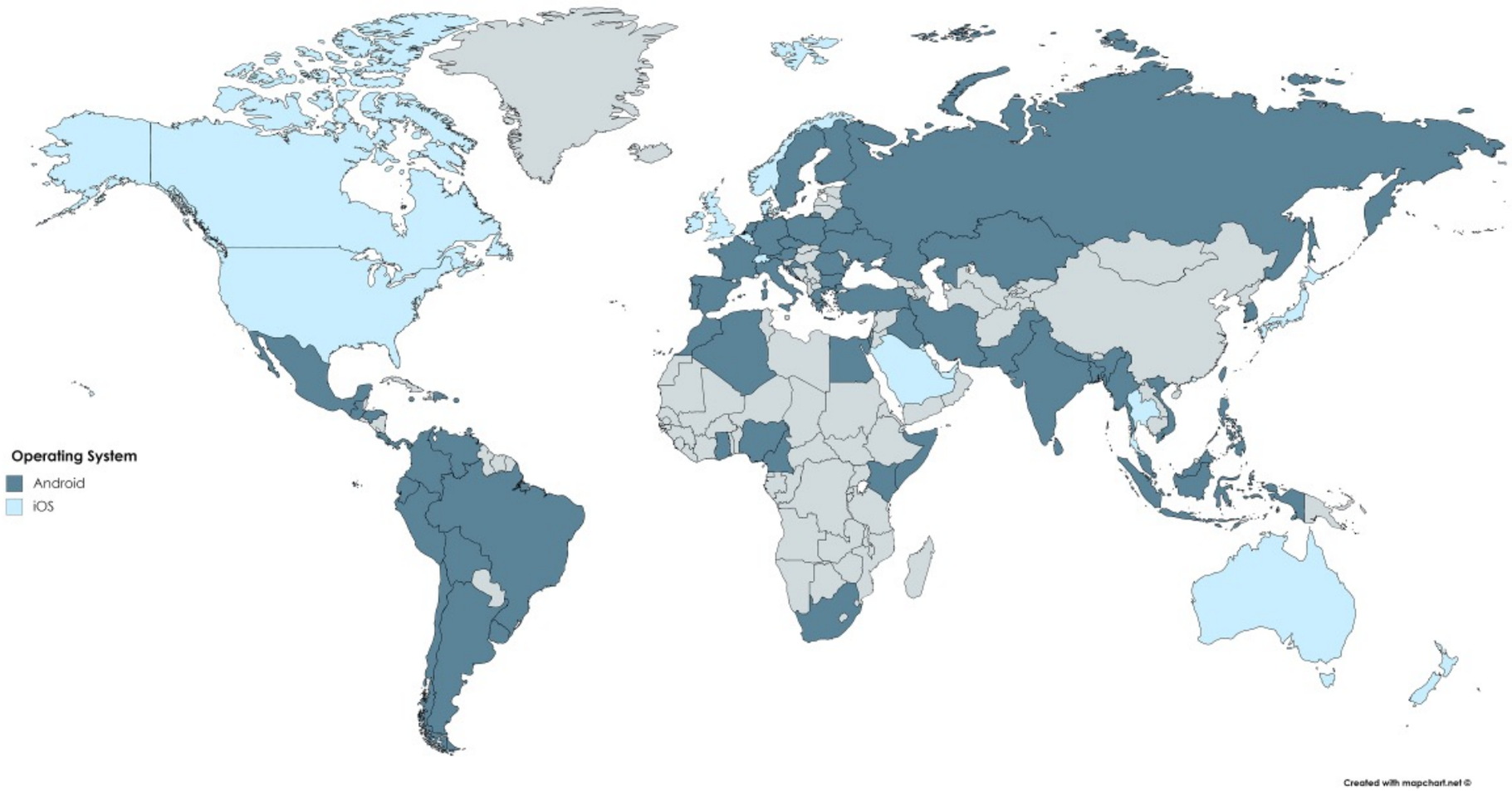
Why not iOS ?

Why not iOS ?

- Familiar language (Java/kotlin) vs *Swift/objective-C*
- Free development tools
- you can make an app and *easily* put on your phone or others
- Larger install base

Mobile Operating Systems

	Android	iOS	Windows 10 mobile	Tizen	Ubuntu Touch
Company	Google	Apple Inc.	Microsoft	Liunux Tizen Samsung Intel	USBports and Ubuntu Commu-nity
Market Share	86.2%	13.7%	0.1%	N/A	N/A
Current Release	Sept 2019	January 2020	Support ended on January 14, 2020	Novemb er 2018	October. 2019
Language	C, C++, Java, Kotlin	C, C++, Objective-C, Swift	.NET, Silverlight C/C++	C++	HTML5, QML, Go, JavaScript C++


Operating System
Android
iOS
Created with mapchart.net ©

Write an App Once and Run it Anywhere!

Hybrid Mobile App Frameworks: cross-platform Apps

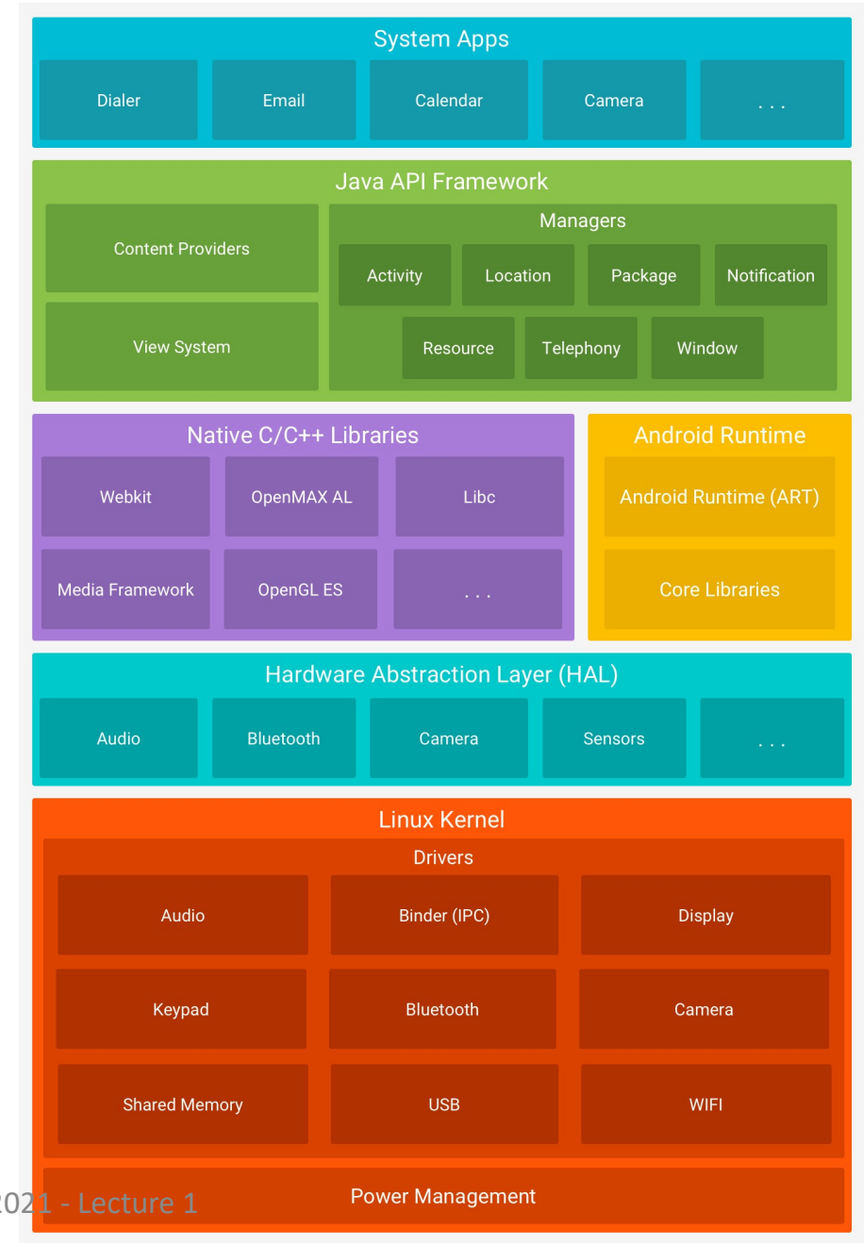
- React Native (JavaScript)
  - Facebook, Skype, Instagram, Uber etc
- Xamarin (.NET)
- Ionic (web technologies (HTML, CSS, and JavaScript)
  -Verizon, IBM, master card, etc.
- **Flutter (C, C++, Dart)**

  Dart:

  - object-oriented, class-based, garbage-collected language with C-style syntax

Android Architecture

- Linux-based software stack created for a wide array of devices and form factors
- Features of the devices used/accessed with libraries



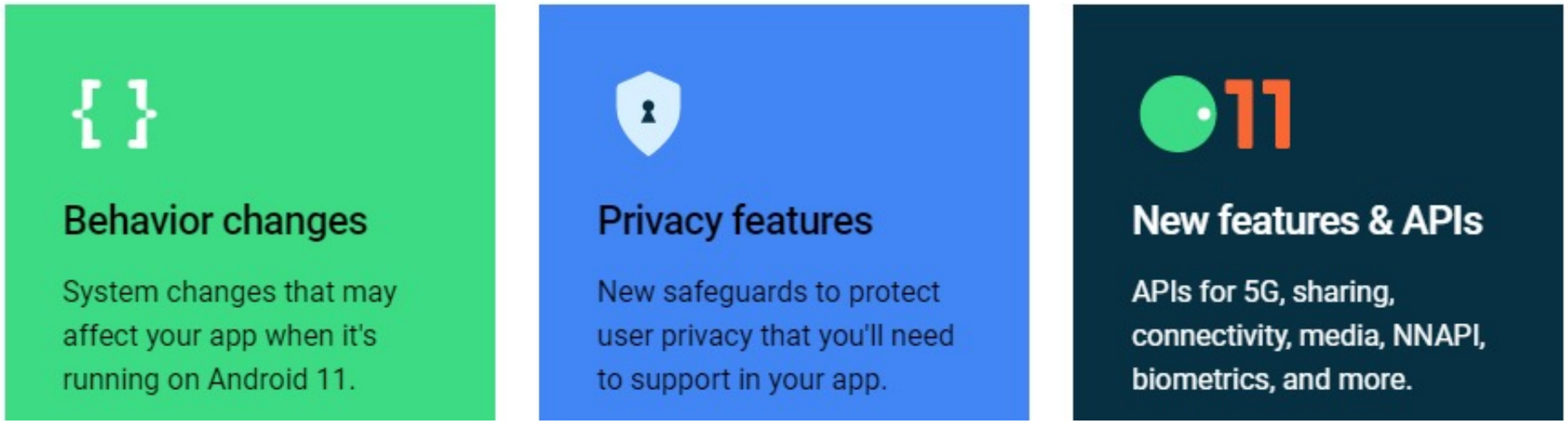
Android Version History

DATE	CODE NAME	API	VERSION NUMBER
September 2008	--	1	1.0
February 2009	Petit Four	2	1.1
April 2009	Cupcake	3	1.5
September 2009	Donut	4	1.6
October 2009	Éclair	5-7	2.0-2.1
May 2010	Froyo	8	2.2-2.2.3
December 2010	Gingergread	9-10	2.3-2.3.7
February 2011	HoneyComb	11-13	3.0-3.2.6
October 2011	Ice Cream Sandwich	14-15	4.0-4.0.4
July 2012	Jelly Bean	16-18	4.1-4.3.1
October 2013	KitKat	10-20	4.4-4.4.4
November 2014	Lollipop	21-22	5.0-5.1.1
October 2015	Marshmallow	23	6.0-6.0.1
August 2016	Nougat	24-25	7.0-7.1.2
August 2017	Oreo	26-27	8.0 – 8.1
August 2018	Pie	28	9.0
September 2019	Android 10	29	10
September 2020	Android 11	30	--

Moto G 3rd Gen

Xiaomi Redmi Note 4

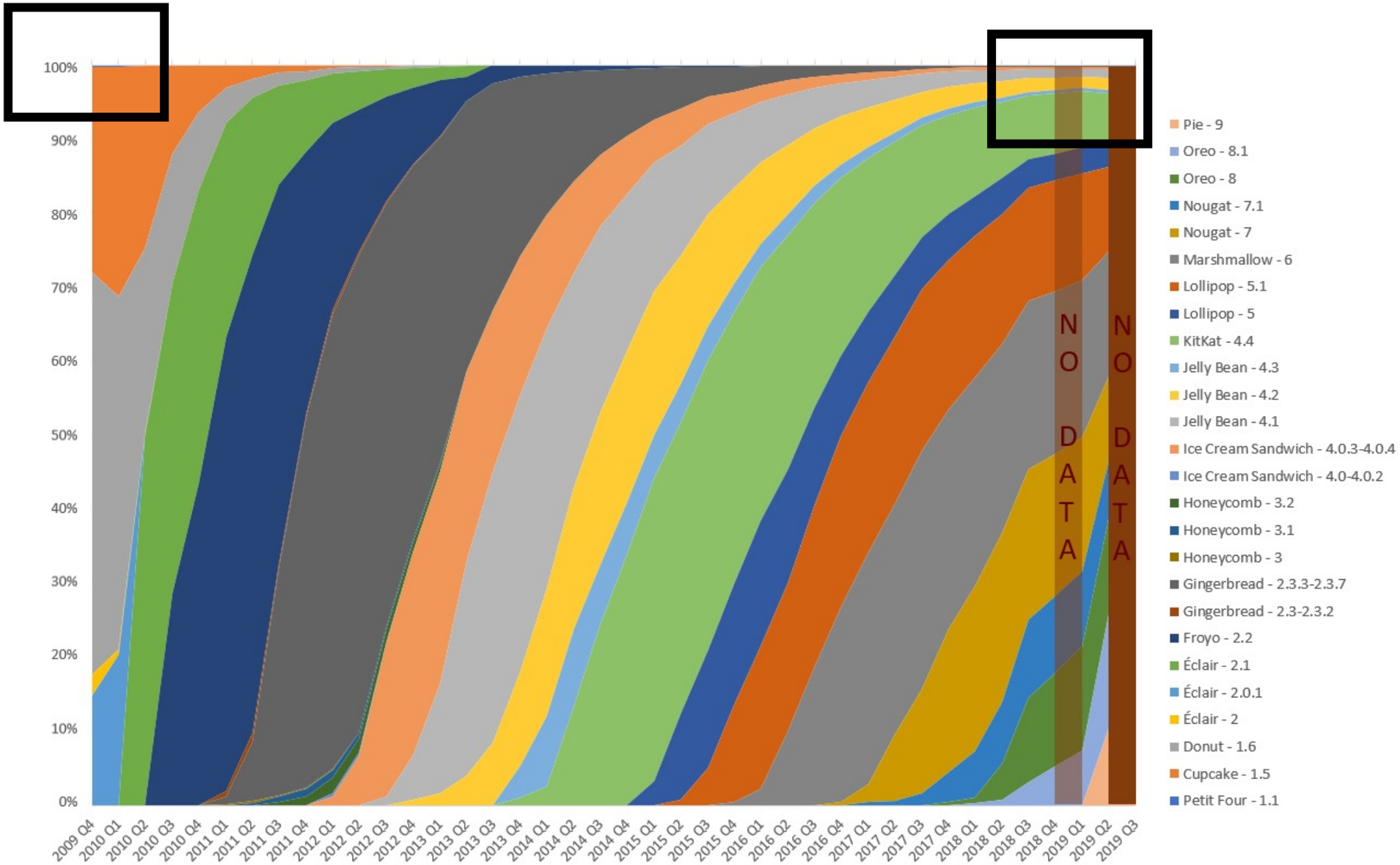
Introducing Android 11



"**Nearby sharing**" -> AirDrop-style feature (iPhone)

- Not tied to Android 11
- Sharing content among android devices
- Uses Wi-Fi (faster than BLE)
- *AndroidBeam* (predecessor) deprecated –> NFC-based

Android Versions History

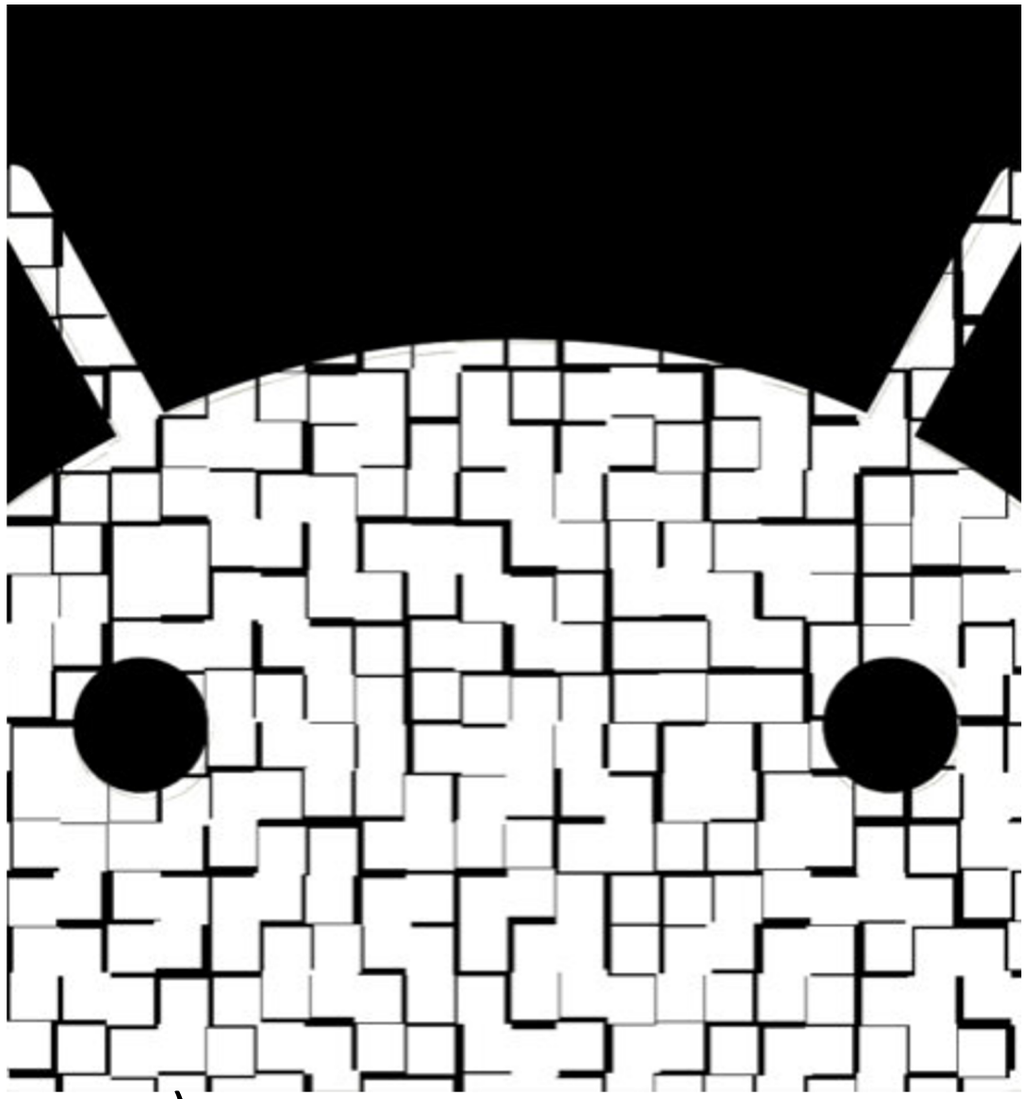


Version Issues

- Why doesn't my phone has the latest Android Version?

  Can't I just update it ?

- There are several companies involved in the process, with different interests
  - OS (Google)
  - Phone manufacturer (Samsung)
  - Service provider (at&t)



Introduction to Kotlin

EE P 523, Lecture 1

Time to Code Our First App !

Android apps can be written using

Kotlin, ***Java***, and ***C++*** languages.

- Java: legacy code
- **Kotlin**: first-class support by Google (since 2017)
- C++: digital signal processing (faster than java/kotlin)

Kotlin

- Developed by JetBrains in 2011

- First-class language for Android Apps
  ->Announced at Google I/O 2017

- Advantages:
  concise syntax, modern features, and seamless interoperability with legacy Java code.

Java vs Kotlin

- 60% of the top 1000 Android apps use Kotlin
  (Posted by David Winer, Kotlin Product Manager -12 March 2020)

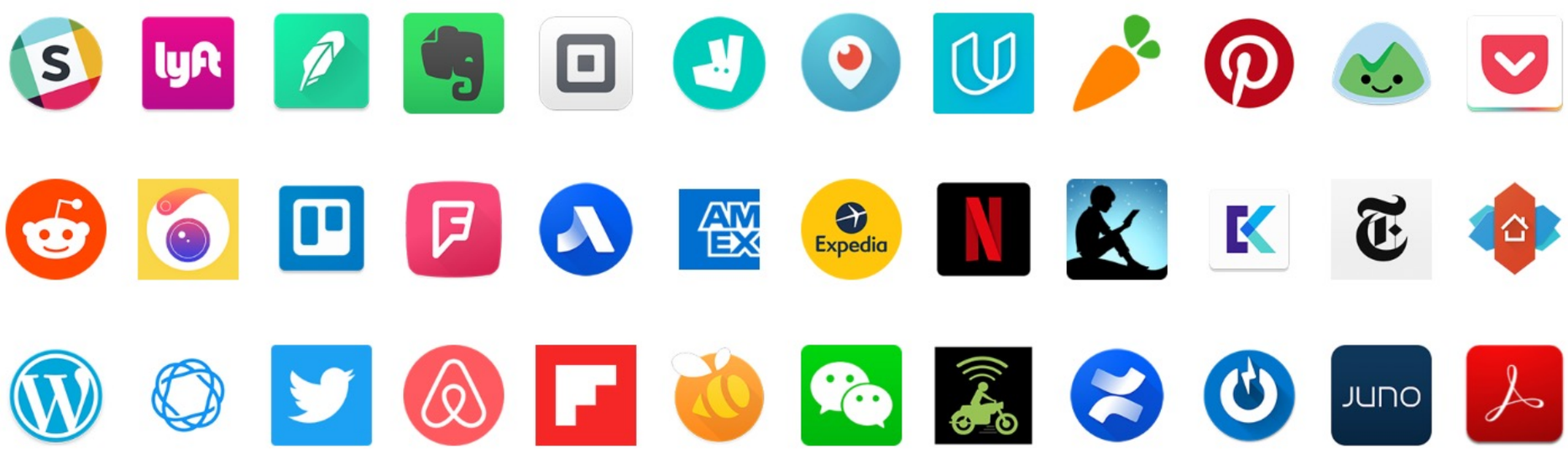
- The Android framework team has started adding @nullable annotations to legacy platform code.

- They have also released more and more Kotlin extensions for Android. And Google is in the process of adding Kotlin examples and support to the official Android documentation.

- The Android framework was originally written in Java. This means most of the Android classes you interact with are Java. Luckily, **Kotlin is interoperable with Java**

What Does Koltin Look Like ?

Kotlin in Action!



Variables

Variables in Kotlin, as in Java, allow us to assign values that can then be modified and used at different points in our program, as long as they are within the scope in which the code is executed.

JAVA	KOTLIN
Integer variableA = new Integer(5); Need to specify type: Int, float, double, String int variableA	Use ***var***->type is inferred var
// Constants final double d = 1.0;	// Constants (read only) Use ***val***->type is inferred val

Variables

JAVA

```
int i = 0;
i += 1;

// Constants
final double d = 1.0;
```

KOTLIN

```
var x = 5 // Int type is inferred
x += 1

// Constants
val i: Int = 10
val i = 1 //type Int inferred

val c: Int //Type required
           if no initializer
c = 10 //deferred assignment
```

Variables: type conversions

JAVA

```
int num = (int) k;
```

casting

note to a: compiler to treat k as an int

KOTLIN

```
var num = k.toInt()
```

FUNctions

JAVA

return type

```
public int sum (int a,int b){
   return a + b;
}
```

Returns nothing

```
public void logSum (int a,int b){
      Log.d("TAG","sum of "+a+"+"+b+" is "+ (a+b));

   }
```

KOTLIN

return type

```
fun sum(a: Int, b: Int): Int {
    return a + b
}
```

Same function!

```
fun sum(a: Int, b: Int) = a + b
```

If you don't care about the return value, you don't need to assign it to anything.

Can be omitted

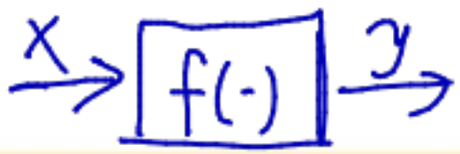
```
fun logSum(a: Int, b: Int): Unit {
    Log.d("sum of $a and $b is ${a + b}")
}
```

Functions: default arguments

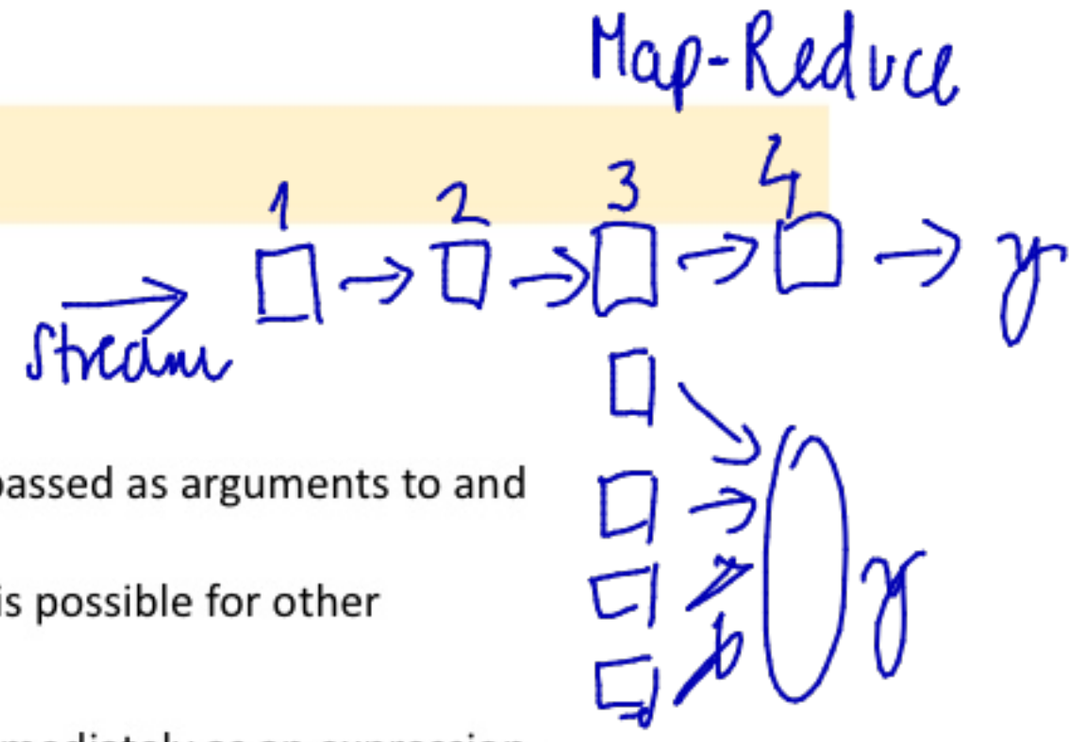
KOTLIN

```
fun sum(a: Int = 1, b: Int = 2) = a + b
```

```
fun sum(a: Int = 0, b: Int) { ... }
sum(b = 1)  // The default value a = 0 is used
```



Lambdas (intro)



KOTLIN

Kotlin functions are *first-class*:

can be stored in variables and data structures, passed as arguments to and returned from other higher-order functions.

You can operate with functions in any way that is possible for other non-function values

Lambda: Function that is not declared, but passed immediately as an expression.

Defined with **{ }**

Example 1

```
max(strings, { a, b -> a.length < b.length })
```

lambda

```
fun compare(a: String, b: String): Boolean = a.length < b.length
```

Example 2

```
val sum = { x: Int, y: Int -> x + y }
```

equivalent

```
val sum: (Int, Int) -> Int = { x, y -> x + y }
```

The body goes after an **->** sign.
If the inferred return type of the lambda is not Unit, the last (or possibly single) expression inside the lambda body is treated as the return value.

Loops

while
for loop
foreach loop

JAVA

```
for(int i = 1; i<20; i++) {…}
for(int j = 1; j<20; j+=2) {…}
for(String s: collection) {…}
```

} for
} for each

data collection

KOTLIN

```
repeat(10) {…}
for(i in 1..20) {…}
for(j in 1..20  step 2) {…}
for(s in collection) {…}
```

List - ArrayList
Map - HashMap
Set - TreeSet, HashSet

Lists

JAVA

```
int[] numList = {0,5,10,15};
ArrayList<String> gear = new ArrayList<>();
gear.add("carabiner");
gear.add("rope");

int courseGrade = numList[0];
String item = gear.get(1);
if (gear.contains("atc")) {…}
```

diamond operator

int[]
Integer[] intArray = new Integer[5];

generics

KOTLIN

```
val numList: List<Int> = listOf(0,5,10,15) // cannot be modified!
val gear = mutableListOf("carabiner", "rope").
words.add("atc")

val courseGrade = numList[0]
val item = gear[1]
if ("atc" in gear) {…}
```

Introduction to Android Studio

EE P 523, Lecture 1

Android Studio

- Official IDE for Android development, and includes everything you need to build Android apps
- Replaces previous Eclipse environment
- Based on IntelliJ IDEA editor (free download and use)

1. Install Java JDK
2. Install android studio Download

JDK ≠ JRE

14

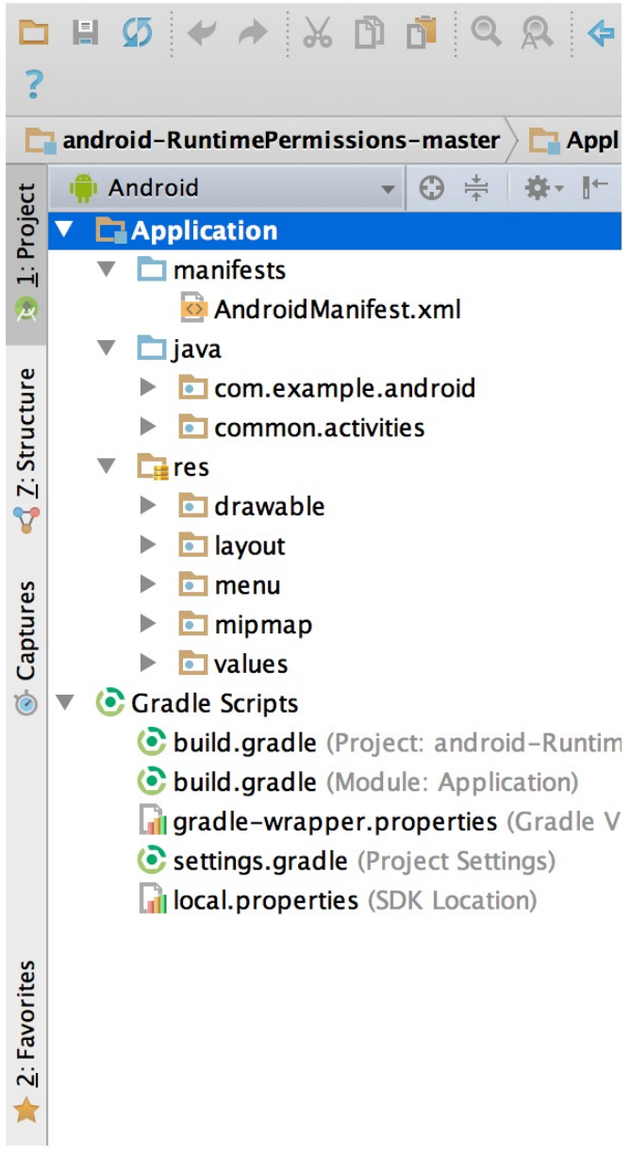
JDK after Java 8

Oracle

Project Structure

Each project in Android Studio contains one or more modules with source code files and resource files.
Types include:

- Android app modules
  - Phone and Tablet
  - Wear OS
  - Android TV
  - Glass
- Library modules
- Google App Engine modules



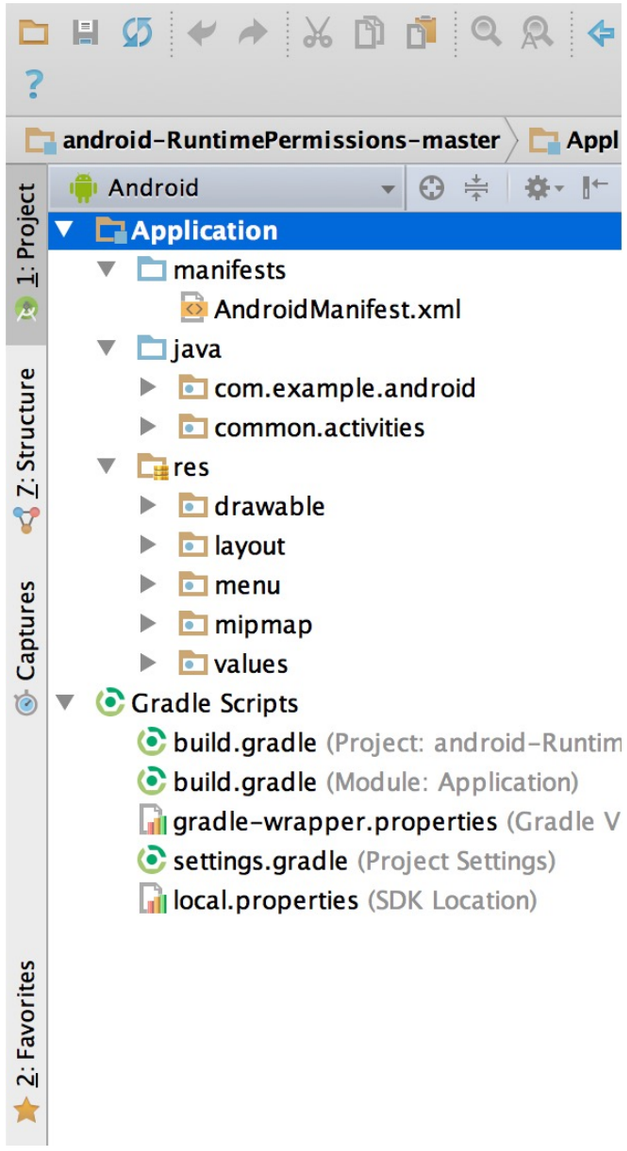
Project Structure

Each app module contains the following folders:

- **manifests**: Contains the AndroidManifest.xml file.

- **java**: Contains the Java source code files, including JUnit test code.

- **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.



Manifest File

- Every app project must have an **AndroidManifest.xml** file
  (with precisely that name)

- Describes essential information about your app to the Android build tools, the Android operating system, and Google Play.

- If you're using Android Studio to build your app, the manifest file is created for you, and most of the essential manifest elements are added as you build your app
  (especially when using code templates).

Manifest File

Package name and application ID

usually matching your project directory structure.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp"
    android:versionCode="1"
    android:versionName="1.0" >
    ...
</manifest>
```

App components

Activity, service, receiver, provider.

```
<manifest package="com.example.myapp" ... >
    <application ... >
        <activity android:name=".MainActivity" ... >
            ...
        </activity>
    </application>
</manifest>
```

Permissions

Android apps must request permission to access sensitive user data (such as contacts and SMS) or certain system features (such as the camera and internet access). Each permission is identified by a unique label.

```
<manifest ... >
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    ...
</manifest>
```

Device compatibility

what types of hardware or software features your app requires, and thus, which types of devices your app is compatible with.

```
<manifest ... >
    <uses-feature android:name="android.hardware.sensor.compass"
                  android:required="true" />
    ...
</manifest>
```

Build Process

The Android build system

compiles app resources and source code,

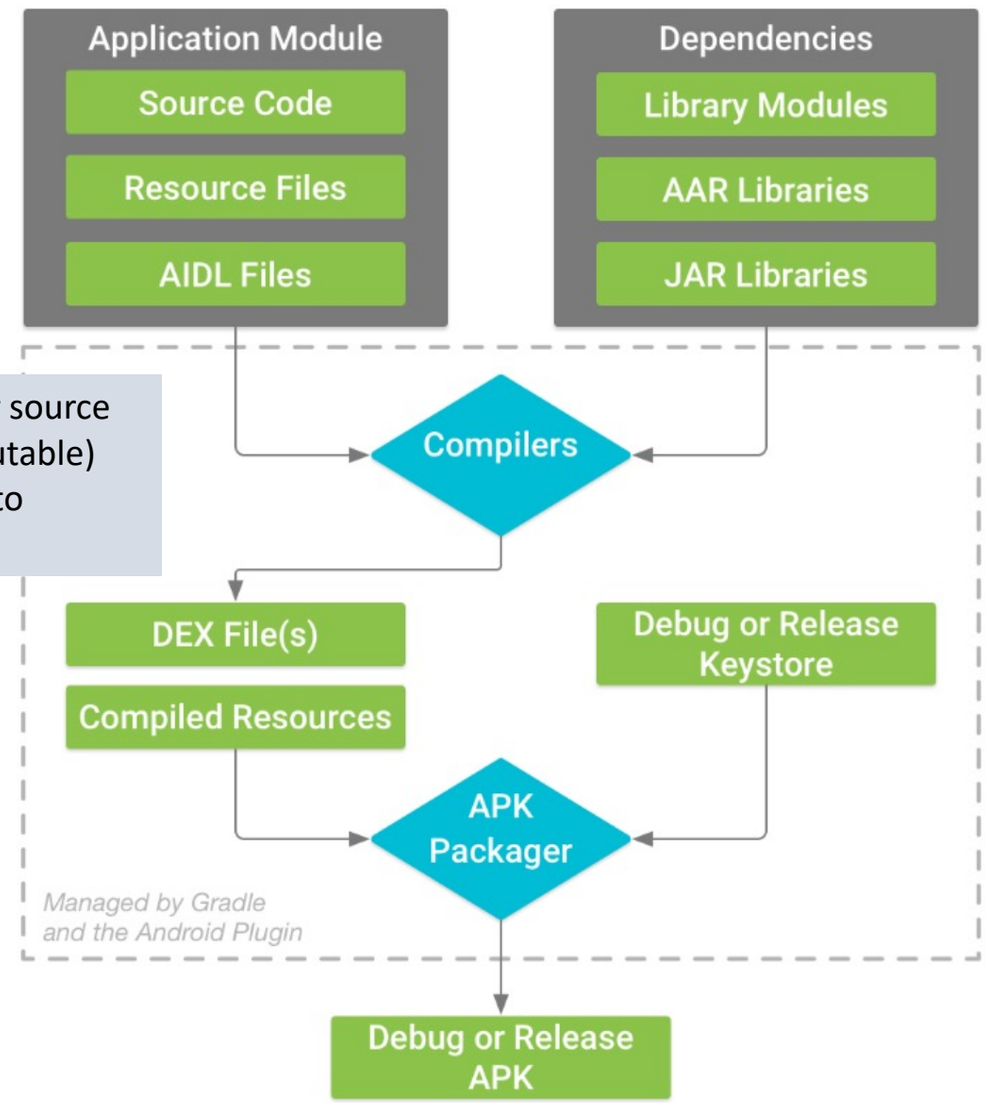
and packages them into **APKs**

(Android Application Package)

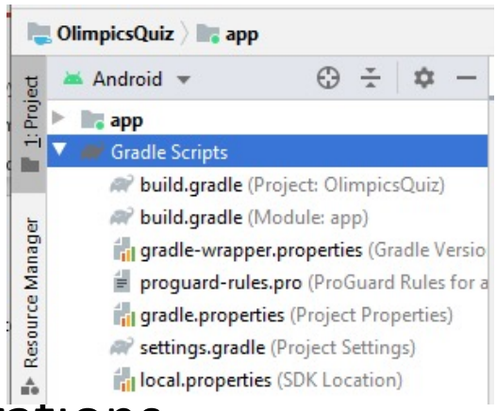
that you can test, deploy, sign, and distribute.

Build Process

The compilers convert your source code into DEX (Dalvik Executable) files and everything else into compiles resources



Gradle Plugin



Android Studio uses **Gradle toolkit**:

- automate and manage the build process,
- allowing you to define flexible custom build configurations.

Gradle and the Android plugin run independent of Android Studio.

You can build your Android apps from

- Within Android Studio
- The command line on your machine
- On machines where Android Studio is not installed (such as continuous integration servers)

App Security Sandbox

Each Android app lives in its own security sandbox

- Android operating system
  - multi-user Linux system
  - each app is a different user.
  - each app runs its own Linux process

- **Each app assigned unique Linux user ID**
  (the ID is used only by the system and is unknown to the app).

- The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.

- Each process has its own virtual machine (VM) -> an app's code runs in isolation from other apps.

Principle of Least Privilege

Each app, by default, has access only to the components that it requires to do its work and no more

Sharing data between aps and access system services:

- 2 apps can share the same Linux user ID - > access each other's files.
- Apps with = user ID: run in the same Linux process, share the same VM.
- Request permission to access device data
  (user's contacts, SMS messages, the mountable storage (SD card), camera, and Bluetooth)
  The user has to explicitly grant these permissions.

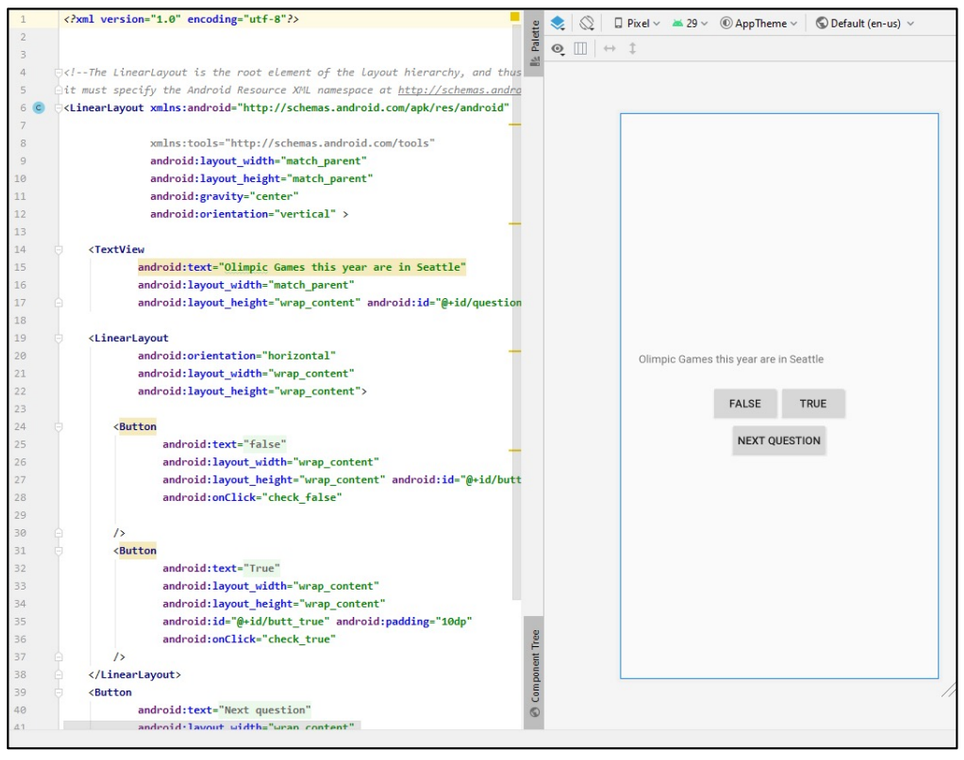
App Basics

The most basic App consists of

an ***activity*** and a ***layout***

```
class MainActivity : AppCompatActivity() {

    private var mCurrentIndex = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        mCurrentIndex = 1
    }

    private val questionBank = listOf(
        Question("This year the Olympic Games are in Seattle", answer: false),
        Question("First modern Olympic games were in 1796", answer: false),
        Question("This year games there will a total of 40 sports", answer: false),
        Question("The Olympic flag contains 5 colors", answer: true),
        Question("Women were allowed to participate in the Olimpics in 1900", answer: true),
        Question("Skateboarding is an Olympic sport", answer: true),
        Question("Gold medals are made of 50% gold", answer: false),
        Question("Michael Phelps is the athlete with the most Olympic medals", answer: true),
        Question("Men and women do NOT compete against each other in any ...", answer: true),
        Question("Swimming obstacle race was Olympic in one Game", answer: false)
    )


    fun updateQuestion(view: View){
        mCurrentIndex ++
        if (mCurrentIndex > questionBank.size) {
            mCurrentIndex = 1
        }
        val questionTextResID = questionBank[mCurrentIndex].textResId
        questionView.setText(questionTextResID)
    }
```

```
<?xml version="1.0" encoding="utf-8"?>


<!--The LinearLayout is the root element of the layout hierarchy, and thus
it must specify the Android Resource XML namespace at http://schemas.andro
<LinearLayout 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"
              android:gravity="center"
              android:orientation="vertical" >

    <TextView
            android:text="Olimpic Games this year are in Seattle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" android:id="@+id/question

    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">

        <Button
                android:text="false"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:id="@+id/butt
                android:onClick="check_false"

        />
        <Button
                android:text="True"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/butt_true" android:padding="10dp"
                android:onClick="check_true"
        />
    </LinearLayout>
    <Button
            android:text="Next question"
            android:layout_width="wrap_content"
```



App Basics

The most basic App consists of an ***activity*** and a ***layout***

activity: instance of class **`Activity`**.

Manage the user interaction with a screen of information

You write subclasses of **`Activity`** in your app .

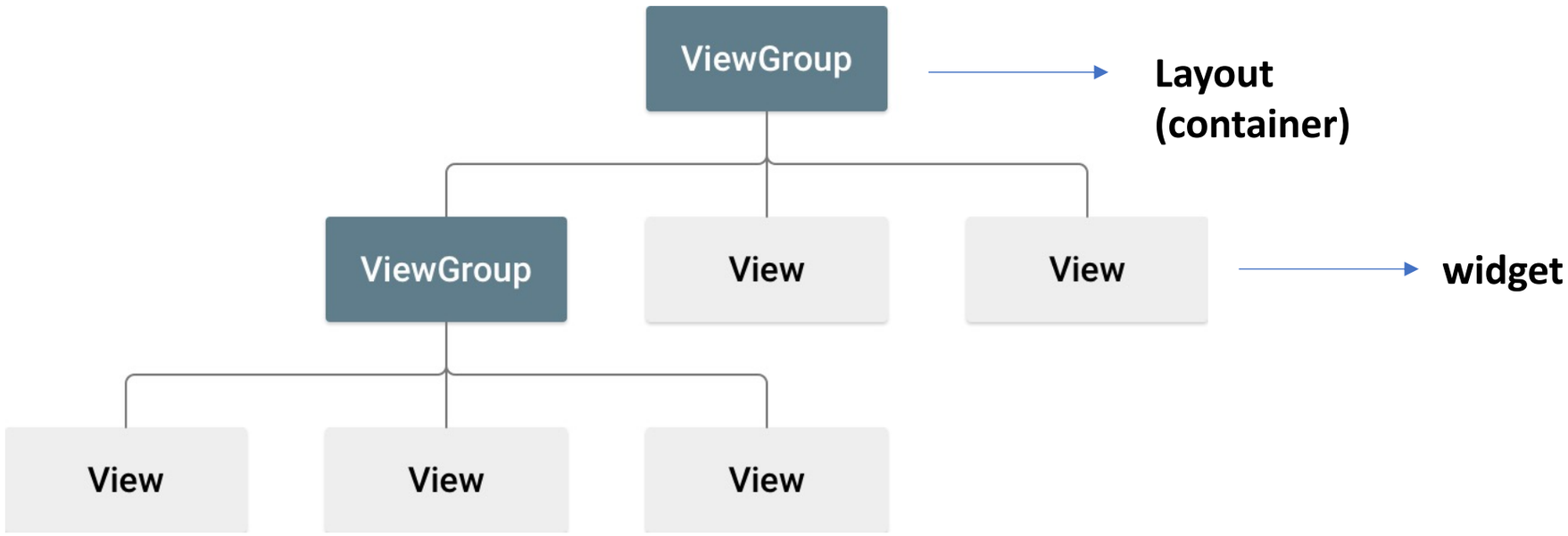
Your App can have one or multiple activities

layout: defines a set of UI objects and the object's positions on the scree.

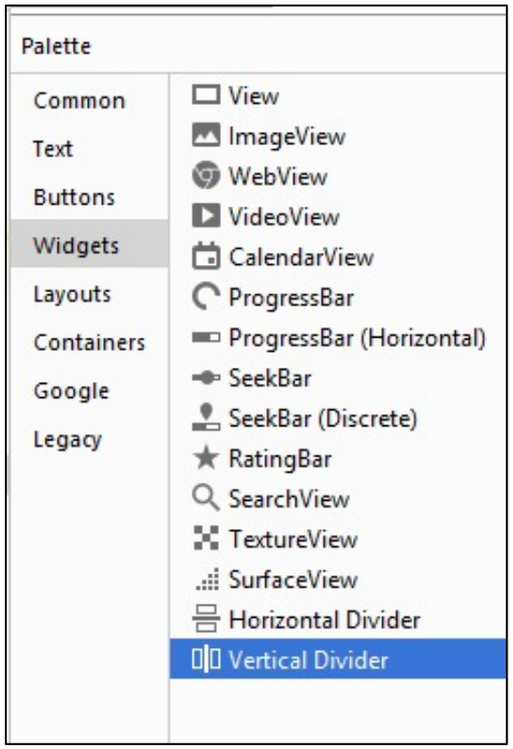
Is made up of definitions written in XML.

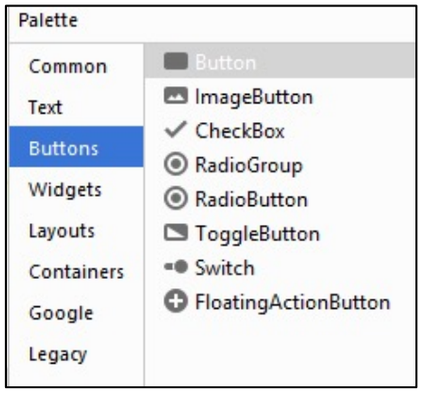
Each definition is used to create an object that appears onscreen like a button or some text

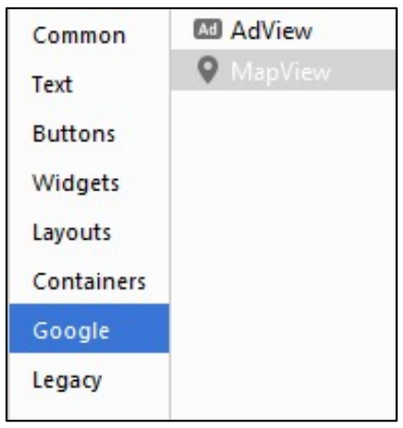
Android Graphical User Interface (GUI)



Android Widgets







The widget package contains (**mostly visual**) UI elements to use on your Application screen.

You can also **design your own** !

Basic App GUI

1. Define a Layout (**Constraint layout** by default)

2. Add widgets to the Layout and configure them
   a. Drag and drop on editor
   b. Edit **XML** file */app/res/layout/activity_main.xml*

3. Add events to capture user input (press a button)

4. Handle events

Extensible Markup Language (XML)

- Set of rules for encoding documents such as is readable both by human and machines

- Tags are not predefined. We must define our own tags.

```
<!-- Base application theme. -->
<style name="AppTheme"
parent="Base.Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

- XML tags can be nested

Extensible Markup Language (XML)

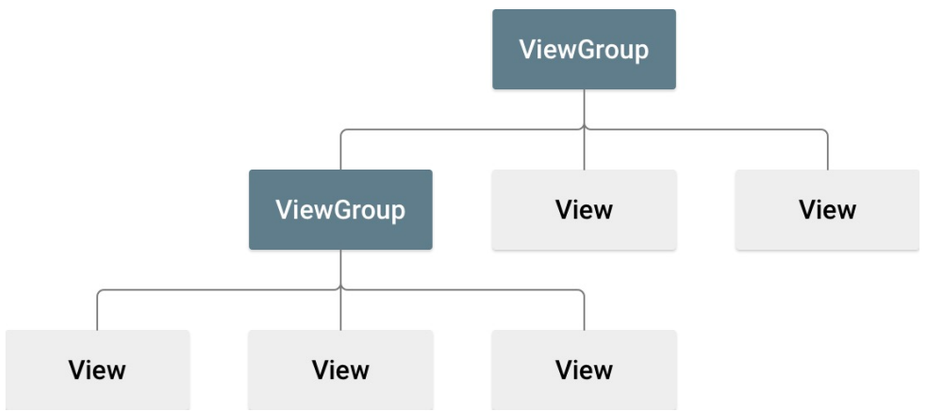
- Uses **tags (not predefined)** to carry data (elements with attributes)

```
<resources>
    <string name="app_name">OlimpicQuizs</string>
</resources>
```

```
<resources>
    <color name="colorPrimary">#3F51B5</color>
</resources>
```

Write the Layout XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```



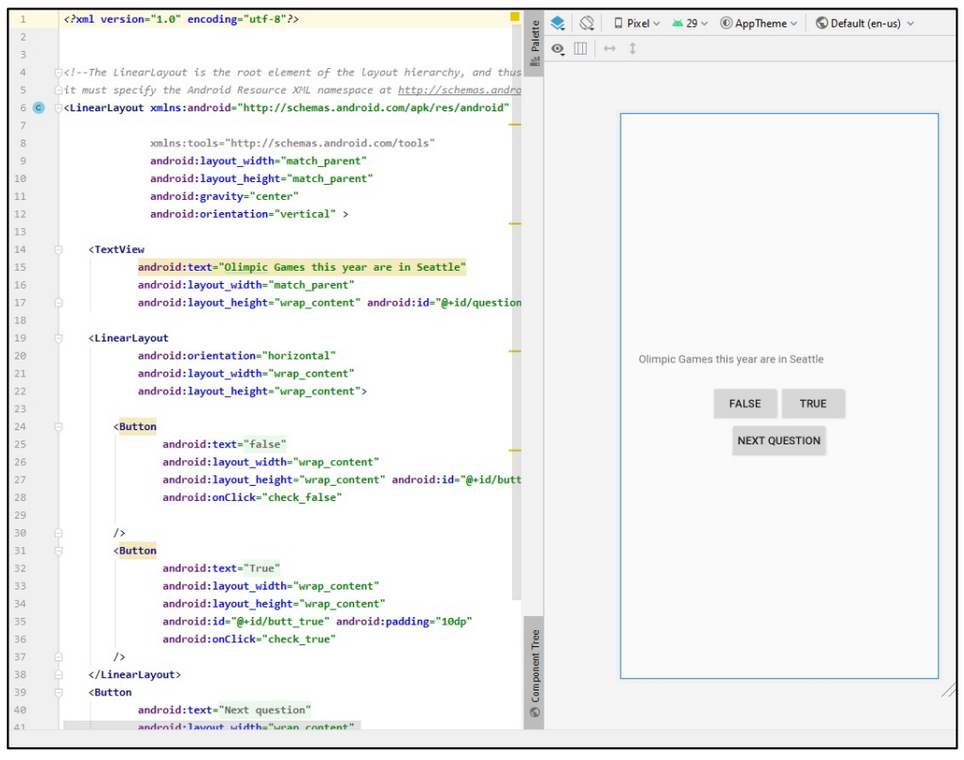
Recap: App Basics

The most basic App consists of

an ***activity*** and a ***layout***

```
class MainActivity : AppCompatActivity() {

    private var mCurrentIndex = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        mCurrentIndex = 1
    }

    private val questionBank = listOf(
        Question("This year the Olympic Games are in Seattle", answer: false),
        Question("First modern Olympic games were in 1796", answer: false),
        Question("This year games there will a total of 40 sports", answer: false),
        Question("The Olympic flag contains 5 colors", answer: true),
        Question("Women were allowed to participate in the Olimpics in 1900", answer: true),
        Question("Skateboarding is an Olympic sport", answer: true),
        Question("Gold medals are made of 50% gold", answer: false),
        Question("Michael Phelps is the athlete with the most Olympic medals", answer: true),
        Question("Men and women do NOT compete against each other in any ...", answer: true),
        Question("Swimming obstacle race was Olympic in one Game", answer: false)
    )


    fun updateQuestion(view: View){
        mCurrentIndex ++
        if (mCurrentIndex > questionBank.size) {
            mCurrentIndex = 1
        }
        val questionTextResID = questionBank[mCurrentIndex].textResId
        questionView.setText(questionTextResID)
    }
```

```
<?xml version="1.0" encoding="utf-8"?>


<!--The LinearLayout is the root element of the layout hierarchy, and thus
it must specify the Android Resource XML namespace at http://schemas.andro
<LinearLayout 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"
              android:gravity="center"
              android:orientation="vertical" >

    <TextView
              android:text="Olimpic Games this year are in Seattle"
              android:layout_width="match_parent"
              android:layout_height="wrap_content" android:id="@+id/question

    <LinearLayout
              android:orientation="horizontal"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content">

        <Button
                  android:text="false"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content" android:id="@+id/butt
                  android:onClick="check_false"

        />
        <Button
                  android:text="True"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:id="@+id/butt_true" android:padding="10dp"
                  android:onClick="check_true"
        />
    </LinearLayout>
    <Button
              android:text="Next question"
```



Android: Basic Terminology

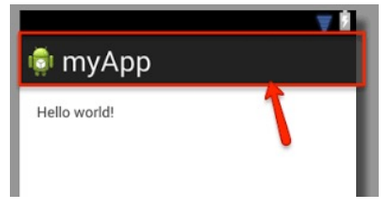
Activity: entry point for interacting with the user.
Single screen with a user interface.

View: Class that represents the basic building block for user interface components.

A View occupies a rectangular area on the screen: drawing and event handling.

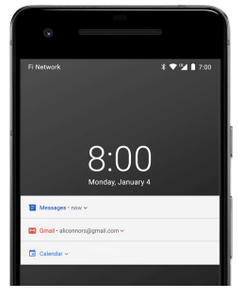
Android: Basic Terminology

Event: action that occurs then the user interacts with widgets (click, scrolling, etc.)



Action bar: A primary toolbar within the activity that may display the activity title, application-level navigation
affordances, and other interactive items.

Notification: is a message that Android displays outside your app's UI to provide the user with reminders,
communication from other people, or other
timely information from your app.



Handle Events

Capture the events from the specific View object that the user interacts with.

`onClick()`

From `View.OnClickListener`. This is called when the user either touches the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses the suitable "enter" key or presses down on the trackball.

`onLongClick()`

From `View.OnLongClickListener`. This is called when the user either touches and holds the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses and holds the suitable "enter" key or presses and holds down on the trackball (for one second).

`onFocusChange()`

From `View.OnFocusChangeListener`. This is called when the user navigates onto or away from the item, using the navigation-keys or trackball.

`onKey()`

From `View.OnKeyListener`. This is called when the user is focused on the item and presses or releases a hardware key on the device.

`onTouch()`

From `View.OnTouchListener`. This is called when the user performs an action qualified as a touch event, including a press, a release, or any movement gesture on the screen (within the bounds of the item).

`onCreateContextMenu()`

From `View.OnCreateContextMenuListener`. This is called when a Context Menu is being built (as the result of a sustained "long click"). See the discussion on context menus in the Menus developer guide.

Respond to a Button Click (1)

1. **Set *onClick* function in the xml file** */app/res/layout/activity_main.xml*

```
<Button
        android:text="Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" tools:layout_editor_absoluteY="43dp"
        tools:layout_editor_absoluteX="0dp"
        android:id="@+id/button"
        android:onClick="butt_action"/>
```

+ Implement function in the Kotlin class **file** *app/java/…/MainActivity.kt*

```
fun butt_action( view:View){
    //perform action
}
```

Respond to a Button Click (2)

2. Set *onClick* Listener

```
val btn_click_me = findViewById(R.id.button) as Button
        btn_click_me.setOnClickListener {
            // your code to perform when the user clicks on the button
            butt_action()
        }
```

With Kotlin Android Extensions (View binding)

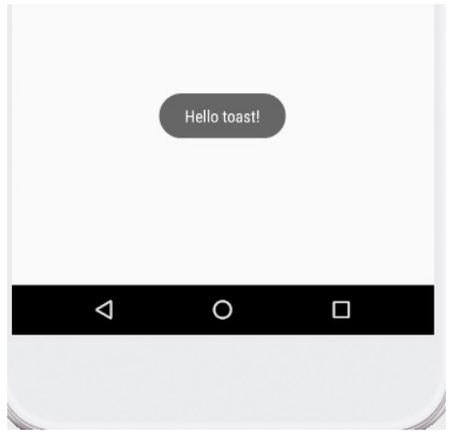
```
import kotlinx.android.synthetic.main.activity_main.*

val btn_click_me = findViewById(R.id.button) as Button
        button.setOnClickListener {
            // your code to perform when the user clicks on the button
            butt_action()
        }
```

Toasts

- Pop-up message that appears on screen for a few seconds
- Two different lengths (*SHORT* and *LONG*)
- Structure
  - `val myToast = Toast.make (applicationContext, text, duration)`
  - `myToast.show()`

```
val text = "Hello toast!"
val duration = Toast.LENGTH_LONG

val toast = Toast.makeText(this, text, duration)
toast.setGravity(Gravity.CENTER or Gravity.BOTTOM, 0, 500)
toast.show()
```

You can also specify the location on the screen



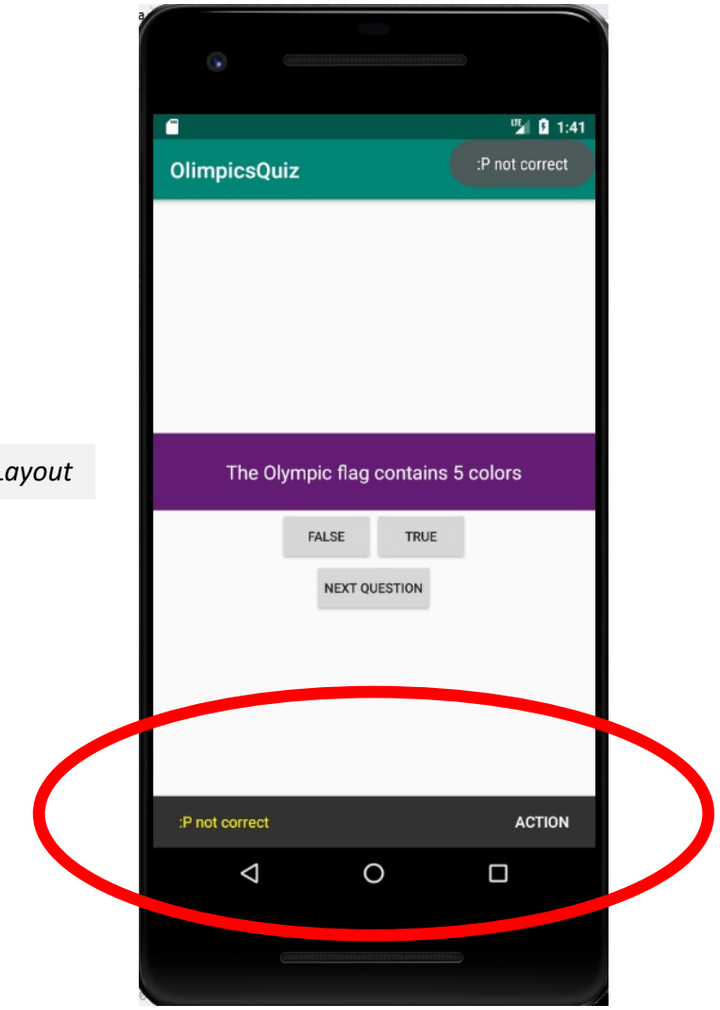
Snackbar

- Pop-up message that appears on screen for a few seconds

```
fun show_snackbar(text:String){
    Snackbar.make(
        myLayout,
        text,
        Snackbar.LENGTH_SHORT
    ).show()
}
```

id of the main xml Layout

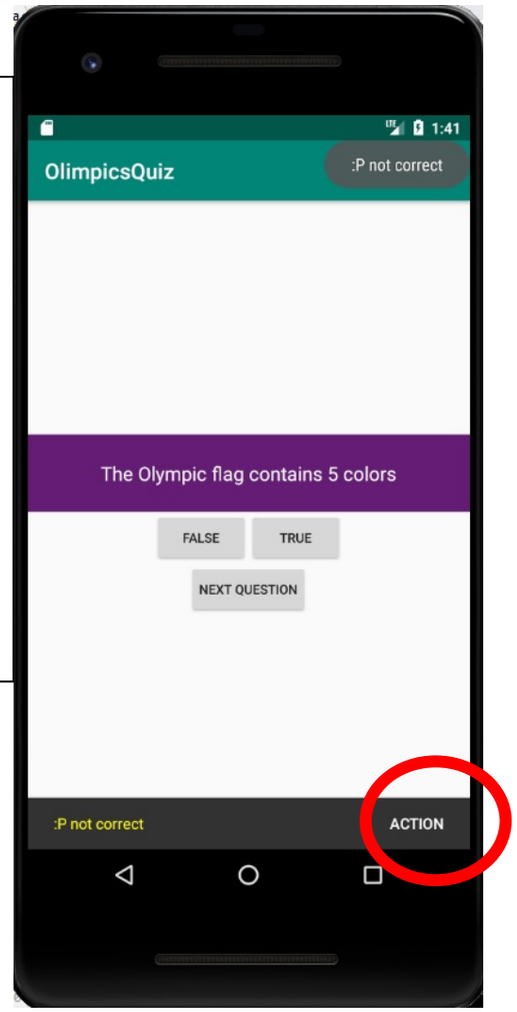
- **Can add action**



Snackbar: add action

Add an action to the sackbar:
in this example, the action is just showing a Toast.
If the user clicks on the “action” text, a Toast will appear on the screen

```
fun show_custom_snackbar(text:String){
    val snackbar =
        Snackbar.make(myLayout, text, Snackbar.LENGTH_LONG)
    snackbar.setAction(
        "ACTION"
    ) { Toast.makeText(applicationContext, "ACTION FROM SNACKBAR",
Toast.LENGTH_SHORT).show() }

    snackbar.setActionTextColor(Color.WHITE)

    val snackbarView = snackbar.view
    val snackbarText =

snackbarView.findViewById<View>(android.support.design.R.id.snackbar_text) as
TextView
    snackbarText.setTextColor(Color.YELLOW)
    snackbar.show()
}
```



Testing your Android App

Option 1: Physical Device

Install App on your device

+ App run faster

+ real final result

Option 2: Virtual Emulator

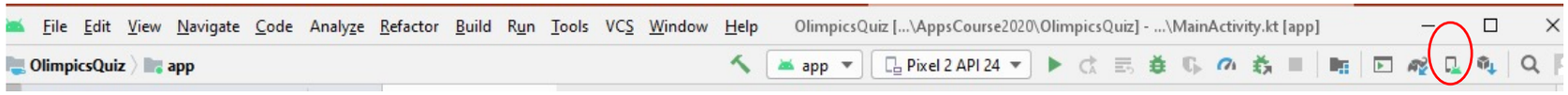
Android Virtual Device (AVD)

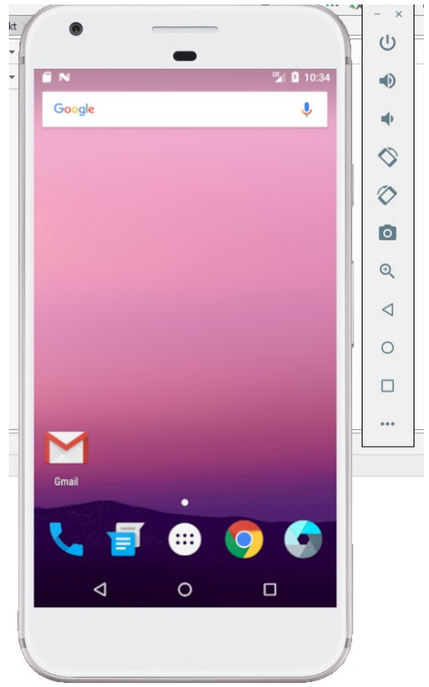
+ Don’t need to plug an Android device

Android Emulator:
Fast and Feature-Rich

Setting Up the AVD

Configuration that defines the characteristics of an Android device that you want to simulate in the Android Emulator.





Debuging your App

- A) The App crashes

- B)The App misbehave

Debuging your App

- Logcat
- Android Lint
- Android breakpoints

Debuging your App

- Enable debugging with real phone

  - **Install LLDB** from SDK Manager **:** If your project includes C/C++ code
  - **Enable debugging on your device:**

    If you're using the emulator, this is enabled by default.

    For a connected device, you need to enable debugging in the device developer options.

Use the system Log

Logcat message format: `Log.d(tag, message)`

Log methods (from highest to lowest priority):

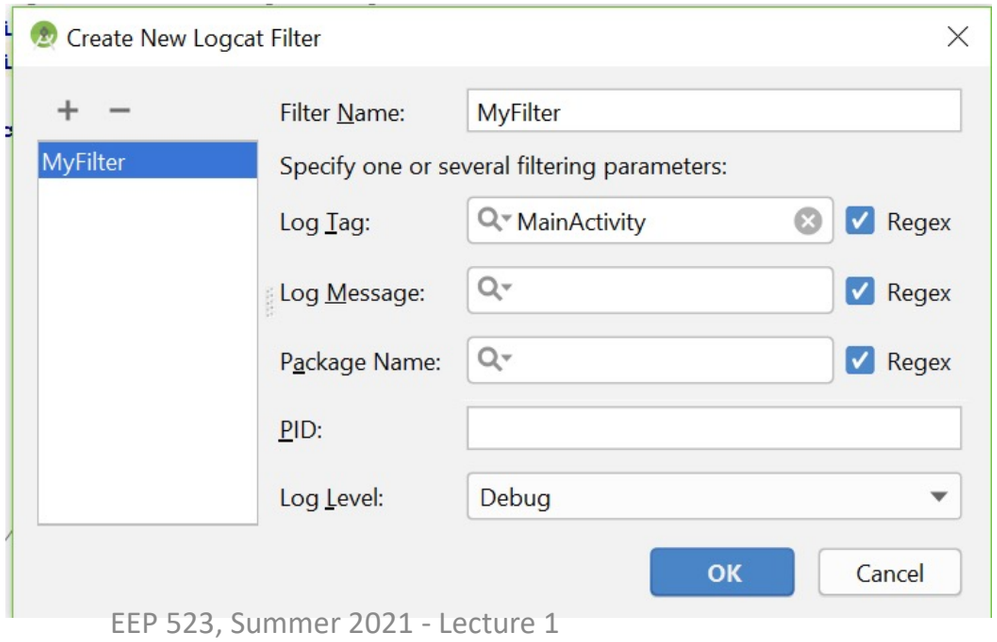
- Log.e (String, String) (error)
- Log.w (String, String) (warning)
- Log.i (String, String) (information)
- Log.d (String, String) (debug)
- Log.v (String, String) (verbose)

Use the system Log

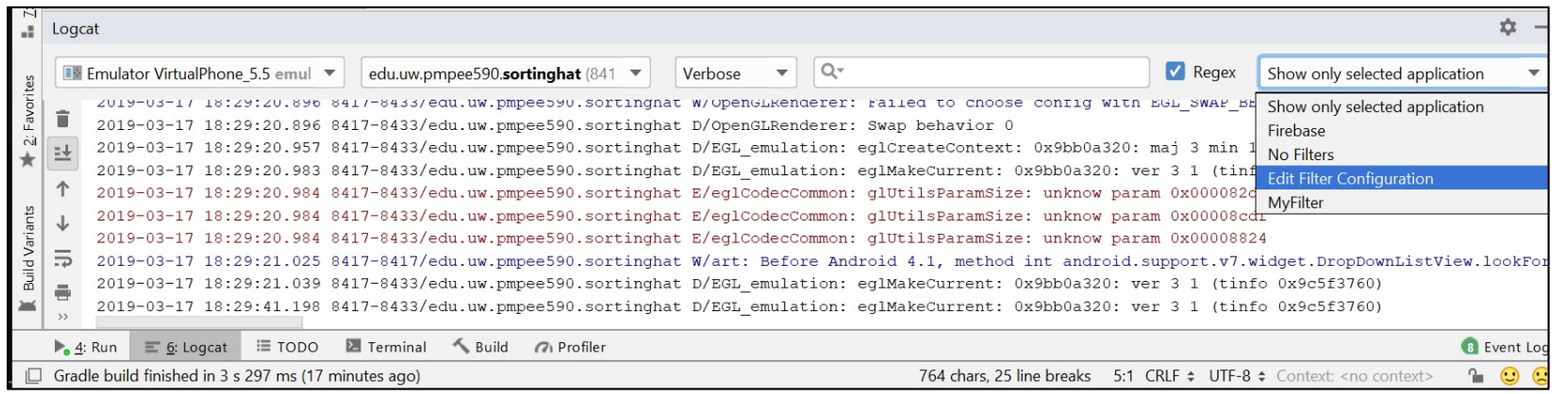
```
...
import android.util.Log
...

class MainActivity : AppCompatActivity() {

    private val TAG: String = MainActivity::class.java.simpleName
    private var index = 0
    ...

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    index +=1
    Log.d(TAG, "App Initialized $index  times")
    ...

}
```

Filter logcat messages

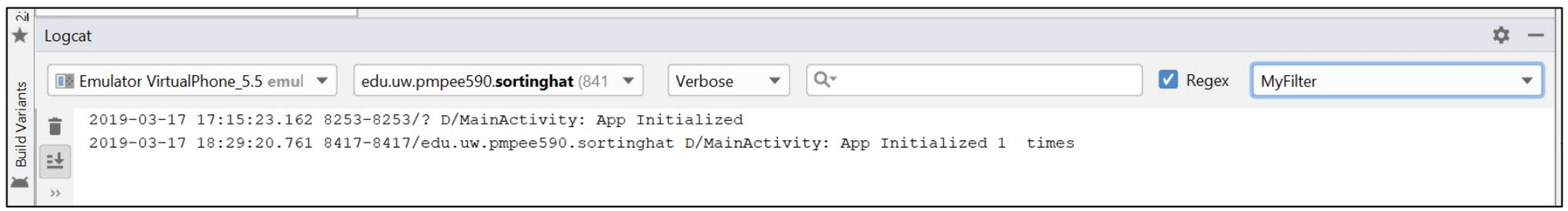
Edit Filter Configuration

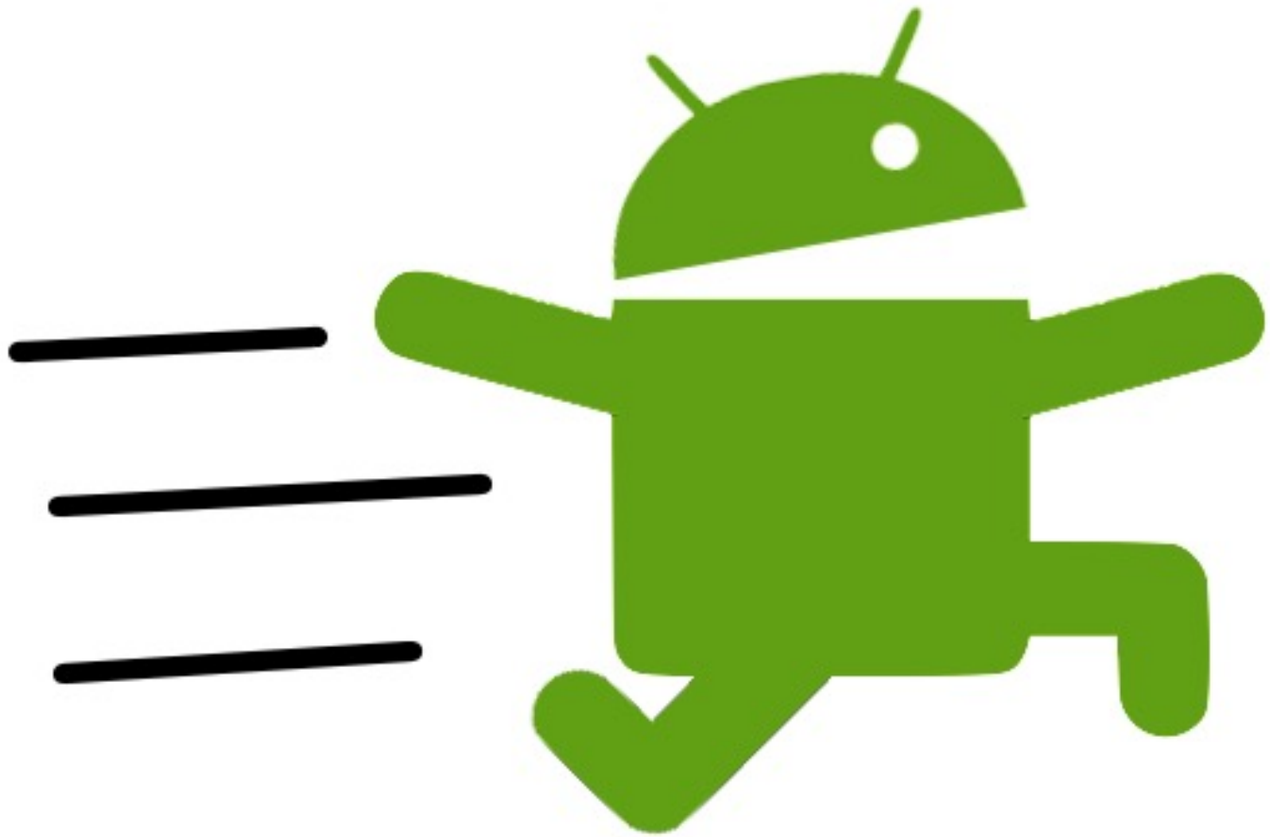


Filter logcat messages



Display only selected filter
"MyFilter"





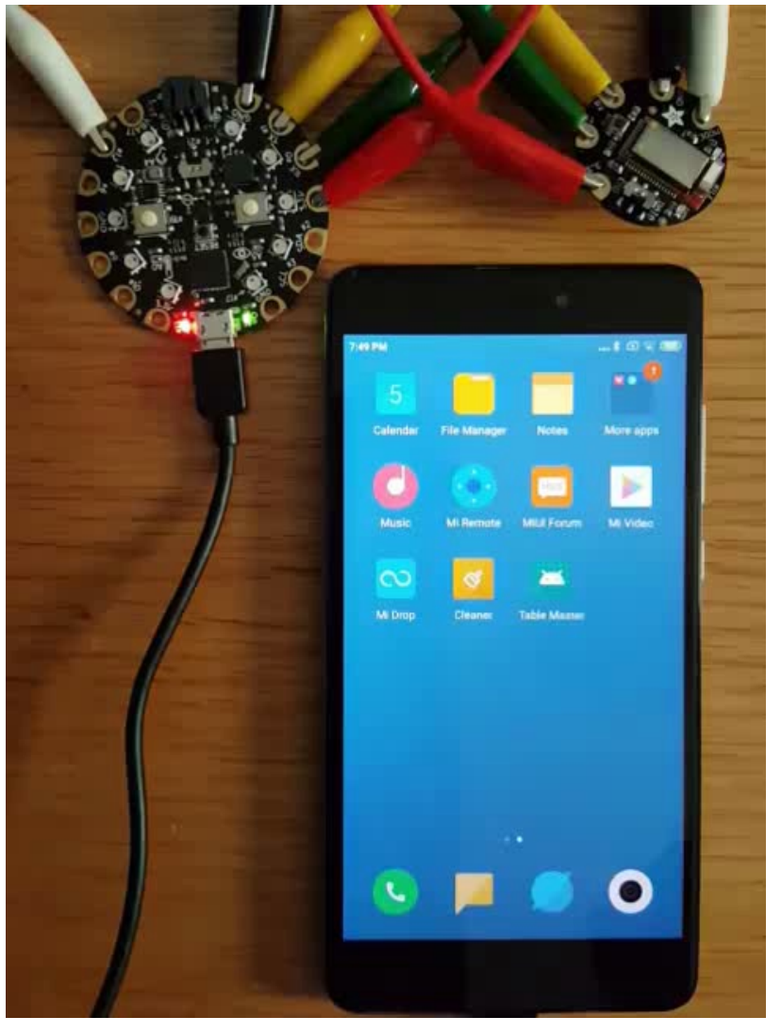
Your Questions

?

Final Projects from Spring 2019

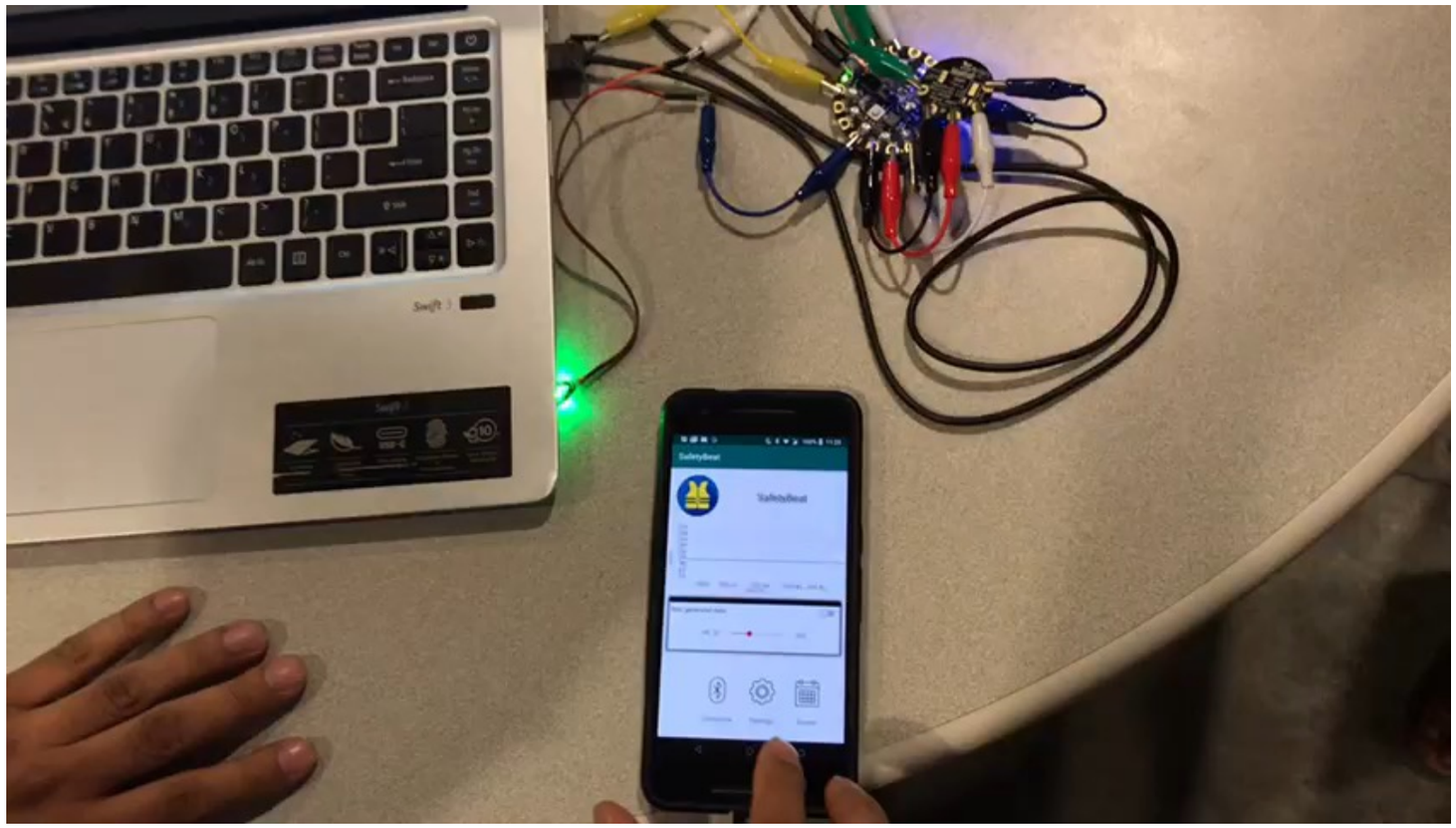
Table Master

Best project award



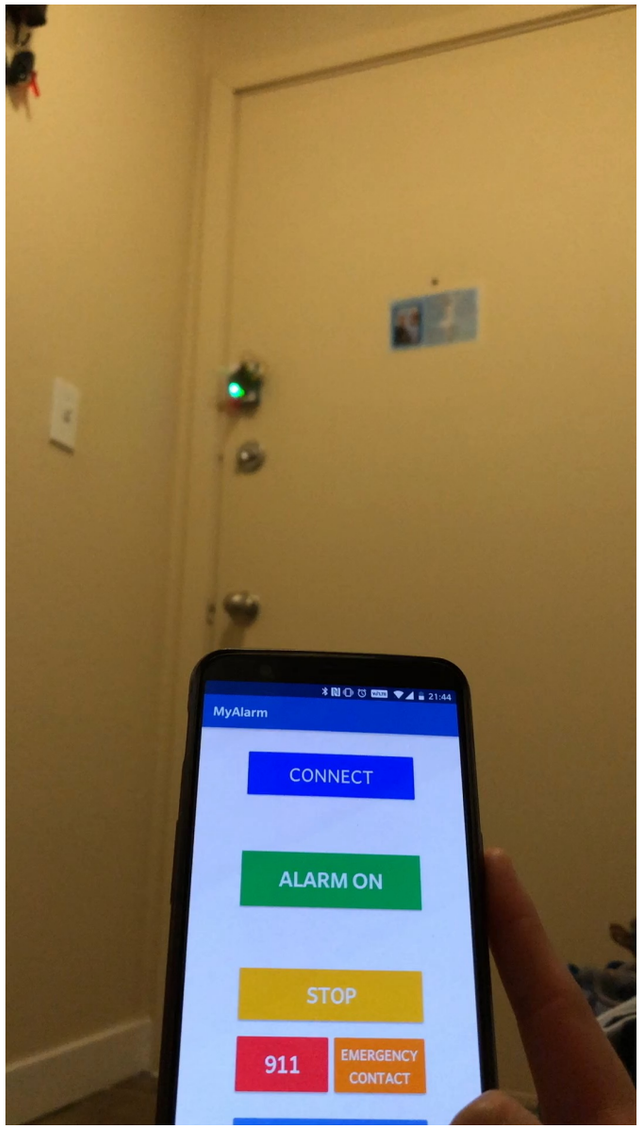
Final Projects from Spring 2019

SafetyBeats



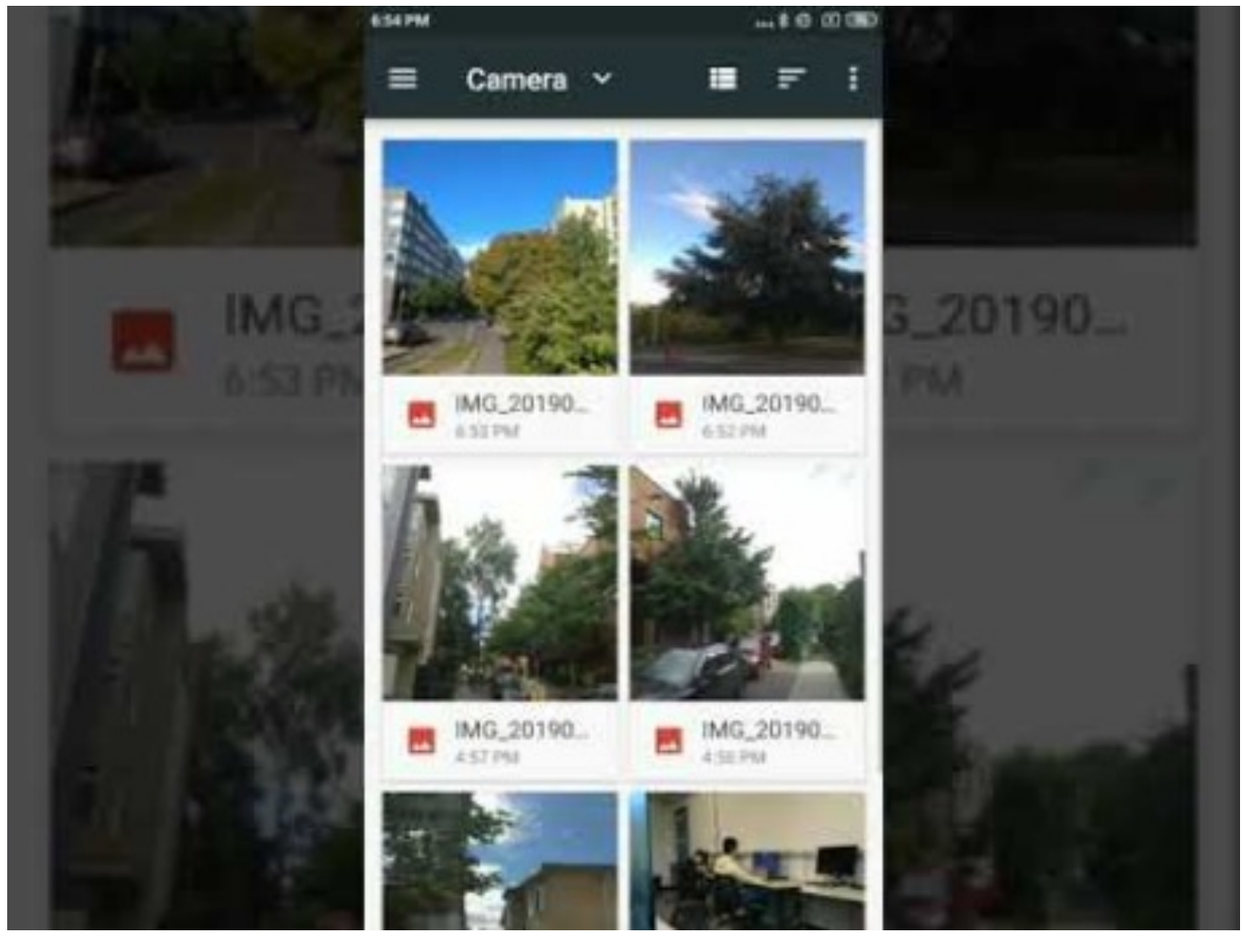
Final Projects from Spring 2019

MyAlarm



Final Projects from Spring 2019

TravKer

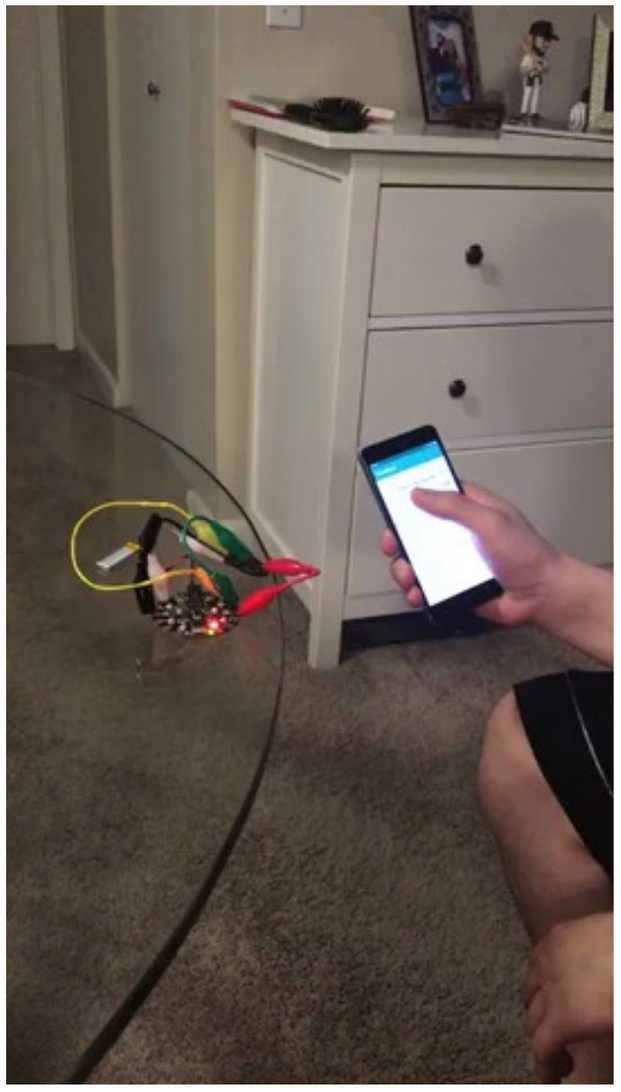


Final Projects from Spring 2019

TreatsDispenser

Final Projects from Spring 2019

GuitarHero



Final Projects from Spring 2019