Android
I - Hello World

Google™

Stefan BORNHOFEN
EISTI

Smartphone?

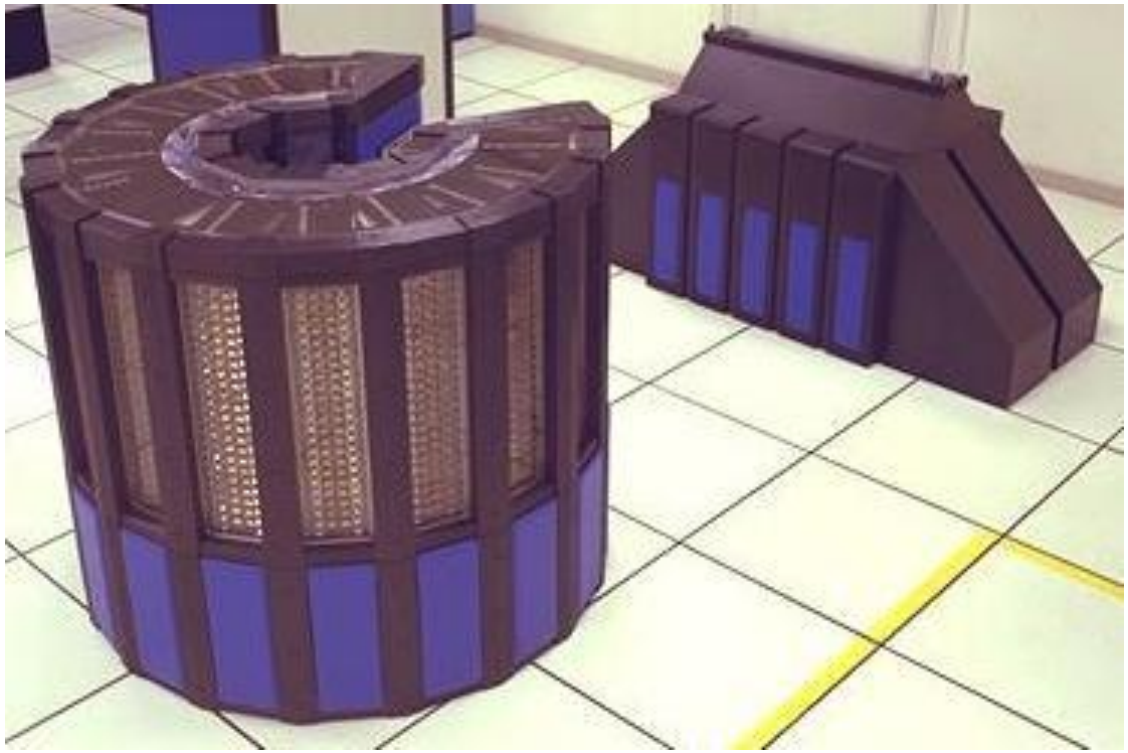
Cell phone

PDA

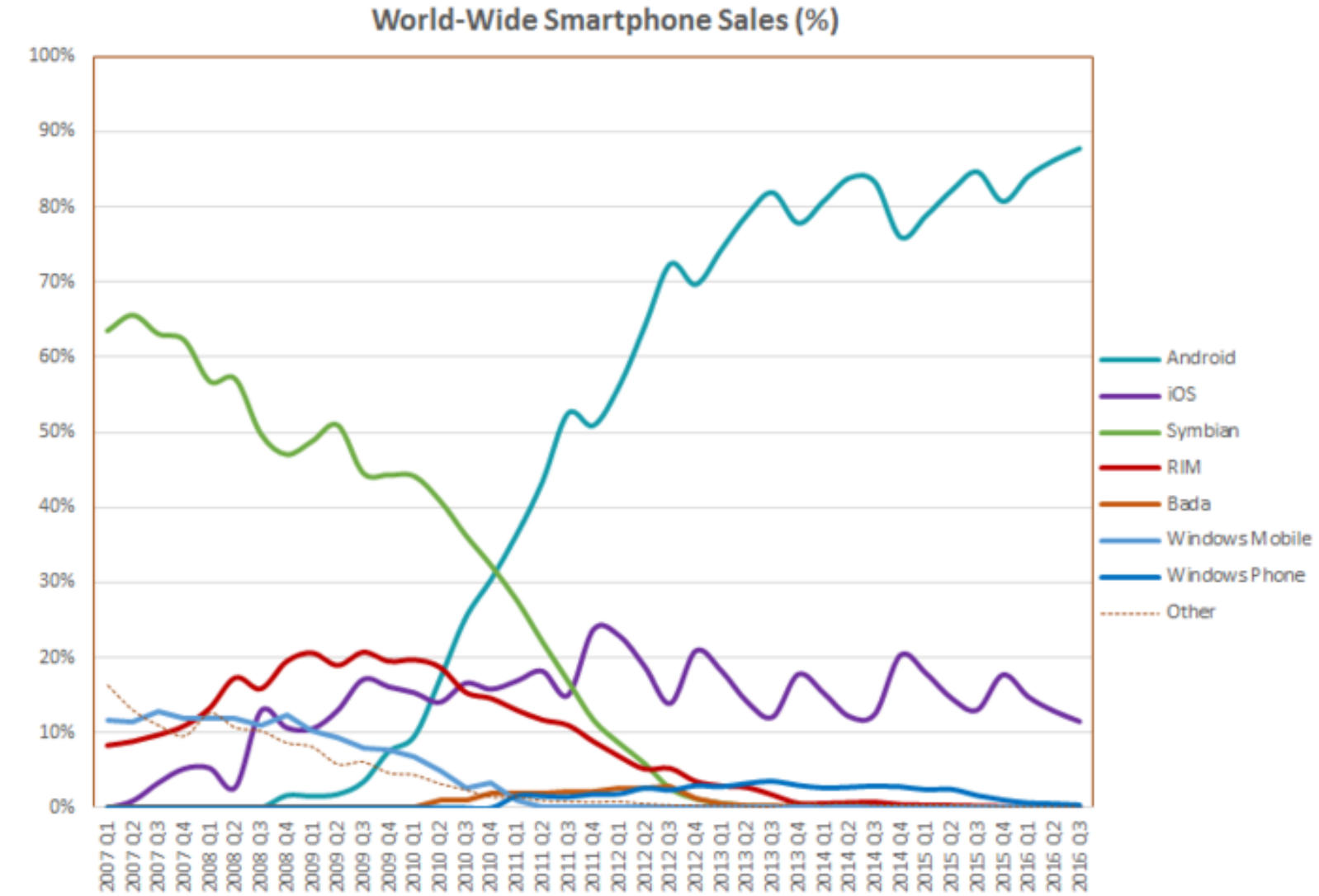
Smartphone!

Current smartphones are as fast as the supercomputers of the 80s.

=



A Cray-2 operated by NASA in the 1980s



source: http://en.wikipedia.org/wiki/Mobile_operating_system

Competitors

- Iphone : somewhat ahead in terms of ergonomics and usability, but expensive and controlled by Apple
- Symbian (Nokia): the oldest of his kind
- RIM (BlackBerry) : for professional use
- Windows Phone : growing slowly
- Bada (Samsung): dying, replaced by Tizen

If you can't afford an iPhone, get an Android.

History

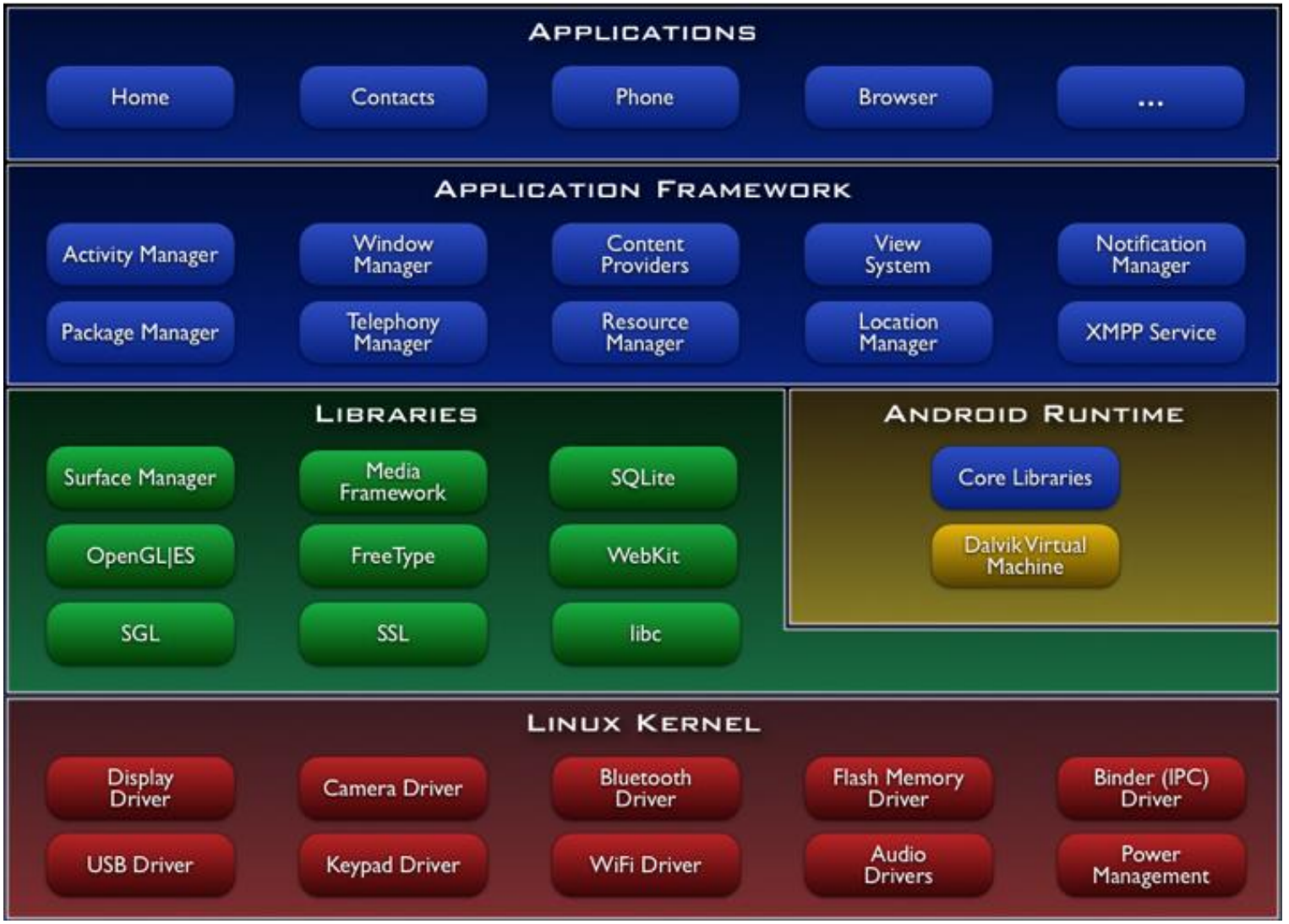
- July 2005 : Google acquires Android (startup)
- November 2007 :
  Open Handset Alliance
  - Google, Samsung, Motorola, LG, Intel, …
  - Android specifications

- October 2008 : Android goes open source
- December 2010 : Android 2.3 *last smartphone-only version*
- January 2011 : Android 3.x *for tablets*
- October 2011 : Android 4.x *unified version (smartphone + tablet)*
- November 2014: Android 5
- October 2015: Android 6
- August 2016: Android 7

Big picture

- Open source project (you can download and recompile the Android source code)
- Linux kernel
- Applications executable in parallel
- Permissions
- 2D and 3D graphics (OpenGL)
- File system
- SQLite database
- Bluetooth, EDGE, 3G, 4G, WiFi
- Camera, GPS, compass, accelerometer
- API: Java (C/C++ is possible)
- SDK: compiler, debugger, device emulator

System Architecture



Web App vs. Native App

Web App

- web site that is specifically optimized for use on a smartphone
- user interface is built with web standard technologies
- available at a URL (public, private, or perhaps behind a login)
- optimized for the specifics of a mobile device
- not installed on the phone
- not available in Google Play
- not written in Java

Native App

- installed on the Android phone
- access to hardware (speakers, accelerometer, camera, etc.)
- written in Java
- Google Play

API

- J2SE : java.util.*, java.io.*, java.lang.*, . . .
- UI : android.widget.*, android.view.*, android.graphics.*
- Phone: android.telephony.*
- SMS : android.telephony.SmsManager
- Web : android.webkit.WebView
- Camera: android.hardware.*
- Database: android.database.*
- Multimedia : android.media.*
- HTTP: org.apache.http.client.*

Dalvik & ART

Android Java Virtual Machine

- use a bytecode format which is different from classic Java bytecode.
- optimized for embedded applications.
- you cannot directly run Java class files on Android.
- do not support awt/swing.

Android 5: Dalvik is replaced by ART

Creating an Android Application

Android application = Java + resources

javac: compiles the Java sources
dx: converts and compresses the Java class files into *.dex (« Dalvik Executable »)
aapt: zips *.dex + resources into *.apk (« Android Package »)
adb: deploys the *.apk on the device

Dont worry, it's just one click.

Running an Android Application

The Android system implements **the principle of least privilege**. Each Android application lives in its own security sandbox.

By default,

- the system assigns each app a unique Linux user ID (unknown to the app)
- every app runs in its own Linux process
- each process has its own virtual machine (VM), so an app's code runs in isolation from other apps.
- the app does not have any permissions. The app can request permission to access device data such as the user's contacts, storage (SD card), camera, etc. All permissions must be granted by the user.

Key components

Activity
- Presentation layer of an Android application
- An Android application can have several Activities

View
- User interface components, e.g. buttons or text fields
- ViewGroups are also called layout managers, can be nested

Intent
- Asynchronous messages which allow the application to request functionality from other components of the Android system
- Direct call (explicit Intent) or ask Android to find registered components (implicit Intent).
- Allow combining loosely coupled components to perform certain tasks

Service
- No visual interface
- Services perform background tasks without providing a user interface
- They can notify the user via the notification framework in Android

ContentProvider
- Share data with other applications
- SQLite database used in conjunction with a ContentProvider

BroadcastReceiver
- Responds to system-wide broadcast announcements (system messages and Intents)
- Will be notified by the Android system, if the specified situation happens

Your Android bible

http://developer.android.com

Installation

- Install the Android Studio

https://developer.android.com/studio/index.html

You might want to connect the
Android SDK source code

  - Download the Android sources in the SDK Manager
  - Restart the Android Studio

Hello World

- New -> Project

Project Architecture

- **manifests/**

AndroidManifest.xml : Fundamental characteristics of the application (permissions, feature requirements, main Activity, …)

- **java/**

Source files

- **res/**

Non-code application resources (images, strings, layout files, etc.).

  - **drawable:** drawable objects (such as bitmaps)
  - **layout:** *XML files* that define the user interface
  - **menu:** *XML files* that define the application menus
  - **values:** various XML files that contain resources, such as string and color definitions.

Gradle:

  - Build automation tool (like Ant, Maven), creates the apk file.

Resources

- An Android application is more than just code
- For every resource that you include in your Android project, the build tool defines a unique integer ID, which you can use to reference the resource from within the code (class “R”)
- Resources must be lowercase

Advantages

- MVC!
- update your application without modifying code
- customize your application for different device configurations (screen resolution, device orientation, language, etc)

```
package fr.eisti.android.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Execution

Run on the Emulator

- Create an Android Virtual Device (Tools -> Android –> AVD Manager)
- Start the virtual device (slooooooow!)
- Click Run from the toolbar.
- Android Studio installs the app on your virtual device and starts it.

Run on a real device

- Is the Google USB driver installed? (Tools -> Android -> Android SDK Manager)
- Plug in your Android-powered device to your machine with a USB cable
- Is your device supported? If Google USB driver does not work, you might need to install a special ADB driver.
- Ensure that USB debugging is enabled in the device settings
- Click Run from the toolbar.
- Android Studio installs the app on your connected device and starts it.

Android Device Monitor

Stand-alone tool for debugging and analysis

The Debug Perspective

- Debug - Displays previously and currently debugged Android applications and its currently running threads
- Variables - When breakpoints are set, displays variable values during code execution
- Breakpoints - Displays a list of the set breakpoints in your application code
- LogCat - Allows you to view system log messages in real time. The LogCat tab is also available in the DDMS perspective.

The DDMS Perspective

- Devices - Shows the list of devices and AVDs that are connected to ADB.
- Emulator Control - Lets you carry out device functions.
- LogCat - Lets you view system log messages in real time.
- Threads - Shows currently running threads within a VM.
- Heap - Shows heap usage for a VM.
- Allocation Tracker - Shows the memory allocation of objects.
- File Explorer - Lets you explore the device's file system.

Log

Log is a logging class that you can utilize to print out messages to the LogCat. Common logging methods include:

- v(String tag, String message): verbose
- d(String tag, String message): debug
- i(String tag, String message): information
- w(String tag, String message): warning
- e(String tag, String message): error

Example

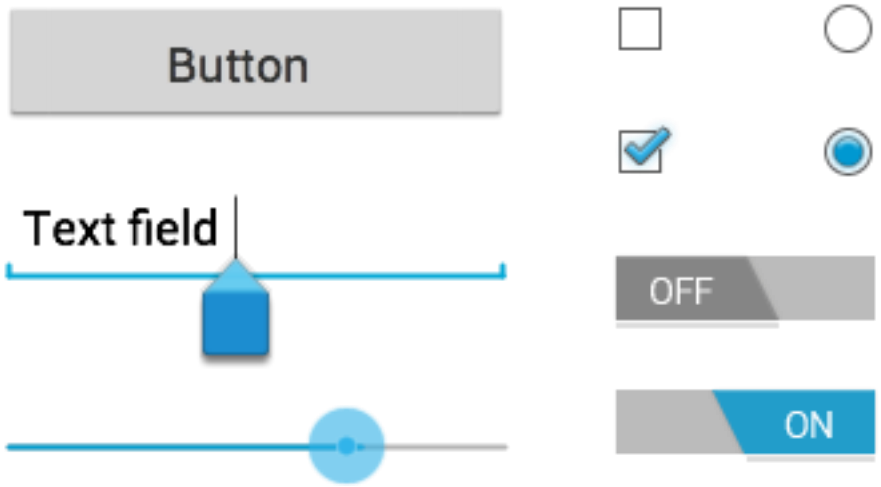
```
Log.i("MyTag", "This is a log entry, my variable
  = " + variable);
```

Activity

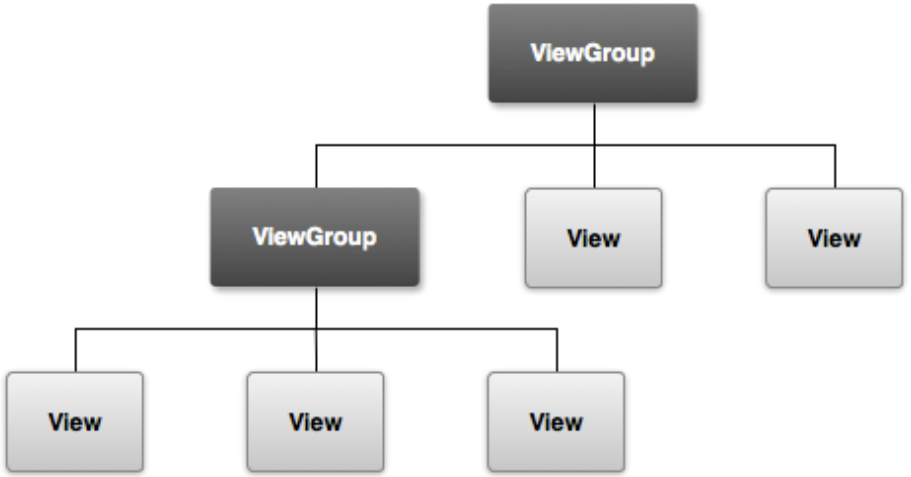
- An application usually consists of multiple activities
- Provides a screen with which users can interact (displays *Views* and handles *Events*).
- Each activity has a window for its user interface
- One activity is specified as the "main" activity, which is presented to the user when launching the application
- Each activity can start another activity. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack ("back stack").
- When the user is done with the current activity and presses the Back button, it is popped from the stack (and destroyed) and the previous activity resumes.

View

- The user interface for an activity is provided by a hierarchy of views—objects derived from the View class.
- Each view controls a particular rectangular space within the activity's window and can respond to user interaction.



Layout



- Layouts are views derived from *ViewGroup* that provide a unique layout model
- XML layout file (saved as a resource)
- Relevant layout managers:
  - LinearLayout
  - GridLayout
  - RelativeLayout
  - AbsoluteLayout

Button click

1) Add the android:onClick attribute to the <Button> element in your XML layout.

```
<Button
    android:id="@+id/mybutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/clickme"
    android:onClick="clicked" />
```

2) Within the Activity, the following method handles the click event:

```
public void clicked(View view) {
    // do something in response to button click
}
```

Android vs. Java Swing

- Activity : JFrame
- View : JComponent
- TextView : JLabel
- EditText : JTextField
- Button : JButton
- RadioButton: JRadioButton
- …

Exercise

Write a basic currency converter with hard coded exchange rates (2p).

Write an improved currency converter (1p).

- Remove the convert button and compute the result after every user interaction
- Change radio groups to spinners
- Convert currency in both directions