Android 101 for iOS Developers

Stephen Barnes - @smbarne

objc.io/issue-11

OVERVIEW

- A Word on UI Design
- Application Structure and Language
- Building Blocks: Activities, Fragments, ListViews, ViewPagers, and more
- Android Lifecycle
- Android Layouts

ASSUMPTIONS

- Working knowledge of Java
- Able to install and use the Android Development Tools.

RECOMMENDATIONS

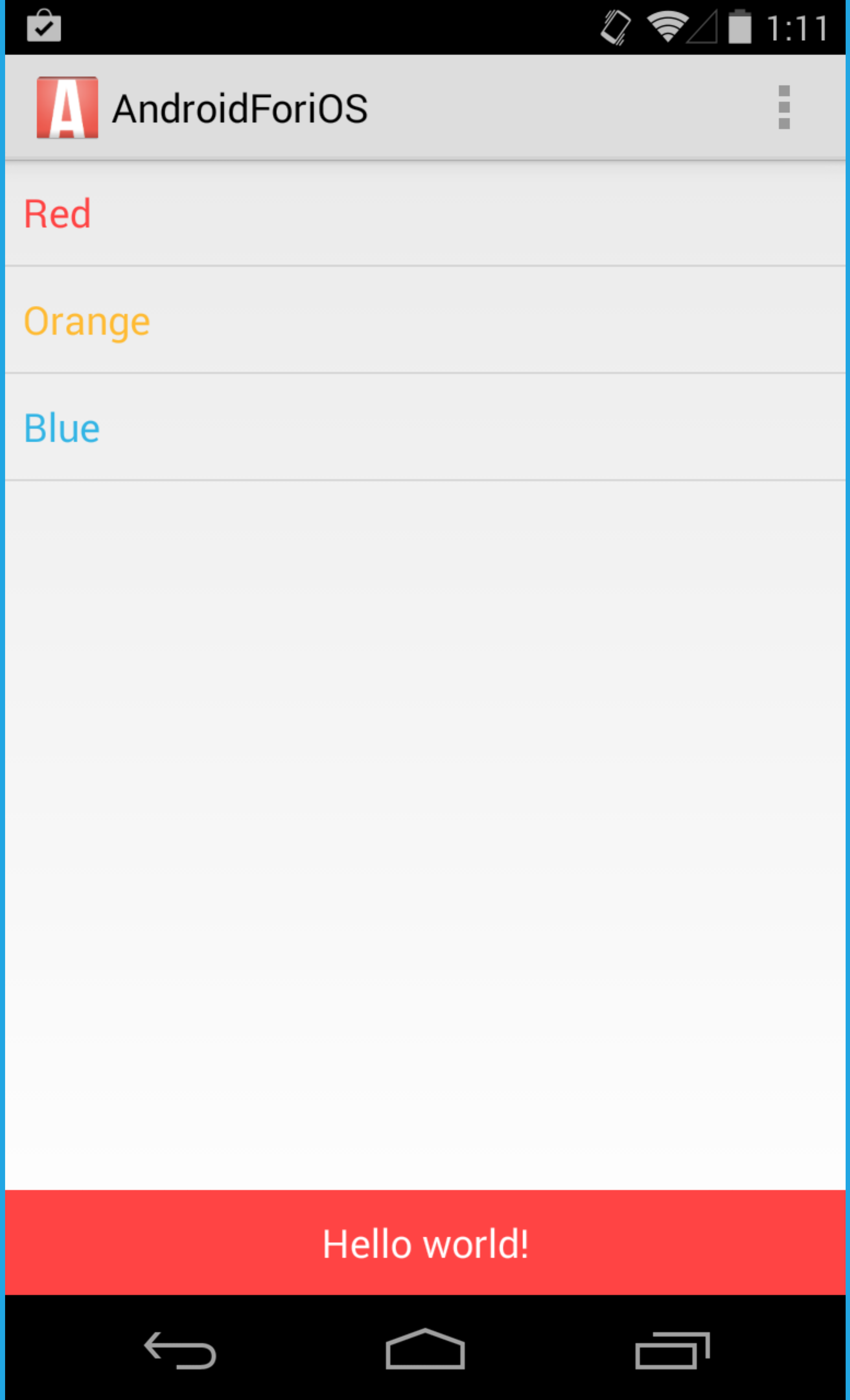
Read through the Building your first app tutorial by Google when possible.

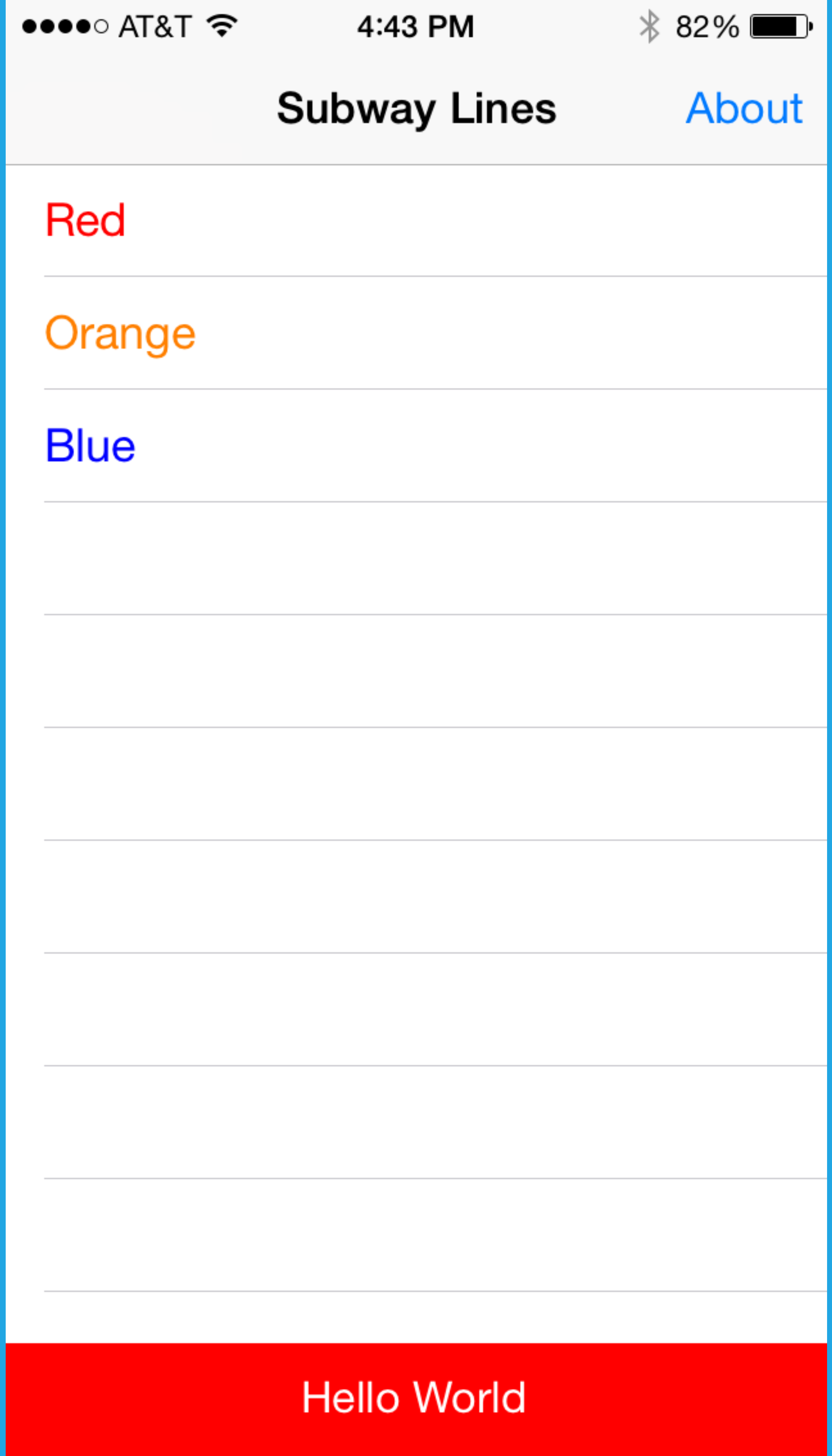
Resources

Objc.io article: objc.io/issue-11/android_101_for_ios_developers.html

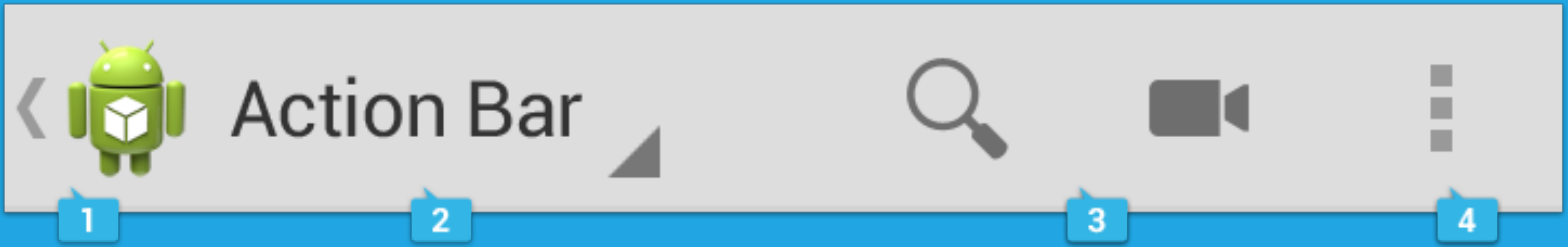
Code: github.com/smbarne/AndroidForiOS

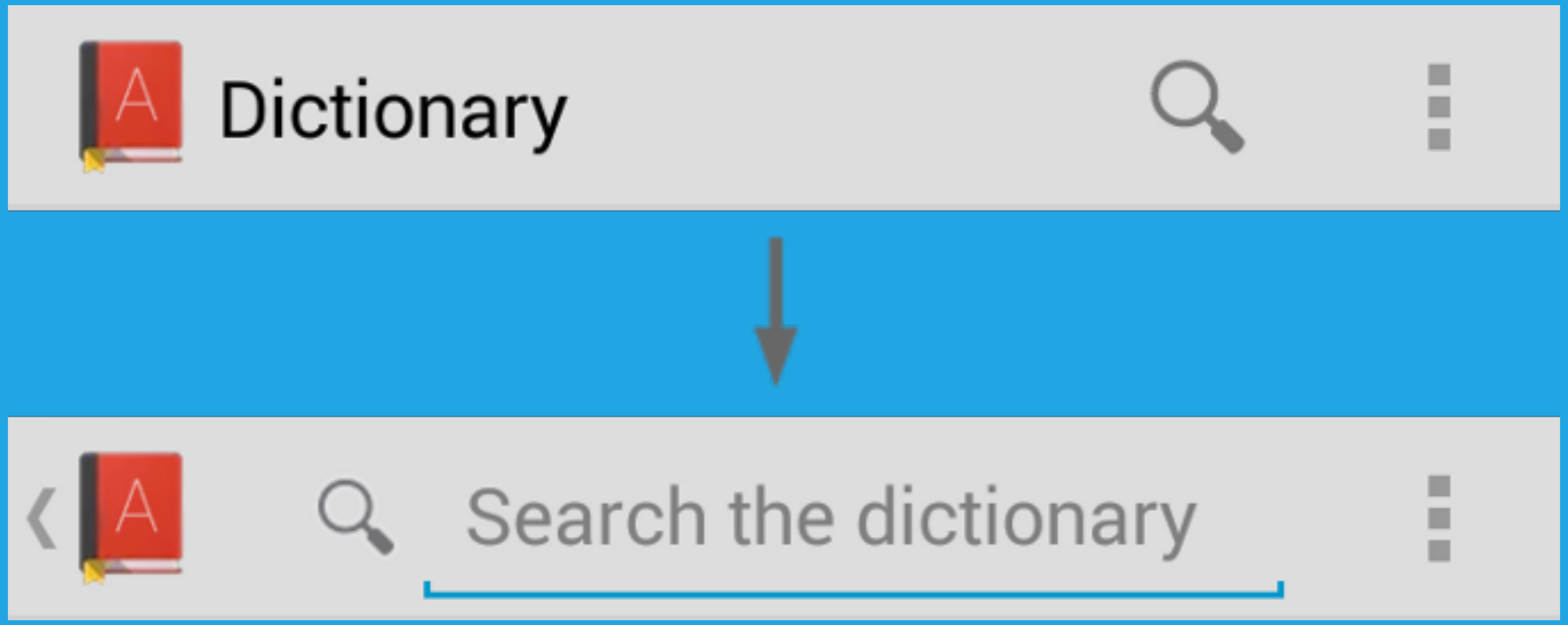
A WORD ON UI DESIGN


1:11
AndroidForiOS
Red
Orange
Blue
Hello world!

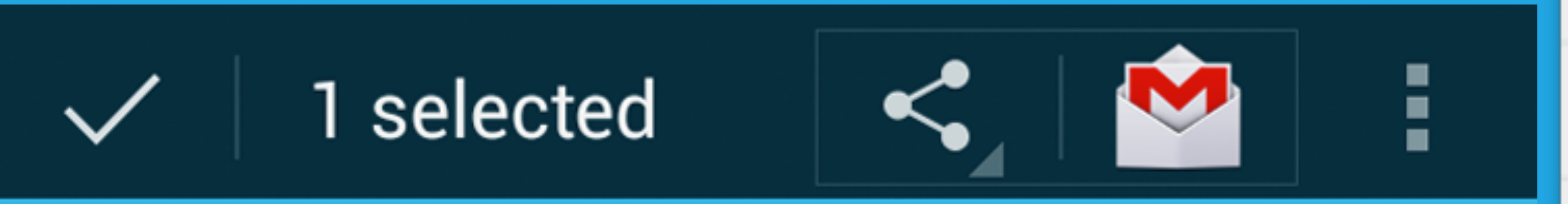

●●●●○ AT&T
4:43 PM
82%
Subway Lines
About
Red
Orange
Blue
Hello World

ACTIONBAR

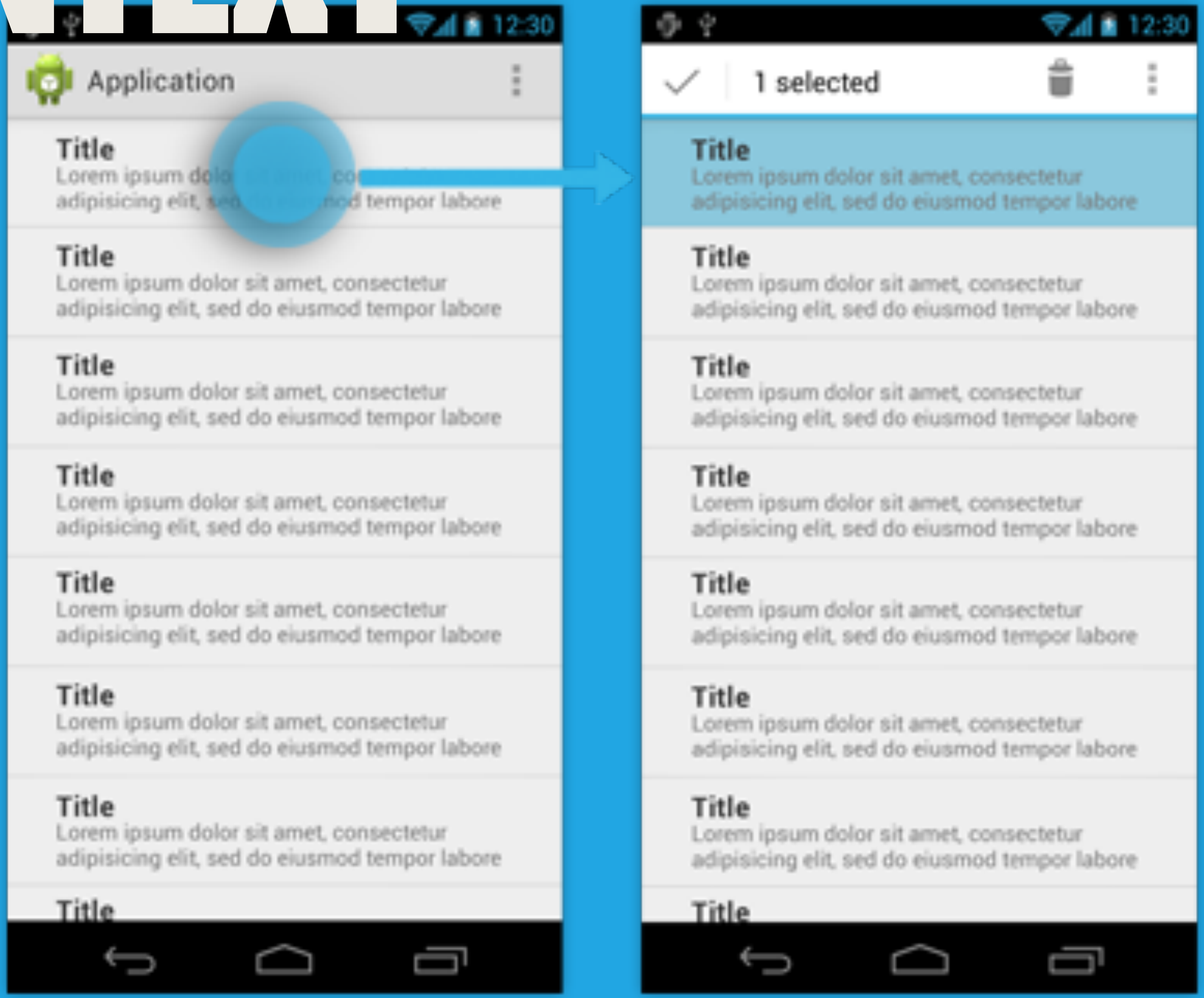




ACTIONBAR CONTEXT



1 selected



12:30

Application

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title

12:30

1 selected

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

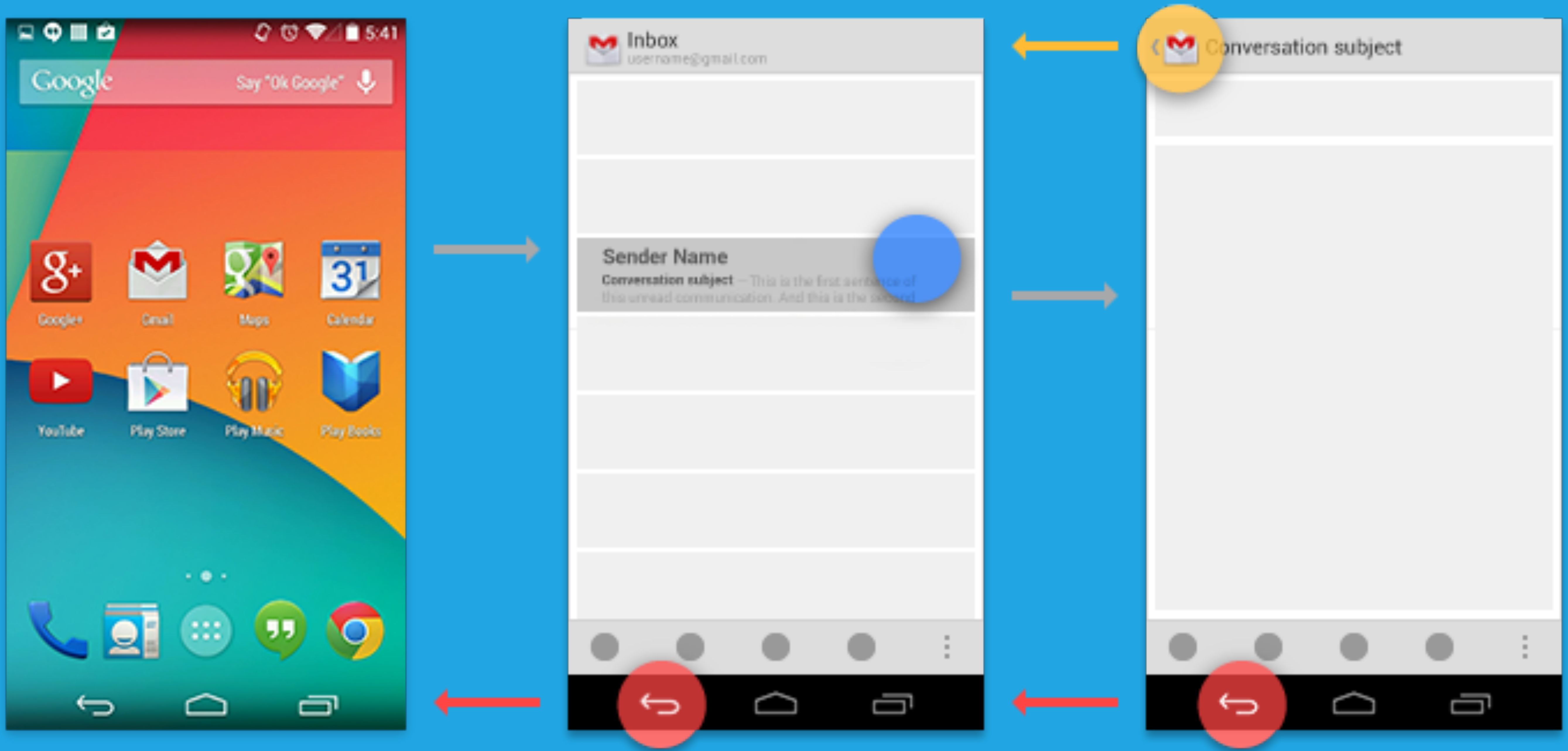
Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor labore

Title

BACK BUTTON AND NAVIGATION

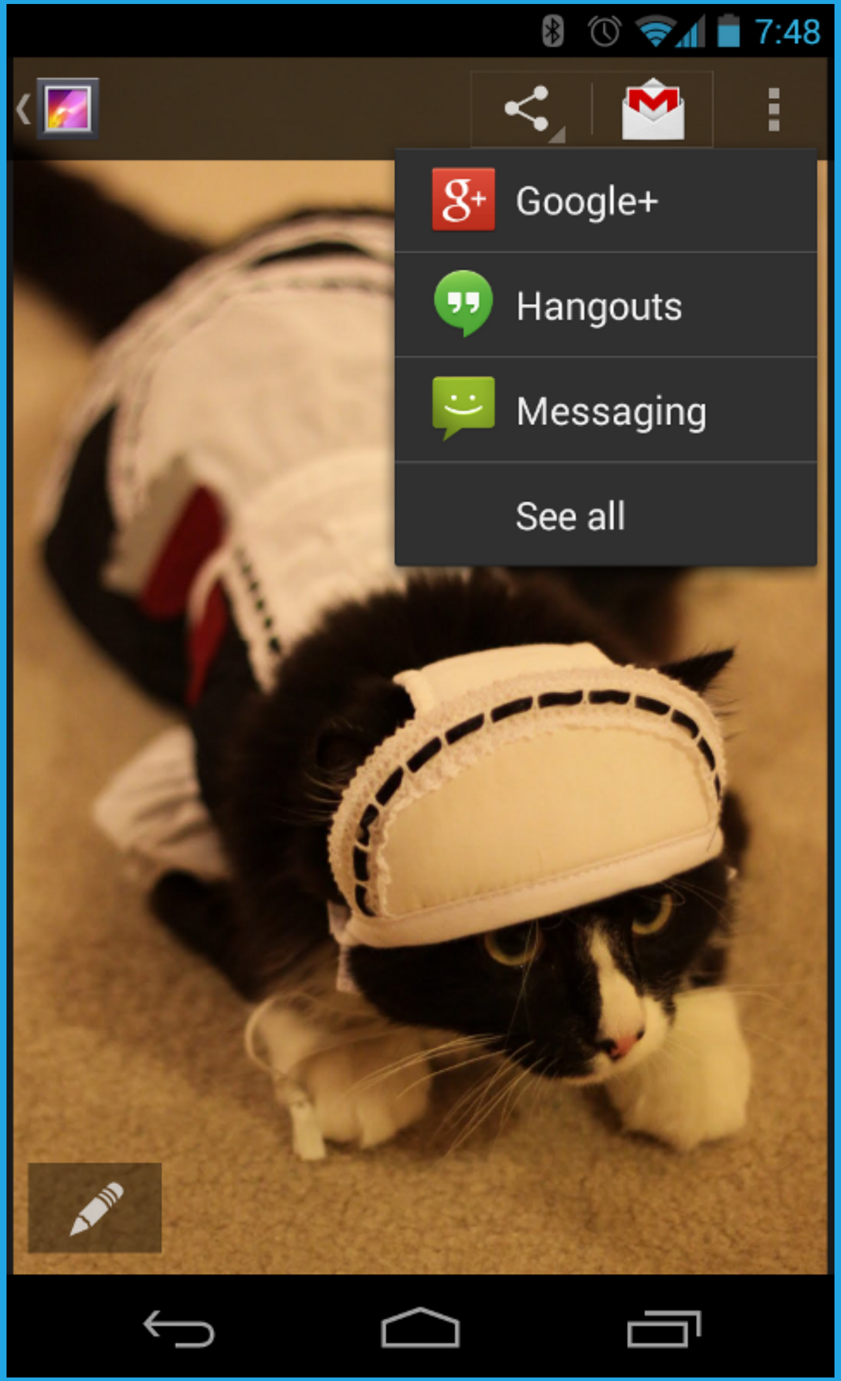


Home

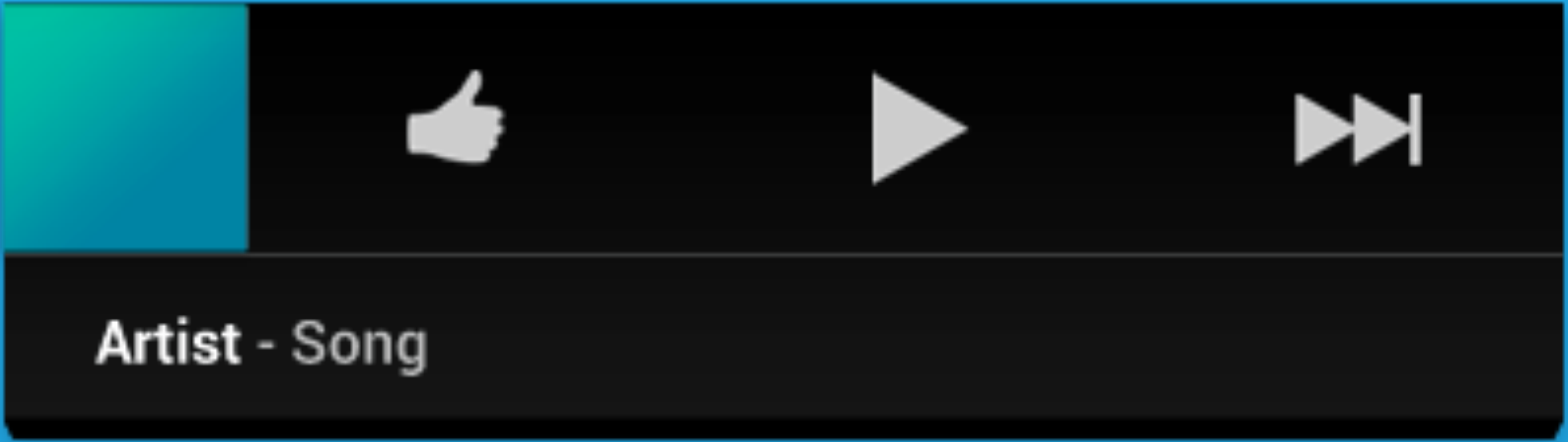
Conversation list

Conversation details

OVERFLOW MENU



WIDGETS



Artist - Song

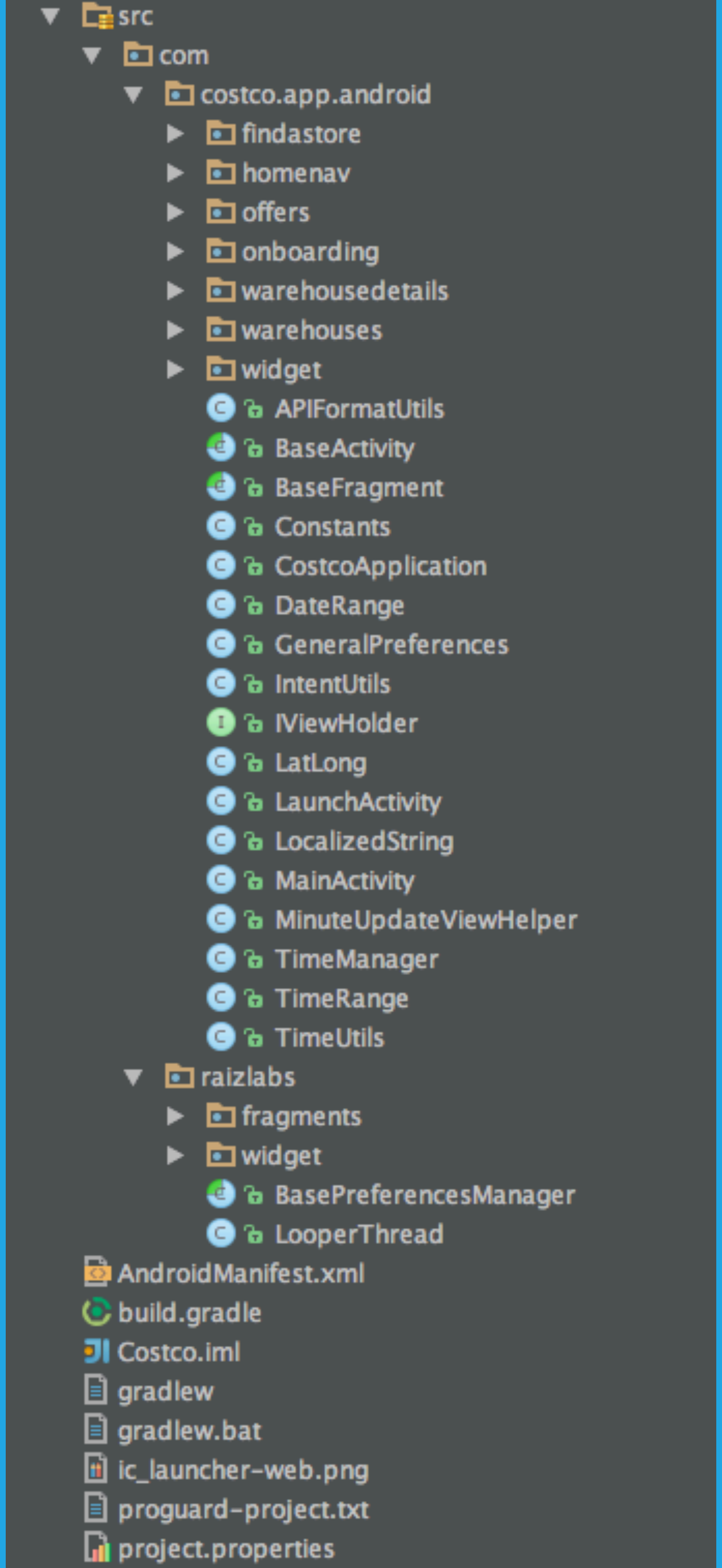
Others

- Phone vs tablet
- Rotation handling
- Aspect ratios
- No overscroll

CODE STYLING AND PATTERNS

- Leave class prefixes at home on Objective-C.
- Instance variables are prefixed with m, not _.
- Take advantage of JavaDoc please.
- Null check! Objective-C handles gracefully handles message sending to nil objects gracefully, but Java does not.
- Say goodbye to properties. You have to remember to call getVariableName() and setVariableName().

PROJECT STRUCTURE



```
▼ src
  ▼ com
    ▼ costco.app.android
      ▶ findastore
      ▶ homenav
      ▶ offers
      ▶ onboarding
      ▶ warehousedetails
      ▶ warehouses
      ▶ widget
        APIFormatUtils
        BaseActivity
        BaseFragment
        Constants
        CostcoApplication
        DateRange
        GeneralPreferences
        IntentUtils
        IViewHolder
        LatLong
        LaunchActivity
        LocalizedString
        MainActivity
        MinuteUpdateViewHelper
        TimeManager
        TimeRange
        TimeUtils
    ▼ raizlabs
      ▶ fragments
      ▶ widget
        BasePreferencesManager
        LooperThread
  AndroidManifest.xml
  build.gradle
  Costco.iml
  gradlew
  gradlew.bat
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

- Folder Structure based on package naming
- AndroidManifest.xml is required and similar to the info.plist on iOS
- build.gradle for each project

RESOURCES (RES FOLDER)

DRAWABLES

drawable

drawable-hdpi

drawable-xhdpi

drawable-xxhdpi

Drawables are images and any other renderable objects that you can define (think XML defined CAShape)

NO MORE @2X - NOW YOU GET MANY, MANY BUCKETS!

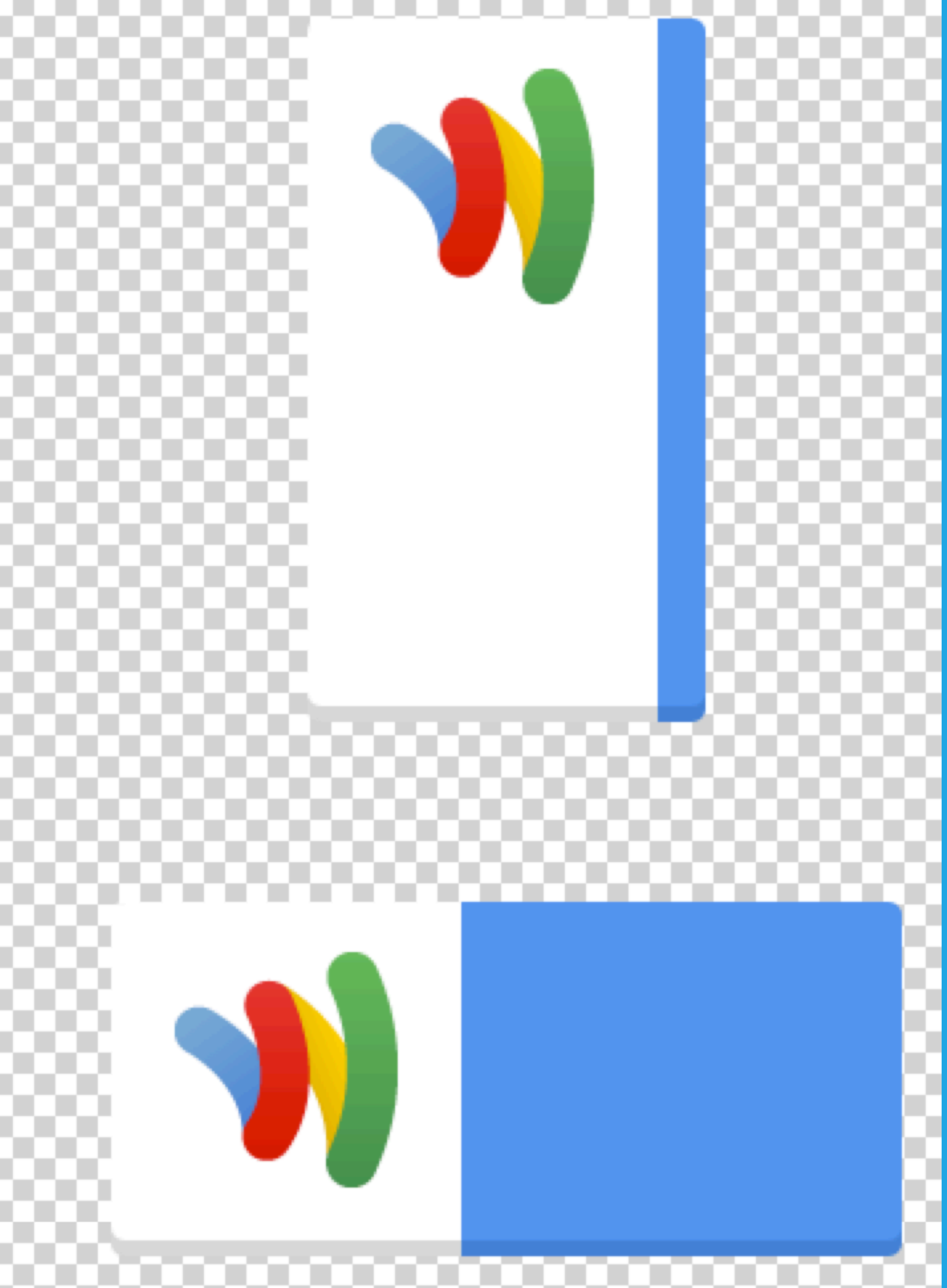
drawable-mdpi, drawable-hdpi, drawable-xhdpi, drawable-xxhdpi, etc

Drawables are typically Images

DPI (Dots Per Inch, or Density Independent Pixel)

- Each device has an internal DPI bucket
- Android picks the resource from the corresponding bucket or the closet one and scales it

9-PATCHES (STRETCHABLE IMAGES)



Dimens and strings

```
<dimen name="Margin.Standard">5dp</dimen>
```

```
<string name="LocationServices.DisabledPrompt.Title">Location services disabled</string>
```

STYLES

```
<style name="Button" parent="@style/TextStyle.Standard">
    <item name="android:textSize">@dimen/TextSize.Large</item>
    <item name="android:background">@color/Panel.White.Transparent</item>
    <item name="android:minWidth">@dimen/MinTouchableSize</item>
    <item name="android:minHeight">@dimen/MinTouchableSize</item>
</style>
```

YOU CAN HAVE DIFFERENT VALUES FOR DIFFERENT API LEVELS!

TOOLS

Build Tools

- Gradle
- Ant

IDEs

- Android Studio (built by IntelliJ)
- Eclipse

Building Blocks

Activities, Fragments, ListViews, ViewPagers, and more

Activities <-> UIViewControllers

Activities are the basic unit of an Android app.

“An application usually consists of multiple activities that are loosely bound to each other.”

One activity in an application is specified as the "main" activity.

Activities can start and return information

- Activities can register that they handle common data such as images
- Activities can also recieve and send specific data, such as an item ID

EXAMPLE: AN ACTIVITY LAUNCHING ANOTHER ACTIVITY AND RESPONDING WHEN IT IS DONE

```java
protected void startNextActivity() {
    Intent nextActivityIntent = NextActivity.getIntent(this);
    startActivityForResult(nextActivityResult, REQUEST_CODE_NEXT_ACTIVITY);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
    case REQUEST_CODE_NEXT_ACTIVITY:
        if (resultCode == RESULT_OK) {
            Toast.makeText(this, "Result OK!", Toast.LENGTH_SHORT).show();
            // We can also do something with returnObject within data here
        }
        return;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

Next Activity Finishing with Data

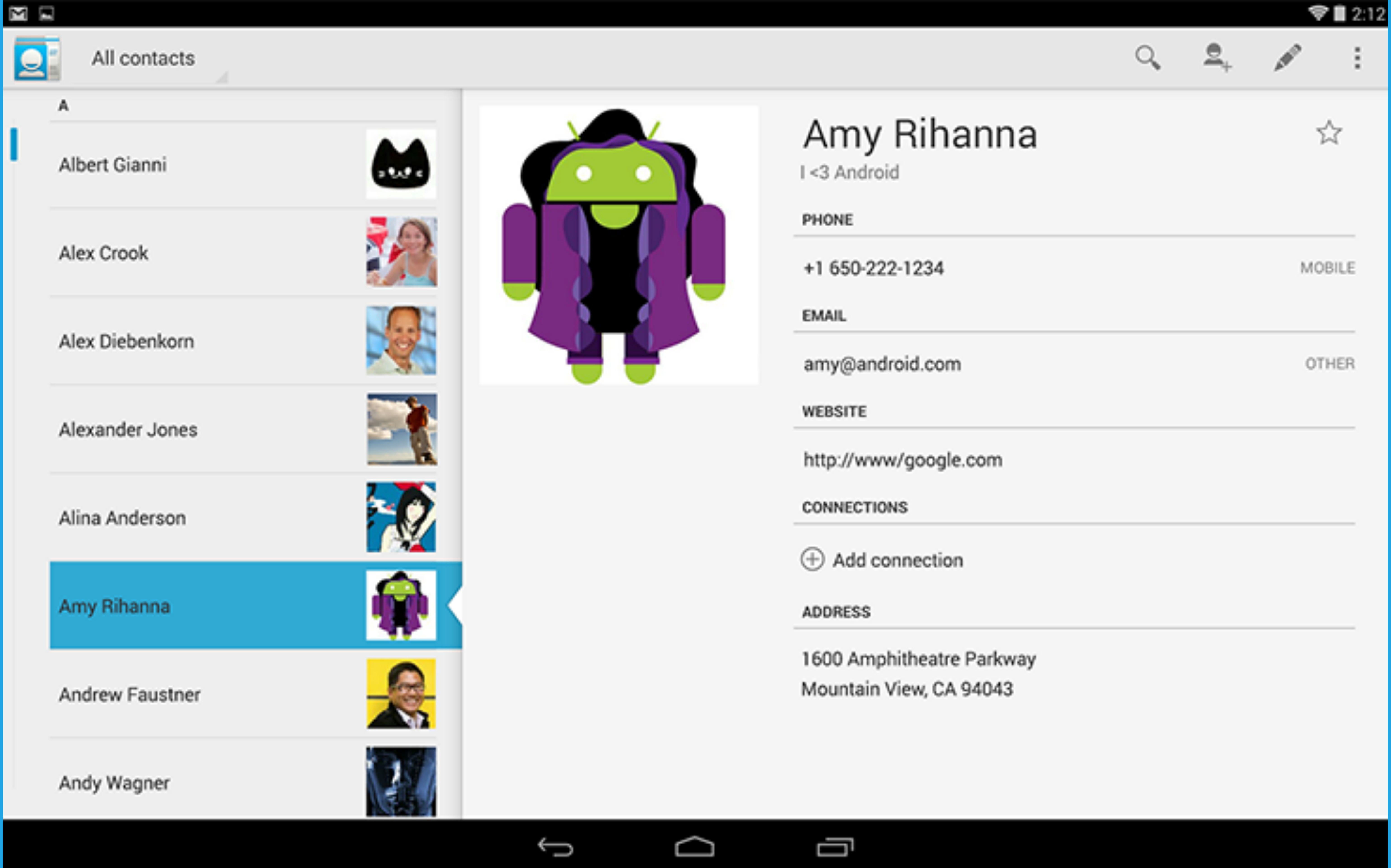
```
public static final String activityResultString = "activityResultString";

/*
 * On completion, place the object ID in the intent and finish with OK.
 * @param returnObject that was processed
 */
private void onActivityResult(Object returnObject) {
    Intent data = new Intent();
    if (returnObject != null) {
        data.putExtra(activityResultString, returnObject.uniqueId);
    }

    setResult(RESULT_OK, data);
    finish();
}
```

FRAGMENTS

“MINI CONTROLLERS THAT CAN BE INSTANTIATED TO POPULATE ACTIVITIES”

Fragments are the new, accepted way to build Android apps

- Store state
- Contain view logic
- Do not have a context, the activity has the context
- Must be tied to an activity



A

Albert Gianni

Alex Crook

Alex Diebenkorn

Alexander Jones

Alina Anderson

Amy Rihanna

Andrew Faustner

Andy Wagner

Amy Rihanna

I <3 Android

PHONE

+1 650-222-1234 MOBILE

EMAIL

amy@android.com OTHER

WEBSITE

http://www/google.com

CONNECTIONS

ADDRESS

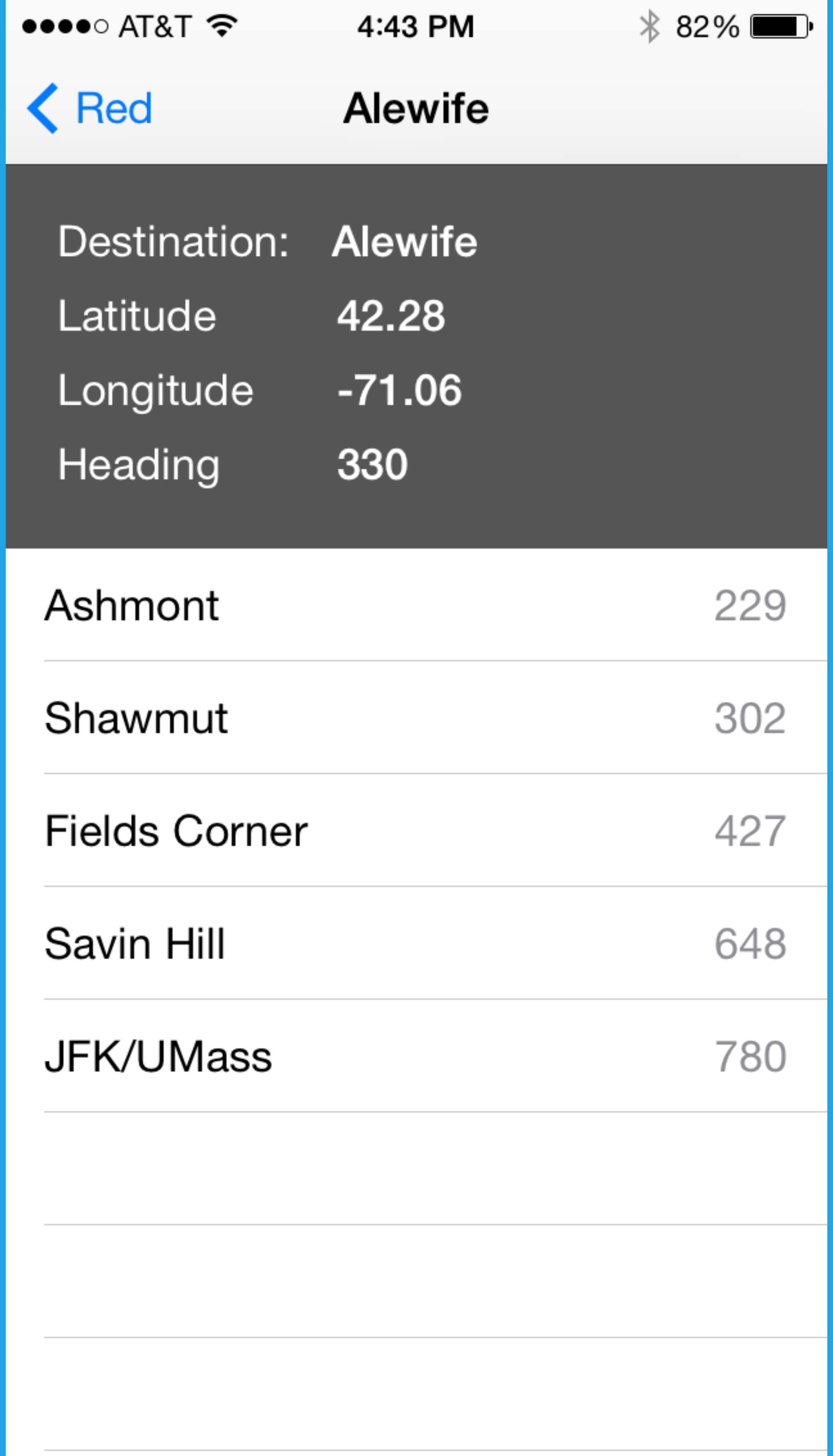
1600 Amphitheatre Parkway
Mountain View, CA 94043

The closest approximation for fragments in iOS is using child view controllers.

Explain it to me with code

Let's look at a sample UITableViewController and a sample ListFragment that show a list of prediction times for a subway trip curtesy of the MBTA.

IOS TABLEVIEW IMPLEMENTATION



Alewife

Destination: **Alewife**

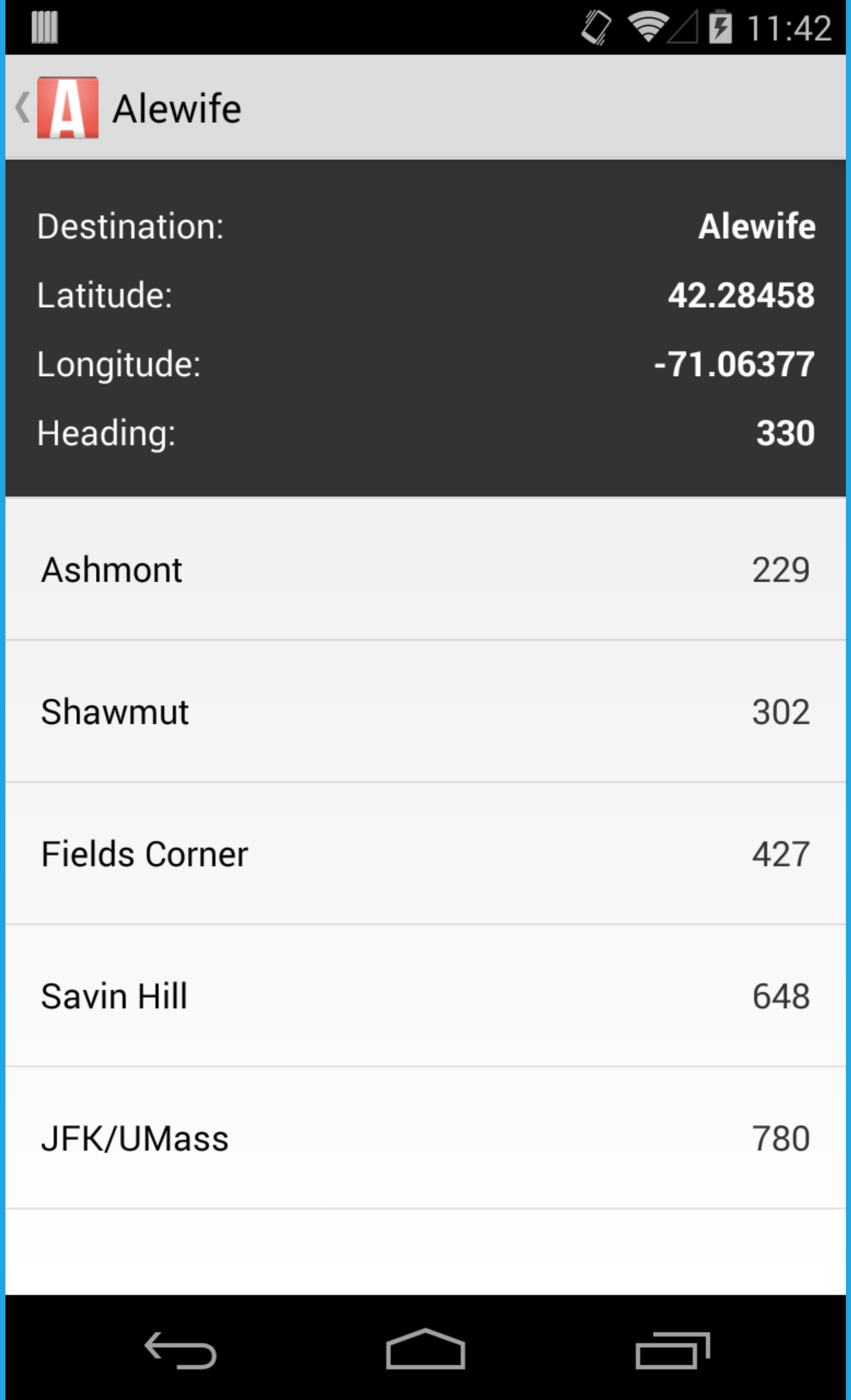
Latitude	**42.28**
Longitude	**-71.06**
Heading	**330**

Ashmont	229
Shawmut	302
Fields Corner	427
Savin Hill	648
JFK/UMass	780

```
@interface MBTASubwayTripTableTableViewController ()
@property (assign, nonatomic) MBTATrip *trip;
@end

@implementation MBTASubwayTripTableTableViewController
- (instancetype)initWithTrip:(MBTATrip *)trip
{
    self = [super initWithStyle:UITableViewStylePlain];
    if (self) {
        _trip = trip;
        [self setTitle:trip.destination];
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    [self.tableView registerClass:[MBTAPredictionCell class]
        forCellReuseIdentifier:[MBTAPredictionCell reuseId]];
    [self.tableView
        registerNib:[UINib nibWithNibName:NSStringFromClass([MBTATripHeaderView class]) bundle:nil]
        forHeaderFooterViewReuseIdentifier:[MBTATripHeaderView reuseId]];
}
```

```
#pragma mark - Table view data source
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return [self.trip.predictions count];
}

#pragma mark - UITableViewDelegate
- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section {
    return [MBTATripHeaderView heightWithTrip:self.trip];
}

- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section {
    MBTATripHeaderView *headerView = [self.tableView
         dequeueReusableHeaderFooterViewWithIdentifier:[MBTATripHeaderView reuseId]];
    [headerView setFromTrip:self.trip];
    return headerView;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
    UITableViewCell *cell = [tableView
        dequeueReusableCellWithIdentifier:[MBTAPredictionCell reuseId]
        forIndexPath:indexPath];

    MBTAPrediction *prediction = [self.trip.predictions objectAtIndex:indexPath.row];
    [(MBTAPredictionCell *)cell setFromPrediction:prediction];

    return cell;
}

- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath {
    return NO;
}

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    [tableView deselectRowAtIndexPath:indexPath animated:YES];
}
@end
```

LISTFRAGMENT IMPLEMENTATION



11:42

Alewife

Destination:	**Alewife**
Latitude:	**42.28458**
Longitude:	**-71.06377**
Heading:	**330**

Ashmont	229
Shawmut	302
Fields Corner	427
Savin Hill	648
JFK/UMass	780

```java
public class TripDetailFragment extends ListFragment {
    /**
     * The configuration flags for the Trip Detail Fragment.
     */
    public static final class TripDetailFragmentState {
        public static final String KEY_FRAGMENT_TRIP_DETAIL = "KEY_FRAGMENT_TRIP_DETAIL";
    }

    protected Trip mTrip;

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param trip the trip to show details
     * @return A new instance of fragment TripDetailFragment.
     */
    public static TripDetailFragment newInstance(Trip trip) {
        TripDetailFragment fragment = new TripDetailFragment();
        Bundle args = new Bundle();
        args.putParcelable(TripDetailFragmentState.KEY_FRAGMENT_TRIP_DETAIL, trip);
        fragment.setArguments(args);
        return fragment;
    }

    public TripDetailFragment() { }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        Prediction[] predictions =
             mTrip.predictions.toArray(new Prediction[mTrip.predictions.size()]);

        PredictionArrayAdapter predictionArrayAdapter =
             new PredictionArrayAdapter(getActivity().getApplicationContext(), predictions);

        setListAdapter(predictionArrayAdapter);
        return super.onCreateView(inflater,container, savedInstanceState);
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        TripDetailsView headerView = new TripDetailsView(getActivity());
        headerView.updateFromTripObject(mTrip);
        getListView().addHeaderView(headerView);
    }
}
```

WAT?

LET'S BREAK DOWN SOME OF THAT PIECE BY PIECE...

LISTVIEWS AND ADAPTERS

UITableView, Meet ListView

Both are sturctured around showing a linear list of Views smoothly

- Android doesn't have cells - instead any view can be used
- Reuse your views in a ListView! This can be even more important to performance than on iOS
- Default ListView views are available that you can populate just like iOS has default UITableViewCells
  - Android's GridView is similar to

ListViews are populated via Adapters

Goodbye Datasources, Hello Adapters

Instead of having a datasource delegate, Android has Adapaters

- An Adapter object acts as a bridge between an AdapterView and the underlying data for that view.
- The Adapter provides access to the data items.
- The Adapter is also responsible for making a View for each item in the data set.

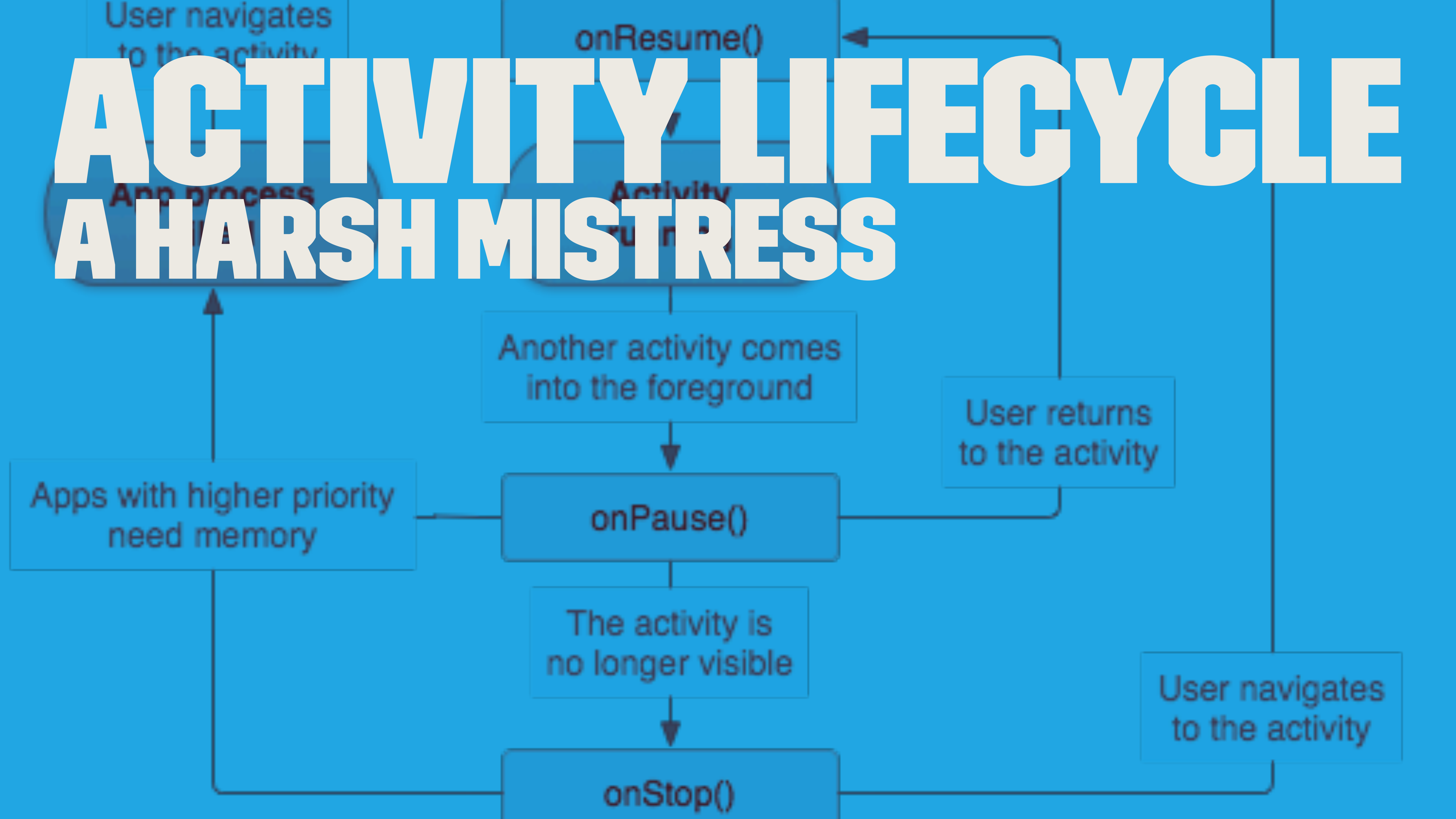
```java
public class PredictionArrayAdapter extends ArrayAdapter<Prediction> {
    int LAYOUT_RESOURCE_ID = R.layout.view_three_item_list_view;

    public PredictionArrayAdapter(Context context) {
        super(context, R.layout.view_three_item_list_view);
    }

    public PredictionArrayAdapter(Context context, Prediction[] objects) {
        super(context, R.layout.view_three_item_list_view, objects);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Prediction prediction = this.getItem(position);
        View inflatedView = convertView;
        if(convertView==null)
        {
            LayoutInflater inflater = (LayoutInflater)getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            inflatedView = inflater.inflate(LAYOUT_RESOURCE_ID, parent, false);
        }

        TextView stopNameTextView = (TextView)inflatedView.findViewById(R.id.view_three_item_list_view_left_text_view);
        TextView middleTextView = (TextView)inflatedView.findViewById(R.id.view_three_item_list_view_middle_text_view);
        TextView stopSecondsTextView = (TextView)inflatedView.findViewById(R.id.view_three_item_list_view_right_text_view);

        stopNameTextView.setText(prediction.stopName);
        middleTextView.setText("");
        stopSecondsTextView.setText(prediction.stopSeconds.toString());

        return inflatedView;
    }
}
```

ASYNCTASKS

```java
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
     protected Long doInBackground(URL... urls) {
         int count = urls.length;
         long totalSize = 0;
         for (int i = 0; i < count; i++) {
             totalSize += Downloader.downloadFile(urls[i]);
             publishProgress((int)((i '/' (float) count) * 100));
             // Escape early if cancel() is called
             if (isCancelled()) {
                break;
            }
         }
         return totalSize;
     }

     protected void onProgressUpdate(Integer... progress) {
         setProgressPercent(progress[0]);
     }

     protected void onPostExecute(Long result) {
         showDialog("Downloaded " + result + " bytes");
     }
 }
```

ACTIVITY LIFECYCLE

A HARSH MISTRESS



User navigates to the activity

onResume()

App process killed

Activity running

Another activity comes into the foreground

User returns to the activity

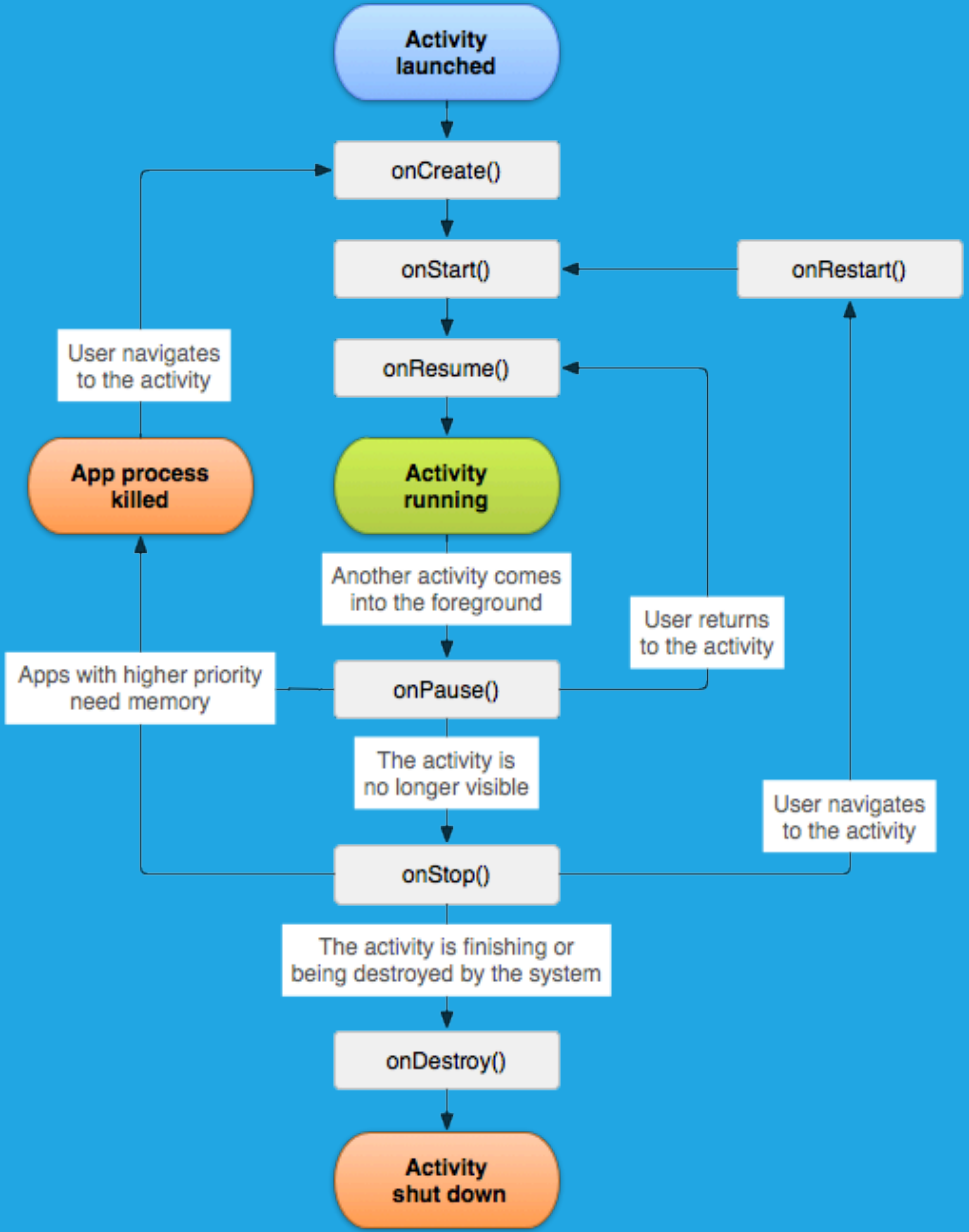
Apps with higher priority need memory

onPause()

The activity is no longer visible

User navigates to the activity

onStop()



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

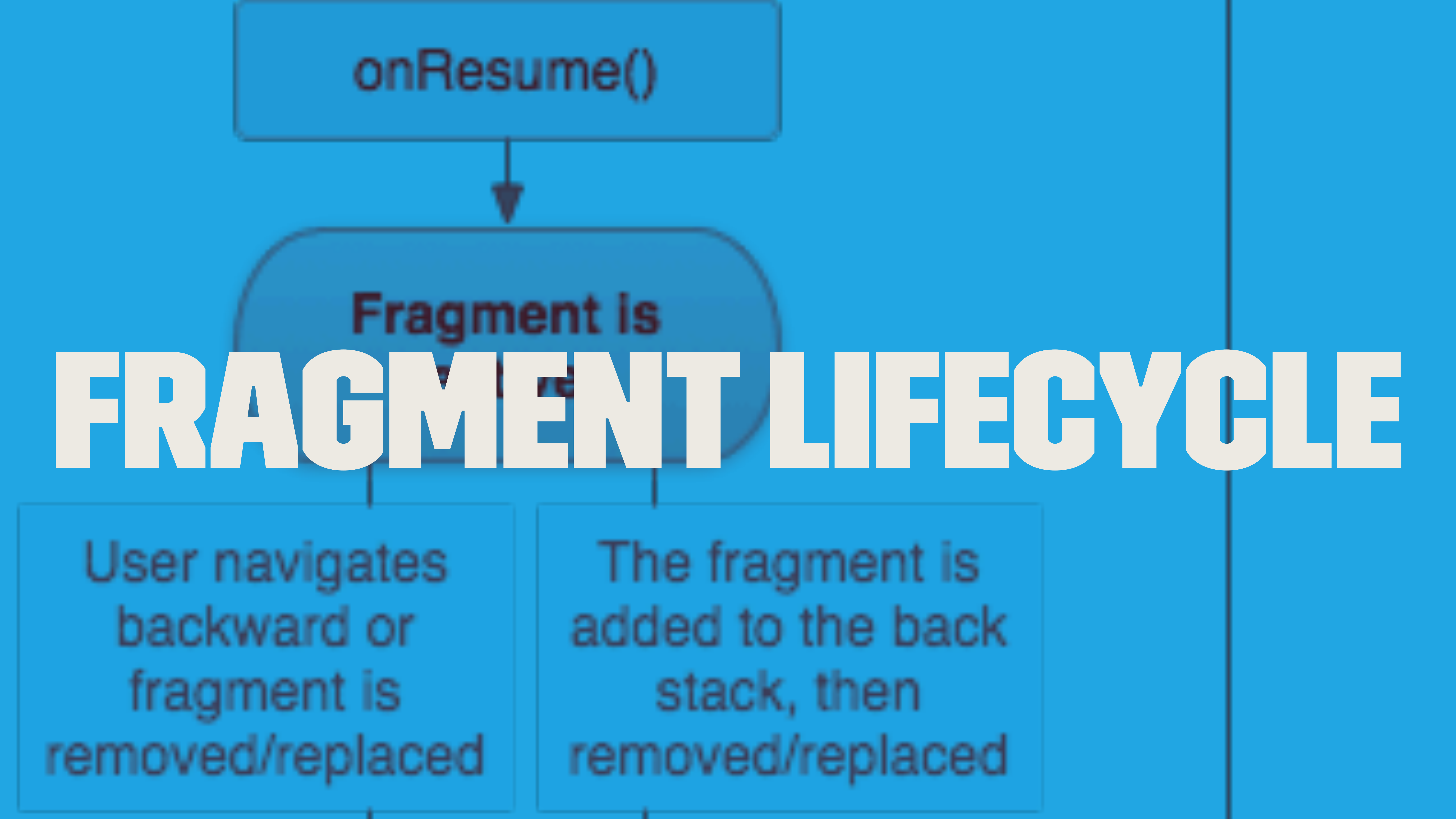
Activities must save their state

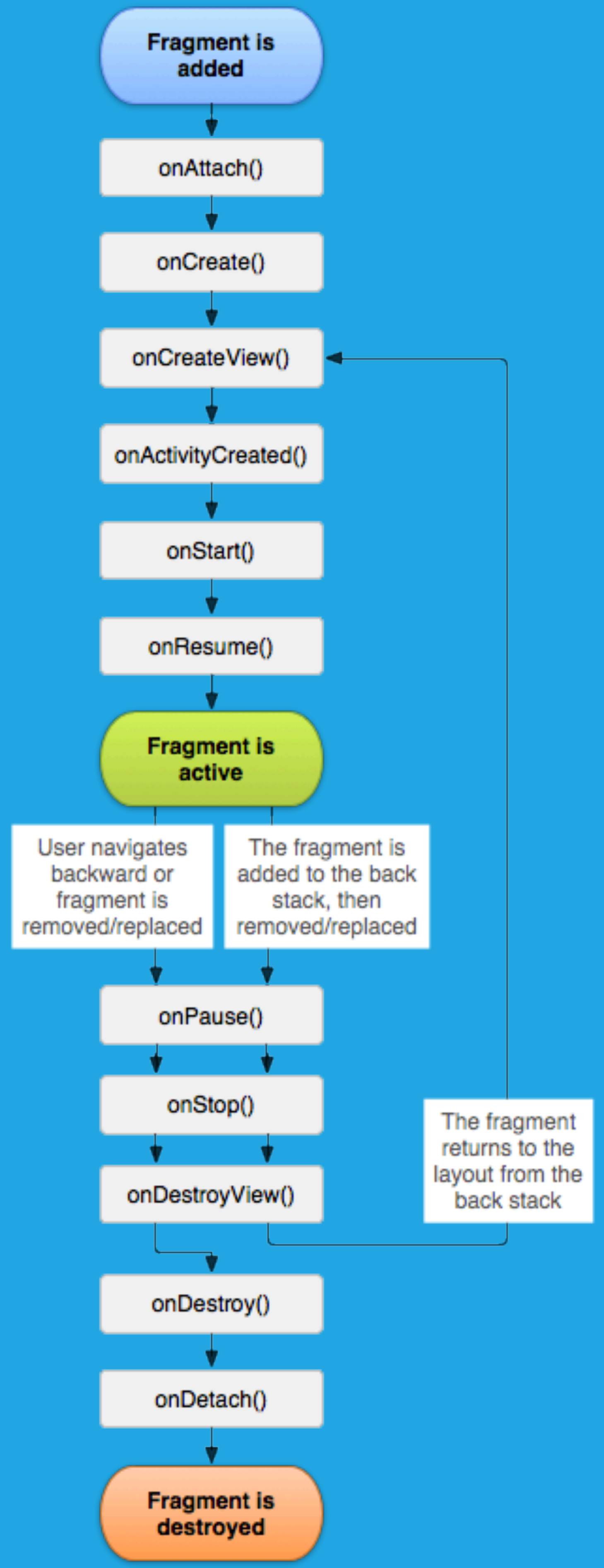
- Place information into the savedInstanceState that you wish to keep and restore it on onCreate()
- Activities can be <u>Destroyed</u> on rotation
  - Dont turn off rotation - it will just hide edge case lifecycle bugs if you do
- Activities and Fragments can be loosely coupled. On creation, look for initialized Fragments before creating them.

EXAMPLE: STORING ACTIVITY DATA

```
public static Intent getTripListActivityIntent(Context context, TripList.LineType lineType) {
    Intent intent = new Intent(context, TripListActivity.class);
    intent.putExtra(TripListActivityState.KEY_ACTIVITY_TRIP_LIST_LINE_TYPE, lineType.getLineName());
    return intent;
}

public static final class TripListActivityState {
    public static final String KEY_ACTIVITY_TRIP_LIST_LINE_TYPE = "KEY_ACTIVITY_TRIP_LIST_LINE_TYPE";
}

TripList.LineType mLineType;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   mLineType = TripList.LineType.getLineType(
        getIntent().getStringExtra(TripListActivityState.KEY_ACTIVITY_TRIP_LIST_LINE_TYPE));
}
```

FRAGMENT LIFECYCLE


onResume()
Fragment is
User navigates backward or fragment is removed/replaced
The fragment is added to the back stack, then removed/replaced



Fragment is added

onAttach()

onCreate()

onCreateView()

onActivityCreated()

onStart()

onResume()

Fragment is active

User navigates backward or fragment is removed/replaced

The fragment is added to the back stack, then removed/replaced

onPause()

onStop()

onDestroyView()

The fragment returns to the layout from the back stack

onDestroy()

onDetach()

Fragment is destroyed

Fragments must save their state

- Fragments have their own savedInstanceState you must use as well.
- Fragments can have Bundled Arguments - this can be a good place to store state data.
- Fragments are created before activities sometimes.
  - Many fragments have Listeners (aka delegates) to their Activities. Don't try to use or reference these before onActivityCreated()*.

EXAMPLE: STORING FRAGMENT DATA

```
/**
 * The configuration flags for the Trip List Fragment.
 */
public static final class TripListFragmentState {
    public static final String KEY_FRAGMENT_TRIP_LIST_LINE_TYPE = "KEY_FRAGMENT_TRIP_LIST_LINE_TYPE";
    public static final String KEY_FRAGMENT_TRIP_LIST_DATA = "KEY_FRAGMENT_TRIP_LIST_DATA";
}
```

```java
/**
 * Use this factory method to create a new instance of
 * this fragment using the provided parameters.
 *
 * @param lineType the subway line to show trips for.
 * @return A new instance of fragment TripListFragment.
 */
public static TripListFragment newInstance(TripList.LineType lineType) {
    TripListFragment fragment = new TripListFragment();
    Bundle args = new Bundle();

    args.putString(TripListFragmentState.KEY_FRAGMENT_TRIP_LIST_LINE_TYPE,
         lineType.getLineName());

    fragment.setArguments(args);
    return fragment;
}
```

```
protected TripList mTripList;

protected void setTripList(TripList tripList) {
    Bundle arguments = this.getArguments();
    arguments.putParcelable(TripListFragmentState.KEY_FRAGMENT_TRIP_LIST_DATA, tripList);
    mTripList = tripList;

    if (mTripArrayAdapter != null) {
        mTripArrayAdapter.clear();
        mTripArrayAdapter.addAll(mTripList.trips);
    }
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (getArguments() != null) {
        mLineType = TripList.LineType.getLineType(
            getArguments().getString(TripListFragmentState.KEY_FRAGMENT_TRIP_LIST_LINE_TYPE));

        mTripList = getArguments().getParcelable(
            TripListFragmentState.KEY_FRAGMENT_TRIP_LIST_DATA);
    }
}
```

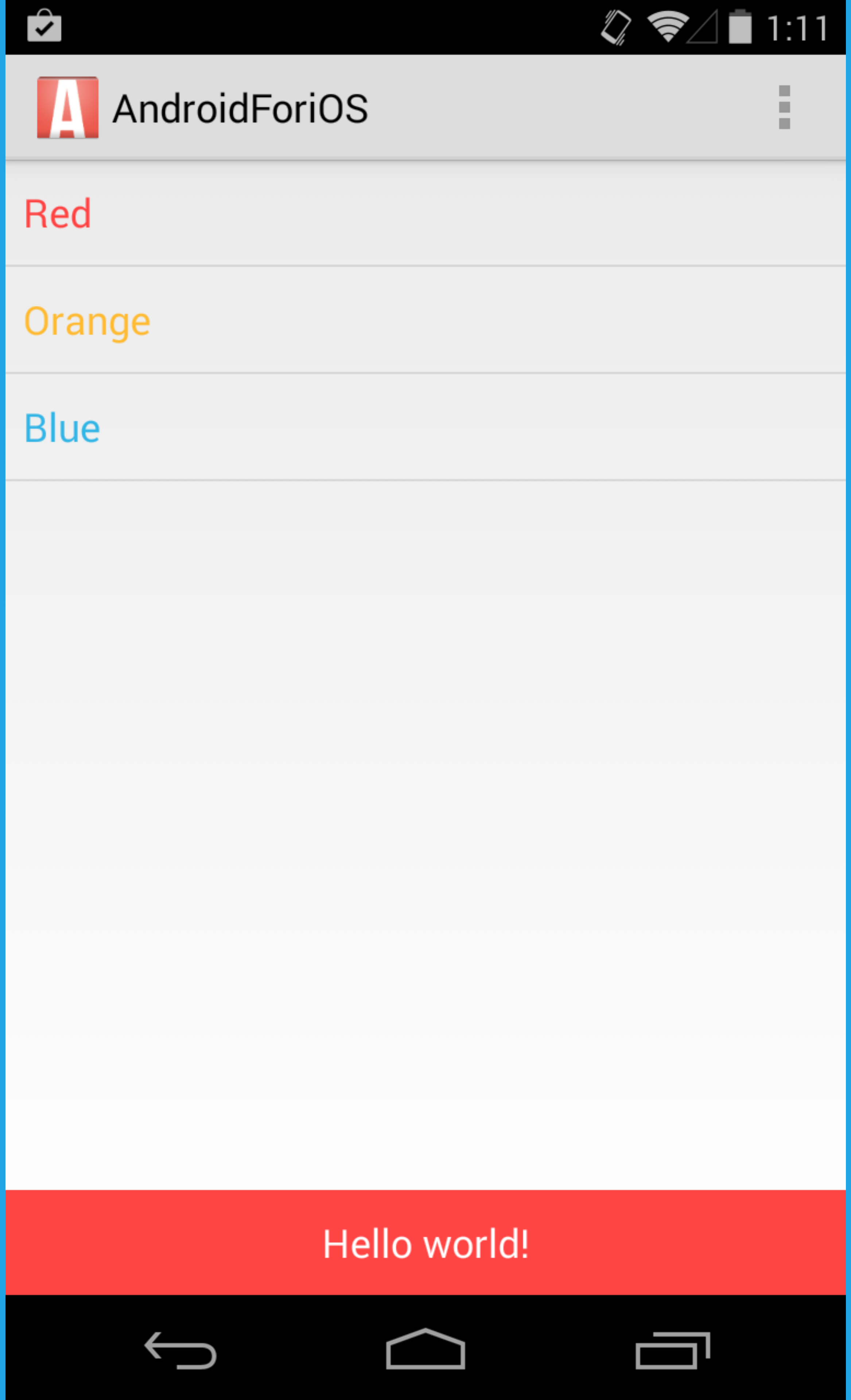
LAYOUTS

IOS IS HEADING MORE AND MORE TOWARDS ANDROID'S LAYOUT STRUCTURE

Layout Basics

- Declare UI elements in XML
- Instantiate layout elements at runtime

SUBWAY LISTVIEW LAYOUT



1:11

AndroidForiOS

Red

Orange

Blue

Hello world!

```
<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="com.example.androidforios.app.activities.MainActivity$PlaceholderFragment">

    <ListView
        android:id="@+id/fragment_subway_list_listview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/Button.Default.Height"/>

    <Button
        android:id="@+id/fragment_subway_list_Button"
        android:layout_width="match_parent"
        android:layout_height="@dimen/Button.Default.Height"
        android:minHeight="@dimen/Button.Default.Height"
        android:background="@drawable/button_red_selector"
        android:text="@string/hello_world"
        android:textColor="@color/Button.Text"
        android:layout_alignParentBottom="true"
        android:gravity="center"/>
</RelativeLayout>
```

LAYOUT TIPS

- Always work in dp (Density-independent Pixels) instead of pixels directly.
- Don't bother nudging items in the visual editor for layouts. Your best bet is to adjust the xml directly. Sorry, this is no interface builder.

DATA

OPTIONS

- Shared Preferences <-> NSUserDefaults
- In memory objects
- Saving to and fetching from file structure via the internal or external file storage <-> saving to the documents directory
- SQLLite <-> Core Data

HOMEWORK

Android Homework

- Learn the UI: Action Bar, Overflow Menu, and the Menu Button
- Cross App Data Sharing
- Intents and Responding to common OS actions
- Java's features: Generics, virtual methods and classes
- Google's Compatability Libraries
- The Android Emulator: Install the x86 HAXM plugin to make the emulator buttery smooth.

Raizlabs Resources

- BaseUtils <-> RZUtils
- WebserviceManager <-> RZWebserviceManager

GO OUT AND CONQUER SOME ANDROID!