WORKING WITH FRAGMENTS

Android Developer Days 2013

Can Elmas

ABOUT ME

- Pozitron, since 2007
- Backend and client side experience
- Software Team Leader at the moment

AGENDA

- Overview
- When to Use?
- Managing Fragments
- Fragment Operations
- Demo(s)

OVERVIEW

- Introduced with Honeycomb (API 11)
- Modular section of an activity / Part of a user interface*
- More dynamic and flexible UI on large screens
- Reusability across activities
- Supported via Compatibility Package

* can also exist without user interface; Headless fragments

OVERVIEW

- Embedded in activities
- Lives in a ViewGroup inside the activity's view hierarchy
- Its own layout and behavior with its own life-cycle

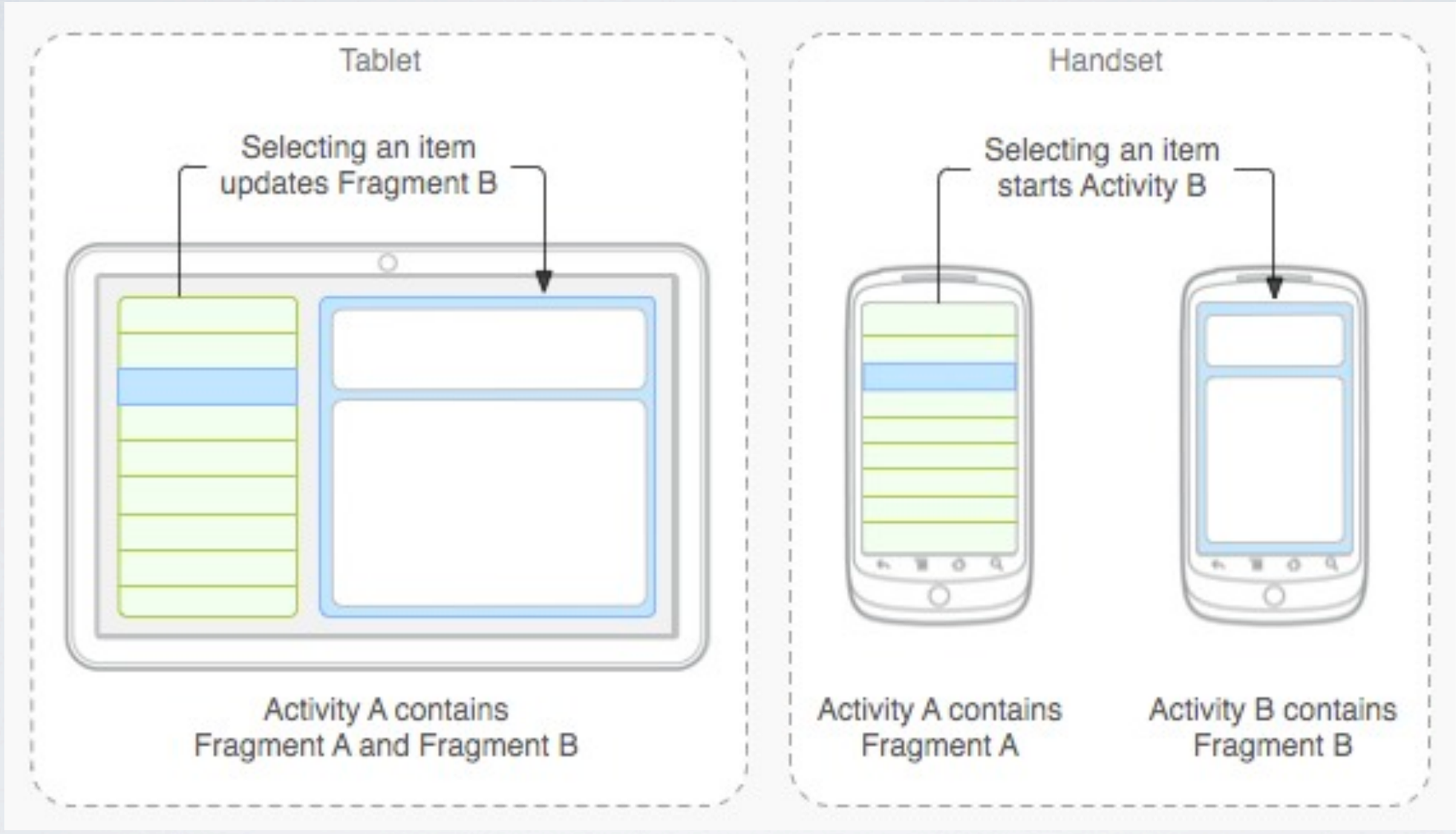
WHEN TO USE?

- Large screens - multi pane, flexible UI

- Combined with other UI widgets, ActionBar, ViewPager, NavigationDrawer etc. - beautiful native app!

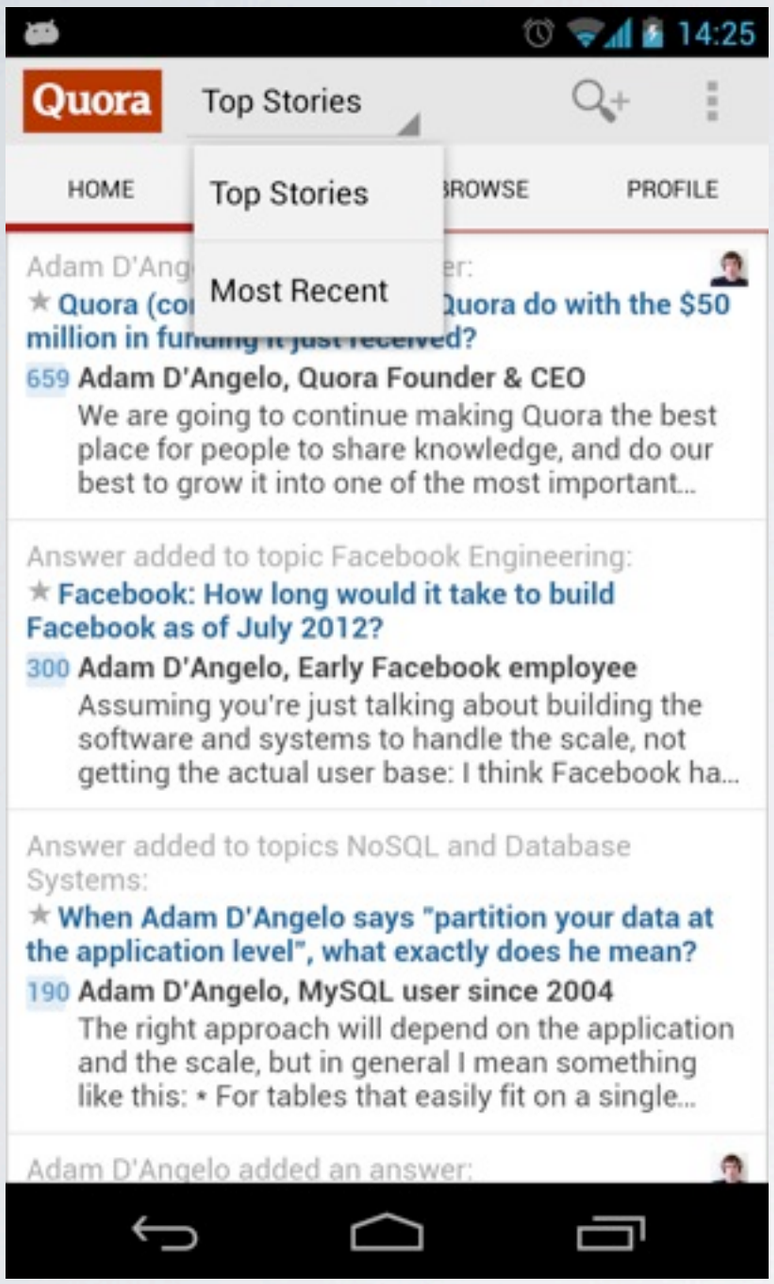
- More flexible, smooth, compact app with less activities; multiple fragments in a single activity

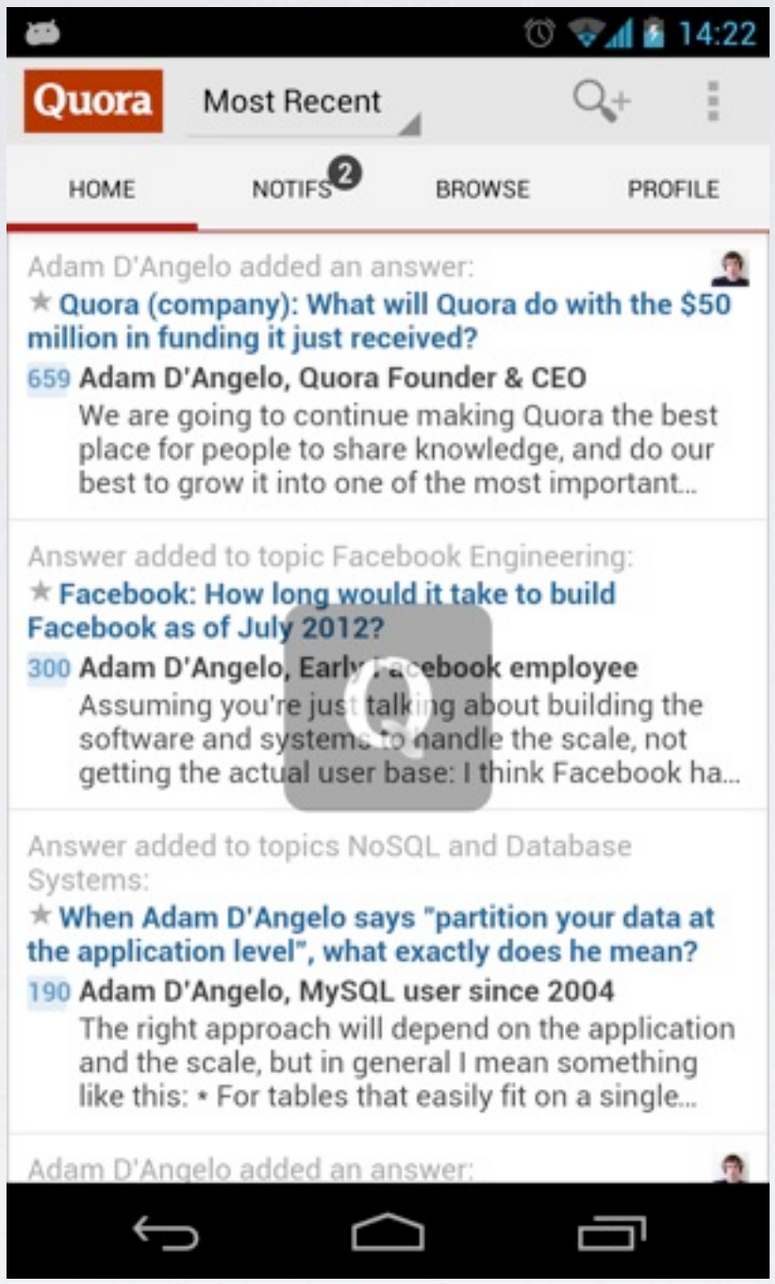
Use Case - Tablet vs Handset

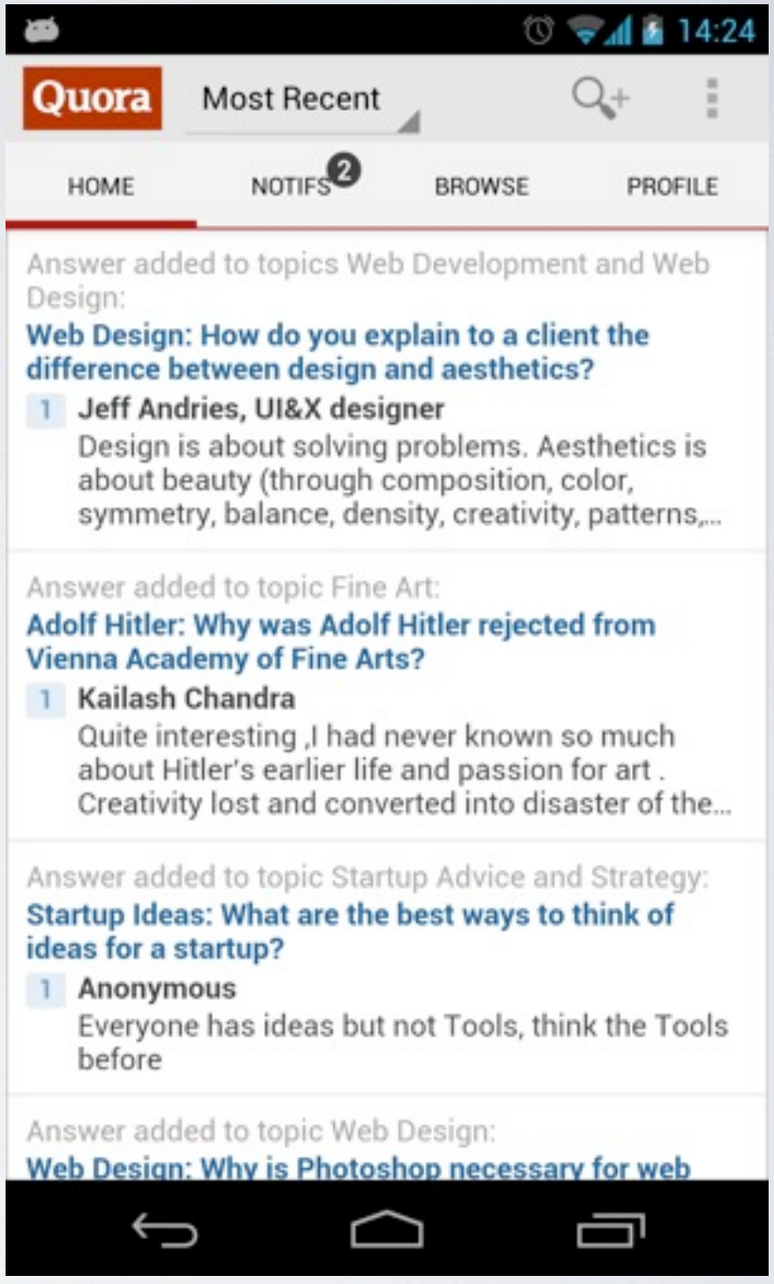


http://developer.android.com/guide/components/fragments.html

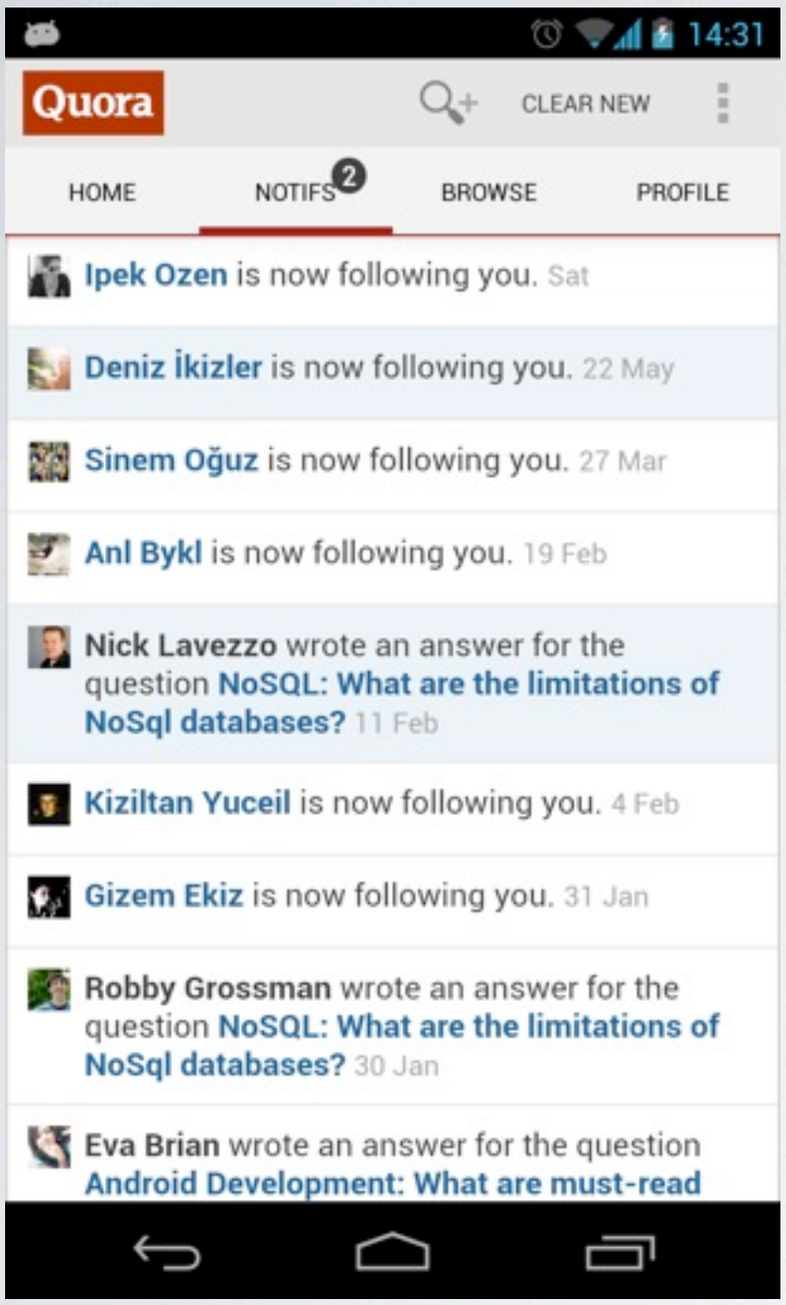
Use Case - Quora App Action Bar Navigation

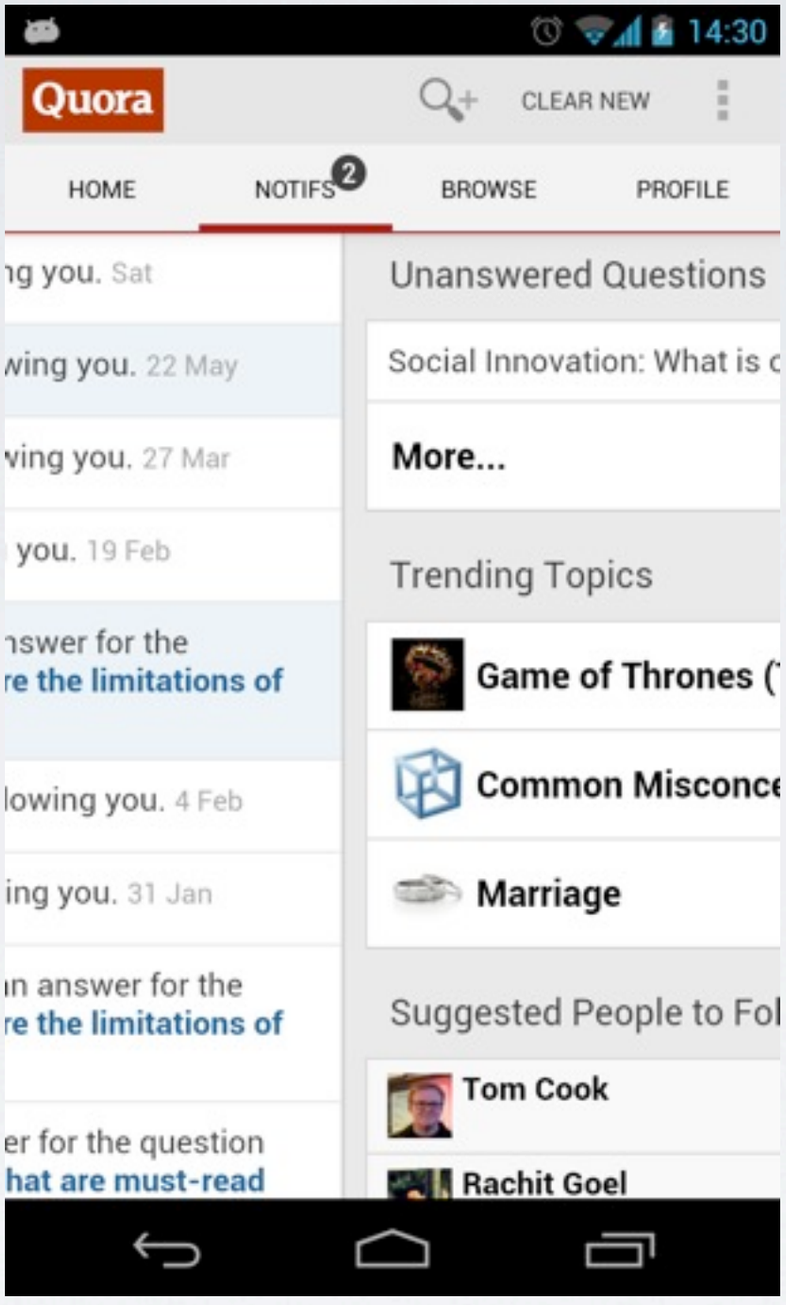


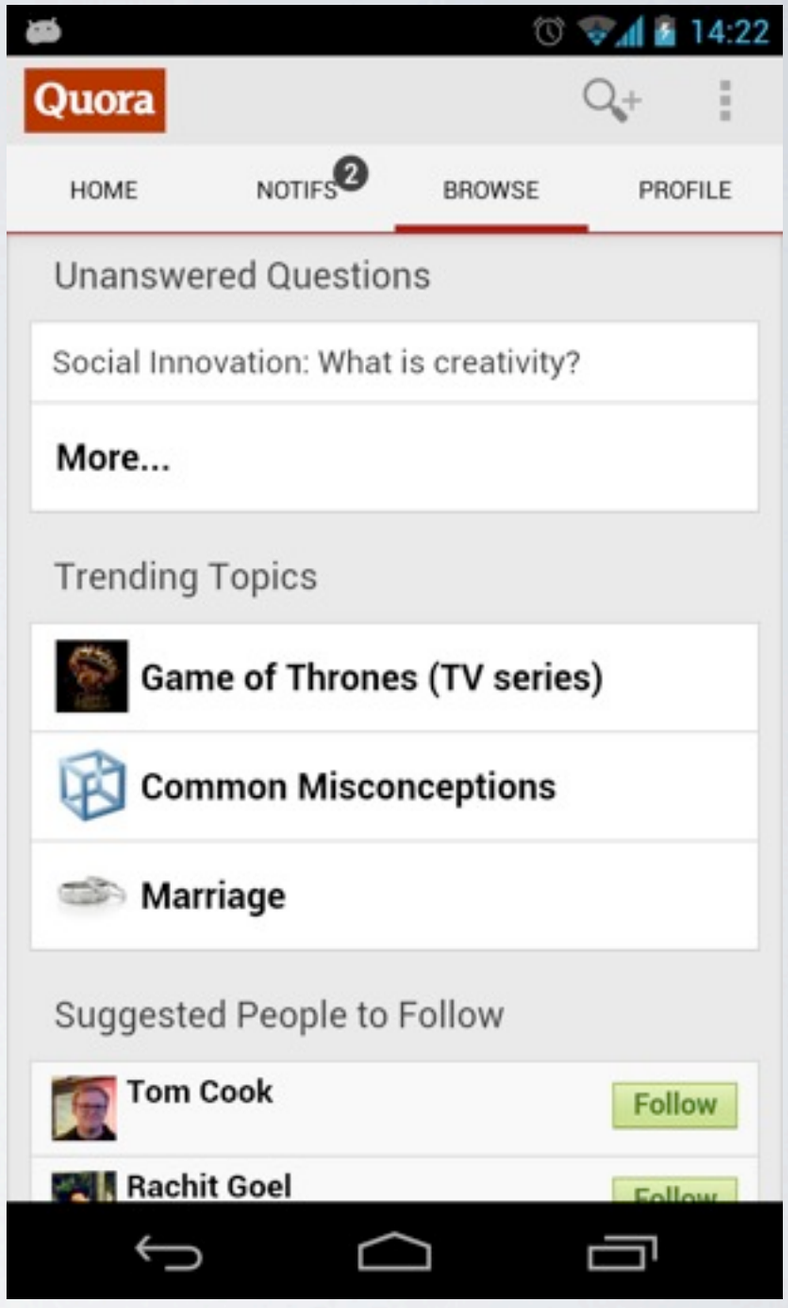




Use Case - Quora App View Pager

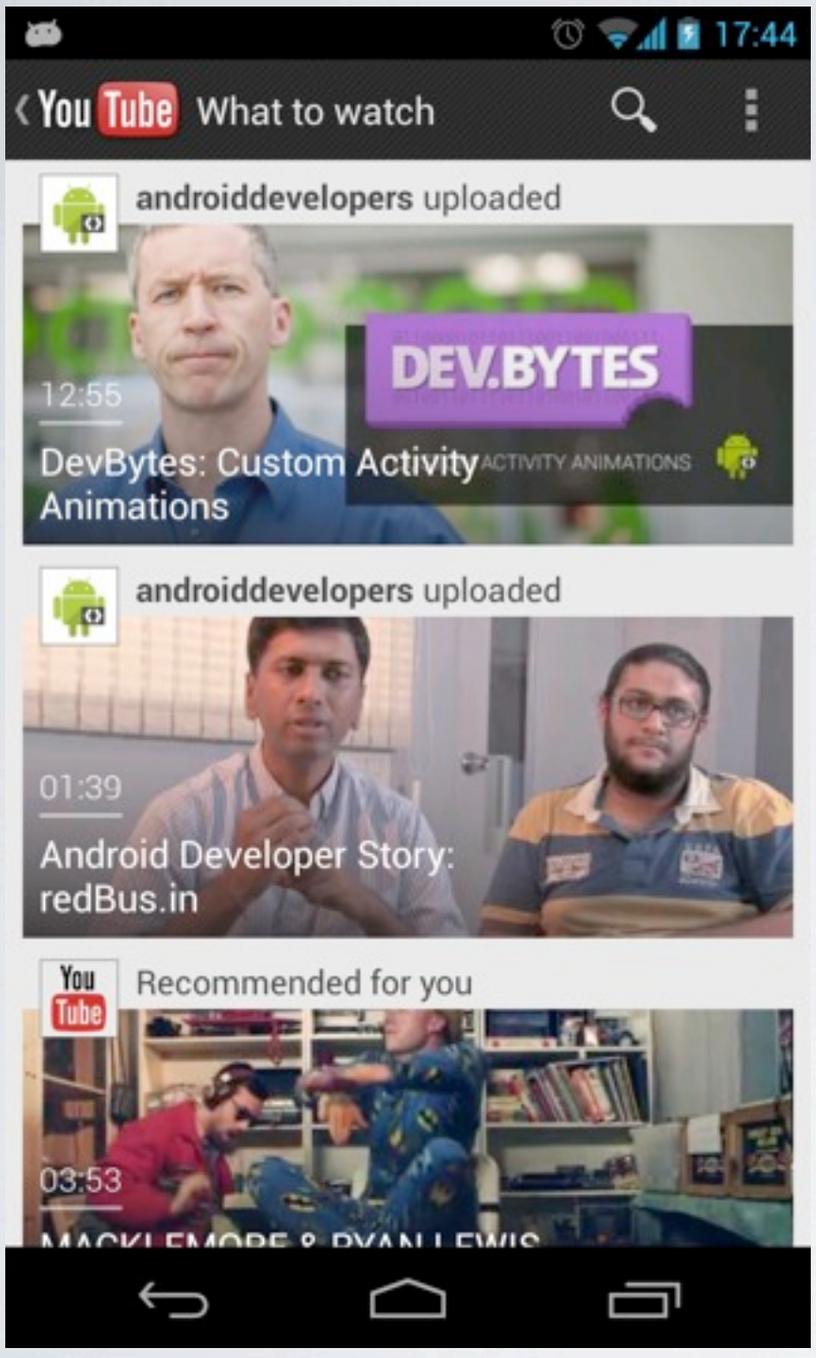


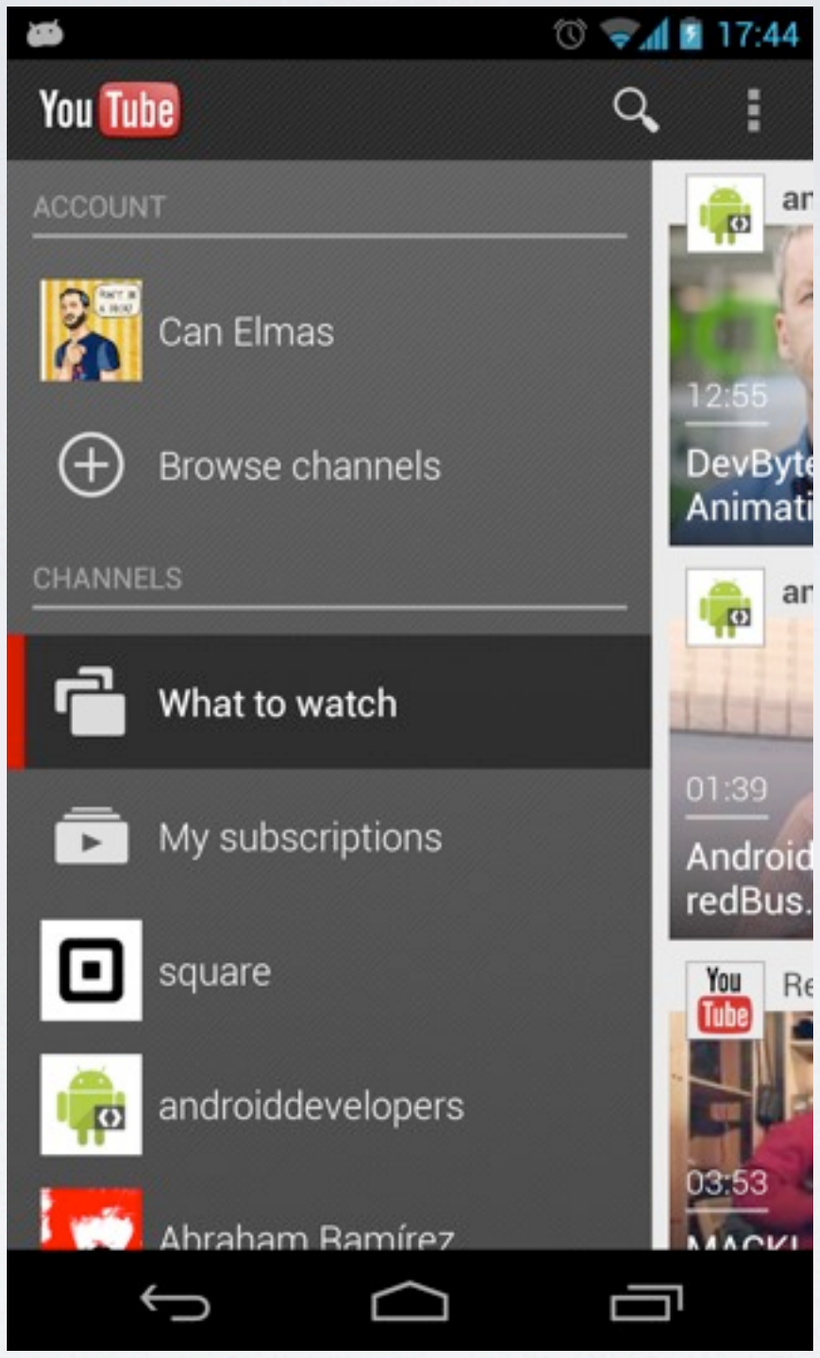


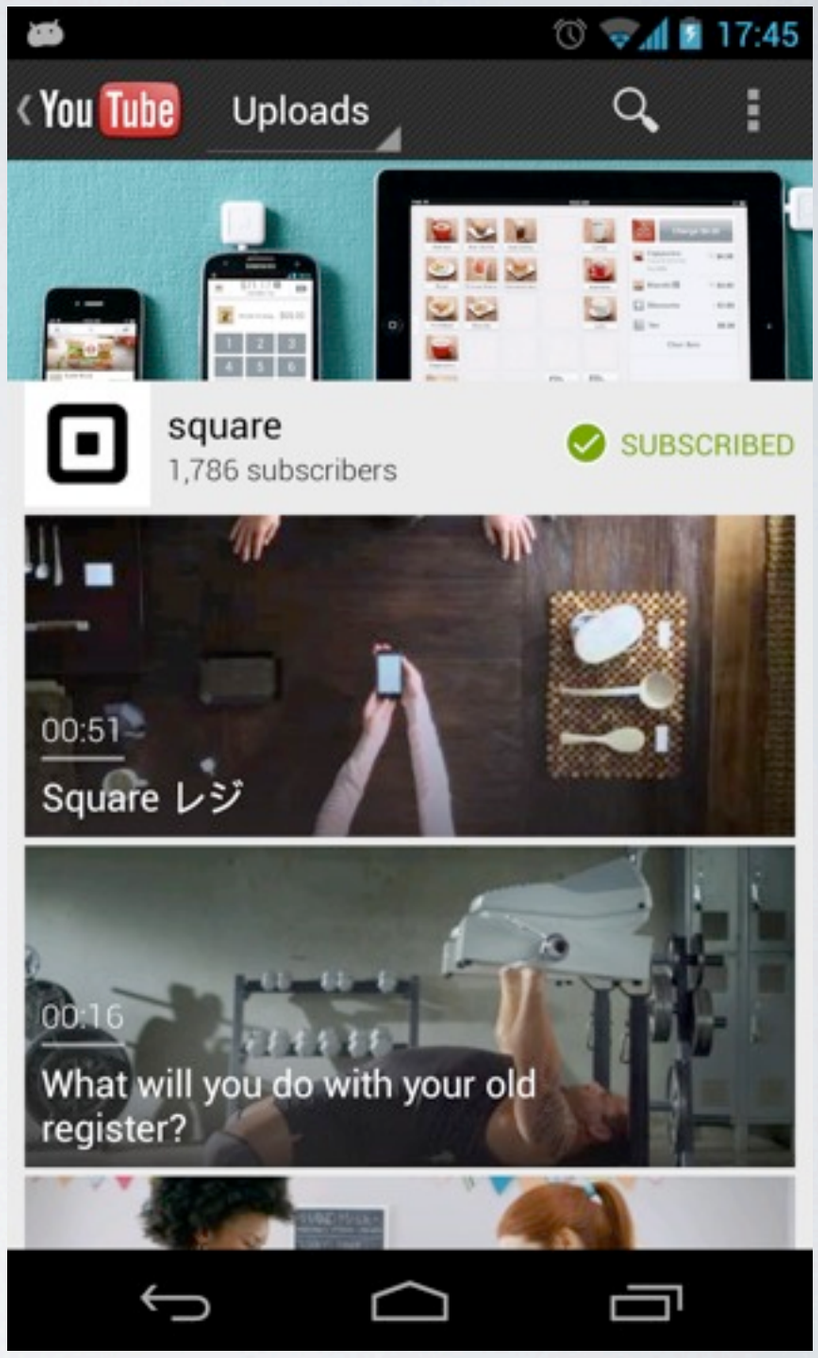


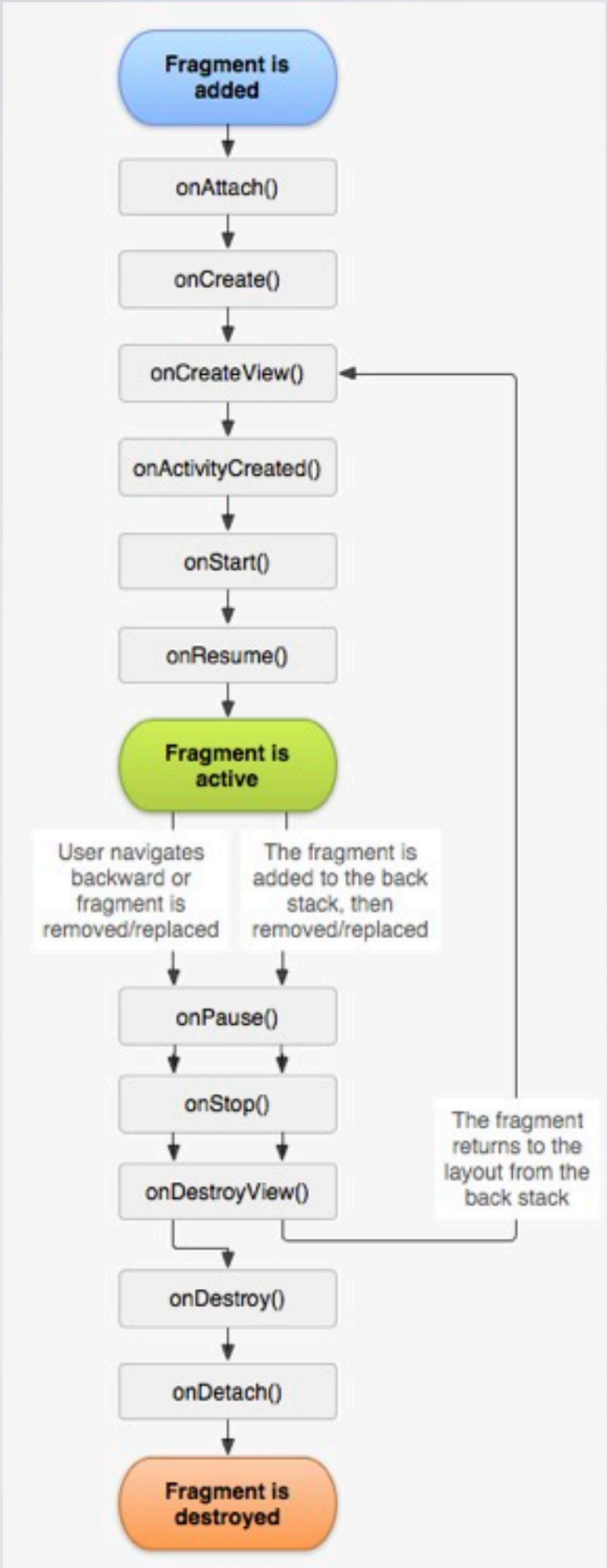
swipe

Use Case - YouTube Navigation Drawer

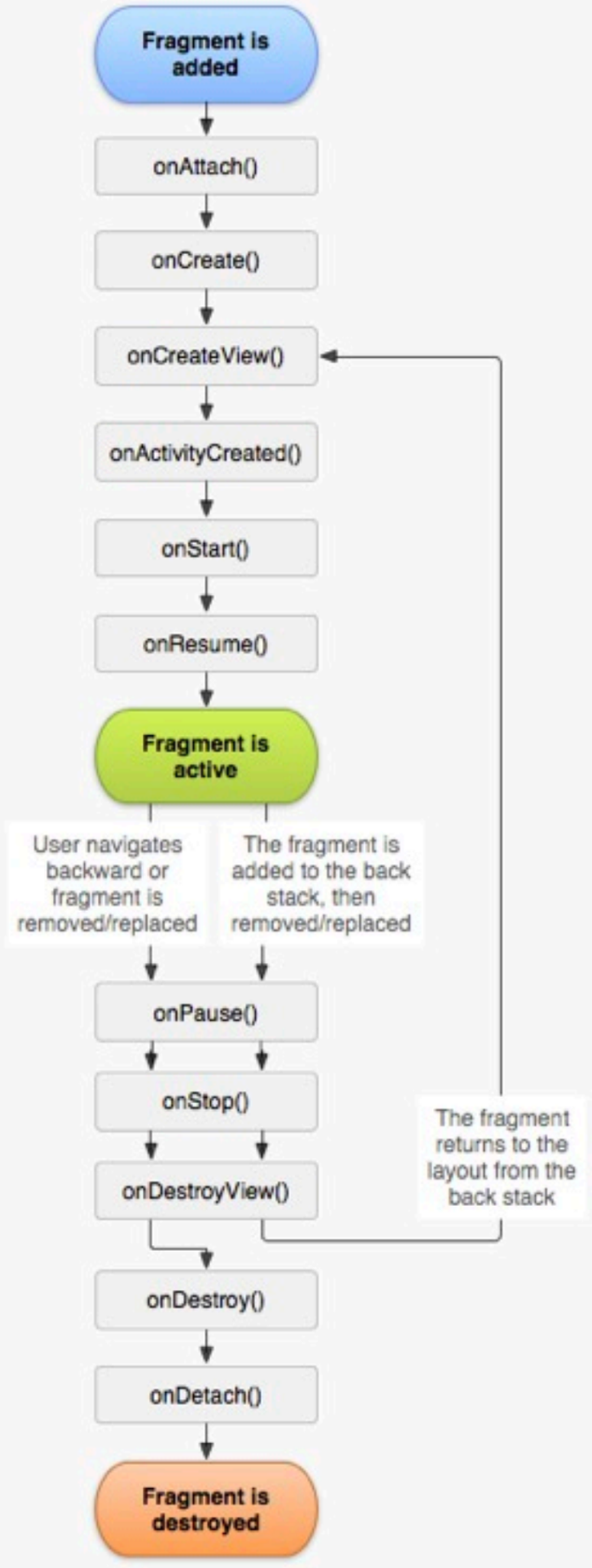







Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
User navigates backward or fragment is removed/replaced
The fragment is added to the back stack, then removed/replaced
onPause()
onStop()
onDestroyView()
The fragment returns to the layout from the back stack
onDestroy()
onDetach()
Fragment is destroyed


Activity State
Fragment Callbacks
Created
onAttach()
onCreate()
onCreateView()
onActivityCreated()
Started
onStart()
Resumed
onResume()
Paused
onPause()
Stopped
onStop()
Destroyed
onDestroyView()
onDestroy()
onDetach()



onCreate(Bundle) called once the fragment is associated with its activity. Activity is still in the process of being created.

Do not depend on activity's layout

onCreateView(LayoutInflater, ViewGroup, Bundle) time to instantiate for the fragment's user interface (optional, return null for non-graphical fragments)

onActivityCreated(Bundle) called once the activity is created and fragment's view is instantiated.

Do final initializations (context dependent instantiations, retrieving views, adding listeners etc.)

onDestroyView() Called when the view has been detached from the fragment. Next time the fragment needs to be displayed, a new view will be created.

MANAGING FRAGMENTS

- ***FragmentManager*** to interact with fragments inside an Activity
  - *Activity.getFragmentManager()*
  - *android.support.v4.app.FragmentActivity.getSupportFragmentManager()*
- ***FragmentTransaction*** to perform fragment operations : add, remove, replace, hide at run-time etc.

ADDING FRAGMENTS - STATIC

- <*fragment*> element in the activity's layout XML
- Preferable if the fragment is consistent in the layout
- Limits run-time fragment operations; can't remove

DEMO

TAKE AWAYS

- Use Support Library for backward compatibility
  - android.support.v4.app.FragmentActivity
  - android.support.v4.app.Fragment
  - FragmentActivity.getSupportFragmentManager()
- Use different layouts to support multiple screen sizes
- Static Instantiation limits run time fragment operations

TAKE AWAYS

- Fragment to Fragment communication via callbacks to the Activity; inner type interfaces defined in fragments
- Avoid tight coupling between activities and fragments
- To access *activity* from *fragment*, *Fragment.getActivity()*
- To access *fragment* from *activity*
  - *FragmentManager.findFragmentById(int id)*
  - *FragmentManager.findFragmentByTag(String tag)*

FRAGMENT OPERATIONS

- **add** - Add a fragment to the activity
- **remove** - Remove an existing fragment; its view is also removed
- **replace** - Remove one fragment from the activity and add another
- **hide** - Hides an existing fragment; its view is hidden
- **show** - Show previously hidden fragment

FRAGMENT TRANSACTIONS

```
FragmentManager fm = getSupportFragmentManager();
FragmentTransaction ft = fm.beginTransaction();
ft.add(R.id.fragment_container, fragment);
ft.commit();
```

- *commit()* resolves in the main thread

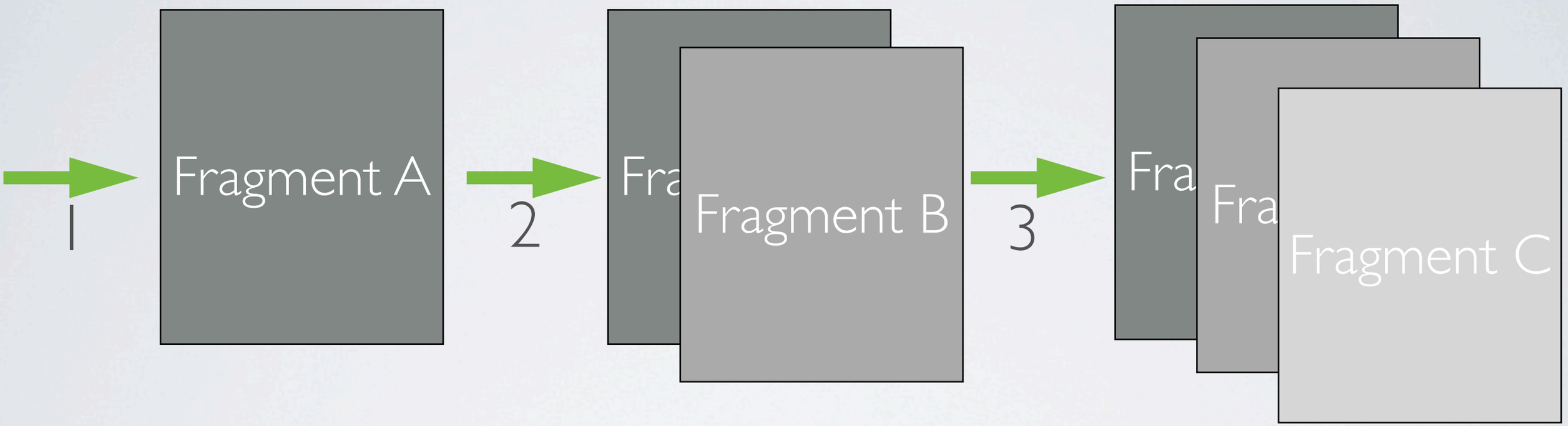
FRAGMENT BACK STACK

- Similar to activity back stack
- Allow user to navigate backward
- Ability to save fragment transaction onto back stack
- Managed by activity on back button press
- Activity destroyed once all fragment transactions removed from the back stack and back button is pressed again

FRAGMENT BACK STACK

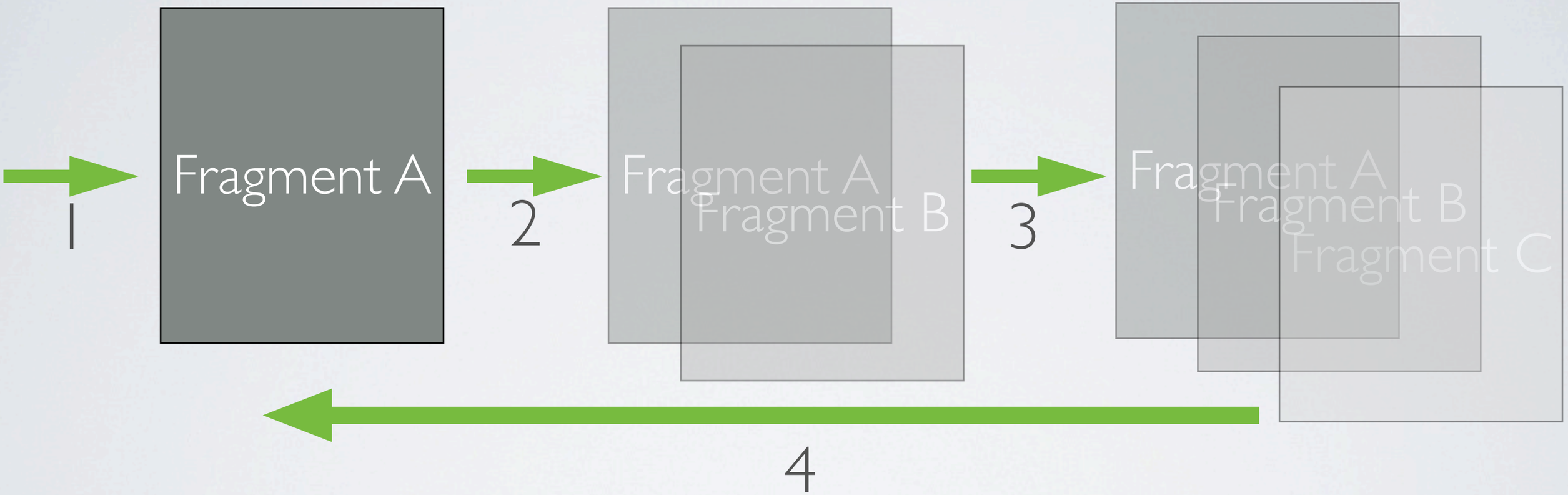
- *FragmentTransaction.addToBackStack(String)*
  - null or optional identifier name for the back stack state
    - Useful for returning to a given state by calling FragmentManager's *popBackStack* methods
- *popBackStack()* : Pop the top state off the back stack
- *popBackStack(String name, int flags)* : pop all states up to the *named* state. Also pops this state if POP_BACK_STACK_INCLUSIVE flag is set.

FRAGMENT BACK STACK



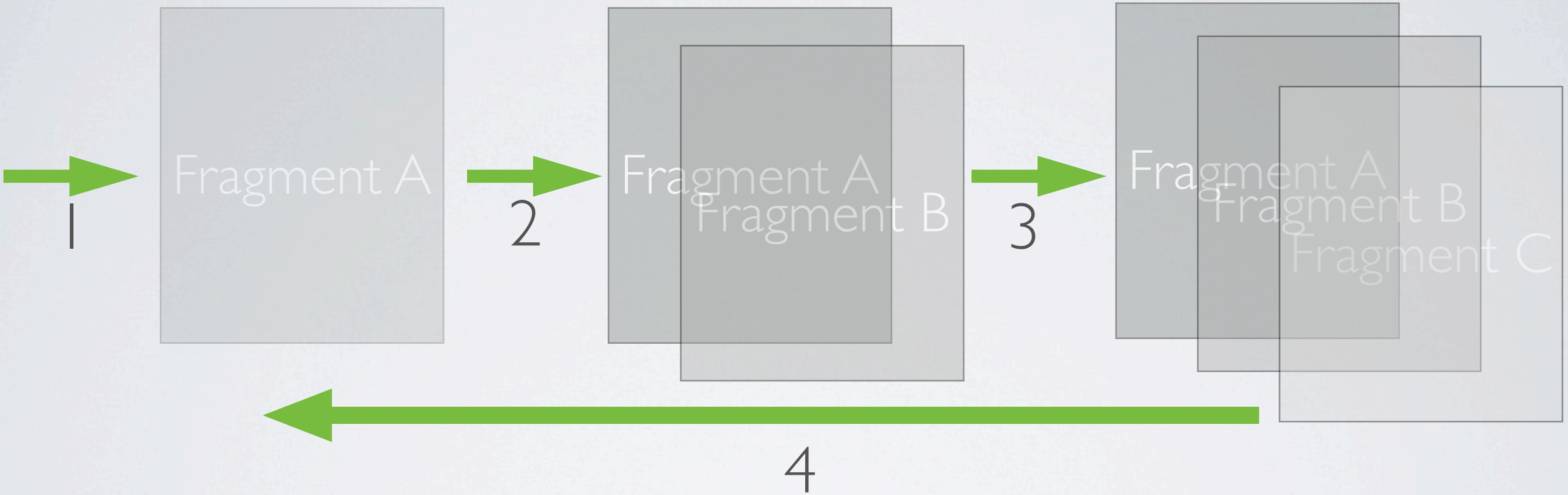
1. ft.add(R.id.fragment_container, fragmentA);
   ft.addToBackStack("fragStackA")
2. ft.add(R.id.fragment_container, fragmentB);
3. ft.add(R.id.fragment_container, fragmentC);

FRAGMENT BACK STACK



4. FragmentManager fm = getSupportFragmentManager();
   fm.popBackStack("fragStackA", 0);

FRAGMENT BACK STACK



4. FragmentManager fm = getSupportFragmentManager();
   fm.popBackStack("fragStackA", POP_BACK_STACK_INCLUSIVE);

DEMO

TAKE AWAYS

- Depending on your requirements choose one of the paths :
  - add add add...
  - replace - navigating back to a removed and back stacked state triggers *onCreateView()* to be called again
  - hide/add - might require keep tracking of current fragment to hide
- re-draw or re-initialize ?

TAKE AWAYS

- Add fragment transactions to back stack when necessary

- Use custom animations for fragment transactions : *FragmentTransaction.setCustomAnimations(enter, exit, popEnter, popExit)*

- ActionBar, ViewPager, MapFragment to provide beautiful, flexible and more native user interfaces

TAKE AWAYS

- ActionBarSherlock (http://actionbarsherlock.com)
  - extension to support library to provide action bar design across all Android versions
  - SherlockFragmentActivity
  - SherlockFragment

TAKE AWAYS

- ViewPager
  - available with support library
    - flip left/right through pages of data
- SupportMapFragment
  - *google_play_service.jar* required
  - google play services should be installed on device

TAKE AWAYS

- Update action bar state (title, options items etc.) from fragments
- "*static factory method*" to initialize fragments
- Pass data to fragments via *Fragment.setArguments(Bundle)* and *Fragments.getArguments()*

```
public static FragSpeaker newInstance(Speaker speaker) {
    FragSpeaker fragment = new FragSpeaker();
    Bundle data = new Bundle();
    data.putSerializable(KEY_SPEAKER, speaker);
    fragment.setArguments(data);
    return fragment;
}
```

FROM HERE

- Nested Fragments (with support library revision 11) - fragments inside fragments; child fragments
- DialogFragment
- PreferenceFragment - similar visual style of system preferences
- Headless Fragments; without UI

Slides and source code

https://github.com/canelmas/add2013

THANK YOU!

can.elmas@pozitron.com
twitter : can_elmas

We are hiring!

ik@pozitron.com
http://www.pozitron.com/career/