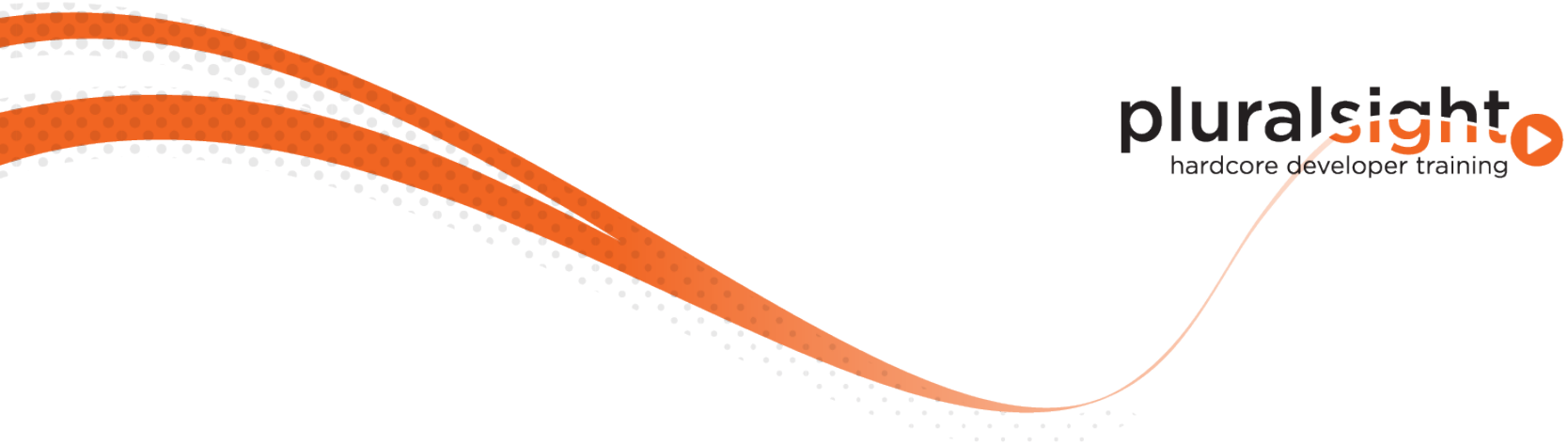
Android Fragment Basics

Jim Wilson

jimw@jwhh.com

@hedgehogjim



pluralsight

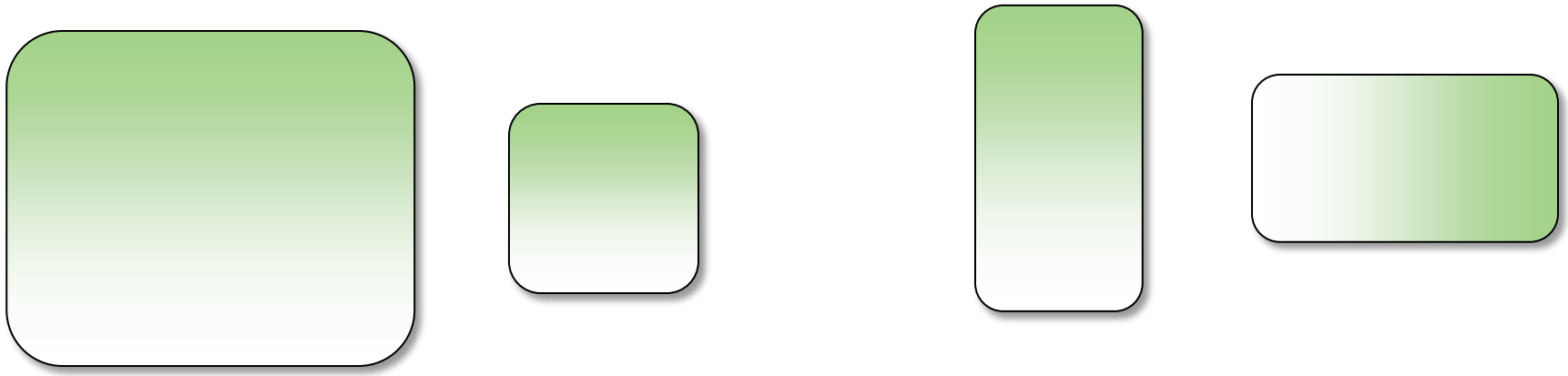
hardcore developer training

Outline

- Fragments and UI modularization
- Creating Fragments
- Coordinating Fragment content
- Maintaining Fragment state
- Supporting Fragments across Activities
- Button click handling and Fragments

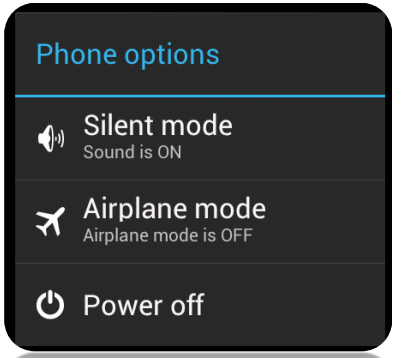
The need for UI modularization

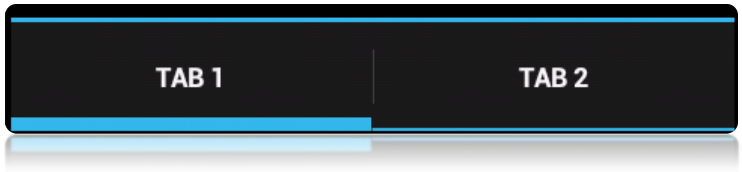
- **Widely varying device experiences**
  - New classes of devices emerging
    - Not just phones anymore
    - Tablet-based computing rapidly gaining acceptance
  - Screen sizes
    - As platform evolves screen resolution and physical size rapidly changing
    - Larger versus smaller screens
    - Portrait versus landscape oriented screens



Fragments are foundation of UI modularization

- **Fragments used throughout Android UI features**
  - Used for making app UIs more adaptable
  - Used for tabbed displays
  - Used for dialog boxes
  - Used for action bar customization
  - And much more …





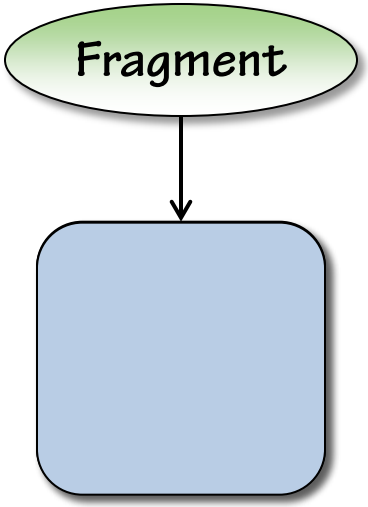
Fragment Availability

- **Fragments supported by 99.8% of active Android devices**
  - Native OS support for devices running Android 3.1 or newer
    - API level 12 and above
  - Compatibility library support for devices running Android 1.6 through 2.3
    - API levels 4 through 10
    - Available from …
      - http://bit.ly/AndroidV4SupportLib
    - Example use …
      - http://bit.ly/AndroidFragmentCompat

*Although Android 3.0 (API Level 11) technically exists – no devices in the marketplace run it

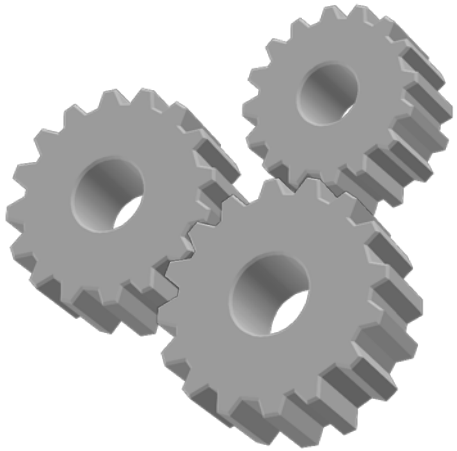
Creating Fragments

- **Creating a Fragment requires a few simple steps**
  - Declare a class that inherits from Fragment class
  - Provide the fragment's display contents
    - Involves overriding onCreateView and/or onCreate
    - Often uses an XML-based layout description much like an Activity
    - Specialized Fragment-derived classes simplify special cases
      - More to come on this
  - Attach your fragment to an Activity
    - Use the <fragment> element in the Activity's XML layout
      - The "class" attribute identifies the fully qualified name of fragment class to create
    - You can attach as many fragments to a view as you'd like



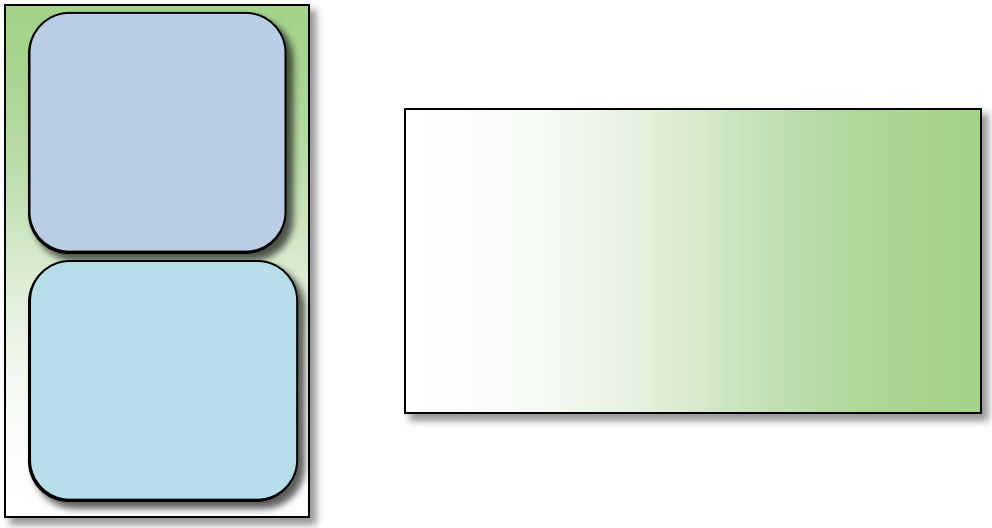
Fragments and UI flexibility

- **Challenging to create apps that support different device experiences**
  - Appropriate user interface experiences very different
  - Historically burden of adaptability has been on individual app developers
- **Fragments simplify supporting differing devices**
  - Goal of simplification is adaptability
    - Automate adaptability when reasonable
    - Allow developers to adapt app behavior with less work



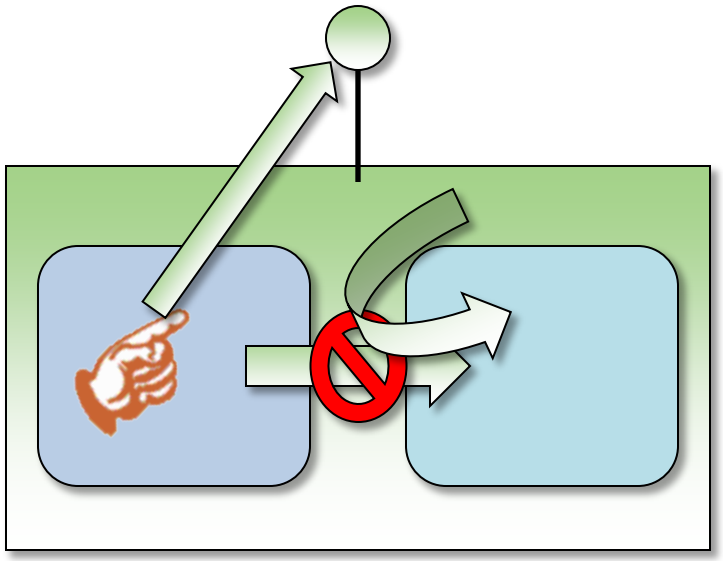
Fragments and UI management

- **A Fragment is a portion of a user interface**
  - Allows you to design your UI in sections
    - Each Fragment can be a self-contained portion of the UI
  - Fragments are then placed on an Activity
    - Fragments can be arranged as needed for different device types



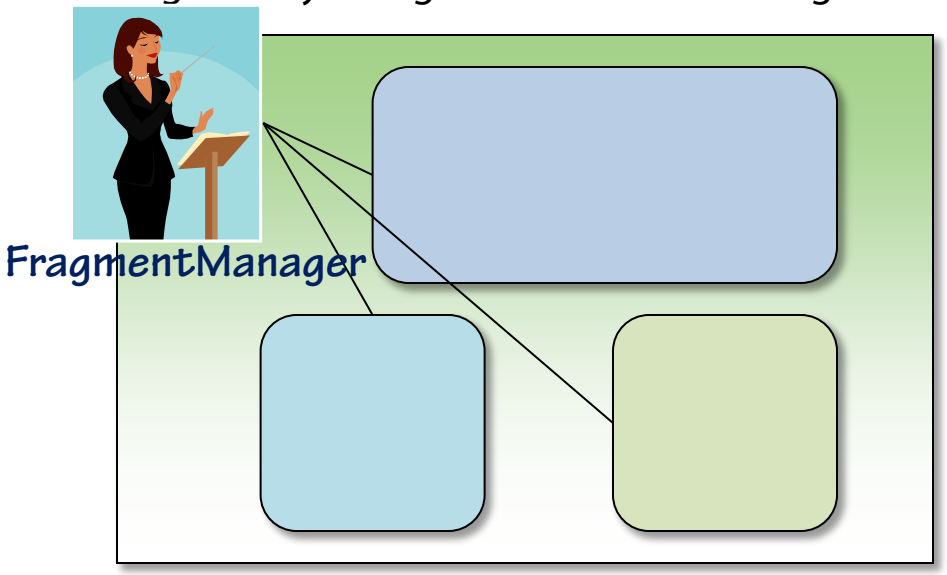
Coordinating fragment content

- **Content of one fragment often affected by user actions in another**
  - Need a way to coordinate fragment content
  - Avoid tightly coupling fragments
  - Best approach is to use an application-defined interface
    - Activity containing the fragment often implements interface
    - Fragments can use interface to notify containing activity of changes
      - Fragment can access Activity through Fragment.getActivity method



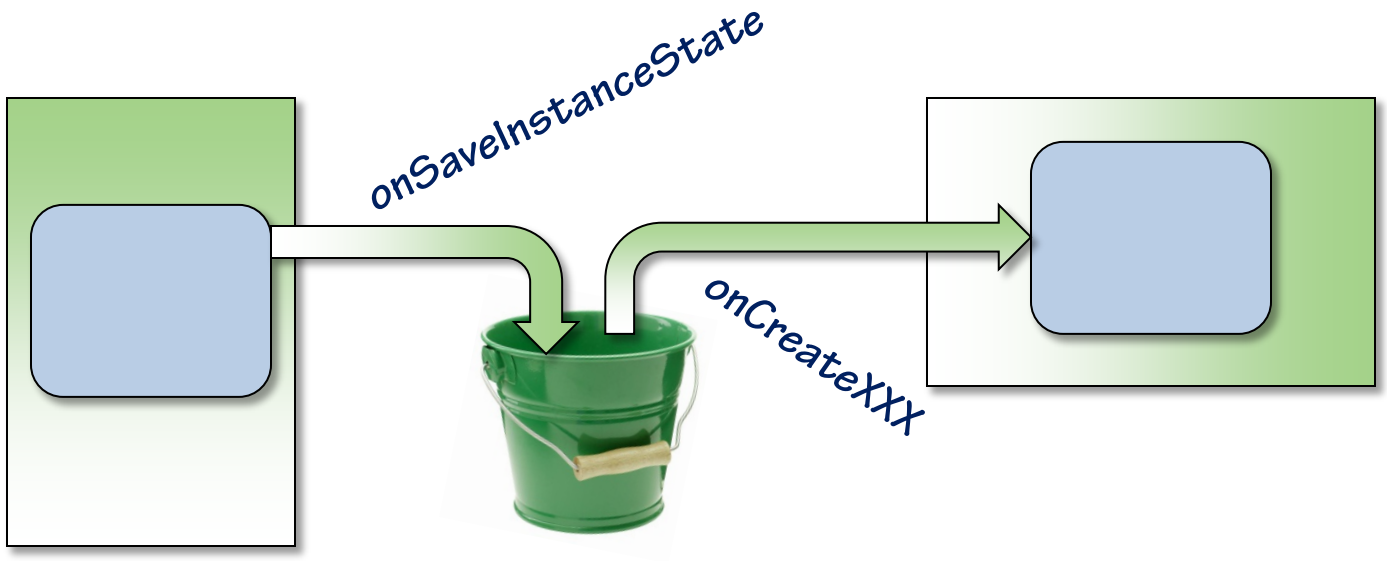
Interacting with fragments

- **The FragmentManager is responsible for fragments w/in an activity**
  - Each activity has it's own FragmentManager
    - Access by calling getFragmentManager
  - Handles details of creating, showing, hiding fragments
  - Holds references to all fragments on the activity
    - Use findFragmentById to get a reference to a fragment instance



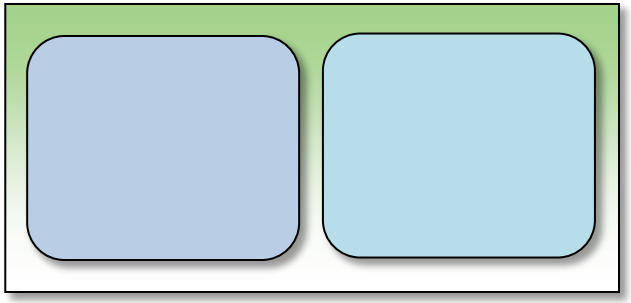
Maintaining fragment state

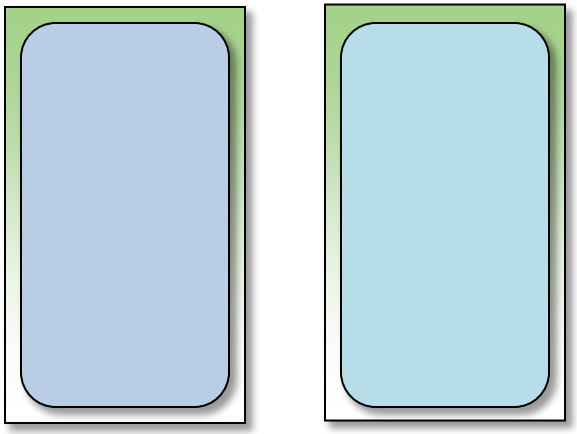
- **Changes in UI layout will often create new fragment instances**
  - New fragment instances may not show the same content as previously
    - Can cause confusion for users
  - You are responsible to update any new instances to match previous ones
    - Use onSaveInstanceState to store current state of a fragment
    - Saved state passed to the new instance's onCreate and onCreateView methods



Supporting Fragments across Activities

- **Applications often must take direct control over layout management**
  - Android built-in layout management OK for basic scenarios
  - Application may need to supplement built-in capabilities
    - May need to switch between single-activity and multiple-activity views
    - In some cases may wish to directly manage hiding/showing individual fragments
  - FragmentManager useful in determining which fragments currently displayed
    - Just because a fragment reference returned doesn't mean it's visible
    - Need to also check fragment's isVisible method





Button click handling and Fragments

- **The Button onClick attribute makes Fragment encapsulation difficult**
  - The onClick attribute requires the handler method to be on the Activity
    - Completely bypasses the Fragment
  - Use explicit OnClickListener interface instead
    - Implement interface on Fragment-derived class
    - Explicitly associate the Fragment with each Button

```
<LinearLayout>
  <RadioButton
    android:id="@+id/myRadioButton"
    android:onClick="onTheButtonClicked"
   />
</LinearLayout>
```

Summary

- **Fragments are the foundation of UI modularization**
- **Fragments support by over 99.8% of active Android devices**
- **Fragments can automatically adapt to many UI differences**
- **FragmentManager handles all Activity interaction with Fragments**
  - Every Activity has its own FragmentManager
- **Use onSaveInstanceState &onCreateXXX callbacks to maintain state**
- **Avoid use of onClick attribute in Fragment layout files**