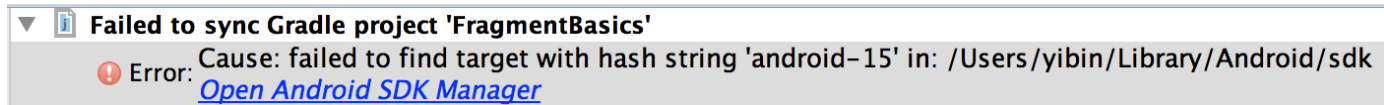


Fragments and Layouts

Download the Sample Code

- <http://developer.android.com/shareables/training/FragmentBasics.zip>
- You may get an error after importing the project



Click to install the SDK

Default Preferences

Appearance & Behavior > System Settings > Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/yibin/Library/Android/sdk [Edit](#)

SDK Platforms | SDK Tools | SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
<input type="checkbox"/> Android N Preview	N	1	Not installed
<input checked="" type="checkbox"/> Android 6.0	23	2	Installed
<input type="checkbox"/> Android 5.1.1	22	2	Not installed
<input type="checkbox"/> Android 5.0.1	21	2	Not installed
<input type="checkbox"/> Android 4.4W.2	20	2	Not installed
<input type="checkbox"/> Android 4.4.2	19	4	Not installed
<input type="checkbox"/> Android 4.3.1	18	3	Not installed
<input type="checkbox"/> Android 4.2.2	17	3	Not installed
<input type="checkbox"/> Android 4.1.2	16	5	Not installed
<input checked="" type="checkbox"/> Android 4.0.3	15	5	Not installed
<input type="checkbox"/> Android 2.3.3	10	2	Not installed
<input type="checkbox"/> Android 2.2	8	3	Not installed

What is a Fragment?


- A Fragment represents a portion of user interface in an Activity
- Combining multiple fragments in a single activity makes you able to
 - Build a multi-pane UI
 - Reuse a fragment in multiple activities



Create a Fragment

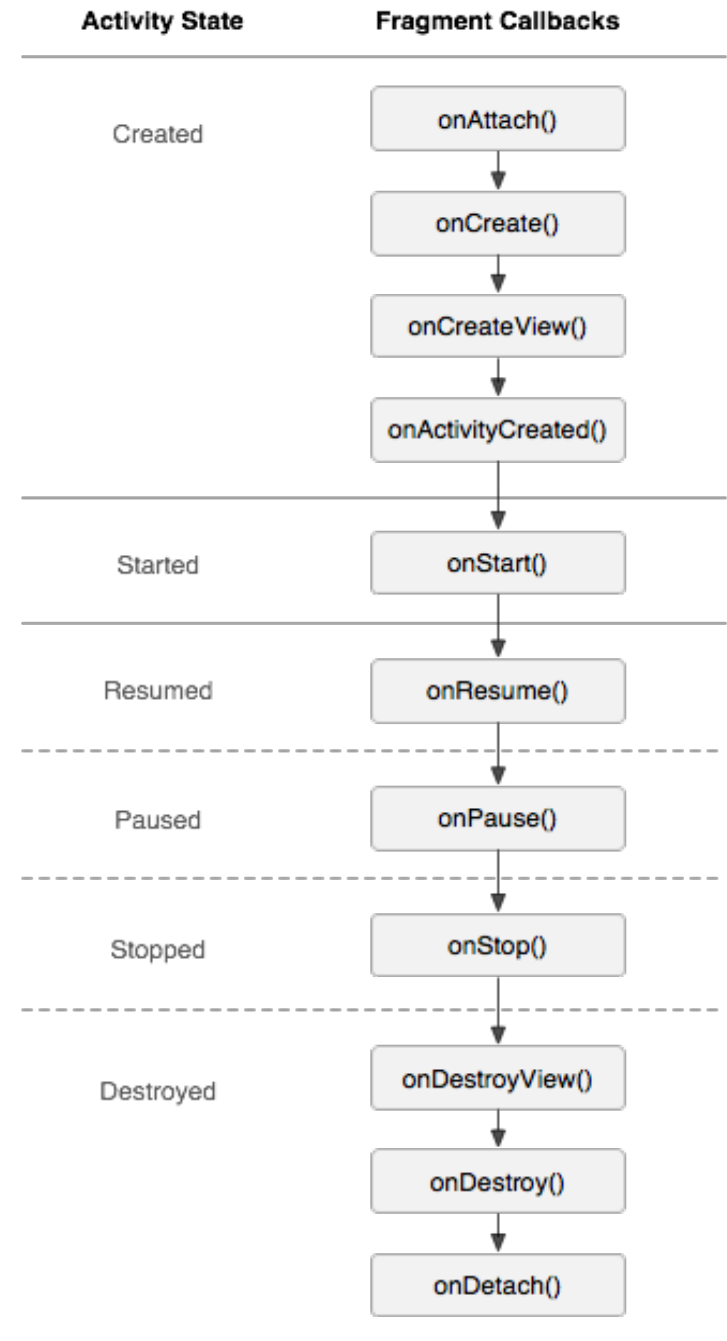
- To create a fragment, extend the Fragment class, then override key lifecycle methods to insert your app logic

```
public class ArticleFragment extends Fragment {           From the activity's layout
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) { // Inflate the layout for this fragment
        return inflater.inflate(R.layout.article_view, container, false);
    }
}
```



Fragment Lifecycle

- A fragment needs to be attached on an activity
- Fragments can be added in “onCreate” state of the activity
- While the state of the activity is changed, the attached fragment(s) will switch to the same state



Adding a Fragment Statically Using XML

A view of your main activity



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <fragment android:name="com.example.android.fragments.HeadlinesFragment"
        android:id="@+id/headlines_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <fragment android:name="com.example.android.fragments.ArticleFragment"
        android:id="@+id/article_fragment"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

Class name of you fragment



Add a Fragment at Runtime

- Adding fragments at runtime is that your activity layout must include a View in which you can insert the fragment

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/fragment_container"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

The Workflow of Adding a Fragment

- Inside your activity, call [getFragmentManager\(\)](#) to get a [FragmentManager](#)
- Call [beginTransaction\(\)](#) to create a [FragmentTransaction](#) and call [add\(\)](#) to add a fragment
- Multiple fragment transactions can be included in the same [FragmentTransaction](#)
- When you're ready to make the changes, you must call [commit\(\)](#)

An Example of Adding a Fragment

```
public class MainActivity extends FragmentActivity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.news_articles);  
  
        // Check that the activity is using the layout version with the fragment_container FrameLayout  
        if (findViewById(R.id.fragment_container) != null) {  
            // Create a new Fragment to be placed in the activity layout  
            HeadlinesFragment firstFragment = new HeadlinesFragment();  
  
            // Add the fragment to the 'fragment_container' FrameLayout  
            getSupportFragmentManager().beginTransaction().add(R.id.fragment_container, firstFragment).commit();  
        }  
    }  
}
```

Replacing a Fragment

- To allow the user to navigate backward through the fragment transactions, you must call [addToBackStack\(\)](#) before you commit the [FragmentManager](#)
- The fragment that is removed is stopped (not destroyed) if you add it to the back stack
 - If the user navigates back to restore the fragment, it restarts
 - If you do not add the transaction to the back stack, then the fragment is destroyed when removed or replaced

An Example of Replacing a Fragment

```
// Create fragment and give it an argument specifying the article it should show
ArticleFragment newFragment = new ArticleFragment();
Bundle args = new Bundle();
args.putInt(ArticleFragment.ARG_POSITION, position);
newFragment.setArguments(args);
FragmentManager transaction = getSupportFragmentManager().beginTransaction();
// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack so the user can navigate back
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);
// Commit the transaction
transaction.commit();
```

Communication between Fragments

- All **Fragment-to-Fragment** communication is done through the **associated Activity**
- In Java if we want to decouple two classes we can use an interface
- Fragments in Android can be regarded as separated classes
- So, we use interface to implement a callback function here

A Fragment with Callback

```
public class HeadlinesFragment extends ListFragment {  
    OnHeadlineSelectedListener mCallback;  
  
    public interface OnHeadlineSelectedListener {  
        // Container Activity must implement this interface  
        public void onArticleSelected(int position);  
    }  
}
```

Pass Messages to The Activity

- For example, the following method in the fragment is called when the user clicks on a list item
- The fragment uses the callback interface to deliver the message to the parent activity

@Override

```
public void onItemClick(ListView l, View v, int position, long id) {  
    // Send the event to the host activity  
    mCallback.onArticleSelected(position);  
}
```

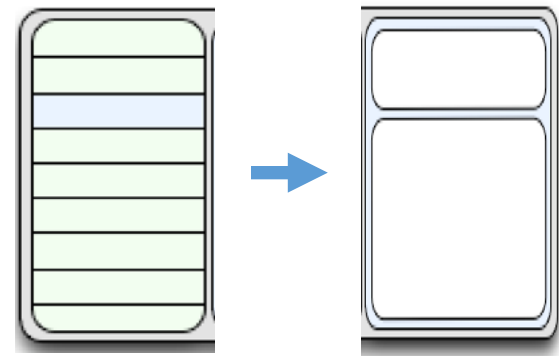
Trigger Another Fragment

The function called by the fragment is implemented in your main activity

```
public static class MainActivity extends Activity
    implements HeadlinesFragment.OnHeadlineSelectedListener{
    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article
        ArticleFragment articleFrag = (ArticleFragment)
            getSupportFragmentManager().findFragmentById(R.id.article_fragment);
        if (articleFrag != null) {
            // If article frag is available, we're in two-pane layout...
            // Call a method in the ArticleFragment to update its content
            articleFrag.updateArticleView(position);
        }
    }
}
```

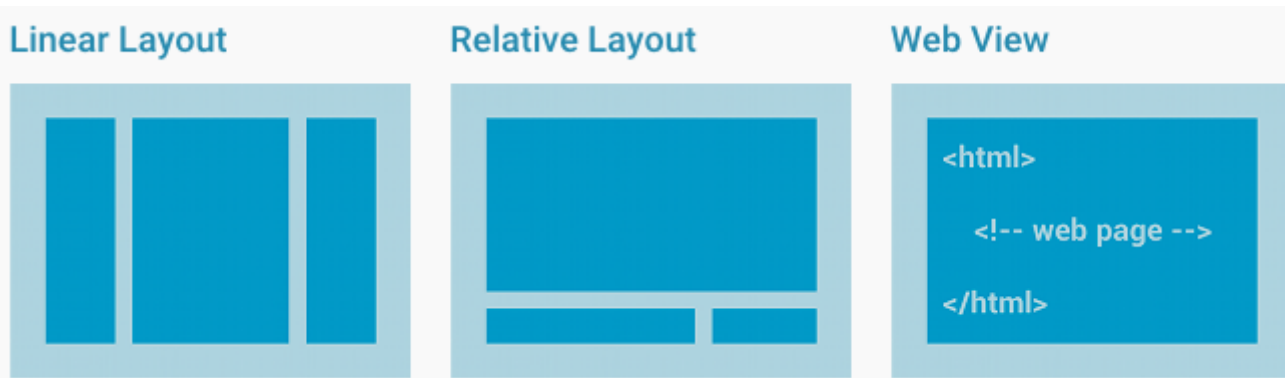


```
else {  
    // Otherwise, we're in the one-pane layout and must swap frags...  
    // Create fragment and give it an argument for the selected article  
    ArticleFragment newFragment = new ArticleFragment();  
    Bundle args = new Bundle();  
    args.putInt(ArticleFragment.ARG_POSITION, position);  
    newFragment.setArguments(args);  
    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();  
    // Replace whatever is in the fragment_container view with this fragment,  
    // and add the transaction to the back stack so the user can navigate back  
    transaction.replace(R.id.fragment_container, newFragment);  
    transaction.addToBackStack(null);  
    transaction.commit(); // Commit the transaction  
}  
}  
}
```



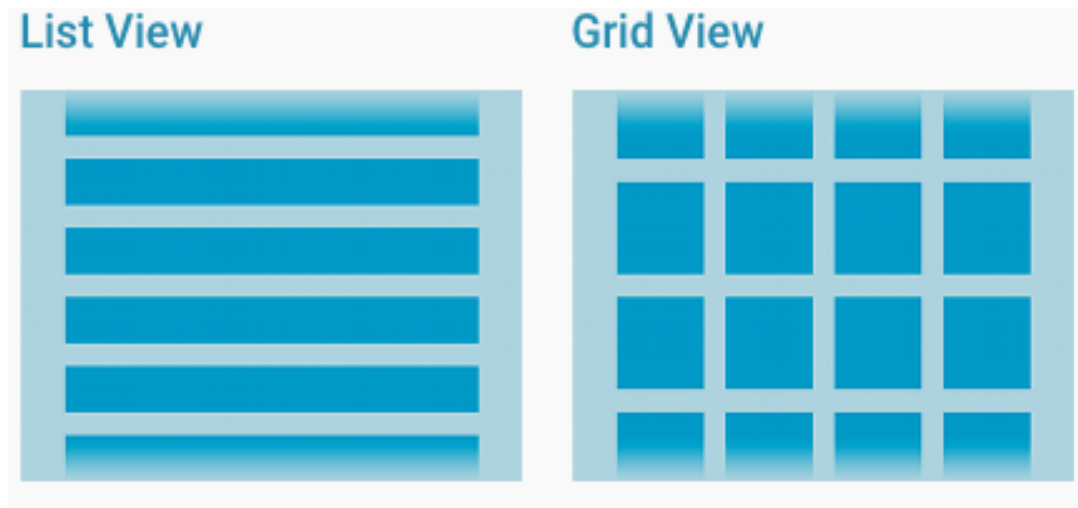
Common Layouts

- Linear Layout: A layout that organizes its children into a single horizontal or vertical row
- Relative Layout: Enables you to specify the location of child objects relative to each other
- Webview: particularly for web pages



Layouts with an Adapter

- Listview: Displays a scrolling single column list
- Gridview: Displays a scrolling multiple columns and rows



Filling ListView With Data

- Binding a ListView instance to an **Adapter** so that it can retrieve data from an external source and creates a View that represents each data entry
 - ArrayAdapter: Use this adapter when your data source is an array
 - SimpleCursorAdapter: Use this adapter when your data comes from a Cursor

ListView with ArrayAdapter

- For example, if you have an array of strings you want to display in a [ListView](#), initialize a new [ArrayAdapter](#) to specify the layout for the string array

```
// We need to use a different list item layout for devices older than Honeycomb int layout =  
Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB ?  
android.R.layout.simple_list_item_activated_1 : android.R.layout.simple_list_item_1;  
  
// Create an array adapter for the list view, using the Ipsum headlines array  
setListAdapter(new ArrayAdapter<String>(getActivity(), layout, Ipsum.Headlines));
```