Custom Views and XML Layouts

The wait was well worth it

XML Layouts

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

XML Layouts

- Defines a view hierarchy
- Defines the layout/position of each view
- Defines the attributes/settings for those views
- Is more succinct than Java

XML Attributes

```
<TextView android:id="@+id/text"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello, I am a TextView" />
```

- XML allows you to set properties on built in views.
- What about your custom view's properties?

Custom Attributes

```
<resources>;
    <declare-styleable name="PieChart">
        <attr name="showText" format="boolean" />
        <attr name="labelPosition" format="enum">
            <enum name="left" value="0"/>
            <enum name="right" value="1"/>
        </attr>
    </declare-styleable>
</resources>
```

res/values/attrs.xml

Custom Attributes in XML

- Import using XML namespace.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:custom="http://schemas.android.com/apk/res/com.example.customviews">
 <com.example.customviews.charting.PieChart
     custom:showText="true"
     custom:labelPosition="left" />
</LinearLayout>
```

AttributeSet

- To apply settings from XML, you must implement the right constructor on your view.
- YourView(Context c, AttributeSet s)

```
public PieChart(Context ctx, AttributeSet attrs) {
    super(ctx, attrs);
    TypedArray a = context.getTheme().obtainStyledAttributes(
         attrs,
         R.styleable.PieChart,
         0, 0);

    try {
        mShowText = a.getBoolean(R.styleable.PieChart_showText, false);
        mTextPos = a.getInteger(R.styleable.PieChart_labelPosition, 0);
    } finally {
        a.recycle();
    }
}
```

Questions?