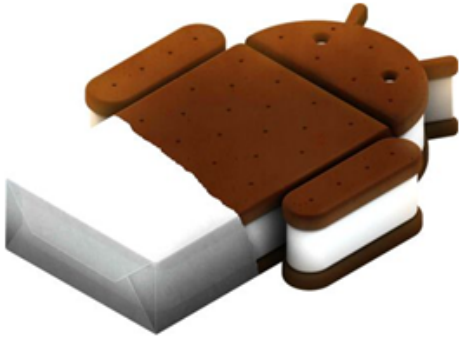
ANDROID UI


OFF
ON
Normal
Focused
Pressed
Focused disabled
Disabled
SECTION HEADER
Normal
Pressed
Activated
Focused
Disabled focused
Snooze length
9
10
minutes
11
Cancel
OK
1 selected

Outline

- Android UI History
- Mobile UX
- Layout Fundamentals
- Pimp my App (Styles)
- Navigation
- Building Blocks
- Lists and Grids
- UI Tools and Tricks

ANDROID UI CHANGES OVER TIME



Android UI Time Pockets

<4.0: “The Good ‘ol Mess” (~7%)
- Use support and compatibility libraries

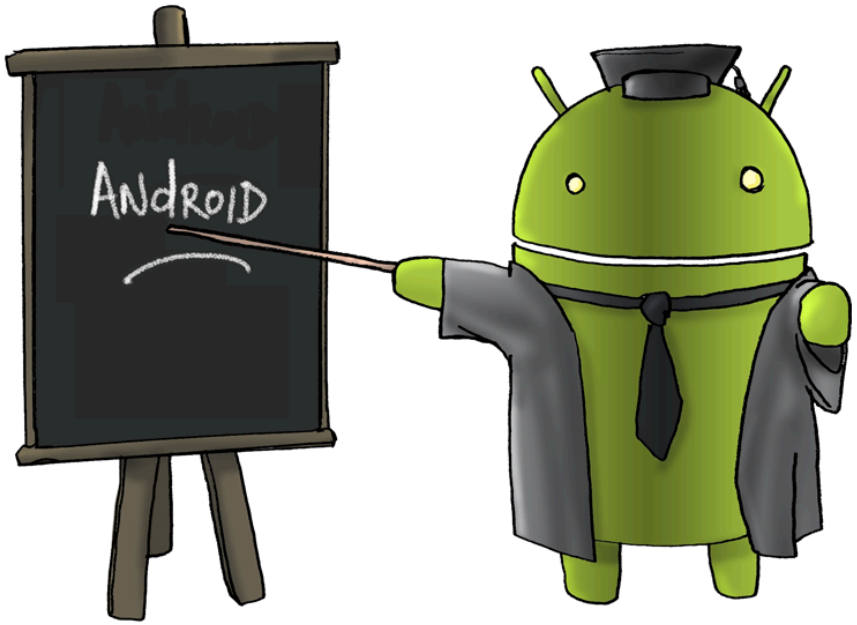
4.0-4.X: “The Current Experience” (~88%)
- Unification of Phones and Tablets
- No more hardware buttons
- No menu or search button
- “Back”, “Home”, Recents”
- New rules and patterns

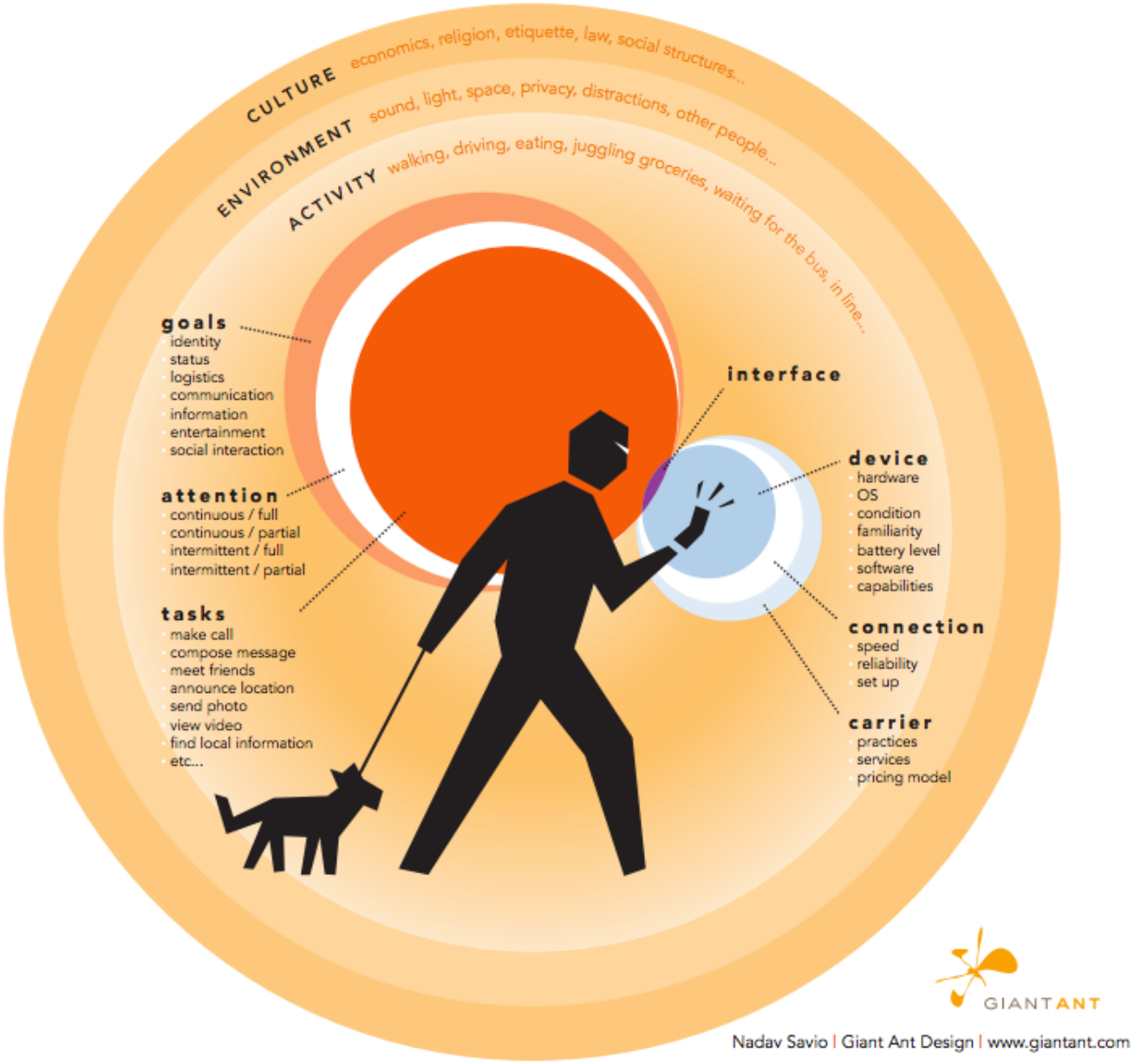
5.0: “New Material” (~5%)
- Material Design
- New rules and patterns
- Wearables and other platforms

ANDROID MOBILE UX

Android Creative Vision

- Enchant me
- Simplify my life
- Make me Amazing




CULTURE economics, religion, etiquette, law, social structures...
ENVIRONMENT sound, light, space, privacy, distractions, other people...
ACTIVITY walking, driving, eating, juggling groceries, waiting for the bus, in line...
goals
identity
status
logistics
communication
information
entertainment
social interaction
attention
continuous / full
continuous / partial
intermittent / full
intermittent / partial
tasks
make call
compose message
meet friends
announce location
send photo
view video
find local information
etc...
interface
device
hardware
OS
condition
familiarity
battery level
software
capabilities
connection
speed
reliability
set up
carrier
practices
services
pricing model
GIANTANT
Nadav Savio | Giant Ant Design | www.giantant.com

Platform Guidelines

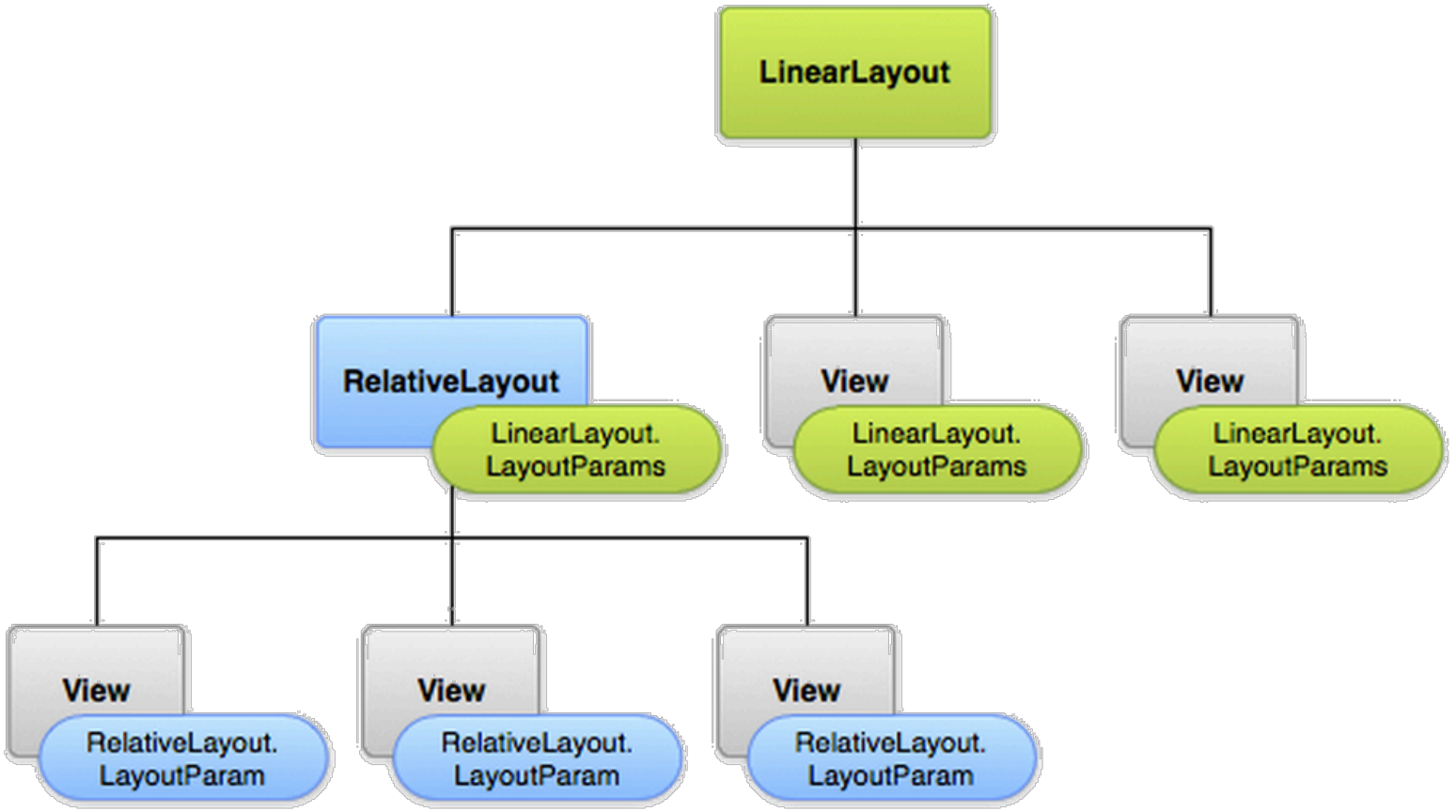
- Follow the Android UI design guidelines that fit with your target users and devices
  - Consistency is key!
- Leverage the platform and don't be afraid of using "standard" elements
- Remember: Android is NOT iOS
  - Don't copy patterns
  - Don't force visuals to look like iOS

Be Quick or Be Dead!

- Users have zero tolerance for lag
- Never block the UI thread
- Only access UI elements from UI thread
- Speed limit enforced!

LAYOUTS

Layouts and LayoutParams



http://developer.android.com/guide/topics/ui/declaring-layout.html

Common Layouts

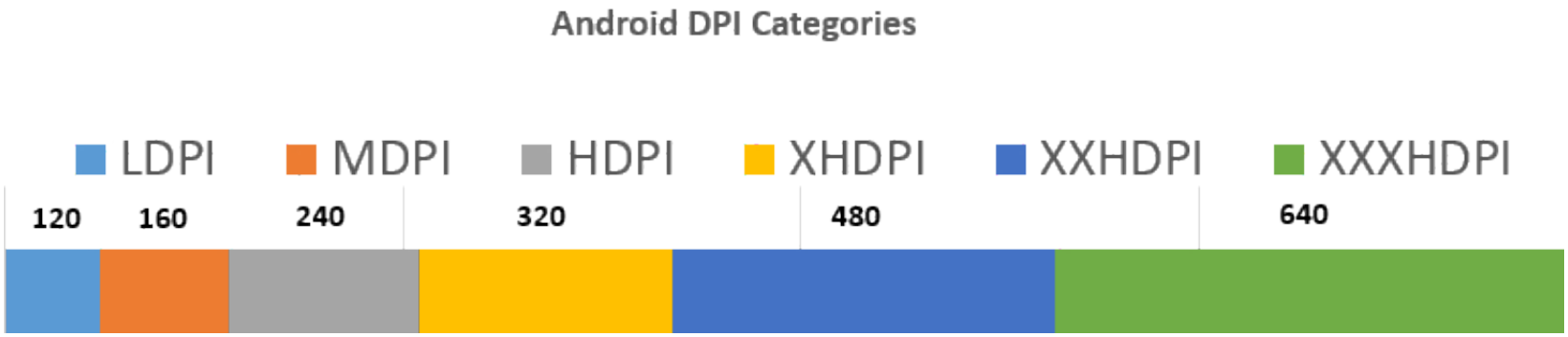
- LinearLayout
  - Horizontal or vertical
- RelativeLayout
  - Each view in specific relation to other defined views and/or parent
- FrameLayout
  - Designed to layout one item but can be used for overlays etc.

Size

- Specified through parameters
  - `layout_width`
  - `layout_height`

- Use
  - `wrap_content`
  - `match_parent`
  - Density independent pixels (dp)

Pixel Density / DPI



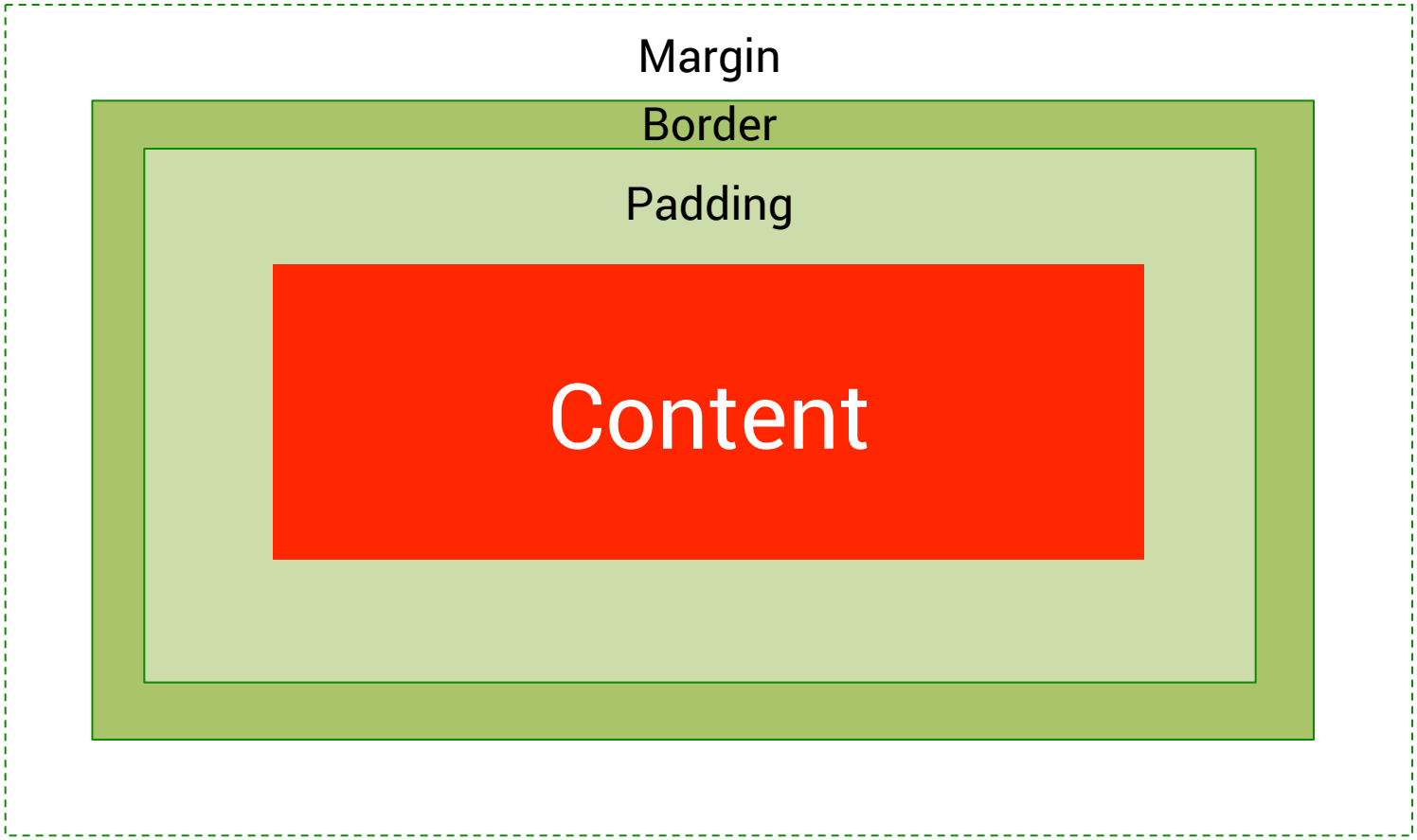
http://pixplicity.com/dp-px-converter/


PIMP MY APP

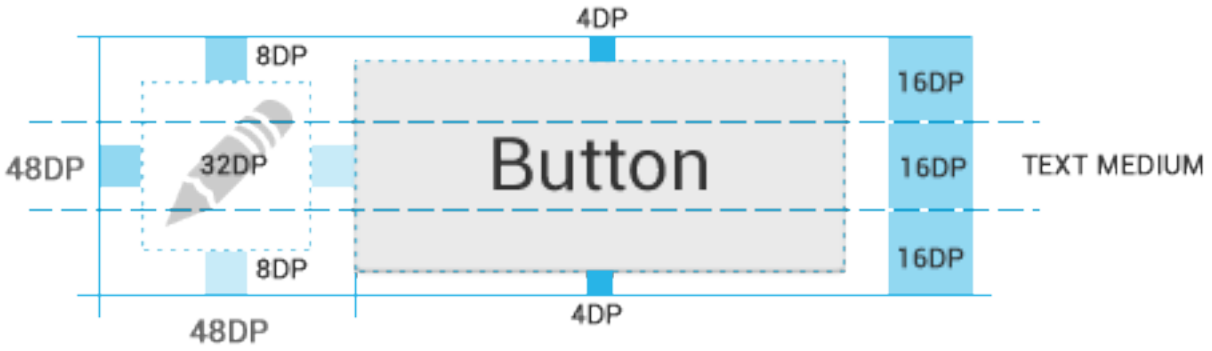
Themes and Styles

- “Themes are Android's mechanism for applying a **consistent style** to an app or activity”
- Themes can be set for the whole app or activities
- Styles can be applied to elements at all levels
- Use Android 4.0 standard themes as a starting point and override
  - Holo Light
  - Holo Dark

Margin, Border and Padding



The Android 48dp “Flow”



http://developer.android.com/design/style/metrics-grids.html

Text and Fonts

- Scale-independent Pixels (sp)

Roboto Thin
Roboto Light
Roboto Regular
Roboto Medium
Roboto Bold
Roboto Black
Roboto Condensed Light
Roboto Condensed
Roboto Condensed Bold

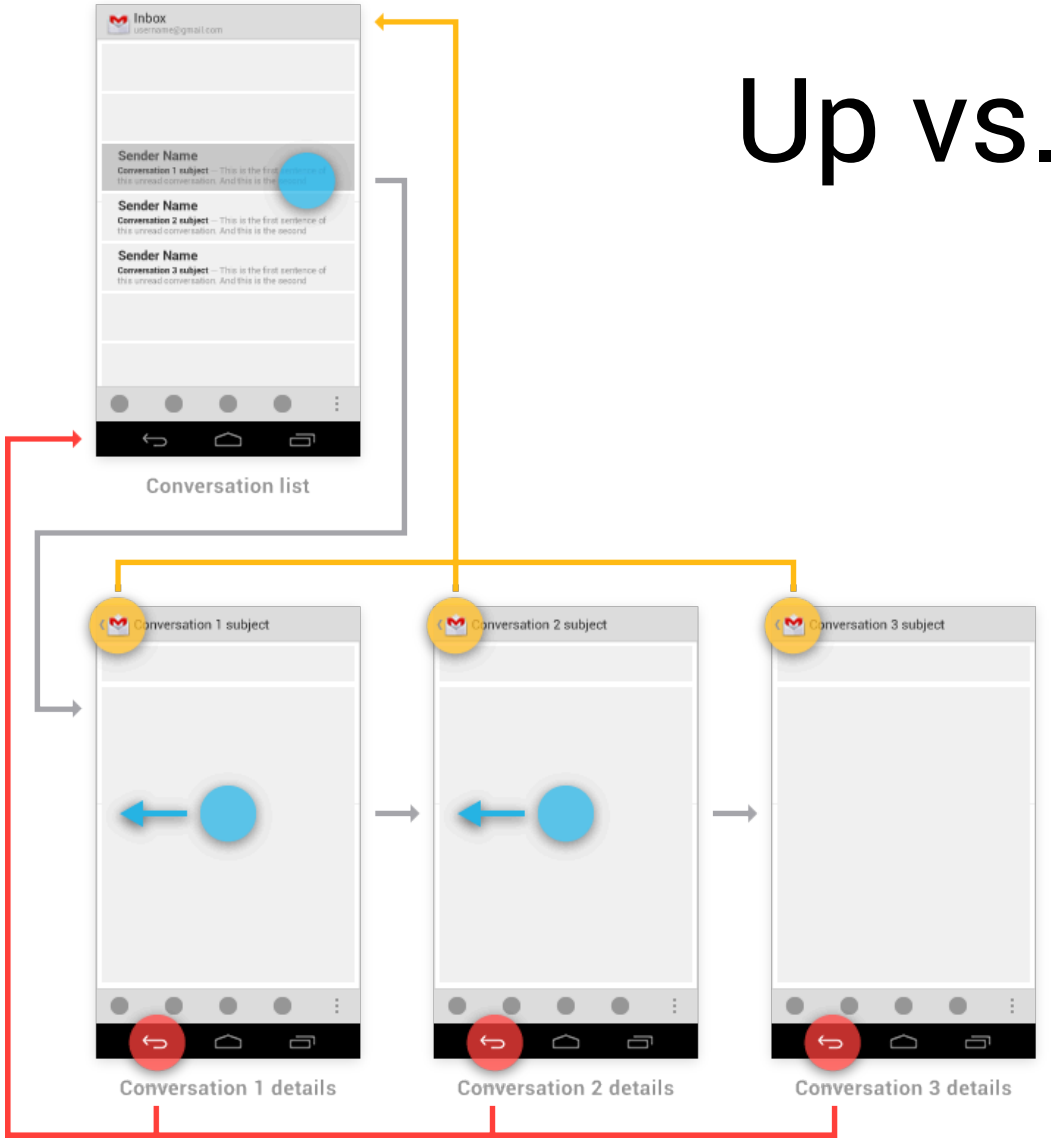
http://developer.android.com/design/style/typography.html

NAVIGATION & BUILDING BLOCKS

Back vs. Up

- Can have big consequences depending on complexity of your app
- Follow the official guidelines to insure consistency on platform
- Generally
  - Up is “hierarchical”
  - Back is “chronological”

Up vs. Back



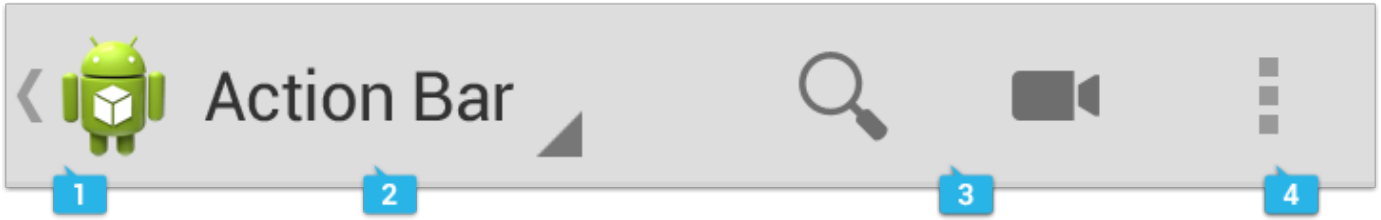
https://developer.android.com/design/patterns/navigation.html

Action Bar

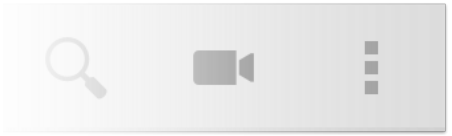
- "The *action bar* is a **dedicated piece of real estate** at the top of each screen that is generally **persistent throughout the app**"
  - "Makes **important actions prominent** and accessible in a predictable way (such as *New* or *Search*)"
  - "Supports **consistent navigation** and view switching within apps"
  - "**Reduces clutter** by providing an action overflow for rarely used actions"
  - "Provides a dedicated space for giving your app an **identity**"

http://developer.android.com/design/patterns/actionbar.html

Action Bar



1. App icon
2. View Control
3. Action Buttons
4. Action Overflow



http://developer.android.com/design/patterns/actionbar.html

Action Buttons

- Decide which actions are FIT for you app:

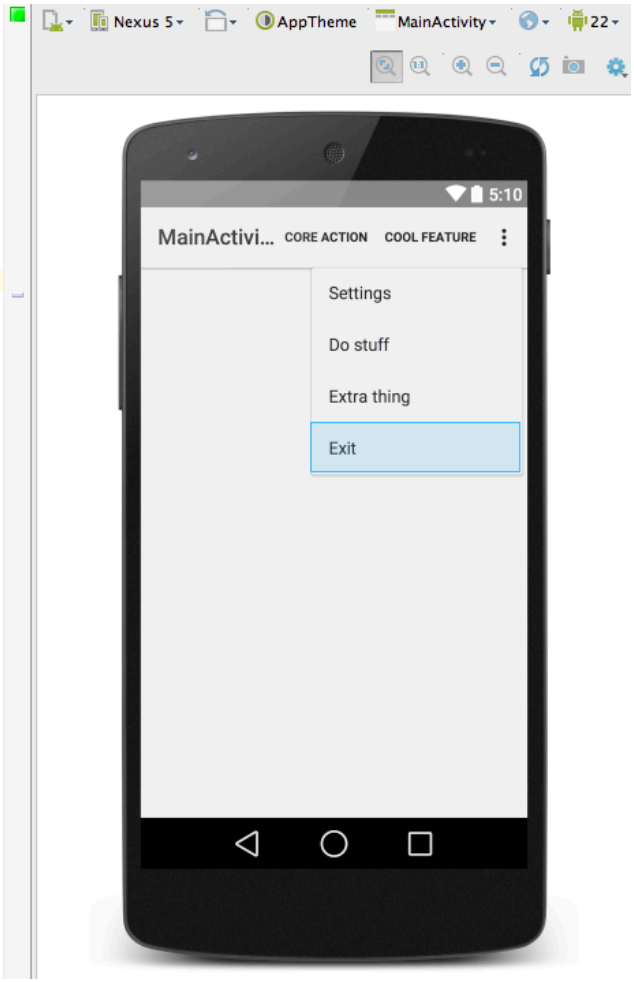
  F = Frequent

  I = Important

  T = Typical
- Only show available actions fitting the given state/context of the app

Define in XML

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.leafcastlelabs.demoui.MainActivity">
    <item android:id="@+id/action_main_thing" android:title="@string/action_core"
        android:orderInCategory="100" android:showAsAction="always" />
    <item android:id="@+id/action_cool_feature" android:title="@string/action_cool_feature"
        android:orderInCategory="100" android:showAsAction="ifRoom" />
    <item android:id="@+id/action_settings" android:title="@string/action_settings"
        android:orderInCategory="100" android:showAsAction="never" />
    <item android:id="@+id/action_do_stuff" android:title="@string/action_do_stuff"
        android:orderInCategory="100" android:showAsAction="never" />
    <item android:id="@+id/action_extra_thing" android:title="@string/action_extra_thing"
    android:orderInCategory="100" android:showAsAction="never" />
    <item android:id="@+id/action_exit" android:title="@string/action_exit"
        android:orderInCategory="100" android:showAsAction="never" />
</menu>
```



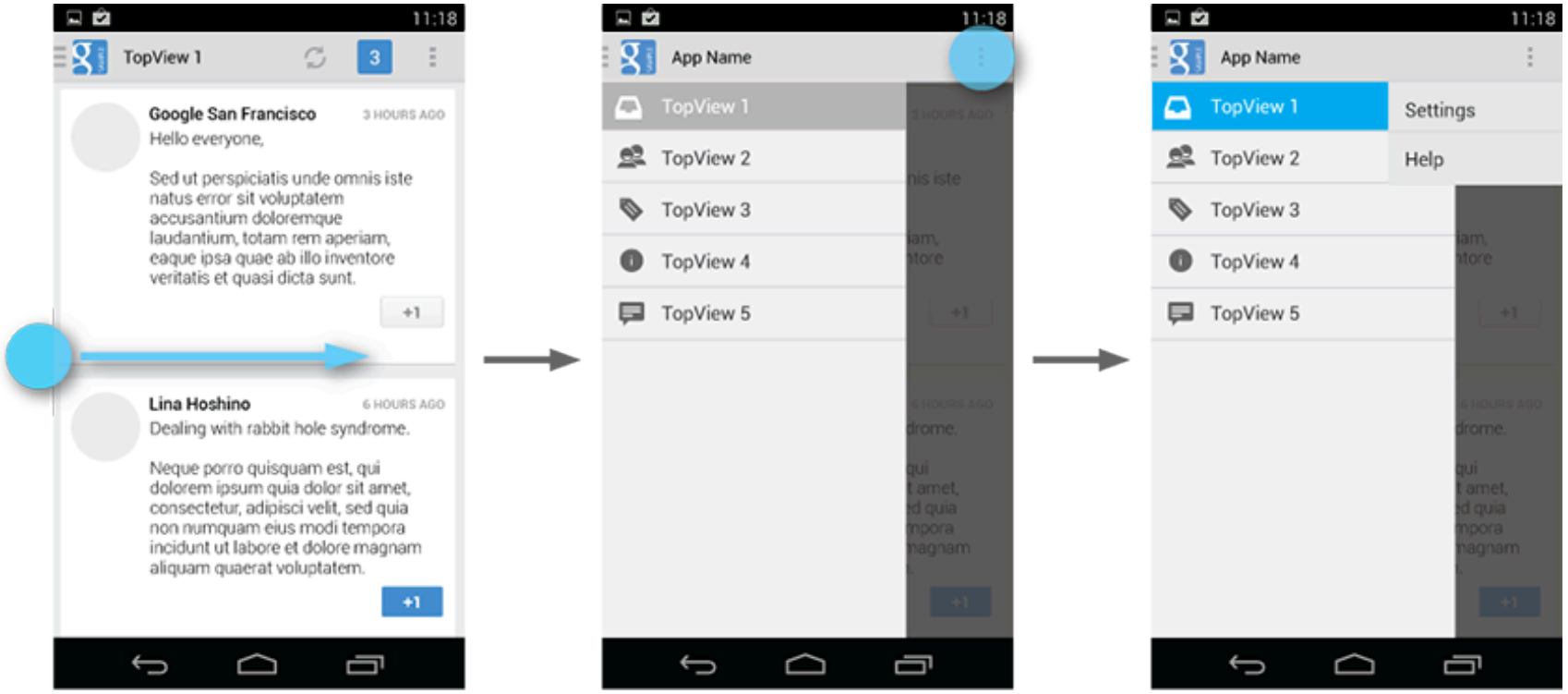
Implementing Action Bar

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

    int id = item.getItemId();
    boolean overrideDefaultHandling = false;
    switch(id){
        case R.id.action_exit:
            overrideDefaultHandling = true;
            finish();
            break;
        case R.id.action_settings:
            overrideDefaultHandling = true;
            break;

    }
    if(overrideDefaultHandling){
        return true;
    } else {
        return super.onOptionsItemSelected(item);
    }
}
```

Contextual Action Bar

- Activate on long press
  - Standard for selection of an item
  - Present specific options
  - allow for more items to be selected
- Overrides normal Action Bar
- Great for multi-select and “bulk actions”

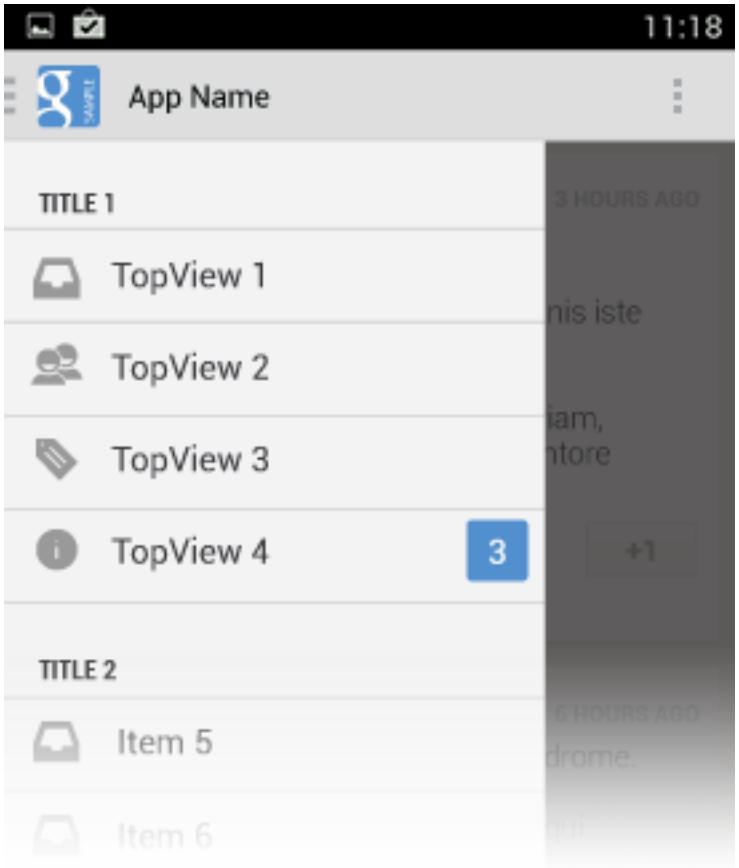
Navigation Drawer



https://developer.android.com/design/patterns/navigation-drawer.html

Navigation Drawer

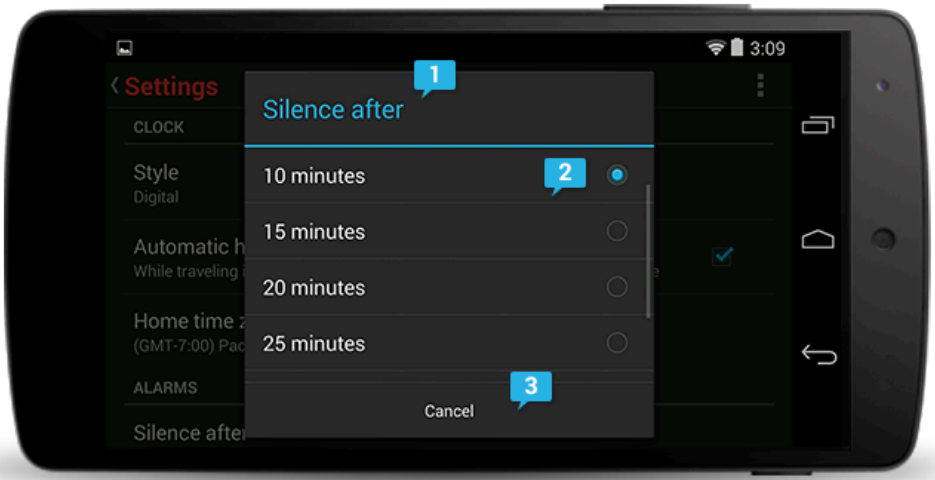
- Only if your app “needs” it
- Titles, icons (and counters)
- Clean up Action Bar on use
  - Only show icon, menu and actions with global scope



https://developer.android.com/design/patterns/navigation-drawer.html

Dialogs

- Generally use `AlertDialog`
- Uses Builder pattern
  1. Header
  2. Content
  3. Action Buttons
- Special subtypes
  - Alerts
  - Popups



http://developer.android.com/design/building-blocks/dialogs.html

Notifications

- Uses Builder pattern
- Requires
  - Icon
  - Title
  - Content Text
- Can contain
  - `PendingIntent`, e.g. start a particular `Activity`
  - Priority
  - Custom layout/style (introduced in 4.1)

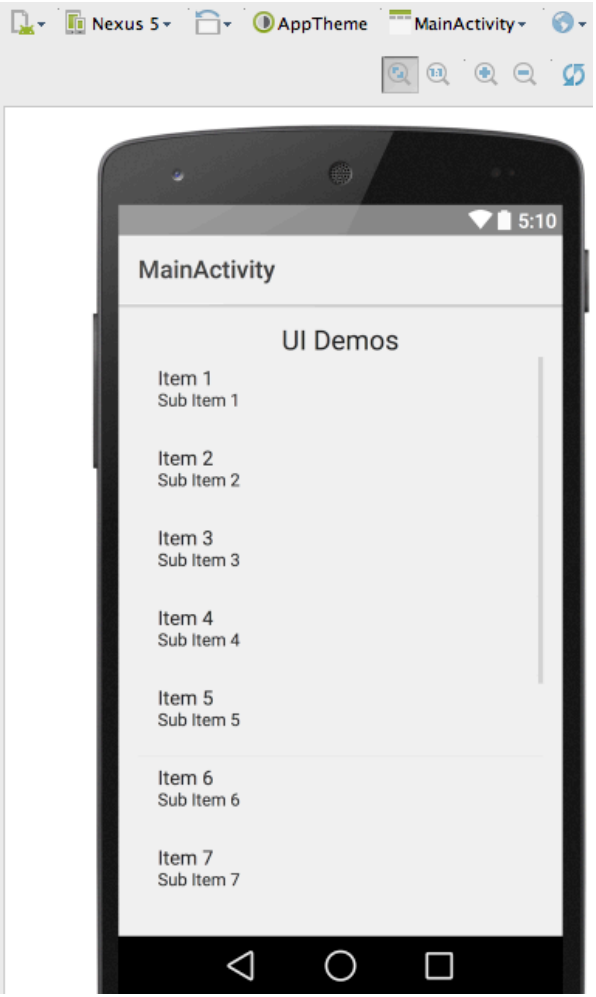
LISTS AND GRIDS

Lists and Grids

- Scrollable views with a bunch of similar items (Views)
- Uses Adaptors to generate UI elements dynamically at runtime using predefined layouts
- Can handle very large datasets efficiently by “recycling” views

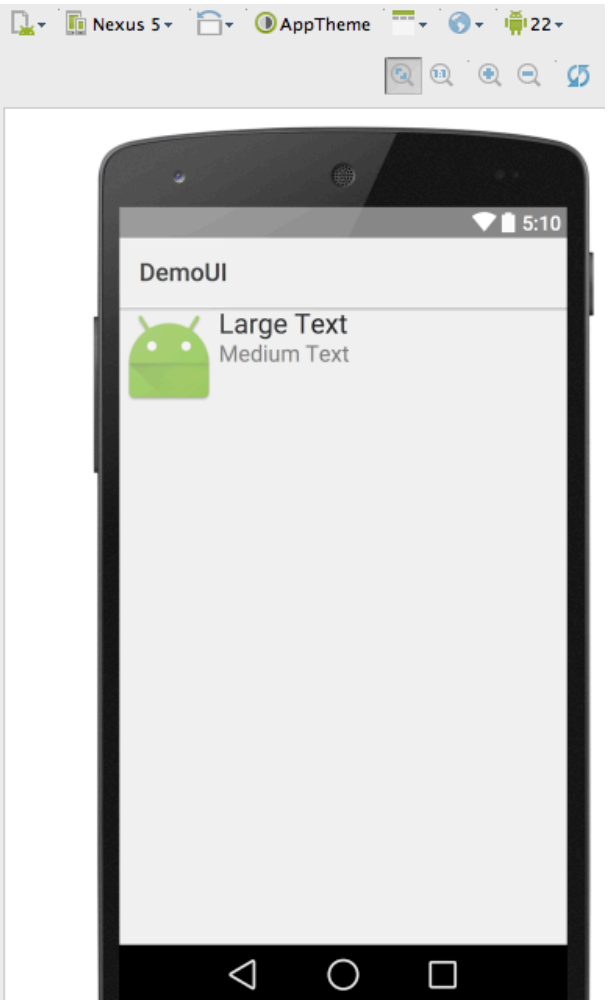
activity_main.xml

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match
    android:layout_height="match_parent" android:paddingLeft="64dp"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context="com.leafcastlelabs.demoui.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="UI Demos"
        android:id="@+id/txtMainHeading"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_below="@+id/txtMainHeading"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```



demo_list_item.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">


    <ImageView
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:id="@+id/imgDemoIcon"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:src="@mipmap/ic_launcher" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/txtDemoTitle"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/imgDemoIcon"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Medium Text"
        android:id="@+id/txtDemoDescription"
        android:layout_below="@+id/txtDemoTitle"
        android:layout_toRightOf="@+id/imgDemoIcon"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />
</RelativeLayout>
```



Demo.java

```
6   public class Demo {
7
8       private String name;
9       private String description;
10      private String intentAction;
11      private int resultCode;
12
13      public Demo(String demoName, String demodescription){
14
15          this(demoName, demodescription, null, 0);
16      }
17
18      public Demo(String demoName, String demodescription, String demoAction, int demoResultCode){
19
20          this.name = demoName;
21          this.description = demodescription;
22          this.intentAction = demoAction;
23          this.resultCode = demoResultCode;
24      }
25
26      public String getName() { return name; }
29
30      public void setName(String name) { this.name = name; }
33
34      public String getDescription() { return description; }
37
38      public void setDescription(String description) { this.description = description; }
41
42      public String getIntentAction() { return intentAction; }
45
46      public void setIntentAction(String intentAction) { this.intentAction = intentAction; }
49
50      public int getResultCode() { return resultCode; }
53
54      public void setResultCode(int resultCode) { this.resultCode = resultCode; }
57  }
58
```

DemoAdaptor.java

```java
public class DemoAdaptor extends BaseAdapter {

    Context context;
    ArrayList<Demo> demos;
    Demo demo;

    public DemoAdaptor(Context c, ArrayList<Demo> demoList){
        this.context = c;
        this.demos = demoList;
    }

    @Override
    public int getCount() {...}

    @Override
    public Object getItem(int position) {...}

    @Override
    public long getItemId(int position) { return position; }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        if (convertView == null) {
            LayoutInflater demoInflator = (LayoutInflater) this.context
                    .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = demoInflator.inflate(R.layout.demo_list_item, null);
        }

        demo = demos.get(position);
        if(demo!=null){
            TextView txtTitle = (TextView) convertView.findViewById(R.id.txtDemoTitle);
            txtTitle.setText(demo.getName());

            TextView txtDescription = (TextView) convertView.findViewById(R.id.txtDemoDescription);
            txtDescription.setText(demo.getDescription());
        }
        return convertView;
    }
}
```

MainActivity.java

```
45          demoAdaptor = new DemoAdaptor(this, demoList);
46          demoListView = (ListView)findViewById(R.id.listView);
47          demoListView.setAdapter(demoAdaptor);
48          demoListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
49              @Override
50              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
51
52                  Intent startDemoIntent = new Intent();
53                  startDemoIntent.putExtra("position", position);
54                  String action = demoList.get(position).getIntentAction();
55                  int demoResultCode = demoList.get(position).getResultCode();
56                  if(action != null && !action.equals("")){
57                      startDemoIntent.setAction(action);
58                      startActivityForResult(startDemoIntent, demoResultCode);
59                  }
60              }
61          });
```

UI Tools and Tricks

EXERCISES