Google
Developers

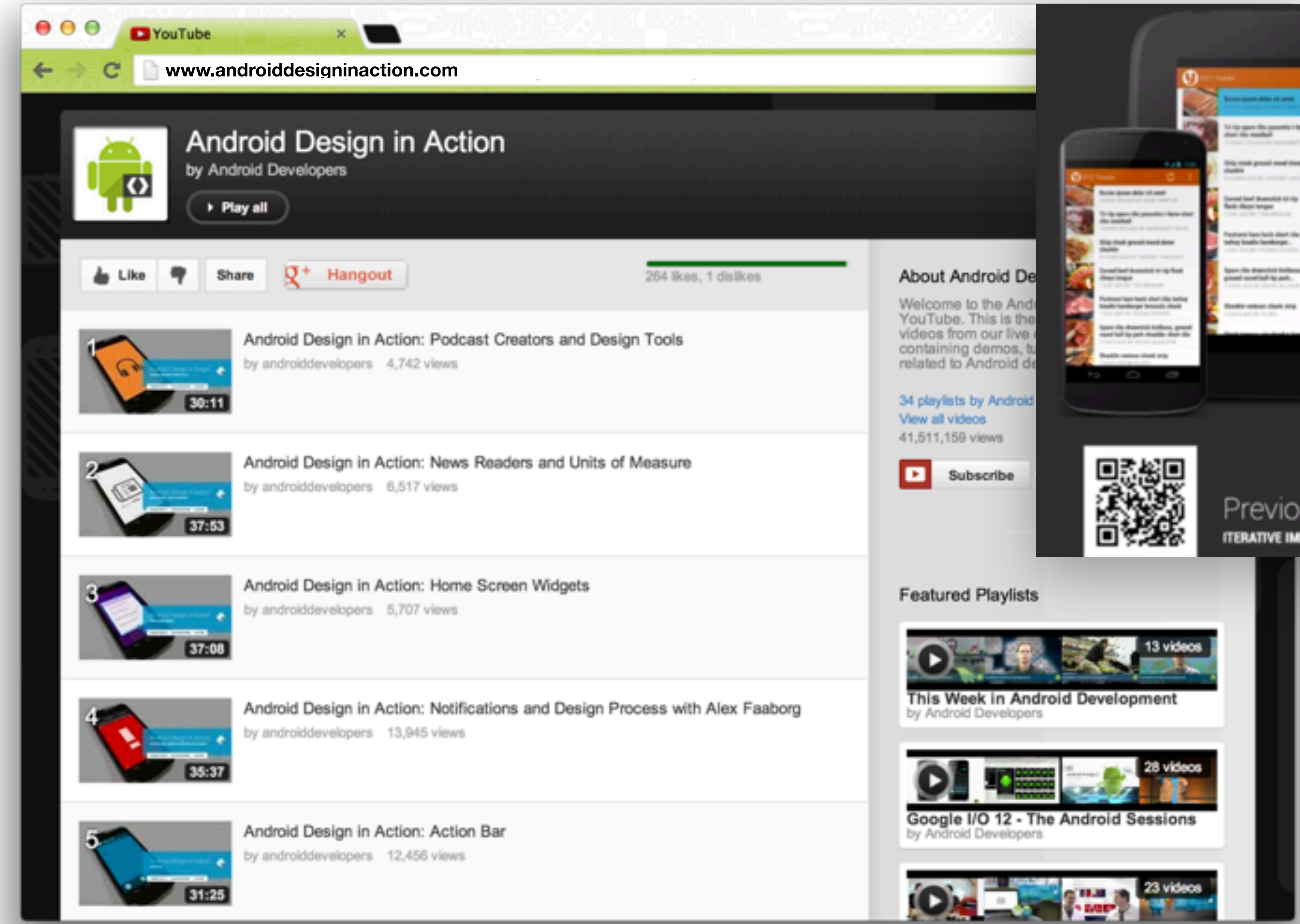
Android Design for UI Developers

Google I/O 13

+Roman Nurik

+Nick Butcher

Android Design in Action



www.androiddesigninaction.com

Android Design in Action
by Android Developers
Play all

Like | Share | Hangout | 264 likes, 1 dislikes

1	Android Design in Action: Podcast Creators and Design Tools — by androiddevelopers 4,742 views — 30:11
2	Android Design in Action: News Readers and Units of Measure — by androiddevelopers 6,517 views — 37:53
3	Android Design in Action: Home Screen Widgets — by androiddevelopers 5,707 views — 37:08
4	Android Design in Action: Notifications and Design Process with Alex Faaborg — by androiddevelopers 13,945 views — 35:37
5	Android Design in Action: Action Bar — by androiddevelopers 12,456 views — 31:25

About Android De
Welcome to the And
YouTube. This is the
videos from our live
containing demos, t
related to Android d

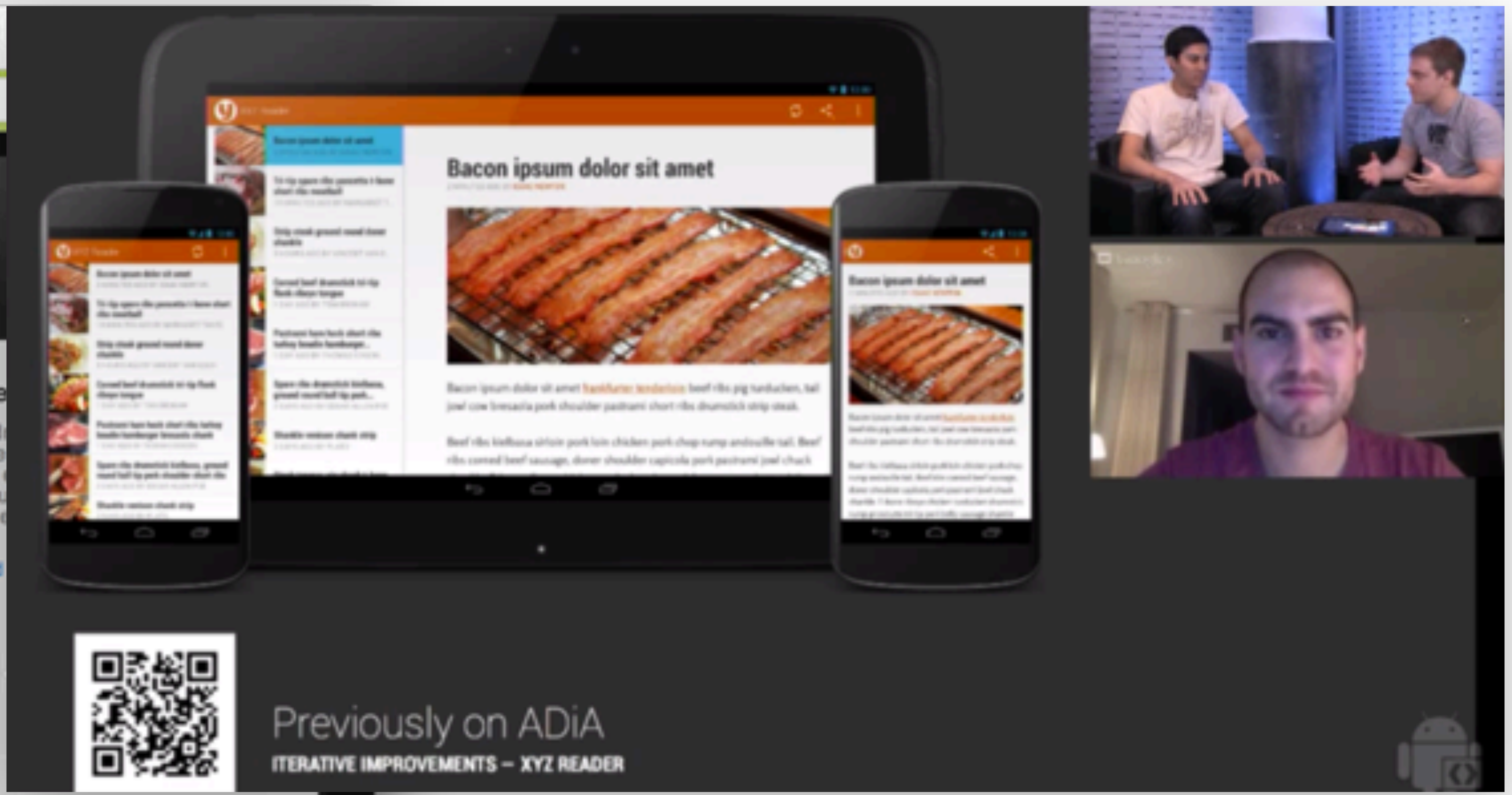
34 playlists by Android
View all videos
41,511,159 views
Subscribe

Featured Playlists

This Week in Android Development — by Android Developers — 13 videos

Google I/O 12 - The Android Sessions — by Android Developers — 28 videos

23 videos


Bacon ipsum dolor sit amet
Previously on ADiA
ITERATIVE IMPROVEMENTS – XYZ READER

Agenda

App Navigation

Lateral navigation

Navigation drawers

Up navigation

Responsive Design

Why responsive

Responsive strategies

Fragments

Resource framework

Holo Visual Language

Dividers and borderless buttons

List headings

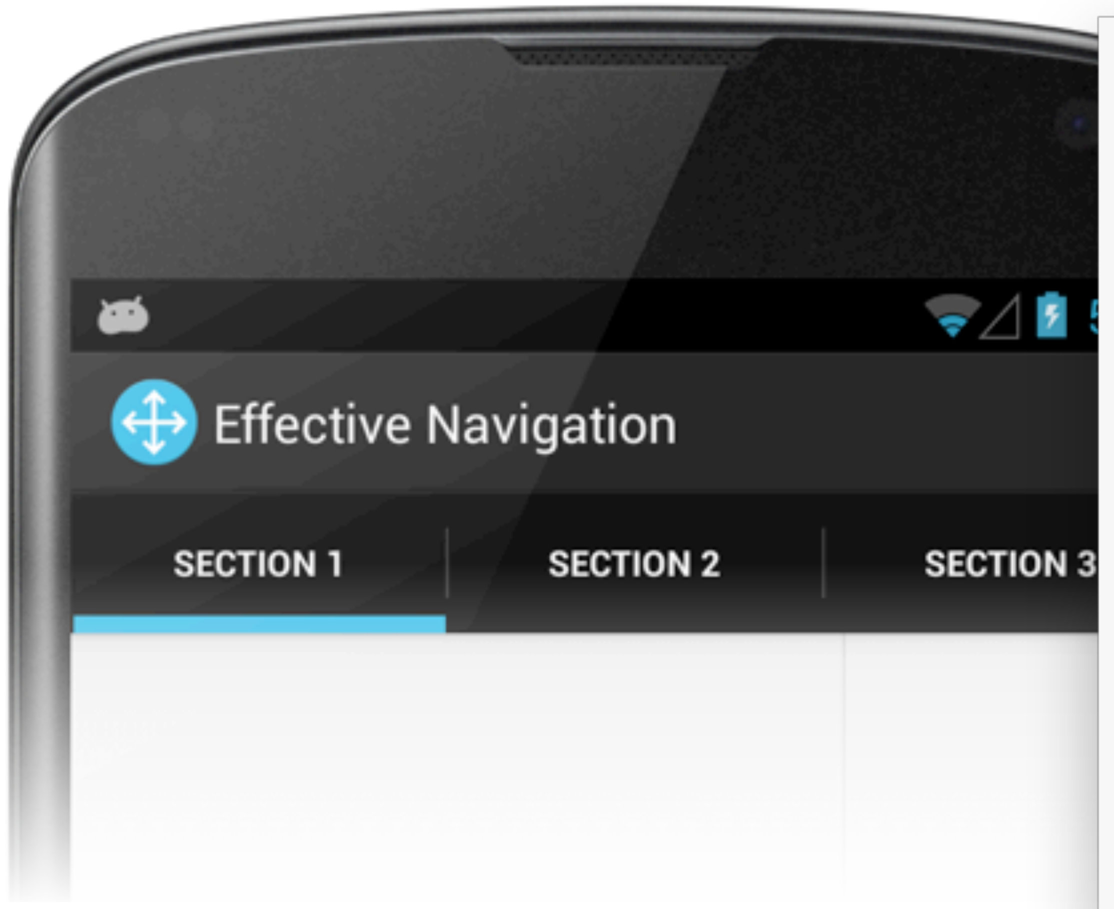
Typography

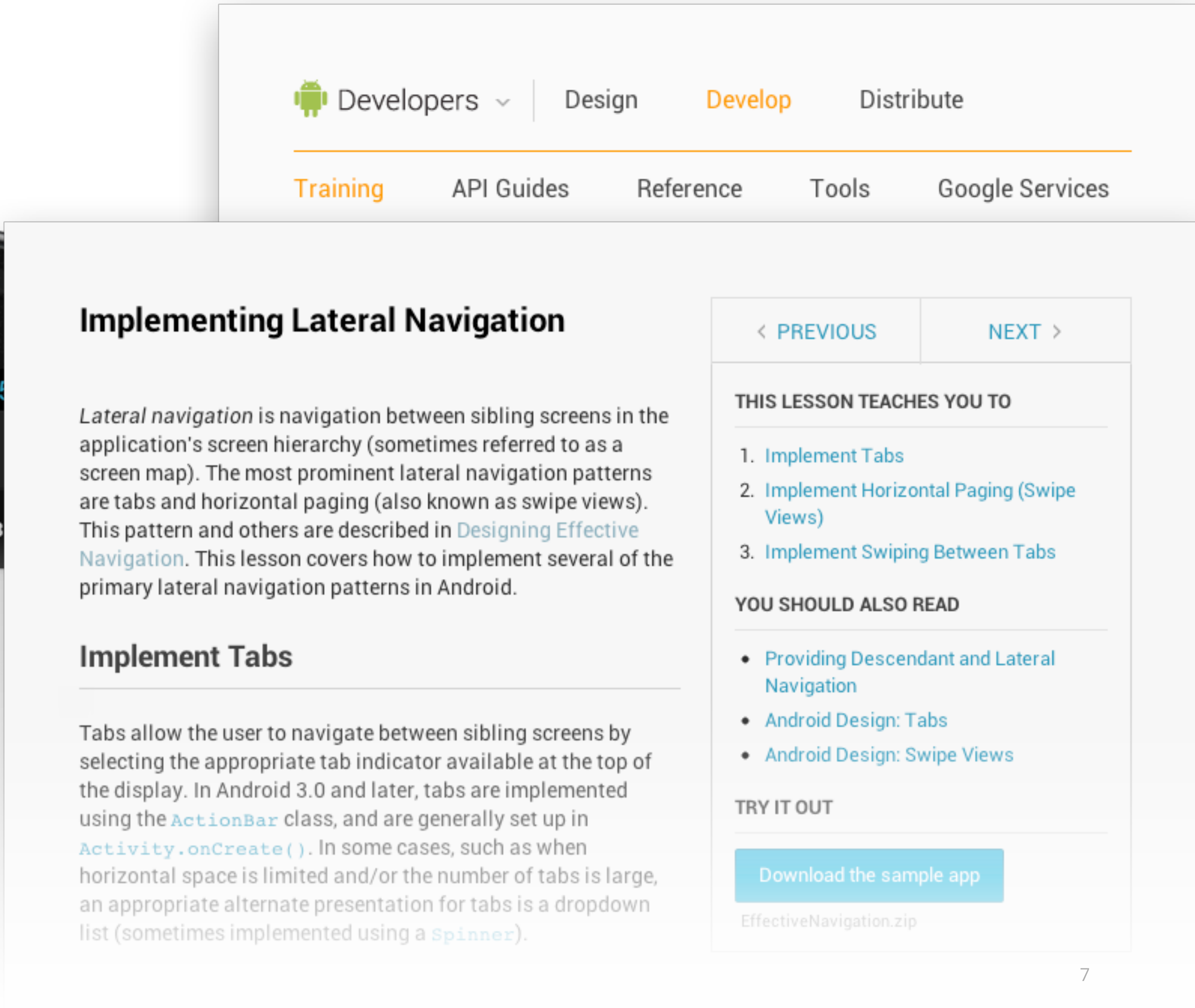
Full-bleed images

Generating assets

App Navigation

Action bar tabs and spinner





Implementing Lateral Navigation

Lateral navigation is navigation between sibling screens in the application's screen hierarchy (sometimes referred to as a screen map). The most prominent lateral navigation patterns are tabs and horizontal paging (also known as swipe views). This pattern and others are described in Designing Effective Navigation. This lesson covers how to implement several of the primary lateral navigation patterns in Android.

Implement Tabs

Tabs allow the user to navigate between sibling screens by selecting the appropriate tab indicator available at the top of the display. In Android 3.0 and later, tabs are implemented using the `ActionBar` class, and are generally set up in `Activity.onCreate()`. In some cases, such as when horizontal space is limited and/or the number of tabs is large, an appropriate alternate presentation for tabs is a dropdown list (sometimes implemented using a `Spinner`).

THIS LESSON TEACHES YOU TO

1. Implement Tabs
2. Implement Horizontal Paging (Swipe Views)
3. Implement Swiping Between Tabs

YOU SHOULD ALSO READ

- Providing Descendant and Lateral Navigation
- Android Design: Tabs
- Android Design: Swipe Views

TRY IT OUT

COMING SOON

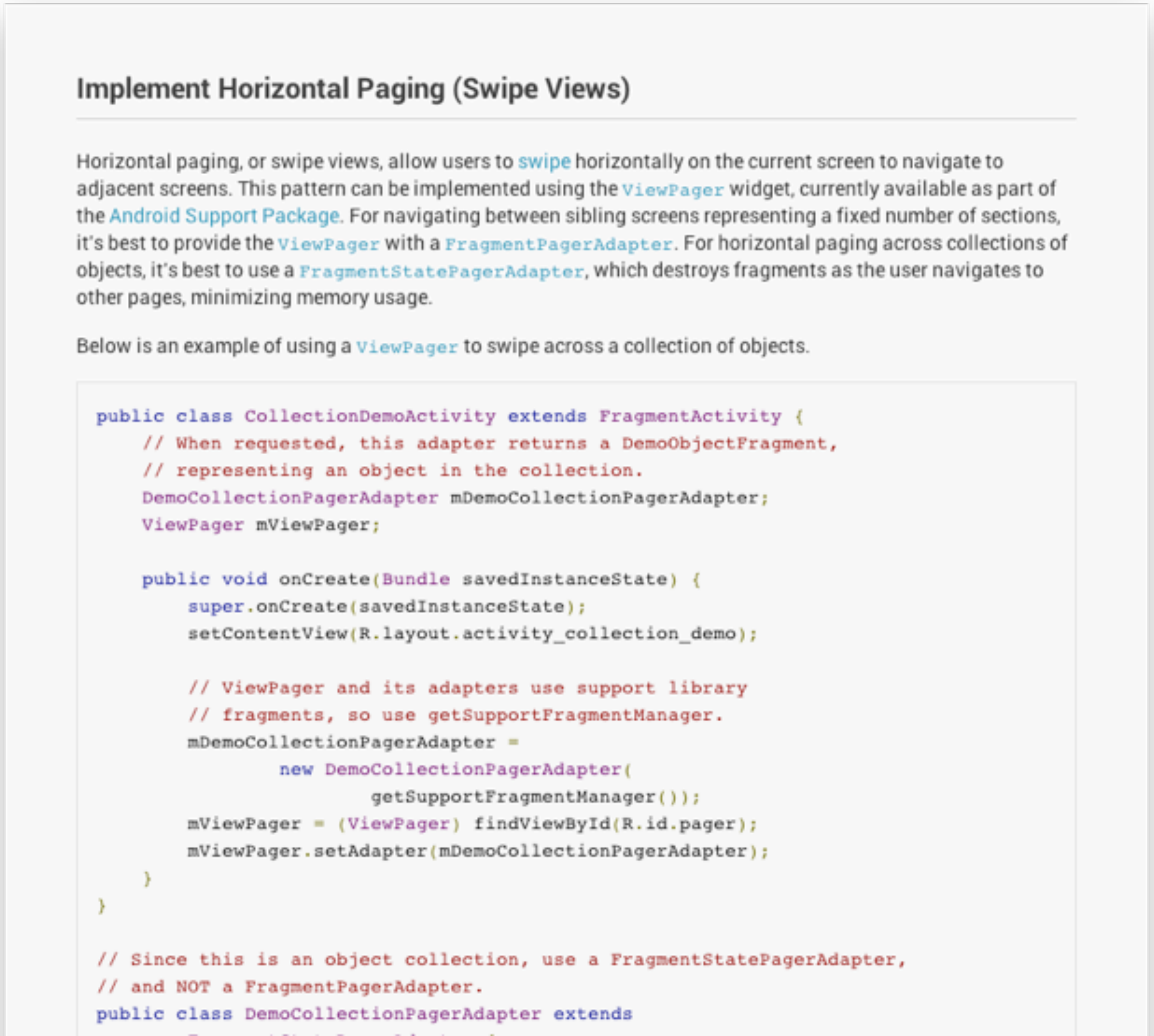
ActionBarCompat

Native API	ActionBarCompat
`Activity` or `FragmentActivity`	`ActionBarActivity`
`getActionBar()`	`getSupportActionBar()`
`Theme.Holo` `Widget.Holo...`	`Theme.AppCompat` `Widget.AppCompat...`
`android:actionBarStyle` `android:displayOptions`	`actionBarStyle` *(no prefix)* `displayOptions` *(no prefix)*
`android:showAsAction`	`yourAppNamespace:showAsAction`

Android 2.1+
(99.9% of devices)

Alternatively, continue using Jake Wharton's incredible `ActionBarSherlock` library

ViewPager

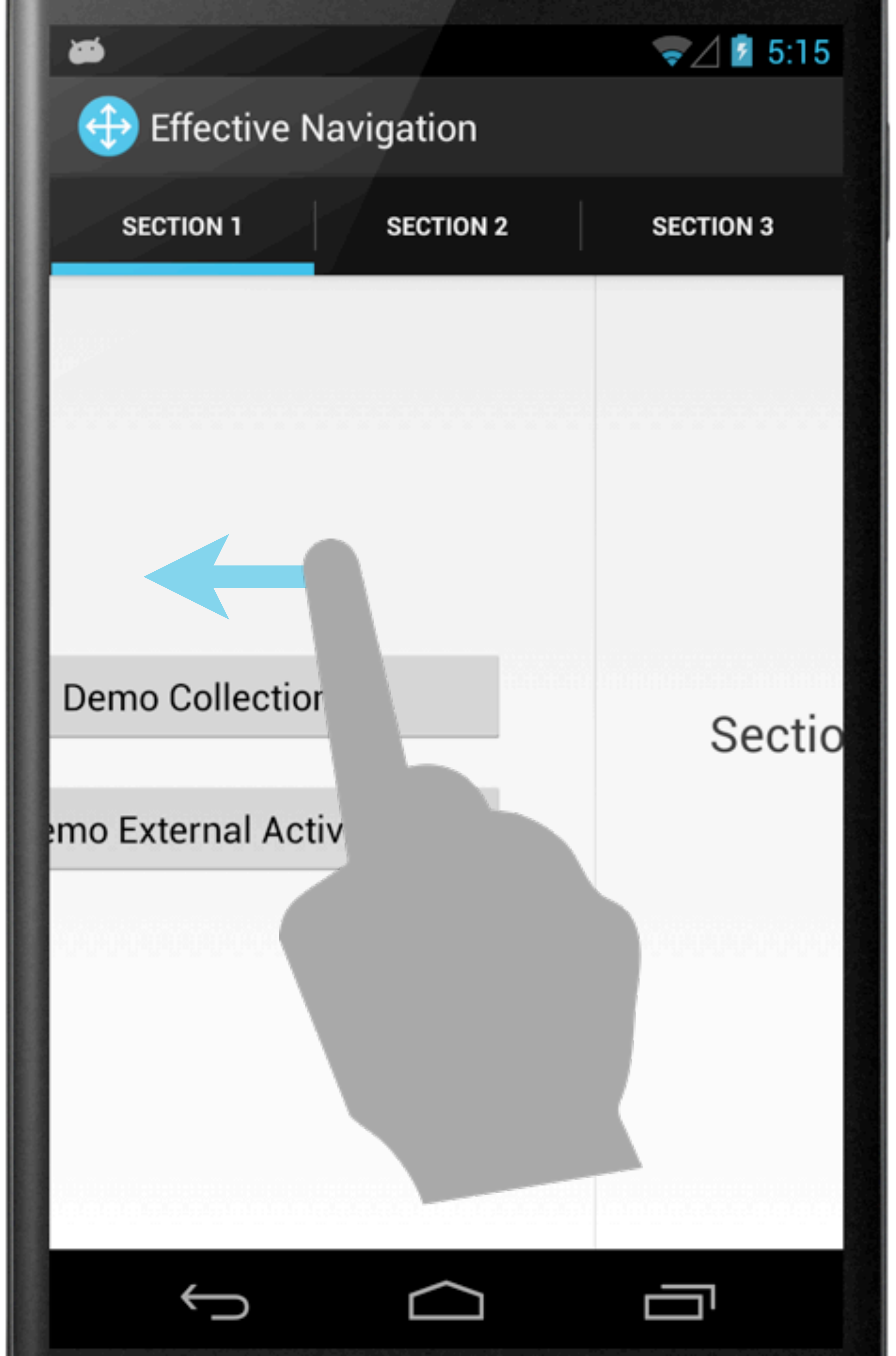


Implement Horizontal Paging (Swipe Views)

Horizontal paging, or swipe views, allow users to swipe horizontally on the current screen to navigate to adjacent screens. This pattern can be implemented using the `ViewPager` widget, currently available as part of the Android Support Package. For navigating between sibling screens representing a fixed number of sections, it's best to provide the `ViewPager` with a `FragmentPagerAdapter`. For horizontal paging across collections of objects, it's best to use a `FragmentStatePagerAdapter`, which destroys fragments as the user navigates to other pages, minimizing memory usage.

Below is an example of using a `ViewPager` to swipe across a collection of objects.

```
public class CollectionDemoActivity extends FragmentActivity {
    // When requested, this adapter returns a DemoObjectFragment,
    // representing an object in the collection.
    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;
    ViewPager mViewPager;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_collection_demo);

        // ViewPager and its adapters use support library
        // fragments, so use getSupportFragmentManager.
        mDemoCollectionPagerAdapter =
                new DemoCollectionPagerAdapter(
                        getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.pager);
        mViewPager.setAdapter(mDemoCollectionPagerAdapter);
    }
}

// Since this is an object collection, use a FragmentStatePagerAdapter,
// and NOT a FragmentPagerAdapter.
public class DemoCollectionPagerAdapter extends
```

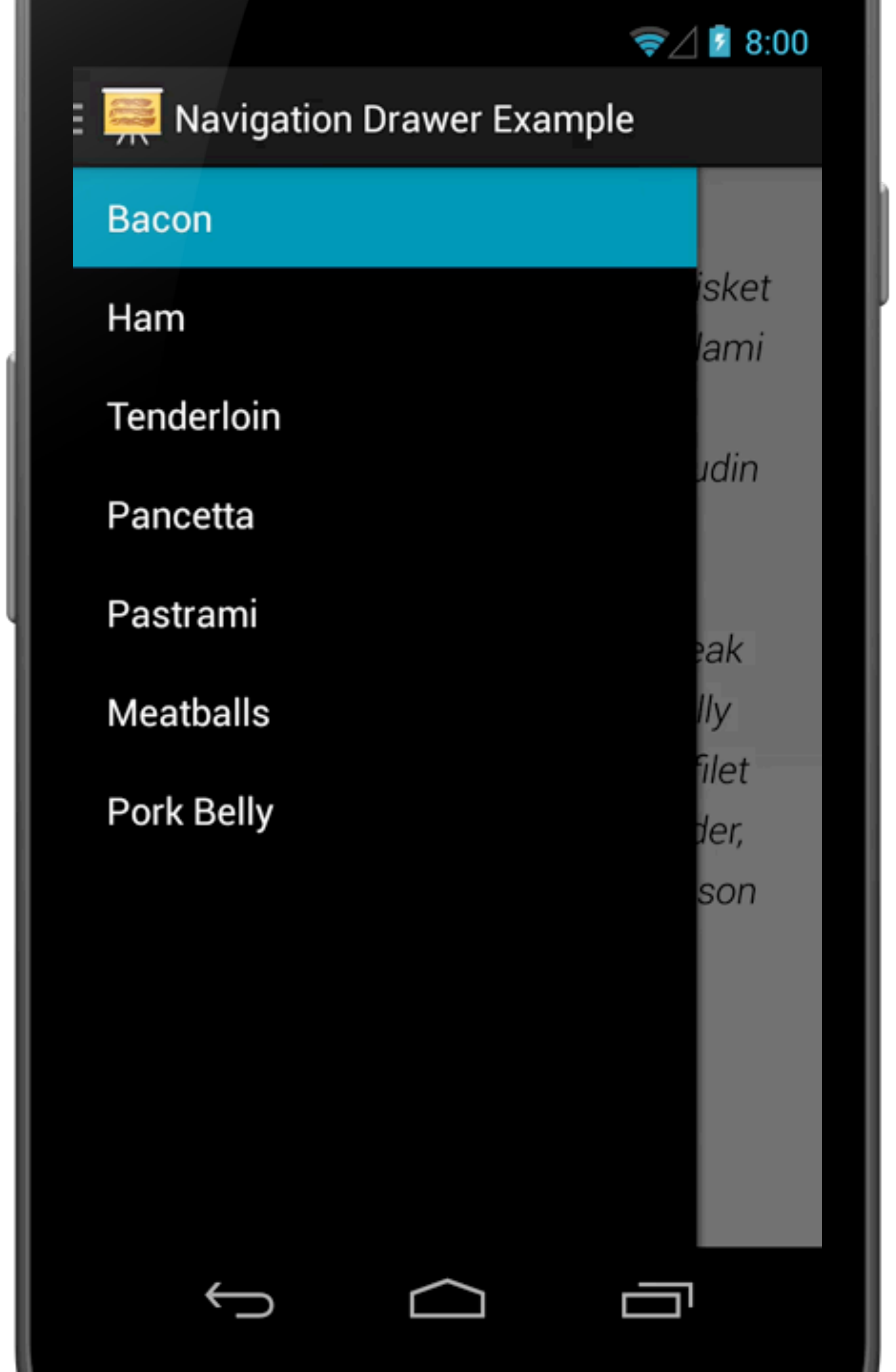


Navigation drawers

Primarily for main app navigation

Only for 3+ top level views of disparate, mutually exclusive content

More at d.android.com/design

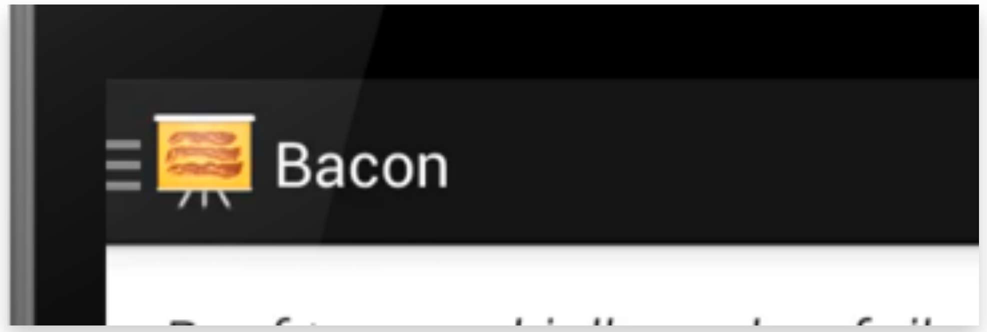


Navigation drawers

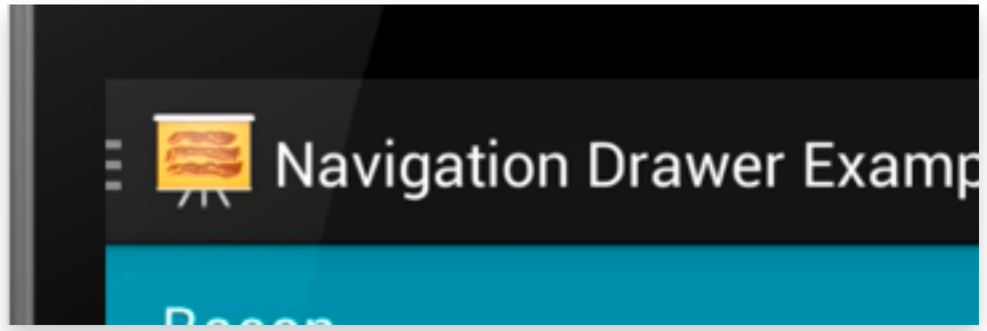
```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- The main content view -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- The navigation drawer -->
    <ListView android:id="@+id/nav_drawer"
        android:layout_gravity="start"
        android:layout_width="@dimen/drawer_width"
        android:layout_height="match_parent"
        android:background="#ffCCCCCC"/>

</android.support.v4.widget.DrawerLayout>
```

ActionBarDrawerToggle and DrawerListener

```
// ActionBarDrawerToggle provides convenient helpers
// for tying together the prescribed interactions
// between a top-level sliding drawer and the action bar.
mDrawerToggle = new ActionBarDrawerToggle(
        this, /* activity */
        mDrawerLayout,
        R.drawable.ic_drawer, /* download available */
        R.string.drawer_open, /* content descriptions */
        R.string.drawer_close) {

    public void onDrawerClosed(View view) {
        // Set the action bar title to the content title.
        getActionBar().setTitle(mTitle);
    }

    public void onDrawerOpened(View drawerView) {
        getActionBar().setTitle("Navigation Drawer Example");
    }
};
mDrawerLayout.setDrawerListener(mDrawerToggle);
```



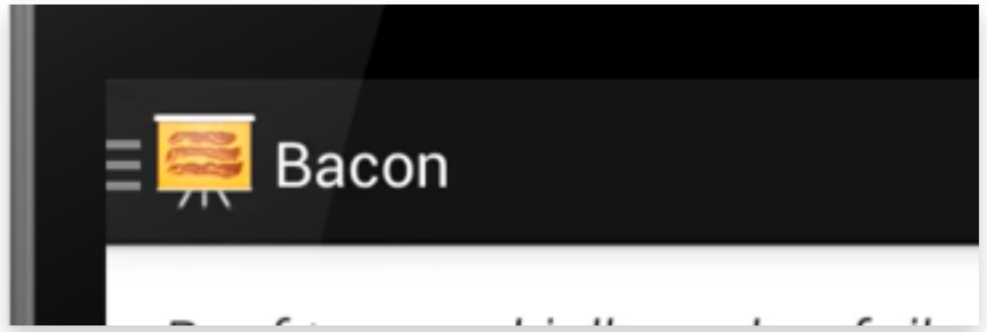
DRAWER CLOSED



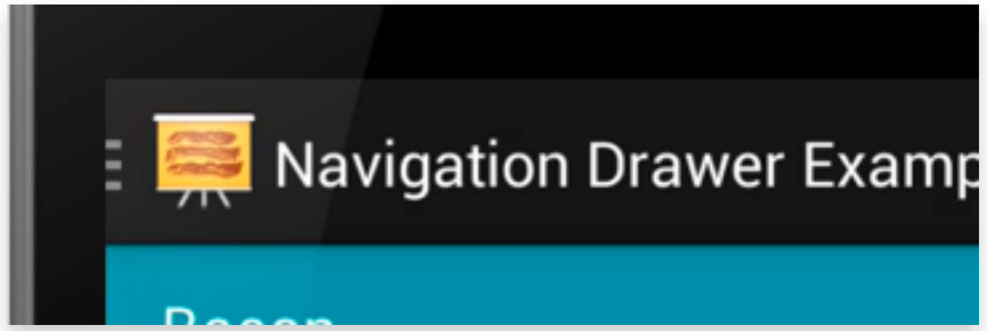
DRAWER OPEN

ActionBarDrawerToggle

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    /*
     * The action bar home/up action should open or
     * close the drawer.
     * ActionBarDrawerToggle will take care of this.
     */
    if (mDrawerToggle.onOptionsItemSelected(item)) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```



DRAWER CLOSED



DRAWER OPEN

Up navigation

```
<style name="ActionBar" parent="android:Widget.Holo.ActionBar">
    <item name="android:displayOptions">showHome|homeAsUp|showTitle</item>
</style>
```

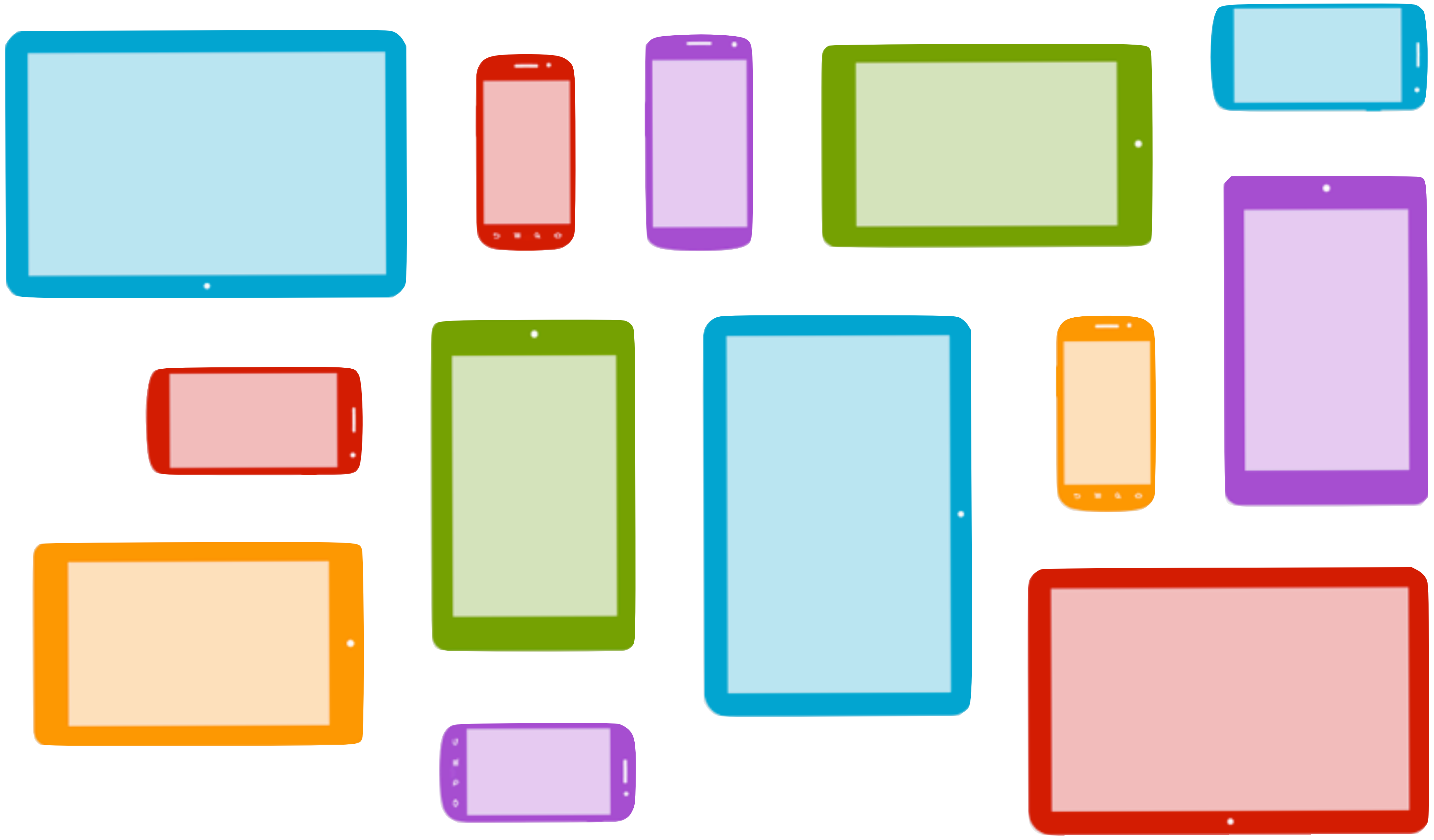
```
<activity
    android:name=".MyChildActivity"
    android:parentActivityName=".MyParentActivity">

    <meta-data android:name="android.support.PARENT_ACTIVITY"
        android:value=".MyParentActivity" />

</activity>
```

Custom Up navigation

```
@Override
public Intent getParentActivityIntent() {
    // Used when navigating within the same task.
    return new Intent(this, MyParentActivity.class)
            .putExtra(START_TAB, 2);
}


@Override
public void onCreateNavigateUpTaskStack(TaskStackBuilder builder) {
    // Used when synthesizing the stack in a new task.
    builder.addNextIntent(new Intent(this, HomeActivity.class))
            .addNextIntent(getParentActivityIntent());
}
```

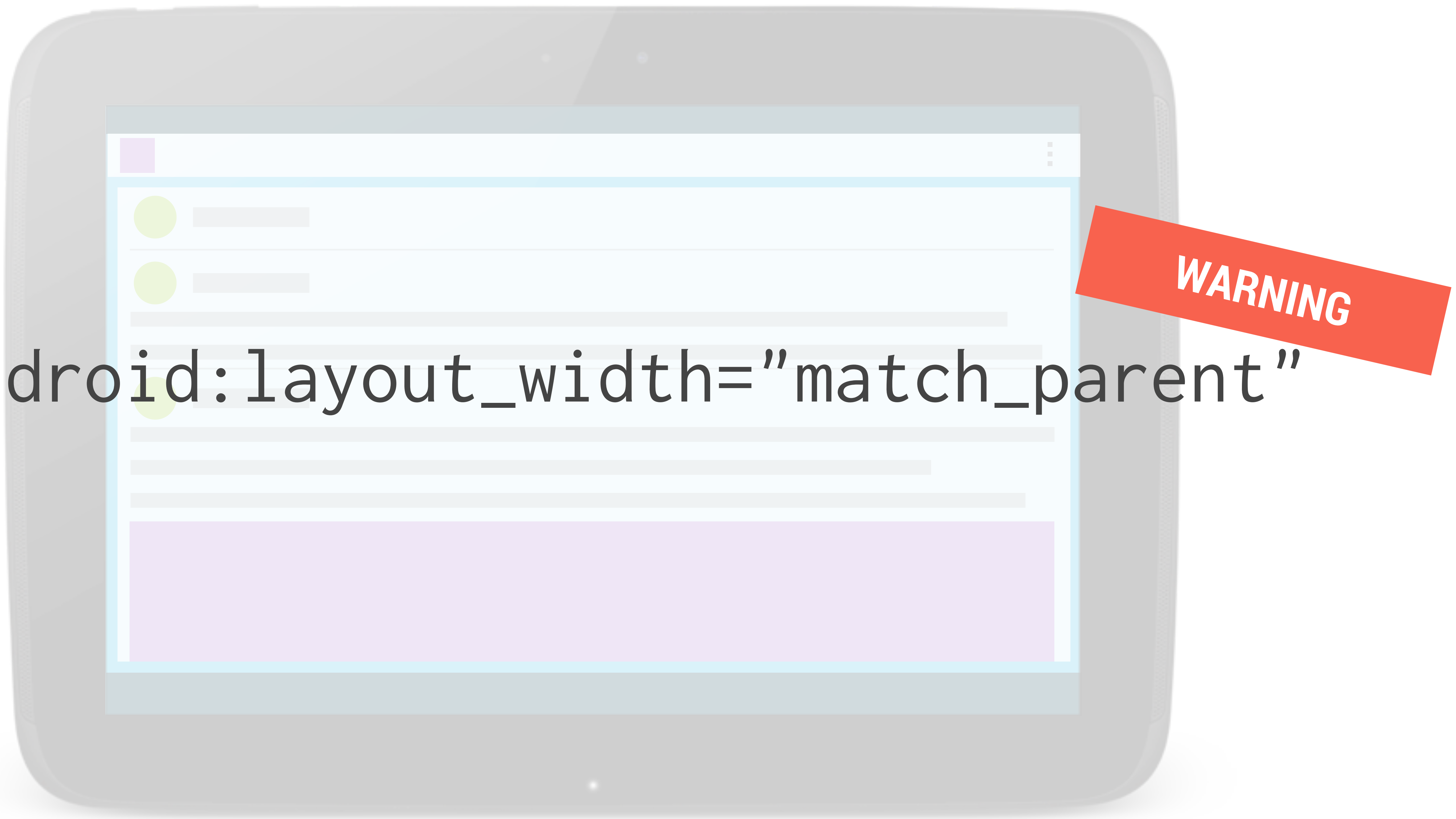
Responsive Design

Device variety



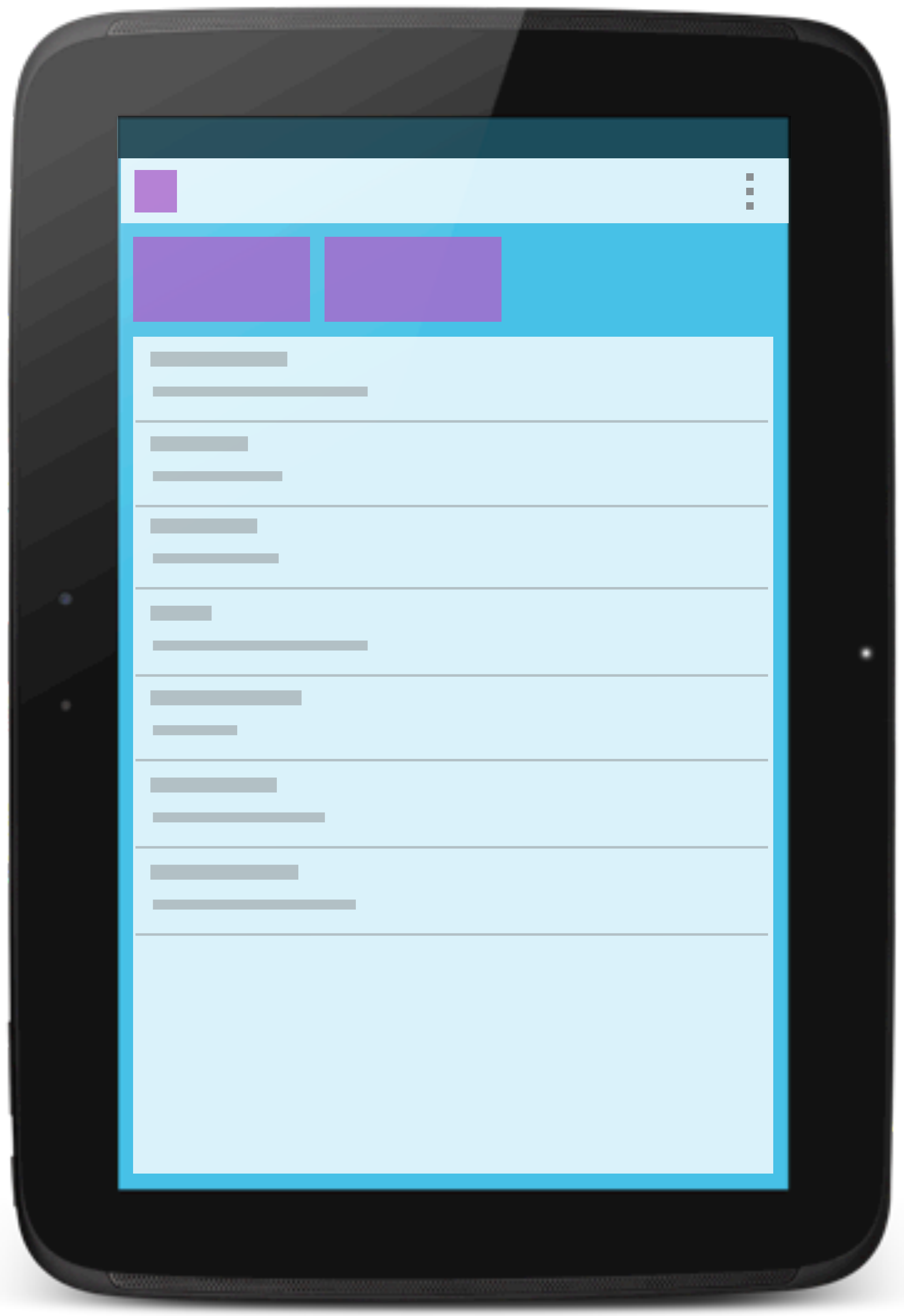
Why responsive?

Why responsive?

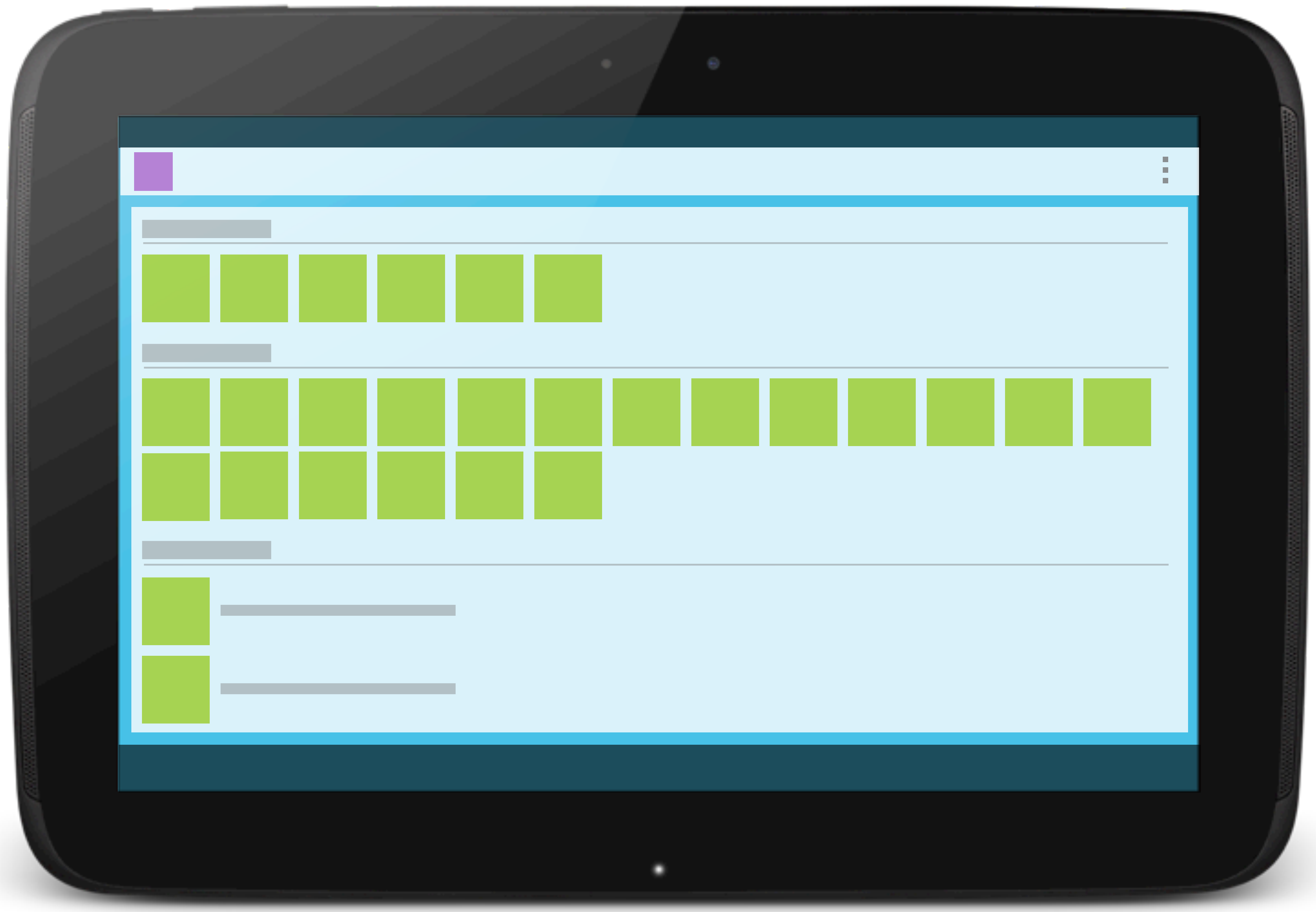


```
android:layout_width=”match_parent”
```

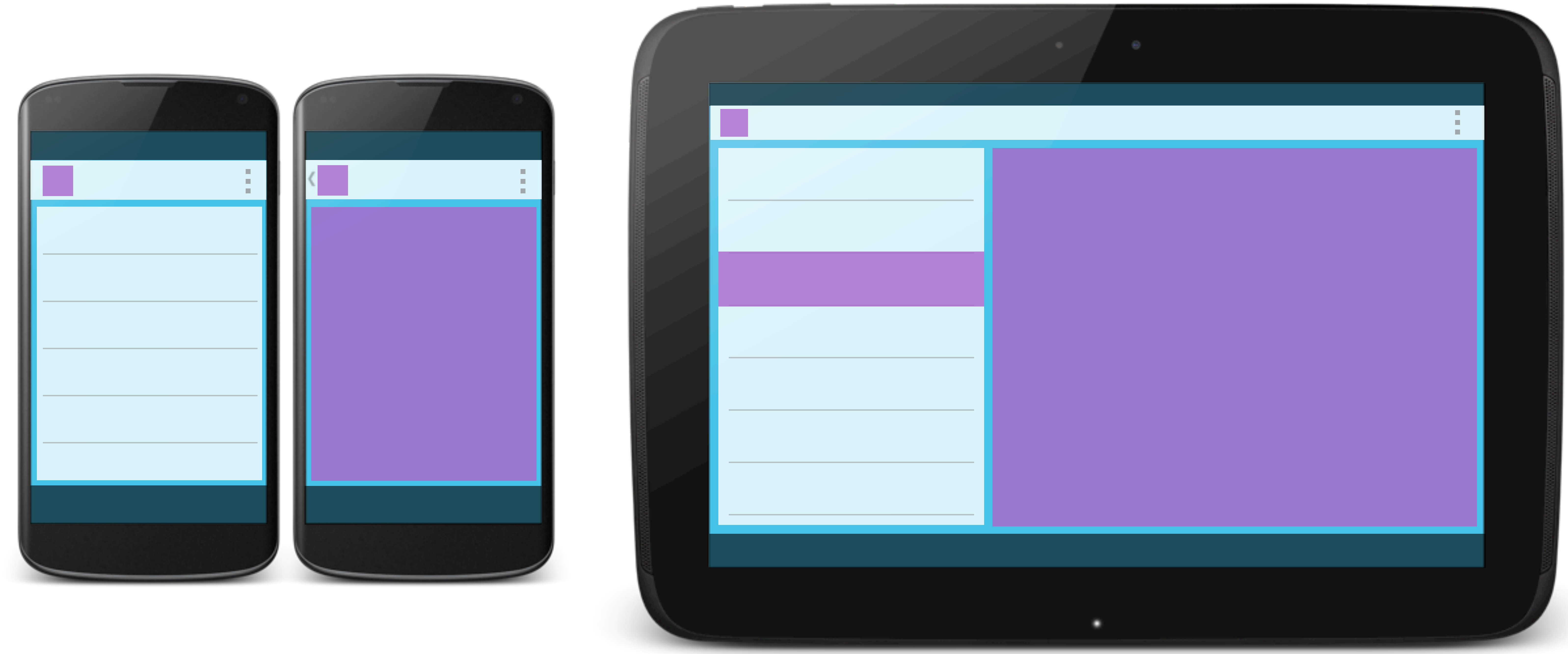
Why responsive?



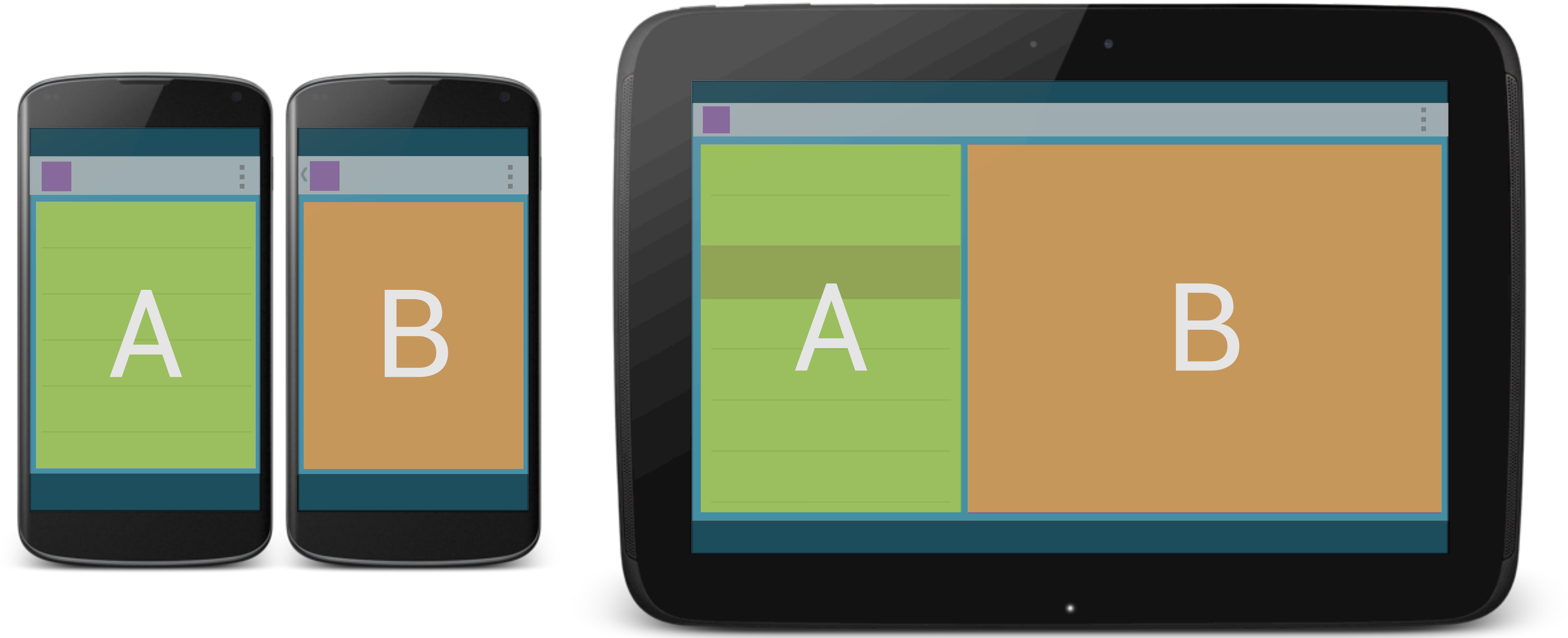
Why responsive?



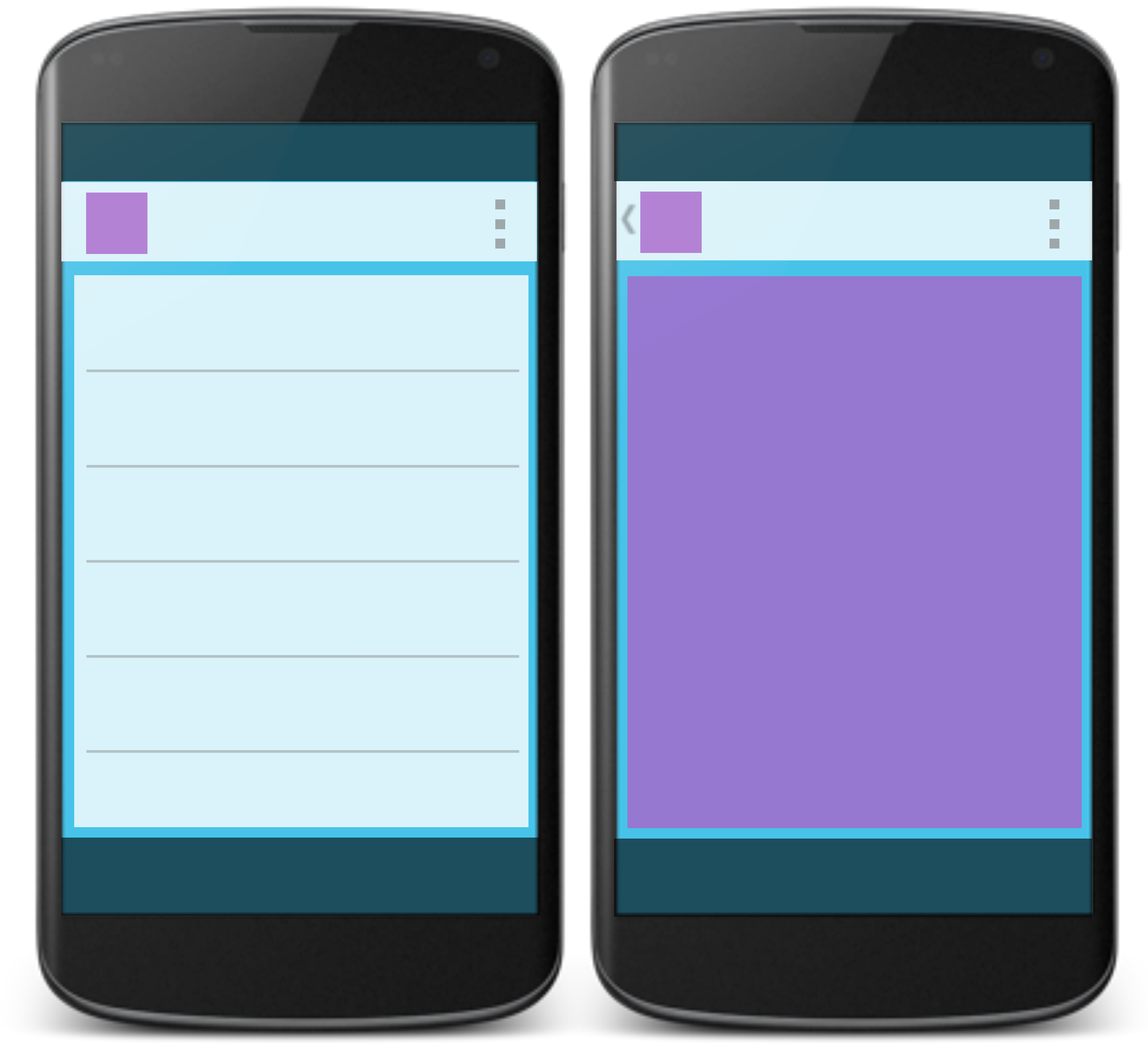
Combination



Fragments



Alternate layouts



*res/layout/**activity_home**.xml*

```
<LinearLayout …>
  <fragment android:name="com.example.ListFragment" />
</LinearLayout>
```

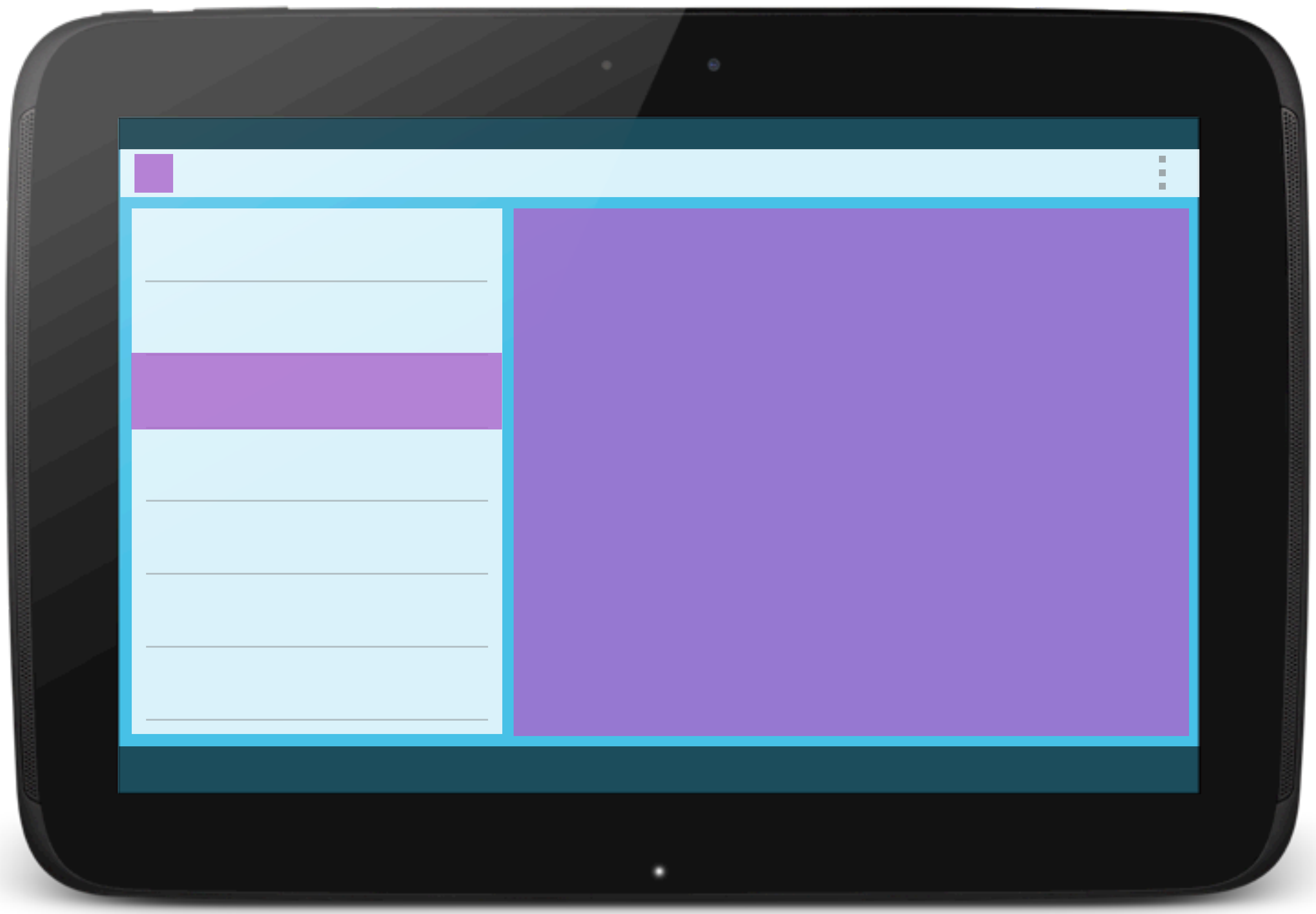
*res/layout/**activity_detail**.xml*

```
<LinearLayout …>
  <fragment android:name="com.example.DetailFragment" />
</LinearLayout>
```

Alternate layouts

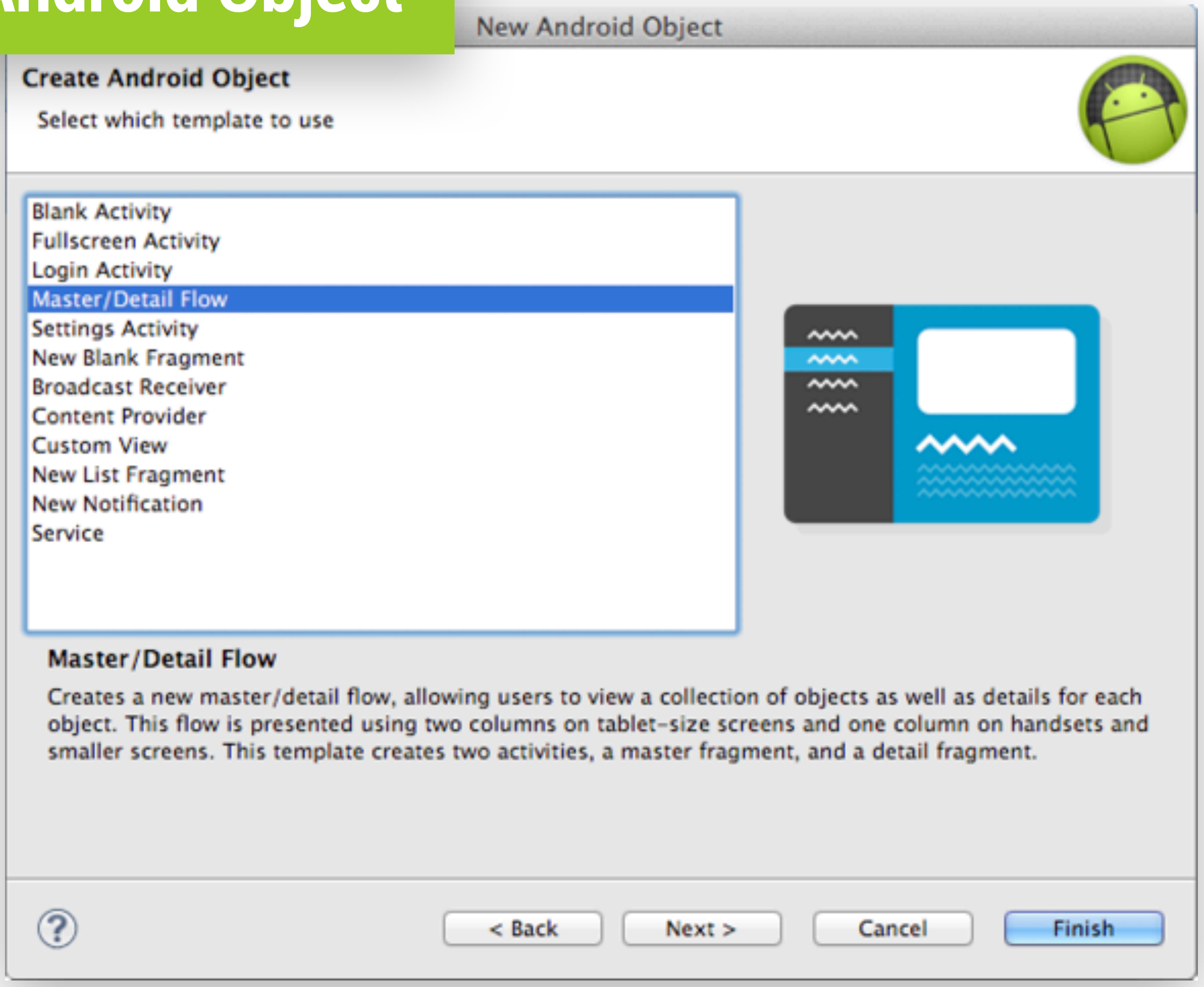
*res/**layout-w600dp**/activity_home.xml*

```
<LinearLayout …>
  <fragment android:name=
    "com.example.ListFragment" />
  <fragment android:name=
    "com.example.DetailFragment" />
</LinearLayout>
```



Master/Detail Flow template

File > New > Android Object


New Android Object

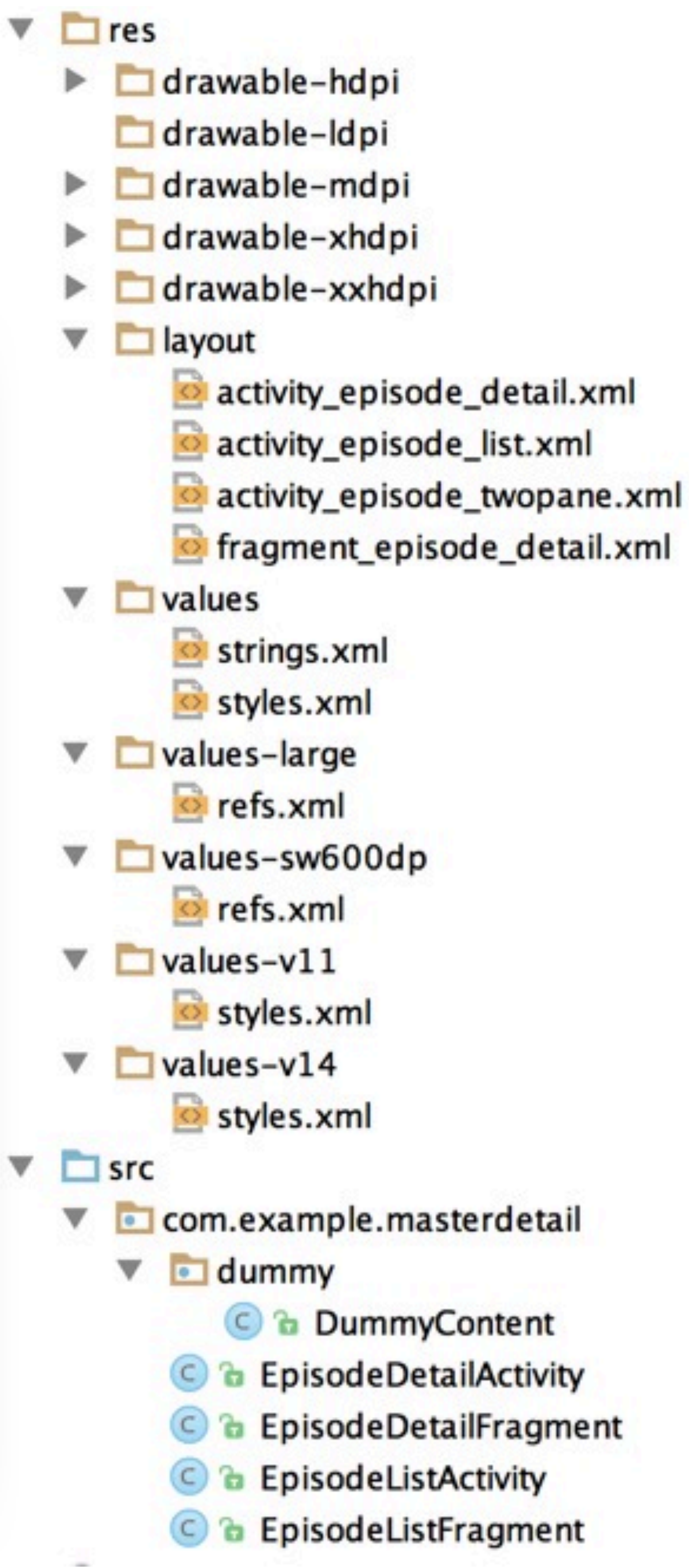
Create Android Object

Select which template to use

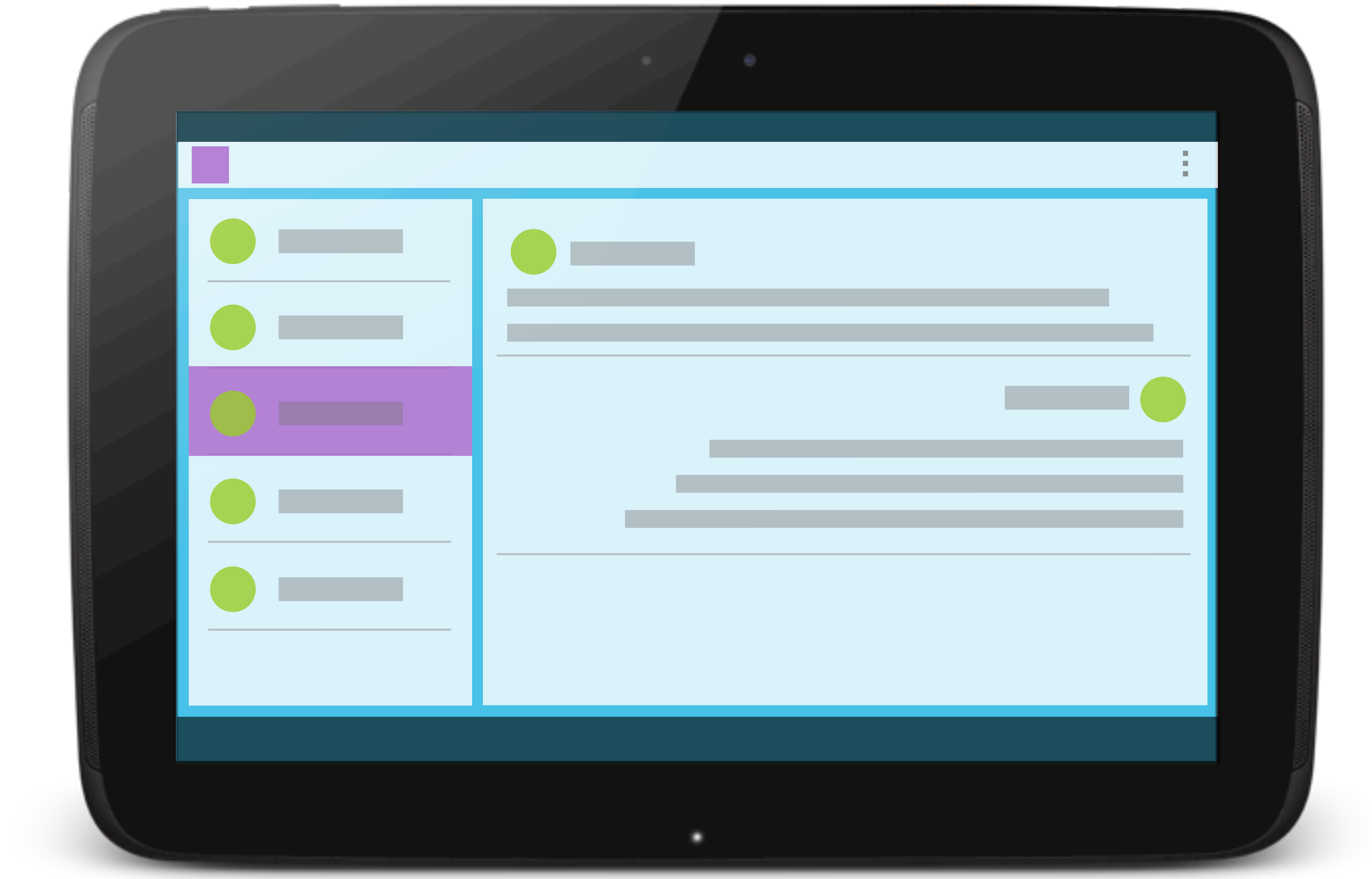
- Blank Activity
- Fullscreen Activity
- Login Activity
- Master/Detail Flow
- Settings Activity
- New Blank Fragment
- Broadcast Receiver
- Content Provider
- Custom View
- New List Fragment
- New Notification
- Service

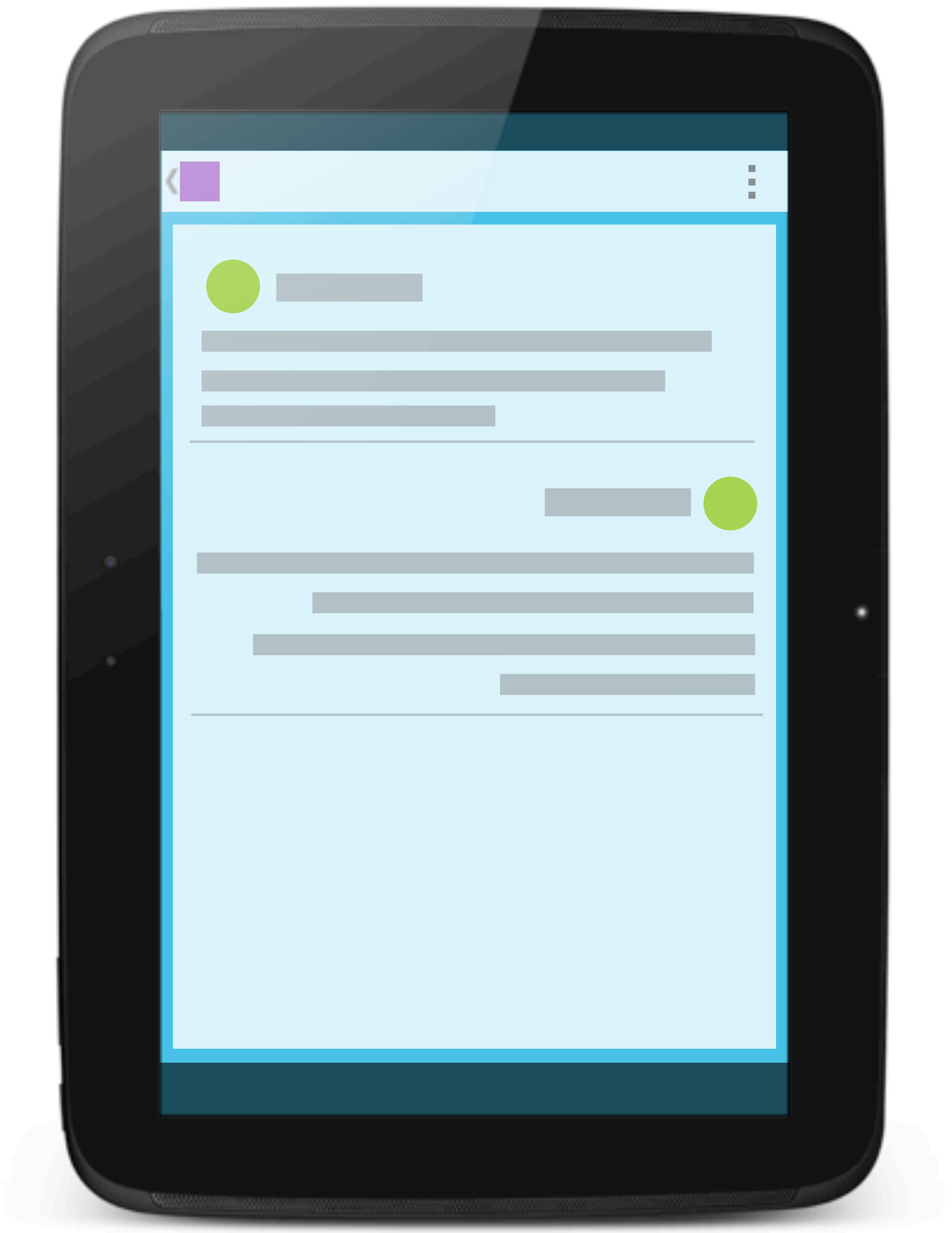
Master/Detail Flow

Creates a new master/detail flow, allowing users to view a collection of objects as well as details for each object. This flow is presented using two columns on tablet-size screens and one column on handsets and smaller screens. This template creates two activities, a master fragment, and a detail fragment.


```
res
  drawable-hdpi
  drawable-ldpi
  drawable-mdpi
  drawable-xhdpi
  drawable-xxhdpi
  layout
    activity_episode_detail.xml
    activity_episode_list.xml
    activity_episode_twopane.xml
    fragment_episode_detail.xml
  values
    strings.xml
    styles.xml
  values-large
    refs.xml
  values-sw600dp
    refs.xml
  values-v11
    styles.xml
  values-v14
    styles.xml
src
  com.example.masterdetail
    dummy
      DummyContent
    EpisodeDetailActivity
    EpisodeDetailFragment
    EpisodeListActivity
    EpisodeListFragment
```

SlidingPaneLayout





SlidingPaneLayout

```
<android.support.v4.widget.SlidingPaneLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/sliding_pane"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- First child is the left pane -->
    <fragment android:name="com.example.ListFragment"
              android:id="@+id/left_pane"
              android:layout_width="280dp"
              android:layout_height="match_parent"
              android:layout_gravity="start" />

    <!-- Second child is the right (content) pane -->
    <fragment android:name="com.example.DetailFragment"
              android:id="@+id/content_pane"
              android:layout_width="600dp"
              android:layout_weight="1"
              android:layout_height="match_parent" />

</android.support.v4.widget.SlidingPaneLayout>
```

SlidingPaneLayout

```
<android.support.v4.widget.SlidingPaneLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/sliding_pane"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- First child is the left pane -->
    <fragment android:name="com.example.ListFragment"
              android:id="@+id/left_pane"
              android:layout_width="280dp"
              android:layout_height="match_parent"
              android:layout_gravity="start" />

    <!-- Second child is the right (content) pane -->
    <fragment android:name="com.example.DetailFragment"
              android:id="@+id/content_pane"
              android:layout_width="600dp"
              android:layout_weight="1"
              android:layout_height="match_parent" />

</android.support.v4.widget.SlidingPaneLayout>
```

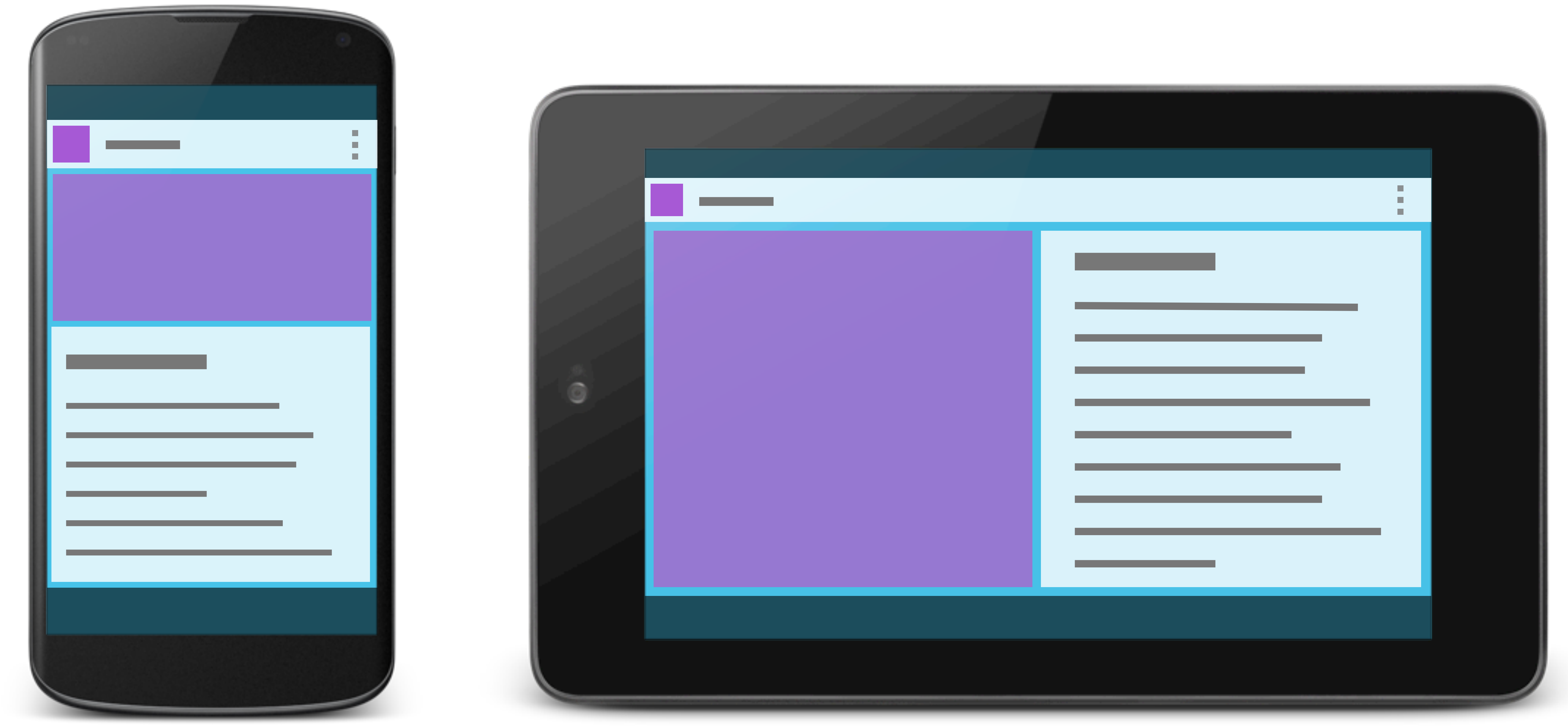
If combined pane widths exceed screen width then right pane overlaps

SlidingPaneLayout

```
<android.support.v4.widget.SlidingPaneLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/sliding_pane"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- First child is the left pane -->
    <fragment android:name="com.example.ListFragment"
              android:id="@+id/left_pane"
              android:layout_width="280dp"
              android:layout_height="match_parent"
              android:layout_gravity="start" />

    <!-- Second child is the right (content) pane -->
    <fragment android:name="com.example.DetailFragment"
              android:id="@+id/content_pane"
              android:layout_width="600dp"
              android:layout_weight="1"
              android:layout_height="match_parent" />
</android.support.v4.widget.SlidingPaneLayout>
```

Grow to consume available space

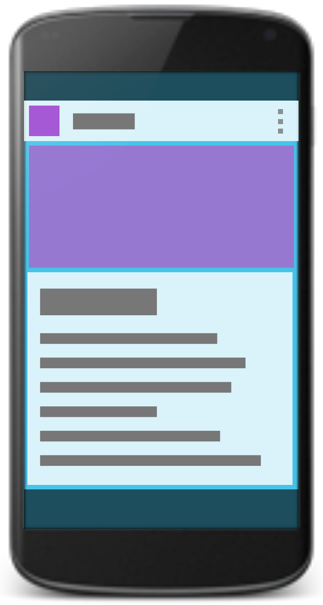
Macro reflow



Alternate layouts

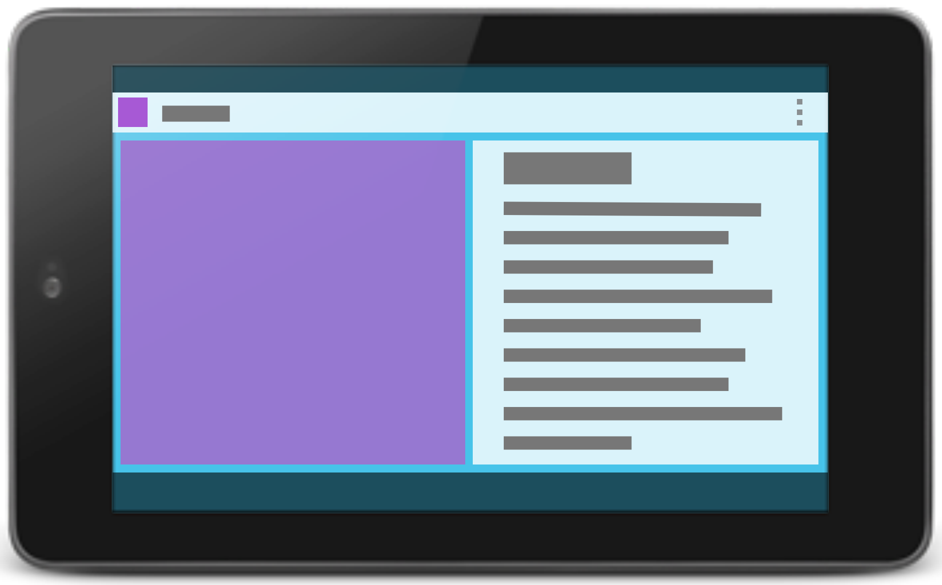
*res/**layout**/activity_home.xml*

```
<LinearLayout
  xmlns:android="…"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

    <fragment … />

    <fragment … />

</LinearLayout>
```

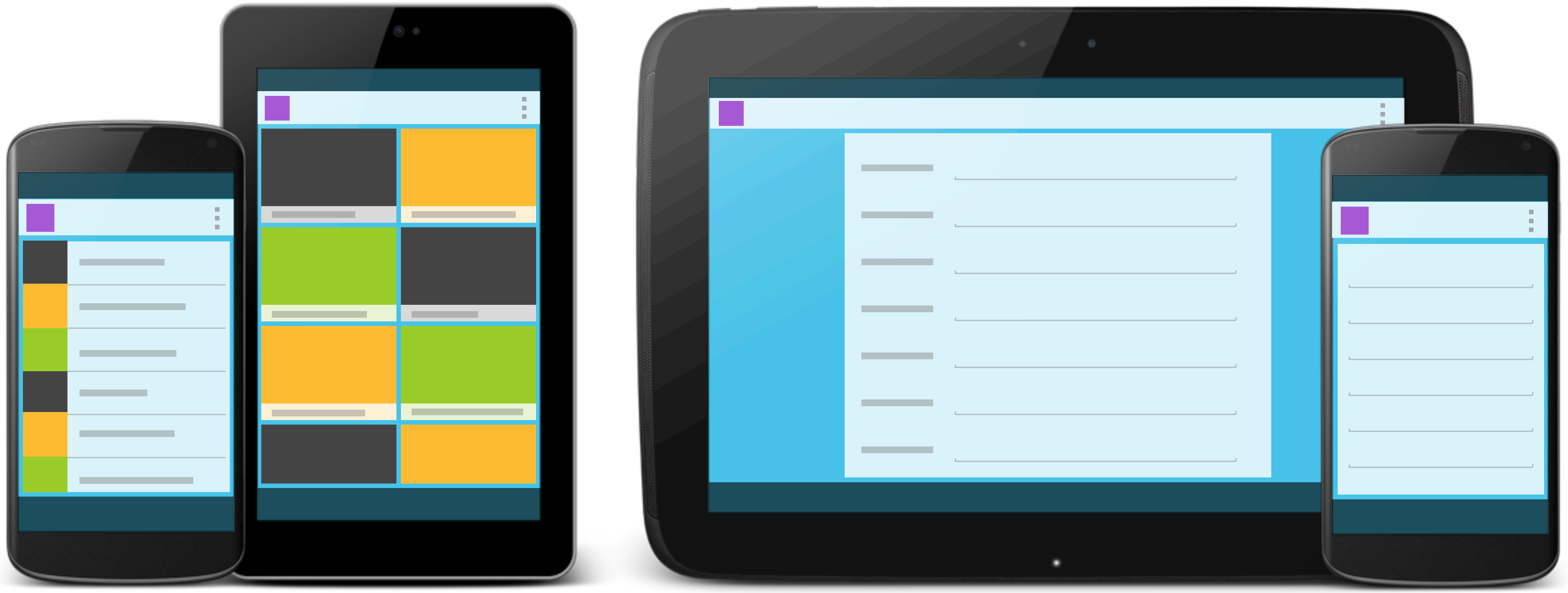


*res/**layout-land**/activity_home.xml*

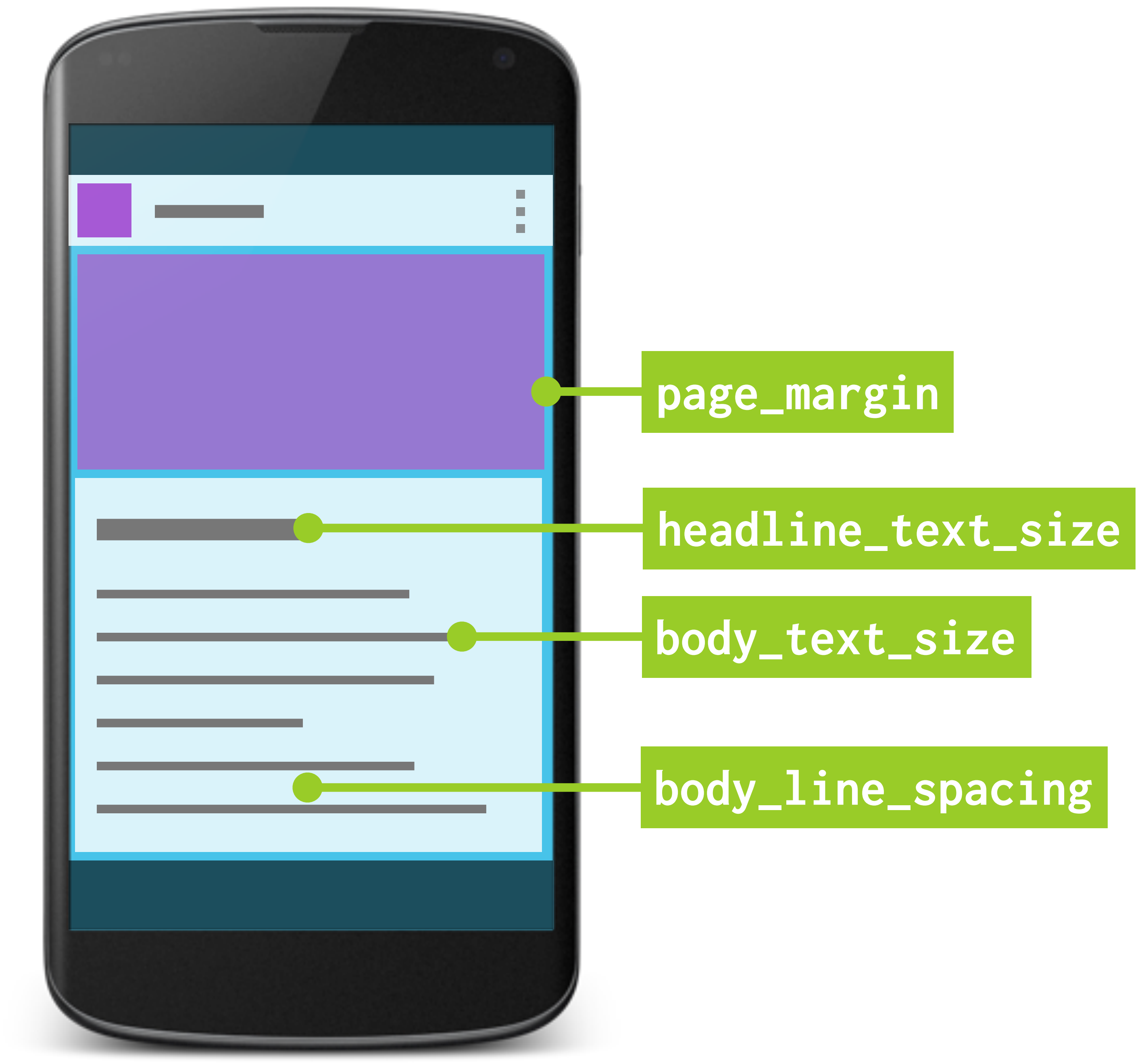
```
<LinearLayout
  xmlns:android="…"
  android:orientation="horizontal"
  android:baselineAligned="false"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

    <fragment … />

    <fragment … />

</LinearLayout>
```



Micro reflow



Dimension files



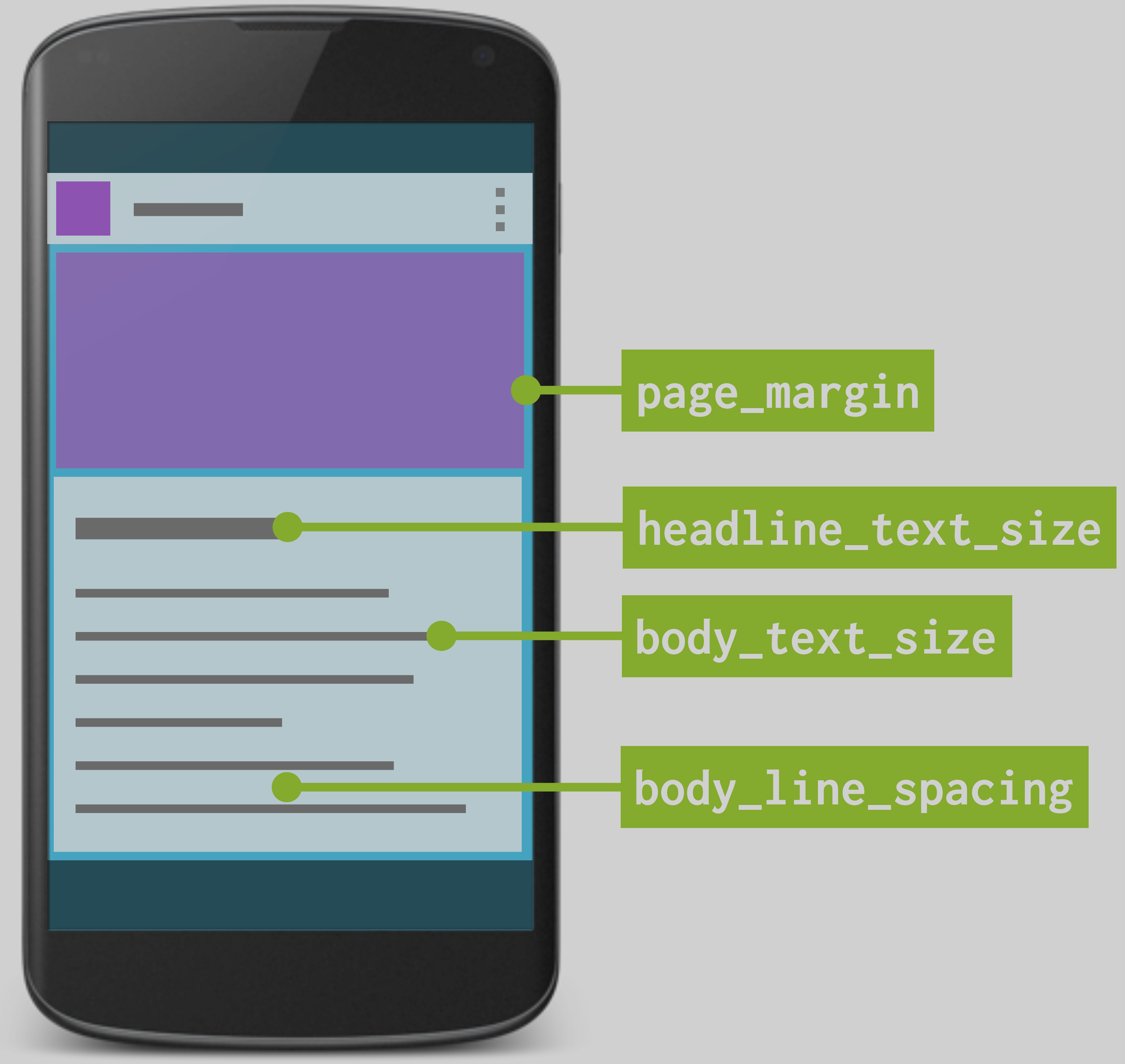
*res/**values**/dimens.xml*

```
<resources>
  …
  <dimen name="page_margin">16dp</dimen>
  <dimen name="body_text_size">18sp</dimen>
  …
</resources>
```

*res/**values-sw720dp**/dimens.xml*

```
<resources>
  …
  <dimen name="page_margin">32dp</dimen>
  <dimen name="body_text_size">22sp</dimen>
  …
</resources>
```

Dimension files



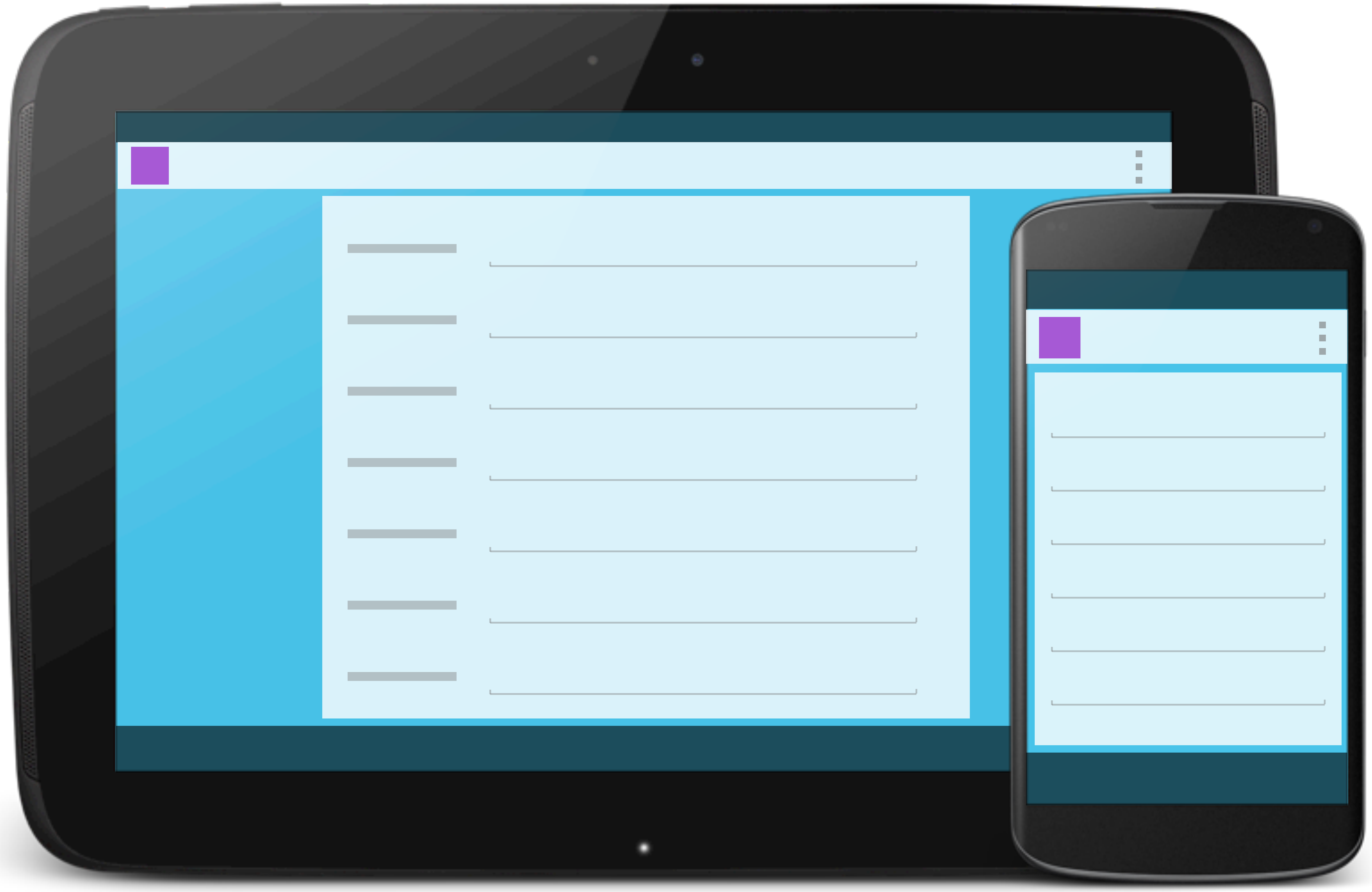
*res/**values**/dimens.xml*

```
<resources>
  …
  <dimen name="page_margin">16dp</dimen>
  <dimen name="body_text_size">18sp</dimen>
  …
</resources>
```

*res/**values-sw720dp**/dimens.xml*

```
<resources>
  …
  <dimen name="page_margin">32dp</dimen>
  <dimen name="body_text_size">22sp</dimen>
  …
</resources>
```

Margin point



Margin point

*res/**layout**/activity_home.xml*

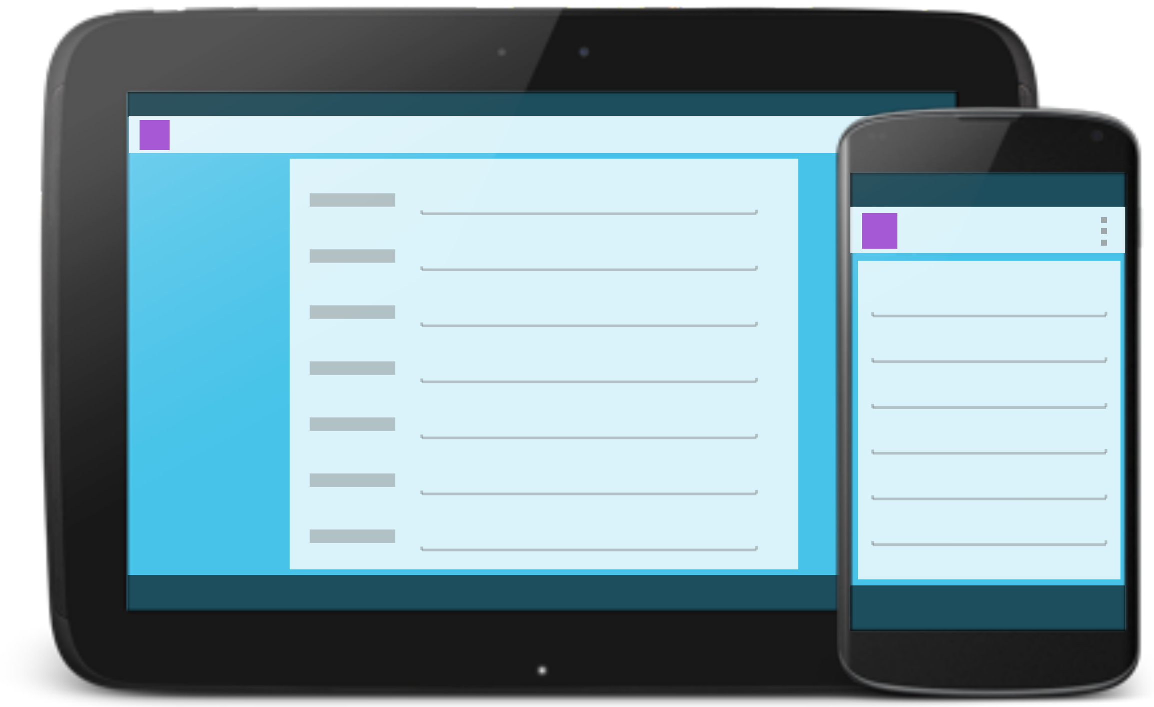
```
<FrameLayout xmlns:android="…"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  <ScrollView style="@style/MarginPoint"
              android:layout_height="match_parent"
              android:scrollbarStyle="outsideOverlay">
```

*res/**values**/styles.xml*

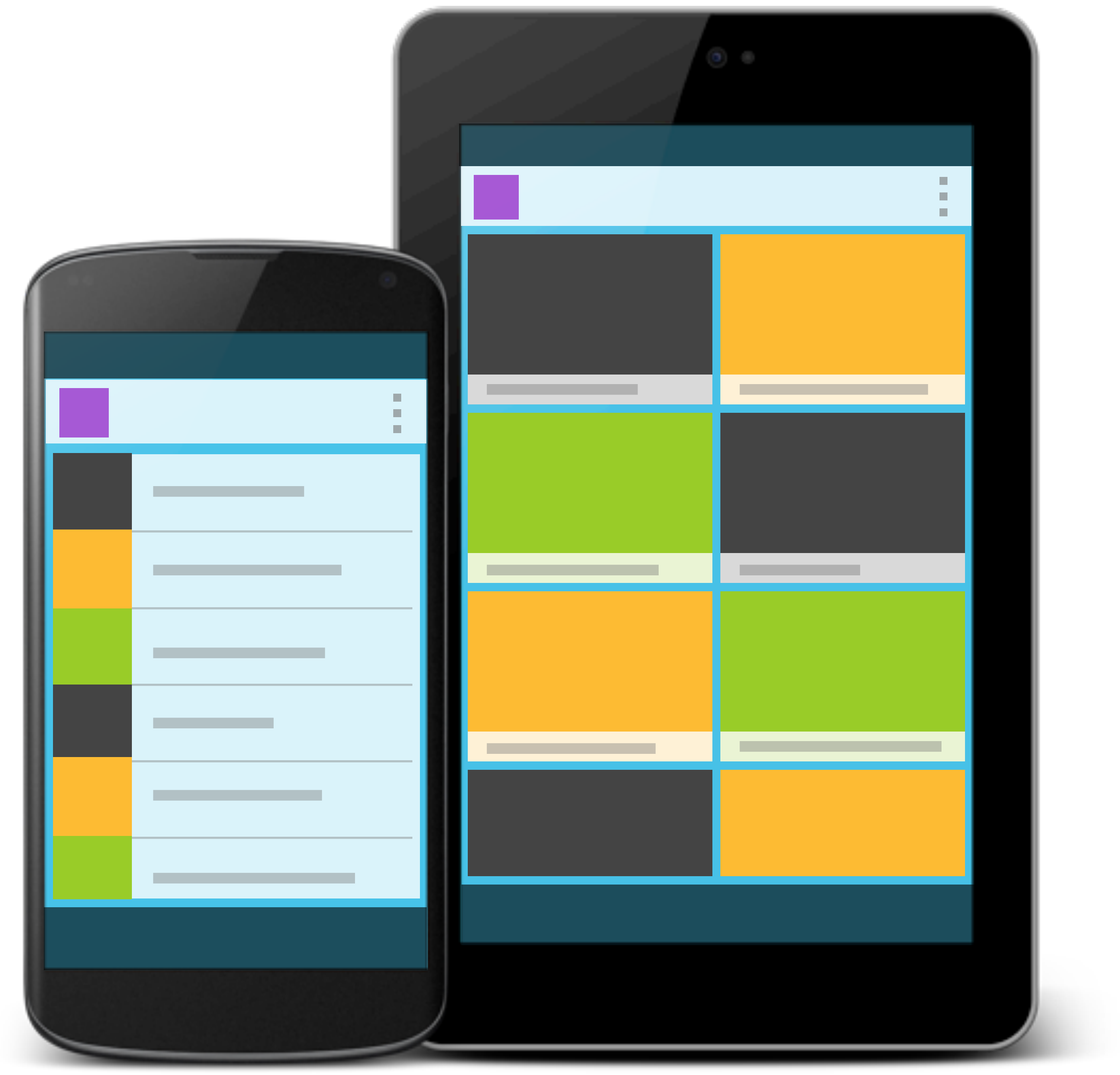
```
<style name="MarginPoint">
  <item name="android:layout_width">match_parent</item>
</style>
```

*res/**values-w600dp**/styles.xml*

```
<style name="MarginPoint">
  <item name="android:layout_width">600dp</item>
  <item name="android:layout_gravity">center_horizontal</item>
</style>
```



Lists to grids



Lists to grids

*res/**layout**/activity_home.xml*

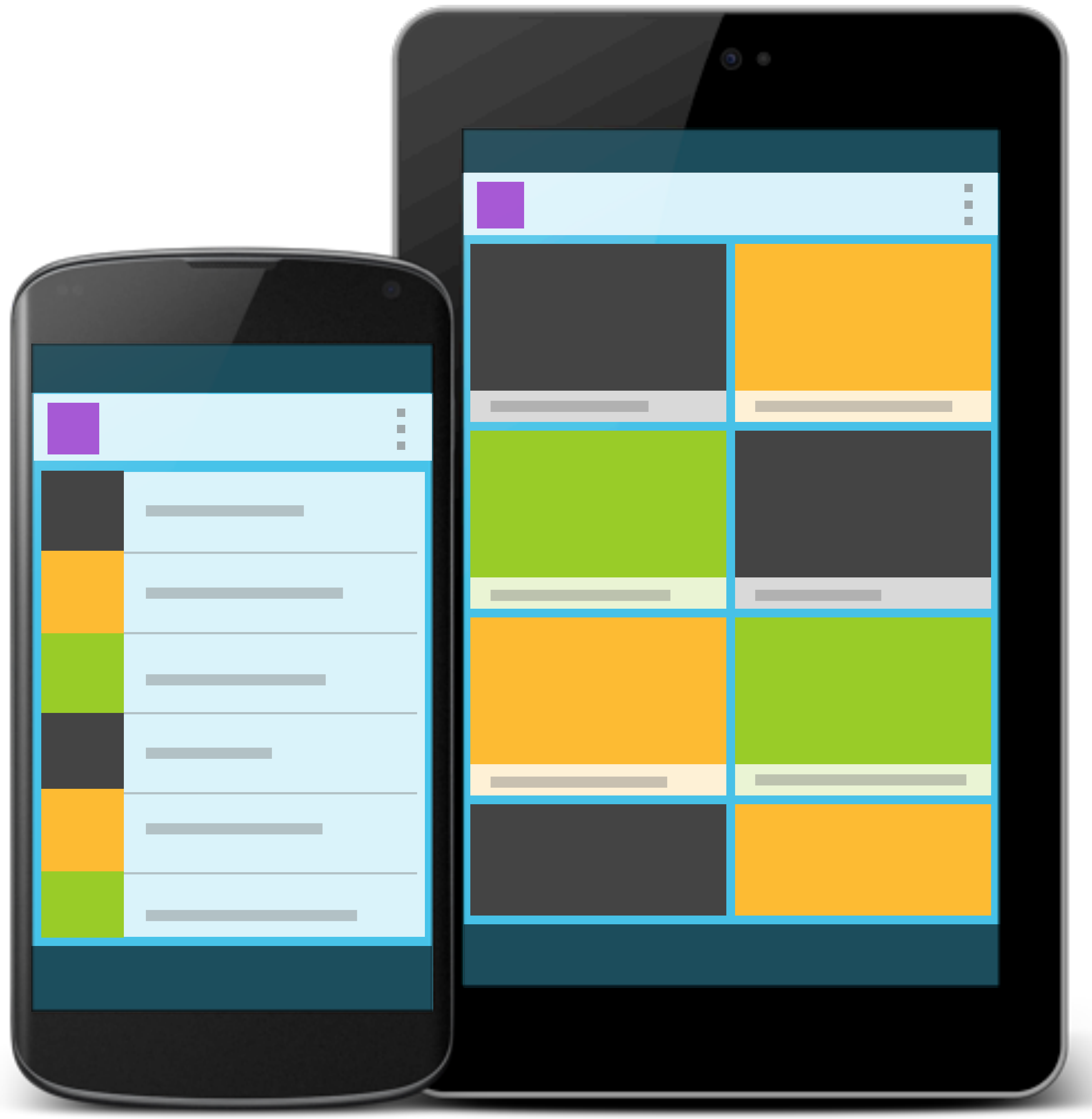
```
<GridView …
  android:numColumns="@integer/num_columns" />
```

*res/**values**/integers.xml*

```
<resources>
  <integer name="num_columns">1</integer>
</resources>
```

*res/**values-w500dp**/integers.xml*

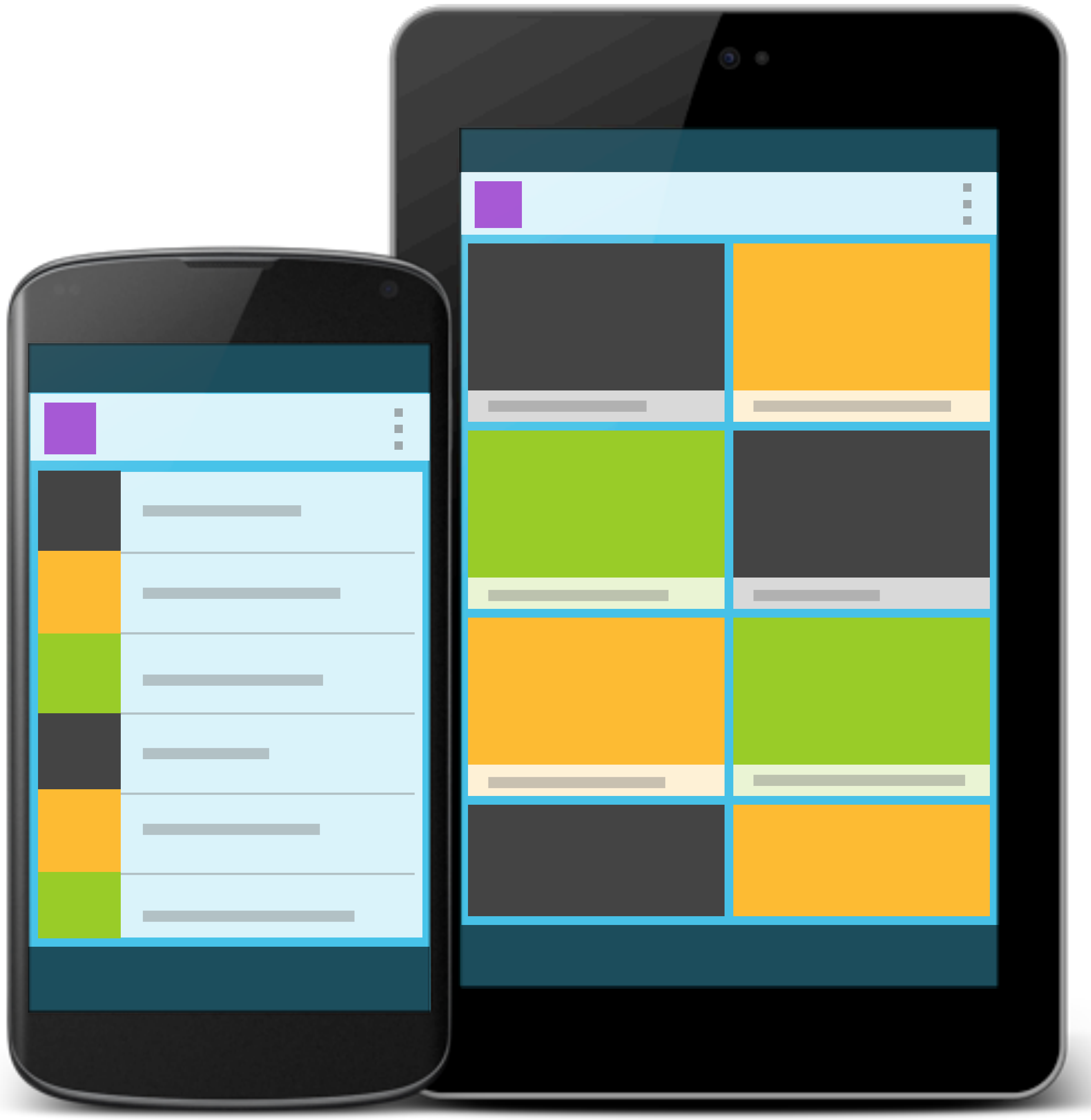
```
<resources>
  <integer name="num_columns">2</integer>
</resources>
```



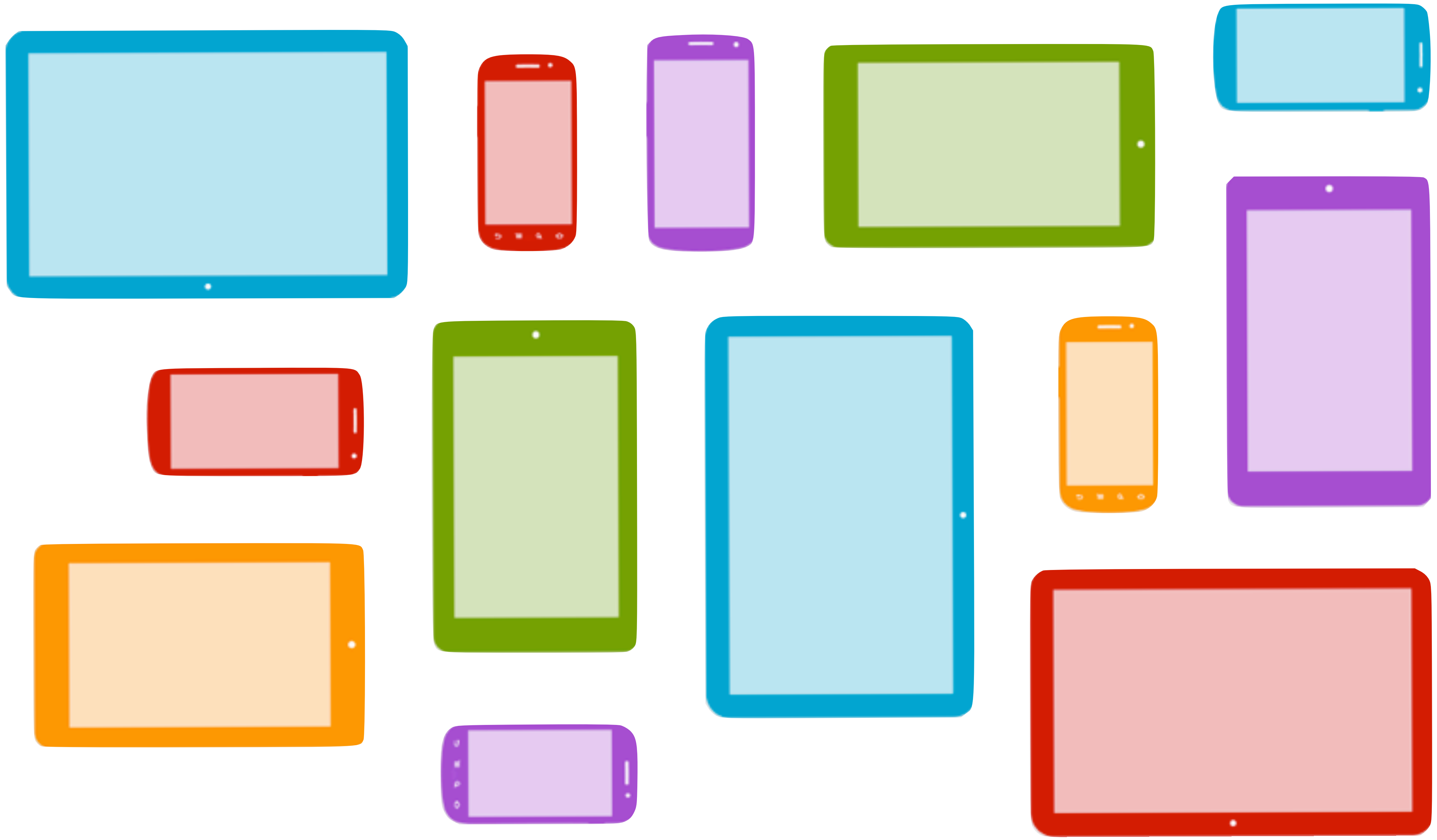
Lists to grids

MyAdapter#getView

```
if (convertView == null) {
   int numColumns =
   getResources().getInteger(R.integer.num_columns);
   if (numColumns == 1) {
      convertView =
      inflater.inflate(R.layout.list_item_layout,
      parent, false);
   } else {
      convertView =
      inflater.inflate(R.layout.grid_item_layout,
      parent, false);
   }
}
```



Responsive design



Holo Visual Language

Style Hierarchy

*res/**values**/styles.xml*

```
<style name="Theme.Base" parent="Theme.Light" />
<style name="Theme.MyTheme" parent="Theme.Base" />
```

*res/**values-v11**/styles.xml*

```
<style name="Theme.Base" parent="Theme.Holo.Light" />
```

Style Hierarchy

*res/**values**/styles.xml*

```
<style name="Theme.Base" parent="Theme.Light" />
<style name="Theme.MyTheme" parent="Theme.Base" />
```

*res/**values-v11**/styles.xml*

```
<style name="Theme.Base" parent="Theme.Holo.Light" />
```

*res/**values**/styles.xml*

COMING SOON

```
<style name="Theme.MyTheme" parent="Theme.AppCompat.Light" />
```

Built-in framework resources

Java: `android.R.attr.foo`

XML: `?android:foo` or `?android:attr/foo`

Example dimension resources:

`?android:actionBarSize`

`?android:listPreferredItemHeight(Small)`

`?android:listPreferredItemPaddingLeft`

Example style resources:

`?android:progressBarStyleLarge`

`?android:borderlessButtonStyle`

`?android:listSeparatorTextViewStyle`

`?android:textAppearanceListItemSmall`

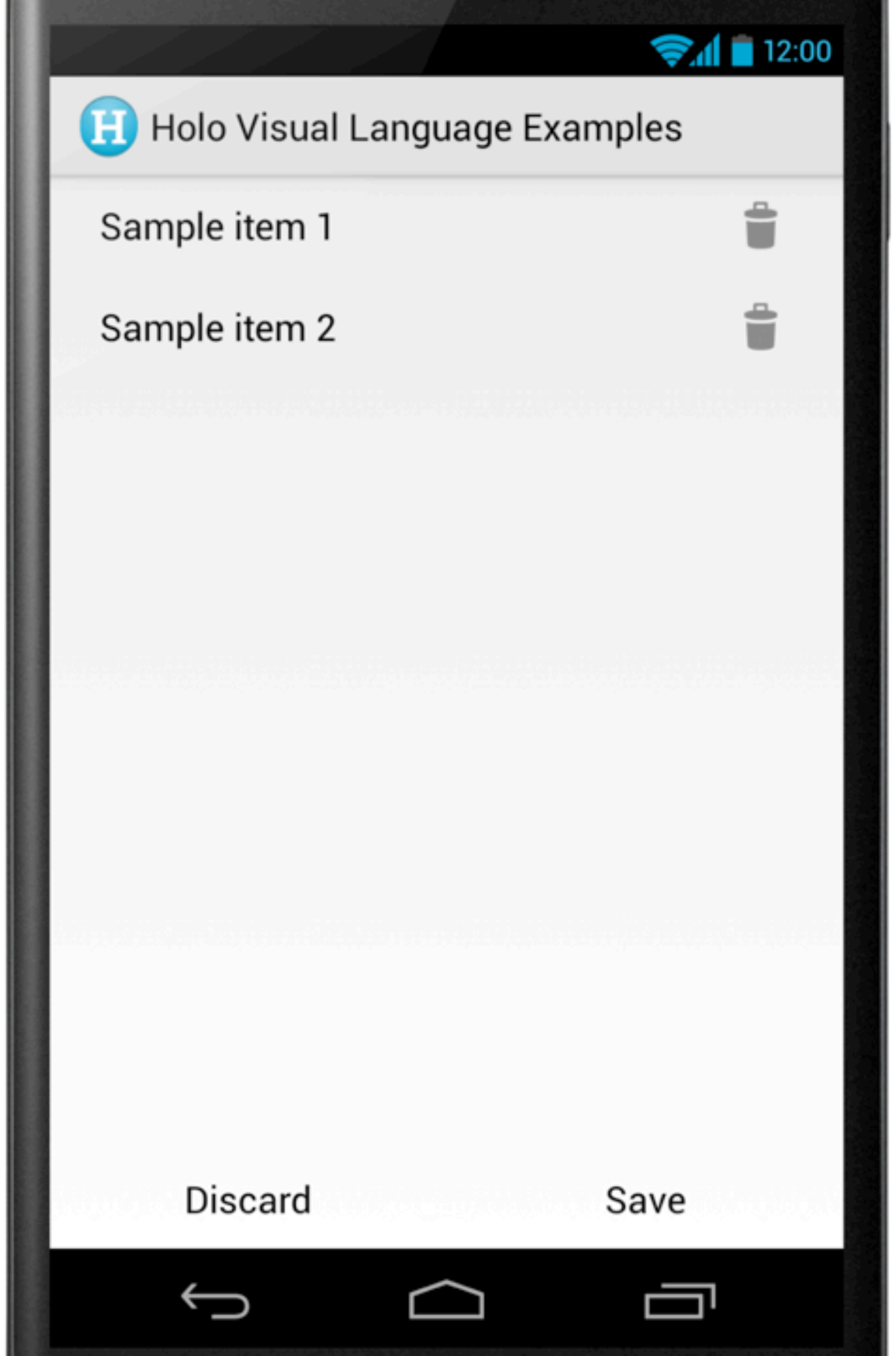
Example drawable resources:

`?android:listChoiceIndicatorSingle`

`?android:dividerHorizontal`

`?android:selectableItemBackground`

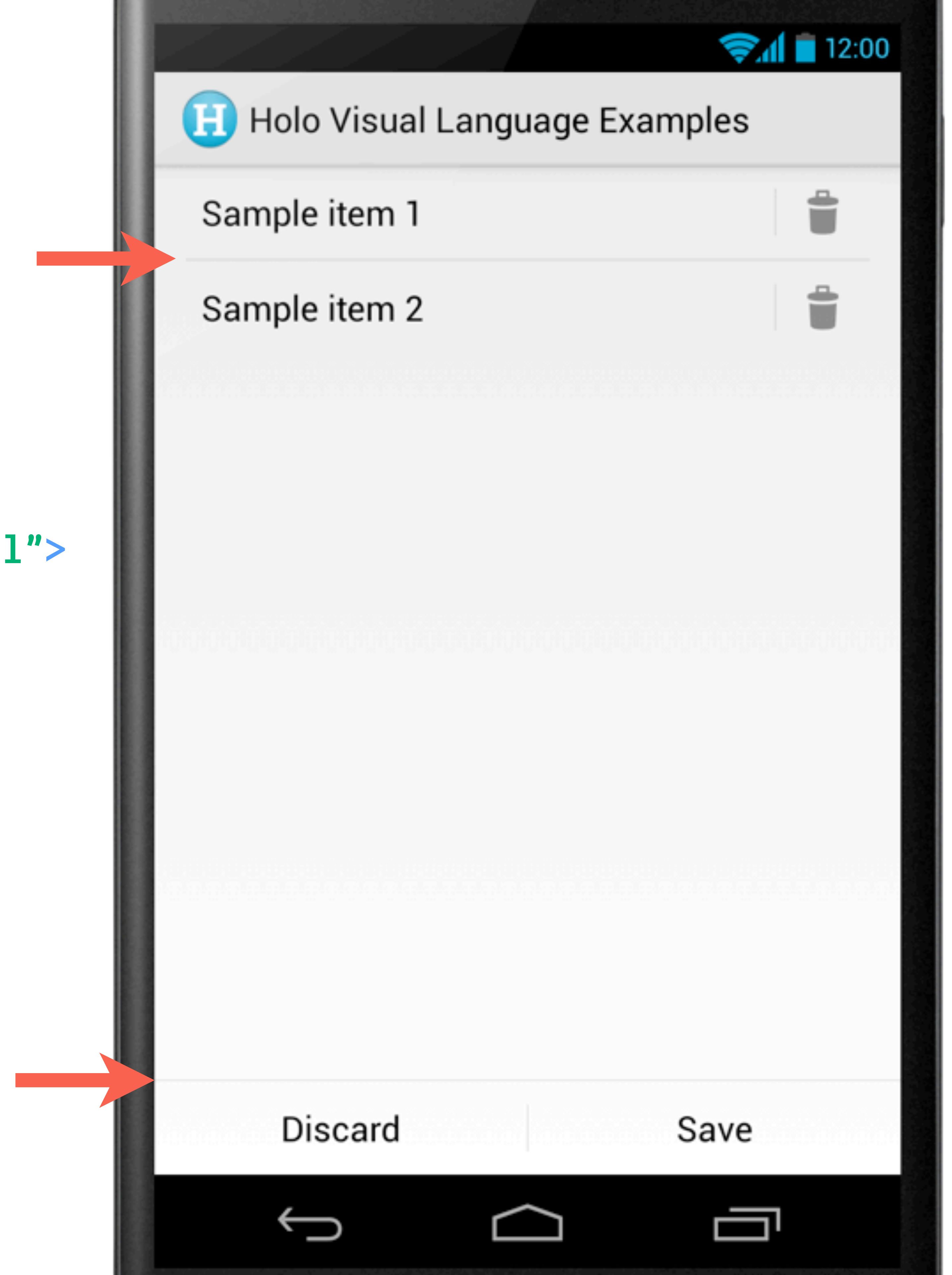
`?android:textSelectHandleLeft`


12:00
Holo Visual Language Examples
Sample item 1
Sample item 2
Discard
Save

Dividers

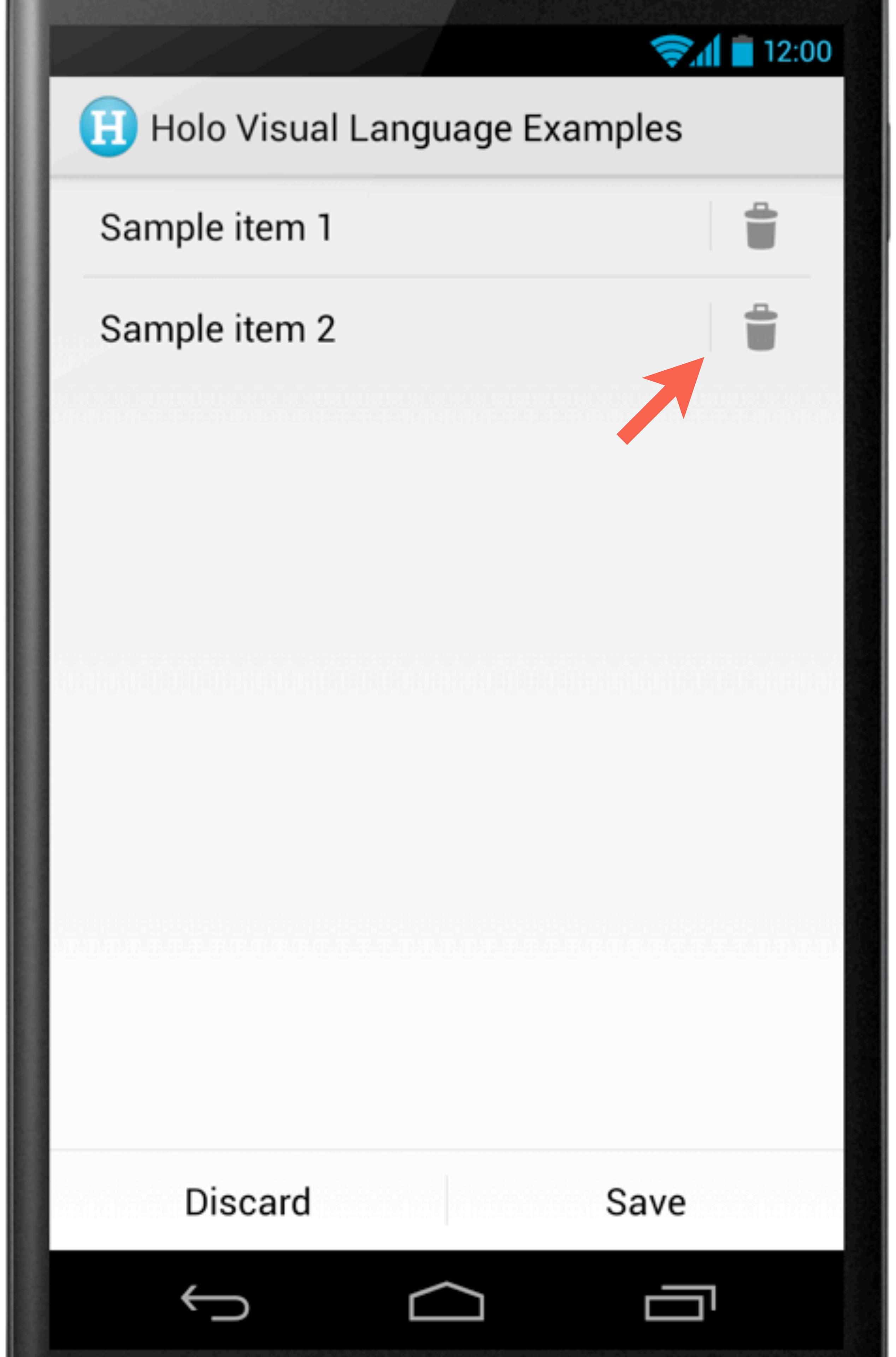
```
<LinearLayout
    android:orientation="vertical"
    ...
    android:showDividers="middle"
    android:divider="?android:dividerHorizontal">
```

Dividers and spacing (e.g. margins) establish hierarchy



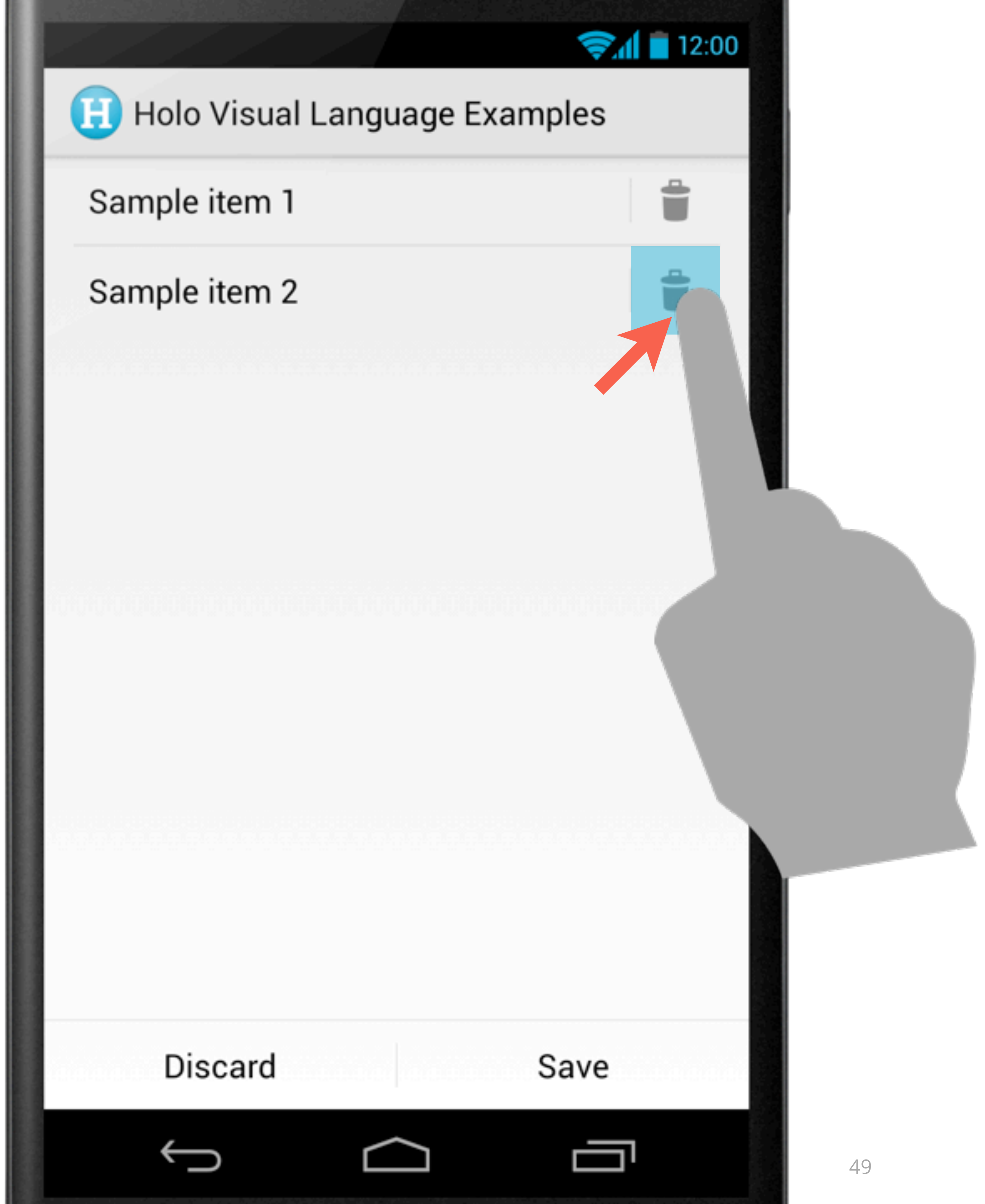
Dividers

```
<LinearLayout
    android:orientation="horizontal"
    ...
    android:showDividers="middle"
    android:divider="?android:dividerVertical"
    android:dividerPadding="8dp"
    android:baselineAligned="false">
```



Borderless buttons

```
<ImageButton
    style="?android:borderlessButtonStyle"
    android:layout_width="48dp"
    android:layout_height="match_parent"
    android:src="@drawable/ic_action_delete"
    android:contentDescription="@string/delete" />
```

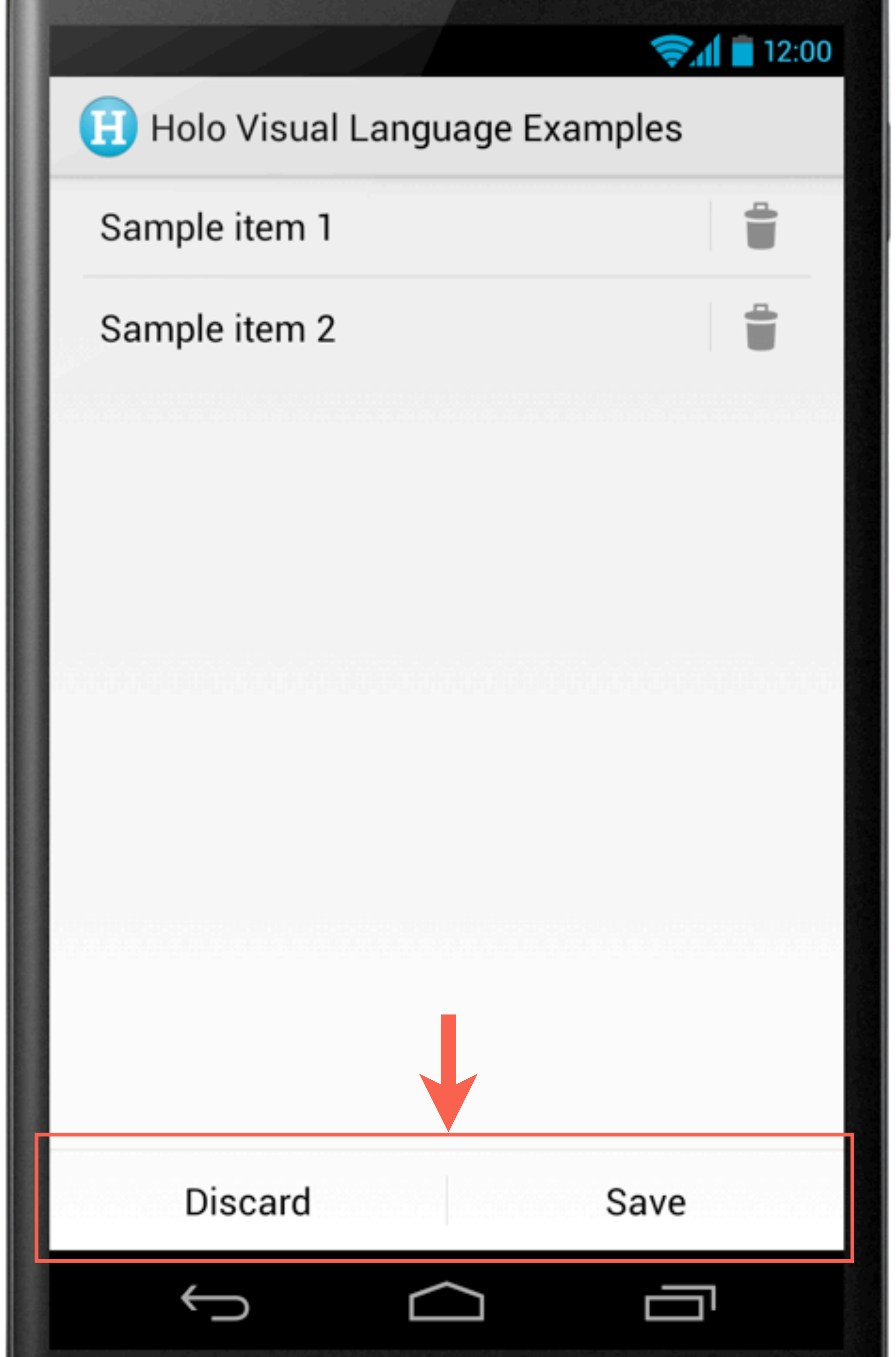


Button bars

```
<LinearLayout style="?android:buttonBarStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
```

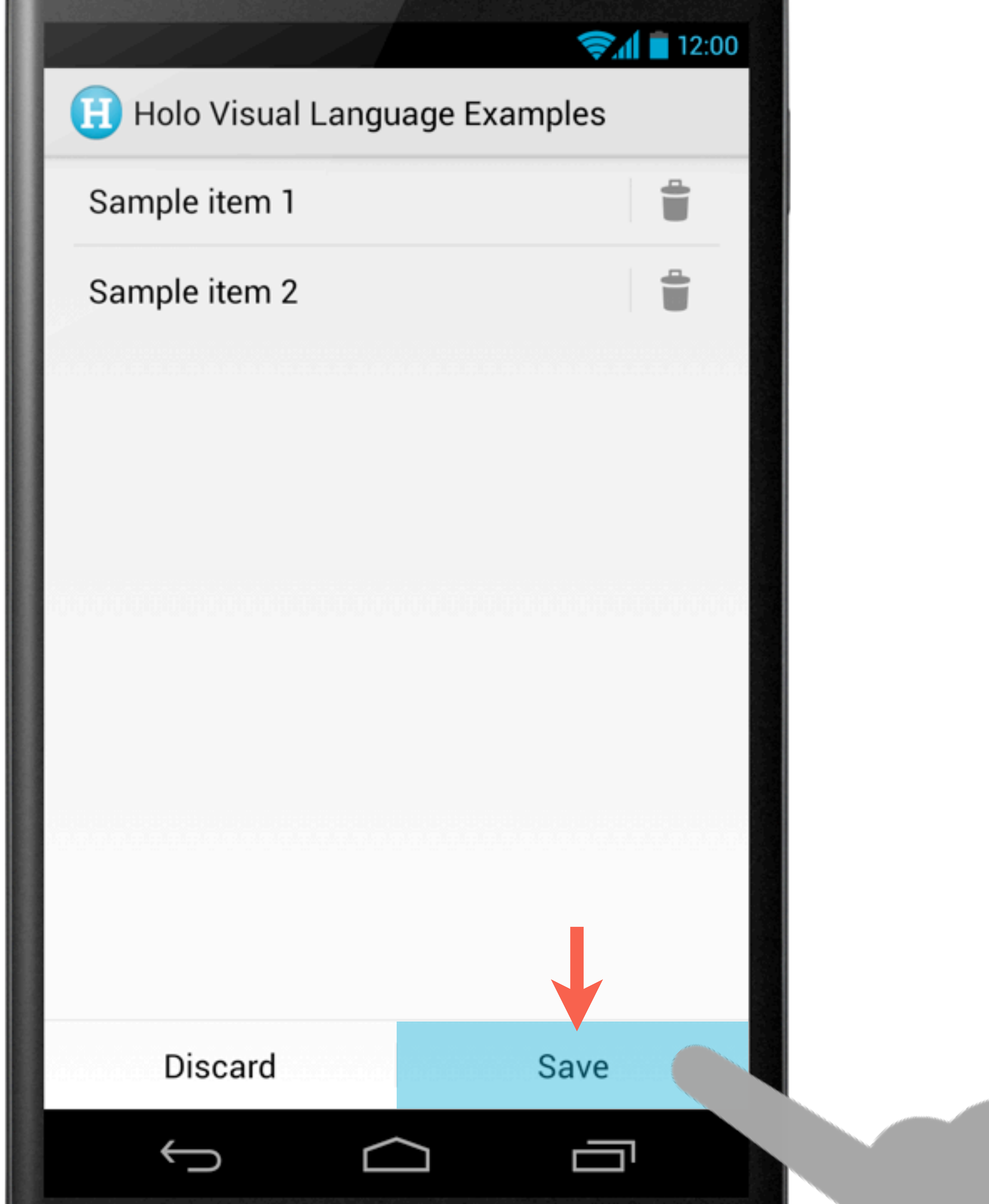
Button bar style automatically adds dividers between buttons

Remember, preferred action on the right!



Button bars

```
<Button style="?android:buttonBarButtonStyle"
    android:layout_width="0dp"
    android:layout_weight="1"
    android:layout_height="wrap_content"
    android:text="@string/save" />
```



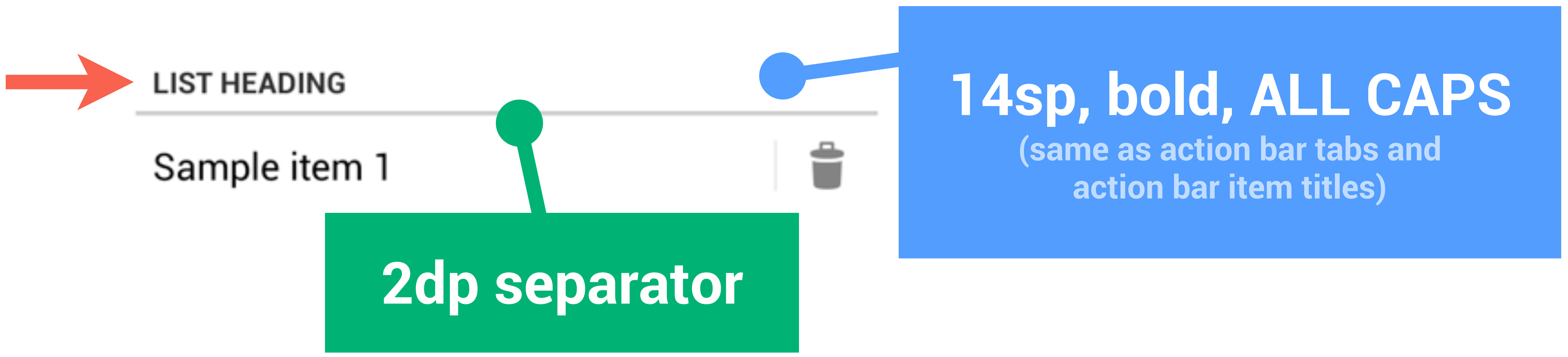
Touch feedback

```
<FrameLayout android:layout_width="match_parent"
    android:layout_height="200dp"
    android:clickable="true"
    android:focusable="true"
    android:foreground="?android:selectableItemBackground"
    android:contentDescription="Item title here">
    ...
```

Or `android:background` *on any view type*

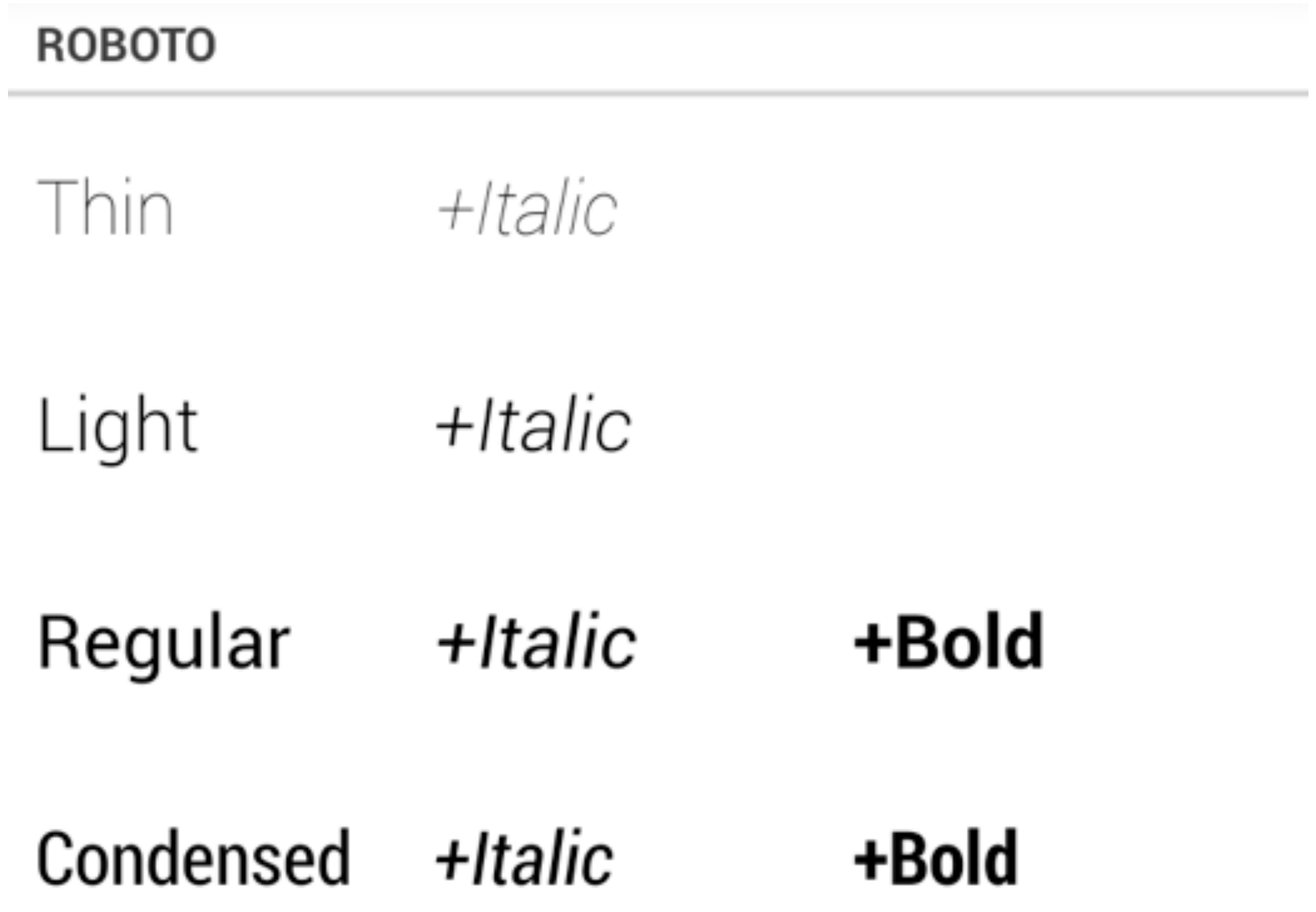
`android:listSelector` *for lists and grids*

List headings



```
<TextView
    style="?android:listSeparatorTextViewStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="List heading" />
```

Sophisticated typography with `TextView`



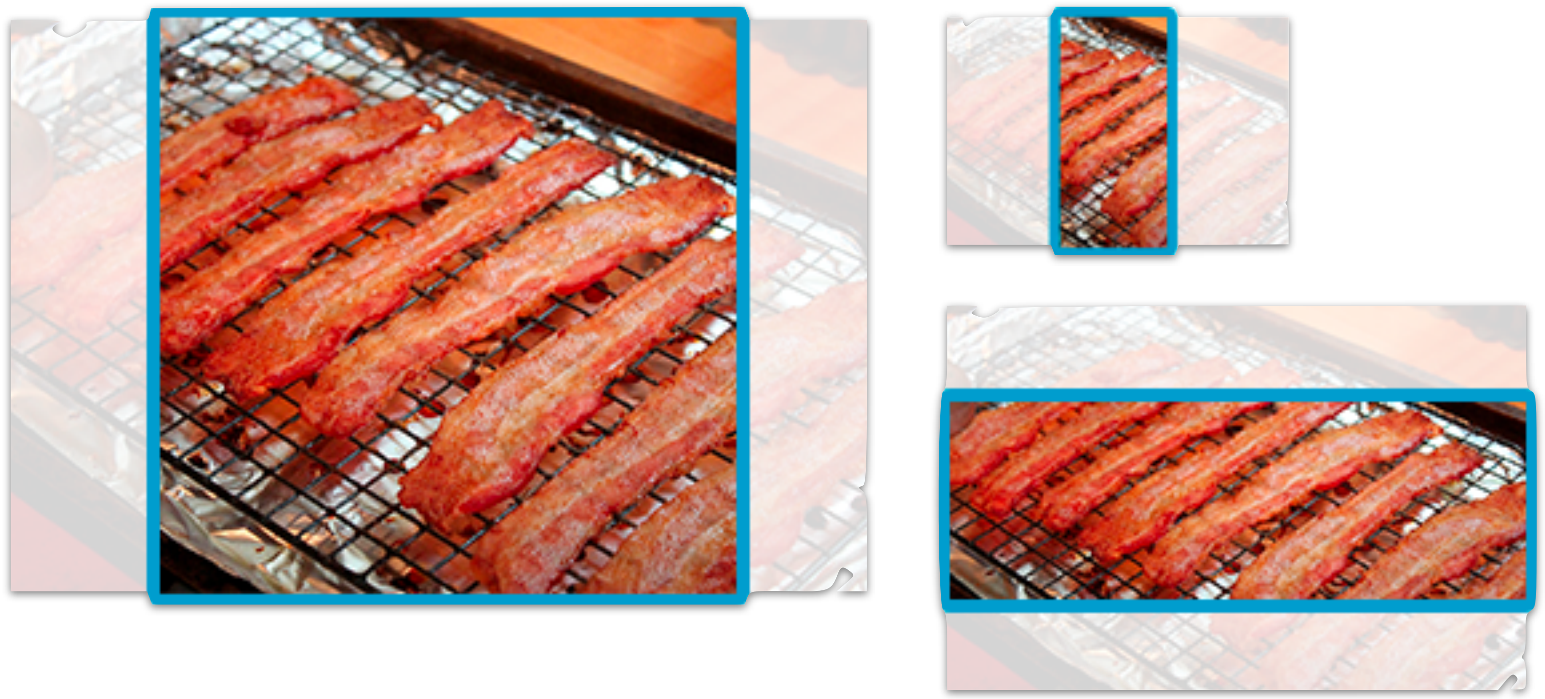
	ROBOTO		
`android:fontFamily="sans-serif-thin"`	Thin	*+Italic*	
`android:fontFamily="sans-serif-light"`	Light	*+Italic*	
(default)	Regular	*+Italic*	**+Bold**
`android:fontFamily="sans-serif-condensed"`	Condensed	*+Italic*	**+Bold**

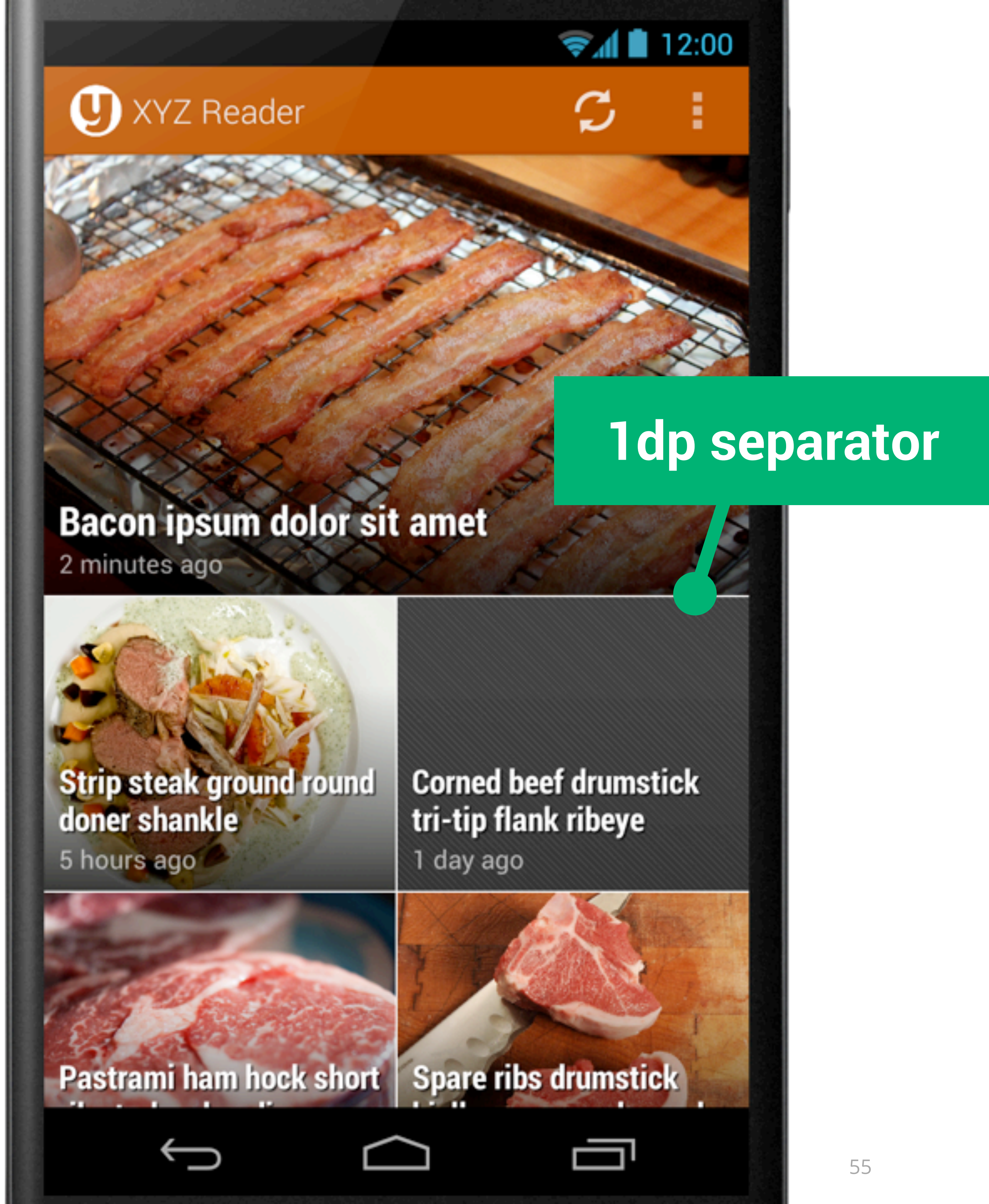
Download standard, condensed, and slab at
Google Fonts

Full-bleed images with consistent aspect ratios

```
<ImageView android:scaleType="centerCrop"
    android:src="@drawable/p1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

centerCrop:




12:00
XYZ Reader
Bacon ipsum dolor sit amet
2 minutes ago
Strip steak ground round doner shankle
5 hours ago
Corned beef drumstick tri-tip flank ribeye
1 day ago
Pastrami ham hock short
Spare ribs drumstick
1dp separator

Generating Holo assets

Android Asset Studio

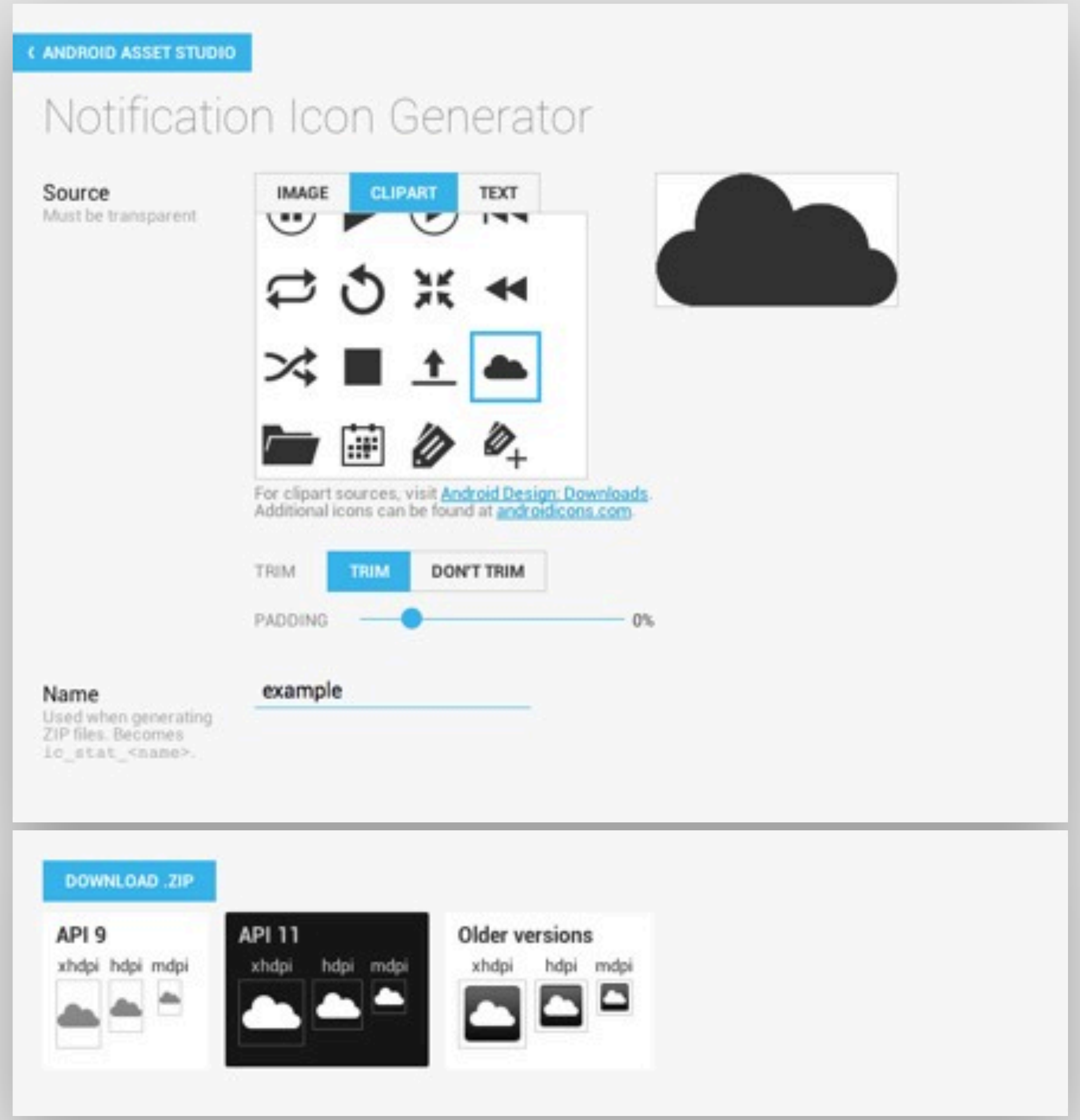
ANDROID-UI-UTILS.GOOGLECODE.COM

Action Bar Style Generator

ACTIONBARSTYLEGENERATOR.COM

Android Holo Colors

ANDROID-HOLO-COLORS.COM


< ANDROID ASSET STUDIO

Notification Icon Generator

Source
Must be transparent

IMAGE [✓ CLIPART] TEXT

For clipart sources, visit Android Design: Downloads
Additional icons can be found at androidicons.com

TRIM [✓ TRIM] DON'T TRIM

PADDING 0%

Name
Used when generating ZIP files. Becomes ic_stat_<name>.

example

DOWNLOAD .ZIP

Generating Holo assets

Android Asset Studio

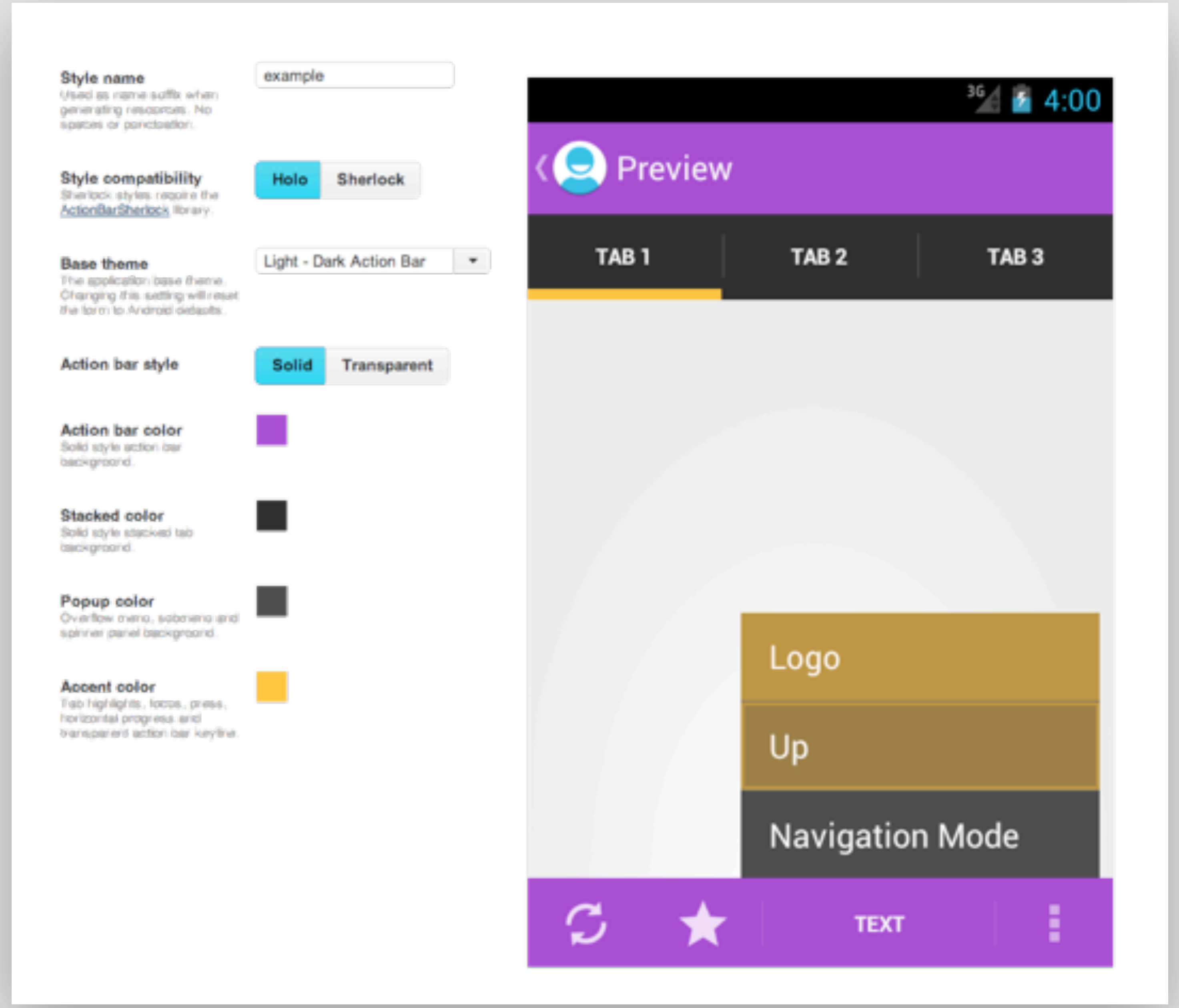
ANDROID-UI-UTILS.GOOGLECODE.COM

Action Bar Style Generator

ACTIONBARSTYLEGENERATOR.COM

Android Holo Colors

ANDROID-HOLO-COLORS.COM



Style name Used as name suffix when generating resources. No spaces or punctuation.	example
Style compatibility Sherlock styles require the ActionBarSherlock library.	[✓ Holo] Sherlock
Base theme The application base theme. Changing this setting will reset the form to Android defaults.	Light - Dark Action Bar
Action bar style	[✓ Solid] Transparent
Action bar color Solid style action bar background.	🟪
Stacked color Solid style stacked tab background.	⬛
Popup color Overflow menu, submenu and spinner panel background.	⬛
Accent color Tab highlights, focus, press, horizontal progress and transparent action bar keyline.	🟨

Generating Holo assets

Android Asset Studio

ANDROID-UI-UTILS.GOOGLECODE.COM

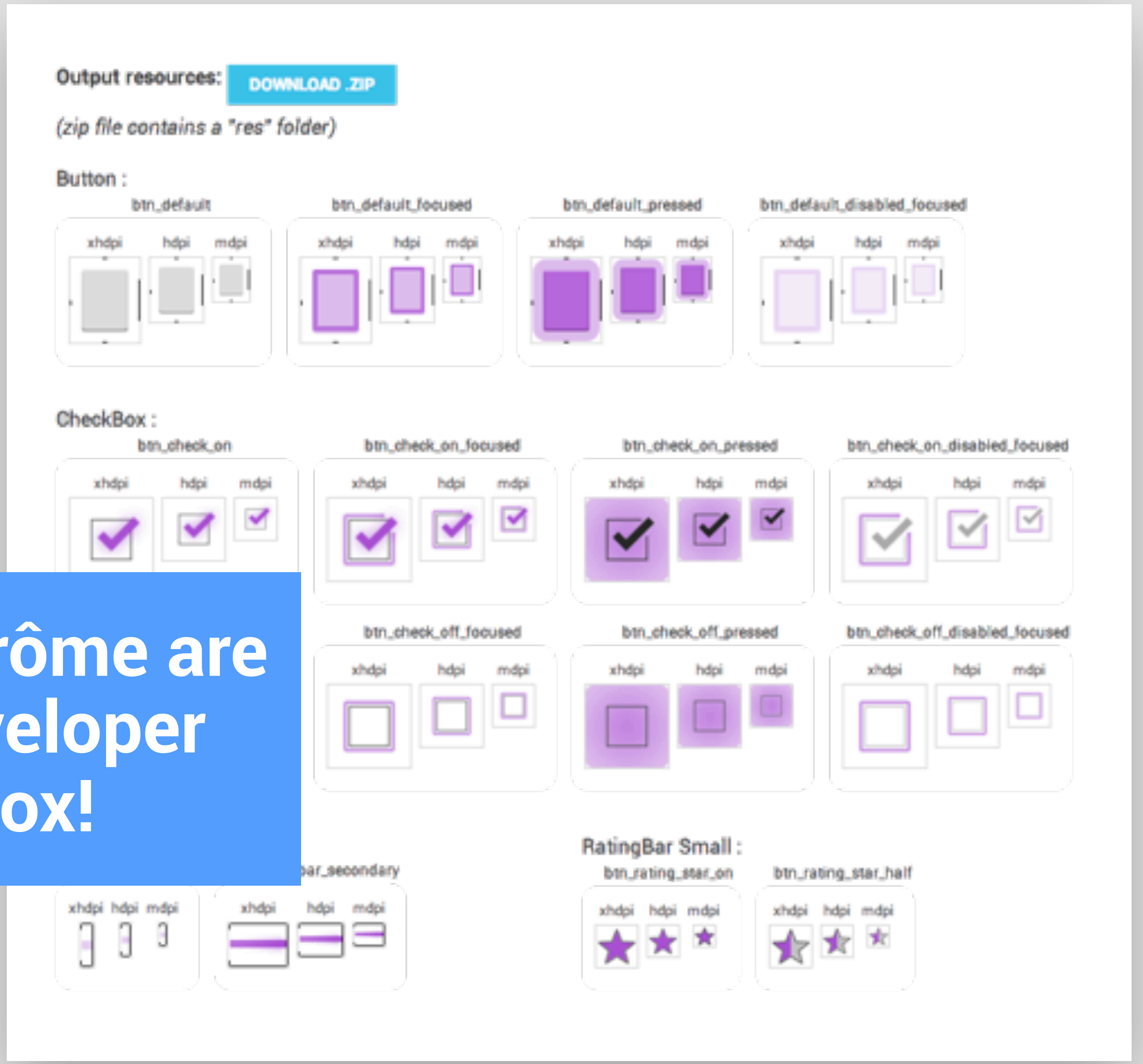
Action Bar Style Generator

ACTIONBARSTYLEGENERATOR.COM

Android Holo Colors

ANDROID-HOLO-COLORS.COM

Jeff and Jérôme are in the Developer Sandbox!



Thank You!

+Roman Nurik
+Nick Butcher

Google Developers