Android Development vol. 3

Tomáš Kypta

@TomasKypta

Agenda

- Dialogs
- Material Design
- UX Design Patterns
- Dependency Injection
- Useful Android Libraries

Database

Database

- SQLiteDatabase
- SQLiteOpenHelper
- can be used with/without content provider

Exercise

1. Check the code of the DB access.

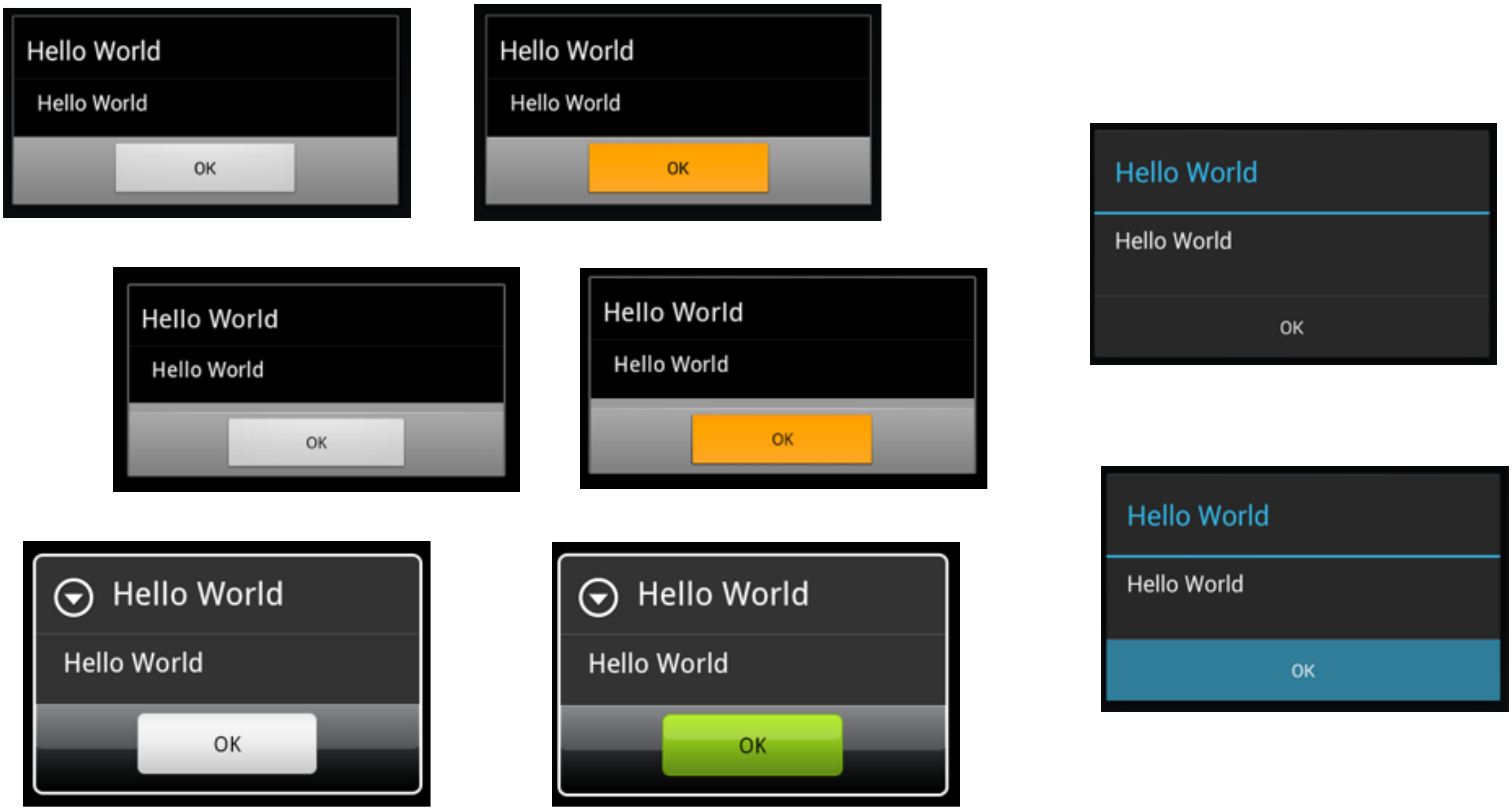
Dialogs

Dialogs

- handled via DialogFragment

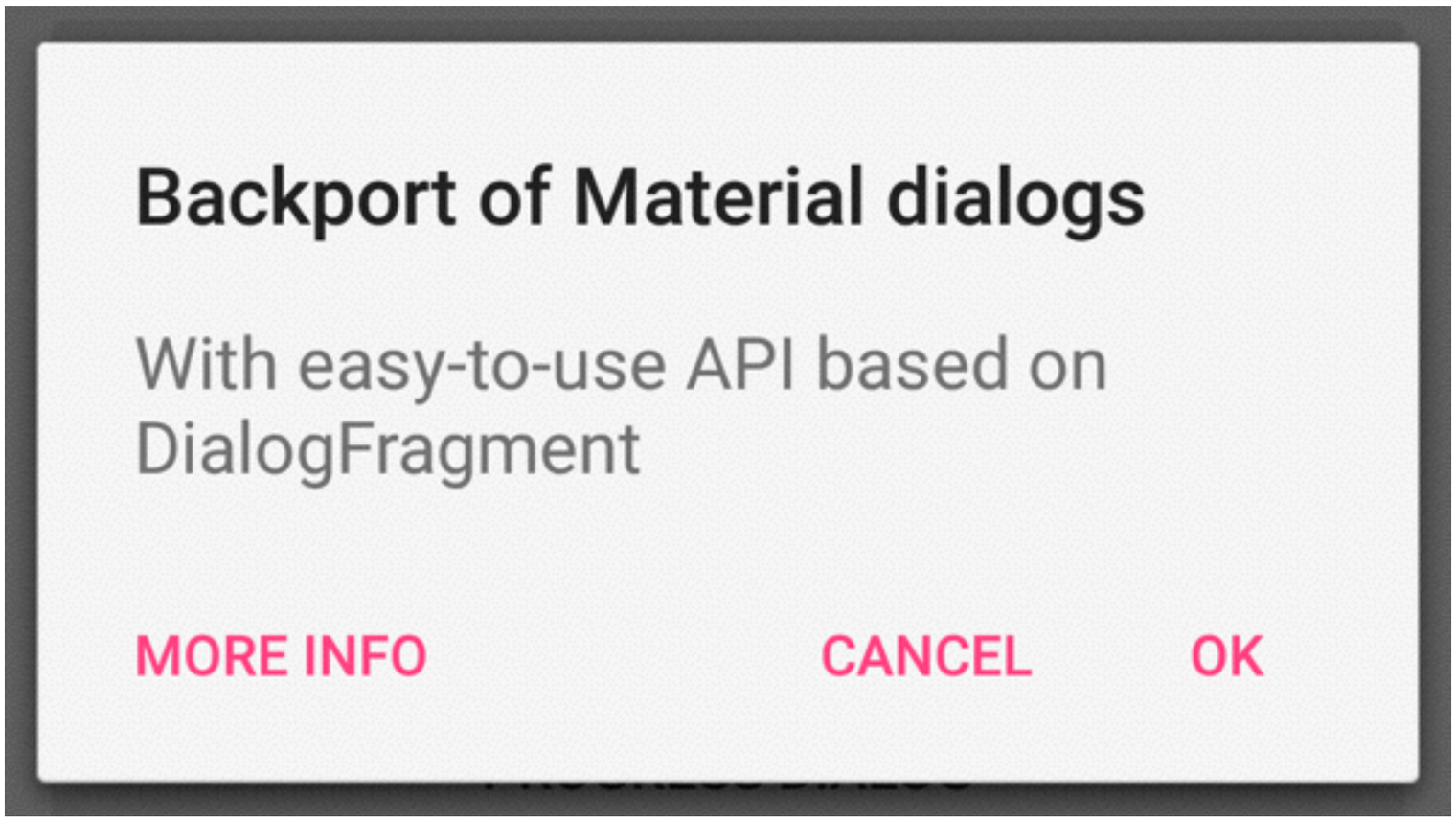
- styling problems

Dialogs


Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK

Dialogs

- android-styled-dialogs
  - https://github.com/avast/android-styled-dialogs



Dialogs

```
SimpleDialogFragment.createBuilder(c,
        getSupportFragmentManager())
.setTitle("Backport of material dialogs")
.setMessage("Lorem ipsum dolor sit amet…")
.setPositiveButtonText("OK")
.setNegativeButtonText("Cancel")
.setNeutralButtonText("Help")
.setRequestCode(REQUEST_SIMPLE_DIALOG)
.show();
```

Dialogs

```
SimpleDialogFragment.createBuilder(c,
        getSupportFragmentManager())
.setTitle(“Backport of material dialogs")
.setMessage(“Lorem ipsum dolor sit amet…”)
.setPositiveButtonText("OK")
.setNegativeButtonText("Cancel")
.setNeutralButtonText("Help")
.setRequestCode(REQUEST_SIMPLE_DIALOG)
.show();
```

Dialogs

```
SimpleDialogFragment.createBuilder(c,
        getSupportFragmentManager())
.setTitle(“Backport of material dialogs")
.setMessage(“Lorem ipsum dolor sit amet…”)
.setPositiveButtonText("OK")
.setNegativeButtonText("Cancel")
.setNeutralButtonText("Help")
.setRequestCode(REQUEST_SIMPLE_DIALOG)
.show();
```

Dialogs

```
SimpleDialogFragment.createBuilder(c,
        getSupportFragmentManager())
.setTitle(“Backport of material dialogs")
.setMessage(“Lorem ipsum dolor sit amet…”)
.setPositiveButtonText("OK")
.setNegativeButtonText("Cancel")
.setNeutralButtonText("Help")
.setRequestCode(REQUEST_SIMPLE_DIALOG)
.show();
```

Dialogs

```
SimpleDialogFragment.createBuilder(c,
        getSupportFragmentManager())
.setTitle(“Backport of material dialogs")
.setMessage(“Lorem ipsum dolor sit amet…”)
.setPositiveButtonText("OK")
.setNegativeButtonText("Cancel")
.setNeutralButtonText("Help")
.setRequestCode(REQUEST_SIMPLE_DIALOG)
.show();
```

Dialogs

- implement listener in activity/fragment to receive callback
  - ISimpleDialogListener
  - IPositiveButtonDialogListener
  - INegativeButtonDialogListener
  - INeutralButtonDialogListener

Dialogs

```
@Override
public void onPositiveButtonClicked(int requestCode) {
    if (requestCode == REQUEST_SIMPLE_DIALOG) {
        Toast.makeText(c,
            "Positive button clicked",
            Toast.LENGTH_SHORT)
            .show();
    }
}
```

Dialogs

```
@Override
public void onPositiveButtonClicked(int requestCode) {
    if (requestCode == REQUEST_SIMPLE_DIALOG) {
        Toast.makeText(c,
            "Positive button clicked",
            Toast.LENGTH_SHORT)
            .show();
    }
}
```

Dialogs

```
@Override
public void onPositiveButtonClicked(int requestCode) {
    if (requestCode == REQUEST_SIMPLE_DIALOG) {
        Toast.makeText(c,
            "Positive button clicked",
            Toast.LENGTH_SHORT)
            .show();
    }
}
```

Exercise

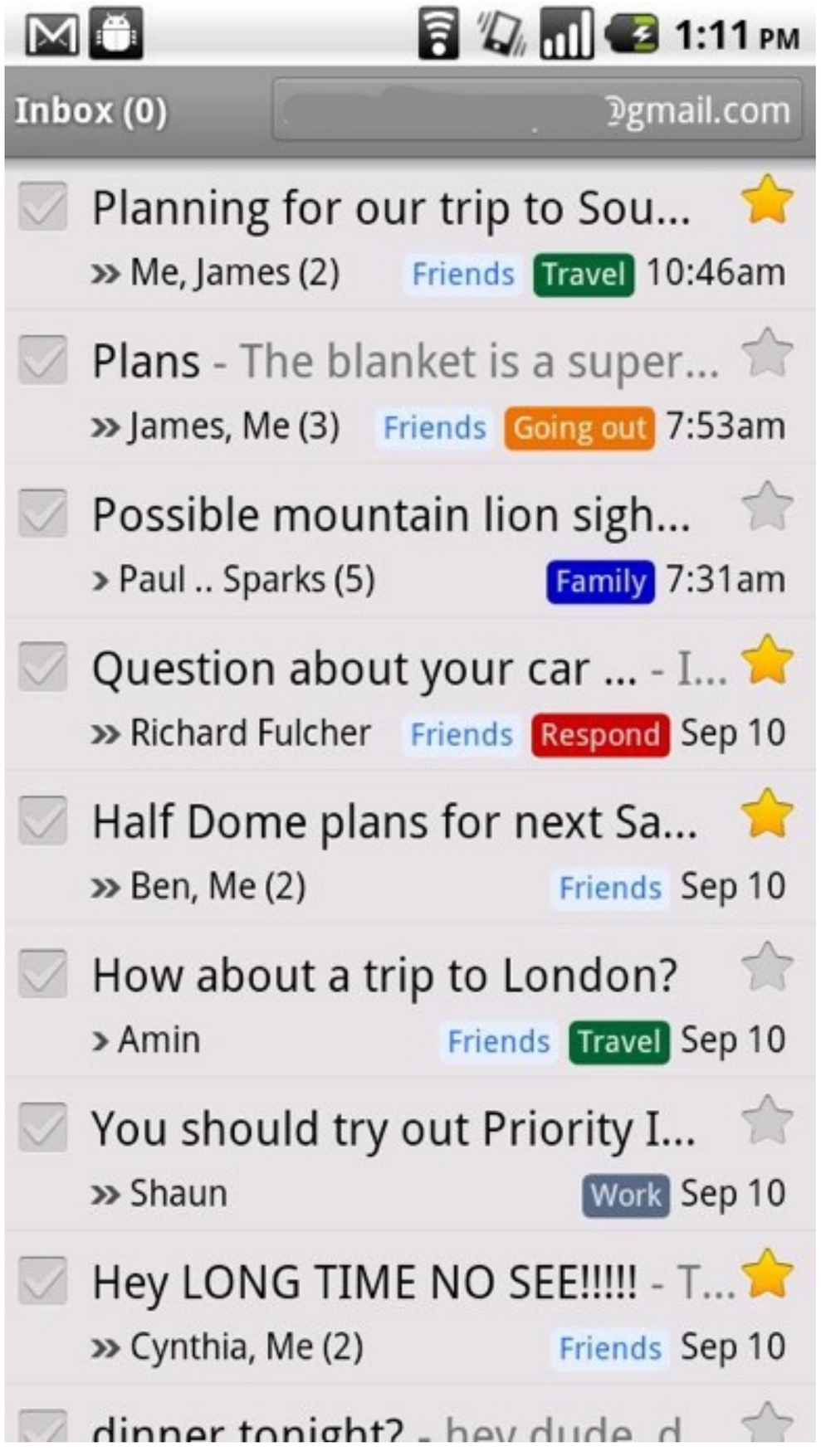
2. Display alert dialog when we click on an account item.

3. Check the code of add dialog.

Material Design

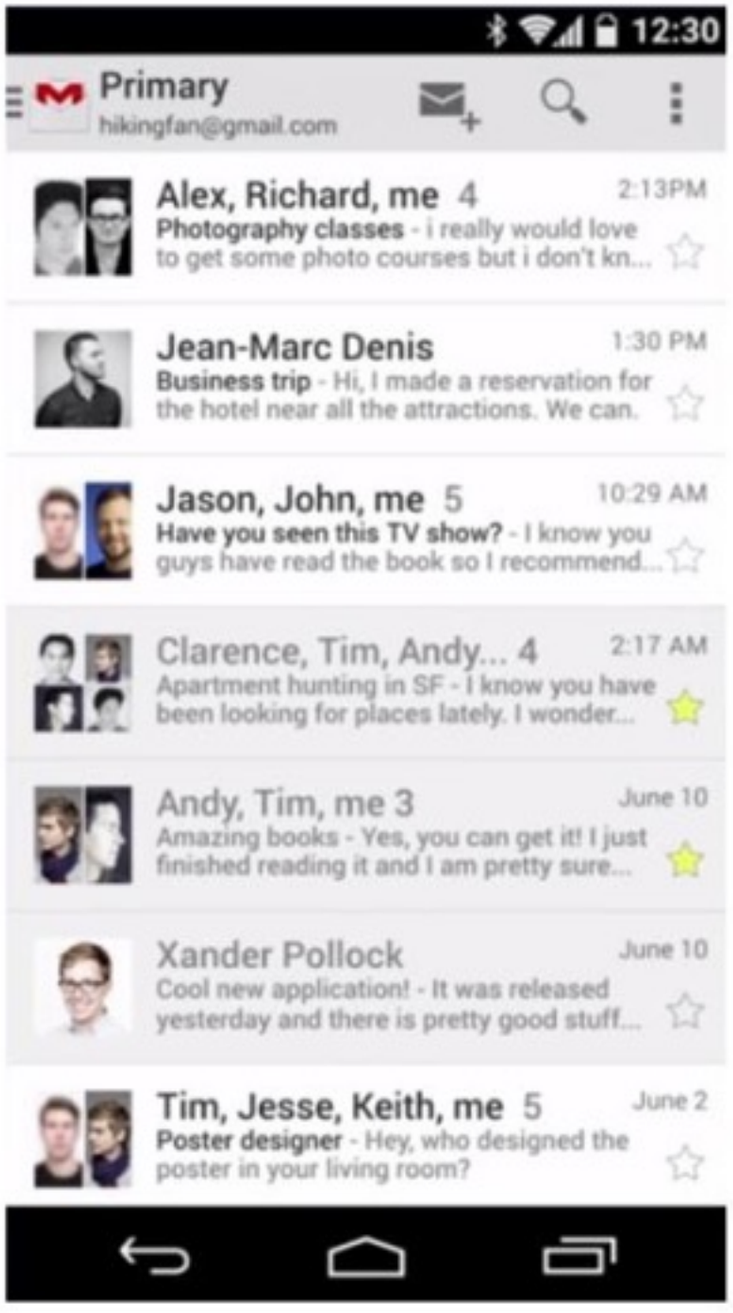
Android Design Evolution

long time ago …

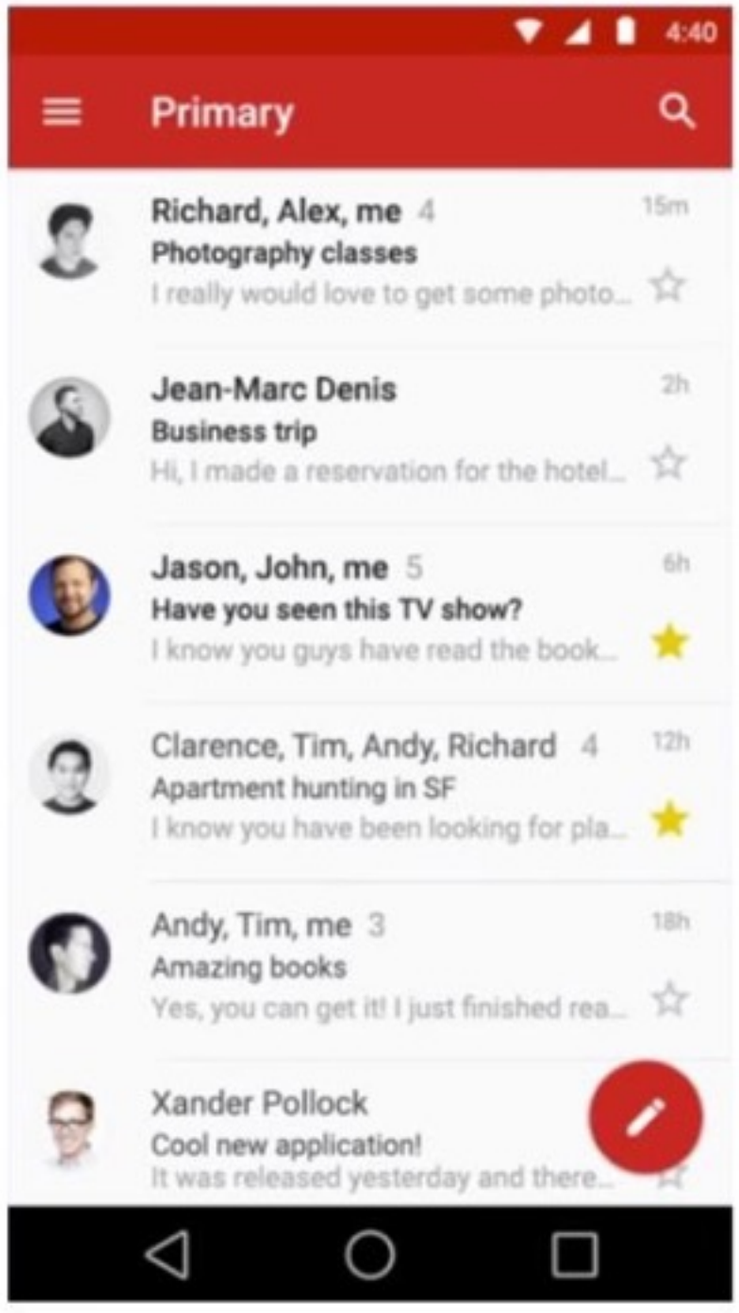


Android Design Evolution

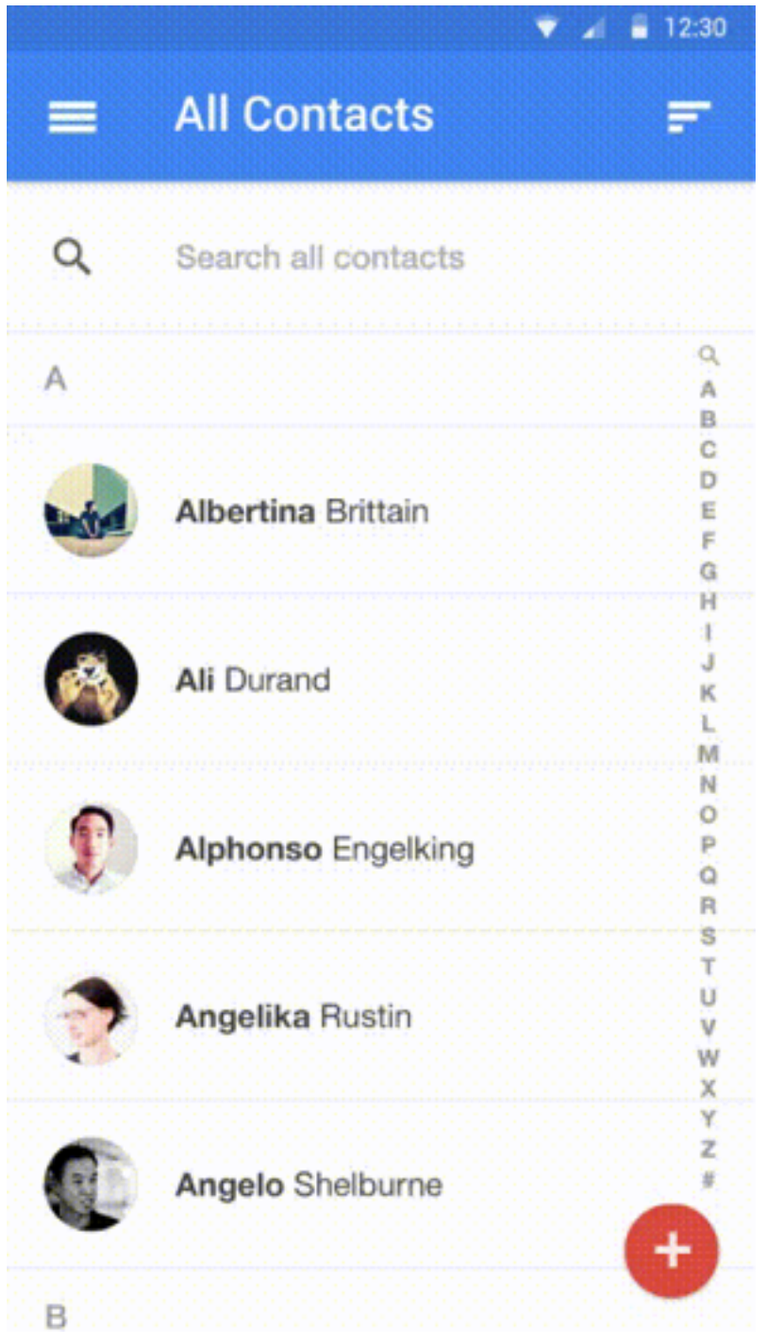
Holo Design

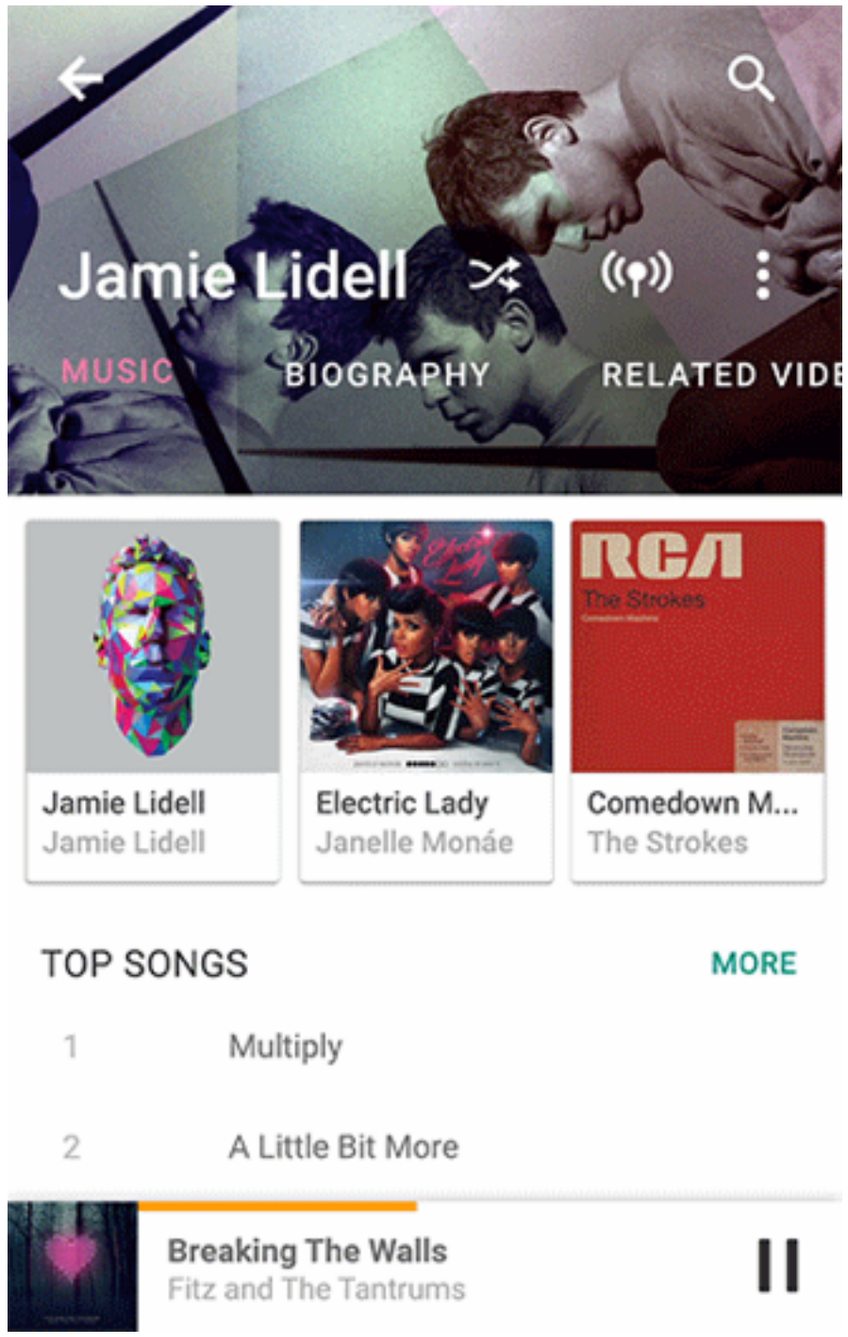


Material Design



Material Design





Material Design

- tangible surfaces and edges
  - shadows
  - elevation
- motion

Material Design

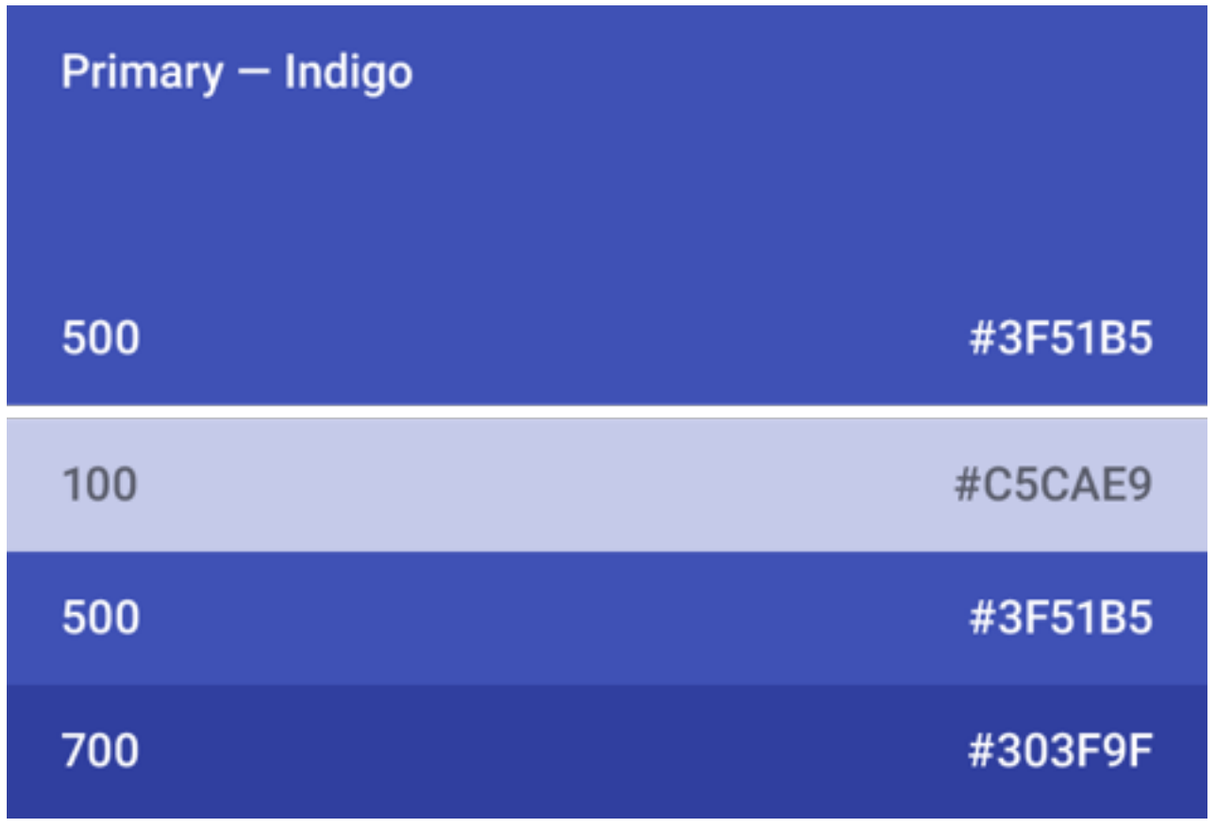
- themes since API level 21
  - `@android:style/Theme.Material` (dark version)
  - `@android:style/Theme.Material.Light` (light version)
  - `@android:style/Theme.Material.Light.DarkActionBar`
  - …

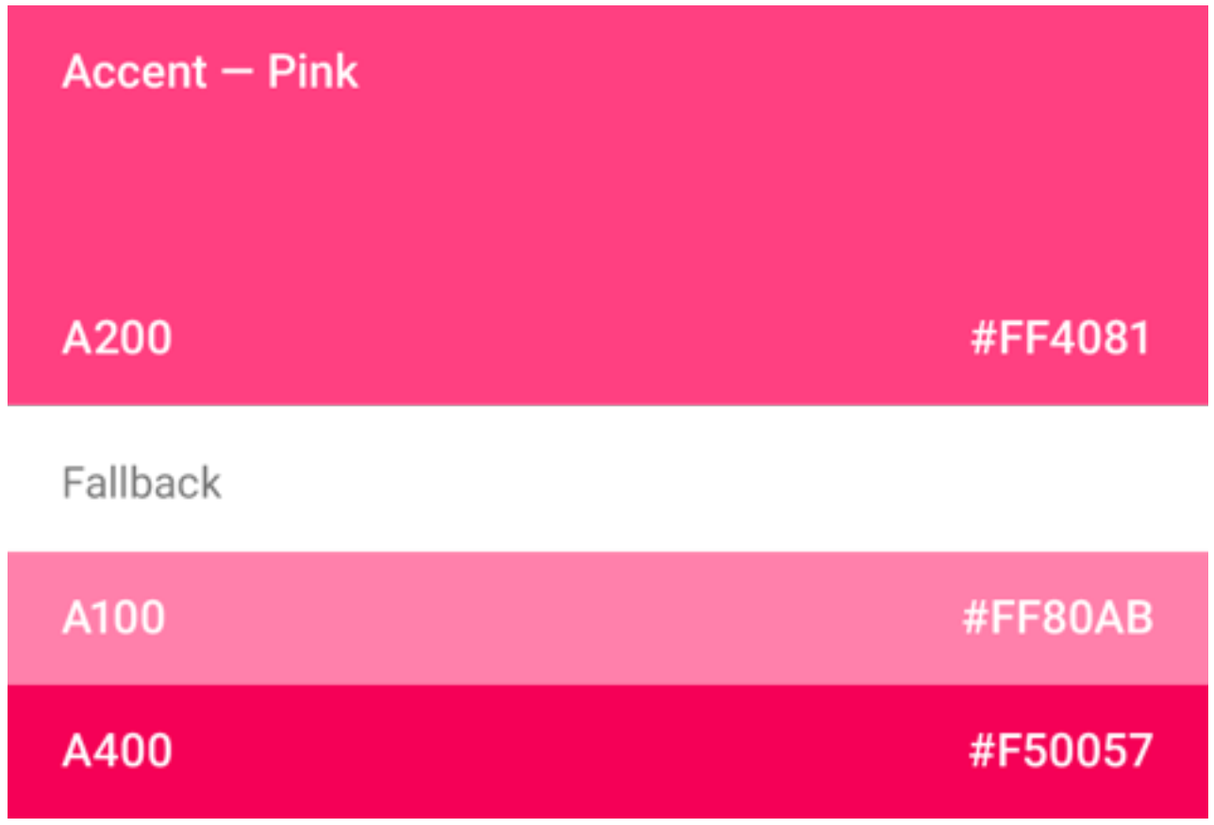
Material Design

- compatibility themes
  - `@style/Theme.AppCompat` (dark version)
  - `@style/Theme.AppCompat.Light` (light version)
  - `@style/Theme.AppCompat.Light.DarkActionBar`
  - …

The Color Palette

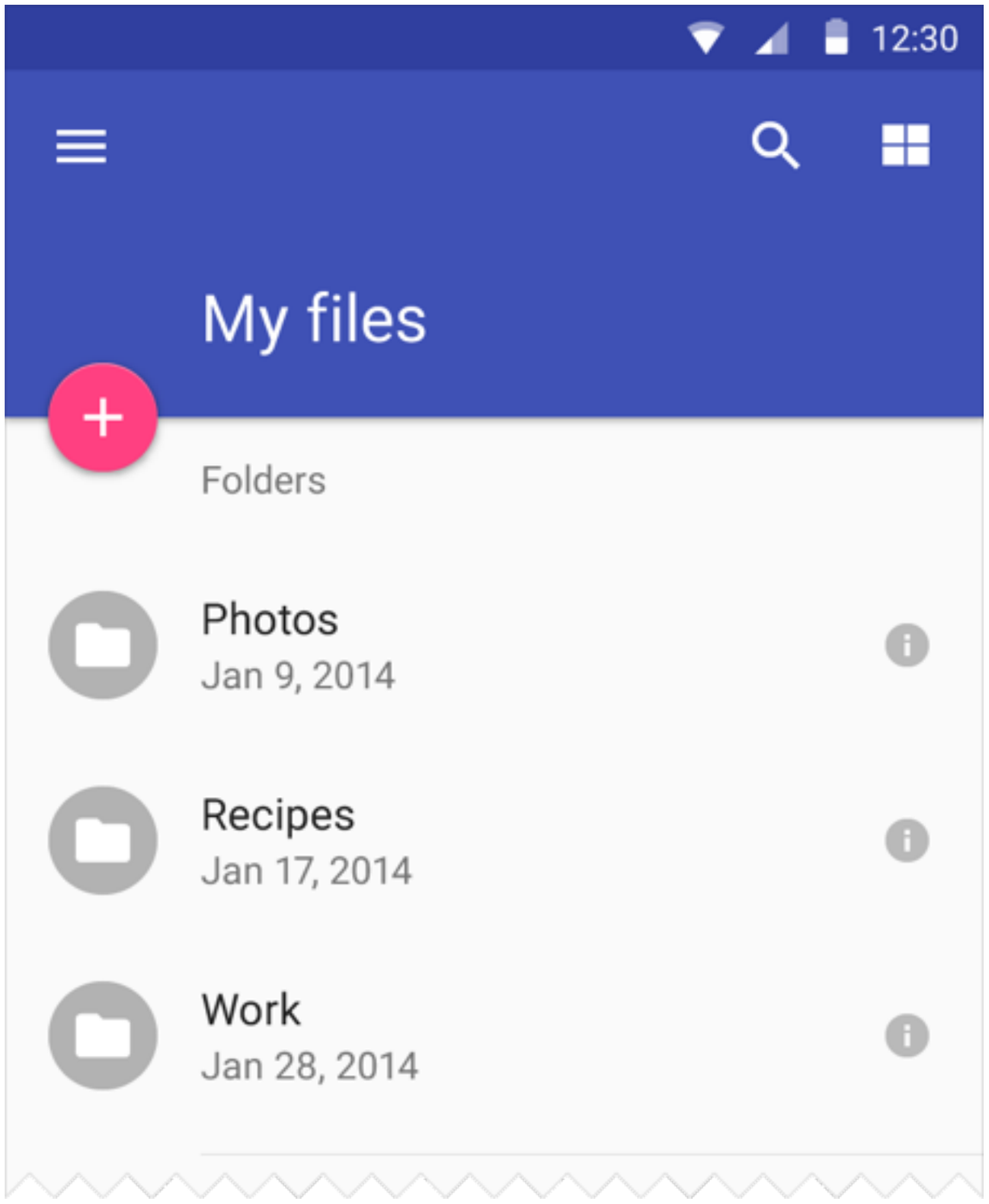
- three color hues from the primary palette
- one accent color from the secondary palette





The Color Palette

- http://www.google.com/design/spec/style/color.html



The Color Palette

```
<!-- inherit from the material theme -->
<style name="AppTheme" parent="“android:Theme.Material">

    <!-- Main theme colors -->
    <!-- your app branding color for the app bar -->
    <item name="“android:colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="“android:colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="android:colorAccent">@color/accent</item>
</style>
```

The Color Palette

```
<!-- inherit from the material theme -->
<style name="AppTheme" parent="“android:Theme.Material">

    <!-- Main theme colors -->
    <!-- your app branding color for the app bar -->
    <item name="“android:colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="“android:colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="android:colorAccent">@color/accent</item>
</style>
```

The Color Palette

```
<!-- inherit from the material theme -->
<style name="AppTheme" parent="“android:Theme.Material">

    <!-- Main theme colors -->
    <!-- your app branding color for the app bar -->
    <item name="“android:colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="“android:colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="android:colorAccent">@color/accent</item>
</style>
```

The Color Palette

```
<!-- inherit from the material theme -->
<style name="AppTheme" parent="“android:Theme.Material">

    <!-- Main theme colors -->
    <!-- your app branding color for the app bar -->
    <item name="“android:colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="“android:colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="android:colorAccent">@color/accent</item>
</style>
```

The Color Palette with AppCompat

```
<!-- AppCompat variant -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- your app branding color for the app bar -->
    <item name="colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="colorAccent">@color/accent</item>
</style>
```

The Color Palette with AppCompat

```
<!-- AppCompat variant -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- your app branding color for the app bar -->
    <item name="colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="colorAccent">@color/accent</item>
</style>
```

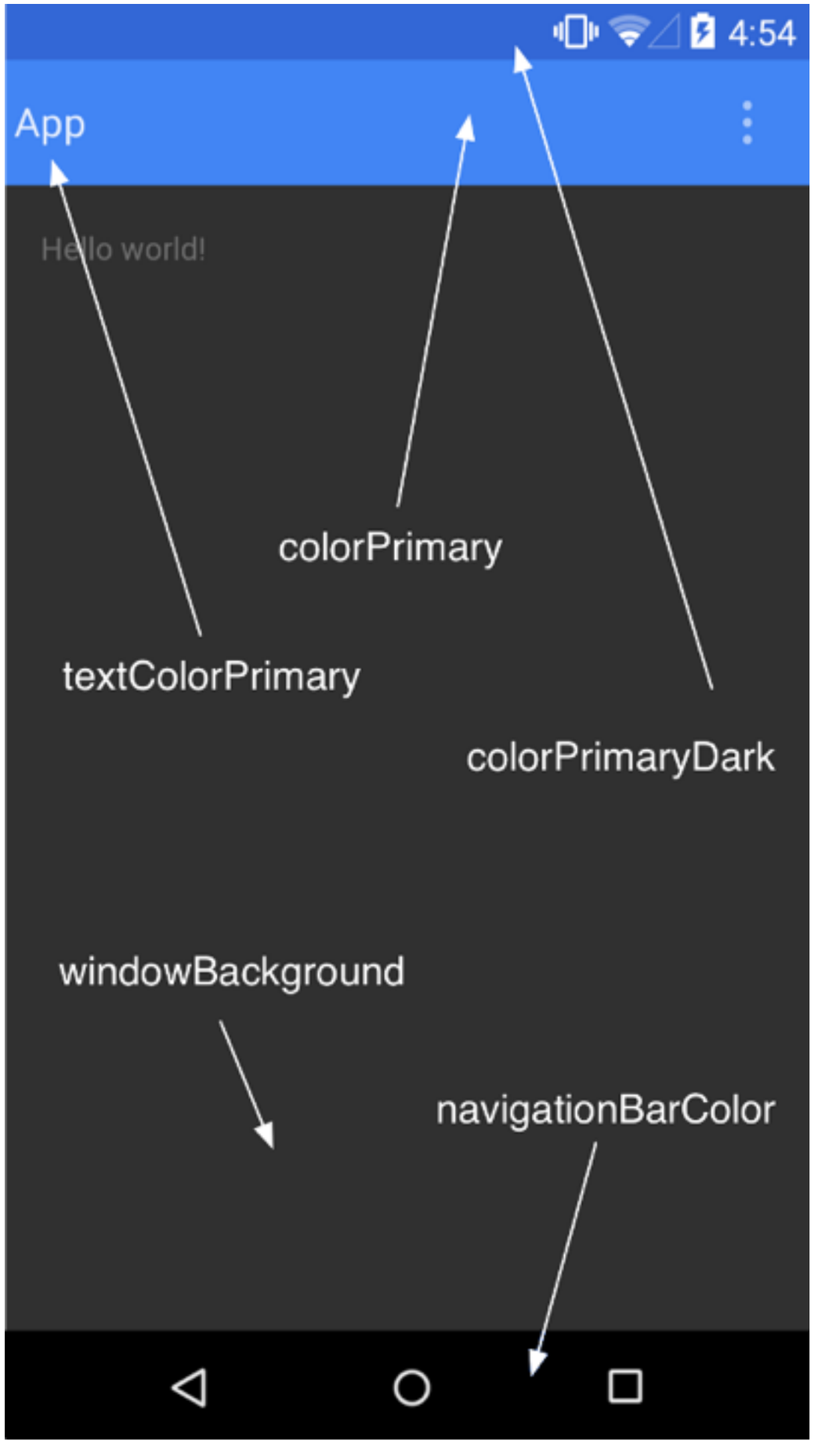
The Color Palette with AppCompat

```
<!-- AppCompat variant -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- your app branding color for the app bar -->
    <item name="colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="colorAccent">@color/accent</item>
</style>
```

The Color Palette with AppCompat

```
<!-- AppCompat variant -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- your app branding color for the app bar -->
    <item name="colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="colorAccent">@color/accent</item>
</style>
```

The Color Palette



Toolbar

Toolbar

- generalization of action bar
- main Android navigation element

Toolbar

- `AppCompatActivity`

```
compile 'com.android.support:appcompat-v7:22.0.0'
```

- inherit styles from **`Theme.AppCompat`**

- for inflating views for action bar use **`getSupportActionBar().getThemedContext()`**

Toolbar

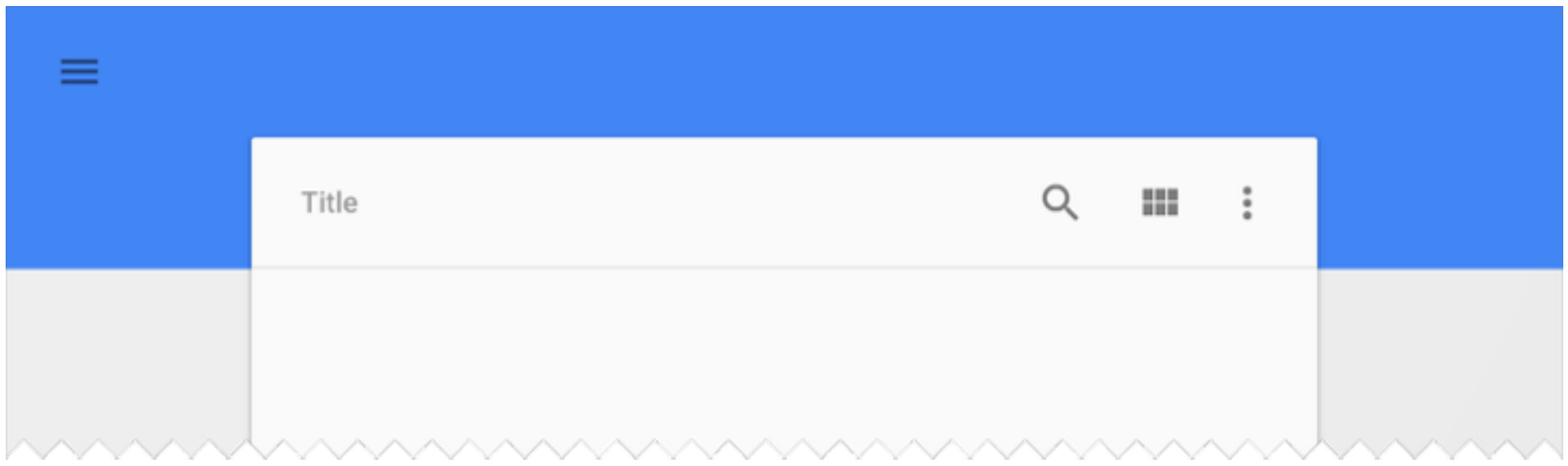
```
<android.support.v7.widget.Toolbar
android:id="@+id/my_toolbar"
android:layout_height="wrap_content"
android:layout_width="match_parent"
android:minHeight="?attr/actionBarSize"
android:background="?attr/colorPrimary" />
```

Toolbar

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Toolbar toolbar = (Toolbar)
        findViewById(R.id.my_toolbar);
    setSupportActionBar(toolbar);
}
```

Toolbar

- standalone mode
- ~~`setSupportActionBar(toolbar);`~~



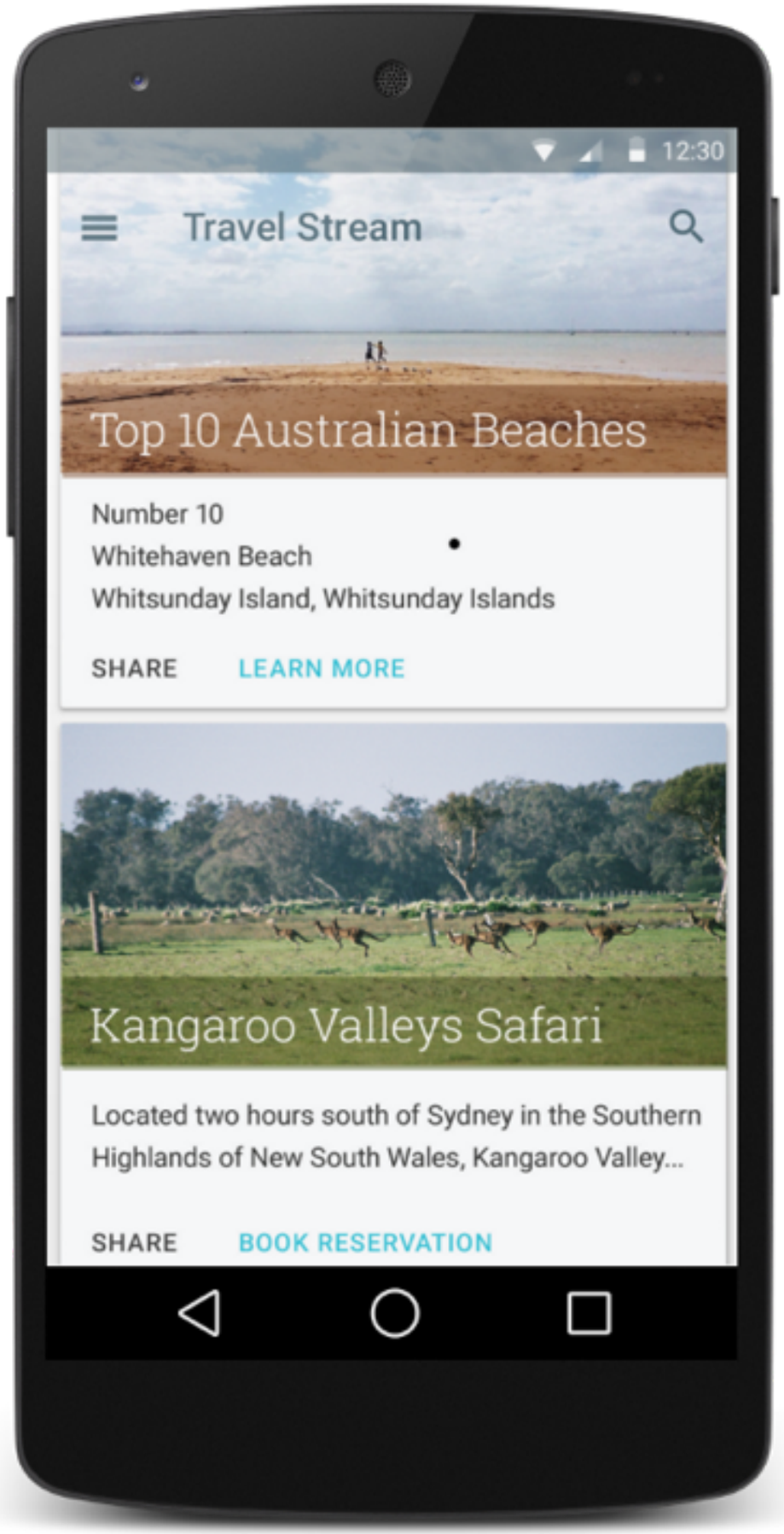
CardView

CardView

- showing information inside cards with consistent look

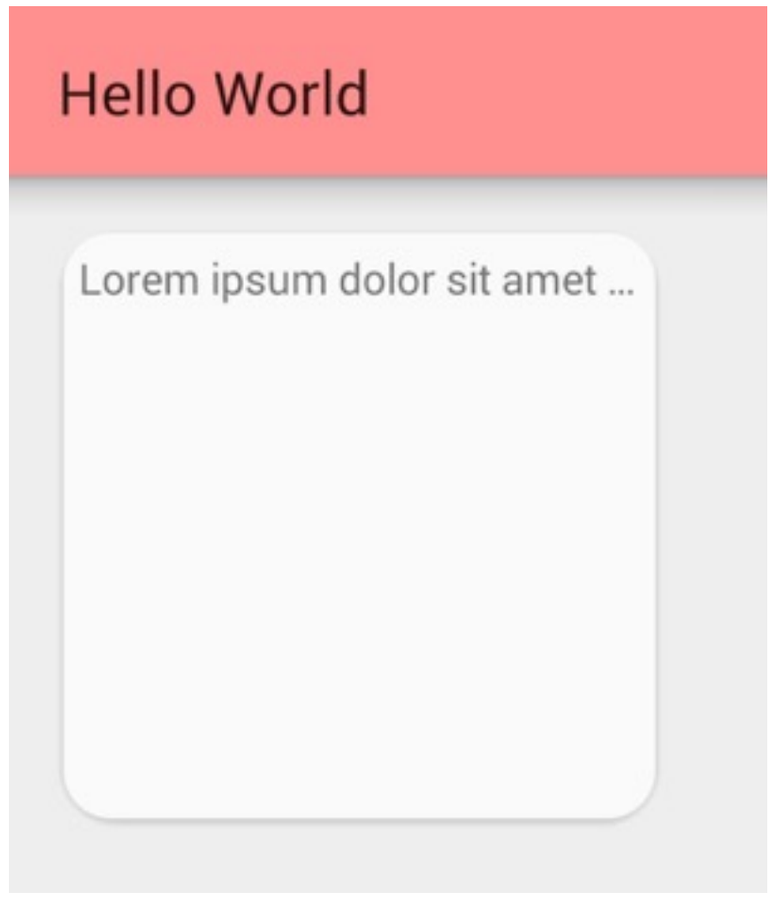
```
compile 'com.android.support:cardview-v7:22.0.0'
```

CardView


12:30
Travel Stream
Top 10 Australian Beaches
Number 10
Whitehaven Beach
Whitsunday Island, Whitsunday Islands
SHARE
LEARN MORE
Kangaroo Valleys Safari
Located two hours south of Sydney in the Southern Highlands of New South Wales, Kangaroo Valley...
SHARE
BOOK RESERVATION

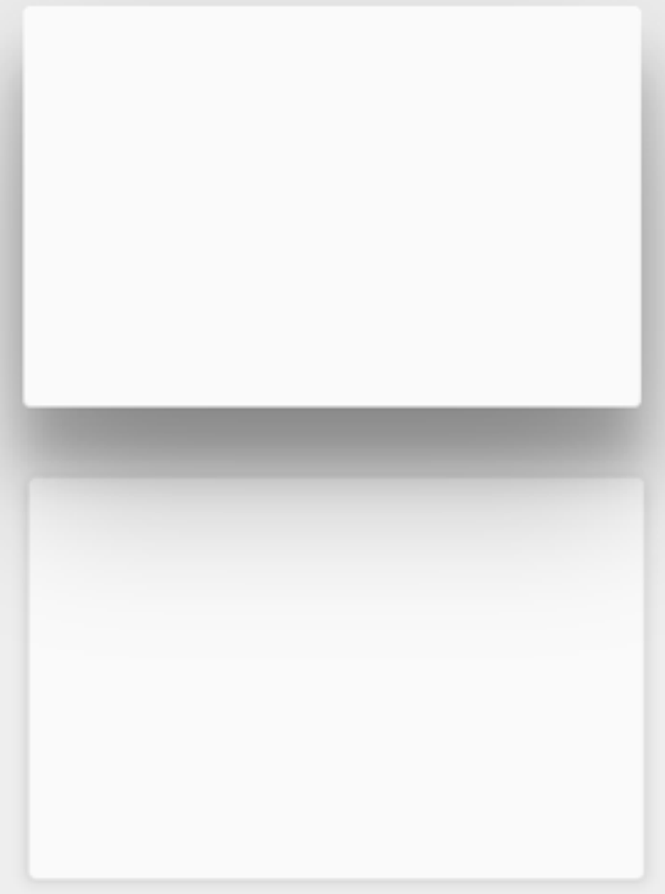
CardView

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_gravity="center"
    android:layout_width="200dp"
    android:layout_height="200dp"
    card_view:cardCornerRadius="16dp">

    …

</android.support.v7.widget.CardView>
```



Elevation

Elevation



Elevation

```
android:elevation=“4dp”

view.setElevation(elevation);
```

Transition

Transition Animation

- since API level 21
- in styles.xml

```
<item name=“android:windowEnterTransition">
        @android:transition/explode</item>
<item name=“android:windowExitTransition">
        @android:transition/explode</item>
```

Transition

```
public class MyActivity1 extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // enable transitions before content is added
        getWindow().requestFeature(
            Window.FEATURE_CONTENT_TRANSITIONS);

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my_1);
        …
    }
}
```

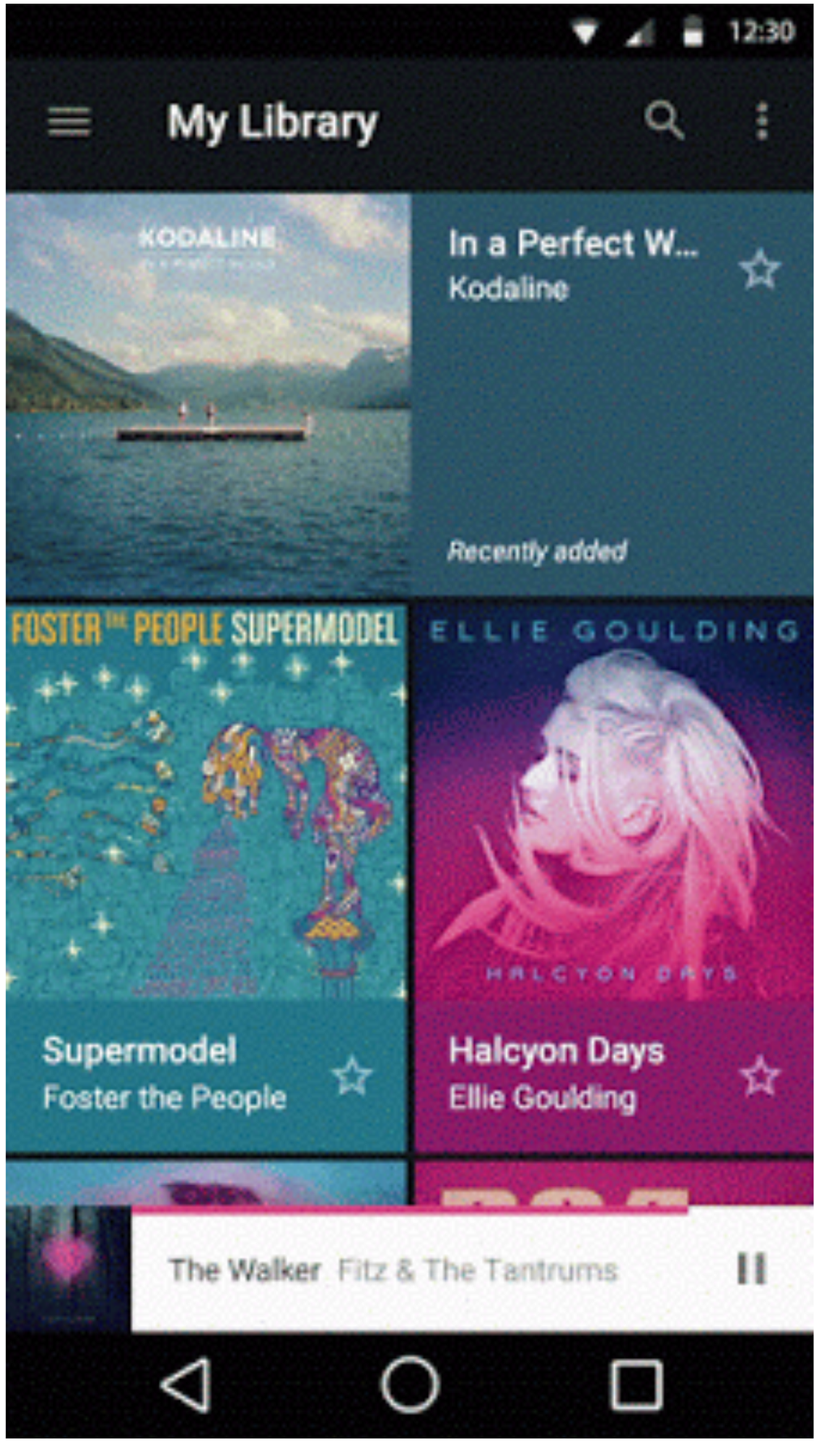
Transition

```
btn.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
            getWindow().setExitTransition(new Explode());
            Intent intent = new Intent(MyActivity1.this,
                MyActivity2.class);
            startActivity(intent,
                ActivityOptions
                    .makeSceneTransitionAnimation(MyActivity1.this)
                    .toBundle());
      }
});
```

Shared Element Transition

- enable window content transitions
- transition for shared element
  - define shared element with `android:transitionName`
    - in both layouts
  - `ActivityOptions.makeSceneTransitionAnimation()`

Shared Element Transition



Ripple

Ripple

- RippleDrawable

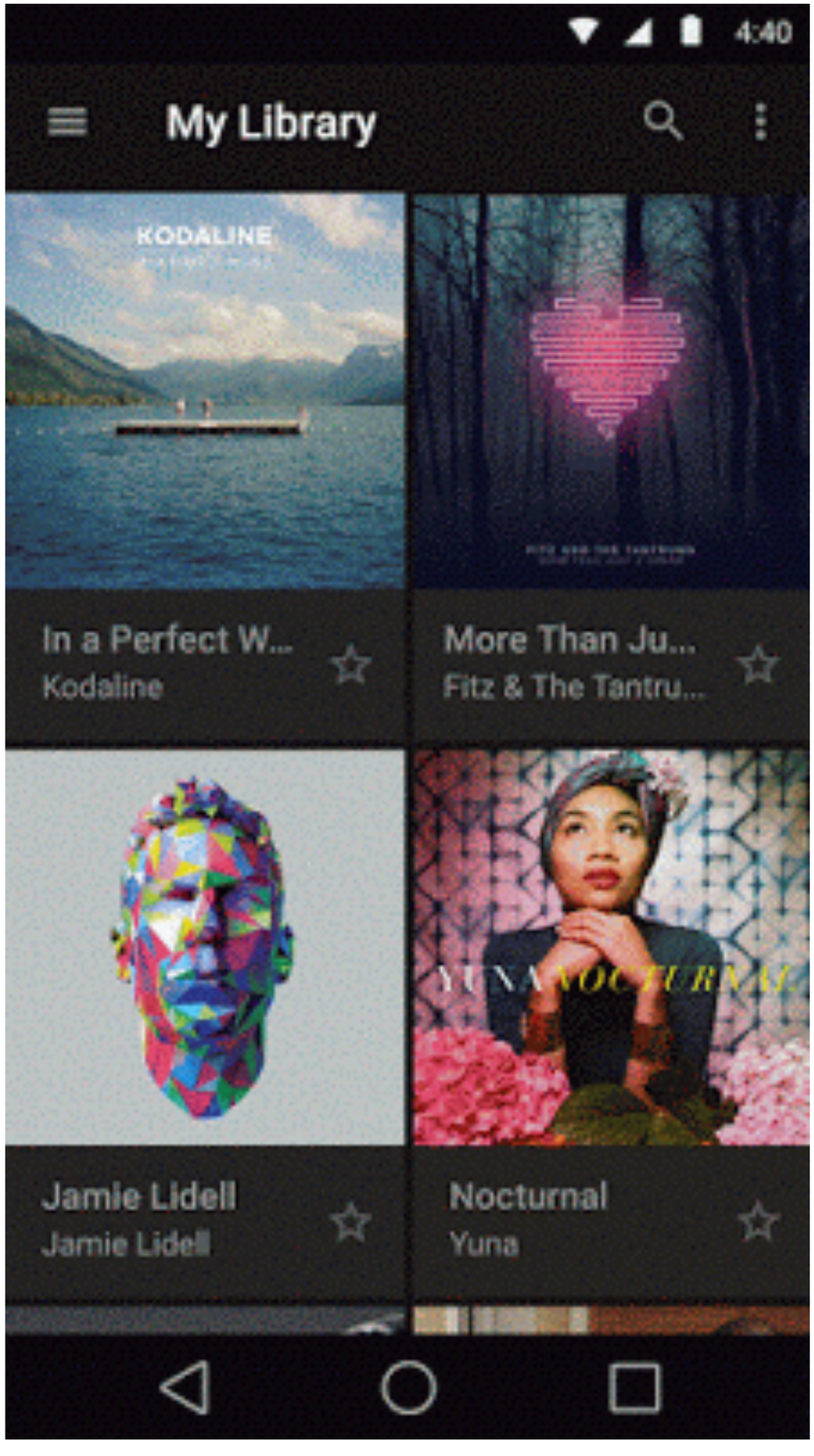
- set as view background

Ripple

Ripple

```
<ripple android:color="#ffff0000"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@android:color/white" />
</ripple>
```

Dynamic color



State List Animator

State List Animator

- materials raise up to meet your finger

- `android:stateListAnimator="@anim/raise"`

State List Animator

… and ripple

State List Animator

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_enabled="true" android:state_pressed="true">
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="10dp"
            android:valueType="floatType" />
    </item>
    <item>
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="0dp"
            android:valueType="floatType" />
    </item>
</selector>
```

State List Animator

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_enabled="true" android:state_pressed="true">
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="10dp"
            android:valueType="floatType" />
    </item>
    <item>
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="0dp"
            android:valueType="floatType" />
    </item>
</selector>
```

State List Animator

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_enabled="true" android:state_pressed="true">
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="10dp"
            android:valueType="floatType" />
    </item>
    <item>
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="0dp"
            android:valueType="floatType" />
    </item>
</selector>
```

Floating Action Button

Exercise

4. Check the code of FAB in project.

UX Design Patterns

Pull-to-Refresh

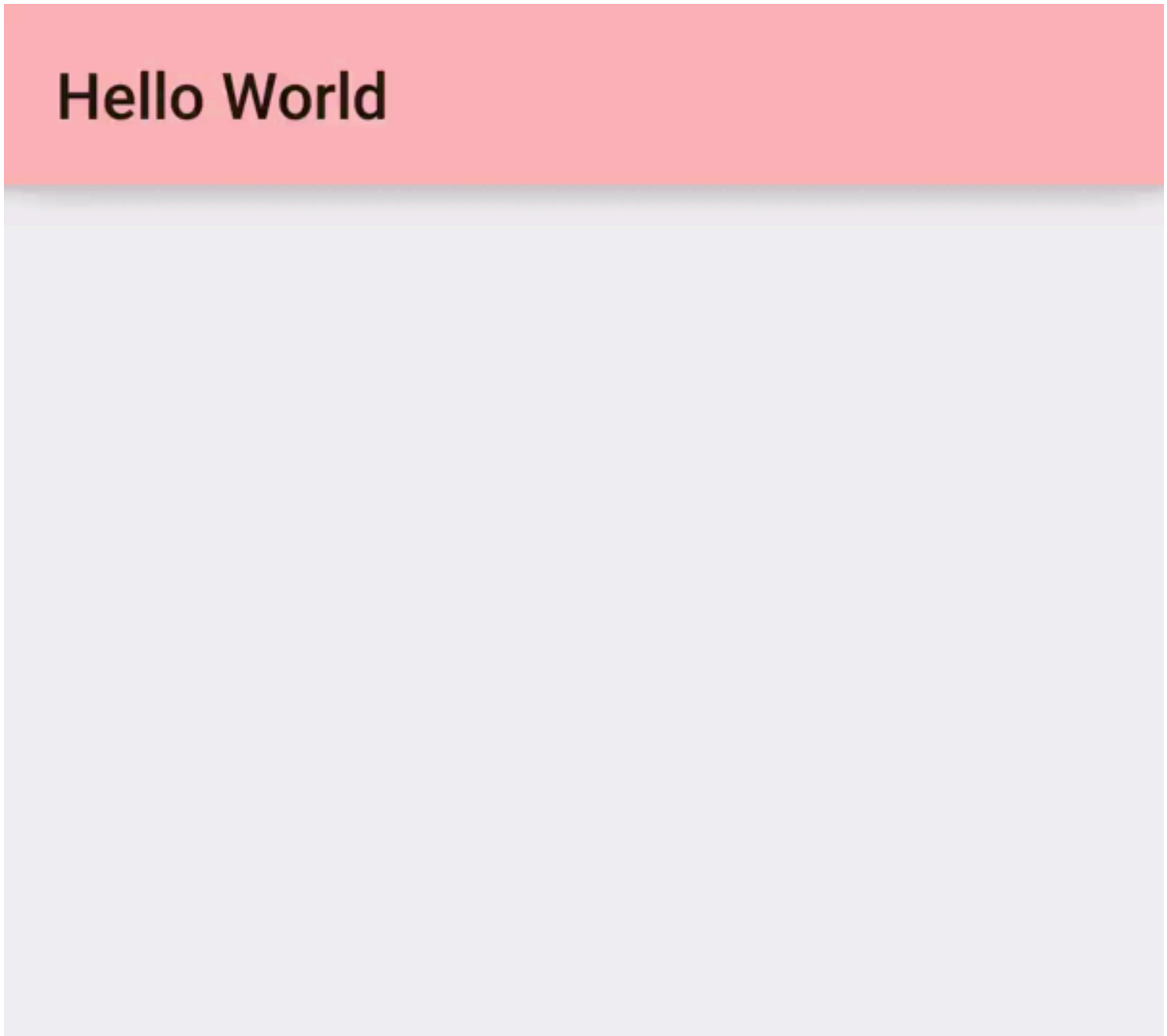
Pull-to-Refresh

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.SwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/swipeContainer"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView android:id="@+id/lvItems"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop=“true" />

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

Pull-to-Refresh

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.SwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/swipeContainer"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView android:id="@+id/lvItems"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop=“true" />

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

Pull-to-Refresh



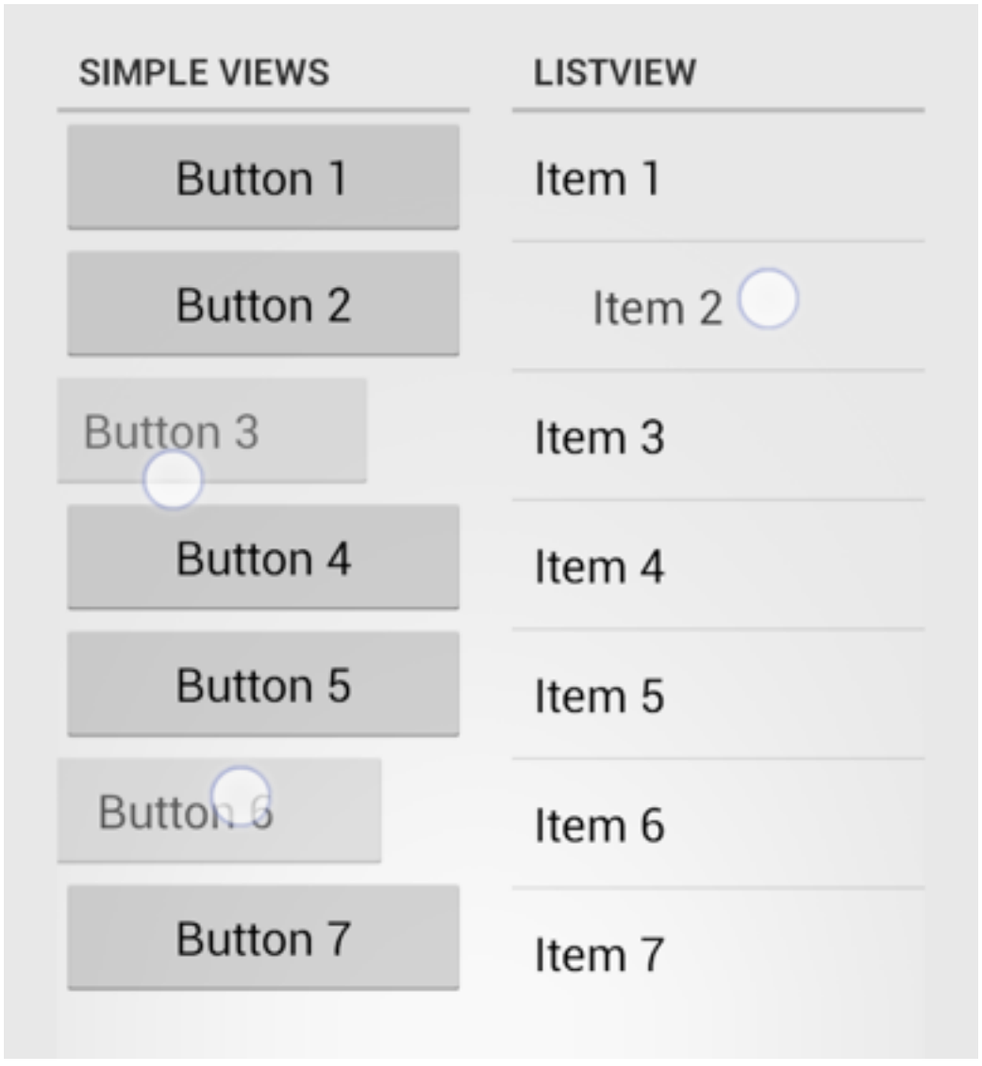
Exercise

5. Add Pull-to-Refresh for account list.

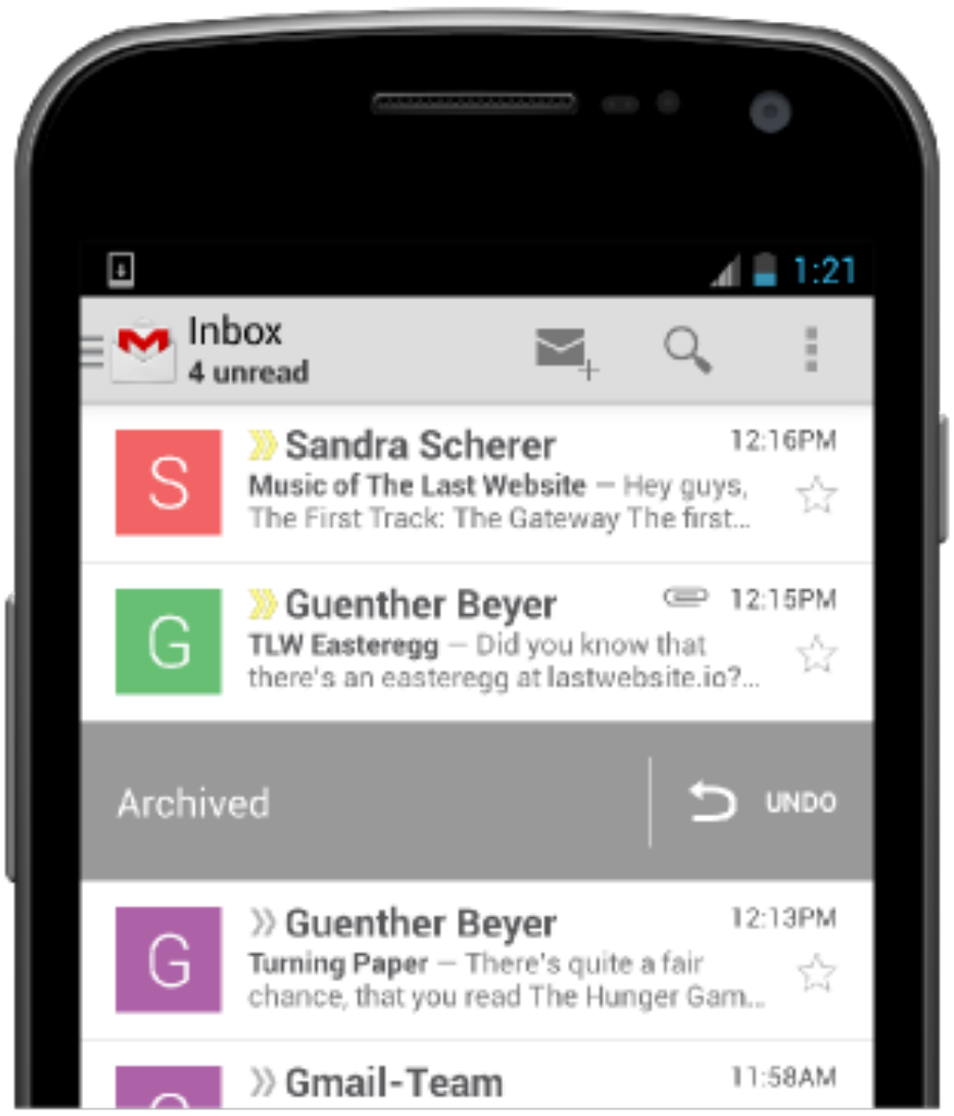
Swipe-to-dismiss

Swipe-to-dismiss

- dismiss list item by swiping left or right



Swipe-to-dismiss with Undo



Snackbar

Snackbar

- Toast-like message

- they provide action

Snackbar

```
Snackbar snackbar = Snackbar.make(
    coordinatorLayout,
    “This is a Snackbar",
    Snackbar.LENGTH_LONG);

snackbar.show();
```

Snackbar

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout,
                "Message was deleted”,
                Snackbar.LENGTH_LONG)
        .setAction("UNDO",
                new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        // do something
                    }
                });

snackbar.show();
```

Exercise

6. Show a snackbar when we display account detail.

Software Design Patterns

Dependency Injection

Service Locator

Service Locator

- via Context

```
Object context.getSystemService(String)
```

Service Locator

```
public class MyApplication extends Application {

    private MyManager mMyManager;

    @Override
    public Object getSystemService(String name) {
        if (MyManager.class.getName().equals(name)) {
            if (mMyManager == null) {
                mMyManager = new MyManager();
            }
            return mMyManager;
        }
        return super.getSystemService(name);
    }
}
```

Service Locator

```
public class MyApplication extends Application {

    private MyManager mMyManager;

    @Override
    public Object getSystemService(String name) {
        if (MyManager.class.getName().equals(name)) {
            if (mMyManager == null) {
                mMyManager = new MyManager();
            }
            return mMyManager;
        }
        return super.getSystemService(name);
    }
}
```

Service Locator

```
public class MyApplication extends Application {

    private MyManager mMyManager;

    @Override
    public Object getSystemService(String name) {
        if (MyManager.class.getName().equals(name)) {
            if (mMyManager == null) {
                mMyManager = new MyManager();
            }
            return mMyManager;
        }
        return super.getSystemService(name);
    }
}
```

Service Locator

```
MyManager myManager = (MyManager) context
    .getApplicationContext()
    .getSystemService(MyManager.class.getName());
```

Service Locator

- always use application context
- can’t be used in libraries
    - you usually don’t control the application object

Exercise

7. Use service locator to access API.

Dagger 2

Dagger 2

- by Google
- evolution of Dagger 1 (by Square)
- no reflection
- generated code in compile time
- constructor and field injection

Dagger 2

- Constructor injection

```
private ProviderC mProviderC;
private ProviderD mProviderD;

@Inject
public ProviderA(ProviderC providerC, ProviderD
providerD) {
    mProviderC = providerC;
    mProviderD = providerD;
}
```

Dagger 2

- Field injection

```
public class MainActivity extends AppCompatActivity {
    @Inject ProviderA mProviderA;
    @Inject ProviderB mProviderB;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ((MyApp) getApplication()).getAppComponent()
            .injectMainActivity(this);
    }
}
```

Dagger 2

- prefer constructor injection wherever possible
- you can test the unit in isolation
- providing mocks is a piece of cake

Dagger 2

- Some catches when using field injection
  - Construct the whole component
    - Implicit provisions doesn’t support overrides!!

```
public class ProviderA {
    …
    @Inject
    public ProviderA(ProviderC providerC) {
     mProviderC = providerC;
    }
}
```

Dagger 2

- Some catches when using field injection
  - Make explicit provisions

```
@Module
public class AppModule {

    @Provides @Singleton
    ProviderB provideProvider2(ProviderC providerC) {
        return new ProviderB(providerC);
    }

    @Provides
    ProviderD provideProvider4() {
        return new ProviderD();
    }
}
```

- Beware marking constructors with **`@Inject`** when providing explicitly
  - may create unwanted double provision

Exercise

8. Use Dagger 2 to access server API.

View Holder

View Holder

```
static class ViewHolder {
  TextView txtName;
  TextView txtDescription;

  public ViewHolder(View view) {
    txtName = (TextView) view.findViewById(R.id.txt_name);
    txtDesc = (TextView) view.findViewById(R.id.txt_desc);
  }
}
```

```
view.setTag(holder);
```

```
ViewHolder holder = (ViewHolder) view.getTag();
```

View Holder

```
static class ViewHolder {
  TextView txtName;
  TextView txtDescription;

  public ViewHolder(View view) {
    txtName = (TextView) view.findViewById(R.id.txt_name);
    txtDesc = (TextView) view.findViewById(R.id.txt_desc);
  }
}


view.setTag(holder);


ViewHolder holder = (ViewHolder) view.getTag();
```

View Holder

```
static class ViewHolder {
  TextView txtName;
  TextView txtDescription;

  public ViewHolder(View view) {
    txtName = (TextView) view.findViewById(R.id.txt_name);
    txtDesc = (TextView) view.findViewById(R.id.txt_desc);
  }
}


view.setTag(holder);


ViewHolder holder = (ViewHolder) view.getTag();
```

Exercise

9. Check code of RecyclerView.Adapter. And finish RecyclerView.Adapter for repositories in RepoFragment.

Useful Android Libraries

Event Bus

Event Bus

- no direct support
- library or custom implementation
- alternative to local broadcasts

Event Bus

- Otto (http://square.github.io/otto)

```
Bus bus = new Bus();
bus.register(this);

bus.unregister(this);

@Subscribe
public void wasLoggedOut(LogoutEvent event) {
    // do some logout action
}
```

Event Bus

```
bus.post(new LogoutEvent(LogoutEvent.LogoutType.MANUAL));
```

```
@Produce
public LogoutEvent produceLogoutEvent() {
    return new LogoutEvent(LogoutEvent.LogoutType.MANUAL);
}
```

Exercise

10.Notify AccountFragment about DB changes.

Image Loaders

Image Loaders

- For loading images from and URL into ImageView

Image Loaders

- Picasso
  - http://square.github.io/picasso/

```
Picasso.with(context)
    .load(URL)
    .into(imageView);
```

Image Loaders

- Glide
  - https://github.com/bumptech/glide
- Android-Universal-Image-Loader
  - https://github.com/nostra13/Android-Universal-Image-Loader

Exercise

11.Load avatar image of an account.

Other Useful Libraries

- ButterKnife
- RxJava / RxAndroid
- Mortar

Exercise

12.Use ButterKnife in RepoFragment.

Useful Libraries
- for unit testing

UI TESTING FOR ANDROID
espresso

Thank You