Android Ninja Academy

Sensei - Joel Sosa

Disclaimer: *GDG Puerto Rico is an independent group; our activities and the opinions expressed should in no way be linked to Google, the corporation.*

Agenda

P1

- Where is Android in the world?
- Platform Dissected
- Android Studio (2.0)
- First Android App
  - Structure (layout, resources)
  - Types of Resources
  - Manifest
- Concepts
  - Activities
  - Elements in Layouts
  - Intents and information sharing

P2

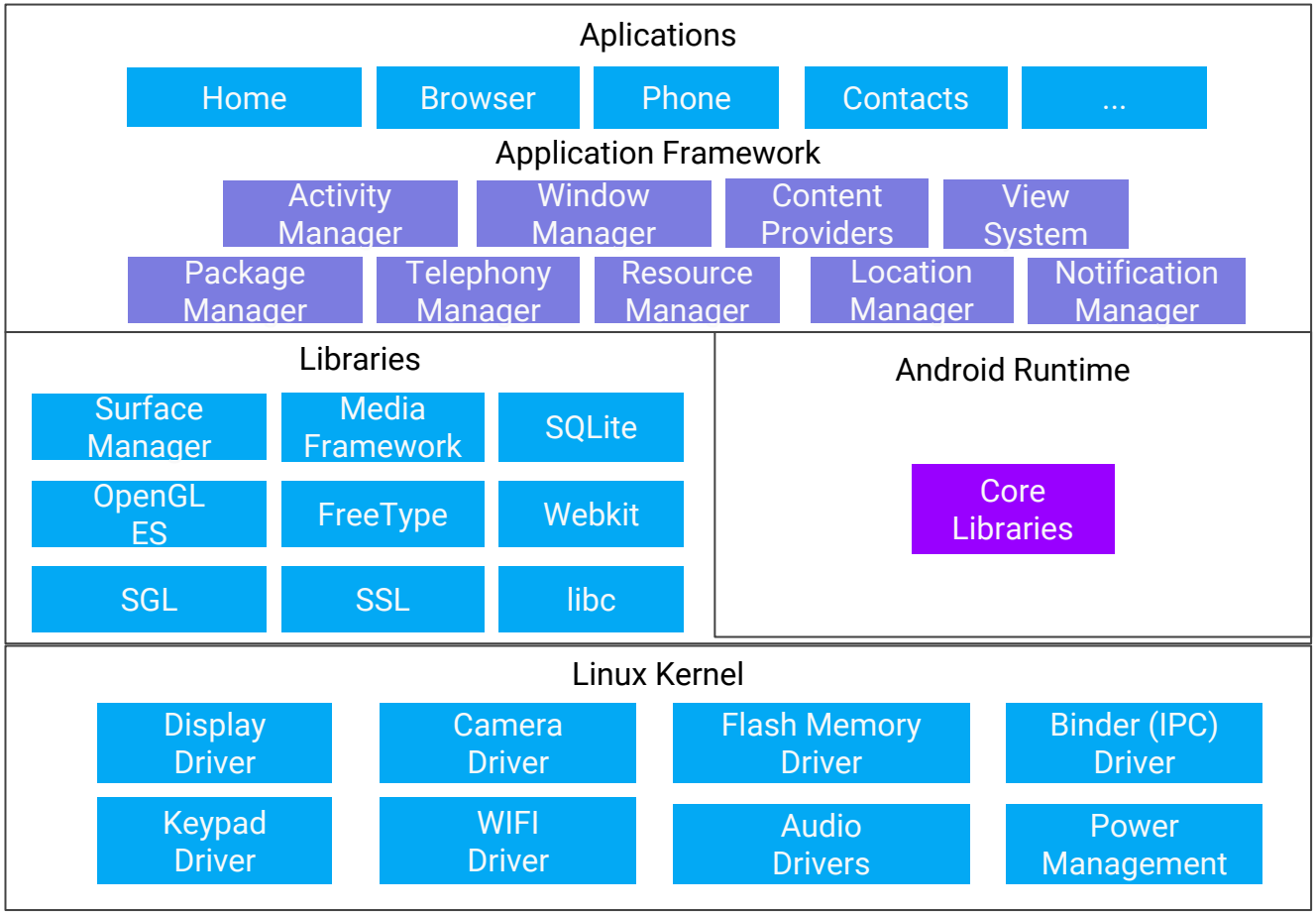
- Libraries
  - RecyclerView
- Adapters
  - RecyclerViewAdapter
  - ViewHolder pattern
  - Custom “item” view
- Extras

Where is Android in the world?

- 1.5 Billion active devices worldwide!
- 2015 Q2 82% market share
- 2 to 3 million activated each day!
- 22+ version codes and counting!
- “Multi form factors” reach (TV, Auto, Mobile, Tablets, Wear)
- Written in Java
- Other languages supported
  - C++ via the NDK
  - Kotlin (Statically typed programming language)
  - C# thouth Xamarin

Sources: IDC, Android developers

Platform Dissection



Android Studio

(2.0 is stable)

- Flexible **Gradle**-based build system
- Build variants and *multiple* apk file generation
- Code templates for common features
- Rich layout editor with drag and drop support theme editing
- lint tools to catch performance, version compatibility and more
- ProGuard and app-signing capabilities
- Build-in support for **Google Cloud Platform**

http://developer.android.com/tools/studio/index.html

Intents

- Represents an abstract description of an operation to be performed.

- Used for lots of things
  - Start an **Activity**
  - **Broadcast** an Intent
  - Start a **Service**
  - Communicate with other **apps**

- They can be:
  - *Implicit* - an intent that we don't know which component should be launched. (example: Launching an url, but with what browser?)
  - *Explicit* - an intent that we define its actions, with a specific component. (example: Launching Activity **B** from Activity **A** within our app)

Example

```
//An implicit intent.
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, messageText);
String chooserTitle = getString(R.string.chooser);
Intent chosenIntent = Intent.createChooser(intent, chooserTitle);
startActivity(chosenIntent);
```

Example

```
//An implicit intent.
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, messageText);
String chooserTitle = getString(R.string.chooser);
Intent chosenIntent = Intent.createChooser(intent, chooserTitle);
startActivity(chosenIntent);
```

Example

```
//An implicit intent.
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, messageText);
String chooserTitle = getString(R.string.chooser);
Intent chosenIntent = Intent.createChooser(intent, chooserTitle);
startActivity(chosenIntent);
```

Example

```
//An implicit intent.
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, messageText);
String chooserTitle = getString(R.string.chooser);
Intent chosenIntent = Intent.createChooser(intent, chooserTitle);
startActivity(chosenIntent);
```

Example

```
//An explicit intent
Intent intent = new Intent(this, ReceiveIntentActivity.class);
intent.putExtra(ReceiveIntentActivity.EXTRA_MESSAGE, messageText);
startActivity(intent);
```

Example

```
Intent intent = new Intent(this, ReceiveIntentActivity.class);
intent.putExtra(ReceiveIntentActivity.EXTRA_MESSAGE, messageText);
startActivity(intent);
```

Example

```
Intent intent = new Intent(this, ReceiveIntentActivity.class);
intent.putExtra(ReceiveIntentActivity.EXTRA_MESSAGE, messageText);
startActivity(intent);
```

Coding time

(pray to the Demo Gods)

Part 2

(getting good @ Android)

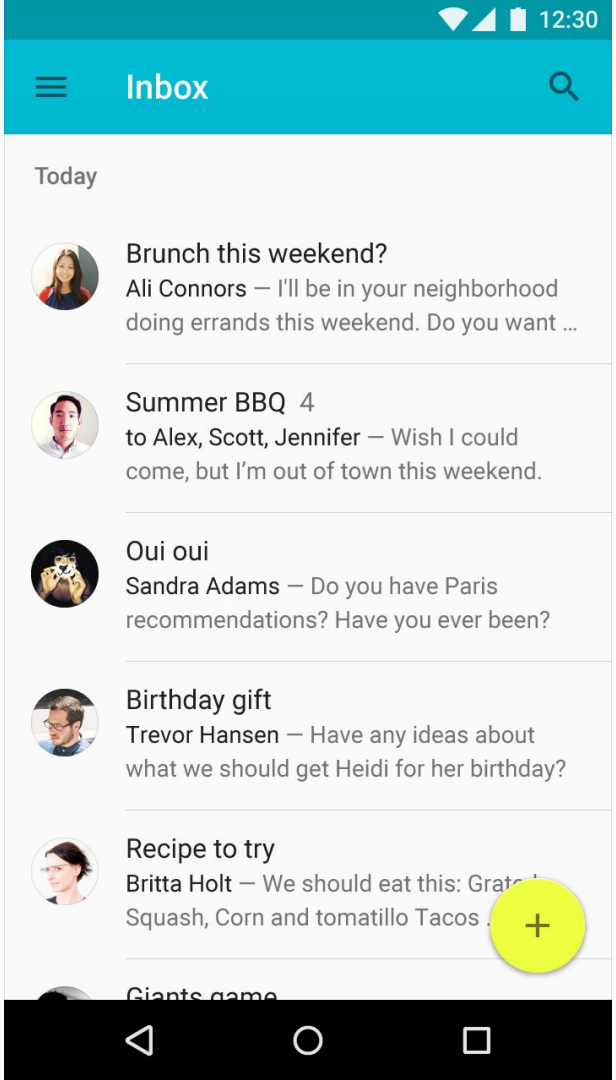
Lists

RecyclerView

- The “standard”
- Flexible
- Decoupled
- Multi-ViewTypes

“RecyclerView has the ‘Recycler’ part at the center of it’s purpose”

Custom Views within RecyclerView



```
public class NameViewHolder extends RecyclerView.ViewHolder {

    public TextView nameTextView, cityTextView;

    public NameViewHolder(View itemView) {
        super(itemView);
        nameTextView = (TextView) itemView.findViewById(R.id.name_text_view);
        cityTextView = (TextView) itemView.findViewById(R.id.city_text_view);
    }
}
```

```
public class NameViewHolder extends RecyclerView.ViewHolder {

    public TextView nameTextView, cityTextView;

    public NameViewHolder(View itemView) {
        super(itemView);
        nameTextView = (TextView) itemView.findViewById(R.id.name_text_view);
        cityTextView = (TextView) itemView.findViewById(R.id.city_text_view);
    }
}
```

```
public class NameViewHolder extends RecyclerView.ViewHolder {

    public TextView nameTextView, cityTextView;

    public NameViewHolder(View itemView) {
        super(itemView);
        nameTextView = (TextView) itemView.findViewById(R.id.name_text_view);
        cityTextView = (TextView) itemView.findViewById(R.id.city_text_view);
    }
}
```

The Adapter

```
public class NamesAdapter extends RecyclerView.Adapter<NameViewHolder> {

    /**
     * We should always a context to which 'construct" call can be made
     * Inflation and so on.
     */
    private Context context;

    public NamesAdapter(Context context) {
        this.context = context;
    }
```

The Adapter

```
public class NamesAdapter extends RecyclerView.Adapter<NameViewHolder> {

    /**
     * We should always have a context to which  a 'construct" call can be made
     * Inflation and so on.
     */
    private Context context;

    public NamesAdapter(Context context) {
        this.context = context;
    }
```

The Adapter

```
public class NamesAdapter extends RecyclerView.Adapter<NameViewHolder> {

@Override
public NameViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            //implement
}

@Override
public void onBindViewHolder(NameViewHolder holder, int position) {
         //implement
}

@Override
public int getItemCount() {
   //implement
}
```

The Adapter

```
public class NamesAdapter extends RecyclerView.Adapter<NameViewHolder> {

@Override
public NameViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(context).inflate(R.layout.item_view, null);
return new NameViewHolder(v);

}

@Override
public void onBindViewHolder(NameViewHolder holder, int position) {
        //implement
}

@Override
public int getItemCount() {
   //implement
}
```

The Adapter

```
public class NamesAdapter extends RecyclerView.Adapter<NameViewHolder> {

@Override
public NameViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            //implement
}

@Override
public void onBindViewHolder(NameViewHolder holder, int position) {
        holder.nameTextView.setText("Joel " + position);
        holder.cityTextView.setText("Ponce " + position);
}
```

The Adapter

```
public class NamesAdapter extends RecyclerView.Adapter<NameViewHolder> {

@Override
public NameViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            //implement
}

@Override
public void onBindViewHolder(NameViewHolder holder, int position) {
        //implement
}

@Override
public int getItemCount() {
   return 10; //in case of Collection you can use collection.size()
}
```

Put it all together

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

    //build upon
    namesRecyclerView = (RecyclerView) findViewById(R.id.names_recycler_view);
    namesRecyclerView.setLayoutManager(new LinearLayoutManager(this));
    namesRecyclerView.setItemAnimator(new DefaultItemAnimator());
    namesRecyclerView.setHasFixedSize(true);
    namesRecyclerView.setAdapter(new NamesAdapter(this));
}
```

Demo 2

(pray once more really hard to the demo Gods)

Bonus

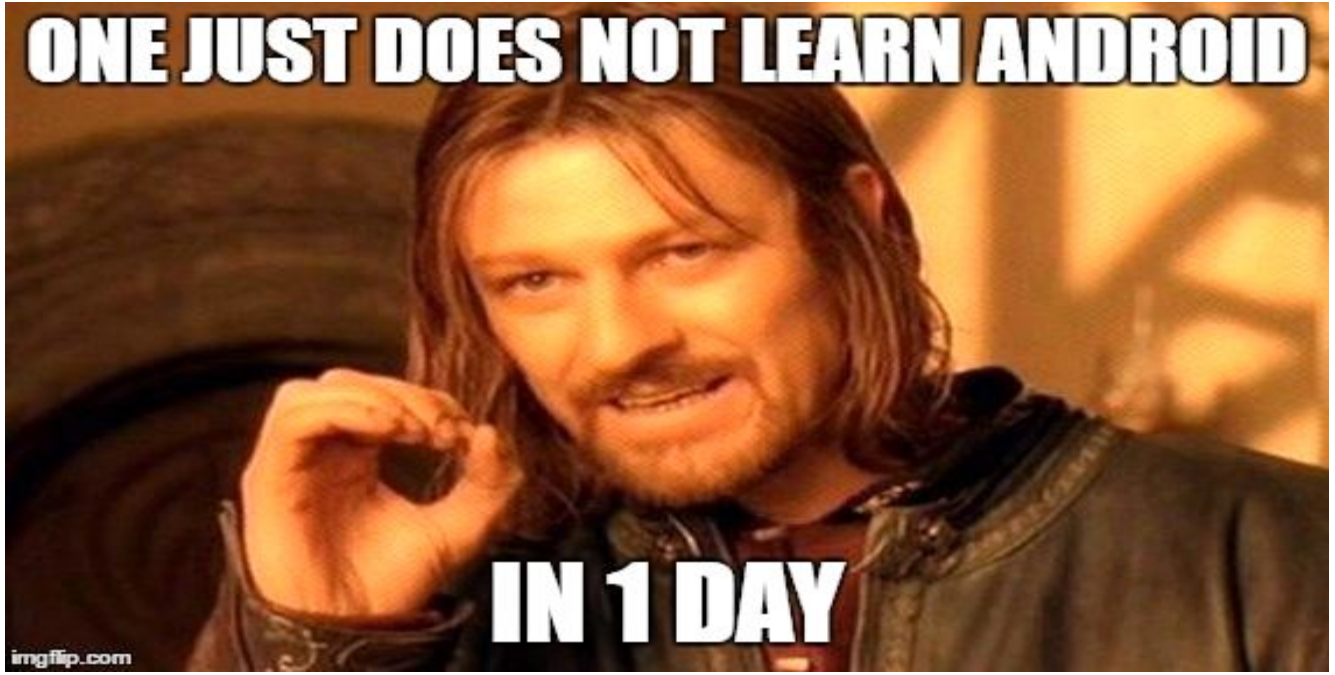
Resources:

Libraries version hub:
http://gradleplease.appspot.com/

Awesome Image downloading and caching
http://square.github.io/picasso/

Upcoming Android blog:
http://nativeandroid.com/

Disclaimer



There still lot’s more!

(too much to present in 30 mins)

- RecyclerView
- CoordinatorLayout
- Snackbars
- <include>’s
- Flavors
- Fragments
- Gradle Tasks
- Unit Testing

- Service
- IntentService
- BroadcastIntent
- Custom Components
- Libraries
- AsynTasks
- Third party API’s
- Many many many… more :)

Thank you

Android Ninja Academy

Sensei - Joel Sosa

@JRSosa

Disclaimer: *GDG Puerto Rico is an independent group; our activities and the opinions expressed should in no way be linked to Google, the corporation.*