Why Jetpack

Kurt Nelson

Jetpack is a collection of Android software components to make it easier for you to develop great Android apps. These components help you follow best practices, free you from writing boilerplate code, and simplify complex tasks, so you can focus on the code you care about.

— developer.android.com

Jetpack comprises the androidx.* package libraries, unbundled from the platform APIs. This means that it offers backward compatibility and is updated more frequently than the Android platform, making sure you always have access to the latest and greatest versions of the Jetpack components.

— developer.android.com

Let's build a brand new app

You're a brand new fresh developer who just finished taking a Java 101 course.

I know java, so I can build my first Android app right?

Let's create a new app in Android Studio!

Automatic Dependencies

```
implementation 'com.android.support:appcompat-v7:28.0.0-rc01'
implementation 'com.android.support.constraint:constraint-layout:1.1.2'
testImplementation 'junit:junit:4.12'
androidTestImplementation 'com.android.support.test:runner:1.0.2'
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
```

How do I get data to the UI?

How can I load an image?

How do I move between screens?

What's this background check?

New Developers on Existing Apps Have The Same Questions.

A solution is an opinionated framework!

At Google

Inside of Google, we built TikTok.

- It came out of Google Plus
- Heavily relied on the google3 monorepo and visibility restrictions
- Handled basically everything Jetpack does, but Google specific
- A tech writer built out guides for common use cases

Big Tech

Outside of Google at large-ish companies, you can have a platform/framework team.

- Reviews any new dependencies
- Resolves disputes during code review
- Provides testing infrastructure
- (Hopefully) has the institutional knowledge on what works for the app

Everyone Else

Google things, pick a library, and hope?

Jetpack is a "right" way to build apps.

Jetpack is a "right" way to build apps.

- You'll be able to get support in the common places.
- If something is broken, you can file a bug on Google.
- The libraries work together

Jetpack is not always the *best* way to build apps.

- Some libraries such as LiveData are far less powerful than popular alternatives like Rx.
- DataBinding is polarizing.
- DownloadManager has security issues on older platforms.
- Navigation works until you need to break up your build into independent targets.
- Single Activity apps have widely different needs.

tldr; Use Jetpack until it doesn't fit your needs

Jetifier

android.support.foo -> androidx.bar.foo

Jetifier

Isn't this just a rename?

For now.

Jetifier

- In the future, more "OS" classes can cleanly move into the APK
- Other libraries can depend on androidx.core without pulling in swaths of unused dependencies
- Views can start being decoupled from the OS version.

Jetifier

- Some third party libraries break right now if you turn it on.
- Libraries that perform code-gen need to be explicity jetpack aware.
- Google is having to support this rename in their monorepo too, they are going through the same suffering.

Let's dive into the components.

Navigation

"Official" Activity Navigation

```
public void sendMessage(View view) {
  Intent intent = new Intent(this, DisplayMessageActivity.class);
  EditText editText = (EditText) findViewById(R.id.editText);
  String message = editText.getText().toString();
  intent.putExtra(EXTRA_MESSAGE, message);
  startActivity(intent);
}
```

Traditional Activity Navigation

```
class SomeActivity extends Activity {
    private static final EXTRA_MESSAGE = "message";

    public Intent buildIntent(Context context, String msg) {
        Intent intent = new Intent(context, SomeActivity.class);
        intent.putExtra(EXTRA_MESSAGE, message);
        return intent;
    }
}
```

```
<navigation xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    app:startDestination="@id/mainFragment">
    <fragment
        android:id="@+id/mainFragment"
        android:name="com.example.cashdog.cashdog.MainFragment"
        android:label="fragment_main"
        tools:layout="@layout/fragment_main" >
        <action
            android:id="@+id/action_mainFragment_to_chooseRecipient"
            app:destination="@id/sendMoneyGraph" />
        <action
            android:id="@+id/action_mainFragment_to_viewBalanceFragment"
            app:destination="@id/viewBalanceFragment" />
    </fragment>
</navigation>
```

```
Navigation.findNavController(view).navigate(R.id.actionName);
```

Navigation

- No popular libraries already exist in this space, most devs have been happy with building intents themselves
- Most beneficial if app is fully using fragments, semi-useful if activity heavy, not a good fit if single activity app
- Does not help break up dependencies

WorkManager

- The best part of jetpack.
- No more need to balance Firebase, Job Scheduler, AlarmManager across API versions.
- Libraries can and should use it.

Room

It's quite nice!

- The built in SQLiteOpenHelper is very powerful but hard to use in a performant non-leaky manner, especially asynchronously.
- Room goes pretty far in the Data Access Object direction.
- SQLBrite/Delight might be better if you have a complex data model and developers who know SQL well.
- Unclear on the binary size hit that Room causes in large apps yet.

ViewModel

It's better loaders.

```
MyViewModel model = ViewModelProviders.of(this).get(MyViewModel.class);
        model.getUsers().observe(this, users -> {
                // update UI
});
```

Lifecycle

OS fragments are deprecated soon meaning libraries can assume support fragments.

LiveData

Differing opinions on when it should be used.

If you are comfortable with Rx, no need to change.

I Still Want

- Image Loading
- Keyboard compat!

Other Sessions

Adventures in Navigation
Android U+2764 (❤) Emoji
Architecting an app with MVP and ViewModels
Firebase and Android Architecture Components: fit like a glove
From AlarmManager to WorkManager
Rinsing the Brush: Picasso 3.0

Slides: github.com/kurtisnelson/presentations