NAMRATA BANDEKAR

VIDEO PROCESSING ON ANDROID

WHO AM I

- OANDA
- raywenderlich.com
- Ginger!

INTRODUCTION TO

VIDEO PROCESSING

POPULAR USE IN

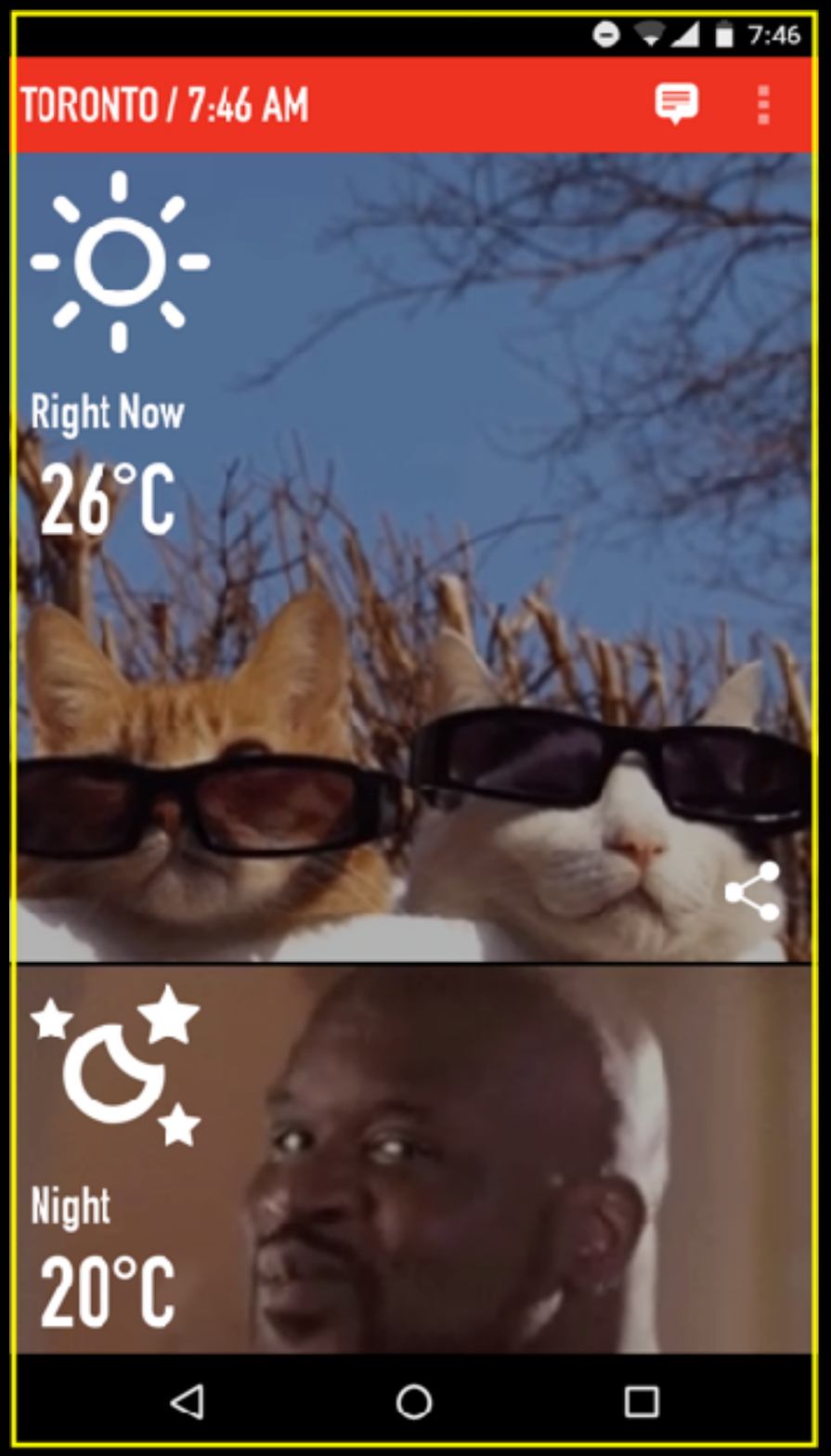
MOBILE APPS

COMMON FEATURES

- Special video effects
- Merging videos
- Cropping and trimming

MY SECOND JOB!

WEATHERGIF



FEATURE LIST?

TORONTO / 4:51 PM

Right Now

24°C

Night

20°C

IOS WEATHERGIF

VIDEO EDITING

ON DEVICE

ANDROID SUPPORT

- MediaCodec
- MediaExtractor
- MediaMuxer

WHAT NEXT?

FFMPEG

WHY FFMPEG?

- Swiss Army Knife for video
- Filters
- Subtitles

FFMPEG ON ANDROID

- Executable binary
- Shared object library

BUILDING THE

FFMPEG EXECUTABLE

DEPENDENCIES

- autoconf
- automake
- libtool

GETTING STARTED

Step 1: Download and unpack NDK

GETTING STARTED

Step 2: Download FFmpeg source code

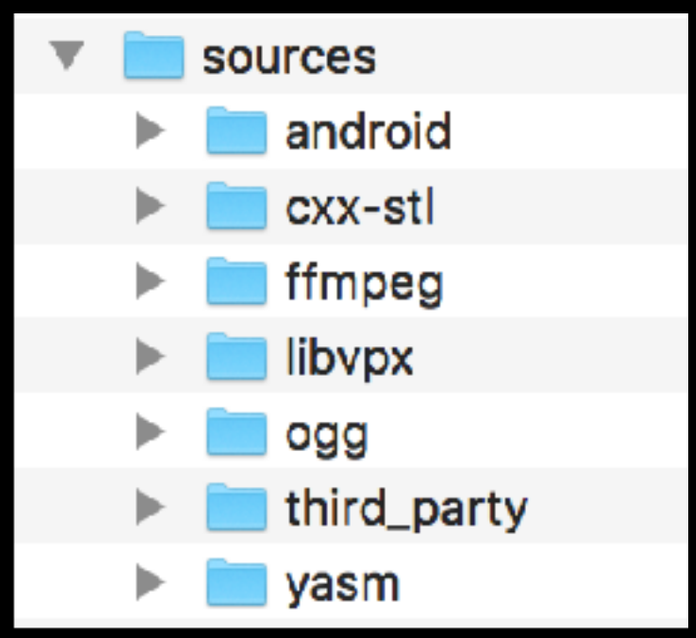
https://github.com/FFmpeg/FFmpeg

GETTING STARTED

Step 3: Download sources for other libraries like yasm and libvpx

GETTING STARTED

Step 4: Put FFmpeg and other libraries under ndk/sources



CONFIGURING...

```
./configure \
--target-os="$TARGET_OS" \
--arch="$NDK_ABI" \
--sysroot="$NDK_SYSROOT" \
--enable-pic \
--enable-libx264 \
\
--enable-decoders --enable-encoders \
--enable-muxers --enable-demuxers \
--enable-filters \
\
--enable-hwaccels --disable-debug \
\
--enable-ffmpeg --disable-ffplay \
--disable-ffprobe --disable-ffserver \
\
--enable-yasm \
--disable-shared
```

CONFIGURING

```
./configure \
--target-os="$TARGET_OS" \
--arch="$NDK_ABI" \
--sysroot="$NDK_SYSROOT" \
```

CONFIGURING

```
--enable-decoders \
--enable-encoders \
--enable-muxers \
--enable-demuxers \
--enable-filters \
```

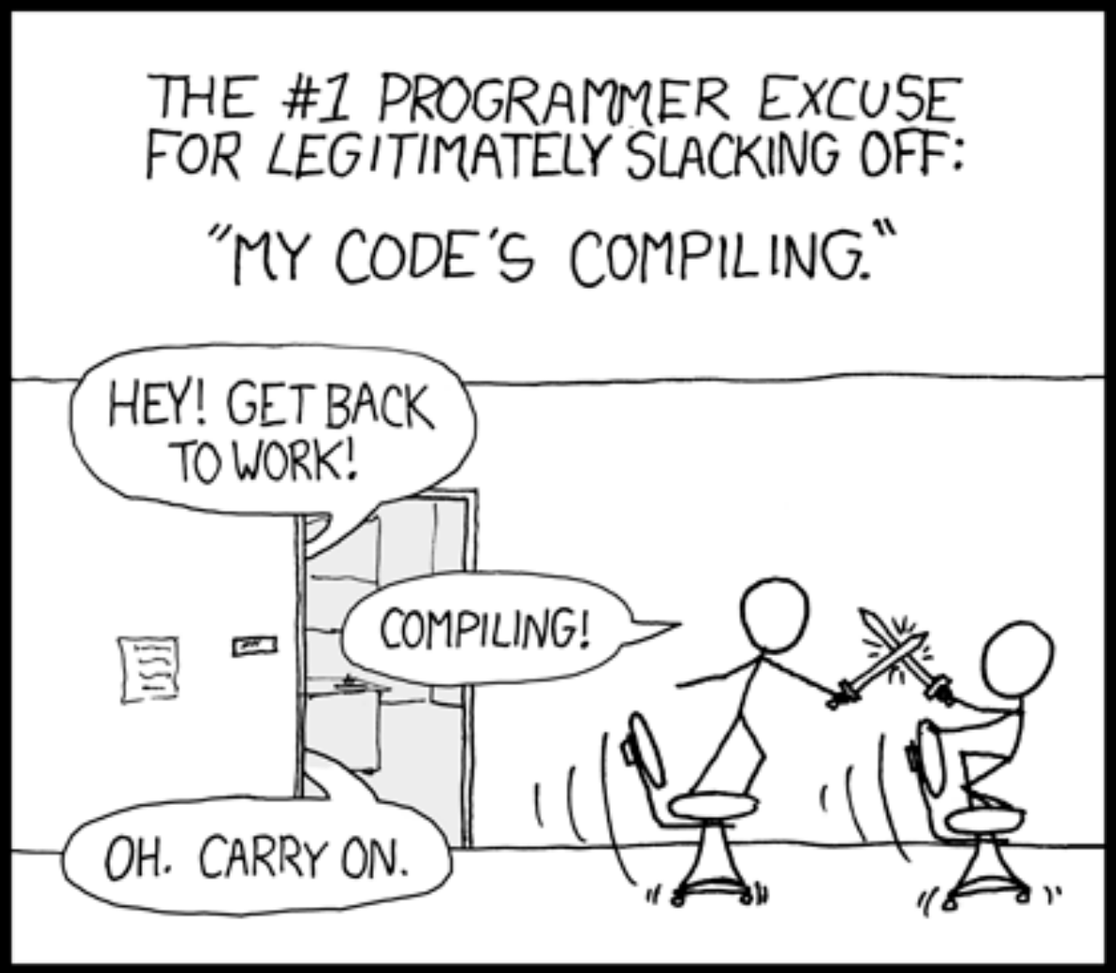
CONFIGURING

```
--enable-decoder=h263 \
--enable-decoder=h264 \
--enable-decoder=mpeg4 \
--enable-decoder=libvpx_vp8 \
--enable-encoder=libvpx_vp8
```

CONFIGURING

```
--enable-libx264 \
```

COMPILING




```
make -j4
make install
```

FFMPEG EXECUTABLE

IN YOUR ANDROID PROJECT

CHANGE PERMISSIONS

```
try {
  File f = new File(ctx.getDir("bin", 0), "FFmpeg");
  if (f.exists()) {
  f.delete();
  }
  copyRawFile(ctx, R.raw.ffmpeg, f);
  // Change the permissions
  String filePath = f.getCanonicalPath();
  Runtime.getRuntime().exec("chmod 0755 "+
    filePath).waitFor();
} catch (Exception e) {
  String errorMsg = e.getLocalizedMessage();
  Log.e(TAG, "installBinary failed: "+errorMsg);
  return null;
}
```

PROCESS BUILDER

```
private int execProcess(List<String> cmds) {
    File dir = new File(ffmpegBin).getParentFile();
    ProcessBuilder pb = new ProcessBuilder(cmds);
    pb.directory(dir);

    Process process = pb.start();

    int exitVal = process.waitFor();
    return exitVal;
}
```

FILTER CHAINING

```
ArrayList<String> cmd = new ArrayList<String>();

cmd.add(mFfmpegBin);
cmd.add("-y");
cmd.add("-i");
cmd.add(new File(inputVideo.path).getCanonicalPath());
cmd.add("-vf");
```

```
cmd.add("movie="+watermarkImg.path+" [logo];
    [in] scale="+width+":"+height+" " +"[scaled];
    [scaled] crop=“+newWidth+":"+newHeight+" [cropped];
    [cropped][logo] overlay=0:0 [out]");

result.path = outputPath;
result.mimeType = "video/mp4";

cmd.add(new File(result.path).getCanonicalPath());
execFFMPEG(cmd, sc);
```

PERFORMANCE

BUILDING FFMPEG AS A

SHARED OBJECT LIBRARY

SETUP FOR FFMPEG.SO

Step 1: JNI project same folder as NDK

SETUP FOR FFMPEG.SO

Step 2: Android.mk file

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := videokit
ANDROID_LIB := -landroid
LOCAL_CFLAGS := -I$(NDK)/sources/ffmpeg
LOCAL_SRC_FILES :=  videokit.c ffmpeg.c cmdutils.c
LOCAL_SHARED_LIBRARIES := libavcodec libavutil libavfilter

include $(BUILD_SHARED_LIBRARY)
$(call import-module,ffmpeg/android/$(CPU))
```

SETUP FOR FFMPEG.SO

Step 3: Application.mk file

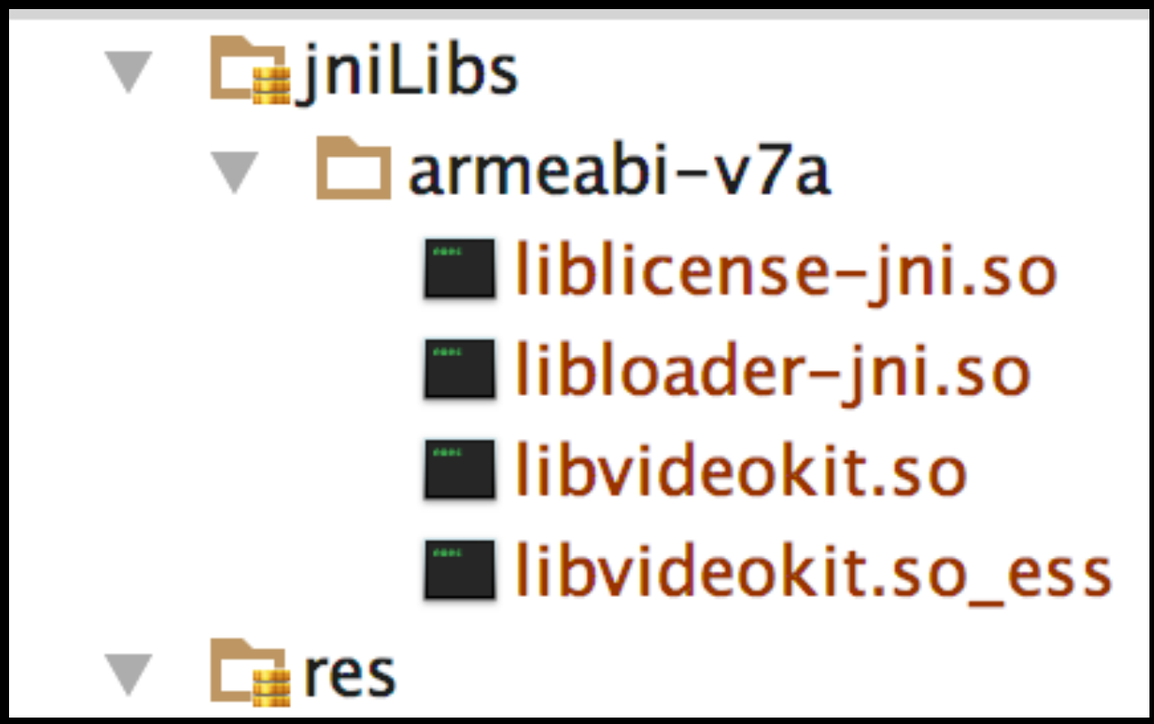
```
APP_OPTIM := release
APP_PLATFORM := $(PLATFORM)
APP_ABI := $(ABI)
NDK_TOOLCHAIN_VERSION=4.9
APP_STL := stlport_shared
```

SETUP FOR FFMPEG.SO

Step 4: Place native code in app/jni

NDK MAGIC

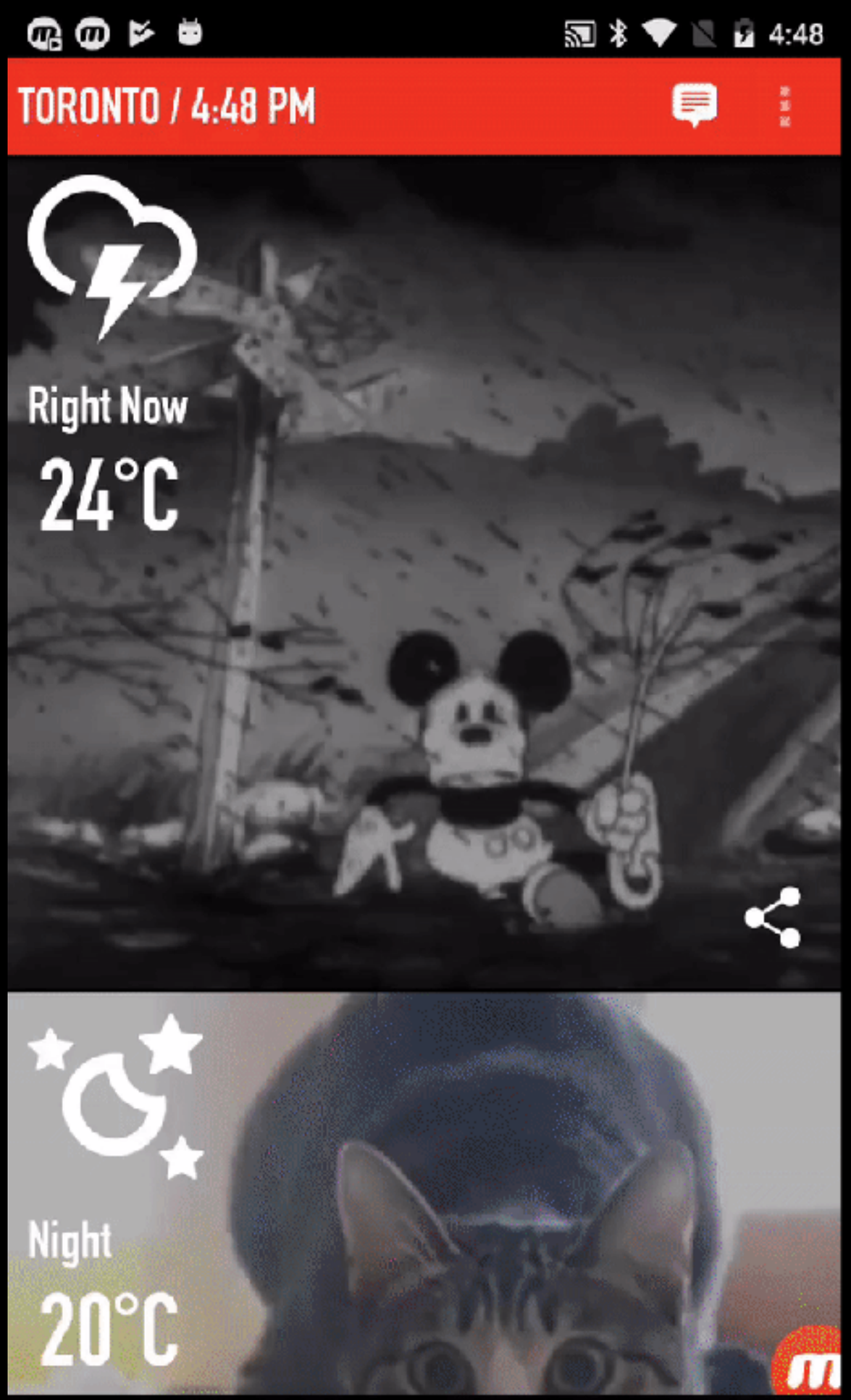
ndk-build



LOAD SHARED LIBRARY

```
public class VideoKit {
  static {
    System.loadLibrary("ffmpeg");
  }
// pass commands to ffmpeg
  private native int run(int loglevel, String[]
args);
}
```

WEATHERGIF DEMO


4:48
TORONTO / 4:48 PM
Right Now
24°C
Night
20°C

LICENSING

License Compliance Checklist

The following is a checklist for LGPL compliance when linking against the FFmp

1. Compile FFmpeg **without** "--enable-gpl" and **without** "--enable-nonfree
2. Use dynamic linking (on windows, this means linking to dlls) for linking v
3. Distribute the source code of FFmpeg, no matter if you modified it or not
4. Make sure the source code corresponds exactly to the library binaries yo
5. Run the command "git diff > changes.diff" in the root directory of the FFm
6. Explain how you compiled FFmpeg, for example the configure line, in a te
7. Use tarball or a zip file for distributing the source code.
8. Host the FFmpeg source code on the same webserver as the binary you a
9. Add "This software uses code of <a href=http://ffmpeg.org>FFmpeg</a
10. Mention "This software uses libraries from the FFmpeg project under the
11. Mention in your EULA that your program uses FFmpeg under the LGPLv2.
12. If your EULA claims ownership over the code, you have to **explicitly** ment
13. Remove any prohibition of reverse engineering from your EULA.
14. Apply the same changes to all translations of your EULA.
15. Do not misspell FFmpeg (two capitals F and lowercase "mpeg").
16. Do not rename FFmpeg dlls to some obfuscated name, but adding a suffi
17. Go through all the items again for any LGPL external library you compiled

LICENSE COMPLIANCE

~~--enable-gpl~~ ~~--enable-nonfree~~

LICENSE COMPLIANCE

- Dynamic linking
- FFmpeg source
- Configure script

LICENSE COMPLIANCE

This software uses code of FFmpeg (http://ffmpeg.org) licensed under the LGPLv2.1 (http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html) and its source can be downloaded here: https://github.com/Yelp/ffmpeg-android

LICENSE COMPLIANCE

```
RIIS-MBPRO-13:assets User$ tree
.
├── COPYING.GPLv2
├── COPYING.GPLv3
├── COPYING.LGPLv2.1
├── COPYING.LGPLv3
├── LICENSE.md
├── arm
│   └── pie
│       └── ffmpeg
├── crashlytics-build.properties
└── x86
    └── ffmpeg
```

FFMPEG PROS

- Modular
- Support for codecs
- Versatile
- Well documented

FFMPEG CHALLENGES

- Licensing
- Slow
- Large

MP4PARSER

MP4PARSER FEATURES

- Concatenation
- Trimming
- Muxing
- Demuxing

MP4PARSER USAGE

```
RxMp4Parser.concatenateInto(
  //The output where the resulting Movie object should be stored
  output,
  //Cropped video
  RxMp4Parser.crop(f, 8.5f, 13f),
  //The entire video
  RxMp4Parser.from(f)
  )
  .subscribe(new Action<File>() {
    @Override
    public void call(File file) {
      mProgressDialog.dismiss();
    }
  }, new Action<Throwable>() {
    @Override
    public void call(Throwable throwable) {
      Toast.makeText(MainActivity.this, "Concatenation failed! " +
        throwable.getMessage(), Toast.LENGTH_SHORT).show();
    }
  });
```

MP4PARSER DEMO

5:01

mp4parser

CONCATENATE VIDEO

MP4PARSER PROS

- Java library
- Fast
- Clean API

MP4PARSER CHALLENGES

- Limited functionality
- No encoding and decoding
- Concatenates only if inputs have same format

RESOURCES

- https://github.com/guardianproject/android-ffmpeg
- https://github.com/Yelp/ffmpeg-android
- https://github.com/inFullMobile/videokit-ffmpeg-android
- https://www.ffmpeg.org/legal.html
- https://github.com/sannies/mp4parser
- https://github.com/TeamWanari/RxMp4Parser

THANKS AND CREDITS

- Dhaval Giani
- Godfrey Nolan
- GDG Toronto
- Toronto Android Developers Meetup
- OANDA

THANK YOU!

QUESTIONS?

@namrataCodes

https://github.com/namrata-b/talks/