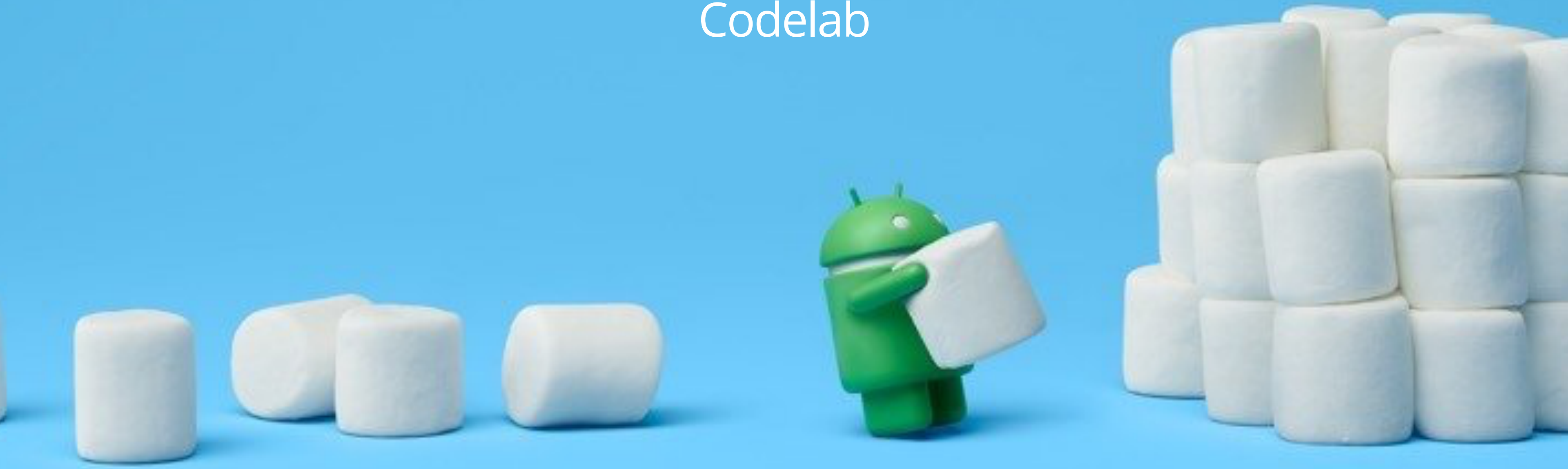
# App Permissions

Codelab



GDG DevFest
2015 Season

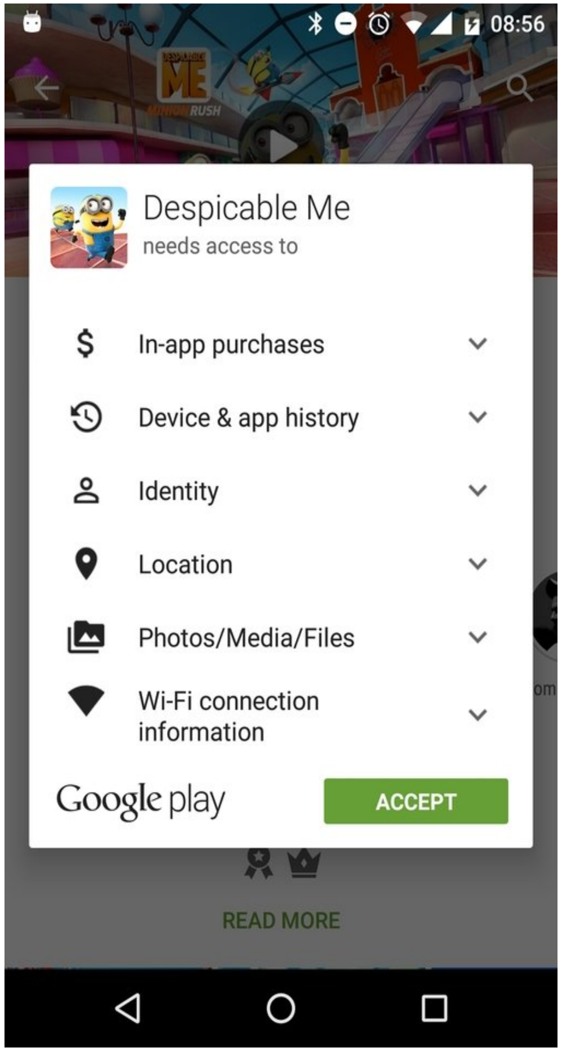
## Who am I?!

Deuphil Kaufmann
@deubaka
SDK Developer @ HyperGrowth
PADC Co-organizer
Java/Android/Mobile

## Prerequisite

- Build tools at least 23.x.x
- Support library at least 23.x.x
- Android 6.0 Emulator/Device

# Permissions - then and now

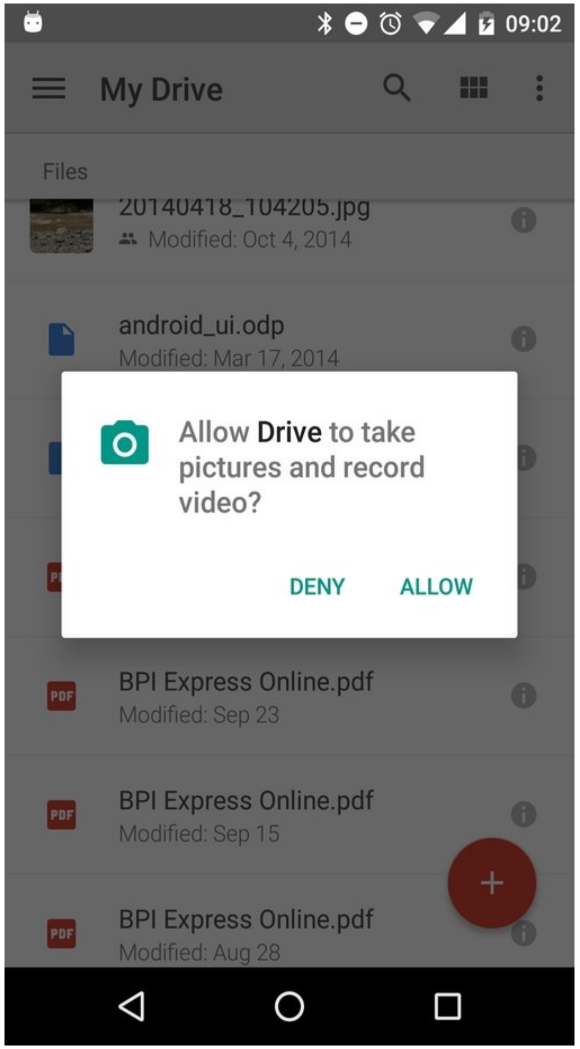


08:56

Despicable Me
needs access to

- In-app purchases
- Device & app history
- Identity
- Location
- Photos/Media/Files
- Wi-Fi connection information

Google play — ACCEPT



09:02

My Drive

Files

20140418_104205.jpg
Modified: Oct 4, 2014

android_ui.odp
Modified: Mar 17, 2014

Allow **Drive** to take pictures and record video?

DENY ALLOW

BPI Express Online.pdf
Modified: Sep 23

BPI Express Online.pdf
Modified: Sep 15

BPI Express Online.pdf
Modified: Aug 28

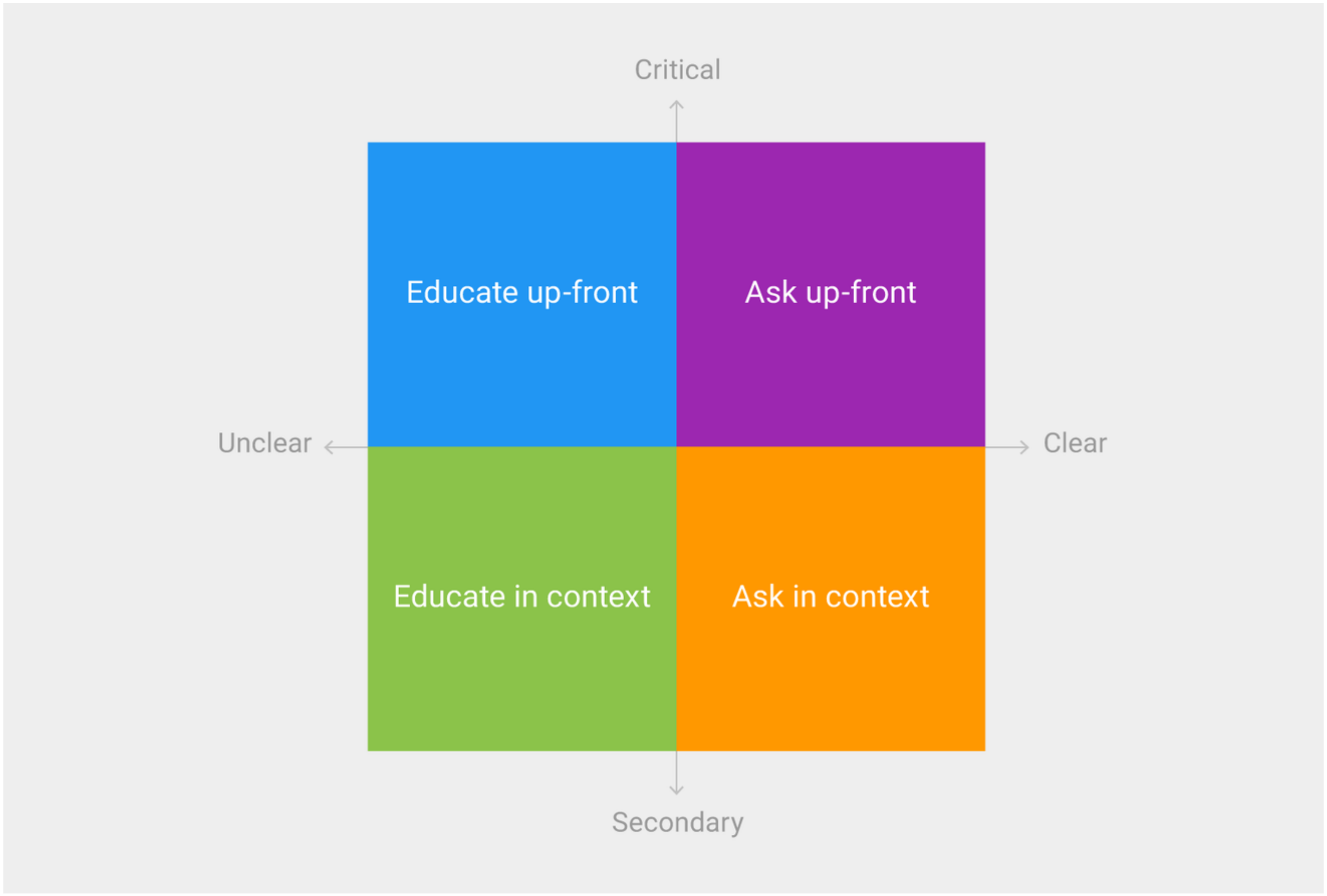
# But why?

- Permissions were out of context and vague
  - Users will tap it anyway
- Created friction during installs/updates
- All or nothing
- No post installation user control

## Which in what?

- **8 Permission Group**
  - Contacts
  - Calendar
  - SMS
  - Location
  - Camera
  - Sensors
  - Microphone
- **2 Protection Categories**
  - Normal - Enabled by default on install time
    - ACCESS_NETWORK_STATE
    - INTERNET
    - BLUETOOTH
  - Dangerous - Must request on runtime

# When and how to ask?



## How to?

- Check for permission before executing affected code
  - Use *ContextCompat*-subclasses to access *checkSelfPermissions*

```
if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
        == PackageManager.PERMISSION_GRANTED) {
    // Do code here
} else {
    // Request permission here
}
```

- Use *ActivityCompat.requestPermissions()* to request permissions

```
if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CAMERA) {
    Snackbar.make(mAnchorLayout, "Camera permission is needed to attach pictures",
            Snackbar.LENGTH_INDEFINITE)
            .setAction(R.string.ok, new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    ActivityCompat.requestPermissions(MainActivity.this,
                            new String[]{Manifest.permission.CAMERA}, REQUEST_CAMERA);
                }
            })
            .show();
} else {
    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
            REQUEST_CAMERA);
}
```

...

- Implement *ActivityCompat.OnRequestPermissionsResultCallback*

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
        @NonNull int[] grantResults) {

    if (requestCode == REQUEST_CAMERA) {
        if (grantResults.length == 1 && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {

            Snackbar.make(mAnchorLayout, "You can now take photos",
                    Snackbar.LENGTH_SHORT).show();
        } else {

            Snackbar.make(mAnchorLayout, "Permission not granted",
                    Snackbar.LENGTH_SHORT).show();
        }
    }
}
```

## Gotchas

- If device is API 22/Lollipop and below OR *targetSdk* is 22 and below, then permissions will be granted at install time
- If device is API 23/Marshmallow and above AND *targetSdk* is 23 and above, then permissions can be denied and are runtime
- If device is API 23/Marshmallow BUT *targetSdk* is 22 and below, then permissions can still be denied, but won't crash
- Once "Never ask again" is ticked, *ActivityCompat.shouldShowRequestPermissionRationale* will always return false
- Option to redirect to app info page only

## Some things to keep in mind

- You still need to define all your app's permission in your AndroidManifest.xml
- Consider using an intent to offload the permission request, ex. launch camera app instead
- Only ask when you need it (see quadrant)
- Be mindful not to overwhelm user
- Explain why you need it (see quadrant again)
- By providing runtime permissions on your app, you create a "trust" between your product and user

End

## References and further readings

- Working with System Permissions
- Permissions Patterns
- Best Practices