Google
Developers

Mobile Best Practices with YouTube

Google I/O 2013

Ikai Lan
Developer Relations, YouTube

Platforms

Agenda

- General tips
- User experience
  - Graceful failover
  - Transitions
- Authentication / Authorization
  - OAuth, Play Services
  - YouTube channels and Google Accounts
- **These best practices came from talking to our partners in the field!**

General best practices

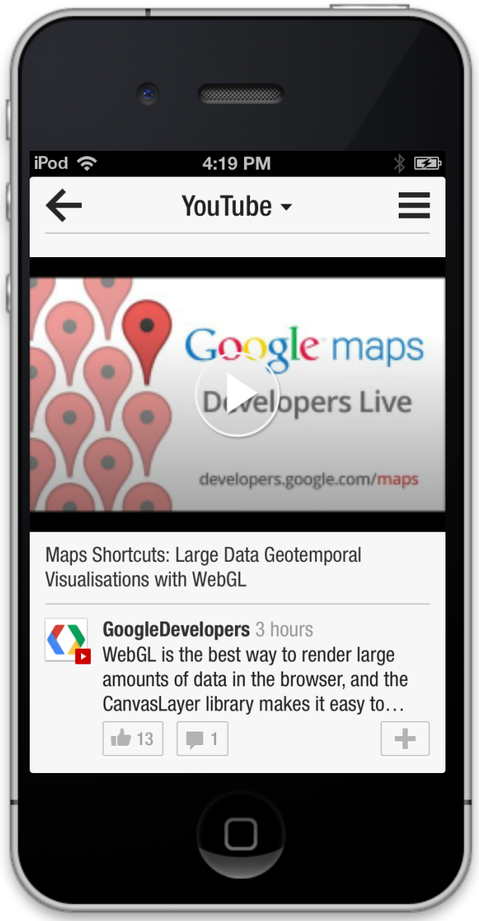
- Follow your platform best practices
  - Don't block the UI thread
  - UX best practices
- Understand YouTube terms of service
  https://developers.google.com/youtube/terms
- Different form factors
  - Phone / "Phablet"
  - 7" tablet
  - 10" tablet
  - @see "Designing products for multi screen world: YouTube"
    Day 2, 3:30pm on the YouTube track

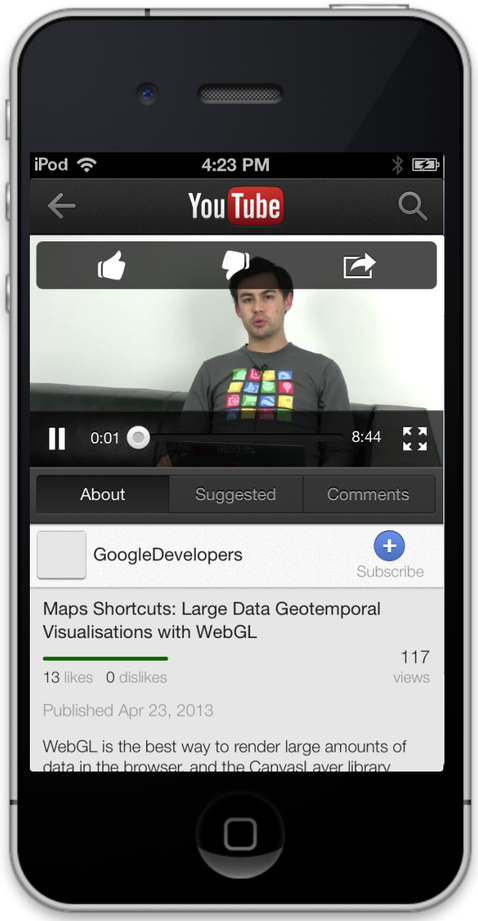
Best Practices

Video playback

iOS playback options

iframe Player API in a WebView





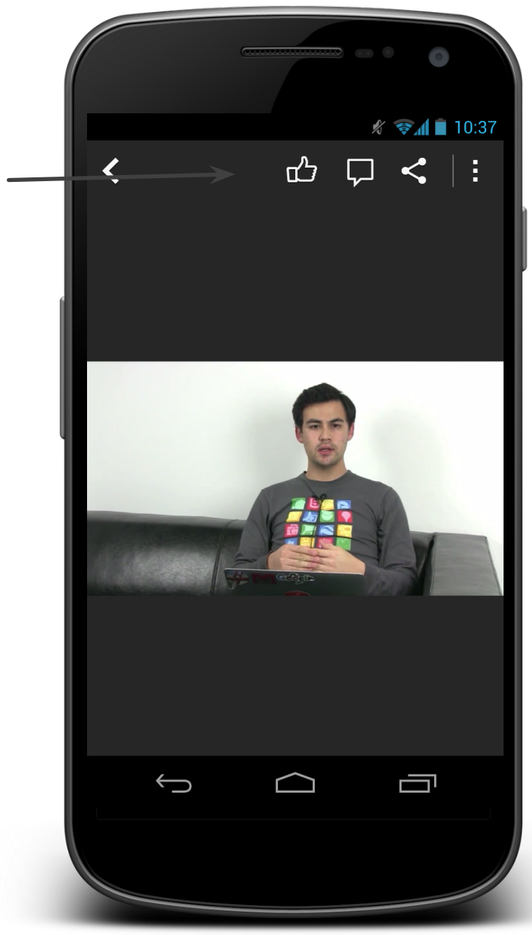
Official YouTube app

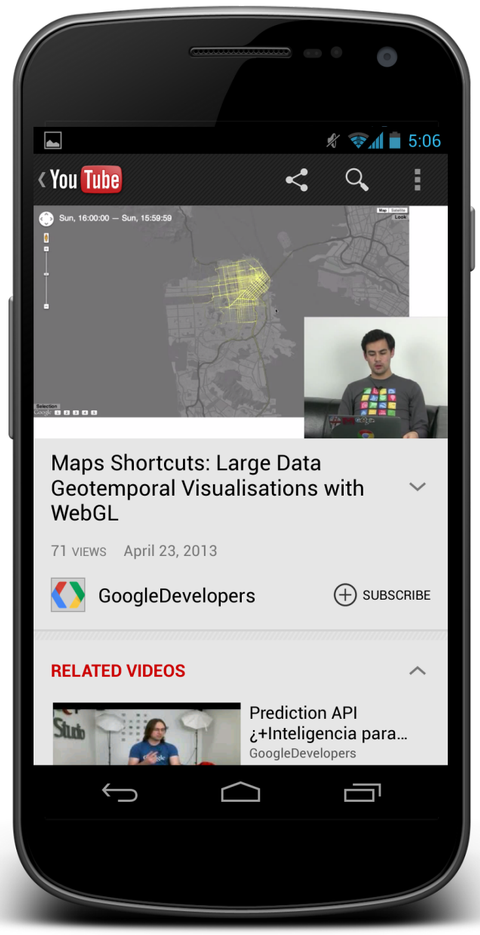
Source: Flipboard (iOS) viewing the Google Developers channel

Android playback options

Custom UI elements to control playback

Android Player API (December 2012)





Official YouTube app

Source: Flipboard (Android) viewing the Google Developers channel

Defining a YouTube view

XML layout

XML

```
<com.google.android.youtube.player.YouTubePlayerView
            android:id="@+id/youtube_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
```

Defining a YouTube view

Java Activity

Java

```
public class VideoPlayerActivity extends YouTubeBaseActivity
  implements
    YouTubePlayer.OnInitializedListener {
     // code goes here ...
```

Best practice: Use the YouTube Android Player API over other options when possible.

YouTube Android Player API: your video toolkit

- Check phone capabilities
- Manage Intents
- Video playback directly in your view
- Playback events callbacks

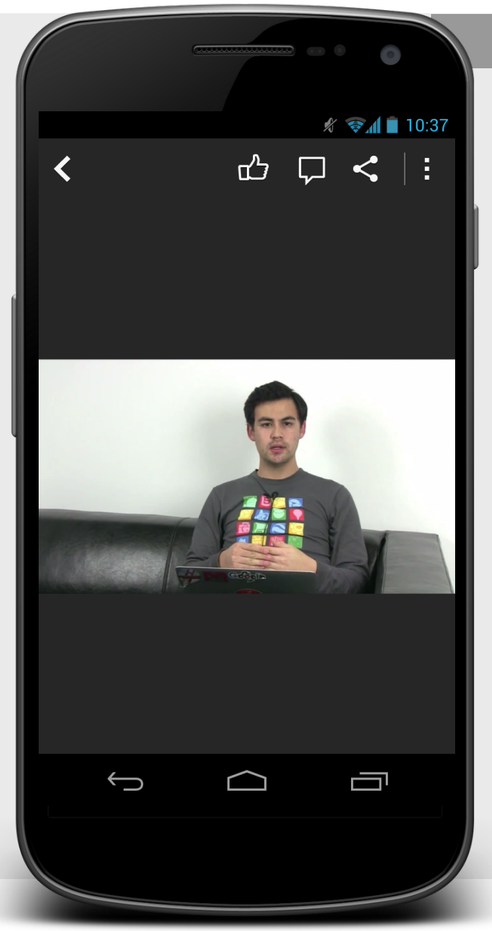
https://developers.google.com/youtube/android/player

Best practice: Gracefully degrade to the best player experience on the platform you're on.

In a perfect world ...

Java

```
// Starts the player while inside your app
YouTubeStandalonePlayer.createVideoIntent(
                this,
                DeveloperKey.DEVELOPER_KEY,
                VIDEO_ID,
                startTimeMillis,
                autoplay,
                lightboxMode);
```



Gracefully falling to the next best option

Best

Java

```
// Starts the player while inside your app
YouTubeStandalonePlayer.createVideoIntent(
                this, DeveloperKey.DEVELOPER_KEY, VIDEO_ID, startTimeMillis, autoplay, lightboxMode);
```

Okay

Java

```
// Starts YouTube app
intent = YouTubeIntents.createPlayVideoIntentWithOptions(this, VIDEO_ID, true, false);
startActivity(intent);
```

Last resort

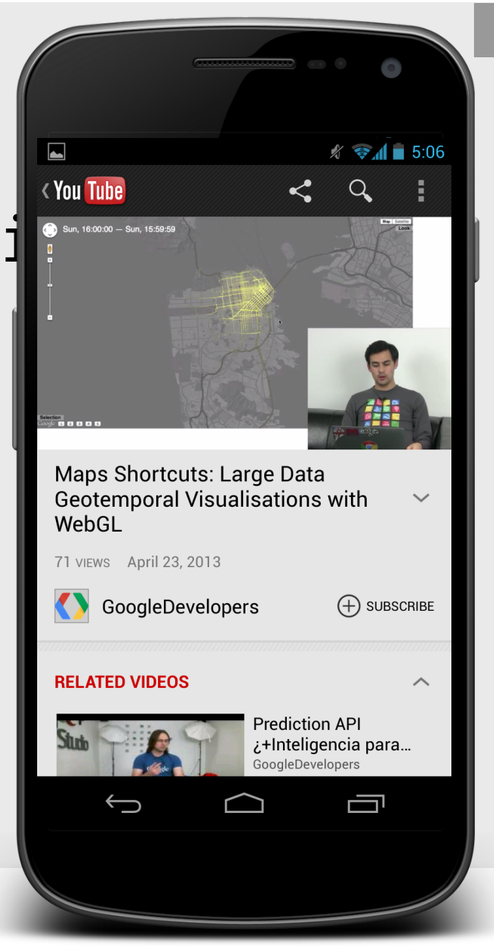
Java

```
// Use a WebView and the iframe API
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.getSettings().setJavaScriptEnabled(true);
myWebView.loadUrl("file:///android_asset/youtube_container.html");
```

Starting the YouTube app

Java

```
// Starts YouTube app
YouTubeIntents.createPlayVideoIntentWithOpti
        this,
        VIDEO_ID,
        true,
        false);
```



Last resort - optional

Java

```
// Use a WebView and the iframe API
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.getSettings().setJavaScriptEnabled(true);
myWebView.loadUrl("file:///android_asset/youtube_container.html");
```

Graceful failover

(We'll go line by line in the next few slides)

Java

```
if(YouTubeIntents.isYouTubeInstalled(context)) {
    if(YouTubeApiServiceUtil.isYouTubeApiServiceAvailable(context) == YouTubeInitializationResult.SUCCESS) {
        // start the YouTube player
        context.startActivity(
            YouTubeStandalonePlayer.createVideoIntent((Activity) context, "developer_key", videoId));
    } else if(YouTubeIntents.canResolvePlayVideoIntent(context)) {
        // Start an intent to the YouTube app
        context.startActivity(
            YouTubeIntents.createPlayVideoIntent(context, videoId));
    }
}
// Falls through: last resort - render a webview with an iframe
```

Is YouTube installed?

Java

```
if(YouTubeIntents.isYouTubeInstalled(context)) {
    if(YouTubeApiServiceUtil.isYouTubeApiServiceAvailable(context)
       == YouTubeInitializationResult.SUCCESS) {
        // start the YouTube player
        context.startActivity(
            YouTubeStandalonePlayer.createVideoIntent(
                context, "developer_key", videoId));
```

Is the Android Player API available?

Java

```
if(YouTubeIntents.isYouTubeInstalled(context)) {
    if(YouTubeApiServiceUtil.isYouTubeApiServiceAvailable(context)
        == YouTubeInitializationResult.SUCCESS) {
        // start the YouTube player
        context.startActivity(
            YouTubeStandalonePlayer.createVideoIntent(
                context, "developer_key", videoId));
```

Start the standalone player

Can also be used to start a player in a fragment

Java

```
if(YouTubeIntents.isYouTubeInstalled(context)) {
    if(YouTubeApiServiceUtil.isYouTubeApiServiceAvailable(context)
       == YouTubeInitializationResult.SUCCESS) {
        // start the YouTube player
        context.startActivity(
            YouTubeStandalonePlayer.createVideoIntent(
                context, "developer_key", videoId));
```

Can the video be played in the official app?

Java

```
} else if(YouTubeIntents.canResolvePlayVideoIntent(context)) {
    // Start an intent to the YouTube app
    context.startActivity(
        YouTubeIntents.createPlayVideoIntent(context, videoId));
}
```

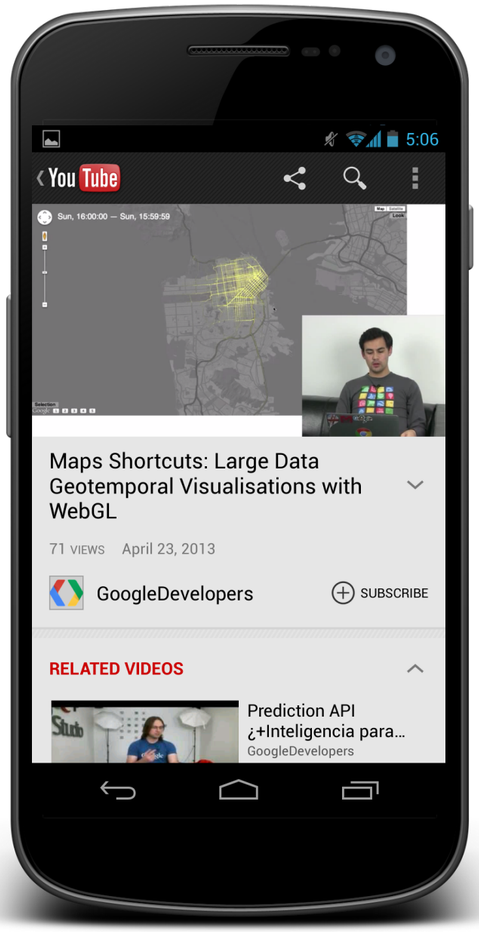
Start an Intent to play it in the app

Java

```
} else if(YouTubeIntents.canResolvePlayVideoIntent(context)) {
    // Start an intent to the YouTube app
    context.startActivity(
        YouTubeIntents.createPlayVideoIntent(context, videoId));
}
```

When should the official Android app be used?

- Full YouTube functionality
  - Subscriptions
  - History
  - Playlists
  - **Rebuilding the YouTube app is a discouraged anti-pattern!**
- Player API or WebView (last resort) - keeping users inside app
- Analogy: Maps application



Tip: Don't rebuild the YouTube app

- Focus on your own app features
- Some overlap is okay - your value add?
- **Use your judgment!**

Best practice: Provide an intermediary screen during loading.

Case Study: Skimble

- One of our earliest partners, demo'd at Google I/O 2012

- Get into shape on the go!

Reminder: Overlays on top of a video will stop video playback.

Case Study: Skimble

Loading ...

Show an app specific intermediary state

Loaded!

Hide message - player does not allow overlays while video is playing

Player callbacks

Determining state, providing a responsive user experience

Java

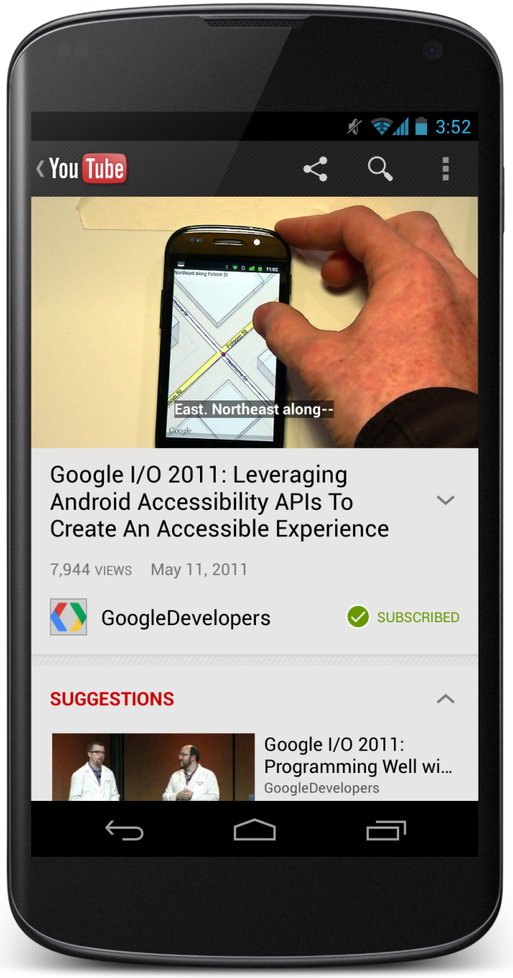
```
@Override
public void onInitializationSuccess(YouTubePlayer.Provider provider,
                                    YouTubePlayer player, boolean wasRestored) {
    // When the player has been initialized, set an event listener for playback events
    this.player = player;
    player.setPlaybackEventListener(playbackEventListener);
}

@Override
public void onPlaying() {
    overlayView.setVisibility(View.INVISIBLE);
}
```

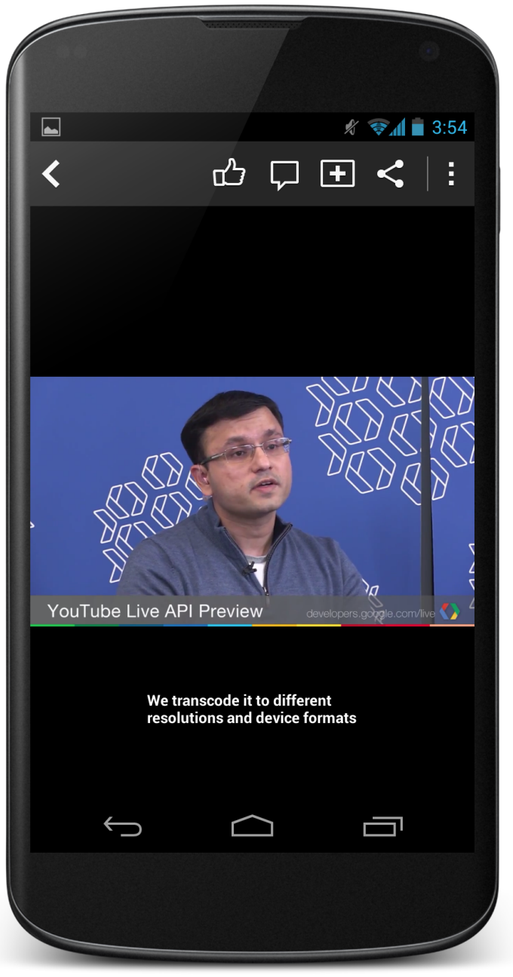
List ALL the callbacks!

- Player changes to fullscreen
- Buffering
- Paused
- Playing
- Seeks to time
- Stopped
- Started ad
- Loading/Loaded
- Ended video
- Started video
- Thumbnail loaded
- Video changes on playlist playback
- **Error**!

One more thing: Captions

- Official app and Player API do support this
- No need to create your own solution!





Best Practices

Authorization

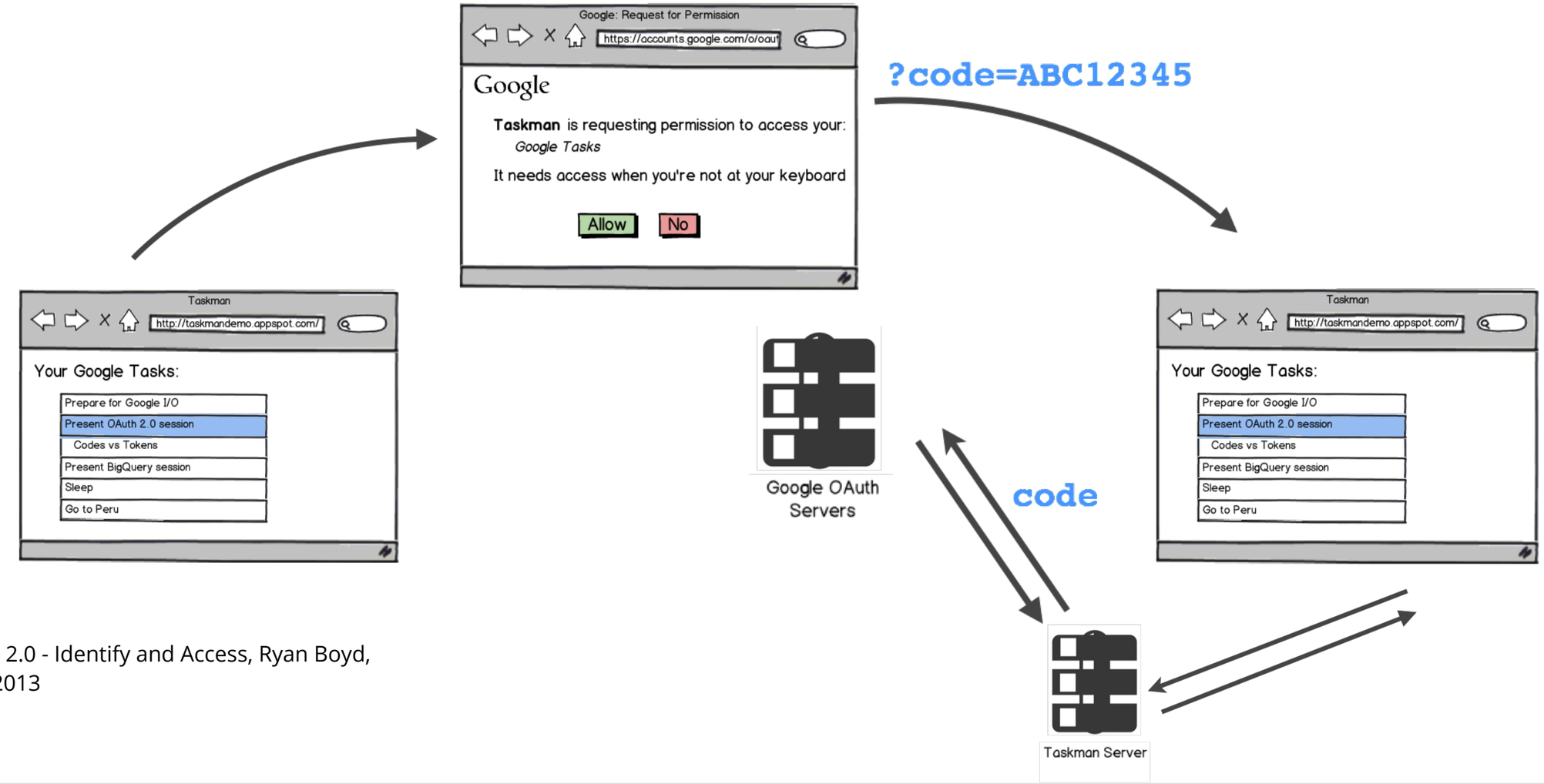
Public Service Announcement

- If you are using the Data API, please migrate to v3
- Don't use ClientLogin

API call requirements

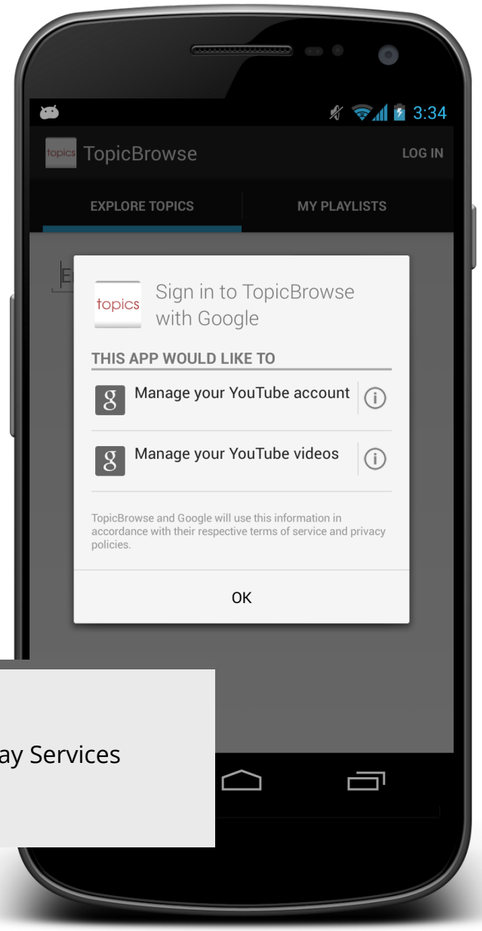
API call	Requirements
Search for videos	App needs a developer key
Get data about a video	App needs a developer key
Android Player API	App needs a developer key
Upload and manage videos	Needs OAuth and YouTube channel
Create and manage playlists	Needs OAuth and YouTube channel

OAuth for the web



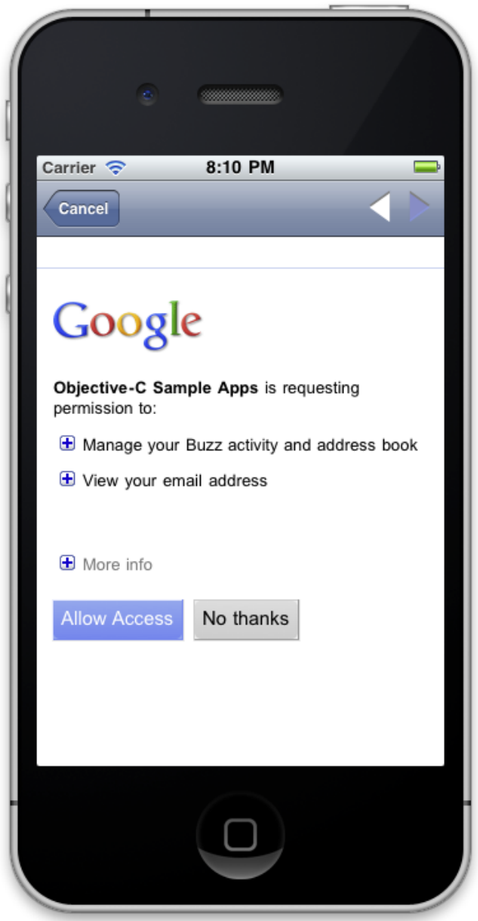
Source: OAuth 2.0 - Identify and Access, Ryan Boyd, SXSW, March 2013

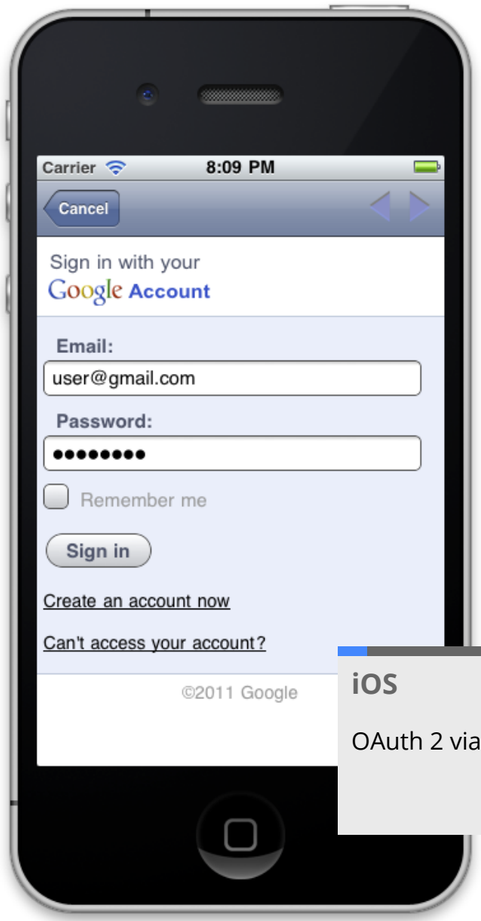
OAuth 2 for phones/tablets



Android

Authorization via Play Services

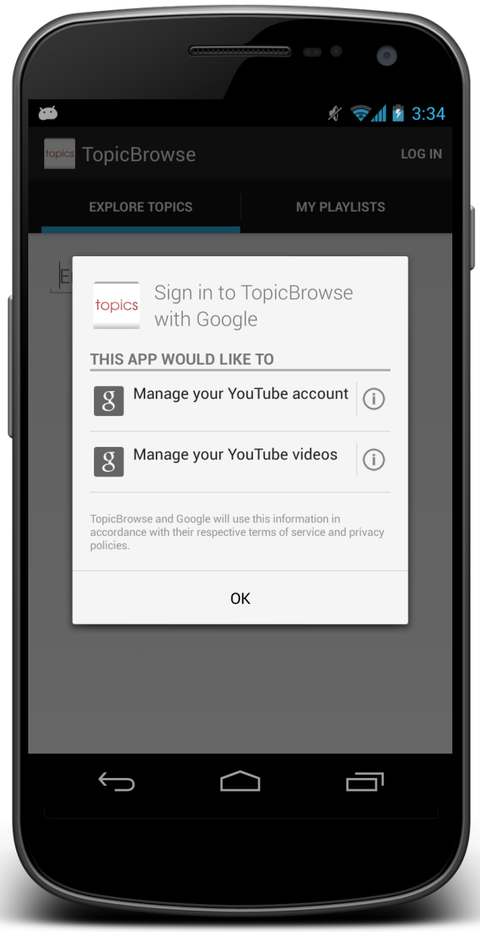


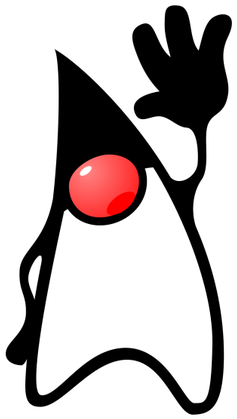


iOS

OAuth 2 via a WebView

Play Services for Android





- Native authorization dialog
- Token management
- Works seamlessly with YouTube v3 API client

Authorizing a channel

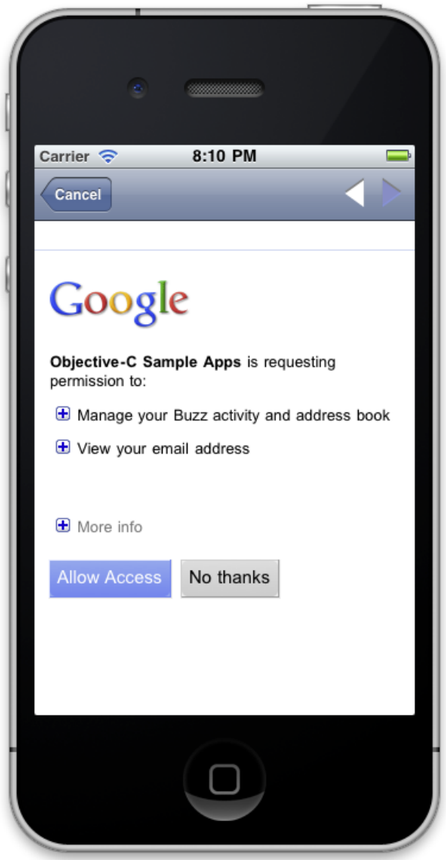
GoogleAuthUtil in action

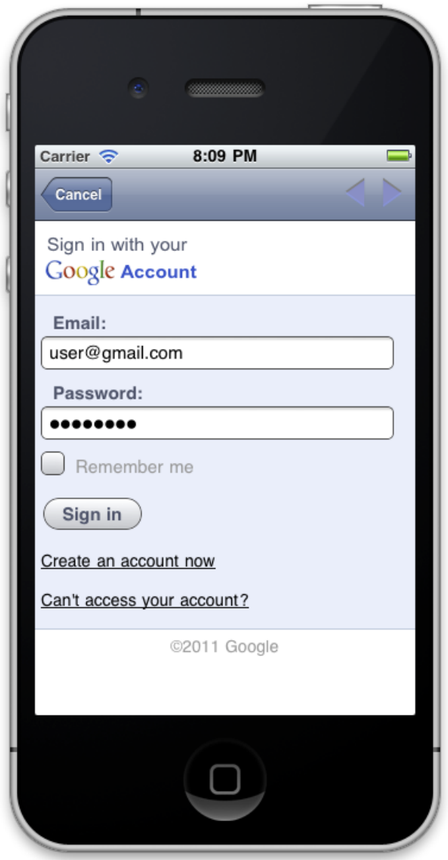
Java

```
public static final int REQUEST_USER_AUTHORIZATION = 1;

try {
    String authToken = GoogleAuthUtil.getToken(mContext,
            accountName, "oauth2:" + YouTubeScopes.YOUTUBE + " "
            + YouTubeAnalyticsScopes.YT_ANALYTICS_READONLY);
    // Make API call with the authToken!
} catch (UserRecoverableAuthException e) {
    // User needs to authorize this application, or take some other action
    Intent authIntent = e.getIntent();
    ((Activity) mContext).startActivityForResult(authIntent, REQUEST_USER_AUTHORIZATION);
}
```

Authorization on iOS

- Google+ Platform for iOS
  https://developers.google.com/+/mobile/ios
- gtm-oauth2
  https://code.google.com/p/gtm-oauth2





gtm-oauth2 in action

Objective-C

```
#import "GTMOAuth2ViewControllerTouch.h"
static NSString *const kKeychainItemName = @"OAuth2 Sample: YouTube API";
NSString *kMyClientID = @"abcd";     // pre-assigned by service
NSString *kMyClientSecret = @"efgh"; // pre-assigned by service
NSString *scope = @"https://www.googleapis.com/auth/youtube"; // scope for YouTube API
GTMOAuth2ViewControllerTouch *viewController;
viewController = [[[GTMOAuth2ViewControllerTouch alloc] initWithScope:scope
             clientID:kMyClientID
         clientSecret:kMyClientSecret
     keychainItemName:kKeychainItemName
             delegate:self
     finishedSelector:@selector(viewController:finishedWithAuth:error:)] autorelease];

[[self navigationController] pushViewController:controller
                                       animated:YES];
```

Tip: Help the user create a channel if they do not already have a YouTube channel.

YouTube channels and Google accounts

- A Google Account may exist without a YouTube channel
- Channels are required for
  - Playlist management
  - Video uploads

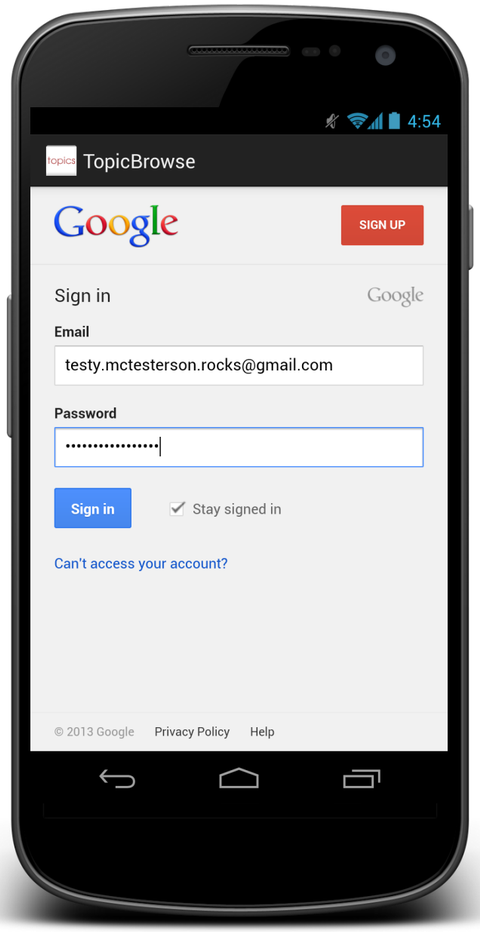
- **You may need to help the user create a channel!**

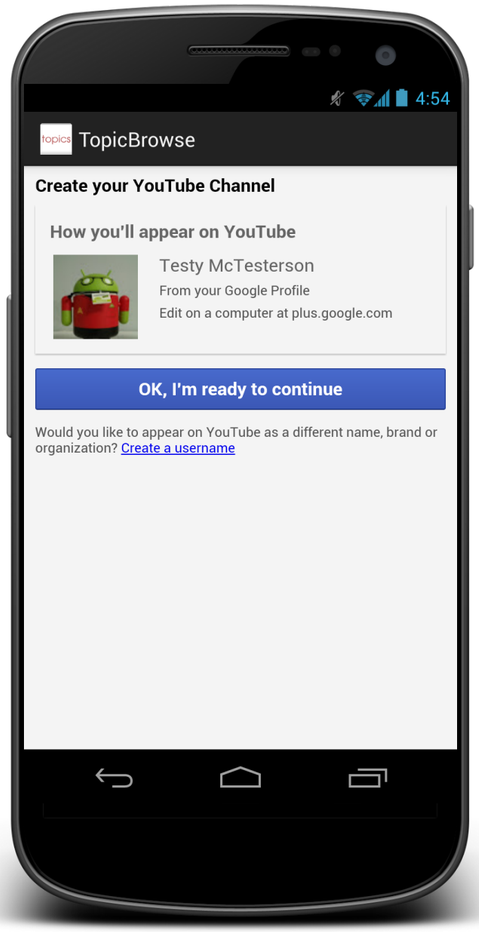
Creating a YouTube channel

YouTube API

HTTP 401 thrown with reason == youtubeSignupRequired

Start Activity





Does the user have a YouTube channel?

Lazy checking

Java

```
try {
    yt.playlistItems().insert("snippet,contentDetails", playlistItem).execute();
} catch (GoogleJsonResponseException e) {
    GoogleJsonError error = e.getDetails();
    for(GoogleJsonError.ErrorInfo errorInfo : error.getErrors()) {
        if(errorInfo.getReason().equals("youtubeSignupRequired")) {
            // Start an activity to create a channel
            Intent createChannelActivity = new Intent(VideoPlayerActivity.this,
CreateChannelActivity.class);
            startActivity(createChannelActivity);
        }
     }
}
```

Make the API call

Java

```
try {
    yt.playlistItems().insert("snippet,contentDetails", playlistItem).execute();
} catch (GoogleJsonResponseException e) {
    GoogleJsonError error = e.getDetails();
    for(GoogleJsonError.ErrorInfo errorInfo : error.getErrors()) {
        if(errorInfo.getReason().equals("youtubeSignupRequired")) {
            // Start an activity to create a channel
            Intent createChannelActivity = new Intent(VideoPlayerActivity.this,
CreateChannelActivity.class);
            startActivity(createChannelActivity);
        }
     }
}
```

Oops! GoogleJsonResponseException thrown!

Java

```
try {
    yt.playlistItems().insert("snippet,contentDetails", playlistItem).execute();
} catch (GoogleJsonResponseException e) {
    GoogleJsonError error = e.getDetails();
    for(GoogleJsonError.ErrorInfo errorInfo : error.getErrors()) {
        if(errorInfo.getReason().equals("youtubeSignupRequired")) {
            // Start an activity to create a channel
            Intent createChannelActivity = new Intent(VideoPlayerActivity.this,
CreateChannelActivity.class);
            startActivity(createChannelActivity);
        }
     }
}
```

Iterate through the errors

Java

```
try {
    yt.playlistItems().insert("snippet,contentDetails", playlistItem).execute();
} catch (GoogleJsonResponseException e) {
    GoogleJsonError error = e.getDetails();
    for(GoogleJsonError.ErrorInfo errorInfo : error.getErrors()) {
        if(errorInfo.getReason().equals("youtubeSignupRequired")) {
            // Start an activity to create a channel
            Intent createChannelActivity = new Intent(VideoPlayerActivity.this,
CreateChannelActivity.class);
            startActivity(createChannelActivity);
        }
     }
}
```

Check for "youtubeSignupRequired"

Java

```
try {
    yt.playlistItems().insert("snippet,contentDetails", playlistItem).execute();
} catch (GoogleJsonResponseException e) {
    GoogleJsonError error = e.getDetails();
    for(GoogleJsonError.ErrorInfo errorInfo : error.getErrors()) {
        if(errorInfo.getReason().equals("youtubeSignupRequired")) {
            // Start an activity to create a channel
            Intent createChannelActivity = new Intent(VideoPlayerActivity.this,
CreateChannelActivity.class);
            startActivity(createChannelActivity);
        }
     }
}
```

Direct the user to creating a channel

Java

```
try {
    yt.playlistItems().insert("snippet,contentDetails", playlistItem).execute();
} catch (GoogleJsonResponseException e) {
    GoogleJsonError error = e.getDetails();
    for(GoogleJsonError.ErrorInfo errorInfo : error.getErrors()) {
        if(errorInfo.getReason().equals("youtubeSignupRequired")) {
            // Start an activity to create a channel
            Intent createChannelActivity = new Intent(VideoPlayerActivity.this,
CreateChannelActivity.class);
            startActivity(createChannelActivity);
        }
     }
}
```

Also: It's possible to check for the presence of a channel ahead of time.

Checking ahead of time for a channel

We'll quickly step through this code

Java

```
GoogleCredential credential = new GoogleCredential();
credential.setAccessToken(authToken);
YouTube youtube = new YouTube.Builder(new NetHttpTransport(), new JacksonFactory(),
        credential).setApplicationName("YouTubeMobileCodeLab/0.1")
        .build();

YouTube.Channels.List channelRequest = youtube.channels().list("status");
channelRequest.setMine("true");
channelRequest.setFields("items/status");
ChannelListResponse channelResult = channelRequest.execute();
List<Channel> channelsList = channelResult.getItems();
for (Channel channel : channelsList) {
    Map<String, Object> status = (Map<String, Object>) channel.get("status");
    if (true == (Boolean) status.get("isLinked")) {
        // Channel is linked to a Google Account
    } else {
        // Channel is NOT linked to a Google Account
    }
}
```

Create a GoogleCredential instance

Java

```
GoogleCredential credential = new GoogleCredential();
credential.setAccessToken(authToken);
YouTube youtube = new YouTube.Builder(new NetHttpTransport(), new JacksonFactory(),
        credential).setApplicationName("YouTubeMobileCodeLab/0.1")
        .build();

YouTube.Channels.List channelRequest = youtube.channels().list("status");
channelRequest.setMine("true");
channelRequest.setFields("items/status");
ChannelListResponse channelResult = channelRequest.execute();
```

Remember me? Put the auth token to use!

Java

```
GoogleCredential credential = new GoogleCredential();
credential.setAccessToken(authToken);
YouTube youtube = new YouTube.Builder(new NetHttpTransport(), new JacksonFactory(),
        credential).setApplicationName("YouTubeMobileCodeLab/0.1")
        .build();

YouTube.Channels.List channelRequest = youtube.channels().list("status");
channelRequest.setMine("true");
channelRequest.setFields("items/status");
ChannelListResponse channelResult = channelRequest.execute();
```

Create an instance of the YouTube API client

Java

```
GoogleCredential credential = new GoogleCredential();
credential.setAccessToken(authToken);
YouTube youtube = new YouTube.Builder(new NetHttpTransport(), new JacksonFactory(),
        credential).setApplicationName("YouTubeMobileCodeLab/0.1")
        .build();

YouTube.Channels.List channelRequest = youtube.channels().list("status");
channelRequest.setMine("true");
channelRequest.setFields("items/status");
ChannelListResponse channelResult = channelRequest.execute();
```

Configure and make the API call

Java

```
GoogleCredential credential = new GoogleCredential();
credential.setAccessToken(authToken);
YouTube youtube = new YouTube.Builder(new NetHttpTransport(), new JacksonFactory(),
        credential).setApplicationName("YouTubeMobileCodeLab/0.1")
        .build();

YouTube.Channels.List channelRequest = youtube.channels().list("status");
channelRequest.setMine("true");
channelRequest.setFields("items/status");
ChannelListResponse channelResult = channelRequest.execute();
```

Iterate through API call result info

Java

```
List<Channel> channelsList = channelResult.getItems();
for (Channel channel : channelsList) {
    Map<String, Object> status = (Map<String, Object>) channel.get("status");
    if (true == (Boolean) status.get("isLinked")) {
        // Channel is linked to a Google Account
    } else {
        // Channel is NOT linked to a Google Account
    }
}
```

Get the status of the API call

Java

```java
List<Channel> channelsList = channelResult.getItems();
for (Channel channel : channelsList) {
    Map<String, Object> status = (Map<String, Object>) channel.get("status");
    if (true == (Boolean) status.get("isLinked")) {
        // Channel is linked to a Google Account
    } else {
        // Channel is NOT linked to a Google Account
    }
}
```

Check to see if the channel is linked

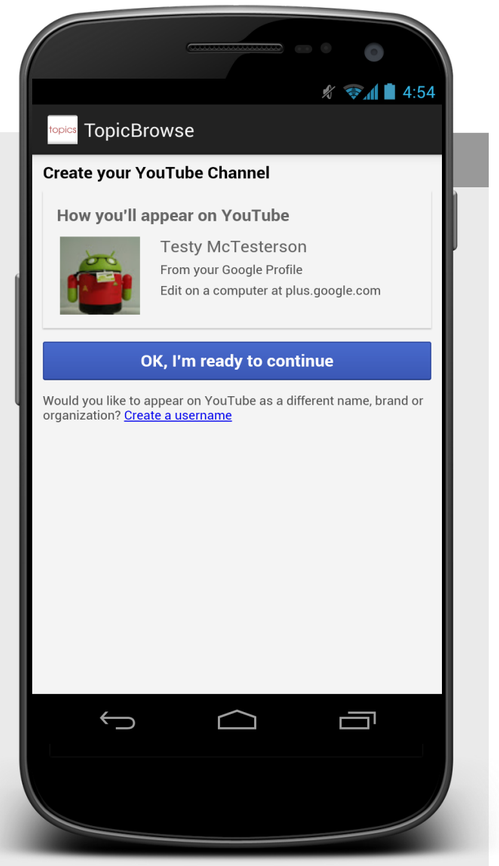
Java

```
List<Channel> channelsList = channelResult.getItems();
for (Channel channel : channelsList) {
    Map<String, Object> status = (Map<String, Object>) channel.get("status");
    if (true == (Boolean) status.get("isLinked")) {
        // Channel is linked to a Google Account
    } else {
        // Channel is NOT linked to a Google Account
    }
}
```

Possible solution: WebView to create channel page

Keeps user within app, but user has to re-type in account info

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.getSettings().setJavaScriptEnabled(true);
myWebView.setWebViewClient(new WebViewClient() {


    @Override
     public void onPageFinished(WebView view, String url) {
         if (url.startsWith("https://m.youtube.com/#/oops")) {
             // Handle the error
         } else if (url.startsWith("https://m.youtube.com/channel_creation_done")) {
             // Return from this activity, the channel was successfully created
             Intent resultData = new Intent();
             setResult(Activity.RESULT_OK, resultData);
             finish();
         }
     }
});
myWebView.loadUrl("http://m.youtube.com/create_channel?chromeless=1&next=/channel_creation_done"));
```



Recap

- Graceful failover
- Handle loading latency
- Data API: Use v3!
- Authorization
  - Android: Google Play Services
  - iOS: Google+ platform or gtm-oauth2
  - User may need a channel to do certain things; help them create one!

Acknowledgements

- Tom Bridgwater, YouTube
- Geert Weening, Flipboard
- John Tighe, Skimble

<Thank You!>

Ikai Lan

ikai@youtube.com
+IkaiLan
@ikai

Google
Developers