Optimizing Android Apps for Chromebooks

David Gassner / LinkedIn Learning

What I Do

- Managing Staff Instructor,
  LinkedIn Learning Technology Library
- Author of over 100 video-based training courses
- Current focus: Android and languages
  - Java, Kotlin, Go, C#, Visual Basic
- My Big Fat Android App: Audio Cues

My Programming Career

Paradox
PAL & ObjectPAL

Android
Java & Kotlin

Flex
ActionScript

Delphi
Pascal

Netiva
Java

ColdFusion
CFML

Other Things I Do

- Theater maker: producer, director, actor
- Manager of a small performance space in Seattle
- Did I mention Audio Cues?

How We Got Here

2014	2016	2018
First beta of ARC introduced	Play Store available in ChromeOS (beta)	Play Store available in ChromeOS 69 (stable)
Apps downloaded from Chrome store	Limited number of Chromebooks supported:	Android supported on nearly all new Chromebooks
	Acer R11	
	Asus Flip	
	Google Pixel	

Debugging Apps, Step 1: Get a Chromebook

Better yet...get 2 or 3!

Vary by

- CPU (Intel, ARM)
- RAM
- Screen size
- Screen resolution
- With/without touchscreen

Debugging Apps, Step 2: Enable Developer Mode

- You’ll get the most recent new features.
- It might be buggy!
- **You can use ADB for debugging!!**
- Steps to enable developer mode vary

www.chromium.org/chromium-os/developer-information-for-chrome-os-devices

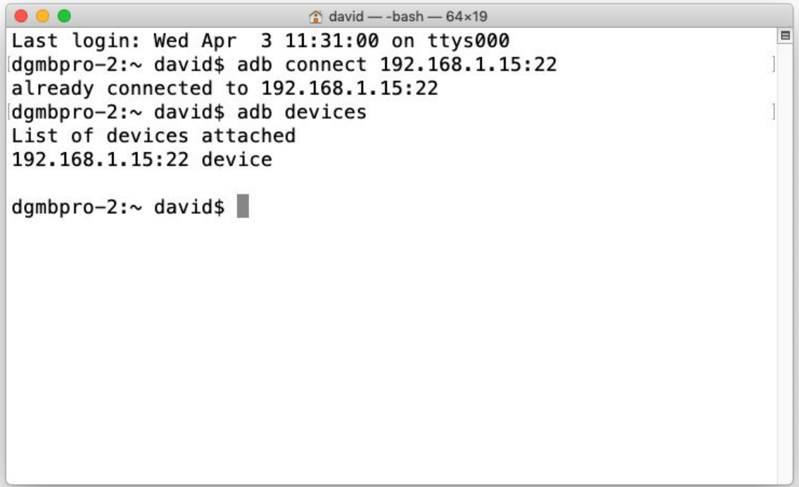
Debugging Apps, Step 2: Enable Developer Mode

- On PixelBook and Pixel Slate:
  - Press Esc + Refresh + power button
  - Boots into Recovery Mode
  - Press Ctrl+D to boot into Developer Mode

Debugging Apps, Step 3: Configure ADB

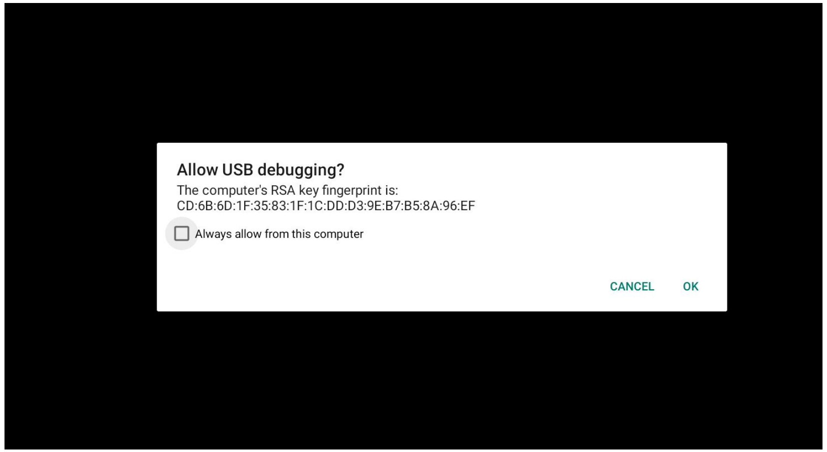
- Chromebooks supporting USB debugging:
  - PixelBook
  - Pixel Slate
  - HP Chromebook X2
- All others: debug over Wi-Fi

developer.android.com/topic/arc/development-environment



```
Last login: Wed Apr  3 11:31:00 on ttys000
dgmbpro-2:~ david$ adb connect 192.168.1.15:22
already connected to 192.168.1.15:22
dgmbpro-2:~ david$ adb devices
List of devices attached
192.168.1.15:22 device

dgmbpro-2:~ david$ 
```



Allow USB debugging?

The computer's RSA key fingerprint is:
CD:6B:6D:1F:35:83:1F:1C:DD:D3:9E:B7:B5:8A:96:EF

☐ Always allow from this computer

On dev computer

On Chromebook

Areas of Optimization

App Manifest
Make your app compatible

Display
Full-screen display
Free-form multi-windowed
Alternate layouts
Alternate bitmap files

Input Devices
Mouse, keyboard, stylus

File System
What's shared with ChromeOS

Support Non-Touchscreen Devices

- If app can work with touchpad or mouse, add to manifest:

```
<uses-feature
    android:name="android.hardware.touchscreen"
    android:required="false" />
```

- Apps require “`faketouch`” feature (mouse, touchpad) by default
- Only allow installation without faketouch if app works with a d-pad or other less common input devices

Require Sensors for Installation

- Sensors that aren’t on most Chromebooks
  - Telephony, NFC, GPS
- Block installation if app doesn’t work without a sensor

```
<uses-feature
    android:name="android.hardware.nfc"
    android:required="true" />
```

- Partial support for other sensors: camera, compass, etc.

Manage Laptop-Style Input Devices

- Add to manifest

```
<uses-feature
    android:name="android.hardware.type.pc"
    android:required="false" />
```

- Lets you develop custom behavior for mouse and touchpad

Chromebook Screen Size and Resolution

- Screen sizes range from 11.6” to very large
- Most common devices have up to 15” screens
- Entry-level screens start at 1366x768
- PixelBook: 2400x1600
- Pixel Slate: 3000x2000
- But remember:

  Chromebooks can be connected to external displays!

Pixel Density: Chromebooks vs. Cell Phones

Device	**Resolution**	**Density**
Lenovo C330	1366x768	**135 PPI**
HP Chromebook 14	1920x1080	**157 PPI**
PixelBook	2400x1600	**235 ppi**
Pixel Slate	3000x2000	**293 ppi**

Device	**Resolution**	**Density**
Nexus 5	1920x1080	**445 PPI**
Pixel 3 XL	2960x1440	**523 PPI**
Galaxy S10	3440x1440	**522 ppi**
Pixel C	2560×1800	**308 ppi**

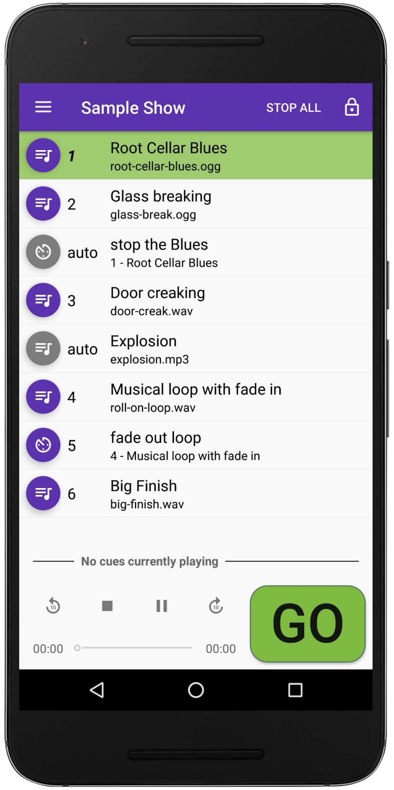
Pretend It’s a Tablet

- Create alternate layouts for different orientations and screen sizes
- Use `VectorDrawable` images when possible
- Create alternate bitmaps for different pixel densities

It Isn’t a Tablet

- Most “true” Android tablets have smaller screens and higher pixel densities
- Tablets display apps with full screen (unless split)
- Depending on alternate resource directories sometimes gets wrong results

False Detection of Landscape Orientation



Sample Show STOP ALL

1 Root Cellar Blues
root-cellar-blues.ogg

2 Glass breaking
glass-break.ogg

auto stop the Blues
1 - Root Cellar Blues

3 Door creaking
door-creak.wav

auto Explosion
explosion.mp3

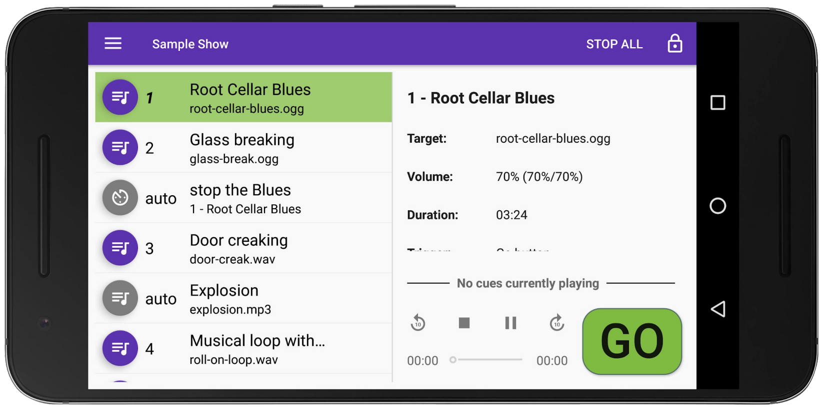
4 Musical loop with fade in
roll-on-loop.wav

5 fade out loop
4 - Musical loop with fade in

6 Big Finish
big-finish.wav

No cues currently playing

00:00 00:00 GO



Sample Show STOP ALL

1 Root Cellar Blues
root-cellar-blues.ogg

2 Glass breaking
glass-break.ogg

auto stop the Blues
1 - Root Cellar Blues

3 Door creaking
door-creak.wav

auto Explosion
explosion.mp3

4 Musical loop with...
roll-on-loop.wav

1 - Root Cellar Blues

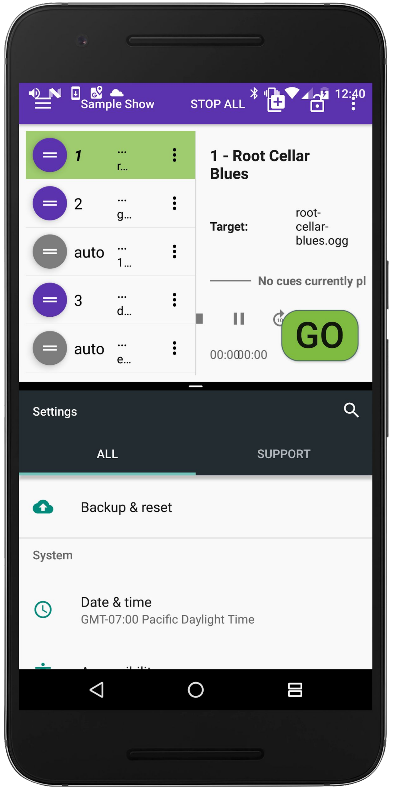
Target: root-cellar-blues.ogg

Volume: 70% (70%/70%)

Duration: 03:24

No cues currently playing

00:00 00:00 GO



Sample Show STOP ALL 12:40

1 ... r...

2 ... g...

auto ... 1...

3 ... d...

auto ... e...

1 - Root Cellar Blues

Target: root-cellar-blues.ogg

No cues currently pl

00:0000:00 GO

Settings

ALL SUPPORT

Backup & reset

System

Date & time
GMT-07:00 Pacific Daylight Time

Step 1: Detect Physical Orientation and Screen Size

```kotlin
var isLandscape = resources.configuration.orientation ==
       Configuration.ORIENTATION_LANDSCAPE

val screenSize = resources.configuration.screenLayout and
       Configuration.SCREENLAYOUT_SIZE_MASK
```

Step 2: Filter Out Phone Devices with Split Display

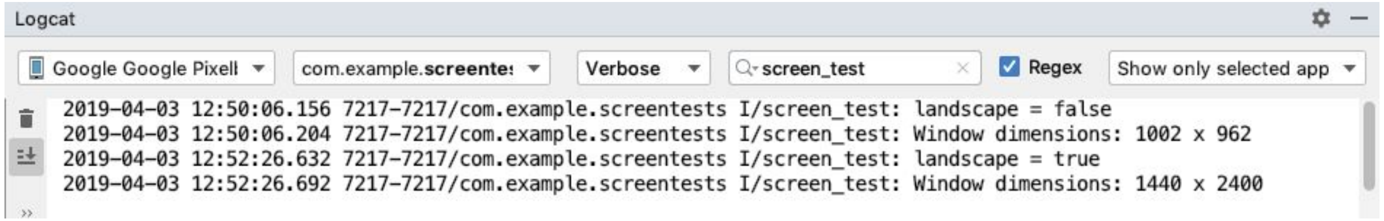
```
if (isLandscape
    && Build.VERSION.SDK_INT >= Build.VERSION_CODES.N
    && isInMultiWindowMode
    && screenSize != Configuration.SCREENLAYOUT_SIZE_LARGE
    && screenSize != Configuration.SCREENLAYOUT_SIZE_XLARGE
) {
    isLandscape = false
}
```

Step 3: Manually Select a Layout File

```
if (isLandscape) {
   setContentView(R.layout.activity_main_landscape)
} else {
   setContentView(R.layout.activity_main)
}
```

Alternate Logic: Measure the Current Layout

```kotlin
val container =
    findViewById<ConstraintLayout?>(R.id.container)
container?.post {
    Log.i(LOG_TAG, "Window dimensions: " +
            "${container.height} x ${container.width}"
    )
}
```



```
2019-04-03 12:50:06.156 7217-7217/com.example.screentests I/screen_test: landscape = false
2019-04-03 12:50:06.204 7217-7217/com.example.screentests I/screen_test: Window dimensions: 1002 x 962
2019-04-03 12:52:26.632 7217-7217/com.example.screentests I/screen_test: landscape = true
2019-04-03 12:52:26.692 7217-7217/com.example.screentests I/screen_test: Window dimensions: 1440 x 2400
```

Aim for Seamless Resizing

- Handle layout changes explicitly for most reliable results
- Handle configuration changes in manifest

  `android:configChanges="screenSize, etc.”`
- Cache data and image files locally to avoid repeated network requests
- Restore state with `onSaveInstanceState()`

Detect Running on a Chromebook

```
val arcDevicePattern = ".+_cheets|cheets_.+"
val isChromeBook =
    Build.DEVICE.matches(arcDevicePattern.toRegex())
```

Handle Mouse and Touchpad Events

- Mouse and touchpad generate `MotionEvent`
- Just like touch events

https://developer.android.com/topic/arc/input-compatibility

Handle Right-Click Mouse Events

- Right click events display *context menus*

```
// Only works on API 23 and later
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
   anyView.setOnContextClickListener {
       Log.i(LOG_TAG, "right click!")
       ... display a popup menu...
       true
   }
}
```

Handle Stylus Events

- Reports events similar to touchscreen with `onTouchEvent()`
- Additional information is sometimes available
    - `MotionEvent.toolType` : distinguish stylus from finger
    - `MotionEvent.pressure` : physical pressure applied to stylus*
    - `MotionEvent.axisValue` : get physical tilt and/or orientation*

* Availability depends on device

Handle Keyboard Events

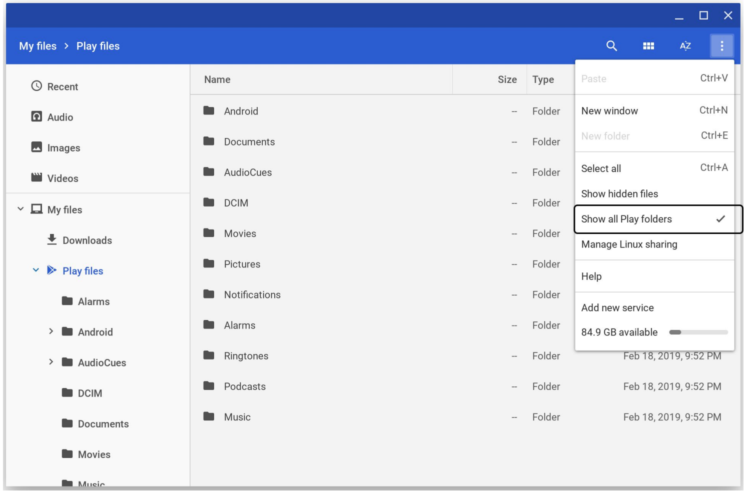
- Like a tablet or phone with Bluetooth keyboard
- Use default functions via `KeyEvent.callback`
- Manage keystrokes manually with `onKeyDown`, `onKeyLongPress`, `onKeyUp`
- Avoid `onKeyPreIME`

Manage Files and Directories

- All Android apps on a device share storage
- Limited access to ChromeOS file system
- Shared storage locations
  - `Downloads` directory
  - External storage (thumb drive, SD card)
- Learn how to use Storage Access Framework (SAF)

Android Files from Chrome Files App (read-only)

- Choose *Show all Play folders* from options menu
- If not appearing, enable flag `show-android-files-in-files-app`



Name	Size	Type	
Android	–	Folder	
Documents	–	Folder	
AudioCues	–	Folder	
DCIM	–	Folder	
Movies	–	Folder	
Pictures	–	Folder	
Notifications	–	Folder	
Alarms	–	Folder	
Ringtones	–	Folder	Feb 18, 2019, 9:52 PM
Podcasts	–	Folder	Feb 18, 2019, 9:52 PM
Music	–	Folder	Feb 18, 2019, 9:52 PM

Market Fragmentation Issues

- Hardware fragmentation
  - Screen, CPU, ports, etc.
- Android version fragmentation
  - Most users upgrade to latest version of ChromeOS
  - Some Chromebooks are on Android 9 Pie; others are still on Android 7 Nougat

Why Bother?

- 10,000,000+ Chromebooks shipped in 2018
- Small fraction of overall Android device market
- Chromebook users may not know what Android is
- But they expect Play Store apps to work!

Bonus Points: Run Android Studio on a Chromebook

- Install Linux
- Download Linux version of Android Studio
- Copy ZIP to `Linux files` in Files app
- In Terminal: unzip package, run `studio.sh`
- Connect ADB to local Android runtime

```
adb connect 100.115.92.2:5555
```

Linked in LEARNING
WITH Lynda.com CONTENT

David Gassner

linkedin.com/in/dgassner