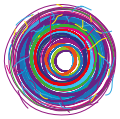
THE AGENCY

WEARABLE INTERNET CHICKEN

Exploring the Android Wear Data Layer API

@ChrisMcKirgan, The Agency

Objective



Introduce android wear's Data Layer API, equipping us with the basic knowledge to craft our own internet enabled wearable applications.

What is Android Wear?

Android on wearable tech

Pretty much just android in a tighter swimsuit

- General APIs are limited
- Limited access to sensors
- No direct http access
- No direct bluetooth access
- No http access ... we'll get onto that

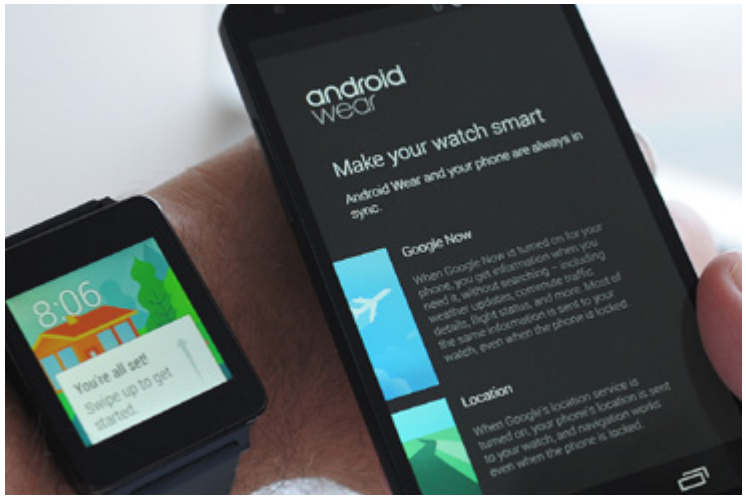
What is the Android Wear API?

Simple, limited range of notifications from your mobile app on your wrist: phone calls, tray notifications, voice input...

Voice input API

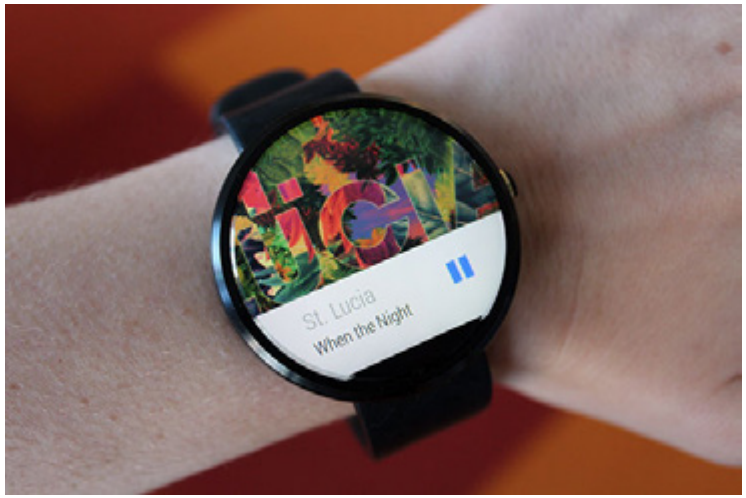
Sensors on wearable; ambient light, step counter, orientation, and the MessageAPI etc.

General development SDK resources



Resources: https://developer.android.com/training/buildingwearables.html

Message API



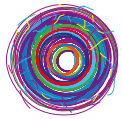
“Fire and forget” API it allows you to send simple data

Useful to do simple things like skip a song track or launch an activity

No receipt of delivery

Resources: https://www.binpress.com/tutorial/aguidetotheandroidwearmessageapi/152

Wearable Data Layer API



Auto sync data anytime between devices

Data sender, data receiver

- Set up at either or both ends

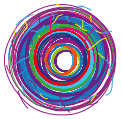
Handles all the complex stuff and gives you beautiful callbacks

Also allows syncing of larger data items such as images

Useful to sync data items between devices, such as fitness data

Resources: https://developer.android.com/training/wearables/datalayer/

Wearable Data Layer API

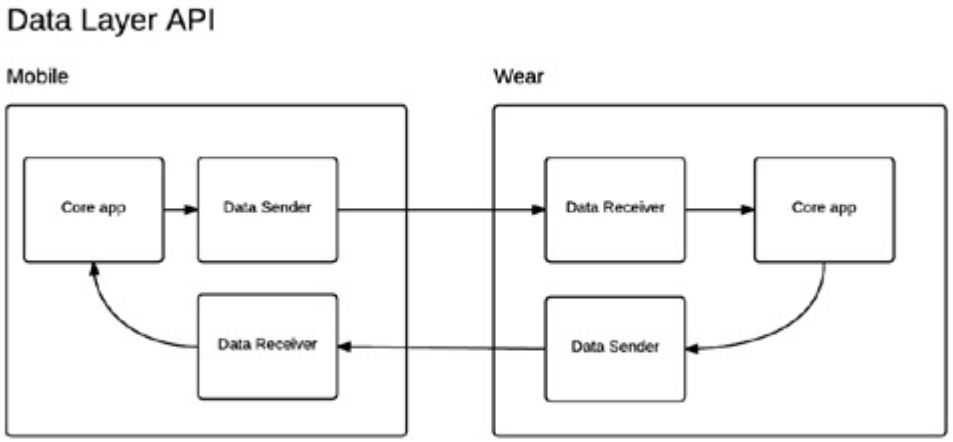


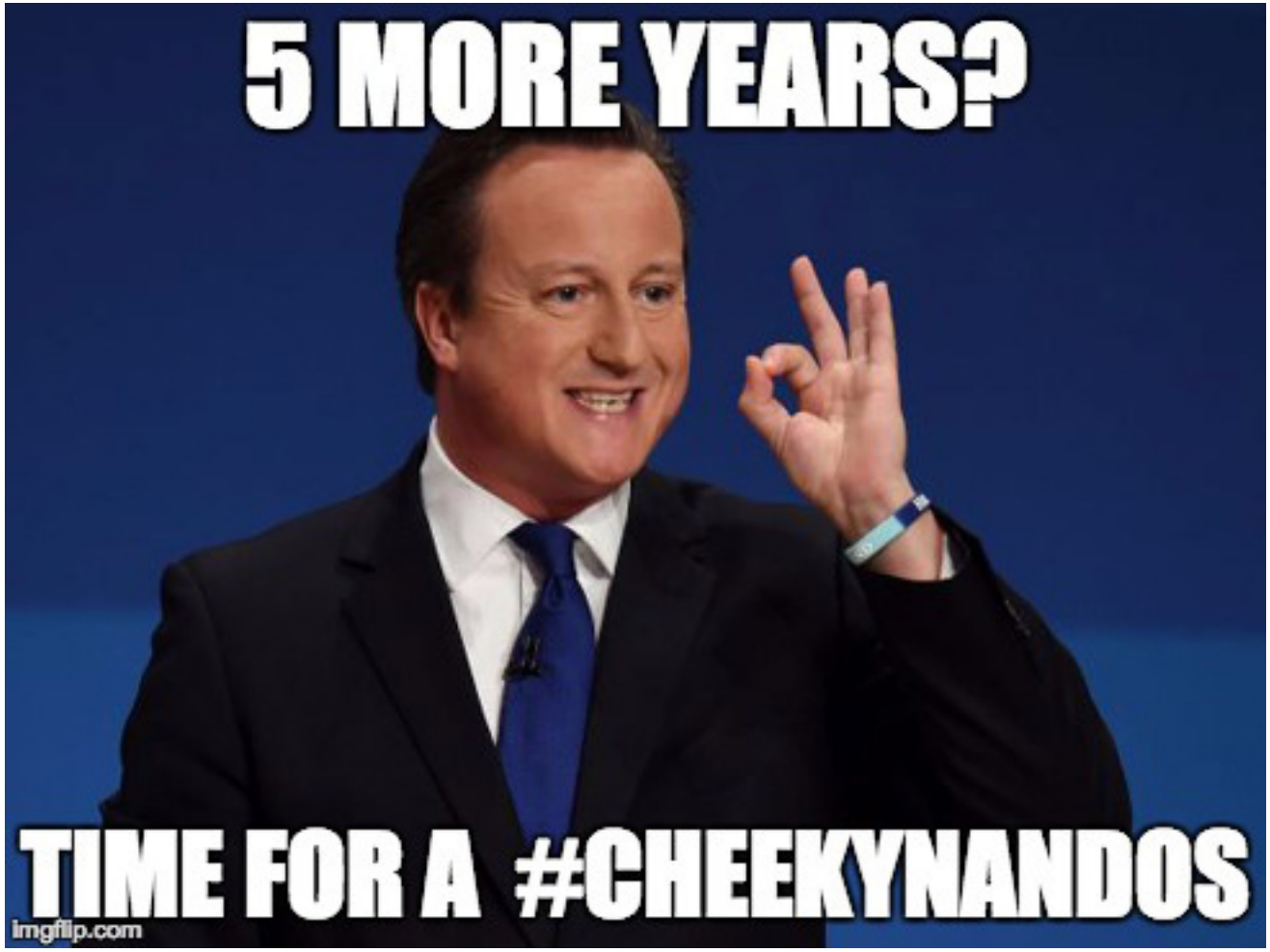
Sender

- Sets up data to be synced
- Sends data to receiver

Reciever

- Listens for changes to data
- Provides callback functions




5 MORE YEARS?
TIME FOR A #CHEEKYNANDOS
imgflip.com

Introducing Fandos

(Nandos concept watch app)

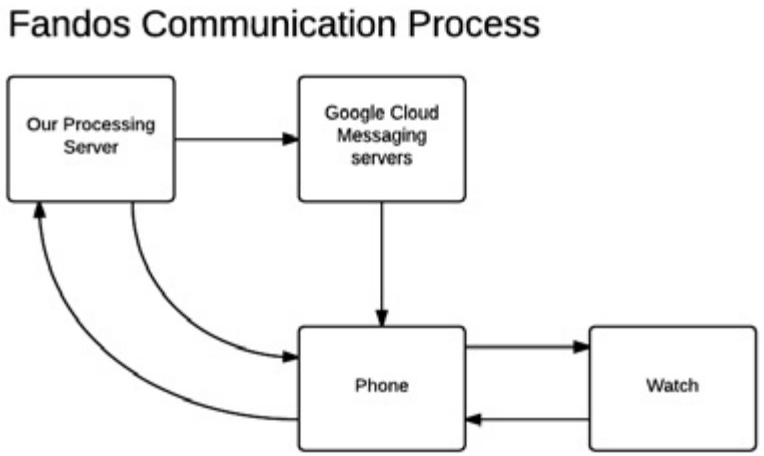
The concept

- Make a purchase in store
- Get notified of reward points on wrist

Fandos

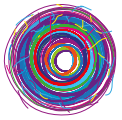
How it works

- Mobile app registers login details & GCM ID with our processing server
- Nandos API hits our processing server
- Processing server passes data onto mobile via GCM
- Mobile passes data to Data Layer API (data sender)
- Mobile data receiver hears event and displays watch app with data



DEMO

Fandos

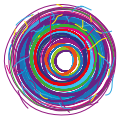


The Code

Sender:

```
104     class SendToDataLayerThread extends Thread {
105         String path;
106         DataMap dataMap;
107
108         // Constructor for sending data objects to the data layer
109         SendToDataLayerThread(String p, DataMap data) {
110             path = p;
111             dataMap = data;
112         }
113
114         public void run() {
115             NodeApi.GetConnectedNodesResult nodes = Wearable.NodeApi.getConnectedNodes(googleClient).await();
116             for (Node node : nodes.getNodes()) {
117
118                 // Construct a DataRequest and send over the data layer
119                 PutDataMapRequest putDMR = PutDataMapRequest.create(path);
120                 putDMR.getDataMap().putAll(dataMap);
121                 PutDataRequest request = putDMR.asPutDataRequest();
122                 DataApi.DataItemResult result = Wearable.DataApi.putDataItem(googleClient,request).await();
123                 if (result.getStatus().isSuccess()) {
124                     Log.v("myTag", "DataMap: " + dataMap + " sent to: " + node.getDisplayName());
125                 } else {
126                     // Log an error
127                     Log.v("myTag", "ERROR: failed to send DataMap");
128                 }
129             }
130         }
131     }
132 }
```

Fandos



The Code

Receiver:

```
15  public class ListenerService extends WearableListenerService {
16      private static final String WEARABLE_DATA_PATH = "/FANDOS_DATA";
17      public final static String EXTRA_MESSAGE = "com.nybblemouse.fandos.MESSAGE";
18
19      @Override
20      public void onDataChanged(DataEventBuffer dataEvents) {
21          Log.v("myTag", "WATCH: DATA RECEIVED");
22          DataMap dataMap;
23          for (DataEvent event : dataEvents) {
24
25              // Check the data type
26              if (event.getType() == DataEvent.TYPE_CHANGED) {
27                  // Check the data path
28                  String path = event.getDataItem().getUri().getPath();
29                  if (path.equals(WEARABLE_DATA_PATH)) {}
30                  dataMap = DataMapItem.fromDataItem(event.getDataItem()).getDataMap();
31                  Log.v("myTag", "DataMap received on watch: " + dataMap);
32
33                  Intent intent = new Intent(this, MainActivity.class);
34                  intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
35                  intent.putExtra(EXTRA_MESSAGE, dataMap.getString("chillis"));
36                  startActivity(intent);
37              }
38          }
39      }
40  }
41
```

THANK YOU

Do you have any questions?

Design by Nick Clarkson @t_pk

Code on github.com/kirgy/Fandos