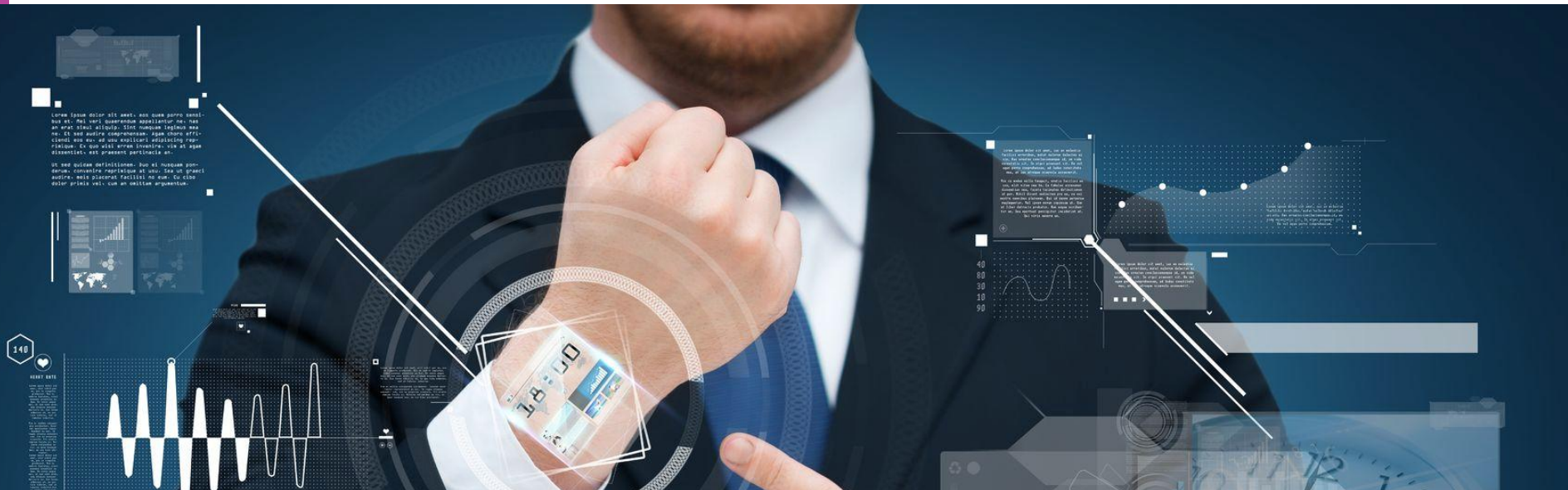
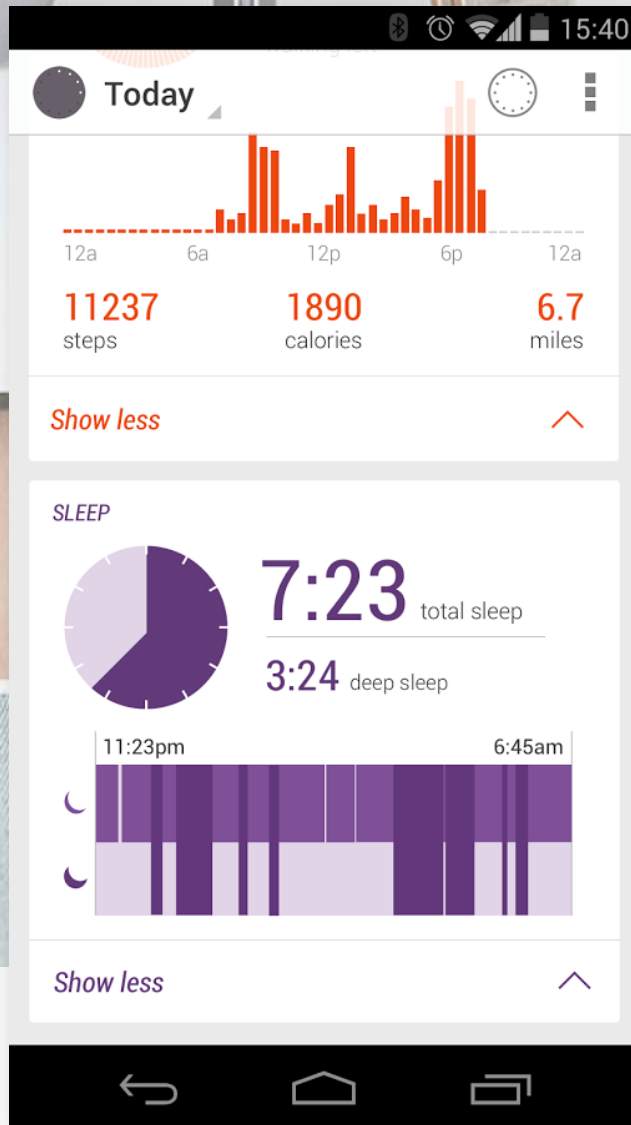


Writing for wearables

Get going with **Android Wear**





Misfit Shine

Wearable Activity Monitor

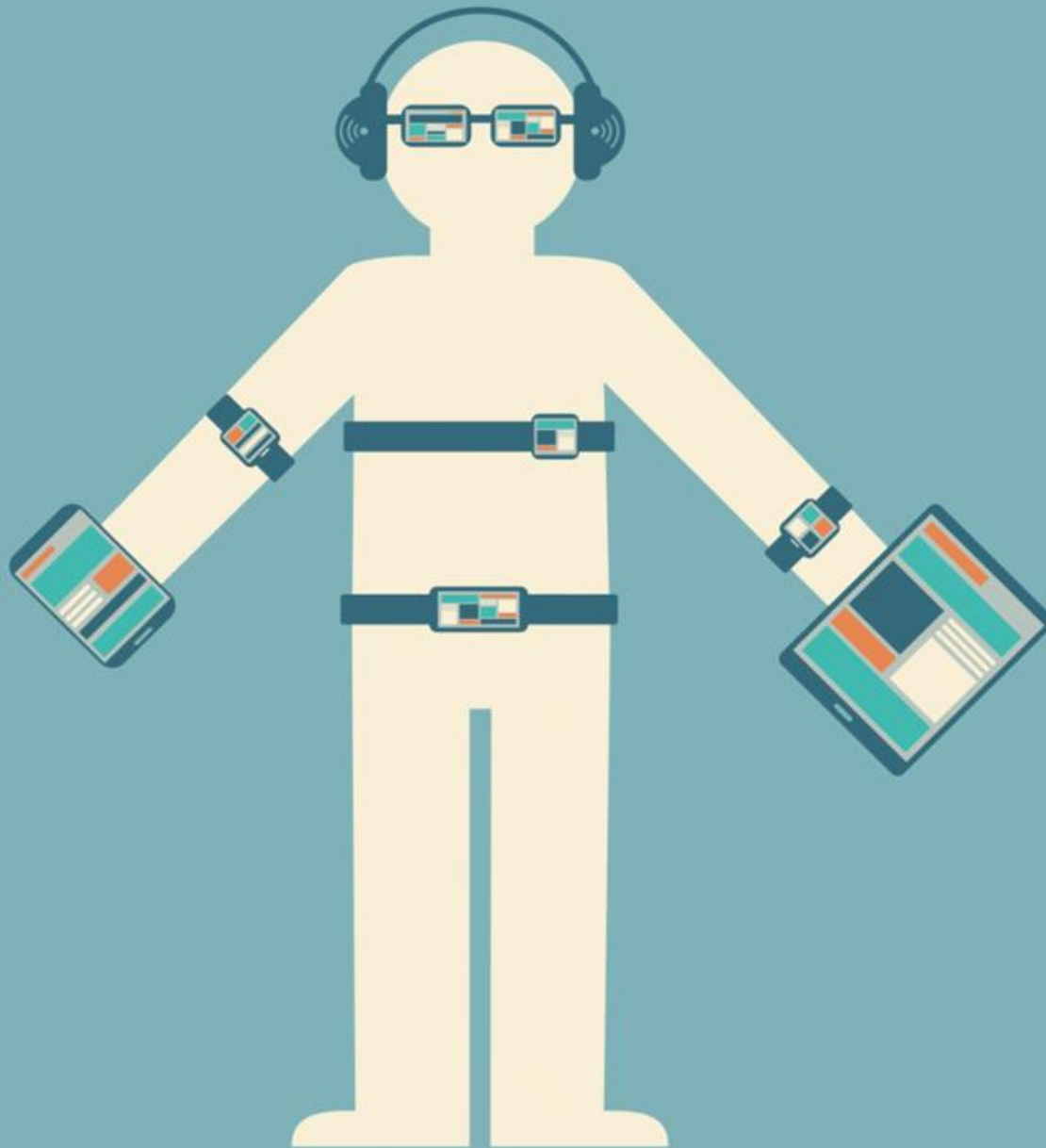
inmeta





GLASS



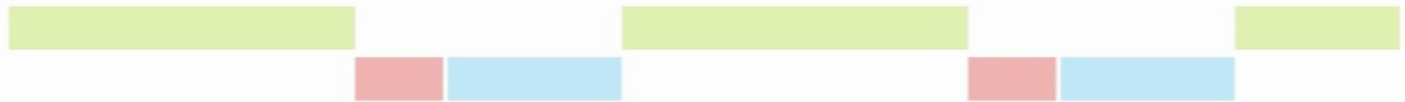


THE SMART WATCH

Phone usage pattern



The promise and value proposition of smart watches





inmeta



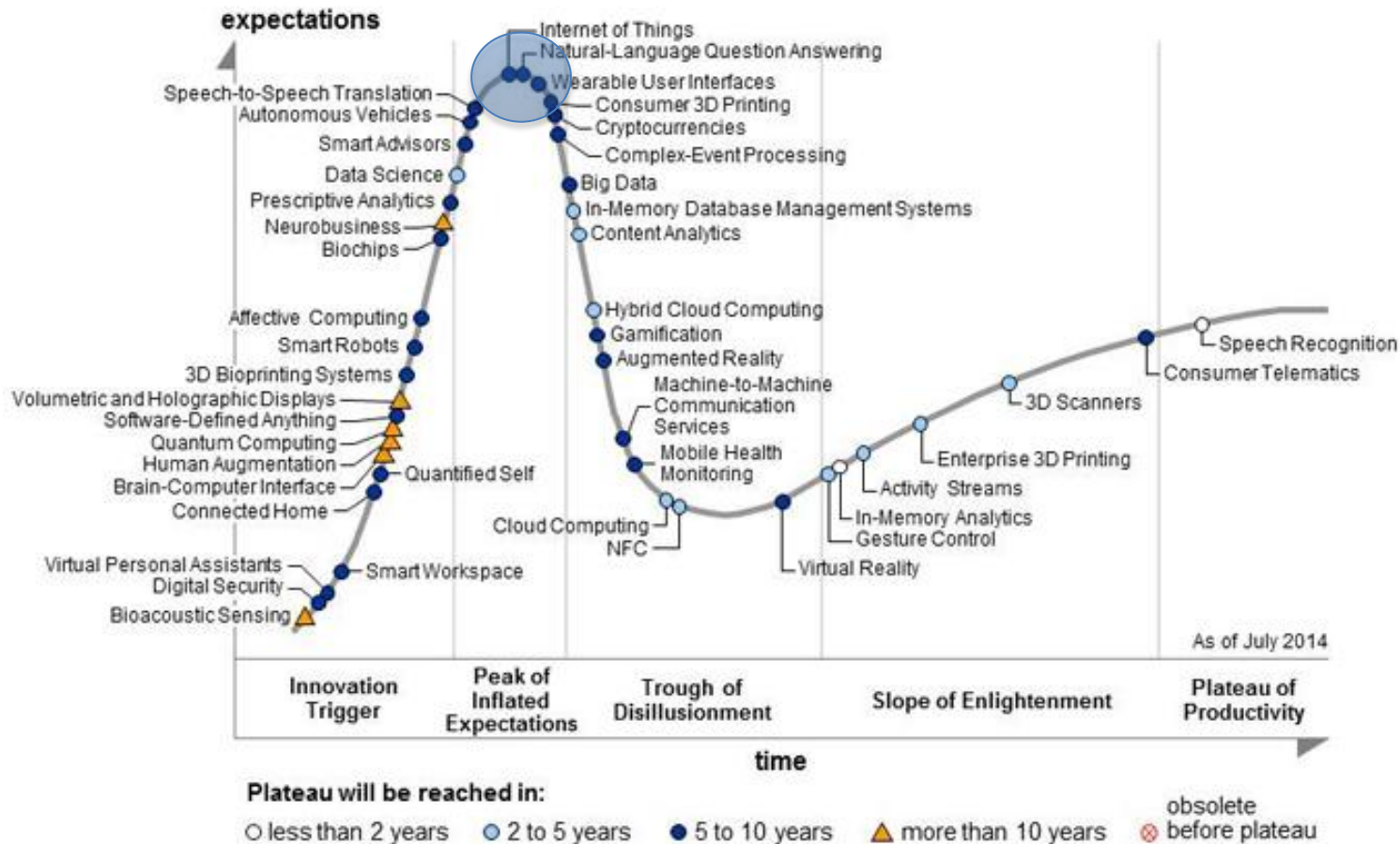
"It keeps me from looking at my phone every two seconds."

A typical android wear “smart” watch

Moto 360 specifications

Chipset	Texas Instruments OMAP 3
Display	1.56-inch Backlit LCD IPS (320 x 290)
Memory	4GB eMMC / 512MB RAM
Battery	320mAh
Operating System	Android Wear (compatible with smartphones running Android 4.3 and above)
Size	46mm diameter
Weight	49g
Connectivity	Bluetooth 4.0 LE
Sensors	9-Axis (Gyro / Accelerometer / Compass), Pedometer, Optical heart-rate monitor

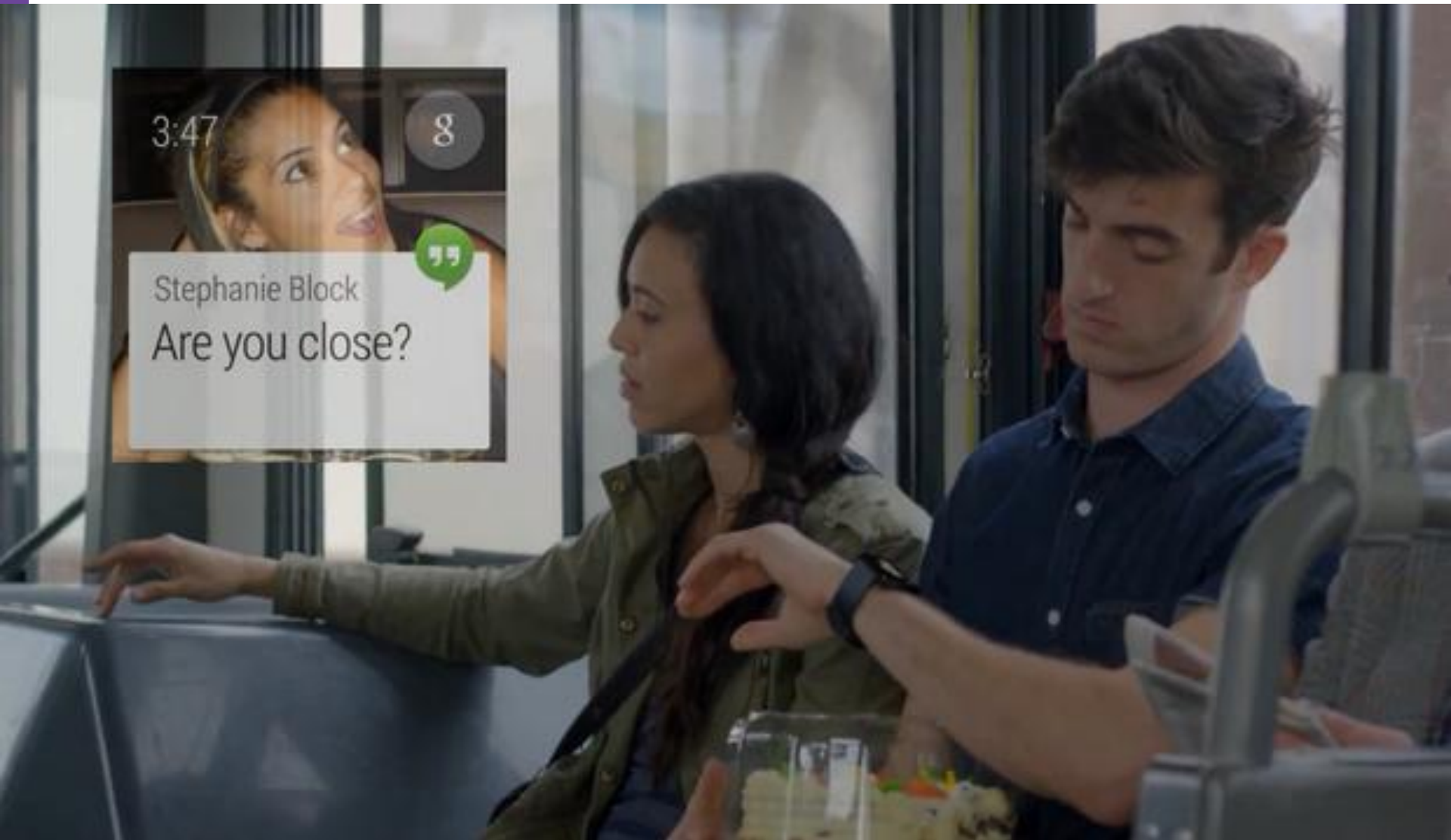




Judge by the potential not the present

ANDROID WEAR

What is android wear?





Joshua Kaplan
Hey dude!

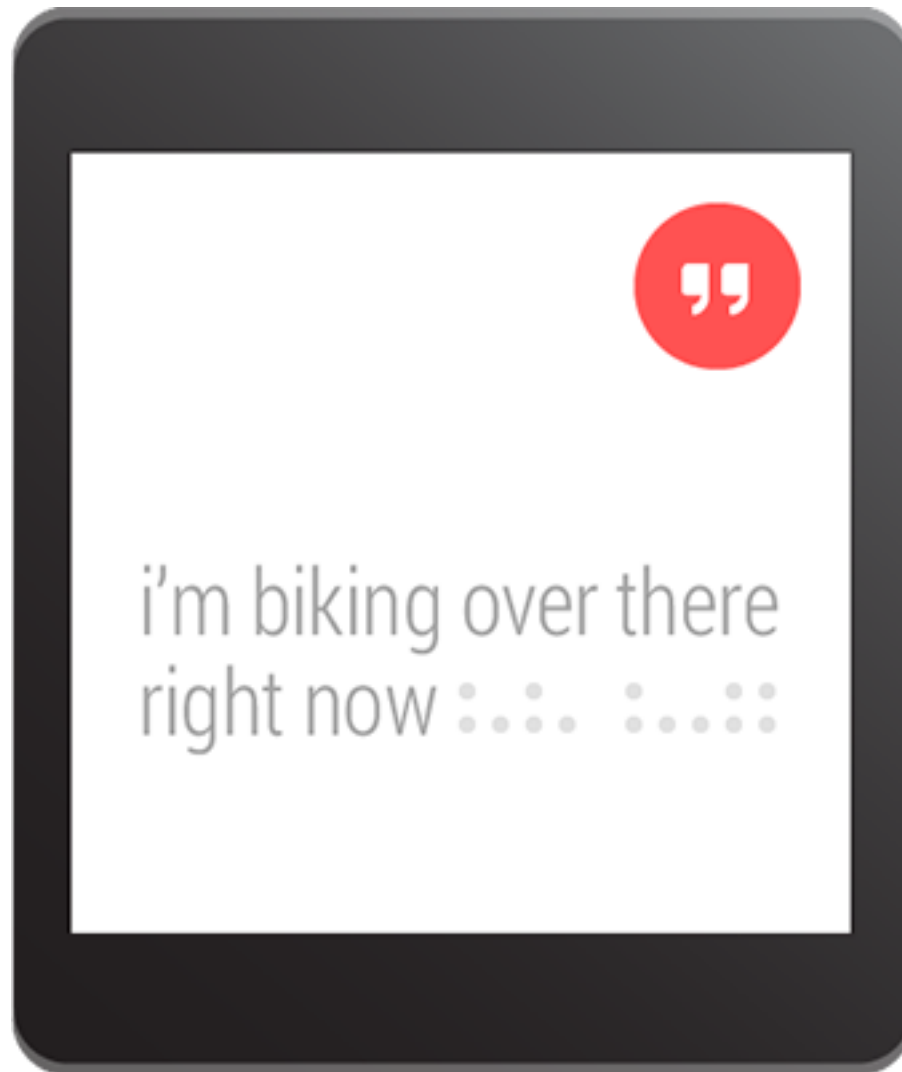
15 mins
To Home
Light traffic on 101 N

72°
San Francisco
49 0%

Turn right on
Washington St
in 50 ft

idea for a movie a
boy travels to mars
with space aliens

556.36



Prefefined semantic voice “intents”

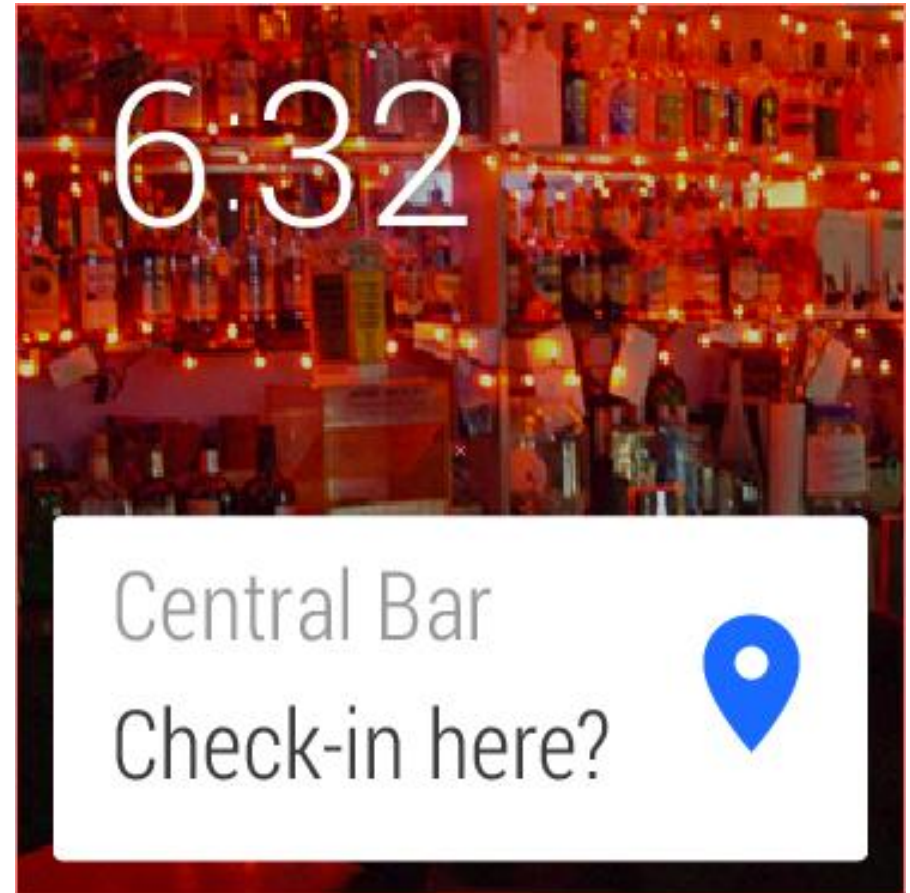
- Call car
- Take note
- Set alarm
- Set timer
- Show heart rate
- Show step count

Don't forget: It's not a small phone!

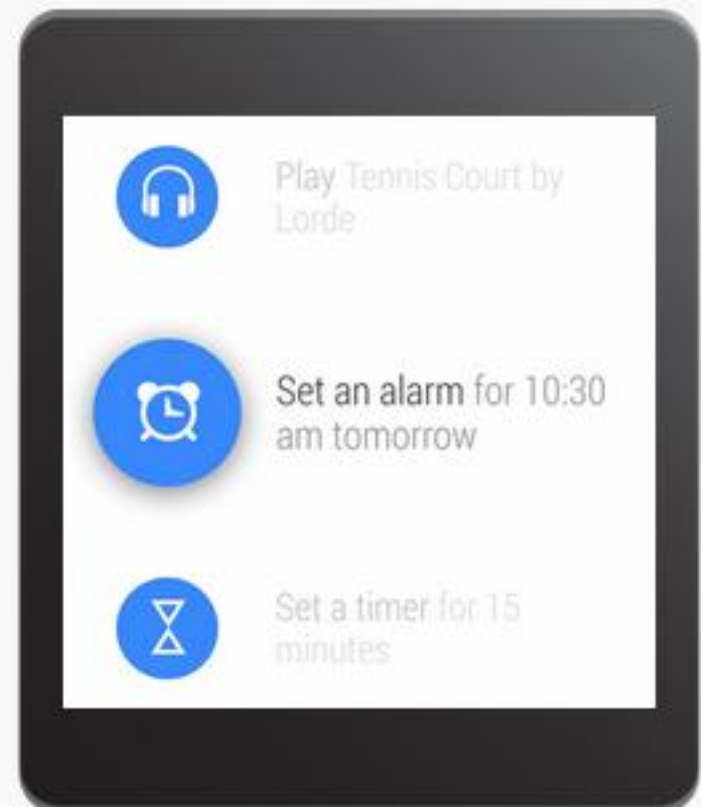
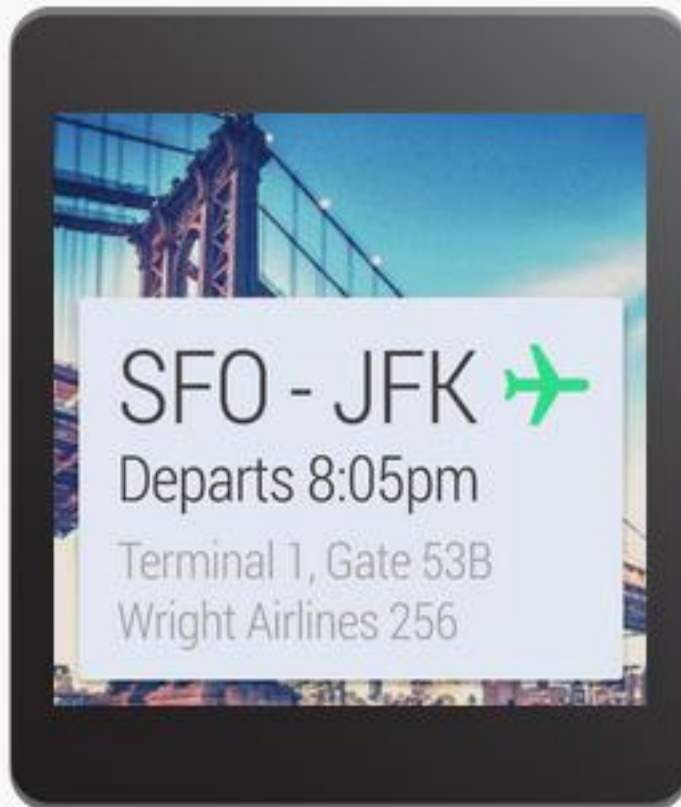
DESIGN PRINCIPLES

Think card streams

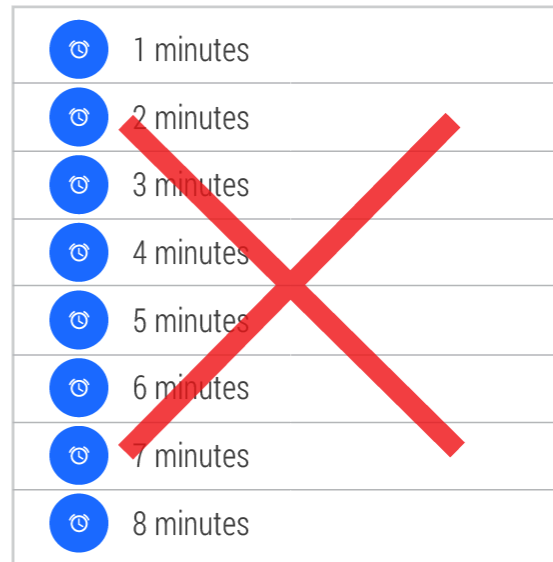
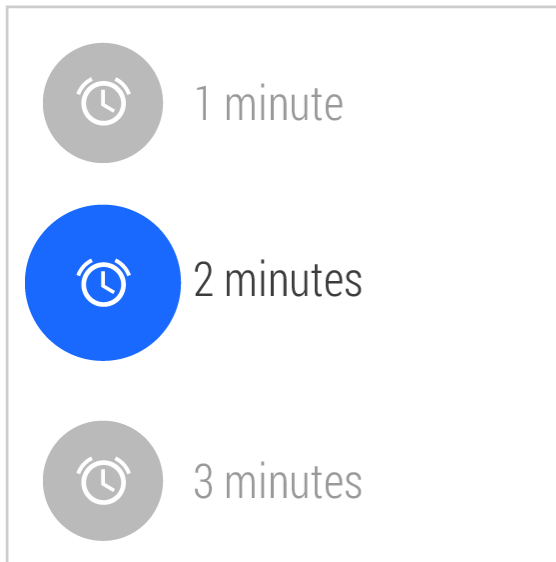
- Put non-intrusive cards in the stream
- Only alert / vibrate when needed – or get blocked
- Remember that cards are removed – have a fallback



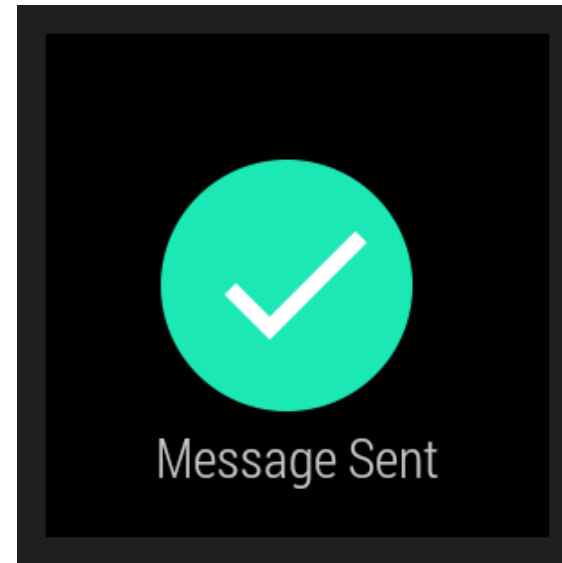
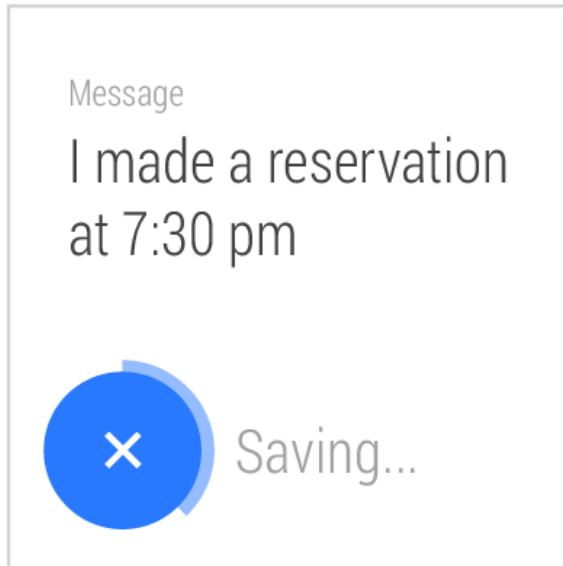
Zero physical interaction



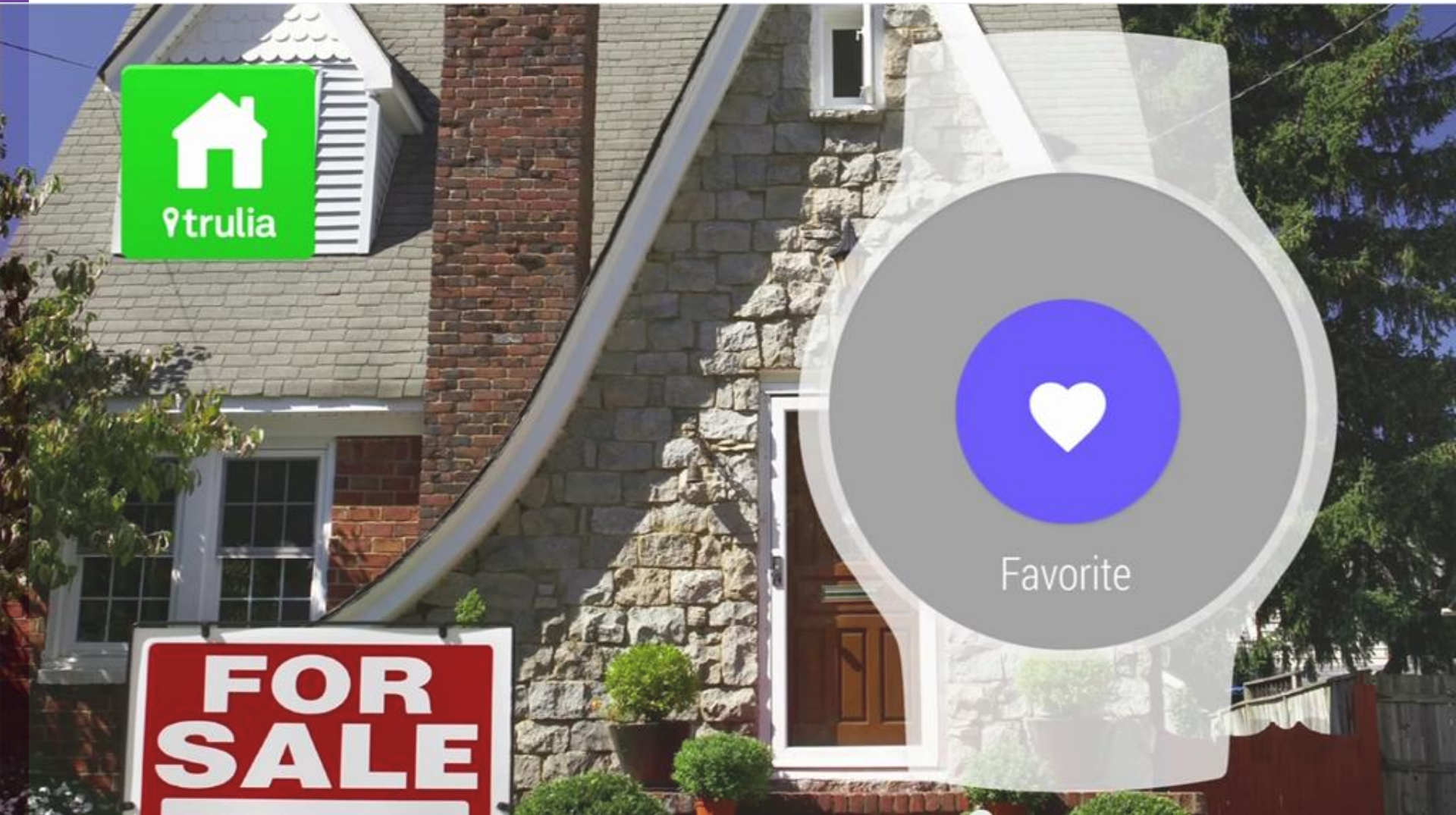
If you must interact



And after the interaction - don't stop the user



Be Contextual and smart



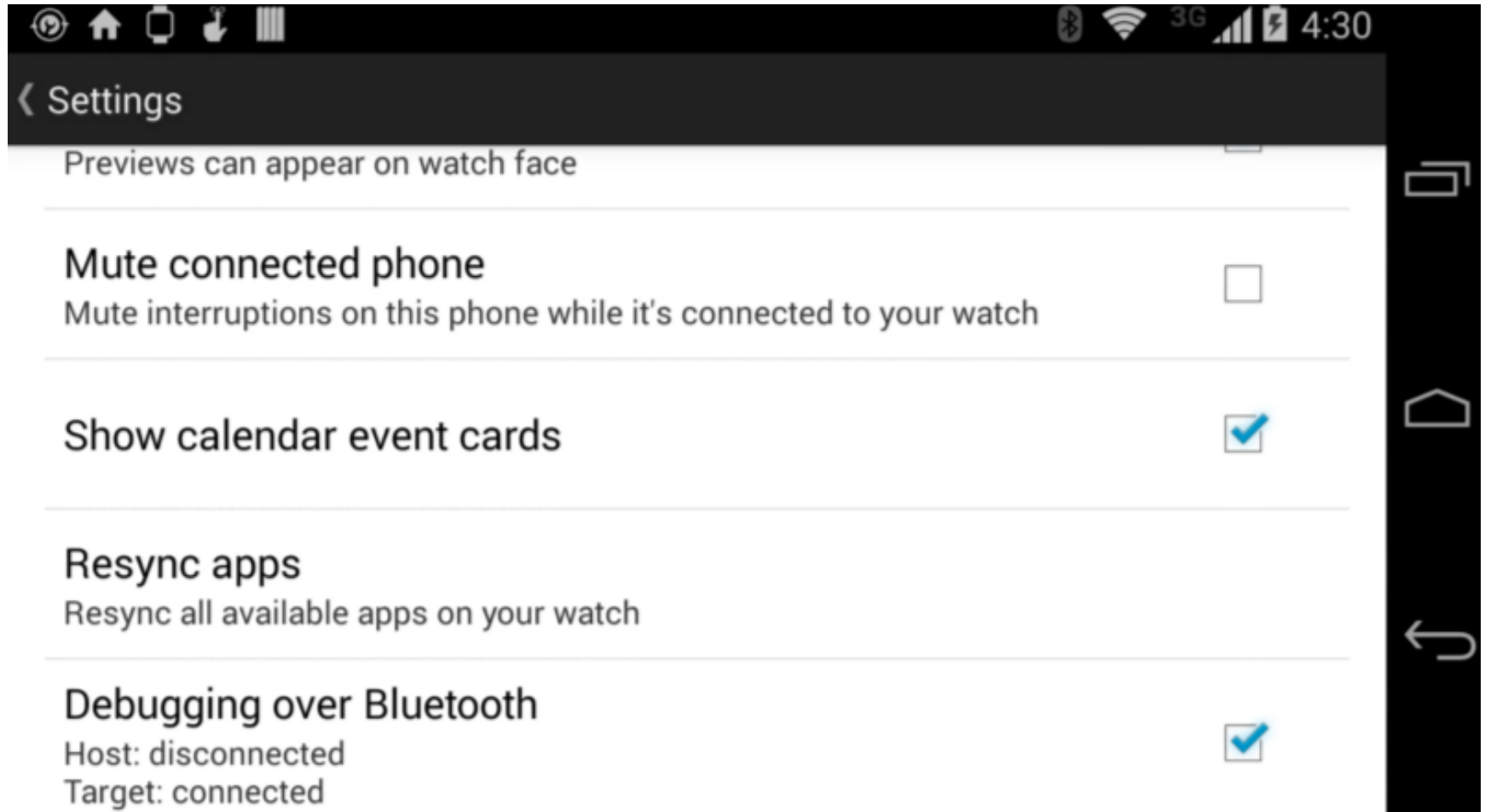
Lets get down to business

DEVELOPING FOR ANDROID WEAR

Dev tool checklist

- Check your phone - (<http://www.android.com/wear/check/>)
- Install Android Developer Tools and Platform 4.3 or a above
- Install SDK tools > v 23
- Update SDK with platform 4.4W.2 (API 20) or 5.0.1L (21)
- Android Studio is the way to go

Bluetooth debugging/transfer of APK



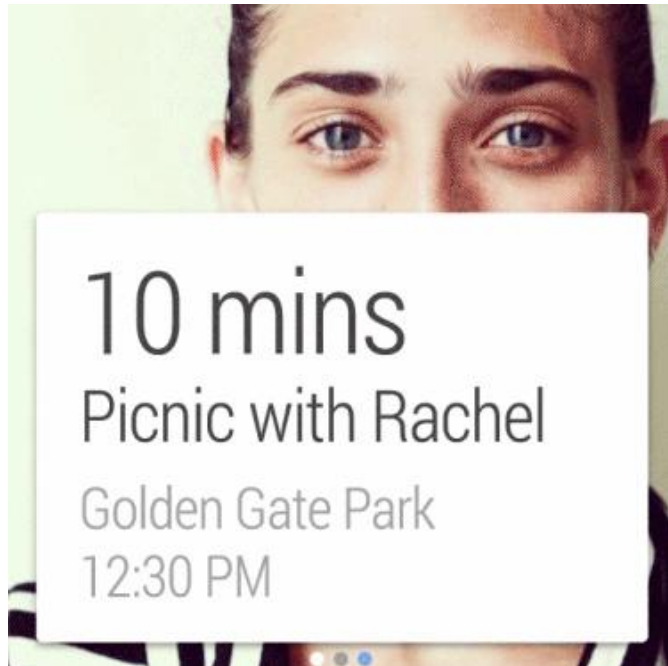
USB Debugging



Application architecture choices

1. Notification strategy

No application needed on the wearable device



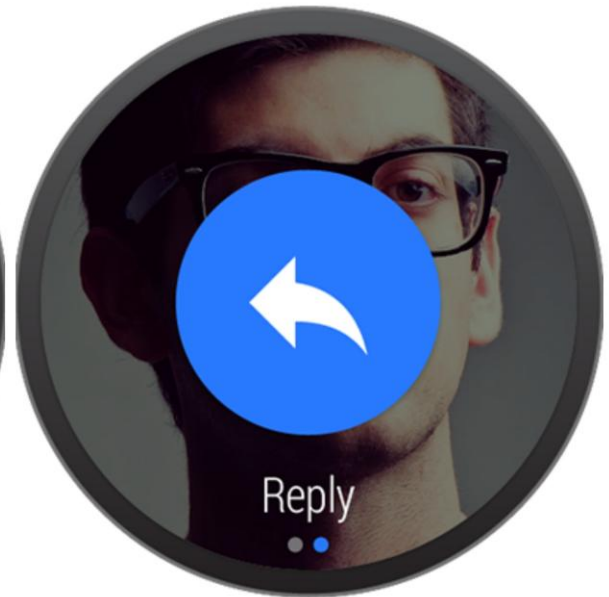
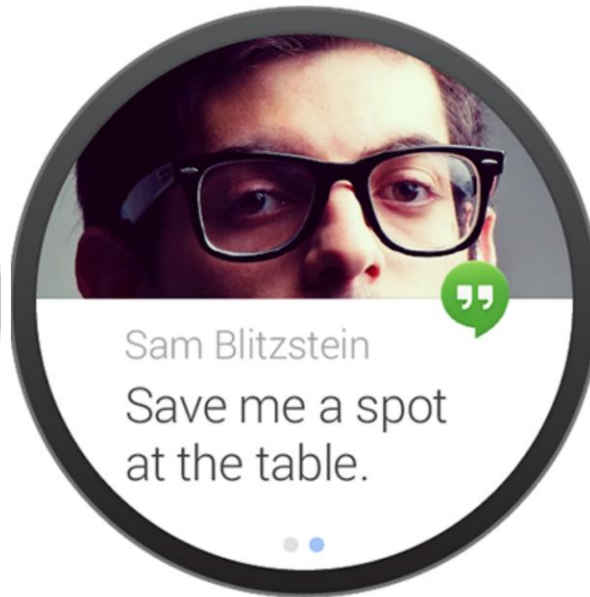
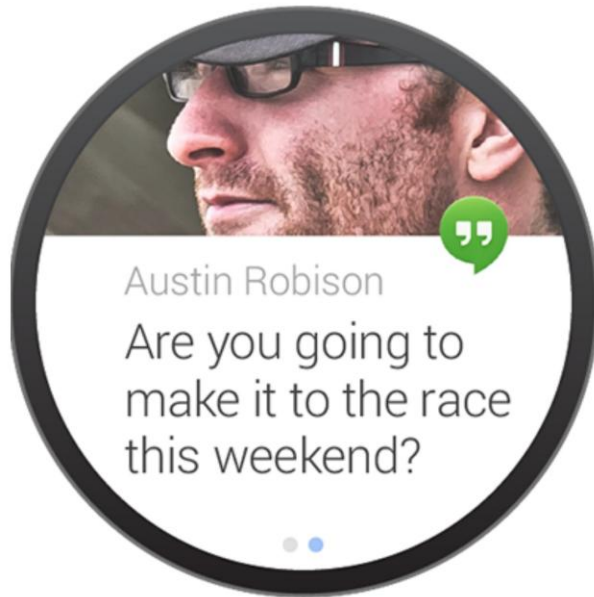
2. Companion app strategy

Application both on wearable and handheld

3. (Wearable only app)

Not very usefull at the moment

Bridged and superpowerful notifications



Creating a notification on wearable + handheld

```
NotificationCompat.Builder notificationBuilder =
    new NotificationCompat.Builder(MainPhoneActivity.this)
        .setSmallIcon(R.drawable.ic_plusone_small_off_client)
        .setLargeIcon(BitmapFactory.decodeResource(
            getResources(), R.drawable.common_full_open_on_phone))
        .setContentTitle("title")
        .setContentText("content")
        .setDefaults(Notification.DEFAULT_ALL)
        .setContentIntent(viewPendingIntent)
        .addAction(R.drawable.ic_plusone_medium_off_client,
            "map", mapPendingIntent);

NotificationManagerCompat notificationManager =
    NotificationManagerCompat.from(MainPhoneActivity.this);
notificationManager.notify(1, notificationBuilder.build());
```

Extending an existing application for wearables

COMPANION APP STRATEGY

Food Facts

Broccoli


Did you know broccoli contains more vitamin C than an orange? (ounce for ounce) Vitamin C is great for your hair, skin, teeth, fighting infections and keeping red blood cells healthy.

Nutrition

34.0kcal / 100g

Carbs	6.6%
Protein	2.8%
Fat	0.4%

Photo credit <http://www.flickr.com/photos/hulagway/>



Baby Foods
329 Food Items

Baked Products
497 Food Items

Beef Products
758 Food Items

Beverages
284 Food Items

Breakfast Cereals
408 Food Items

Cereal Grains And Pasta
184 Food Items

SUMMARY

VITAMINS AND MINERALS

CANDIES,ALMOND JOY BITES

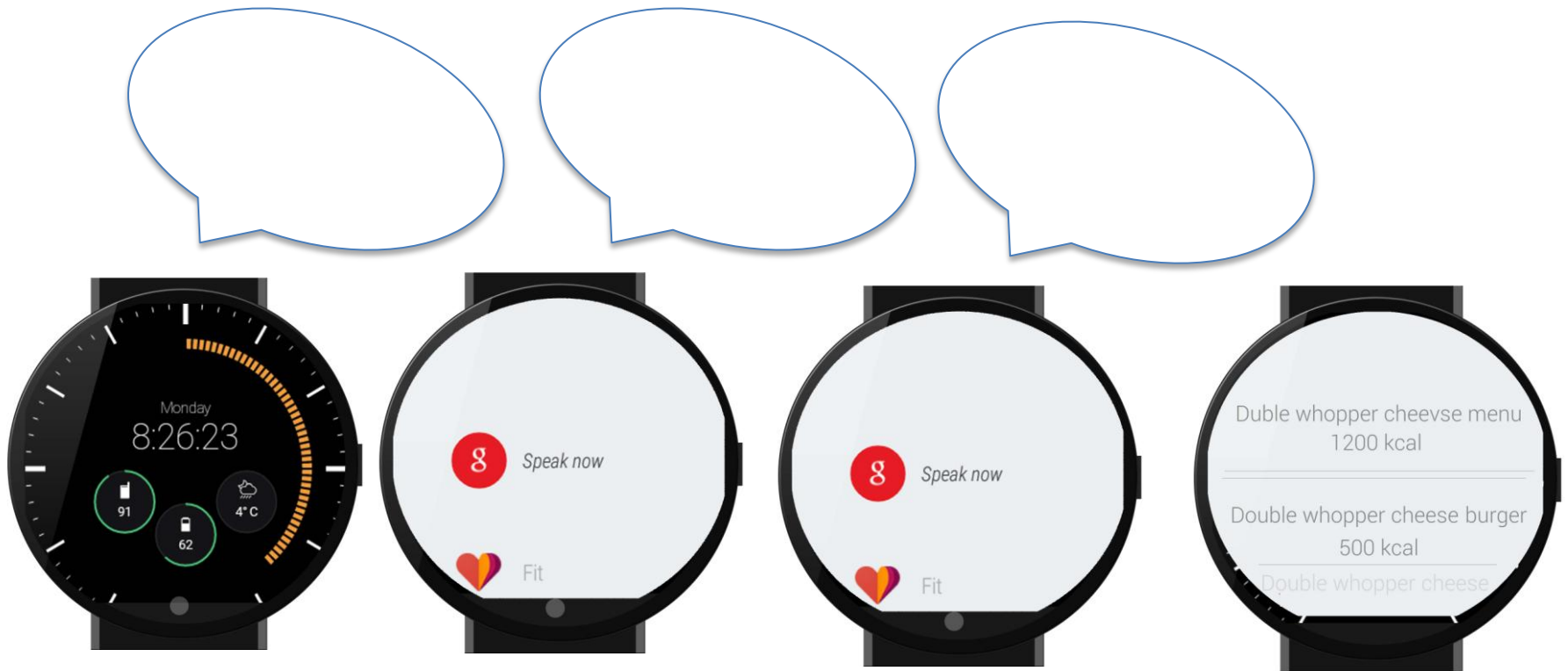
400.0 grams total.

Vitamin B5	10%	1 mg
Vitamin B6	12%	0,24 ug
Vitamin B12	0%	0 mcg
Vitamin C	6,7%	4 ug
Vitamin D	0%	0 iu
Vitamin E	0%	0 mg
Folate	0%	0 m
Calcium	43,6%	436 mg
Iron	35,5%	5,32 mg
Sodium	6,5%	156 mg
Selenium	0%	0 mcg

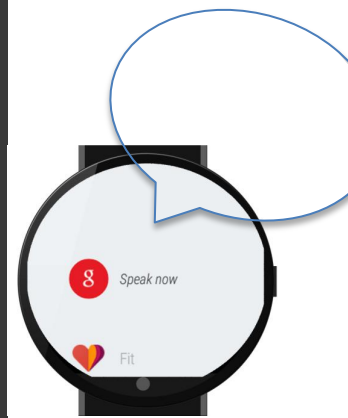
Antipattern



The wearable use case



Application launch by “start” keyword



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.glennbech.mywearproject">

    <uses-feature android:name="android.hardware.type.watch" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.DeviceDefault">
        <activity
            android:name=".WatchActivity"
            android:label="Food Search">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

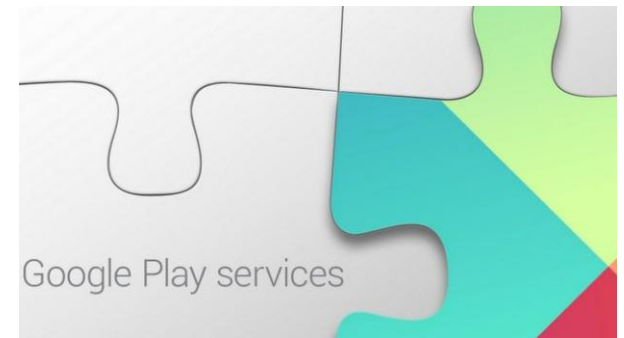
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```


Architecture



Google Play Services

- It was Ninja installed-and around Google IO 2013
- Was “Side loaded” through the Google Play Store app
- Enables Google to update important APIs without platform updates
- Android 2.3 + devices with the play store application receive updates
- Provides APIs for Maps, Fit, Drive, Wallet, Analytics, Gaming..
- And Phone – Wearable Communications



Create a Google API Client reference

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //....
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(new GoogleApiClient.ConnectionCallbacks() {
            @Override
            public void onConnected(Bundle connectionHint) {
                Log.d(TAG, "onConnected: " + connectionHint);
                displaySpeechRecognizer();
            }

            @Override
            public void onConnectionSuspended(int cause) {
                Log.d(TAG, "onConnectionSuspended: " + cause);
            }
        })
        .addOnConnectionFailedListener((result) -> {
            Log.d(TAG, "onConnectionFailed: " + result);
        })
        .addApi(Wearable.API)
        .build();
}
```

Connecting to the Google API Client

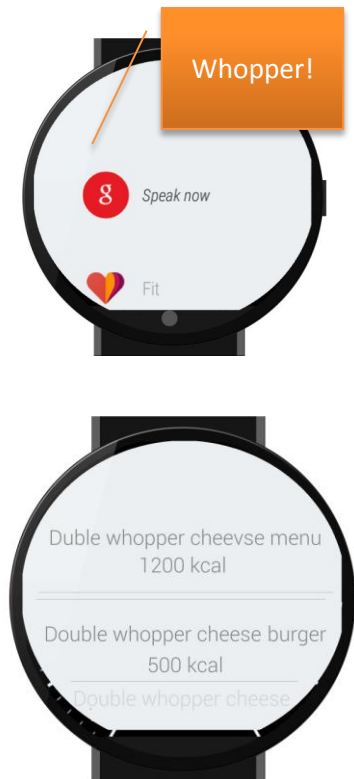
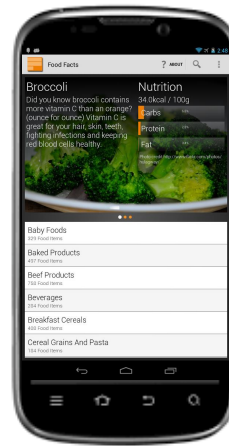
```
@Override
protected void onStart() {
    super.onStart();
    Log.i(TAG, "On Start - connecting to Google client.");
    mGoogleApiClient.connect();
}
```

```
@Override
protected void onStop() {
    super.onStop();
    mGoogleApiClient.disconnect();
}
```

Architecture



<https://www.mashape.com/glennbech/food-facts>



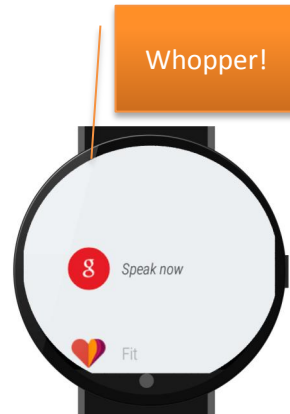
inmeta

“Talk about where
I smoke the patient”

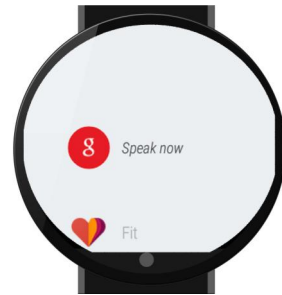
(voice input is not that reliable outdoors)

Voice input

```
private void displaySpeechRecognizer() {  
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);  
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,  
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);  
    Log.i(TAG, "Starting speech recognizer.");  
  
    startActivityForResult(intent, SPEECH_REQUEST_CODE);  
}
```



Voice input - results



```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
    if (requestCode == SPEECH_REQUEST_CODE && resultCode == RESULT_OK) {
        List<String> results = data.getStringArrayListExtra(
            RecognizerIntent.EXTRA_RESULTS);
        String spokenText = results.get(0);
        Log.i(TAG, "Spoken text: " + spokenText);
        sendMessage(spokenText);
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

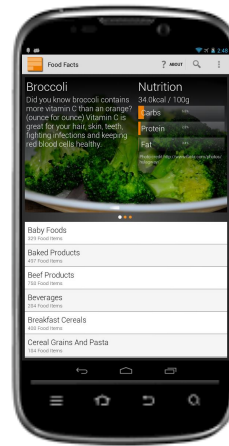
Sending the message

```
public void sendMessage(final String message) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            NodeApi.GetConnectedNodesResult nodes =
                NodeApi.getConnectedNodes(mGoogleApiClient).await();
            for (Node node : nodes.getNodes()) {
                SendMessageResult result =
                    MessageApi.sendMessage(
                        mGoogleApiClient, node.getId(),
                        PATH, message.getBytes()).await();
                Log.i(TAG, result.getStatus().getStatusMessage());
            }
        }
    }).start();
    Log.i(TAG, "Send Message finished.");
}
```

Architecture



<https://www.mashape.com/glennbech/food-facts>



inmeta

Receive & reply by phone

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.glennbech.mywearproject">

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="MyWearProject"
        android:theme="@style/AppTheme">

        ....
        <service android:name=".WearListenerService">
            <intent-filter>
                <action android:name="com.google.android.gms.wearable.BIND_LISTENER" />
            </intent-filter>
        </service>
    </application>
</manifest>
```

Receive & reply by phone

```
@Override
public void onMessageReceived(MessageEvent messageEvent) {
    super.onMessageReceived(messageEvent);

    if (Log.isLoggable(TAG, Log.DEBUG)) {
        Log.d(TAG, "onDataReceived: " + messageEvent);
    }

    GoogleApiClient googleApiClient = new GoogleApiClient.Builder(this)
        .addApi(Wearable.API)
        .build();

    ConnectionResult connectionResult =
        googleApiClient.blockingConnect(30, TimeUnit.SECONDS);

    String search = new String(messageEvent.getData());
    String resultJSON = "..."; // REST invocation

    Wearable.MessageApi.sendMessage(googleApiClient, messageEvent.getSourceNodeId(),
        DATA_ITEM_RECEIVED_PATH, resultJSON.getBytes());

    if (!connectionResult.isSuccess()) {
        Log.e(TAG, "Failed to connect to GoogleApiClient.");
        return;
    }
}
```

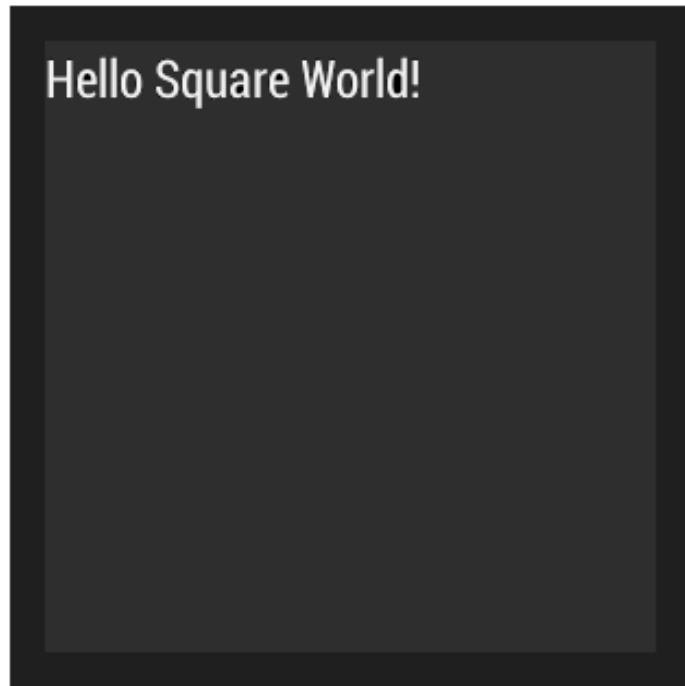
Architecture



Parsing and displaying the data

```
public class WatchActivity extends Activity implements MessageListener {  
    /// .....  
  
    @Override  
    public void onMessageReceived(MessageEvent messageEvent) {  
        Gson gson = new Gson();  
        List items = gson.fromJson(new  
            InputStreamReader(  
                new ByteArrayInputStream(messageEvent.getData()), ArrayList.class);  
    }  
}
```


Custom UI



Unique layouts for square and round

```
<android.support.wearable.view.WatchViewStub
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/watch_view_stub"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:rectLayout="@layout/rect_activity_wear"
    app:roundLayout="@layout/round_activity_wear">
</android.support.wearable.view.WatchViewStub>
```

Gotcha

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    WatchViewStub stub = (WatchViewStub) findViewById(R.id.watch_view_stub);
    stub.setOnLayoutInflatedListener(new WatchViewStub.OnLayoutInflatedListener() {
        @Override
        public void onLayoutInflated(WatchViewStub stub) {
            // Now you can access your views
            TextView tv = (TextView) stub.findViewById(R.id.text);
        }
    });
}
```

Shape Aware Layout

```
<android.support.wearable.view.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="15dp">
</android.support.wearable.view.BoxInsetLayout>
```

A large dark gray circle with a thin white border. Inside the circle, at the top, is a gray square. The text "Hello Round World!" is written in white at the top left of the square.

Hello Round World!

Packaging & distribution of apps

- Wearable apps are packaged within handheld apps
- Handheld devices install them automatically
- Actually packacked in the raw folder

*Thank
You*