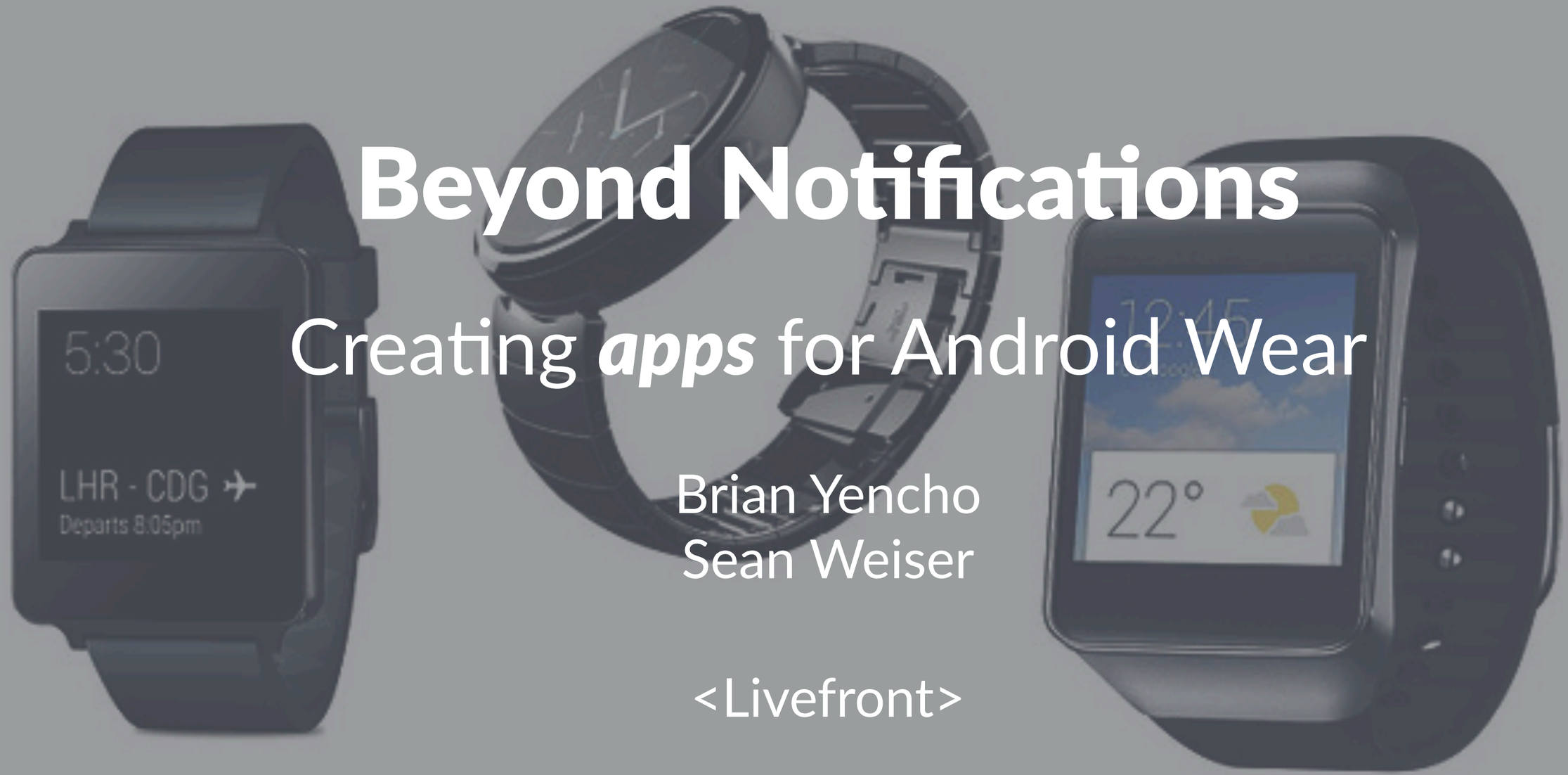


Beyond Notifications

Creating ***apps*** for Android Wear

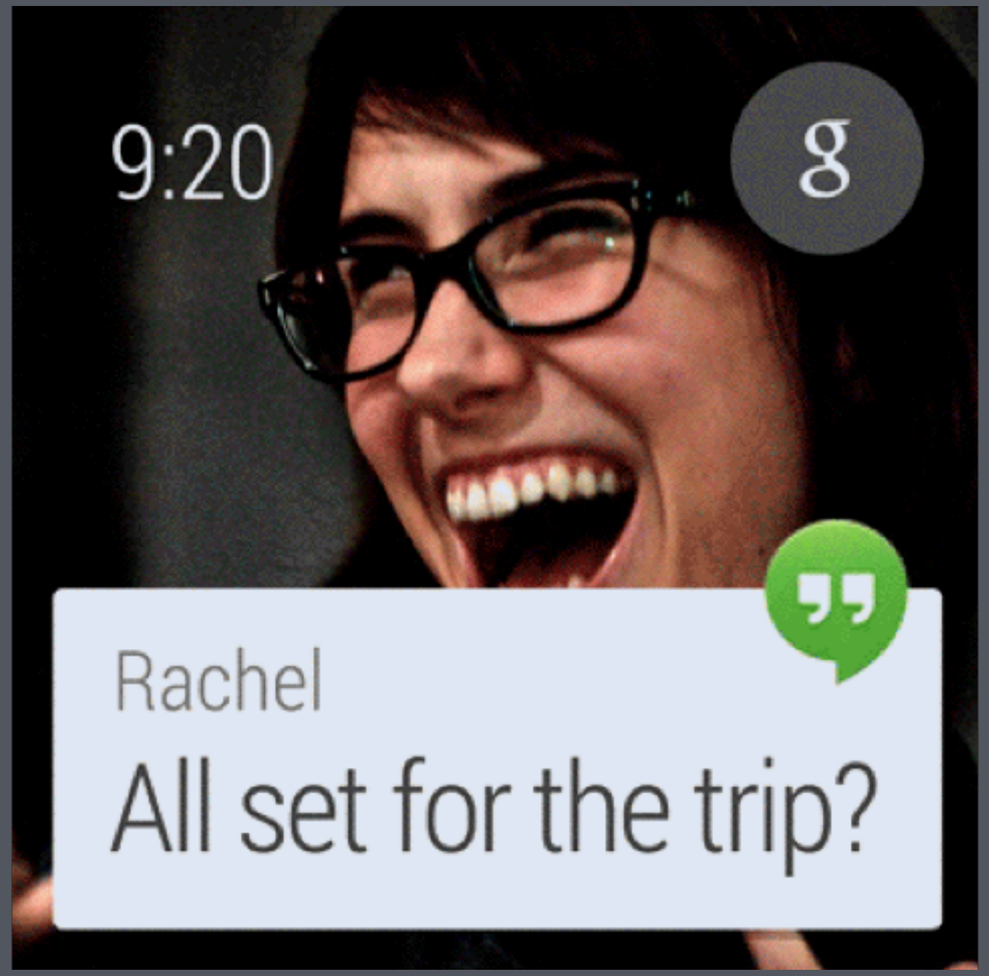
Brian Yencho
Sean Weiser

<Livefront>

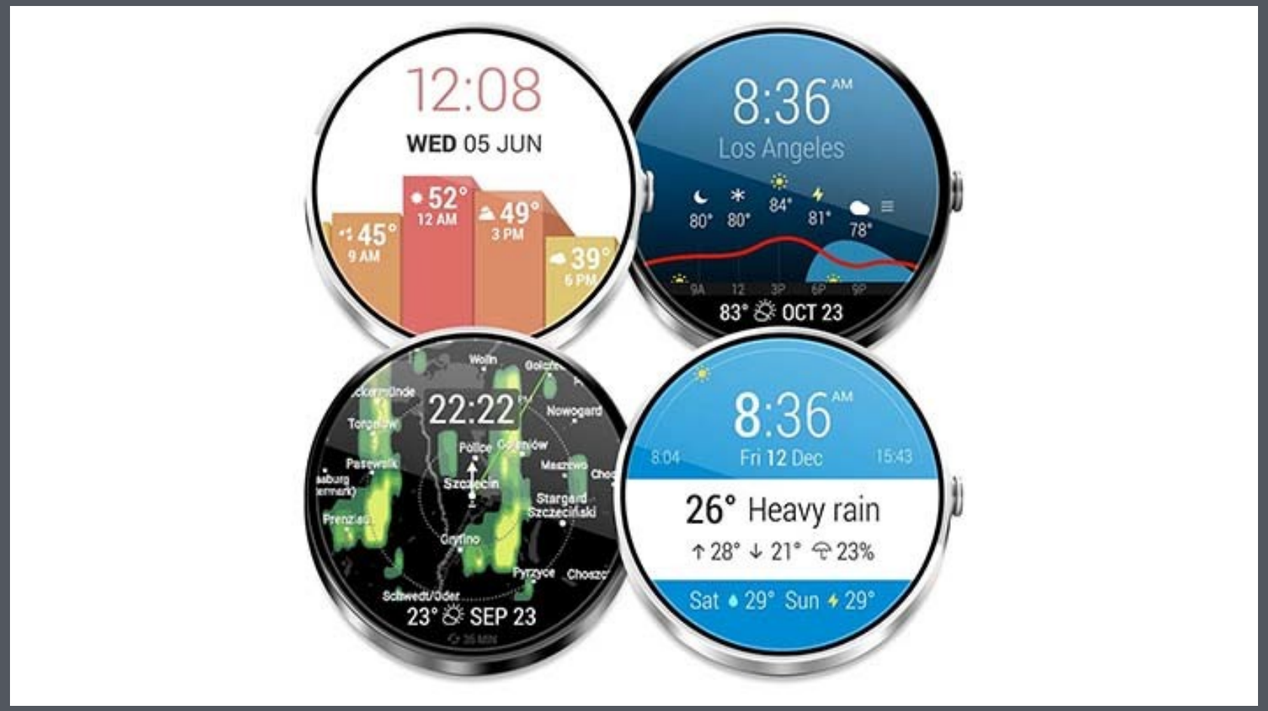
Part 1

Custom UI

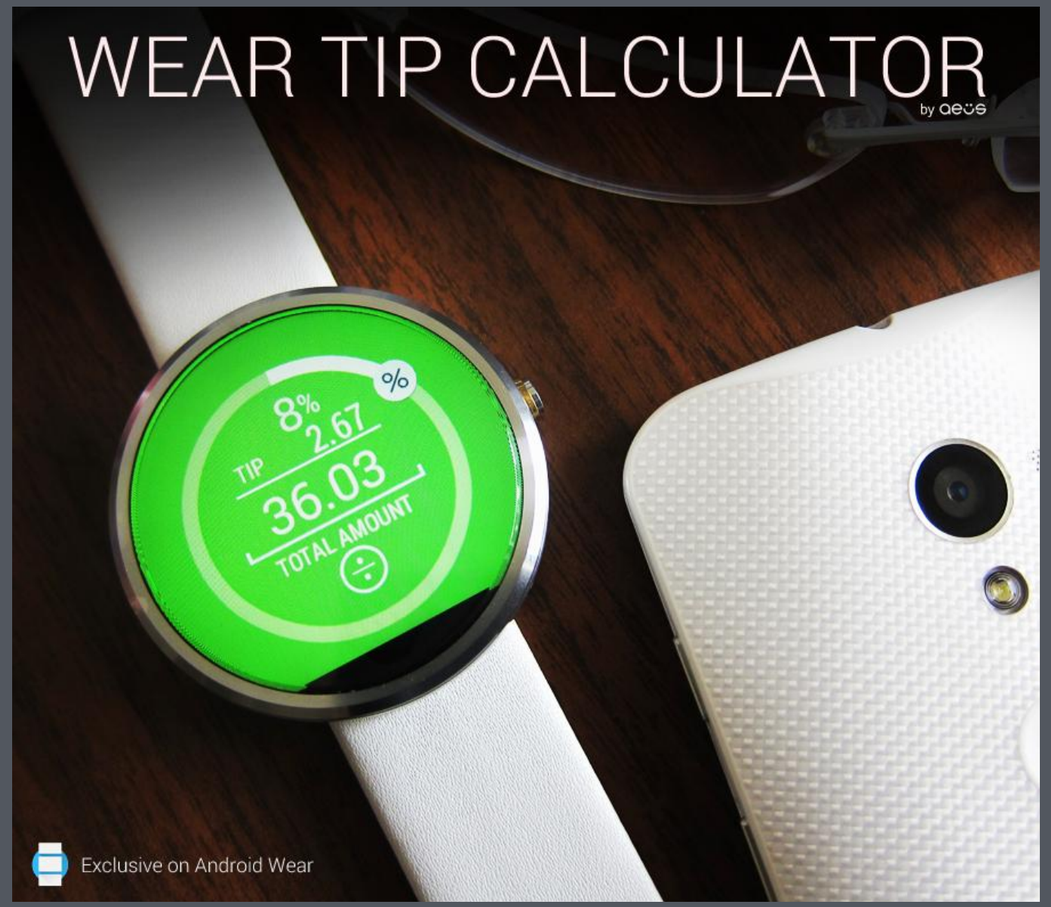
Emphasis on notifications



Full screen apps possible



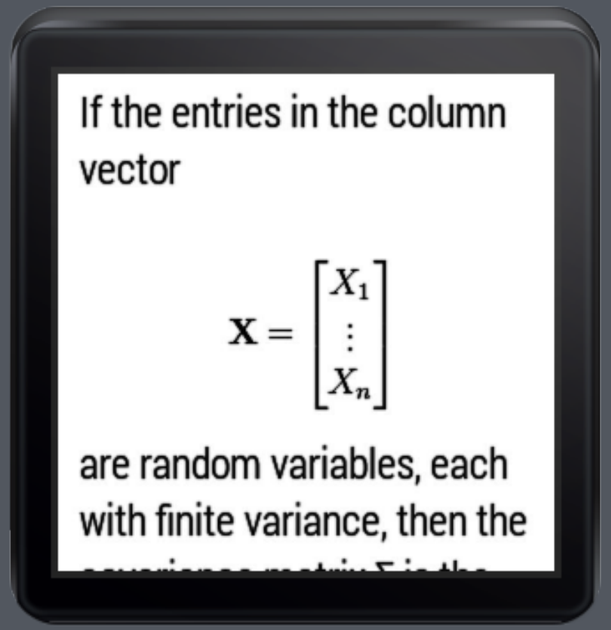
InstaWeather for Android Wear



Wear Tip Calculator


ATTOPEDIA
CONTENT SOURCED FROM WIKIPEDIA
Tap screen to begin search


ATTOPEDIA
CONTENT SOURCED FROM WIKIPEDIA
Covariance matrix
In probability theory and statistics, a covariance matrix (also known as dispersion matrix or variance–covariance matrix) is a matrix whose


If the entries in the column vector
$\mathbf{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$
are random variables, each with finite variance, then the

Attopedia

Android Wear ***is*** Android

It's not about how ***can*** you create custom UI

It's about how ***should*** you do it

Android Wear devices

- **square or round**
  - layouts should be "shape-aware"
- **small**
  - low information density
  - big gestures

Wearable Support Library

In build.gradle:

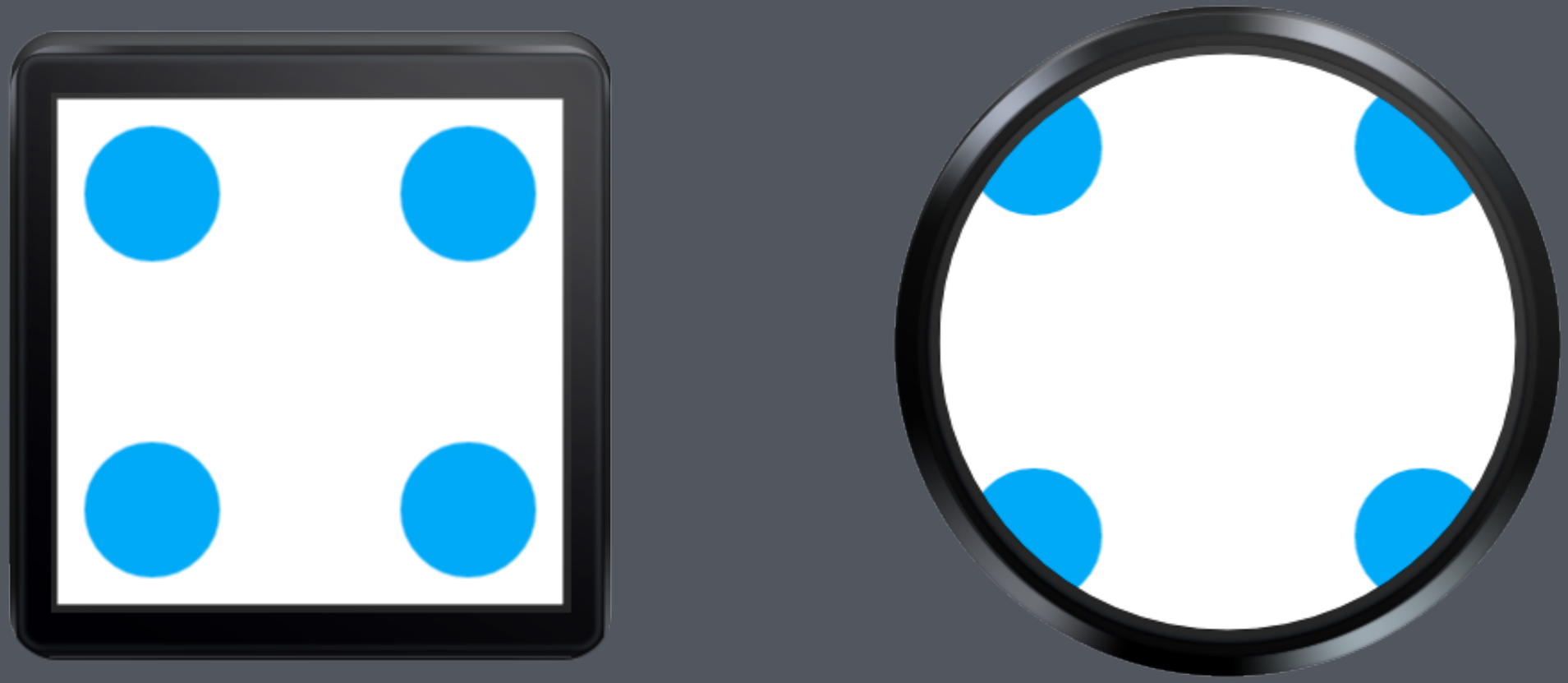
```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.android.support:wearable:+'
    compile 'com.google.android.gms:play-services-wearable:+'
}
```

Note: the Wear Support library is ***subject to change***

- BoxInsetLayout
- CardFragment / CardFrame
- ConfirmationActivity
- DismissOverlayView
- GridViewPager / GridPagerAdapter / FragmentGridPagerAdapter
- WatchViewStub
- WearableListView

Shape-Aware Layouts

FrameLayout with a view in each corner:

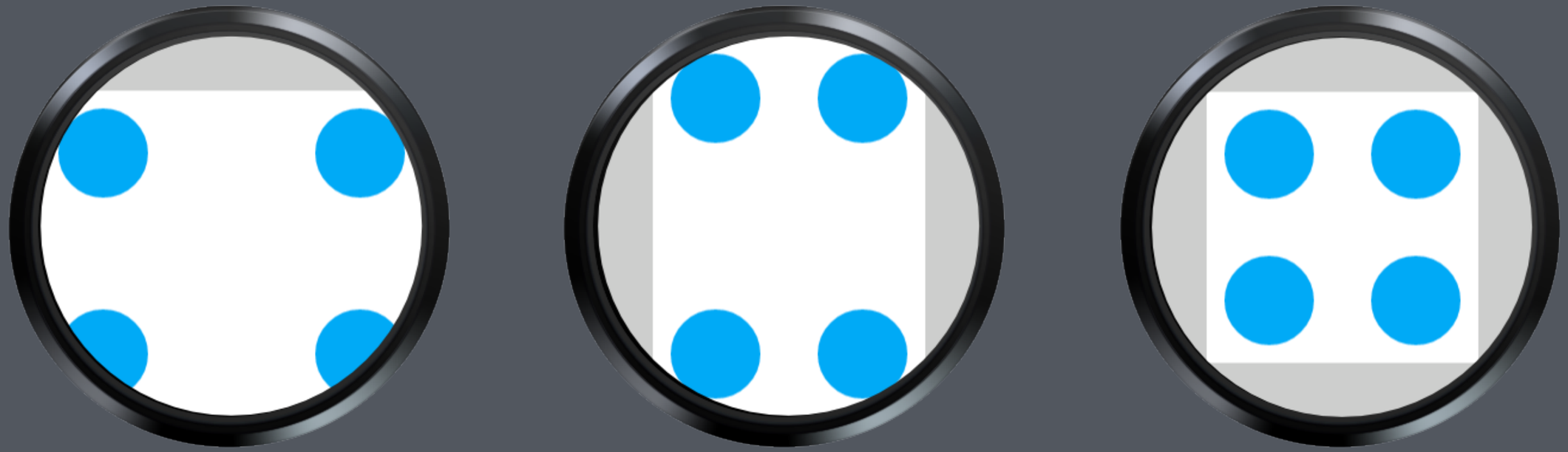


BoxInsetLayout

BoxInsetLayout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@android:color/white"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <!-- Same layout as before -->
    <!-- Note the app:layout_box attribute -->
    <include layout="@layout/activity_bad_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_box="all" />

</android.support.wearable.view.BoxInsetLayout>
```

`app:layout_box="all"`
(left, right, top, bottom, all)



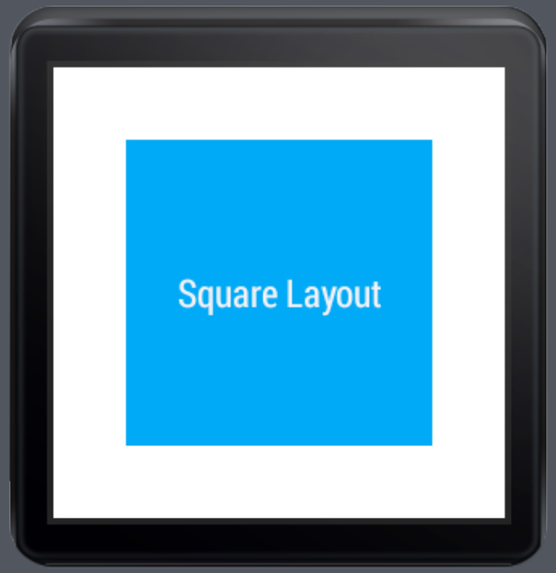
```
app:layout_box="top"
app:layout_box="left|right"
app:layout_box="all"
```

WatchViewStub

WatchViewStub

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.WatchViewStub
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    ...
    app:rectLayout="@layout/rect_activity_watch_view_stub"
    app:roundLayout="@layout/round_activity_watch_view_stub" />
```

WatchViewStub

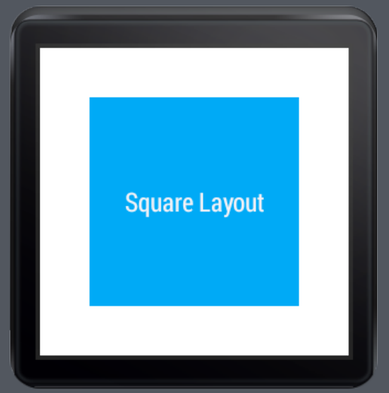



```
app:rectLayout="@layout/rect_activity"
app:roundLayout="@layout/round_activity"
```

WatchViewStub

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_watch_view_stub);

    final WatchViewStub stub = (WatchViewStub) findViewById(R.id.watch_view_stub);
    stub.setOnLayoutInflatedListener(new WatchViewStub.OnLayoutInflatedListener() {
        @Override
        public void onLayoutInflated(WatchViewStub stub) {
            // If you want to update the TextView you'll have to do it here, otherwise you'll
            // get an NPE because the WatchViewStub doesn't immediately inflate its contents.
            mTextView = (TextView) stub.findViewById(R.id.text);
        }
    });
}
```

BoxInsetLayout / WatchViewStub often the root of any layout



Cards


Card title
Card text


Sample text 1
Sample text 2
Sample text 3


Card title
Card text


Sample text 1
Sample text 2
Sample text 3

CardFragment

CardFragment

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_card_fragment);

    // Create fragment
    CardFragment cardFragment =
        CardFragment.create(getString(R.string.card_fragment_title), // title
                            getString(R.string.card_fragment_text),  // text
                            R.drawable.card_icon);                   // icon
    cardFragment.setCardGravity(Gravity.BOTTOM);

    // Add to container layout
    ...
}
```

CardFrame

CardFrame

Root layout > CardScrollView > CardFrame > arbitrary content

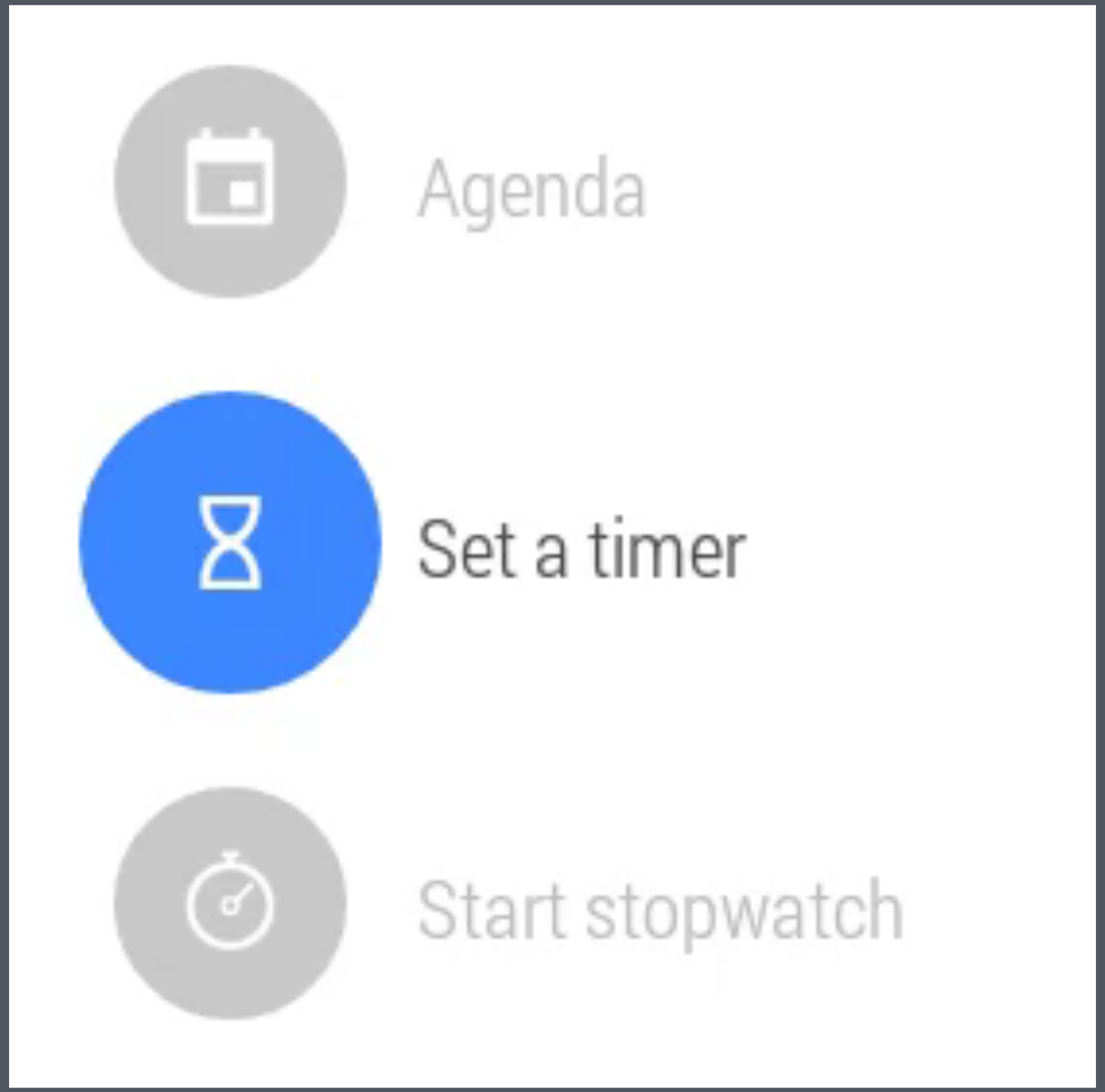
CardFrame

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@color/demo_item_circle_color_blue"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <android.support.wearable.view.CardScrollView
        android:id="@+id/card_scroll_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_box="bottom">

        <android.support.wearable.view.CardFrame
            android:id="@+id/card_frame"
            android:layout_width="match_parent"
            android:layout_height="match_parent" >

            <!-- Single child with arbitrary content -->
            <LinearLayout
                android:orientation="vertical"
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                ...
```

CardFrame

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_card_frame);

    // CardFrame should sit inside a CardScrollView to control
    // gravity and card expansion
    CardScrollView cardScrollView =
            (CardScrollView) findViewById(R.id.card_scroll_view);
    cardScrollView.setCardGravity(Gravity.BOTTOM);
}
```

Lists


Agenda
Set a timer
Start stopwatch

WearableListView

WearableListView

WearableListView

Three pieces:

1. WearableListView

2. WearableListView.Adapter

3. WearableListView.OnCenterProximityListener (optional)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@android:color/white"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <!-- Just include like a normal ListView -->
    <android.support.wearable.view.WearableListView
        android:id="@+id/wearable_list"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        app:layout_box="left|right"/>

</android.support.wearable.view.BoxInsetLayout>
```

WearableListView.Adapter

Main interface:

```
public WearableListView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType);

public void onBindViewHolder(WearableListView.ViewHolder viewHolder, int position);

public int getItemCount();
```

Just like RecyclerView.Adapter

WearableListView.Adapter

WearableListView.ViewHolder

```
private static class ItemViewHolder extends WearableListView.ViewHolder {
    TextView text;
    public ItemViewHolder(View itemView) {
        super(itemView);
        text = (TextView) itemView.findViewById(R.id.name);
    }
}
```

WearableListView.OnCenterProximityListener (Optional)

Interface:

```
public void onCenterPosition(boolean animate);

public void onNonCenterPosition(boolean animate);
```

WearableListView.OnCenterProximityListener (Optional)

Create custom view class

```
public class DemoItemLayout extends LinearLayout implements WearableListView.OnCenterProximityListener {
    ...
}
```

Inflate in your adapter

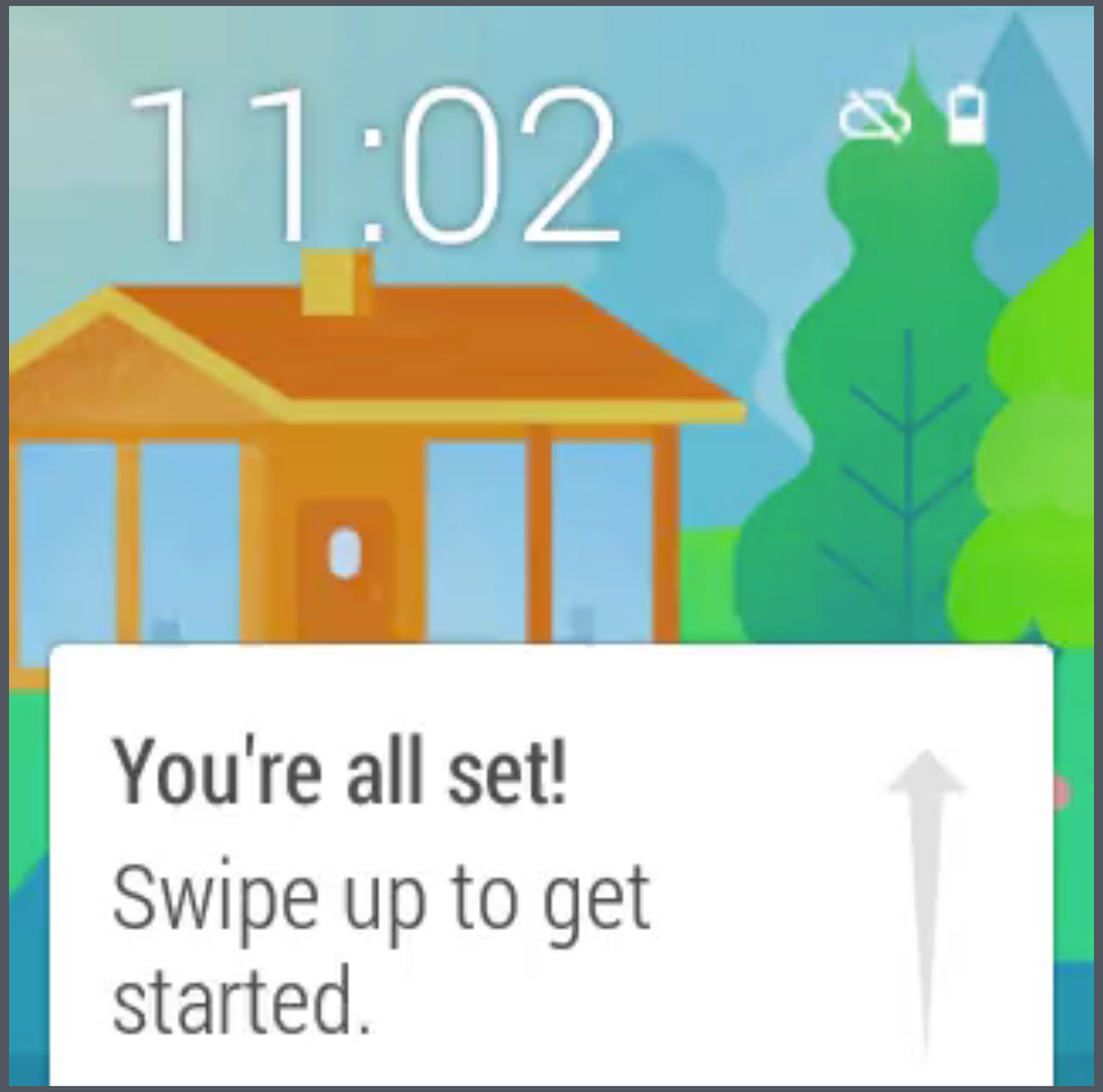
```
@Override
public WearableListView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
    return new ItemViewHolder(LayoutInflater.from(viewGroup.getContext())
            .inflate(R.layout.view_demo_item_layout, viewGroup, false));
}
```

```
@Override
public void onCenterPosition(boolean animate) {
    // Update icon color
    ((GradientDrawable) mCircle.getDrawable()).setColor(mColorBlue);

    // Animate or set values
    if (animate) {
        mCircle.animate()
                .scaleX(SCALE_MAX)
                .scaleY(SCALE_MAX)
                .setDuration(ANIMATION_DURATION)
                .start();
        mText.animate()
                .alpha(ALPHA_MAX)
                .setDuration(ANIMATION_DURATION)
                .start();
    } else {
        mCircle.setScaleX(SCALE_MAX);
        mCircle.setScaleY(SCALE_MAX);
        mText.setAlpha(ALPHA_MAX);
    }
}
```

```
@Override
public void onNonCenterPosition(boolean animate) {
    // Update icon color
    ((GradientDrawable) mCircle.getDrawable()).setColor(mColorGreen);

    // Animate or set values
    if (animate) {
        mCircle.animate()
                .scaleX(SCALE_MIN)
                .scaleY(SCALE_MIN)
                .setDuration(ANIMATION_DURATION)
                .start();
        mText.animate()
                .alpha(ALPHA_MIN)
                .setDuration(ANIMATION_DURATION)
                .start();
    } else {
        mCircle.setScaleX(SCALE_MIN);
        mCircle.setScaleY(SCALE_MIN);
        mText.setAlpha(ALPHA_MIN);
    }
}
```

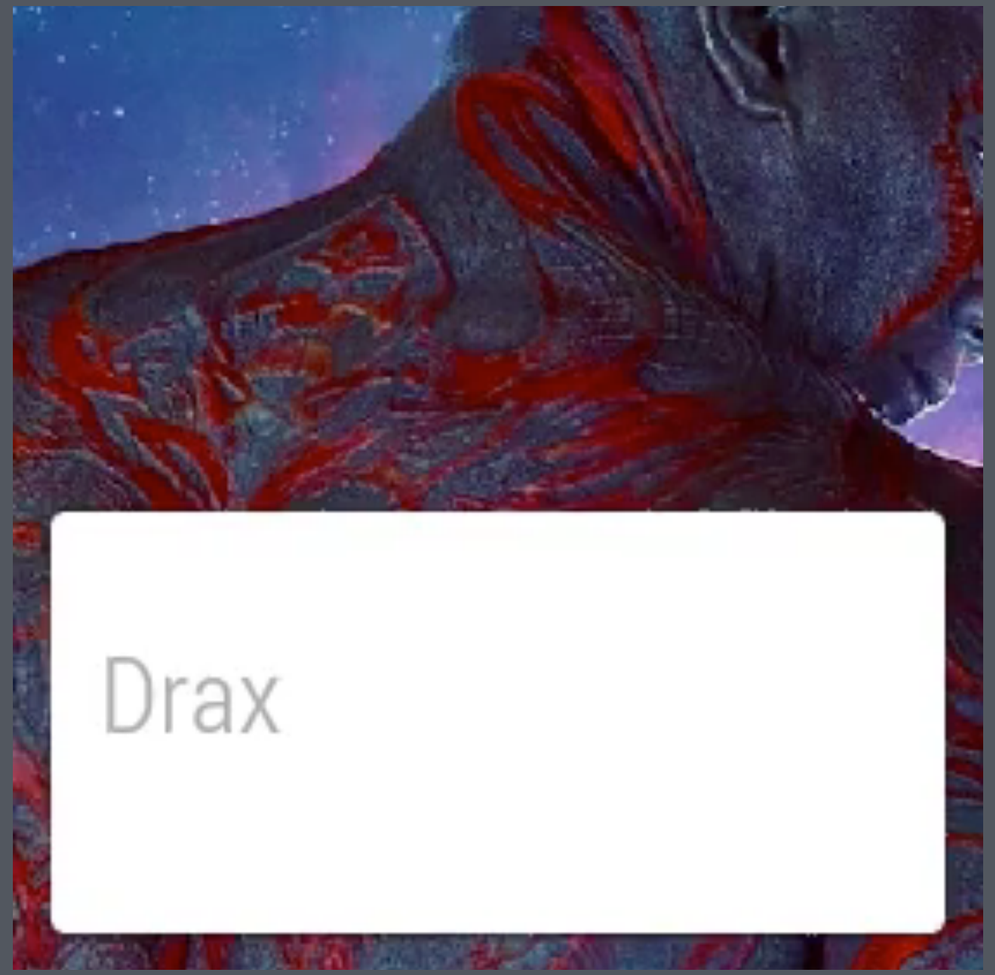

Bad Layout Demo
BoxInset Demo

2D Pickers


11:02
You're all set!
Swipe up to get started.

GridViewPager

GridViewPager



GridViewPager

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@android:color/black"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <!-- Include just like a normal ViewPager -->
    <android.support.wearable.view.GridViewPager
        android:id="@+id/pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_box="bottom"/>

</android.support.wearable.view.BoxInsetLayout>
```

GridViewPager

Two possible adapters:

1. GridPagerAdapter

2. FragmentGridPagerAdapter

FragmentGridPagerAdapter

Interface

```
public Fragment getFragment(int row, int column);

public int getRowCount();

public int getColumnCount(int row);
```

Optional overrides:

```
// Handle cross-fades between rows and parallax within rows
public Drawable getBackgroundForRow(int row);

// Handles cross-fades across each cell
public Drawable getBackgroundForPage(int row, int column);
```

FragmentGridPagerAdapter

```
@Override
public Drawable getBackgroundForRow(int row) {
    int imageResource = mData.get(row).getImageResource();
    return mContext.getResources().getDrawable(imageResource);
}
```


Drax

Delayed Confirmations


Launch the
Nukes?

DelayedConfirmationView

DelayedConfirmationView

DelayedConfirmationView

Cancellation

DelayedConfirmationView

Confirmation

DelayedConfirmationView

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@android:color/white"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <!-- Other content -->

    <android.support.wearable.view.DelayedConfirmationView
        android:id="@+id/delayed_confirm"
        android:layout_width="@dimen/delayed_confirmation_size"
        android:layout_height="@dimen/delayed_confirmation_size"
        android:layout_gravity="center"
        android:visibility="gone"
        android:src="@drawable/ic_full_cancel"
        app:layout_box="all"
        app:circle_color="@color/demo_blue"
        app:circle_border_color="@color/demo_green"
        app:circle_border_width="@dimen/delayed_confirmation_border_width"
        app:circle_radius="@dimen/delayed_confirmation_radius" />

</android.support.wearable.view.BoxInsetLayout>
```

DelayedConfirmationView

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_delayed_confirmation_view);

    ...

    mDelayedView = (DelayedConfirmationView)
            findViewById(R.id.delayed_confirm);

    // Set the timer to 2 seconds
    mDelayedView.setTotalTimeMs(2000);

    // Set this activity as the listener
    mDelayedView.setListener(this);

    // Start countdown
    mDelayedView.start();
}
```

DelayedConfirmationView

DelayedConfirmationView.DelayedConfirmationListener

```
// For confirmation
public void onTimerFinished(View view);

// For cancellation etc.
public void onTimerSelected(View view);
```

DelayedConfirmationView

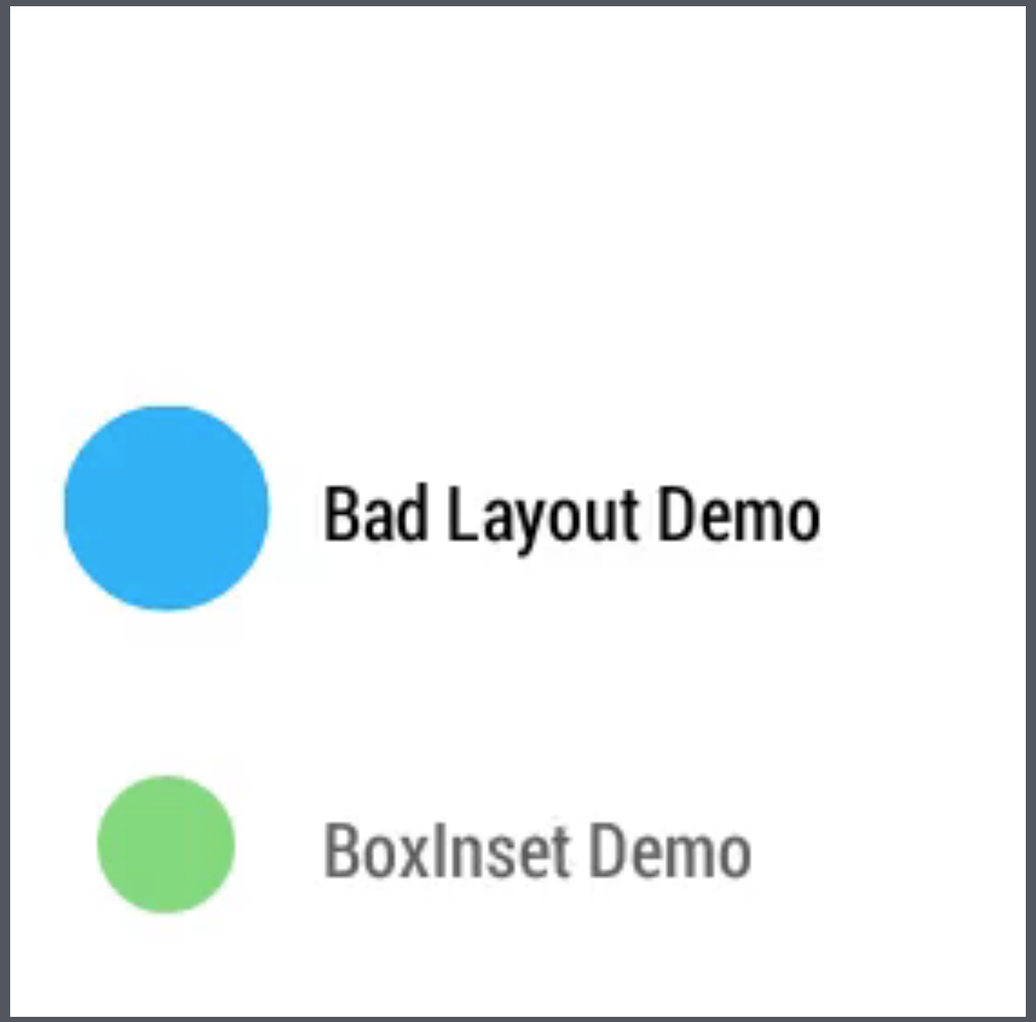
```
@Override
public void onTimerSelected(View view) {
    // Indicate that the timer should do nothing when it finishes
    mCanceled = true;

    // Show a cancellation toast
    Toast.makeText(this, getString(R.string.message_canceled),
                   Toast.LENGTH_SHORT).show();
    finish();
}
```

DelayedConfirmationView

```
@Override
public void onTimerFinished(View view) {
    if (mCanceled) {
        // Timer was canceled, do nothing
        return;
    }

    // Show the system success animation
    Intent intent = new Intent(this, ConfirmationActivity.class);
    intent.putExtra(ConfirmationActivity.EXTRA_ANIMATION_TYPE,
            ConfirmationActivity.SUCCESS_ANIMATION);
    intent.putExtra(ConfirmationActivity.EXTRA_MESSAGE,
            getString(R.string.message_confirmed));

    // We'll dismiss this activity when the animation ends
    startActivityForResult(intent, CONFIRMATION_REQUEST_CODE);
}
```

Long-press-to-dismiss

Swipe-to-dismiss is the system default



Some apps may need to disable the swipe-to-dismiss feature

```
<style name="AppTheme" parent="Theme.DeviceDefault">
    <item name="android:windowSwipeToDismiss">false</item>
</style>
```

They are then *required* to implement the **long-press-to-dismiss** pattern

DismissOverlayView

DismissOverlayView

DismissOverlayView

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@color/demo_blue"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <!-- Content to lie underneath DismissOverlayView -->

    <android.support.wearable.view.DismissOverlayView
        android:id="@+id/dismiss_overlay"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        app:layout_box="all" />

</android.support.wearable.view.BoxInsetLayout>
```

DismissOverlayView

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_dismiss_overlay_view);

    mDismissOverlay = (DismissOverlayView) findViewById(R.id.dismiss_overlay);

    // Trigger the dismiss overlay view on a long press
    mDetector = new GestureDetector(this, new GestureDetector.SimpleOnGestureListener() {
        public void onLongPress(MotionEvent ev) {
            mDismissOverlay.show();
        }
    });

    ...
}
```

DismissOverlayView

```
@Override
public boolean dispatchTouchEvent(MotionEvent ev) {
    // Always check for long presses before dispatching any event. Note that if we do this
    // check in onTouchEvent instead of here when swipeToDismiss is enabled, the long press
    // event would be consumed by the system before ever reaching the gesture detector.
    return mDetector.onTouchEvent(ev) || super.dispatchTouchEvent(ev);
}
```


Long press to dismiss

Part 2

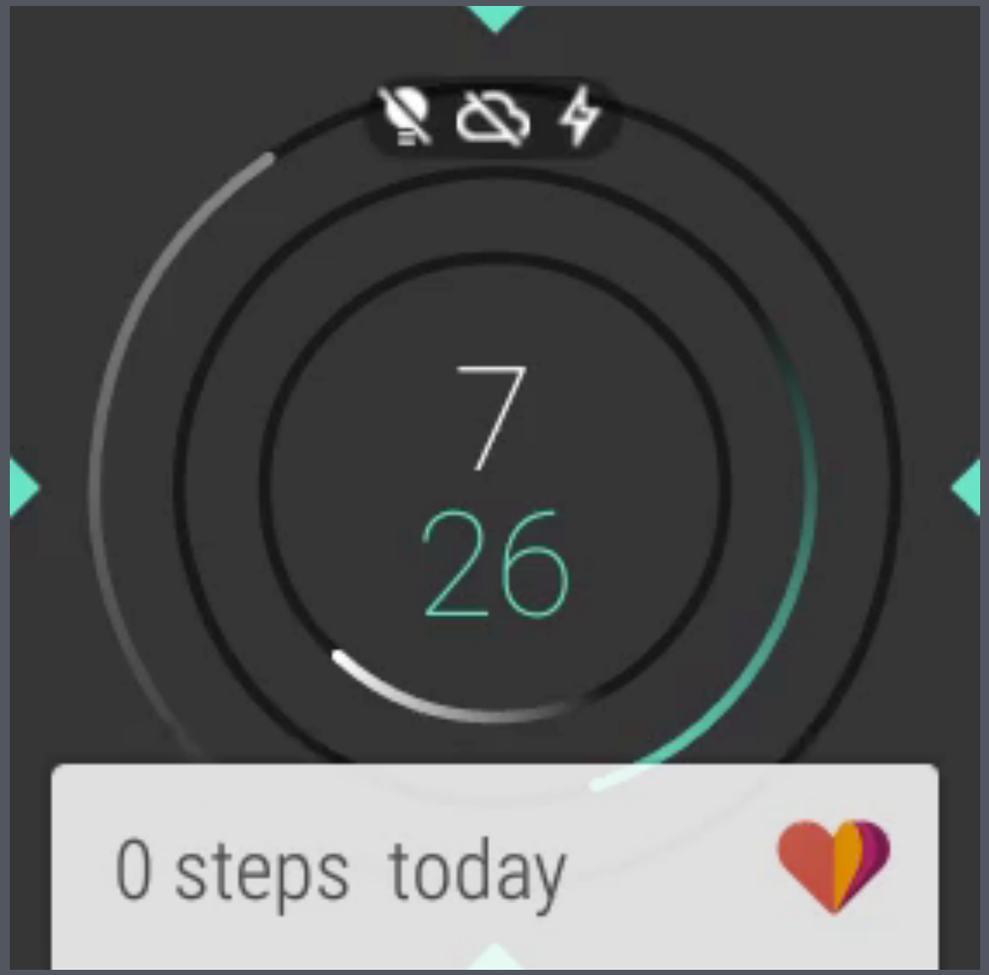
Custom Watch Faces

Outline

1. The Basics
2. Customization

Modes

Interactive



Ambient

The Basics

Permissions

```
<manifest ...>
    <uses-permission
        android:name="com.google.android.permission.PROVIDE_BACKGROUND" />
    <uses-permission
        android:name="android.permission.WAKE_LOCK" />
    ...
</manifest>
```

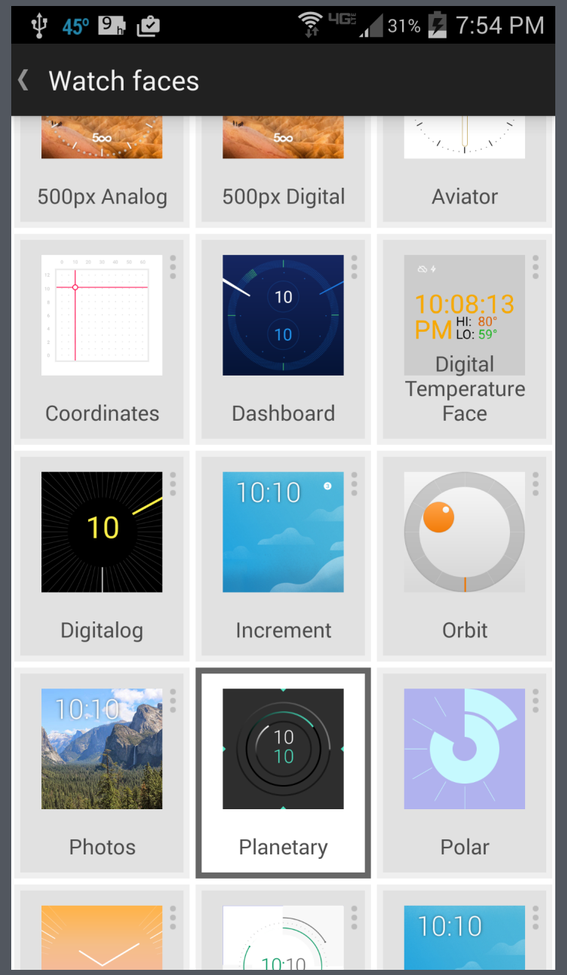
Needs to be added to manifest of the both the mobile and wearable app

Registering the service

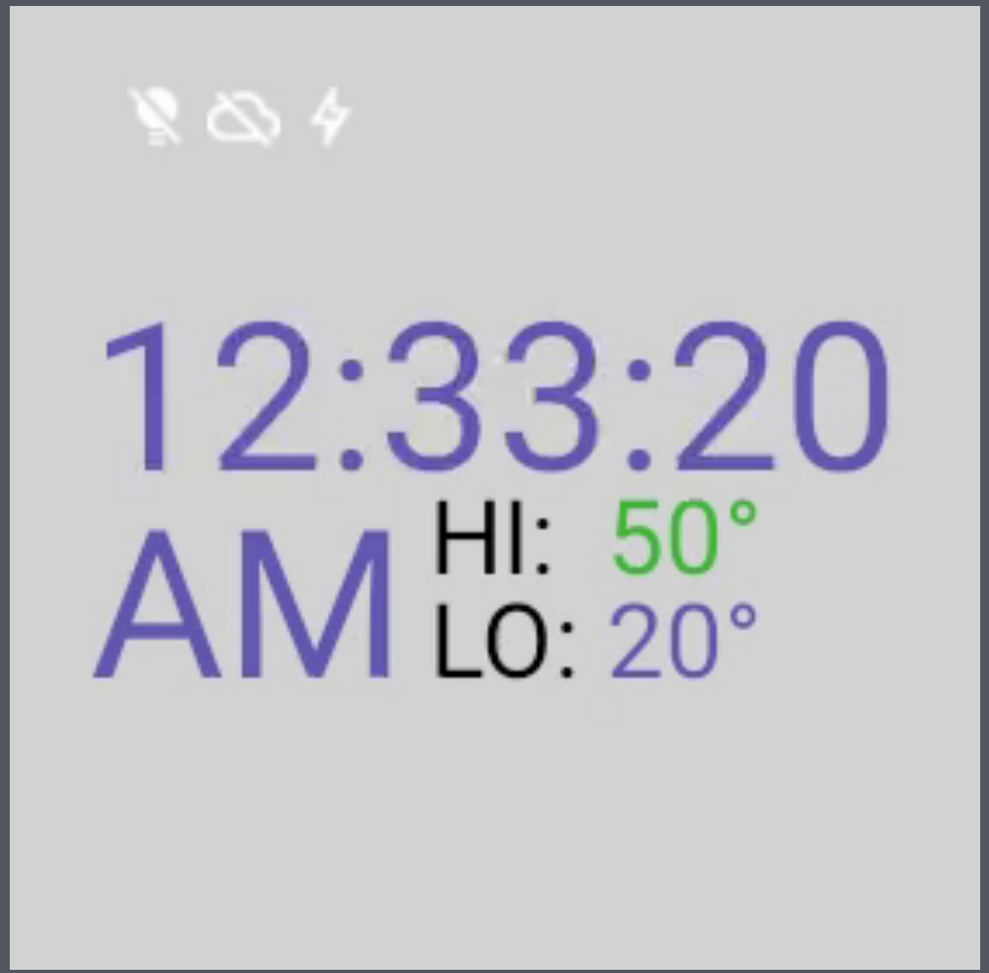
Place in the wearable app's manifest

```
<service
    android:name=".WatchFaceService"
    android:label="@string/watch_name"
    android:allowEmbedded="true"
    android:taskAffinity=""
    android:permission="android.permission.BIND_WALLPAPER" >
    <meta-data
        android:name="android.service.wallpaper"
        android:resource="@xml/watch_face" />
    <meta-data
        android:name="com.google.android.wearable.watchface.preview"
        android:resource="@drawable/watch_preview" />
    <meta-data
        android:name="com.google.android.wearable.watchface.preview_circular"
        android:resource="@drawable/watch_preview_circular" />
    <intent-filter>
        <action android:name="android.service.wallpaper.WallpaperService" />
        <category
            android:name=
            "com.google.android.wearable.watchface.category.WATCH_FACE" />
    </intent-filter>
</service>
```

Watch face previews



Demo watch face



CanvasWatchFaceService

```
public class TemperatureWatchFaceService extends CanvasWatchFaceService {
    ...
    @Override
    public Engine onCreateEngine() {
        return new Engine();
    }
    ...
}
```

CanvasWatchFaceService.Engine

```
private class Engine extends CanvasWatchFaceService.Engine {
    @Override
    public void onCreate(SurfaceHolder holder) {
        super.onCreate(holder);
    }

    @Override
    public void onPropertiesChanged(Bundle properties) {
       super.onPropertiesChanged(properties);
    }

    @Override
    public void onTimeTick() {
        super.onTimeTick();
    }

    @Override
    public void onAmbientModeChanged(boolean inAmbientMode) {
       super.onAmbientModeChanged(inAmbientMode);
    }

    @Override
    public void onDraw(Canvas canvas, Rect bounds) {

    }

    @Override
    public void onVisibilityChanged(boolean visible) {
       super.onVisibilityChanged(visible);
    }
}
```

onCreate()

- Configure the system UI
- Create styles and colors
- Load and save any resources

```
@Override
public void onCreate(SurfaceHolder holder) {
    super.onCreate(holder);

    // Setup system UI
    setWatchFaceStyle(new WatchFaceStyle.Builder(TemperatureWatchFaceService.this)
            .setCardPeekMode(WatchFaceStyle.PEEK_MODE_SHORT)
            .setBackgroundVisibility(WatchFaceStyle.BACKGROUND_VISIBILITY_INTERRUPTIVE)
            .setShowSystemUiTime(false)
            .build());

    // Load resources (strings, dimensions, colors, images)
    Resources resources = TemperatureWatchFaceService.this.getResources();
    mAmbientBackgroundColor = resources.getColor(R.color.ambient_background);
    ...

    // Setup paints
    mBackgroundPaint = new Paint();
    mBackgroundPaint.setColor(mInteractiveBackgroundColor);

    mTimePaint = new Paint();
    mTimePaint.setColor(mInteractiveDefaultColor);
    mTimePaint.setAntiAlias(true);
    mTimePaint.setTextSize(resources.getDimension(R.dimen.text_size));
    ...

    // Load time
    mTime = new Time();
    ...

    // Start timer
    updateTimer();
}
```

Custom timer

```
...

private void updateTimer() {
    mUpdateTimeHandler.removeMessages(MSG_UPDATE_TIME);
    if (shouldTimerBeRunning()) {
        mUpdateTimeHandler.sendEmptyMessage(MSG_UPDATE_TIME);
    }
}

...

private final Handler mUpdateTimeHandler = new Handler() {
    @Override
    public void handleMessage(Message message) {
        switch (message.what) {
            case MSG_UPDATE_TIME:
                // Invalidate to indicate we want to redraw
                invalidate();

                if (shouldTimerBeRunning()) {
                    long timeMs = System.currentTimeMillis();

                    // Trigger on the next half-second
                    long delayMs = UPDATE_TIME_RATE_MS - (timeMs % UPDATE_TIME_RATE_MS);
                    mUpdateTimeHandler.sendEmptyMessageDelayed(MSG_UPDATE_TIME, delayMs);
                }
                break;
        }
    }
};

...
```

onPropertiesChanged()

Here is where we detect whether the wearable supports low-bit ambient and burn-in protection

```
@Override
public void onPropertiesChanged(Bundle properties) {
    super.onPropertiesChanged(properties);
    mLowBitAmbient = properties.getBoolean(PROPERTY_LOW_BIT_AMBIENT, false);

    // Also possible to get whether burn-in protection is turned on
}
```

onTimeTick()

Called once a minute when in ambient mode

```
@Override
public void onTimeTick() {
    super.onTimeTick();

    // Invalidate to update the watch
    invalidate();
}
```

onAmbientModeChanged()

```
@Override
public void onAmbientModeChanged(boolean inAmbientMode) {
    super.onAmbientModeChanged(inAmbientMode);

    // Set text and background paint colors
    if (inAmbientMode) {
        mTimePaint.setColor(mAmbientTextColor);
        ...
    } else {
        setTimePaintColor(mTimePaint, mCurrentTemperature);
        ...
        mBackgroundPaint.setColor(mInteractiveBackgroundColor);
    }

    // Set antialiasing
    if (mLowBitAmbient) {
        mTimePaint.setAntiAlias(!inAmbientMode);
        ...
    }

    invalidate();
}
```

onVisibilityChanged()

```
@Override
public void onVisibilityChanged(boolean visible) {
    super.onVisibilityChanged(visible);

    if (visible) {
        registerReceiver();

        // Update time zone
        mTime.clear(TimeZone.getDefault().getID());
        mTime.setToNow();
    } else {
        unregisterReceiver();
    }

    updateTimer();
}

private void registerReceiver() {
    if (mRegisteredTimeZoneReceiver) {
        return;
    }
    mRegisteredTimeZoneReceiver = true;
    IntentFilter filter = new IntentFilter(Intent.ACTION_TIMEZONE_CHANGED);
    TemperatureWatchFaceService.this.registerReceiver(mTimeZoneReceiver, filter);
}

private void unregisterReceiver() {
    if (!mRegisteredTimeZoneReceiver) {
        return;
    }
    mRegisteredTimeZoneReceiver = false;
    TemperatureWatchFaceService.this.unregisterReceiver(mTimeZoneReceiver);
}
```

onDraw()

```
@Override
public void onDraw(Canvas canvas, Rect bounds) {
    mTime.setToNow();
    boolean shouldDrawColon = (System.currentTimeMillis() % 1000) < 500;

    // Draw the background.
    canvas.drawRect(0, 0, bounds.width(), bounds.height(), mBackgroundPaint);

    String time = "";
    if (isInAmbientMode()) {
        time = mTime.format("%I:%M");
    } else if (shouldDrawColon) {
        time = mTime.format("%I:%M:%S");
    } else {
        time = mTime.format("%I %M %S");
    }

    canvas.drawText(time, mXOffset, mYOffset, mTimePaint);

    String amPm = mTime.format("%p");
    canvas.drawText(amPm, mXOffset, mYOffset + mLineHeight, mTimePaint);

    // Draw temperature related strings
    ...
}
```

Other data sources

Customization

1. System UI options
2. User configuration options

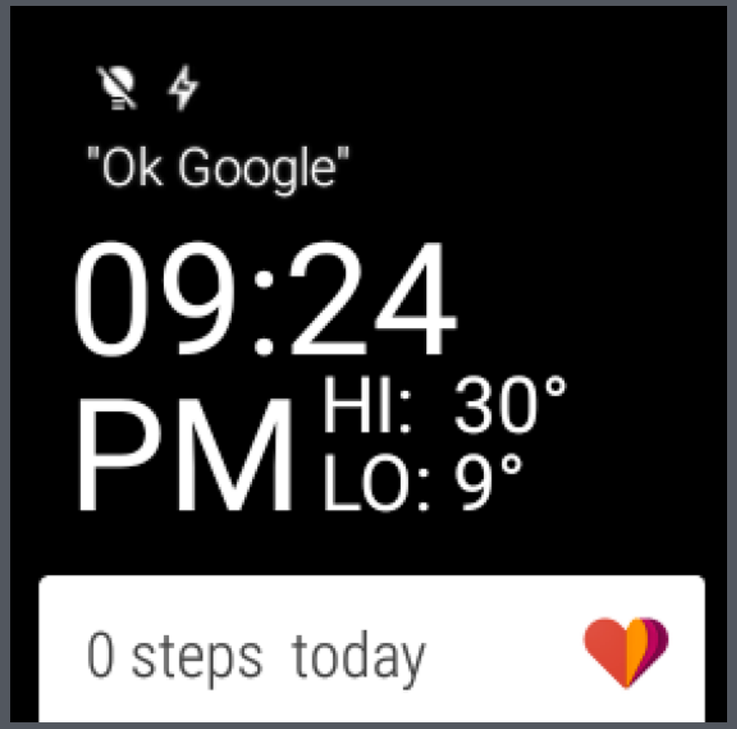
System UI options

From the demo:

```
setWatchFaceStyle(new WatchFaceStyle.Builder(TemperatureWatchFaceService.this)
        .setCardPeekMode(WatchFaceStyle.PEEK_MODE_SHORT)
        .setBackgroundVisibility(WatchFaceStyle.BACKGROUND_VISIBILITY_INTERRUPTIVE)
        .setShowSystemUiTime(false)
        .build());
```

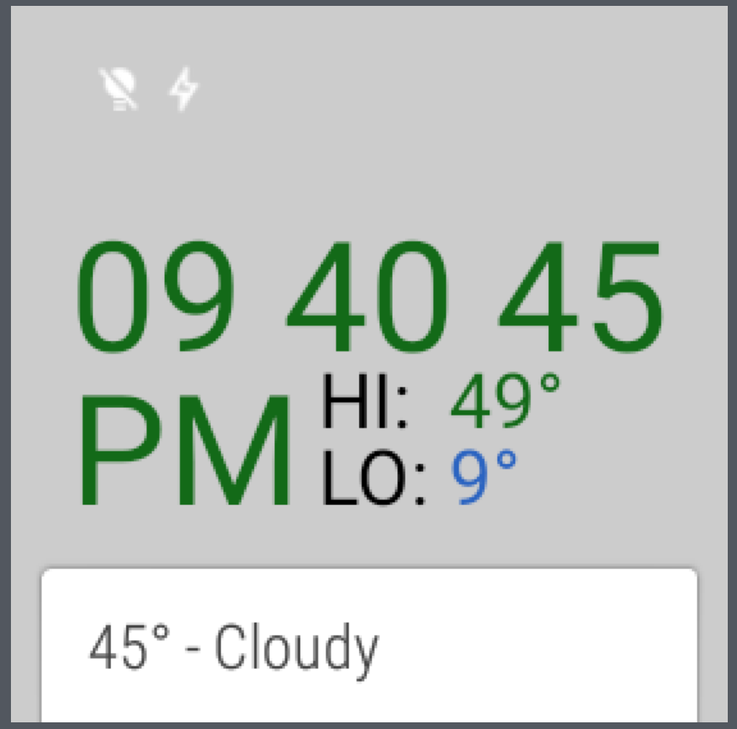
setAmbientPeekMode()

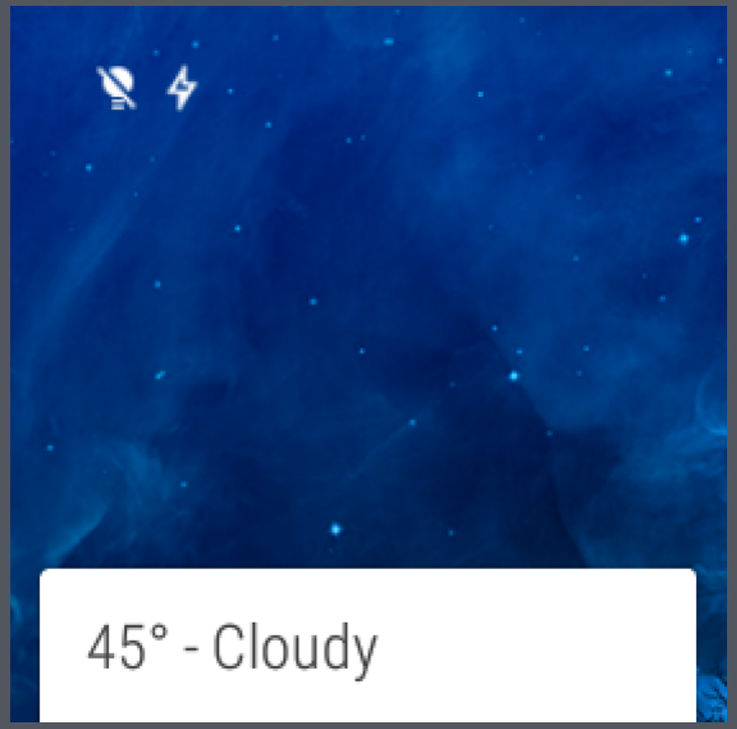
must be either `AMBIENT_PEEK_MODE_VISIBLE` or `AMBIENT_PEEK_MODE_HIDDEN`



setBackgroundVisibility()

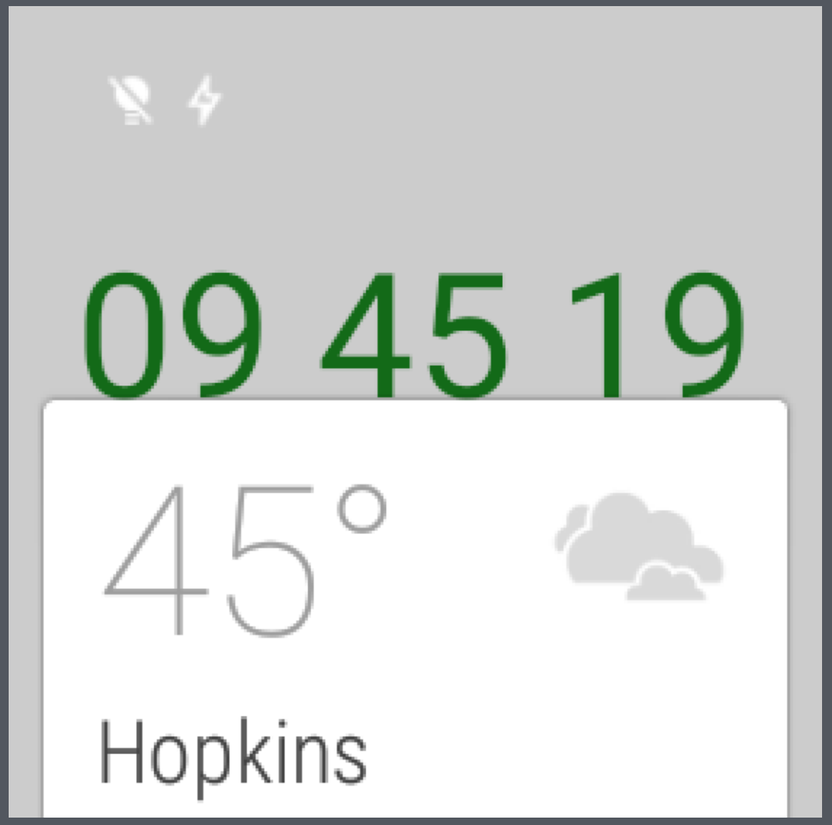
must be either `BACKGROUND_VISIBILITY_INTERRUPTIVE` or `BACKGROUND_VISIBILITY_PERSISTENT`

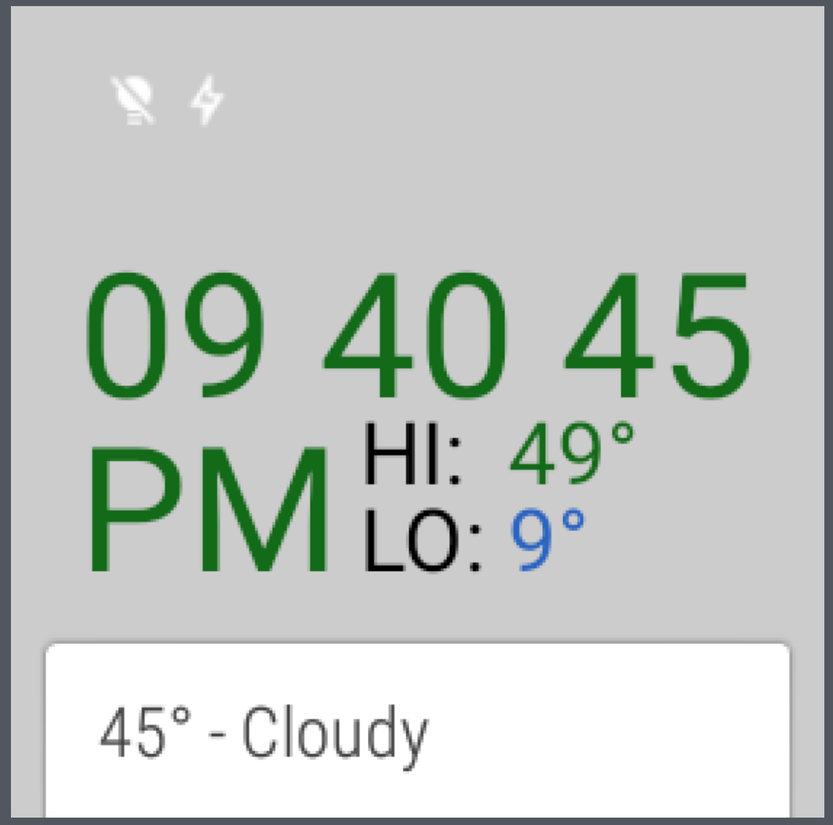




setCardPeekMode()

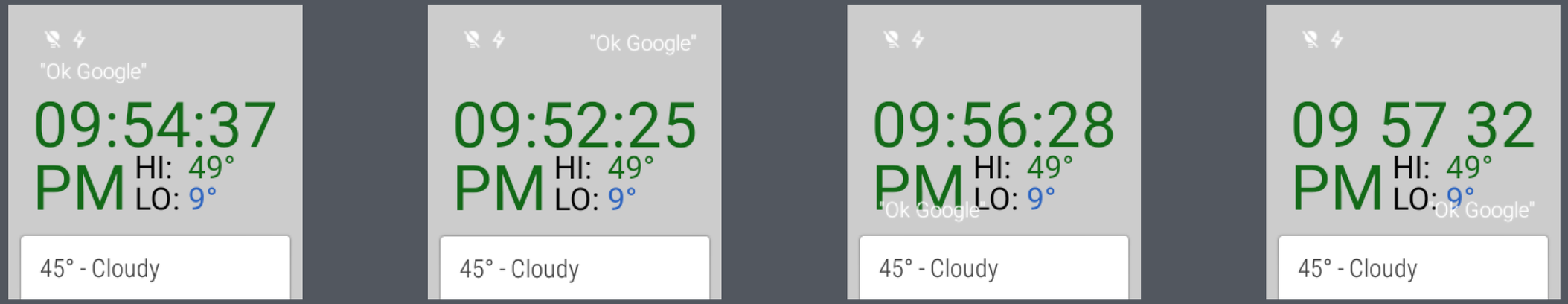
must be either `PEEK_MODE_VARIABLE` or `PEEK_MODE_SHORT`





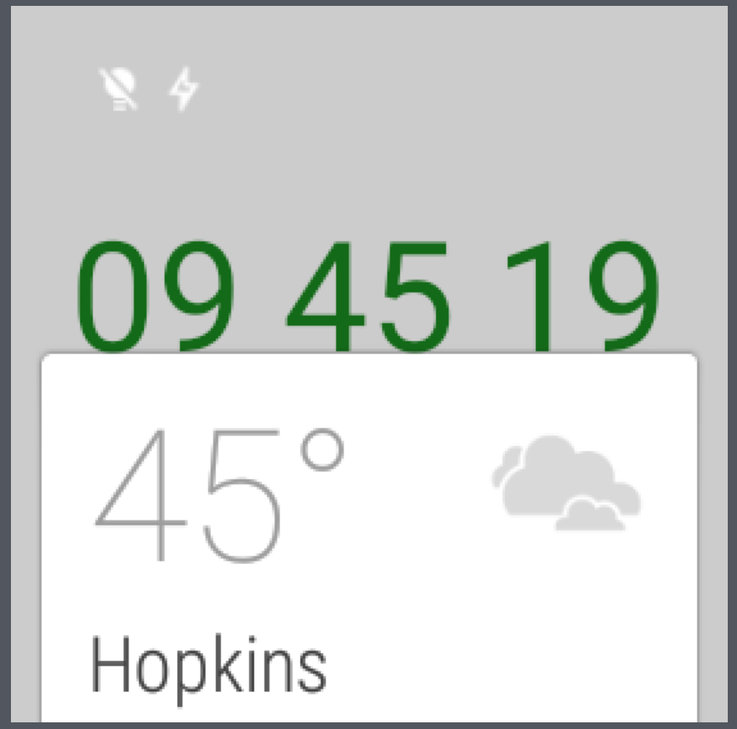
setHowWordIndicatorGravity()

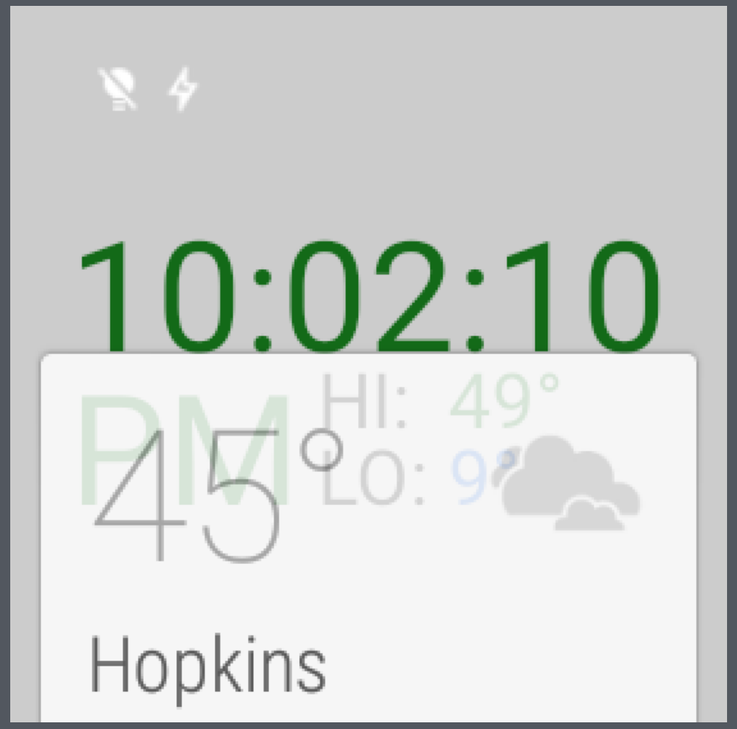
This must be any combination of horizontal Gravity constant (`LEFT`, `CENTER_HORIZONTAL`, `RIGHT`) and vertical Gravity constants (`TOP`, `CENTER_VERTICAL`, `BOTTOM`), e.g. `Gravity.LEFT | Gravity.BOTTOM`. On circular screens, only the vertical gravity is respected.



setPeekOpacityMode()

must be either `PEEK_OPACITY_MODE_OPAQUE` or `PEEK_OPACITY_MODE_TRANSLUCENT`





setStatusBarGravity()

This must be any combination of horizontal Gravity constant (`LEFT`, `CENTER_HORIZONTAL`, `RIGHT`) and vertical Gravity constants (`TOP`, `CENTER_VERTICAL`, `BOTTOM`), e.g. `Gravity.LEFT | Gravity.BOTTOM`. On circular screens, only the vertical gravity is respected.

setViewProtection()

must be any combination of `PROTECT_STATUS_BAR`, `PROTECT_HOTWORD_INDICATOR` and `PROTECT_WHOLE_SCREEN`

setShowSystemUiTime()

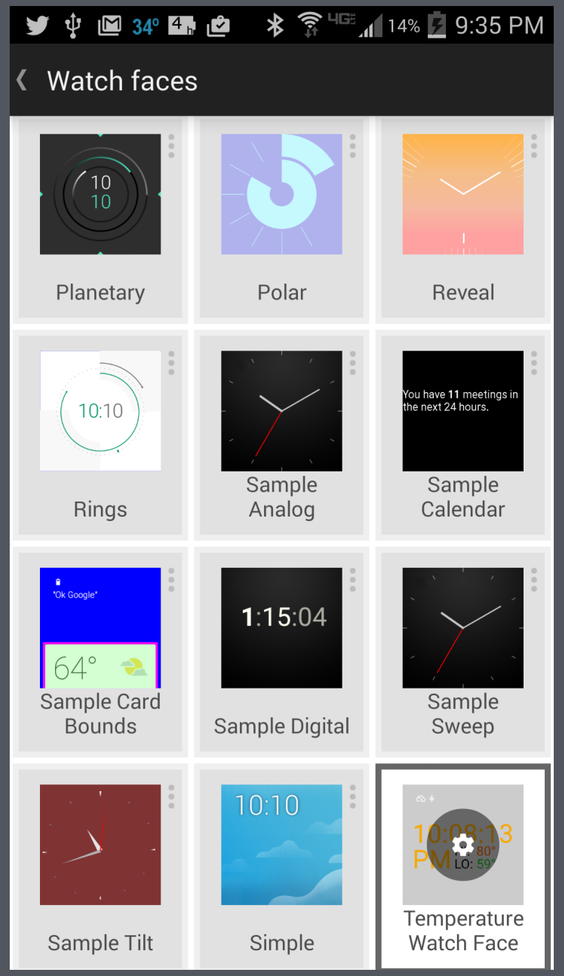
Set this to false if you already draw or clearly represent the time on your watch face.

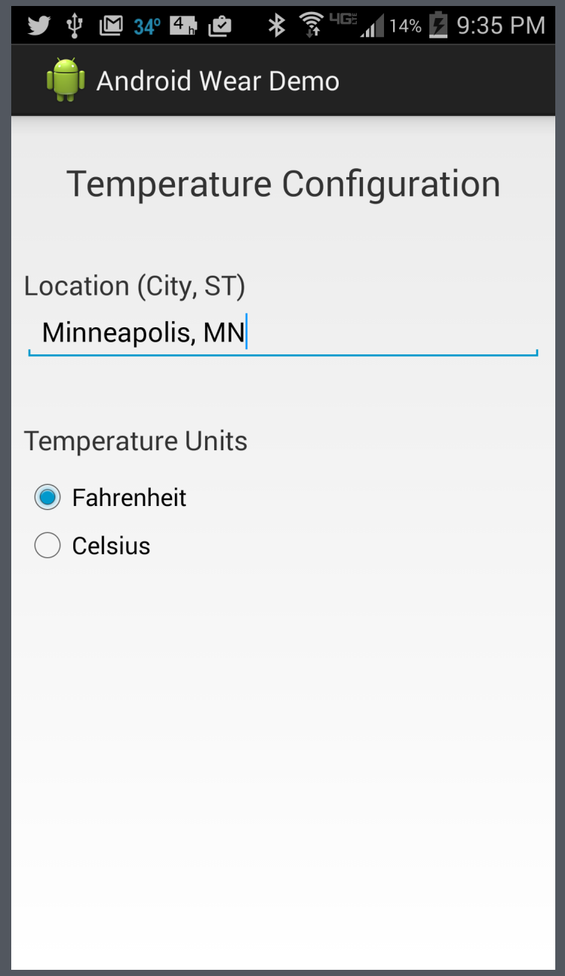
setShowUnreadCountIndicator()

if true an indicator will be shown

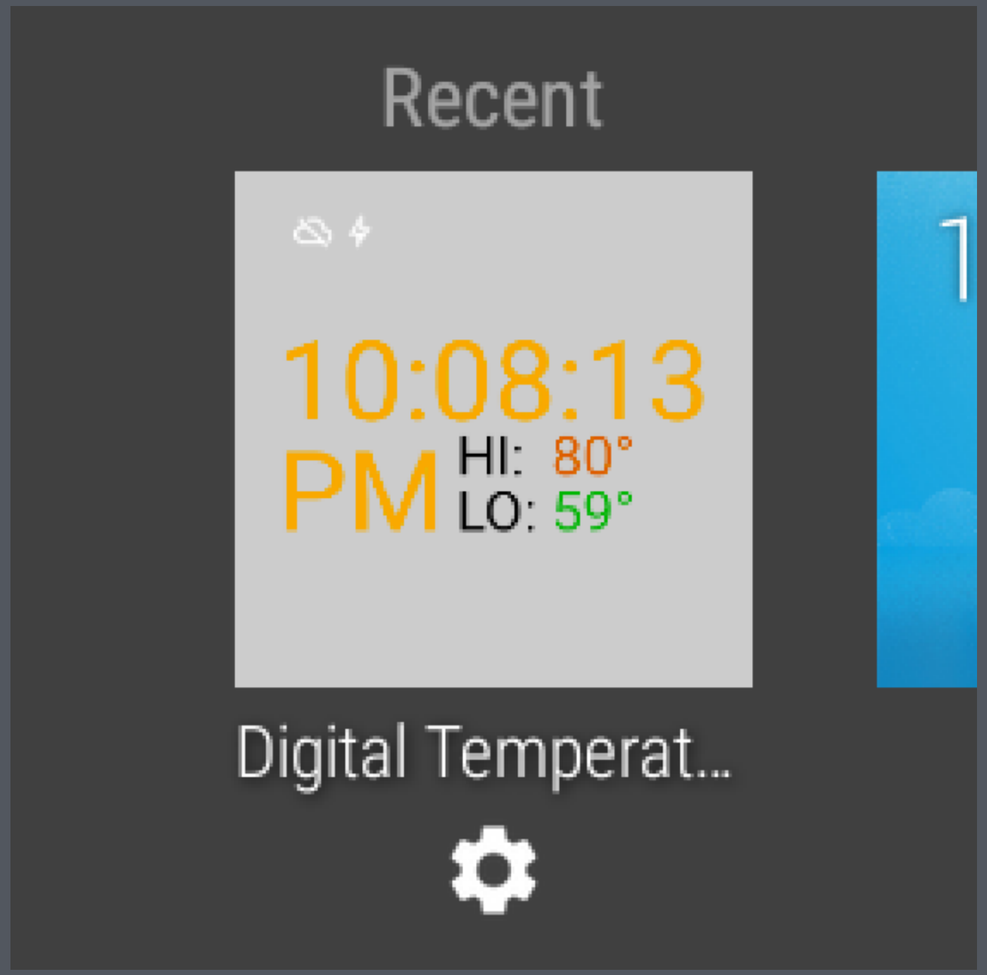
User configuration options

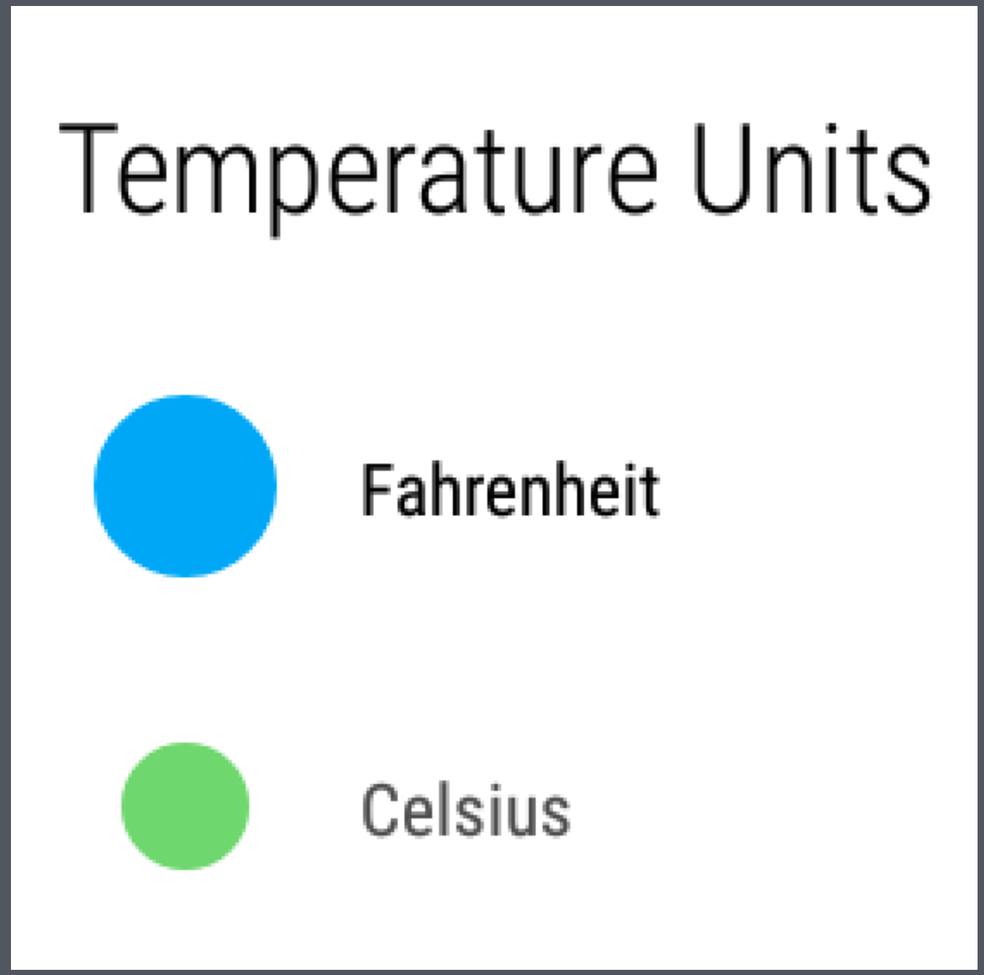
Companion configuration activity





Wearable configuration activity





Resources

https://developer.android.com/training/wearables/watch-faces/index.html

http://developer.android.com/samples/WatchFace/index.html

Beyond Notifications

Creating ***apps*** for Android Wear

Brian Yencho
Sean Weiser

https://github.com/livefront/android-wear-demo