Custom Views for Profit and Pleasure

October 18, 2014

Dan Lew

Custom Views

(as viewed by me, years ago)

Image source: http://goo.gl/JLdYHW

Framework Views are huge!

View	Lines
View	20,992
TextView	9,477
ImageView	1,423
ViewGroup	7,285
LinearLayout	1,931
RelativeLayout	1,822
AdapterView	1,212
AbsListView	7,314
ListView	3,905

(Data as of API 21)

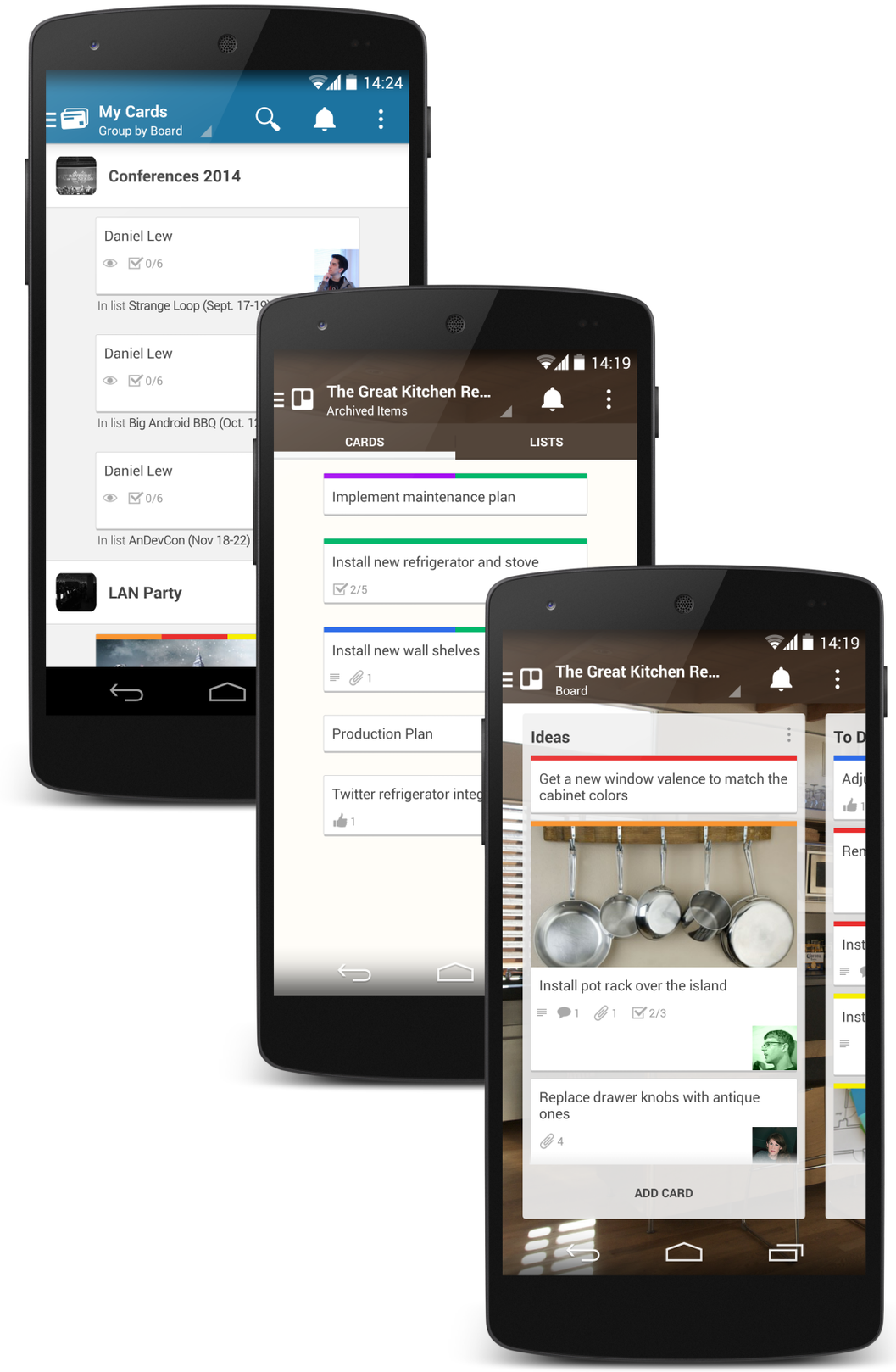
Too Many Methods!

Category	Methods
Creation	Constructors
	onFinishInflate()
Layout	onMeasure()
	onLayout()
Drawing	onSizeChanged()
	onDraw()
Event processing	onKeyDown()
	onKeyUp()
	onTrackballEvent()
	onTouchEvent()
Focus	onFocusChanged()
	onWindowFocusChanged()
Attaching	onAttachedToWindow()
	onDetachedFromWindow()
	onWindowVisibilityChanged()

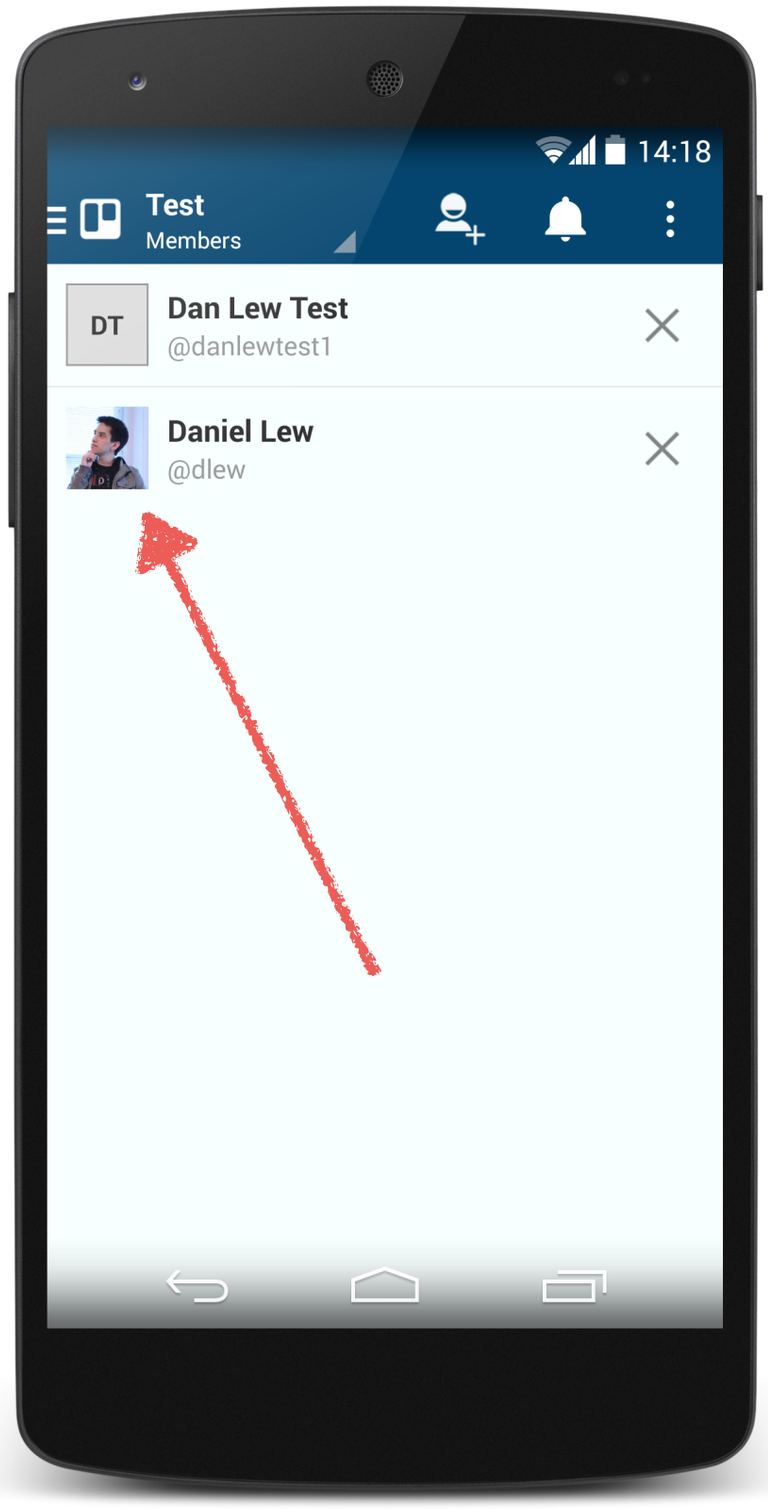
Library Views
!=
Custom Views

Why custom Views?

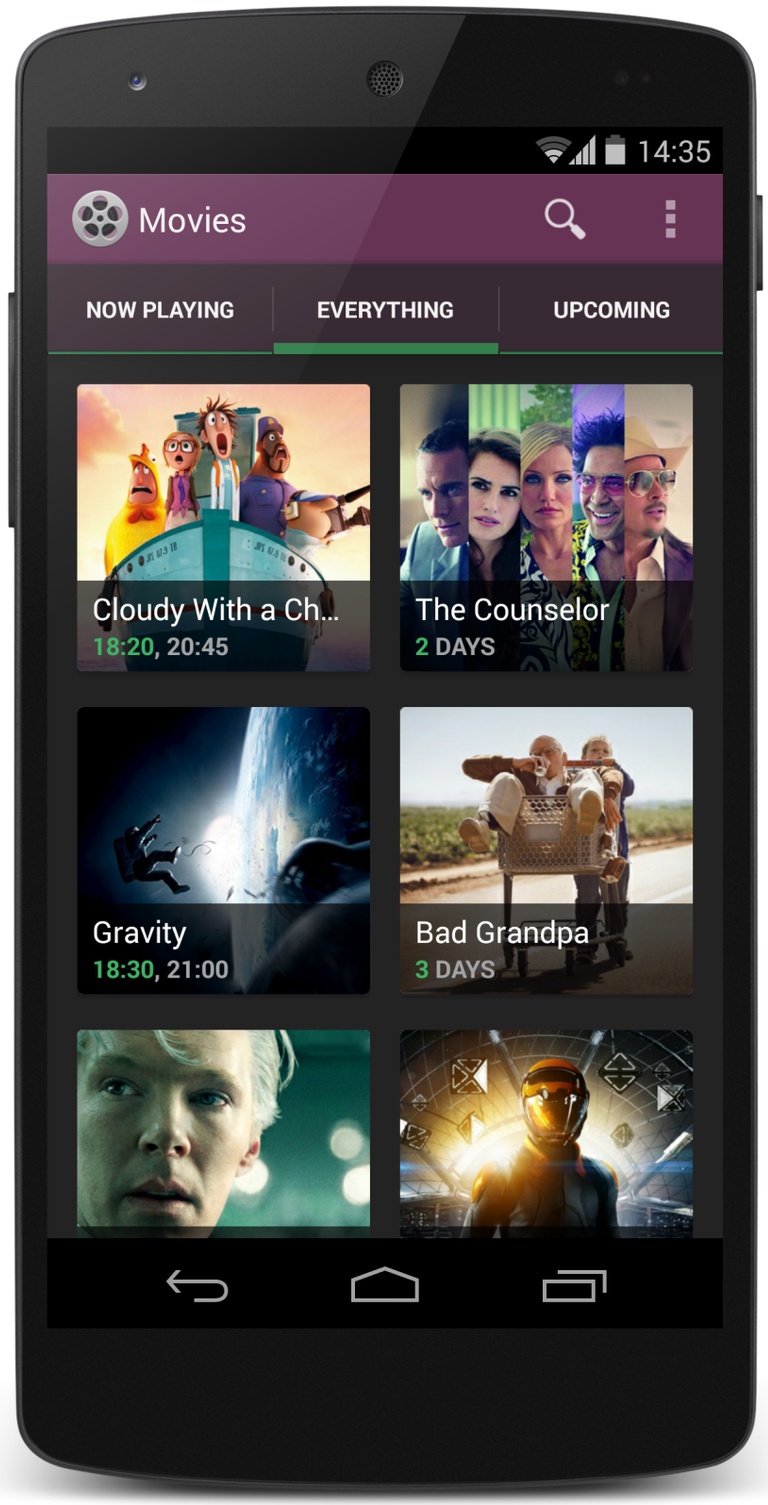
- View reuse


14:24
My Cards
Group by Board
Conferences 2014
Daniel Lew
0/6
In list Strange Loop (Sept. 17-19
Daniel Lew
0/6
In list Big Android BBQ (Oct. 1
Daniel Lew
0/6
In list AnDevCon (Nov 18-22)
LAN Party
14:19
The Great Kitchen Re...
Archived Items
CARDS
LISTS
Implement maintenance plan
Install new refrigerator and stove
2/5
Install new wall shelves
1
Production Plan
Twitter refrigerator integ
1
14:19
The Great Kitchen Re...
Board
Ideas
Get a new window valence to match the cabinet colors
Install pot rack over the island
1
1
2/3
Replace drawer knobs with antique ones
4
ADD CARD
To D

- View reuse
- Encapsulation



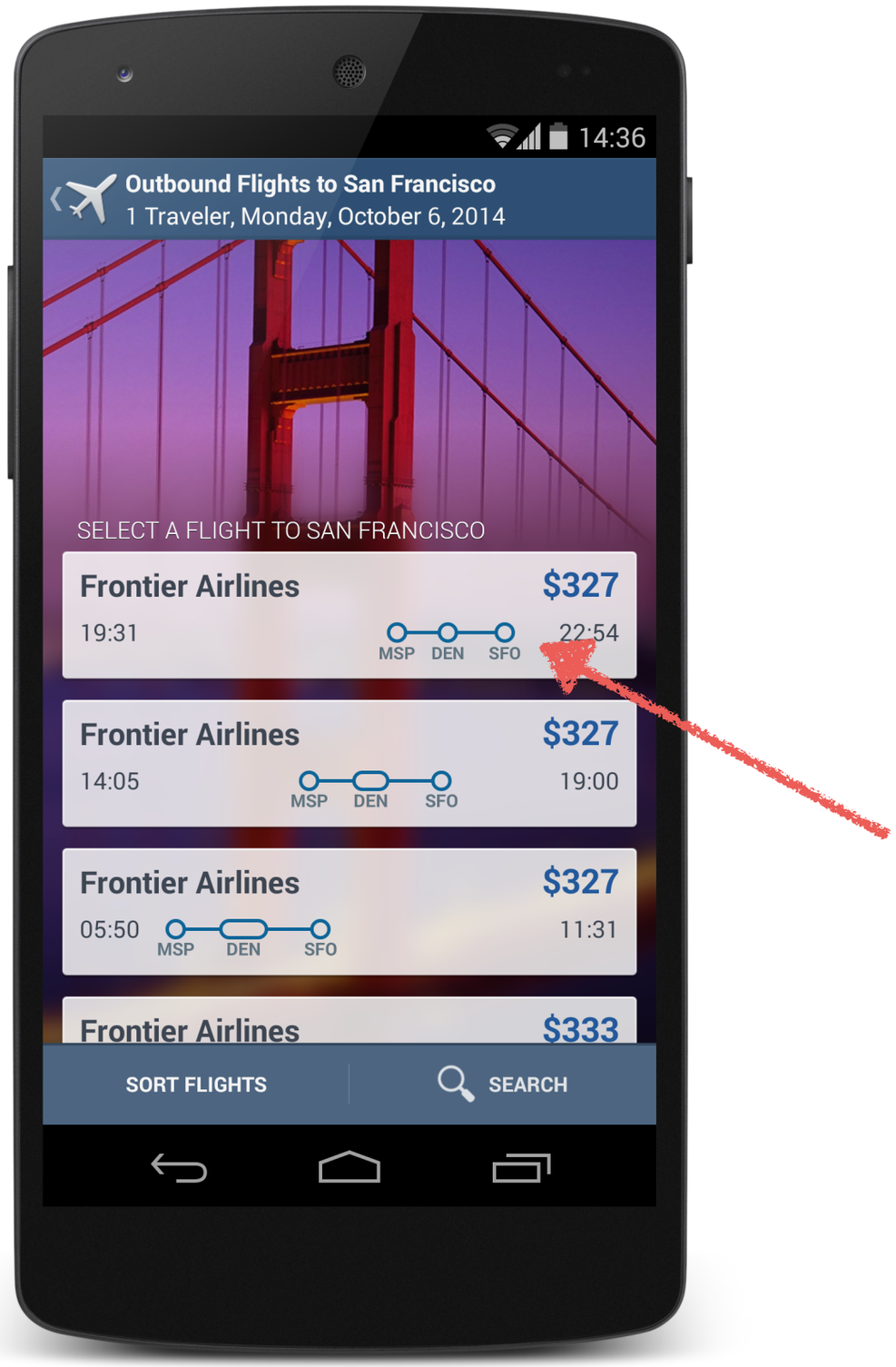
- View reuse
- Encapsulation
- Compound Views



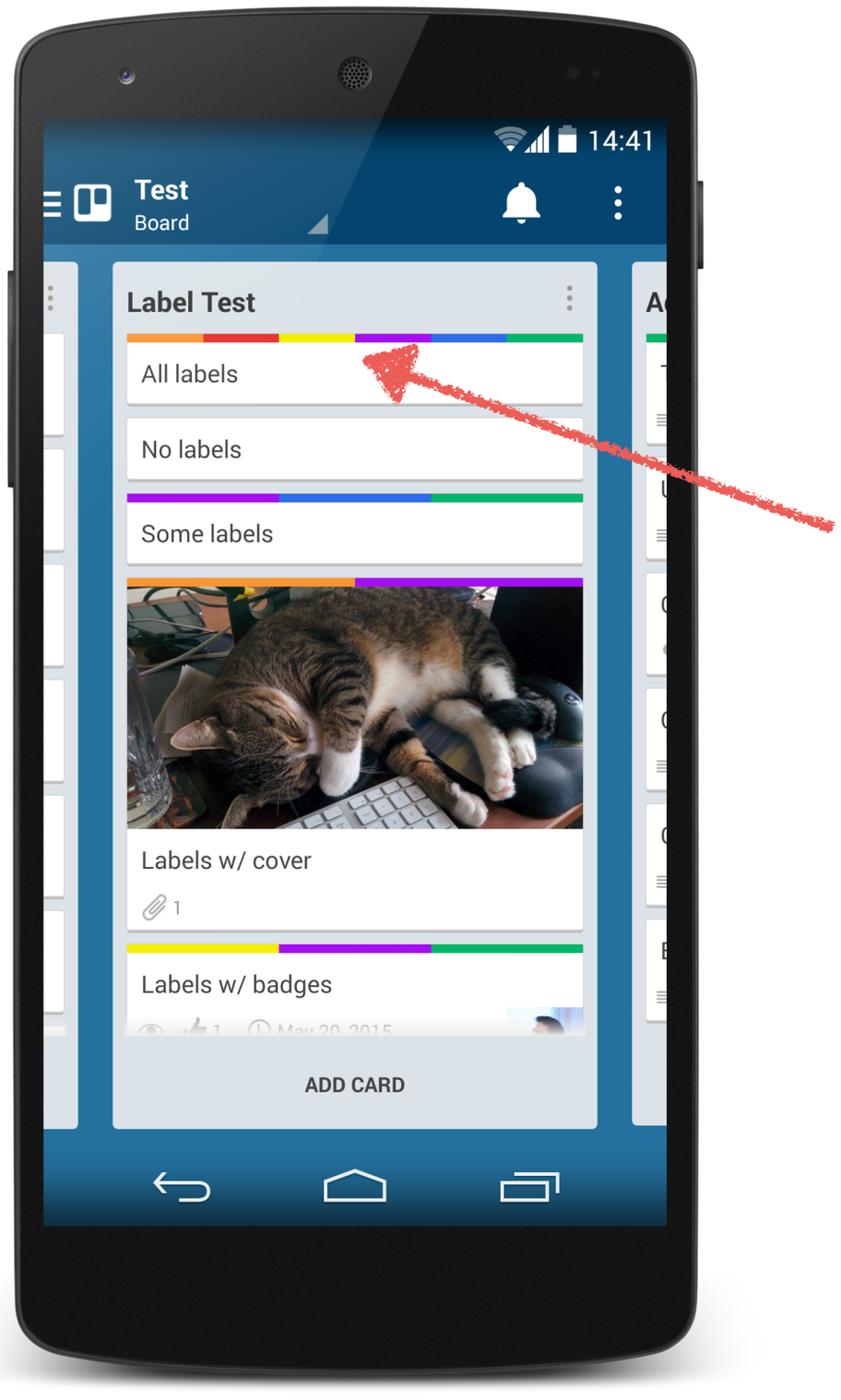
- View reuse
- Encapsulation
- Compound Views
- XML styling

```
<net.danlew.app.FontTextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:font="comicSans" />

<net.danlew.app.FontTextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
app:font="papyrus" />
```

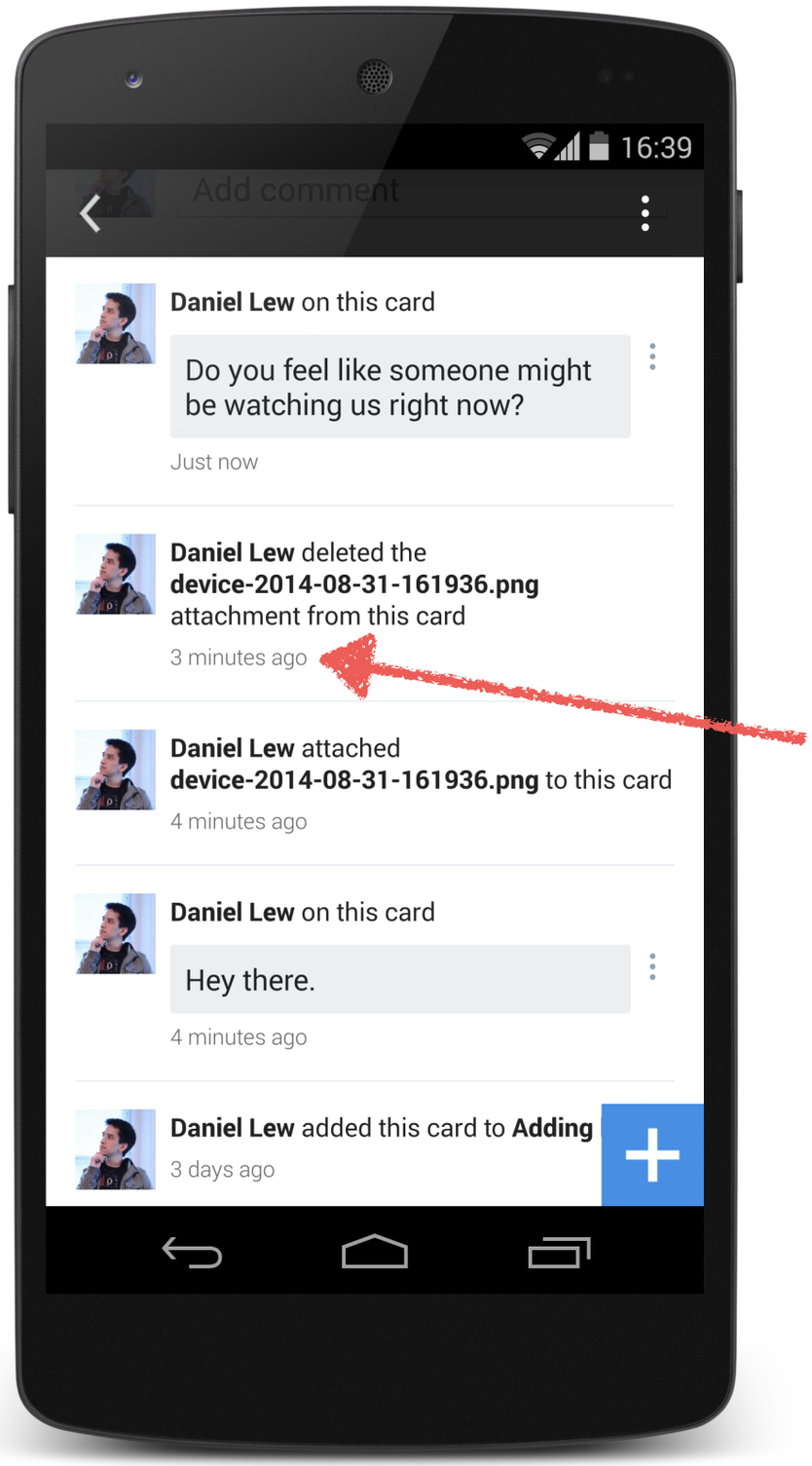
- View reuse
- Encapsulation
- Compound Views
- XML styling
- Custom drawing



- View reuse
- Encapsulation
- Compound Views
- XML styling
- Custom drawing
- Performance



- View reuse
- Encapsulation
- Compound Views
- XML styling
- Custom drawing
- Performance
- Awesomeness



A Simple Custom View

Step 1: Subclass

```
public class CustomView extends View {



}
```

Step 2: Constructor

```
public class CustomView extends View {
    public CustomView(Context context) {
        super(context);
    }
}
```

Step 3: Add

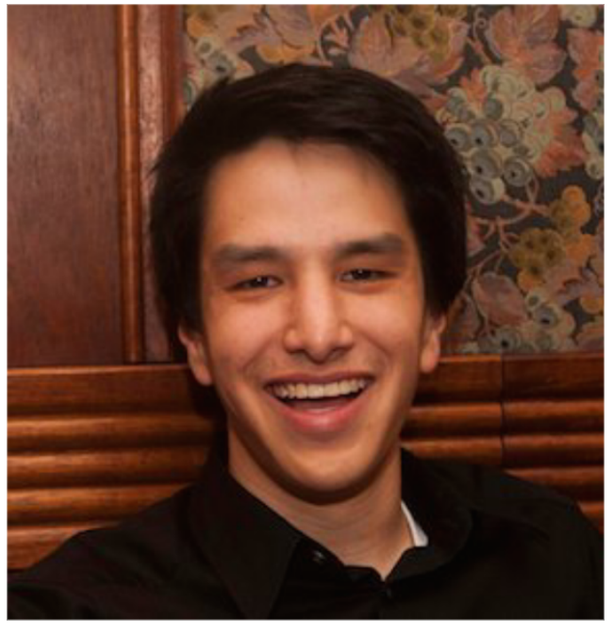
```
CustomView customView = new CustomView(context);
viewGroup.addView(customView);
```

You're done!

Dan Lew

UserView

Our brave illustrative warrior



Dan Lew

Without Custom Views

Without Custom Views

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/icon"
        android:layout_width="128dp"
        android:layout_height="128dp" />

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:textSize="22sp" />

</LinearLayout>
```

Without Custom Views

```
View userView = LayoutInflater.from(context)
    .inflate(R.layout.user_view, parent, false);

TextView nameView = (TextView) view.findViewById(R.id.name);
nameView.setText(user.getName());

ImageView iconView = (ImageView) view.findViewById(R.id.icon);
iconView.setImageResource(user.getIcon());
```

Compound Views

```
public class UserView extends LinearLayout {

    private ImageView mIconView;
    private TextView mNameView;

    public UserView(Context context) {
        super(context);

        setOrientation(LinearLayout.VERTICAL);
        setGravity(Gravity.CENTER);

        LayoutInflater.from(context).inflate(R.layout.user_view_merge, this);
        mIconView = (ImageView) findViewById(R.id.icon);
        mNameView = (TextView) findViewById(R.id.name);
    }

    public void setIcon(int drawable) {
        mIconView.setImageResource(drawable);
    }

    public void setName(CharSequence name) {
        mNameView.setText(name);
    }
}
```

Compound Views

```
<merge>

    <ImageView
        android:id="@+id/icon"
        android:layout_width="128dp"
        android:layout_height="128dp" />

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:textSize="22sp" />

</merge>
```

Compound Views

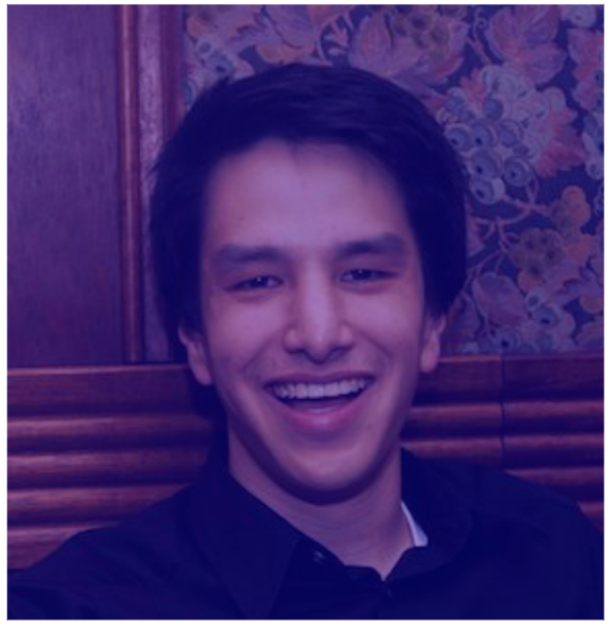
```
UserView userView = new UserView(context);
userView.setName(user.getName());
userView.setIcon(user.getIcon());
```

Encapsulation

```
public class UserView extends LinearLayout {
    /* ...same code as before... */

    public void bind(User user) {
        mIconView.setImageResource(user.getIcon());
        mNameView.setText(user.getName());
    }
}
```

Encapsulation

```
UserView userView = new UserView(this);
userView.bind(user);
```



Dan Lew

XML Styling

XML Styling

```
<resources>
    <declare-styleable name="UserView">
        <attr name="tint" format="color" />
    </declare-styleable>
</resources>
```

XML Styling

```
<net.danlew.customviews.view.UserView
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
app:tint="#500F" />
```

Reading Styles in Code

```
public class UserView extends LinearLayout {

    public UserView(Context context, AttributeSet attrs) {
        super(context, attrs);

        /* ...Same constructor code as before... */

        TypedArray ta = context.obtainStyledAttributes(attrs,
            R.styleable.UserView);
        setTint(ta.getColor(R.styleable.UserView_tint,
            Color.TRANSPARENT));
        ta.recycle();
    }

    public void setTint(int color) {
        mIconView.setColorFilter(color);
    }
}
```

Note on Constructors

- Code constructor:

```
public View(Context context)
```

- XML constructor:

```
public View(Context context, AttributeSet attrs)
```

- XML w/ styled defaults (rarely necessary):

```
public View(Context context, AttributeSet attrs, int defStyleAttr)
public View(Context context, AttributeSet attrs, int defStyleAttr,
                  int defStyleRes)
```

Code + XML Constructors

```
public class UserView extends LinearLayout {

    public UserView(Context context) {
        this(context, null);
    }

    public UserView(Context context, AttributeSet attrs) {
        super(context, attrs);

        LayoutInflater.from(context).inflate(R.layout.user_view_merge, this);

        // Etc...
    }
}
```

Dan Lew

Custom Drawing

Custom Drawing

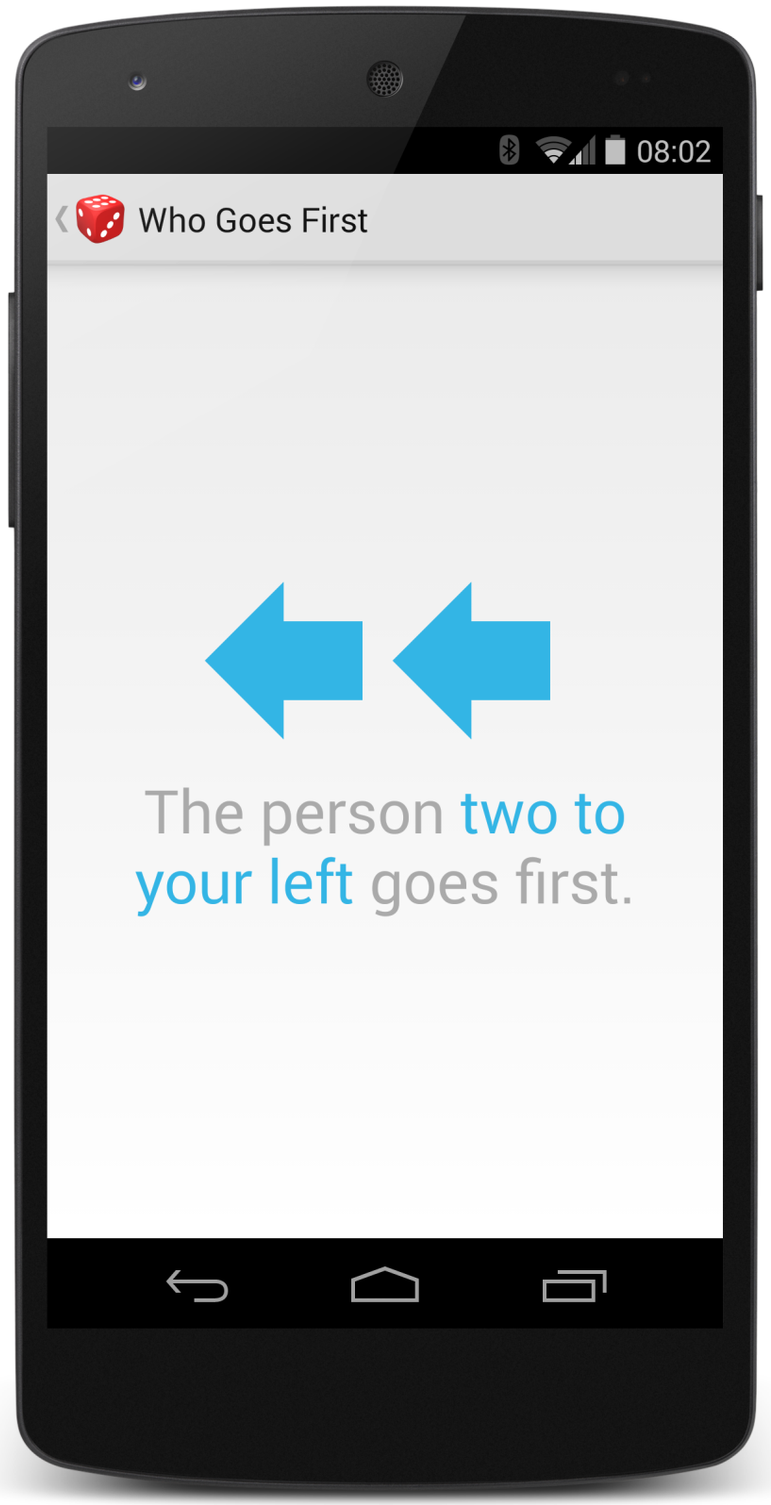
```
public class CircleView extends View {

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        /*
          ...Configure paint here to draw icon...
          ...yadda yadda yadda...
        */

        int radius = getWidth() / 2;
        canvas.drawCircle(radius, radius, radius, paint);
    }
}
```

yadda: http://goo.gl/W8d17V

Advanced Practices

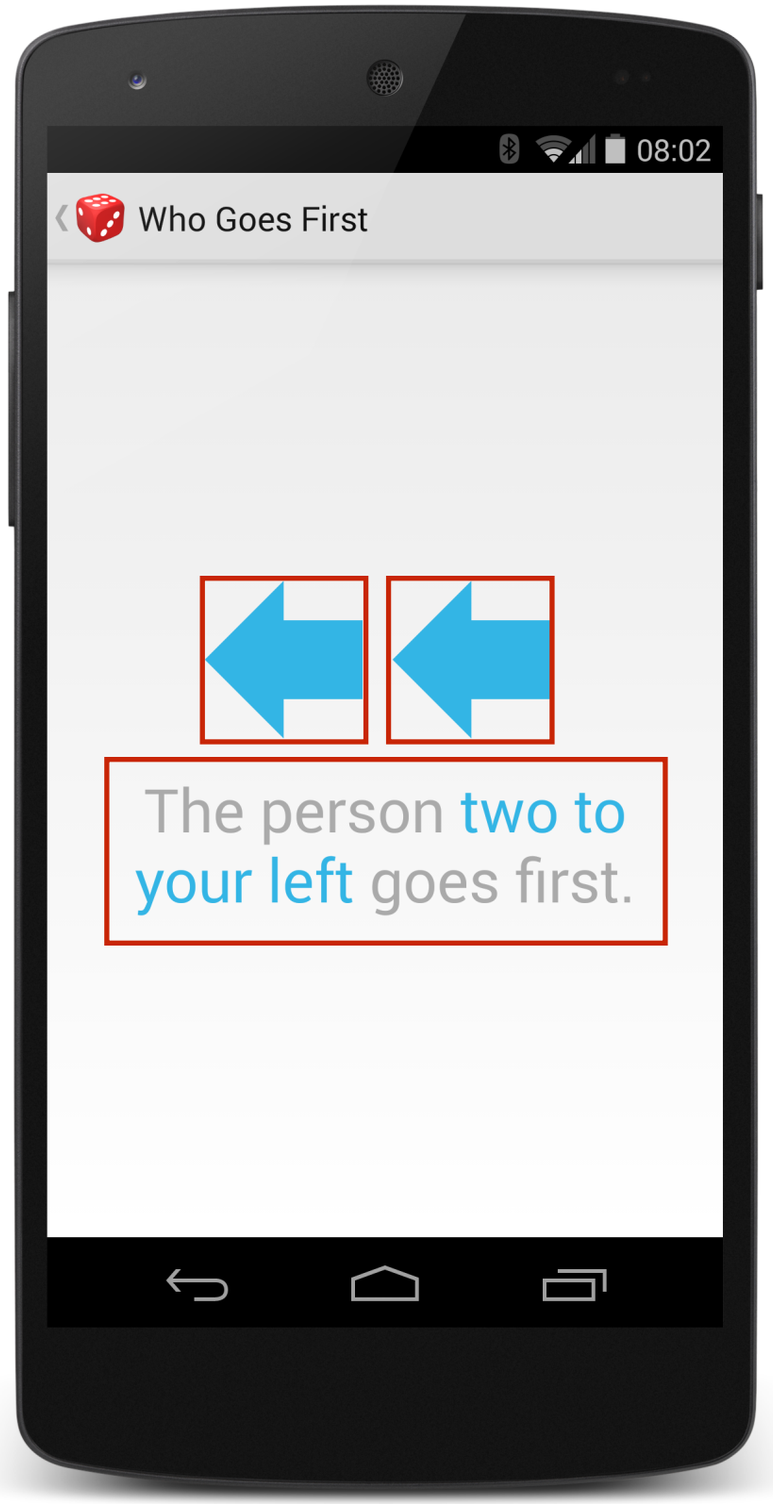
- Measure / Layout
- Saving state
- Touch events
- Custom animations

Measurement vs. Layout



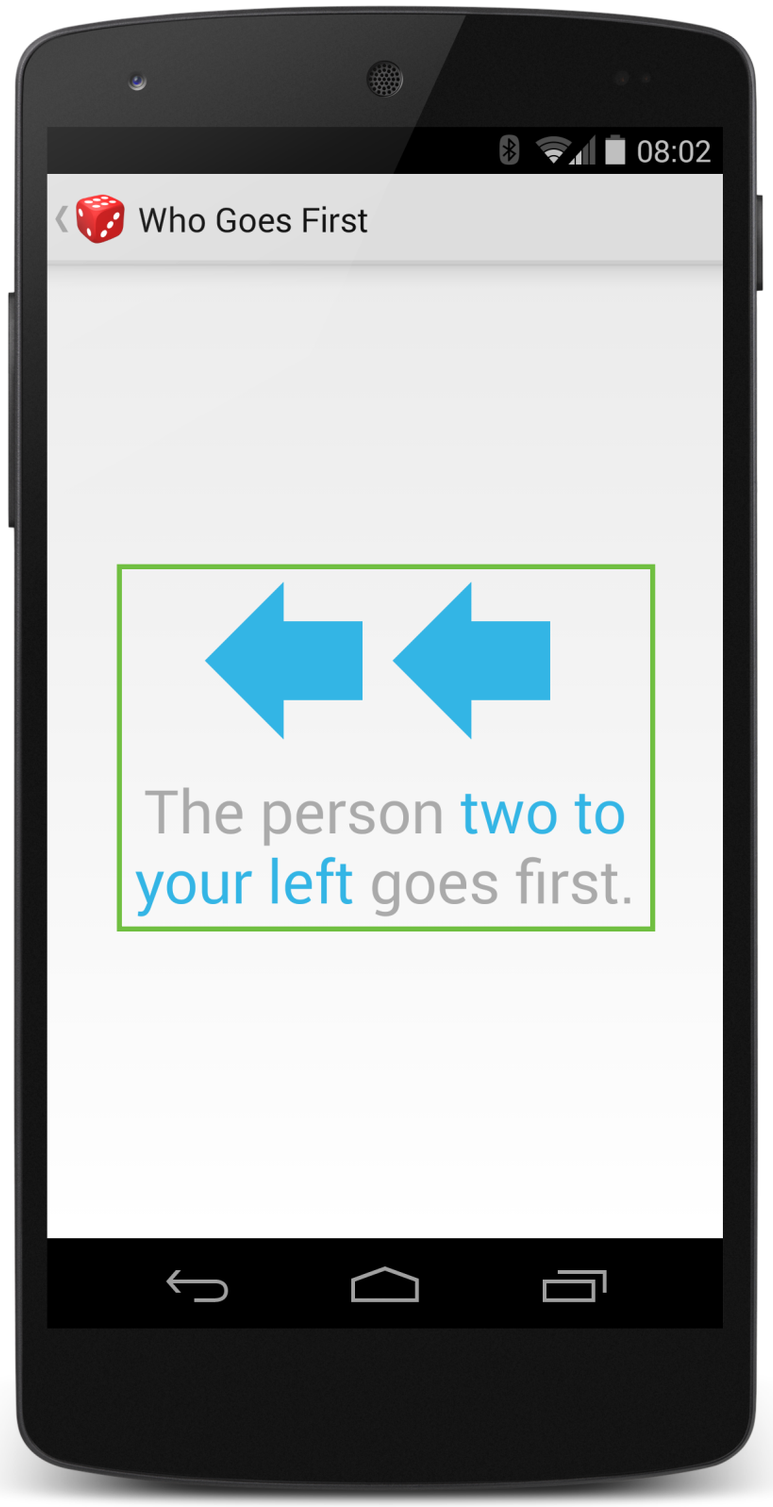
onMeasure()

View size



onLayout()

Child View location/size



Measurement and Layout Pro-Tip

Don’t bother!

(most of the time)

onMeasure()

- onMeasure() signature

```
void onMeasure(int widthMeasureSpec, int heightMeasureSpec)
```

- measureSpec - packed integer

```
int widthMode = MeasureSpec.getMode(widthMeasureSpec);
int widthSize = MeasureSpec.getSize(widthMeasureSpec);
int heightMode = MeasureSpec.getMode(heightMeasureSpec);
int heightSize = MeasureSpec.getSize(heightMeasureSpec);
```

Measure Modes

- MeasureSpec.EXACTLY - Must be that size
- MeasureSpec.AT_MOST - Maximum width
- MeasureSpec.UNDEFINED - Ideal width

onMeasure()

```
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int widthMode = MeasureSpec.getMode(widthMeasureSpec);
    int widthSize = MeasureSpec.getSize(widthMeasureSpec);

    int width;
    if (widthMode == MeasureSpec.EXACTLY) {
        width = widthSize;
    }
    else {
        int desiredWidth = 500; // Whatever calculation you want

        if (widthMode == MeasureSpec.AT_MOST) {
            width = Math.min(desiredWidth, widthSize);
        }
        else {
            width = desiredWidth;
        }
    }

    // ...to be continued...
}
```

onMeasure()

```
@Override
protected void onMeasure(int widthMeasureSpec,
        int heightMeasureSpec) {
    int width;
    int height;

    // ...Calculate width and height...

    setMeasuredDimension(width, height);
}
```

onLayout()

- Seriously, don’t bother!

- For each child, call: child.layout()

onLayout()

```
@Override
protected void onLayout(boolean changed, int l, int t,
        int r, int b) {

    for (int a = 0; a < getChildCount(); a++) {
        View child = getChildAt(a);

        // ...Calculate left, top, right, bottom…
        // (This should be easy, right?)

        child.layout(left, top, right, bottom);
    }

}
```

Detecting Size Changes

- Update config based on size
- onLayout()
- onSizeChanged()

Saving State

- Encapsulate state in Views
- onSaveInstanceState() / onRestoreInstanceState()
- Uses Parcelables
- Parcelables - harder (but fast)
- Bundles - easy (but a bit slower)

Saving State

```
private String mValue;

@Override
protected Parcelable onSaveInstanceState() {
    Bundle bundle = new Bundle();
    bundle.putString("Key", mValue);
    return bundle;
}

@Override
protected void onRestoreInstanceState(Parcelable state) {
    Bundle bundle = (Bundle) state;
    mValue = bundle.getString("Key");
}
```

Parent State

- View parent has state, too!
- Must wrap/unwrap parent state
- Still simple with Bundles

Parent State

```
@Override
protected Parcelable onSaveInstanceState() {
    Bundle bundle = new Bundle();
    bundle.putParcelable("SuperState", super.onSaveInstanceState());
    // ...Put whatever you want in the Bundle...
    return bundle;
}

@Override
protected void onRestoreInstanceState(Parcelable state) {
    Bundle bundle = (Bundle) state;
    super.onRestoreInstanceState(bundle.getParcelable("SuperState"));
    // ...Restore whatever you put into the Bundle...
}
```

Touch Events

- Hard mode: onTouchEvent()
- Easy mode: GestureDetector
- Handles complex logic (e.g. touch slop)
- If you want to know everything, Dave Smith Talk: http://goo.gl/CtzHaN

GestureDetector

- Step 1: Create OnGestureListener

```
private OnGestureListener mListener = new SimpleOnGestureListener() {

    @Override
    public boolean onDoubleTap(MotionEvent e) {
        Log.i("Tag", "Double tap detected!");
        return true;
    }

};
```

GestureDetector

- Step 2: Create GestureDetector

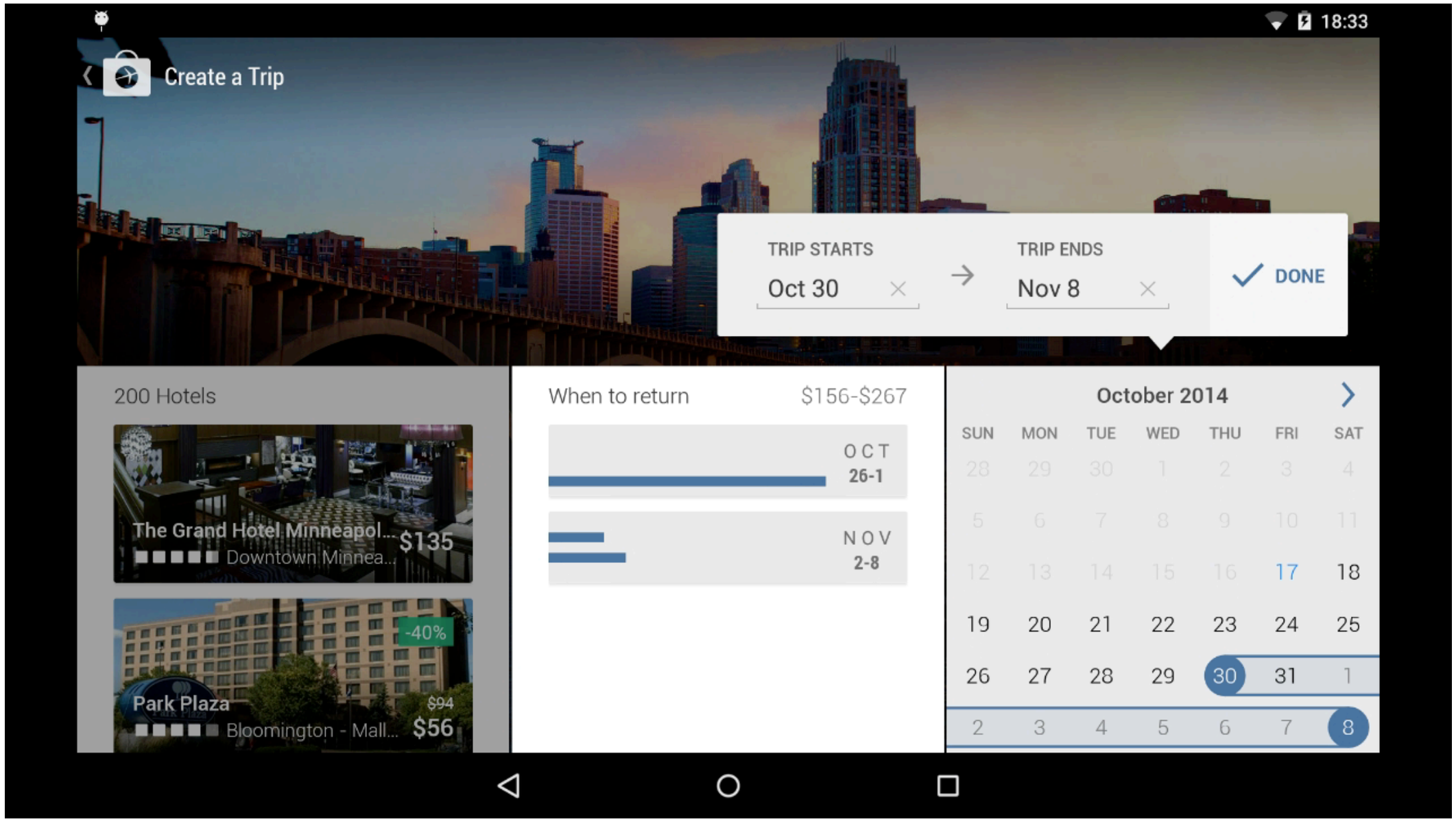
```
public class CustomView extends View {

    private GestureDetector mGestureDetector;

    public CustomView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mGestureDetector = new GestureDetector(context, mListener);
    }

}
```

GestureDetector

- Step 3: Hook up GestureDetector in onTouchEvent()

```
public class CustomView extends View {
    // ...Previous code...

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        return mGestureDetector.onTouchEvent(event)
            || super.onTouchEvent(event);
    }

}
```

Custom Animation



Custom Draw Property

```
public class ColorView extends View {
    private int mColor;

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawColor(mColor);
    }
}
```

What I Want

```
// Animate the alpha
ObjectAnimator.ofFloat(myView, "alpha", 1.0f).start();

// Change color as it fades in
ObjectAnimator.ofInt(myView, "color", Color.RED).start();
```

Property Setter/Getter

```
public class ColorView extends View {

    private int mColor;

    public int getColor() {
        return mColor;
    }

    public void setColor(int color) {
        mColor = color;
        invalidate();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawColor(mColor);
    }
}
```

Invalidate vs. Layout

- invalidate() - View needs to redraw contents
- requestLayout() - View needs to change its size
    - Rubbish for animation
- Animation talk: http://goo.gl/uwRBso
    - …By yours truly, Dan Lew

One downside…

…no custom View composition

Avoid Custom Views

- …when hooks exist already:

```
view.setOnClickListener(new OnClickListener() { ... });
```

- …when you could just use a custom Drawable:

```
imageView.setImageDrawable(new CustomDrawable());
```

See “Mastering Android drawables”: http://goo.gl/ENfzW6

- …when Google is doing tricky stuff (e.g. widget tinting in appcompat)

Summary

- Custom Views can be simple!
- Encapsulating logic makes your code simpler.
- Custom Views can help you fine tune your app.
- …But don’t go crazy with them.

Thank you!

- Samples: https://github.com/dlew/android-custom-views-sample
- http://danlew.net/
- @danlew42
- +DanielLew