Mobile Application Programming: Android

View Measurement
Activities

- Apps are composed of activities
- Activities are self-contained tasks made up of one screen-full of information
- Activities start one another and are destroyed commonly
- Apps can use activities belonging to another app
Android Layout

Creating a Custom Control

- Create subclass of View class
- Override:
  - `onDraw(Canvas c)`
  - `onMeasure(int wMeasure, int hMeasure)`
- Add listener interface and listener property for the interesting events the control generates and call on... methods when events occur
Drawing

- `onDraw(Canvas c)`
  - Call `super` classes’ `onDraw`
  - Clear background (if opaque)
  - Build `Paint` object for draws
  - Make calls to `canvas.draw...()`
  - Determine what data should be made available to `onMeasure` through methods or constants
LinearLayout
I want to wrap my content in height, but be as large as I can be in width.
I want to wrap my content in height, but be as large as I can be in width.

I want to be as large as I can be in width, and be as large as I can be in height.

I want to wrap my content in width, and be wrap my content in height.
How tall do you need to be if you can be at most 800 pixels wide?
How tall do you need to be if you can be at most **800** pixels wide?

How wide do you need to be if you can be at most **400** pixels tall?
You need to be exactly **630** pixels wide. How tall do you need to be now?
You need to be exactly 630 pixels wide. How tall do you need to be now?

How tall do you need to be if you can be at most 230 pixels wide?
LinearLayout

CircleLayout
- Layout(10,30,400,30)
  - View
    - LayoutParams
  - View
    - LayoutParams
  - Layout(10,60,300,30)
    - View
      - LayoutParams
    - Layout(10,30,400,30)

Button
- Layout(10,400,400,300)
  - LayoutParams

FrameLayout
- Layout(10,700,400,100)
  - LayoutParams
  - View
    - LayoutParams
  - View
    - LayoutParams
Measuring

- `onMeasure(int wMeasure, int hMeasure)`
  - Get suggested size with `View.MeasureSpec.getSize`
  - Get mode with `View.MeasureSpec.getMode`
  - Choose size for view respecting mode (view specific)
  - Also respect min using `getSuggestedMinimum`...
  - Call `resolveSize` to ensure `MeasureSpec` is respected
  - Call `setMeasuredDimension` (exception raised if not!)