Incorporating Custom View Interactivity and Configurability

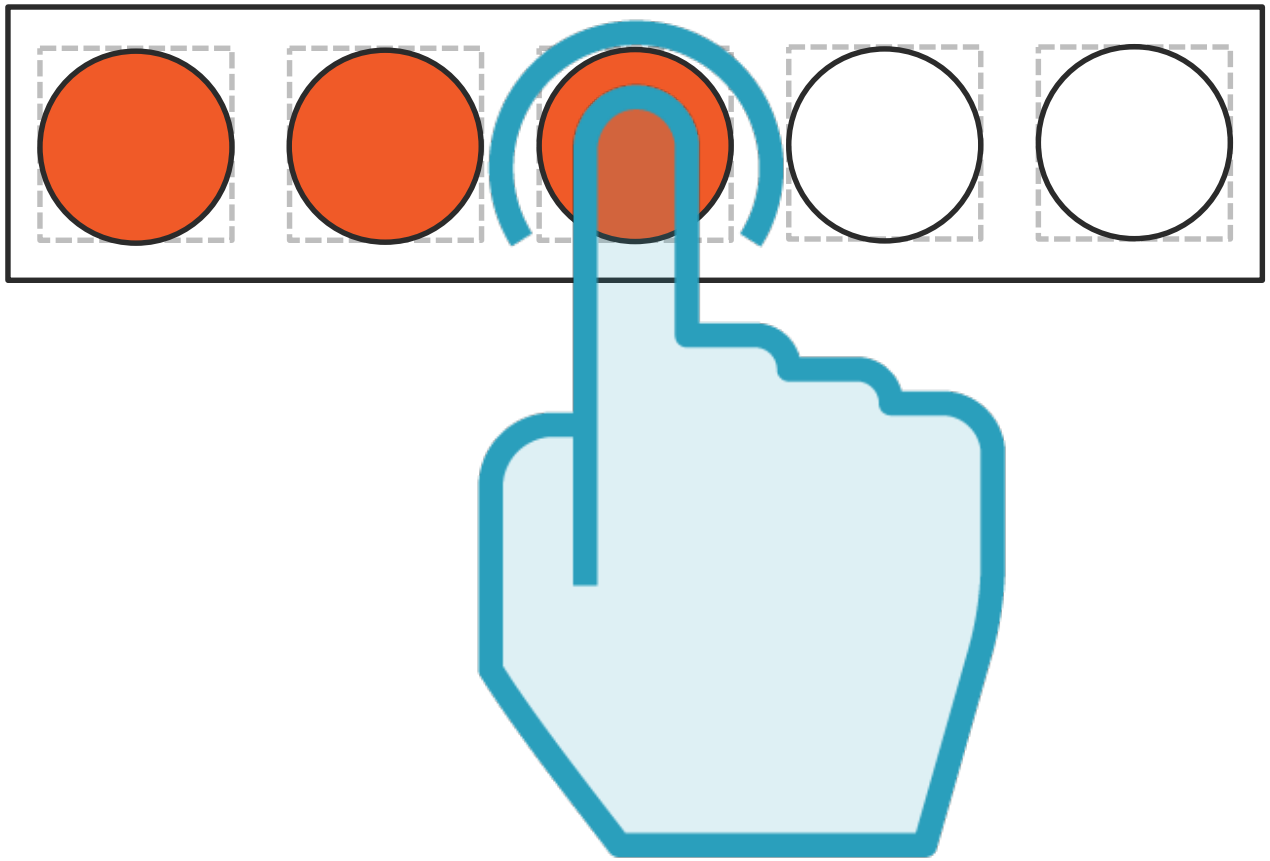
Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim blog.jwhh.com

What to Expect from This Module

- Custom View Interaction from Code
- Adding Custom View Touch Support
- Updating Custom View Drawing
- Custom View Configuration Attributes
- Screen Density Independent Drawing

Adding Touch Support to Our Custom View



Touch Support

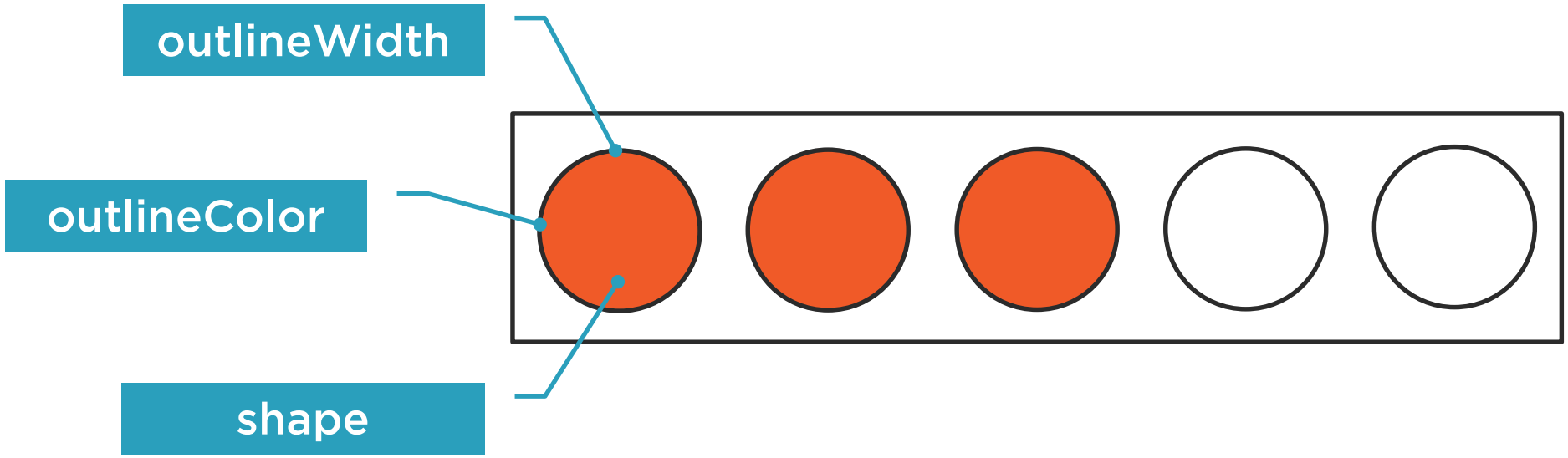
onTouchEvent method

- Override to handle touch actions
- Return true for handled actions
- Call super class for non-handled actions
- Receives a MotionEvent as parameter

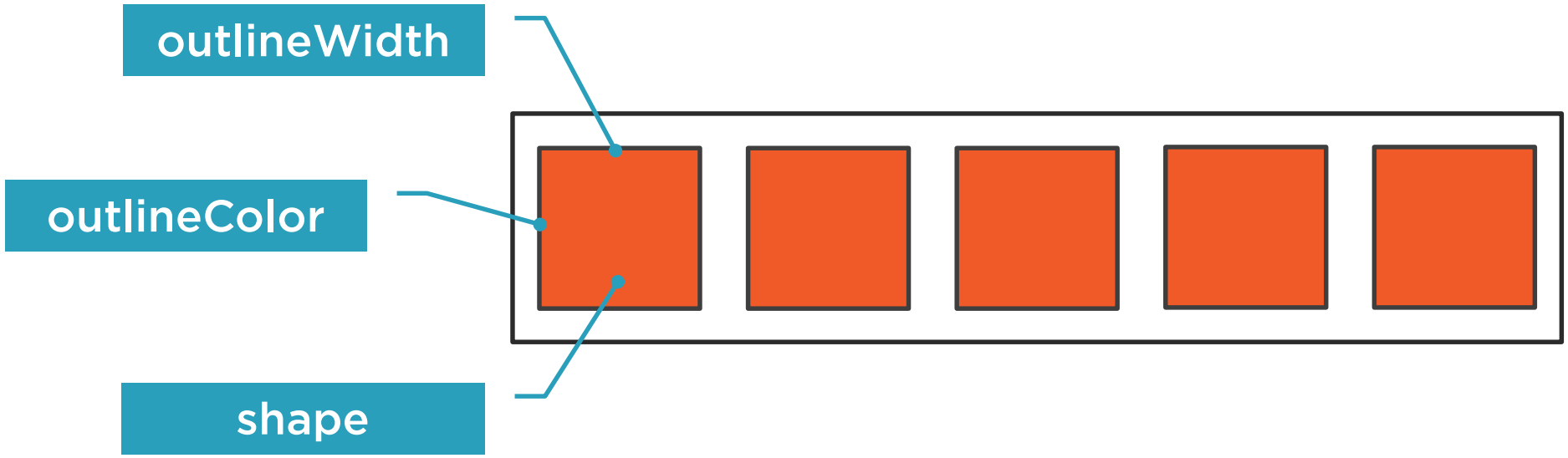
MotionEvent class

- Includes the specific type of action
- Includes x and y coordinates

Adding Configurability to Our Custom View



Adding Configurability to Our Custom View



Configurability

Should provide design-time support

- Configurable attributes need to be available in designer Properties window

Use declare-styleable resource

- Describes view's configurable attributes
- Each attribute has a name
- Each attribute has a format which identifies valid values

Configurability

Receiving design-time attribute values

- Passed to view constructor
- Received as AttributeSet
- Raw values from properties window
- Convert to TypedArray with Context.obtainStyledAttributes method

Configurability

TypedArray

- Attribute values in more useable form
- Get methods for each attribute format
- Access attribute values with R.stylable.XXX constants

Must recycle TypedArray when done

- Use TypedArray.recycle method

Summary

Accessing custom views from code

- Use findViewById to get view reference
- Use returned reference to call methods

Handling custom view touch events

- Override onTouchEvent
- MotionEvent class contains event details

Initiating custom view drawing

- Call invalidate method
- Will trigger a call to the onDraw method

Summary

Configuration attributes

- Described in declare-styleable resource
- Makes attributes available to designer

Accessing attribute values

- Passed to constructor as AttributeSet
- Convert to TypedArray with Context.obtainStyledAttributes method

TypedArray may be reused by system

- Call recycle method when done

Summary

Dimension configuration attributes

- Can be expressed in a variety of units
- TypedArray.getDimension method returns value as physical pixels

Drawing dimension constants in code

- Should use device independent pixels
- Use DisplayMetrics class to get physical pixel conversion factor at runtime