Google
Developers

Writing Custom Views for Android

Adam Powell
Romain Guy

Google I/O

Why custom views?

Isn't the framework good enough?

Why Custom Views?

- Unique presentations

Why Custom Views?

- Unique presentations
- Custom interactions

Why Custom Views?

- Unique presentations
- Custom interactions
- Layout optimizations

Why Custom Views?

- Unique presentations
- Custom interactions
- Layout optimizations
- Hero moments

What should I know?

- Important View events
- Measurement and Layout
- Drawing

What should I know?

- Important View events
- Measurement and Layout
- Drawing
- Design guidelines
- Guarantees
- Common pitfalls
- Tips and Tricks

Life of a View

Things every developer should know

Life of a View

- Attachment/detachment
- Traversals
- State save/restore

Attachment

onAttachedToWindow()

- Call super.onAttachedToWindow()!
- Perform any relevant state resets
- Start listening for state changes

Attachment

onDetachedFromWindow()

- Call super.onDetachedFromWindow()!
- Remove any posted Runnables
- Stop listening for data changes
- Clean up resources
  - Bitmaps
  - Threads

Life of a View

- What's missing?

View abstraction

- Views are at a lower level of abstraction than Activities/Fragments

```
android.app
...
android.widget
android.view
...
```

View abstraction

- Views are at a lower level of abstraction than Activities/Fragments
- Views are ignorant of Activity lifecycle

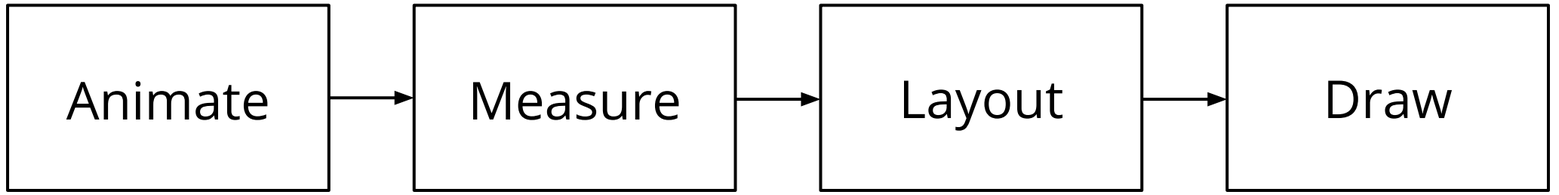
View abstraction

- Views are at a lower level of abstraction than Activities/Fragments
- Views are ignorant of Activity lifecycle
- Use listeners/callbacks

```
public interface CustomListener {
    onCustomEvent(CustomView v);
}
```

Traversals

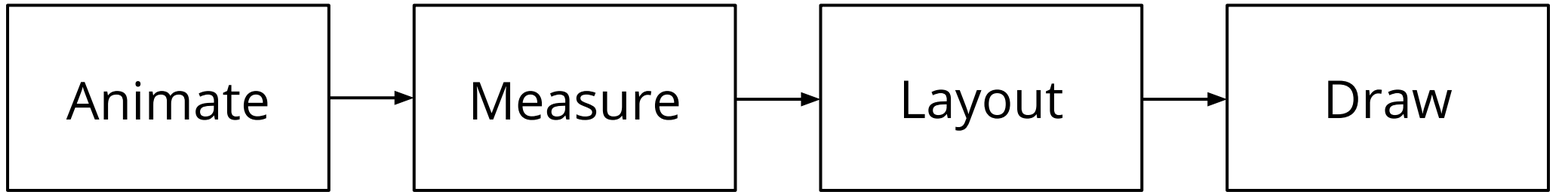
- Scheduled by animation, requestLayout(), invalidate()
- Phases



Traversals

Guarantees

- Animation events will always happen before measure
- A required measure will always happen before layout
- A required layout will always happen before draw



Measurement and Layout

Implementing your own ViewGroup

requestLayout()

- Schedules view hierarchy traversal for the upcoming frame

requestLayout()

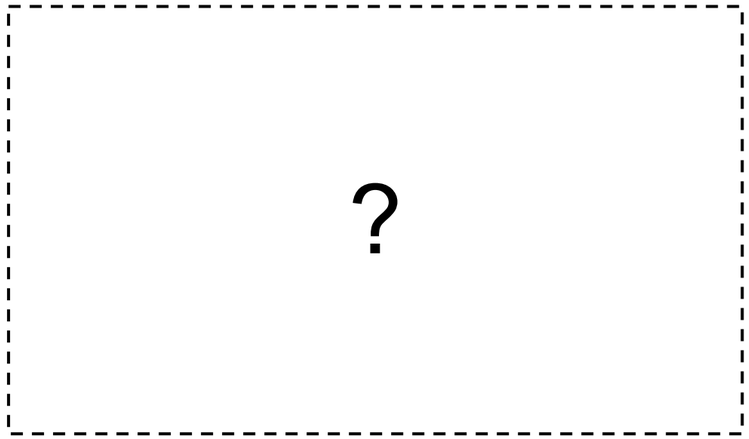
- Schedules view hierarchy traversal for the upcoming frame
- Calls requestLayout() on the view's parent
  - ...and its parent's parent
    - ...and its parent's parent's parent...

requestLayout()

- Schedules view hierarchy traversal for the upcoming frame
- Calls requestLayout() on the view's parent
  - ...and its parent's parent
    - ...and its parent's parent's parent...
- Anything can change!

onMeasure()

- Determines a size for the view and its children



onMeasure()

- Determines a size for the view and its children
- Accepts packed MeasureSpec parameters

```
int widthSpec = MeasureSpec.makeMeasureSpec(sizeInPx, MeasureSpec.EXACTLY);
int widthSize = MeasureSpec.getSize(widthSpec);
int widthMode = MeasureSpec.getMode(widthSpec);
```

onMeasure()

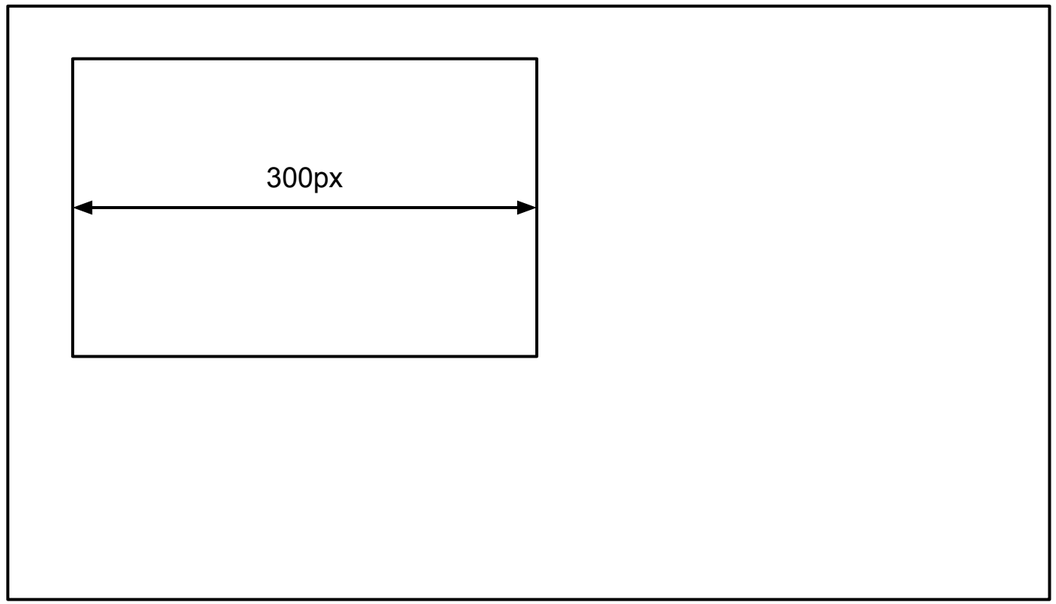
- Determines a size for the view and its children
- Accepts packed MeasureSpec parameters
- measure()s all child views

onMeasure()

- Determines a size for the view and its children
- Accepts packed MeasureSpec parameters
- measure()s all child views
- Must call setMeasuredDimension() before returning

MeasureSpec
Modes explained

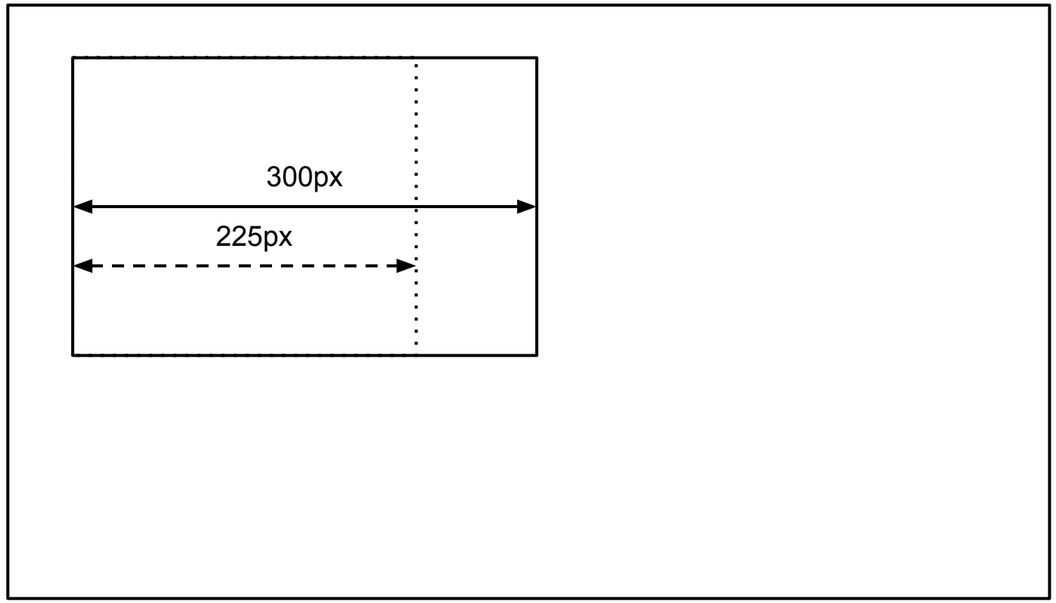
- MeasureSpec.EXACTLY
  - Be precisely this size
- Examples
  - android:layout_width="150dp"
  - android:layout_width="match_parent"
  - android:layout_weight="1"




```
MeasureSpec.makeMeasureSpec(300, MeasureSpec.
EXACTLY);
```

MeasureSpec
Modes explained

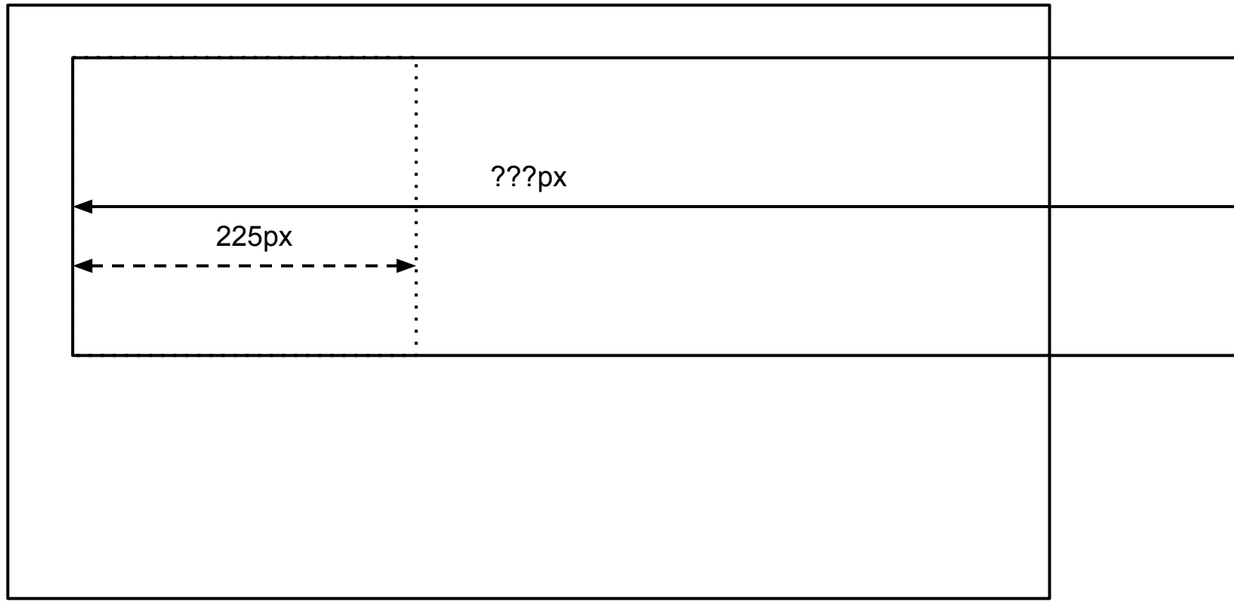
- MeasureSpec.AT_MOST
  - Be *up to* this size
- Examples
  - android:layout_width="wrap_content"




```
MeasureSpec.makeMeasureSpec(300, MeasureSpec.
AT_MOST);
```

MeasureSpec
Modes explained

- MeasureSpec.UNSPECIFIED
  - ???
- Examples
  - ScrollView
  - ListView




```
MeasureSpec.makeMeasureSpec(0, MeasureSpec.
UNSPECIFIED);
```

MeasureSpec
Creating the right specs

- ViewGroup.getChildMeasureSpec(int spec, int padding, int childDimension)
  - spec: the parent's MeasureSpec
  - padding: extra space in the parent not available
  - childDimension: how big the child wants to be
    - understands match_parent/wrap_content!

getChildMeasureSpec()
Example

- Parent: 300px EXACTLY
- Child: WRAP_CONTENT
- Result: ???

getChildMeasureSpec()
Example

- Parent: 300px EXACTLY
- Child: WRAP_CONTENT
- Result: 300px AT_MOST

onLayout()
The easy part

- Position child views

onLayout()
The easy part

- Position child views
- Happens once in a traversal
  - Save more expensive work until layout if you can

onLayout()
The easy part

- Position child views
- Happens once in a traversal
  - Save more expensive work until layout if you can
- Use info already computed in onMeasure()
  - Use getMeasuredWidth()/getMeasuredHeight()
  - getWidth()/getHeight() become valid on a view after layout() returns

LayoutParams

Specialized parameters for your ViewGroups

LayoutParams

- LayoutParams extends ViewGroup.LayoutParams
  - width/height come standard

LayoutParams

- LayoutParams extends ViewGroup.LayoutParams
  - width/height come standard
- Store arguments specific to your ViewGroup
  - e.g. weight (LinearLayout)

LayoutParams

- LayoutParams extends ViewGroup.LayoutParams
  - width/height come standard
- Store arguments specific to your ViewGroup
  - e.g. weight (LinearLayout)
- Can inflate from layout xml

LayoutParams

- LayoutParams extends ViewGroup.LayoutParams
  - width/height come standard
- Store arguments specific to your ViewGroup
  - e.g. weight (LinearLayout)
- Can inflate from layout xml
- Validation/conversion
  - checkLayoutParams(ViewGroup.LayoutParams)
  - generateLayoutParams(ViewGroup.LayoutParams)
  - generateLayoutParams(AttributeSet)
  - generateDefaultLayoutParams()

LayoutParams

- LayoutParams extends ViewGroup.LayoutParams
  - width/height come standard
- Store arguments specific to your ViewGroup
  - e.g. weight (LinearLayout)
- Can inflate from layout xml
- Validation/conversion
  - checkLayoutParams(ViewGroup.LayoutParams)
  - generateLayoutParams(ViewGroup.LayoutParams)
  - generateLayoutParams(AttributeSet)
  - generateDefaultLayoutParams()
- Can store private data for layout

Drawing

Android's UI renderer

invalidate()

- Schedules view hierarchy traversal for the upcoming frame

invalidate()

- Schedules view hierarchy traversal for the upcoming frame
- Calls invalidateChild() on the view's parent
  - ...and its parent's parent
    - ...and its parent's parent's parent...

invalidate()

- Calling invalidate() marks the entire View as dirty
- invalidate(int, int, int, int) can be used to dirty a subregion of the View
  - Always prefer this version if you can

draw()

- Draws the View and its children

draw()

- Draws the View and its children
- Invokes other drawing method that you can override

draw()

- Draws the View and its children
- Invokes other drawing method that you can override
- onDraw()
  - Draws the content, e.g. text in TextView
  - The background is drawn before onDraw()
  - Skipped if setWillNotDraw(true) is set (default in ViewGroup)

draw()

- Draws the View and its children
- Invokes other drawing method that you can override
- onDraw()
  - Draws the content, e.g. text in TextView
  - The background is drawn before onDraw()
  - Skipped if setWillNotDraw(true) is set (default in ViewGroup)
- dispatchDraw()
  - Calls draw() on every child via ViewGroup.drawChild()
  - drawChild() handles transforms, animations, etc.

Software vs Hardware

- Android 3.0 introduced hardware rendering

Software vs Hardware

- Android 3.0 introduced hardware rendering
- View.draw() is handled differently

Software vs Hardware

- Android 3.0 introduced hardware rendering
- View.draw() is handled differently
- In software
  - View.draw() is called on every parent of the invalidate() source
  - And on every view that intersects the dirty region

Software vs Hardware

- Android 3.0 introduced hardware rendering
- View.draw() is handled differently
- In software
  - View.draw() is called on every parent of the invalidate() source
  - And on every view that intersects the dirty region
- In hardware
  - View.draw() is called only on the invalidate() source

Software vs Hardware

- Android 3.0 introduced hardware rendering
- View.draw() is handled differently
- In software
  - View.draw() is called on every parent of the invalidate() source
  - And on every view that intersects the dirty region
- In hardware
  - View.draw() is called only on the invalidate() source
- Watch “Accelerated Android Rendering” from Google I/O 2011
  - http://goo.gl/ANmIW

Instance State

Saving it for later

Instance State

- View methods
  - Parcelable onSaveInstanceState()
  - onRestoreInstanceState(Parcelable)
- SavedState extends View.BaseSavedState
- View must have an id

Instance State

What to save

- Data model should be separate from view state
- Save user interactions in progress
  - Scroll position
  - Active selections
  - Active modes
  - Uncommitted user input
    - e.g. text entered

Touch Events

Responding to the user

Touch Events

Event stream consistency

- A valid touch event stream is:

Touch Events

Event stream consistency

- A valid touch event stream is:
  - ACTION_DOWN

Touch Events

Event stream consistency

- A valid touch event stream is:
  - ACTION_DOWN
  - Matched pairs of ACTION_POINTER_DOWN/ACTION_POINTER_UP

Touch Events

Event stream consistency

- A valid touch event stream is:
  - ACTION_DOWN
  - Matched pairs of ACTION_POINTER_DOWN/ACTION_POINTER_UP
  - Zero or more ACTION_MOVEs
    - Must only contain valid pointers!

Touch Events

Event stream consistency

- A valid touch event stream is:
  - ACTION_DOWN
  - Matched pairs of ACTION_POINTER_DOWN/ACTION_POINTER_UP
  - Zero or more ACTION_MOVEs
    - Must only contain valid pointers!
  - ACTION_UP/ACTION_CANCEL

Touch Events

Claiming the event stream

- onTouchEvent
  - Returns true to claim events

Touch Events

Claiming the event stream

- onTouchEvent
  - Returns true to claim events
- onInterceptTouchEvent
  - Also returns true to claim events
  - ...but can observe events sent to children

Touch Events

Claiming the event stream

- onTouchEvent
  - Returns true to claim events
- onInterceptTouchEvent
  - Also returns true to claim events
  - ...but can observe events sent to children
- requestDisallowInterceptTouchEvent
  - Block onInterceptTouchEvent for parents
  - Sticky until the end of the gesture

Touch Events

Claiming the event stream

- onTouchEvent
  - Returns true to claim events
- onInterceptTouchEvent
  - Also returns true to claim events
  - ...but can observe events sent to children
- requestDisallowInterceptTouchEvent
  - Block onInterceptTouchEvent for parents
  - Sticky until the end of the gesture

Touch Events
Sloppy intercept

- Wait to intercept until a slop distance is crossed
- ViewConfiguration#getScaledTouchSlop()

<Thank You!>

Find us on Google+

http://google.com/+AdamWPowell
http://google.com/+RomainGuy
#io2013 #swag #holoyolo

Google
Developers

Presentation Bullet Slide Layout

- Titles are formatted as Open Sans with bold applied and font size is set at 45
  - Y coordinates for title is .18in
  - Y coordinates for bullet text is 2.03in
- Title capitalization is title case
- Subtitle capitalization is title case

Presentation Bullet Slide Layout

Subtitle Placeholder

- Titles are formatted as Open Sans with bold applied and font size is set at 45
  - Y coordinates for title is .18in
  - Y coordinates for subtitle is 1.48in
  - Y coordinates for bullet text is 2.72in
- Title capitalization is title case
- Subtitle capitalization is title case
- Titles and subtitles should never have a period at the end

Color Palette

Flat Color

67
135
253

244
74
63

255
209
77

13
168
97

Secondary

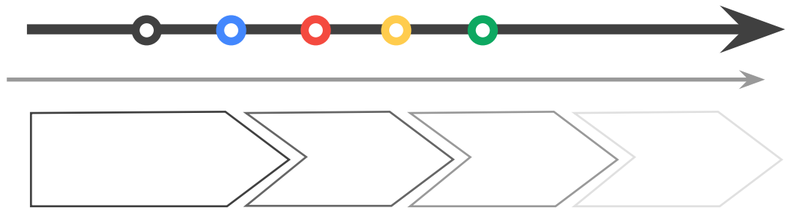
Gradient

Grays

Graphic Element Styles and Arrows

Rounded Box

Code Boxes

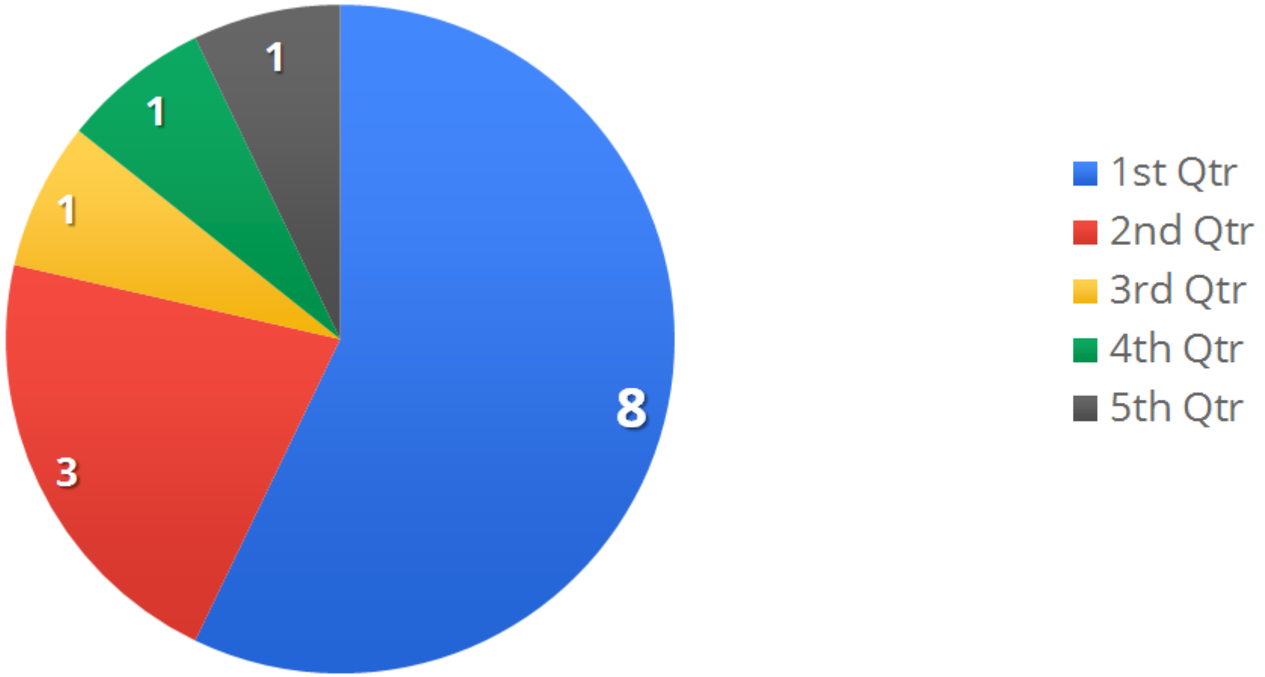


Arrows

Content Container Boxes

Pie Chart Example

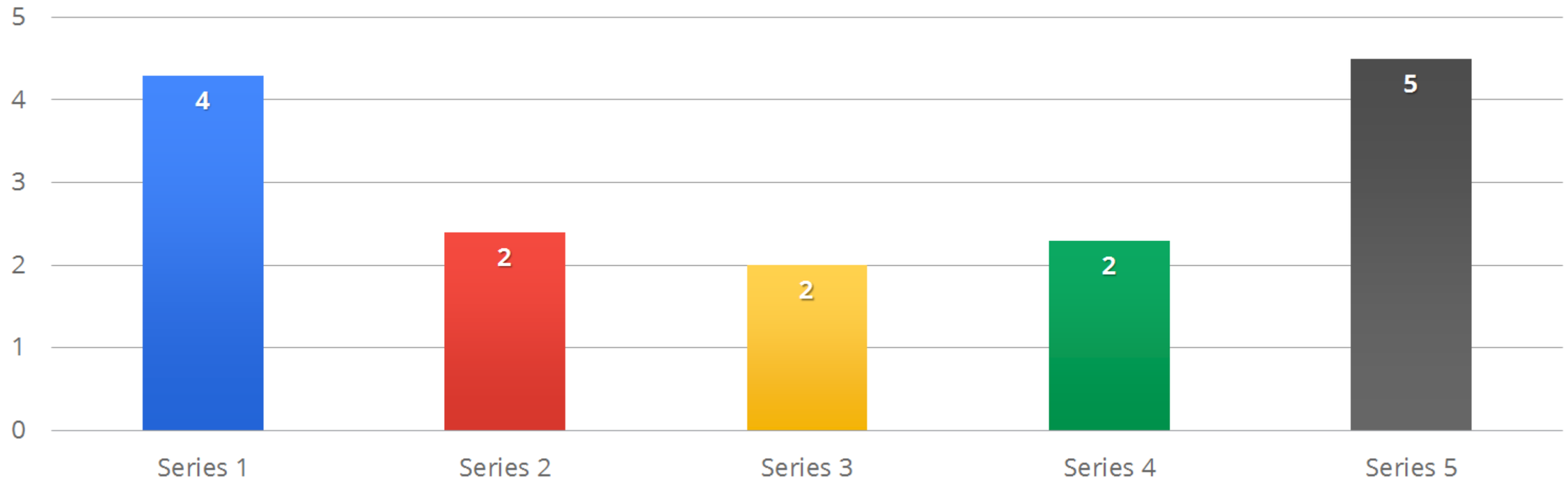
Subtitle Placeholder



source: place source info here

Column Chart Example

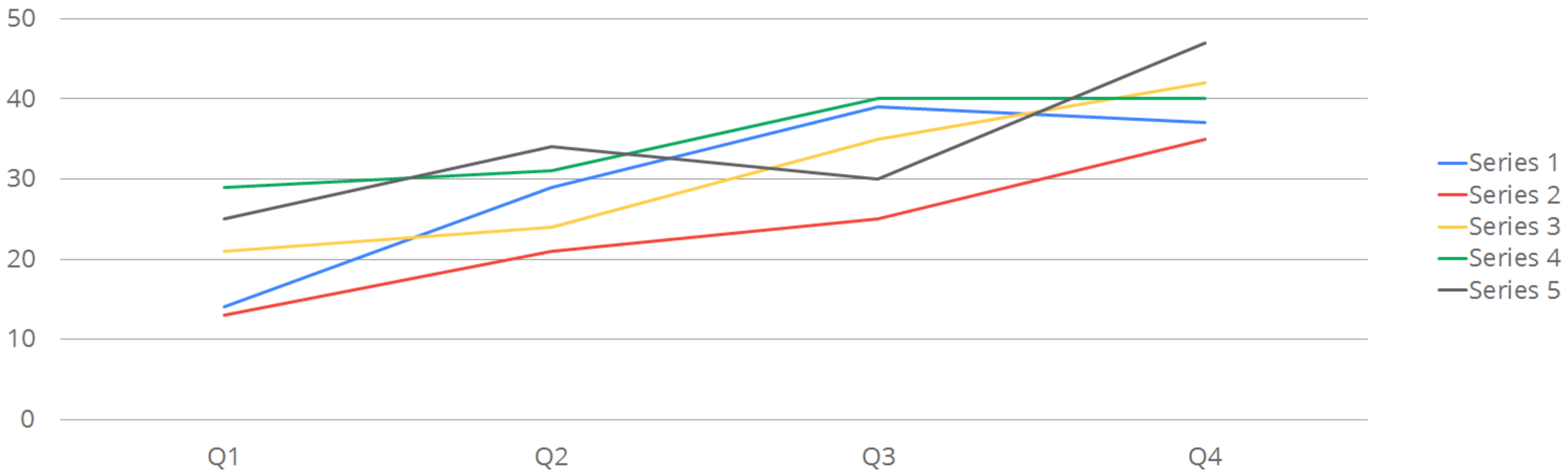
Subtitle Placeholder



source: place source info here

Line Chart Example

Subtitle Placeholder



source: place source info here

Table Option A

Subtitle Placeholder

	Column 1	Column 2	Column 3	Column 4
Row 1	placeholder	placeholder	placeholder	placeholder
Row 2	placeholder	placeholder	placeholder	placeholder
Row 3	placeholder	placeholder	placeholder	placeholder
Row 4	placeholder	placeholder	placeholder	placeholder
Row 5	placeholder	placeholder	placeholder	placeholder
Row 6	placeholder	placeholder	placeholder	placeholder
Row 7	placeholder	placeholder	placeholder	placeholder

Table Option B

Subtitle Placeholder

Header 1	placeholder	placeholder	placeholder
Header 2	placeholder	placeholder	placeholder
Header 3	placeholder	placeholder	placeholder
Header 4	placeholder	placeholder	placeholder
Header 5	placeholder	placeholder	placeholder

Segue Slide

Subtitle Placeholder

“ This is an example of quote text.”

Name

Company

Code Slide With Subtitle Placeholder

Subtitle Placeholder

HTML

```
<script type='text/javascript'>
  // Say hello world until the user starts questioning
  // the meaningfulness of their existence.
  function helloWorld(world) {
    for (var i = 42;--i >= 0;) {
      alert ('Hello' + String(world));
  }
}
</script>
<style>
p { color: pink }
p { color: blue }
u { color: 'umber' }
</style>
```