Tampa Bay Android Developers Group

Build a Faster UI with ConstraintLayout

Scott A.Thisse
July 17, 2017

About Me

- Lead the TAMPA BAY ANDROID DEVELOPERS GROUP 
- Help lead  GDG Sun Coast
- Lead developer for centripetalfusion  which does remote Android development and on-site training

What is ContraintLayout?

Android's newest layout:

- more flexibility

What is ContraintLayout?

Android's newest layout:

- more flexibility
- shallower view hierarchies

Requirements for Use?

- Android 2.3 Gingerbread (API 9) or higher

Requirements for Use?

- Android 2.3 Gingerbread (API 9) or higher
- Android Studio 2.2 or higher

Requirements for Use?

- Android 2.3 Gingerbread (API 9) or higher
- Android Studio 2.2 or higher
- Build tools 23.0.2 or higher

Requirements for Use?

- Android 2.3 Gingerbread (API 9) or higher
- Android Studio 2.2 or higher
- Build tools 23.0.2 or higher
- Gradle dependency:

  com.android.support.constraint:constraint-layout:1.0.2

Requirements for Use?

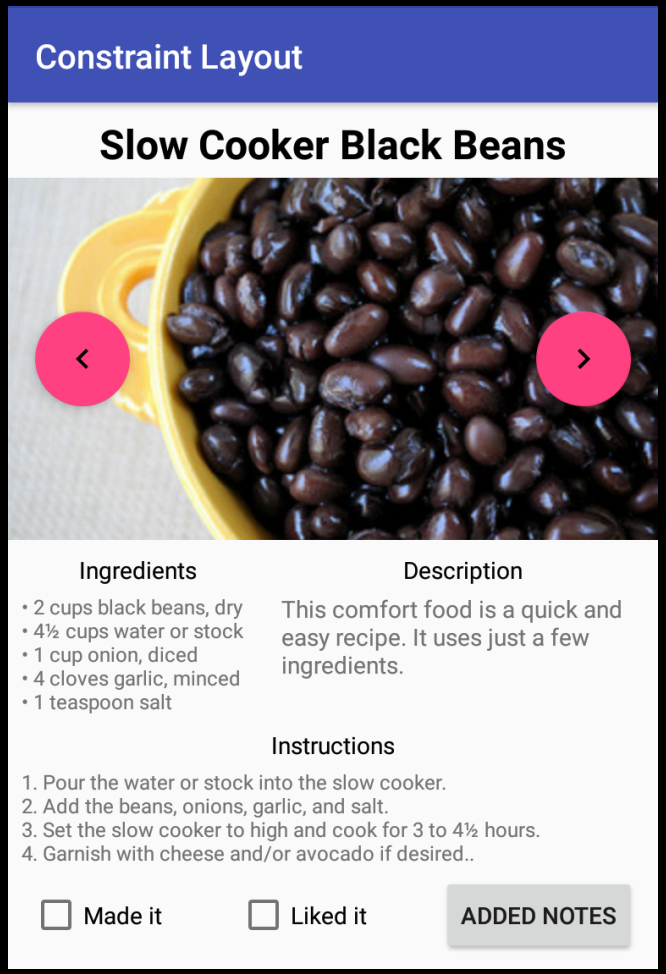
- Android 2.3 Gingerbread (API 9) or higher
- Android Studio 2.2 or higher
- Build tools 23.0.2 or higher
- Gradle dependency:

  com.android.support.constraint:constraint-layout:1.0.2
- This talk uses Android Studio 3.0 Canary 5

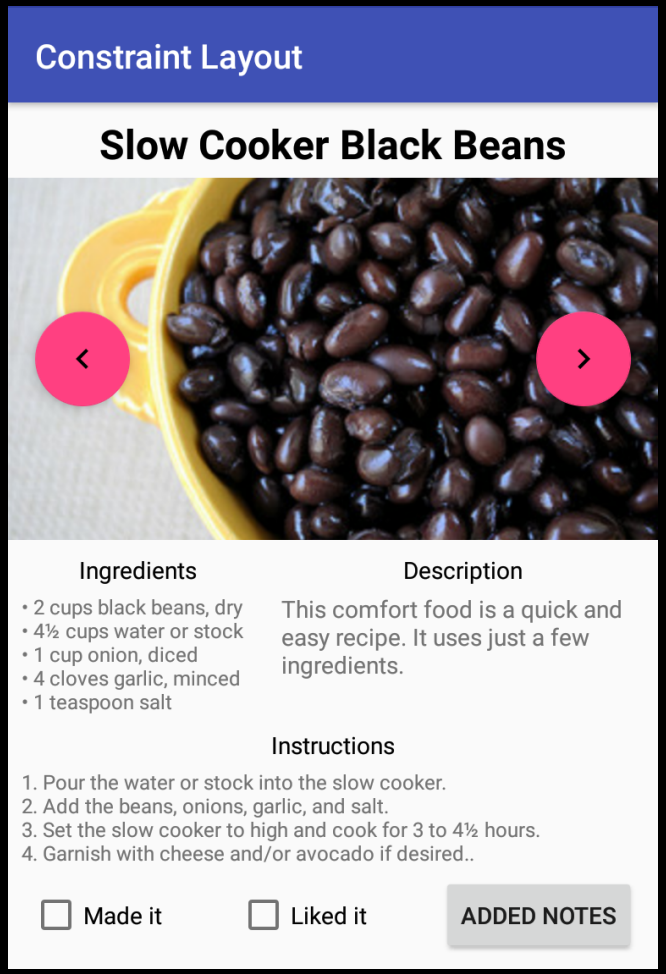
A Fairly Complicated Layout

* image from https://www.flickr.com/photos/cookbookman/5684475740

How to Create?

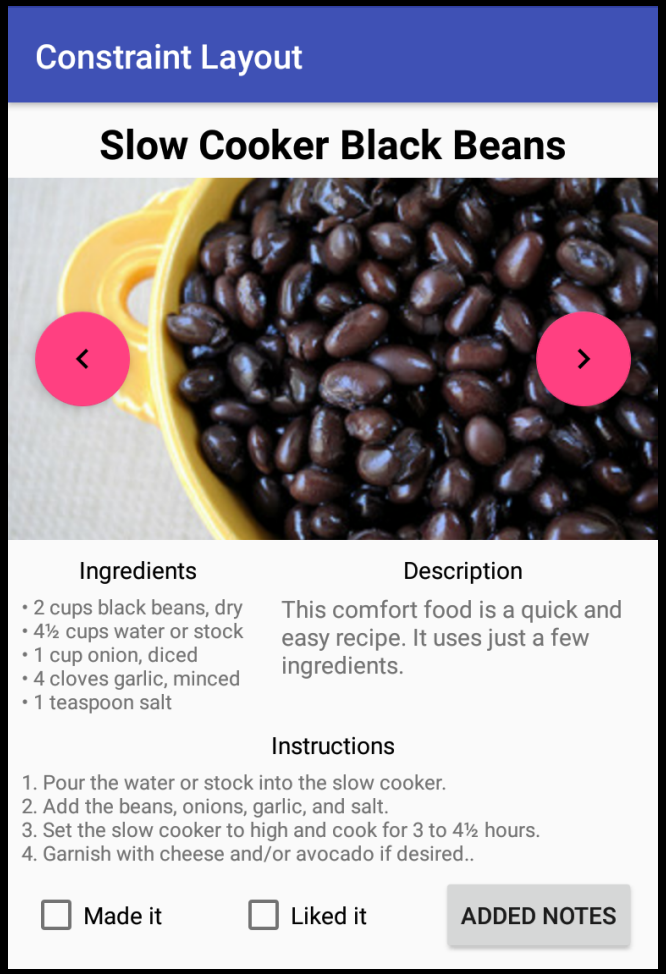


How to Create?



Use RelativeLayout?

How to Create?

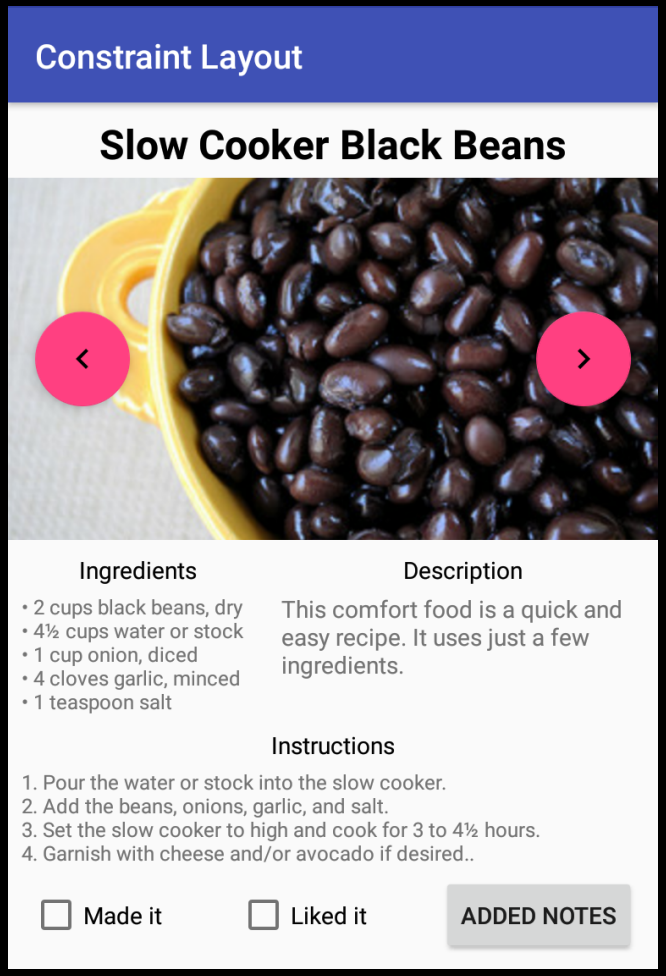


Use RelativeLayout?

Trouble with:

- 40/60 column split

How to Create?

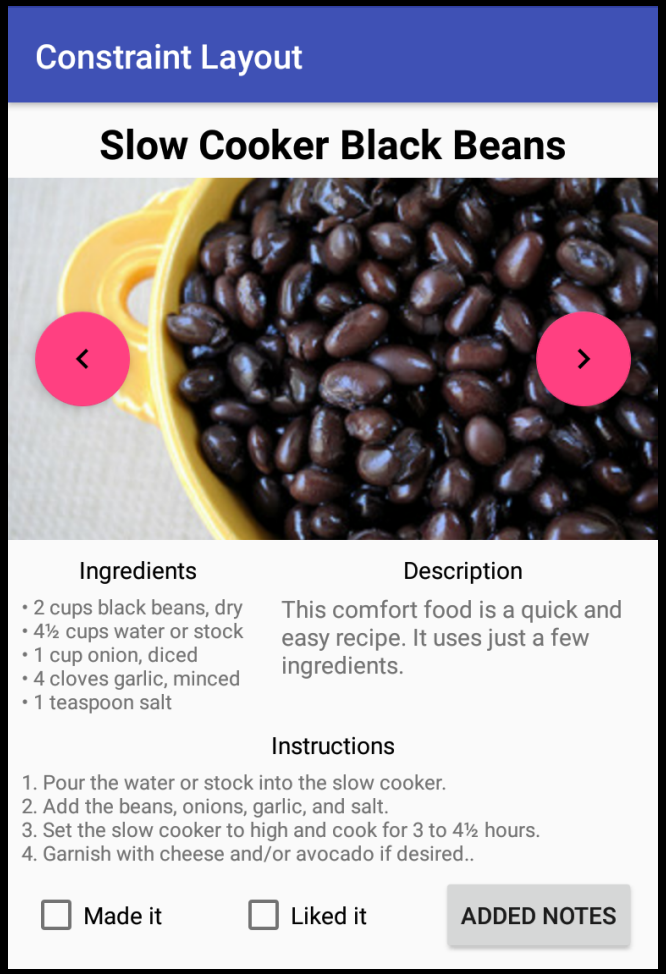


Use RelativeLayout?

Trouble with:

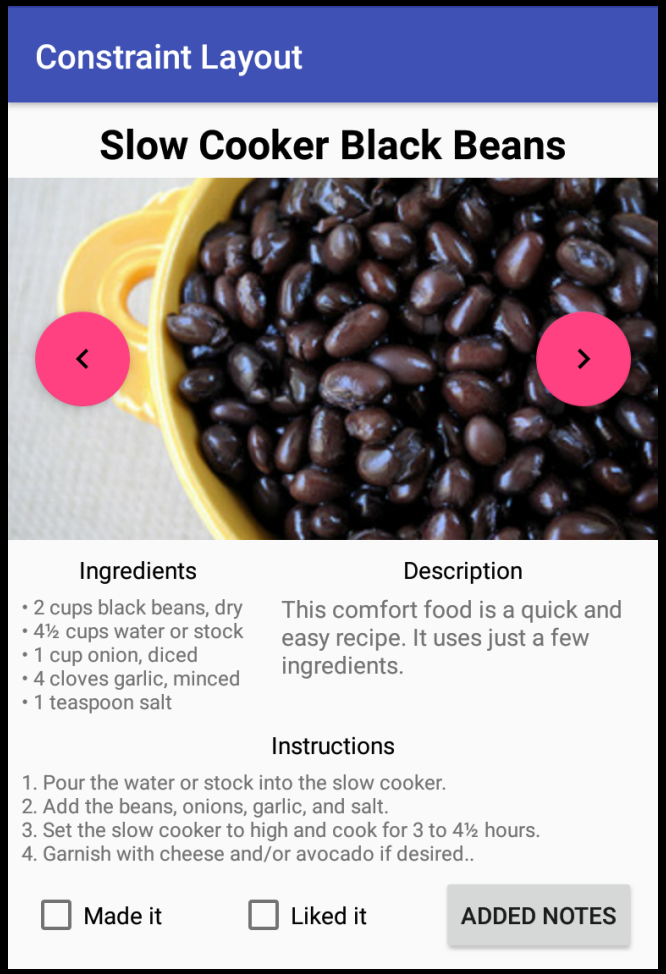
- 40/60 column split
- Positioning *Liked It*

How to Create?



Use LinearLayout?

How to Create?

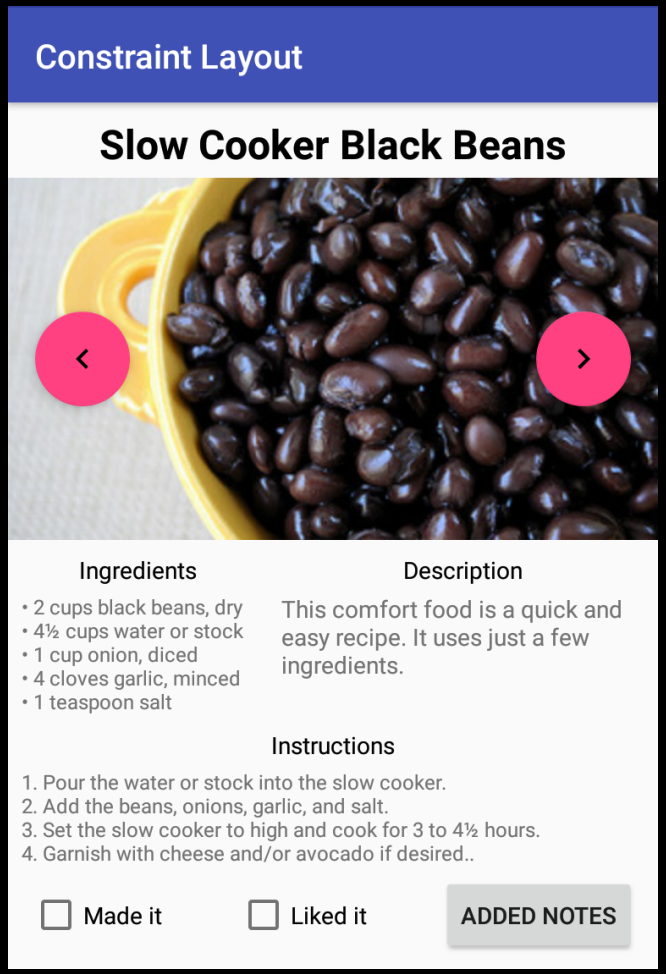


Use LinearLayout?

Trouble with:

- Overlayed nav buttons

How to Create?



Use LinearLayout?

Trouble with:

- Overlayed nav buttons
- Positioning *Liked It* still needs manual tweaking

Old Style Layout

- LinearLayout: vertical
  - TextView *title*
  - RelativeLayout
    - ImageView *photo*
    - FloatingActionButton *prev*
    - FloatingActionButton *next*
  - LinearLayout: horizontal, weighted
    - LinearLayout: vertical
      - TextView *ingredients_lbl*
      - TextView *ingredients*
    - LinearLayout: vertical
      - TextView *description_lbl*
      - TextView *description*
  - TextView *instructions_lbl*
  - TextView *instructions*
  - LinearLayout: horizontal
    - CheckBox *made_it*
    - CheckBox *liked_it*
    - Button *notes*

40/60 Column Split

LinearLayout: horizontal

- LinearLayout: vertical with weight 40
  - TextView *ingredients_lbl*
  - TextView *ingredients*
- LinearLayout: vertical with weight 60
  - TextView *description_lbl*
  - TextView *description*

Overlayed Navigation Buttons

RelativeLayout

- ImageView *photo*
- FloatingActionButton *prev*
- FloatingActionButton *next*

Positioning *Liked It*

LinearLayout: horizontal

- CheckBox *made_it*
- CheckBox *liked_it* with tweaked margins
- Button *notes*

ConstraintLayout

Let's consider its layout components...

Handles

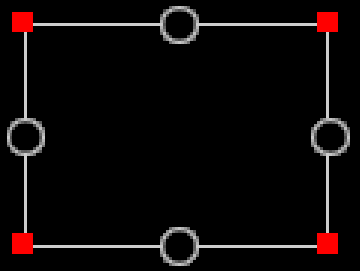
– connection points used to specify constraints

Handle Types

- *resize* – used to resize the view

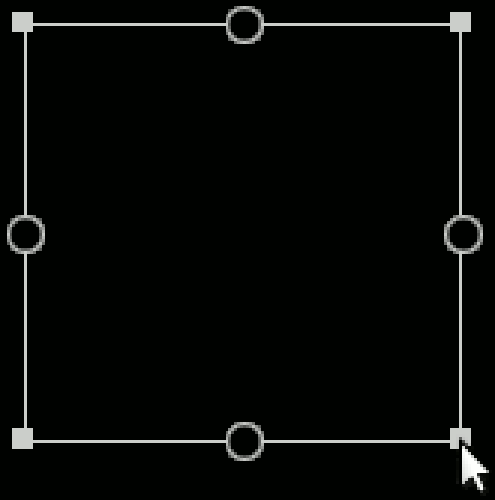
Handle Types

- *resize* – used to resize the view



Handle Types

- *resize* – used to resize the view



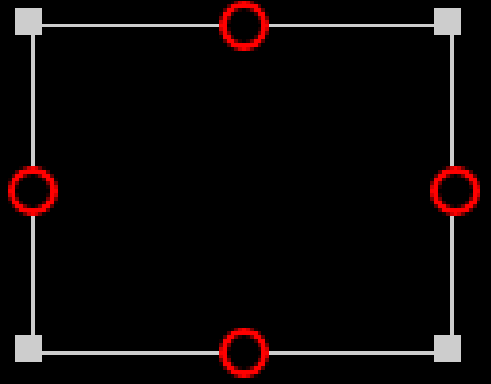
<Animation>

Handle Types

- *side* – used for positioning and alignment

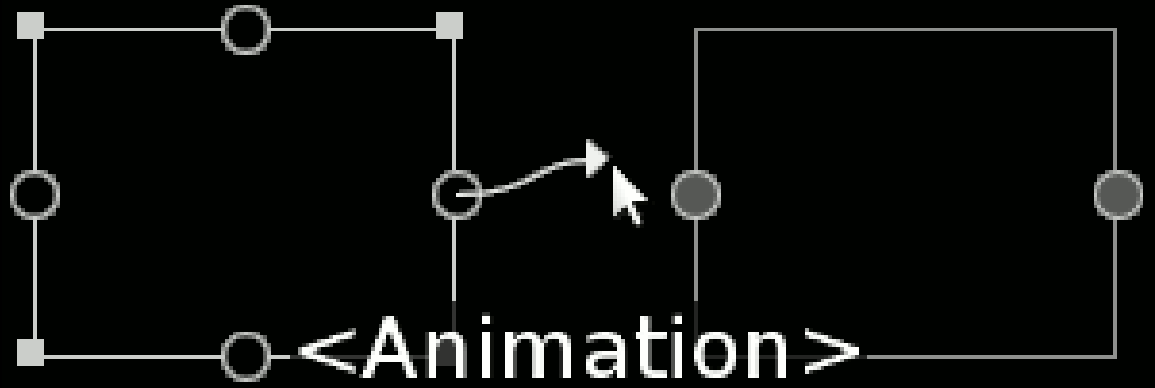
Handle Types

- *side* – used for positioning and alignment



Handle Types

- *side* – used for positioning and alignment



Handle Types

- *baseline* – used for aligning text baselines

Handle Types

- *baseline* – used for aligning text baselines

(click 

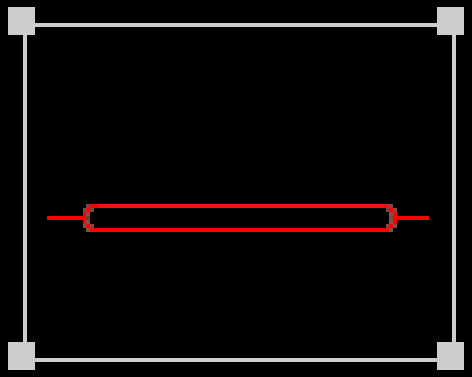
to toggle handle visibility)

Handle Types

- *baseline* – used for aligning text baselines

(click 

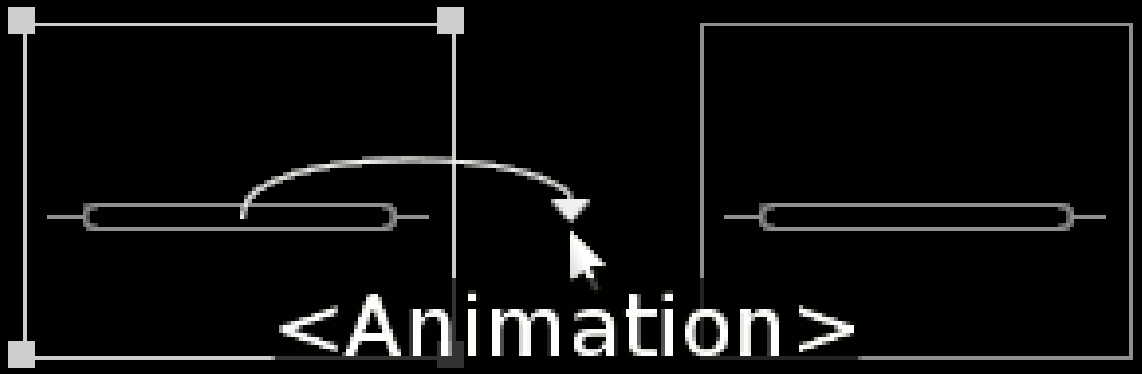
to toggle handle visibility)



Handle Types

- *baseline* – used for aligning text baselines

(click  to toggle handle visibility)



Handle Rules

- Can't mix handle types (*side* vs. *baseline*)

Handle Rules

- Can't mix handle types (*side* vs. *baseline*)
- Can't mix directions (vertical vs horizontal)

Constraints

– rules used to size and position views

Working with Constraints

To create:

- (Select a view if its handles aren't visible)

Working with Constraints

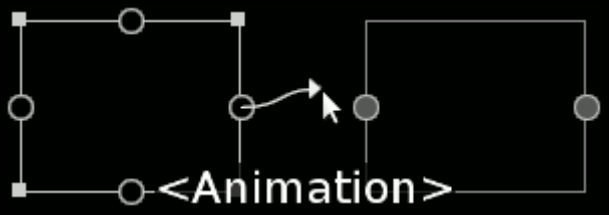
To create:

- (Select a view if its handles aren't visible)
- Click on an anchor handle and drag to a second anchor handle

Working with Constraints

To create:

- (Select a view if its handles aren't visible)
- Click on an anchor handle and drag to a second anchor handle



Working with Constraints

To delete:

- Click on an anchor handle to delete its attached constraints

Working with Constraints

To delete:

- Click on an anchor handle to delete its attached constraints
- Click on  to delete all constraints for a view

Chains

– used to divvy up space between views

Working with Chains

- To create:
  - Add bidirectional constraints

Working with Chains

- To create:
  - Add bidirectional constraints
  - Or select  (*Create Horizontal Chain*)

    or  (*Create Vertical Chain*)

Working with Chains

- Represented in XML as a pair of constraints

Working with Chains

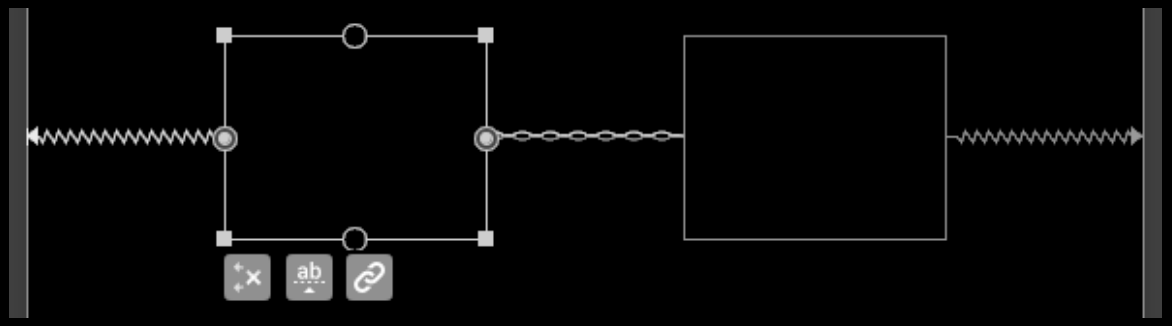
- Represented in XML as a pair of constraints
- Click on  to cycle through chain modes

Chain Modes

- *spread* – evenly distribute available space

Chain Modes

- *spread* – evenly distribute available space

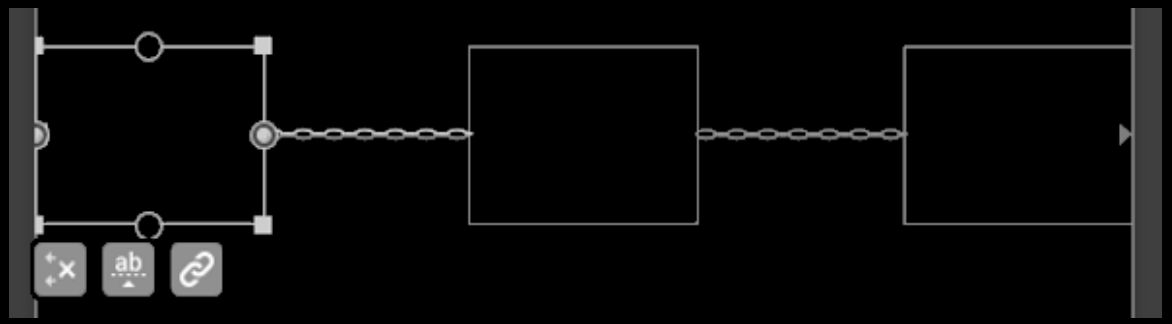


Chain Modes

- *spread_inside* – also move outer views to edges

Chain Modes

- *spread_inside* – also move outer views to edges

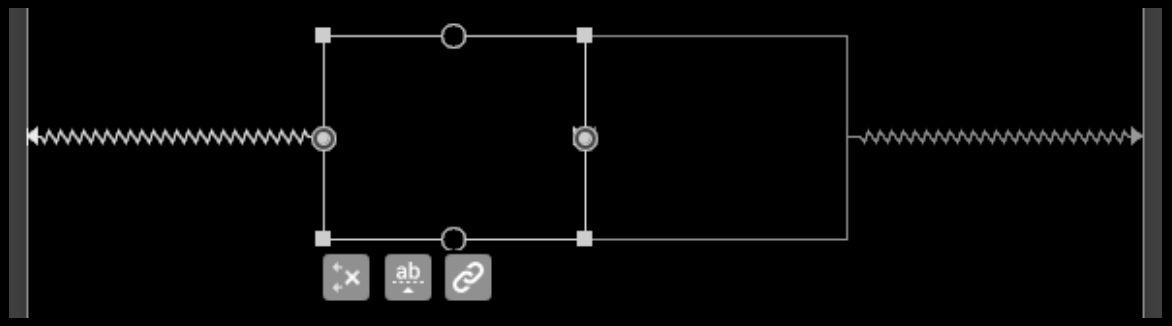


Chain Modes

- *packed* – pack views together

Chain Modes

- *packed* – pack views together



More Chain Info

- Margins can be added and will be honored

More Chain Info

- Margins can be added and will be honored
- Spread modes can use weights

More Chain Info

- Margins can be added and will be honored
- Spread modes can use weights
- Packed mode can use bias

Guidelines

– design-time visual guides used for alignment

Guideline Info

- Used to implement Material Design *keylines*

Guideline Info

- Used to implement Material Design *keylines*
- Used as constraint anchor points

Guideline Info

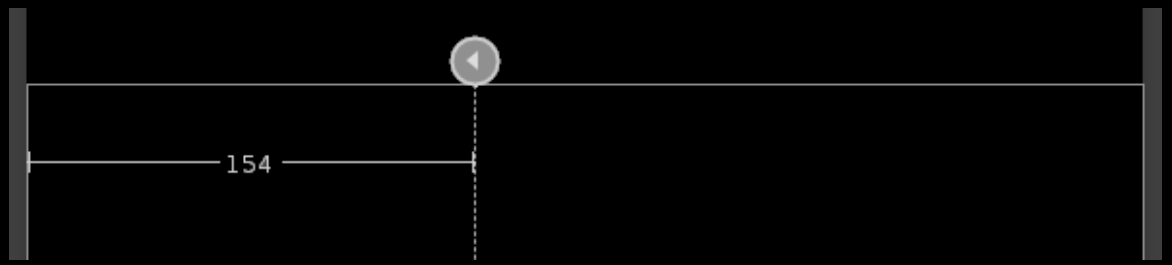
- Used to implement Material Design *keylines*
- Used as constraint anchor points
- Click on  to create

Guideline Types

- *begin* – number of pixels from the left/top

Guideline Types

- *begin* – number of pixels from the left/top

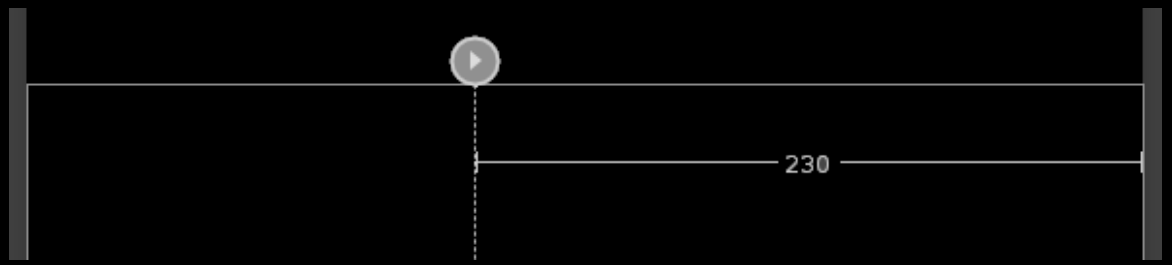


Guideline Types

- *end* – number of pixels from the right/bottom

Guideline Types

- *end* – number of pixels from the right/bottom

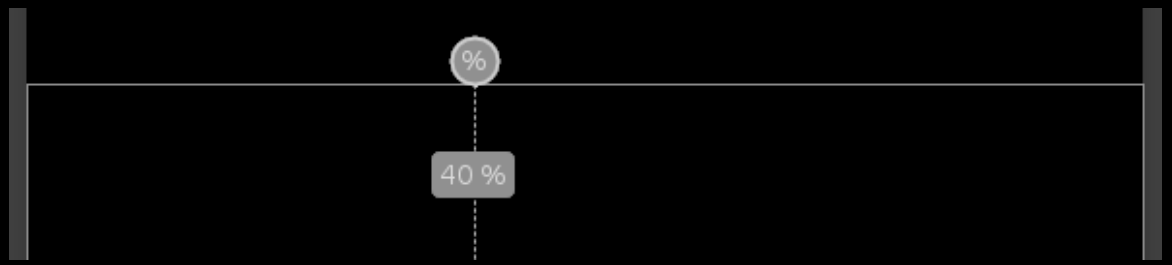


Guideline Types

- *percent* – percentage from the left/top

Guideline Types

- *percent* – percentage from the left/top



Moving Guideline Position

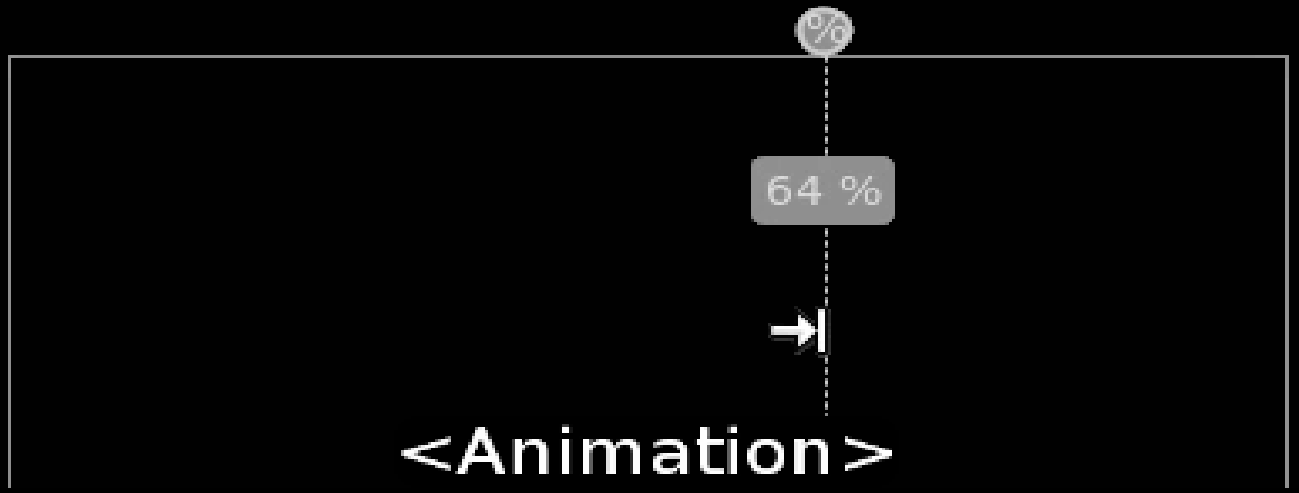
- Click on guideline line and release

Moving Guideline Position

- Click on guideline line and release
- Click on line again and drag

Moving Guideline Position

- Click on guideline line and release
- Click on line again and drag

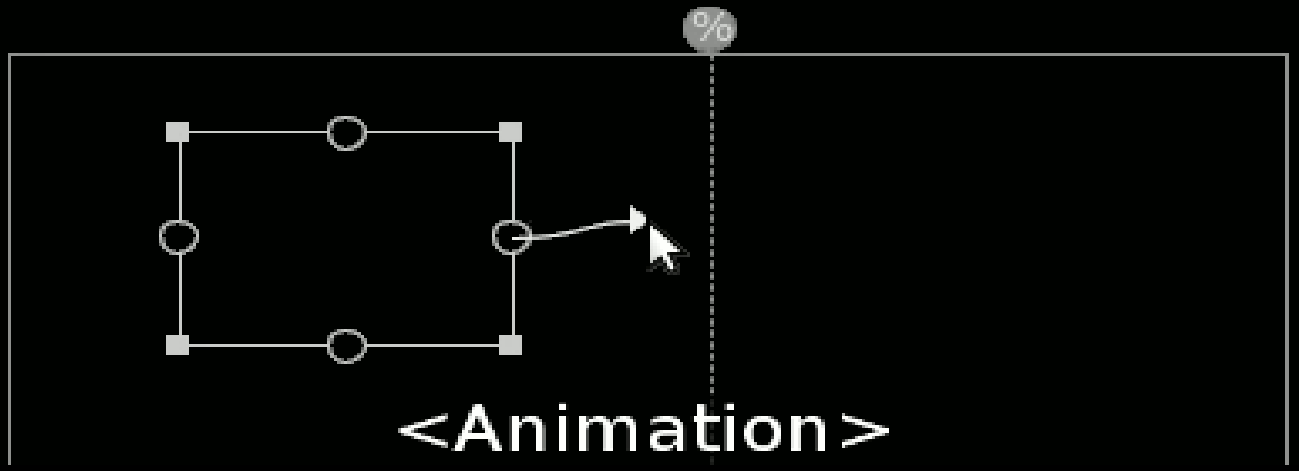


Connecting to Guideline

- Click on a view’s anchor handle and drag to a guideline

Connecting to Guideline

- Click on a view’s anchor handle and drag to a guideline



Layout Editor Views

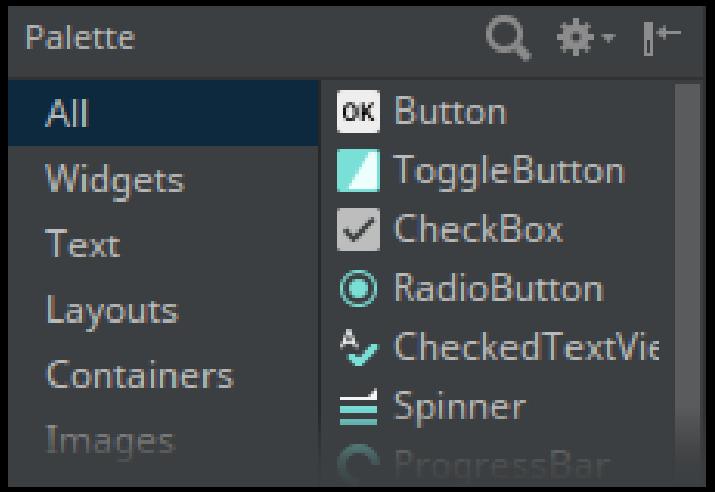
- Familiar text view with preview

Layout Editor Views

- Familiar text view with preview
- Design view with several panels

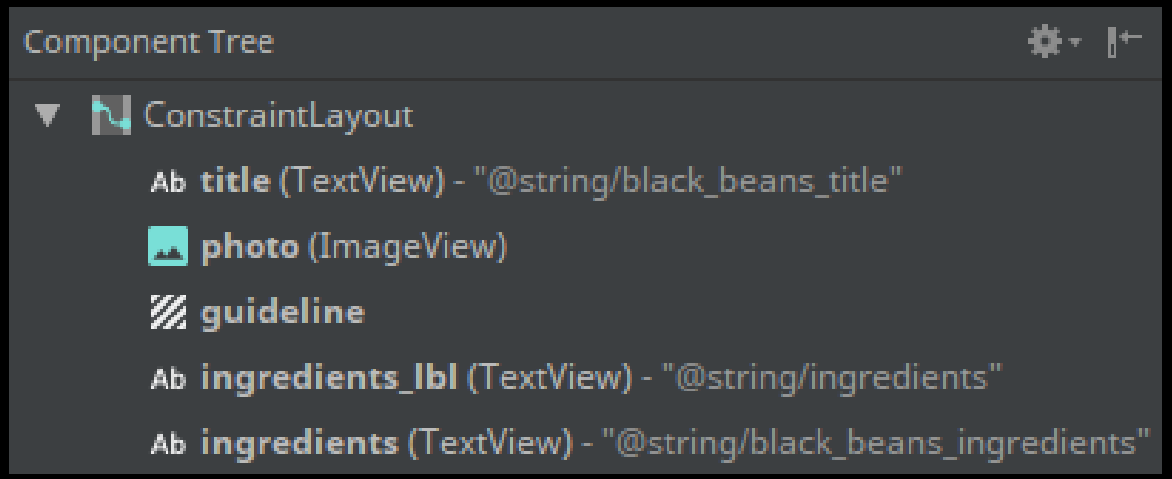
Design View Panels

- Palette – categorized list of available widgets



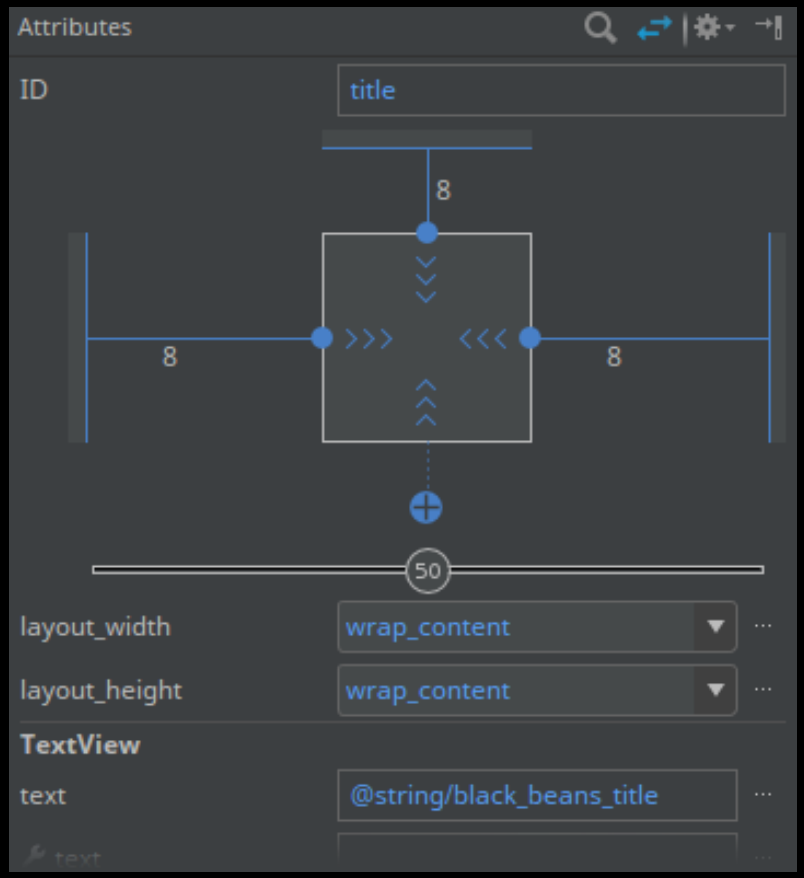
Design View Panels

- Component Tree – currently defined views



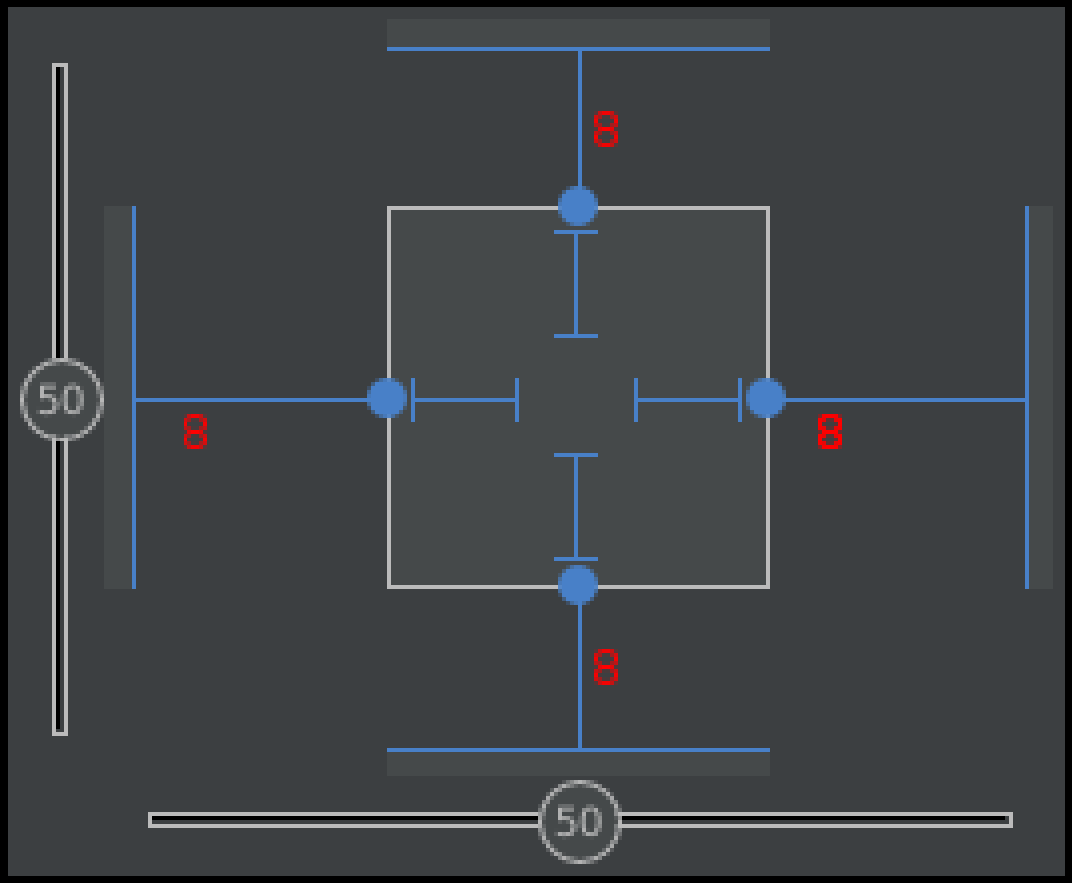
Design View Panels

- Attributes – view inspector and properties



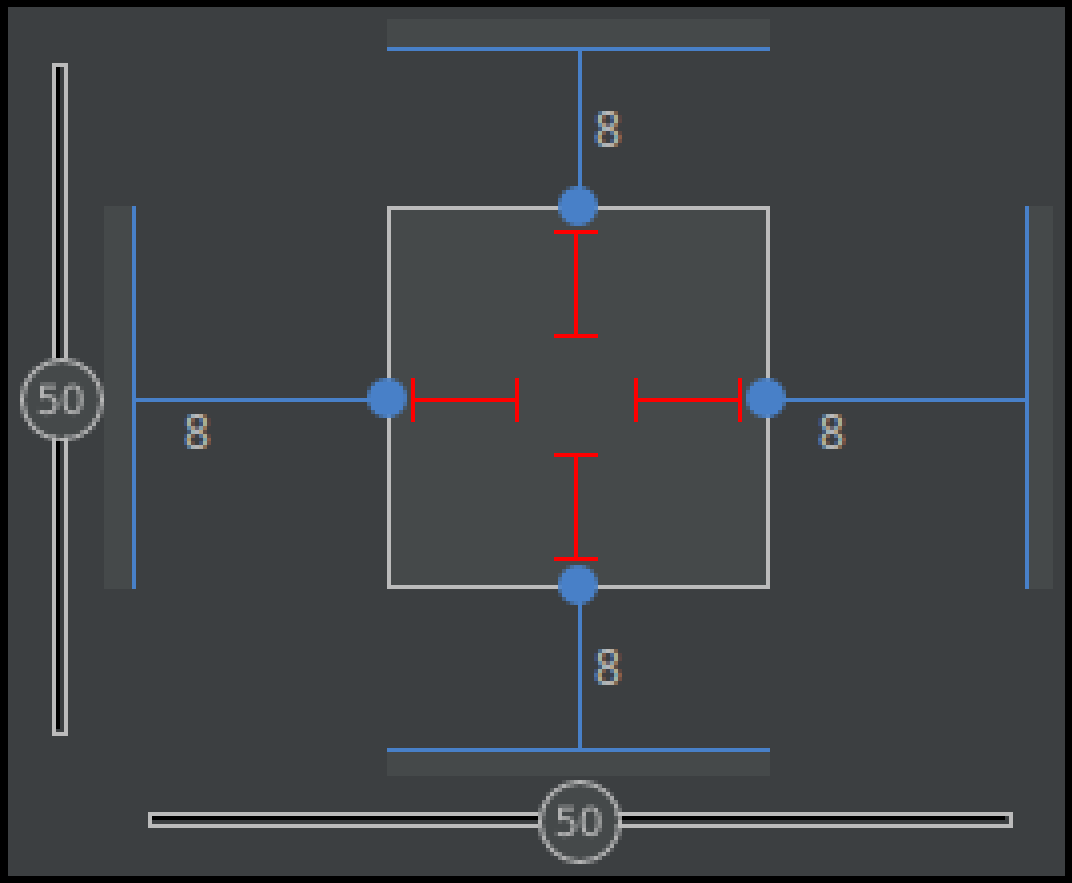
Inspector Graph Elements

- Numbers outside box are margins



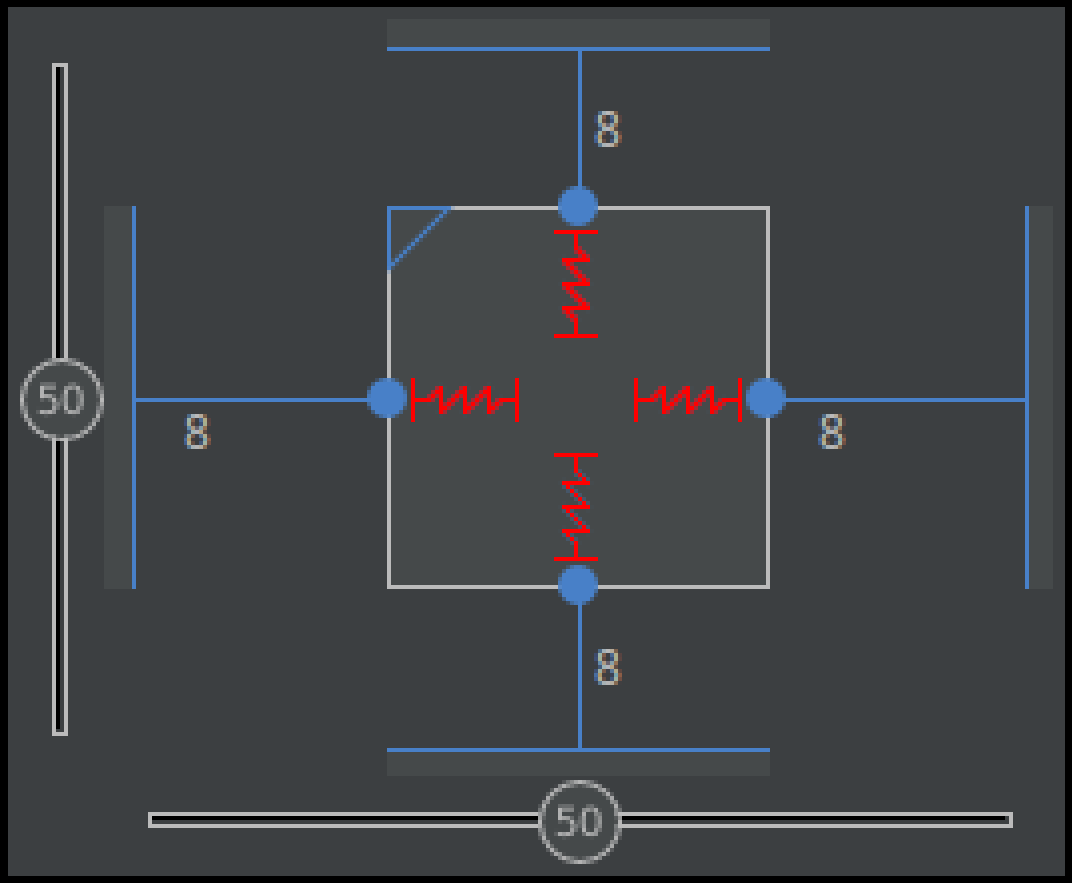
Inspector Graph Elements

- Size: fixed



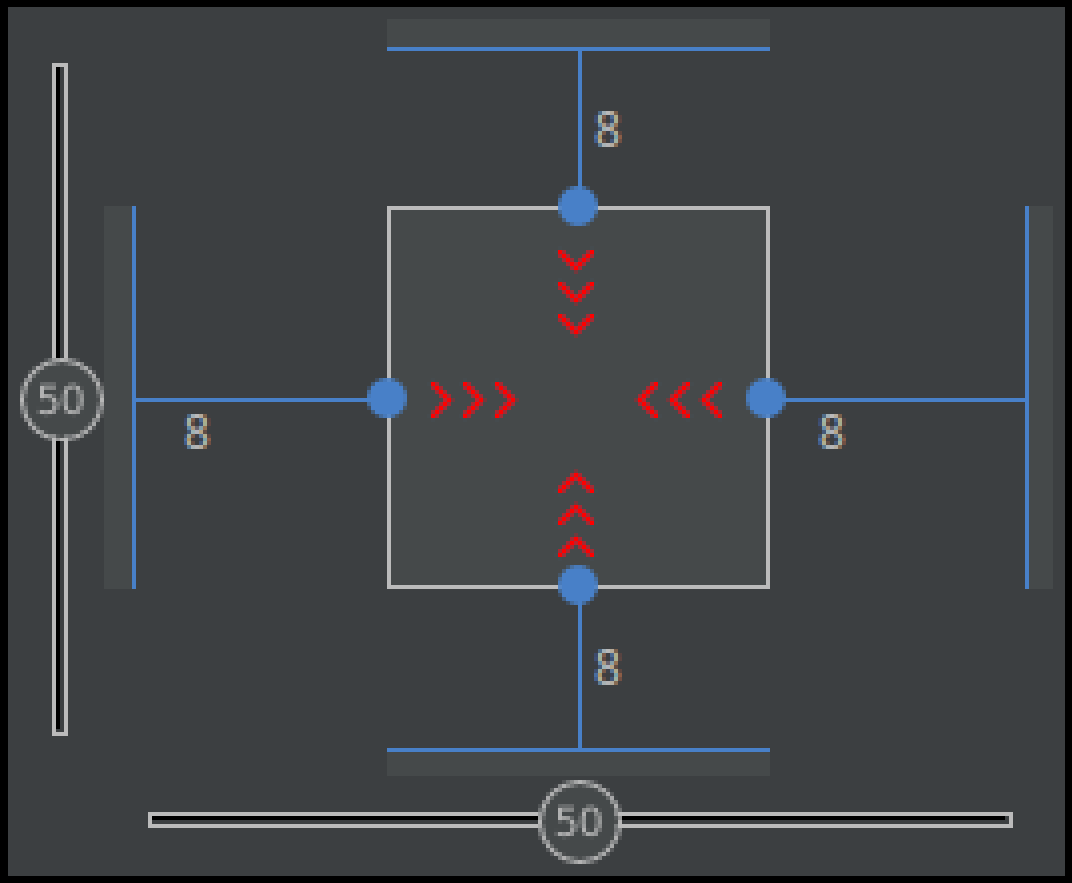
Inspector Graph Elements

- Size: match_constraint



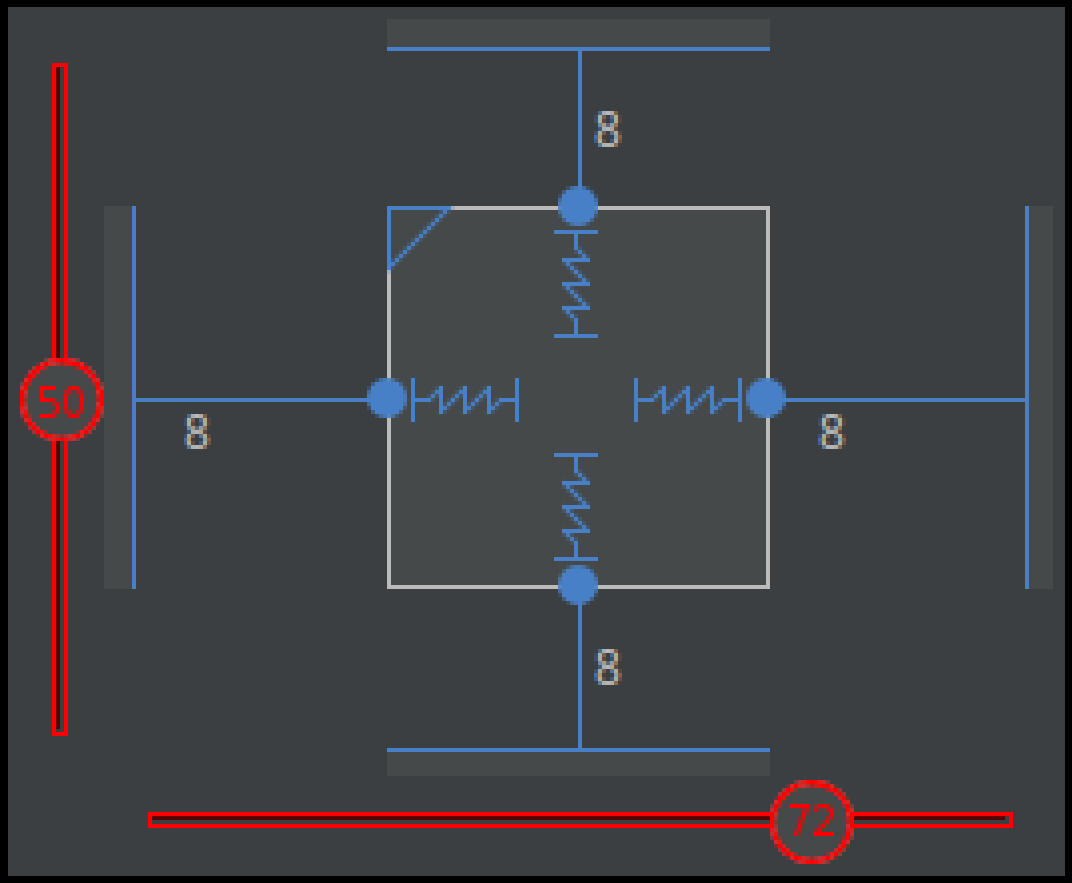
Inspector Graph Elements

- Size: wrap_content



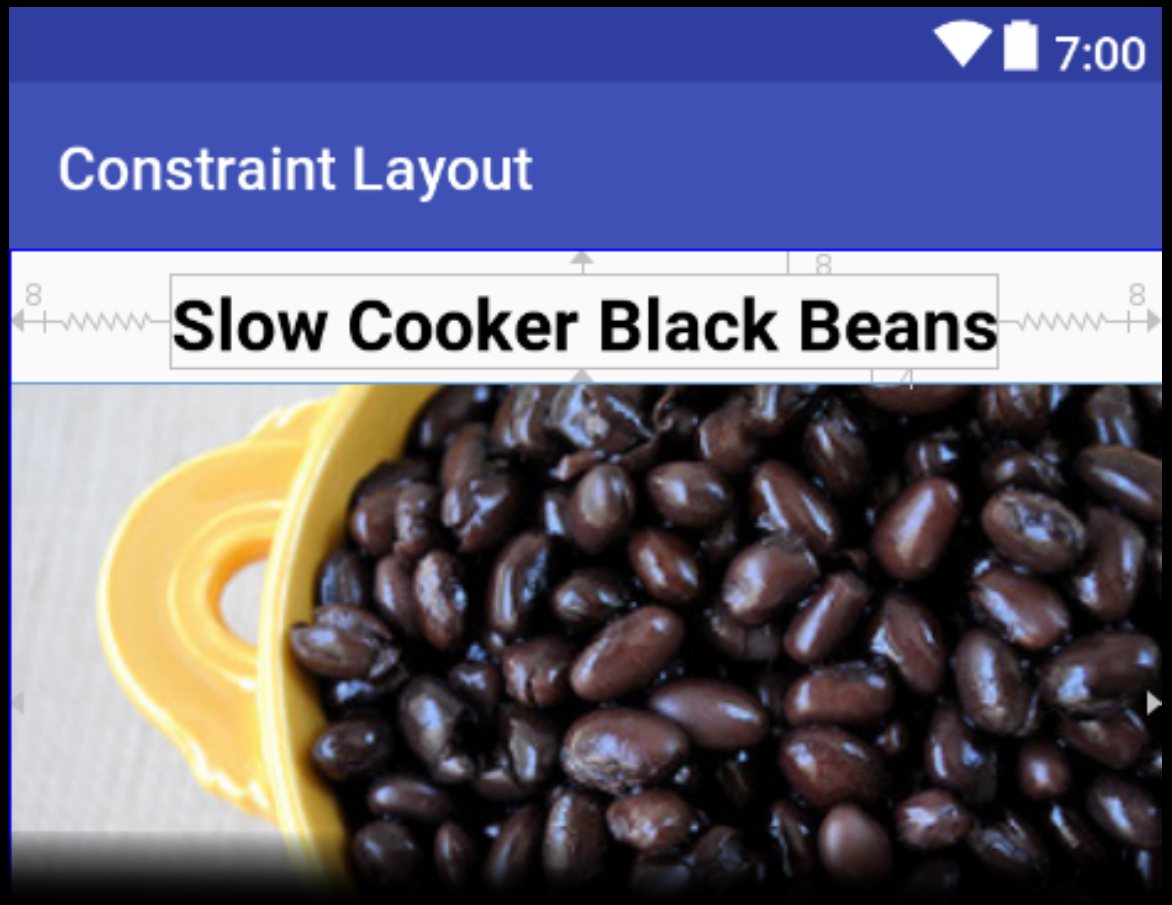
Inspector Graph Elements

- Bias



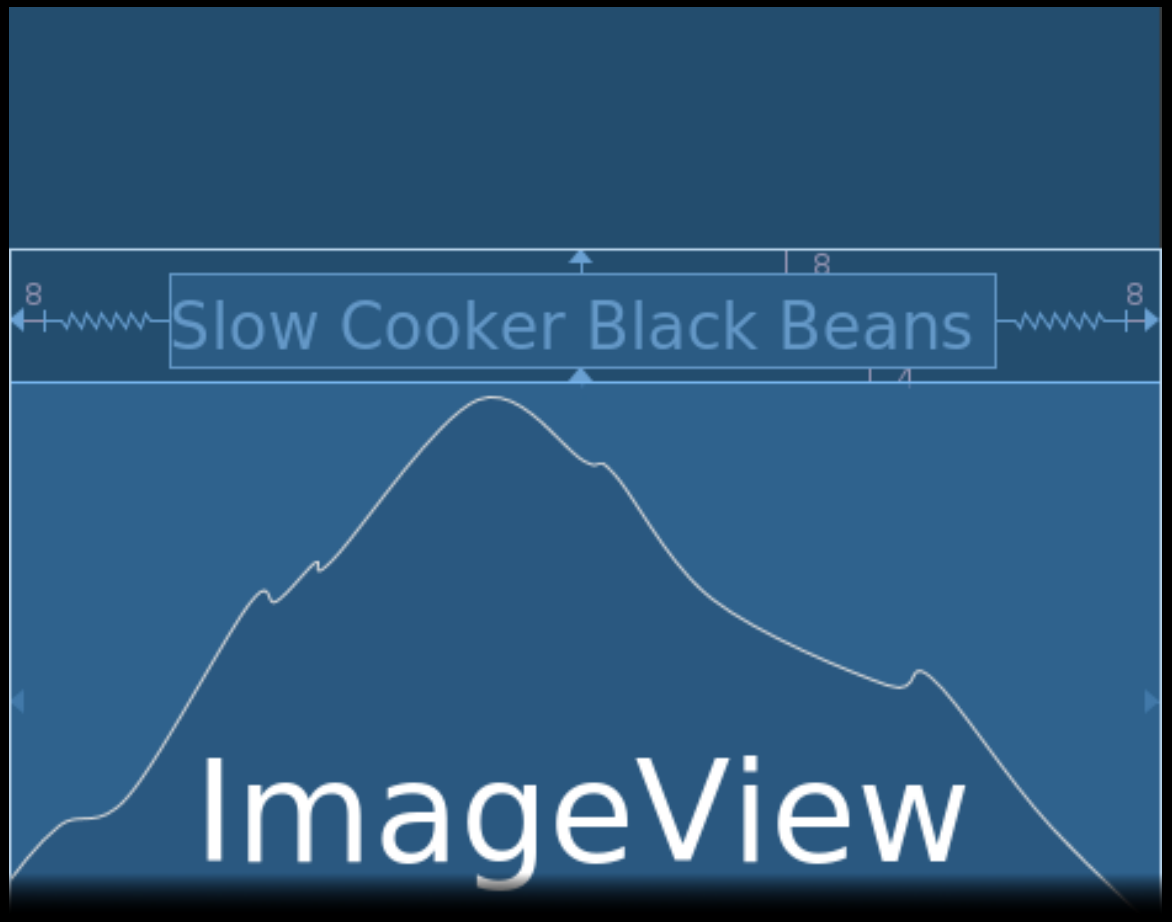
Design View Panels

- Design – mostly WYSIWYG version of layout



Design View Panels

- Blueprint – architectural version of layout



Toolbar Items

Click on:

-  to select layout versions to display

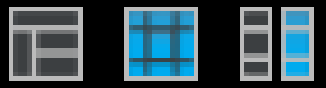
Toolbar Items

Click on:

-  to select layout versions to display
-  to show/hide constraints/margins

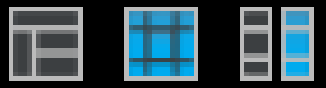
Toolbar Items

Click on:

-  to select layout versions to display
-  to show/hide constraints/margins
-  to toggle autoconnect mode

Toolbar Items

Click on:

-  to select layout versions to display
-  to show/hide constraints/margins
-  to toggle autoconnect mode
-  to delete ALL constraints

Toolbar Items

Click on:

-  to infer constraints for selected view

Toolbar Items

Click on:

-  to infer constraints for selected view
-  to set defaults margins

Toolbar Items

Click on:

-  to infer constraints for selected view
-  to set defaults margins
-  to distribute or size views

Toolbar Items

Click on:

-  to infer constraints for selected view
-  to set defaults margins
-  to distribute or size views
-  to align or center views (or create chains)

Autoconnect and Infer Constraints

- Autoconnect *mode* adds constraints between an added / moved view and its neighbors

Autoconnect and Infer Constraints

- Autoconnect *mode* adds constraints between an added / moved view and its neighbors
- Infer Constraints *action* adds constraints for all the views in the layout when it is clicked

Dimension Ratio

- Allows views to have explicit aspect ratio

Dimension Ratio

- Allows views to have explicit aspect ratio
- Set one dimension explicitly & other to “0dp”

Dimension Ratio

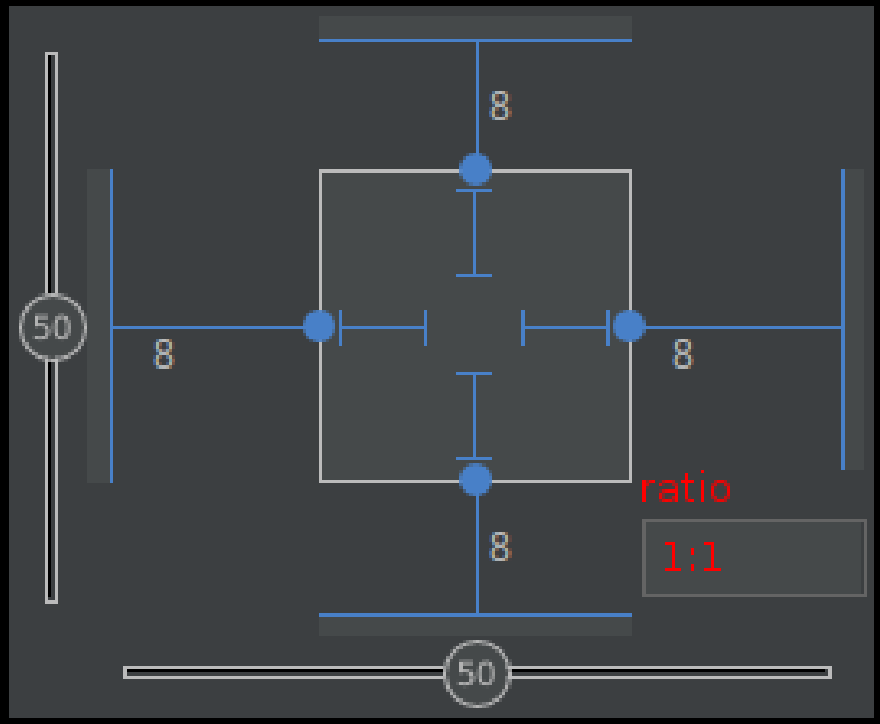
- Allows views to have explicit aspect ratio
- Set one dimension explicitly & other to “0dp”
- Ratio of width to height can then be specified

Dimension Ratio

- Manually add *layout_constraintDimensionRatio* to get ratio input will appear in the Inspector

Dimension Ratio

- Manually add *layout_constraintDimensionRatio* to get ratio input will appear in the Inspector



Replicating LinearLayout

- As views are added, add constraints from their top/left to bottom/right of the previous view.

Replicating LinearLayout

- As views are added, add constraints from their top/left to bottom/right of the previous view.
- Use weighted chains to replicate weights

Replicating RelativeLayout

- Use equivalent constraints instead of relative attributes such as *layout_ToStartOf*

Replicating RelativeLayout

- Use equivalent constraints instead of relative attributes such as *layout_ToStartOf*
- For a full list of equivalents, see chart at constraintlayout.github.io/layouts/relativelayout.html

Replicating RelativeLayout

- Use equivalent constraints instead of relative attributes such as *layout_ToStartOf*
- For a full list of equivalents, see chart at constraintlayout.github.io/layouts/relativelayout.html
- Constrain to opposite sides to replicate *centerHorizontal / Vertical* or *centerInParent*

Replicating PercentLayout

- Use percent-type guidelines for positioning

Replicating PercentLayout

- Use percent-type guidelines for positioning
- Use aspect ratio (*constraintDimensionRatio*) for sizing

Some Pitfalls

- Aspect Ratio doesn't appear in Inspector window until after being added manually

Some Pitfalls

- Aspect Ratio doesn't appear in Inspector window until after being added to XML
- Grab guidelines by their lines, not indicator

Some Pitfalls

- Aspect Ratio doesn't appear in Inspector window until after being added to XML
- Grab guidelines by their lines, not indicator
- Guidelines might not appear when first added until the layout is clicked

Some Pitfalls

- Sometimes hard to see constraints if views or margins are small

Some Pitfalls

- Sometimes hard to see constraints if views or margins are small
- Sometimes helpful to start with view dimensions set to absolute values

References

1. Google. “ConstraintLayout.” *Android Developers*, developer.android.com/reference/android/support/constraint/ConstraintLayout.html. Accessed June 5, 2017.

2. Google. “Using ConstraintLayout to design your views.” *Google Developers,* codelabs.developers.google.com/codelabs/constraint-layout. Accessed June 5, 2017.

3. Google. “What's New in Android Design Tools - New Features for Rapid UI Development (Google I/O '17).” *YouTube*, uploaded May 18, 2017, www.youtube.com/watch?v=nYb4FUdILZE.

4. Hugo, Huyen, Mark, Seb, Taylor, Wiebe, and Wolfram. “ConstraintLayout.” *ConstraintLayout.com*, constraintlayout.com. Accessed June 5, 2017.

5. Thisse, Scott. “ConstraintLayout-Talk”. *GitHub,* github.com/sathisse/ConstraintLayout-Talk. Accessed June 15, 2017.

Let's Create the Complicated Layout

The code is at:

github.com/sathisse/ConstraintLayout-Talk

References

1. Google. “ConstraintLayout.” *Android Developers*, developer.android.com/reference/android/support/constraint/ConstraintLayout.html. Accessed June 5, 2017.

2. Google. “Using ConstraintLayout to design your views.” *Google Developers,* codelabs.developers.google.com/codelabs/constraint-layout. Accessed June 5, 2017.

3. Google. “What's New in Android Design Tools - New Features for Rapid UI Development (Google I/O '17).” *YouTube*, uploaded May 18, 2017, www.youtube.com/watch?v=nYb4FUdILZE.

4. Hugo, Huyen, Mark, Seb, Taylor, Wiebe, and Wolfram. “ConstraintLayout.” *ConstraintLayout.com*, constraintlayout.com. Accessed June 5, 2017.

5. Thisse, Scott. “ConstraintLayout-Talk”. *GitHub,* github.com/sathisse/ConstraintLayout-Talk. Accessed June 15, 2017.

Q & A

Questions?