ConstraintLayout

```
public class ConstraintLayout
extends ViewGroup (http://developer.android.com/reference/android/view/ViewGroup.html)
```

java.lang.Object (http://developer.android.com/reference/java/lang/Object.html)
  ↳ android.view.View (http://developer.android.com/reference/android/view/View.html)
    ↳ android.view.ViewGroup (http://developer.android.com/reference/android/view/ViewGroup.html)
      ↳ android.support.constraint.ConstraintLayout

A `ConstraintLayout` is a `ViewGroup` (http://developer.android.com/reference/android/view/ViewGroup.html) which allows you to position and size widgets in a flexible way.

Note: `ConstraintLayout` is available as a support library that you can use on Android systems starting with API level 9 (Gingerbread). As such, we are planning on enriching its API and capabilities over time. This documentation will reflect those changes.

There are currently various types of constraints that you can use:

- Relative positioning (#RelativePositioning)
- Margins (#Margins)
- Centering positioning (#CenteringPositioning)
- Circular positioning (#CircularPositioning)
- Visibility behavior (#VisibilityBehavior)
- Dimension constraints (#DimensionConstraints)
- Chains (#Chains)
- Virtual Helpers objects (#VirtualHelpers)

- Optimizer (#Optimizer)

Note that you cannot have a circular dependency in constraints.

Also see `ConstraintLayout.LayoutParams` (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.LayoutParams.html) for layout attributes

Developer Guide

Relative positioning

Relative positioning is one of the basic building block of creating layouts in ConstraintLayout. Those constraints allow you to position a given widget relative to another one. You can constrain a widget on the horizontal and vertical axis:

- Horizontal Axis: left, right, start and end sides
- Vertical Axis: top, bottom sides and text baseline

The general concept is to constrain a given side of a widget to another side of any other widget.

For example, in order to position button B to the right of button A (Fig. 1):

Fig. 1 - Relative Positioning Example

you would need to do:

```
<Button android:id="@+id/buttonA" ... />
        <Button android:id="@+id/buttonB" ...
                app:layout_constraintLeft_toRightOf="@+id/buttonA" />
```

This tells the system that we want the left side of button B to be constrained to the right side of button A. Such a position constraint means that the system will try to have both sides share the same location.

Fig. 2 - Relative Positioning Constraints

Here is the list of available constraints (Fig. 2):

- `layout_constraintLeft_toLeftOf`
- `layout_constraintLeft_toRightOf`
- `layout_constraintRight_toLeftOf`
- `layout_constraintRight_toRightOf`
- `layout_constraintTop_toTopOf`
- `layout_constraintTop_toBottomOf`
- `layout_constraintBottom_toTopOf`
- `layout_constraintBottom_toBottomOf`
- `layout_constraintBaseline_toBaselineOf`
- `layout_constraintStart_toEndOf`
- `layout_constraintStart_toStartOf`
- `layout_constraintEnd_toStartOf`
- `layout_constraintEnd_toEndOf`

They all take a reference `id` to another widget, or the `parent` (which will reference the parent container, i.e. the ConstraintLayout):

```
<Button android:id="@+id/buttonB" ...
                app:layout_constraintLeft_toLeftOf="parent" />
```

Margins

Fig. 3 - Relative Positioning Margins

If side margins are set, they will be applied to the corresponding constraints (if they exist) (Fig. 3), enforcing the margin as a space between the target and the source side. The usual layout margin attributes can be used to this effect:

- `android:layout_marginStart`
- `android:layout_marginEnd`
- `android:layout_marginLeft`

- `android:layout_marginTop`
- `android:layout_marginRight`
- `android:layout_marginBottom`

Note that a margin can only be positive or equals to zero, and takes a `Dimension`.

Margins when connected to a GONE widget

When a position constraint target's visibility is `View.GONE`, you can also indicate a different margin value to be used using the following attributes:

- `layout_goneMarginStart`
- `layout_goneMarginEnd`
- `layout_goneMarginLeft`
- `layout_goneMarginTop`
- `layout_goneMarginRight`
- `layout_goneMarginBottom`

Centering positioning and bias

A useful aspect of `ConstraintLayout` is in how it deals with "impossible" constrains. For example, if we have something like:

```
<android.support.constraint.ConstraintLayout ...>
            <Button android:id="@+id/button" ...
                app:layout_constraintLeft_toLeftOf="parent"
                app:layout_constraintRight_toRightOf="parent/>
        </>
```

Unless the `ConstraintLayout` happens to have the exact same size as the `Button`, both constraints cannot be satisfied at the same time (both sides cannot be where we want them to be).

Fig. 4 - Centering Positioning

What happens in this case is that the constraints act like opposite forces pulling the widget apart equally (Fig. 4); such that the widget will end up being centered in the parent container. This will apply similarly for vertical constraints.

Bias

The default when encountering such opposite constraints is to center the widget; but you can tweak the positioning to favor one side over another using the bias attributes:

- `layout_constraintHorizontal_bias`
- `layout_constraintVertical_bias`

Fig. 5 - Centering Positioning with Bias

For example the following will make the left side with a 30% bias instead of the default 50%, such that the left side will be shorter, with the widget leaning more toward the left side (Fig. 5):

```
<android.support.constraint.ConstraintLayout ...>
            <Button android:id="@+id/button" ...
                app:layout_constraintHorizontal_bias="0.3"
                app:layout_constraintLeft_toLeftOf="parent"
                app:layout_constraintRight_toRightOf="parent/>
        </>
```

Using bias, you can craft User Interfaces that will better adapt to screen sizes changes.

Circular positioning (**Added in 1.1**)

You can constrain a widget center relative to another widget center, at an angle and a distance. This allows you to position a widget on a circle (see Fig. 6). The following attributes can be used:

- `layout_constraintCircle` : references another widget id
- `layout_constraintCircleRadius` : the distance to the other widget center
- `layout_constraintCircleAngle` : which angle the widget should be at (in degrees, from 0 to 360)

Fig. 6 - Circular Positioning

```
<Button android:id="@+id/buttonA" ... />
  <Button android:id="@+id/buttonB" ...
      app:layout_constraintCircle="@+id/buttonA"
      app:layout_constraintCircleRadius="100dp"
      app:layout_constraintCircleAngle="45" />
```

Visibility behavior

`ConstraintLayout` has a specific handling of widgets being marked as `View.GONE`.

`GONE` widgets, as usual, are not going to be displayed and are not part of the layout itself (i.e. their actual dimensions will not be changed if marked as `GONE`).

But in terms of the layout computations, `GONE` widgets are still part of it, with an important distinction:

- For the layout pass, their dimension will be considered as zero (basically, they will be resolved to a point)
- If they have constraints to other widgets they will still be respected, but any margins will be as if equals to zero

Fig. 7 - Visibility Behavior

This specific behavior allows to build layouts where you can temporarily mark widgets as being `GONE`, without breaking the layout (Fig. 7), which can be particularly useful when doing simple layout animations.

Note: The margin used will be the margin that B had defined when connecting to A (see Fig. 7 for an example). In some cases, this might not be the margin you want (e.g. A had a 100dp margin to the side of its container, B only a 16dp to A, marking A as gone, B will have a margin of 16dp to the container). For this reason, you can specify an alternate margin value to be used when the connection is to a widget being marked as gone (see the section above about the gone margin attributes (#GoneMargin)).

Dimensions constraints

Minimum dimensions on ConstraintLayout

You can define minimum and maximum sizes for the `ConstraintLayout` itself:

- `android:minWidth` set the minimum width for the layout
- `android:minHeight` set the minimum height for the layout
- `android:maxWidth` set the maximum width for the layout
- `android:maxHeight` set the maximum height for the layout

Those minimum and maximum dimensions will be used by `ConstraintLayout` when its dimensions are set to `WRAP_CONTENT`.

Widgets dimension constraints

The dimension of the widgets can be specified by setting the `android:layout_width` and `android:layout_height` attributes in 3 different ways:

- Using a specific dimension (either a literal value such as `123dp` or a `Dimension` reference)
- Using `WRAP_CONTENT`, which will ask the widget to compute its own size
- Using `0dp`, which is the equivalent of "`MATCH_CONSTRAINT`"

Fig. 8 - Dimension Constraints

The first two works in a similar fashion as other layouts. The last one will resize the widget in such a way as matching the constraints that are set (see Fig. 8, (a) is wrap_content, (b) is 0dp). If margins are set, they will be taken in account in the computation (Fig. 8, (c) with 0dp).

Important: `MATCH_PARENT` is not recommended for widgets contained in a `ConstraintLayout`. Similar behavior can be defined by using `MATCH_CONSTRAINT` with the corresponding left/right or top/bottom constraints being set to `"parent"`.

WRAP_CONTENT : enforcing constraints (*Added in 1.1*)

If a dimension is set to `WRAP_CONTENT`, in versions before 1.1 they will be treated as a literal dimension -- meaning, constraints will not limit the resulting dimension. While in general this is enough (and faster), in

some situations, you might want to use `WRAP_CONTENT`, yet keep enforcing constraints to limit the resulting dimension. In that case, you can add one of the corresponding attribute:

- `app:layout_constrainedWidth="true|false"`
- `app:layout_constrainedHeight="true|false"`

MATCH_CONSTRAINT dimensions (*Added in 1.1*)

When a dimension is set to `MATCH_CONSTRAINT`, the default behavior is to have the resulting size take all the available space. Several additional modifiers are available:

- `layout_constraintWidth_min` and `layout_constraintHeight_min` : will set the minimum size for this dimension
- `layout_constraintWidth_max` and `layout_constraintHeight_max` : will set the maximum size for this dimension
- `layout_constraintWidth_percent` and `layout_constraintHeight_percent` : will set the size of this dimension as a percentage of the parent

Min and Max

The value indicated for min and max can be either a dimension in Dp, or "wrap", which will use the same value as what `WRAP_CONTENT` would do.

Percent dimension

To use percent, you need to set the following:

- The dimension should be set to `MATCH_CONSTRAINT` (0dp)
- The default should be set to percent `app:layout_constraintWidth_default="percent"` or `app:layout_constraintHeight_default="percent"`
  (**Note:** this is necessary in 1.1-beta1 and 1.1-beta2, but will not be needed in following versions if the percent attribute is defined)
- Then set the `layout_constraintWidth_percent` or `layout_constraintHeight_percent` attributes to a value between 0 and 1

Ratio

You can also define one dimension of a widget as a ratio of the other one. In order to do that, you need to have at least one constrained dimension be set to `0dp` (i.e., `MATCH_CONSTRAINT`), and set the attribute `layout_constraintDimensionRatio` to a given ratio. For example:

```
<Button android:layout_width="wrap_content"
          android:layout_height="0dp"
          app:layout_constraintDimensionRatio="1:1" />
```

will set the height of the button to be the same as its width.

The ratio can be expressed either as:

- a float value, representing a ratio between width and height
- a ratio in the form "width:height"

You can also use ratio if both dimensions are set to `MATCH_CONSTRAINT` (0dp). In this case the system sets the largest dimensions the satisfies all constraints and maintains the aspect ratio specified. To constrain one specific side based on the dimensions of another, you can pre append `W,`" or `H,` to constrain the width or height respectively. For example, If one dimension is constrained by two targets (e.g. width is 0dp and centered on parent) you can indicate which side should be constrained, by adding the letter `W` (for constraining the width) or `H` (for constraining the height) in front of the ratio, separated by a comma:

```
<Button android:layout_width="0dp"
          android:layout_height="0dp"
          app:layout_constraintDimensionRatio="H,16:9"
          app:layout_constraintBottom_toBottomOf="parent"
          app:layout_constraintTop_toTopOf="parent"/>
```

will set the height of the button following a 16:9 ratio, while the width of the button will match the constraints to parent.

Chains

Chains provide group-like behavior in a single axis (horizontally or vertically). The other axis can be constrained independently.

Creating a chain

A set of widgets are considered a chain if they are linked together via a bi-directional connection (see Fig. 9, showing a minimal chain, with two widgets).

Fig. 9 - Chain

Chain heads

Chains are controlled by attributes set on the first element of the chain (the "head" of the chain):

Fig. 10 - Chain Head

The head is the left-most widget for horizontal chains, and the top-most widget for vertical chains.

Margins in chains

If margins are specified on connections, they will be taken in account. In the case of spread chains, margins will be deducted from the allocated space.

Chain Style

When setting the attribute `layout_constraintHorizontal_chainStyle` or `layout_constraintVertical_chainStyle` on the first element of a chain, the behavior of the chain will change according to the specified style (default is `CHAIN_SPREAD`).

- `CHAIN_SPREAD` -- the elements will be spread out (default style)
- Weighted chain -- in `CHAIN_SPREAD` mode, if some widgets are set to `MATCH_CONSTRAINT`, they will split the available space
- `CHAIN_SPREAD_INSIDE` -- similar, but the endpoints of the chain will not be spread out
- `CHAIN_PACKED` -- the elements of the chain will be packed together. The horizontal or vertical bias attribute of the child will then affect the positioning of the packed elements

Fig. 11 - Chains Styles

Weighted chains

The default behavior of a chain is to spread the elements equally in the available space. If one or more elements are using `MATCH_CONSTRAINT`, they will use the available empty space (equally divided among themselves). The attribute `layout_constraintHorizontal_weight` and `layout_constraintVertical_weight` will control how the space will be distributed among the elements using `MATCH_CONSTRAINT`. For exemple, on a chain containing two elements using `MATCH_CONSTRAINT`, with the first element using a weight of 2 and the second a weight of 1, the space occupied by the first element will be twice that of the second element.

Margins and chains (*in 1.1*)

When using margins on elements in a chain, the margins are additive.

For example, on a horizontal chain, if one element defines a right margin of 10dp and the next element defines a left margin of 5dp, the resulting margin between those two elements is 15dp.

An item plus its margins are considered together when calculating leftover space used by chains to position items. The leftover space does not contain the margins.

Virtual Helper objects

In addition to the intrinsic capabilities detailed previously, you can also use special helper objects in `ConstraintLayout` to help you with your layout. Currently, the `Guideline` object allows you to create Horizontal and Vertical guidelines which are positioned relative to the `ConstraintLayout` container. Widgets can then be positioned by constraining them to such guidelines. In **1.1**, `Barrier` and `Group` were added too.

Optimizer (*in 1.1*)

In 1.1 we exposed the constraints optimizer. You can decide which optimizations are applied by adding the tag *app:layout_optimizationLevel* to the ConstraintLayout element.

- **none** : no optimizations are applied
- **standard** : Default. Optimize direct and barrier constraints only
- **direct** : optimize direct constraints
- **barrier** : optimize barrier constraints
- **chain** : optimize chain constraints (experimental)

- **dimensions** : optimize dimensions measures (experimental), reducing the number of measures of match constraints elements

This attribute is a mask, so you can decide to turn on or off specific optimizations by listing the ones you want. For example: *app:layout_optimizationLevel="direct|barrier|chain"*

See also:

`Guideline` (https://developer.android.com/reference/android/support/constraint/Guideline.html)

Summary

Nested classes

`class`	`ConstraintLayout.LayoutParams` (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.LayoutParams.html) This class contains the different attributes specifying how a view want to be laid out inside a `ConstraintLayout` (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html) .

Inherited constants

From class `android.view.ViewGroup`

`int`	`CLIP_TO_PADDING_MASK`
`int`	`FOCUS_AFTER_DESCENDANTS`
`int`	`FOCUS_BEFORE_DESCENDANTS`
`int`	`FOCUS_BLOCK_DESCENDANTS`
`int`	`LAYOUT_MODE_CLIP_BOUNDS`

int	LAYOUT_MODE_OPTICAL_BOUNDS
int	PERSISTENT_ALL_CACHES
int	PERSISTENT_ANIMATION_CACHE
int	PERSISTENT_NO_CACHE
int	PERSISTENT_SCROLLING_CACHE

From class `android.view.View`

int	ACCESSIBILITY_LIVE_REGION_ASSERTIVE
int	ACCESSIBILITY_LIVE_REGION_NONE
int	ACCESSIBILITY_LIVE_REGION_POLITE
int	DRAG_FLAG_GLOBAL
int	DRAG_FLAG_GLOBAL_PERSISTABLE_URI_PERMISSION
int	DRAG_FLAG_GLOBAL_PREFIX_URI_PERMISSION
int	DRAG_FLAG_GLOBAL_URI_READ
int	DRAG_FLAG_GLOBAL_URI_WRITE
int	DRAG_FLAG_OPAQUE
int	DRAWING_CACHE_QUALITY_AU

	O
int	DRAWING_CACHE_QUALITY_HI H
int	DRAWING_CACHE_QUALITY_LO
int	FIND_VIEWS_WITH_CONTENT_ ESCRIPTION
int	FIND_VIEWS_WITH_TEXT
int	FOCUSABLES_ALL
int	FOCUSABLES_TOUCH_MODE
int	FOCUS_BACKWARD
int	FOCUS_DOWN
int	FOCUS_FORWARD
int	FOCUS_LEFT
int	FOCUS_RIGHT
int	FOCUS_UP
int	GONE
int	HAPTIC_FEEDBACK_ENABLED
int	IMPORTANT_FOR_ACCESSIBIL TY_AUTO

int	IMPORTANT_FOR_ACCESSIBILITY_NO
int	IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS
int	IMPORTANT_FOR_ACCESSIBILITY_YES
int	INVISIBLE
int	KEEP_SCREEN_ON
int	LAYER_TYPE_HARDWARE
int	LAYER_TYPE_NONE
int	LAYER_TYPE_SOFTWARE
int	LAYOUT_DIRECTION_INHERIT
int	LAYOUT_DIRECTION_LOCALE
int	LAYOUT_DIRECTION_LTR
int	LAYOUT_DIRECTION_RTL
int	MEASURED_HEIGHT_STATE_SHIFT
int	MEASURED_SIZE_MASK
int	MEASURED_STATE_MASK
int	MEASURED_STATE_TOO_SMALL

int	NO_ID
int	OVER_SCROLL_ALWAYS
int	OVER_SCROLL_IF_CONTENT_S ROLLS
int	OVER_SCROLL_NEVER
int	SCREEN_STATE_OFF
int	SCREEN_STATE_ON
int	SCROLLBARS_INSIDE_INSET
int	SCROLLBARS_INSIDE_OVERLA
int	SCROLLBARS_OUTSIDE_INSET
int	SCROLLBARS_OUTSIDE_OVERL Y
int	SCROLLBAR_POSITION_DEFAU T
int	SCROLLBAR_POSITION_LEFT
int	SCROLLBAR_POSITION_RIGHT
int	SCROLL_AXIS_HORIZONTAL
int	SCROLL_AXIS_NONE
int	SCROLL_AXIS_VERTICAL

int	SCROLL_INDICATOR_BOTTOM
int	SCROLL_INDICATOR_END
int	SCROLL_INDICATOR_LEFT
int	SCROLL_INDICATOR_RIGHT
int	SCROLL_INDICATOR_START
int	SCROLL_INDICATOR_TOP
int	SOUND_EFFECTS_ENABLED
int	STATUS_BAR_HIDDEN
int	STATUS_BAR_VISIBLE
int	SYSTEM_UI_FLAG_FULLSCREE
int	SYSTEM_UI_FLAG_HIDE_NAVI ATION
int	SYSTEM_UI_FLAG_IMMERSIVE
int	SYSTEM_UI_FLAG_IMMERSIVE STICKY
int	SYSTEM_UI_FLAG_LAYOUT_FU LSCREEN
int	SYSTEM_UI_FLAG_LAYOUT_HI E_NAVIGATION
int	SYSTEM_UI_FLAG_LAYOUT_ST BLE

int	SYSTEM_UI_FLAG_LIGHT_STA US_BAR
int	SYSTEM_UI_FLAG_LOW_PROFI E
int	SYSTEM_UI_FLAG_VISIBLE
int	SYSTEM_UI_LAYOUT_FLAGS
int	TEXT_ALIGNMENT_CENTER
int	TEXT_ALIGNMENT_GRAVITY
int	TEXT_ALIGNMENT_INHERIT
int	TEXT_ALIGNMENT_TEXT_END
int	TEXT_ALIGNMENT_TEXT_STAR
int	TEXT_ALIGNMENT_VIEW_END
int	TEXT_ALIGNMENT_VIEW_STAR
int	TEXT_DIRECTION_ANY_RTL
int	TEXT_DIRECTION_FIRST_STR NG
int	TEXT_DIRECTION_FIRST_STR NG_LTR
int	TEXT_DIRECTION_FIRST_STR NG_RTL

int	TEXT_DIRECTION_INHERIT
int	TEXT_DIRECTION_LOCALE
int	TEXT_DIRECTION_LTR
int	TEXT_DIRECTION_RTL
String (http://developer.android.com/reference/java/lang/String.html)	VIEW_LOG_TAG
int	VISIBLE

Inherited fields

From class android.view.View

public static final Property (http://developer.android.com/reference/android/util/Property.htm

protected static final int[]
protected static final int[]
protected static final int[]
protected static final int[]
protected static final int[]
protected static final int[]
protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

protected static final int[]

public static final Property (http://developer.android.com/reference/android/util/Property.htm

public static final Property (http://developer.android.com/reference/android/util/Property.htm

public static final Property (http://developer.android.com/reference/android/util/Property.htm

public static final Property (http://developer.android.com/reference/android/util/Property.htm

public static final Property (http://developer.android.com/reference/android/util/Property.htm

protected static final int[]

`protected static final int[]`

`public static final Property` (http://developer.android.com/reference/android/util/Property.htm

`public static final Property` (http://developer.android.com/reference/android/util/Property.htm

`public static final Property` (http://developer.android.com/reference/android/util/Property.htm

`protected static final int[]`

`public static final Property` (http://developer.android.com/reference/android/util/Property.htm

`public static final Property` (http://developer.android.com/reference/android/util/Property.htm

`public static final Property` (http://developer.android.com/reference/android/util/Property.htm

Public constructors

`ConstraintLayout`
(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#ConstraintLayout(android.content.Context))
(`Context` (http://developer.android.com/reference/android/content/Context.html) `context`)

`ConstraintLayout`
(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#ConstraintLayout(android.content.Context, android.util.AttributeSet))
(`Context` (http://developer.android.com/reference/android/content/Context.html) `context`, `AttributeSet` (http://developer.android.com/reference/android/util/AttributeSet.html) `attrs`)

`ConstraintLayout`
(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#ConstraintLayout(android.content.Context, android.util.AttributeSet, int))
(`Context` (http://developer.android.com/reference/android/content/Context.html) `context`, `AttributeSet` (http://developer.android.com/reference/android/util/AttributeSet.html) `attrs, int defStyleAttr`)

Public methods

int	getMaxHeight (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#getMaxHeight()) () The maximum height of this view.
int	getMaxWidth (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#getMaxWidth()) ()
int	getMinHeight (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#getMinHeight()) () The minimum height of this view.
int	getMinWidth (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#getMinWidth()) () The minimum width of this view.
int	getOptimizationLevel (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#getOptimizationLevel()) () Return the current optimization level for the layout resolution
void	requestLayout (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#requestLayout()) ()
void	setConstraintSet (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setConstraintSet(android.support.constraint.ConstraintSet)) (ConstraintSet

	(https://developer.android.com/reference/android/support/constraint/ConstraintSet.html) `set)` Sets a ConstraintSet object to manage constraints.

`void`	`setMaxHeight`
	(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setMaxHeight(int))
	`(int value)`
	
	Set the max height for this view

`void`	`setMaxWidth`
	(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setMaxWidth(int))
	`(int value)`
	
	Set the max width for this view

`void`	`setMinHeight`
	(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setMinHeight(int))
	`(int value)`
	
	Set the min height for this view

`void`	`setMinWidth`
	(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setMinWidth(int))
	`(int value)`
	
	Set the min width for this view

`void`	`setOptimizationLevel`
	(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setOptimizationLevel(int))
	`(int level)`
	
	Set the optimization for the layout resolution.

Protected methods

`boolean`

`ConstraintLayout.LayoutParams`
(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.LayoutParams.html

`ViewGroup.LayoutParams`
(http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html)

`void`

`void`

Inherited methods

From class `android.view.ViewGroup` (http://developer.android.com/reference/android/view/View(

`void`

`void`

`boolean`

`void`

void

void

void

void

void

boolean

boolean

void

void

void

boolean

boolean

void

void

void
void
void
void
void
void
void
void
[**WindowInsets**](http://developer.android.com/reference/android/view/WindowInsets.html)
void
void
boolean

void
void
void
boolean
boolean
boolean
boolean
boolean

boolean

void

void

void

void

void

void

void

void

boolean

`boolean`
`boolean`
`void`
`void`
`void`
`void`
`boolean`
`void`
`void`
[`View`](http://developer.android.com/reference/android/view/View.html)
`void`

View (http://developer.android.com/reference/android/view/View.html)

void

boolean

ViewGroup.LayoutParams (http://developer.android.com/reference/android/view/ViewGroup.LayoutP

ViewGroup.LayoutParams (http://developer.android.com/reference/android/view/ViewGroup.LayoutP

ViewGroup.LayoutParams (http://developer.android.com/reference/android/view/ViewGroup.LayoutP

CharSequence (http://developer.android.com/reference/java/lang/CharSequence.html)

View (http://developer.android.com/reference/android/view/View.html)

int

int

`static int`
`boolean`
`boolean`
`boolean`
`boolean`
`int`
`View` (http://developer.android.com/reference/android/view/View.html)
`LayoutAnimationController` (http://developer.android.com/reference/android/view/animation/Layo
`Animation.AnimationListener` (http://developer.android.com/reference/android/view/animation/A
`int`
`LayoutTransition` (http://developer.android.com/reference/android/animation/LayoutTransition.html)
`int`
`ViewOverlay` (http://developer.android.com/reference/android/view/ViewOverlay.html)
`int`
`boolean`
`boolean`

`boolean`
`boolean`
`int`
`final void`
[`ViewParent`](http://developer.android.com/reference/android/view/ViewParent.html)
`boolean`
`boolean`
`boolean`
`boolean`
`boolean`
`boolean`
`void`
`final void`
`void`
`void`

void

void

final void

final void

void

int[]

void

boolean

boolean

abstract void

`boolean`

`boolean`

`boolean`

`void`

`void`

`void`

`boolean`

boolean
[PointerIcon](http://developer.android.com/reference/android/view/PointerIcon.html)
boolean
void
void
void
void
void

void
void
void
void
void
void
void
void
boolean
void
boolean

boolean
void
void
void
void
void
void
void
void
void
void
void

void

void

void

void

void

void

void

void

void

boolean

boolean

boolean

ActionMode (http://developer.android.com/reference/android/view/ActionMode.html)

ActionMode (http://developer.android.com/reference/android/view/ActionMode.html)

void

void

void

From class **android.view.View** (http://developer.android.com/reference/android/view/View.html)

void

void

void
void
void
void
ViewPropertyAnimator (http://developer.android.com/reference/android/view/ViewPropertyAnimator
void

boolean

boolean

boolean

void

void

void

void

boolean

boolean

boolean

boolean

boolean

boolean

boolean

final void

void

final void

boolean

void

void

`static int`

`int`

`int`

`int`

`void`

`WindowInsets` (http://developer.android.com/reference/android/view/WindowInsets.html)

`int`

`int`

`int`

`AccessibilityNodeInfo` (http://developer.android.com/reference/android/view/accessibility/Accessi

`void`

`void`

`WindowInsets` (http://developer.android.com/reference/android/view/WindowInsets.html)

void

void

boolean

void

void

void

boolean

boolean

boolean

boolean

boolean

boolean

boolean

boolean

boolean

boolean

boolean

boolean

boolean

void

void

void

void
void
void
void
void
boolean
boolean
boolean
void
void

`void`
`void`
`void`
`void`
`void`
[`View`](http://developer.android.com/reference/android/view/View.html)
`final` [`View`](http://developer.android.com/reference/android/view/View.html)
`final` [`View`](http://developer.android.com/reference/android/view/View.html)
`void`
`boolean`

View (http://developer.android.com/reference/android/view/View.html)

void

void

static int

CharSequence (http://developer.android.com/reference/java/lang/CharSequence.html)

int

AccessibilityNodeProvider (http://developer.android.com/reference/android/view/accessibility/Ac

int

int

float

Animation (http://developer.android.com/reference/android/view/animation/Animation.html)

IBinder (http://developer.android.com/reference/android/os/IBinder.html)

Drawable (http://developer.android.com/reference/android/graphics/drawable/Drawable.html)

ColorStateList (http://developer.android.com/reference/android/content/res/ColorStateList.html)

PorterDuff.Mode (http://developer.android.com/reference/android/graphics/PorterDuff.Mode.html)

int

final int

float

int

float

boolean

Rect (http://developer.android.com/reference/android/graphics/Rect.html)
final boolean
CharSequence (http://developer.android.com/reference/java/lang/CharSequence.html)
final Context (http://developer.android.com/reference/android/content/Context.html)
ContextMenu.ContextMenuInfo (http://developer.android.com/reference/android/view/ContextMenu
static int
Display (http://developer.android.com/reference/android/view/Display.html)
final int[]
Bitmap (http://developer.android.com/reference/android/graphics/Bitmap.html)
Bitmap (http://developer.android.com/reference/android/graphics/Bitmap.html)
int
int
void
long
float
boolean
boolean
ArrayList (http://developer.android.com/reference/java/util/ArrayList.html)<View (http://developer.and

`void`

`Drawable` (http://developer.android.com/reference/android/graphics/drawable/Drawable.html)

`int`

`ColorStateList` (http://developer.android.com/reference/android/content/res/ColorStateList.html)

`PorterDuff.Mode` (http://developer.android.com/reference/android/graphics/PorterDuff.Mode.html)

`final boolean`

`boolean`

`Handler` (http://developer.android.com/reference/android/os/Handler.html)

`final boolean`

`final int`

`void`

`int`

`int`

`int`
`int`
`boolean`
`KeyEvent.DispatcherState` (http://developer.android.com/reference/android/view/KeyEvent.Dispatc
`int`
`int`
`int`
`ViewGroup.LayoutParams` (http://developer.android.com/reference/android/view/ViewGroup.LayoutP
`final int`
`float`
`int`
`final boolean`
`void`
`void`
`Matrix` (http://developer.android.com/reference/android/graphics/Matrix.html)
`final int`
`final int`
`final int`
`final int`

`final int`
`int`
`int`
`int`
`int`
`int`
`int`
`int`
`int`
`View.OnFocusChangeListener` (http://developer.android.com/reference/android/view/View.OnFocus
`ViewOutlineProvider` (http://developer.android.com/reference/android/view/ViewOutlineProvider.htn
`int`
`ViewOverlay` (http://developer.android.com/reference/android/view/ViewOverlay.html)
`int`
`int`
`int`
`int`
`int`
`int`
`int`
`final ViewParent` (http://developer.android.com/reference/android/view/ViewParent.html)
`ViewParent` (http://developer.android.com/reference/android/view/ViewParent.html)
`float`
`float`
`PointerIcon` (http://developer.android.com/reference/android/view/PointerIcon.html)
`Resources` (http://developer.android.com/reference/android/content/res/Resources.html)

`final int`
`float`
`int`
`View` (http://developer.android.com/reference/android/view/View.html)
`WindowInsets` (http://developer.android.com/reference/android/view/WindowInsets.html)
`float`
`float`
`float`
`float`
`float`
`int`
`int`
`int`
`int`
`int`
`final int`
`final int`
`int`
`StateListAnimator` (http://developer.android.com/reference/android/animation/StateListAnimator.ht
`int`
`int`
`int`

Object (http://developer.android.com/reference/java/lang/Object.html)
Object (http://developer.android.com/reference/java/lang/Object.html)
int
int
final int
float
int
TouchDelegate (http://developer.android.com/reference/android/view/TouchDelegate.html)
ArrayList (http://developer.android.com/reference/java/util/ArrayList.html)<View (http://developer.and
String (http://developer.android.com/reference/java/lang/String.html)
float
float
float
int
int
int
ViewTreeObserver (http://developer.android.com/reference/android/view/ViewTreeObserver.html)
int
final int
int
WindowId (http://developer.android.com/reference/android/view/WindowId.html)
int

IBinder (http://developer.android.com/reference/android/os/IBinder.html)
int
void
float
float
float
boolean
boolean
boolean
boolean
boolean
boolean
boolean
static View (http://developer.android.com/reference/android/view/View.html)
void
void

void
void
void
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
final boolean
final boolean
boolean
boolean
boolean
boolean
boolean

boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
final boolean
boolean

`boolean`

`boolean`

`boolean`

`void`

`void`

`final void`

`static int[]`

`void`

`void`

`void`

`void`

`WindowInsets` (http://developer.android.com/reference/android/view/WindowInsets.html)

`void`

`void`

`boolean`

`void`

`void`
`int[]`
[`InputConnection`](http://developer.android.com/reference/android/view/inputmethod/InputConnectio
`void`
`void`
`boolean`
`void`
`void`
`final void`

boolean

void

void

void

boolean

void

boolean

void

void

boolean

boolean

boolean

boolean

boolean

boolean

void
void
void
void
void
void
PointerIcon (http://developer.android.com/reference/android/view/PointerIcon.html)
void

void
[Parcelable](http://developer.android.com/reference/android/os/Parcelable.html)
void
void
boolean
void
void
boolean
boolean
void
void
void

void
void
boolean
boolean
boolean
boolean
boolean
boolean
boolean
boolean
void
boolean
boolean

void
void
void
void
void
void
void
void
boolean
void

void

void

void

final boolean

final boolean

boolean

final boolean

void

boolean

boolean

final void

static int

static int

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void
void
void
final void
void
void
void
void
void
void

void
void
void
void
void
void
void
void
void
void
void
void
void
void

void
void
void
void
void
void
void
void
void
void
void
void
void

void

final void

void

final void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void

void
void
void
void
void
void
void
final void
void
void
void
void
void
void
void

void
void
void
void
void
void
void
void
void
void
void
void
void
void
void

void
void
final void
void
final void
void
void
void
void
void
void
void
void
void

`void`

`void`

`void`

`boolean`

`boolean`

`ActionMode` (http://developer.android.com/reference/android/view/ActionMode.html)

`ActionMode` (http://developer.android.com/reference/android/view/ActionMode.html)

`void`

`final boolean`

`final boolean`

`boolean`

`void`

`String` (http://developer.android.com/reference/java/lang/String.html)

`void`

`void`

`final void`

`boolean`

boolean	
boolean	

From class java.lang.Object (http://developer.android.com/reference/java/lang/Object.html)

Object (http://developer.android.com/reference/java/lang/Object.html)	clone()
boolean	equals(Object (htt
void	finalize()
final Class (http://developer.android.com/reference/java/lang/Class.html)<?>	getClass()
int	hashCode()
final void	notify()
final void	notifyAll()
String (http://developer.android.com/reference/java/lang/String.html)	toString()
final void	wait(long arg0,
final void	wait(long arg0)
final void	wait()

From interface android.view.ViewParent (http://developer.android.com/reference/android/view/V

abstract void	br (ht
abstract boolean	can
abstract boolean	can
abstract boolean	can
abstract void	chi (htt
abstract void	chi (htt

	arg
abstract void	**cle** (htt
abstract void	**cre** (htt
abstract View (http://developer.android.com/reference/android/view/View.html)	**foc** (htt **arg**
abstract void	**foc** (htt
abstract boolean	**get** (htt (htt (htt
abstract int	**get**
abstract ViewParent (http://developer.android.com/reference/android/view/ViewParent.html)	**get**
abstract ViewParent (http://developer.android.com/reference/android/view/ViewParent.html)	**get**
abstract int	**get**
abstract int	**get**
abstract void	**inv** (htt (htt
abstract ViewParent (http://developer.android.com/reference/android/view/ViewParent.html)	**inv** (htt
abstract boolean	**isL**
abstract boolean	**isL**
abstract boolean	**isT**
abstract boolean	**isT**
abstract void	**not** (htt

	(htt **arg**
abstract boolean	**onN** (htt **arg**
abstract boolean	**onN** (htt **arg**
abstract boolean	**onN** (htt **arg** **arg**
abstract void	**onN** (htt **arg**
abstract void	**onN** (htt **arg**
abstract void	**onN** (htt (htt **arg**
abstract boolean	**onS** (htt (htt **arg**
abstract void	**onS** (htt
abstract void	**rec** (htt
abstract void	**req** (htt (htt
abstract boolean	**req** (htt

	(htt **boo**
abstract void	**req**
abstract void	**req**
abstract void	**req**
abstract boolean	**req** (htt **Acc** (htt t.htr **arg**
abstract void	**req** (htt
abstract boolean	**sho** (htt
abstract boolean	**sho** (htt **arg**
abstract ActionMode (http://developer.android.com/reference/android/view/ActionMode.html)	**sta** (htt **Act** (htt **arg**
abstract ActionMode (http://developer.android.com/reference/android/view/ActionMode.html)	**st** (ht **Ac** (ht **ar**

From interface **android.view.ViewManager** (http://developer.android.com/reference/android/view/

abstract void	**addView(View** (http://developer.android.com/reference/android/view (http://developer.android.com/reference/android/view/ViewGroup.Lay
abstract void	**removeView(View** (http://developer.android.com/reference/android/vi
abstract void	**updateViewLayout(View** (http://developer.android.com/reference/a (http://developer.android.com/reference/android/view/ViewGroup.Lay

From interface android.graphics.drawable.Drawable.Callback (http://developer.android.cor

abstract void	invalidateDrawable(Drawable (http://developer.android.com/ref
abstract void	scheduleDrawable(Drawable (http://developer.android.com/referen (http://developer.android.com/reference/java/lang/Runnable.html) arg
abstract void	unscheduleDrawable(Drawable (http://developer.android.com/ref (http://developer.android.com/reference/java/lang/Runnable.html) ar

From interface android.view.KeyEvent.Callback (http://developer.android.com/reference/andro

abstract boolean	onKeyDown(int arg0, KeyEvent (http://developer.android.com/re
abstract boolean	onKeyLongPress(int arg0, KeyEvent (http://developer.android.c
abstract boolean	onKeyMultiple(int arg0, int arg1, KeyEvent (http://develop
abstract boolean	onKeyUp(int arg0, KeyEvent (http://developer.android.com/refe

From interface android.view.accessibility.AccessibilityEventSource (http://developer.a

abstract void	sendAccessibilityEvent(int arg0)
abstract void	sendAccessibilityEventUnchecked(AccessibilityEvent (http://developer.android.com/reference/android/view/accessibility/A

Public constructors

ConstraintLayout

ConstraintLayout (Context (http://developer.android.com/reference/android/content/Context.html) c

Parameters

context	Context

ConstraintLayout

```
ConstraintLayout (Context (http://developer.android.com/reference/android/content/Context.html) c
                  AttributeSet (http://developer.android.com/reference/android/util/AttributeSet.htm
```

Parameters

context	**Context**
attrs	**AttributeSet**

ConstraintLayout

```
ConstraintLayout (Context (http://developer.android.com/reference/android/content/Context.html) c
                  AttributeSet (http://developer.android.com/reference/android/util/AttributeSet.htm
                  int defStyleAttr)
```

Parameters

context	**Context**
attrs	**AttributeSet**
defStyleAttr	**int**

Public methods

getMaxHeight

```
int getMaxHeight ()
```

The maximum height of this view.

Returns

`int`	The maximum height of this view

See also:

`setMaxHeight(int)`
(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setMaxHeight(int))

getMaxWidth

```
int getMaxWidth ()
```

Returns

`int`

getMinHeight

```
int getMinHeight ()
```

The minimum height of this view.

Returns

`int`	The minimum height of this view

See also:

`setMinHeight(int)`
(https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setMinHeight(int))

getMinWidth

```
int getMinWidth ()
```

The minimum width of this view.

Returns

int	The minimum width of this view

See also:

setMinWidth(int) (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#setMinWidth(int))

getOptimizationLevel

```
int getOptimizationLevel ()
```

Return the current optimization level for the layout resolution

Returns

int	the current level

requestLayout

```
void requestLayout ()
```

setConstraintSet

```
void setConstraintSet (ConstraintSet (https://developer.android.com/reference/android/support/
```

Sets a ConstraintSet object to manage constraints. The ConstraintSet overrides LayoutParams of child views.

Parameters

`set`	`ConstraintSet`: Layout children using ConstraintSet

setMaxHeight

```
void setMaxHeight (int value)
```

Set the max height for this view

setMaxWidth

```
void setMaxWidth (int value)
```

Set the max width for this view

setMinHeight

```
void setMinHeight (int value)
```

Set the min height for this view

setMinWidth

```
void setMinWidth (int value)
```

Set the min width for this view

setOptimizationLevel

```
void setOptimizationLevel (int level)
```

Set the optimization for the layout resolution. The optimization can be any of the following:

- Optimizer.OPTIMIZATION_NONE
- Optimizer.OPTIMIZATION_STANDARD
- a mask composed of specific optimizations

The mask can be composed of any combination of the following:

- Optimizer.OPTIMIZATION_DIRECT
- Optimizer.OPTIMIZATION_BARRIER
- Optimizer.OPTIMIZATION_CHAIN (experimental)
- Optimizer.OPTIMIZATION_DIMENSIONS (experimental)

Note that the current implementation of Optimizer.OPTIMIZATION_STANDARD is as a mask of DIRECT and BARRIER.

Parameters

`level`	`int`: optimization level

Protected methods

checkLayoutParams

`boolean checkLayoutParams (ViewGroup.LayoutParams (http://developer.android.com/reference/a`

Parameters

`p`	`ViewGroup.LayoutParams`

Returns

`boolean`

generateDefaultLayoutParams

ConstraintLayout.LayoutParams (https://developer.android.com/reference/android/support/constrair

Returns

ConstraintLayout.LayoutParams (https://developer.android.com/reference/android/support/constraint/ConstraintLayout.LayoutParams.html)	

generateLayoutParams

ViewGroup.LayoutParams (http://developer.android.com/reference/android/view/ViewGroup.LayoutPara

Parameters

p	ViewGroup.LayoutParams

Returns

ViewGroup.LayoutParams (http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html)	

onLayout

```
void onLayout (boolean changed,
               int left,
               int top,
               int right,
               int bottom)
```

Parameters

`changed`	`boolean`
`left`	`int`
`top`	`int`
`right`	`int`
`bottom`	`int`

onMeasure

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

Parameters

`widthMeasureSpec`	`int`
`heightMeasureSpec`	`int`

Twitter
Follow @AndroidDev on Twitter

Google+
Follow Android Developers on Google+

YouTube
Check out Android Developers on YouTube