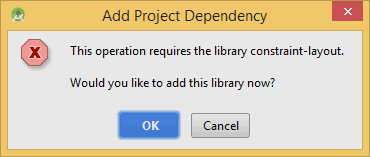
ConstraintLayout

Новый макет **ConstraintLayout** появился в Android Studio 2.2 и доступен для устройств с версии Android 2.3. Его можно найти в разделе **Layouts**. Гугл очень расхваливает данный макет и советует всем переходить на него и даже создал специальный конвертер для этой задачи.

Upd. В Android Studio 2.3 в шаблонах по умолчанию теперь используется **ConstraintLayout**, поэтому часть текста в статье можно пропустить.





В **build.gradle** модуля прописывается ссылка на библиотеку.

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

Если вы будете создавать макет с нуля и выберите данный макет для экрана с панели инструментов, то вам также предложат установить библиотеку.

**ConstraintLayout** является наследником **ViewGroup** и местами похож на **RelativeLayout**, но более продвинут. Код разметки в XML-представлении:

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/activity\_main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="ru.alexanderklimov.as22.MainActivity">

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="16dp"

android:text="Hello World!"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

tools:layout\_constraintLeft\_creator="1"

tools:layout\_constraintRight\_creator="1" />

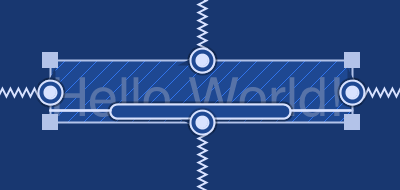
</android.support.constraint.ConstraintLayout>

Управление компонентами внутри данного контейнера достаточно сильно отличается от старого взаимодействия. Придётся всем переучиваться.

При его использовании нет смысла использовать XML-представление, только в режиме **Design**, когда вы можете подвигать все компоненты в визуальном редакторе.

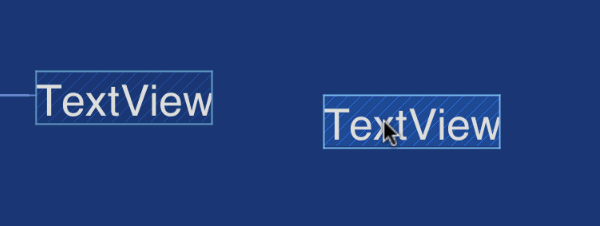
Рассмотрим отдельные элементы, которые используются для редактирования макета. Переключитесь в режим **Blueprint** http://developer.alexanderklimov.ru/android/layout/blueprint.png, чтобы ничего нас не отвлекало от работы.

Так выглядит выбранный компонент.



Квадратные опорные точки в углах компонента позволяют изменять его размеры. Круглые опорные точки по краям позволяют управлять отступами от краёв экрана и других компонентов.

Продолговатый закругленный прямоугольник указывает на базовую линию текста. Пригодится при выравнивании по базовой линии другого компонента.



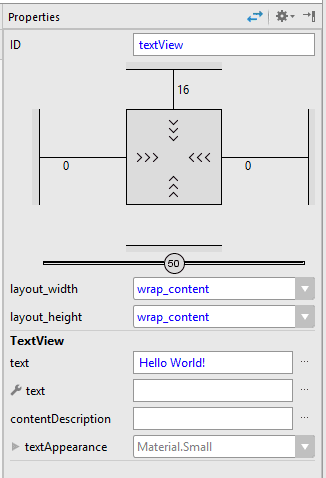
Также можно увидеть зигзагообразные направляющие-пружинки, которые нужны для особых случаев. Если провести мышкой над опорной точкой пружинки, она красиво активируется. Последующий щелчок удаляет пружинку. Если компонент удерживался двумя пружинками сверху и снизу, то удаление одной из них притянет компонент к верхней или нижней части экрана. Лучше самостоятельно проверить у себя, так как словами трудно описать.

Пружинки также можно легко восстановить, если щёлкнуть по круглой опорной точке и потянуть её к краю экрана.

Наверху на панели инструментов можно активировать автоматический режим размещения компонентов **Autoconnect** в виде магнита http://developer.alexanderklimov.ru/android/layout/constraintlayout10.png. В этом случае вы просто бросаете на экран нужный компонент, а далее студия сама пытается найти нужные параметры для него. Всё это происходит с красивой анимацией. Естественно, вы можете всё переиграть и поставить в нужную позицию.

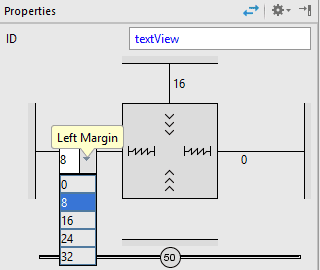
На той же панели есть инструмент **Infern Constraints** в виде пары звёздочек http://developer.alexanderklimov.ru/android/layout/constraintlayout11.png. Действует как **Autoconnect**, только работает не с одним компонентом, редактируемым в данный момент, а со всем макетом сразу, используя математические расчёты, чтобы определить, какие компоненты нужно привязать к другим, исходя из их местоположения на экране.

Теперь рассмотрим настройки в панели **Properties**.



Набор из трёх стрелок http://developer.alexanderklimov.ru/android/layout/constraintlayout7.png внутри квадрата означают атрибут **wrap\_content**. Если вы измените этот атрибут с помощью выпадающего списка или вручную напишите размер в **dp**, то увидите, что стрелки заменятся на прямые (фиксированный размер http://developer.alexanderklimov.ru/android/layout/constraintlayout8.png) или зигзаги (any size http://developer.alexanderklimov.ru/android/layout/constraintlayout9.png, который является приблизительным аналогом атрибута **match\_parent**).

По бокам квадрата имеются числа. Если подвести к ним мышку, то появится выпадающий список с определёнными значениями: 0, 8, 16, 24, 32. Они отвечают за атрибут **margin** (отступы).



По бокам можно увидеть также ползунки с числами в кружочке. В примере виден только горизонтальный ползунок, при других настройках увидите и вертикальный. С их помощью можно центрировать компонент относительно экрана и других объектов.

GuideLine

На панели инструментов также имеется значок **GuideLines** с двумя опциями: **Add Vertical GuideLine** и **Add Horizontal GuideLine**. Если ими воспользоваться, то в XML-файле появятся такие строчки:

<android.support.constraint.Guideline

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/guideline"

android:orientation="vertical"

app:layout\_constraintGuide\_begin="72dp"

tools:layout\_editor\_absoluteY="0dp"

tools:layout\_editor\_absoluteX="72dp" />

По сути, это **View**, размер которого 0, что соответствует **View.GONE**. На этапе разработки мы видим только полоски, а во время работы приложения ничего не видим. Данные элементы помогают разместить компоненты аккуратно относительно линии.

Не всегда с помощью визуального редактора можно добиться нужного результата, тогда нужно переключиться в XML-режим. Один из таких случаев описан в статье [Square Island: Constraint Layout: Icon Label Text](http://blog.sqisland.com/2016/10/constraint-layout-icon-label-text.html).

Если есть желание работать через XML, то следует запомнить очень много атрибутов, например, для выравнивания относительно друг друга:

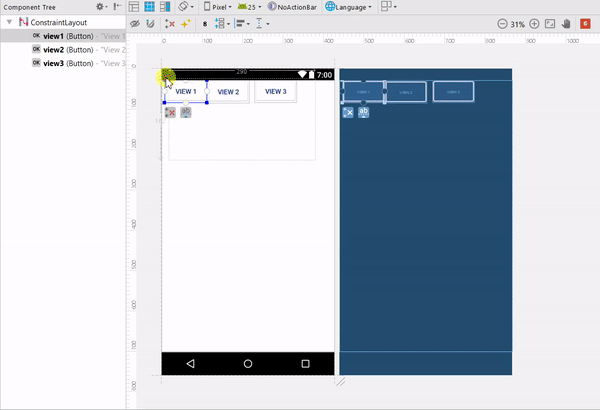
* app:layout\_constraintStart\_toStartOf="@id/view"
* app:layout\_constraintLeft\_toLeftOf="@id/view"
* app:layout\_constraintEnd\_toEndOf="@id/view"
* app:layout\_constraintRight\_toRightOf="@id/view"
* app:layout\_constraintTop\_toTopOf="@id/view"
* app:layout\_constraintBaseline\_toBaselineOf="@id/view"
* app:layout\_constraintBottom\_toBottomOf="@id/view"
* app:layout\_constraintStart\_toEndOf="@id/view"
* app:layout\_constraintLeft\_toRightOf="@id/view"
* app:layout\_constraintEnd\_toStartOf="@id/view"
* app:layout\_constraintRight\_toLeftOf="@id/view"
* app:layout\_constraintTop\_toBottomOf="@id/view"
* app:layout\_constraintBottom\_toTopOf="@id/view"

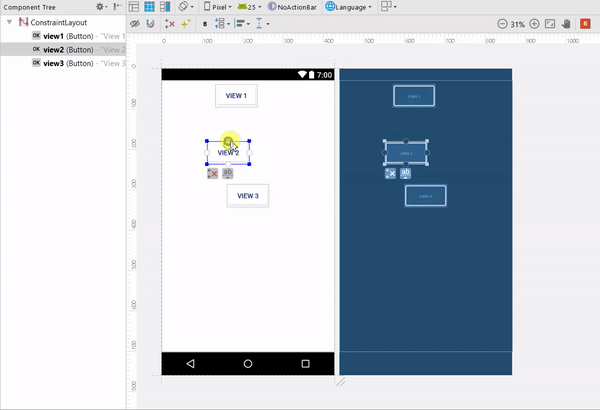
Атрибут **app:layout\_constraintHorizontal\_bias** используется float-значения от 0 до 1, для выравнивания по оси.

Атрибут **app:layout\_constraintDimensionRatio="4:3"** указывает, что нужно использовать данные пропорции по высоте и ширине для данного компонента. Также встречается модификация атрибута **app:layout\_constraintDimensionRatio="H, 1:1"**.

Chains - Скованные одной цепью

Несколько элементов можно сковать одной цепью. Допустим, у нас есть три кнопки. Выделяем их и через контекстное меню выбираем **Center Horizontally**. Снизу у выбранных компонентов появится символ цепи, а между ними будут нарисована связь в виде цепей. Если последовательно щёлкать по значку цепи, то увидите, как кнопки будут центрироваться с разными стилями - равномерно между собой и краями экрана, сбившись в одну кучку в центре или крайние прижмутся к краям, а центральная кнопка будет по центру. Аналогично это работает и для выравнивания по вертикали.





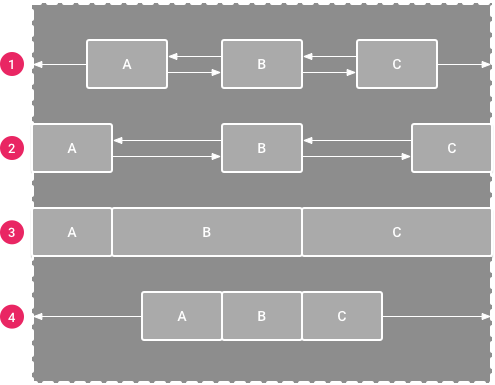
За цепи отвечают стили.

* **app:layout\_constraintHorizontal\_chainStyle="spread"**
* **app:layout\_constraintVertical\_chainStyle="spread"**
* **app:layout\_constraintHorizontal\_chainStyle="spread\_inside"**
* **app:layout\_constraintVertical\_chainStyle="spread\_inside"**
* **app:layout\_constraintHorizontal\_chainStyle="packed"**
* **app:layout\_constraintVertical\_chainStyle="packed"**

Также можно присвоить кнопкам вес, когда одна кнопка может быть больше остальных, это поведение знакомо по **LinearLayout**. Для этих целей используются атрибуты

* **layout\_constraintHorizontal\_weight**
* **layout\_constraintVertical\_weight**

На рисунке этот вариант представлен в третьем примере.



Дополнительное чтение

[Build a Responsive UI with ConstraintLayout | Android Developers](https://developer.android.com/training/constraint-layout/index.html)

[Watch ConstraintLayout: It Can Do What Now?](https://realm.io/news/constraintlayout-it-can-do-what-now/)