**ConstraintLayout**

**ConstraintLayout** представляет новый тип контейнеров, который позволяет создавать гибкие и масштабируемые интерфейсы.

Начнем с самых азов. Чтобы вы могли размещать на экране различные компоненты (кнопки, поля ввода, чекбоксы и т.п.), необходимо использовать специальный контейнер. Именно в него вы будете помещать компоненты. В Android компоненты называются View, а контейнер - ViewGroup.

Существуют несколько типов ViewGroup: LinearLayout, RelativeLayout, FrameLayout, TableLayout, ConstraintLayout и т.д.

Вообще, слово Constraint переводится как ограничение, принуждение. Но как по мне, так это не совсем точно отражает смысл. Самое подходящее слово, которое я могу тут подобрать - это привязка. Его и буду использовать.

Создаем пустой проект.

Обратите внимание, что TextView как бы вложен в ConstraintLayout. Это то, о чем я говорил в начале. ConstraintLayout - контейнер, а внутри него находятся различные View, в нашем случае - TextView. Также, можно сказать, что ConstraintLayout является родителем или родительским ViewGroup для TextView.

Давайте удалим TextView с экрана. Для этого просто выделите его на экране или в Component Tree и нажмите кнопку Del на клавиатуре.

Теперь ConstraintLayout пуст и экран ничего не отображает.

Давайте добавим на экран какой-нибудь компонент, например, снова TextView. Для этого просто перетащите компонент мышкой из Palette на экран.

После этого TextView появился на экране и в Component Tree.

Запустим приложение и посмотрим, как этот текст будет выглядеть.

Видим, что TextView уехал влево и вверх. Что-то явно пошло не так.

Если вы откроете текстовое представление вашего экрана (вкладка Text слева-снизу), то увидите, что элемент TextView подчеркнут красной линией.

Если навести на него мышкой, то он покажет ошибку:  
This view is not constrained, it only has designtime positions, so it will jump to (0,0) unless you add constraints.

Этим сообщением студия говорит нам, что View не привязано. Его текущее положение на экране актуально только для разработки (т.е. только в студии). А при работе приложения, это положение будет проигнорировано, и View уедет в точку (0,0), т.е. влево-вверх (что мы и наблюдали при запуске).

Как сделать так, чтобы View в ConstraintLayout оставалось на месте и не смещалось в угол? Необходимо добавить привязки (constraints). Они будут задавать положение View на экране относительно каких-либо других элементов или относительно родительского View.

Давайте добавим привязки для нашего TextView.

Если вы выделите на экране TextView, то можете видеть 4 круга по его бокам.

Эти круги используются, чтобы создавать привязки.

Существует два типа привязок: одни задают положение View по горизонтали, а другие - по вертикали.

Создадим горизонтальную привязку. Привяжем положение TextView к левому краю его родителя. Напомню, что родителем TextView является ConstraintLayout, который в нашем случае занимает весь экран. Поэтому края ConstraintLayout совпадают с краями экрана.

Чтобы создать привязку, нажмите мышкой на TextView, чтобы выделить его. Затем зажмите левой кнопкой мыши левый кружок и тащите его к левой границе.

TextView также уехал влево. Он привязался к левой границе своего родителя.

Но вовсе необязательно они должны быть вплотную. Мы можем задать отступ. Для этого просто зажмите левой кнопкой мыши TextView, перетащите вправо и отпустите.

Давайте создадим вертикальную привязку, чтобы закрепить TextView и по вертикали.

Теперь TextView привязан и по горизонтали, и по вертикали. Т.е. он точно знает, где он должен находиться на экране во время работы приложения.

Запускаем, чтобы проверить

Давайте добавим еще одно View, например, кнопку - Button.

Если сейчас запустить приложение, то кнопка уедет влево-вверх потому что она ни к чему не привязана.

Мы можем привязывать не только к границам родителя, но и к другим View. Давайте привяжем кнопку к TextView.

Т.к. кнопка привязана к TextView, то, если мы сейчас будем перемещать TextView, то кнопка будет также перемещаться.

Как удалить привязку

Чтобы удалить привязку, надо просто нажать на соответствующий кружок. Удалим привязки у кнопки.

Добавим три кнопки.

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. Контекстное меню -> “Cycle Chain Mode”, то увидите, как кнопки будут центрироваться с разными стилями:

* **spread** - Свободное пространство равномерно распределяется между выбранными компонентами и краями родителя (например, экрана)
* **spread\_inside** - Крайние компоненты прижимаются к границам родителя, свободное пространство равномерно распределяется только между остальными компонентами
* **packed** - Свободное пространство равномерно распределяется между крайними компонентами и границами родителя. Вы можете использовать **margin** для отступов