# Ресурсы и макеты экрана приложения, ConstraintLayout в Android Studio

Как правило, каждый экран в вашем приложении для Android связан с одним классом Java или Kotlin, известным как **Activity**.  Этот термин можно перевести как активность или деятельность, но мы будем использовать термин активити,  без перевода. Единственный экран с отображенным «Hello World» создается с помощью активити  MainActivity.kt. Это активити было создано  средой разработки, когда вы создали свой новый проект. Каждое видимое активити в приложении для Android имеет макет, который определяет пользовательский интерфейс для активити. Android Studio имеет редактор макетов, в котором вы можете создавать и определять макеты.

Макеты описаны в XML. Редактор макета позволяет вам определять и изменять макет либо путем кодирования XML, либо с помощью интерактивного визуального редактора. Вы можете переключаться между редактированием макета в XML и в редакторе визуального дизайна и создавать макеты любым способом.

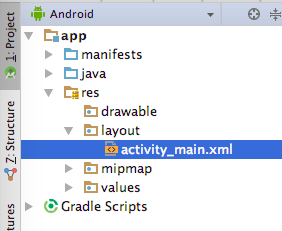
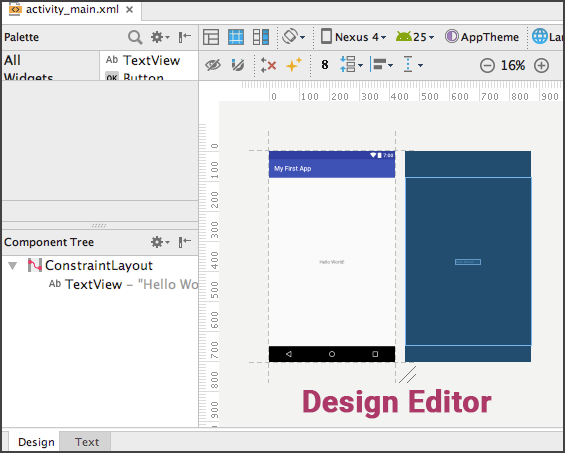
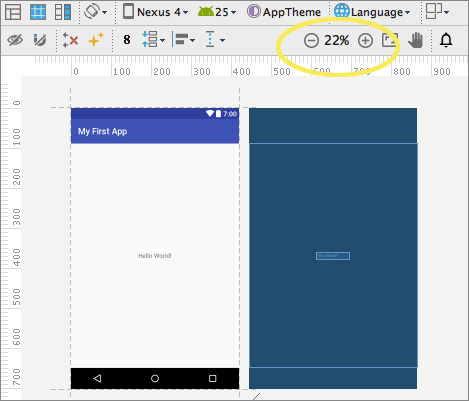
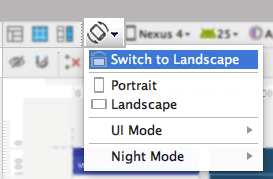
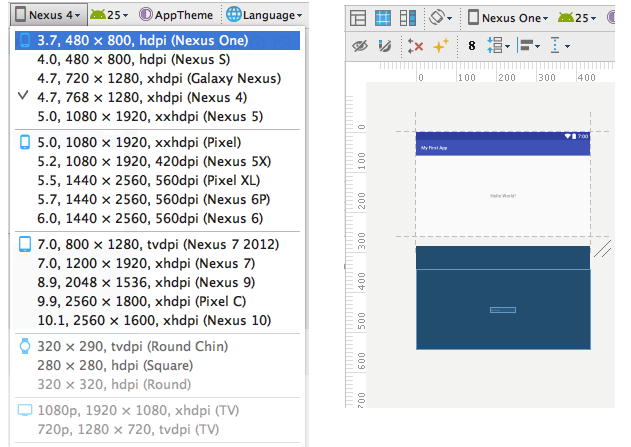
**1 . Редактор макета в Android Studio**

Каждый элемент в макете представляет собой view, или представление. В этой задаче изучим интерфейс редактора макетов в Android Studio и разберемся, как изменять значения свойств для view.

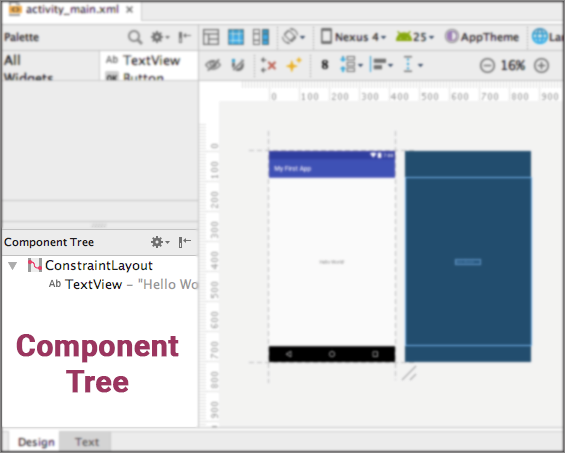
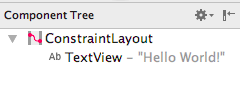
**Что вы узнаете**

* Как использовать редактор макетов.
* Как установить значения свойств.
* Как добавить строковые ресурсы.
* Как добавить цветовые ресурсы.

**Откройте редактор макета**

1. Найдите и откройте папку, в которой хранятся все макеты **res/layout**.  
   
2. Дважды щелкните файл разметки макета экрана **activity\_main.xml**.Панель справа, называемая *редактором макетов* , отображает содержимое этого файла.
3. В нижней части панели обратите внимание на две вкладки: « **Дизайн** и **текст»** . Откройте вкладку «Текст».XML-код, определяющий макет, отображается на вкладке «Текст».
4. Откройте вкладку «Дизайн», чтобы отобразить визуальный редактор макета.
5. В правом верхнем углу редактора макета вы видите кнопки **+** и **—** для увеличения и уменьшения масштаба. Используйте эти кнопки для настройки вида или нажмите кнопку **Zoom to Fit Screen** (справа от кнопок масштабирования), чтобы обе панели соответствовали вашему экрану.На этом скриншоте *макет дизайна* слева показывает, как ваше приложение появляется на устройстве. Схема *чертежа* , показанная справа, представляет собой схематический вид макета.
6. Используйте значки компоновки  в левом верхнем углу панели инструментов дизайна, чтобы отобразить проектный вид, вид чертежа или оба вида вместе.
7. Используйте значок ориентации, чтобы изменить ориентацию макета.
8. Используйте меню выбора устройства, чтобы увидеть, как будет выглядеть макет на разных андроид-устройствах.

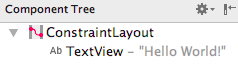
**Изучите и измените размер дерева компонентов**

1. Посмотрите на дерево компонентов в левом нижнем углу вкладки «Дизайн».  
   На этой панели отображается иерархия представлений view в вашем макете.
2. Отрегулируйте стороны панелей так, чтобы вы могли видеть все в Дереве компонентов.  
   
3. Щелкните значок «Закрыть панель» в правом верхнем углу дерева компонентов.Панель Component Tree закрывается.
4. Верните панель Component Tree, щелкнув по вертикальной метке «Дерево компонентов» слева.Теперь, когда вы знаете, как изменять размер, закрывать и открывать панель Component Tree, пришло время изучить иерархии представлений view.

**Исследуйте иерархии представлений view**

1. В дереве компонентов обратите внимание, что корень иерархии представлений view представляет собой компонент разметки ConstraintLayout.Каждый макет должен иметь корневой view, содержащий все остальные view. Корневой view всегда представляет собой группу представлений ViewGroup, которая представляет собой view, которое содержит другие view. ConstraintLayout — один из примеров группы представлений ViewGroup. Его название содержит слово constraint – ограничения, и это отражает суть взаимодействия элементов, добавляемых в этот корневой view.

Далее в этом уроке мы рассмотрим работу с ConstraintLayout более подробно.

1. Обратите внимание, что ConstraintLayout содержит TextView.  
   
2. Просмотрите XML-код для макета, щелкнув вкладку « **Текст** » в нижней части вкладки «Дизайн».
3. В коде XML обратите внимание, что корневой элемент <android.support.constraint.ConstraintLayout>. Корневой элемент содержит один <TextView> элемент.



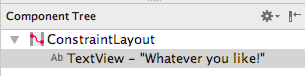
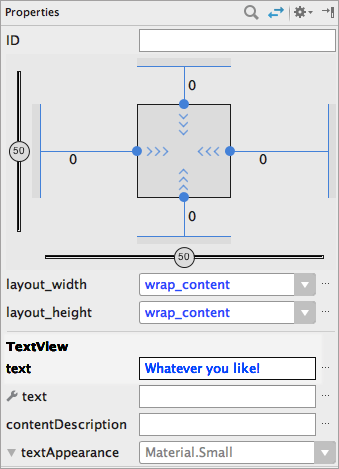
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <android.support.constraint.ConstraintLayout    ... >       <TextView        ...  />    </android.support.constraint.ConstraintLayout> |

**Изменение значений свойств**

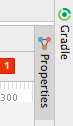
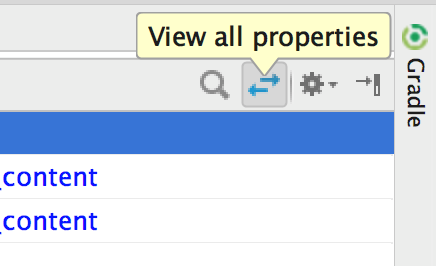
1. На вкладке «Текст» изучите свойства элемента TextView.Пока что не беспокойтесь о каких-либо инструментах и свойствах макета – мы рассмотрим их позже.

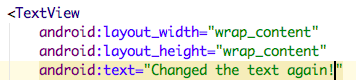


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <TextView       android:layout\_width="wrap\_content"       android:layout\_height="wrap\_content"       android:text="Hello World!"       ... /> |

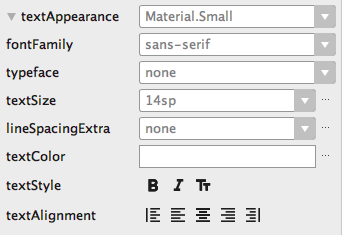
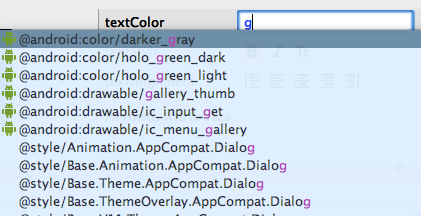
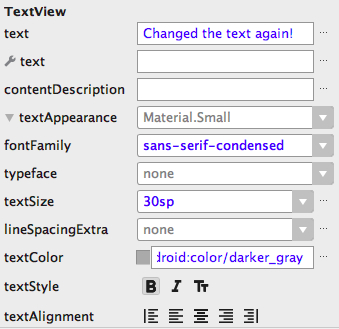
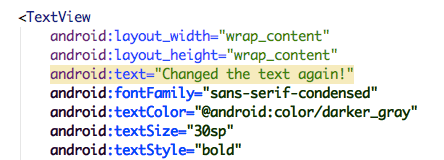
1. Измените значение свойства **text** в кавычках на любое.
2. Вернитесь на вкладку «Дизайн».
3. Выберите **TextView** в **дереве компонентов** .  
   
4. Посмотрите на панель «Свойства» справа.  
   

**Устранение неполадок этого шага :**

* Если панель «Свойства» не отображается, щелкните вертикальную метку « **Свойства»** в правом верхнем углу.
* Если панель свойств отличается от отображаемой, щелкните значок « **Просмотреть все свойства»,**чтобы переключить представление.

1. В текстовом поле TextView на панели «Свойства» снова измените значение текста и нажмите « **Ввод»,**чтобы завершить выборhttps://www.fandroid.info/wp-content/uploads/2017/12/tekstovom-pole-TextView.png
2. Вернитесь на вкладку «Текст», чтобы убедиться, что текст был изменен в XML.
3. Запустите приложение, чтобы увидеть изменение.

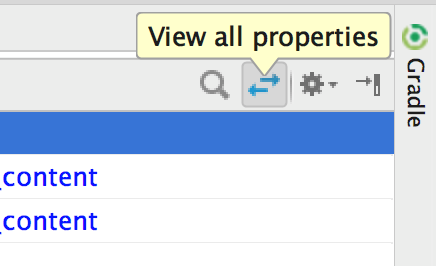
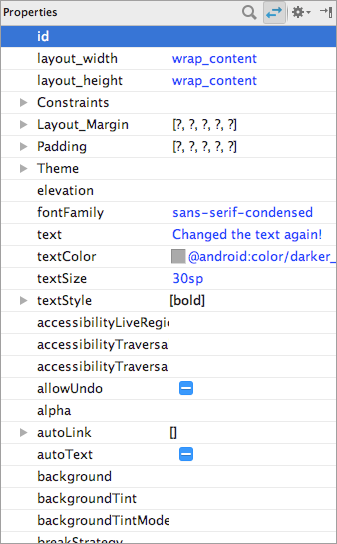
**Изменение свойств отображения текста**

1. На вкладке «Дизайн» разверните поле **textAppearance** для TextView.
2. Измените некоторые свойства текста. Например, измените семейство шрифтов, увеличьте размер текста и выберите жирный стиль. (Возможно, вам потребуется прокрутить панель, чтобы увидеть все поля.)
3. Измените цвет текста. Щелкните в поле **textColor** и введите букву g.Появится меню с возможными значениями завершения. Этот список содержит предварительно определенные цвета. Позже разберемся, как быстро определить свои собственные цвета.
4. Выберите **@android:color/darker\_gray** и нажмите **Enter** .
5. На вкладке «Текст» просмотрите XML для TextView. Вы видите, что новые свойства были добавлены.

Если вы видите ошибку «Hardcoded string should use @string resource,» не беспокойтесь, вы скоро узнаете, как исправить это.

1. Запустите приложение еще раз.

**Показать все свойства**

1. Вернитесь на вкладку «Дизайн». В правом верхнем углу панели «Свойства» щелкните значок « **Просмотреть все свойства»** . Если вы не видите никаких свойств, убедитесь, что TextView все еще выбран в дереве компонентов.Панель «Свойства» обновляется, чтобы показать все свойства TextView.
2. Прокрутите список, чтобы получить представление о свойствах, которые вы можете установить для TextView.

**2. Добавление строковых и цветовых ресурсов**

Разберемся, как создавать ресурсы. Использование ресурсов позволяет задавать значения однажды и затем использовать одни и те же значения в нескольких местах. Таким образом, созданный ресурс вы можете использовать в любом месте приложения, а при изменении ресурса в месте хранения изменения будут отображаться везде, где он используется.

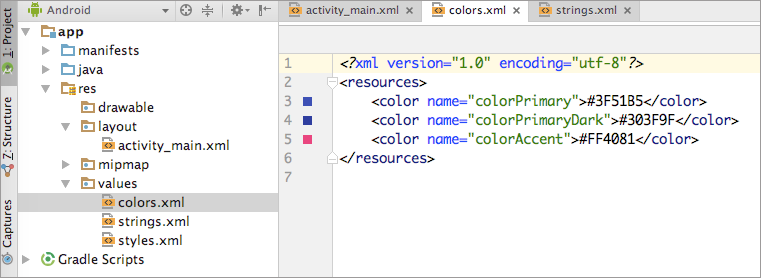
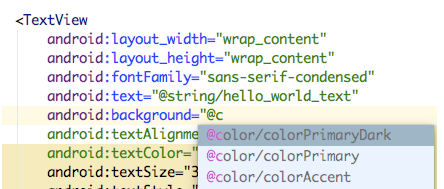
**Что вы узнаете**

* Как извлечь значения свойств в ресурсы.
* Как определяются ресурсы.
* Добавление и использование цветовых ресурсов.
* Результаты изменения высоты и ширины макета.

**Добавить цветовые ресурсы**

Теперь разберемся, как добавлять новые цветовые ресурсы.

**Изменение цвета текста и фона TextView**

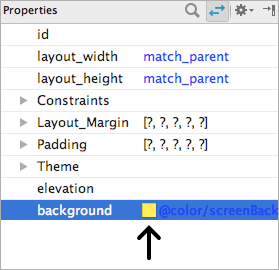
1. На панели «Проект» слева в папке **res**/**values**  откройте файл цветовых ресурсов **colors.xml.**Файл colors.xml открывается в редакторе. В нем определено три цвета. Это основные цветовые ресурсы для оформления приложения в стиле материального дизайна. Более подробно о принципах материального дизайна смотрите по ссылке. [/urok-1-kak-obnovlyat-vashi-prilozheniya-do-osobennostej-material-design/](https://www.fandroid.info/urok-1-kak-obnovlyat-vashi-prilozheniya-do-osobennostej-material-design/)
2. Вернитесь к Activity\_main.xml на вкладке «Текст», чтобы вы могли увидеть XML-код макета.
3. Добавьте новое свойство в TextView под названием **android: background** и начните печатать значение **@color** .Появится меню, предлагающее предопределенные цветовые ресурсы:
4. Выберите **цвет @ colorPrimaryDark.**
5. Измените свойство **android: textColor** и дайте ему значение **@android:color/white** .Android Studio определяет диапазон цветов, включая белый, поэтому вам не нужно самостоятельно определять белый цвет.
6. Вернитесь на вкладку «Дизайн» в Activity\_main.xml. Скройте панель «Свойства», если она загромождает ваше представление view.Вы видите, что TextView теперь имеет темно-синий фон, а текст отображается белым цветом.

**Добавить новый цвет для использования в качестве цвета фона экрана**

1. Вернитесь в colors.xml, добавьте новый цветовой ресурс:

< **color name = «screenBackground»** > #FFEE58 </**color** >

Этот цвет желтый.

1. Вернитесь на вкладку «Дизайн» в Activity\_main.xml. В дереве компонентов выберите ConstraintLayout.
2. На панели «Свойства» отобразите все свойства. Выберите свойство **Background** и нажмите **Enter** . Введите «c» в появившемся поле.
3. В появившемся меню выберите **цвет @color/screenBackground** . Нажмите « **Enter** **»,** чтобы завершить выбор.
4. Дважды щелкните желтый значок слева от значения цвета в поле **Background**.Откроется страница «Цвета» в редакторе ресурсов, в которой отображается список цветов, определенных в вашем проекте, а также цвета, определенные платформой Android. Он также предлагает интерактивный выбор цвета.
5. Не стесняйтесь изменять значение цвета **screenBackground** , но убедитесь, что окончательный цвет заметно отличается от colorPrimary и colorPrimaryDark.

**Исследуйте свойства ширины и высоты**

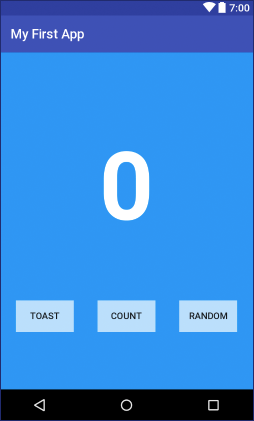
Теперь, когда у вас есть новый цвет фона экрана, вы будете использовать его для изучения эффектов изменения свойств ширины и высоты view.

1. Перейдите на вкладку «Дизайн» для Activity\_main.xml. В дереве компонентов выберите ConstraintLayout.
2. На панели «Свойства» отображается минимизированный вид свойств. Переключитесь на значок « **Просмотреть все свойства»,** если нужно. Значения **Layout\_width** и **layout\_height**  установлены в **match\_parent** . ConstraintLayout является корневым представлением, поэтому размер «родительского» макета фактически является размером экрана на устройстве. ConstraintLayout будет иметь ту же ширину и высоту, что и экран.

**Совет.** Все представления должны иметь свойства **layout\_width** и **layout\_height** .

1. На вкладке «Дизайн» обратите внимание, что весь фон экрана использует цвет **экрана** .
2. Теперь выберите TextView и измените ширину и высоту на **match\_parent** .TextView заполняет весь экран, потому что его ширина и высота такие же, как и ConstraintLayout, который, в свою очередь, заполняет весь экран.
3. Посмотрите, что произойдет, если ширина **match\_parent,** а высота — **wrap\_content** и наоборот.
4. Установите как ширину, так и высоту TextView обратно в **wrap\_content** .

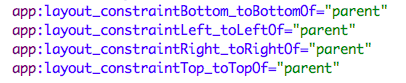
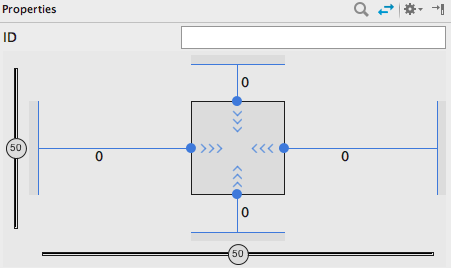
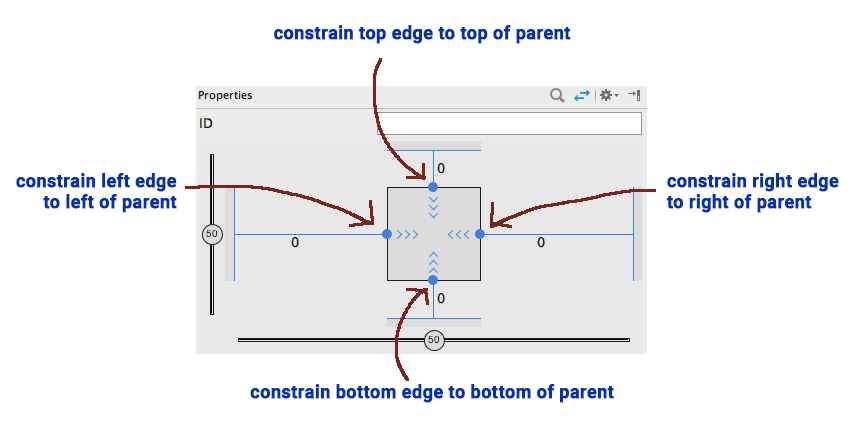
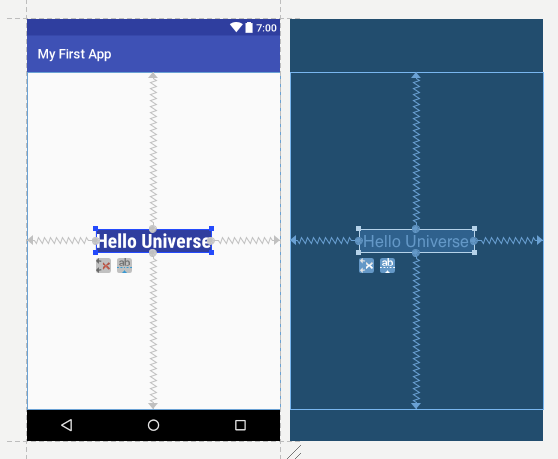
**3. Добавим views и  constraints (ограничения)**

В этой задаче вы добавите три кнопки в свой пользовательский интерфейс, как показано здесь.

**Что вы узнаете**

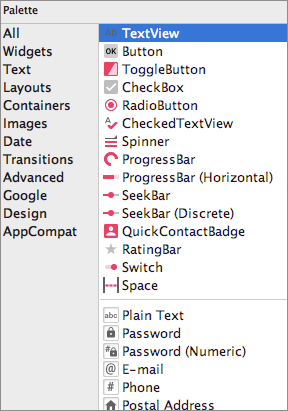
* Как добавить новые view  в макет.
* Как ограничить позицию представления другим view.

**Просмотр свойств ограничений**

1. На вкладке «Текст» просмотрите свойства ограничения для TextView.Эти свойства определяют положение TextView. Прочтите их внимательно.Вы можете ограничить верхнюю, нижнюю, левую и правую стороны сверху, снизу, слева и справа от других view. В этом случае единственное другое view — это корневой  ConstraintLayout, который является родительским элементом этого TextView.
2. На вкладке «Дизайн» выберите TextView в дереве компонентов и посмотрите на Inspector Constraint в верхней части панели «Свойства».Каждая из синих точек представляет собой constrain — ограничение.
3. Обратите внимание, что в представлении *чертежа*  также показаны ограничения при выборе определенного view. Переместите курсор над представлением проекта, и он также отобразит ограничения.

**Добавьте кнопки и ограничьте их позиции**

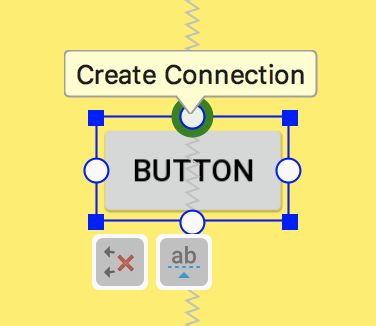
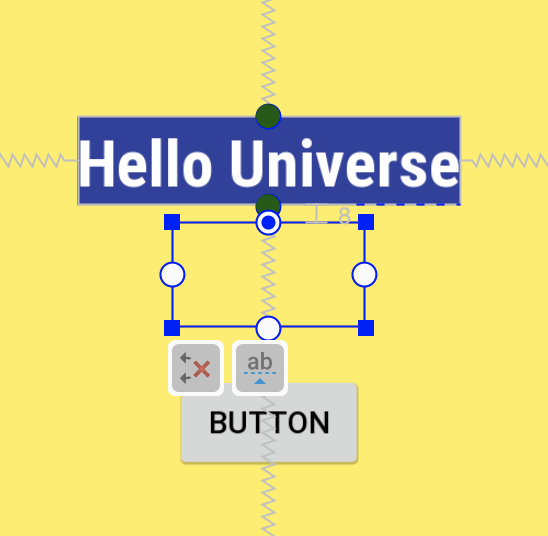
Чтобы узнать, как использовать ограничения для подключения view друг к другу, добавьте три кнопки в макет.

1. Обратите внимание на палитру в левом верхнем углу редактора компоновки. Переместите стороны так, чтобы вы могли видеть многие элементы в палитре.
2. Прокрутите перечисленные элементы, чтобы получить представление о том, что доступно.
3. Выберите **кнопку** , которая находится рядом с верхней частью, и перетащите ее на макет проекта, поместив ее под TextView.

Обратите внимание, что кнопка добавлена ​​в дерево компонентов под ConstraintLayout.

**Добавьте ограничение на кнопку**

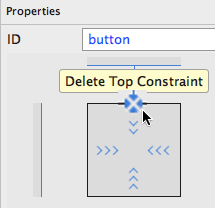
Теперь вы ограничиваете верхнюю часть кнопки до нижней части TextView.

1. Переместите курсор по кругу в середине верхней части кнопки.
2. Перетяните круг в верхней части кнопки на круг в нижней части TextView.Кнопка перемещается вверх, чтобы разместиться чуть ниже TextView. Верхняя часть кнопки теперь *ограничена*нижней частью TextView.
3. Взгляните на инспектора ограничений на панели «Свойства». Он показывает одно ограничение для кнопки.
4. На вкладке «Текст» ознакомьтесь с кодом XML для кнопки. Он включает атрибут, который ограничивает верхнюю часть кнопки до нижней части TextView, а также имеет верхний отступ, который был добавлен автоматически.ограничивает верхнюю часть кнопки до нижней части TextVi

**Добавьте вторую кнопку**

1. Добавьте еще одну кнопку в макет и поместите ее куда угодно. (Не беспокойтесь, если она улетит куда-нибудь в макете)
2. Играйте с позициями и ограничениями двух кнопок. Попробуйте добавить и удалить ограничения.

Чтобы удалить ограничение:

* В представлении «Дизайн» переместите курсор по кругу для ограничения, пока круг не станет красным, а затем щелкните по нему.
* Или на панели «Свойства» переместите курсор по кругу для ограничения, пока он не покажет x, а затем щелкните по нему.

Если вы удалили ограничение и хотите его вернуть, отмените действие или создайте новое ограничение.

1. Когда разберетесь, как добавлять и удалять ограничения, добавьте ограничения, чтобы разместить кнопки в их конечной позиции. Левая кнопка должна находиться слева, а правая кнопка — справа на экране. Вертикально обе кнопки отображаются между TextView и нижней частью экрана (см. Рисунок ниже).Сделайте это, добавив ограничения на две кнопки, чтобы соблюдались условия:

* верхняя часть каждой кнопки ограничена нижней стороной TextView.
* нижняя часть каждой кнопки ограничена нижней стороной экрана.
* левая часть левой кнопки ограничена левой стороной экрана.
* правая часть правой кнопки ограничена правой стороной экрана.

**Добавьте третью кнопку**

Добавьте третью кнопку между двумя существующими кнопками.

1. Добавьте третью кнопку.
2. Сдвиньте левую и правую стороны новой кнопки на соседние кнопки.
3. Ограничьте верхнюю и нижнюю части новой кнопки так же, как существующие кнопки. Другими словами, верхняя часть новой кнопки ограничена нижней стороной TextView. Низ ограничен нижней стороной экрана.
4. Изучите XML-код на вкладке «Текст». У каждой из кнопок есть атрибут**:** **app:layout\_constraintVertical\_bias**  
   Ограничения «смещения» позволяют сместить положение view на большее значение, чем на другой стороне, когда обе стороны [ограничены в противоположных направлениях](https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html#CenteringPositioning) . Например, если верхняя и нижняя стороны view ограничены по верху и низу экрана, вы можете использовать вертикальное смещение, чтобы поместить view ближе к верхней, чем к нижней части экрана.
5. Код XML для новой кнопки, скорее всего, появится ниже кода двух других кнопок в файле XML. Переместите код для новой кнопки, чтобы он находился между двумя другими кнопками, таким образом порядок кнопок в файле XML отражает порядок, который они отображают в дизайне. Это изменение не влияет на функциональность или внешний вид макета, но это поможет вам сопоставить представления на вкладке «Текст» с представлениями на вкладке «Дизайн».

Вот код XML для макета. В вашем макете могут быть немного другие поля и, возможно, некоторые ограничения по вертикали или горизонтали. Точные значения атрибутов для внешнего вида TextView также могут отличаться у вас.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65 | <?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:layout\_width="match\_parent"     android:layout\_height="match\_parent"     android:background="@color/screenBackground"     tools:context="com.example.android.myfirstapp.MainActivity">       <TextView         android:id="@+id/textView"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"          android:background="@color/colorPrimaryDark"         android:fontFamily="sans-serif"         android:text="@string/hello\_world\_text"         android:textAlignment="center"         android:textColor="@android:color/white"         android:textSize="30sp"         android:textStyle="bold"         app:layout\_constraintBottom\_toBottomOf="parent"         app:layout\_constraintLeft\_toLeftOf="parent"         app:layout\_constraintRight\_toRightOf="parent"         app:layout\_constraintTop\_toTopOf="parent"      />       <Button         android:id="@+id/button"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Button"         android:layout\_marginLeft="8dp"         app:layout\_constraintLeft\_toLeftOf="parent"         app:layout\_constraintBottom\_toBottomOf="parent"         android:layout\_marginBottom="8dp"         android:layout\_marginTop="8dp"         app:layout\_constraintTop\_toBottomOf="@+id/textView" />       <Button         android:id="@+id/button3"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Button"         app:layout\_constraintBottom\_toBottomOf="parent"         android:layout\_marginBottom="8dp"         android:layout\_marginTop="8dp"         app:layout\_constraintTop\_toBottomOf="@+id/textView"         app:layout\_constraintRight\_toLeftOf="@+id/button2"         android:layout\_marginRight="8dp"         app:layout\_constraintLeft\_toRightOf="@+id/button"         android:layout\_marginLeft="8dp" />       <Button         android:id="@+id/button2"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Button"         android:layout\_marginRight="8dp"         app:layout\_constraintRight\_toRightOf="parent"         app:layout\_constraintBottom\_toBottomOf="parent"         android:layout\_marginBottom="8dp"         android:layout\_marginTop="8dp"         app:layout\_constraintTop\_toBottomOf="@+id/textView" />    </android.support.constraint.ConstraintLayout> |

**Подготовьте пользовательский интерфейс для следующей задачи**

Следующая задача — заставить кнопки выполнить какое-то действие при нажатии. Прежде чем научиться настраивать интерактивные кнопки, завершите подготовку пользовательского интерфейса.

1. Измените текст TextView, чтобы показать **0** (ноль).
2. Измените текст и id кнопки. Вы можете отредактировать XML-код на вкладке «Текст» или задать значения свойств на панели «Свойства» в редакторе проекта. Установите текст и, id как показано ниже:

|  |  |  |
| --- | --- | --- |
| **Кнопка** | **Текст** | **Идентификатор** |
| Левая кнопка | Toast | @+id/toast\_button |
| Средняя кнопка | Count | @+id/count\_button |
| Правая кнопка | Random | @+id/random\_button |

**При необходимости исправить ошибки**

Если вы отредактировали XML для макета, код XML для средней кнопки, которая теперь является кнопкой **Count** , показывает некоторые ошибки. Также макет испорчен на вкладке «Дизайн».  
Ошибки возникают из-за того, что кнопки слева и справа изменили их id.

1. Если у вас есть эти ошибки, исправьте их, обновив id кнопки в ограничениях, подчеркнутых красным.



|  |  |
| --- | --- |
| 1  2 | app:layout\_constraintRight\_toLeftOf="@+id/toast\_button"  app:layout\_constraintLeft\_toRightOf="@+id/random\_button" |

**Обновите внешний вид кнопок и TextView**

Макет вашего приложения теперь в основном завершен, но его внешний вид можно улучшить несколькими небольшими изменениями.

**Добавить новые цветовые ресурсы**

1. В файле colors.xml измените значение screenBackground на #2196F3, что является синим оттенком в [палитре Material Design](https://material.io/guidelines/style/color.html#color-color-palette) .
2. Добавьте новый цвет с именем «buttonBackground». Используйте значение #BBDEFB, которое является более светлым оттенком в синей палитре.



|  |  |
| --- | --- |
| 1 | <color name="buttonBackground">#BBDEFB</color> |

**Добавить цвет фона для кнопок**

1. В макете добавьте цвет фона для каждой из кнопок. (Вы можете отредактировать XML или использовать панель «Свойства», в зависимости от того, что вы предпочитаете.)



|  |  |
| --- | --- |
| 1 | android:background="@color/buttonBackground" |

**Изменение полей левой и правой кнопок**

1. Дайте левой кнопке левый край 24dp и дайте правой кнопке правый край 24dp.



|  |  |
| --- | --- |
| 1 | android:layout\_marginLeft="24dp"android:layout\_marginRight="24dp" |

**Обновите внешний вид TextView**

1. Удалите фоновый цвет TextView, либо очистив значение на панели «Свойства», либо удалив атрибут android:background из кода XML.После этого, фон представления станет прозрачным.
2. Увеличьте размер текста TextView до 72sp.



|  |  |
| --- | --- |
| 1 | android:textSize="72sp" |

1. Измените семейство шрифтов TextView на sans-serif .
2. Добавьте свойство app:layout\_constraintVertical\_bias в TextView, чтобы он был более равномерно распределен по вертикали на экране. Не стесняйтесь настраивать значение этого ограничения по своему усмотрению. (Зайдите на вкладку «Дизайн», чтобы посмотреть, как выглядит макет).



|  |  |
| --- | --- |
| 1 | app:layout\_constraintVertical\_bias="0.3" |

**Единицы измерения dp и sp – это специальные относительные единицы, которые используются для размеров экранных компонентов и текста в Android Studio. Благодаря их использованию экранные элементы выглядят примерно одинаково на экранах разных размеров.**

**Совет** . Использование атрибута смещения вместо полей или отступов приводит к более удобному расположению на разных размерах экрана и ориентациях.

* Если представление ограничено другими видами как на верхнем, так и на нижнем краях, используйте вертикальное смещение, чтобы настроить его вертикальное положение.
* Если представление ограничено как на его левом, так и на правом краях, используйте горизонтальное смещение, чтобы настроить его горизонтальное положение.

**Запустите приложение**

Если вы внесете все обновления, ваше приложение будет выглядеть следующим образом. Если вы использовали разные цвета и шрифты, ваше приложение будет выглядеть несколько иначе.

Вы добавили кнопки на главный экран вашего приложения, но в настоящее время кнопки не реагируют на нажатия. Мы исправим это в следующем уроке.