## **Основы разработки в Android Studio**

## **Представления**

Пользовательский интерфейс состоит из иерархии объектов, называемых представлениями — каждый элемент экрана представляет собой View. Класс View представляет собой базовый строительный блок для всех компонентов пользовательского интерфейса и базовый класс для классов, которые предоставляют интерактивные компоненты пользовательского интерфейса, такие как кнопки, флажки и поля ввода текста.

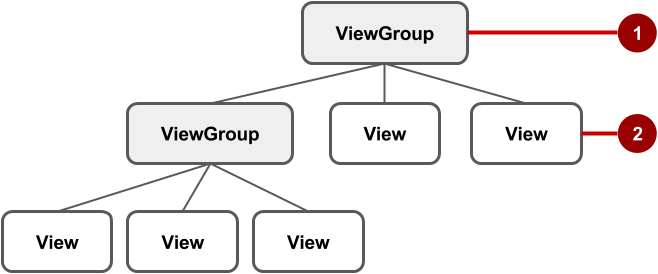
View имеет свойство местоположения, выраженное в виде пары координат слева и сверху, и два измерения, выраженных в ширине и высоте. Единицей измерения местоположения и размеров является пиксель, не зависящий от плотности (dp).

Система Android предоставляет сотни предопределенных View подклассов. Часто используемые подклассы View:

* TextView для отображения текста
* EditText чтобы пользователь мог вводить и редактировать текст
* Button и другие интерактивные элементы (такие как RadioButton, CheckBox и Spinner) для обеспечения интерактивного поведения
* ScrollView и RecyclerView для отображения прокручиваемых элементов
* ImageView для отображения изображений
* Layout для размещения других представлений и их позиционирования

**Группы ViewGroup**

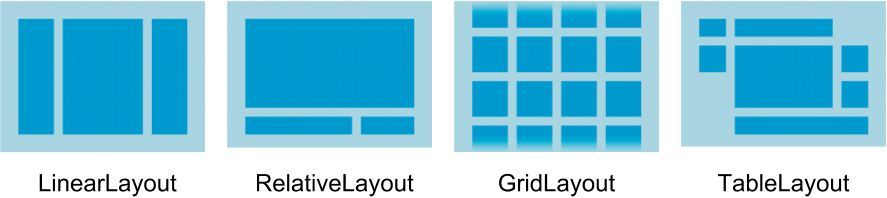
View элементы могут быть сгруппированы внутри ViewGroup, который действует как контейнер. Отношения родитель-потомок, в которых родительским является ViewGroup, а дочерним является View или другой ViewGroup.

View-элементы экрана организованы в иерархию. В корне этой иерархии находится a ViewGroup, который содержит макет всего экрана. ViewGroup может содержать дочерние View элементы или другие ViewGroup, как показано на следующем рисунке. 

На рисунке выше:

1. Root ViewGroup.
2. Первый набор дочерних View элементов и ViewGroup групп, родительским элементом которых является root.

Большинство ViewGroup группы называют контейнерами или макетами, потому что они организуют дочерние View элементы определенным образом и обычно используются в качестве корневых ViewGroup. Вот несколько примеров:

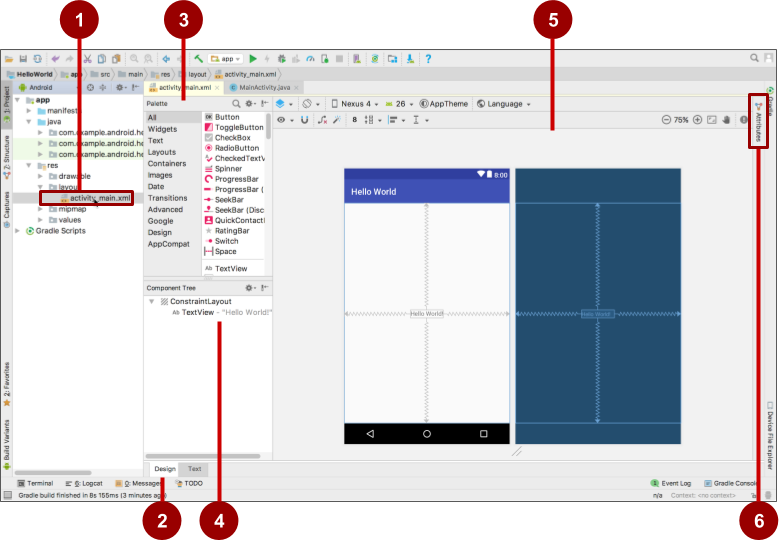
* ConstraintLayout: Группа дочерних View элементов, использующих ограничения, границы и направляющие для управления расположением элементов относительно других элементов в макете.
* LinearLayout: Группа дочерних View элементов, расположенных и выровненных по горизонтали или вертикали.
* RelativeLayout: Группа дочерних View элементов, в которой каждый элемент позиционируется и выравнивается относительно других элементов внутри ViewGroup. Другими словами, положения дочерних View элементов могут быть описаны по отношению друг к другу или к родительским ViewGroup.
* TableLayout: Группа дочерних View элементов, упорядоченных по строкам и столбцам.
* FrameLayout: Группа дочерних View элементов в стеке. FrameLayout предназначен для блокирования области на экране для отображения одного View. Дочерние View элементы отображаются в виде стека, сверху которого находится самый последний добавленный дочерний элемент. Размер FrameLayout - это размер самого большого дочернего View элемента.
* GridLayout: Группа, которая помещает свои дочерние View элементы в прямоугольную сетку, которую можно прокручивать. 

Иерархия макетов может стать сложной для приложения, которое отображает много View элементов на экране. Важно понимать иерархию, поскольку она влияет на то, видны ли View элементы и насколько эффективно они отрисовываются.

**Редактор**

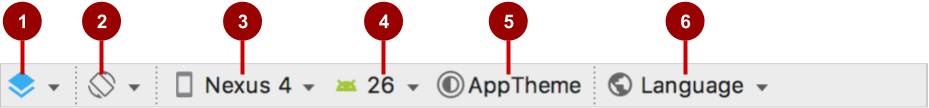
Контейнеры определяются в редакторе или путем ввода XML-кода.

Редактор отображает визуальное представление XML-кода. Вы можете перетаскивать элементы в область дизайна или схемы элементов и упорядочивать, изменять размер и задавать для них атрибуты. Вы сразу видите эффект от вносимых вами изменений.

Чтобы использовать редактор макетов, дважды щелкните файл XML-макета (activity\_main.xml). Появится редактор макетов с выделенной вкладкой Дизайн внизу. (Если выделена вкладка "Текст" и вы видите XML-код, перейдите на вкладку "Дизайн".) Для пустого шаблона Activity макет такой, как показано на рисунке ниже. 

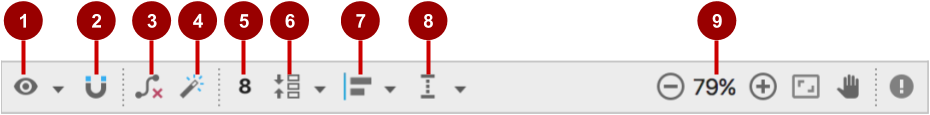
1. XML-файл макета (activity\_main.xml).
2. Дизайн и текстовые вкладки. Нажмите Дизайн, чтобы увидеть редактор, или Текст, чтобы увидеть XML-код.
3. Панель Палитра. Панель палитры предоставляет список элементов пользовательского интерфейса. Добавьте элементов пользовательский интерфейс, перетащив его на панель дизайна.
4. Дерево компонентов. На панели "Дерево компонентов" показана иерархия контейнеров. Щелкните View элемент или ViewGroup на этой панели, чтобы выбрать его. View элементы организованы в древовидную иерархию родительских и дочерних элементов, в которой дочерний элемент наследует атрибуты своего родительского элемента. На рисунке выше TextView является дочерним элементом ConstraintLayout.
5. Панели "Дизайн" и "Схема". Перетащите View элементы из панели "Палитра" на панель "Дизайн" или "Схема", чтобы разместить их в контейнере. На рисунке выше показан только один элемент: TextView который отображает "Hello World".
6. Вкладка Атрибуты. Щелкните по Атрибутам, чтобы отобразить панель Атрибутов для настройки атрибутов для View элемента.

### **Панели инструментов редактора**

Панели инструментов редактора содержат кнопки для настройки вашего макета и изменения его внешнего вида. Верхняя панель инструментов позволяет настроить внешний вид предварительного просмотра макета в редакторе: 

На рисунке выше показана верхняя панель инструментов редактора макетов:

1. Выберите Design Surface: Выберите Дизайн, чтобы отобразить предварительный просмотр элементов пользовательского интерфейса в цвете в вашем макете, или Схему элементов, чтобы отобразить только контуры элементов. Чтобы увидеть обе панели рядом, выберите Дизайн + Схема.
2. Ориентация в редакторе: Выберите книжную или альбомную ориентацию, чтобы отобразить предварительный просмотр в вертикальной или горизонтальной ориентации. Настройка ориентации позволяет просматривать ориентации макета или контейнеров, не запуская приложение на эмуляторе или устройстве. Чтобы создать альтернативные макеты, выберите Создать альбомный вариант или другие варианты.
3. Устройство в редакторе: Выберите тип устройства (телефон / планшет, Android TV или Android Wear).
4. Версия API в редакторе: Выберите версию Android для отображения предварительного просмотра.
5. Тема в редакторе: Выберите тему (например, AppTheme) для применения к предварительному просмотру.
6. Локализация в редакторе: Выберите язык и локаль для предварительного просмотра. В этом списке отображаются только языки, доступные в строковых ресурсах (подробнее о том, как добавлять языки, смотрите в уроке по локализации). Вы также можете выбрать Предварительный просмотр справа налево, чтобы просмотреть макет так, как если бы был выбран язык RTL.

Редактор также предлагает вторую панель инструментов, которая позволяет настраивать внешний вид элементов пользовательского интерфейса и увеличивать и уменьшать масштаб предварительного просмотра: 

На рисунке выше показана ConstraintLayout панель инструментов редактирования:

1. Показать: выберите Показать ограничения и Показать поля, чтобы отобразить их в предварительном просмотре или прекратить их отображение.
2. Автоматическое подключение: включить или отключить автоматическое подключение. При включенном автоматическом подключении вы можете перетащить любой элемент (например, Button) в любую часть макета, чтобы создать ограничения для родительского макета.
3. Очистить все ограничения: Очистить все ограничения во всем макете.
4. Вывод ограничений: создание ограничений путем вывода.
5. Поля по умолчанию: установите поля по умолчанию.
6. Упаковать: упаковать или развернуть выбранные элементы.
7. Выровнять: выровнять выбранные элементы.
8. Рекомендации: Добавьте вертикальные или горизонтальные направляющие.
9. Элементы управления масштабированием: увеличение или уменьшение масштаба.

## **Прямое редактирование XML**

Быстрее и проще редактировать XML-код напрямую, особенно при копировании и вставке кода для похожих элементов.

Чтобы просмотреть и отредактировать XML-код, откройте файл XML layout. Появится редактор с выделенной вкладкой Дизайн сверху. Перейдите на вкладку Код, чтобы увидеть XML-код.

### **XML-атрибуты (свойства элементов)**

Представления имеют свойства , которые определяют, где представление отображается на экране, его размер, как оно соотносится с другими представлениями и как оно реагирует на ввод данных пользователем. При определении представлений в XML или на панели "Атрибуты" редактора макетов свойства называются атрибутами.

Например, в следующем TextView используются, в android:id, android:layout\_width, android:layout\_height, android:background. TextView, являются XML-атрибутами, которые автоматически преобразуются в свойства:

<TextView

android:id="@+id/show\_count"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@color/myBackgroundColor"

android:textStyle="bold"

android:text="@string/count\_initial\_value" />

Атрибуты обычно имеют такую форму:

android:имя\_атрибута="значение"

Например:

android:textStyle="bold"

Если значением является ресурс, например цвет, то @ символ указывает, что это за ресурс. Например:

android:background="@color/myBackgroundColor"

Для атрибута background задается цветовой ресурс, обозначенный как myBackgroundColor, который объявлен как #FFF043.

Некоторые атрибуты специфичны для определенных элементов. Например, TextView подкласс поддерживает атрибут textSize. Любые элементы, расширяющие TextView, наследуют эти атрибуты, специфичные для подкласса.

Некоторые атрибуты являются общими для всех View элементов, потому что они унаследованы от корневого [View](https://translated.turbopages.org/proxy_u/en-ru.ru.ec792d66-66cb12d3-d2ec7855-74722d776562/https/developer.android.com/reference/android/view/View.html) класса. Один из примеров - android:id.

**Определение элемента**

Чтобы однозначно идентифицировать View и ссылаться на него в своем коде, вы должны присвоить ему значение id. android:id атрибут позволяет указать уникальный id идентификатор ресурса для View.

Например:

android:id="@+id/button\_count"

@+id/button\_count Часть атрибута создает id вызываемый button\_count для Button (подкласс View). Вы используете символ плюс (+), чтобы указать, что вы создаете новый id.

**Ссылка на элемент**

Чтобы ссылаться на существующий идентификатор ресурса, опустите символ плюс (+). Например, чтобы ссылаться на a View по его id имени в другом атрибуте, таком как android:layout\_toLeftOf (описанном в следующем разделе) для управления положением a View, вы должны использовать:

android:layout\_toLeftOf="@id/show\_count"

В приведенном выше атрибуте @id/show\_count ссылается на View с идентификатором ресурса show\_count. Атрибут позиционирует элемент так, чтобы он находился "слева" от show\_count View.