Макеты

Макет определяет визуальную структуру пользовательского интерфейса, например, пользовательского интерфейса [операции](https://developer.android.com/guide/components/activities.html?hl=ru) или [виджета приложения](https://developer.android.com/guide/topics/appwidgets/index.html?hl=ru). Существует два способа объявить макет:

* **Объявление элементов пользовательского интерфейса в XML**. В Android имеется удобный справочник XML-элементов для классов View и их подклассов, например таких, которые используются для виджетов и макетов.
* **Создание экземпляров элементов во время выполнения**. Ваше приложение может программным образом создавать объекты View и ViewGroup (а также управлять их свойствами).

Платформа Android предоставляет вам гибкость при использовании любого из этих способов для объявления пользовательского интерфейса приложения и его управления. Например, вы можете объявить в XML макеты по умолчанию, включая элементы экрана, которые будут отображаться в макетах, и их свойства. Затем вы можете добавить в приложение код, который позволяет изменять состояние объектов на экране (включая объявленные в XML) во время выполнения.

Преимущество объявления пользовательского интерфейса в файле XML заключается в том, что таким образом вы можете более эффективно отделить представление своего приложения от кода, который управляет его поведением. Описания пользовательского интерфейса находятся за пределами кода вашего приложения. Это означает, что вы можете изменять или адаптировать интерфейс без необходимости вносить правки в исходный код и повторно компилировать его. Например, можно создать разные файлы XML макета для экранов разных размеров и разных ориентаций экрана, а также для различных языков. Кроме того, объявление макета в XML упрощает визуализацию структуры пользовательского интерфейса, благодаря чему отладка проблем также становится проще. В данной статье мы научим вас объявлять макет в XML. Если вы предпочитаете создавать экземпляры объектов View во время выполнения, обратитесь к справочной документации для классов [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html?hl=ru) и [View](https://developer.android.com/reference/android/view/View.html?hl=ru).

Как правило, справочник XML-элементов для объявления элементов пользовательского интерфейса точно следует структуре и правилам именования для классов и методов — названия элементов соответствуют названиям классов, а названия атрибутов соответствуют методам. Фактически, соответствие зачастую такое точное, что вы можете с легкостью догадаться, какой атрибут XML соответствует тому или иному методу класса, или какой класс соответствует заданному элементу XML. Однако следует отметить, что не все справочники являются идентичными. В некоторых случаях названия могут несколько отличаться. Например, у элемента EditText есть атрибут text, который соответствует методу EditText.setText().

**Совет.** Дополнительные сведения о различных типах макетов представлены в разделе [Часто используемые макеты](https://developer.android.com/guide/topics/ui/layout-objects.html?hl=ru). Также в инструкциях по построению [объектов представления Hello](https://developer.android.com/resources/tutorials/views/index.html?hl=ru)изложены сведения о создании различных макетов.

Создание XML

С помощью справочника XML-элементов, который имеется в Android, можно быстро и просто создавать макеты пользовательского интерфейса и содержащиеся в нем элементы, точно так же, как при создании веб-страниц в HTML — с помощью вложенных элементов.

В каждом файле макета должен быть всего один корневой элемент, в качестве которого должен выступать объект представления (View) или представления группы (ViewGroup). После определения корневого элемента можно приступать к добавлению дополнительных объектов макета или виджетов в качестве дочерних элементов для постепенного формирования иерархии представлений, которая определяет ваш макет. Ниже представлен пример макета XML, в котором используется вертикальный объект [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html?hl=ru), в котором размещены элементы [TextView](https://developer.android.com/reference/android/widget/TextView.html?hl=ru) и [Button](https://developer.android.com/reference/android/widget/Button.html?hl=ru).

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
              android:layout\_width="match\_parent"  
              android:layout\_height="match\_parent"  
              android:orientation="vertical" >  
    <TextView android:id="@+id/text"  
              android:layout\_width="wrap\_content"  
              android:layout\_height="wrap\_content"  
              android:text="Hello, I am a TextView" />  
    <Button android:id="@+id/button"  
            android:layout\_width="wrap\_content"  
            android:layout\_height="wrap\_content"  
            android:text="Hello, I am a Button" />  
</LinearLayout>

После объявления макета в файле XML сохраните файл с расширением .xml в каталог res/layout/своего проекта Android для последующей компиляции.

Дополнительные сведения о синтаксисе для файла XML макета представлены в документе [Ресурсы макета](https://developer.android.com/guide/topics/resources/layout-resource.html?hl=ru).

Загрузка ресурса XML

Во время компиляции приложения каждый файл XML макета компилируется в ресурс [View](https://developer.android.com/reference/android/view/View.html?hl=ru). Вам необходимо загрузить ресурс макета в коде приложения в ходе реализации метода обратного вызова[Activity.onCreate()](https://developer.android.com/reference/android/app/Activity.html?hl=ru#onCreate(android.os.Bundle)). Для этого вызовите метод [setContentView()](https://developer.android.com/reference/android/app/Activity.html?hl=ru" \l "setContentView(int)), передайте в него ссылку на ресурс макета в следующей форме: R.layout.*layout\_file\_name*. Например, если макет XML сохранен как файл main\_layout.xml, загрузить его для вашей операции необходимо следующим образом:

public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main\_layout);  
}

Метод обратного вызова onCreate() в вашей операции вызывается платформой Android во время запуска операции (см. сведения о жизненных циклах в документе [Операции](https://developer.android.com/guide/components/activities.html?hl=ru#Lifecycle)).

Атрибуты

Каждый объект View и ViewGroup поддерживают свои собственные атрибуты XML. Некоторые атрибуты характерны только для объекта View (например, объект TextView поддерживает атрибут textSize), однако эти атрибуты также наследуются любыми объектами View, которые могут наследовать этот класс. Некоторые атрибуты являются общими для всех объектов View, поскольку они наследуются от корневого класса View (такие как атрибут id). Любые другие атрибуты рассматриваются как «параметры макета». Такие атрибуты описывают определенные ориентации макета для объекта View, которые заданы родительским объектом ViewGroup такого объекта.

Идентификатор

У любого объекта View может быть связанный с ним целочисленный идентификатор, который служит для обозначения уникальности объекта View в иерархии. Во время компиляции приложения этот идентификатор используется как целое число, однако идентификатор обычно назначается в файле XML макета в виде строки в атрибуте id. Этот атрибут XML является общим для всех объектов View (определенных классом [View](https://developer.android.com/reference/android/view/View.html?hl=ru)), который вы будете использовать довольно часто. Синтаксис для идентификатора внутри тега XML следующий:

android:id="@+id/my\_button"

Символ @ в начале строки указывает на то, что обработчику XML следует выполнить синтаксический анализ остальной части идентификатора, выполнить ее синтаксический анализ и определить ее в качестве ресурса идентификатора. Символ плюса (+) обозначает, что это имя нового ресурса, который необходимо создать и добавить к нашим ресурсам (в файле R.java). В Android существует ряд других ресурсов идентификатора. При ссылке на идентификатор ресурса Android вам не нужно указывать символ плюса, однако необходимо добавить пространство имен пакета android, как указано ниже:

android:id="@android:id/empty"

После добавления пространства имен пакета android можно сослаться на идентификатор из класса ресурсовandroid.R, а не из локального класса ресурсов.

Чтобы создать представления и сослаться на них из приложения, обычно следует выполнить указанные ниже действия.

1. Определите представление или виджет в файле макета и присвойте ему уникальный идентификатор:

<Button android:id="@+id/my\_button"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="@string/my\_button\_text"/>

1. Затем создайте экземпляр объекта представления и выполните его захват из макета (обычно с помощью метода [onCreate()](https://developer.android.com/reference/android/app/Activity.html?hl=ru" \l "onCreate(android.os.Bundle))):

Button myButton = (Button) findViewById(R.id.my\_button);

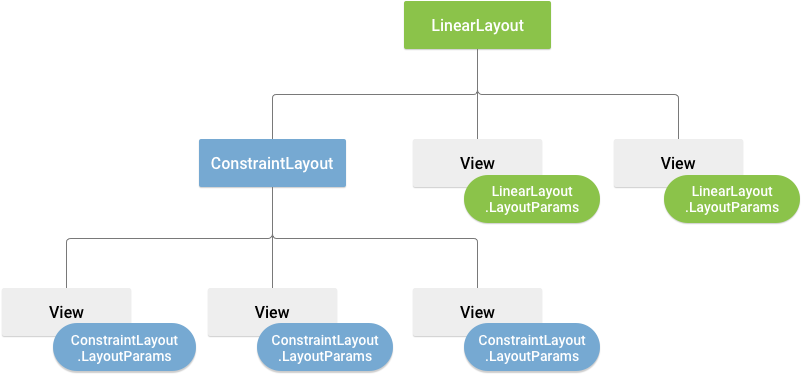
Определение идентификаторов для объектов представления имеет важное значение при создании объекта [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html?hl=ru). В относительном макете универсальные идентификаторы используются для расположения представлений относительно друг друга.

Идентификатор не обязательно должен быть уникальным в рамках всей иерархии, а только в той ее части, где вы выполняете поиск (зачастую это может быть как раз вся иерархия, поэтому при возможности идентификаторы должны быть полностью уникальными).

Параметры макета

Атрибуты макета XML, которые называются layout\_*something*, определяют параметры макета для объекта представления, подходящие для класса ViewGroup, в котором он находится.

Каждый класс ViewGroup реализует вложенный класс, который наследует [ViewGroup.LayoutParams](https://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html?hl=ru). В этом подклассе имеются типы свойств, которые определяют размер и положение каждого дочернего представления, подходящие для его группы. На рисунке 1 показано, что родительская группа представлений определяет параметры макета для каждого дочернего представления (включая дочернюю группу представлений).



**Рисунок 1.** Графическое представление иерархии представления с параметрами макета каждого представления.

Обратите внимание, что подкласс LayoutParams имеет собственный синтаксис для задания значений. Каждый дочерний элемент должен определять LayoutParams, которые подходят для его родительского элемента, тогда как он сам может определять другие LayoutParams для своих дочерних элементов.

Все группы представлений включают в себя параметры ширины и высоты (layout\_width иlayout\_height), и каждое представление должно определять их. Многие LayoutParams также включают дополнительные параметры полей и границ.

Для параметров ширины и высоты можно указать точные значения, хотя, возможно, вам не захочется делать это часто. Обычно для задания значений ширины и высоты используется одна из следующих констант:

* ***wrap\_content*** — размер представления задается по размерам его содержимого;
* ***match\_parent*** (которая до API уровня 8 ***называлась fill\_parent*** ) — размер представления определяется ограничениями, задаваемыми его родительской группой представлений.

Как правило, не рекомендуется задавать абсолютные значения ширины и высоты макета (например, в пикселах). Вместо этого используйте относительные единицы измерения, такие как пикселы, не зависящие от разрешения экрана (***dp***), ***wrap\_content***или ***match\_parent***Это гарантирует одинаковое отображение вашего приложения на устройствах с экранами разных размеров. Принятые типы измерения определены в документе [Доступные ресурсы](https://developer.android.com/guide/topics/resources/available-resources.html?hl=ru#dimension).

Размещение макета

Представление имеет прямоугольную форму. Расположение представления определяется его координатами *слева* и *сверху*, а его размеры — параметрами ширины и высоты. Расположение измеряется в пикселах.

Расположение представления можно получить путем вызова методов [getLeft()](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "getLeft()) и [getTop()](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "getTop()). Первый возвращает координату слева (по оси X) для прямоугольника представления. Второй возвращает верхнюю координату (по оси Y) для прямоугольника представления. Оба этих метода возвращают расположение представления относительно его родительского элемента. Например, когда метод getLeft() возвращает 20, это означает, что представление находится на расстоянии 20 пикселов от левого края его непосредственного родительского элемента.

Кроме того, имеется несколько удобных методов ([getRight()](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "getRight()) и [getBottom()](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "getBottom())), которые позволяют избежать лишних вычислений. Эти методы возвращают координаты правого и нижнего краев прямоугольника представления. Например, вызов метода [getRight()](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "getRight()) аналогичен следующему вычислению: getLeft() + getWidth().

Размер, отступ и поля

Размер представления выражается его шириной и высотой. Фактически, представление обладает двумя парами значений «ширина-высота».

Первая пара — это *измеренная ширина* и *измеренная высота*. Эти размеры определяют размер представления в границах своего родительского элемента. Измеренные размеры можно получить, вызвав методы [getMeasuredWidth()](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "getMeasuredWidth()) и[getMeasuredHeight()](https://developer.android.com/reference/android/view/View.html?hl=ru#getMeasuredHeight()).

Вторая пара значений — это просто *ширина* и *высота* (иногда они называются *чертежная ширина* и *чертежная высота*). Эти размеры определяют фактический размер представления на экране после разметки во время их отрисовки. Эти значения могут отличаться от измеренных ширины и высоты, хотя это и не обязательно. Значения ширины и высоты можно получить, вызвав методы [getWidth()](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "getWidth()) и [getHeight()](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "getHeight()).

При измерении своих размеров представление учитывает заполнение. Отступ выражается в пикселах для левой, верхней, правой и нижней частей представления. Отступ можно использовать для смещения содержимого представления на определенное количество пикселов. Например, значение отступа слева, равное 2, приведет к тому, что содержимое представления будет смещено на 2 пиксела вправо от левого края представления. Для задания отступов можно использовать метод [setPadding(int, int, int, int)](https://developer.android.com/reference/android/view/View.html?hl=ru" \l "setPadding(int,%20int,%20int,%20int)). Чтобы запросить отступ, используются методы [getPaddingLeft()](https://developer.android.com/reference/android/view/View.html?hl=ru#getPaddingLeft()), [getPaddingTop()](https://developer.android.com/reference/android/view/View.html?hl=ru#getPaddingTop()),[getPaddingRight()](https://developer.android.com/reference/android/view/View.html?hl=ru#getPaddingRight()) и [getPaddingBottom()](https://developer.android.com/reference/android/view/View.html?hl=ru#getPaddingBottom()).

Даже если представление может определить отступ, в нем отсутствует поддержка полей. Такая возможность имеется у группы представлений. Дополнительные сведения представлены в справке по объектам [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html?hl=ru) и [ViewGroup.MarginLayoutParams](https://developer.android.com/reference/android/view/ViewGroup.MarginLayoutParams.html?hl=ru).

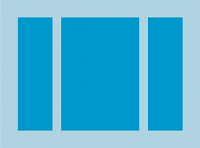
Дополнительные сведения о размерах представлены в разделе [Значения размеров](https://developer.android.com/guide/topics/resources/more-resources.html?hl=ru#Dimension).

Часто используемые макеты

В каждом подклассе класса [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html?hl=ru) имеется уникальный способ отображения вложенных в него представлений. Ниже представлены некоторые из наиболее часто используемых типов макетов, которые входят в состав платформы Android.

**Примечание.** Несмотря на то, что для формирования пользовательского интерфейса один макет может содержать один или несколько вложенных макетов, рекомендуется использовать как можно более простую иерархию макетов. Чем меньше в макете вложенных элементов, тем быстрее выполняется его отрисовка (горизонтальная иерархия представлений намного лучше вертикальной).

[Линейный макет](https://developer.android.com/guide/topics/ui/layout/linear.html?hl=ru)

[](https://developer.android.com/guide/topics/ui/layout/linear.html?hl=ru)

Макет, в котором дочерние элементы представлены в горизонтальных или вертикальных столбцах. Если длина окна больше длины экрана, в нем создается полоса прокрутки.

[Относительный макет](https://developer.android.com/guide/topics/ui/layout/relative.html?hl=ru)

[](https://developer.android.com/guide/topics/ui/layout/relative.html?hl=ru)

В этом макете можно задавать расположение дочерних объектов относительно друг друга (дочерний элемент А находится слева от дочернего элемента Б) или относительно родительского объекта (выравнивание относительно верхнего края родительского элемента).

[Представление веб-страницы](https://developer.android.com/guide/webapps/webview.html?hl=ru)

[](https://developer.android.com/guide/webapps/webview.html?hl=ru)

Отображение веб-страниц.

Создание макетов с помощью адаптера

Если содержимое макета является динамическим или не определено заранее, можно использовать макет, который создает подклассы класса [AdapterView](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru) для заполнения макета представлениями во время выполнения. Подкласс класса [AdapterView](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru) использует [Adapter](https://developer.android.com/reference/android/widget/Adapter.html?hl=ru) для привязки данных к своему макету. [Adapter](https://developer.android.com/reference/android/widget/Adapter.html?hl=ru) выступает в качестве посредника между источником данных и макетом [AdapterView](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru) — [Adapter](https://developer.android.com/reference/android/widget/Adapter.html?hl=ru) извлекает данные (из источника, например, массива или запроса к базе данных) и преобразует каждую запись в представление, которое можно добавить в макет [AdapterView](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru).

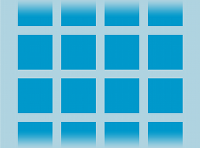
Часто используемые макеты, для создания которых можно использовать адаптер, включают в себя следующие:

[Представление в виде списка](https://developer.android.com/guide/topics/ui/layout/listview.html?hl=ru)

[](https://developer.android.com/guide/topics/ui/layout/listview.html?hl=ru)

Отображение списка в один столбец с возможностью прокрутки.

[Представление в виде сетки](https://developer.android.com/guide/topics/ui/layout/gridview.html?hl=ru)

[](https://developer.android.com/guide/topics/ui/layout/gridview.html?hl=ru)

Отображение сетки из столбцов и строк с возможностью прокрутки.

Заполнение представления адаптера данными

Чтобы заполнить адаптер [AdapterView](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru), такой как [ListView](https://developer.android.com/reference/android/widget/ListView.html?hl=ru) или [GridView](https://developer.android.com/reference/android/widget/GridView.html?hl=ru), свяжите экземпляр [AdapterView](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru) с приложением [Adapter](https://developer.android.com/reference/android/widget/Adapter.html?hl=ru), которое извлекает данные из внешнего источника и создает объект [View](https://developer.android.com/reference/android/view/View.html?hl=ru), который представляет каждую запись данных.

В Android предусмотрено несколько подклассов адаптера [Adapter](https://developer.android.com/reference/android/widget/Adapter.html?hl=ru), которые полезно использовать для извлечения данных различных видов и создания представлений для [AdapterView](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru). Вот два наиболее часто используемых адаптера:

[ArrayAdapter](https://developer.android.com/reference/android/widget/ArrayAdapter.html?hl=ru)

Этот адаптер используется в случае, когда в качестве источника данных выступает массив. По умолчанию [ArrayAdapter](https://developer.android.com/reference/android/widget/ArrayAdapter.html?hl=ru) создает представление для каждого элемента массива путем вызова метода [toString()](https://developer.android.com/reference/java/lang/Object.html?hl=ru" \l "toString()) для каждого элемента и помещения его содержимого в объект [TextView](https://developer.android.com/reference/android/widget/TextView.html?hl=ru).

Например, если имеется массив строк, которые необходимо отобразить в объекте [ListView](https://developer.android.com/reference/android/widget/ListView.html?hl=ru), запустите новый [ArrayAdapter](https://developer.android.com/reference/android/widget/ArrayAdapter.html?hl=ru) с помощью конструктора, чтобы указать макет для каждой строки и массива строк:

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
        android.R.layout.simple\_list\_item\_1, myStringArray);

Аргументы для такого конструктора следующие:

* ваше приложение [Context](https://developer.android.com/reference/android/content/Context.html?hl=ru);
* макет, в котором содержится объект [TextView](https://developer.android.com/reference/android/widget/TextView.html?hl=ru) для каждой строки в массиве;
* массив строк.

Затем достаточно вызвать метод [setAdapter()](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru" \l "setAdapter(T)) для своего объекта [ListView](https://developer.android.com/reference/android/widget/ListView.html?hl=ru):

ListView listView = (ListView) findViewById(R.id.listview);  
listView.setAdapter(adapter);

Чтобы настроить внешний вид каждого элемента, можно переопределить метод [toString()](https://developer.android.com/reference/java/lang/Object.html?hl=ru" \l "toString()) для объектов в массиве. Либо можно создать представление для каждого элемента, который отличается от[TextView](https://developer.android.com/reference/android/widget/TextView.html?hl=ru) (например, если для каждого элемента массива требуется объект[ImageView](https://developer.android.com/reference/android/widget/ImageView.html?hl=ru)), наследовать класс [ArrayAdapter](https://developer.android.com/reference/android/widget/ArrayAdapter.html?hl=ru) и переопределить метод [getView()](https://developer.android.com/reference/android/widget/ArrayAdapter.html?hl=ru" \l "getView(int,%20android.view.View,%20android.view.ViewGroup)), чтобы возвратить требуемый тип представления для каждого элемента.

[SimpleCursorAdapter](https://developer.android.com/reference/android/widget/SimpleCursorAdapter.html?hl=ru)

Этот адаптер используется в случае, когда в качестве источника данных выступает объект [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru). Если используется [SimpleCursorAdapter](https://developer.android.com/reference/android/widget/SimpleCursorAdapter.html?hl=ru), необходимо указать макет, который будет использоваться для каждой строки в объекте [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru), а также в какие представления макета необходимо вставить столбцы в объекте [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru) . Например, если необходимо создать список имен людей с их номерами телефонов, можно выполнить запрос, который возвращает объект [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru), содержащий строку для каждого человека и столбцы для имен и номеров телефонов. Затем создается массив строк с указанием столбцов из объекта [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru), которые необходимо поместить в макет для каждого результата, а также массив целых чисел, в котором указаны соответствующие представления для вставки каждого столбца:

String[] fromColumns = {ContactsContract.Data.DISPLAY\_NAME,  
                        ContactsContract.CommonDataKinds.Phone.NUMBER};  
int[] toViews = {R.id.display\_name, R.id.phone\_number};

При создании экземпляра объекта [SimpleCursorAdapter](https://developer.android.com/reference/android/widget/SimpleCursorAdapter.html?hl=ru) передайте в него макет, который будет использоваться для каждого результата, объект [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru) с результатами и два следующих массива:

SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,  
        R.layout.person\_name\_and\_number, cursor, fromColumns, toViews, 0);  
ListView listView = getListView();  
listView.setAdapter(adapter);

Затем [SimpleCursorAdapter](https://developer.android.com/reference/android/widget/SimpleCursorAdapter.html?hl=ru) создает представление для каждой строки в объекте [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru) с помощью предоставленного макета путем вставки каждого элемента fromColumns в соответствующее представление toViews.

.

Если в течение жизненного цикла вашего приложения вы вносите изменения в соответствующие данные, чтение которых выполняет адаптер, вам следует вызвать метод [notifyDataSetChanged()](https://developer.android.com/reference/android/widget/ArrayAdapter.html?hl=ru" \l "notifyDataSetChanged()). Это позволит уведомить связанное представление о том, что данные были изменены и ему требуется выполнить обновление.

Обработка нажатий

Чтобы организовать реагирование на нажатие каждого элемента в [AdapterView](https://developer.android.com/reference/android/widget/AdapterView.html?hl=ru), реализуйте интерфейс [AdapterView.OnItemClickListener](https://developer.android.com/reference/android/widget/AdapterView.OnItemClickListener.html?hl=ru). Например:

// Create a message handling object as an anonymous class.  
private OnItemClickListener mMessageClickedHandler = new OnItemClickListener() {  
    public void onItemClick(AdapterView parent, View v, int position, long id) {  
        // Do something in response to the click  
    }  
};  
  
listView.setOnItemClickListener(mMessageClickedHandler);

Единицы измерения

Исторически так сложилось, что разработчики всегда использовали пиксели при создании компьютерных интерфейсов. Но с развитием технологий данный подход стал источником проблем - на многих новых дисплеях элементы стали очень мелкими при установленных высоких разрешениях. Поэтому стали появляться новые единицы измерения, не зависящие от разрешения экрана.

Android поддерживает несколько стандартных единиц измерения. Вкратце перечислим их.

* px (pixels) — пиксели. Точки на экране - минимальные единицы измерения;
* dp (density-independent pixels) — независимые от плотности пиксели. Абстрактная единица измерения, основанная на физической плотности экрана с разрешением 160 dpi. В этом случае 1dp = 1px;
* dip - синоним для dp. Иногда используется в примерах Google;
* sp (scale-independent pixels) — независимые от масштабирования пиксели. Допускают настройку размеров, производимую пользователем. Полезны при работе с шрифтами;
* in (inches) — дюймы, базируются на физических размерах экрана. Можно измерить обычной линейкой;
* mm (millimeters) — миллиметры, базируются на физических размерах экрана. Можно измерить обычной линейкой;
* pt (points) — 1/72 дюйма, базируются на физических размерах экрана;

Как правило, при установке размера текста используются единицы измерения **sp**, которые наиболее корректно отображают шрифты:

android:textSize="48sp"

В остальных случаях рекомендуется использовать **dp**.

Переводим dp в пиксели

Так как на разных устройствах **dp** может различаться, то для получения величин в пикселях и наоборот используйте методы:

private float dpFromPx(float px) {

return px

/ getApplicationContext().getResources().getDisplayMetrics().density;

}

private float pxFromDp(float dp) {

return dp

\* getApplicationContext().getResources().getDisplayMetrics().density;

}

Если вы не определили размеры в XML, то их можно задать программно с помощью следующего кода (устанавливаем отступы для компонента):

final float scale = getResources().getDisplayMetrics().density;

int padding\_5dp = (int) (5 \* scale + 0.5f);

int padding\_20dp = (int) (20 \* scale + 0.5f);

int padding\_50dp = (int) (50 \* scale + 0.5f);

RadioButton rb = new RadioButton(this);

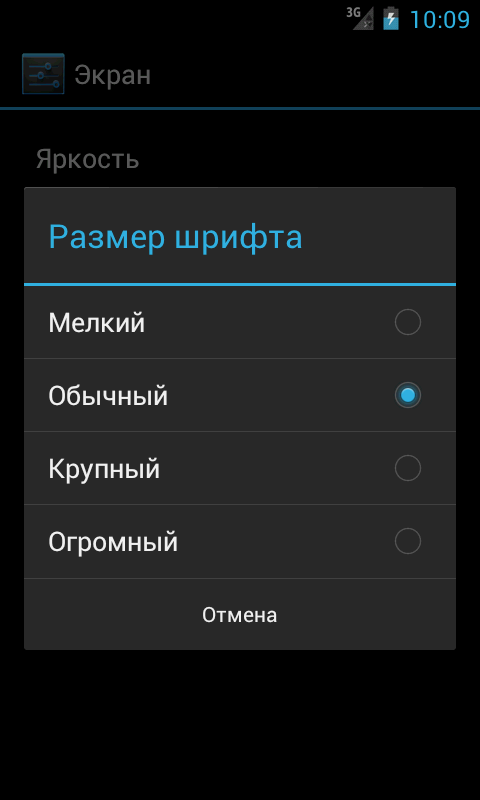
rb.setText("My Radio Button");

rb.setLayoutParams(new LayoutParams(padding\_50dp,padding\_50dp));

rb.setPadding(padding\_20dp,padding\_5dp,padding\_5dp,padding\_5dp);

Настройка шрифтов

Давайте чуть подробнее поговорим о работе со шрифтами, чтобы лучше понять специфику работы с текстами. Все люди разные - у кого-то зрение хорошое, у кого-то плохое. Android позволяет в настройках задать размеры шрифта в четырёх вариантах: Мелкий, Обычный, Крупный, Огромный. Для этого нужно зайти в Настройки | Экран | Размер шрифта.



Можно узнать программно выбранный вариант через свойство **fontScale**:

float fontScale = getResources().getConfiguration().fontScale;

mInfoTextView.setText("fontScale:: " + fontScale);

Обычному шрифту соответствует значение 1, мелкому - 0.9, крупному - 1.1, огромному - 1.15.

Если вы хотите, чтобы ваш текст мог меняться в зависимости от выбора пользователя, то используйте единицы измерения SP:

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Текст с размером 26sp"

android:textSize="26sp" />

В тех случаях, когда изменять текст по желанию пользователя не следует, но при этом текст должен отображаться корректно в зависимости от разрешения экрана устройства, то используйте единицы измерения DP.

Третий вариант - если вы ни при каких обстоятельствах (какой же вы упрямый) не хотите зависеть от предпочтений пользователя и разрешения экрана, то пользуйтесь PX (пиксели). Среда разработки будет сопротивляться вашему желанию и выводить предупреждающие сообщения. Подумайте ещё раз о своём решении.

Используем стандартные системные размеры шрифтов

В Android зашиты три системный размера шрифтов, основанных на SP: Small, Medium и Large. Вы можете использовать их в стандартных случаях, когда вам не нужно задавать конкретные значения (атрибут style):

<TextView

android:id="@+id/textView1"

style="@android:style/TextAppearance.Small"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста - Small" />

<TextView

android:id="@+id/textView2"

style="@android:style/TextAppearance.Medium"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста - Medium" />

<TextView

android:id="@+id/textView3"

style="@android:style/TextAppearance.Large"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста - Large" />

На самом деле стилей **@android:style/TextAppearance.\*** гораздо больше. Если вы вдруг забыли про названия стилей, то можете использовать встроенные возможности среды разработки. На панели инструментов виджет TextView представлен в четырёх вариантах: TextView, Large, Medium, Small, и в них используется атрибут **android:textAppearance**.

Создадим проект со всеми возможными вариантами и посмотрим на результат. В первом случае будем использовать стандартные настройки шрифта, во втором - увеличим его.

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical" >

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. Размер 20px"

android:textSize="20px" />

<TextView

android:id="@+id/textView2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. Размер не указан" />

<TextView

android:id="@+id/textView3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. textAppearanceSmall"

android:textAppearance="?android:attr/textAppearanceSmall" />

<TextView

android:id="@+id/textView4"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. textAppearanceMedium"

android:textAppearance="?android:attr/textAppearanceMedium" />

<TextView

android:id="@+id/textView5"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. textAppearanceLarge"

android:textAppearance="?android:attr/textAppearanceLarge" />

<TextView

android:id="@+id/textView6"

style="@android:style/TextAppearance.Small"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. TextAppearance.Small" />

<TextView

android:id="@+id/textView7"

style="@android:style/TextAppearance.Medium"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. TextAppearance.Medium" />

<TextView

android:id="@+id/textView8"

style="@android:style/TextAppearance.Large"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. TextAppearance.Large" />

<TextView

android:id="@+id/textView9"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Пример текста. Размер 30sp"

android:textSize="30sp" />

</LinearLayout>

