Два самых важных элемента интерфейса Android приложения — это **Activity** и **View**.

**View** — это практически любой элемент интерфейса. Кнопка, поле для ввода текста, элемент, отображающий текст, и многое другое — это все разновидности View.

**Activity** — это совокупность View на одном экране. По сути, Activity — это экран приложения.

Также давайте познакомимся с понятием **ViewGroup**.

Если View — это просто элемент интерфейса, то ViewGroup — это View, который может содержать внутри другие View, то есть контейнер.

Layout

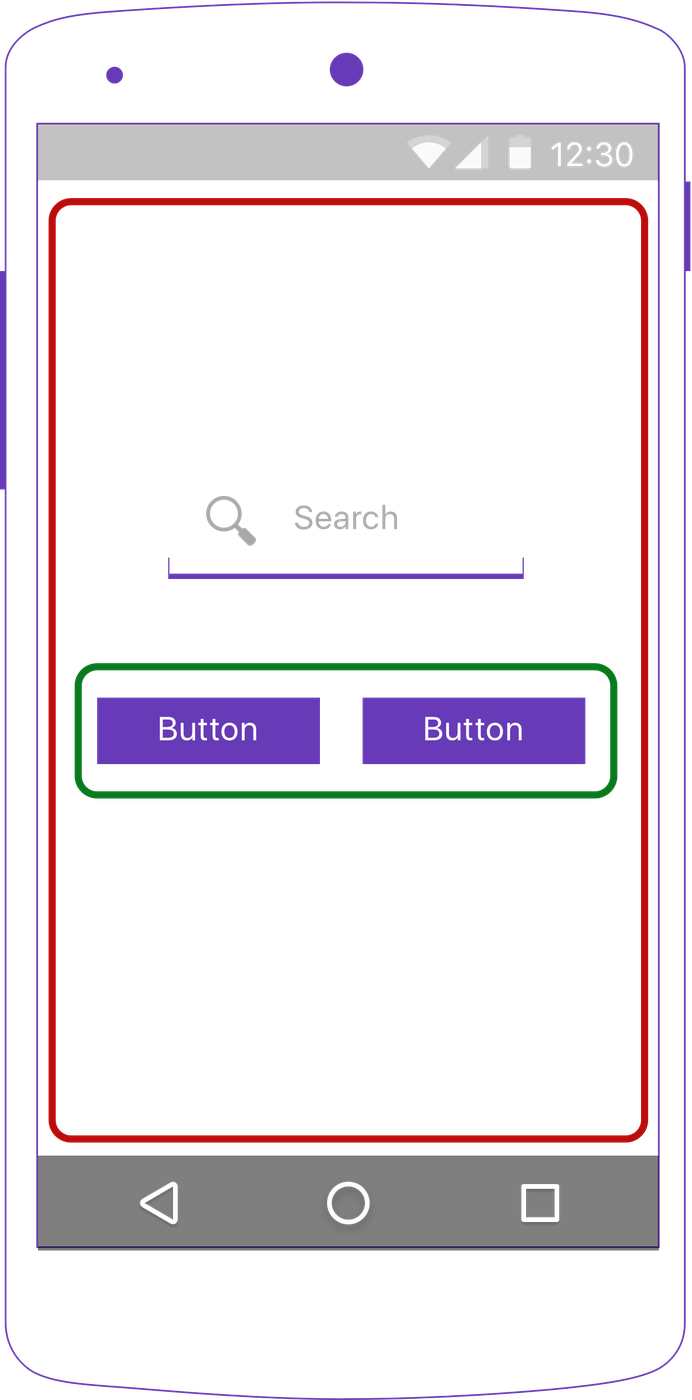
**Layout** — это категория ViewGroup, созданная для того, чтобы мы могли располагать элементы интерфейса так, как нам это нужно.

Самые популярные виды Layout:

* LinearLayout
* RelativeLayout
* FrameLayout

На самом деле их намного больше, но большинство задач можно решить, используя эти три вида.

Все еще непонятно, что такое **Layout**? Посмотрите на эту картинку:

[](https://skillberg.com/media/uploads/2017/11/17/09a0adaf3ce344c9a1563d3db5f39d47.png)Layout в Android

Кнопки и поле "Search" — это View. Зеленой рамочкой вокруг кнопок обозначены границы Layout (т.е. ViewGroup), внутри которого они находятся. А этот Layout и поле "Search", в свою очередь, находятся внутри другого Layout, границы которого обозначены красным цветом.

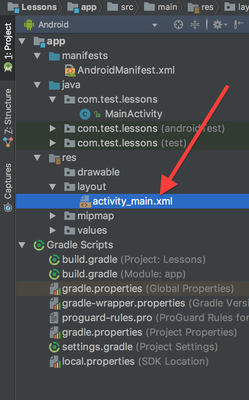
А весь белый фон — это Activity, экран, в котором располагаются остальные элементы интерфейса.

Создание интерфейса

Основной подход при разработке интерфейса — декларативный. Это значит, что вы создаете интерфейс не в Java-коде, а описываете его в специальных XML-файлах. Почти как верстка HTML :)

Помните, в прошлом уроке мы с вами говорили о ресурсах приложения? Так вот, верстка экранов — один из типов ресурсов. Файлы верстки находятся в директории res/layout.

Взгляните, в новом проекте уже есть один файл!

[](https://skillberg.com/media/uploads/2017/11/17/932c7b84e6bb4e52ae17c1a9bd3c6aaa.png)Лэйаут Activity

Этот файл описывает интерфейс нашей единственной Activity.

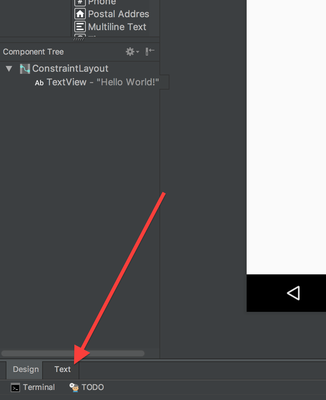
**Совет по организации файлов layout:** в отличие от исходного кода Java, тут нет возможности создавать вложенные директории. Поэтому в сколько-нибудь сложном проекте в директории layout появляется очень большое количество файлов. Чтобы не запутаться что есть что, следуйте простому правилу именования файлов:

* **activity**\_name.xml — для вёрстки Activity.
* **fragment**\_name.xml — для вёрстки Fragment (о них поговорим в одном из следующих уроков).
* **view**\_name.xml — для вёрстки View.

Здесь "name" — имя элемента интерфейса. То есть файл для Activity с названием MainActivity должен называться activity\_main.xml, а для PostsActivity — activity\_posts.xml.

Итак, вернемся к нашему activity\_main.xml.

Если вы откроете этот файл, то увидите визуальный редактор. На самом деле, намного удобнее и надежнее работать с XML-кодом, нежели с редактором. Внизу вы увидите две вкладки — **Design** и **Text**. Переключитесь на **Text**.

[](https://skillberg.com/media/uploads/2017/11/17/66083ed5c31141bb83fb14575baab439.png)Переключение на редактирование XML

Вероятнее всего, вы увидите вот такой код:

<?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"

tools:context="com.test.lessons.MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>

Это ConstraintLayout. Мы обязательно поработаем с ним позже, но пока что он слишком сложен. Замените код на такой:

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.test.lessons.MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:text="Hello World!" />

</FrameLayout>

Запустите приложение. Как видите, визуально ничего не поменялось.

Давайте же разберем, что тут написано.

Корневой элемент — FrameLayout. Это самый простой из возможных Layout-ов.

Внутри него находится TextView — это View, задачей которого является отображение текста. Получается, FrameLayout является контейнером для TextView.

Чтобы было понятнее, покажу структуру, отбросив все атрибуты:

<FrameLayout>

<TextView />

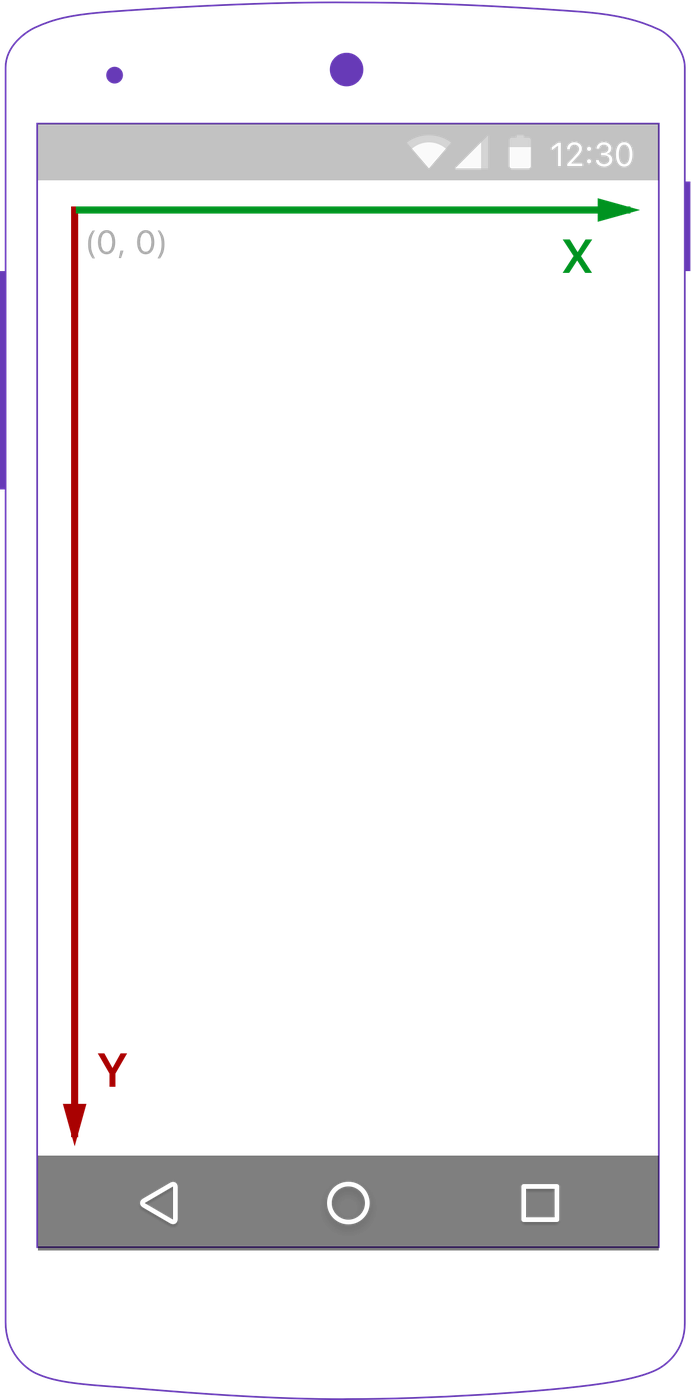
</FrameLayout>

Теперь давайте посмотрим, что такое FrameLayout.

FrameLayout

Все, что может FrameLayout — отображать View друг над другом (по оси **Z**). Вспомните курс математики, точнее, систему координат.

В двумерном пространстве есть две оси — **X** (слева направо) и **Y** (снизу вверх). В Android, кстати, это работает немного иначе — ось **Y** растет вниз, а не вверх.

[](https://skillberg.com/media/uploads/2017/11/17/6cc8d1cf3a924d8e99968823715f864e.png)Оси координат в Android

В трёхмерном пространстве появляется ось **Z**, идущая "на нас". В контексте интерфейсов **Z** обозначает расположение элемента один над другим.

Давайте поэкспериментируем с FrameLayout. Замените TextView на такой View:

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.test.lessons.MainActivity">

<View

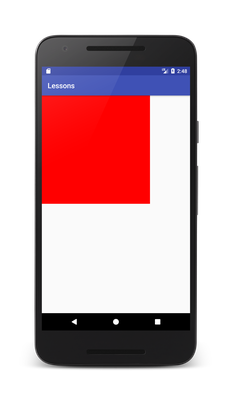
android:layout\_width="300dp"

android:layout\_height="300dp"

android:background="#ff0000" />

</FrameLayout>

Запустив приложение, вы увидите такую картинку:

[](https://skillberg.com/media/uploads/2017/11/17/fd12f48e8adf4aaa9541a84d4ea7b521.png)FrameLayout

Как видите, View расположился в верхнем левом углу — по умолчанию View позиционируются с точки (0, 0).

Обратите внимание на атрибуты у XML-тегов. Давайте рассмотрим их.

С помощью этого атрибута android:layout\_width задается ширина элемента. Может принимать значения:

* match\_parent— занимает столько же места, сколько и родительский элемент.
* wrap\_content — занимает ровно столько места, сколько нужно, чтобы вместить весь контент внутри View.
* Значение в пикселях (px), независимых от плотности пикселях (dp), пикселях, масштабируемых под размер шрифта (sp), дюймах (in) или миллиметрах (mm).

Пока что мы будем использовать только константы match\_parent и wrap\_content, а так же значения в dp. Что такое dp мы разберем попозже.

android:layout\_height — то же самое, что android:layout\_width, только для высоты.

FrameLayout должен занимать все пространство на экране, поэтому мы задали константу match\_parent для ширины и высоты. В итоге, все белое пространство (и пространство под красным View) — это FrameLayout.

Для View же мы задали фиксированный размер в dp, поэтому он именно такого размера.

android:background — может принимать следующие значения:

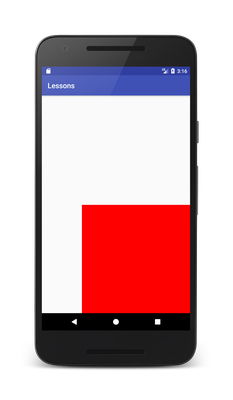
* ссылка на другой ресурс (об этом позже)
* цвет в формате #rgb
* цвет в формате #argb
* цвет в формате #rrggbb
* цвет в формате #aarrggbb

Мы задали цвет в формате #rrggbb, равный #ff0000, то есть красный.

В Android также есть такое понятие, как LayoutParams — параметры отображения View в контейнере. Они разные для каждого контейнера. Для FrameLayout LayoutParams содержат лишь один атрибут — android:layout\_gravity. Этот атрибут позволяет задать расположение View внутри FrameLayout. Возможные значения:

* bottom — объект "прижимается" к низу контейнера.
* top — объект прижимается к верху контейнера.
* left — объект прижимается к левой стороне контейнера. Использовать это значение не рекомендуется из-за отсутствия поддержки RTL-локалей.
* right — объект прижимается к правой стороне контейнера. Использовать это значение не рекомендуется из-за отсутствия поддержки RTL-локалей.
* start — объект прижимается к "началу" контейнера. В локалях с расположением элементов слева направо, LTR (то есть в большинстве) это аналог left, а в RTL — right.
* end — объект прижимается к "концу" контейнера. В локалях с расположением элементов слева направо, LTR (то есть в большинстве) это аналог right, а в RTL — left.
* center\_horizontal — объект центрируется относительно контейнера по оси **X**.
* center\_vertical — объект центрируется относительно контейнера по оси **Y**.
* center — объект располагается по центру контейнера.

Значения можно комбинировать, используя символ "|", например android:layout\_gravity="end|bottom" в LTR локалях прижмет View в правый нижний угол, а в RTL — в левый нижний:

[](https://skillberg.com/media/uploads/2017/11/17/b6715519c065441f80378fe38052c2e3.png)FrameLayout — View в правом нижнем углу

Изменим код на такой:

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.test.lessons.MainActivity">

<View

android:layout\_width="300dp"

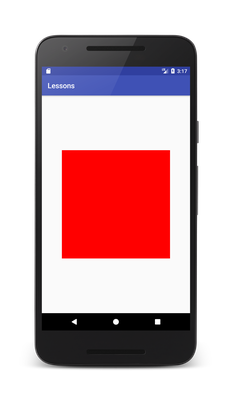
android:layout\_height="300dp"

android:layout\_gravity="center"

android:background="#ff0000" />

</FrameLayout>

Получим такой результат:

[](https://skillberg.com/media/uploads/2017/11/17/2b3de335c174404bb2a6d26dd653ade7.png)FrameLayout — View в центре

Теперь давайте разберемся, как FrameLayout выстраивает дочерние View по оси **Z**.

Добавим еще 2 View:

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.test.lessons.MainActivity">

<View

android:layout\_width="300dp"

android:layout\_height="300dp"

android:layout\_gravity="center"

android:background="#ff0000" />

<View

android:layout\_width="200dp"

android:layout\_height="200dp"

android:layout\_gravity="center"

android:background="#00ff00" />

<View

android:layout\_width="100dp"

android:layout\_height="100dp"

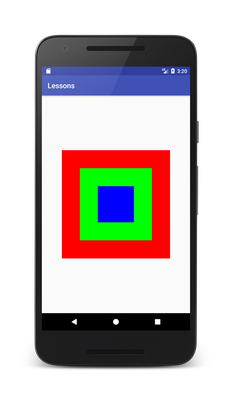
android:layout\_gravity="center"

android:background="#0000ff" />

</FrameLayout>

Как вы могли заметить, я изменил цвета и размеры View, чтобы можно было понять, какой из них где находится.

Запустите приложение:

[](https://skillberg.com/media/uploads/2017/11/17/d25f0a82c8ad4bc1974e7ad09dc22eb7.png)FrameLayout: View друг над другом в центре

Видно, что красный View находится ниже всех (он был объявлен первым), зеленый — над ним (он второй) и синий "выше" всех (он был объявлен последним).

Что мы еще можем сделать с этими View? Например, раскидать их по разным углам. Уменьшим размер всех View до 100dp и добавим еще один View:

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.test.lessons.MainActivity">

<View

android:layout\_width="100dp"

android:layout\_height="100dp"

android:layout\_gravity="start|top"

android:background="#ff0000" />

<View

android:layout\_width="100dp"

android:layout\_height="100dp"

android:layout\_gravity="end|top"

android:background="#00ff00" />

<View

android:layout\_width="100dp"

android:layout\_height="100dp"

android:layout\_gravity="start|bottom"

android:background="#0000ff" />

<View

android:layout\_width="100dp"

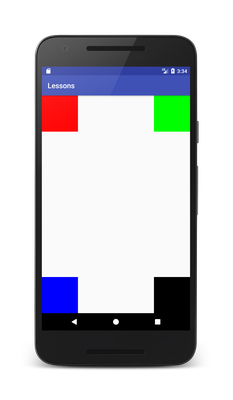
android:layout\_height="100dp"

android:layout\_gravity="end|bottom"

android:background="#000000" />

</FrameLayout>

Получим вот такой результат:

[](https://skillberg.com/media/uploads/2017/11/17/25534bb731244017b10832600847b3b2.png)FrameLayout: View по углам

Пожалуй, для этого урока хватит :)

В следующем уроке мы рассмотрим LinearLayout.