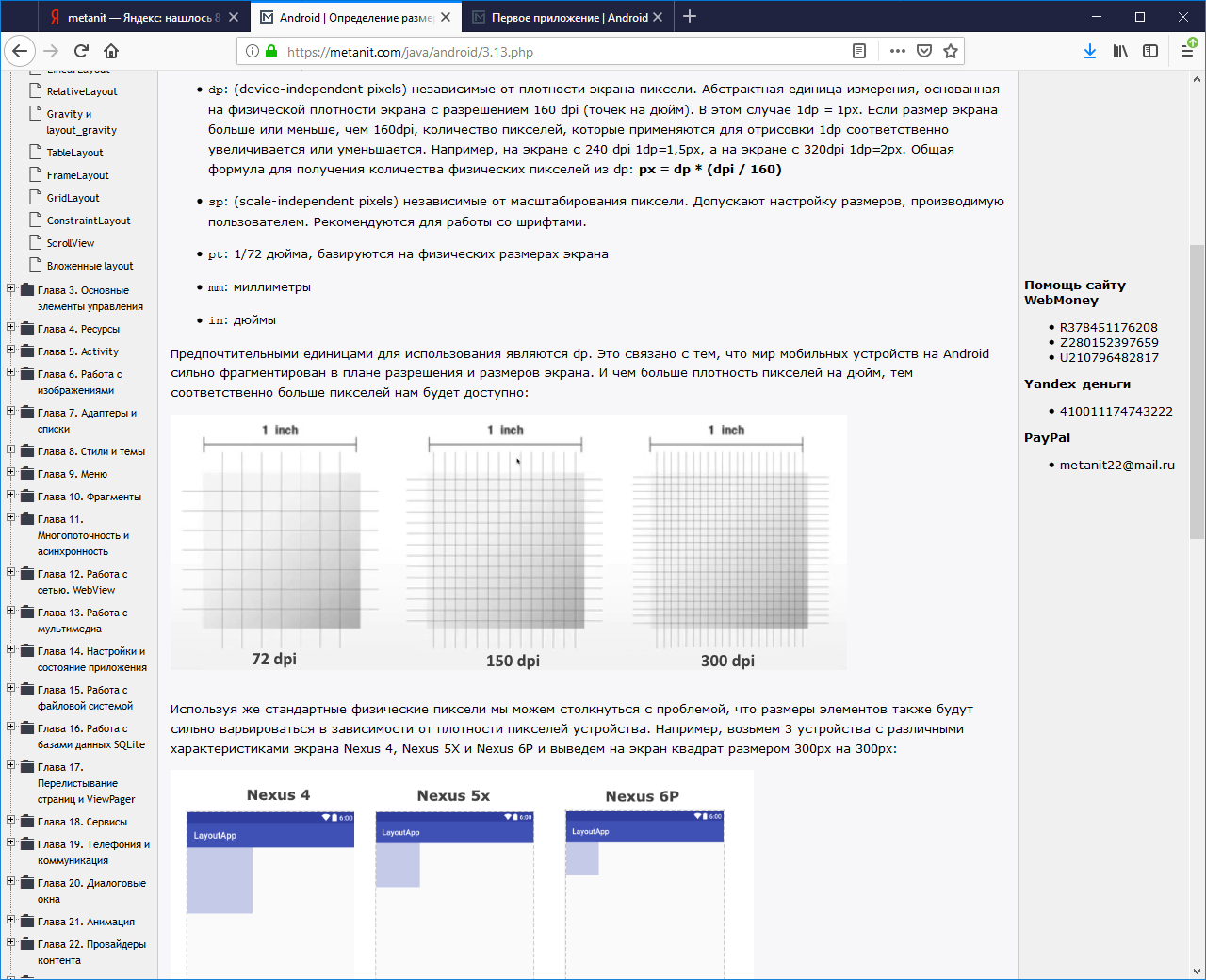
**Определение размеров**

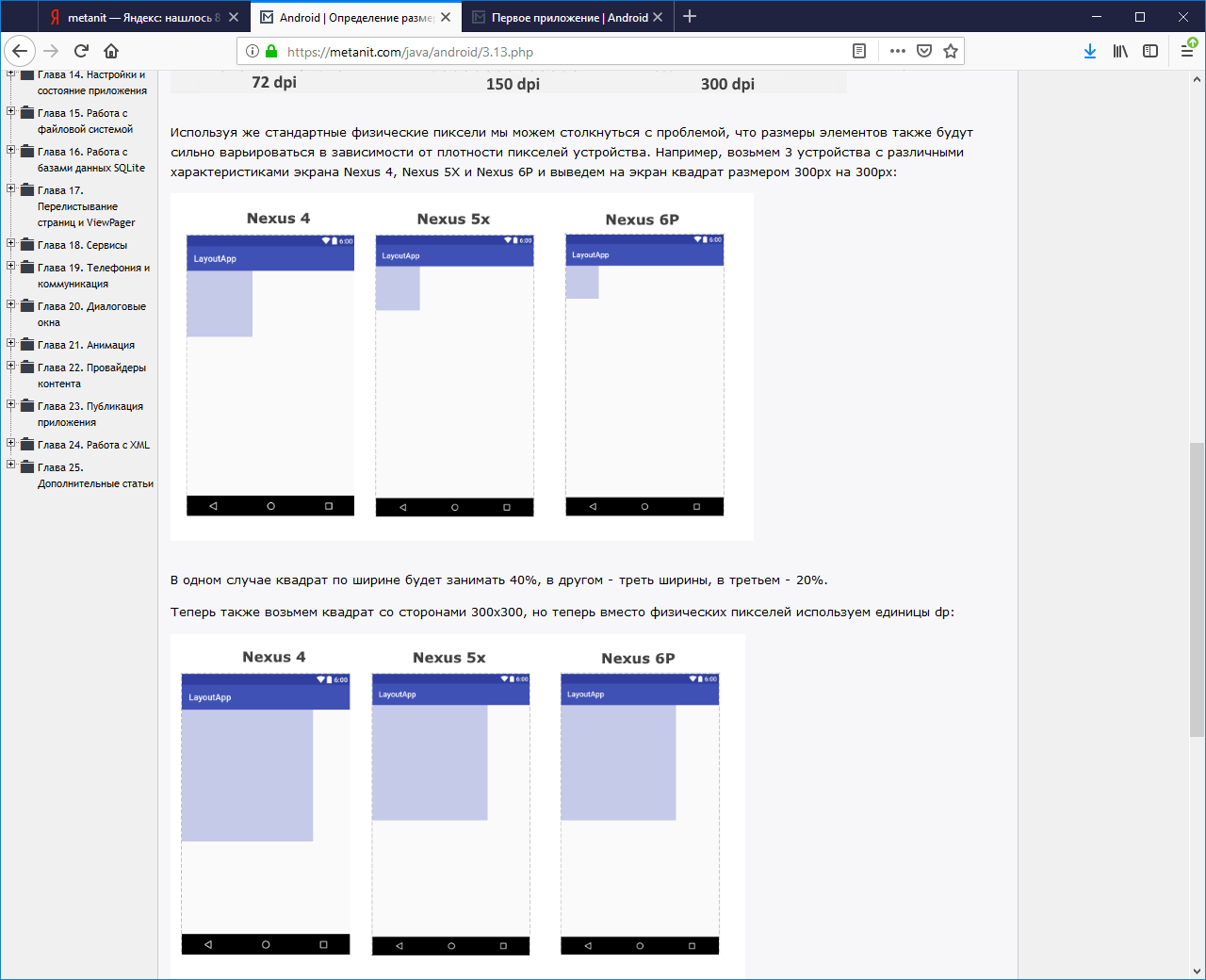
В ОС Android мы можем использовать различные типы измерений:

* px: пиксели текущего экрана. Однако эта единица измерения не рекомендуется, так как реальное представление внешнего вида может изменяться в зависимости от устройства; каждое устройство имеет определенный набор пикселей на дюйм, поэтому количество пикселей на экране может также меняться
* dp: (device-independent pixels) независимые от плотности экрана пиксели. Абстрактная единица измерения, основанная на физической плотности экрана с разрешением 160 dpi (точек на дюйм). В этом случае 1dp = 1px. Если размер экрана больше или меньше, чем 160dpi, количество пикселей, которые применяются для отрисовки 1dp соответственно увеличивается или уменьшается. Например, на экране с 240 dpi 1dp=1,5px, а на экране с 320dpi 1dp=2px. Общая формула для получения количества физических пикселей из dp: px = dp \* (dpi / 160)
* sp: (scale-independent pixels) независимые от масштабирования пиксели. Допускают настройку размеров, производимую пользователем. Рекомендуются для работы со шрифтами.
* pt: 1/72 дюйма, базируются на физических размерах экрана
* mm: миллиметры
* in: дюймы

Предпочтительными единицами для использования являются dp. Это связано с тем, что мир мобильных устройств на Android сильно фрагментирован в плане разрешения и размеров экрана. И чем больше плотность пикселей на дюйм, тем соответственно больше пикселей нам будет доступно:

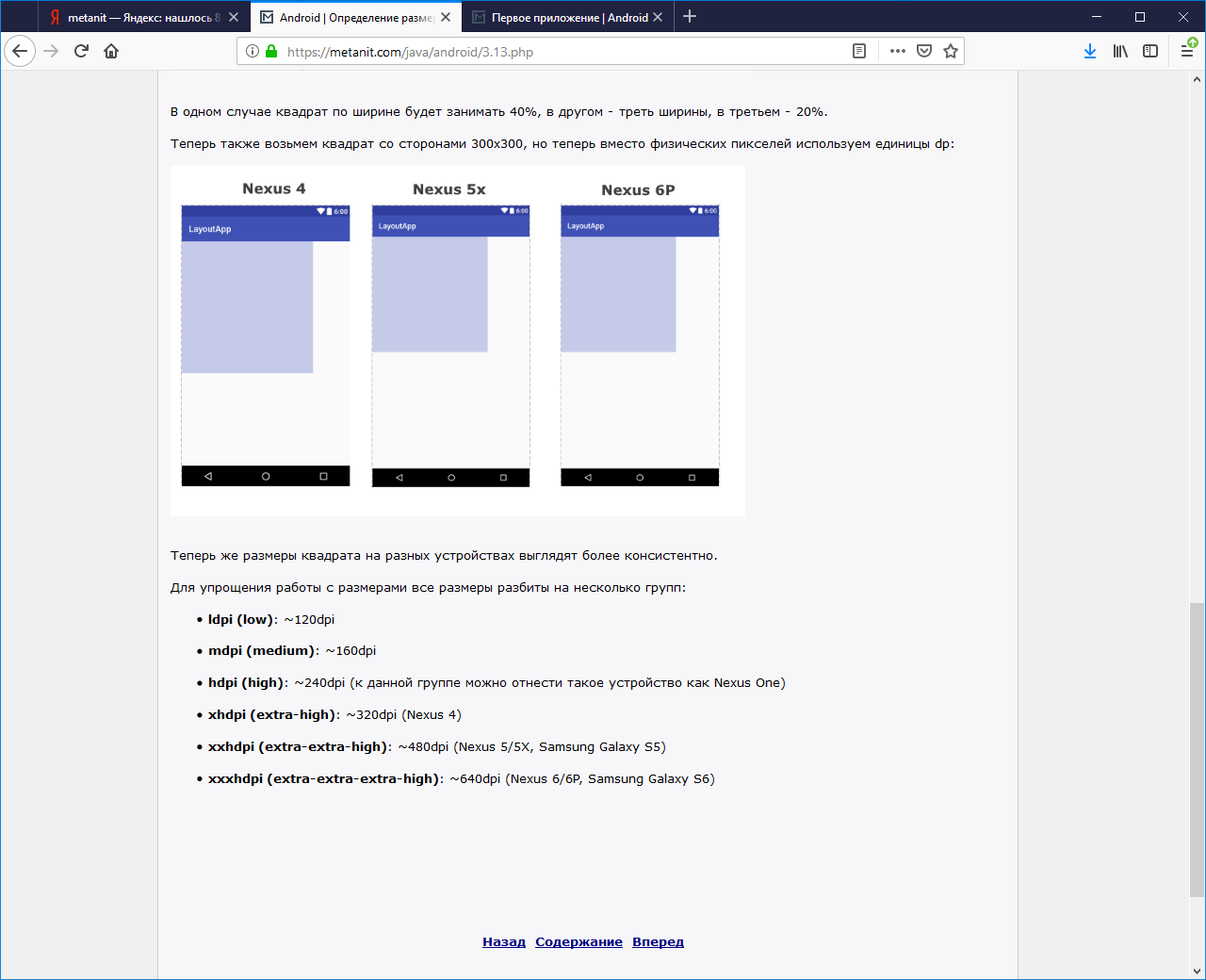


Используя же стандартные физические пиксели мы можем столкнуться с проблемой, что размеры элементов также будут сильно варьироваться в зависимости от плотности пикселей устройства. Например, возьмем 3 устройства с различными характеристиками экрана Nexus 4, Nexus 5X и Nexus 6P и выведем на экран квадрат размером 300px на 300px:



В одном случае квадрат по ширине будет занимать 40%, в другом - треть ширины, в третьем - 20%.

Теперь также возьмем квадрат со сторонами 300х300, но теперь вместо физических пикселей используем единицы dp:



Теперь же размеры квадрата на разных устройствах выглядят более консистентно.

Для упрощения работы с размерами все размеры разбиты на несколько групп:

* ldpi (low): ~120dpi
* mdpi (medium): ~160dpi
* hdpi (high): ~240dpi (к данной группе можно отнести устройство Nexus One)
* xhdpi (extra-high): ~320dpi (Nexus 4)
* xxhdpi (extra-extra-high): ~480dpi (Nexus 5/5X, Samsung Galaxy S5)
* xxxhdpi (extra-extra-extra-high): ~640dpi (Nexus 6/6P, Samsung Galaxy S6)

**Ширина и высота элементов**

Все визуальные элеметы, которые мы используем в приложении, как правило, упорядочиваются на экране с помощью контейнеров. В Android подобными контейнерами служат такие классы как RelativeLayout, LinearLayout, GridLayout, TableLayout, ConstraintLayout, FrameLayout. Все они по разному располагают элементы и управляют ими, но есть некоторые общие моменты при компоновке визуальных компонентов, которые мы сейчас рассмотрим.

Для организации элементов внутри контейнера используются параметры разметки. Для их задания в файле xml используются атрибуты, которые начинаются с префикса layout\_. В частности, к таким параметрам относятся атрибуты layout\_height и layout\_width, которые используются для установки размеров и могут принимать одно из следующих значений:

* точные размеры элемента, например 96 dp
* значение wrap\_content: элемент растягивается до тех границ, которые достаточны, чтобы вместить все его содержимое
* значение match\_parent: элемент заполняет всю область родительского контейнера

Например:

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

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

    android:id="@+id/activity\_main"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    >

    <TextView

        android:text="Hello Android 7"

        android:layout\_height="wrap\_content"

        android:layout\_width="wrap\_content"

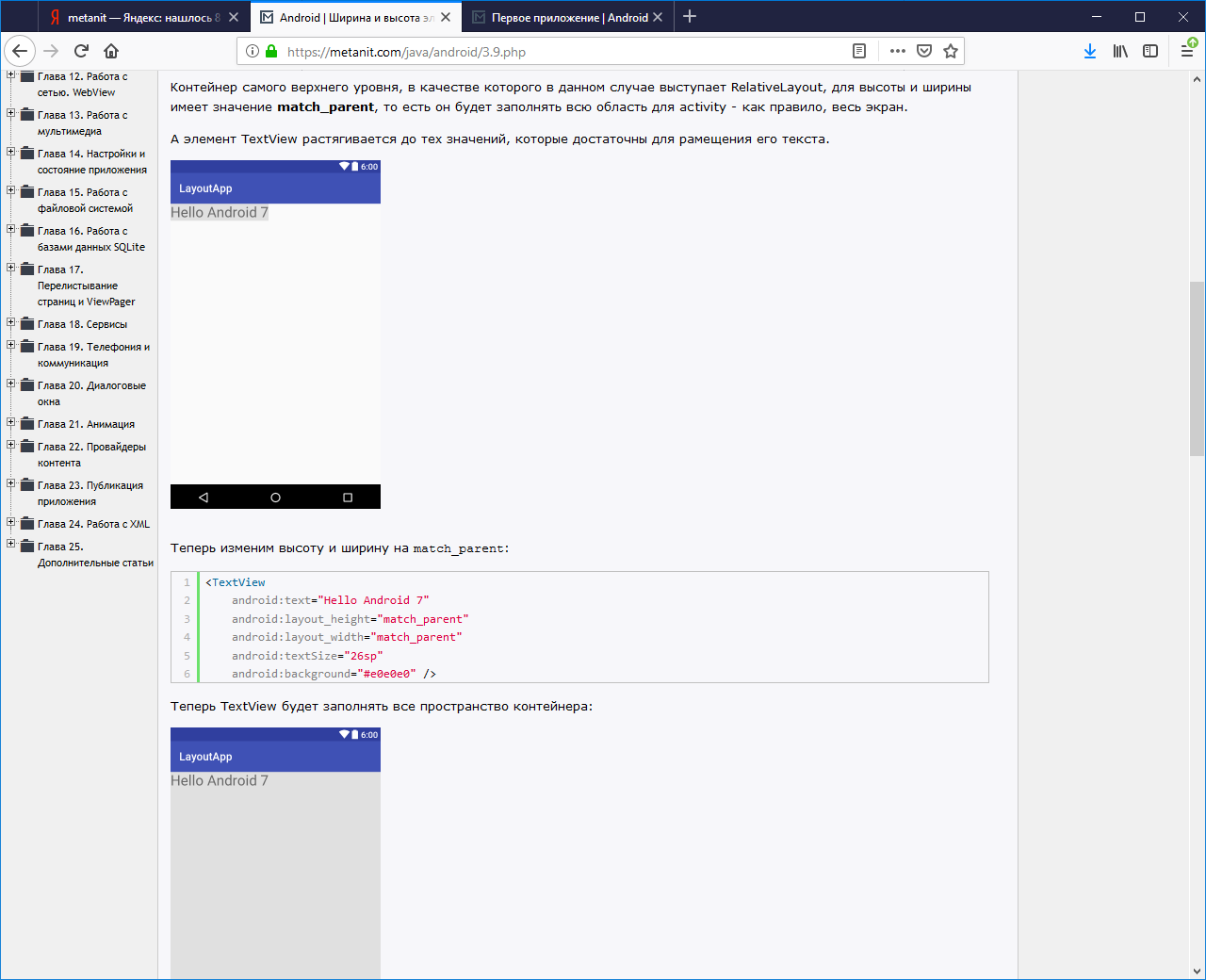
        android:textSize="26sp"

        android:background="#e0e0e0" />

</RelativeLayout>

Контейнер самого верхнего уровня, в качестве которого в данном случае выступает RelativeLayout, для высоты и ширины имеет значение match\_parent, то есть он будет заполнять всю область для activity - как правило, весь экран.

А элемент TextView растягивается до тех значений, которые достаточны для рамещения его текста.



Теперь изменим высоту и ширину на match\_parent:

<TextView

    android:text="Hello Android 7"

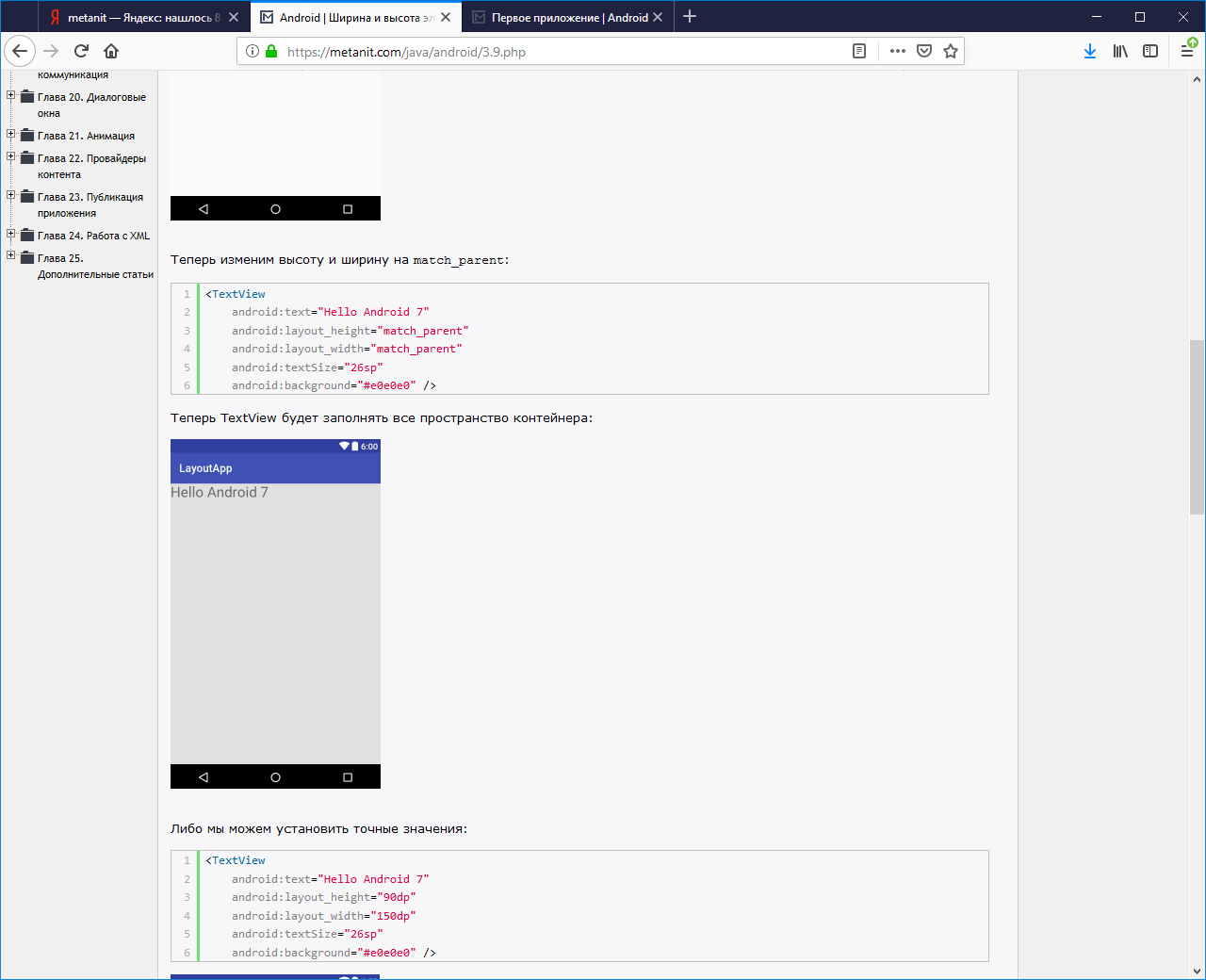
    android:layout\_height="match\_parent"

    android:layout\_width="match\_parent"

    android:textSize="26sp"

    android:background="#e0e0e0" />

Теперь TextView будет заполнять все пространство контейнера:



Либо мы можем установить точные значения:

<TextView

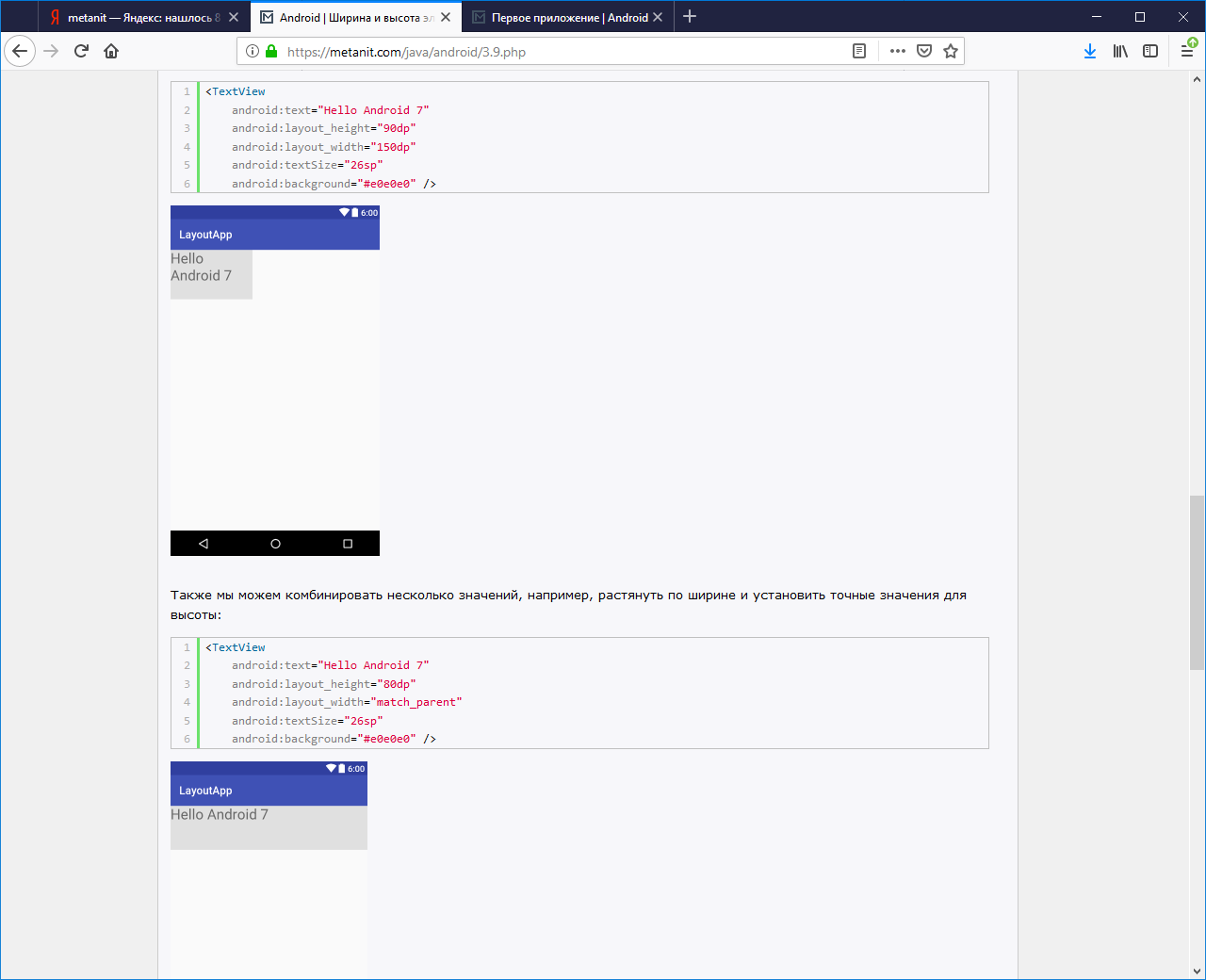
    android:text="Hello Android 7"

    android:layout\_height="90dp"

    android:layout\_width="150dp"

    android:textSize="26sp"

    android:background="#e0e0e0" />



Также мы можем комбинировать несколько значений, например, растянуть по ширине и установить точные значения для высоты:

<TextView

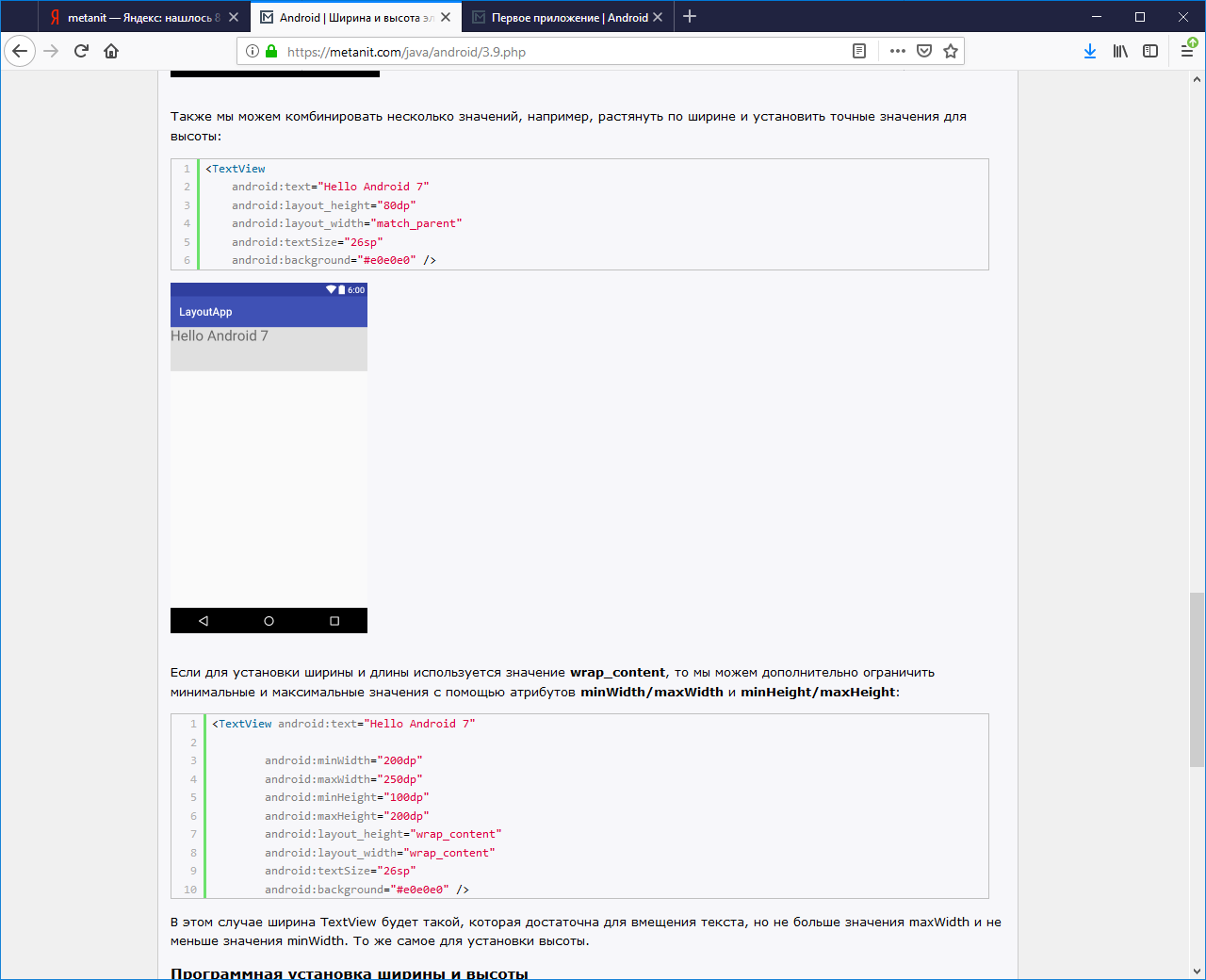
    android:text="Hello Android 7"

    android:layout\_height="80dp"

    android:layout\_width="match\_parent"

    android:textSize="26sp"

    android:background="#e0e0e0" />



Если для установки ширины и длины используется значение **wrap\_content**, то мы можем дополнительно ограничить минимальные и максимальные значения с помощью атрибутов **minWidth/maxWidth и minHeight/maxHeight**:

<TextView android:text="Hello Android 7"

        android:minWidth="200dp"

        android:maxWidth="250dp"

        android:minHeight="100dp"

        android:maxHeight="200dp"

        android:layout\_height="wrap\_content"

        android:layout\_width="wrap\_content"

        android:textSize="26sp"

        android:background="#e0e0e0" />

В этом случае ширина TextView будет такой, которая достаточна для вмещения текста, но не больше значения **maxWidth** и не меньше значения **minWidth**. То же самое для установки высоты.

**Программная установка ширины и высоты**

Если элемент, к примеру, тот же TextView создается в коде java, то для установки высоты и ширины можно использовать метод **setLayoutParams().** Так, изменим код MainActivity:

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.ViewGroup;

import android.widget.RelativeLayout;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        RelativeLayout relativeLayout = new RelativeLayout(this);

        TextView textView1 = new TextView(this);

        textView1.setText("Hello Android 7");

        textView1.setTextSize(26);

        // устанавливаем размеры

        textView1.setLayoutParams(new ViewGroup.LayoutParams

                (ViewGroup.LayoutParams.WRAP\_CONTENT, ViewGroup.LayoutParams.WRAP\_CONTENT));

        // добавляем TextView в RelativeLayout

        relativeLayout.addView(textView1);

        setContentView(relativeLayout);

    }

}

В метод setLayoutParams() передается объект ViewGroup.LayoutParams. Этот объект инициализируется двумя параметрами: шириной и высотой. Для указания ширины и высоты можно использовать одну из констант ViewGroup.LayoutParams.WRAP\_CONTENT или ViewGroup.LayoutParams.MATCH\_PARENT.

Также мы можем передать точные значения или комбинировать типы значений:

textView1.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, 200));