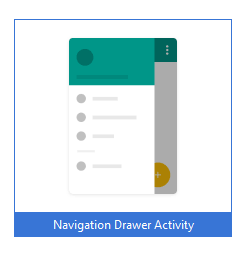
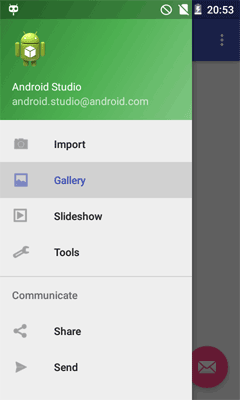
Шаблон Navigation Drawer Activity

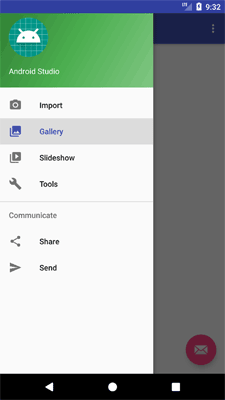
Рассмотрим новый шаблон **Navigation Drawer Activity**. Создадим новый проект и выберем нужный шаблон.



Для беглого знакомства можете сразу запустить проект и посмотреть его в действии. При запуске приложение выглядит как обычная программа "Hello World". Но есть и отличия. Нажмите на значок в виде трёх горизонтальных полосок в заголовке. Значок в документации называется "гамбургером" (Hamburger menu). Это официальная позиция Гугла. Но в реальности значок символизирует полосатых котов (никому не рассказывайте). При нажатии слева вылезет навигационная шторка. Шторка работает как обычная шторка в ванной. По высоте она занимает весь экран, включая системную область. Можете подвигать шторку вперёд-назад, чтобы увидеть, что верхняя кромка шторки в системной области полупрозрачна и не закрывает системные значки. Подобное поведение доступно на устройствах под Android 5 и выше. На старых устройствах шторка находится под системной панелью. Недавно стал проверять работу под Android 8.0 и увидел, что шторка теперь не закрывает системную панель. Ниже для сравнения я привёл два варианта.



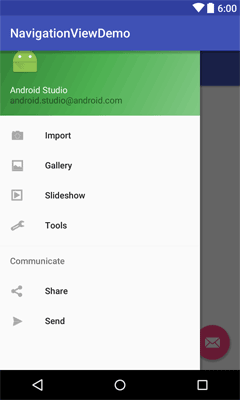
Шторка закрывает системную панель

Шторка не закрывает системную панель

Сама шторка состоит из двух основных частей - в верхней части находится картинка и текст, а в нижней - меню со значками. Меню в свою очередь разделено на две группы. В верхней части значки можно выбрать и выбранный пункт останется выделенным. В нижней части меню пункты не выделяются. Уберите шторку обратно и вызовите теперь её не нажатием на значок гамбургера, а движением пальца от края экран в центр. Получилось? Отлично, а теперь выдвигайте шторку медленно и наблюдайте за значком гамбургера. Вы увидите, что во время движения значок трансформируется. К сожалению, шторка закрывает значок и непонятно, во что превращаются три полоски. А превращаются они в  стрелку. Позже я покажу, как увидеть её. А может не покажу, я ещё не решил.

Возвращаемся в студию и начинаем изучать код проекта.

Если открыть файл **activity\_main.xml** в режиме **Design**, то можно увидеть, как будет выглядеть приложение с открытой шторкой.



Как видите, небольшие расхождения имеются, но в целом совпадает с реальным приложением.

Посмотрим на его содержание.

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

<android.support.v4.widget.DrawerLayout 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:id="@+id/drawer\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:fitsSystemWindows="true"

tools:openDrawer="start">

<include

layout="@layout/app\_bar\_main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

<android.support.design.widget.NavigationView

android:id="@+id/nav\_view"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

android:fitsSystemWindows="true"

app:headerLayout="@layout/nav\_header\_main"

app:menu="@menu/activity\_main\_drawer" />

</android.support.v4.widget.DrawerLayout>

Сейчас важно запомнить, что за выдвигающую шторку отвечает элемент **NavigationView**, который входит последним в контейнере **DrawerLayout** и представляет собой навигационное меню. А перед меню находится вставка **include**, указывающая на разметку **app\_bar\_main.xml**.

Атрибут **tools:openDrawer** позволяет указать студии, что навигационное меню нужно отобразить в раскрытом виде в режиме просмотра разметки.

NavigationView

В 2014 году Google показал новый дизайн и различные новые примеры по навигации. Но вначале они использовали подручные средства, которые были под рукой - фрагменты.

Спустя год компания разработала на основе предка **FrameLayout** новый компонент **NavigationView**, который стал частью библиотеки [Android Design Support Library](http://developer.alexanderklimov.ru/android/android-support-design.php).

Новый подход оказался неожиданным, но логичным. Раз выдвижная шторка содержит навигационное меню, то и класс был спроектирован как меню. Вы можете создать элементы меню в ресурсах **res/menu** стандартным способом и получить готовую навигацию.

Необходимые рекомендации по созданию навигационной выдвижной шторки можно найти на странице [Navigation drawer - Patterns](https://material.io/guidelines/patterns/navigation-drawer.html).

Перейдём к деталям.

Теперь в рекомендациях не указаны точные размеры шторки, хотя раньше нужно было самостоятельно указать ширину шторки. Видимо, новый класс сам обеспечивает необходимую ширину. Интерес представляют два последних атрибута.

Тег **NavigationView** содержит ссылку на собственную разметку в атрибуте **app:headerLayout**, который указывает на файл **nav\_header\_main.xml** (верхняя часть шторки), а также на меню в атрибуте **app:menu**, который ссылается на ресурс меню **menu/activity\_main\_drawer.xml**.

Откроем файл **nav\_header\_main.xml** и посмотрим на разметку шторки.

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

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

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout\_width="match\_parent"

android:layout\_height="@dimen/nav\_header\_height"

android:background="@drawable/side\_nav\_bar"

android:gravity="bottom"

android:orientation="vertical"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:theme="@style/ThemeOverlay.AppCompat.Dark">

<ImageView

android:id="@+id/imageView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingTop="@dimen/nav\_header\_vertical\_spacing"

app:srcCompat="@android:drawable/sym\_def\_app\_icon" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="@dimen/nav\_header\_vertical\_spacing"

android:text="Android Studio"

android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="android.studio@android.com" />

</LinearLayout>

Разметка состоит из **ImageView** и двух **TextView**, размещённых в контейнере **LinearLayout**. Фон контейнера определён в ресурсе **drawable/side\_nav\_bar.xml** и представляет собой градиент.

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

android:shape="rectangle">

<gradient

android:angle="135"

android:centerColor="#4CAF50"

android:endColor="#2E7D32"

android:startColor="#81C784"

android:type="linear" />

</shape>

Остальные атрибуты понятны и не требуют пояснений.

Можно (но не нужно) настроить верхнюю часть шторки не через XML, а программно.

NavigationView navigationView = (NavigationView) findViewById(R.id.nav\_view);

// Наполняем шапку элементами

View headerLayout = navigationView.inflateHeaderView(R.layout.nav\_header\_main);

ImageView headerImageView = headerLayout.findViewById(R.id.imageView);

После обновления одной из версий библиотеки **Design Support**, доступ к шапке осуществляется теперь через другой код.

View headerLayout = navigationView.getHeaderView(0);

Теперь рассмотрим ресурс навигационного меню **res/menu/activity\_main\_drawer.xml**.

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

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

<group android:checkableBehavior="single">

<item

android:id="@+id/nav\_camera"

android:icon="@drawable/ic\_menu\_camera"

android:title="Import" />

<item

android:id="@+id/nav\_gallery"

android:icon="@drawable/ic\_menu\_gallery"

android:title="Gallery" />

<item

android:id="@+id/nav\_slideshow"

android:icon="@drawable/ic\_menu\_slideshow"

android:title="Slideshow" />

<item

android:id="@+id/nav\_manage"

android:icon="@drawable/ic\_menu\_manage"

android:title="Tools" />

</group>

<item android:title="Communicate">

<menu>

<item

android:id="@+id/nav\_share"

android:icon="@drawable/ic\_menu\_share"

android:title="Share" />

<item

android:id="@+id/nav\_send"

android:icon="@drawable/ic\_menu\_send"

android:title="Send" />

</menu>

</item>

</menu>

Принцип создания элементов меню остался стандартным. Каждый пункт меню представляет собой тег **item** с указанием значка и текста. Для группировки используется элемент **group**. Поведение элементов меню в группе регулируется атрибутом **android:checkableBehavior**. В примере используется значение **single** - при нажатии на пункт меню, он останется выделенным (принцип переключателя **RadioButton**). Всего доступно три варианта.

* **single** – можно выбрать один элемент группы (переключатель)
* **all** - можно выбрать все элементы группы (флажок)
* **none** – элементы не выбираются

В библиотеке **Android Support Design** версии 23 вариант **all** не работает и будет действовать, как со значением **single**.

Также следует обратить внимание, что теперь проект ссылается на векторные рисунки, которые находятся в папке **drawable-21**.

Осталось рассмотреть тег **include**, который ссылается на файл ресурса **res/layout/app\_bar\_main.xml**. Он вам будет знаком по шаблону **Blank Activity**, который мы изучали в статье [Библиотека Android Support Design](http://developer.alexanderklimov.ru/android/android-support-design.php). Только там он находился в файле **activity\_main.xml**, а здесь его перенесли в файл **app\_bar\_main.xml**. Всё остальное осталось без изменений. Повторяться не будем.

Теперь изучим код активности для работы со шторкой.

В классе активности реализуется интерфейс **OnNavigationItemSelectedListener** с его методом **onNavigationItemSelected()**:

public class MainActivity extends AppCompatActivity

**implements NavigationView.OnNavigationItemSelectedListener** {

@SuppressWarnings("StatementWithEmptyBody")

@Override

public boolean onNavigationItemSelected(MenuItem item) {

// Handle navigation view item clicks here.

int id = item.getItemId();

if (id == R.id.nav\_camara) {

// Handle the camera action

} else if (id == R.id.nav\_gallery) {

} else if (id == R.id.nav\_slideshow) {

} else if (id == R.id.nav\_manage) {

} else if (id == R.id.nav\_share) {

} else if (id == R.id.nav\_send) {

}

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);

drawer.closeDrawer(GravityCompat.START);

return true;

}

}

Логика кода ничем не отличается о работы с обычным меню. Определяется идентификатор выбранного пункта и далее вам нужно написать свой код. Затем вызывается метод **closeDrawer()** для закрытия шторки.

Добавим код для первого пункта меню.

if (id == R.id.nav\_camera) {

// Handle the camera action

Toast.makeText(getApplicationContext(), "Вы выбрали камеру", Toast.LENGTH\_SHORT).show();

}

При нажатии кнопки "Назад" проверяется состояние шторки. Если шторка открыта (**isDrawerOpen()**), то её закрываем с помощью метода **closeDrawer()**.

@Override

public void onBackPressed() {

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);

if (drawer.isDrawerOpen(GravityCompat.START)) {

drawer.closeDrawer(GravityCompat.START);

} else {

super.onBackPressed();

}

}

В методе **onCreate()** происходит инициализация шторки.

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

...

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);

ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(

this, drawer, toolbar, R.string.navigation\_drawer\_open, R.string.navigation\_drawer\_close);

drawer.(toggle);

toggle.syncState();

NavigationView navigationView = (NavigationView) findViewById(R.id.nav\_view);

navigationView.setNavigationItemSelectedListener(this);

}

Метод **setDrawerListener()** помечен как устаревший. Замените на строчку:

drawer.addDrawerListener(toggle);

Возможно, в следующих версиях студии код поправят.

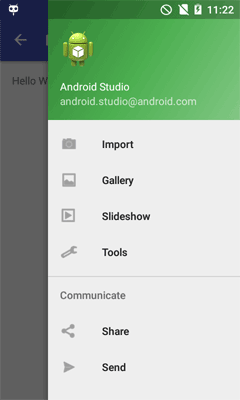
Теперь поговорим об изменениях, которые можно внести в проект.

Хотите выдвигать шторку справа? Установите значение **end** у атрибута **layout\_gravity**. Обычно используется для стран с обратным порядком букв.

<android.support.design.widget.NavigationView

...

android:layout\_gravity="end"/>



На самом деле смысла в этом не оказалось. Да, шторка выдвигается вручную. Но если нажать на значок гамбургера, то приложение валится с ошибкой. Любое нажатие в меню шторки также приводит к ошибке. Теоретически можно написать код, который исправит проблему, но он будет сложным. Забудьте об этом совете.

Тем не менее, можно реализовать забавный эффект - добавить вторую шторку на экран. Первая будет работать главной и реагировать на нажатие значка, а вторая будет дополнительной для вывода какой-то информации. Достаточно в разметку добавить второй **NavigationView** с атрибутом **android:layout\_gravity="end"**

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

<android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"

...

<android.support.design.widget.NavigationView

...

android:layout\_gravity="start"

... />

<android.support.design.widget.NavigationView

android:id="@+id/nav\_view2"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

**android:layout\_gravity="end"**

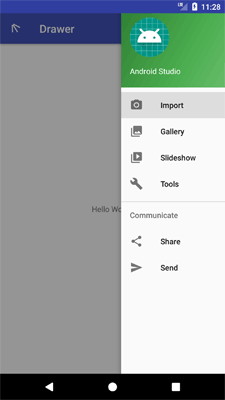
android:fitsSystemWindows="true"

app:headerLayout="@layout/nav\_header\_main"

app:menu="@menu/activity\_main\_drawer" />

</android.support.v4.widget.DrawerLayout>

Кстати, если выдвигать правую шторку, то можно наблюдать трансформацию значка без помех. Скриншот во время частичного выдвижения шторки справа.



Для изменения цвета значков и текста в навигационном меню используйте атрибуты **app:itemIconTint** и **app:itemTextColor**.

<android.support.design.widget.NavigationView

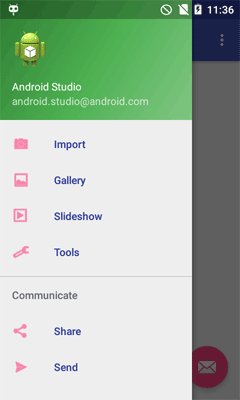
...

app:itemIconTint="@color/colorAccent"

app:itemTextColor="@color/colorPrimaryDark"/>

Данным атрибутам соответствуют методы **setItemIconTintList()** и **setItemTextColor()** компонента **NavigationView**.

﻿



Так как наличие шторки не совсем очевидно, можно при первом запуске показать шторку в раскрытом состоянии. Далее можно запомнить состояние шторки при первом запуске в настройках, чтобы во второй раз выводить её уже в закрытом состоянии. Можете самостоятельно реализовать эту возможность.

// показываем в открытом состоянии в onCreate()

drawer.openDrawer(GravityCompat.START);

Напоследок покажу превращение значка гамбургера в стрелку в явном виде, как и обещал в начале статьи. Напомню, что по рекомендации Material Design шторка должна закрывать всю область экрана. Но если вы хотите поместить шторку под заголовком, то следует немного поправить разметку. Откроем файл **app\_bar\_main.xml** и вырежем из него небольшой кусок. Затем в файле **activity\_main.xml** добавим **LinearLayout** в качестве корневого контейнера и вставим скопированный ранее кусок кода.

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

<LinearLayout

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

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

**<android.support.design.widget.AppBarLayout**

**android:layout\_height="wrap\_content"**

**android:layout\_width="match\_parent"**

**android:theme="@style/AppTheme.AppBarOverlay">**

**<android.support.v7.widget.Toolbar**

**android:id="@+id/toolbar"**

**android:layout\_width="match\_parent"**

**android:layout\_height="?attr/actionBarSize"**

**android:background="?attr/colorPrimary"**

**app:popupTheme="@style/AppTheme.PopupOverlay"/>**

**</android.support.design.widget.AppBarLayout>**

<android.support.v4.widget.DrawerLayout

android:id="@+id/drawer\_layout"

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"

android:fitsSystemWindows="true"

tools:openDrawer="start">

<include

layout="@layout/app\_bar\_main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

<android.support.design.widget.NavigationView

android:id="@+id/nav\_view"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

android:fitsSystemWindows="true"

app:headerLayout="@layout/nav\_header\_main"

app:menu="@menu/activity\_main\_drawer"/>

</android.support.v4.widget.DrawerLayout>

</LinearLayout>

Смотрим на значок.

Сама анимация значка зависит от переменной **toggle** (объект класса **ActionBarDrawerToggle**). Если вы её уберёте, то никакого значка в заголовке приложения не будет.

Можно поменять цвет значка гамбургера. Откроем файл стилей **res/values/styles.xml** и добавим:

<!-- Base application theme. -->

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<item name="drawerArrowStyle">@style/DrawerArrowStyle</item>

</style>

<style name="DrawerArrowStyle" parent="Widget.AppCompat.DrawerArrowToggle">

<item name="spinBars">true</item>

<item name="color">@android:color/white</item>

</style>

Элемент **spinBars** со значением **true** позволяет использовать анимацию. В противном случае значок будет статичным.

Навигация

В шаблоне присутствует метод **onNavigationItemSelected()** с аннотацией**@SuppressWarnings("StatementWithEmptyBody")** (Оператор с пустым телом). Нам нужно добавить свой код для навигации, который должен реагировать на нажатия в меню шторки. Нам понадобятся фрагменты. Для примера создадим первый фрагмент.

package ru.alexanderklimov.navigationdrawerdemo;

import android.os.Bundle;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

public class FirstFragment extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment\_first, container, false);

}

}

Разметка для первой активности.

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@android:color/holo\_green\_dark">

<TextView

android:id="@+id/textView3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:text="First Fragment"

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

</RelativeLayout>

По такому же образу создайте второй фрагмент **SecondFragment** и т.д.

Определим **RelativeLayout** в файле **content\_main.xml** в качестве контейнера.

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

<RelativeLayout

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:id="@+id/container"**

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"

tools:context="ru.alexanderklimov.navigationdrawerdemo.MainActivity"

tools:showIn="@layout/app\_bar\_main">

<!--<TextView-->

<!--android:layout\_width="wrap\_content"-->

<!--android:layout\_height="wrap\_content"-->

<!--android:text="Hello World!"/>-->

</RelativeLayout>

Теперь можем написать недостающий код для навигации по фрагментам в **MainActivity**.

@SuppressWarnings("StatementWithEmptyBody")

@Override

public boolean onNavigationItemSelected(MenuItem item) {

// Создадим новый фрагмент

Fragment fragment = null;

Class fragmentClass = null;

// Handle navigation view item clicks here.

int id = item.getItemId();

if (id == R.id.nav\_camera) {

// Handle the camera action

fragmentClass = FirstFragment.class;

} else if (id == R.id.nav\_gallery) {

fragmentClass = SecondFragment.class;

} else if (id == R.id.nav\_slideshow) {

} else if (id == R.id.nav\_manage) {

} else if (id == R.id.nav\_share) {

} else if (id == R.id.nav\_send) {

}

try {

fragment = (Fragment) fragmentClass.newInstance();

} catch (Exception e) {

e.printStackTrace();

}

// Вставляем фрагмент, заменяя текущий фрагмент

FragmentManager fragmentManager = getSupportFragmentManager();

fragmentManager.beginTransaction().replace(R.id.container, fragment).commit();

// Выделяем выбранный пункт меню в шторке

item.setChecked(true);

// Выводим выбранный пункт в заголовке

setTitle(item.getTitle());

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);

drawer.closeDrawer(GravityCompat.START);

return true;

}

Добавление собственных компонентов в меню шторки

В версии 23.1.0 появилась возможность добавлять дополнительные компоненты в меню шторки. Создадим новый файл разметки **nav\_header\_switch.xml**.

<?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="horizontal">

<android.support.v7.widget.SwitchCompat

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:text="Сытые коты"/>

</LinearLayout>

Связываем разметку с новым элементом меню.

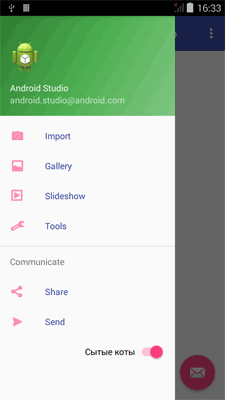
<item

android:id="@+id/nav\_switch"

app:actionLayout="@layout/nav\_header\_switch"

android:title=""/>

Атрибут **android:title** я оставил пустым, так как текст уже задан в разметке. Вы можете задать свой текст здесь.



Добавляем счётчик в меню шторки

Откройте файл **res/menu/activity\_main\_drawer.xml** и добавьте атрибут **app:actionViewClass="android.widget.TextView"** ко второму и третьему элементу меню из шаблона. Теперь эти элементы будут связаны с текстовыми метками.

<item

android:id="@+id/nav\_gallery"

android:icon="@drawable/ic\_menu\_gallery"

**app:actionViewClass="android.widget.TextView"**

android:title="Gallery"/>

<item

android:id="@+id/nav\_slideshow"

android:icon="@drawable/ic\_menu\_slideshow"

**app:actionViewClass="android.widget.TextView"**

android:title="Slideshow"/>

Объявим текстовые метки и инициализируем их в методе **onCreate()**. В отдельном методе будем управлять их свойствами.

private TextView mSlideshowTextView;

private TextView mGalleryTextView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

...

mGalleryTextView = (TextView) MenuItemCompat.getActionView(navigationView.getMenu().

findItem(R.id.nav\_gallery));

mSlideshowTextView = (TextView) MenuItemCompat.getActionView(navigationView.getMenu().

findItem(R.id.nav\_slideshow));

// метод для счетчиков

initializeCountDrawer();

}

private void initializeCountDrawer() {

mGalleryTextView.setGravity(Gravity.CENTER\_VERTICAL);

mGalleryTextView.setTypeface(null, Typeface.BOLD);

mGalleryTextView.setTextColor(getResources().getColor(R.color.colorAccent));

mGalleryTextView.setText("99+");

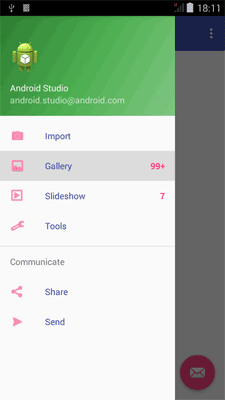
mSlideshowTextView.setGravity(Gravity.CENTER\_VERTICAL);

mSlideshowTextView.setTypeface(null, Typeface.BOLD);

mSlideshowTextView.setTextColor(getResources().getColor(R.color.colorAccent));

mSlideshowTextView.setText("7");

}



Вы можете переделать метод под себя, чтобы динамически изменять показания счётчика.