**Android 5.0新特性**

**1. 了解Material Design**

* 概念：融合卡片式，立体式的设计风格,强调层次感，动画，阴影等元素
* 国内翻译介绍：查看 [http://design.1sters.com](http://design.1sters.com/)
* 官网介绍：<http://developer.android.com/training/material>
* 演示Android5.0 Demo
* Android UI样式风格发展：2.3版本(黄色丑陋版)->4.0(Holo)->5.0(MaterialDesign)

**2. 动态替换Theme**

* MaterialTheme配色方案：[http://www.materialpalette.com](http://www.materialpalette.com/)
* 修改状态栏，ActionBar，界面背景，NavigationBar的颜色。让Activity使用自定义的Theme。
* <style name="AppTheme" parent="@android:style/Theme.Material">
* <!--状态栏颜色-->
* <item name="android:colorPrimaryDark">#f00</item>
* <!--ActionBar颜色-->
* <item name="android:colorPrimary">#ff0</item>
* <!--界面背景颜色-->
* <item name="android:windowBackground">@color/colorWindowBackground</item>
* <!--导航栏颜色-->
* <item name="android:navigationBarColor">#00f</item>
* </style>
* 动态替换Theme的步骤：
  + 定义至少2套theme
  + 调用setTheme方法设置当前的theme，但是该方法要在setContentView之前，如:
  + setTheme(mTheme);
  + setContentView(R.layout.activity\_main);
  + 设置了Theme，需要finish当前Activity，然后重启当前Activity，让Theme生效
  + Intent intent = getActivity().getIntent();
  + getActivity().finish();//结束当前的Activity
  + getActivity().overridePendingTransition(0,0);//不要动画

startActivity(intent);

**3. View的高度与阴影**

* 官网介绍：<https://developer.android.com/intl/zh-tw/training/material/shadows-clipping.html>
* View新增属性z轴，用来体现Material Design中的层次，影响因素2个：elevation 和 translationZ

View高度 = elevation + translationZ

* + elevation表示view的高度，高度越大，阴影越大，可以在xml中直接使用属性， 也可以在代码中使用view.setEvelvation();

android:elevation="10dp"

* + transtionZ属性表示view在Z方向移动的距离，一般用于属性动画中

android:translationZ="10dp"

* 高度影响View的绘制顺序，过去是按View添加顺序绘制，先添加的先绘制，现在高度小的先绘制，因为高度小的，层级低，在下面， 高度相同的，按添加顺序绘制
* 注意：
  + 如果View的背景色为透明，则不会显示出阴影效果
  + 只有子View的大小比父View小时，阴影才能显示出来

**4. View的轮廓与裁剪(在Android5.1以及以上才有效果)**

* 官网介绍：<https://developer.android.com/intl/zh-tw/training/material/shadows-clipping.html>
* View增加了轮廓概念,轮廓用来表示怎么显示阴影,也就是说轮廓什么形状，阴影就显示什么形状。
  + View的轮廓可以通过outlineProvider属性设置，默认是依据于background的，还有其他3个取值：bounds,none,paddingBounds

android:outlineProvider="bounds"

none：即使设置了evaluation也不显示阴影

background：按背景来显示轮廓，如果background是颜色值，则轮廓就是view的大小，如果是shape，则按shape指定的形状作为轮廓

bounds: View的矩形大小作轮廓

paddedBounds: View的矩形大小减去padding的值后的大小作轮廓。

* + 可以通过setOutlineProvider()方法自定义轮廓:
  + tv\_blue.setOutlineProvider(new ViewOutlineProvider() {
  + @TargetApi(Build.VERSION\_CODES.LOLLIPOP)
  + @Override
  + public void getOutline(View view, Outline outline) {
  + outline.setOval(0,0,
  + view.getWidth(),view.getHeight());
  + }

});

* + 注意：如果background是图片，那只能通过代码setOutlineProvider()来指定轮廓
* View的裁剪是指将View按照轮廓裁剪，能改变View的形状，如圆形头像：
  + 先设置轮廓:
  + 再设置根据轮廓裁剪View,目前只支持对矩形，圆形，圆角矩形的裁剪:
  + //设置对View进行裁剪

tv\_clip.setClipToOutline(true);

**5. Palette的使用**

* 使用Palette可以让我们从一张图片中拾取颜色，将拾取到的颜色赋予ActionBar，StatusBar以及背景色可以让界面色调实现统一
* 使用Palette需要添加以下依赖：

compile 'com.android.support:palette-v7:23.0.0+'

* Palette提供的API
  + 传入Bitmap即可获取Palette对象,以下是同步和异步使用方式:
  + //同步获取，需要在子线程中使用
  + Palette palette = Palette.from(drawable.getBitmap()).generate();
  + //异步获取，可以在主线程中使用
  + Palette.from(drawable.getBitmap()).generate(new Palette.PaletteAsyncListener() {
  + @Override
  + public void onGenerated(Palette palette) {
  + //...
  + }

});

* + 得到Palette对象后，获取其中的颜色，颜色对应如下：
  + vibrant - 有活力的颜色
  + lightVibrant - 有活力的亮色
  + darkVibrant - 有活力的暗色
  + muted - 柔和暗淡的颜色
  + lightMuted - 柔和的亮色

darkMuted - 柔和的暗色

* + 获取指定颜色的采样对象，获取采样得到的颜色:
  + //我们可以直接使用palette获取指定颜色：
  + palette.getLightMutedColor(defaultColor);
  + //一般也可以先获取采样对象Swatch,从Swatch中获取我们需要的颜色:
  + //获取有活力颜色的采样对象

Palette.Swatch vibrantSwatch = palette.getVibrantSwatch();

* + 采样对象Swatch提供了以下方法来获取颜色：
  + //swatch.getPopulation(): the amount of pixels which this swatch represents.
  + //swatch.getRgb(): the RGB value of this color.
  + //swatch.getHsl(): the HSL value of this color，即色相，饱和度，明度.
  + //swatch.getBodyTextColor(): the RGB value of a text color which can be displayed on top of this color.
  + //swatch.getTitleTextColor(): the RGB value of a text color which can be displayed on top of this color
  + //一般会将getRgb设置给控件背景色，getBodyTextColor()设置给文字颜色
  + textView.setBackgroundColor(vibrantSwatch.getRgb());

textView.setTextColor(vibrantSwatch.getBodyTextColor());

**6. 水波纹动画，自定义水波纹动画以及状态选择器动画**

* 首先，在Android5.0以上，点击效果默认自带水波纹效果，并且有2种选择：
* //矩形边框水波纹
* android:background="?android:attr/selectableItemBackground"
* //无边框限制水波纹

android:background="?android:attr/selectableItemBackgroundBorderless"

* 自定义水波纹动画
  + 使用ViewAnimationUtils创建圆形水波纹动画，注意该动画不能在Activity的onCreate方法中执行：
  + Animator circularReveal = ViewAnimationUtils.createCircularReveal(text, 0, text.getHeight() , 1f, text.getWidth()\*2);
  + circularReveal.setDuration(1000);

circularReveal.start();

* + 使用ripple标签或者RippleDrawable可以更改控件水波纹动画颜色：
  + <ripple xmlns:android="http://schemas.android.com/apk/res/android" android:color="#00ff00">
  + <item android:id="@android:id/mask" ><color android:color="#0000ff" />
* 定义带有属性动画的状态选择器
  + 通过stateListAnimator属性指定状态选择器的动画：

android:stateListAnimator="@drawable/selector\_anim"

* + 状态选择器文件中需要加入objectAnimator标签：
  + <selector xmlns:android = "http://schemas.android.com/apk/res/android" >
  + <item android:statepressed = "true" >
  + <objectAnimator android:propertyName = "scaleX"
  + android:duration = "@android:integer/configshortAnimTime"
  + android:valueTo = "0.2"
  + android:valueFrom = "1"
  + android:valueType = "floatType" >
  + //...
  + 同样，状态选择器动画可以用代码方式加载
  + //加载动画
  + AnimatorInflater.loadStateListAnimator();
  + //设置动画
  + View.setStateListAnimator();
* 定义带有帧动画的状态选择器，需要设置给background属性，不是stateListAnimator，如下所示:
* <animated-selector xmlns:android = "http://schemas.android.com/apk/res/android" >
* <item android:id = "@+id/pressed" android:drawable = "@drawable/drawableP"
* android:state\_pressed = "true" />
* <item android:id = "@id/default"
* android:drawable = "@drawable/drawableD" />
* <!-- 指定帧动画 - ->
* <transition android:fromId = "@+id/default" android:toId = "@+id/pressed" >
* <animation-list>
* <item android:duration = "15" android:drawable = "@drawable/dt1 " />
* <item android:duration = "15" android:drawable = "@drawable/dt2" />
* ...
* </animation-list>

</animated-selector>

**7. CardView的使用**

* CardLayout拥有高度和阴影，以及轮廓裁剪，圆角等功能
* 各属性说明:
* 1.设置圆角：card\_view:cardCornerRadius="10dp"
* 2.设置高度：card\_view:cardElevation="10dp"
* 3.设置内边距：card\_view:contentPadding="10dp"

4.设置背景色：card\_view:cardBackgroundColor="?android:attr/colorPrimary"

**8. RecyclerView的使用**

* <https://developer.android.com/intl/zh-tw/training/material/lists-cards.html>
* 先添加依赖 compile 'com.android.support:recyclerview-v7:23.1.1'
* 设置LayoutManager：控制RecyclerView如何显示布局,系统提供3个布局管理器：
  + LinearLayoutManager：线性布局，有横向和竖直方向显示
  + GridLayoutManager：网格布局，有横向和竖直方向显示
  + StaggeredGridLayoutManager： 瀑布流布局，有横向和竖直方向显示
* 然后给RecyclerView设置Adapter<RecyclerView.ViewHolder>
* 设置点击事件，由于RecyclerView没有setOnItemClickListener，只能在Adapter中给View设置Click事件

**9. ToolBar的使用**

* 它用来代替ActionBar，但是比ActionBar更加灵活，相当于可以写在布局文件中的ActionBar；与DrawerLayout的使用的时候，DrawerLayout可以覆盖在ToolBar上，并且ToolBar和ActionBar不能同时使用
* 使用ToolBar的步骤：
  + 先隐藏ActionBar，可以继承一个不带ActionBar的Theme，如:

style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar"

* + 然后在Activity中设置ToolBar替代ActionBar：

setSupportActionBar(toolBar);

* + 最后设置ToolBar的显示内容：
  + toolBar.setTitle("ToolBar");//设置标题
  + toolBar.setNavigationIcon(iconRes);//设置图标

toolBar.setOnMenuItemClickListener();//设置Menu Item点击

**10. Android 5.0新特性的向下兼容**

* 可以通过Support Library使用的新特性可以向下兼容，如：
  + RecyclerView (recyclerview-v7)
  + CardView (cardview-v7)
  + Palette颜色识别 (palette-v7)
  + ToolBar (appcompat-v7)
  + SwipeRefreshLayout (v4)
* 定义针对版本的资源目录
  + layout：使用新API的布局放在res/layout-v21中，其他的放res/layout
  + drawable：使用新属性的drawable文件放到 res/drawable-v21，其他的放 res/drawable
  + values: 新的主题放到 res/values-v21, 其他的放 res/values
* 在代码中对系统Version做判断，使用对应的效果，如:
* if(Build.VERSION.SDK\_INT> 21){
* //使用新动画
* ...

}