自定义控件分类：

一、组合自定义控件（组合系统控件） 继承布局

有特殊需求的组合控件可以继承View或者是布局，然后自定义属性

二、部分自定义控件（增强已有控件功能） 继承某个UI 如:TextView跑马灯

三、完全自定义控件 继承View 或者 ViewGroup

四、使用图片结合动画组合控件

五、使用图片（或系统控件）组合继承View或者是布局，自定义属性自定义控件定义attrs.xml文件写规则

继承ViewGroup与继承ViewGroup实现类容器的在onMeasure方法中不同之处

继承ViewGroup要主动去测量子布局的宽高

```
for (int i = 0; i < getChildCount(); i++) {
        measureChild(getChildAt(i), widthMeasureSpec, heightMeasureSpec);
    }
```

继承ViewGroup实现类容器的不同之处（如继承：RelativeLayout，LinearLayout,FrameLayout）

可以直接获取子布局的宽高

```
// 获取尺寸大小
//      mWidth = getMeasuredWidth();
//      mHeight = getMeasuredHeight();
//      menuWidth = menuView.getMeasuredWidth();
```

引入图片设置图片大小适配不用屏幕手机的的比例

图片的像素宽 高 除以 图片的倍数 得到相应的图片宽高dp值。

mdpi:1 --- 1px （1倍图） hdpi:1.5 --- 1.5px （1.5倍图）

xhdpi:2 --- 2px （2倍图） xxhdpi:2.5 ---2.5 px （2.5倍图）

图片大小与不同手机屏幕的适配方法

图片的像素宽除以 N倍图比例 ， 高除以 N倍图比例 得到相应的 宽高dp值

X轴轮播广告系统控件

android.support.v4.view.ViewPager

类似于ListView的用法

用法：

1、布局定义控件

2、代码找到控件

3、设置数据适配器setAdapter

4、自定义类继承extends PagerAdapter

5、重写4个方法：

getCount()　　获取一共有多少条目

isViewFromObject(View view, Object object)

instantiateItem(ViewGroup container, int position)　　设置添加数据到控件中

destroyItem(ViewGroup container, int position, Object object)

6、设置页改变监听器addOnPageChangeListener(new ViewPager.OnPageChangeListener()

7、设置点击事件　　setOnTouchListener(new View.OnTouchListener()

ViewPager---PagerAdapter

核心方法：

将图片资源转变为图片

```
ImageView imageView = new ImageView(this);
imageView.setBackgroundResource(IMG_IDS[i]);
```

将某UI组件设置到布局中

```
ImageView dot=new ImageView(this)
dot.setBackgroundResource(R.drawable.dot_selector);
```

---px ---dp 参数接受PX值

```
LinearLayout.LayoutParams params=new
LinearLayout.LayoutParams(dp2px(8),dp2px(8));
params.leftMargin=dp2px(8)
布局ID.addView(dot,params);
```

设置集合中第一个条目在Item中的位置

```
setCurrentItem(index);
```

继承View自定义控件

```java
//设置控件自己的大小
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    super.onMeasure(widthMeasureSpec, heightMeasureSpec);
    setMeasuredDimension(mBJimg.getWidth(), mBJimg.getHeight());
}

//设置自己的样子
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    //画一张图片作为背景开关图
    canvas.drawBitmap(mBJimg, 0, 0, new Paint());
    //画一张图片作为滑动开关图
    canvas.drawBitmap(mHDTPS, mLeft, 0, new Paint());
}
```

```java
@Override  //事件分发机制   让开关可以滑动的核心方法
public boolean onTouchEvent(MotionEvent event) {
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:      //按下
            mDownx = event.getX();
            break;
        case MotionEvent.ACTION_MOVE:
            float movex = event.getX();
            float v = (int) (movex - mDownx + 0.5f);
            mLeft += v;
            if (mLeft < 0) {
                mLeft = 0;
            } else if (mLeft > mMaxpy) {
                mLeft = mMaxpy;
            }
            invalidate(); //核心方法    onDraw,要在UI线程里面进行调用，假如要在非UI线程里面，调用postInvalidate();
            mDownx = movex;
            break;
        case MotionEvent.ACTION_UP:      //滑动结束后处理滑动到一半的情况
            if (mLeft > mMaxpy / 2) {
                mLeft = mMaxpy;
                if(mOnSwitchListener != null && mIscheck != mLeft > mMaxpy / 2){
```

```java
public zhidingyiview(Context context, AttributeSet attrs, int defStyleAttr) {
    super(context, attrs, defStyleAttr);
    TypedArray ta = context.obtainStyledAttributes(attrs, R.styleable.zhidingyiview);
    //获取背景图片
    int BJID = ta.getResourceId(R.styleable.zhidingyiview_con_bj, -1);
    mBJimg = BitmapFactory.decodeResource(getResources(), BJID);
    //获取滑动图片
    int HDPJ = ta.getResourceId(R.styleable.zhidingyiview_con_tp, -1);
    mHDTPS = BitmapFactory.decodeResource(getResources(), HDPJ);
    //获取状态开关
    mIscheck = ta.getBoolean(R.styleable.zhidingyiview_con_chenk, false);
    //最大移动的位置
    mMaxpy = mBJimg.getWidth() - mHDTPS.getWidth();

    ta.recycle();
    setCheck();
}
```

完全自定义控件继承View 或者继承 ViewGroup

绘制流程

1、测量 控件大小并设置 measure 重写onMeasure(int widthMeasureSpec, int heightMeasureSpec)方法

2、布局（设置控件的摆放位置）layout

3、画（画控件的样子）draw 重写 onDraw(Canvas canvas)方法

View 还是 ViewGroup

View

measure（onMeasure）----draw（onDraw）

ViewGroup

measure（onMeasure）（设置自己大小，测量）-----layout(onLayout给控件布局)

4、设置与用户的交互 重写 onTouchEvent(MotionEvent ev)方法

5、交互设置接口，定义要实现的方法

继承ViewGoup自定义容器控件

```
@Override    //指定布局的位置
protected void onLayout(boolean changed, int l, int t, int r, int b) {
    //设置左边View的位置
    /**
     * 距离左边的位置,相对于父母，距离上边的位置,相对于父母,
     * 距离右边的位置,相对于父母，距离下边的位置,相对于父母,
     */
    mViewleft.layout(-mLeftwidth, 0, 0, mSlidingMenuHeight);
    //设置主内容VIEW的位置
    mContentView.layout(0, 0, mSlidingMenuWidth, mSlidingMenuHeight);
}

//onFinishInflate解析布局完成之后会调用一个方法
@Override
protected void onFinishInflate() {
    super.onFinishInflate();
    //获取左边的布局
    mViewleft = getChildAt(0);
    LayoutParams params = mViewleft.getLayoutParams();
    mLeftwidth = params.width;
    //获取内容的布局
    mContentView = getChildAt(1);
}
```

```java
//测量并设置大小
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    //给左边的布局测量
    int widthMespecs = MeasureSpec.makeMeasureSpec(mLeftwidth, MeasureSpec.EXACTLY);
    //设置左边的宽高
    mViewleft.measure(widthMespecs, heightMeasureSpec);
    //设置右边的宽高
    mContentView.measure(widthMeasureSpec, heightMeasureSpec);

    //设置自己的大小
    mSlidingMenuWidth = MeasureSpec.getSize(widthMeasureSpec);
    mSlidingMenuHeight = MeasureSpec.getSize(heightMeasureSpec);
    //将自己的大小设置到实际测量的尺寸里
    setMeasuredDimension(mSlidingMenuWidth, mSlidingMenuHeight);
}
```

给子布局测量时需要一个模式 makeMeasureSpec方法：基于提供的尺寸和测量规范模式

模式：

UNSPECIFIED：系统用，或者测量时候，不知道父亲实现

EXACTLY：自己知道确切的大小尺寸

AT_MOST:wrap_content

MeasureSpec.makeMeasureSpec(mLeftwidth, MeasureSpec.EXACTLY);

返回值是 int值 前2位为模式 后30位为子控件的大小

1、先获取布局中的子控件 onFinishInflate() 解析布局完成之后会调用一个方法可以获取到布局中的view

2、测量设置子控件的大小,设置自己的大小 onMeasure(int widthMeasureSpec, int heightMeasureSpec)

3、决定子控件的摆放位置 onLayout(boolean changed, int l, int t, int r, int b)

4、点击事件onTouchEvent(MotionEvent event)

5、交互设置接口，定义要实现的方法

继承UI的自定义控件测量加入的某布局的方法 PopupWindow()中常用测量来确定UI的大小

```
mHead = View.inflate(getContext(), R.layout.view_head, null);
measure(0, 0);
mHeadCLHeight = mHead.getMeasuredHeight();        测量高度
```

事件分发机制

一个控件点击会不会有效果取决于布局结构及布局结构中的事件分发机制
父布局都有三个方法
dispatchTouchEvent(MotionEvent ev) 是否分发事件 返回：
onInterceptTouchEvent(MotionEvent ev) 是否拦截事件   返回：
onTouchEvent(MotionEvent event)   消费
执行顺序：
鼠标按下：
如果上层分发且不拦截 ————————〉走   下层判断是否分发 或拦截   如果不分发或分发拦截 ——>走自己消费——走上层消费

鼠标抬起：
如果上层分发且不拦截 ————————〉走   下层判断是否分发 或拦截   如果不分发或分发拦截 ——>走自己消费——走上层消费
点击事件结束   点击事件执行失败

如果每一层都发分不拦截最后走Button的消费并走onclick方法   点击事件触发成功

最后一层UI 没有拦截事件
dispatchTouchEvent(MotionEvent ev)
onTouchEvent(MotionEvent event)
onclick(View v)

屏幕滑动事件

在onTouchEvent(MotionEvent event)方法中

如果down按下事件返回true,那么后面的MOVE事件就无法响应，滑动事件处理失败

自定义控件的思想

onMeasure 埋设 测量（任意尺寸，确切的尺寸，最大尺寸）

onDraw 做 绘制