为什么要自定义控件

1、特定的显示风格

2、处理特有的用户交互

3、优化我们的布局

4、封装等...

如何自定义控件

1、自定义属性的声明与获取

2、测量onMeasure

3、绘制onDraw

4、状态的存储与恢复

自定义属性声明与获取

1、分析需要的自定义属性

2、在res/valus/attrs.xml定义声明

3、在layout xml文件中进行使用

4、在View的构造方法中进行获取

自定义属性声明与获取

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <attr name="icon" format="reference"></attr>
    <attr name="color" format="color"></attr>
    <attr name="text" format="string"></attr>
    <attr name="text_size" format="dimension"></attr>
    <declare-styleable name="ChangeColorIconWithText">
        <attr name="icon"></attr>
        <attr name="color"></attr>
        <attr name="text"></attr>
        <attr name="text_size"></attr>
    </declare-styleable>
</resources>
```

```java
TypedArray a = context.obtainStyledAttributes(attrs,
        R.styleable.ChangeColorIconWithText);
int n = a.getIndexCount();
for (int i = 0; i < n; i++)
{
    int attr = a.getIndex(i);
    switch (attr)
    {
    case R.styleable.ChangeColorIconWithText_icon:
        BitmapDrawable drawable = (BitmapDrawable) a.getDrawable(attr);
        mIconBitmap = drawable.getBitmap();
        break;
    case R.styleable.ChangeColorIconWithText_color:
        mColor = a.getColor(attr, 0xFF45C01A);
        break;
    case R.styleable.ChangeColorIconWithText_text:
        mText = a.getString(attr);
        break;
    case R.styleable.ChangeColorIconWithText_text_size:
        mTextSize = (int) a.getDimension(attr, TypedValue
                .applyDimension(TypedValue.COMPLEX_UNIT_SP, 12,
                        getResources().getDisplayMetrics()));
        break;
    }
}
a.recycle();
```

测量onMeasure

1、EXACTLY , AT_MOST , UNSPECIFIED

2、MeasureSpec

3、setMeasuredDimension

4、requestLayout()

测量onMeasure

```
private int measureHeight(int heightMeasureSpec)
{
    int result = 0;
    int mode = MeasureSpec.getMode(heightMeasureSpec);
    int size = MeasureSpec.getSize(heightMeasureSpec);

    if (mode == MeasureSpec.EXACTLY)
    {
        result = size;
    } else
    {
        result = getNeedHeight() + getPaddingTop()
                + getPaddingBottom();//计算自身需要的高度
        if (mode == MeasureSpec.AT_MOST)
        {
            result = Math.min(result, size);
        }
    }
    return result;
}
```

绘制onDraw

1、绘制内容区域

2、invalidate() ， postInvalidate();

3、Canvas.drawXXX

4、translate、rotatescale、skew

5、save()、restore()

```
@Override
protected synchronized void onDraw(Canvas canvas)
{
    //使用Canvas相关API绘制anything you want ;
}
```

状态的存储与恢复

1、onSaveInstanceState

2、onRestoreInstanceState

```
@Override
protected Parcelable onSaveInstanceState()
{
    Bundle bundle = new Bundle();
    bundle.putParcelable(INSTANCE_STATUS, super.onSaveInstanceState());
    bundle.putFloat(STATUS_ALPHA, mAlpha);
    return bundle;
}

@Override
protected void onRestoreInstanceState(Parcelable state)
{
    if (state instanceof Bundle)
    {
        Bundle bundle = (Bundle) state;
        mAlpha = bundle.getFloat(STATUS_ALPHA);
        super.onRestoreInstanceState(bundle.getParcelable(INSTANCE_STATUS));
        return;
    }
    super.onRestoreInstanceState(state);
}
```

Thank You !