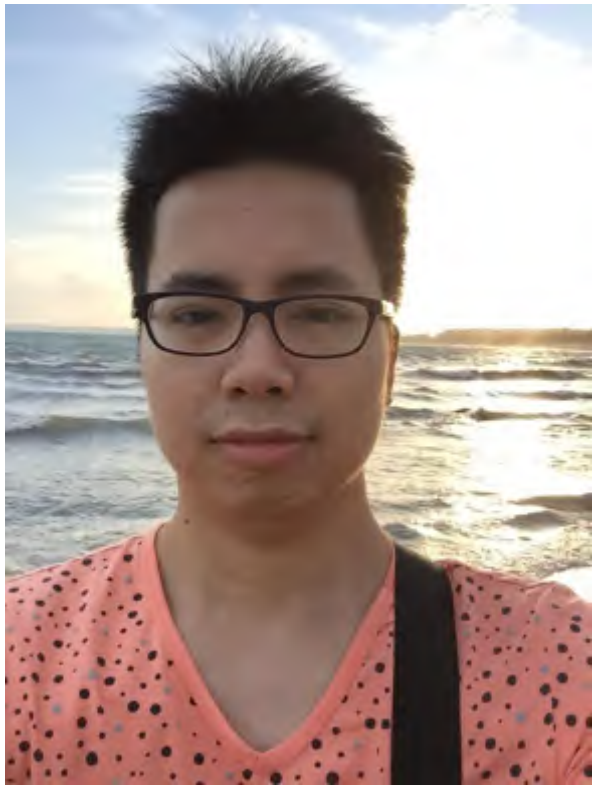
MDCC 2016
中国移动开发者大会
Mobile Developer Conference China 2016

从手Q开发谈Android无障碍化实现原理及优化

alberthe | 何金源
2016年9月



何金源 毕业于华南理工大学

手机QQ

基础Android开发组

目前负责

Android手Q无障碍化

多人聊天

基础资料卡

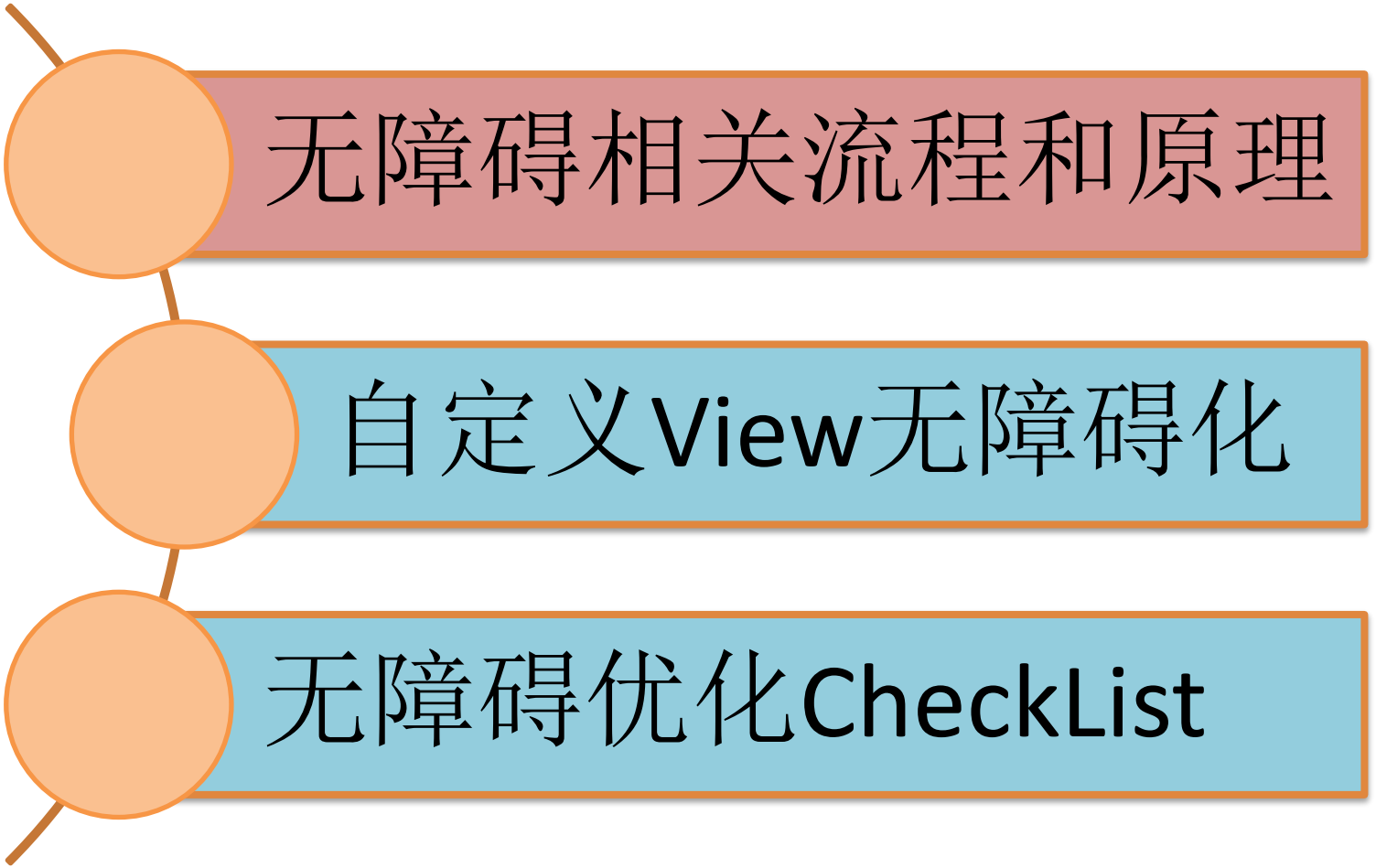

无障碍相关流程和原理
自定义View无障碍化
无障碍优化CheckList

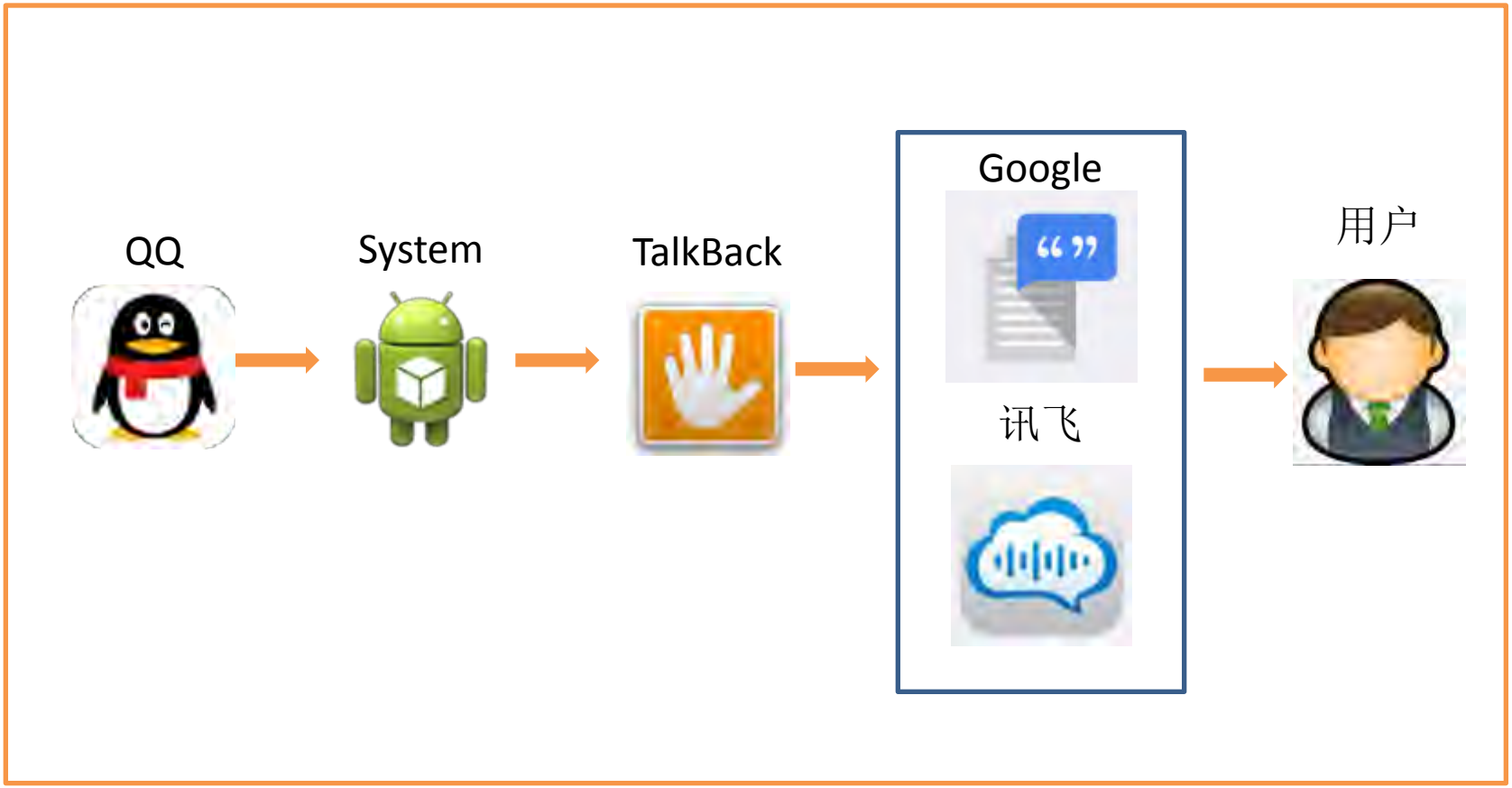
- 无障碍化，是指针对听障、视障、肢障的用户所增加辅助项目，可以方便社会上此类有需要的人士有机会成功使用我们的应用。

- 操作方式：

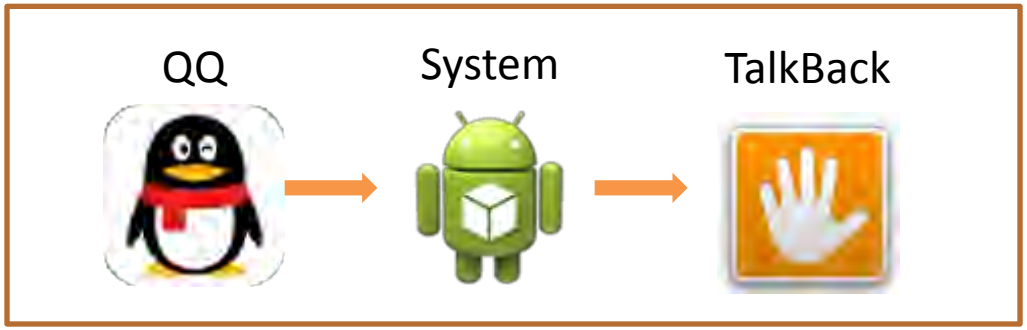
选择（Hover）：单击
开启（Click）：双击
滚动：双指往上、下、左、右
选择上或下一个项目：单指往上、下、左、右
快速回到主画面：单指上滑+左滑
返回键：单指下滑+左滑
最近画面键：单指左滑+上滑
通知栏：单指右滑+下滑

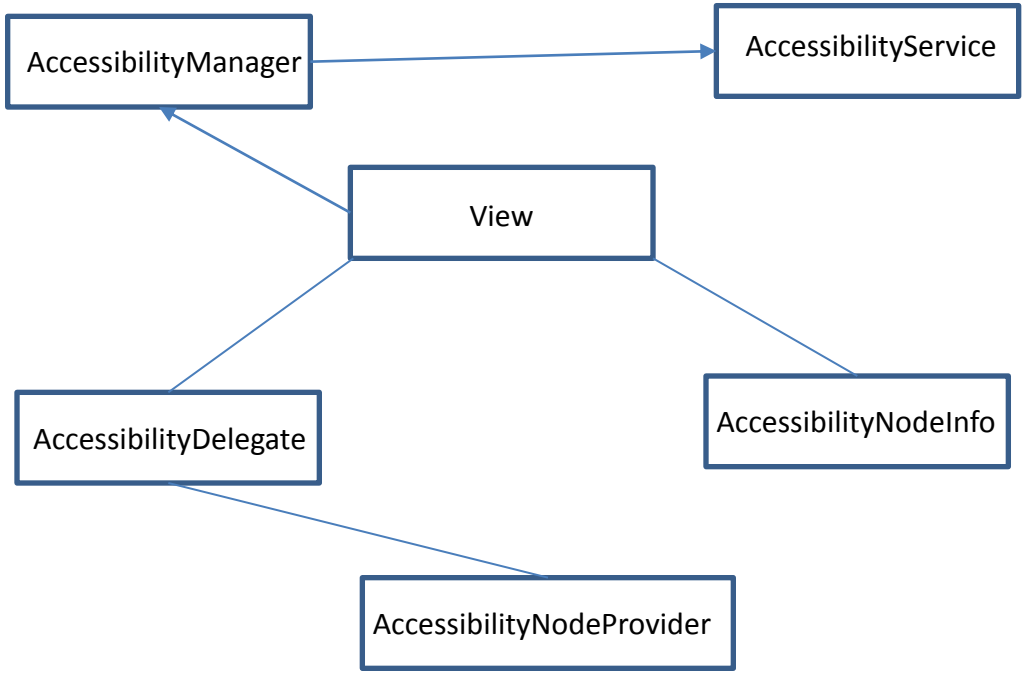
- 可覆盖在任意View上
- 在屏幕上用绿色方框标明
- TalkBack根据用户交互来分配
- 表示当前活跃的元素


无障碍相关流程和原理
自定义View无障碍化
无障碍优化CheckList

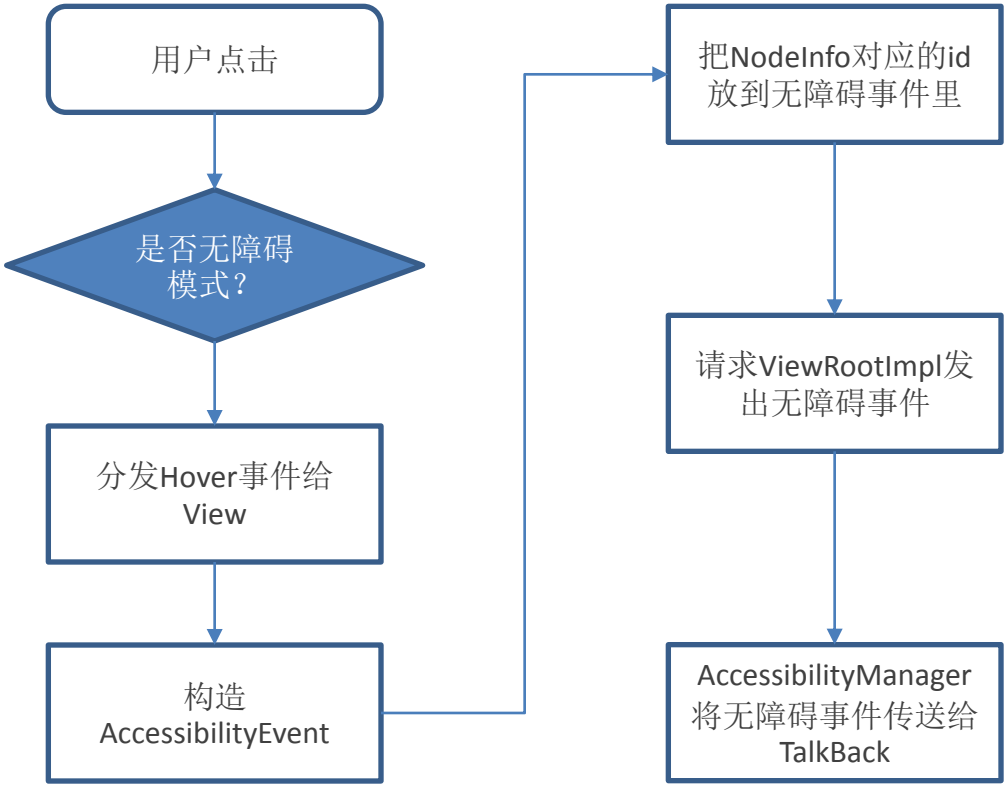

QQ
System
TalkBack
Google
讯飞
用户

无障碍相关类

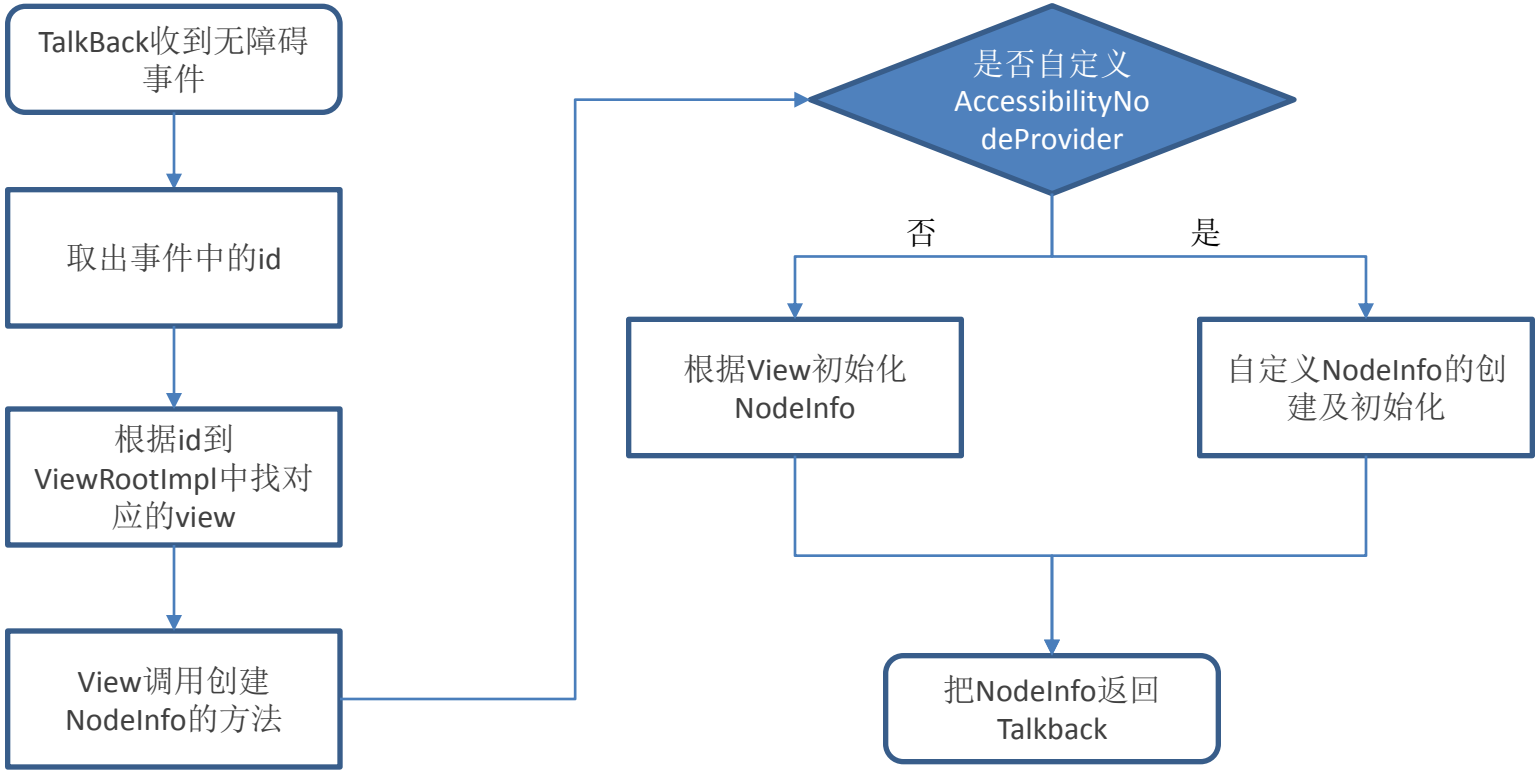

QQ
System
TalkBack

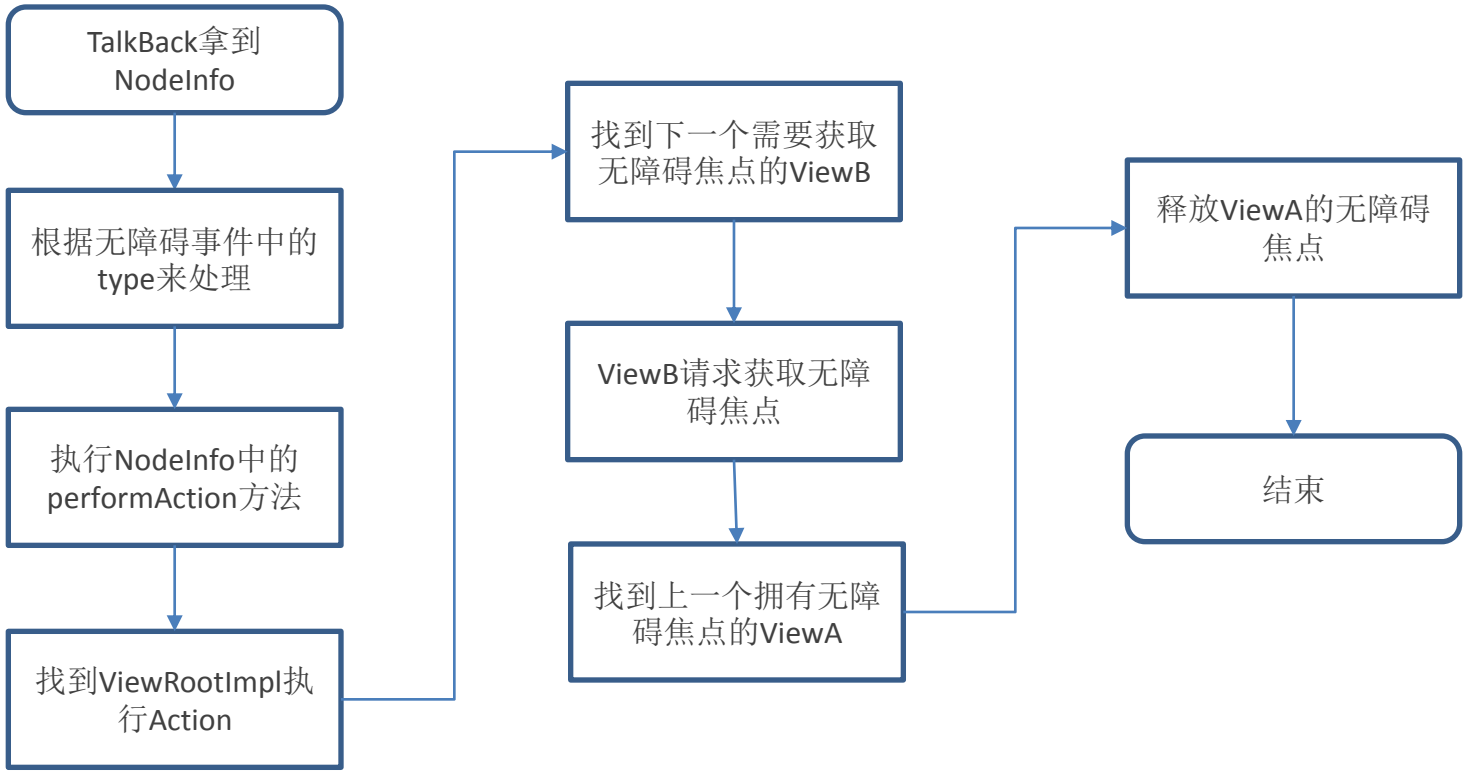

AccessibilityManager
AccessibilityService
View
AccessibilityDelegate
AccessibilityNodeInfo
AccessibilityNodeProvider

发出AccessibilityEvent

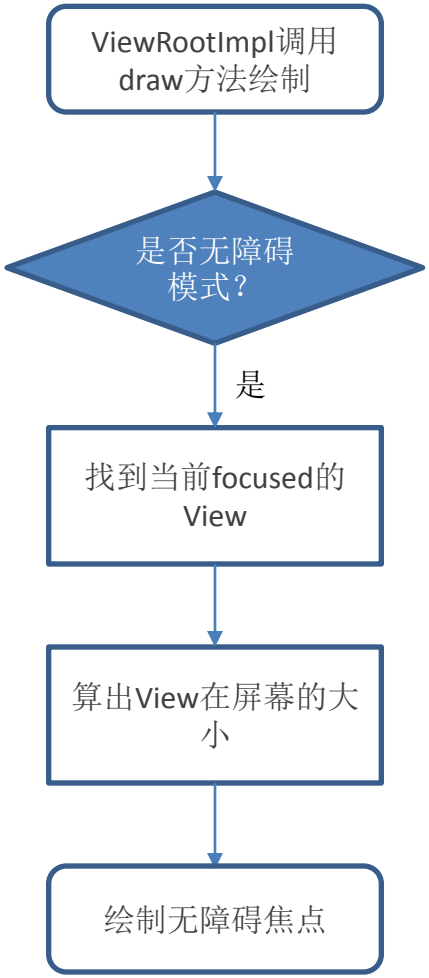


创建AccessibilityNodeInfo

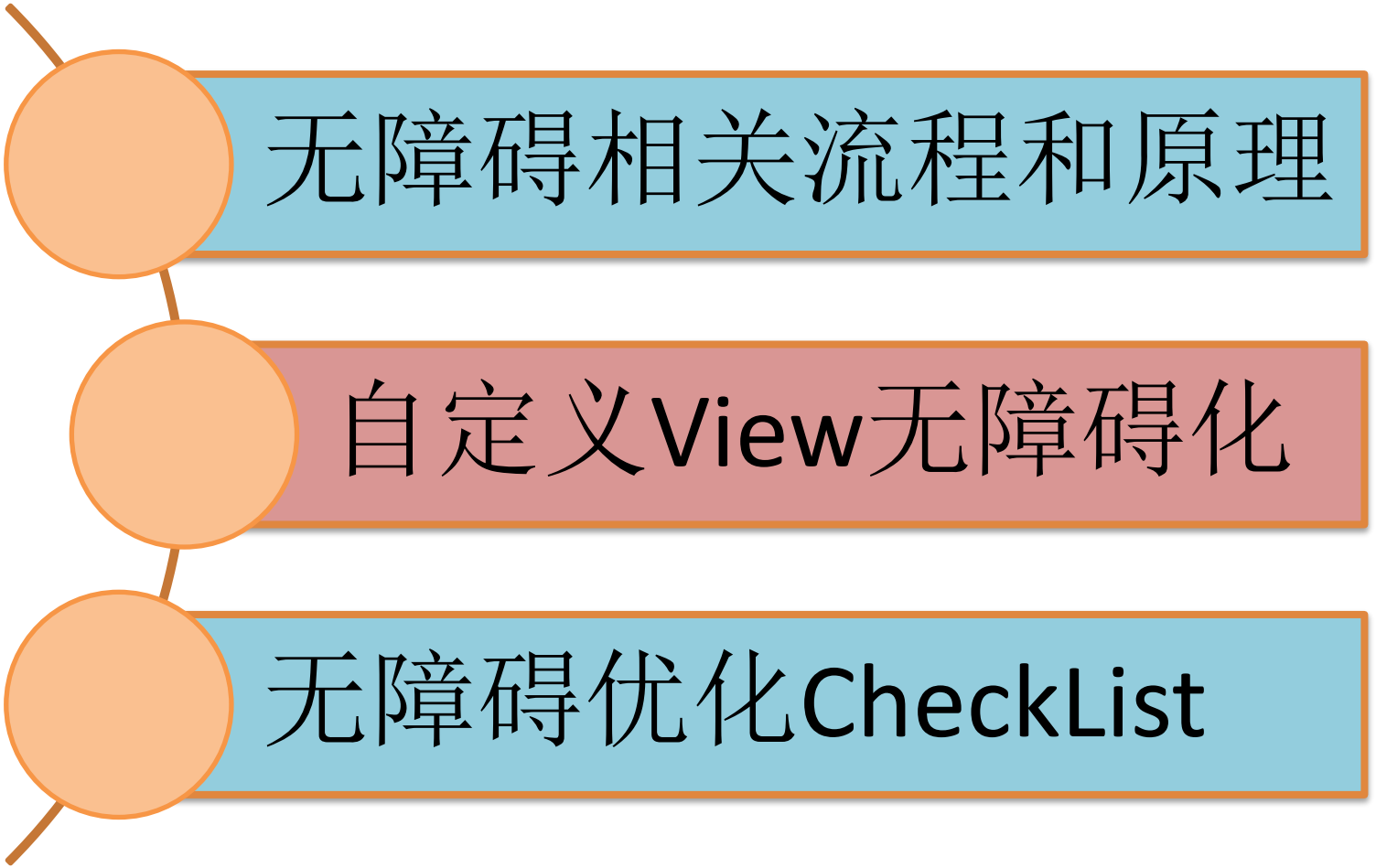



TalkBack拿到NodeInfo
根据无障碍事件中的type来处理
执行NodeInfo中的performAction方法
找到ViewRootImpl执行Action
找到下一个需要获取无障碍焦点的ViewB
ViewB请求获取无障碍焦点
找到上一个拥有无障碍焦点的ViewA
释放ViewA的无障碍焦点
结束

绘制无障碍焦点


ViewRootImpl调用draw方法绘制
是否无障碍模式？
是
找到当前focused的View
算出View在屏幕的大小
绘制无障碍焦点

- UI界面元素发生变化时（比如View被点击，View的焦点切换等），发出AccessibilityEvent
- AccessibilityService接收这些AccessibilityEvent后，根据AccessibiltyEvent里的accessibilityId来获取AccessibilityNodeInfo
- AccessibilityNodeInfo由View来创建，或者由AccessibilityNodeProvider来创建
- AccessibilityService根据AccessibilityNodeInfo的信息提供无障碍服务
- AccessibilityService通过AccessibilityNodeInfo来告知UI元素作出处理


无障碍相关流程和原理
自定义View无障碍化
无障碍优化CheckList

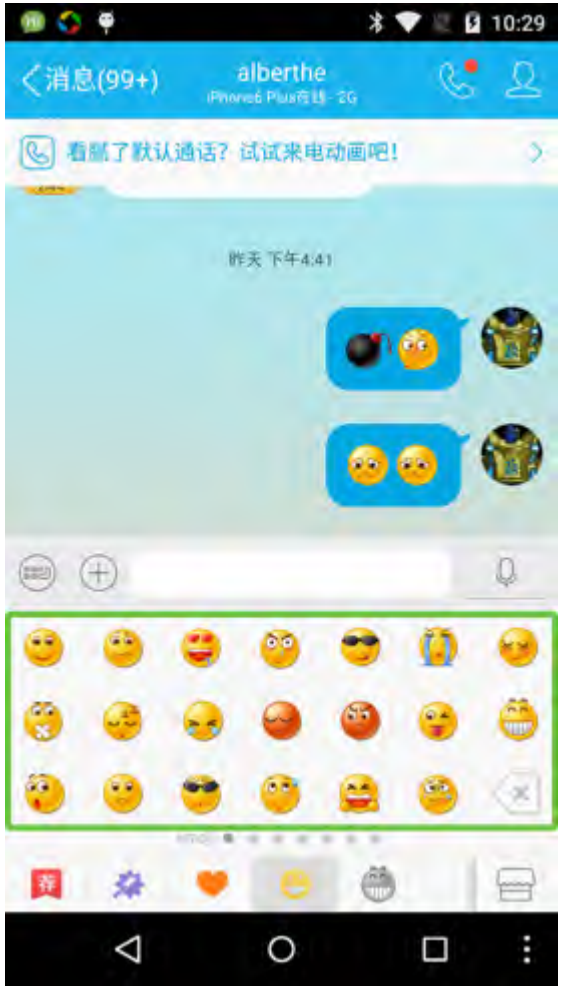
一般情况下的无障碍化

按钮添加
contentdescription

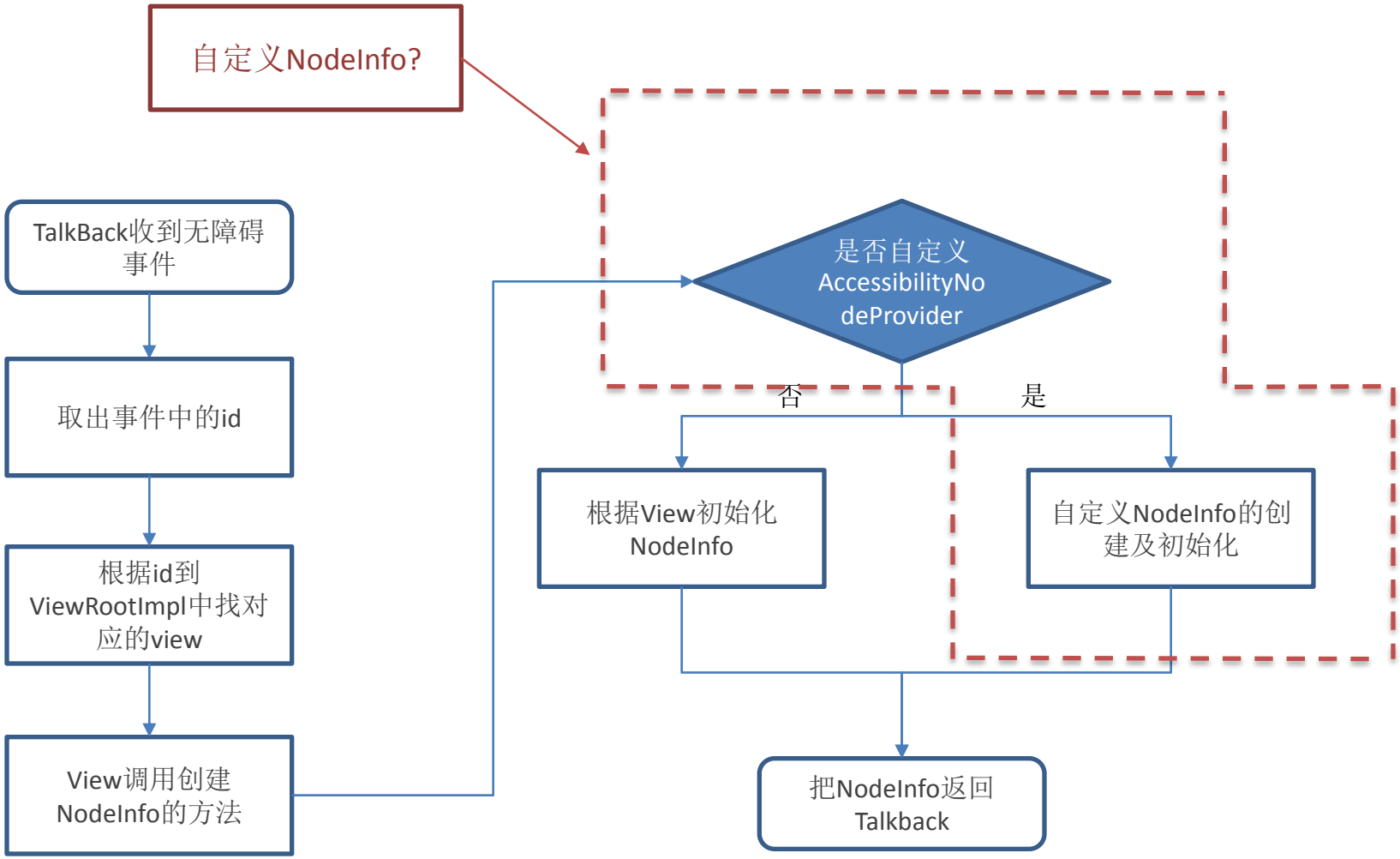
ListView的Item

会变化的元素

困难场景


10:29
消息(99+)
alberthe
看腻了默认通话？试试来电动画吧！
昨天 下午4:41


10:25
消息(99+)
alberthe
看腻了默认通话？试试来电动画吧！
游戏设计之美-Lancelot:抢这么快啊


自定义NodeInfo?
TalkBack收到无障碍事件
取出事件中的id
根据id到ViewRootImpl中找对应的view
View调用创建NodeInfo的方法
是否自定义AccessibilityNodeProvider
否
是
根据View初始化NodeInfo
自定义NodeInfo的创建及初始化
把NodeInfo返回Talkback

- 自定义AccessibilityNodeProvider

```
public class CustomNodeProvider extends AccessibilityNodeProvider {
    @Override
    public AccessibilityNodeInfo createAccessibilityNodeInfo(
            int virtualViewId) {
        // TODO Auto-generated method stub
        return super.createAccessibilityNodeInfo(virtualViewId);
    }

    @Override
    public boolean performAction(int virtualViewId, int action,
            Bundle arguments) {
        // TODO Auto-generated method stub
        return super.performAction(virtualViewId, action, arguments);
    }
}
```

- 创建根AccessibilityNodeInfo

```
@Override
public AccessibilityNodeInfo createAccessibilityNodeInfo(int virtualViewId) {
    AccessibilityNodeInfo info = null;
    if (virtualViewId == View.NO_ID) {
        info = AccessibilityNodeInfo.obtain(CustomView.this);
        onInitializeAccessibilityNodeInfo(info);
        List<VirtualView> children = mChildren;
        final int childCount = children.size();
        for (int i = 0; i < childCount; i++) {
            VirtualView child = children.get(i);
            info.addChild(CustomView.this, child.mId);
        }
    } else {
        return createChildNodeInfo(virtualViewId);
    }
    return info;
}
```

- 创建子AccessibilityNodeInfo

添加Action

设置子节点的边框

设置状态等

```
private AccessibilityNodeInfo createChildNodeInfo(int virtualViewId) {
    AccessibilityNodeInfo info = null;
    VirtualView virtualView = findVirtualViewById(virtualViewId);
    if (virtualView == null) {
        return null;
    }
    info = AccessibilityNodeInfo.obtain();
    info.addAction(AccessibilityNodeInfo.ACTION_CLICK);
    info.addAction(AccessibilityNodeInfo.ACTION_ACCESSIBILITY_FOCUS);
    info.addAction(AccessibilityNodeInfo.ACTION_CLEAR_ACCESSIBILITY_FOCUS);

    info.setPackageName(getContext().getPackageName());
    info.setClassName(virtualView.getClass().getName());
    info.setSource(CustomView.this, virtualViewId);

    Rect mTempRect = new Rect();
    getLocalVisibleRect(mTempRect);

    info.setBoundsInParent(mTempRect);

    int[] mTempGlobalRect = new int[2];
    getLocationOnScreen(mTempGlobalRect);
    final int offsetX = mTempGlobalRect[0];
    final int offsetY = mTempGlobalRect[1];
    Rect mTempScreenRect = new Rect();
    mTempScreenRect.set(mTempRect);
    mTempScreenRect.offset(offsetX, offsetY);
    info.setBoundsInScreen(mTempScreenRect);

    info.setParent(CustomView.this);
    info.setText(virtualView.mText);
    info.setVisibleToUser(true);
    info.setAccessibilityFocused(true);
    info.setFocusable(true);
    info.setClickable(true);
    info.setEnabled(isEnabled());
    info.setContentDescription(virtualView.mText);
    return info;
}
```

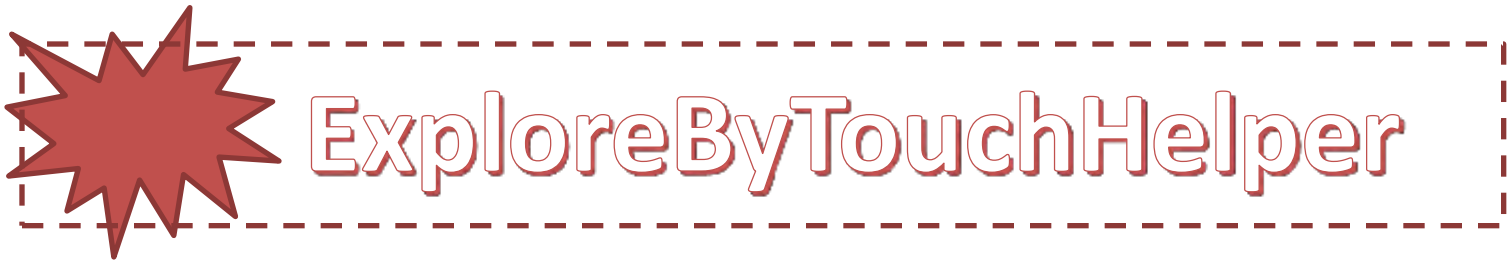
- 实现performAction方法

```
@Override
public boolean performAction(int virtualViewId, int action, Bundle arguments) {
    switch (virtualViewId) {
        case View.NO_ID:
            return performActionForHost(action, arguments);
        default:
            return performActionForChild(virtualViewId, action, arguments);
    }
}
```

创建好根节点和子节点后，AccessibilityNodeProvider会被系统调用performAction，来执行指定的无障碍操作（Action），根据virtualViewId判断由根节点还是子节点来执行。

- 分发HoverEvent到子节点，发出Hover无障碍事件

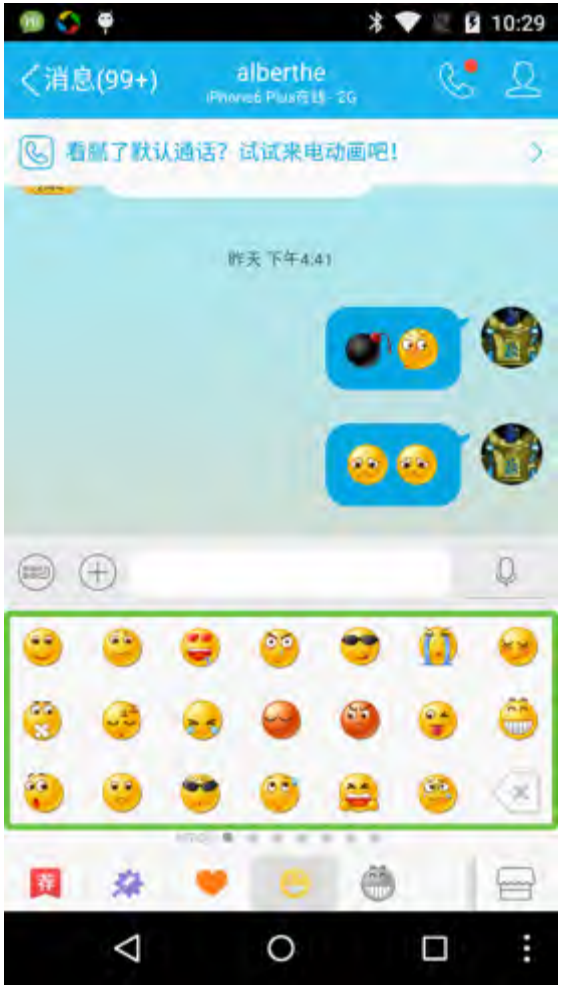
```
@Override
protected boolean dispatchHoverEvent(MotionEvent event) {
    if (!mManager.isEnabled() || !mManager.isTouchExplorationEnabled()) {
        return false;
    }
    switch (event.getAction()) {
        case MotionEvent.ACTION_HOVER_MOVE:
        case MotionEvent.ACTION_HOVER_ENTER:
            final int virtualViewId = getVirtualViewAt(event.getX(), event.getY());
            // 设置当前Hovered的子节点，发出Hover无障碍事件
            updateHoveredVirtualView(virtualViewId);
            return (virtualViewId != INVALID_ID);
        case MotionEvent.ACTION_HOVER_EXIT:
            if (mFocusedVirtualViewId != INVALID_ID) {
                updateHoveredVirtualView(INVALID_ID);
                return true;
            }
            return false;
        default:
            return false;
    }
}
```



~~AccessibilityNodeProvider~~

- 简化虚拟节点层次结构的实现
  只要实现五个抽象方法
- 隐藏AccessibilityNodeProvider的实现
- 完善控制Hover事件、无障碍事件
- 兼容性好

- 委托处理无障碍
- 标记虚拟节点ID
- 丰富无障碍信息
- 提供用户交互支持



委托ExploreByTouchHelper

```
1  private ClassicEmotionTouchHelper mTouchHelper;
2
3  public ClassicEmoticonPanelView(Context context, AttributeSet attrs) {
4      super(context, attrs);
5      ...
6
7      if(AppSetting.enableTalkBack){
8          mTouchHelper = new ClassicEmotionTouchHelper(this);
9          ViewCompat.setAccessibilityDelegate(this, mTouchHelper);
10         ViewCompat.setImportantForAccessibility(this, ViewCompat.IMPORTANT_FOR_ACCESSIBILITY_YES);
11     }
12 }
13
14 @Override
15 protected boolean dispatchHoverEvent(MotionEvent event) {
16     if(AppSetting.enableTalkBack && mTouchHelper.dispatchHoverEvent(event)){
17         return true;
18     }
19     return super.dispatchHoverEvent(event);
20 }
```

- ~~委托处理无障碍~~
- 标记虚拟节点ID
- 丰富无障碍信息
- 提供用户交互支持

- 界面上的元素使用无障碍节点id标记
- 无障碍节点id需要满足：
  - id是一个接一个的
  - id是稳定的
  - id非负整数

标记界面元素的无障碍节点id

```
private class ClassicEmotionTouchHelper extends ExploreByTouchHelper{
    ...

    @Override
    protected int getVirtualViewAt(float x, float y) {
        int index = getEmotionIndex(x,y);
        if( index >= 0){
            return index;
        }
        return ExploreByTouchHelper.INVALID_ID;
    }

    @Override
    protected void getVisibleVirtualViews(List<Integer> virtualViewIds) {
        for(int i = 0; i < mEmoticonList.size(); i++){
            virtualViewIds.add(i);
        }
        // 加上删除按钮
        virtualViewIds.add(mEmoticonList.size());
    }

    ...
}
```

- ~~委托处理无障碍~~
- ~~标记虚拟节点ID~~
- 丰富无障碍信息
- 提供用户交互支持

填充无障碍节点的属性

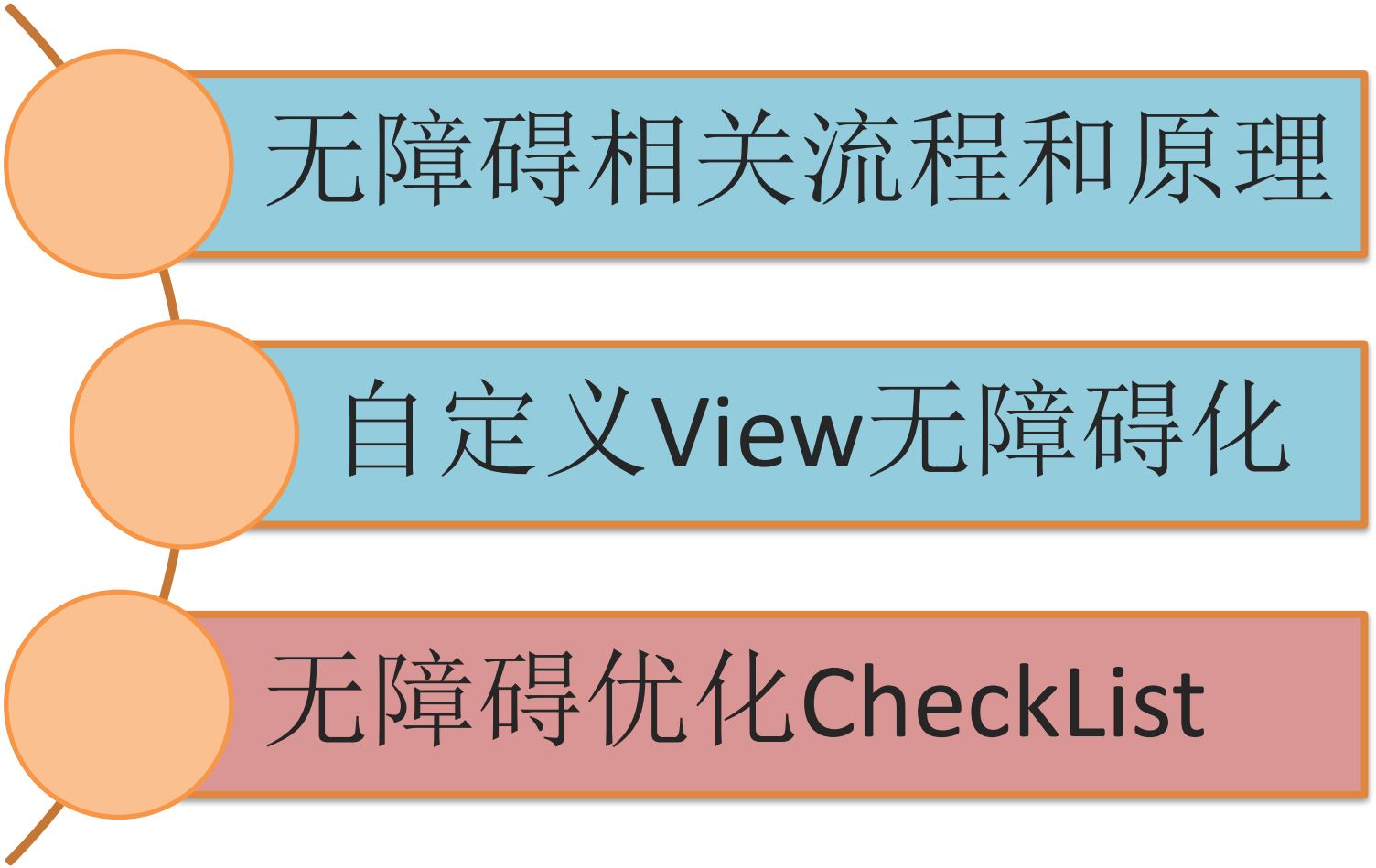
```
private class ClassicEmotionTouchHelper extends ExploreByTouchHelper{
    ...

    @Override
    protected void onPopulateEventForVirtualView(int virtualViewId, AccessibilityEvent event) {
        CharSequence desc = getDescriptionForIndex(virtualViewId);
        event.setContentDescription(desc);
    }

    @Override
    protected void onPopulateNodeForVirtualView(int virtualViewId, AccessibilityNodeInfoCompat node) {
        // 必须设置AccessibilityNodeInfo的Text或者contentDescription
        CharSequence desc = getDescriptionForIndex(virtualViewId);
        node.setContentDescription(desc);

        node.addAction(AccessibilityNodeInfoCompat.ACTION_CLICK);

        // 必须设置BoundsInParent
        Rect bounds = getBoundsForIndex(virtualViewId);
        node.setBoundsInParent(bounds);
    }

    ...
}

```

- ~~委托处理无障碍~~
- ~~标记虚拟节点ID~~
- ~~丰富无障碍信息~~
- 提供用户交互支持

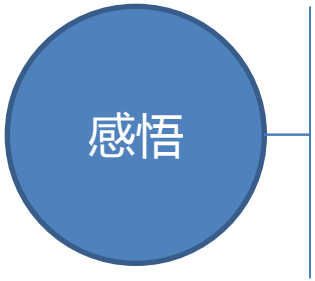
提供用户交互支持

```
private class ClassicEmotionTouchHelper extends ExploreByTouchHelper{
    ...

    @Override
    protected boolean onPerformActionForVirtualView(int virtualViewId, int action, Bundle arguments) {
        switch(action){
        case AccessibilityNodeInfoCompat.ACTION_CLICK:
            onEmotionClick(virtualViewId);
            return true;
        }

        return false;
    }

    ...
}

public void onEmotionClick(int virtualViewId) {
    ...

    mTouchHelper.invalidateVirtualView(virtualViewId);
    mTouchHelper.sendEventForVirtualView(virtualViewId, AccessibilityEvent.TYPE_VIEW_CLICKED);

}
```

- ~~委托处理无障碍~~
- ~~标记虚拟节点ID~~
- ~~丰富无障碍信息~~
- ~~提供用户交互支持~~


无障碍相关流程和原理
自定义View无障碍化
无障碍优化CheckList

- setContentDescription
- Focusable
- Custom View
- 可变元素



1. 细节决定成败

2. 无障碍要迭代

《Android无障碍宝典》
http://geek.csdn.net/news/detail/93269

谢谢