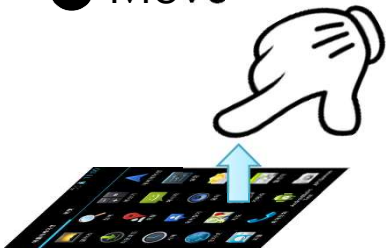
# 이벤트

# 터치 이벤트란?



1 Down

2 Move



3 Up

# 터치 이벤트 수신 함수



Down

**Activity**

```
public boolean
dispatchTouchEvent(MotionEvent ev) {
    return super.dispatchTouchEvent(ev);
}
```

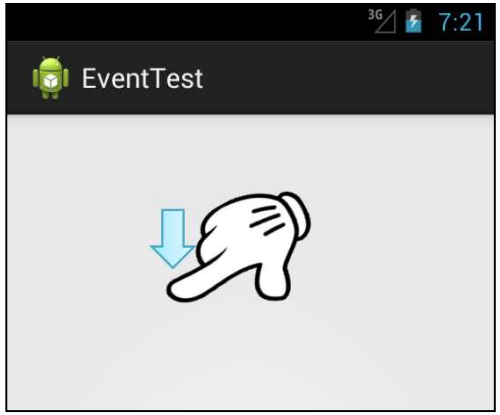
```
public boolean
onTouchEvent(MotionEvent event) {
    return super.onTouchEvent(event);
}
```

# 터치 이벤트 수신 함수



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    /* 액티비티의 dispatchTouchEvent 함수를 재정의한다. */
    @Override
    public boolean dispatchTouchEvent(MotionEvent ev) {
        // 터치 이벤트 정보에 대한 로그를 남긴다.
        Log.d("superdroid", "====================================");
        Log.d("superdroid", "dispatchTouchEvent()");
        Log.d("superdroid", "- ActionCode : " + ev.getAction());
        Log.d("superdroid", "- XY Position : " + ev.getX() + "," + ev.getY());
        Log.d("superdroid", "- Event Time : " + ev.getEventTime());
        Log.d("superdroid", "- Down Event Time : " + ev.getDownTime());

        return super.dispatchTouchEvent(ev);
    }

    /* 액티비티의 onTouchEvent 함수를 재정의한다. */
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Log.i("superdroid", "====================================");
        Log.i("superdroid", "onTouchEvent()");
        Log.i("superdroid", "- ActionCode : " + event.getAction());
        Log.i("superdroid", "- XY Position : " + event.getX() + "," + event.getY());
        Log.i("superdroid", "- Event Time : " + event.getEventTime());
        Log.i("superdroid", "- Down Event Time : " + event.getDownTime());

        return super.onTouchEvent(event);
    }
}
```

# 터치 이벤트 수신 함수



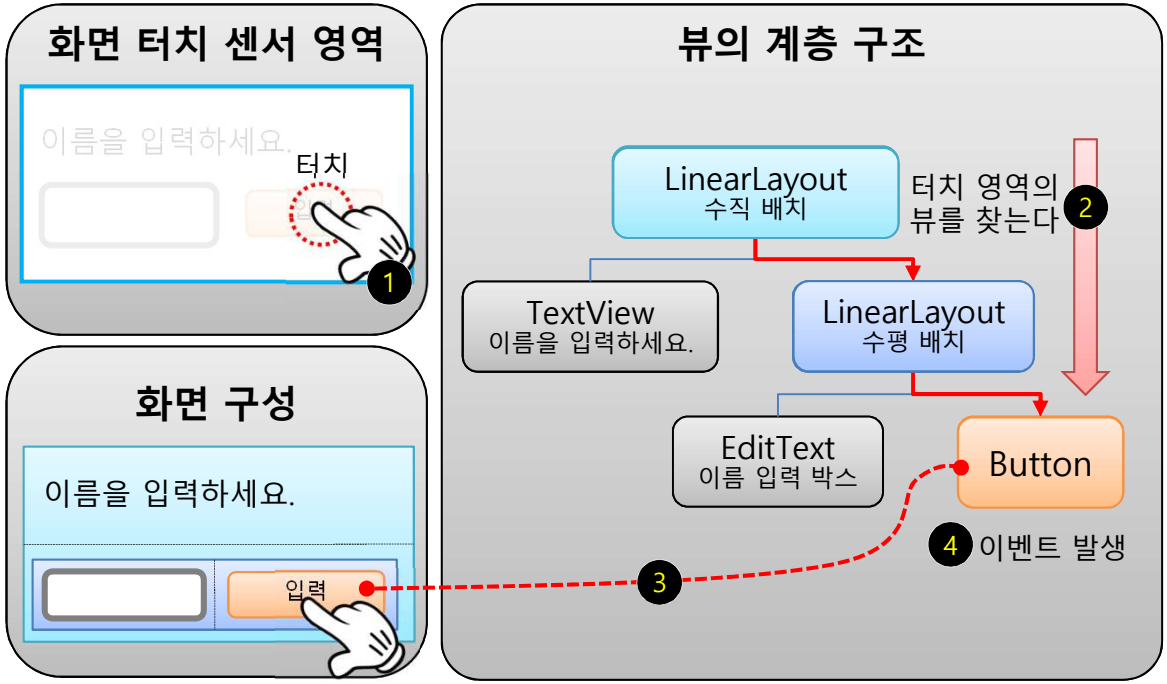

● 로그 출력 결과

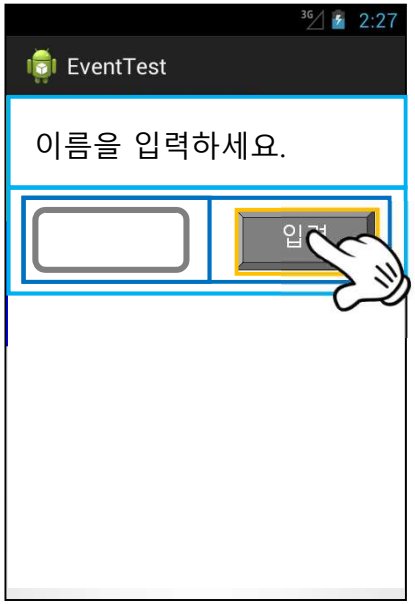
```
==================================
dispatchTouchEvent()
- ActionCode : 0
- XY Position : 26.0,141.0
- Event Time : 1326475
- Down Event Time : 1326475
==================================
onTouchEvent()
- ActionCode : 0
- XY Position : 26.0,141.0
- Event Time : 1326475
- Down Event Time : 1326475
```

| 함수명 | 설명 |
|---|---|
| getAction | 터치 이벤트의 액션값이 전달되며, 종류는 다음과 같다.<br>ACTION_DOWN = 0<br>ACTION_UP = 1<br>ACTION_MOVE = 2<br>ACTION_CANCEL = 3 |
| getX | 이벤트가 발생한 X축 좌표 위치. |
| getY | 이벤트가 발생한 Y축 좌표 위치. |
| getEventTime | 이벤트가 발생한 시간(밀리세컨드, 즉 1초의 1000분의 1).<br>※ 시간 정보는 부팅한 시점부터 이벤트가 발생한 시점까지다. |
| detDownTime | 다운 이벤트가 발생한 시간(밀리세컨드 단위).<br>다운, 이동, 업 이벤트를 하나의 단위로 보아야 한다. 그런 취지에서 활용 가치가 있는 이벤트의 시작인 다운 이벤트 발생 시간 정보를 제공한다. |

# 터치 이벤트 전달 과정

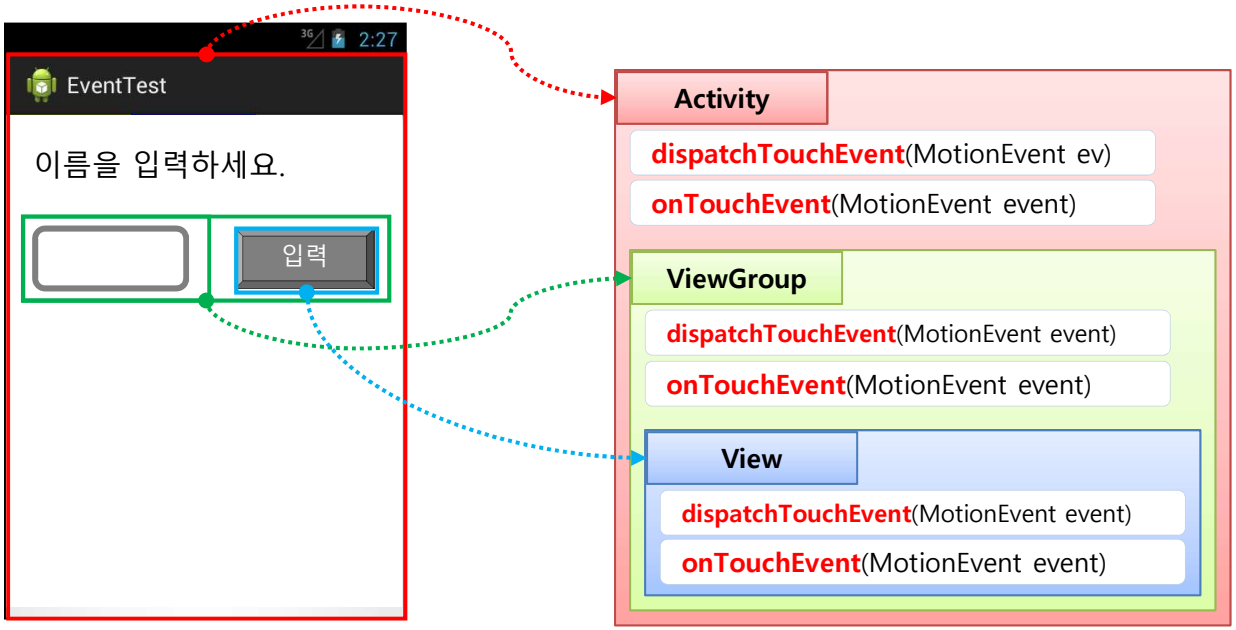



화면 터치 센서 영역
이름을 입력하세요.
터치
1
화면 구성
이름을 입력하세요.
입력
3
뷰의 계층 구조
LinearLayout
수직 배치
터치 영역의
뷰를 찾는다
2
TextView
이름을 입력하세요.
LinearLayout
수평 배치
EditText
이름 입력 박스
Button
4 이벤트 발생


2:27
EventTest
이름을 입력하세요.
입력

# 터치 이벤트 전달 과정

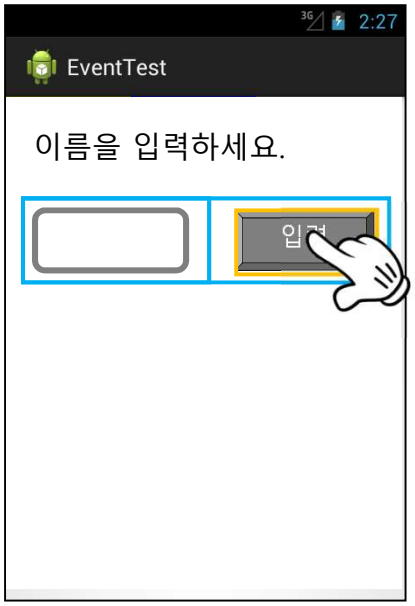


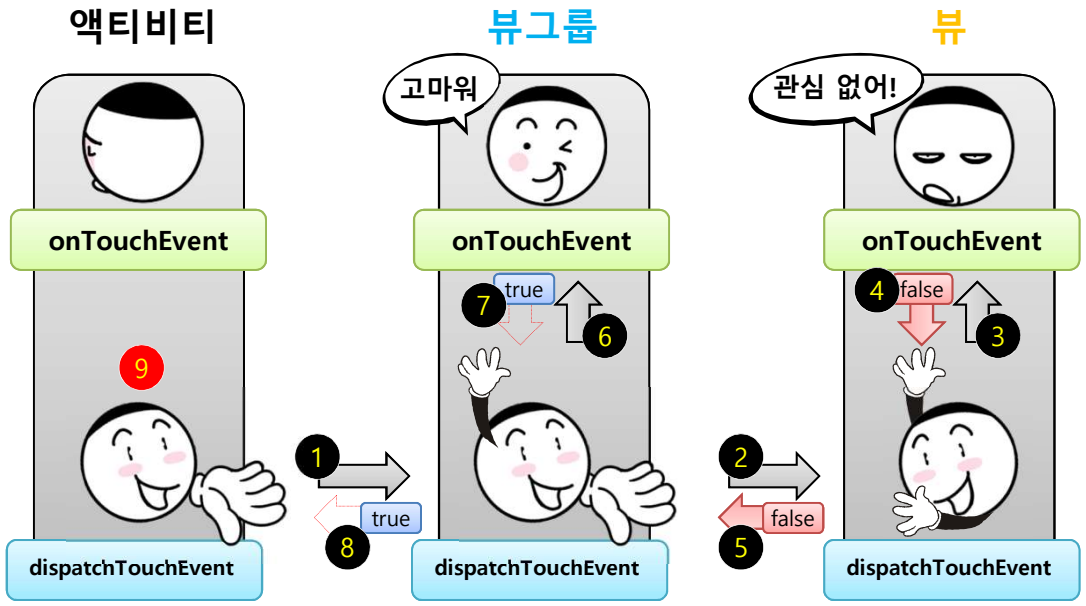



액티비티뿐만 아니라 뷰/뷰그룹에도 존재하는 터치 이벤트 관련 함수

# 터치 이벤트 전달 과정




2:27
EventTest
이름을 입력하세요.
입력


액티비티
뷰그룹
뷰
고마워
관심 없어!
onTouchEvent
onTouchEvent
onTouchEvent
true
false
1
2
3
4
5
6
7
8
9
true
false
dispatchTouchEvent
dispatchTouchEvent
dispatchTouchEvent

# 터치 이벤트 전달 과정



```
class CustomView extends View {

    public CustomView(Context context)
    {
        super(context);
    }

    /* 뷰의 dispatchTouchEvent 함수를 재정의한다. */
    @Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        Log.d("superdroid", "CustomView dispatchTouchEvent() >> " + event.getAction());

        return super.dispatchTouchEvent(event);
    }

    /* 뷰의 onTouchEvent 함수를 재정 한다. */
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Log.d("superdroid", "CustomView onTouchEvent() >> "  + event.getAction());

        return super.onTouchEvent(event);
    }
}
```

# 터치 이벤트 전달 과정



```
class CustomViewGroup extends FrameLayout {

    public CustomViewGroup(Context context)
    {
        super(context);
    }

    /* 뷰그룹의 dispatchTouchEvent 함수를 재정의한다. */
    @Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        Log.i("superdroid", "CustomViewGroup dispatchTouchEvent() >> "  + event.getAction());

        return super.dispatchTouchEvent(event);
    }

    /* 뷰그룹의 onTouchEvent 함수를 재정의한다. */
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Log.i("superdroid", "CustomViewGroup onTouchEvent() >> " + event.getAction());

        return super.onTouchEvent(event);
    }
}
```

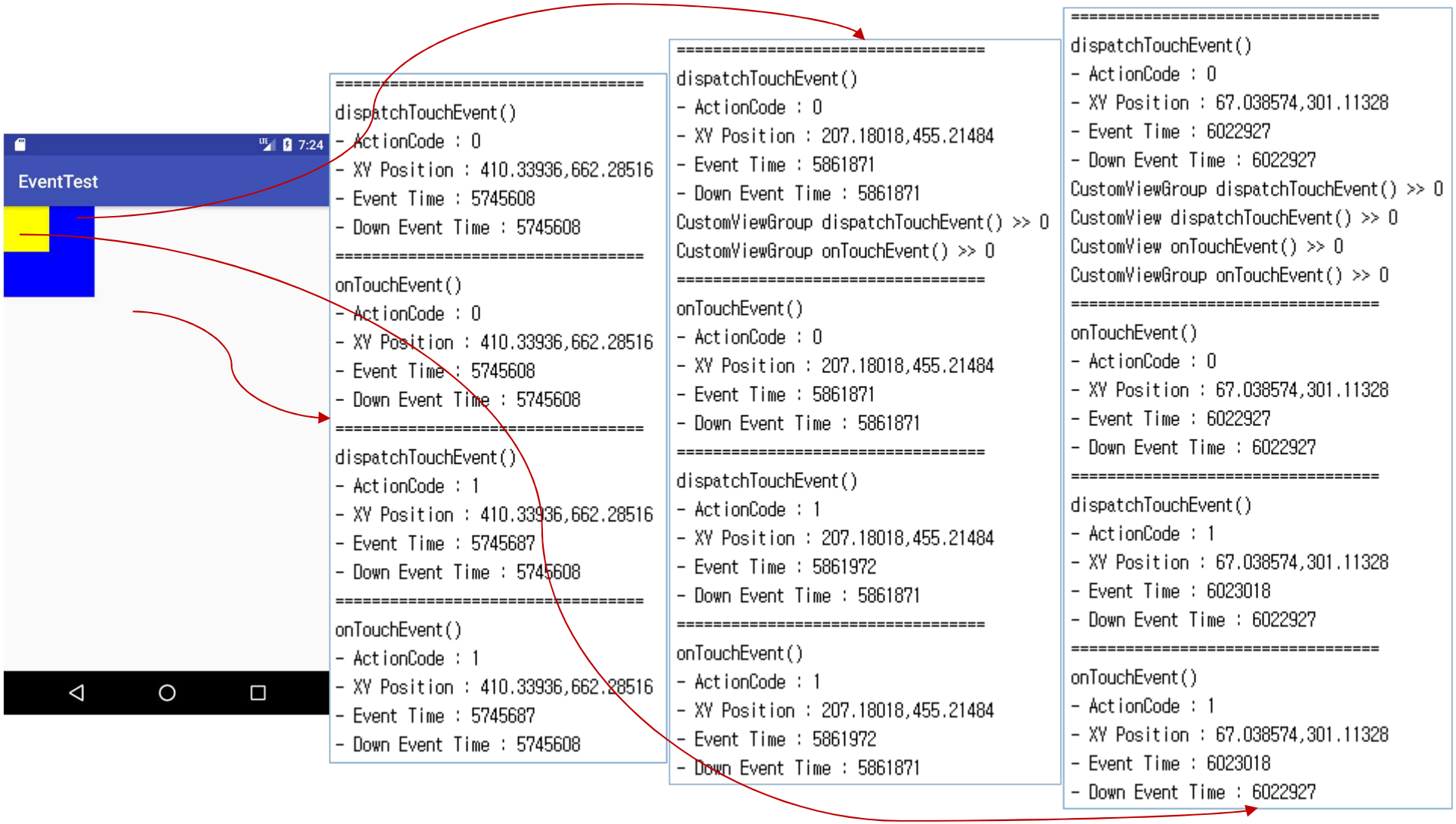
# 터치 이벤트 전달 과정



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 1. 뷰 그룹을 생성한다.
        // ============================================================================
        CustomViewGroup viewGroup = new CustomViewGroup(this);
        viewGroup.setBackgroundColor(Color.BLUE);
        FrameLayout.LayoutParams viewGroupLp = new FrameLayout.LayoutParams(300, 300);
        // ============================================================================

        // 2. 뷰를 생성한다.
        // ============================================================================
        CustomView view = new CustomView(this);
        view.setBackgroundColor(Color.YELLOW);
        FrameLayout.LayoutParams viewLp = new FrameLayout.LayoutParams(150,150);
        // ============================================================================

        // 3. 생성된 뷰를 뷰그룹에 추가한다.
        // ============================================================================
        viewGroup.addView(view, viewLp);
        // ============================================================================

        // 4. 컨텐츠 영역에 생성된 뷰그룹을 추가한다.
        // ============================================================================
        setContentView(viewGroup, viewGroupLp);
        // ============================================================================
    }

    ...
}
```

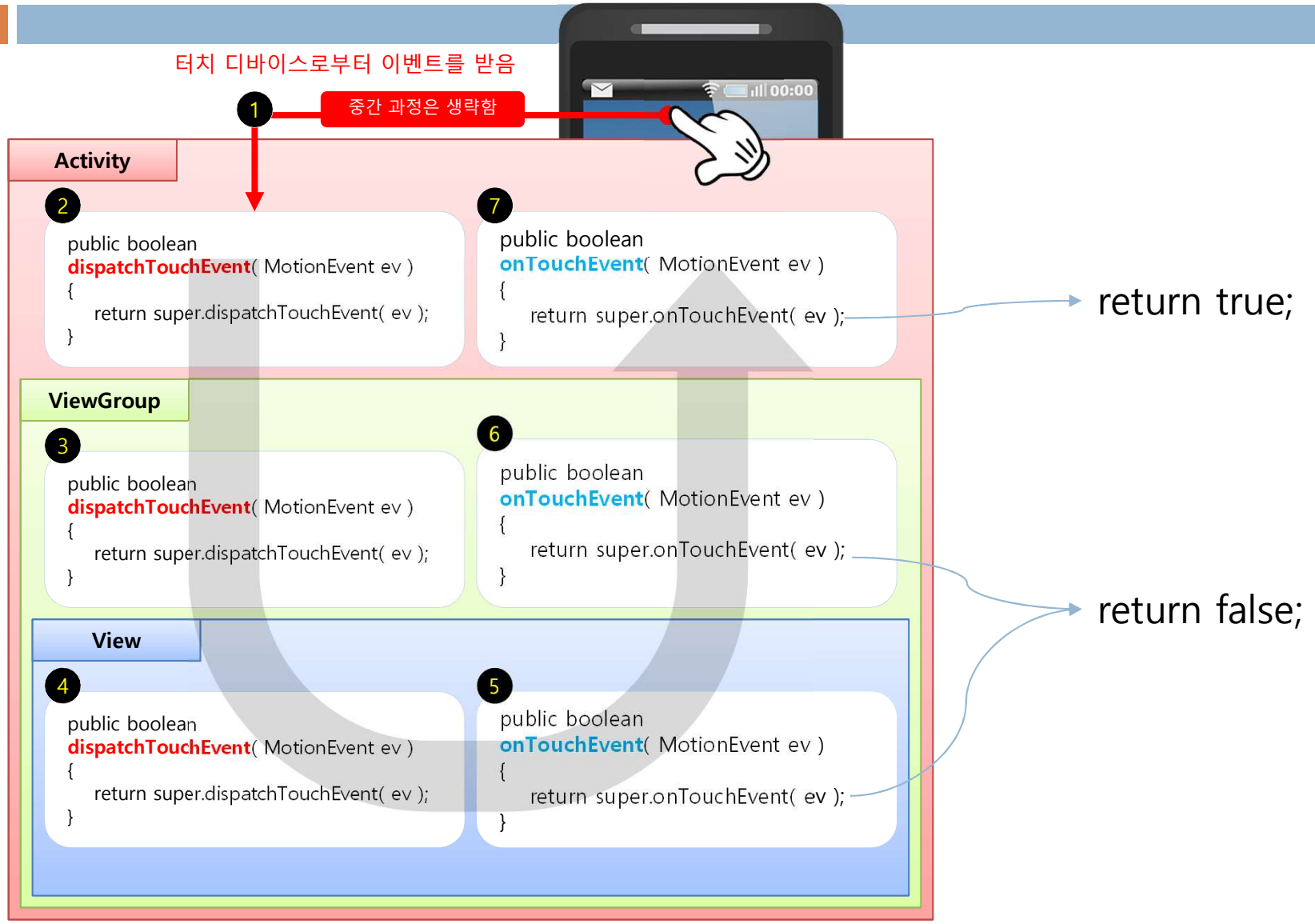
# 터치 이벤트 전달 과정



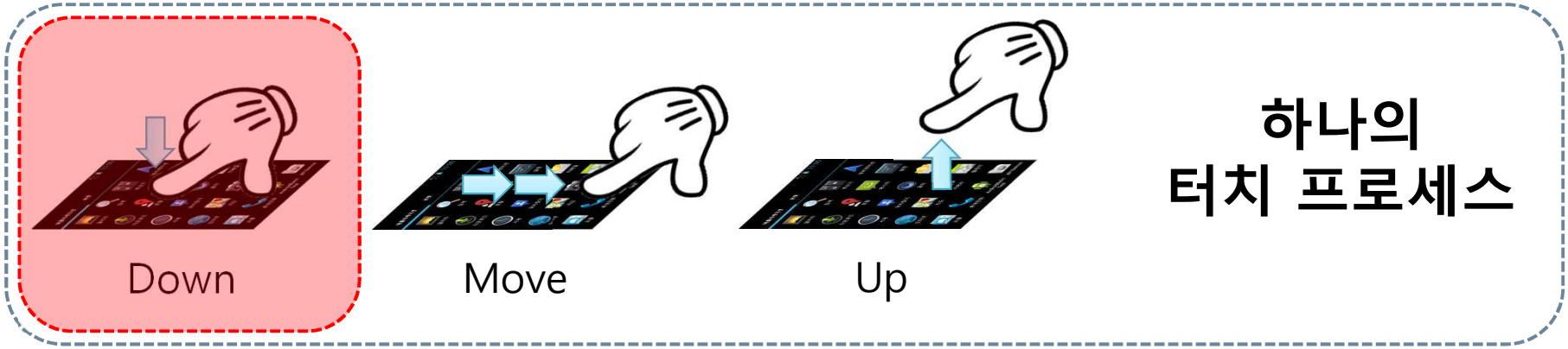

7:24
EventTest
==========
dispatchTouchEvent()
- ActionCode : 0
- XY Position : 410.33936,662.28516
- Event Time : 5745608
- Down Event Time : 5745608
==========
onTouchEvent()
- ActionCode : 0
- XY Position : 410.33936,662.28516
- Event Time : 5745608
- Down Event Time : 5745608
==========
dispatchTouchEvent()
- ActionCode : 1
- XY Position : 410.33936,662.28516
- Event Time : 5745687
- Down Event Time : 5745608
==========
onTouchEvent()
- ActionCode : 1
- XY Position : 410.33936,662.28516
- Event Time : 5745687
- Down Event Time : 5745608
==========
dispatchTouchEvent()
- ActionCode : 0
- XY Position : 207.18018,455.21484
- Event Time : 5861871
- Down Event Time : 5861871
CustomViewGroup dispatchTouchEvent() >> 0
CustomViewGroup onTouchEvent() >> 0
==========
onTouchEvent()
- ActionCode : 0
- XY Position : 207.18018,455.21484
- Event Time : 5861871
- Down Event Time : 5861871
==========
dispatchTouchEvent()
- ActionCode : 1
- XY Position : 207.18018,455.21484
- Event Time : 5861972
- Down Event Time : 5861871
==========
onTouchEvent()
- ActionCode : 1
- XY Position : 207.18018,455.21484
- Event Time : 5861972
- Down Event Time : 5861871
==========
dispatchTouchEvent()
- ActionCode : 0
- XY Position : 67.038574,301.11328
- Event Time : 6022927
- Down Event Time : 6022927
CustomViewGroup dispatchTouchEvent() >> 0
CustomView dispatchTouchEvent() >> 0
CustomView onTouchEvent() >> 0
CustomViewGroup onTouchEvent() >> 0
==========
onTouchEvent()
- ActionCode : 0
- XY Position : 67.038574,301.11328
- Event Time : 6022927
- Down Event Time : 6022927
==========
dispatchTouchEvent()
- ActionCode : 1
- XY Position : 67.038574,301.11328
- Event Time : 6023018
- Down Event Time : 6022927
==========
onTouchEvent()
- ActionCode : 1
- XY Position : 67.038574,301.11328
- Event Time : 6023018
- Down Event Time : 6022927

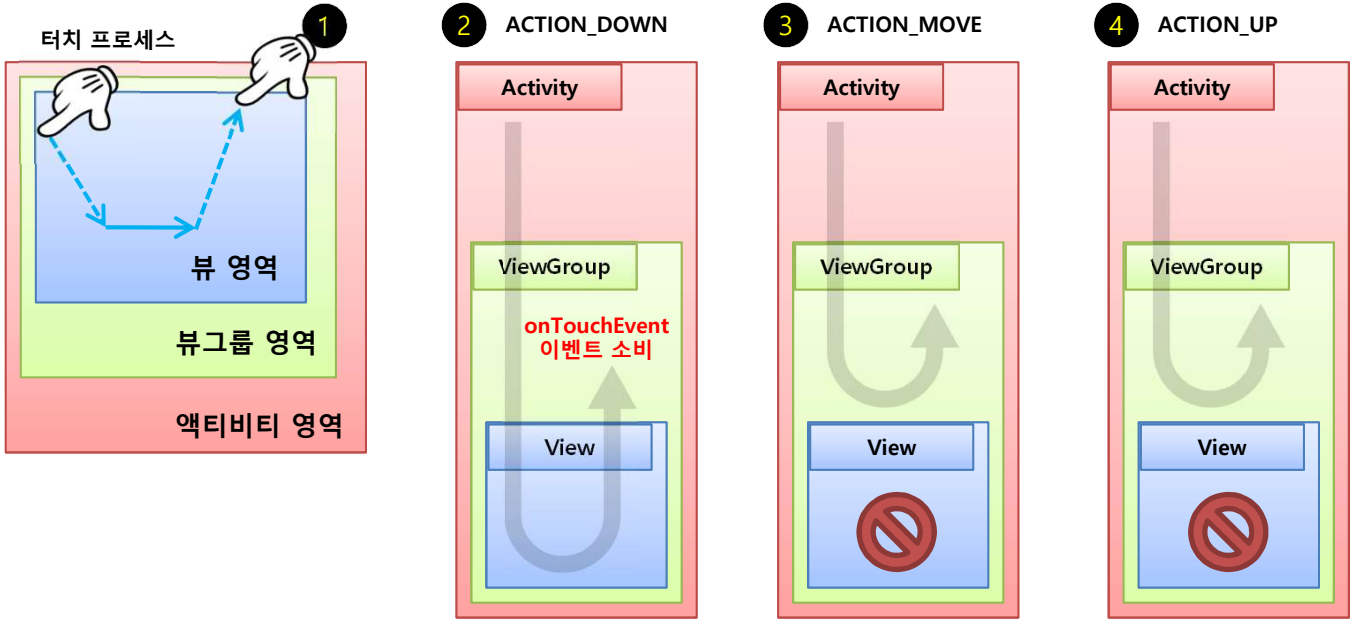
# 터치 이벤트 전달 과정 정리





# 터치 다운 이벤트


Down
Move
Up
하나의
터치 프로세스


터치 프로세스
1
뷰 영역
뷰그룹 영역
액티비티 영역
2
ACTION_DOWN
Activity
ViewGroup
onTouchEvent
이벤트 소비
View
3
ACTION_MOVE
Activity
ViewGroup
View
4
ACTION_UP
Activity
ViewGroup
View

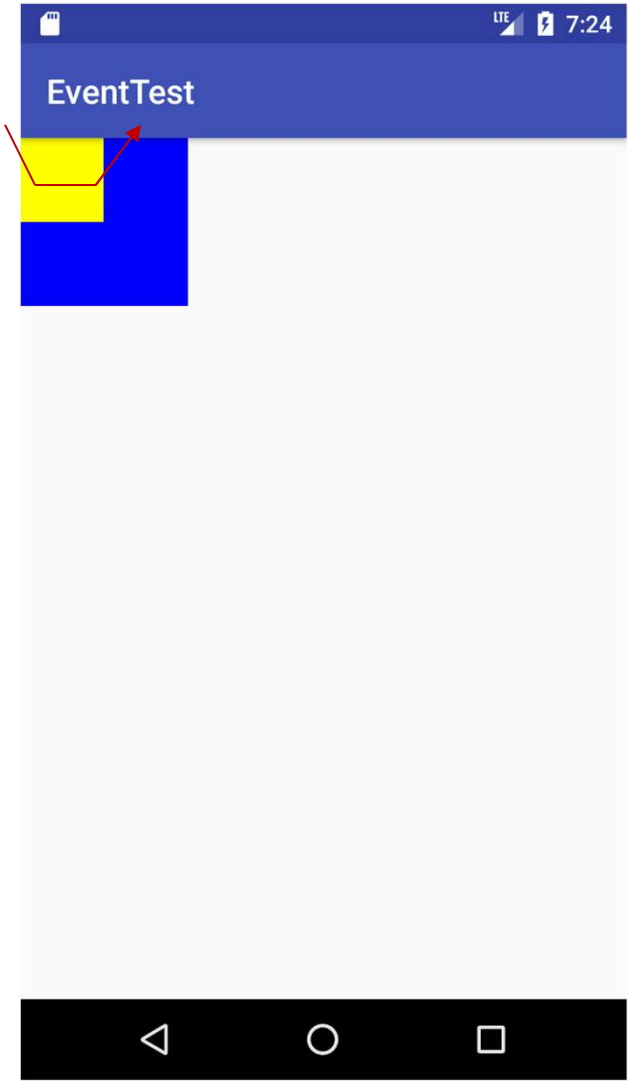
# 터치 다운 이벤트



```
class CustomViewGroup extends FrameLayout {

    public CustomViewGroup(Context context)
    {
        super(context);
    }

    /* 뷰그룹의 dispatchTouchEvent 함수를 재정의한다. */
    @Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        Log.i("superdroid", "CustomViewGroup dispatchTouchEvent() >> "  + event.getAction());

        return super.dispatchTouchEvent(event);
    }

    /* 뷰그룹의 onTouchEvent 함수를 재정의한다. */
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Log.i("superdroid", "CustomViewGroup onTouchEvent() >> " + event.getAction());

        return true;
    }
}
```

# 터치 다운 이벤트

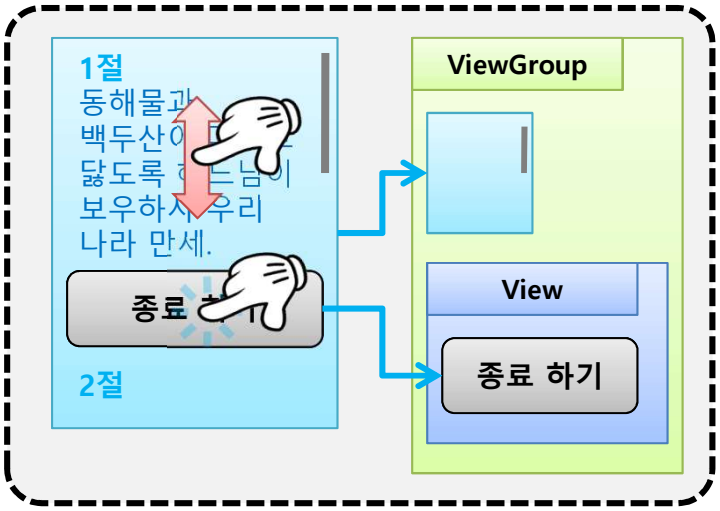


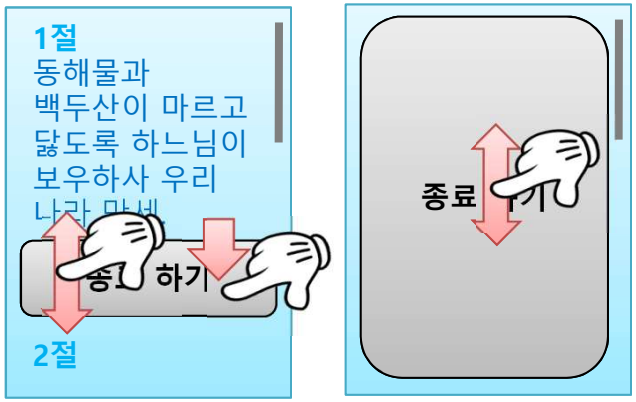



```
==================================
dispatchTouchEvent()
- ActionCode : 0
- XY Position : 20.006104,286.11328
- Event Time : 7174493
- Down Event Time : 7174493
CustomViewGroup dispatchTouchEvent() >> 0
CustomView dispatchTouchEvent() >> 0
CustomView onTouchEvent() >> 0
CustomViewGroup onTouchEvent() >> 0
==================================
dispatchTouchEvent()
- ActionCode : 2
- XY Position : 23.005371,290.09766
- Event Time : 7174575
- Down Event Time : 7174493
CustomViewGroup dispatchTouchEvent() >> 2
CustomViewGroup onTouchEvent() >> 2
==================================
dispatchTouchEvent()
- ActionCode : 1
- XY Position : 86.05591,283.125
- Event Time : 7175095
- Down Event Time : 7174493
CustomViewGroup dispatchTouchEvent() >> 1
CustomViewGroup onTouchEvent() >> 1
```

# 영역이 교차하는 뷰와 뷰그룹 모두 다운 이벤트가 필요한 상황의 문제점





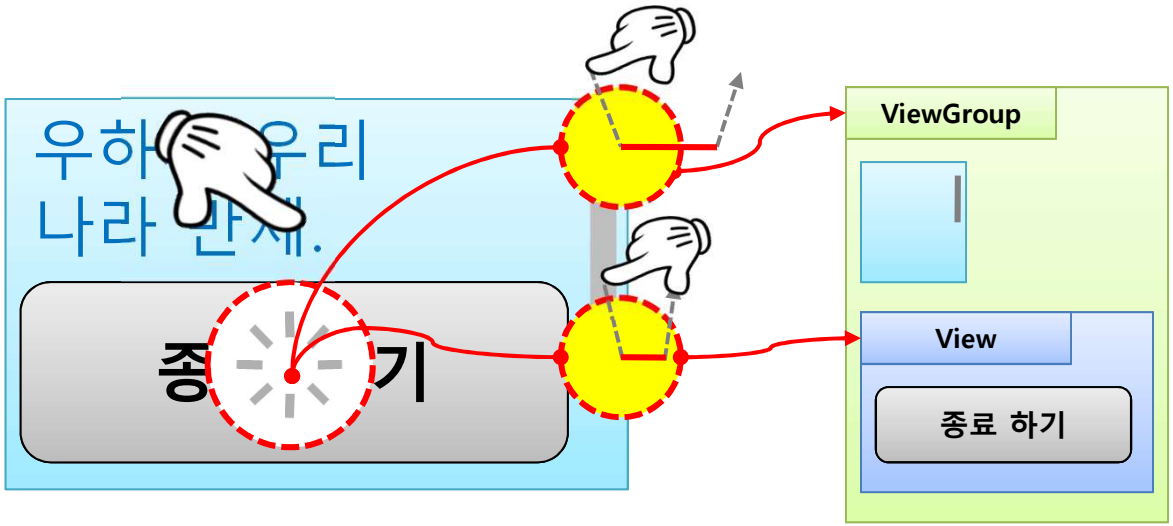




**뷰와 부모뷰의 이벤트가 겹침**

**부모뷰가 이벤트를 받지 못함**

# onInterceptTouchEvent 함수를 이용한 문제 해결법





# onInterceptTouchEvent 함수



```
class CustomView extends View {

    public CustomView(Context context)
    {
        super(context);
    }

    /* 뷰의 dispatchTouchEvent 함수를 재정의한다. */
    @Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        Log.d("superdroid", "CustomView dispatchTouchEvent() >> " + event.getAction());

        return super.dispatchTouchEvent(event);
    }

    /* 뷰의 onTouchEvent 함수를 재정 한다. */
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Log.d("superdroid", "CustomView onTouchEvent() >> "  + event.getAction());

        // 1) 뷰가 이벤트를 받으면 특정 처리를 하기 위해 true를 리턴한다.
        return true;
    }
}
```

# onInterceptTouchEvent 함수

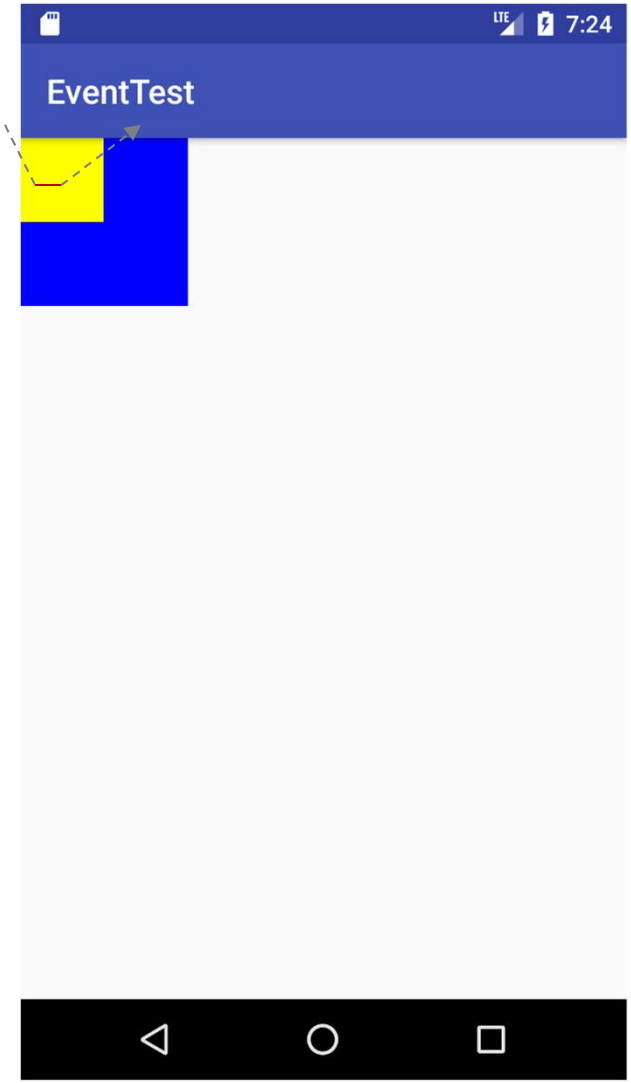


```
class CustomViewGroup extends FrameLayout {

    float initialY = 0;

    ...

    /* 뷰그룹의 onTouchEvent 함수를 재정의한다. */
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Log.i("superdroid", "CustomViewGroup onTouchEvent() >> " + event.getAction());

        // 1) 뷰 그룹이 이벤트를 받으면 특정 처리를 하기 true를 리턴한다.
        return true;
    }

    /* 뷰그룹의 onInterceptTouchEvent 함수를 재정의한다.*/
    @Override
    public boolean onInterceptTouchEvent(MotionEvent ev) {
        Log.i("superdroid", "CustomViewGroup onInterceptTouchEvent()" + ev.getAction());

        switch(ev.getAction()) {
            case MotionEvent.ACTION_DOWN:
                // 1) 터치 다운 위치의 Y 위치를 기억해 둔다.
                initialY = ev.getY();
                break;

            case MotionEvent.ACTION_MOVE:
                // 2) 터치 다운 Y 위치에서 20픽셀을 초과 이동되면 이벤트를 가로챈다.
                if(Math.abs(initialY - ev.getY()) >= 20) return true;
        }

        return super.onInterceptTouchEvent(ev);
    }
}
```

이 메소드는 ViewGroup의 메소드

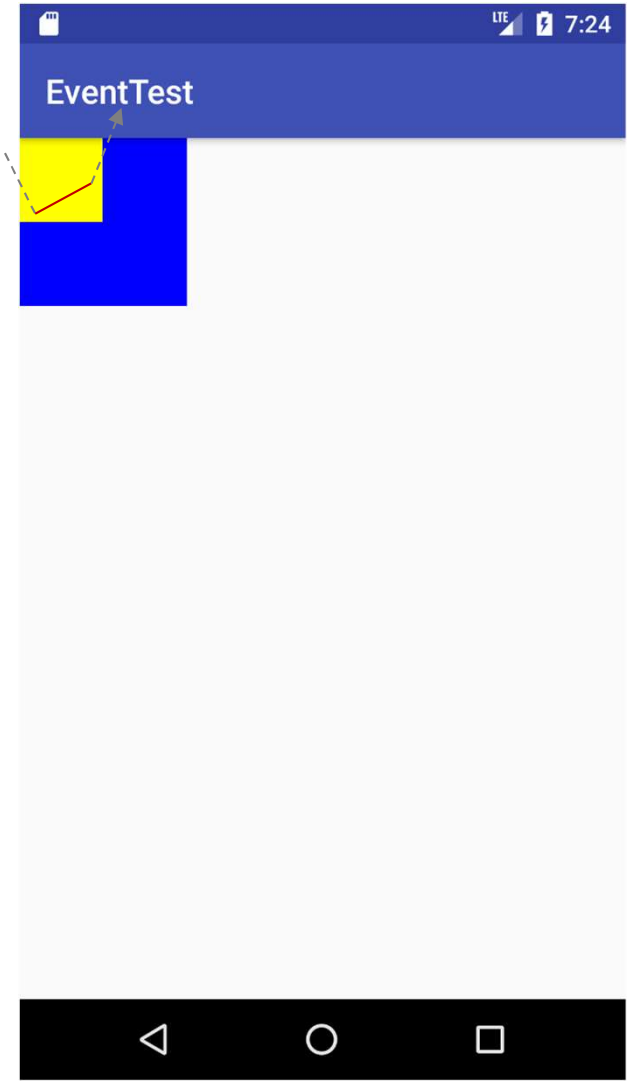
# onInterceptTouchEvent 함수






```
==================================
dispatchTouchEvent()
- ActionCode : 0
- XY Position : 69.04907,273.10547
- Event Time : 10507671
- Down Event Time : 10507671
CustomViewGroup dispatchTouchEvent() >> 0
CustomViewGroup onInterceptTouchEvent()0
CustomView dispatchTouchEvent() >> 0
CustomView onTouchEvent() >> 0
==================================
dispatchTouchEvent()
- ActionCode : 2
- XY Position : 69.04907,276.21106
- Event Time : 10507776
- Down Event Time : 10507671
CustomViewGroup dispatchTouchEvent() >> 2
CustomViewGroup onInterceptTouchEvent()2
CustomView dispatchTouchEvent() >> 2
CustomView onTouchEvent() >> 2
==================================
dispatchTouchEvent()
- ActionCode : 1
- XY Position : 72.06482,293.08594
- Event Time : 10508204
- Down Event Time : 10507671
CustomViewGroup dispatchTouchEvent() >> 1
CustomViewGroup onInterceptTouchEvent()1
CustomView dispatchTouchEvent() >> 1
CustomView onTouchEvent() >> 1
```

# onInterceptTouchEvent 함수

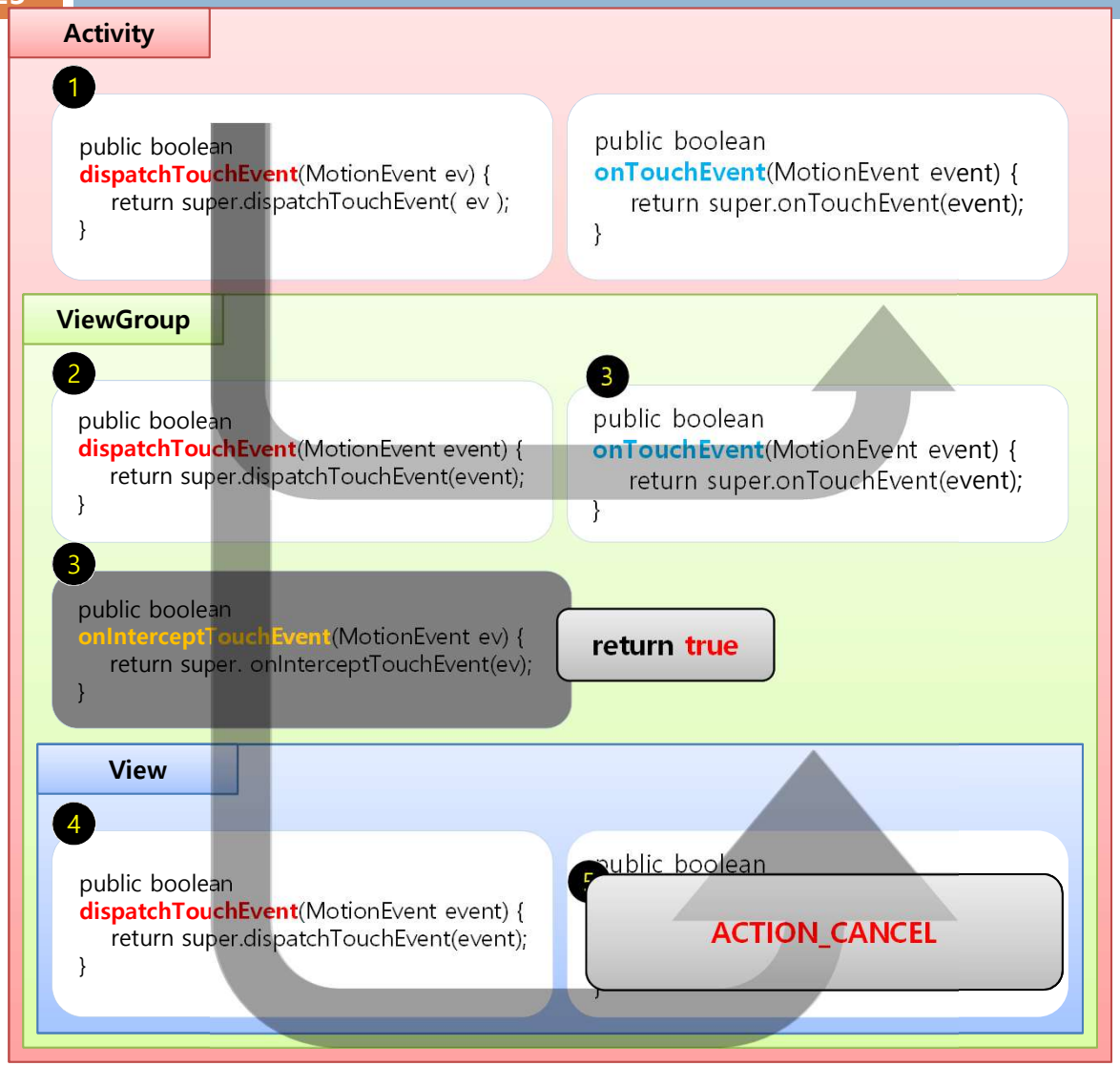



```
==================================
dispatchTouchEvent()
- ActionCode : 0
- XY Position : 89.055176,369.14062
- Event Time : 11584830
- Down Event Time : 11584830
CustomViewGroup dispatchTouchEvent() >> 0
CustomViewGroup onInterceptTouchEvent()0
CustomView dispatchTouchEvent() >> 0
CustomView onTouchEvent() >> 0
==================================
dispatchTouchEvent()
- ActionCode : 2
- XY Position : 89.055176,362.62372
- Event Time : 11584960
- Down Event Time : 11584830
CustomViewGroup dispatchTouchEvent() >> 2
CustomViewGroup onInterceptTouchEvent()2
CustomView dispatchTouchEvent() >> 2
CustomView onTouchEvent() >> 2
==================================
dispatchTouchEvent()
- ActionCode : 2
- XY Position : 89.055176,347.64423
- Event Time : 11584976
- Down Event Time : 11584830
CustomViewGroup dispatchTouchEvent() >> 2
CustomViewGroup onInterceptTouchEvent()2
CustomView dispatchTouchEvent() >> 3
CustomView onTouchEvent() >> 3
==================================
dispatchTouchEvent()
- ActionCode : 2
- XY Position : 89.055176,334.04565
- Event Time : 11584993
- Down Event Time : 11584830
CustomViewGroup dispatchTouchEvent() >> 2
CustomViewGroup onTouchEvent() >> 2
```

```
==================================
dispatchTouchEvent()
- ActionCode : 1
- XY Position : 93.07617,260.09766
- Event Time : 11585571
- Down Event Time : 11584830
CustomViewGroup dispatchTouchEvent() >> 1
CustomViewGroup onTouchEvent() >> 1
```

# onInterceptTouchEvent 함수의 활용







Down
Move
Move
Move

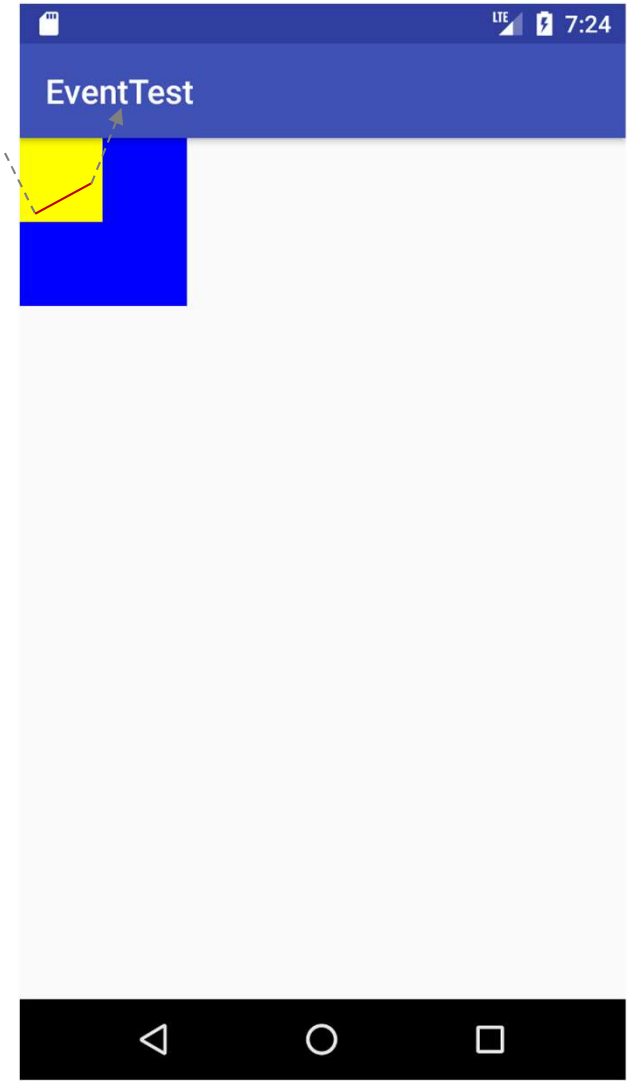
# requestDisallowInterceptTouchEvent 함수



자식 뷰가 부모 뷰그룹에게 이벤트를 가로채지 않도록 요청한다. **단 한번의 터치 프로세스에만 유효하다.**

```
class CustomView extends View {
    private boolean isFirstDown = true;
    public CustomView(Context context)
    {
        super(context);
    }

    /* 뷰의 dispatchTouchEvent 함수를 재정의한다. */
    @Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        Log.d("superdroid", "CustomView dispatchTouchEvent() >> " + event.getAction());

        return super.dispatchTouchEvent(event);
    }

    /* 뷰의 onTouchEvent 함수를 재정 한다. */
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Log.d("superdroid", "CustomView onTouchEvent() >> "  + event.getAction());

        // 1) 뷰에게 전달되는 첫 번째 터치 이벤트 프로세스는 부모 뷰그룹이
        // 가로채지 않도록 설정한다.
        // ============================================================
        if(isFirstDown == true && event.getAction() == MotionEvent.ACTION_DOWN ) {
            getParent().requestDisallowInterceptTouchEvent(true);
            isFirstDown = false;
        }
        // ============================================================
        return true;
    }
}
```

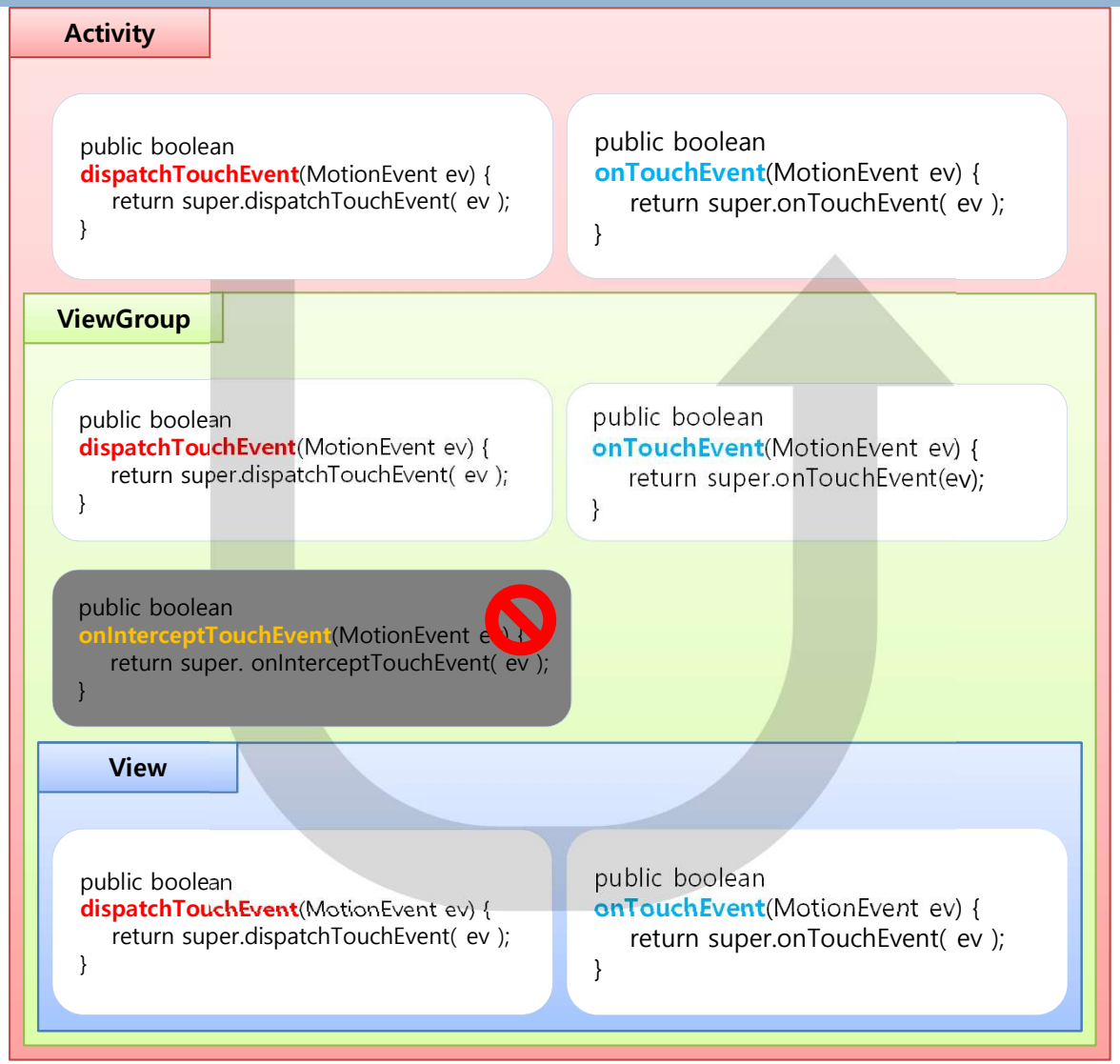
# requestDisallowInterceptTouchEvent 함수




```
================================
dispatchTouchEvent()
- ActionCode : 0
- XY Position : 49.04297,257.10938
- Event Time : 13935254
- Down Event Time : 13935254
CustomViewGroup dispatchTouchEvent() >> 0
CustomViewGroup onInterceptTouchEvent()0
CustomView dispatchTouchEvent() >> 0
CustomView onTouchEvent() >> 0
================================
dispatchTouchEvent()
- ActionCode : 2
- XY Position : 49.04297,255.11719
- Event Time : 13935469
- Down Event Time : 13935254
CustomViewGroup dispatchTouchEvent() >> 2
CustomView dispatchTouchEvent() >> 2
CustomView onTouchEvent() >> 2
================================
dispatchTouchEvent()
- ActionCode : 2
- XY Position : 49.04297,263.16537
- Event Time : 13935493
- Down Event Time : 13935254
CustomViewGroup dispatchTouchEvent() >> 2
CustomView dispatchTouchEvent() >> 2
CustomView onTouchEvent() >> 2
================================
dispatchTouchEvent()
- ActionCode : 1
- XY Position : 86.8265,358.88858
- Event Time : 13936061
- Down Event Time : 13935254
CustomViewGroup dispatchTouchEvent() >> 1
CustomView dispatchTouchEvent() >> 1
CustomView onTouchEvent() >> 1
```

# requestDisallowInterceptTouchEvent 함수





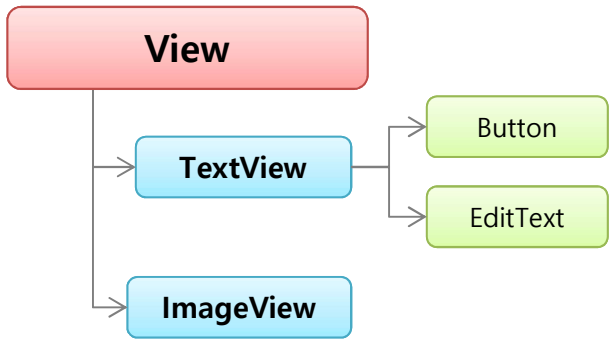
# 터치 이벤트 리스너



- 터치 이벤트 리스너의 필요성
  - 터치 이벤트를 받기 위해서는 늘 뷰를 상속받아 재정의해야만 한다.
  - 특히 뷰/뷰그룹을 레이아웃 XML을 통해 생성하는 경우에는 상속받기가 애매해져 버린다.

src/CustomView.java

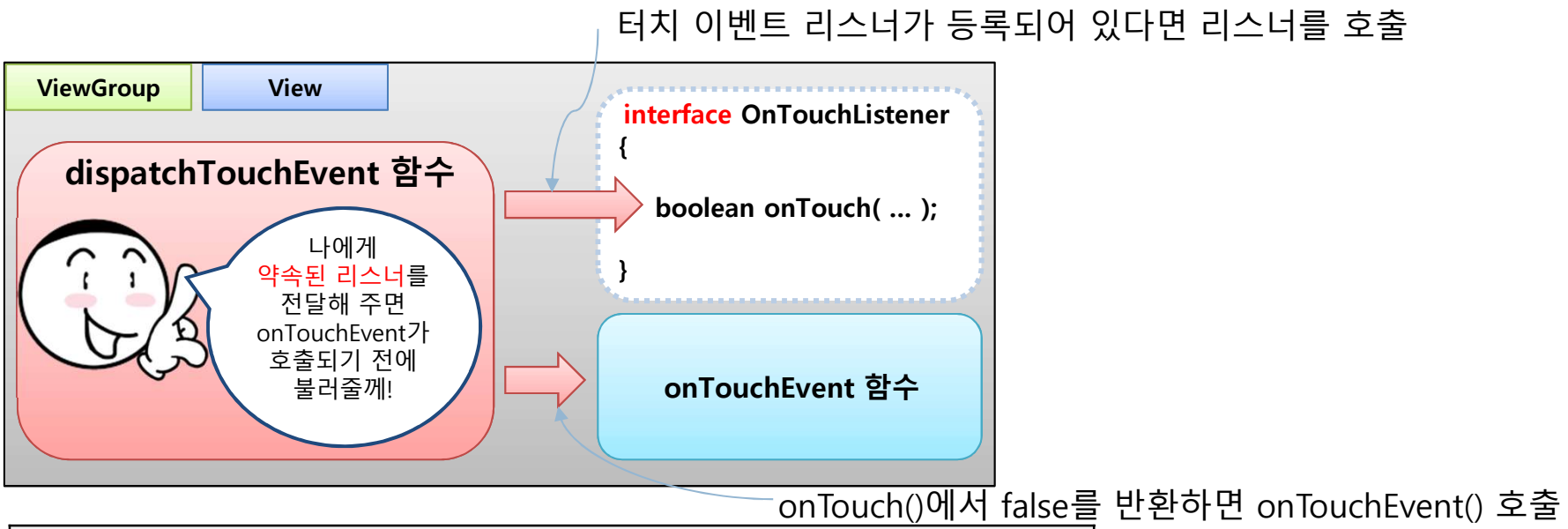
```
class CustomView extends View {


    @Override
    public boolean onTouchEvent(MotionEvent event){

    }
}
```



# 터치 이벤트 리스너



- 터치 이벤트 리스너란?




View.java

```
public class View ...
{
    ...
    public interface OnTouchListener {
        boolean onTouch(View v, MotionEvent event);
    }
    ...
```

# 터치 리스너 등록을 위한 레이아웃 XML



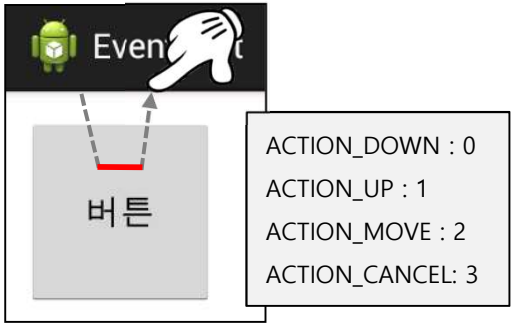
**res/layout/activity_main.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="20px">

    <Button android:id="@+id/button_view"
        android:text="버튼"
        android:layout_width="100dp"
        android:layout_height="100dp"/>

</LinearLayout>
```

# 액티비티에 터치 이벤트 리스너 클래스 구현



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 1. 레이아웃을 콘텐트 영역에 설정한다.
        setContentView(R.layout.activity_main);

        // 2. 레이아웃에 포함된 버튼 객체를 참조한다.
        Button button = (Button) findViewById(R.id.button_view);

        // 3. 버튼에 터치 이벤트 리스너 객체를 생성 및 설정한다.
        button.setOnTouchListener(new MyTouchListener());
    }

    /* 버튼 뷰의 터치 이벤트 리스너 클래스 구현 */
    private class MyTouchListener implements View.OnTouchListener {
        // 4. 각종 터치 이벤트는 onTouch 함수를 통해 전달된다.
        // ================================================================
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            Log.d("superdroid", "Button View onTouch() >> "  + event.getAction());

            // 5. 이벤트를 소비하진 않도록 한다.
            return false;
        }
        // ================================================================
    }
}
```




● 로그 출력 결과

```
Button View onTouch() >> 0
Button View onTouch() >> 2
Button View onTouch() >> 2
Button View onTouch() >> 2
Button View onTouch() >> 1
```

# 액티비티가 View.OnTouchListener를 상속받아 구현



```
// 1. 액티비티가 View.OnTouchListener
public class MainActivity extends AppCompatActivity implements View.OnTouchListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        Button button = (Button) findViewById(R.id.button_view);

        // 2. 버튼에 터치 이벤트 리스너 객체를 생성 및 설정한다.
        button.setOnTouchListener(this);
    }

    // 3. 각종 터치 이벤트는 onTouch 함수를 통해 전달된다.
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        Log.d("superdroid", "Button View onTouch() >> "  + event.getAction());

        return false;
    }
}
```

# 익명의 내부 클래스 구현



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        Button button = (Button) findViewById(R.id.button_view);

        // 1. 버튼에 터치 이벤트 리스너 객체를 생성 및 설정한다.
        button.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent motionEvent) {
                Log.d("superdroid", "Button View onTouch() >> "  + motionEvent.getAction());

                return false;
            }
        });
    }

}
```

# 터치 이벤트 확장



- 터치는 다운, 이동, 업 이벤트가 전부며, 이 세 가지 이벤트의 조합으로 다양한 동작을 처리할 수 있다.
  - 그 중에서도 빈번히 사용되는 이벤트 조합들의 경우
    - 뷰 내부에서 확장된 이벤트 리스너를 제공한다.

# 터치 이벤트 확장 – 클릭 리스너



- 클릭 리스너는 뷰 영역을 클릭했을 때 호출.
- 안드로이드에서 클릭이란 뷰 영역을 손가락으로 눌렀다 (ACTION_DOWN) 바로 떼는(ACTION_UP) 조합된 터치 동작을 말함.

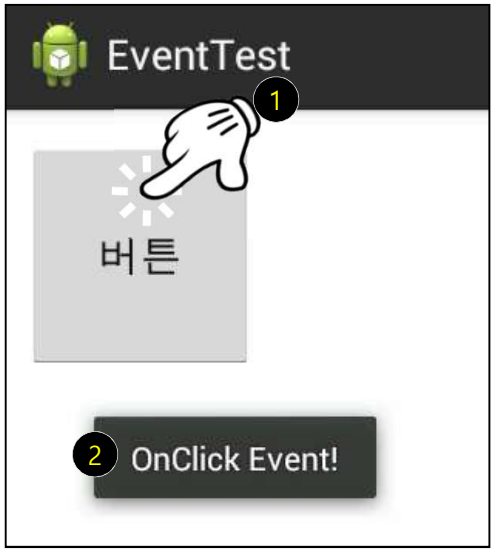

View.java

```
public class View ... {
    ...
    public interface OnClickListener {
        void onClick(View v);
    }
    ...
```

# 터치 이벤트 확장 – 클릭 리스너



```
// 1. 클릭 인터페이스를 액티비티가 View.OnClickListener 상속받아 구현한다.
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    // 2. 클릭 인터페이스 핸들러 함수 구현
    @Override
    public void onClick(View v) {
        switch(v.getId()) {
            case R.id.button_view:
                // 3. 버튼이 클릭되었을 때 토스트 팝업을 이용해 화면에 알림
                Toast.makeText( this, "OnClick Event!", Toast.LENGTH_LONG).show();
                break;
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button) findViewById(R.id.button_view);

        // 3. 버튼에 클릭 리스너 객체를 생성 및 설정한다.
        button.setOnClickListener(this);
    }
}
```



# 터치 이벤트 확장 – 클릭 리스너



- 클릭 리스너는 매우 많이 사용되므로 좀 더 용이한 방법 한 가지를 더 제공한다.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_margin="20px">

   <Button android:id="@+id/button_view"
      android:text="버튼"
      android:layout_width="100dp"
      android:layout_height="100dp"
      android:clickable="true"
      android:onClick="onMyButtonClick"/>

</LinearLayout>
```

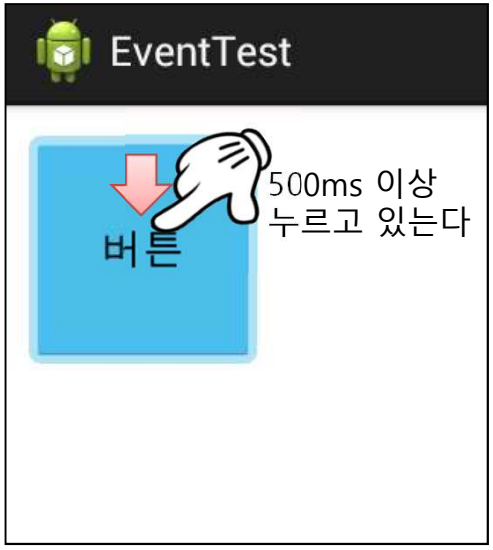
```
public class MainActivity extends AppCompatActivity {

    public void onMyButtonClick(View v) {
        switch(v.getId()) {
            case R.id.button_view:
                Toast.makeText( this, "OnClick Event!", Toast.LENGTH_LONG).show();
                break;
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

**View의 android:clickable 속성**: 이 속성은 클릭 이벤트 활성화 유무를 설정한다. 이 속성이 활성화되어야만 클릭 리스너가 호출된다. 버튼같은 특정 뷰들은 이 속성의 기본값이 true이다. 또한 클릭 리스너 등록 함수인 setOnClickListener를 사용해도 clickable 속성이 활성화된다.

# 터치 이벤트 확장 – 롱클릭 리스너



- 롱클릭 리스너는 뷰를 500ms 이상 누르고 있을 때 호출.
  - ACTION_DOWN 이벤트가 발생한 후 ACTION_UP이 500ms 이내에 오지 않으면 롱클릭 이벤트가 됨.




```
View.java
public class View ... {
    ...
    public interface OnLongClickListener {
        boolean onLongClick(View v);
    }
    ...
```

# 터치 이벤트 확장 – 롱클릭 리스너



```
//① 롱클릭 인터페이스를 액티비티가 View.OnLongClickListener 상속받아 구현한다.
public class MainActivity extends AppCompatActivity implements View.OnLongClickListener {

    // ② 롱클릭 인터페이스 핸들러 함수 구현
    @Override
    public boolean onLongClick(View v) {
        switch(v.getId()) {
            case R.id.button_view:
                Toast.makeText( this, "OnLongClick Event!", Toast.LENGTH_LONG).show();
                return true;
        }
        return false;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button) findViewById(R.id.button_view);

        button.setOnLongClickListener(this);
    }
}
```