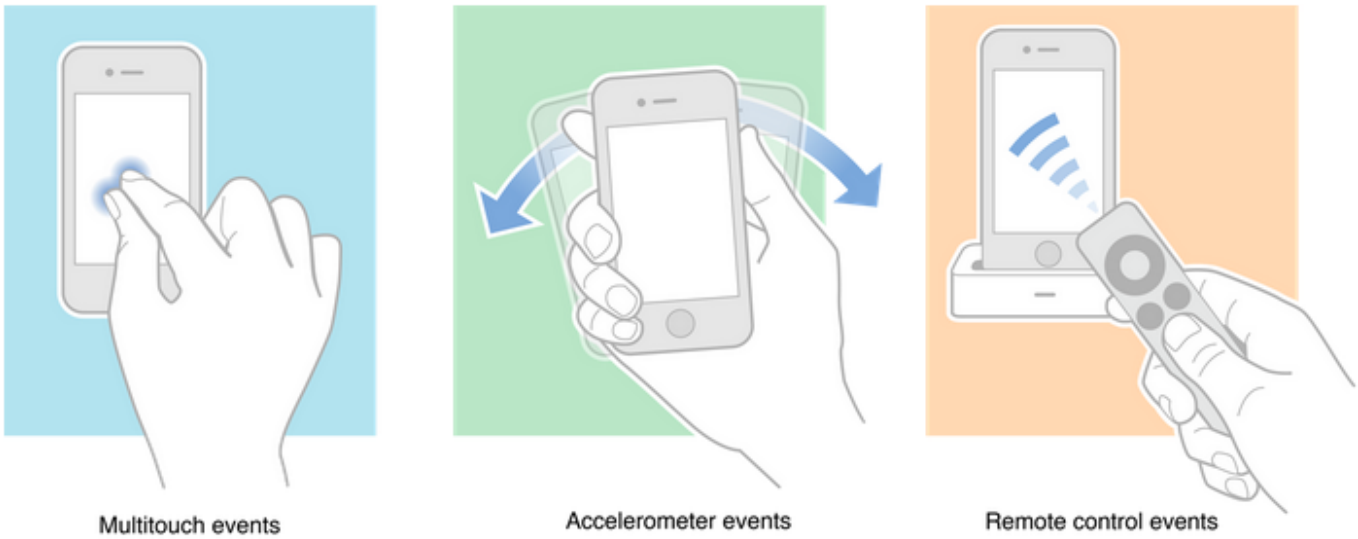
iOS事件机制(一)

Dec 7th, 2013

> 运用的前提是掌握
> 掌握的本质是理解



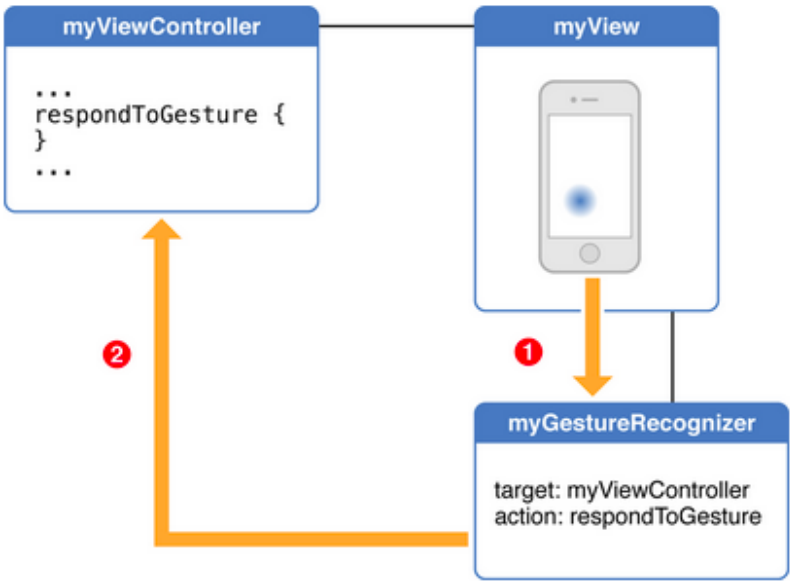
本篇内容将围绕iOS中事件及其传递机制进行学习和分析。在iOS中，事件分为三类：

- 触控事件（单点、多点触控以及各种手势操作）
- 传感器事件（重力、加速度传感器等）
- 远程控制事件（远程遥控iOS设备多媒体播放等）

这三类事件共同构成了iOS设备丰富的操作方式和使用体验，本次就首先来针对第一类事件：触控事件，进行学习和分析。

Gesture Recognizers

Gesture Recognizers是一类手势识别器对象，它可以附属在你指定的View上，并且为其设定指定的手势操作，例如是点击、滑动或者是拖拽。当触控事件 发生时，设置了Gesture Recognizers的View会先通过识别器去拦截触控事件，如果该触控事件是事先为View设定的触控监听事件，那么Gesture Recognizers将会发送动作消息给目标处理对象，目标处理对象则对这次触控事件进行处理，先看看如下流程图。



在iOS中，View就是我们在屏幕上看到的各种UI控件，当一个触控事件发生时，Gesture Recognizers会先获取到指定的事件，然后发送动作消息(action message)给目标对象(target)，目标对象就是ViewController，在ViewController中通过事件方法完成对该事件的处理。Gesture Recognizers能设置诸如单击、滑动、拖拽等事件，通过Action-Target这种设计模式，好处是能动态为View添加各种事件监听，而不用去实现一个View的子类去完成这些功能。

以上过程就是我们在开发中在方法中常见的设置action和设置target，例如为UIButton设置监听事件等。

常用手势识别类

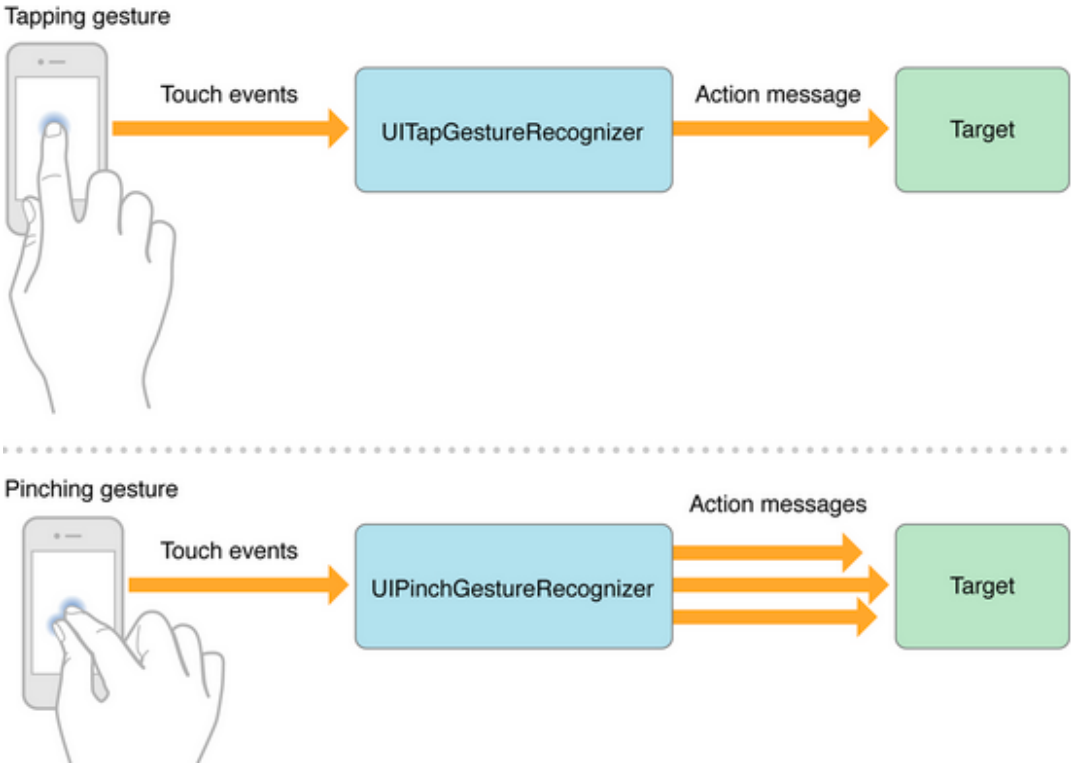
在UIKit框架中，系统为我们事先定义好了一些常用的手势识别器，包括点击、双指缩放、拖拽、滑动、旋转以及长按。通过这些手势识别器我们可以构造丰富的操作方式。

Gesture	UIKit class
Tapping (any number of taps)	`UITapGestureRecognizer`
Pinching in and out (for zooming a view)	`UIPinchGestureRecognizer`
Panning or dragging	`UIPanGestureRecognizer`
Swiping (in any direction)	`UISwipeGestureRecognizer`
Rotating (fingers moving in opposite directions)	`UIRotationGestureRecognizer`
Long press (also known as "touch and hold")	`UILongPressGestureRecognizer`

在上表中可以看到，UIKit框架中已经提供了诸如UITapGestureRecognizer在内的六种手势识别器，如果你需要实现自定义的手势识别器，也可以通过继承UIGestureRecognizer类并重写其中的方法来完成，这里我们就不详细讨论了。

每一个Gesture Recognizer关联一个View，但是一个View可以关联多个Gesture Recognizer，因为一个View可能还能响应多种触控操作方式。当一个触控事件发生时，Gesture Recognizer接收一个动作消息要先于View本身，结果就是Gesture Recognizer作为View处理触控事件的代表，或者叫代理。当Gesture Recognizer接收到指定的事件时，它就会发送一条动作消息(action message)给ViewController并处理。

连续和不连续动作



触控动作同时分为连续动作(continuous)和不连续动作(discrete)，连续动作例如滑动和拖拽，它会持续一小段时间，而不连续动作例如单击，它瞬间就会完成，在这两类事件的处理上又稍有不同。对于不连续动作，Gesture Recognizer只会给ViewContoller发送一个单一的动作消息(action message),而对于连续动作，Gesture Recognizer会发送多条动作消息给ViewController，直到所有的事件都结束。

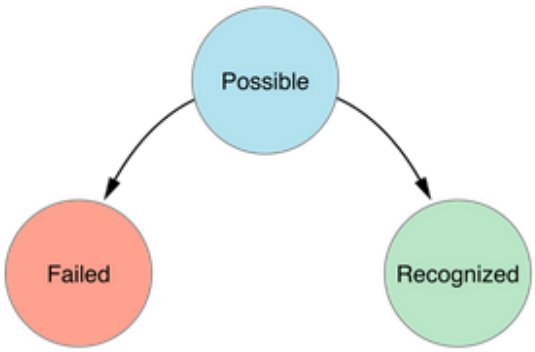
为一个View添加GestureRecognizer有两种方式，一种是通过InterfaceBuilder实现，另一种就是通过代码实现，我们看看通过代码来如何实现。

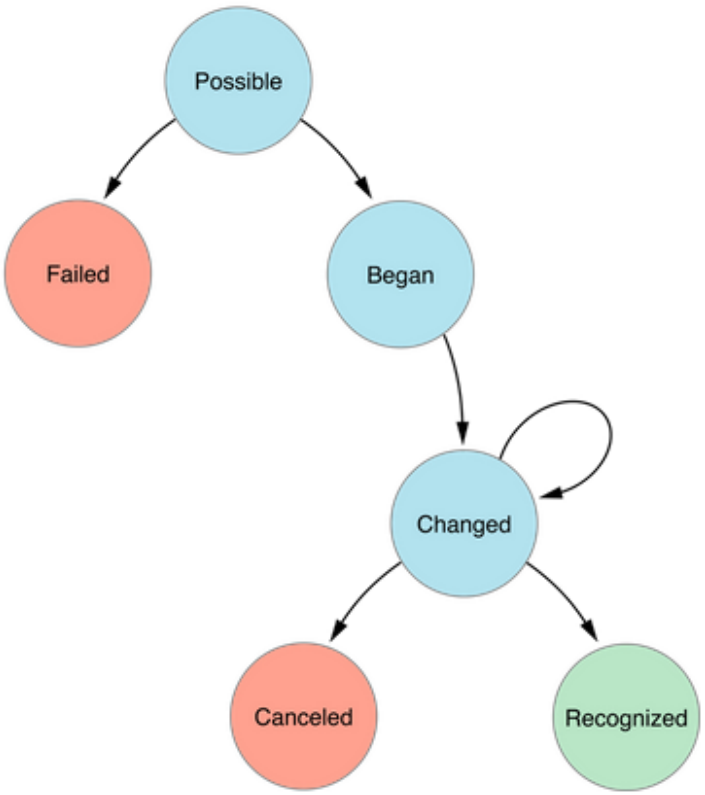
MyViewContoller.m

```
1  - (void)viewDidLoad {
2        [super viewDidLoad];
3
4        // 创建并初始化手势对象
5        UITapGestureRecognizer *tapRecognizer = [[UITapGestureRecognizer alloc]
6             initWithTarget:self action:@selector(respondToTapGesture:)];
7
8        // 指定操作为单击一次
9        tapRecognizer.numberOfTapsRequired = 1;
10
11       // 为当前View添加GestureRecognizer
12       [self.view addGestureRecognizer:tapRecognizer];
13
14       // ...
15 }
```

通过上述代码，我们实现了为当前MyViewController的View添加一个单击事件，首先构造了UITapGestureRecognizer对象，指定了target为当前ViewController本身，action就是后面自己实现的处理方法，这里就呼应了前文提到的Action-Target模式。

在事件处理过程中，这两种方式所处的状态又各有不同，首先，所有的触控事件最开始都是处于可用状态(Possible)，对应UIKit里面的UIGestureRecognizerStatePossible类，如果是不连续动作事件，则状态只会从Possible转变为已识别状态(Recognized,UIGestureRecognizerStateRecognized)或者是失败状态(Failed,UIGestureRecognizerStateFailed)。例如一次成功的单击动作，就对应了Possible-Recognized这个过程。

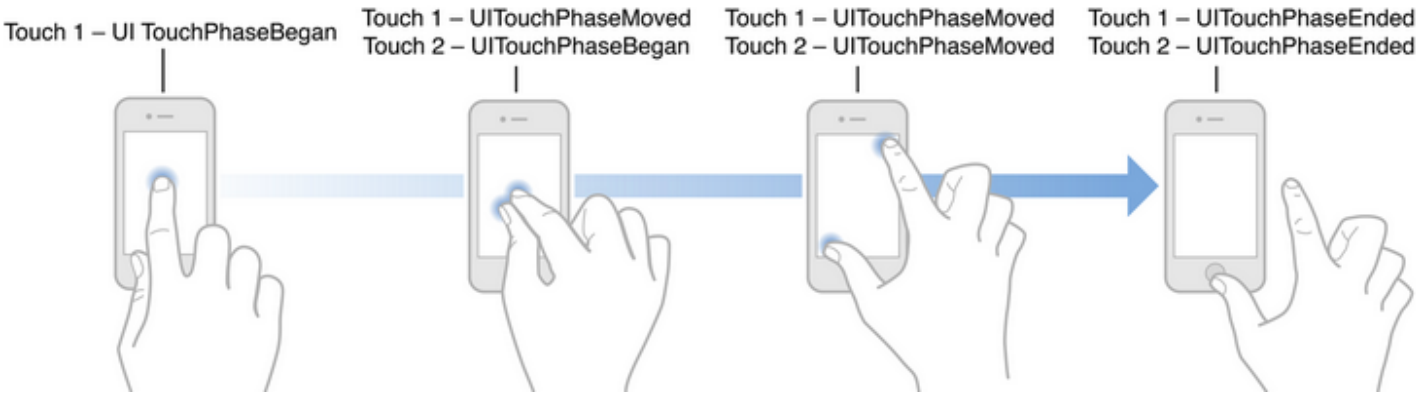




如果是连续动作事件，如果事件没有失败并且连续动作的第一个动作被成功识别(Recognized)，则从Possible状态转移到Began(UIGestureRecognizerStateBegan)状态，这里表示连续动作的开始，接着会转变为Changed(UIGestureRecognizerStateChanged)状态，在这个状态下会不断循环的处理连续动作，直到动作执行完成变转变为Recognized已识别状态，最终该动作会处于完成状态(UIGestureRecognizerStateEnded)，另外，连续动作事件的处理状态会从Changed状态转变为Canceled(UIGestureRecognizerStateCancelled)状态，原因是识别器认为当前的动作已经不匹配当初对事件的设定了。每个动作状态的变化，Gesture Recognizer都会发送消息(action message)给Target，也就是ViewController，它可以根据这些动作消息进行相应的处理。例如一次成功的滑动手势动作就包括按下、移动、抬起的过程，分别对应了Possible-Began-Changed-Recognized这个过程。

UITouch & UIEvent

在屏幕上的每一次动作事件都是一次Touch，在iOS中用UITouch对象表示每一次的触控，多个Touch组成一次Event，用UIEvent来表示一次事件对象。



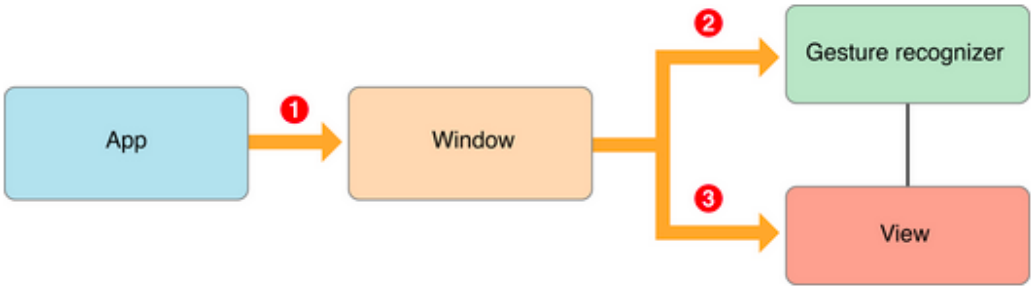
在上述过程中，完成了一次双指缩放的事件动作，每一次手指状态的变化都对应事件动作处理过程中得一个阶段。通过Began-Moved-Ended这几个阶段的动作(Touch)共同构成了一次事件(Event)。在事件响应对象UIResponder中有对应的方法来分别处理这几个阶段的事件。

- touchesBegan:withEvent:

- touchesMoved:withEvent:
- touchesEnded:withEvent:
- touchesCancelled:withEvent:

后面的参数分别对应UITouchPhaseBegan、UITouchPhaseMoved、UITouchPhaseEnded、UITouchPhaseCancelled这几个类。用来表示不同阶段的状态。

事件传递



如上图，iOS中事件传递首先从App(UIApplication)开始，接着传递到Window(UIWindow)，在接着往下传递到View之前，Window会将事件交给GestureRecognizer，如果在此期间，GestureRecognizer识别了传递过来的事件，则该事件将不会继续传递到View去，而是像我们之前说的那样交给Target(ViewController)进行处理。

响应者链(Responder Chain)

通常，一个iOS应用中，在一块屏幕上通常有很多的UI控件，也就是有很多的View，那么当一个事件发生时，如何来确定是哪个View响应了这个事件呢，接下来我们就一起来看看。

寻找hit-test view

什么是hit-test view呢？简单来说就是你触发事件所在的那个View，寻找hit-test view的过程就叫做Hit-Testing。那么，系统是如何来执行Hit-Testing呢，首先假设现在有如下这么一个UI布局，一种有ABCDE五个View。

假设一个单击事件发生在了View D里面，系统首先会从最顶层的View A开始寻找，发现事件是在View A或者其子类里面，那么接着从B和C找，发现事件是在C或者其子类里面，那么接着到C里面找，这时发现事件是在D里面，并且D已经没有子类了，那么hit-test view就是View D啦。

响应者对象(Responsder Object)

响应者对象是能够响应并且处理事件的对象，UIResponder是所有响应者对象的父类，包括UIApplication、UIView和UIViewController都是UIResponder的子类。也就意味着所有的View和ViewController都是响应者对象。

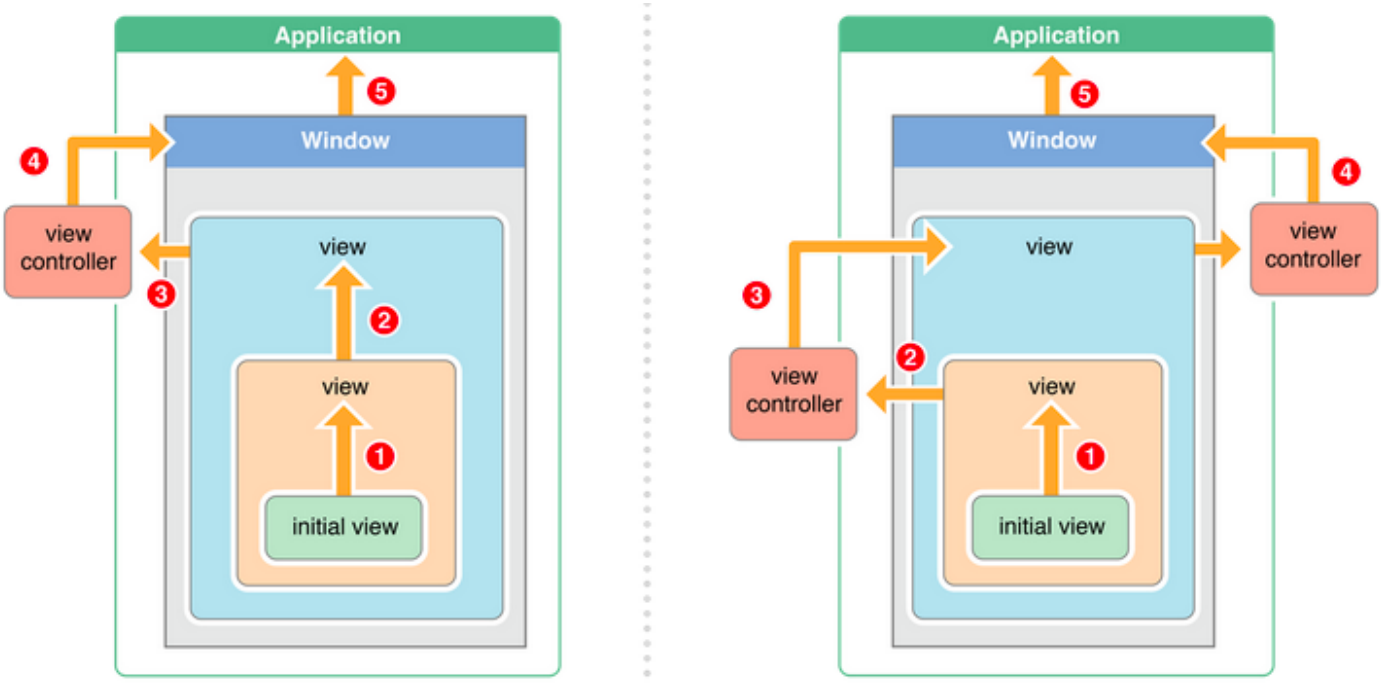
第一响应者(First Responder)

第一响应者是第一个接收事件的View对象，我们在Xcode的Interface Builder画视图时，可以看到视图结构中就有First Responder。

这里的First Responder就是UIApplication了。另外，我们可以控制一个View让其成为First Responder，通过实现 canBecomeFirstResponder方法并返回YES可以使当前View成为第一响应者，或者调用View的becomeFirstResponder方法也可以，例如当UITextField调用该方法时会弹出键盘进行输入，此时输入框控件就是第一响应者。

事件传递机制

如上所说，，如果hit-test view不能处理当前事件，那么事件将会沿着响应者链(Responder Chain)进行传递，知道遇到能处理该事件的响应者(Responsder Object)。通过下图，我们来看看两种不同情况下得事件传递机制。



左边的情况，接收事件的initial view如果不能处理该事件并且她不是顶层的View，则事件会往它的父

View进行传递。initial view的父View获取事件后如果仍不能处理，则继续往上传递，循环这个过程。如果顶层的View还是不能处理这个事件的话，则会将事件传递给它们的ViewController，如果ViewController也不能处理，则传递给Window(UIWindow)，此时Window不能处理的话就将事件传递给Application(UIApplication)，最后如果连Application也不能处理，则废弃该事件。

右边图的流程唯一不同就在于，如果当前的ViewController是由层级关系的，那么当子ViewController不能处理事件时，它会将事件继续往上传递，直到传递到其Root ViewController，后面的流程就跟之前分析的一样了。

这就是事件响应者链的传递机制，通过这些内容，我们可以更深入的了解事件在iOS中得传递机制，对我们在实际开发中更好的理解事件操作的原理有很大的帮助，也对我们实现复杂布局进行事件处理时增添了多一份的理解。

总结

通过前面的内容分析，我们已经学习并了解了如下内容：

- Gesture Recognizers，是用来控制手势识别的过程和方法，并且其通过Action-Target模式与ViewController的通信的方式。连续和不连续手势动作情况下GestureRecognizer的状态转变。
- UITouch和UIEvent对象，他们都是UIKit中来进行事件处理的对象，多个UITouch对象构成一个UIEvent对象，重写相应的方法可以控制和处理事件各个阶段的操作。
- 系寻找hit-test view的方式、事件传递机、制响应者链

后记：本篇是iOS事件传递机制的上篇，下篇将继续讨论多点触控事件和手势操作的内容！

原文地址： http://ryantang.me/blog/2013/12/07/ios-event-dispatch-1/