超出父视图事件响应

1. 概述：

   通常来说，一个控件超出了父视图之后，它就失去了响应事件的能力了，为什么？这和事件的响应链有关系，其实，要想让控件在父视图范围外也响应事件，只要干预下响应链的相关响应事件逻辑即可。

2. 实现

   非常简单，代码如下

```
#pragma mark - 决定由哪个视图来响应事件
- (UIView *)hitTest:(CGPoint)point withEvent:(UIEvent *)event{
  if (!self.isUserInteractionEnabled || self.isHidden || self.alpha <= 0.01) {
    return nil;
  }

  if ([self pointInside:point withEvent:event]) {//第一步：判断点在自己身上
    for (UIView *subview in [self.subviews reverseObjectEnumerator]) {//第二步：决定让那个子视图或自己来响应事件
      CGPoint convertedPoint = [subview convertPoint:point fromView:self];
      UIView *hitTestView = [subview hitTest:convertedPoint withEvent:event];
      if ([hitTestView isKindOfClass:[MYImageButton class]]) {//优先让目标子视图响应事件
        return hitTestView;
      }else{
        if(hitTestView){
          return hitTestView;
        }
      }
    }
    return self;//子视图无响应，自己响应事件
  }else{
    return nil;//点不在自己身上，自己不响应
  }
}

#pragma mark - 触摸点是否在自己身上(不重写的话，点击超出父视图时结果都为false，重写目的是只要点有在子视图身上，就算超出自己范围也算点了自己)
- (BOOL)pointInside:(CGPoint)point withEvent:(UIEvent *)event{
  NSArray *subViews = self.subviews;
  if ([subViews count] > 1)
  {
    for (UIView *aSubView in subViews)
    {
      if ([aSubView pointInside:[self convertPoint:point toView:aSubView] withEvent:event])
      {
        return YES;
      }
    }
  }

  return [super pointInside:point withEvent:event];
```

在父类重写下这两个方法，最下面的方法是判断触摸点是否在自己范围内，系统判断由哪个子视图响应事件的逻辑默认是：通过下面的方法判断点是否在自己身上，如果返回为 true 的话，那么父视图就会问自己的子视图看谁能处理此次事件响应。也就是说，第一个方法是猜想系统默认实现写的，注释掉后发现，仍然 ok，说明系统默认可能也是这种实现逻辑。