**iOS9 3D Touch 标签菜单 peek and pop  force 压力触控 功能开发高级版教程**

**教程地址**

**http://blog.csdn.net/u011363981/article/details/50589754**

 3D Touch  Demo  百度网盘<http://pan.baidu.com/s/1Mtaj0>

github地址： <https://github.com/wyk125/3D-Touch-Demo.git>  觉得不错，记得Start哦，谢过！

前言

技术的本身并不是难点，而是怎样优雅地把这些 技术平滑的融入到我们的app，我们的工作，我们的生活，才是学习之本，学习之道。

3D Touch 标签 视频演示（6s，6Ps且iOS9以上支持3D Touch，越狱的不在此列）：

<http://pan.baidu.com/s/1mhd8FYK>

3D Touch peek and pop 视频演示

<http://player.ku6.com/refer/pTdE_6IsnauaoQ_xoqaoiQ../v.swf>

在Mac电脑上为iPhone或iPad录屏的方法 需要的请get  <http://www.weste.net/2015/04-07/102560.html>

app启动过程  <http://blog.csdn.net/u011363981/article/details/50589880>

3D Touch  苹果官方文档

<https://developer.apple.com/library/prerelease/ios/documentation/UserExperience/Conceptual/MobileHIG/3DTouch.html#//apple_ref/doc/uid/TP40006556-CH71>

**3D Touch的主要应用**

       3D Touch给用户带来的额外维度上的交互，这个设计之所以叫做3D Touch，其原理上是增加了一个压力的感触，通过区分轻按和重按来进行不同的用户交互。官方文档给出的应用介绍主要有两块：

第一部分的应用是我们可以通过3D手势，在主屏幕上的应用Icon处，直接进入应用的响应功能模块。这个功能就例如我们上面的日历示例，会在Icon旁边出现一个菜单，点击菜单我们可以进入相应的功能单元。3D Touch功能，push消息功能加上iOS8推出的扩展today功能，这三个机制使iOS应用变得无比灵活方便，用户可以不需付出寻找的时间成本来快速使用自己需要的功能。

第二部分是对app的一个优化，用户可以通过3D Touch手势在view上来预览一些预加载信息，这样的设计可以使app更加简洁大方，交互性也更强。

在模拟器上学习和测试3D Touch（目前这个方法不一定行的通，可以试下，后续更新）

     3D Touch是一个很新颖的设计，可是苹果文档说，只有xcode7以上才支持3D Touch开发的，模拟器并不支持这个压力手势，我们只能在真机上进行学习与测试，但是在IT的世界，从来都不缺拯救世界的人物，github上有人为我们提供了这样的一个插件，可以让我们在模拟器上进行3D Touch的效果测试：

git地址：<https://github.com/DeskConnect/SBShortcutMenuSimulator>

附.SBShortcutMenuSimulator的安装和使用

        其实安装和使用并不需要怎么介绍，git主页里介绍的很清楚，这里在记录一遍，其中只有一点需要注意，如果你的电脑中装有Xcode6和Xcode7两个版本，那个Xcode的编译路径，需要做一下修改。

安装：

在终端中一次运行如下指令：

[?](http://my.oschina.net/u/2340880/blog/511509?fromerr=lTrt5rgh)

1

2

3

git clone https://github.com/DeskConnect/SBShortcutMenuSimulator.git

|  |
| --- |
| cd SBShortcutMenuSimulator  make |

如果电脑中有多个Xcode版本，先做如下操作，如果只有Xcode7，则可以跳过

[?](http://my.oschina.net/u/2340880/blog/511509?fromerr=lTrt5rgh)

1

|  |
| --- |
| sudo xcode-select -switch /Applications/Xcode2.app/Contents/Developer/ |

注意：上面命令中，Xcode2.app是你电脑中Xcode的名字，这里如要特别注意，如果名字中有空格，需要修改一下，把空格去掉，否则会影响命令的执行。

之后在SBShortcutMenuSimulator的目录中执行如下操作：

[?](http://my.oschina.net/u/2340880/blog/511509?fromerr=lTrt5rgh)

1

2

xcrun simctl spawn booted launchctl debug system/com.apple.SpringBoard --environment DYLD\_INSERT\_LIBRARIES=$PWD/SBShortcutMenuSimulator.dylib

xcrun simctl spawn booted launchctl stop com.apple.SpringBoard

如果没有报错，我们可以通过向指定端口发送消息的方法来在模拟器上模拟3D Touch的效果：

[?](http://my.oschina.net/u/2340880/blog/511509?fromerr=lTrt5rgh)

1

|  |
| --- |
| echo 'com.apple.mobilecal' | nc 127.0.0.1 8000 |

其中，com.apple.mobilecal是应用的Bundle ID ，如果要测试我们的应用，将其改为我们应用的BundleID即可，可以看到模拟器的效果如下：

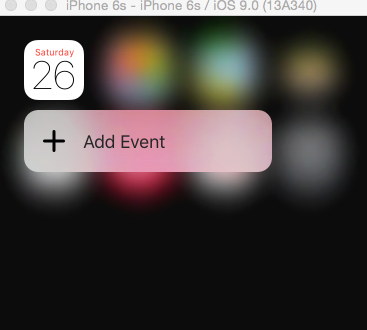
[](http://static.oschina.net/uploads/space/2015/0926/110201_r1gd_2340880.png)

                                图1

我试了几次 却显示如下图2所示，后面找到解决思路，或有更好的方法 ，我会第一时间分享。

 图 2

**3D Touch的三大模块**

     在我们的app中使用3D Touch功能，主要分为以下三个模块：

Home Screen Quick Actions

   通过主屏幕的应用Icon，我们可以用3D Touch呼出一个菜单，进行快速定位应用功能模块相关功能的开发。如图3，图4所示。

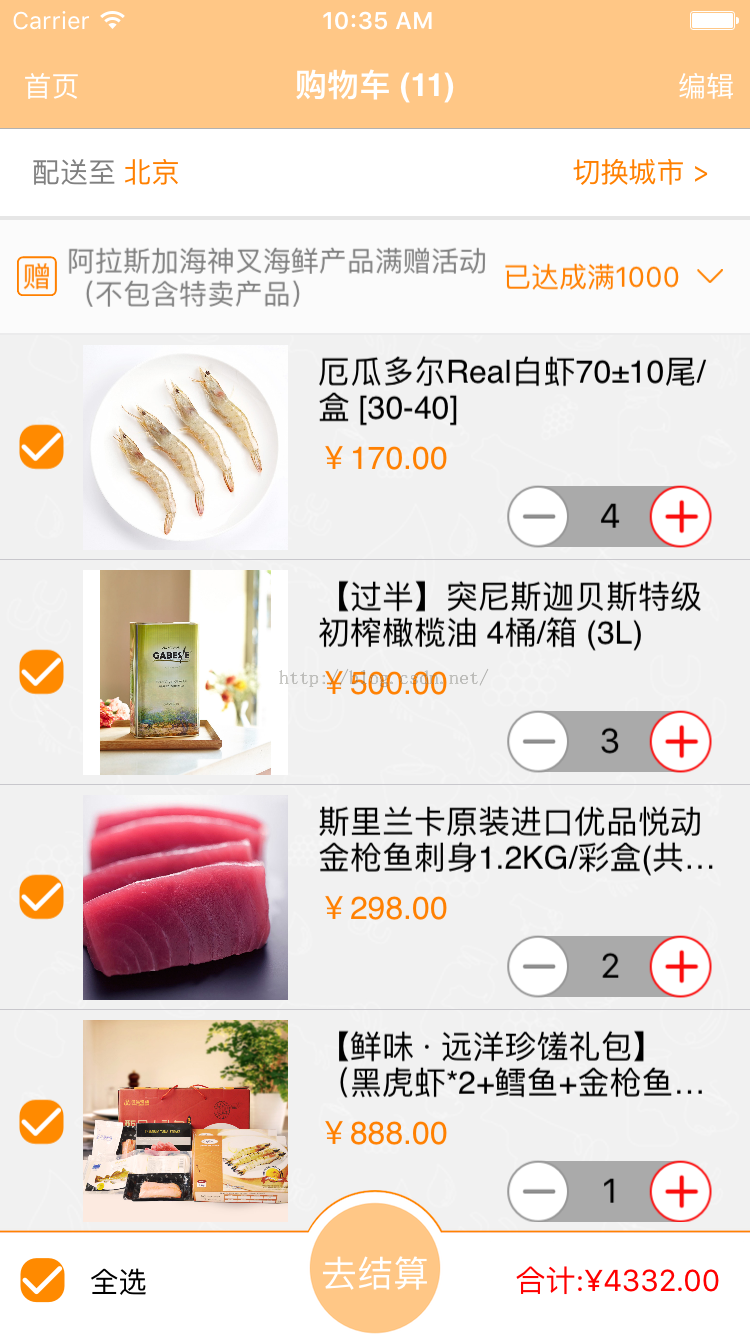
  

                图 3                                                                                                                                          图4

peek and pop

        这个功能是一套全新的用户交互机制，在使用3D Touch时，ViewController中会有如下三个交互阶段：

        （1）提示用户这里有3D Touch的交互，会使交互控件周围模糊  如 图5



                 图 5

        （2）继续深按，会出现预览视图 ，图  6

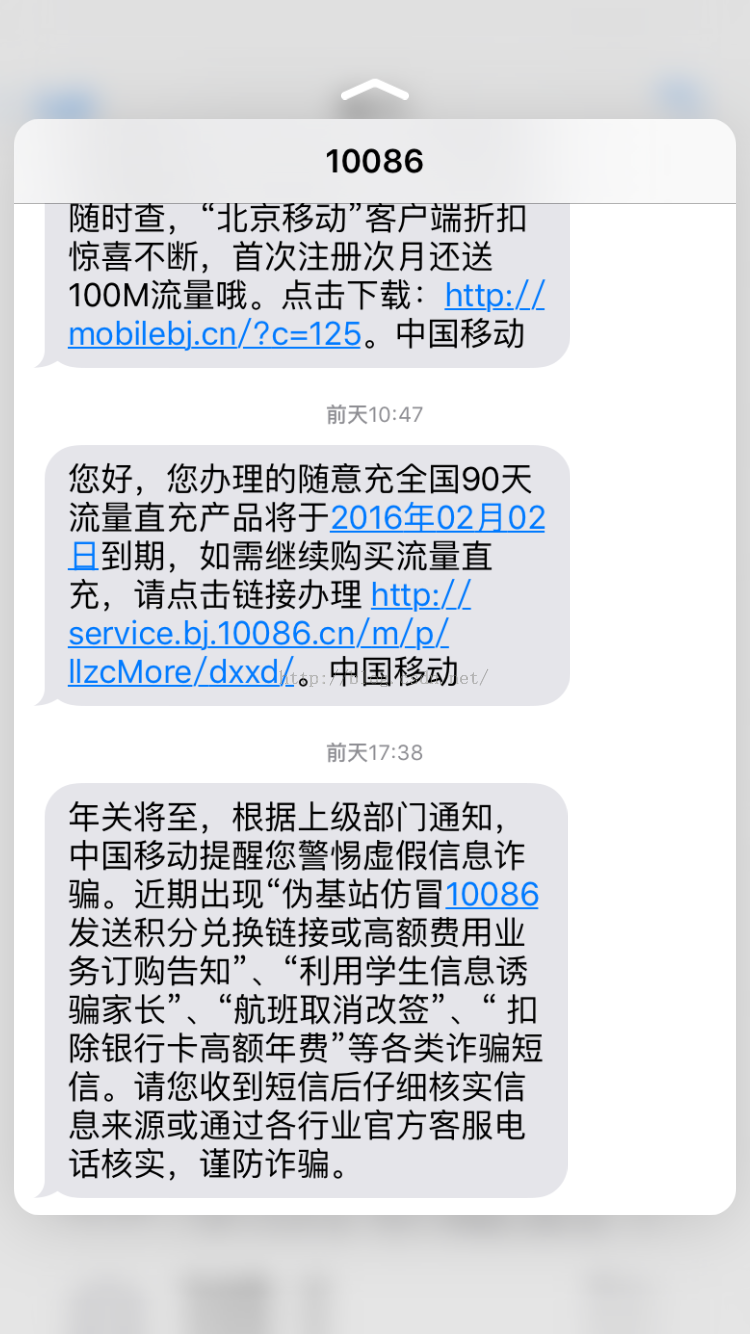


        图6

   （3）通过视图上的交互控件进行进一步交互  如 图7

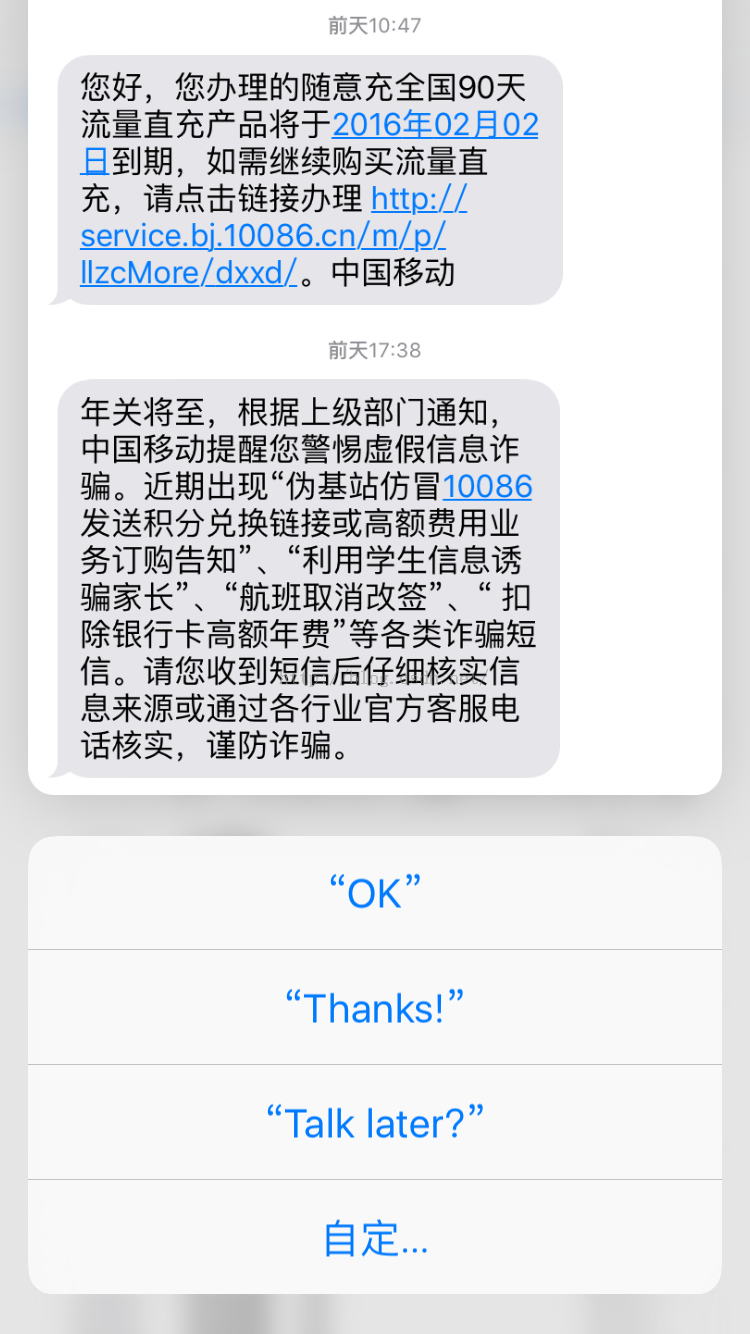


                       图 7

这个模块的设计可以在网址连接上进行网页的预览交互。

Force Properties

        iOS9为我们提供了一个新的交互参数:力度。我们可以检测某一交互的力度值，来做相应的交互处理。例如，我们可以通过力度来控制快进的快慢，音量增加的快慢等。

**3D Touch 的Home Screen Quick Action 使用与相关api详解**

 iOS9为我们提供了两种屏幕标签，分别是静态标签和动态标签。两点注意：

1、快捷标签最多可以创建四个，包括静态的和动态的。

2、每个标签的题目和icon最多两行，多出的会用...省略

**静态标签**

    静态标签是我们在项目的配置plist文件中配置的标签，在用户安装程序后就可以使用，并且排序会在动态标签的前面。

我们先来看静态标签的配置：

必填项（下面两个键值是必须设置的）：

UIApplicationShortcutItemType 这个键值设置一个快捷通道类型的字符串

UIApplicationShortcutItemTitle 这个键值设置标签的标题

选填项（下面这些键值不是必须设置的）：

UIApplicationShortcutItemSubtitle 设置标签的副标题

UIApplicationShortcutItemIconType 设置标签Icon类型

UIApplicationShortcutItemIconFile  设置标签的Icon文件

UIApplicationShortcutItemUserInfo 设置信息字典(用于传值)

首先，找到项目的info.plist文件，右击文件，再将鼠标放在 Open As  上，点击 Source  Code。

以XML格式打开 info.plist文件 ，插入以下  代码 即可。或者你也可以直接在plist 文件中，直接键入key值再 设置。

静态标签的设置顺序和显示顺序 是相反的，进栈出栈。

**[html]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. **<array>**
2. **<span** style="white-space:pre"**>**    **</span><dict>**
3. **<key>**UIApplicationShortcutItemIconType**</key>** <!--注释行    -->
4. **<string>**UIApplicationShortcutIconTypeDate**</string>**
5. **<key>**UIApplicationShortcutItemIconFile**</key>** <!--自定义图片文件 替代不理想的系统图标   -->
6. **<string>**tabbar\_shoppingcart**</string>**    <!--图片名称 -->
7. **<span** style="white-space:pre"**>** **</span>** **<span** style="white-space:pre"**>**  **</span><key>**UIApplicationShortcutItemTitle**</key>**
8. **<span** style="white-space:pre"**>**        **</span><string>**购物车**</string>**
9. **<span** style="white-space:pre"**>**        **</span><key>**UIApplicationShortcutItemType**</key>**
10. **<span** style="white-space:pre"**>**        **</span><string>**YPYD.UITouchText.cart**</string>**
11. **<span** style="white-space:pre"**>**    **</span></dict>**
12. **<span** style="white-space:pre"**>**    **</span><dict>**
13. **<span** style="white-space:pre"**>**        **</span><key>**UIApplicationShortcutItemIconType**</key>**    <!--  标签的类型key   -->
14. **<span** style="white-space:pre"**>**        **</span><string>**UIApplicationShortcutIconTypeSearch**</string>**  <!--  标签的类型vale   -->
15. **<span** style="white-space:pre"**>**        **</span><key>**UIApplicationShortcutItemTitle**</key>**    <!--  必写标签的标题key   -->
16. **<span** style="white-space:pre"**>**        **</span><string>**搜索商品**</string>**                       <!--  必写标签的标题vale   -->
17. **<span** style="white-space:pre"**>**        **</span><key>**UIApplicationShortcutItemType**</key>**  <!--必写  标签的唯一标识Key 自由定义的一个字符串-->
18. **<span** style="white-space:pre"**>**        **</span><string>**YPYD.UITouchText.search**</string>**    <!--  必写标签的唯一标识vale -->
19. **<span** style="white-space:pre"**>**    **</span></dict>**
20. **<span** style="white-space:pre"**>**    **</span><dict>**
21. **<span** style="white-space:pre"**>**        **</span><key>**UIApplicationShortcutItemIconType**</key>**
22. **<span** style="white-space:pre"**>**        **</span><string>**UIApplicationShortcutIconTypeHome**</string>**
23. **<span** style="white-space:pre"**>**        **</span><key>**UIApplicationShortcutItemTitle**</key>**
24. **<span** style="white-space:pre"**>**        **</span><string>**首页**</string>**
25. **<span** style="white-space:pre"**>**        **</span><key>**UIApplicationShortcutItemType**</key>**
26. **<span** style="white-space:pre"**>**        **</span><string>**YPYD.UITouchText.home**</string>**
27. **<span** style="white-space:pre"**>**    **</span></dict>**
28. **</array>**

如果你的手机支持 3D Touch 真机运行程序，会有图3所示的效果。

**动态标签**

动态标签是我们在程序中，通过代码添加的，与之相关的类，主要有三个：

UIApplicationShortcutItem 创建3DTouch标签的类

UIMutableApplicationShortcutItem 创建可变的3DTouch标签的类

UIApplicationShortcutIcon 创建标签中图片Icon的类

因为这些类是iOS9中新增加的类，所以其api的复杂程度并不大，下面我们来对其中方法与属性进行简要讲解：

**UIApplicationShortcutItem**

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. **@interface** UIApplicationShortcutItem : NSObject <NSCopying, NSMutableCopying>
2. //下面是两个初始化方法 通过设置type，title等属性来创建一个标签，这里的icon是UIApplicationShortcutIcon对象，我们后面再说
3. - (instancetype)initWithType:(**NSString** \*)type localizedTitle:(**NSString** \*)localizedTitle localizedSubtitle:(**nullableNSString** \*)localizedSubtitle icon:(**nullableUIApplicationShortcutIcon** \*)icon userInfo:(**nullableNSDictionary** \*)userInfoNS\_DESIGNATED\_INITIALIZER;
4. - (instancetype)initWithType:(**NSString** \*)type localizedTitle:(**NSString** \*)localizedTitle;
5. //下面这是一些只读的属性，获取相应的属性值
6. **@property** (**nonatomic**,**copy**,**readonly**)**NSString** \*type;
7. **@property** (**nonatomic**,**copy**,**readonly**)**NSString** \*localizedTitle;
8. **@property** (nullable,**nonatomic**,**copy**,**readonly**)**NSString** \*localizedSubtitle;
9. **@property** (nullable,**nonatomic**,**copy**,**readonly**)**UIApplicationShortcutIcon** \*icon;//系统定义的图标
10. **@property** (nullable,**nonatomic**,**copy**,**readonly**)NSDictionary<**NSString** \*,**id** <NSSecureCoding>> \*userInfo;

**UIMutableApplicationShortcutItem**

//这个类继承于 UIApplicationShortcutItem，创建的标签可变

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. **@interface** UIMutableApplicationShortcutItem : UIApplicationShortcutItem
2. **@property** (**nonatomic**,**copy**) **NSString** \*type;
3. **@property** (**nonatomic**,**copy**)**NSString** \*localizedTitle;
4. **@property** (nullable,**nonatomic**,**copy**)**NSString** \*localizedSubtitle;
5. **@property** (nullable,**nonatomic**,**copy**)**UIApplicationShortcutIcon** \*icon;
6. **@property** (nullable,**nonatomic**,**copy**)NSDictionary<**NSString** \*,**id** <NSSecureCoding>> \*userInfo;
7. **@end**

**UIApplicationShortcutIcon**

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. //这个类创建标签中的icon
2. **@interface** UIApplicationShortcutIcon : NSObject <NSCopying>
3. //创建系统风格的icon
4. + (instancetype)iconWithType:(UIApplicationShortcutIconType)type;
5. //创建自定义的图片icon
6. + (instancetype)iconWithTemplateImageName:(**NSString** \*)templateImageName;
7. **@end**

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. /\*\* 创建shortcutItems \*/
2. - (**void**)configShortCutItems {
3. **NSMutableArray** \*shortcutItems = [NSMutableArray array];
5. **UIApplicationShortcutItem** \*item1 = [[UIApplicationShortcutItem alloc]initWithType:@"YPYD.UITouchText.home" localizedTitle:@"测试1" localizedSubtitle**:nil** icon:[UIApplicationShortcutIcon iconWithType:UIApplicationShortcutIconTypeHome] userInfo**:nil**];
7. **UIApplicationShortcutItem** \*item2 = [[UIApplicationShortcutItem alloc]initWithType:@"YPYD.UITouchText.search" localizedTitle:@"测试2" localizedSubtitle:@"测试2副标题" icon:[UIApplicationShortcutIcon iconWithType:UIApplicationShortcutIconTypeSearch]
8. userInfo**:nil**];
9. [shortcutItems addObject:item2];
10. [shortcutItems addObject:item1];
11. [[UIApplication sharedApplication] setShortcutItems:shortcutItems];
12. }

创建好标签后，将其添加如application的hortcutItems数组中即可，示例如图 8：

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. </pre><p style="margin-top:0px; margin-bottom:0px; font-size:14px; line-height:normal; font-family:Menlo"></p><p style="color:rgb(51,51,51); font-family:Menlo; font-size:14px; margin-top:0px; margin-bottom:0px; line-height:normal"></p><div style="color:rgb(51,51,51); font-family:Verdana,sans-serif,宋体; font-size:16px"></div><p style="margin-top:0px; margin-bottom:0px; line-height:normal"><img src="http://img.blog.csdn.net/20160128105549280?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center" width="375" height="667" alt="" /></p><p style="color:rgb(51,51,51); font-family:Helvetica; font-size:16px; margin-top:0px; margin-bottom:0px; line-height:normal"><img src="" alt="" /></p><p style="color:rgb(51,51,51); font-family:Helvetica; font-size:16px; margin-top:0px; margin-bottom:0px; line-height:normal">图 8 </p><p style="color:rgb(51,51,51); font-family:Helvetica; font-size:16px; margin-top:0px; margin-bottom:0px; line-height:normal"><span style="color:rgb(51,51,51); font-family:Verdana,sans-serif,宋体; font-size:16px; letter-spacing:0.5px">这里，将系统风格icon的枚举列举如下：</span></p><p style="color:rgb(51,51,51); font-family:Helvetica; font-size:16px; margin-top:0px; margin-bottom:0px; line-height:normal"><span style="color:rgb(51,51,51); font-family:Verdana,sans-serif,宋体; font-size:16px; letter-spacing:0.5px"></span><pre name="code" **class**="objc">**typedef** NS\_ENUM(NSInteger, UIApplicationShortcutIconType) {
2. UIApplicationShortcutIconTypeCompose,
3. UIApplicationShortcutIconTypePlay,
4. UIApplicationShortcutIconTypePause,
5. UIApplicationShortcutIconTypeAdd,
6. UIApplicationShortcutIconTypeLocation,
7. UIApplicationShortcutIconTypeSearch,
8. UIApplicationShortcutIconTypeShare,
9. UIApplicationShortcutIconTypeProhibit       NS\_ENUM\_AVAILABLE\_IOS(9\_1),
10. UIApplicationShortcutIconTypeContact        NS\_ENUM\_AVAILABLE\_IOS(9\_1),
11. UIApplicationShortcutIconTypeHome           NS\_ENUM\_AVAILABLE\_IOS(9\_1),
12. UIApplicationShortcutIconTypeMarkLocation   NS\_ENUM\_AVAILABLE\_IOS(9\_1),
13. UIApplicationShortcutIconTypeFavorite       NS\_ENUM\_AVAILABLE\_IOS(9\_1),
14. UIApplicationShortcutIconTypeLove           NS\_ENUM\_AVAILABLE\_IOS(9\_1),
15. UIApplicationShortcutIconTypeCloud          NS\_ENUM\_AVAILABLE\_IOS(9\_1),
16. UIApplicationShortcutIconTypeInvitation     NS\_ENUM\_AVAILABLE\_IOS(9\_1),
17. UIApplicationShortcutIconTypeConfirmation   NS\_ENUM\_AVAILABLE\_IOS(9\_1),
18. UIApplicationShortcutIconTypeMail           NS\_ENUM\_AVAILABLE\_IOS(9\_1),
19. UIApplicationShortcutIconTypeMessage        NS\_ENUM\_AVAILABLE\_IOS(9\_1),
20. UIApplicationShortcutIconTypeDate           NS\_ENUM\_AVAILABLE\_IOS(9\_1),
21. UIApplicationShortcutIconTypeTime           NS\_ENUM\_AVAILABLE\_IOS(9\_1),
22. UIApplicationShortcutIconTypeCapturePhoto   NS\_ENUM\_AVAILABLE\_IOS(9\_1),
23. UIApplicationShortcutIconTypeCaptureVideo   NS\_ENUM\_AVAILABLE\_IOS(9\_1),
24. UIApplicationShortcutIconTypeTask           NS\_ENUM\_AVAILABLE\_IOS(9\_1),
25. UIApplicationShortcutIconTypeTaskCompleted  NS\_ENUM\_AVAILABLE\_IOS(9\_1),
26. UIApplicationShortcutIconTypeAlarm          NS\_ENUM\_AVAILABLE\_IOS(9\_1),
27. UIApplicationShortcutIconTypeBookmark       NS\_ENUM\_AVAILABLE\_IOS(9\_1),
28. UIApplicationShortcutIconTypeShuffle        NS\_ENUM\_AVAILABLE\_IOS(9\_1),
29. UIApplicationShortcutIconTypeAudio          NS\_ENUM\_AVAILABLE\_IOS(9\_1),
30. UIApplicationShortcutIconTypeUpdate         NS\_ENUM\_AVAILABLE\_IOS(9\_1)
31. }

如果上述系统默认图标没你想要的 ，你还可以调用这个方法 手动 自定义图标。

+ (instancetype)iconWithTemplateImageName:(NSString \*)templateImageName;

如果上述系统默认图标没你想要的 ，你还可以调用这个方法 手动 自定义图标。

**响应标签的行为**

类似推送，当我们点击标签进入应用程序时，也可以进行一些操作，我们可以看到，在applocation中增加了这样一个方法：

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. - (**void**)application:(**UIApplication** \*)application performActionForShortcutItem:(**UIApplicationShortcutItem** \*)shortcutItem completionHandler:(**void**(^)(**BOOL** succeeded))completionHandler NS\_AVAILABLE\_IOS(9\_0);

当我们通过标签进入app时，就会在appdelegate中调用这样一个回调，我们可以获取shortcutItem的信息进行相关逻辑操作。

这里有一点需要注意：我们在app的入口函数：

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. - (**BOOL**)application:(**UIApplication** \*)application didFinishLaunchingWithOptions:(**NSDictionary** \*)launchOptions;

也需要进行一下判断，在launchOptions中有UIApplicationLaunchOptionsShortcutItemKey这样一个键，通过它，我们可以区别是否是从标签进入的app，如果是则处理结束逻辑后，返回NO，防止处理逻辑被反复回调。

**实现3D Touch 标签 跳转的参考方法**

因为app的启动会首先加载

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. - (**BOOL**)application:(**UIApplication** \*)application didFinishLaunchingWithOptions:(**nullableNSDictionary** \*)launchOptions ;

即为app启动，并已存在self.window 和 UITabBarController 视图控制器，因此，你可以通过 通知的方式来实现

performActionForShortcutItem 的 相关 跳转显示。

// iOS9 的 3D Touch 响应通知

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. - (**void**)application:(**UIApplication** \*)application performActionForShortcutItem:(**UIApplicationShortcutItem** \*)shortcutItem completionHandler:(**void**(^)(**BOOL** succeeded))completionHandler
2. {
3. **if** ([[[UIDevice currentDevice] systemVersion] floatValue] >= 9.0 &&**self**.window.traitCollection.forceTouchCapability == UIForceTouchCapabilityAvailable)
4. {
5. YPYDLog(@"你的手机支持3D Touch!");
6. **YPYDNetCountManager** \* sharedNetCountManager = [YPYDNetCountManager sharedNetCountManager];
7. sharedNetCountManager.applicationShortcutItemTitle = shortcutItem.type;

10. //首页
11. **if**([shortcutItem.type isEqualToString:@"YPYD.UITouchText.home"])
12. {
13. [[NSNotificationCenter defaultCenter] postNotificationName:@"YPYD.UITouchText.home" object**:nil** userInfo**:nil**];
14. }
15. //搜索商品
16. **if**([shortcutItem.type isEqualToString:@"YPYD.UITouchText.search"])
17. {
18. [[NSNotificationCenter defaultCenter] postNotificationName:@"YPYD.UITouchText.search" object**:nil** userInfo**:nil**];
19. }
20. //购物车
21. **if**([shortcutItem.type isEqualToString:@"YPYD.UITouchText.cart"])
22. {
23. [[NSNotificationCenter defaultCenter] postNotificationName:@"YPYD.UITouchText.cart" object**:nil** userInfo**:nil**];
24. }
25. //我的U
26. **if**([shortcutItem.type isEqualToString:@"YPYD.UITouchText.myU"])
27. {
28. [[NSNotificationCenter defaultCenter] postNotificationName:@"YPYD.UITouchText.home" object**:nil** userInfo**:nil**];
29. }
30. }
31. **else**
32. {
33. YPYDLog(@"你的手机暂不支持3D Touch!");
34. }
35. }

// 3D Touch 具体通知响应

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. - (**void**)applicationShortcutItemResponse
2. {
3. **YPYDNetCountManager** \* sharedNetCountManager = [YPYDNetCountManagersharedNetCountManager];
4. // 配合系统 判断是否已经完成登录方法 主要用于，app处于关闭状态时，先登录账号，再进入相关功能页面
5. //    YPYDLog(@"netGoodsListCount  %d", sharedNetCountManager.hadAutoLogin);
6. //    if (!sharedNetCountManager.hadAutoLogin)
7. //    {
8. //        return;
9. //    }
10. //    else
12. //首页
13. **if**([ sharedNetCountManager.applicationShortcutItemTitleisEqualToString:@"YPYD.UITouchText.home"])
14. {
15. **self**.selectedIndex =0;
16. [[NSNotificationCenterdefaultCenter]postNotificationName:@"UITouchText.home"object**:nil**userInfo**:nil**];
17. }
18. //搜索商品
19. **if**([ sharedNetCountManager.applicationShortcutItemTitleisEqualToString:@"YPYD.UITouchText.search"])
20. {
21. **self**.selectedIndex =0;
23. **YPYDSearchController** \* searchController = [[YPYDSearchControlleralloc]init];
24. searchController.navigationItem.title =@"搜索";
25. YPYDLog(@"self.selectedViewController  %@",**self**.selectedViewController.childViewControllers[0]);
26. [**self**.selectedViewController.childViewControllers[0].navigationControllerpushViewController:searchControlleranimated:**YES**];
27. }
28. //购物车
29. **if**([ sharedNetCountManager.applicationShortcutItemTitleisEqualToString:@"YPYD.UITouchText.cart"])
30. {
31. **self**.selectedIndex =2;
32. }
33. //我的 U
34. **if**([ sharedNetCountManager.applicationShortcutItemTitleisEqualToString:@"YPYD.UITouchText.myU"])
35. {
36. **self**.selectedIndex =3;
37. }
38. }

**3D Touch 的 peek and pop 使用与相关api详解**

根据苹果的介绍，Peek手势允许用户通过短时间按压屏幕进行操作，可在邮件、照片等应用弹出全新功能菜单，给出预览内容。如果按压力度加大，则是Pop手势功能，会让被点击内容完全呈现，这些内容可以是文字、图像、网页以及其他各种内容。**简单来说，Peek专注于预览，Pop可以全面展现内容。**如 图9

图 10。



图9 peek 预览图 图 10 pop 视图

**实现peek and pop步骤**

 1、遵守协议 UIViewControllerPreviewingDelegate

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. **@interface** YPYDHomeViewController ()
2. <UITableViewDelegate,UITableViewDataSource,UIViewControllerPreviewingDelegate>

 2、注册Peek和Pop方法

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. [**self** registerForPreviewingWithDelegate:**self** sourceView:**self**.view];

 3、实现代理方法

#pragma mark peek && pop 代理方法轻按进入浮动预览页面

/\*\* peek手势  \*/

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. - (nullable **UIViewController** \*)previewingContext:(**id** <UIViewControllerPreviewing>)previewingContext viewControllerForLocation:(CGPoint)location
2. {
3. **YPYDPOPTestViewController1** \*childVC = [[YPYDPOPTestViewController1alloc]init];
4. // 获取用户手势点所在cell的下标。同时判断手势点是否超出tableView响应范围。
5. **if** (![selfgetShouldShowRectAndIndexPathWithLocation:location])
6. **return nil**;
7. previewingContext.sourceRect = **self**.sourceRect;
9. // 加个白色背景
10. **UIView** \*bgView =[[UIViewalloc]initWithFrame:CGRectMake(20,10,ScreenWidth -40,ScreenHeight -20 -6**64** \*2)];
11. bgView.backgroundColor = [UIColorwhiteColor];
12. bgView.layer.cornerRadius =10;
13. bgView.clipsToBounds = **YES**;
14. [childVC.view addSubview:bgView];
16. // 加个lable
17. **UILabel** \*lable = [[UILabelalloc]initWithFrame:bgView.bounds];
18. lable.textAlignment =NSTextAlignmentCenter;
19. lable.numberOfLines = 3;
20. lable.text =[NSStringstringWithFormat:@"按着的是 %@\n用力，再按重一点...",**self**.items[**self**.indexPath.row]];
21. [bgView addSubview:lable];
23. **return** childVC;
24. }

/\*\* 获取用户手势点所在cell的下标。同时判断手势点是否超出tableView响应范围。\*/

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. #pragma mark  比较巧妙 准确的 获取高亮区域的方法
2. /\*\* 获取用户手势点所在cell的下标。同时判断手势点是否超出tableView响应范围。\*/
3. - (**BOOL**)getShouldShowRectAndIndexPathWithLocation:(CGPoint)location
4. {
5. YPYDLog(@"%f",location.y);
6. // 根据手指按压的区域，结合 tableView 的 Y 偏移量（上下）
7. location.y = **self**.homeTableView.contentOffset.y+location.y;
8. //定位到当前，按压的区域处于哪个 cell  获得 cell 的indexPath
9. **self**.indexPath = [**self**.homeTableView indexPathForRowAtPoint:location];
10. // 根据cell 的indexPath 取出 cell
11. **UITableViewCell** \* cell = [**self**.homeTableView cellForRowAtIndexPath:**self**.indexPath];
12. //    cell.backgroundColor = [UIColor redColor];
13. // 根据 获得cell ，确定高亮的区域，记得 高亮区域是相对于屏幕  位置来算，记得减去 tableView 的 Y偏移量
14. **self**.sourceRect = CGRectMake(cell.frame.origin.x, cell.frame.origin.y-**self**.homeTableView.contentOffset.y, cell.frame.size.width,cell.frame.size.height);
15. // 如果row越界了，返回NO 不处理peek手势
16. **return** (**self**.indexPath.row >= **self**.items.count &&**self**.indexPath.row<0) ? **NO** : **YES**;
17. }

/\*\* pop手势  \*/

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. - (**void**)previewingContext:(**id** <UIViewControllerPreviewing>)previewingContext commitViewController:(**UIViewController** \*)viewControllerToCommit
2. {
3. [selftableView:**self**.homeTableViewdidSelectRowAtIndexPath:**self**.indexPath];
4. }
5. - (**void**)tableView:(**UITableView** \*)tableView didSelectRowAtIndexPath:(**NSIndexPath** \*)indexPath
6. {
7. **YPYDPOPTestViewController1** \* testVC = [[YPYDPOPTestViewController1alloc]init];
8. [**self**.navigationControllerpushViewController:testVCanimated:**YES**];
9. }

完成后，即可实现图9、图10的效果。

**预览时上滑底部菜单的添加**

在我们刚刚创建的预览控制器YPYDPOPTestViewController1里实现

如 图 11。

//遵守协议

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. **@interface** YPYDPOPTestViewController1 ()<UIViewControllerPreviewingDelegate>

//预览页面

底部Action Items

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. - (NSArray<**id**<UIPreviewActionItem>> \*)previewActionItems
2. {
3. **UIPreviewAction** \*p1 =[UIPreviewActionactionWithTitle:@"分享"style:UIPreviewActionStyleDefaulthandler:^(**UIPreviewAction** \*\_Nonnull action, **UIViewController** \*\_Nonnull previewViewController) {       NSLog(@"点击了分享");    }];
4. **UIPreviewAction** \*p2 =[UIPreviewActionactionWithTitle:@"收藏"style:UIPreviewActionStyleDefaulthandler:^(**UIPreviewAction** \*\_Nonnull action, **UIViewController** \*\_Nonnull previewViewController) {       NSLog(@"点击了收藏");    }];
5. **NSArray** \*actions = @[p1,p2];
6. **return** actions;
7. }



图 11

**3D Touch 的Force Properties**

**使用与相关api详解**

在iOS 9中，UITouch类支持3D触摸应用程序定制实现了两个新的属性：force 和 maximumpossibleforce 。

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. // Force of the touch, where 1.0 represents the force of an average touch
2. **@property**(**nonatomic**,**readonly**) CGFloat force NS\_AVAILABLE\_IOS(9\_0);
3. // Maximum possible force with this input mechanism
4. **@property**(**nonatomic**,**readonly**) CGFloat maximumPossibleForce NS\_AVAILABLE\_IOS(9\_0);

首次在iOS设备上，这些特性让你检测和响应触摸在UIEvent对象接收应用程序的压力。一个触摸的力量有一个高动态范围，可作为一个浮点值应用到您的应用程序。

**[objc]** view plain copy

print?[](https://code.csdn.net/snippets/1568137)[](https://code.csdn.net/snippets/1568137/fork)

1. #pragma mark 获取压力的大小非常简单，我们可以通过UITouch类中的一些属性来完成。我们只需要在ToucheMoved的事件中捕获这些信息，请注意，不用判断 x,y 值变化
2. //我这里 只做了一个简单的  压力感应示例，如需更复杂的效果，可以自行编制。
3. - (**void**)touchesMoved:(NSSet<**UITouch** \*> \*)touches withEvent:(**UIEvent** \*)event
4. {
5. //宣告一个UITouch的指标来存放事件触发时所撷取到的状态
6. **UITouch** \*touch = [[event allTouches] anyObject];
7. \_forceLab.text = [NSString stringWithFormat:@"当前压力值为: %f",touch.force];
8. //   YPYDLog(@"最大压力值  %f",touch.maximumPossibleForce);最大压力值  6.666667
9. **if** (touch.force>0.2)
10. {
11. \_lab.font = [UIFont systemFontOfSize:20\*touch.force];
12. \_lab.textColor = YPYDColor(255\*touch.force/6.5, 0, 0, 1);
13. }
14. }

**结语**

        关于3DTouch功能 更详细的实现步骤，你可以参考下方的3D

Touch  Demo，如有疑问，可以在下方评论留言，或者QQ：823023927，最后，如有疏漏和错误之处，欢迎指正。

 3D Touch  Demo  百度网盘 <http://pan.baidu.com/s/1Mtaj0>

github地址： <https://github.com/wyk125/3D-Touch-Demo.git>  觉得不错，记得Start哦，谢过！