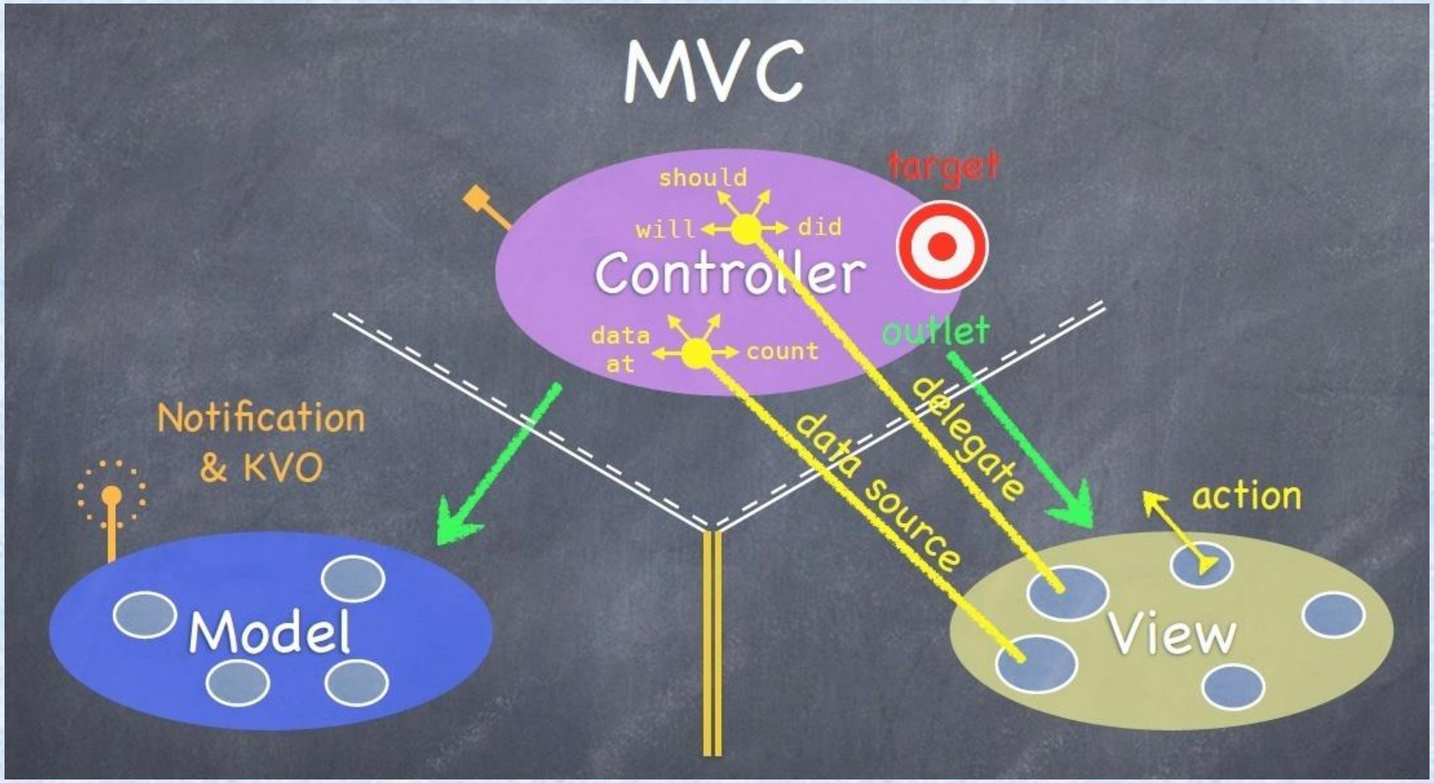
通知、手势、警告视图与触摸事件

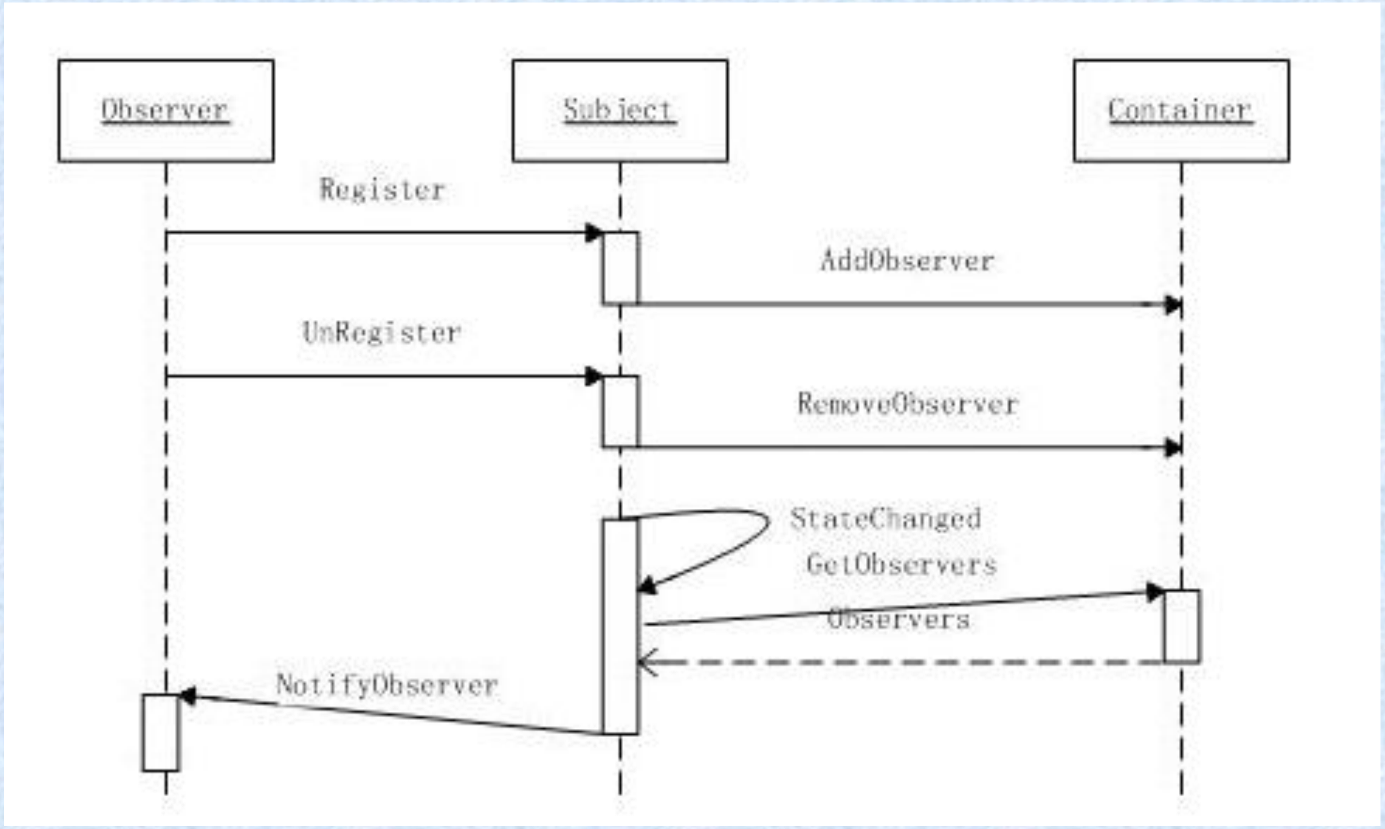
一、iOS中的通知


MVC
Controller
should
will
did
data
at
count
target
outlet
delegate
data source
action
Notification
& KVO
Model
View

观察者模式

观察者模式又叫做发布-订阅(Publish/Subscribe)模式。观察者模式定义了一种一对多地依赖模式，让多个观察者同时监听某一个主题对象。这个主题对象在状态发生变化时，会通知所有的观察者对象，使它们能够自动更新自己。这里的主题对象就是指通知者，又叫做发布者。观察者又叫订阅者。

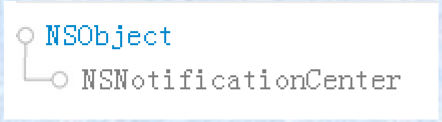
eg:银行排队的例子


Observer
Subject
Container
Register
AddObserver
UnRegister
RemoveObserver
StateChanged
GetObservers
Observers
NotifyObserver

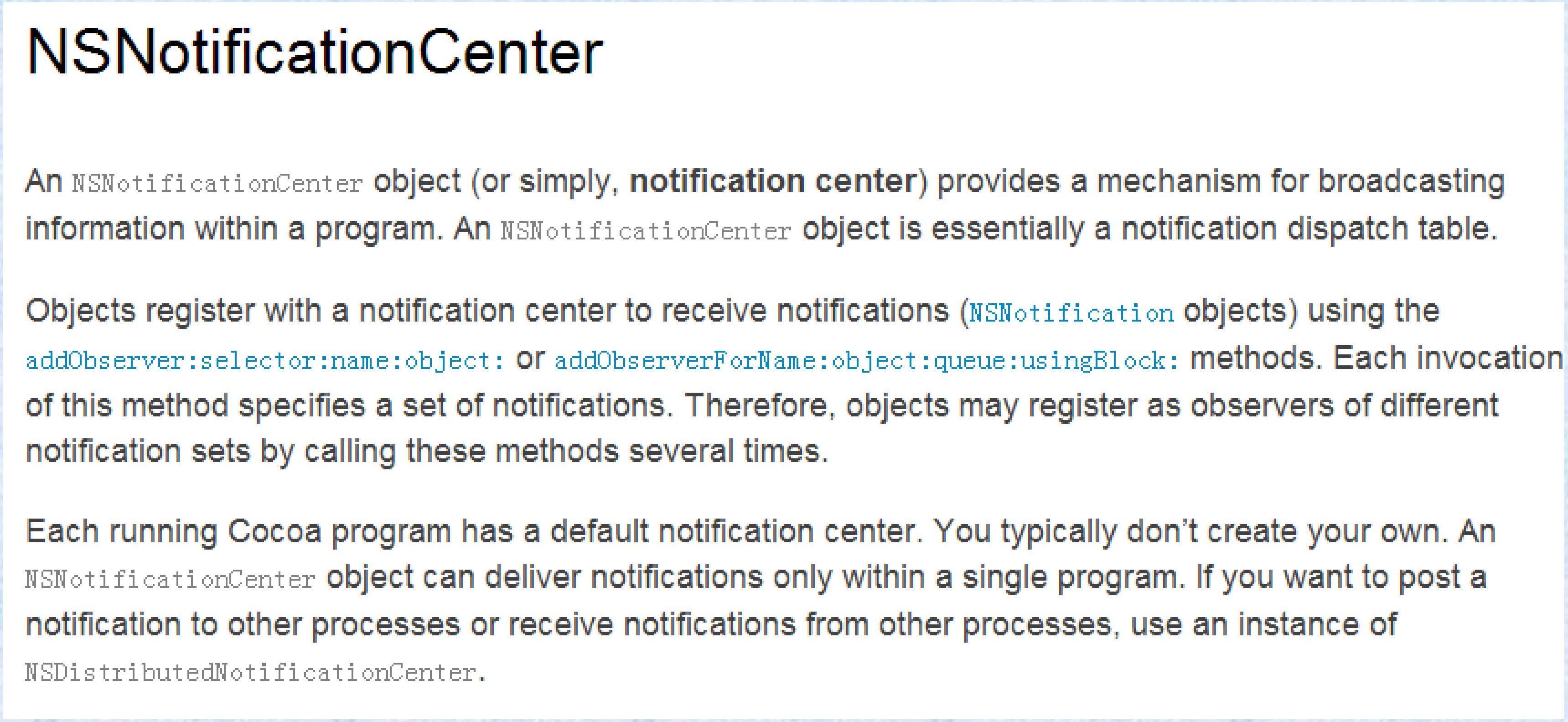
1、iOS中使用通知的步骤

步骤1： 创建一个通知对象。
步骤2： 注册通知。
步骤3： 发送通知。
步骤4： 移除通知。

2、通知涉及到类


NSObject
NSNotificationCenter

通知中心

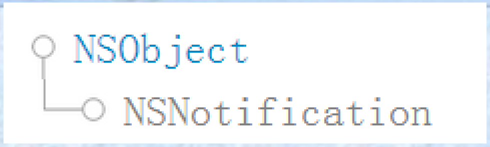

NSNotificationCenter

An NSNotificationCenter object (or simply, **notification center**) provides a mechanism for broadcasting information within a program. An NSNotificationCenter object is essentially a notification dispatch table.

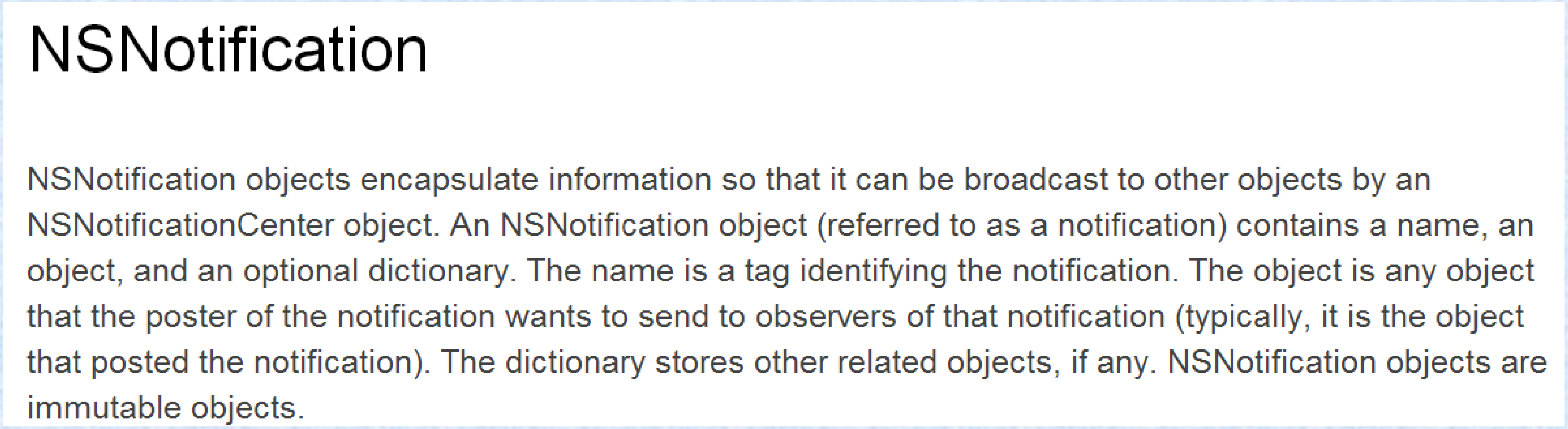
Objects register with a notification center to receive notifications (NSNotification objects) using the addObserver:selector:name:object: or addObserverForName:object:queue:usingBlock: methods. Each invocation of this method specifies a set of notifications. Therefore, objects may register as observers of different notification sets by calling these methods several times.

Each running Cocoa program has a default notification center. You typically don't create your own. An NSNotificationCenter object can deliver notifications only within a single program. If you want to post a notification to other processes or receive notifications from other processes, use an instance of NSDistributedNotificationCenter.

3、通知



NSObject
NSNotification



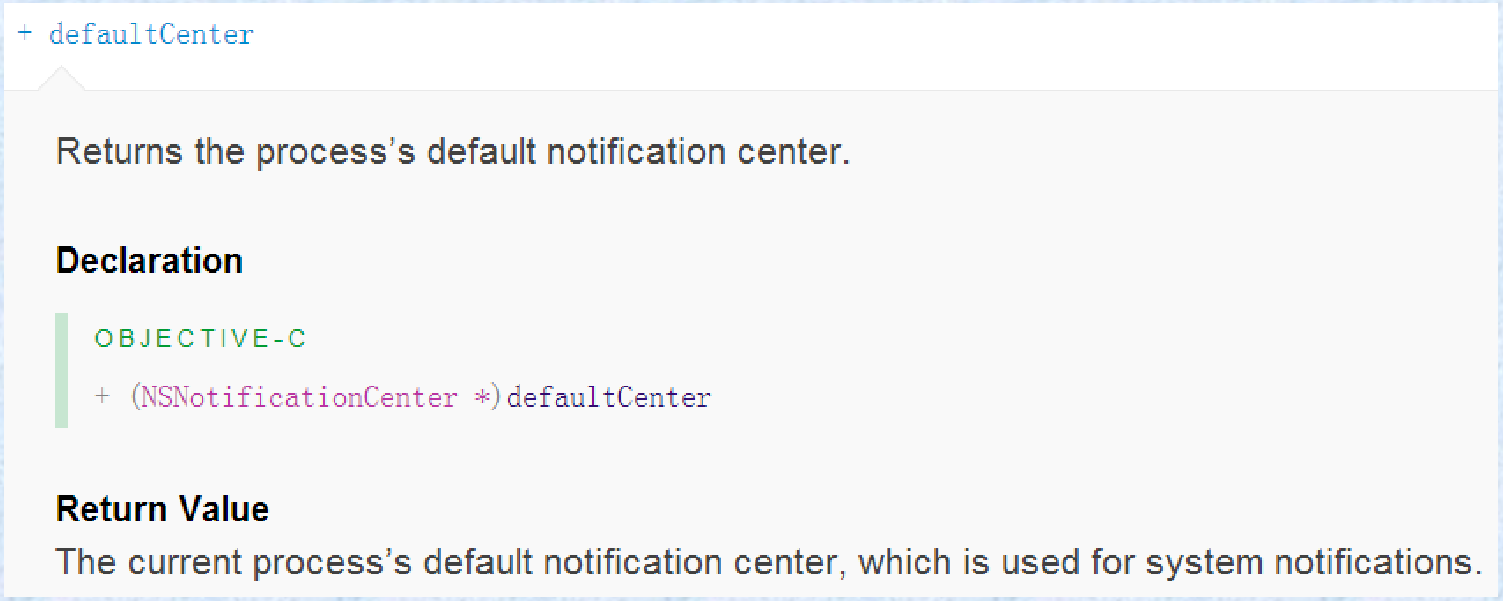
NSNotification

NSNotification objects encapsulate information so that it can be broadcast to other objects by an NSNotificationCenter object. An NSNotification object (referred to as a notification) contains a name, an object, and an optional dictionary. The name is a tag identifying the notification. The object is any object that the poster of the notification wants to send to observers of that notification (typically, it is the object that posted the notification). The dictionary stores other related objects, if any. NSNotification objects are immutable objects.

三个属性

```
@property(readonly, copy) NSString *name
@property(readonly, retain) id object
@property(readonly, copy) NSDictionary *userInfo
```

4、创建一个通知对象


```
+ defaultCenter
```

Returns the process's default notification center.

Declaration

OBJECTIVE-C

```
+ (NSNotificationCenter *)defaultCenter
```

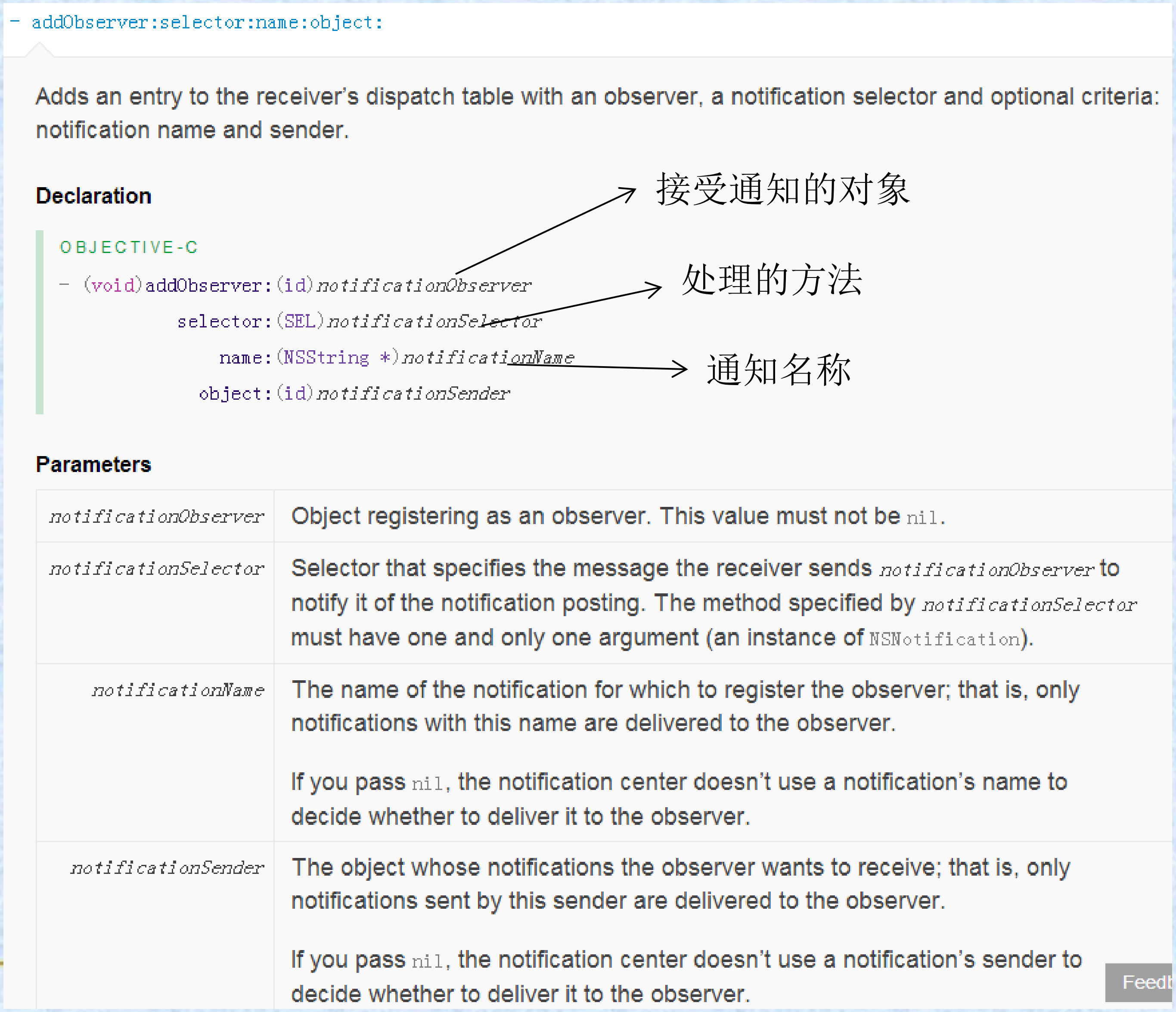
Return Value

The current process's default notification center, which is used for system notifications.

```
NSNotificationCenter *center = [NSNotificationCenter defaultCenter];
```

每一个程序都有一个自己的通知中心，即NSNotificationCenter对象。采用defaultCenter方法就可以获得唯一的NSNotificationCenter对象

5、注册通知



- addObserver:selector:name:object:

Adds an entry to the receiver's dispatch table with an observer, a notification selector and optional criteria: notification name and sender.

Declaration

OBJECTIVE-C

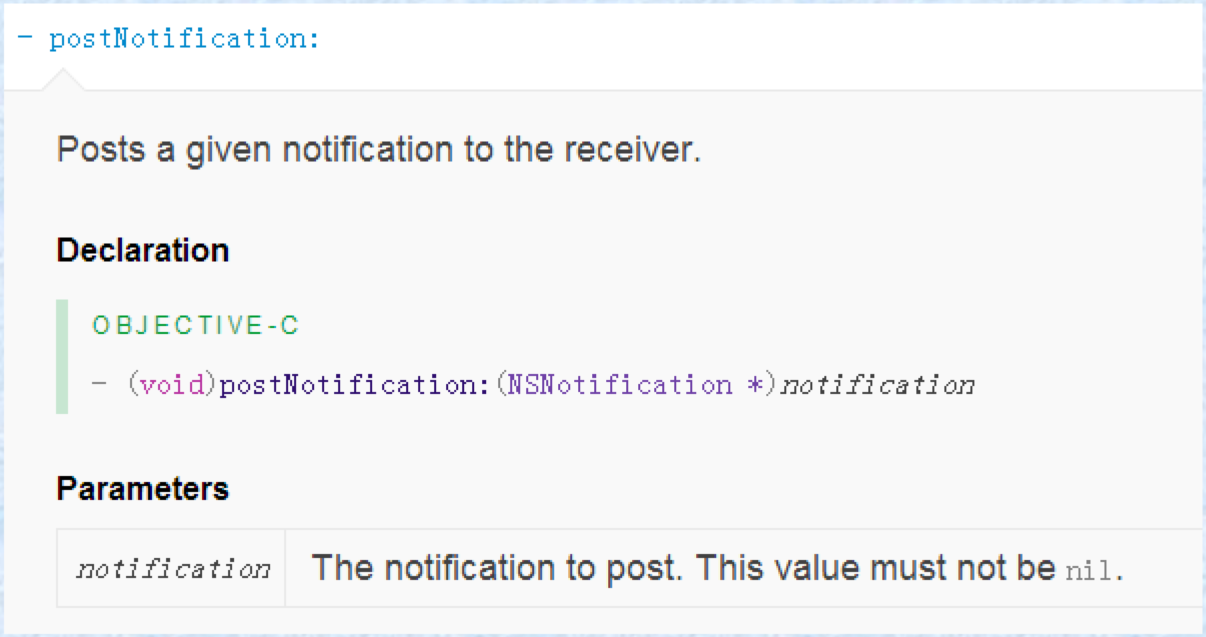
```
- (void)addObserver:(id)notificationObserver
           selector:(SEL)notificationSelector
               name:(NSString *)notificationName
             object:(id)notificationSender
```

Parameters

notificationObserver	Object registering as an observer. This value must not be `nil`.
notificationSelector	Selector that specifies the message the receiver sends *notificationObserver* to notify it of the notification posting. The method specified by *notificationSelector* must have one and only one argument (an instance of `NSNotification`).
notificationName	The name of the notification for which to register the observer; that is, only notifications with this name are delivered to the observer. If you pass `nil`, the notification center doesn't use a notification's name to decide whether to deliver it to the observer.
notificationSender	The object whose notifications the observer wants to receive; that is, only notifications sent by this sender are delivered to the observer. If you pass `nil`, the notification center doesn't use a notification's sender to decide whether to deliver it to the observer.

Feedb

6、发送通知



- postNotification:

Posts a given notification to the receiver.

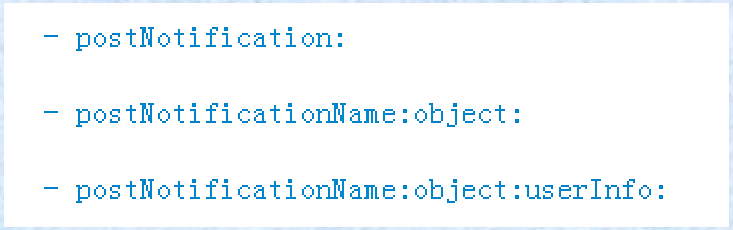
Declaration

OBJECTIVE-C

```
- (void)postNotification:(NSNotification *)notification
```

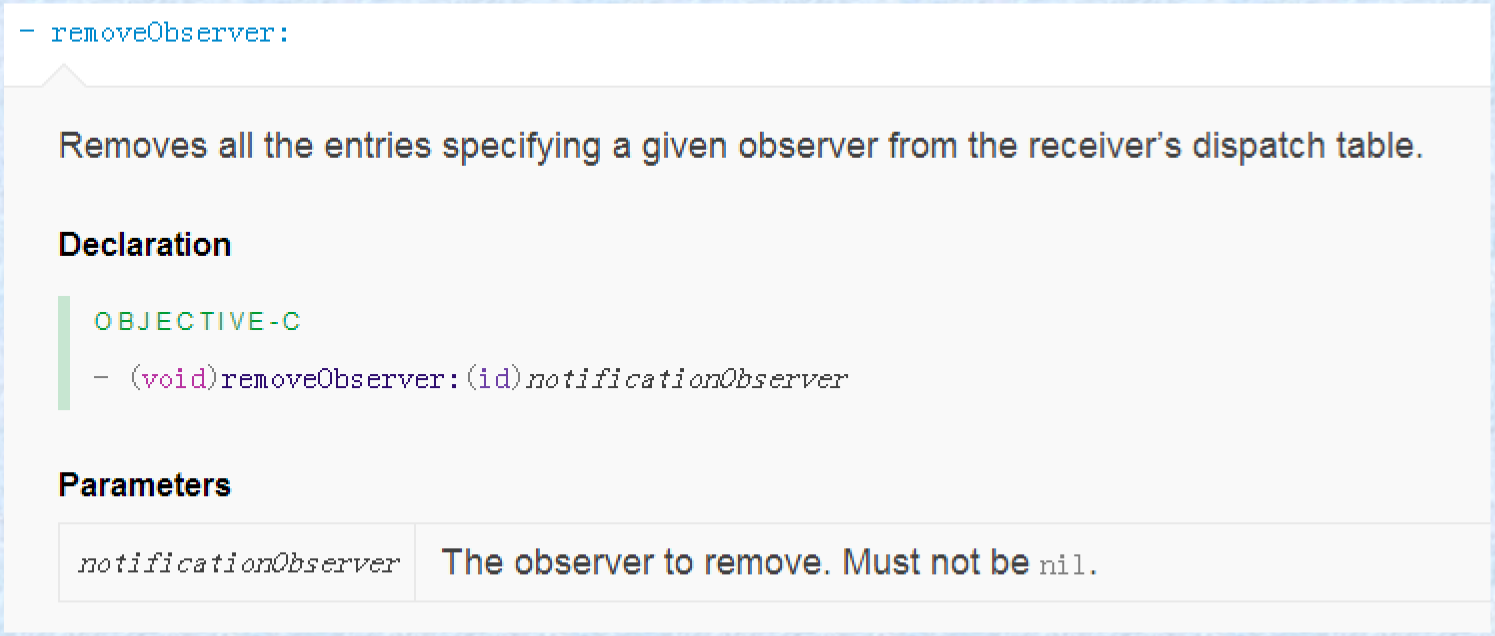
Parameters

notification	The notification to post. This value must not be `nil`.



- postNotification:
- postNotificationName:object:
- postNotificationName:object:userInfo:

7、移除通知



- removeObserver:

Removes all the entries specifying a given observer from the receiver's dispatch table.

Declaration

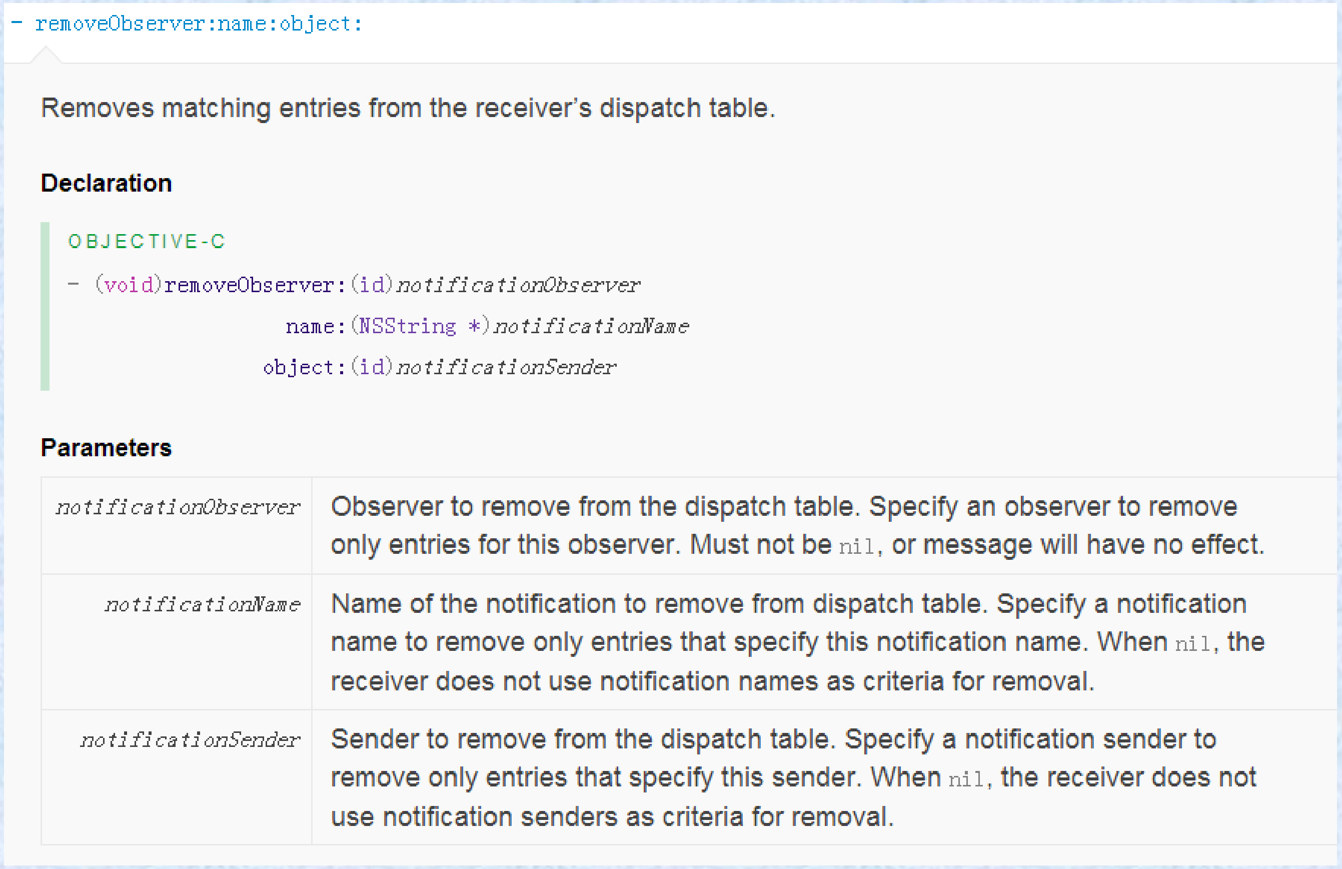
OBJECTIVE-C

```
- (void)removeObserver:(id)notificationObserver
```

Parameters

notificationObserver	The observer to remove. Must not be `nil`.

```
[center removeObserver:self];
```


- removeObserver:name:object:

Removes matching entries from the receiver's dispatch table.

Declaration

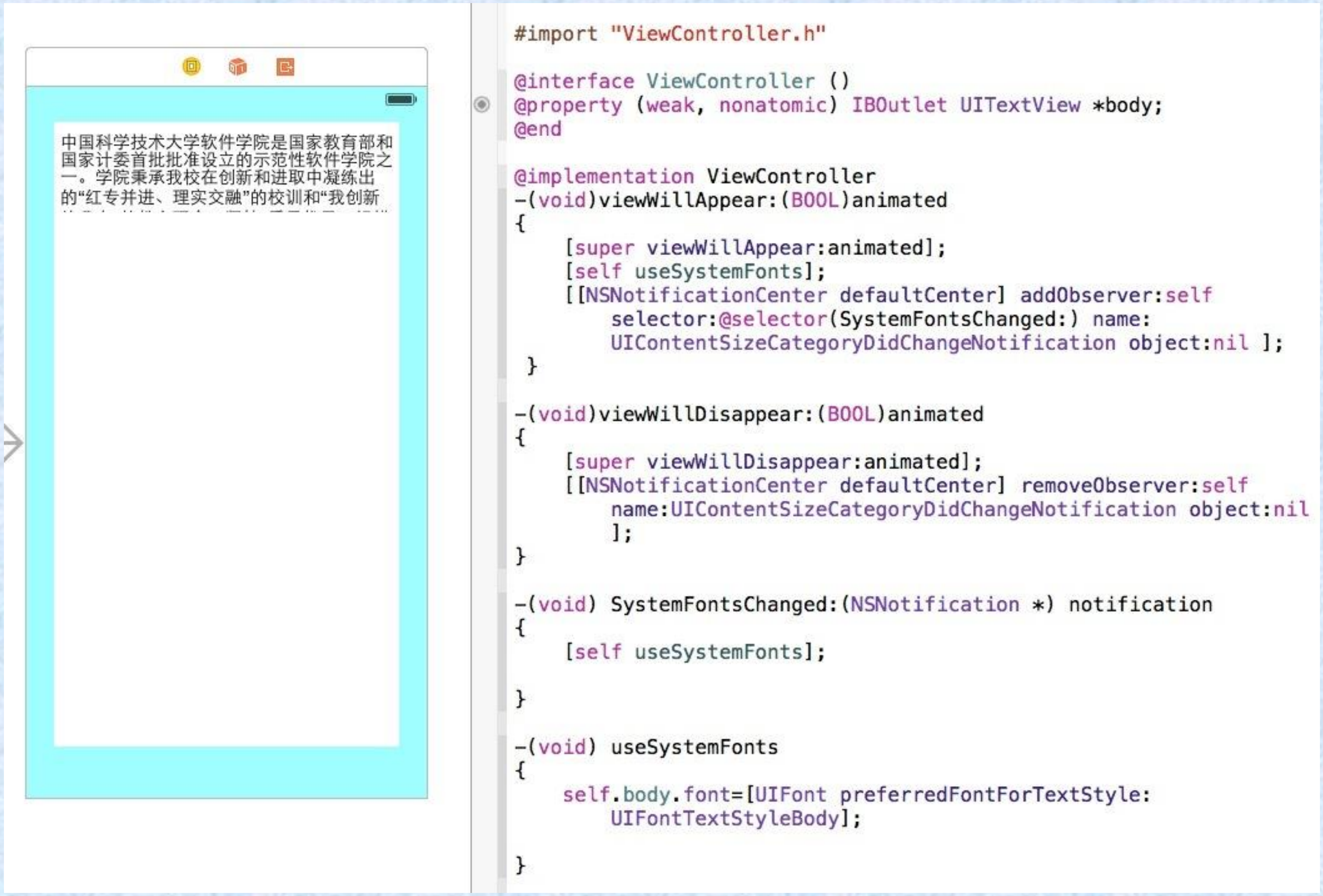
OBJECTIVE-C

```
- (void)removeObserver:(id)notificationObserver
                  name:(NSString *)notificationName
                object:(id)notificationSender
```

Parameters

notificationObserver	Observer to remove from the dispatch table. Specify an observer to remove only entries for this observer. Must not be `nil`, or message will have no effect.
notificationName	Name of the notification to remove from dispatch table. Specify a notification name to remove only entries that specify this notification name. When `nil`, the receiver does not use notification names as criteria for removal.
notificationSender	Sender to remove from the dispatch table. Specify a notification sender to remove only entries that specify this sender. When `nil`, the receiver does not use notification senders as criteria for removal.

[center removeObserver:self name:UIContentSizeCategoryDidChangeNotification object:nil];



中国科学技术大学软件学院是国家教育部和国家计委首批批准设立的示范性软件学院之一。学院秉承我校在创新和进取中凝练出的“红专并进、理实交融”的校训和“我创新

```objc
#import "ViewController.h"

@interface ViewController ()
@property (weak, nonatomic) IBOutlet UITextView *body;
@end

@implementation ViewController
-(void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    [self useSystemFonts];
    [[NSNotificationCenter defaultCenter] addObserver:self
        selector:@selector(SystemFontsChanged:) name:
        UIContentSizeCategoryDidChangeNotification object:nil ];
 }

-(void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
    [[NSNotificationCenter defaultCenter] removeObserver:self
        name:UIContentSizeCategoryDidChangeNotification object:nil
        ];
}

-(void) SystemFontsChanged:(NSNotification *) notification
{
    [self useSystemFonts];

}

-(void) useSystemFonts
{
    self.body.font=[UIFont preferredFontForTextStyle:
        UIFontTextStyleBody];

}
```

应用示例

二、KVO

KVO(key-value-observing)就是用一个key来找属性，并监听，当属性值改变后，通知监听对象。

KVO的使用

使用KVO的要求是对象必须能支持KVC机制——所有NSObject的子类都支持这个机制。

第一步：添加观察者(注册)
第二步：在观察者中实现监听方法，observeValueForKeyPath: ofObject: change: context:
第三步：移除观察者

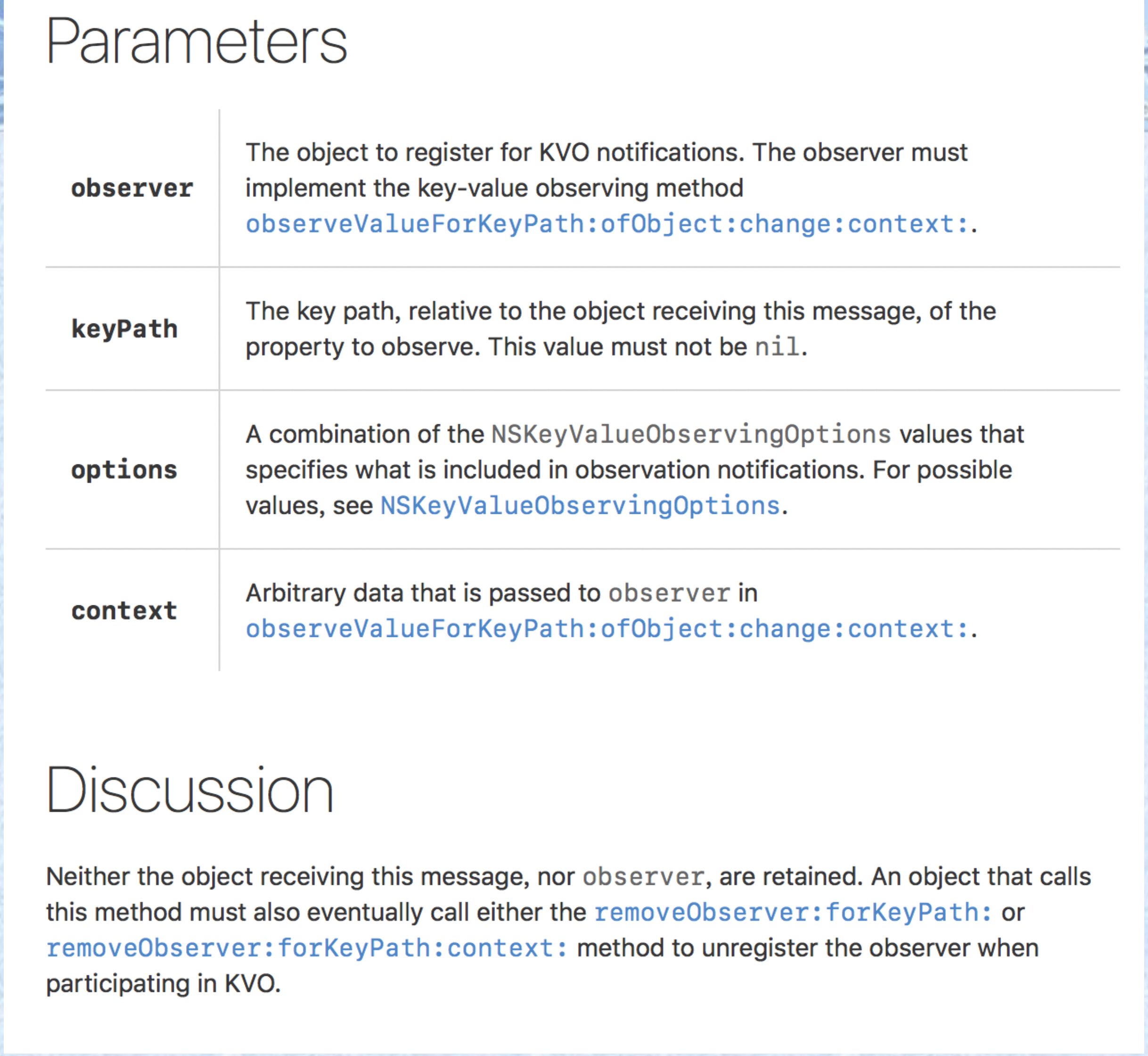
注册

观察者

观察的属性

属性的配置

上下文
（传递的数据）



Parameters

observer	The object to register for KVO notifications. The observer must implement the key-value observing method `observeValueForKeyPath:ofObject:change:context:`.
keyPath	The key path, relative to the object receiving this message, of the property to observe. This value must not be `nil`.
options	A combination of the `NSKeyValueObservingOptions` values that specifies what is included in observation notifications. For possible values, see `NSKeyValueObservingOptions`.
context	Arbitrary data that is passed to `observer` in `observeValueForKeyPath:ofObject:change:context:`.

Discussion

Neither the object receiving this message, nor `observer`, are retained. An object that calls this method must also eventually call either the `removeObserver:forKeyPath:` or `removeObserver:forKeyPath:context:` method to unregister the observer when participating in KVO.

```
-(void)addObserver:(NSObject *)anObserver
        forKeyPath:(NSString *)keyPath
           options:(NSKeyValueObservingOptions)options
           context:(void *)context
```

options：KVO的一些属性配置；有四个选项。

NSKeyValueObservingOptionNew：
change字典包括改变后的值

NSKeyValueObservingOptionOld:
change字典包括改变前的值

NSKeyValueObservingOptionInitial:
注册后立刻触发KVO通知

NSKeyValueObservingOptionPrior:
值改变前是否也要通知（这个key决定了是否在改变前改变后通知两次）

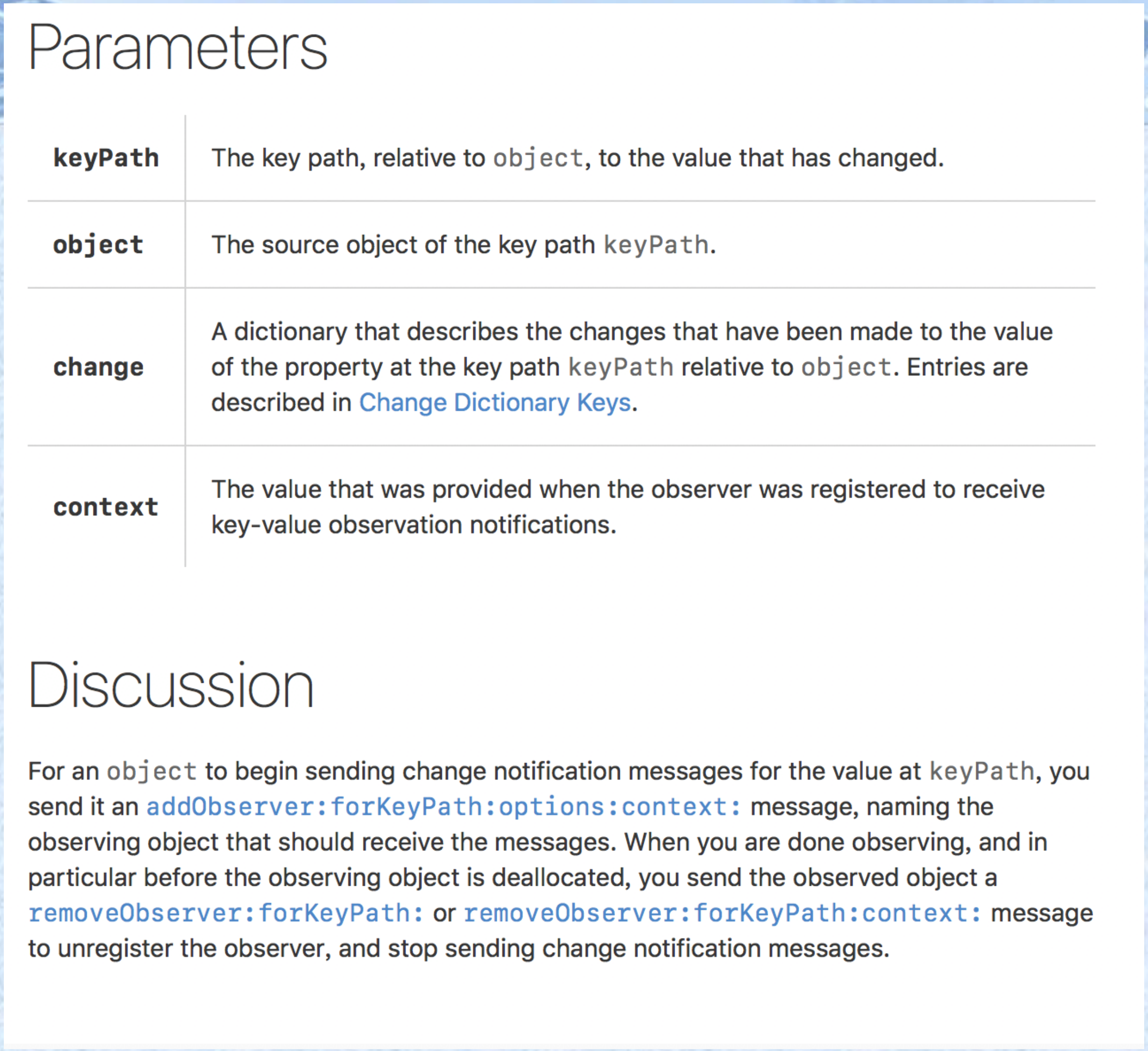
监听

监听的属性

监听的对象

新值和旧值

额外的数据



Parameters

keyPath	The key path, relative to `object`, to the value that has changed.
object	The source object of the key path `keyPath`.
change	A dictionary that describes the changes that have been made to the value of the property at the key path `keyPath` relative to `object`. Entries are described in Change Dictionary Keys.
context	The value that was provided when the observer was registered to receive key-value observation notifications.

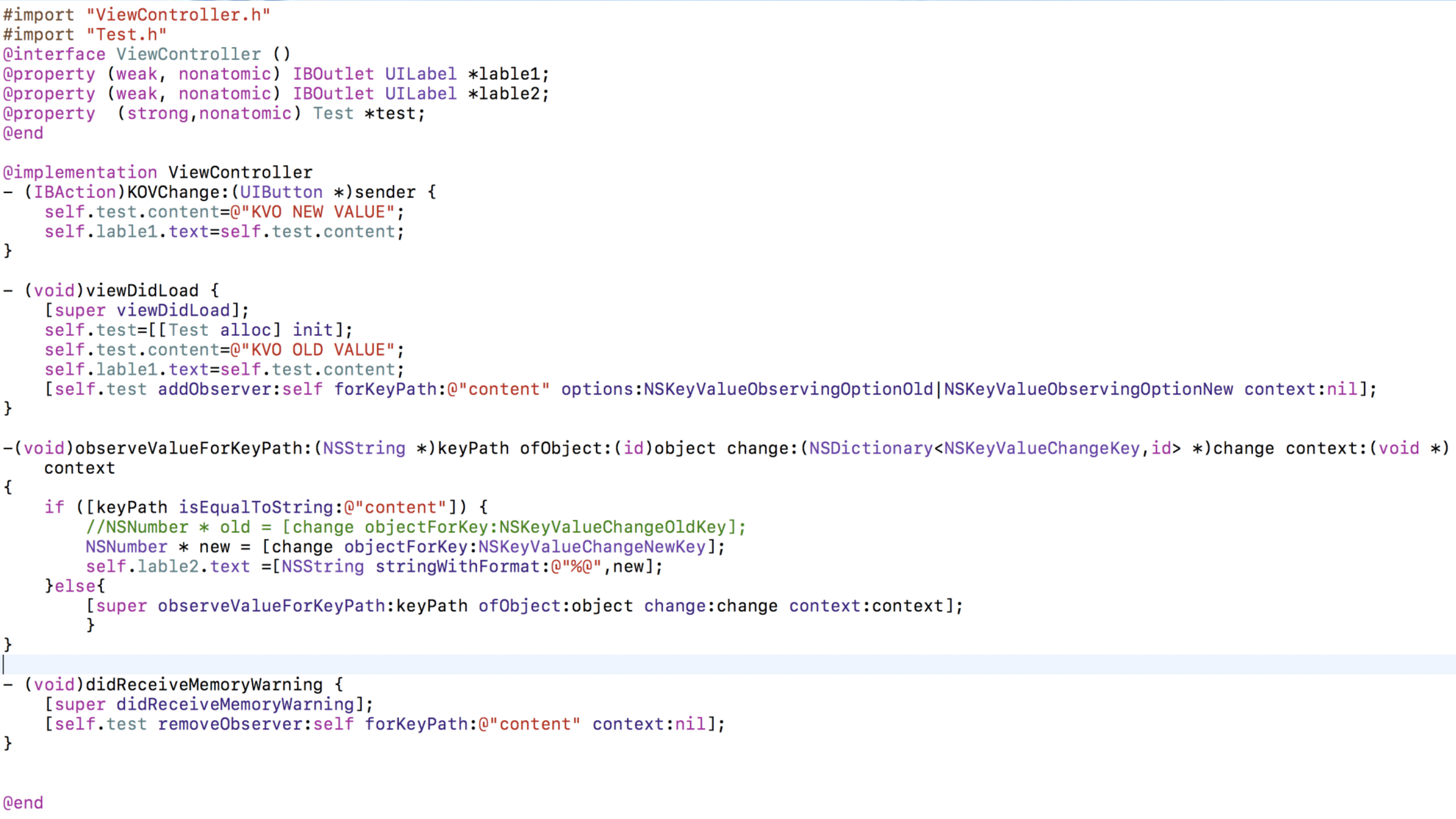
Discussion

For an `object` to begin sending change notification messages for the value at `keyPath`, you send it an `addObserver:forKeyPath:options:context:` message, naming the observing object that should receive the messages. When you are done observing, and in particular before the observing object is deallocated, you send the observed object a `removeObserver:forKeyPath:` or `removeObserver:forKeyPath:context:` message to unregister the observer, and stop sending change notification messages.

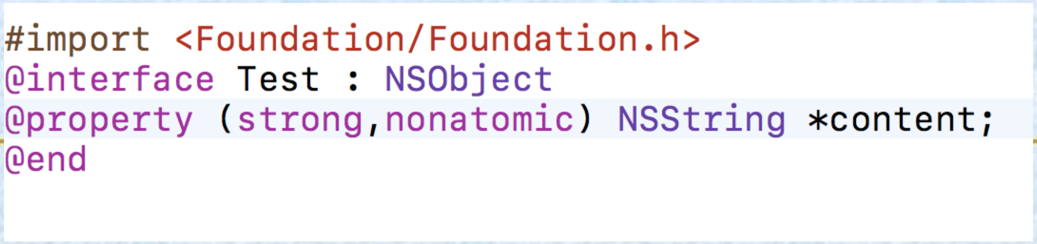
```
-(void)observeValueForKeyPath:(NSString *)keyPath
                     ofObject:(id)object
                       change:(NSDictionary *)change
                      context:(void *)context
```

反注册

```
- (void)removeObserver:(NSObject *)observer forKeyPath:(NSString *)keyPath;
```



```
#import "ViewController.h"
#import "Test.h"
@interface ViewController ()
@property (weak, nonatomic) IBOutlet UILabel *lable1;
@property (weak, nonatomic) IBOutlet UILabel *lable2;
@property  (strong,nonatomic) Test *test;
@end

@implementation ViewController
- (IBAction)KOVChange:(UIButton *)sender {
    self.test.content=@"KVO NEW VALUE";
    self.lable1.text=self.test.content;
}

- (void)viewDidLoad {
    [super viewDidLoad];
    self.test=[[Test alloc] init];
    self.test.content=@"KVO OLD VALUE";
    self.lable1.text=self.test.content;
    [self.test addObserver:self forKeyPath:@"content" options:NSKeyValueObservingOptionOld|NSKeyValueObservingOptionNew context:nil];
}

-(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object change:(NSDictionary<NSKeyValueChangeKey,id> *)change context:(void *)
    context
{
    if ([keyPath isEqualToString:@"content"]) {
        //NSNumber * old = [change objectForKey:NSKeyValueChangeOldKey];
        NSNumber * new = [change objectForKey:NSKeyValueChangeNewKey];
        self.lable2.text =[NSString stringWithFormat:@"%@",new];
    }else{
        [super observeValueForKeyPath:keyPath ofObject:object change:change context:context];
        }
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    [self.test removeObserver:self forKeyPath:@"content" context:nil];
}


@end
```



```
#import <Foundation/Foundation.h>
@interface Test : NSObject
@property (strong,nonatomic) NSString *content;
@end
```

二、iOS中使用手势

1、使用手势的步骤

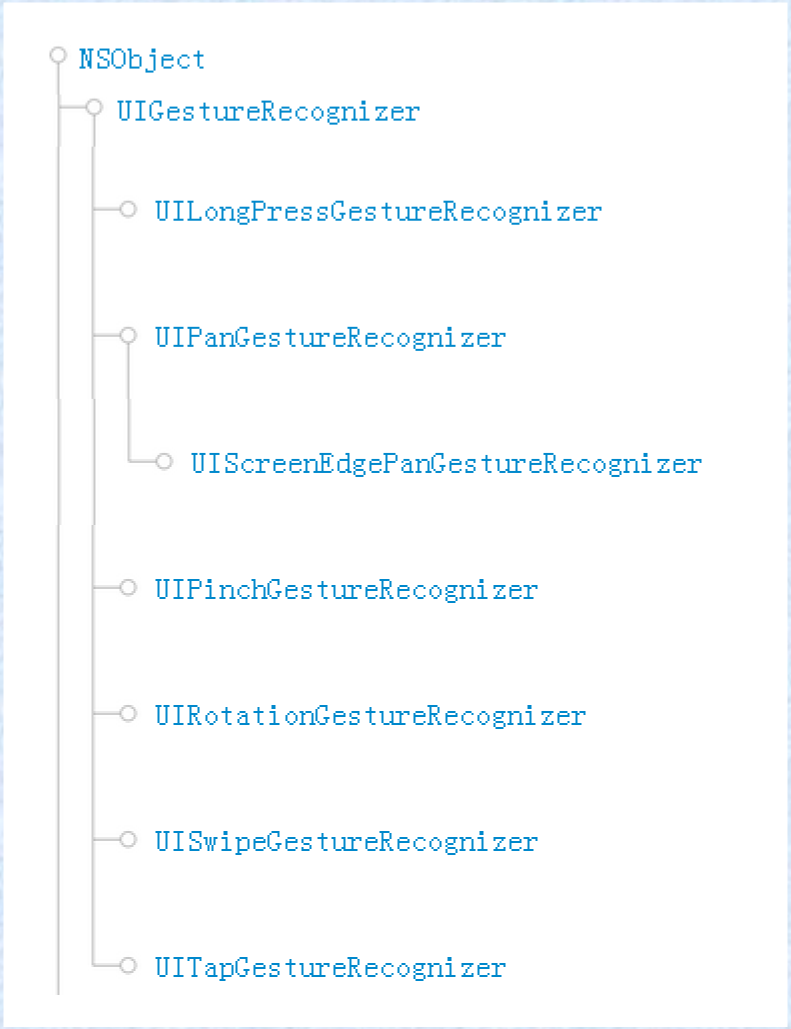
步骤1： 创建一个手势识别器对象。给需要
　　　 手势功能的视图添加手势识别器对象。

步骤2： 添加一个方法来响应手势。

2、手势涉及到类

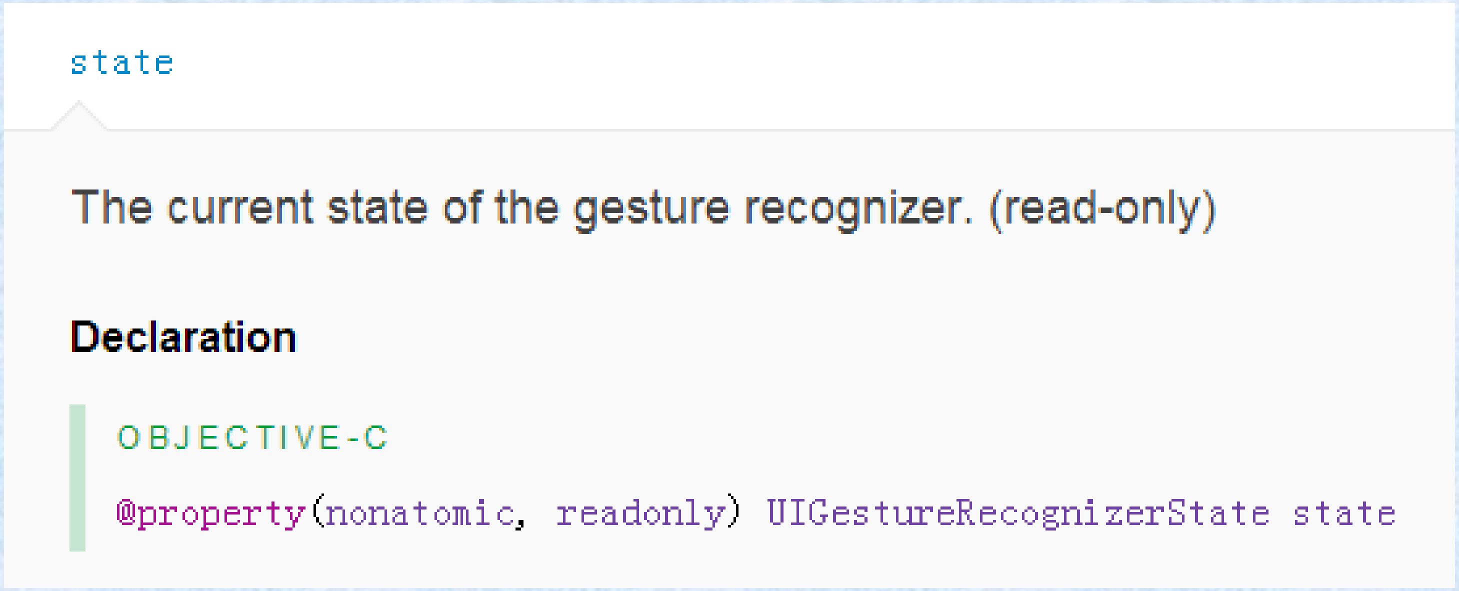
UIGestureRecognizer

UIGestureRecognizer is an **abstract base class** for concrete gesture-recognizer classes. A gesture-recognizer object—or, simply, a gesture recognizer—decouples the logic for recognizing a gesture and acting on that recognition. When one of these objects recognizes a common gesture or, in some cases, a change in the gesture, it sends an action message to each designated target object.



- NSObject
  - UIGestureRecognizer
    - UILongPressGestureRecognizer
    - UIPanGestureRecognizer
      - UIScreenEdgePanGestureRecognizer
    - UIPinchGestureRecognizer
    - UIRotationGestureRecognizer
    - UISwipeGestureRecognizer
    - UITapGestureRecognizer

3、UIGestureRecognizer的几个重要的属性


state

The current state of the gesture recognizer. (read-only)

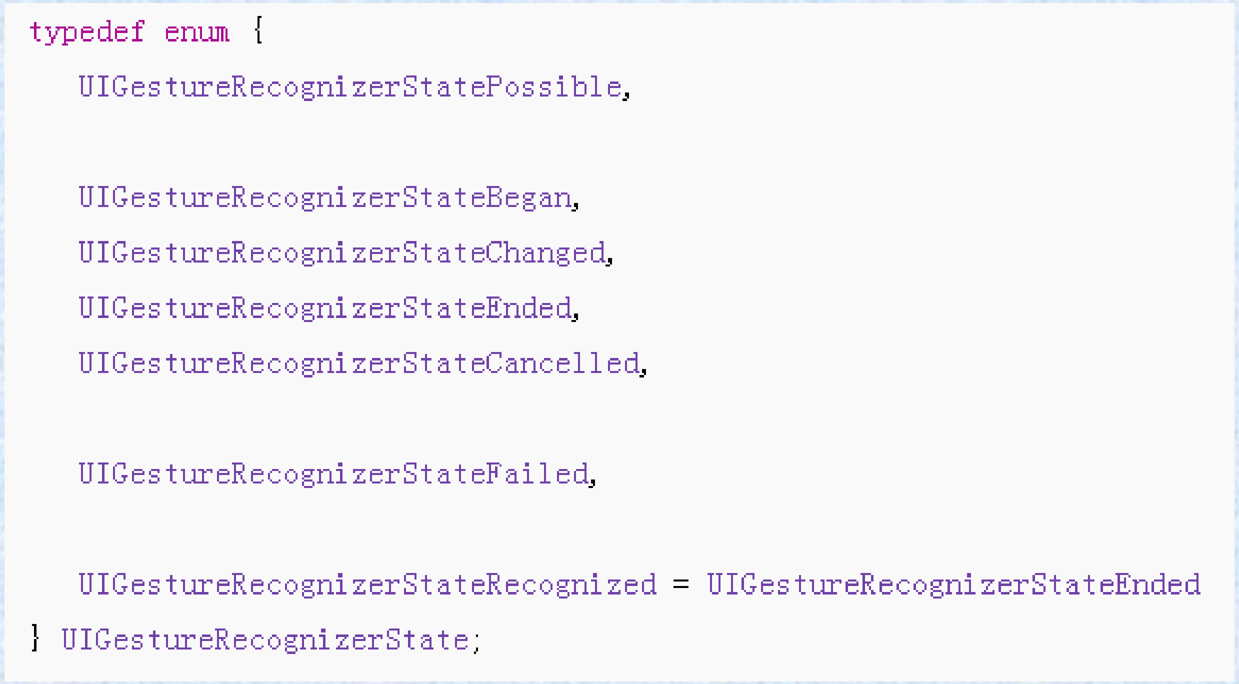
Declaration

OBJECTIVE-C

```
@property(nonatomic, readonly) UIGestureRecognizerState state
```

通过该属性可以知道手势识别器的当前状态

UIGestureRecognizerState



```
typedef enum {
   UIGestureRecognizerStatePossible,

   UIGestureRecognizerStateBegan,
   UIGestureRecognizerStateChanged,
   UIGestureRecognizerStateEnded,
   UIGestureRecognizerStateCancelled,

   UIGestureRecognizerStateFailed,

   UIGestureRecognizerStateRecognized = UIGestureRecognizerStateEnded
} UIGestureRecognizerState;
```

● UIGestureRecognizerStatePossible
The gesture recognizer has not yet recognized its gesture, but may be evaluating touch events. This is the default state.

● UIGestureRecognizerStateBegan
The gesture recognizer has received touch objects recognized as a continuous gesture. It sends its action message (or messages) at the next cycle of the run loop.

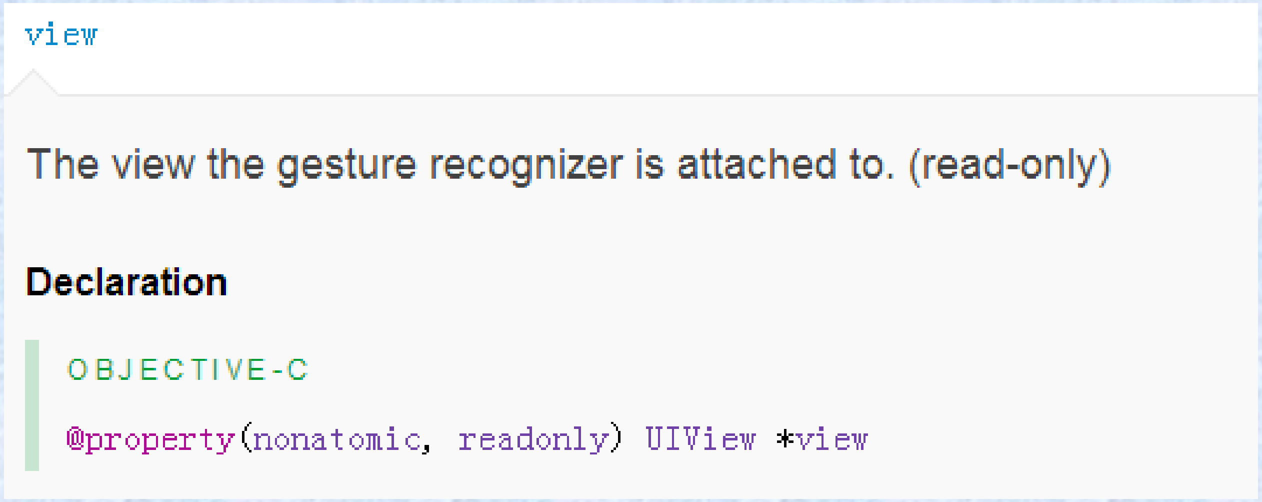
● UIGestureRecognizerStateChanged
The gesture recognizer has received touches recognized as a change to a continuous gesture. It sends its action message (or messages) at the next cycle of the run loop.

● UIGestureRecognizerStateEnded
The gesture recognizer has received touches recognized as the end of a continuous gesture. It sends its action message (or messages) at the next cycle of the run loop and resets its state to UIGestureRecognizerStatePossible.

● UIGestureRecognizerStateCancelled
The gesture recognizer has received touches resulting in the cancellation of a continuous gesture. It sends its action message (or messages) at the next cycle of the run loop and resets its state to UIGestureRecognizerStatePossible.

● UIGestureRecognizerStateFailed
The gesture recognizer has received a multi-touch sequence that it cannot recognize as its gesture. No action message is sent and the gesture recognizer is reset to UIGestureRecognizerStatePossible.

● UIGestureRecognizerStateRecognized
The gesture recognizer has received a multi-touch sequence that it recognizes as its gesture. It sends its action message (or messages) at the next cycle of the run loop and resets its state to UIGestureRecognizerStatePossible.


view

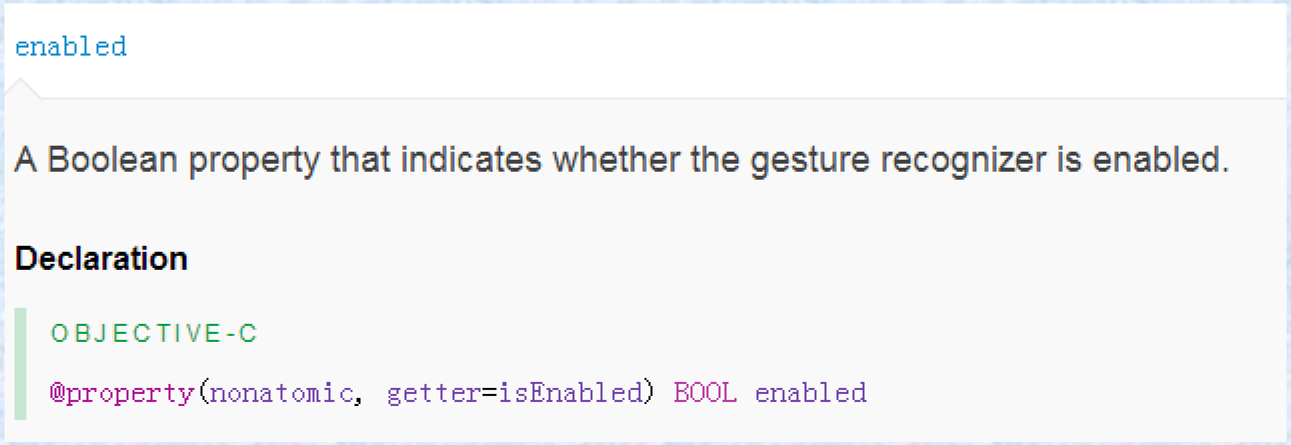
The view the gesture recognizer is attached to. (read-only)

Declaration

OBJECTIVE-C

```
@property(nonatomic, readonly) UIView *view
```

通过该属性可以知道手势识别器的和那个视图关联


enabled

A Boolean property that indicates whether the gesture recognizer is enabled.

Declaration

OBJECTIVE-C

```
@property(nonatomic, getter=isEnabled) BOOL enabled
```

通过该属性决定手势识别器是否可用

4、创建和初始化手势识别器对象

- 方法一：利用代码生成

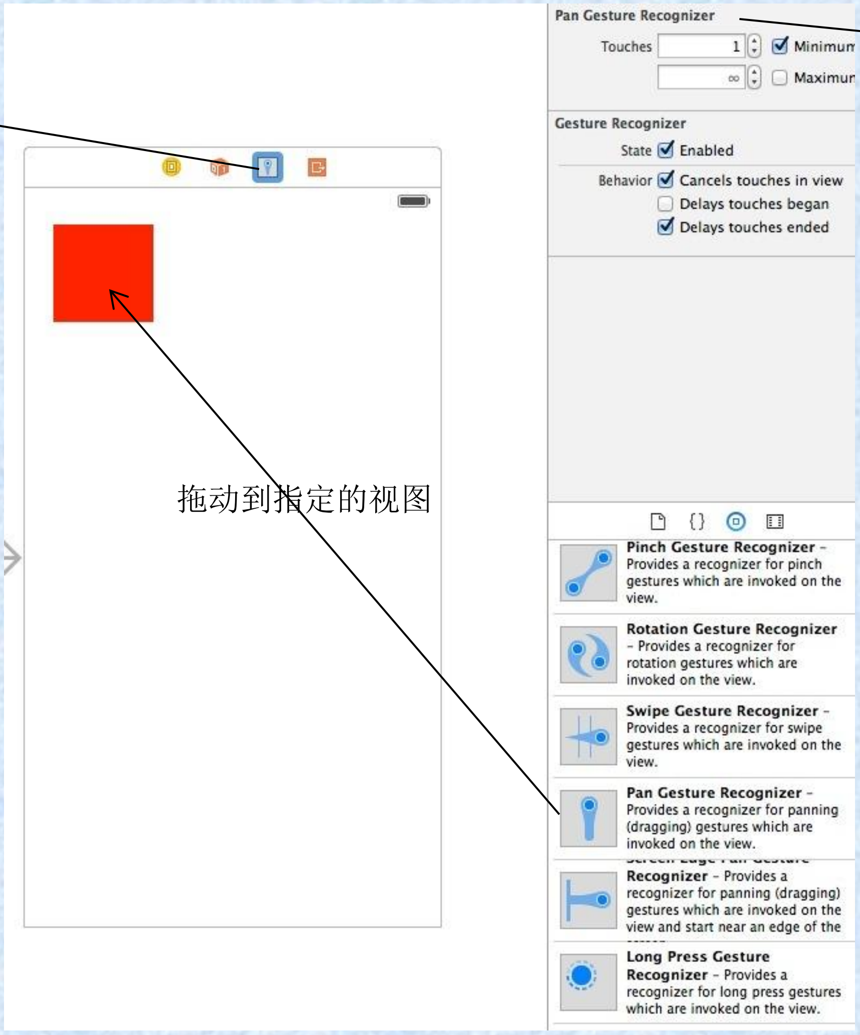
```
UIPanGestureRecognizer *gr =
                    [[UIPanGestureRecognizer alloc] initWithTarget:self.view action:@selector(pan:)];
[self.view  addGestureRecognizer:pangr];
```

添加手势的视图

处理手势的方法

```
-(void)pan:(UIPanGestureRecognizer *)recognizer
{

        //处理手势的代码

}
```

• 方法二：在Storyboard利用IB创建



Pan Gesture Recognizer

Touches	1	✓ Minimum
	∞	Maximum

Gesture Recognizer

State ✓ Enabled

Behavior ✓ Cancels touches in view
Delays touches began
✓ Delays touches ended

Pinch Gesture Recognizer – Provides a recognizer for pinch gestures which are invoked on the view.

Rotation Gesture Recognizer – Provides a recognizer for rotation gestures which are invoked on the view.

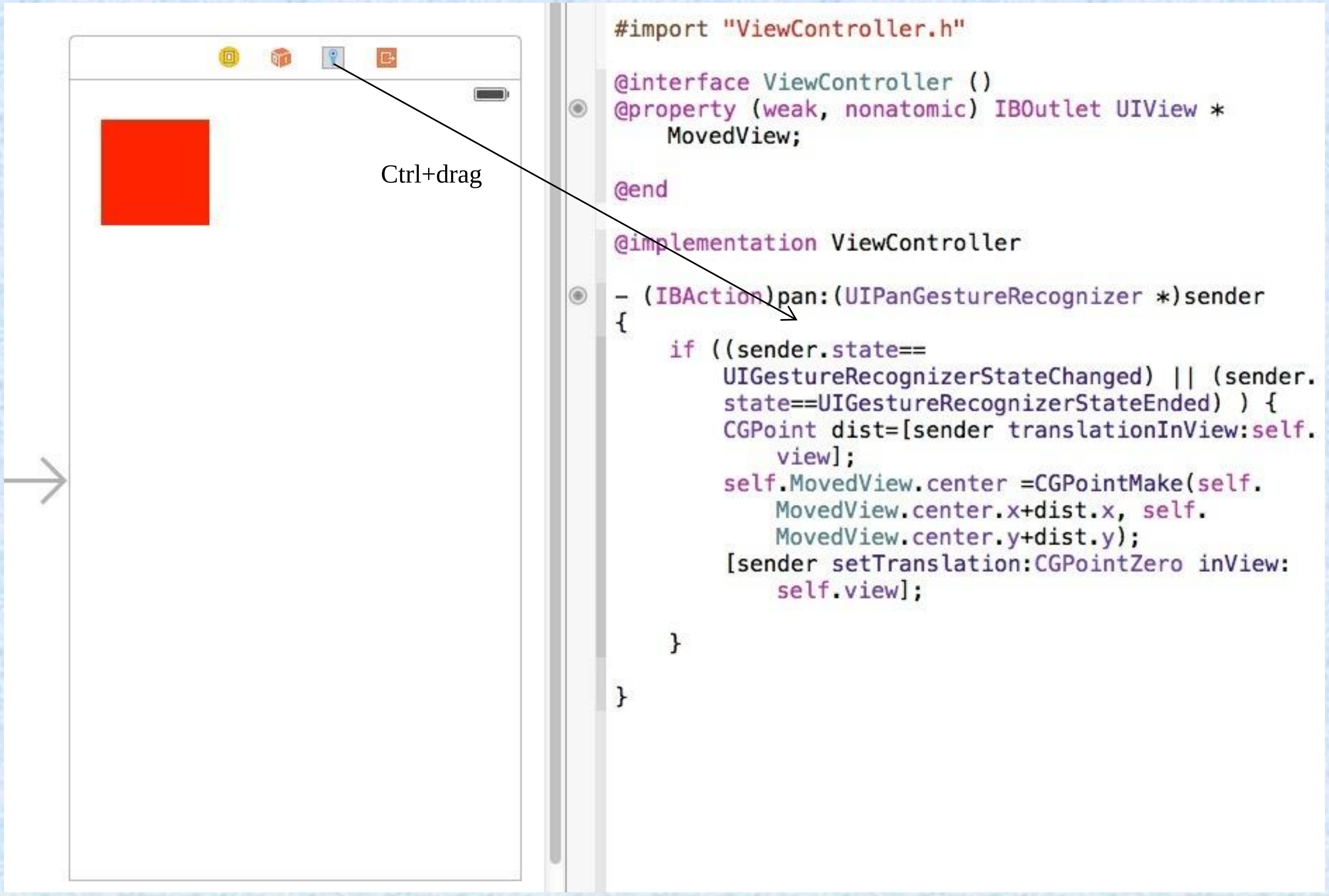
Swipe Gesture Recognizer – Provides a recognizer for swipe gestures which are invoked on the view.

Pan Gesture Recognizer – Provides a recognizer for panning (dragging) gestures which are invoked on the view.

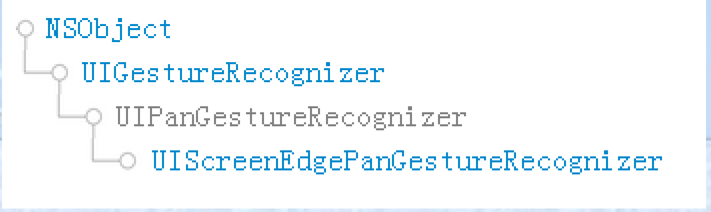
Recognizer – Provides a recognizer for panning (dragging) gestures which are invoked on the view and start near an edge of the

Long Press Gesture Recognizer – Provides a recognizer for long press gestures which are invoked on the view.

推荐使用Storyboard进行创建



```
#import "ViewController.h"

@interface ViewController ()
@property (weak, nonatomic) IBOutlet UIView *
    MovedView;

@end

@implementation ViewController

- (IBAction)pan:(UIPanGestureRecognizer *)sender
{
    if ((sender.state==
        UIGestureRecognizerStateChanged) || (sender.
        state==UIGestureRecognizerStateEnded) ) {
        CGPoint dist=[sender translationInView:self.
            view];
        self.MovedView.center =CGPointMake(self.
            MovedView.center.x+dist.x, self.
            MovedView.center.y+dist.y);
        [sender setTranslation:CGPointZero inView:
            self.view];

    }

}
```

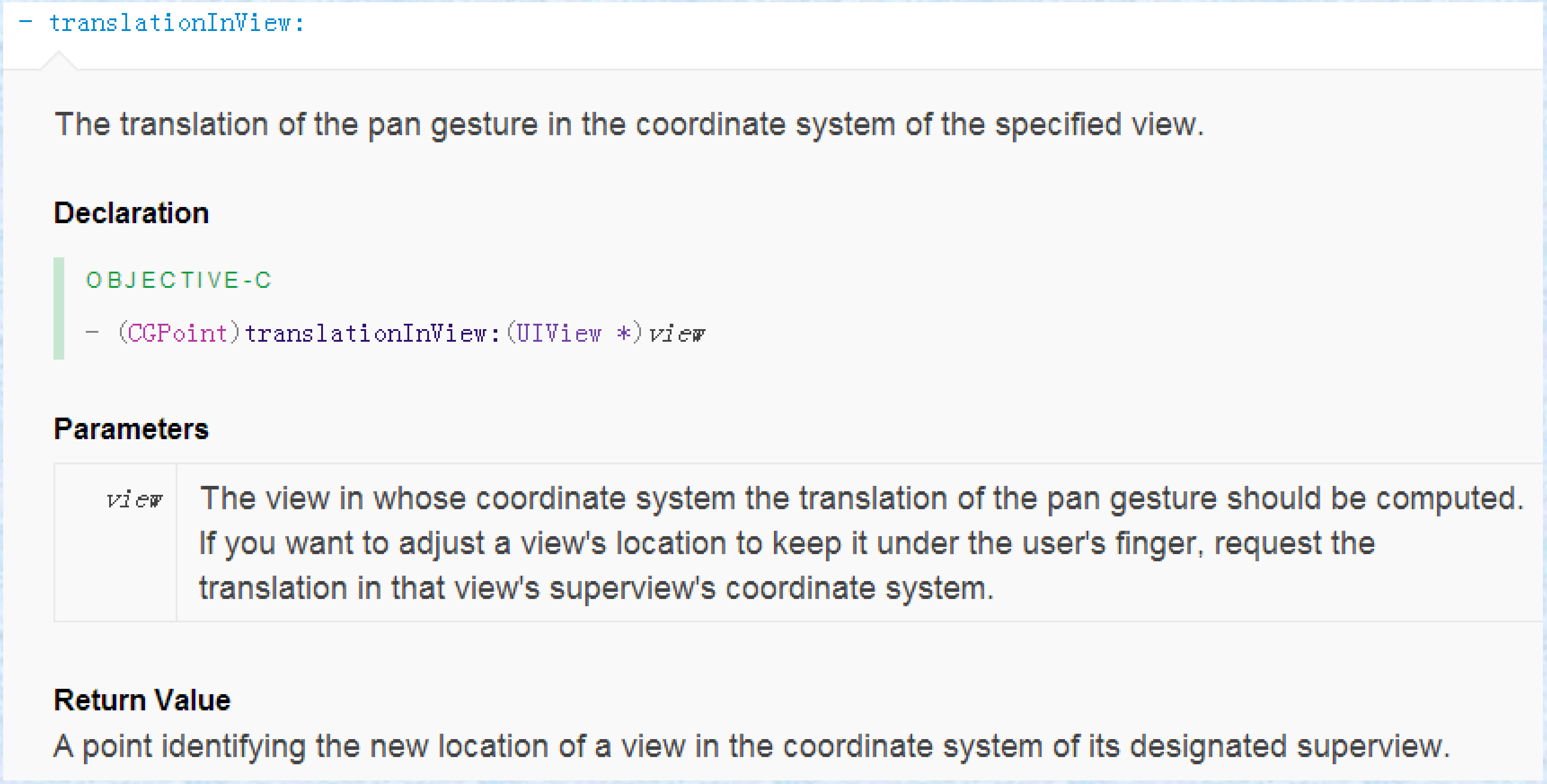


5、 UIPanGestureRecognizer

UIPanGestureRecognizer is a concrete subclass of UIGestureRecognizer that looks for panning (dragging) gestures. The user must be pressing one or more fingers on a view while they pan it. Clients implementing the action method for this gesture recognizer can ask it for the current translation and velocity of the gesture.

A panning gesture is continuous. It begins (UIGestureRecognizerStateBegan) when the minimum number of fingers allowed (minimumNumberOfTouches) has moved enough to be considered a pan. It changes (UIGestureRecognizerStateChanged) when a finger moves while at least the minimum number of fingers are pressed down. It ends (UIGestureRecognizerStateEnded) when all fingers are lifted.

UIGestureRecognizer的几个重要的方法



- translationInView:

The translation of the pan gesture in the coordinate system of the specified view.

Declaration

OBJECTIVE-C

```
- (CGPoint)translationInView:(UIView *)view
```

Parameters

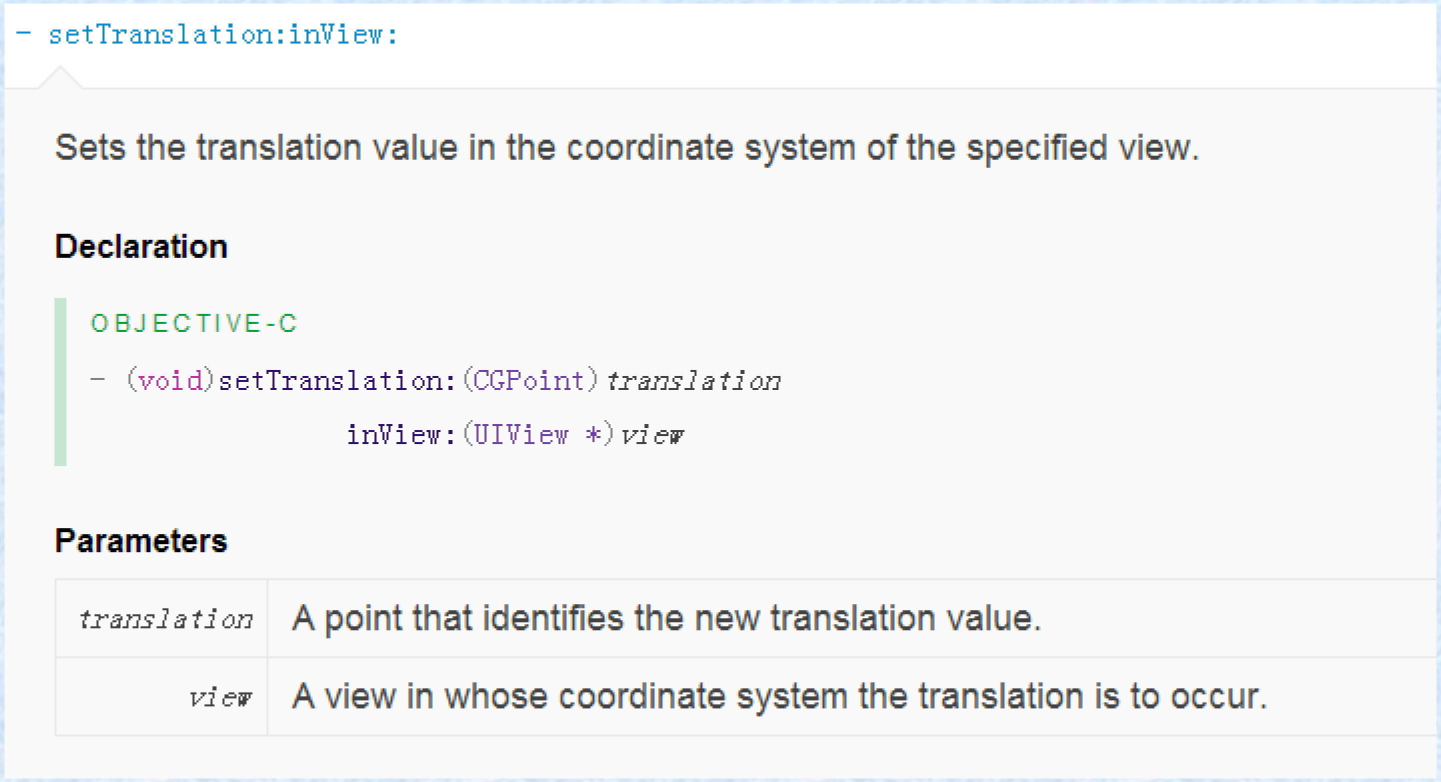
view	The view in whose coordinate system the translation of the pan gesture should be computed. If you want to adjust a view's location to keep it under the user's finger, request the translation in that view's superview's coordinate system.

Return Value

A point identifying the new location of a view in the coordinate system of its designated superview.

通过该方法可获得移动的位置距离

UIGestureRecognizer的几个重要的方法



- setTranslation:inView:

Sets the translation value in the coordinate system of the specified view.

Declaration

OBJECTIVE-C

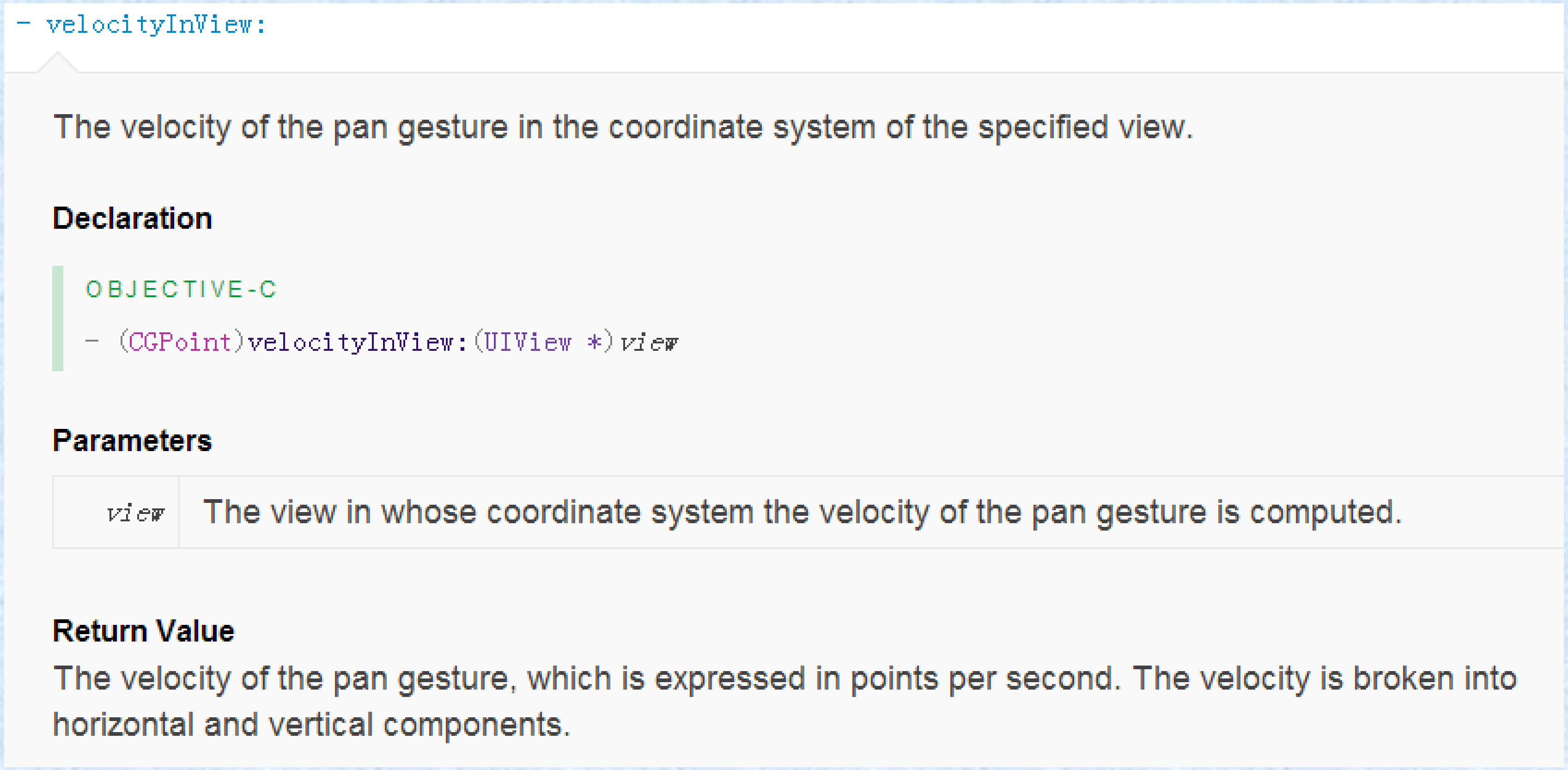
```
- (void)setTranslation:(CGPoint)translation
                inView:(UIView *)view
```

Parameters

translation	A point that identifies the new translation value.
view	A view in whose coordinate system the translation is to occur.

通过该方法可设置移动的视图的初始位置

UIGestureRecognizer的几个重要的方法



```
- velocityInView:
```

The velocity of the pan gesture in the coordinate system of the specified view.

Declaration

OBJECTIVE-C

```
- (CGPoint)velocityInView:(UIView *)view
```

Parameters

view	The view in whose coordinate system the velocity of the pan gesture is computed.

Return Value

The velocity of the pan gesture, which is expressed in points per second. The velocity is broken into horizontal and vertical components.

通过该方法获得移动速度相关信息

5、 其它的一些手势识别器

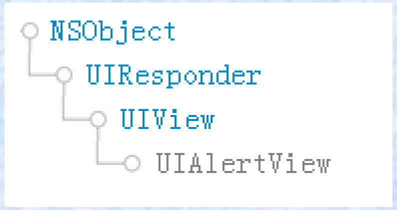
UIPinchGestureRecognizer
@property CGFloat scale; // 只读，放大或缩小的尺度
@property (readonly) CGFloat velocity; // 缩放的速度

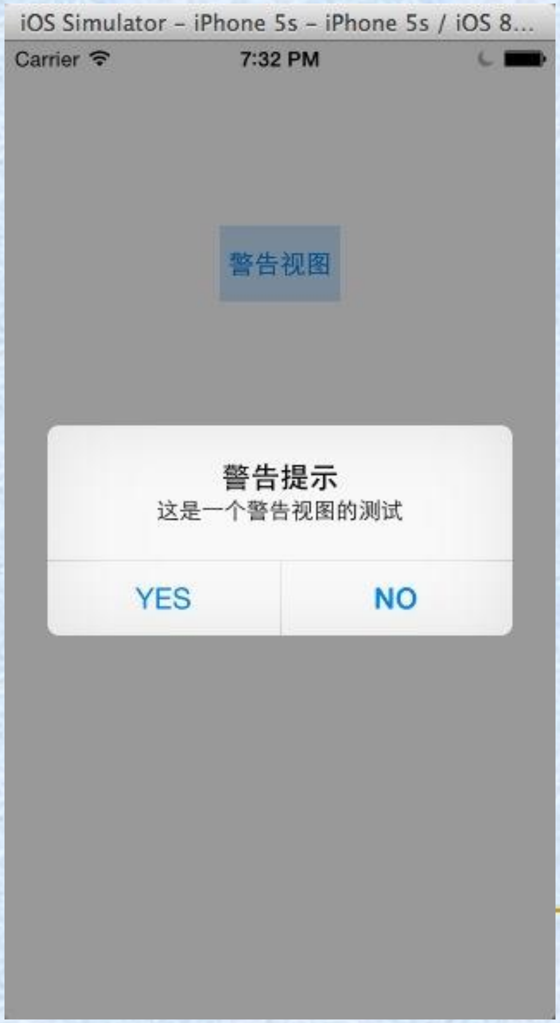
UIRotationGestureRecognizer
@property CGFloat rotation; // 旋转的角度
@property (readonly) CGFloat velocity; // 只读，旋转的速度

UITapGestureRecognizer
@property NSUInteger numberOfTapsRequired; // 单次还是多次轻拍
@property NSUInteger numberOfTouchesRequired; // 手指数量

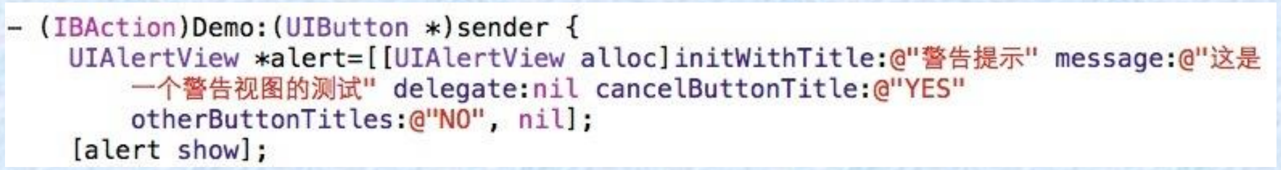
三、警告视图与动作表单

1、警告视图（UIAlertView）

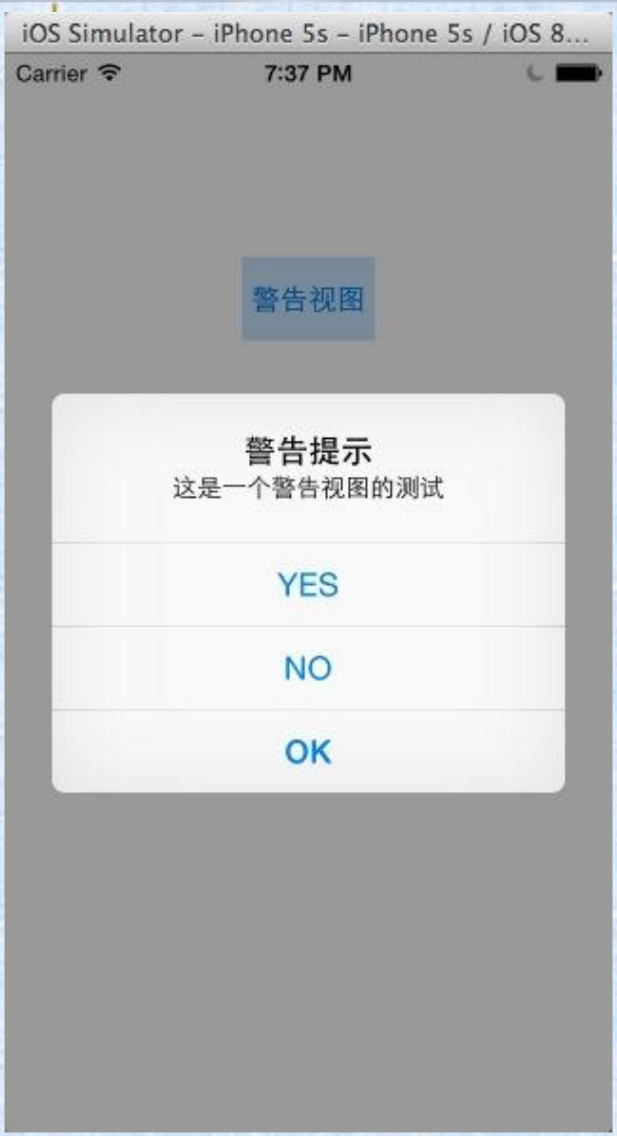




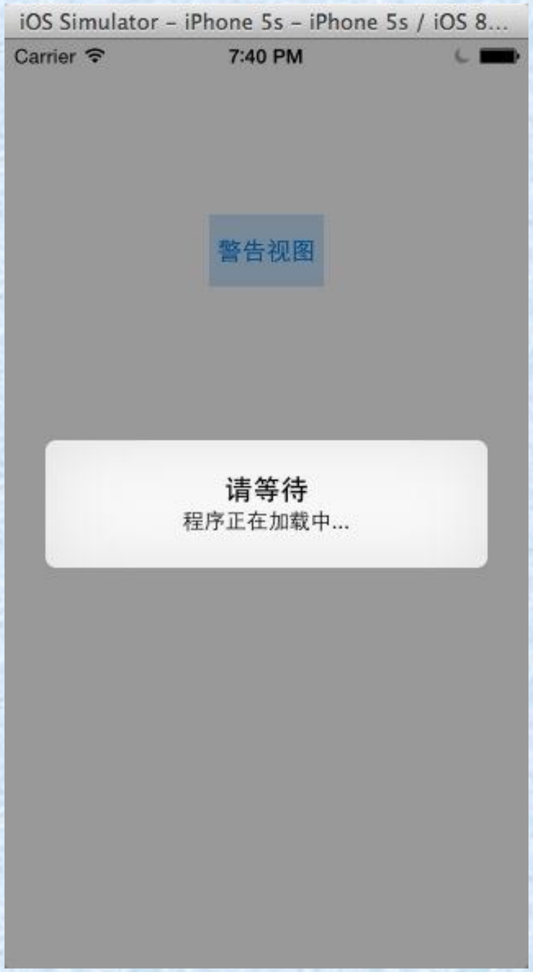
警告提示
这是一个警告视图的测试

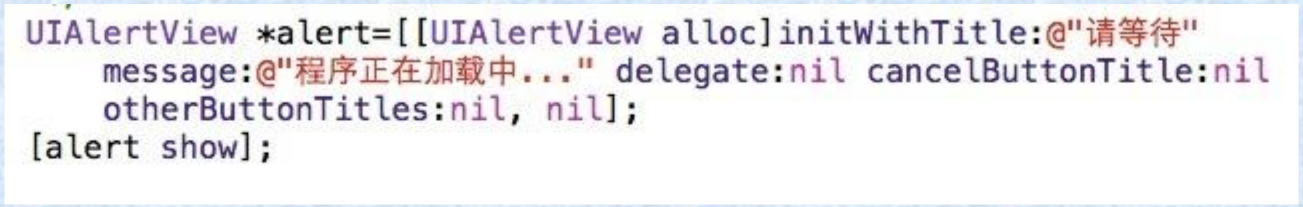


```
- (IBAction)Demo:(UIButton *)sender {
    UIAlertView *alert=[[UIAlertView alloc]initWithTitle:@"警告提示" message:@"这是
        一个警告视图的测试" delegate:nil cancelButtonTitle:@"YES"
        otherButtonTitles:@"NO", nil];
    [alert show];
```

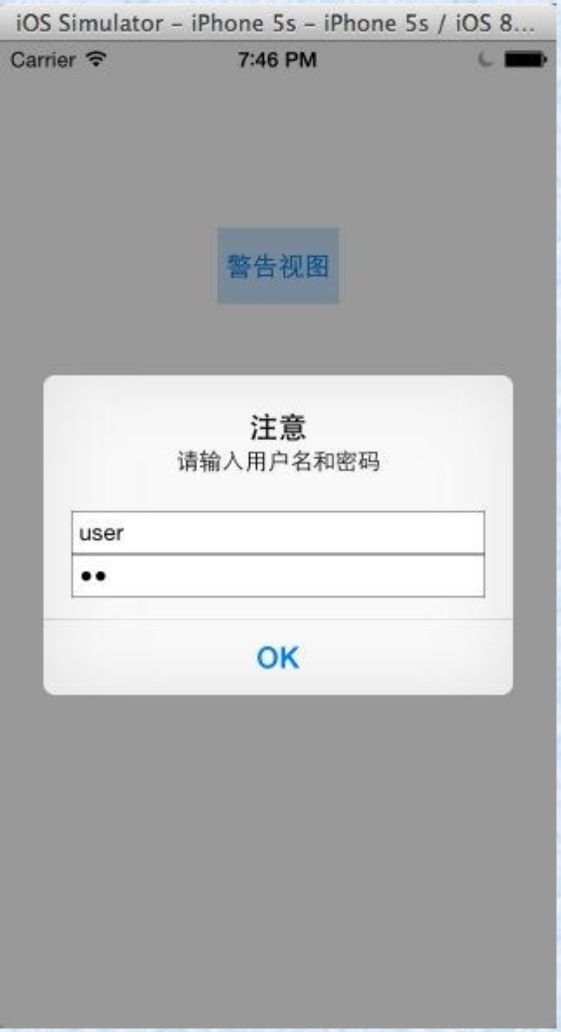


```
UIAlertView *alert=[[UIAlertView alloc]initWithTitle:@"警告提示" message:@"这是
    一个警告视图的测试" delegate:nil cancelButtonTitle:@"OK"
    otherButtonTitles:@"YES",@"NO", nil];
[alert show];
```





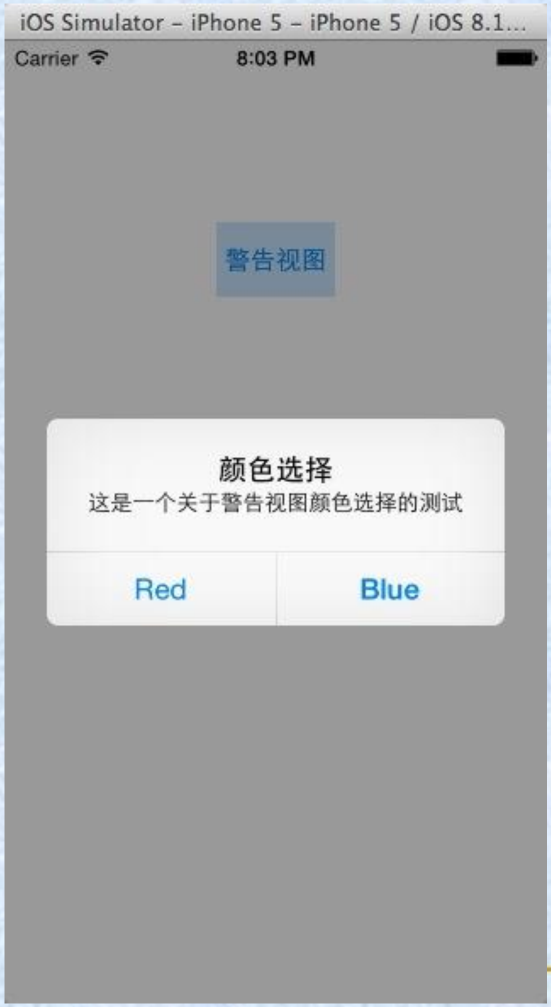
```
UIAlertView *alert=[[UIAlertView alloc]initWithTitle:@"请等待"
    message:@"程序正在加载中..." delegate:nil cancelButtonTitle:nil
    otherButtonTitles:nil, nil];
[alert show];
```



```
UIAlertView *alert=[[UIAlertView alloc]initWithTitle:@"注意"
    message:@"请输入用户名和密码" delegate:nil cancelButtonTitle:@"OK"
    otherButtonTitles:nil, nil];
alert.alertViewStyle=UIAlertViewStyleLoginAndPasswordInput;
[alert show];
```

```
typedef enum {
   UIAlertViewStyleDefault  = 0,
   UIAlertViewStyleSecureTextInput ,
   UIAlertViewStylePlainTextInput ,
   UIAlertViewStyleLoginAndPasswordInput
} UIAlertViewStyle;
```

响应警告视图


iOS Simulator – iPhone 5 – iPhone 5 / iOS 8.1…
Carrier
8:03 PM
警告视图
颜色选择
这是一个关于警告视图颜色选择的测试
Red
Blue

```
UIAlertView *alert=[[UIAlertView alloc]initWithTitle:@"颜色选择"
    message:@"这是一个关于警告视图颜色选择的测试" delegate:self
    cancelButtonTitle:@"Red" otherButtonTitles:@"Blue", nil];
[alert show];
```

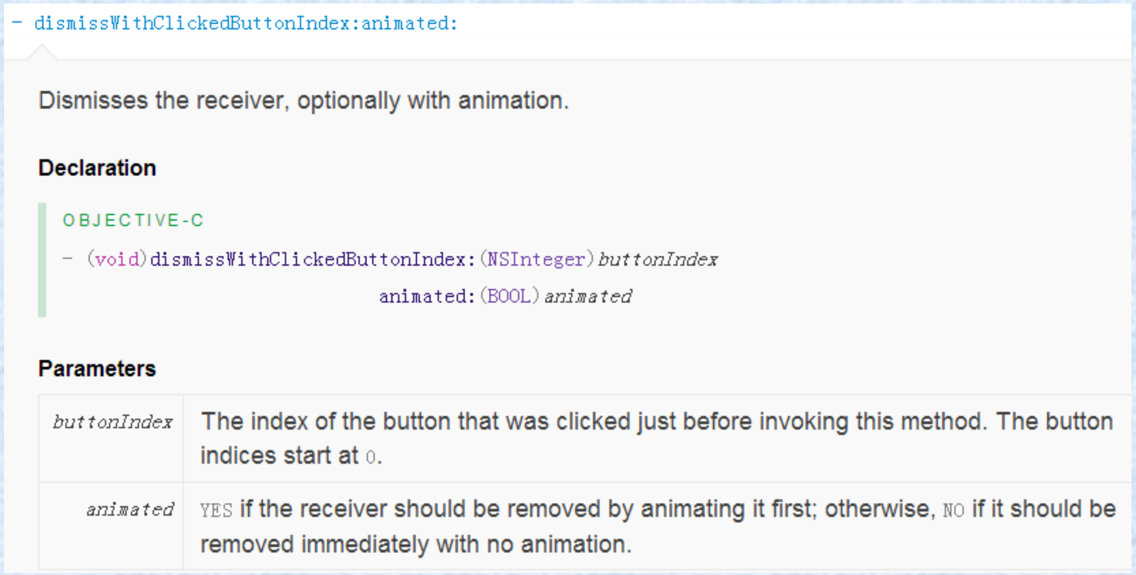
委托给自己

```
-(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex
{
    if ([[alertView buttonTitleAtIndex:buttonIndex] isEqualToString:@"Red"])  {
        self.view.backgroundColor=[UIColor redColor];
    }else{
      self.view.backgroundColor=[UIColor blueColor];
        }
}
```

```
@interface ViewController : UIViewController<UIAlertViewDelegate>


@end
```

委托协议

如何让上面警告视图消失？



- dismissWithClickedButtonIndex:animated:

Dismisses the receiver, optionally with animation.

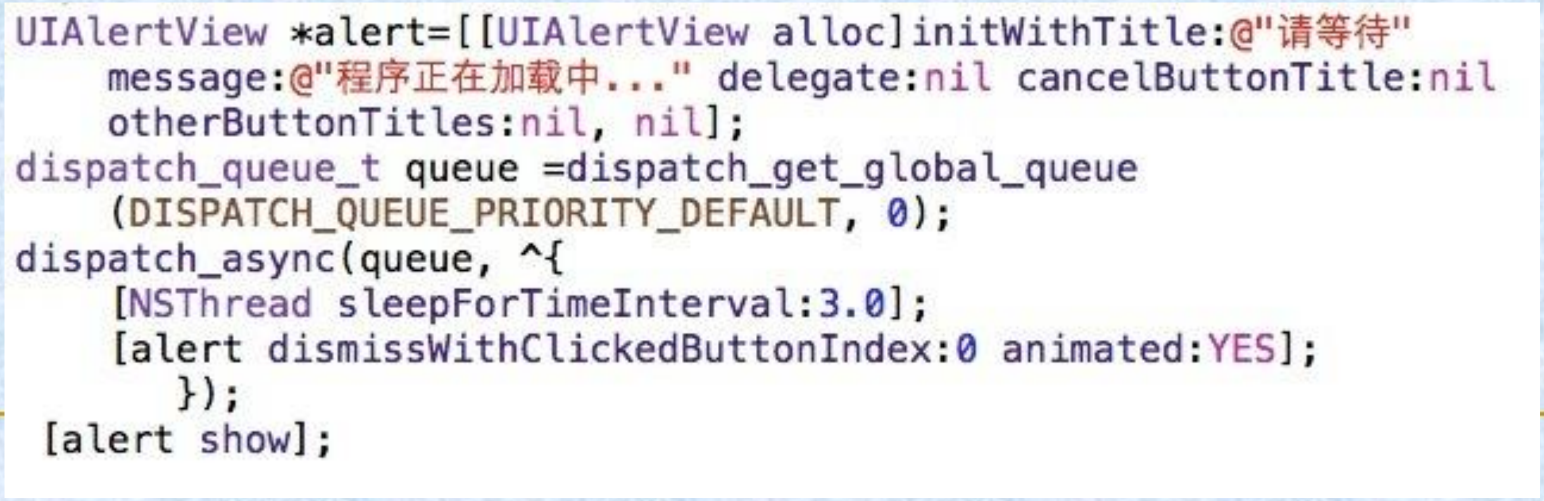
Declaration

OBJECTIVE-C

```
- (void)dismissWithClickedButtonIndex:(NSInteger)buttonIndex
                             animated:(BOOL)animated
```

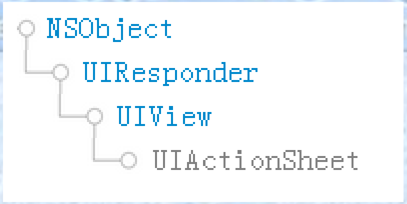
Parameters

buttonIndex	The index of the button that was clicked just before invoking this method. The button indices start at 0.
animated	YES if the receiver should be removed by animating it first; otherwise, NO if it should be removed immediately with no animation.

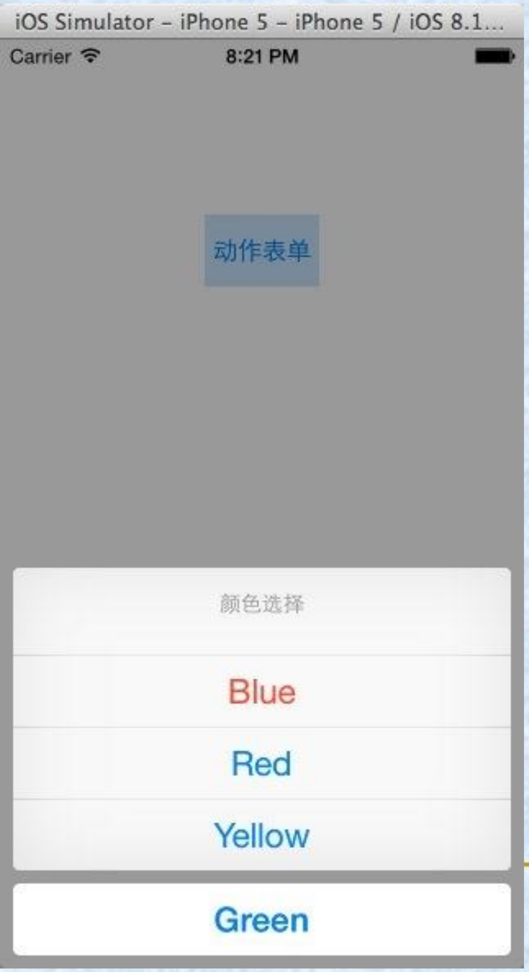


```
UIAlertView *alert=[[UIAlertView alloc]initWithTitle:@"请等待"
    message:@"程序正在加载中..." delegate:nil cancelButtonTitle:nil
    otherButtonTitles:nil, nil];
dispatch_queue_t queue =dispatch_get_global_queue
    (DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
dispatch_async(queue, ^{
    [NSThread sleepForTimeInterval:3.0];
    [alert dismissWithClickedButtonIndex:0 animated:YES];
      });
 [alert show];
```

2、动作表单（UIActionSheet）



```
NSObject
  UIResponder
    UIView
      UIActionSheet
```



```
UIActionSheet *actionsheet=[[UIActionSheet alloc] initWithTitle:@"颜色选择"
    delegate:self cancelButtonTitle:@"Green" destructiveButtonTitle:@"Blue"
    otherButtonTitles:@"Red",@"Yellow", nil];
[actionsheet showInView:self.view];
```

委托给自己

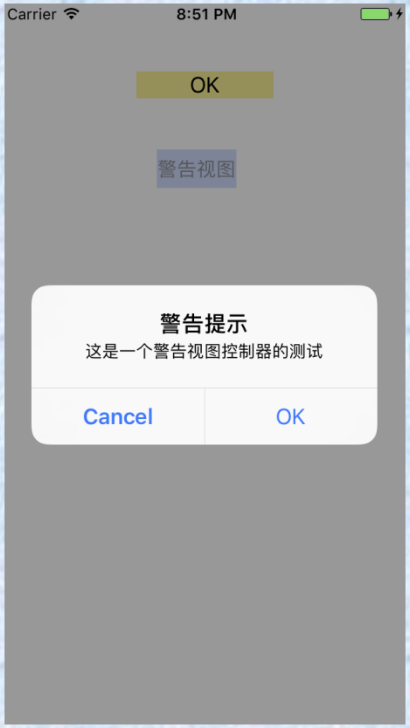
```
-(void)actionSheet:(UIActionSheet *)actionSheet clickedButtonAtIndex:(NSInteger)
    buttonIndex
{
    if ([[actionSheet buttonTitleAtIndex:buttonIndex] isEqualToString:@"Red"])  {
        self.view.backgroundColor=[UIColor redColor];
    }else if ([[actionSheet buttonTitleAtIndex:buttonIndex] isEqualToString:@"Blue"]){
        self.view.backgroundColor=[UIColor blueColor];
    }else if ([[actionSheet buttonTitleAtIndex:buttonIndex] isEqualToString:@"Yellow"]
        ){
        self.view.backgroundColor=[UIColor yellowColor];}
        else
           {
          self.view.backgroundColor=[UIColor greenColor];
        }
}
```

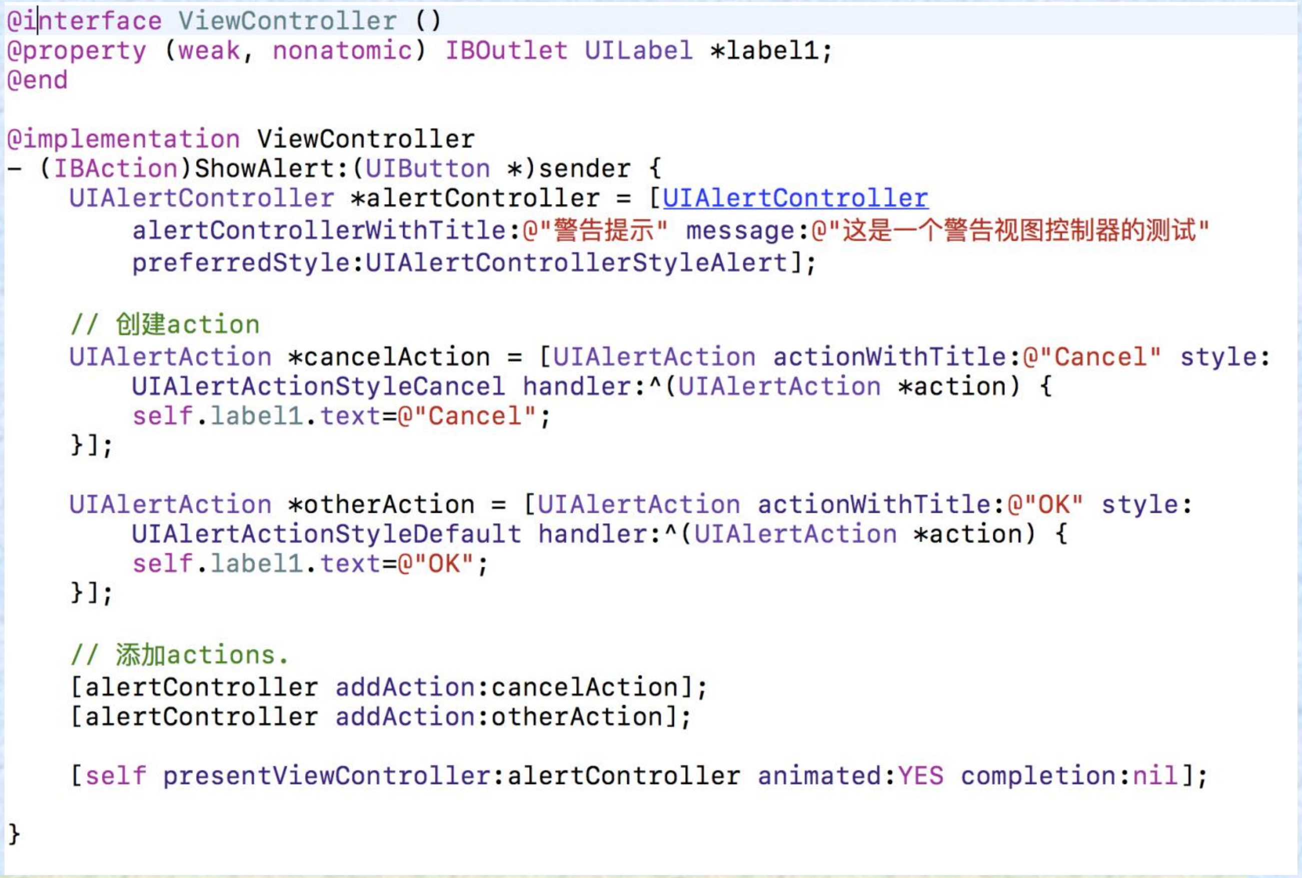
委托协议

```
@interface ViewController:UIViewController<UIActionSheetDelegate>


@end
```

3、UIAlertController

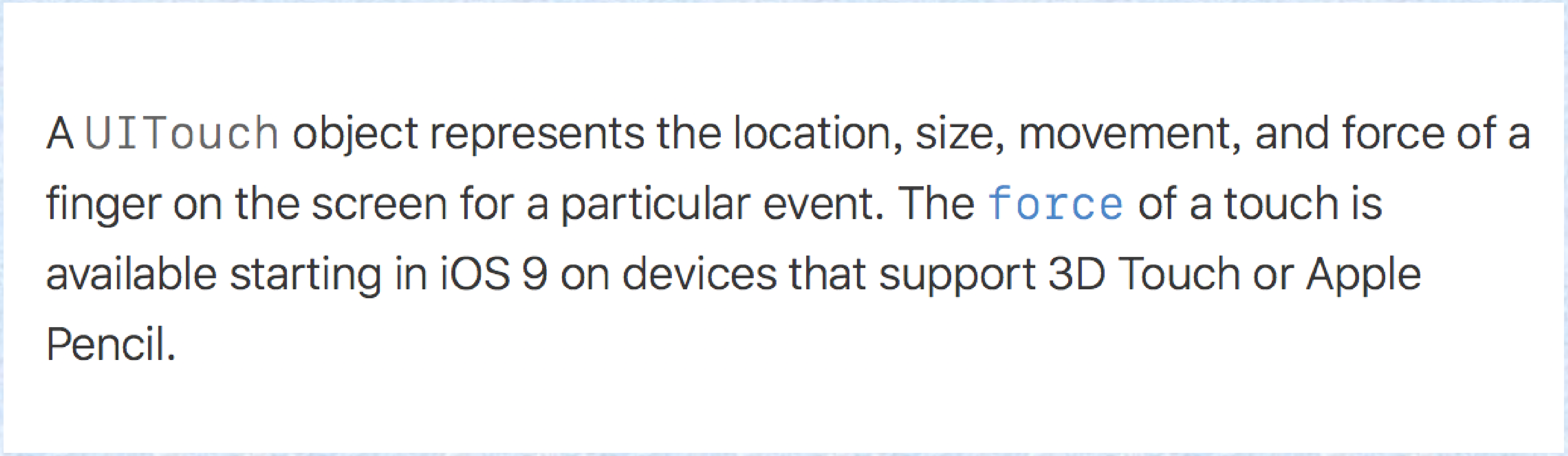
iOS9后建议使用UIAlertController



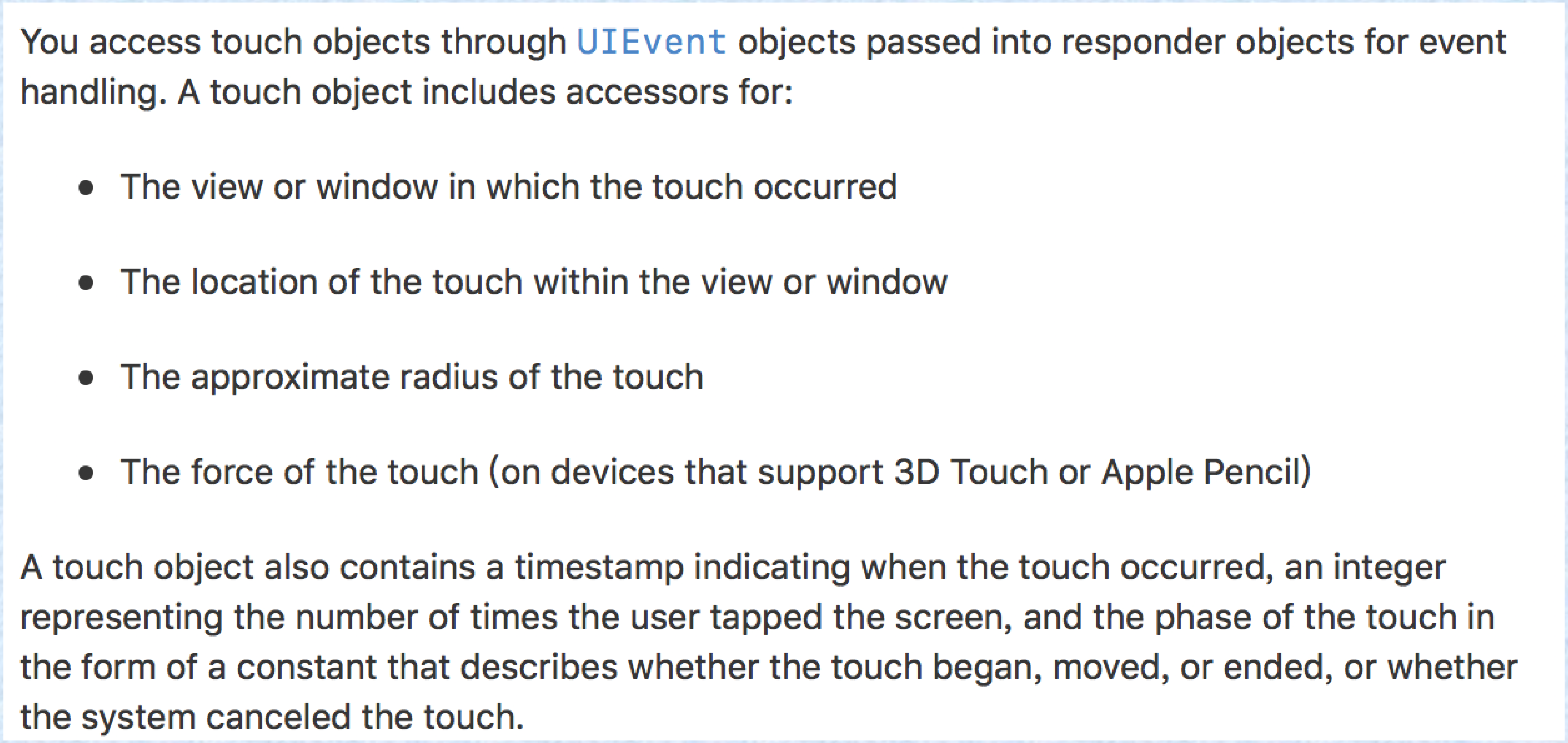


```
@interface ViewController ()
@property (weak, nonatomic) IBOutlet UILabel *label1;
@end

@implementation ViewController
- (IBAction)ShowAlert:(UIButton *)sender {
    UIAlertController *alertController = [UIAlertController
        alertControllerWithTitle:@"警告提示" message:@"这是一个警告视图控制器的测试"
        preferredStyle:UIAlertControllerStyleAlert];

    // 创建action
    UIAlertAction *cancelAction = [UIAlertAction actionWithTitle:@"Cancel" style:
        UIAlertActionStyleCancel handler:^(UIAlertAction *action) {
        self.label1.text=@"Cancel";
    }];

    UIAlertAction *otherAction = [UIAlertAction actionWithTitle:@"OK" style:
        UIAlertActionStyleDefault handler:^(UIAlertAction *action) {
        self.label1.text=@"OK";
    }];

    // 添加actions.
    [alertController addAction:cancelAction];
    [alertController addAction:otherAction];

    [self presentViewController:alertController animated:YES completion:nil];

}
```

四、触摸事件UITouch



A `UITouch` object represents the location, size, movement, and force of a finger on the screen for a particular event. The `force` of a touch is available starting in iOS 9 on devices that support 3D Touch or Apple Pencil.

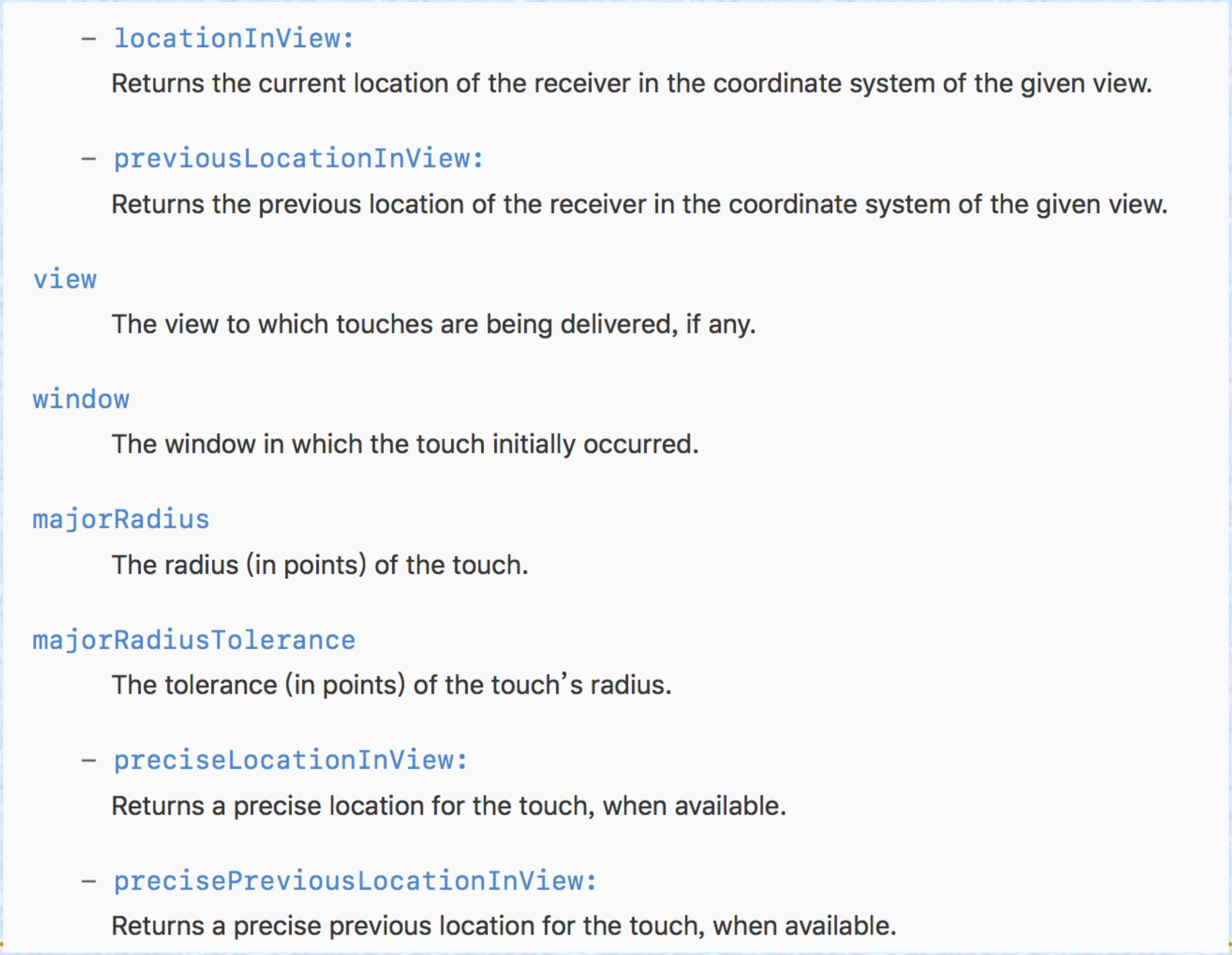


You access touch objects through `UIEvent` objects passed into responder objects for event handling. A touch object includes accessors for:

- The view or window in which the touch occurred
- The location of the touch within the view or window
- The approximate radius of the touch
- The force of the touch (on devices that support 3D Touch or Apple Pencil)

A touch object also contains a timestamp indicating when the touch occurred, an integer representing the number of times the user tapped the screen, and the phase of the touch in the form of a constant that describes whether the touch began, moved, or ended, or whether the system canceled the touch.

重要的属性与方法



– `locationInView:`

Returns the current location of the receiver in the coordinate system of the given view.

– `previousLocationInView:`

Returns the previous location of the receiver in the coordinate system of the given view.

`view`

The view to which touches are being delivered, if any.

`window`

The window in which the touch initially occurred.

`majorRadius`

The radius (in points) of the touch.

`majorRadiusTolerance`

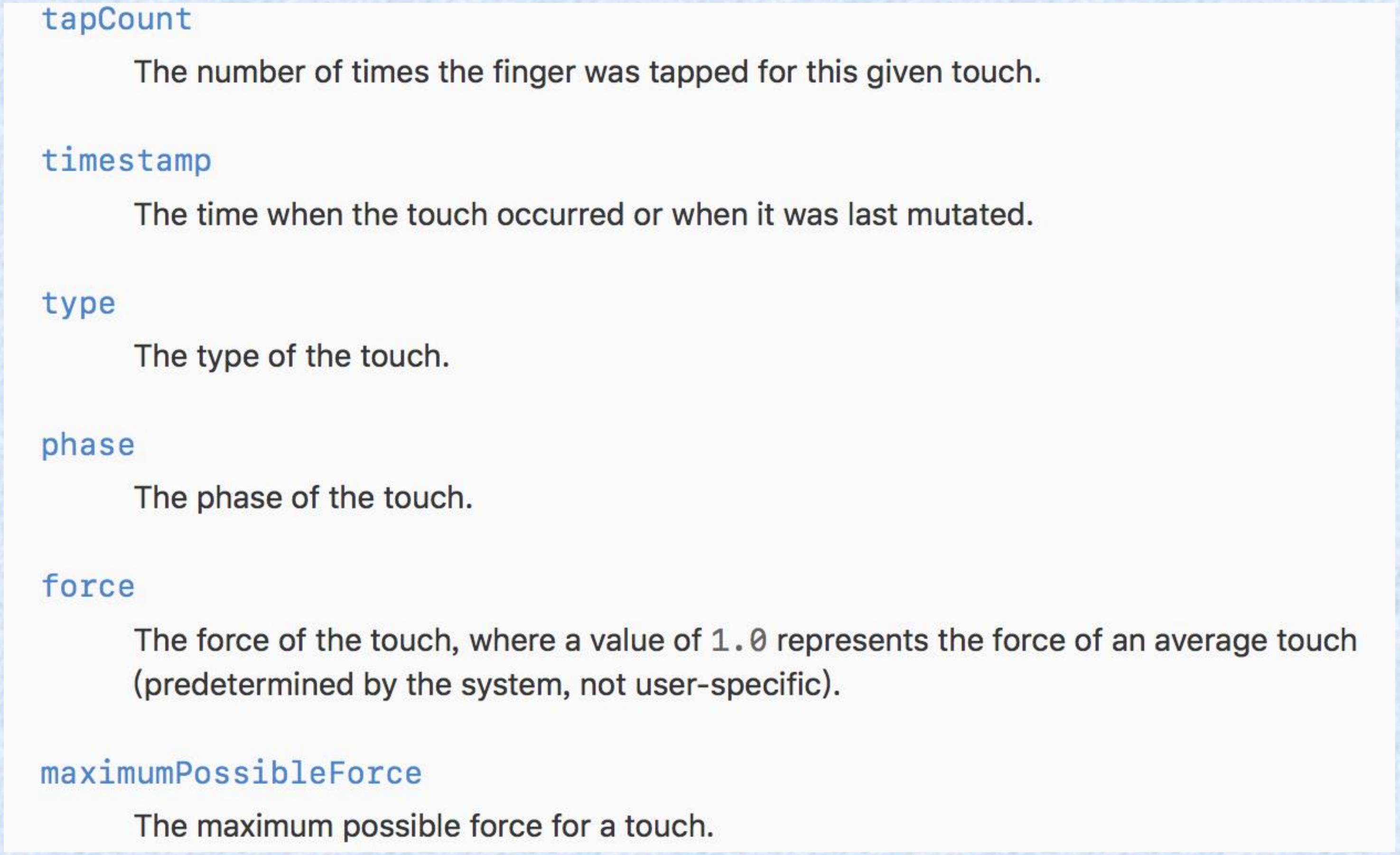
The tolerance (in points) of the touch’s radius.

– `preciseLocationInView:`

Returns a precise location for the touch, when available.

– `precisePreviousLocationInView:`

Returns a precise previous location for the touch, when available.



`tapCount`

The number of times the finger was tapped for this given touch.

`timestamp`

The time when the touch occurred or when it was last mutated.

`type`

The type of the touch.

`phase`

The phase of the touch.

`force`

The force of the touch, where a value of `1.0` represents the force of an average touch (predetermined by the system, not user-specific).

`maximumPossibleForce`

The maximum possible force for a touch.

phase的几个状态

UITouchPhaseBegan	触摸开始
UITouchPhaseMoved	接触点移动
UITouchPhaseStationary	接触点无移动
UITouchPhaseEnded	触摸结束
UITouchPhaseCancelled	触摸取消

事件处理方法

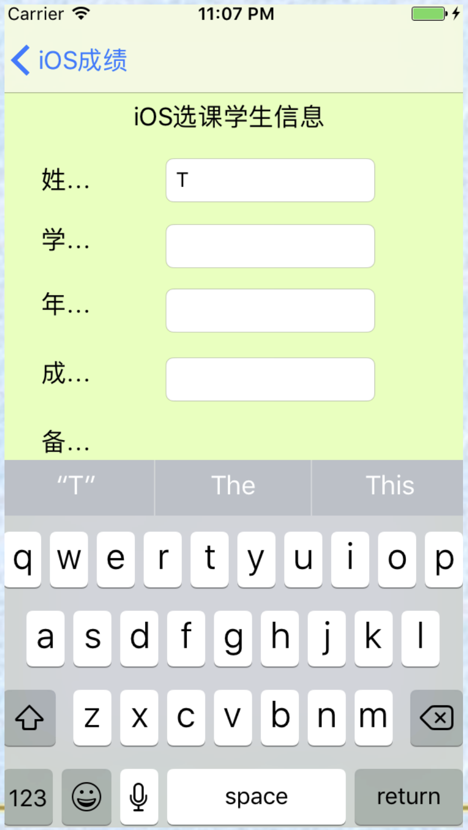
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
开始时触发

-(void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
移动时触发

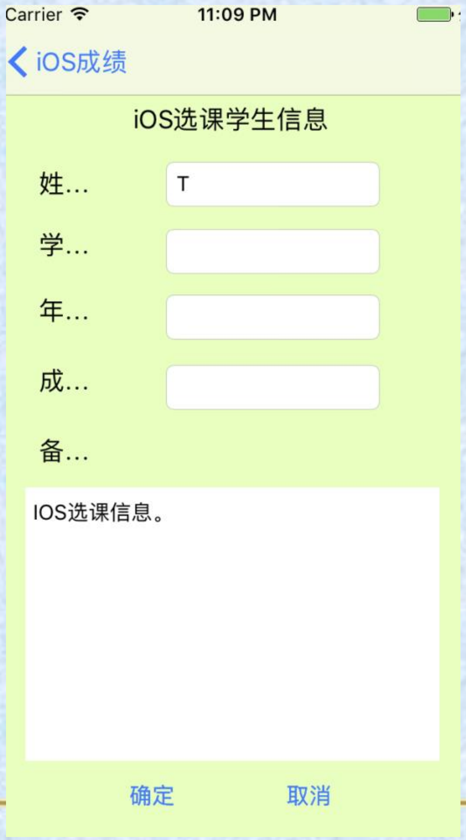
-(void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
结束时触发

-(void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event;
取消时触发

```
-(void)touchesBegan:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event
{

    [self.TxtMemo resignFirstResponder];
    [self.TxtName resignFirstResponder];
    [self.TxtNumber resignFirstResponder];
    [self.TxtAge resignFirstResponder];
    [self.TxtScore resignFirstResponder];

}
```

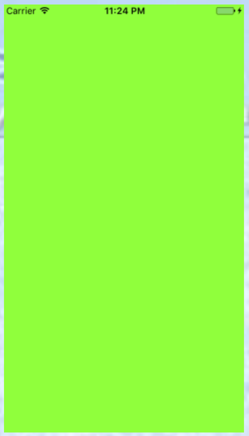


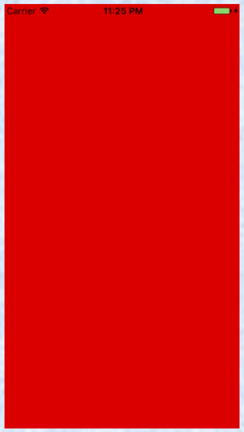
弹出键盘

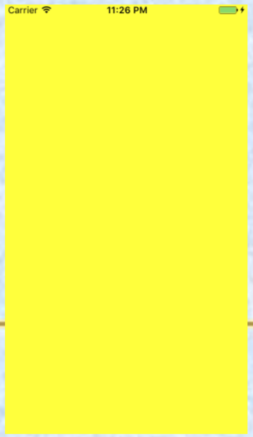


收起键盘

```
@interface ViewController ()
@property (strong, nonatomic) IBOutlet UIView *myview;
@end

@implementation ViewController

-(void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)
    event
{
    UITouch *touch =  [touches anyObject];
    if(touch.tapCount == 1)
    {
        self.myview.backgroundColor = [UIColor greenColor];
    }
    if(touch.tapCount == 2)
    {
        self.myview.backgroundColor = [UIColor redColor];
    }
    if(touch.tapCount == 3)
    {
        self.myview.backgroundColor = [UIColor yellowColor];
    }

}
```


Carrier
11:24 PM


Carrier
11:25 PM


Carrier
11:26 PM