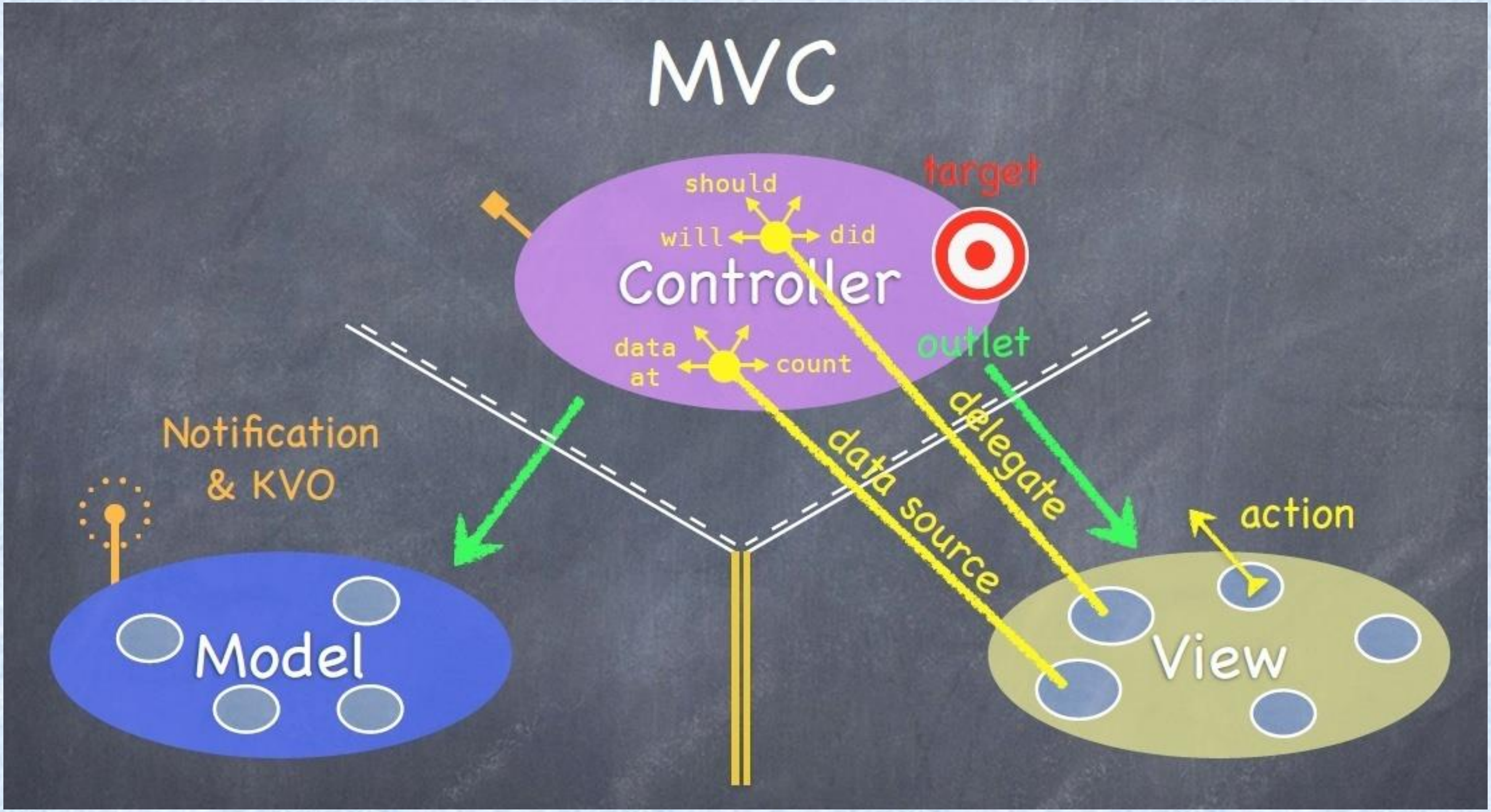
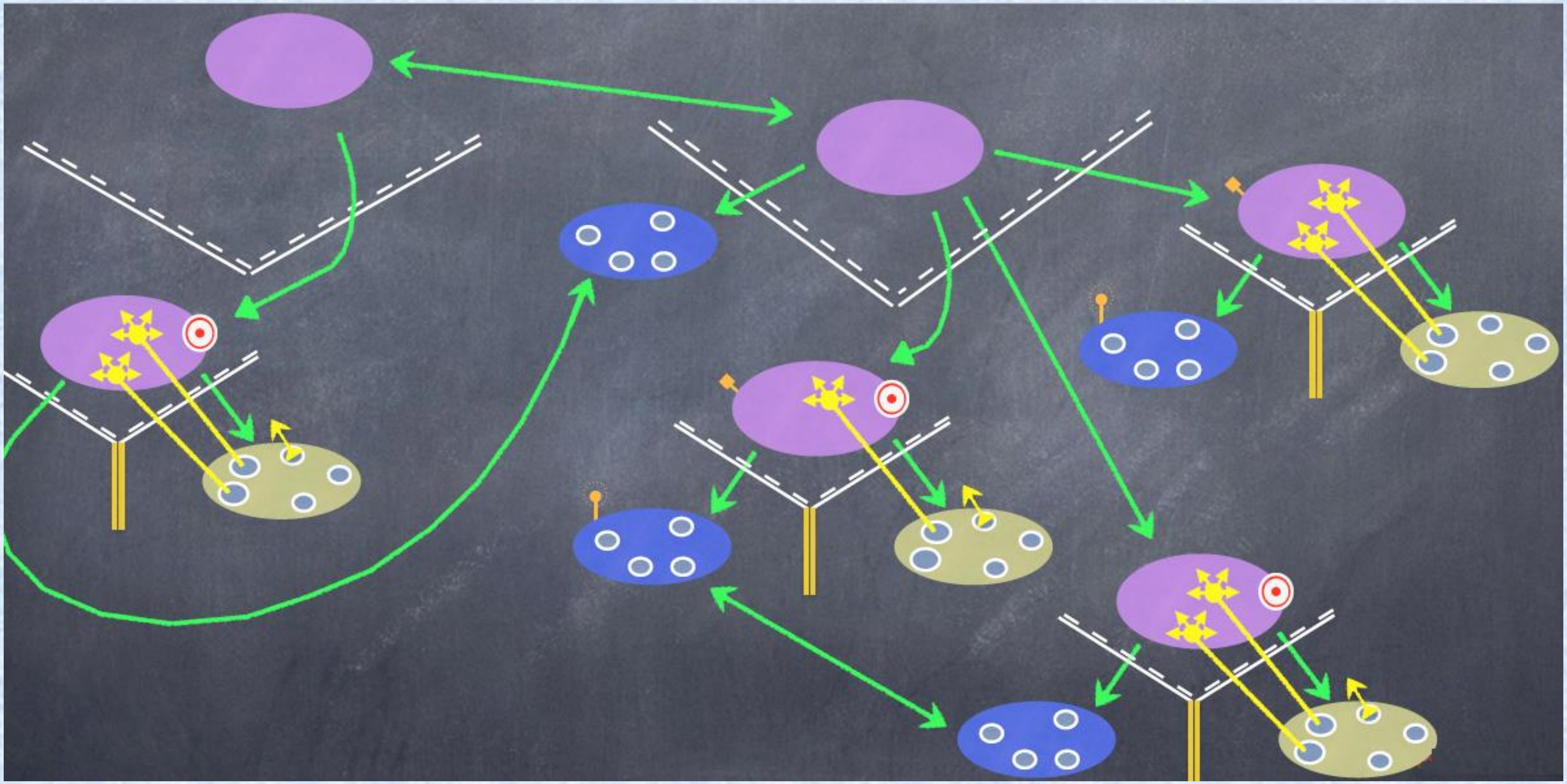
视图与视图控制器

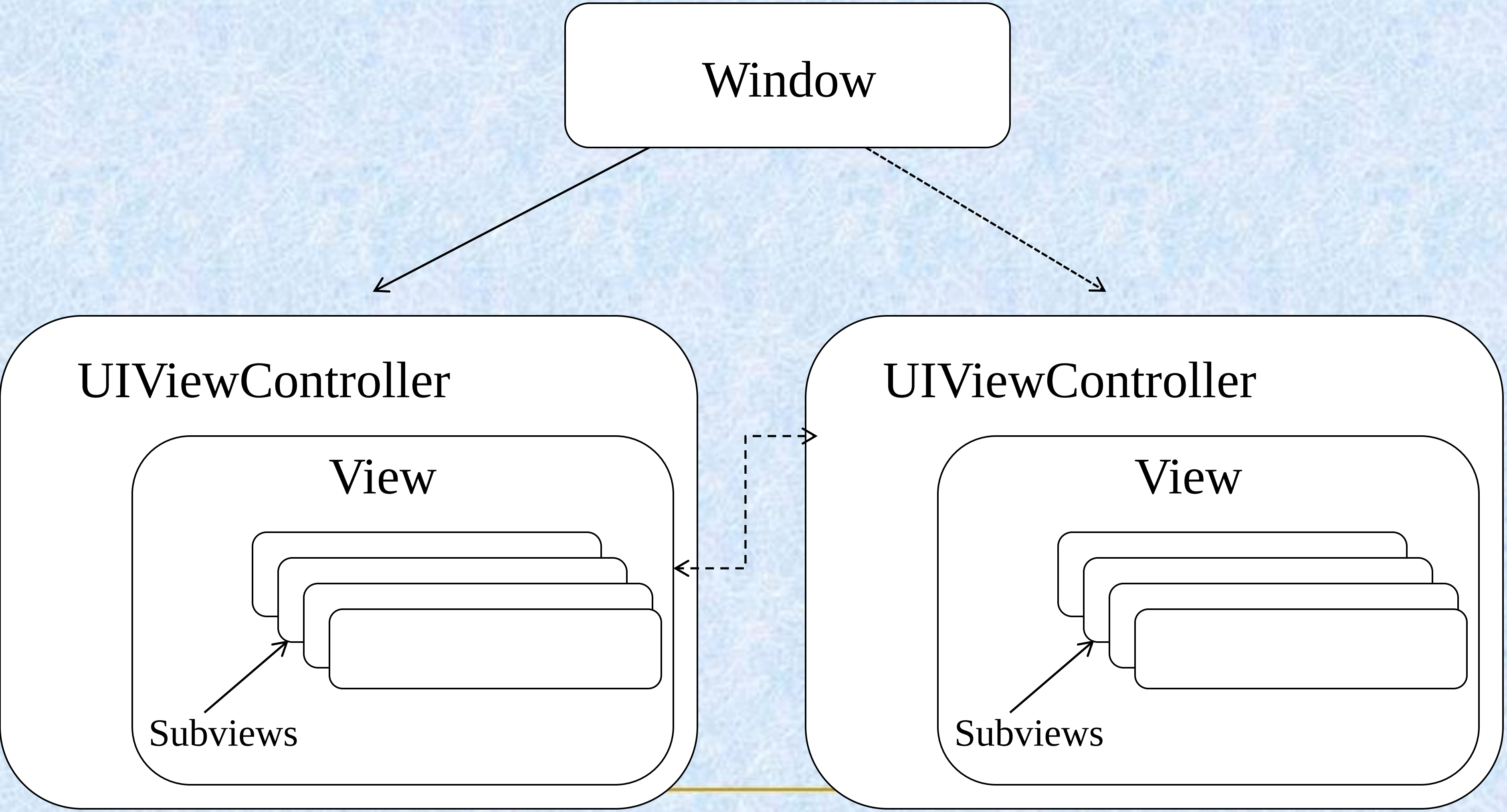
一、MVC

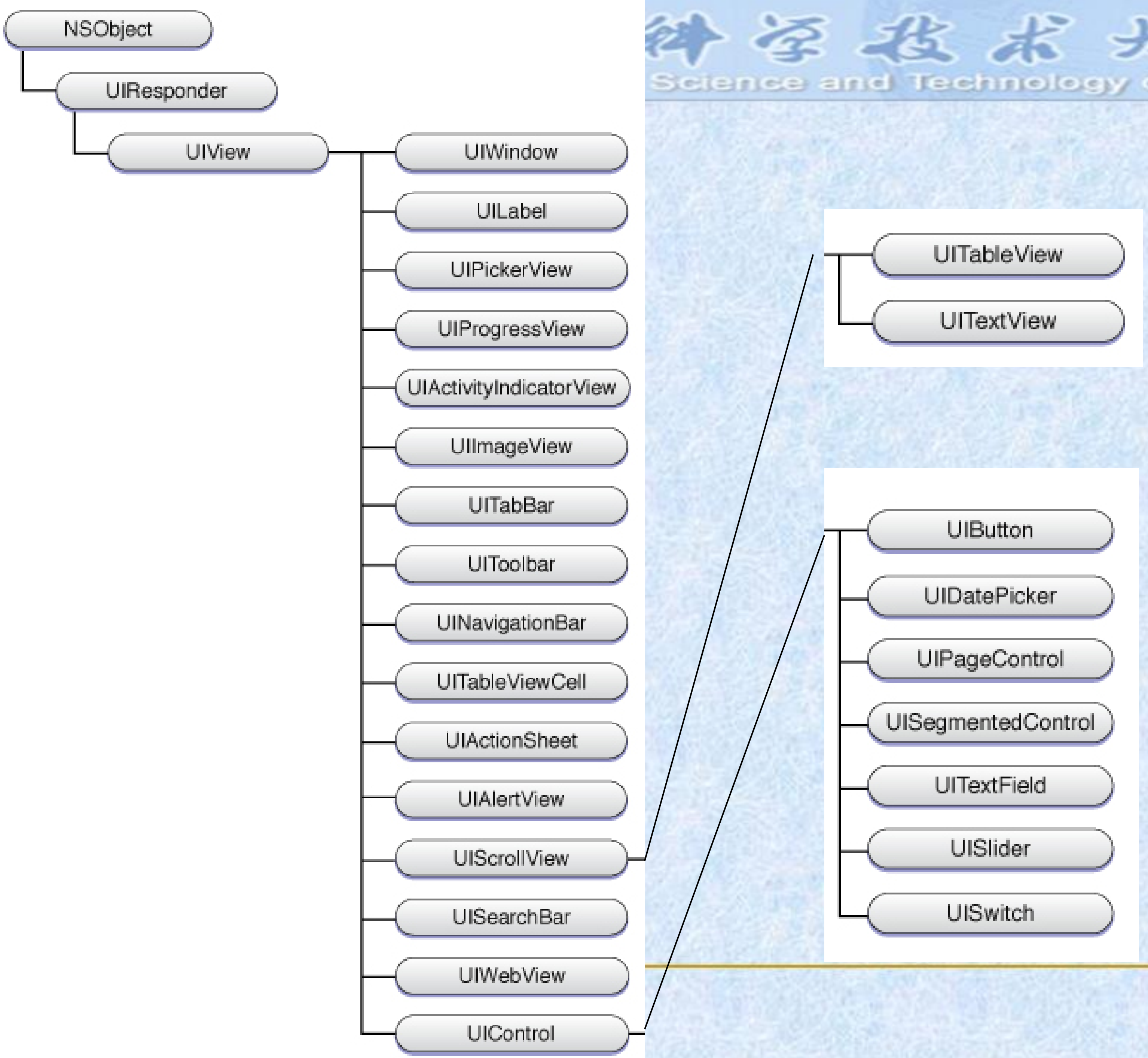




多个MVC协同工作

二、窗口、视图与视图控制器的关系



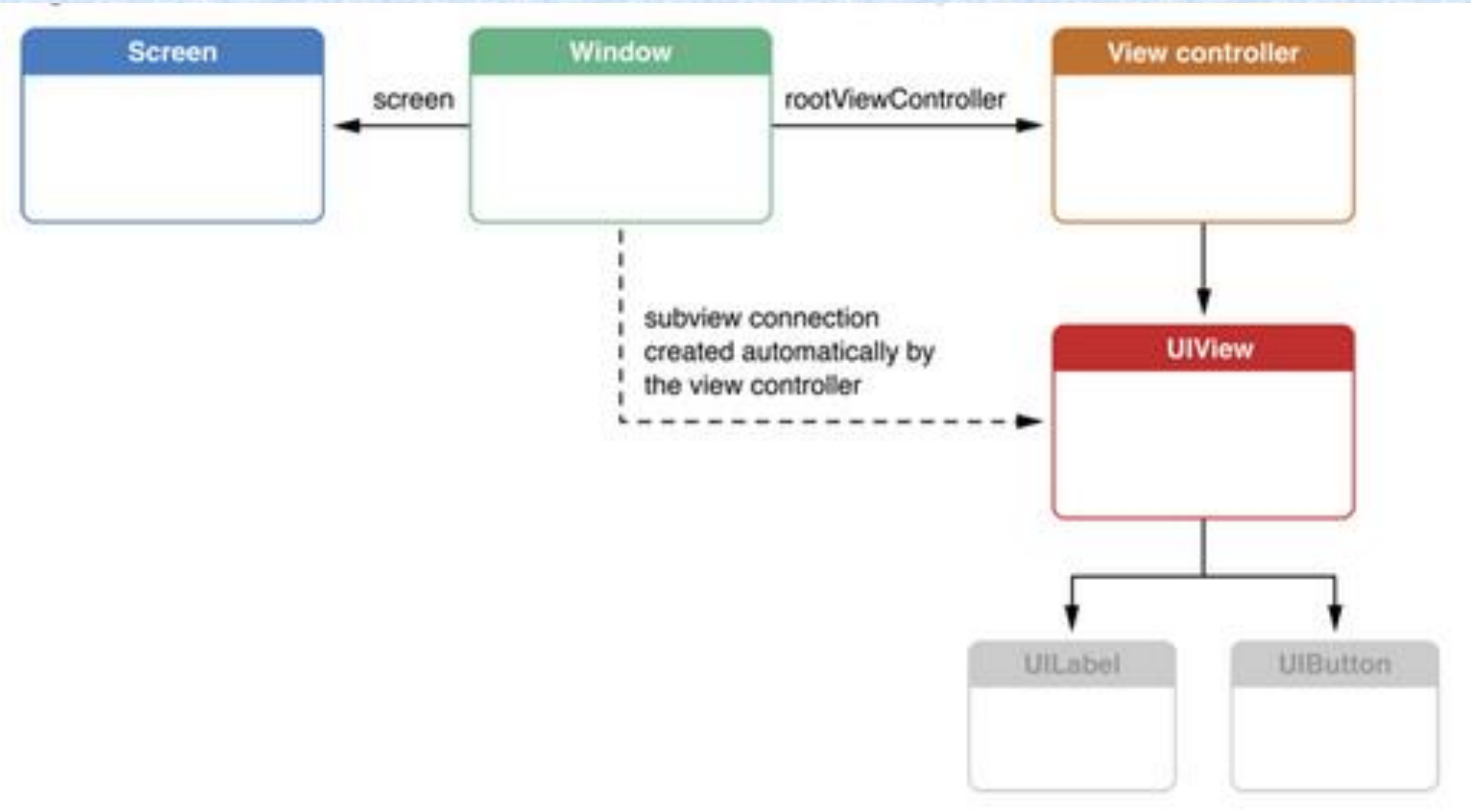


- NSObject
  - UIResponder
    - UIView
      - UIWindow
      - UILabel
      - UIPickerView
      - UIProgressView
      - UIActivityIndicatorView
      - UIImageView
      - UITabBar
      - UIToolbar
      - UINavigationBar
      - UITableViewCell
      - UIActionSheet
      - UIAlertView
      - UIScrollView
        - UITableView
        - UITextView
      - UISearchBar
      - UIWebView
      - UIControl
        - UIButton
        - UIDatePicker
        - UIPageControl
        - UISegmentedControl
        - UITextField
        - UISlider
        - UISwitch

三、窗口

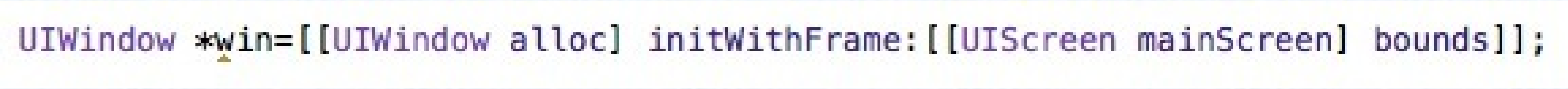
●在iOS中窗口是一种特殊的视图，通常在一个app中只会有*一个窗口*。

● iOS程序启动完毕后，创建的第一个视图控件就是窗口，接着创建控制器的视图，最后将控制器的视图添加到窗口上，于是控制器的视图就显示在屏幕上了



先创建UIwindow,再创建控制器，创建控制器的view，然后将控制器的view添加到UIWindow上

● 窗口对应的类是UIWindow。默认情况下其大小与屏幕一致，请勿修改。可以通过下面代码获得窗口对象



```
UIWindow *win=[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
```

● 当在故事板（storyboard）中使用IB创建UI时，会自动帮助创建UIWindow.

注意： Mac中编程可能会用到多个窗口，所有需要用UIWindow,但在iOS中只有一个窗口，所以基本不用UIWindow, 只是使用UIViews

四、视图

1、什么是视图?

●视图表示屏幕上的一块矩形区域，它在App中占有绝对重要的地位，因为iOS中几乎所有可视化控件都是视图或其子类。

●和窗口不同，允许在一个窗口中包含多个视图，并且视图的尺寸是任意尺寸。并且视图可以进行嵌套和响应事件。

- 视图对应的类是UIView及其子类:

创建视图可以有两种方法:

1、 在故事板（storyboard）中利用IB创建。推荐

2、 代码的方法。

视图对象是应用程序和用户交互的主要途径，其主要作用有

●描画和动画

视图负责对其所属的矩形区域进行描画。
某些视图属性变量可以以动画的形式过渡到新的值。
布局和子视图管理

●视图管理着一个子视图列表。

视图定义了自身相对于其父视图的尺寸调整行为。
必要时，视图可以通过代码调整其子视图的尺寸和位置。
视图可以将其坐标系统下的点转换为其它视图或窗口坐标系统下的点。

●事件处理

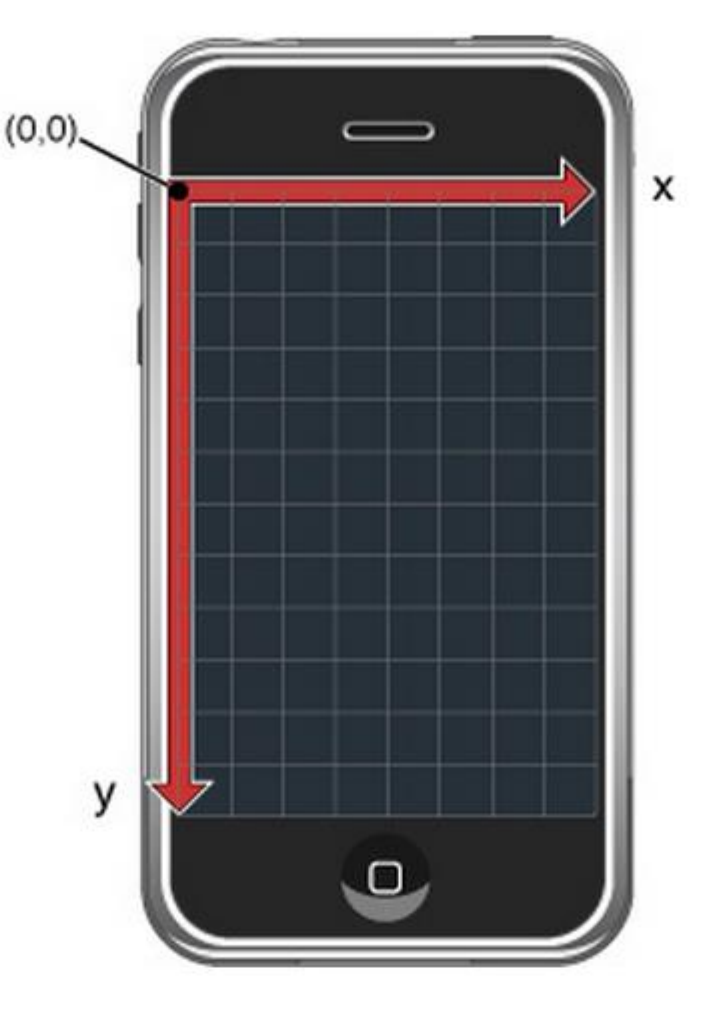
视图可以接收触摸事件。
视图是响应者链的参与者。

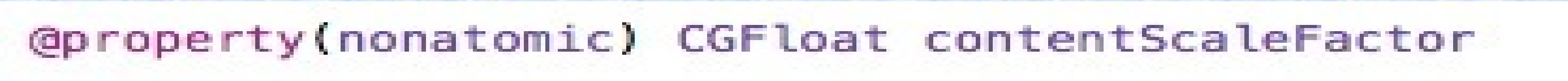
2、视图坐标系统

● iOS中视图的坐标系统是以左上为原点的。这点和MAC OS中坐标系统不同（左下）。

●同时要注意的是如果我们直接使用框架(eg:Quartz 2D) 进行绘图，那么原点默认为左下。

● 要注意，视图的坐标系统中的每个点并不指定是像素，一个点可能包括多个像素。
可以使用下面视图的属性来得到每点像素数量。





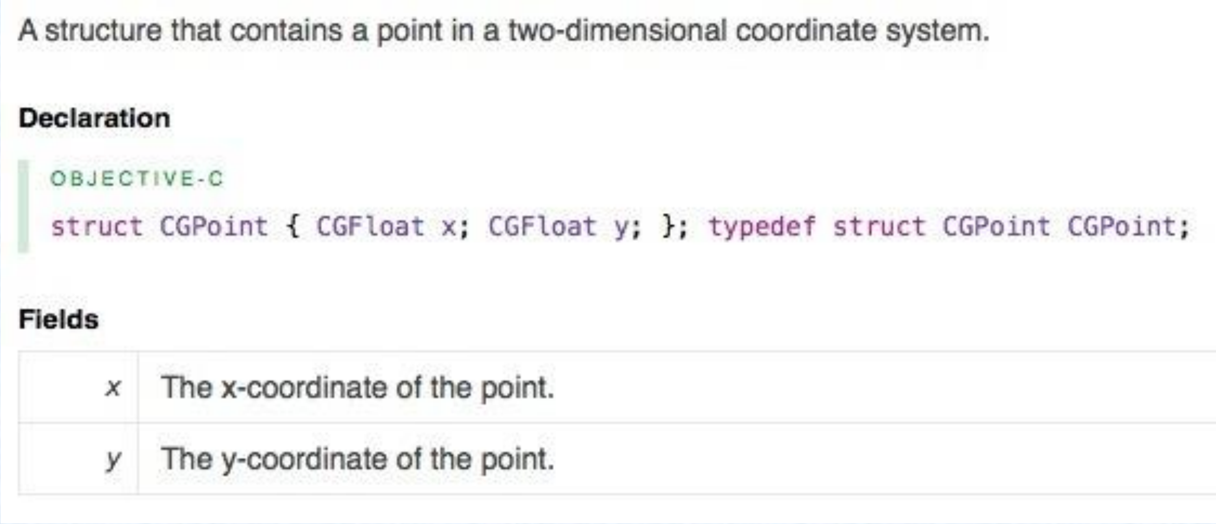
```
@property(nonatomic) CGFloat contentScaleFactor
```

3、关于视图坐标的数据类型

●CGFloat

为了兼容32位和64位系统

●CGPoint



A structure that contains a point in a two-dimensional coordinate system.

Declaration

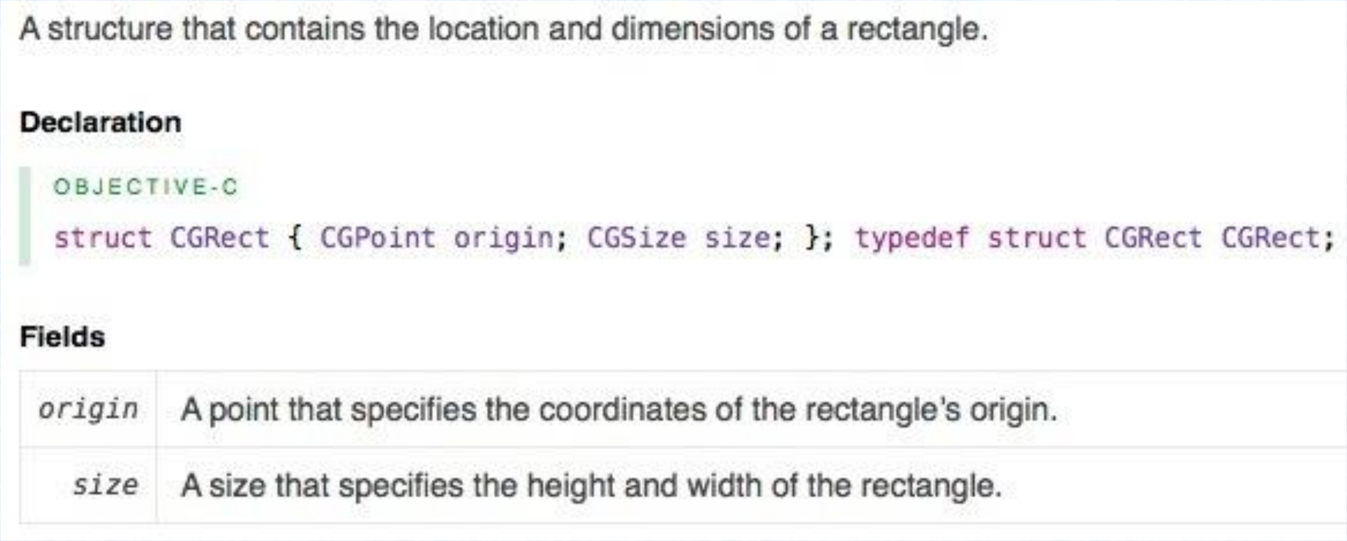
OBJECTIVE-C

```
struct CGPoint { CGFloat x; CGFloat y; }; typedef struct CGPoint CGPoint;
```

Fields

x	The x-coordinate of the point.
y	The y-coordinate of the point.

- CGRect



A structure that contains the location and dimensions of a rectangle.

Declaration

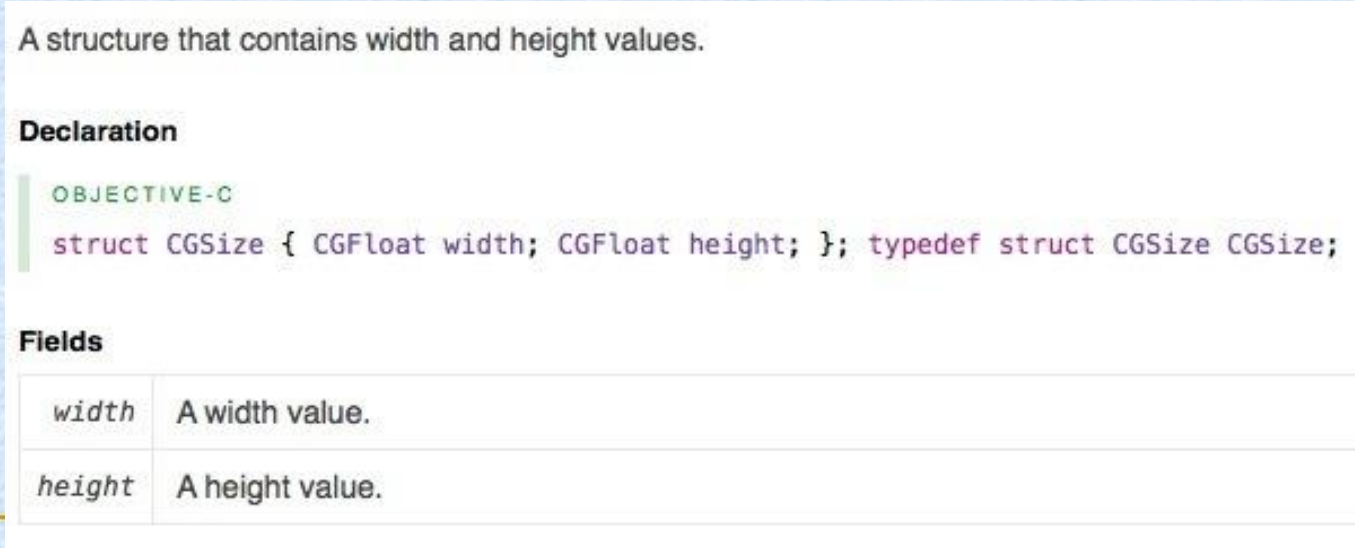
OBJECTIVE-C

```
struct CGRect { CGPoint origin; CGSize size; }; typedef struct CGRect CGRect;
```

Fields

origin	A point that specifies the coordinates of the rectangle's origin.
size	A size that specifies the height and width of the rectangle.

- CGSize



A structure that contains width and height values.

Declaration

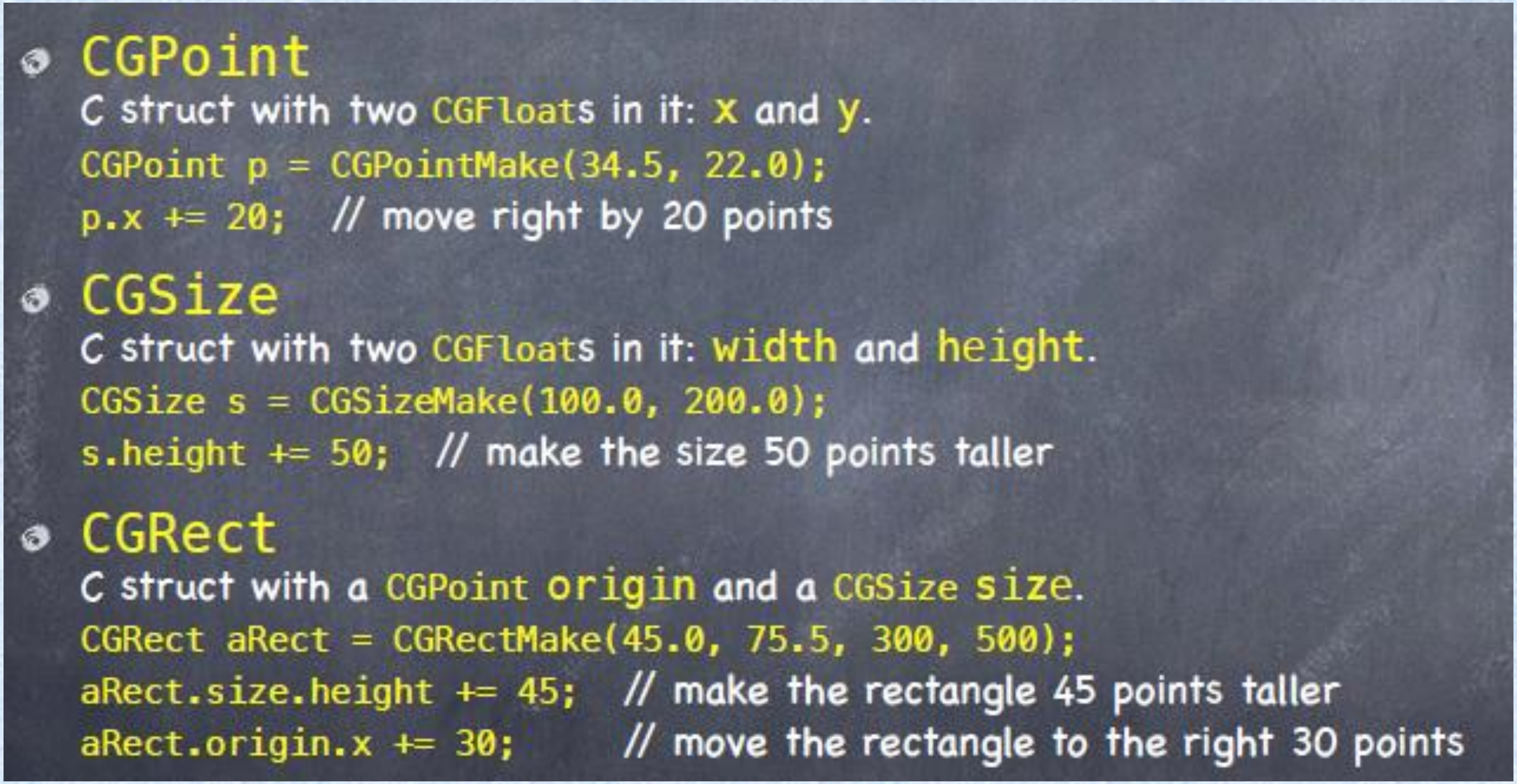
OBJECTIVE-C

```
struct CGSize { CGFloat width; CGFloat height; }; typedef struct CGSize CGSize;
```

Fields

width	A width value.
height	A height value.

几个例子：



- CGPoint

  C struct with two CGFloats in it: x and y.

  ```
  CGPoint p = CGPointMake(34.5, 22.0);
  p.x += 20;   // move right by 20 points
  ```

- CGSize

  C struct with two CGFloats in it: width and height.

  ```
  CGSize s = CGSizeMake(100.0, 200.0);
  s.height += 50;   // make the size 50 points taller
  ```

- CGRect

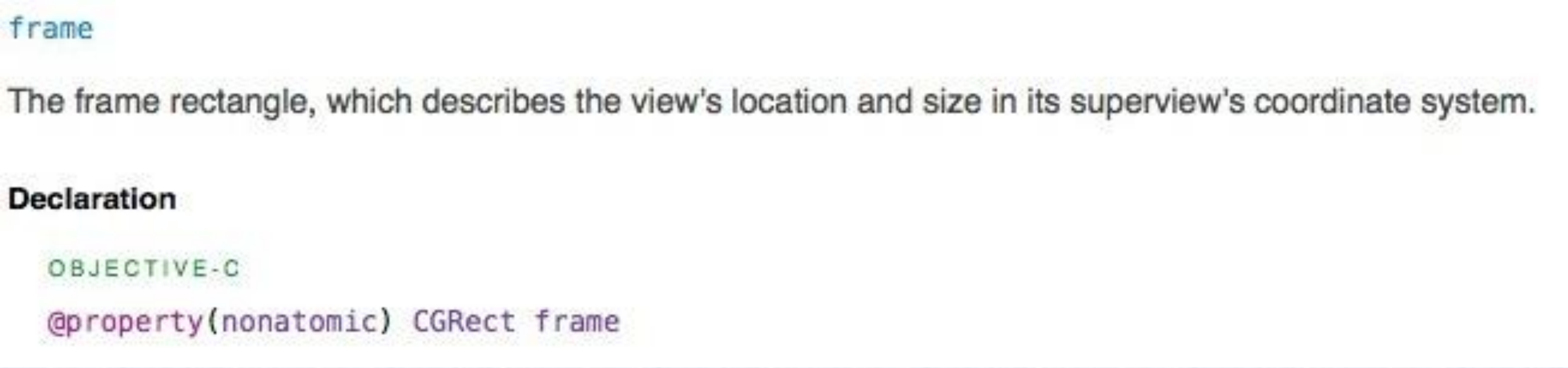
  C struct with a CGPoint origin and a CGSize size.

  ```
  CGRect aRect = CGRectMake(45.0, 75.5, 300, 500);
  aRect.size.height += 45;   // make the rectangle 45 points taller
  aRect.origin.x += 30;      // move the rectangle to the right 30 points
  ```

4、边框、边界和中心

视图UIView对象通过frame、bounds、和center属性声明来跟踪自己的大小和位置。

●边框frame



frame

The frame rectangle, which describes the view's location and size in its superview's coordinate system.

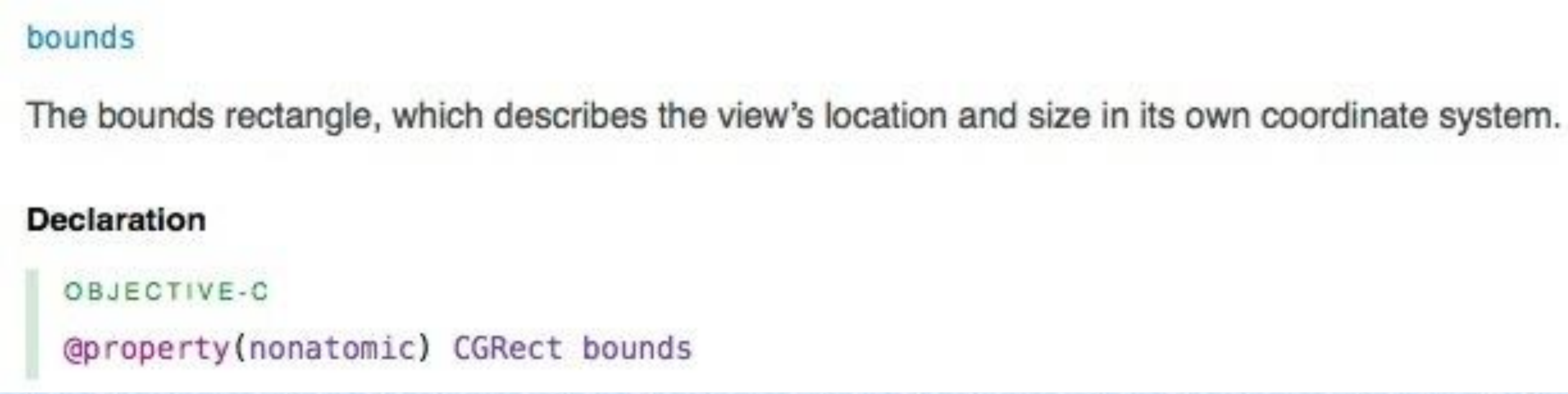
Declaration

OBJECTIVE-C

```
@property(nonatomic) CGRect frame
```

- 边界bounds

bounds属性也包含一个矩形，边即边界矩形，负责定义视图相对于本地坐标系统的位置和大小。边界矩形的原点通常被设置为 (0, 0)。也可以改动原点位置。



bounds

The bounds rectangle, which describes the view’s location and size in its own coordinate system.

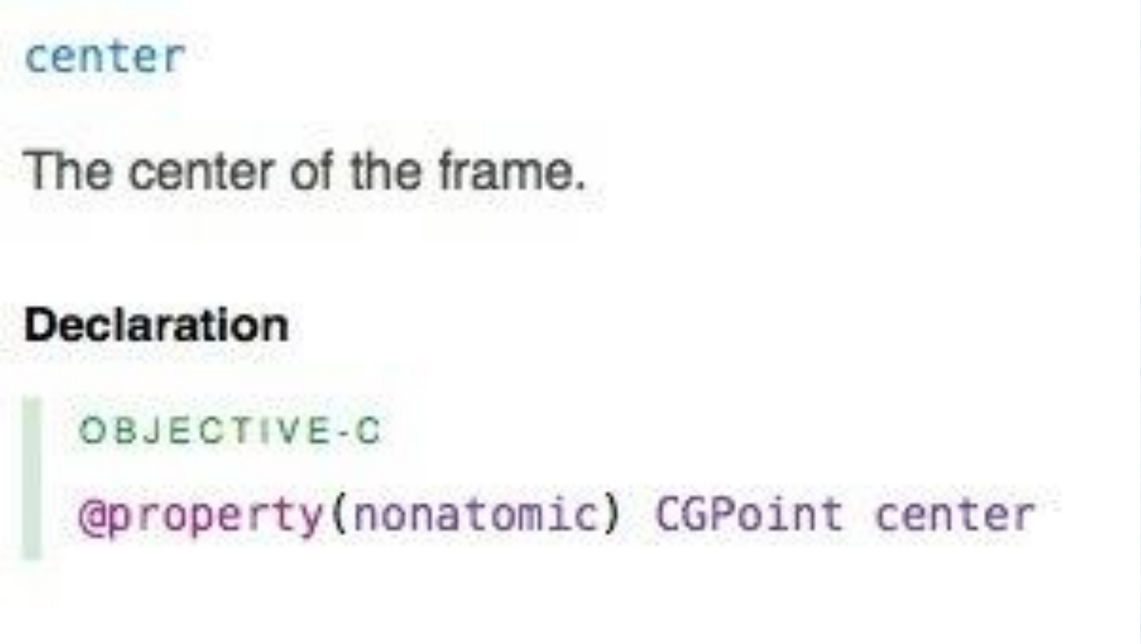
Declaration

OBJECTIVE-C

```
@property(nonatomic) CGRect bounds
```

- 中心center

center属性包含边框矩形的中心点，用CGPoint表示矩形中心点在其父视图中的位置。中心点代表视图的中心。改变中心点一直是移动视图位置的最好方法。



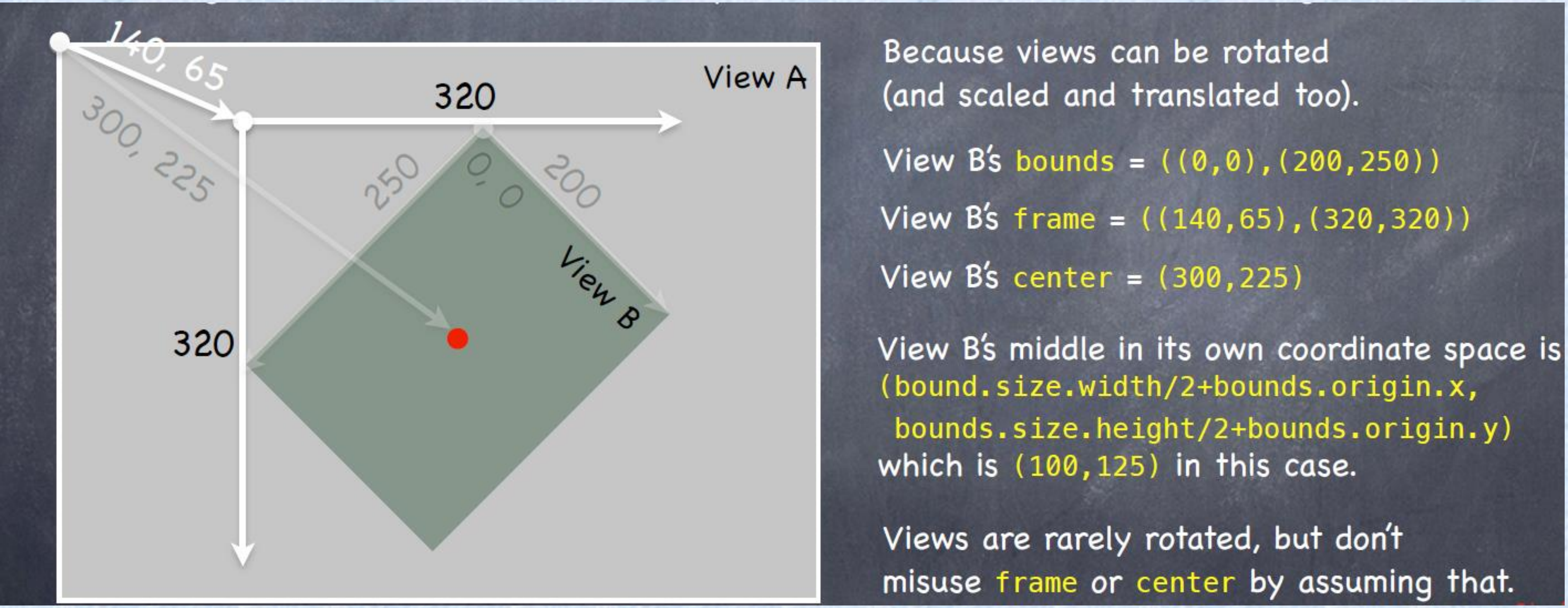
center

The center of the frame.

Declaration

OBJECTIVE-C

```
@property(nonatomic) CGPoint center
```



Because views can be rotated (and scaled and translated too).

View B's bounds = ((0,0),(200,250))

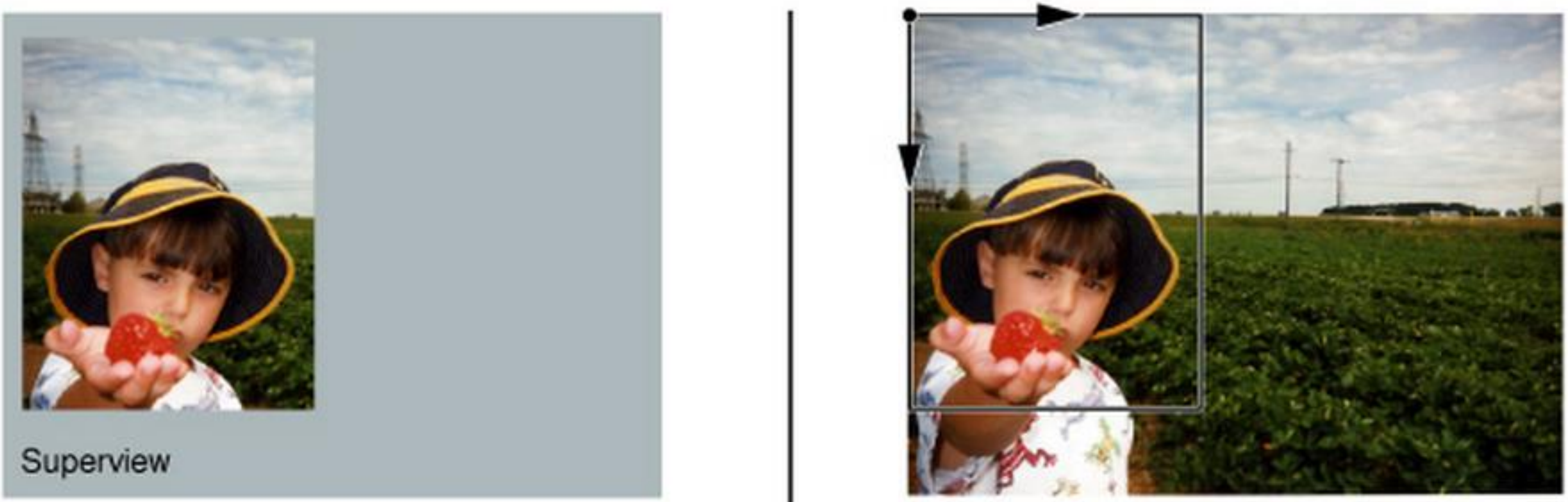
View B's frame = ((140,65),(320,320))

View B's center = (300,225)

View B's middle in its own coordinate space is (bound.size.width/2+bounds.origin.x, bounds.size.height/2+bounds.origin.y) which is (100,125) in this case.

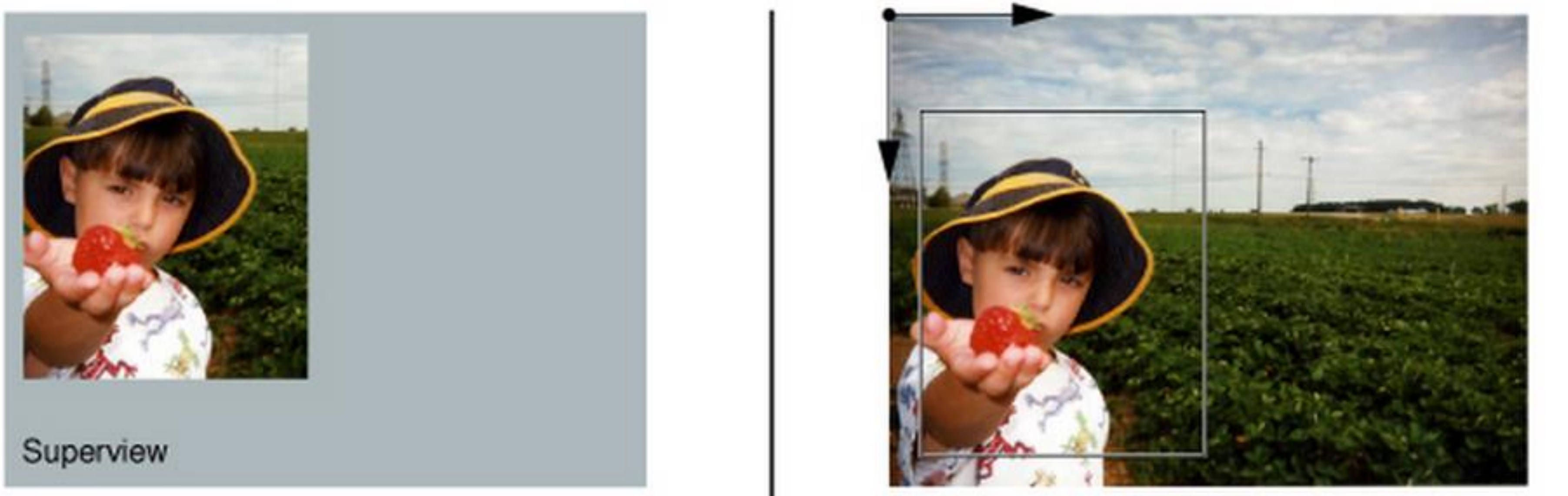
Views are rarely rotated, but don't misuse frame or center by assuming that.

frame、bounds和center三个属性是相互关联、相互影响的，其中一个属性发生变化，其他属性也会跟着变化。



Frame rectangle at (5.0, 5.0), size (73.0, 98.0)

Bounds rectangle at (0.0, 0.0), size (73.0, 98.0)



Frame rectangle at (5.0, 5.0), size (73.0, 88.0)

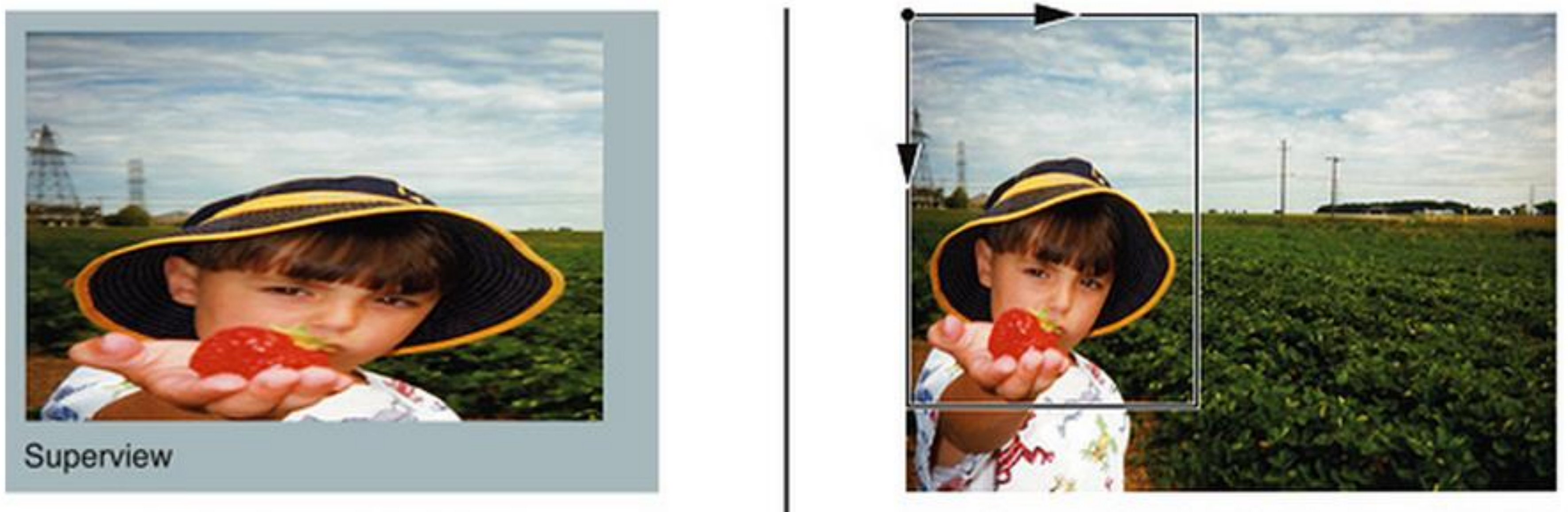
Bounds rectangle at (8.0, 24.0), size (73.0, 88.0)

注意：缺省情况下，视图的边框并不会被父视图的边框裁剪。如果您希望让一个视图裁剪其子视图，需要将其clipsToBounds属性设置为YES。

5、内容模式和比例缩放

改变视图的边界或者将一个比例因子应用到视图的transform属性声明时，边框矩形会发生等量的变化。根据内容模式的不同，视图的内容也可能被缩放或重新定位，以反映上述的变化。

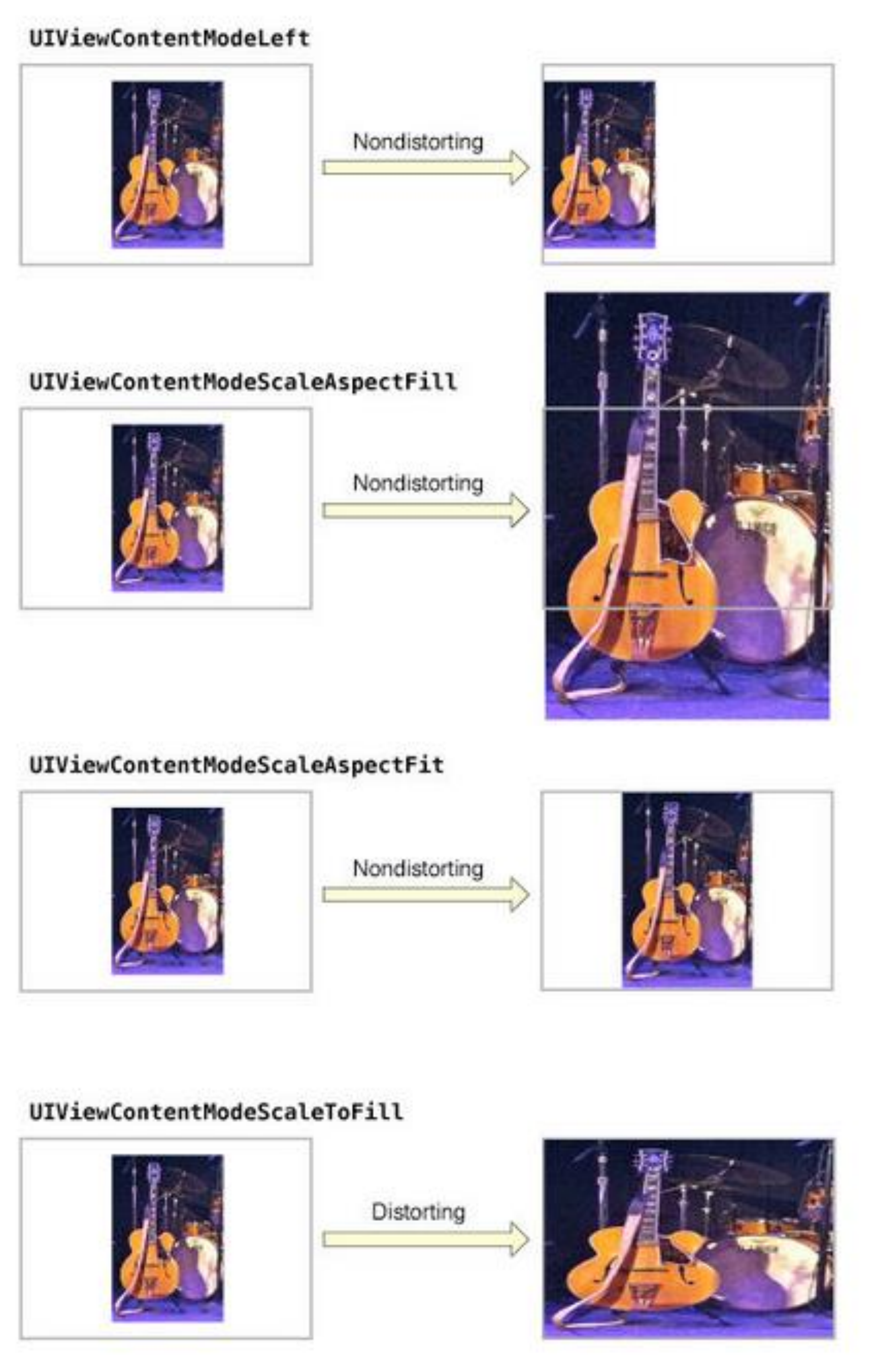
视图的contentMode属性决定了边界变化和缩放操作作用到视图上产生的效果。缺省情况下，这个属性的值被设置为UIViewContentModeScaleToFill，意味着视图内容总是被缩放，以适应新的边框尺寸。



View with transform set

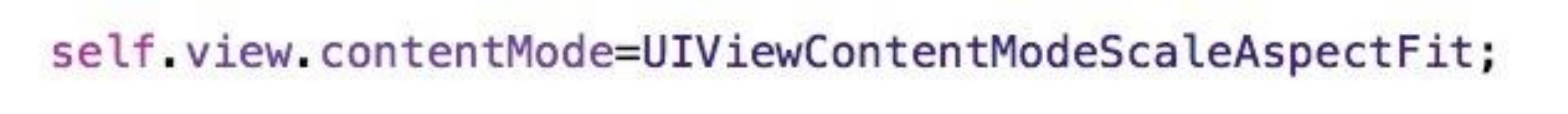
Bounds rectangle at (0.0, 0.0), size (73.0, 98.0)

常用的几种模式


UIViewContentModeLeft
Nondistorting
UIViewContentModeScaleAspectFill
Nondistorting
UIViewContentModeScaleAspectFit
Nondistorting
UIViewContentModeScaleToFill
Distorting

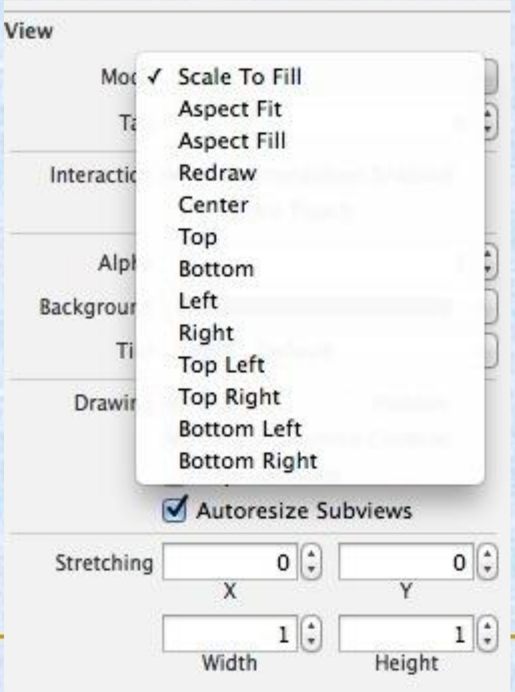
如何设置contentMode属性

方法一：代码方式。



```
self.view.contentMode=UIViewContentModeScaleAspectFit;
```

方法二：故事板(Storyboard),属性检测器。

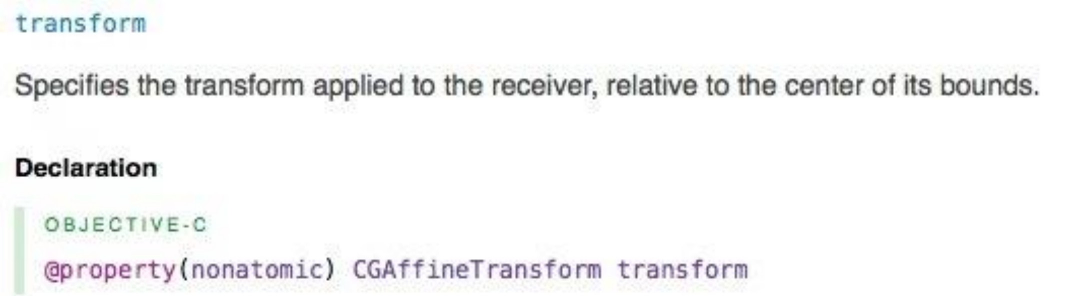


✓ Scale To Fill

☑ Autoresize Subviews

Stretching 0 0
X Y
1 1
Width Height

视图的transform属性



transform

Specifies the transform applied to the receiver, relative to the center of its bounds.

Declaration

OBJECTIVE-C

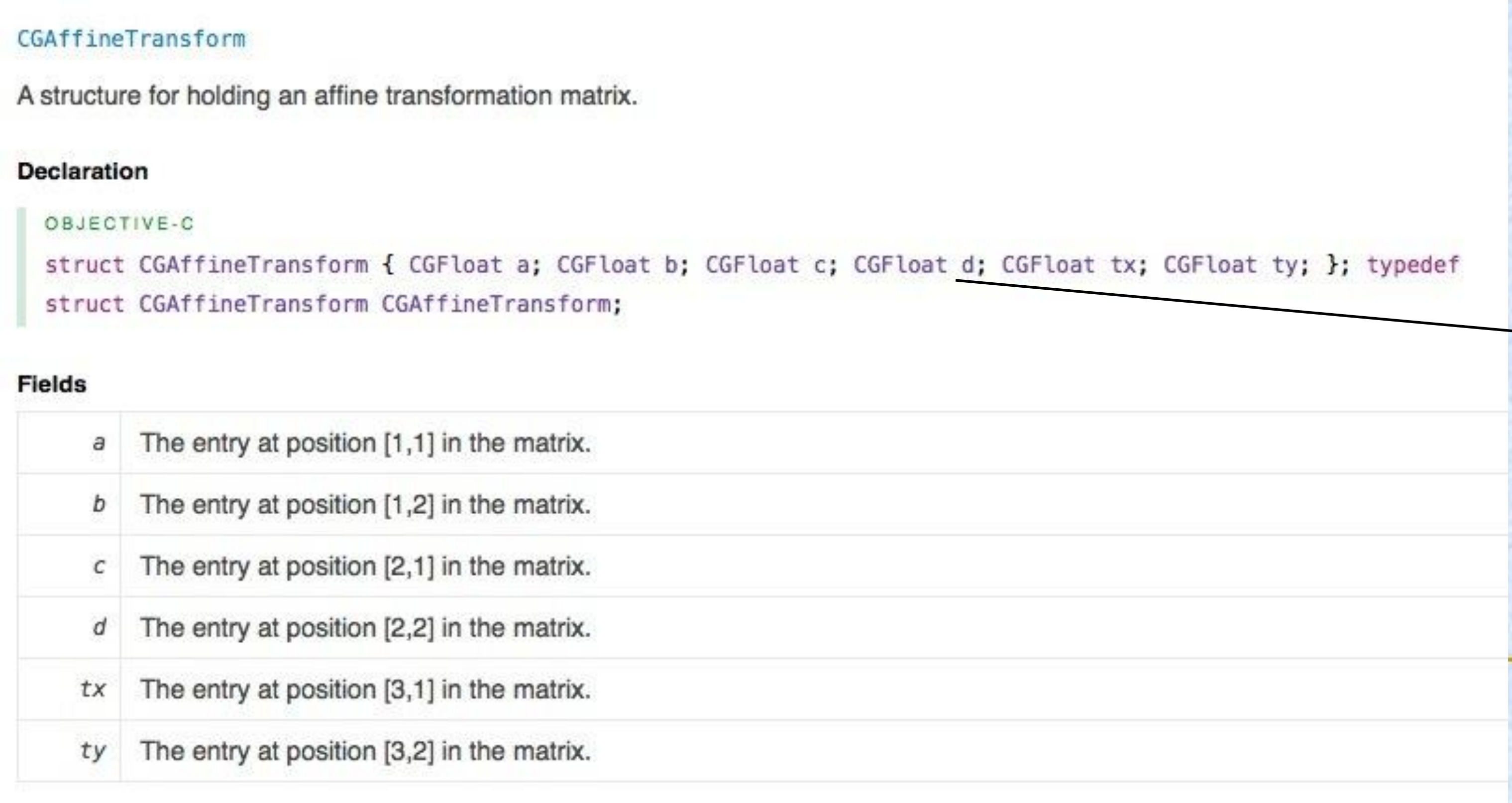
```
@property(nonatomic) CGAffineTransform transform
```

$$[x' \quad y' \quad 1]=[x \quad y \quad 1]\times\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

仿射变换

$$x' = ax + cy + t_x$$
$$y' = bx + dy + t_y$$

仿射矩阵的数据结构



CGAffineTransform

A structure for holding an affine transformation matrix.

Declaration

OBJECTIVE-C

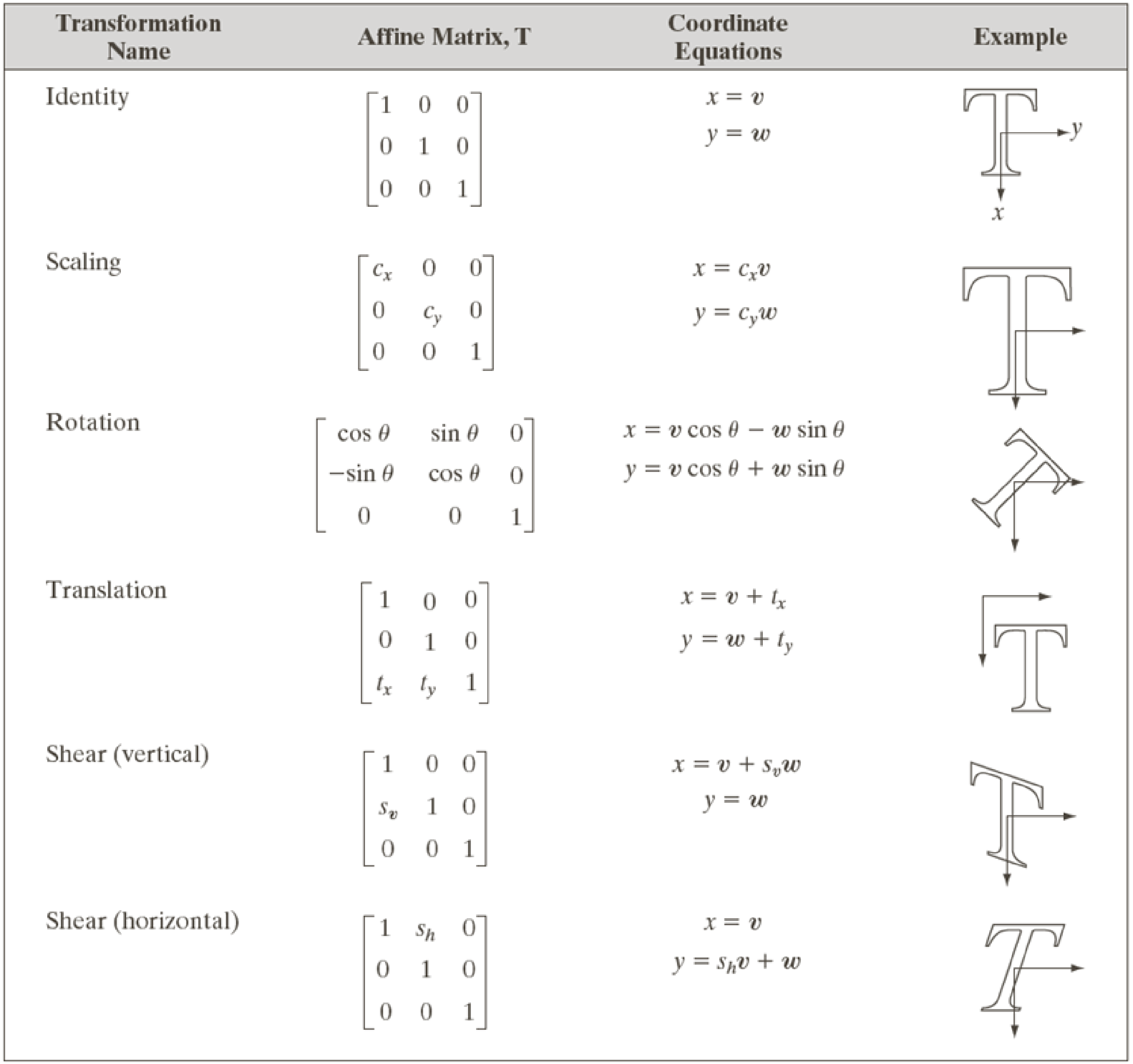
```
struct CGAffineTransform { CGFloat a; CGFloat b; CGFloat c; CGFloat d; CGFloat tx; CGFloat ty; }; typedef
struct CGAffineTransform CGAffineTransform;
```

Fields

a	The entry at position [1,1] in the matrix.
b	The entry at position [1,2] in the matrix.
c	The entry at position [2,1] in the matrix.
d	The entry at position [2,2] in the matrix.
tx	The entry at position [3,1] in the matrix.
ty	The entry at position [3,2] in the matrix.

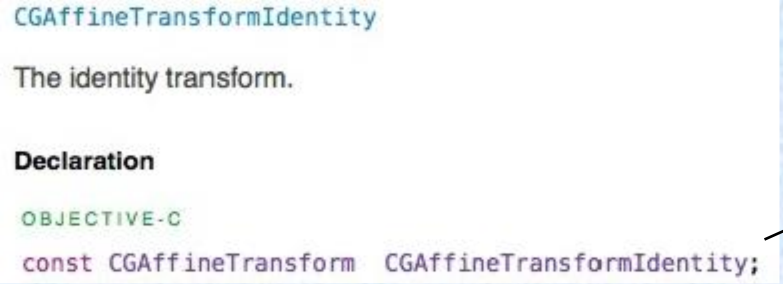
$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

仿射矩阵



Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

恒等变换



CGAffineTransformIdentity

The identity transform.

Declaration

OBJECTIVE-C

```
const CGAffineTransform  CGAffineTransformIdentity;
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

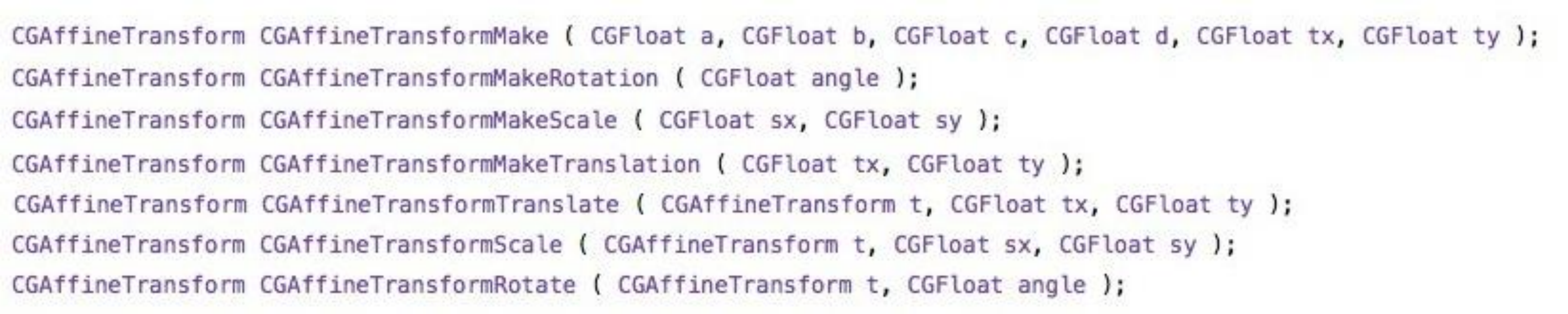
执行恒等变换。如果改变过一个视图的transform属性后又要恢复默认状态进行重置可以使用：

view.transform = CGAffineTransformIdentity

变换方式

1、常用方法是利用Core Graphics框架(quartz 2D)中的变换函数。

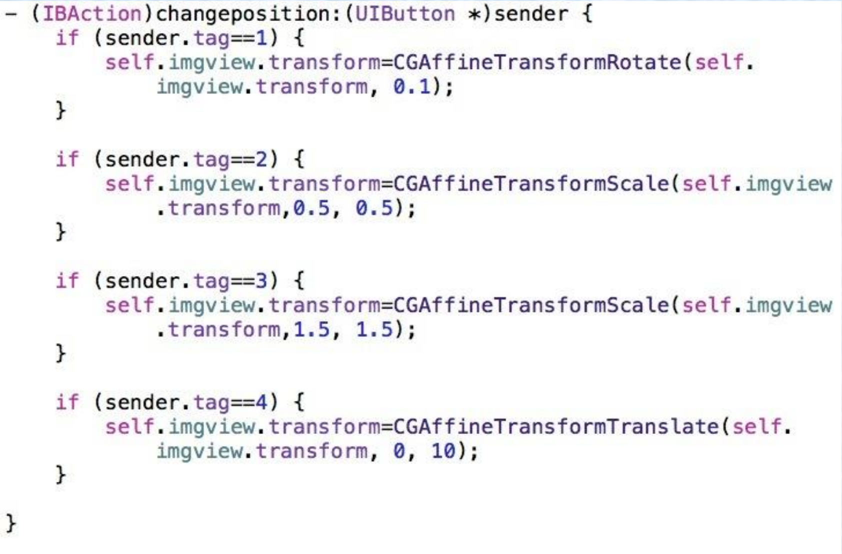
eg:



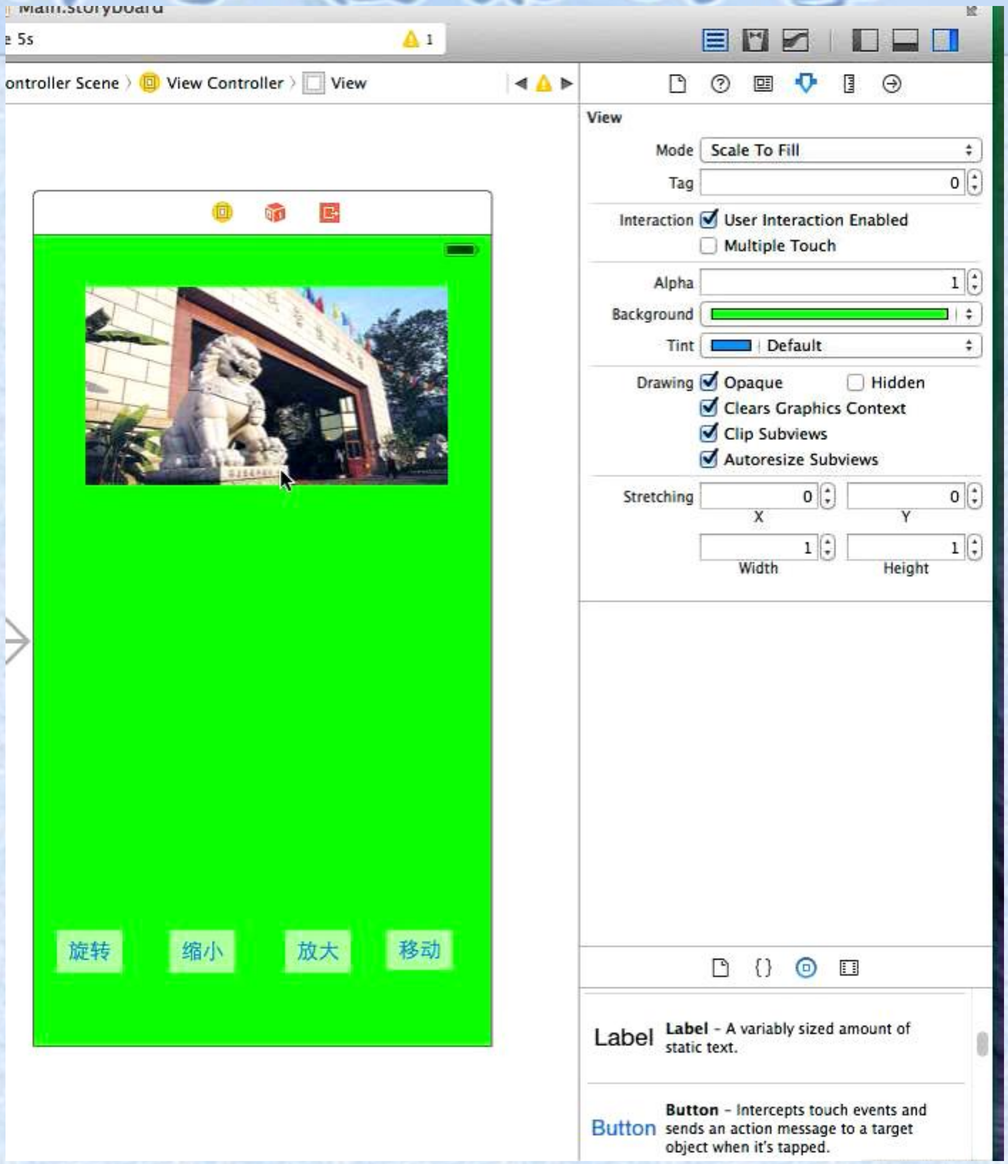
```
CGAffineTransform CGAffineTransformMake ( CGFloat a, CGFloat b, CGFloat c, CGFloat d, CGFloat tx, CGFloat ty );
CGAffineTransform CGAffineTransformMakeRotation ( CGFloat angle );
CGAffineTransform CGAffineTransformMakeScale ( CGFloat sx, CGFloat sy );
CGAffineTransform CGAffineTransformMakeTranslation ( CGFloat tx, CGFloat ty );
CGAffineTransform CGAffineTransformTranslate ( CGAffineTransform t, CGFloat tx, CGFloat ty );
CGAffineTransform CGAffineTransformScale ( CGAffineTransform t, CGFloat sx, CGFloat sy );
CGAffineTransform CGAffineTransformRotate ( CGAffineTransform t, CGFloat angle );
```

2、直接计算并设置变换矩阵。

示例



```
- (IBAction)changeposition:(UIButton *)sender {
    if (sender.tag==1) {
        self.imgview.transform=CGAffineTransformRotate(self.
            imgview.transform, 0.1);
    }

    if (sender.tag==2) {
        self.imgview.transform=CGAffineTransformScale(self.imgview
            .transform,0.5, 0.5);
    }

    if (sender.tag==3) {
        self.imgview.transform=CGAffineTransformScale(self.imgview
            .transform,1.5, 1.5);
    }

    if (sender.tag==4) {
        self.imgview.transform=CGAffineTransformTranslate(self.
            imgview.transform, 0, 10);
    }

}
```



View

Mode: Scale To Fill

Tag: 0

Interaction: ☑ User Interaction Enabled

□ Multiple Touch

Alpha: 1

Background

Tint: Default

Drawing: ☑ Opaque □ Hidden

☑ Clears Graphics Context

☑ Clip Subviews

☑ Autoresize Subviews

Stretching: X 0, Y 0, Width 1, Height 1

Label — Label - A variably sized amount of static text.

Button — Button - Intercepts touch events and sends an action message to a target object when it's tapped.

6、视图间的层级关系

一个视图拥有一个父视图，也可能包含多个子视图。

●获取父视图可通过属性superview

(UIView *) superview

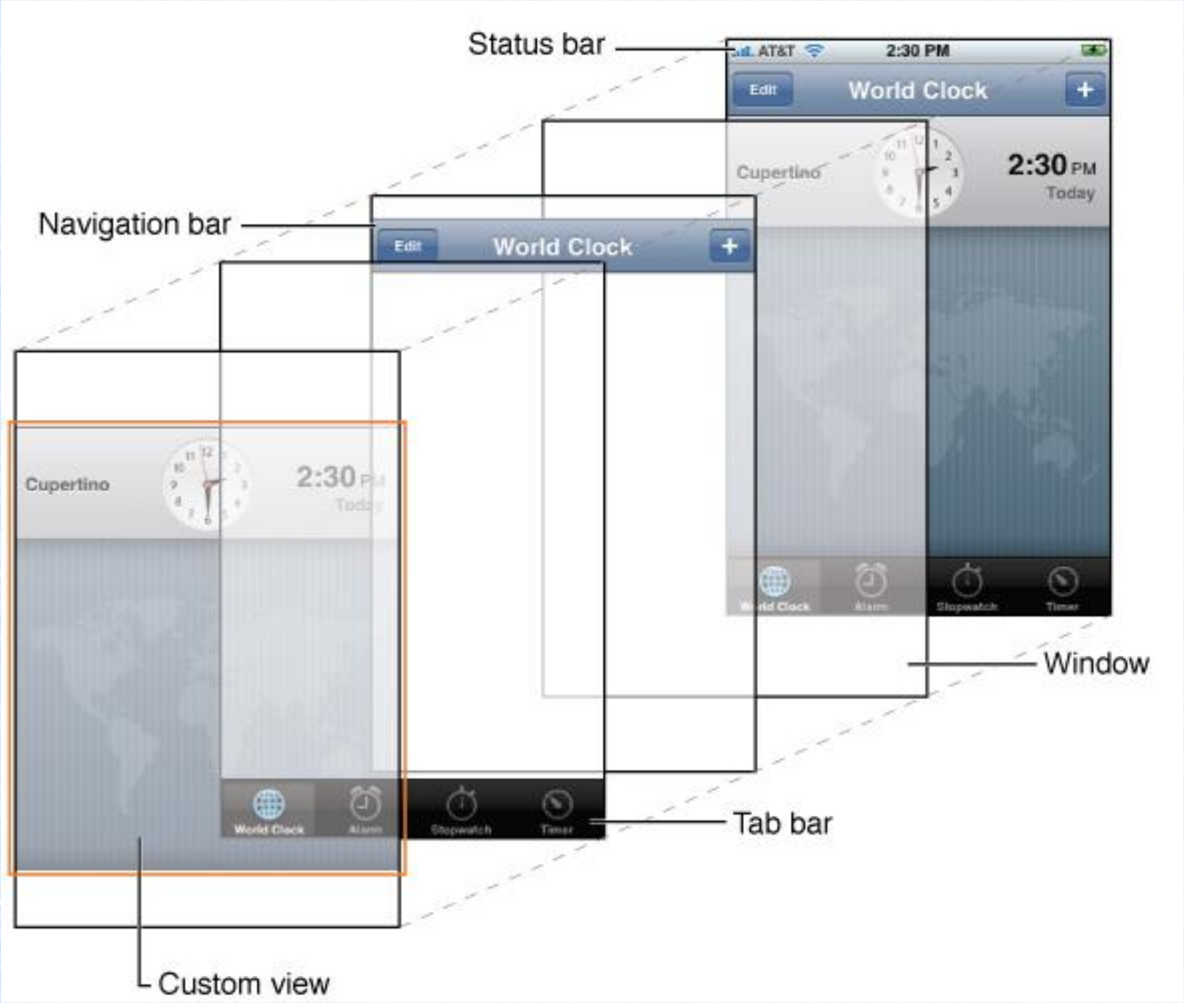
●获取所有子视图可通过属性subviews

(NSArray *) subviews

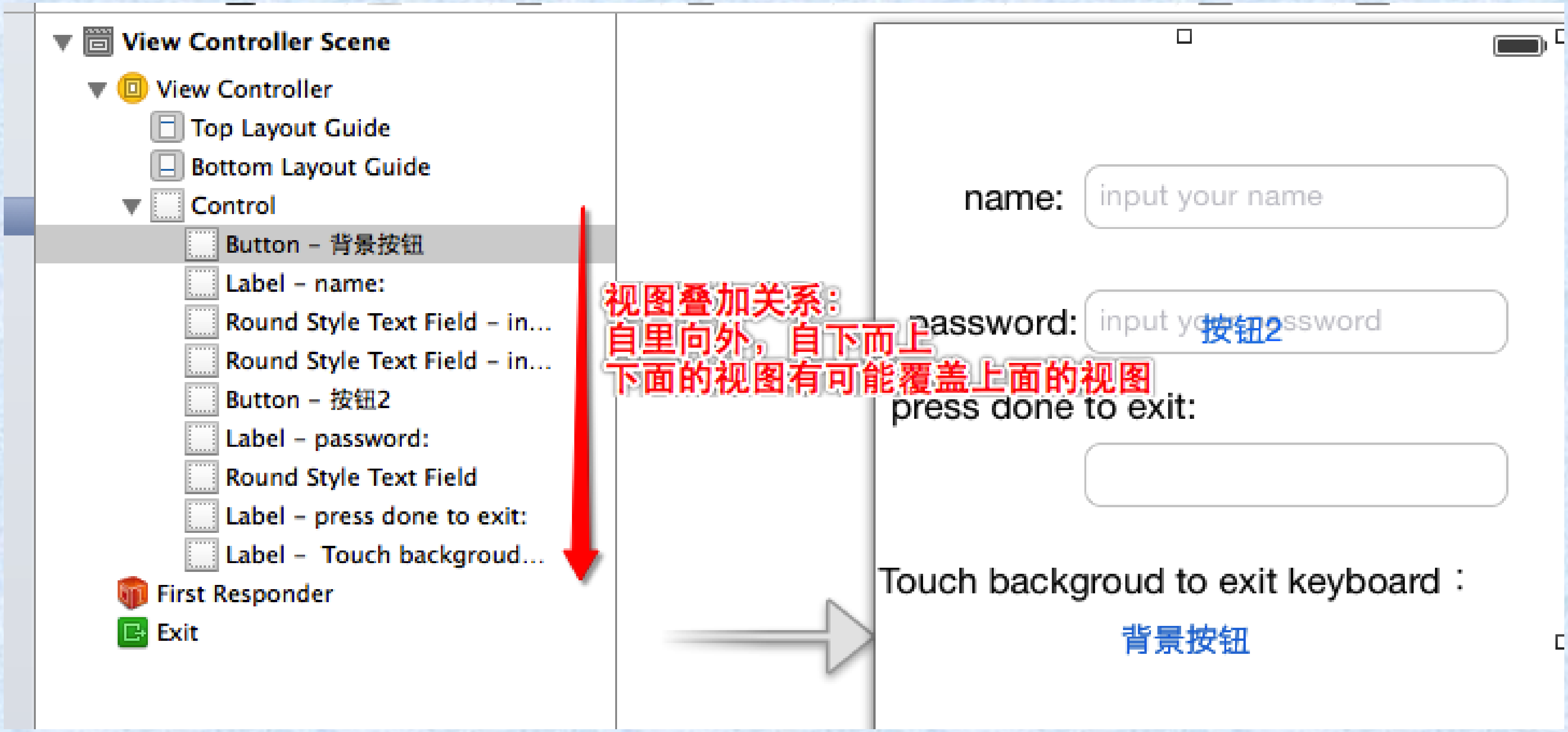
●可通过tag属性标示一个视图对象(整数)

NSInteger tag

在 storyboard中利用IB可以通过拖拽视图来布局，但当视图很多时可能会出现遮挡，此时就需要注意下视图间的层级。所谓层级，就是哪个视图在上，哪个在下，有没有一个视图覆盖其他视图的情况。

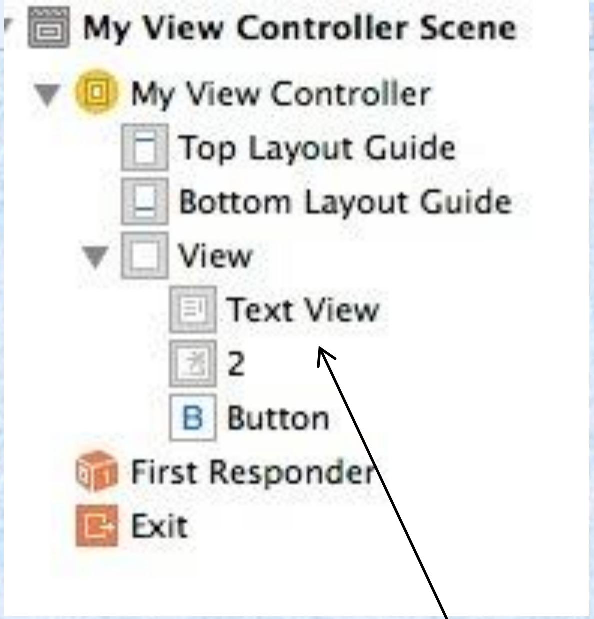


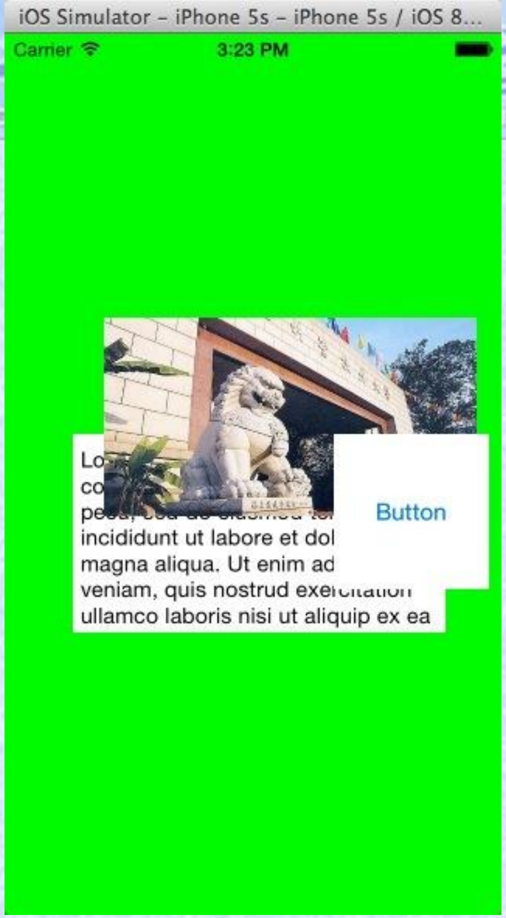
Clock程序的视图层



View Controller Scene
- View Controller
  - Top Layout Guide
  - Bottom Layout Guide
  - Control
    - Button - 背景按钮
    - Label - name:
    - Round Style Text Field - in...
    - Round Style Text Field - in...
    - Button - 按钮2
    - Label - password:
    - Round Style Text Field
    - Label - press done to exit:
    - Label - Touch backgroud...
  - First Responder
  - Exit

视图叠加关系：
自里向外，自下而上
下面的视图有可能覆盖上面的视图





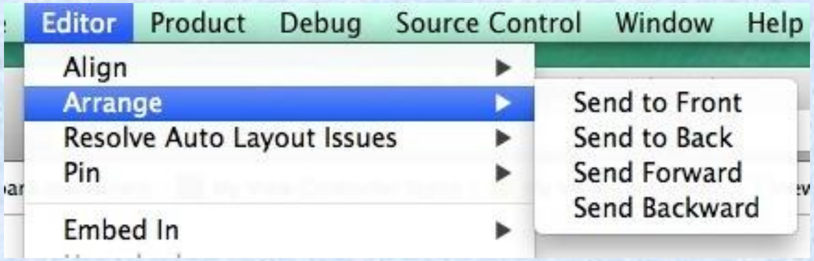
Lo
co
pe
incididunt ut labore et dol
magna aliqua. Ut enim ad
veniam, quis nostrud exe
ullamco laboris nisi ut aliquip ex ea

调整方法

1:直接在大纲中拖动设置先后次序

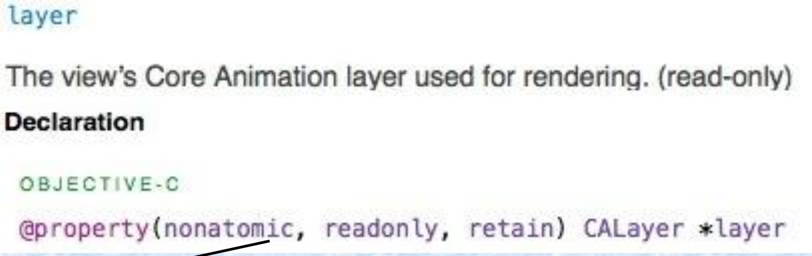
2:在菜单中调整。

3:在代码中调整，调整在subviews中的位置。靠后的在上面。也可以使用bringSubviewToFront等方法



7、视图的层

在UIView中有一个属性layer,它的类型是CALayer.



layer

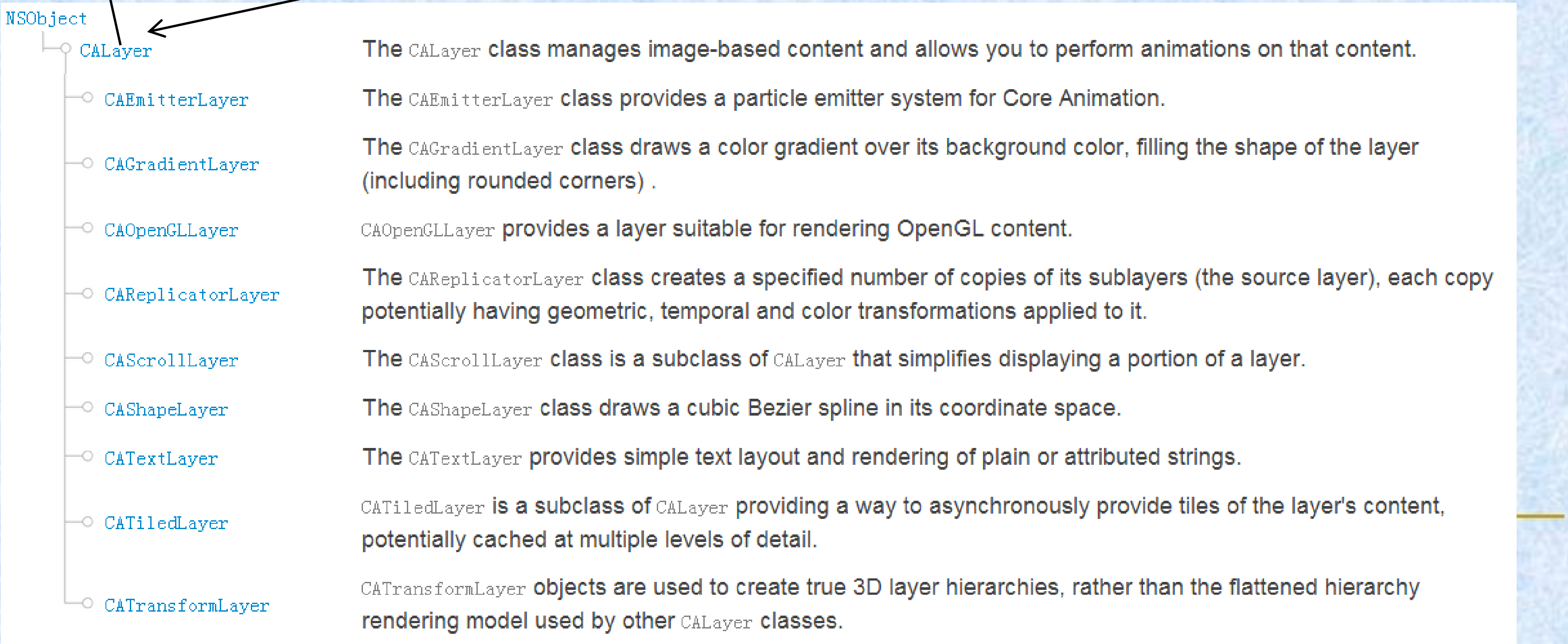
The view's Core Animation layer used for rendering. (read-only)

Declaration

OBJECTIVE-C

```
@property(nonatomic, readonly, retain) CALayer *layer
```

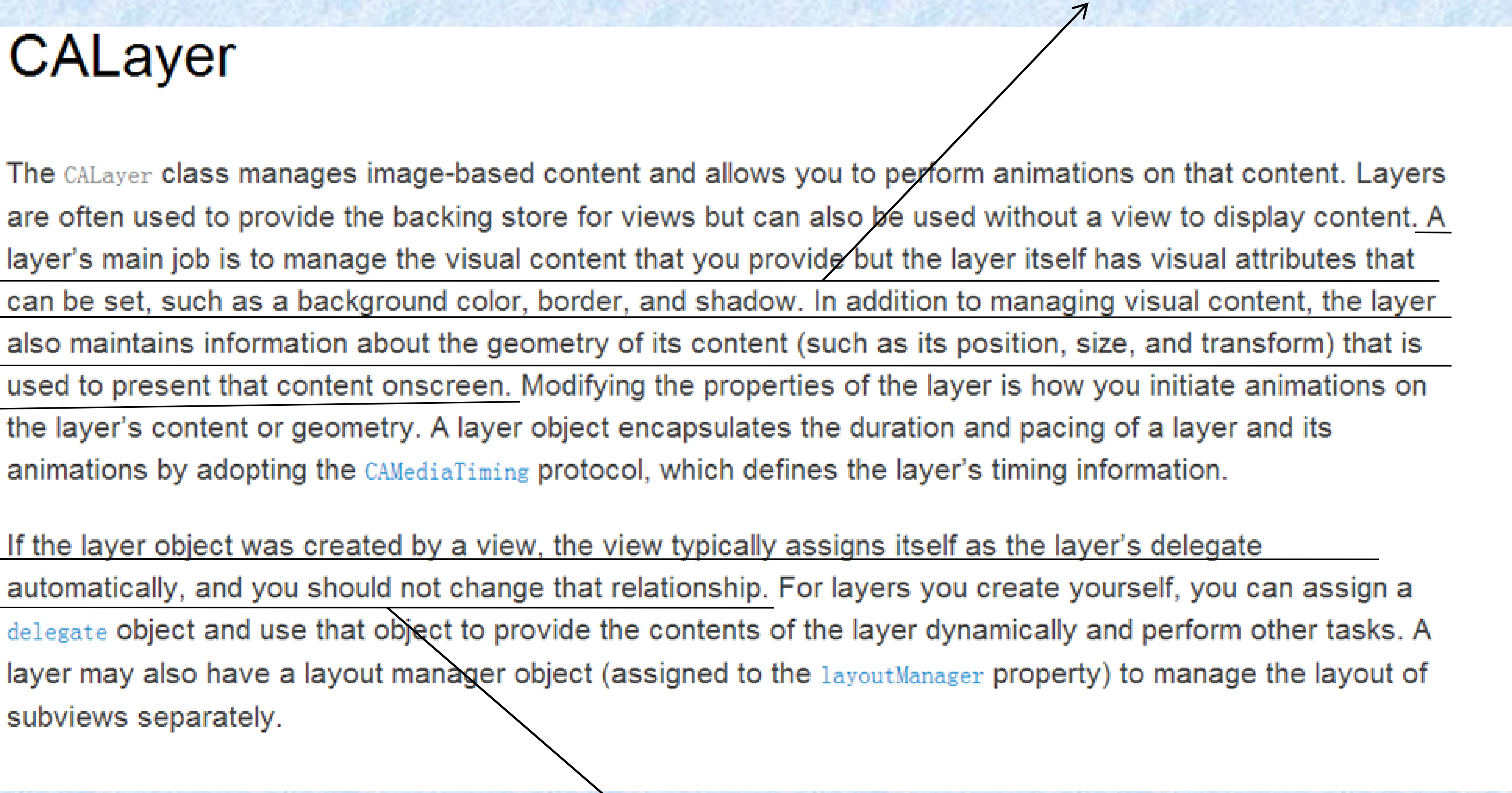
Core Animation
(Quartz Core Framework)



NSObject

CALayer	The CALayer class manages image-based content and allows you to perform animations on that content.
CAEmitterLayer	The CAEmitterLayer class provides a particle emitter system for Core Animation.
CAGradientLayer	The CAGradientLayer class draws a color gradient over its background color, filling the shape of the layer (including rounded corners) .
CAOpenGLLayer	CAOpenGLLayer provides a layer suitable for rendering OpenGL content.
CAReplicatorLayer	The CAReplicatorLayer class creates a specified number of copies of its sublayers (the source layer), each copy potentially having geometric, temporal and color transformations applied to it.
CAScrollLayer	The CAScrollLayer class is a subclass of CALayer that simplifies displaying a portion of a layer.
CAShapeLayer	The CAShapeLayer class draws a cubic Bezier spline in its coordinate space.
CATextLayer	The CATextLayer provides simple text layout and rendering of plain or attributed strings.
CATiledLayer	CATiledLayer is a subclass of CALayer providing a way to asynchronously provide tiles of the layer's content, potentially cached at multiple levels of detail.
CATransformLayer	CATransformLayer objects are used to create true 3D layer hierarchies, rather than the flattened hierarchy rendering model used by other CALayer classes.

真正的绘图部分

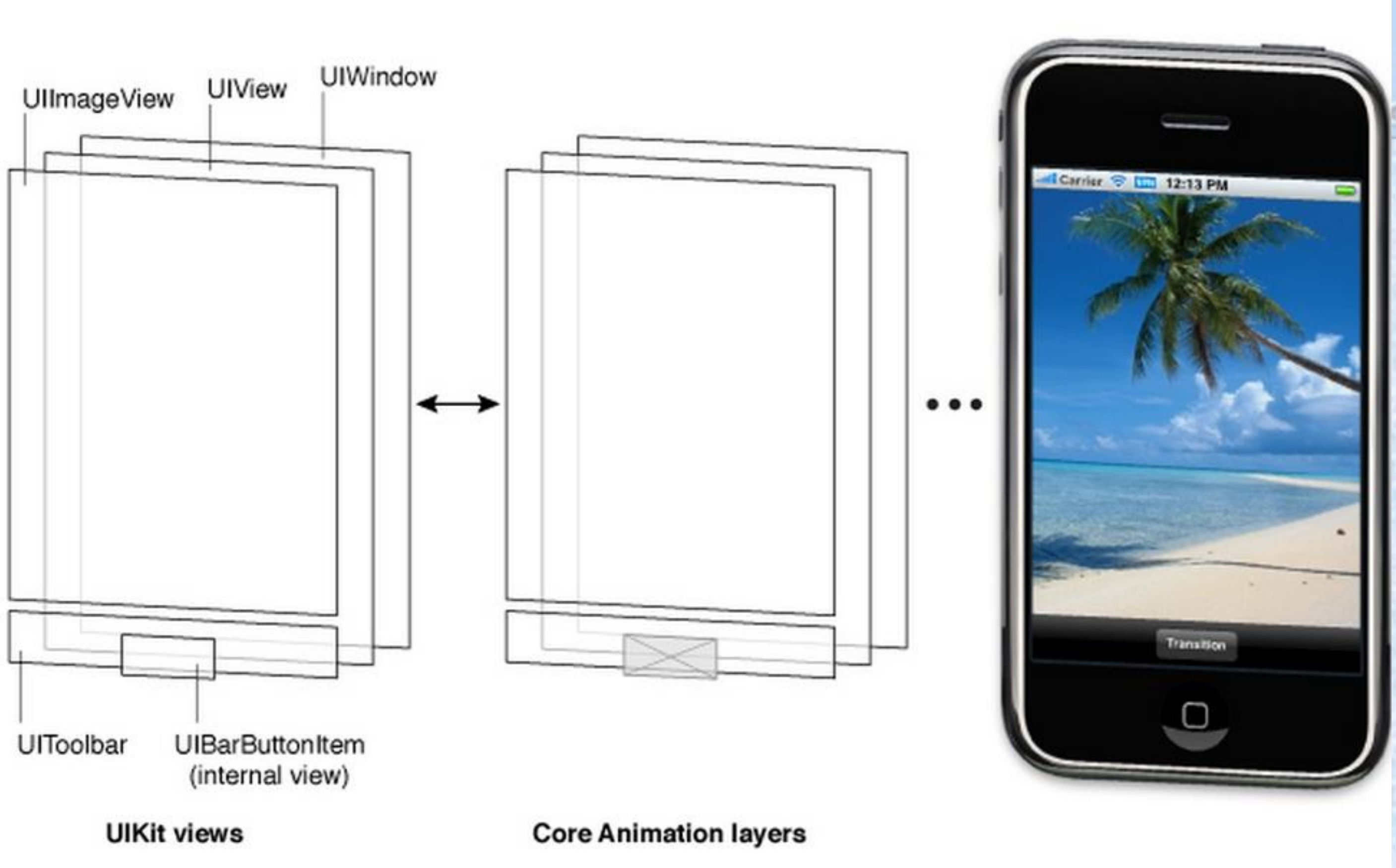


CALayer

The CALayer class manages image-based content and allows you to perform animations on that content. Layers are often used to provide the backing store for views but can also be used without a view to display content. A layer's main job is to manage the visual content that you provide but the layer itself has visual attributes that can be set, such as a background color, border, and shadow. In addition to managing visual content, the layer also maintains information about the geometry of its content (such as its position, size, and transform) that is used to present that content onscreen. Modifying the properties of the layer is how you initiate animations on the layer's content or geometry. A layer object encapsulates the duration and pacing of a layer and its animations by adopting the CAMediaTiming protocol, which defines the layer's timing information.

If the layer object was created by a view, the view typically assigns itself as the layer's delegate automatically, and you should not change that relationship. For layers you create yourself, you can assign a delegate object and use that object to provide the contents of the layer dynamically and perform other tasks. A layer may also have a layout manager object (assigned to the layoutManager property) to manage the layout of subviews separately.

视图都被一个层对象支持



UIView来自CALayer，高于CALayer，是CALayer高层实现与封装。UIView的所有特性来源于CALayer支持。

CALayer为视图内容的布局、渲染和合成动画提供基础支撑

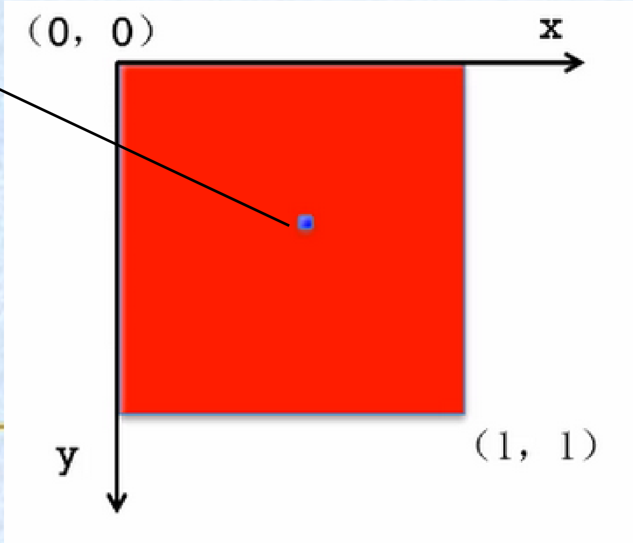
与视图一样，层也有父层，子层概念。也有frame,bounds，但多了position和anchorPoint属性

- CGPoint position

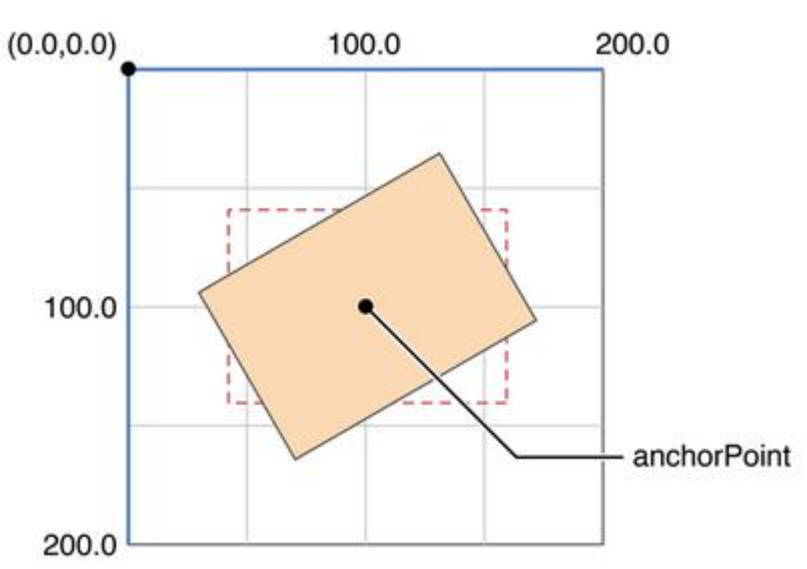
用来设置CALayer在父层中的位置，以父层的左上角为原点(0, 0)

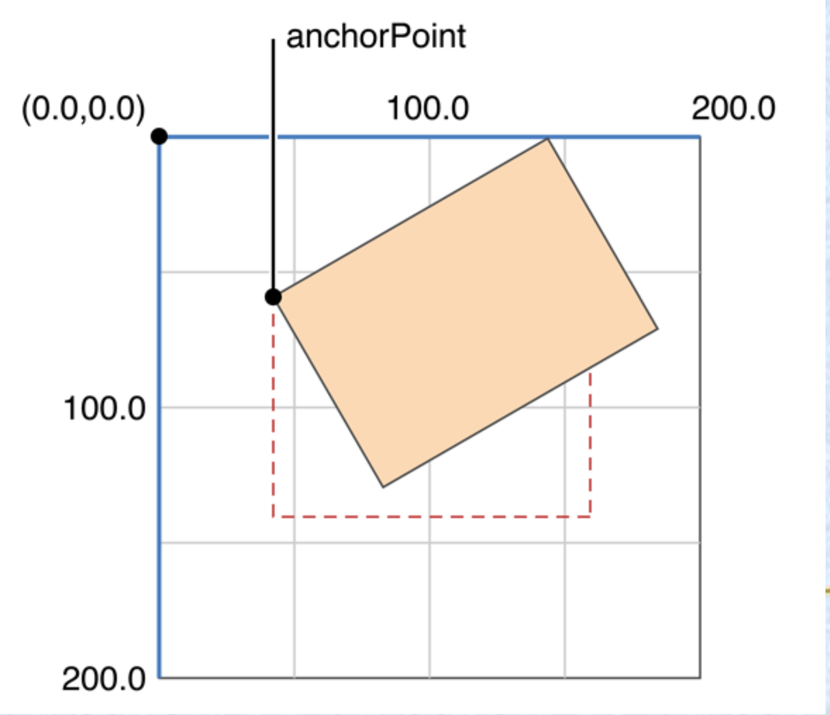
- CGPoint anchorPoint

称为“定位点”、“锚点”
以自己的左上角为原点(0, 0)，它的x、y取值范围都是0~1，默认值为（0.5, 0.5).

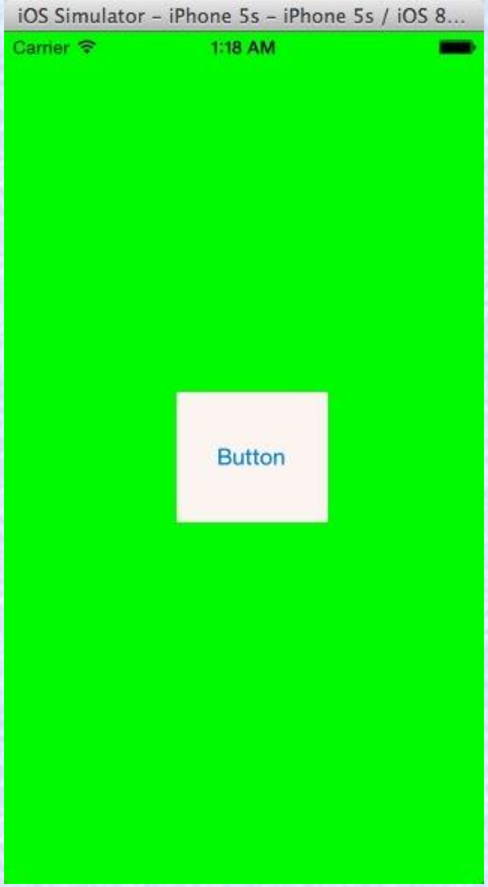


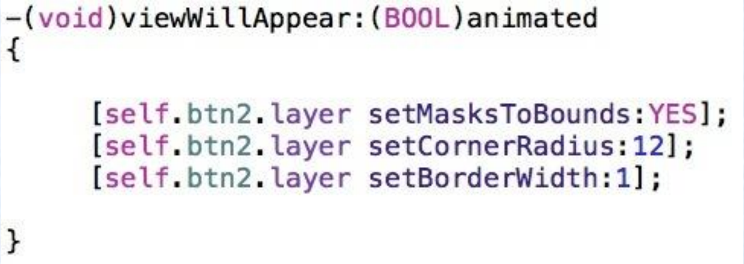
锚点的选择


(0.0,0.0)
100.0
200.0
100.0
200.0
anchorPoint

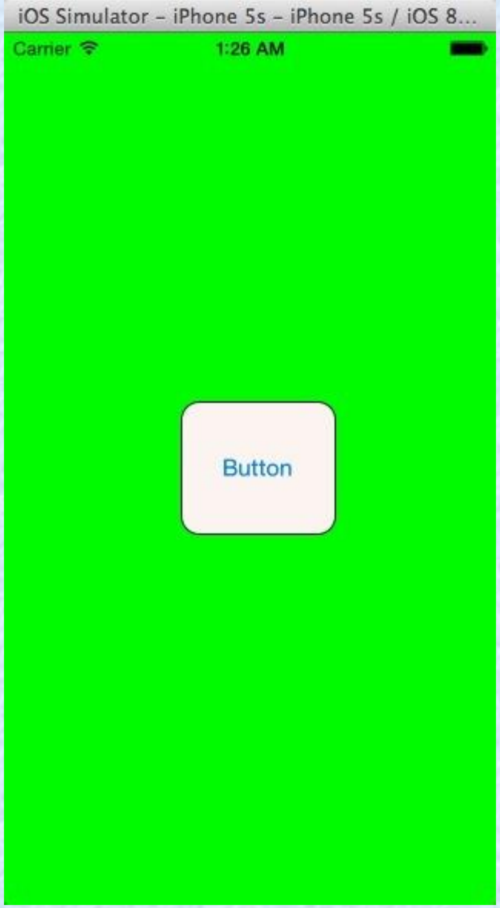

anchorPoint
(0.0,0.0)
100.0
200.0
100.0
200.0

例子



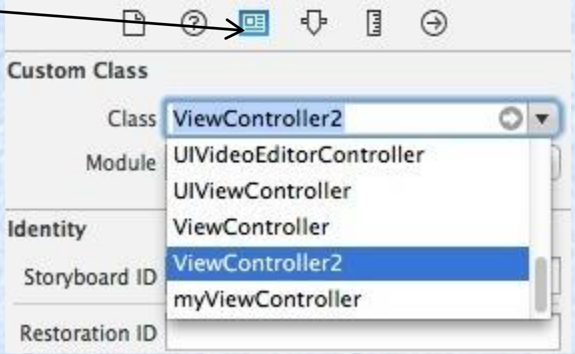


```
-(void)viewWillAppear:(BOOL)animated
{

    [self.btn2.layer setMasksToBounds:YES];
    [self.btn2.layer setCornerRadius:12];
    [self.btn2.layer setBorderWidth:1];

}
```



8、视图的创建、添加和移除

方法一：在故事板Storyboard中利用IB。
(注意要为IB创建的视图设置对应的类)



Custom Class

Class ViewController2

Module

Identity

Storyboard ID

Restoration ID

方法二：代码方式

```
- (void)addSubview:(UIView *)view
```

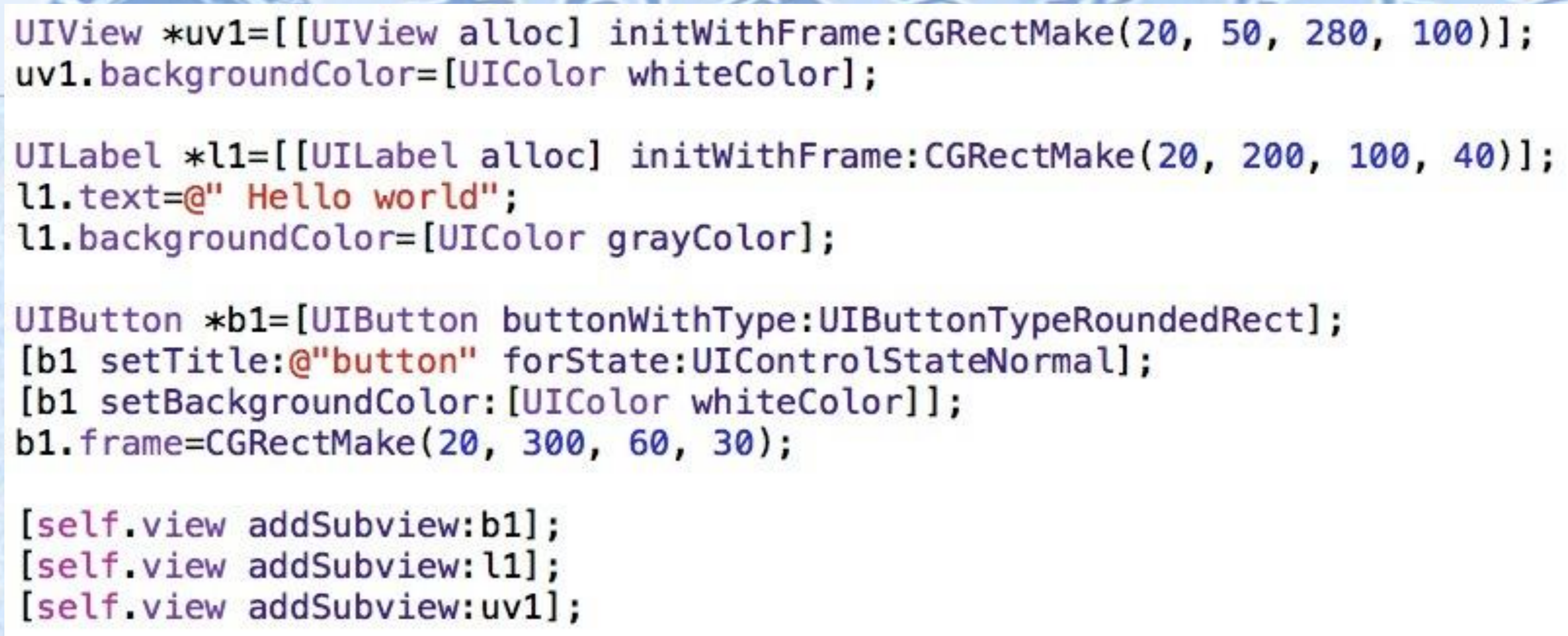
添加一个子视图，并在最上面显示出来。

初始化返回一个新的拥有特定帧矩形的视图对象

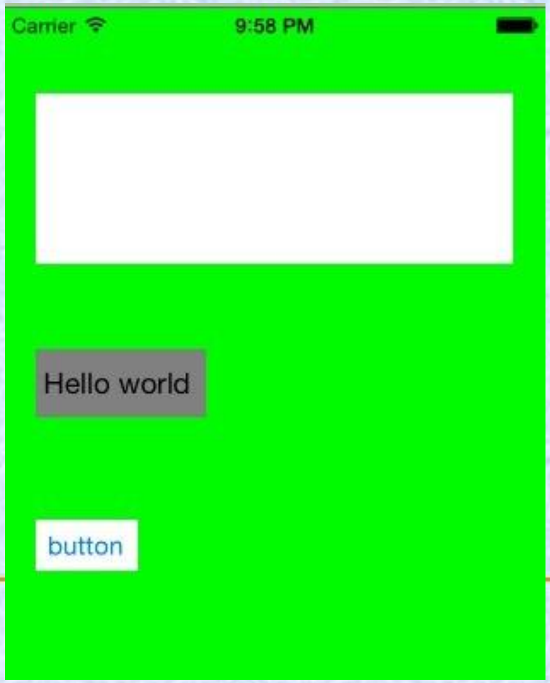
```
- (id)initWithFrame:(CGRect)aRect
```

从父视图或者窗口移除，并在响应链中移除。

```
- (void)removeFromSuperview
```



```
UIView *uv1=[[UIView alloc] initWithFrame:CGRectMake(20, 50, 280, 100)];
uv1.backgroundColor=[UIColor whiteColor];

UILabel *l1=[[UILabel alloc] initWithFrame:CGRectMake(20, 200, 100, 40)];
l1.text=@" Hello world";
l1.backgroundColor=[UIColor grayColor];

UIButton *b1=[UIButton buttonWithType:UIButtonTypeRoundedRect];
[b1 setTitle:@"button" forState:UIControlStateNormal];
[b1 setBackgroundColor:[UIColor whiteColor]];
b1.frame=CGRectMake(20, 300, 60, 30);

[self.view addSubview:b1];
[self.view addSubview:l1];
[self.view addSubview:uv1];
```



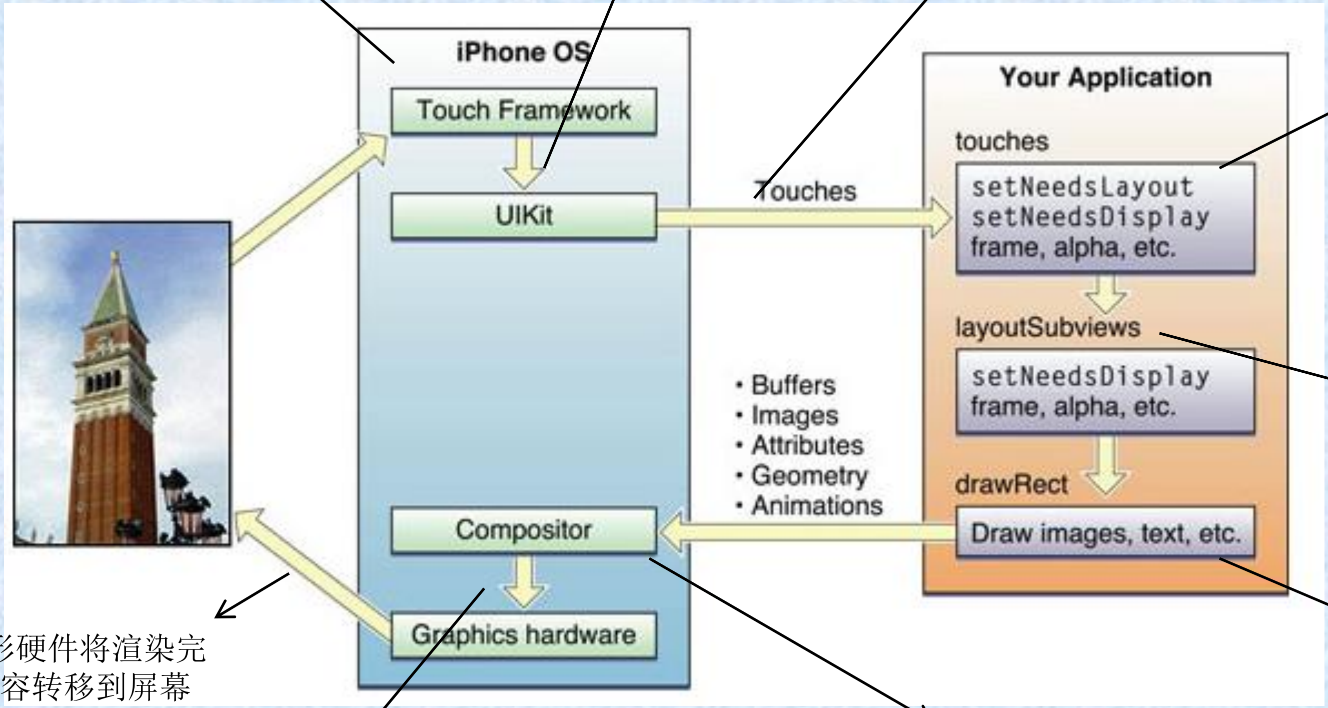
常用初始化代码：

```
- (void)setup { ... }
-(void)awakeFromNib
 {
   [self setup];
 }
- (id)initWithFrame:(CGRect)aRect
{
self = [super initWithFrame:aRect];
[self setup];
return self;
}
```

适用于在故事板上使用Interface Builder创建时

9、视图的交互

当进行交互时或以编程方式进行某些修改时，UIKit内部都会产生一系列复杂的事件。



1 用户触屏

2 硬件报告触摸事件给UIKit框架

3、UIKit框架将触摸事件发给适合的视图

4、视图中的事件处理代码进行响应。

5、如果视图需要重新布局，layoutSubviews方法将被调用。

6、如果任何视图的任何部分被标记为需要重画，UIKit会要求视图重画自身

7、任何已经更新的视图会与应用余下的可视内容组合在一起

8、发送到图形硬件

9、图形硬件将渲染完成的内容转移到屏幕

几点注意：

●可以覆盖drawRect方法，但绝不允许直接发送消息至视图的drawRect方法。

●如果需要重绘视图可以直接发送消息至视图的setNeedsDisplay或setNeedsDisplayInRect方法来间接触发drawRect方法重绘视图。

●setNeedsDisplay和setNeedsDisplayInRect方法异步的。

●layoutSubviews方法可以覆盖也可以直接发消息。当需要重新布局视图元素时候会使用到该方法

layoutSubviews在以下情况下会被调用：

- 用initWithFrame 进行初始化且rect的值不为CGRectZero时会触发
- addSubview会触发layoutSubviews
- 设置view的Frame(frame的值发生变化)会触发layoutSubviews
- 滚动一个UIScrollView会触发layoutSubviews
- 旋转Screen会触发父UIView上的layoutSubviews事件
- 改变一个UIView大小的时候也会触发父UIView上的layoutSubviews事件
- 直接发消息至视图的setLayoutSubviews。

注意：一般不直接发消息至视图的setLayoutSubviews。如果需要重写布局视图元素可使用下面方法。

-(void)layoutIfNeeded

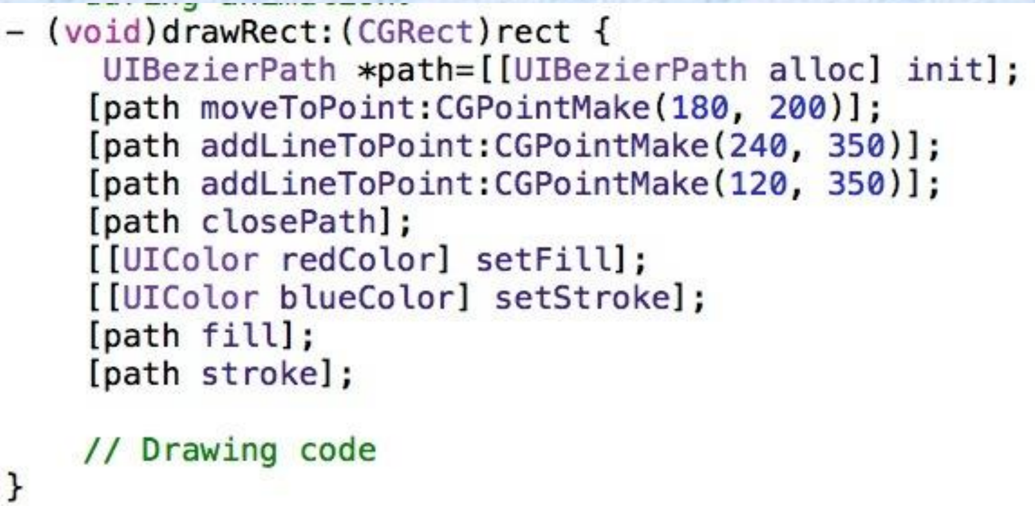
标记为需要重新布局，异步调用layoutIfNeeded刷新布局，不立即刷新，但layoutSubviews一定会被调用

-(void)setNeedsLayout

如果，有需要刷新的标记，立即调用layoutSubviews进行布局（如果没有标记，不会调用layoutSubviews）

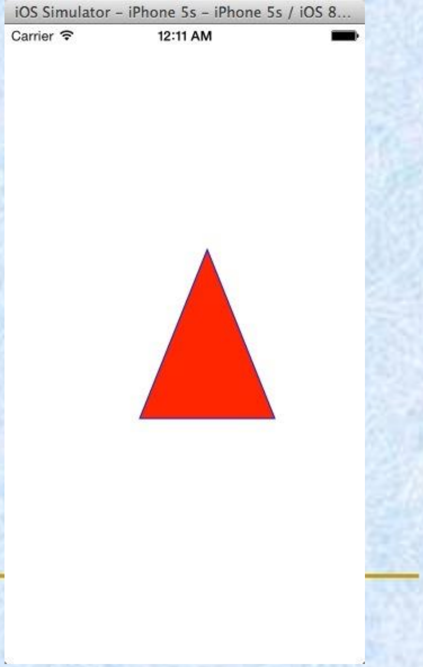
说明：如果要立即刷新，需先调用[view setNeedsLayout]，把标记设为需要布局，然后马上调用[view layoutIfNeeded]，实现布局

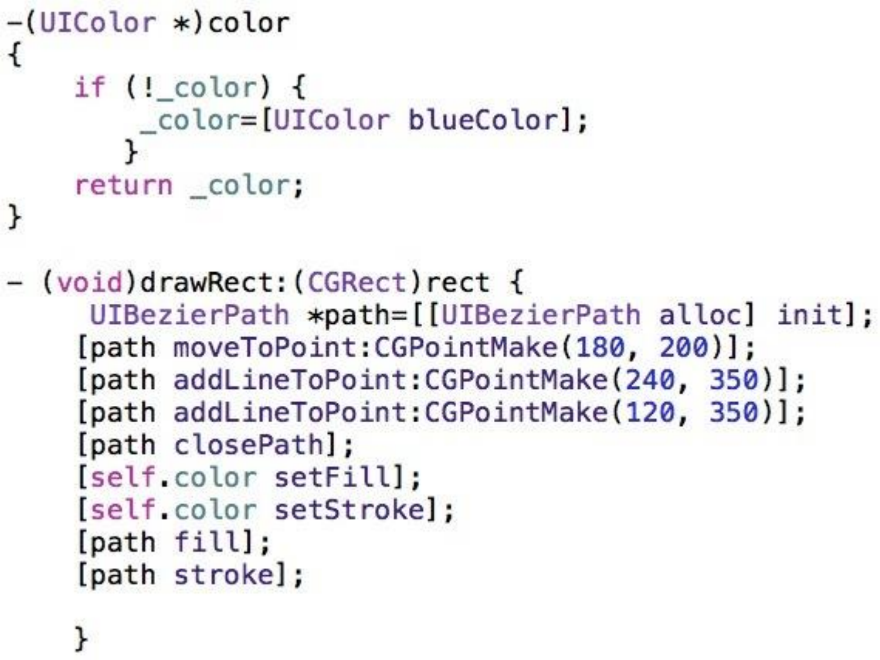
未覆盖drawRect方法前



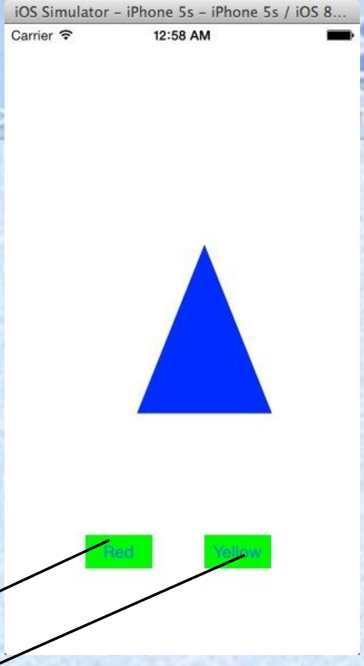
```
- (void)drawRect:(CGRect)rect {
     UIBezierPath *path=[[UIBezierPath alloc] init];
    [path moveToPoint:CGPointMake(180, 200)];
    [path addLineToPoint:CGPointMake(240, 350)];
    [path addLineToPoint:CGPointMake(120, 350)];
    [path closePath];
    [[UIColor redColor] setFill];
    [[UIColor blueColor] setStroke];
    [path fill];
    [path stroke];

    // Drawing code
}
```

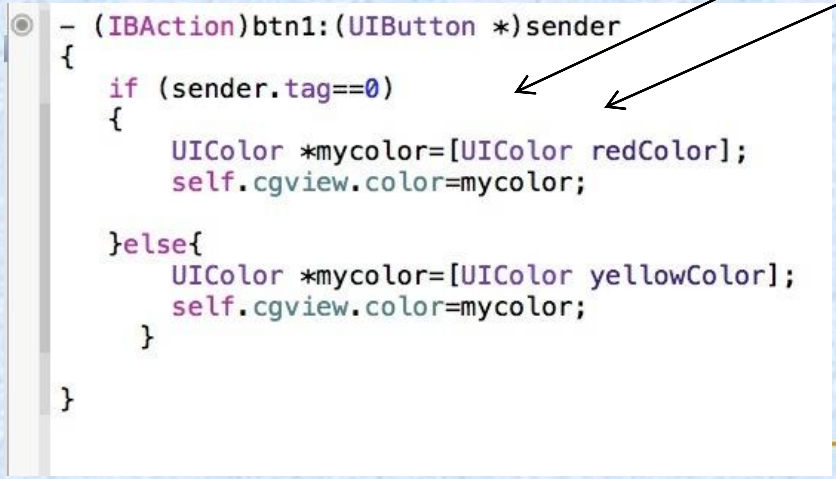
iOS Simulator – iPhone 5s – iPhone 5s / iOS 8...
Carrier
12:14 AM





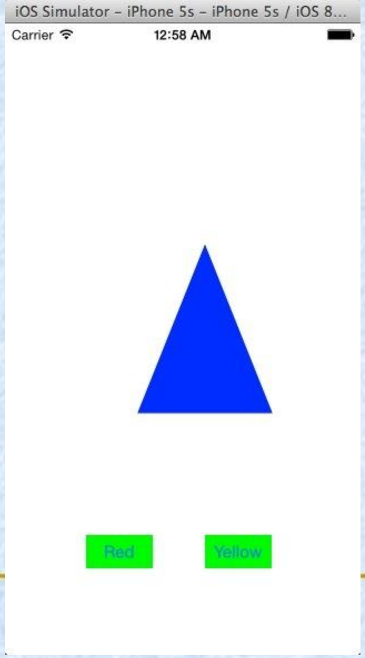
```
-(UIColor *)color
{
    if (!_color) {
        _color=[UIColor blueColor];
       }
    return _color;
}

- (void)drawRect:(CGRect)rect {
     UIBezierPath *path=[[UIBezierPath alloc] init];
    [path moveToPoint:CGPointMake(180, 200)];
    [path addLineToPoint:CGPointMake(240, 350)];
    [path addLineToPoint:CGPointMake(120, 350)];
    [path closePath];
    [self.color setFill];
    [self.color setStroke];
    [path fill];
    [path stroke];

    }
```

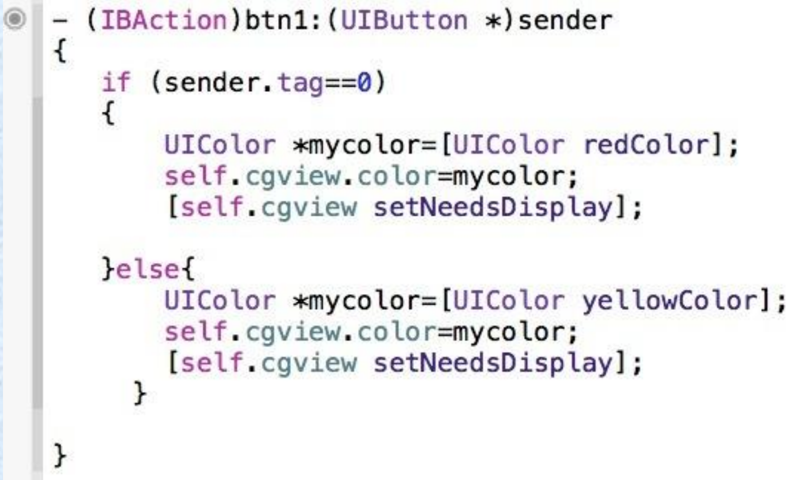




```
- (IBAction)btn1:(UIButton *)sender
{
   if (sender.tag==0)
   {
       UIColor *mycolor=[UIColor redColor];
       self.cgview.color=mycolor;

   }else{
       UIColor *mycolor=[UIColor yellowColor];
       self.cgview.color=mycolor;
      }

}
```

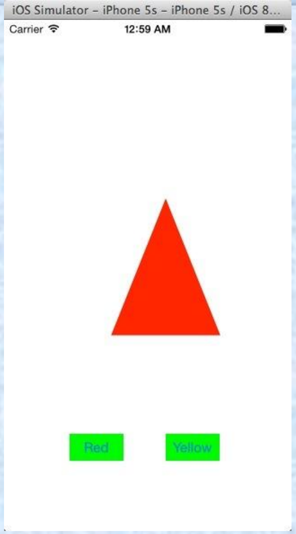
无变化

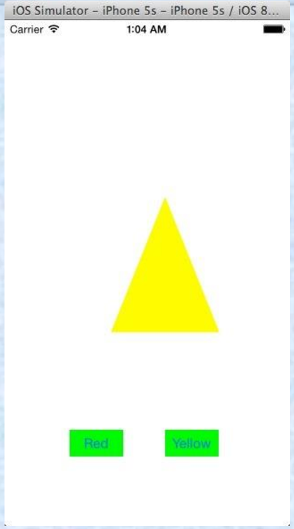


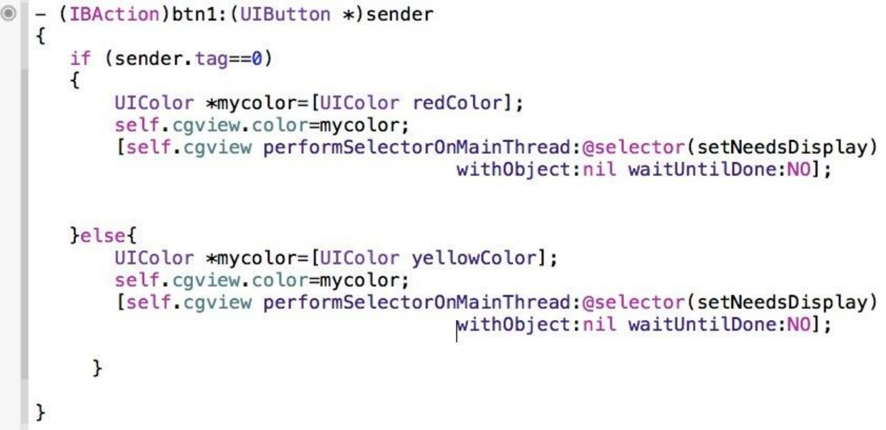


```
- (IBAction)btn1:(UIButton *)sender
{
    if (sender.tag==0)
    {
        UIColor *mycolor=[UIColor redColor];
        self.cgview.color=mycolor;
        [self.cgview setNeedsDisplay];

    }else{
        UIColor *mycolor=[UIColor yellowColor];
        self.cgview.color=mycolor;
        [self.cgview setNeedsDisplay];
      }

}
```

异步

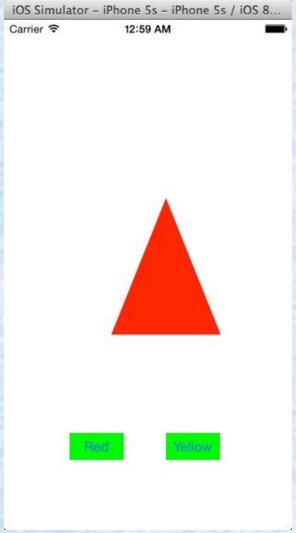


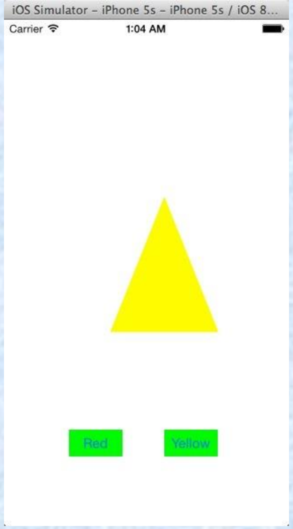




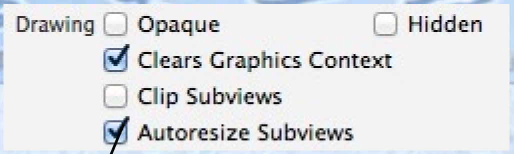
```
- (IBAction)btn1:(UIButton *)sender
{
    if (sender.tag==0)
    {
        UIColor *mycolor=[UIColor redColor];
        self.cgview.color=mycolor;
        [self.cgview performSelectorOnMainThread:@selector(setNeedsDisplay)
                                      withObject:nil waitUntilDone:NO];


    }else{
        UIColor *mycolor=[UIColor yellowColor];
        self.cgview.color=mycolor;
        [self.cgview performSelectorOnMainThread:@selector(setNeedsDisplay)
                                      withObject:nil waitUntilDone:NO];

      }

}
```

同步





10、视图的自动调整



Drawing □ Opaque □ Hidden

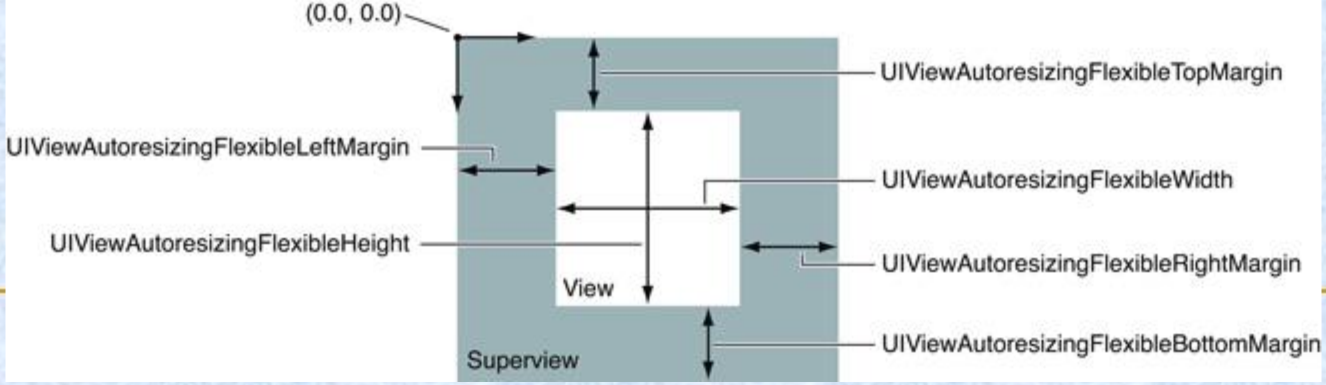
☑ Clears Graphics Context

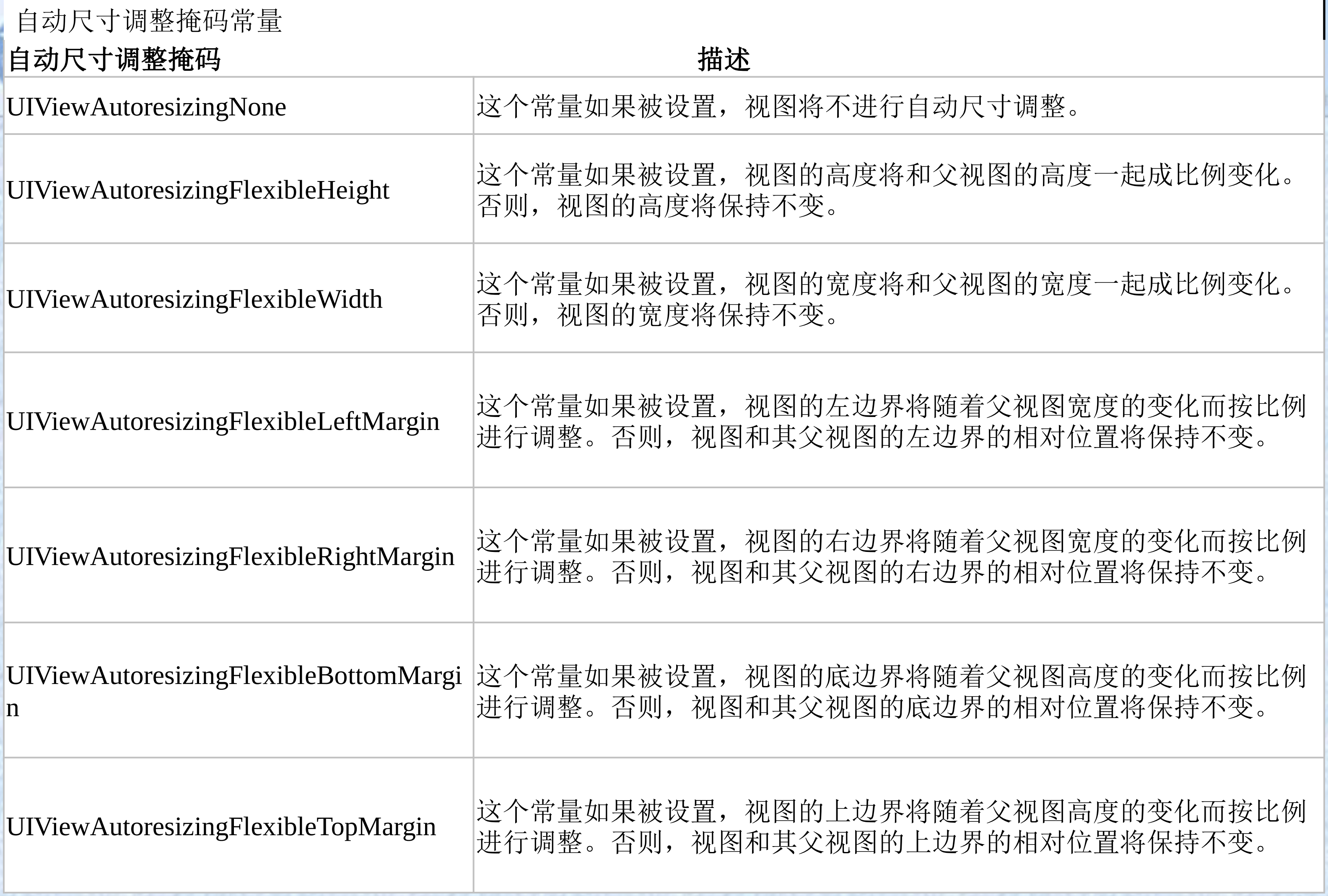
□ Clip Subviews

☑ Autoresize Subviews

当改变视图的边框矩形时，其内嵌子视图的位置和尺寸往往也需要改变，以适应原始视图的新尺寸。如果autoresizesSubviews属性被设置为YES，则其子视图会根据autoresizingMask属性的值自动进行尺寸调整。

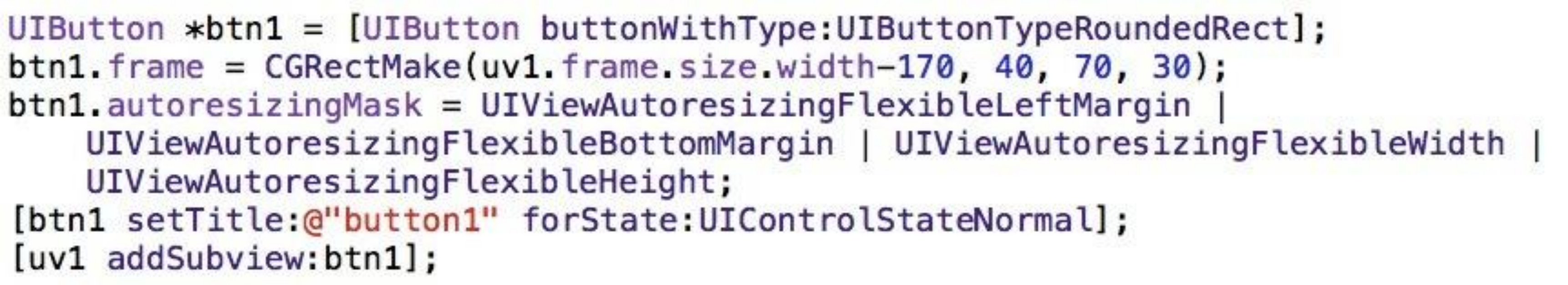
简单配置一下视图的自动尺寸调整掩码常常就能使应用程序得到合适的行为；否则必须通过覆盖layoutSubviews方法来提供自己的实现。


(0.0, 0.0)
UIViewAutoresizingFlexibleTopMargin
UIViewAutoresizingFlexibleLeftMargin
UIViewAutoresizingFlexibleWidth
UIViewAutoresizingFlexibleHeight
UIViewAutoresizingFlexibleRightMargin
View
UIViewAutoresizingFlexibleBottomMargin
Superview



自动尺寸调整掩码常量

自动尺寸调整掩码	描述
UIViewAutoresizingNone	这个常量如果被设置，视图将不进行自动尺寸调整。
UIViewAutoresizingFlexibleHeight	这个常量如果被设置，视图的高度将和父视图的高度一起成比例变化。否则，视图的高度将保持不变。
UIViewAutoresizingFlexibleWidth	这个常量如果被设置，视图的宽度将和父视图的宽度一起成比例变化。否则，视图的宽度将保持不变。
UIViewAutoresizingFlexibleLeftMargin	这个常量如果被设置，视图的左边界将随着父视图宽度的变化而按比例进行调整。否则，视图和其父视图的左边界的相对位置将保持不变。
UIViewAutoresizingFlexibleRightMargin	这个常量如果被设置，视图的右边界将随着父视图宽度的变化而按比例进行调整。否则，视图和其父视图的右边界的相对位置将保持不变。
UIViewAutoresizingFlexibleBottomMargin	这个常量如果被设置，视图的底边界将随着父视图高度的变化而按比例进行调整。否则，视图和其父视图的底边界的相对位置将保持不变。
UIViewAutoresizingFlexibleTopMargin	这个常量如果被设置，视图的上边界将随着父视图高度的变化而按比例进行调整。否则，视图和其父视图的上边界的相对位置将保持不变。



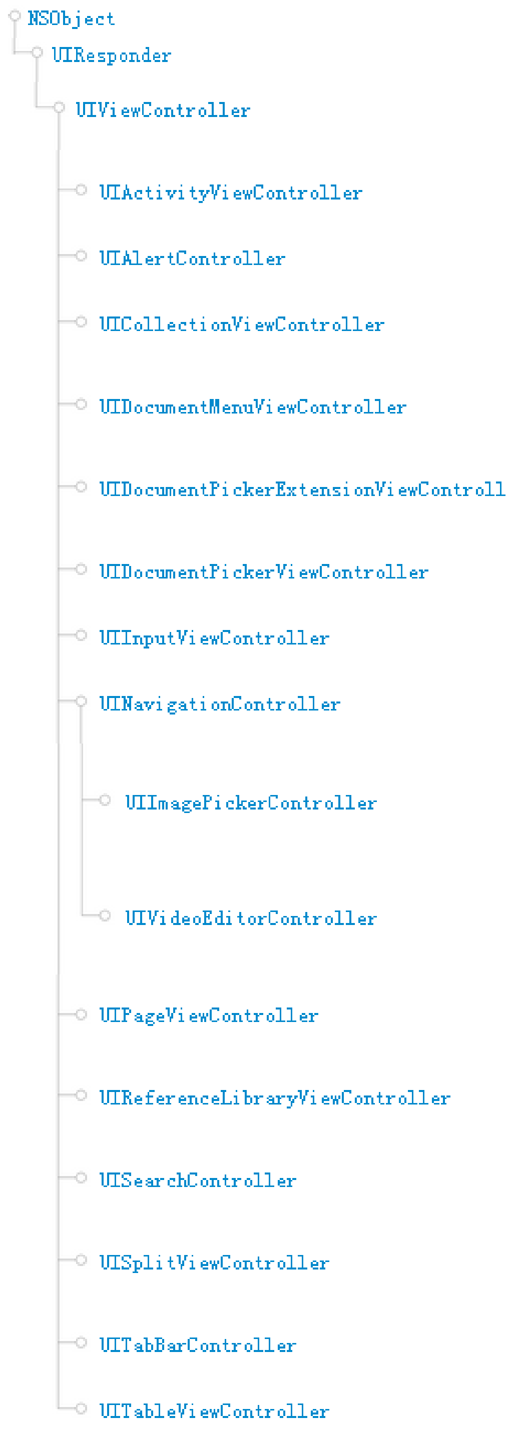
```
UIButton *btn1 = [UIButton buttonWithType:UIButtonTypeRoundedRect];
btn1.frame = CGRectMake(uv1.frame.size.width-170, 40, 70, 30);
btn1.autoresizingMask = UIViewAutoresizingFlexibleLeftMargin |
    UIViewAutoresizingFlexibleBottomMargin | UIViewAutoresizingFlexibleWidth |
    UIViewAutoresizingFlexibleHeight;
[btn1 setTitle:@"button1" forState:UIControlStateNormal];
[uv1 addSubview:btn1];
```

五、视图控制器

1、视图控制器的作用

- 创建和管理视图。
- 管理视图上显示的数据。
- 设备方向变化，调整视图大小以适应屏幕。
- 负责视图和模型之间的数据及请示的传递。

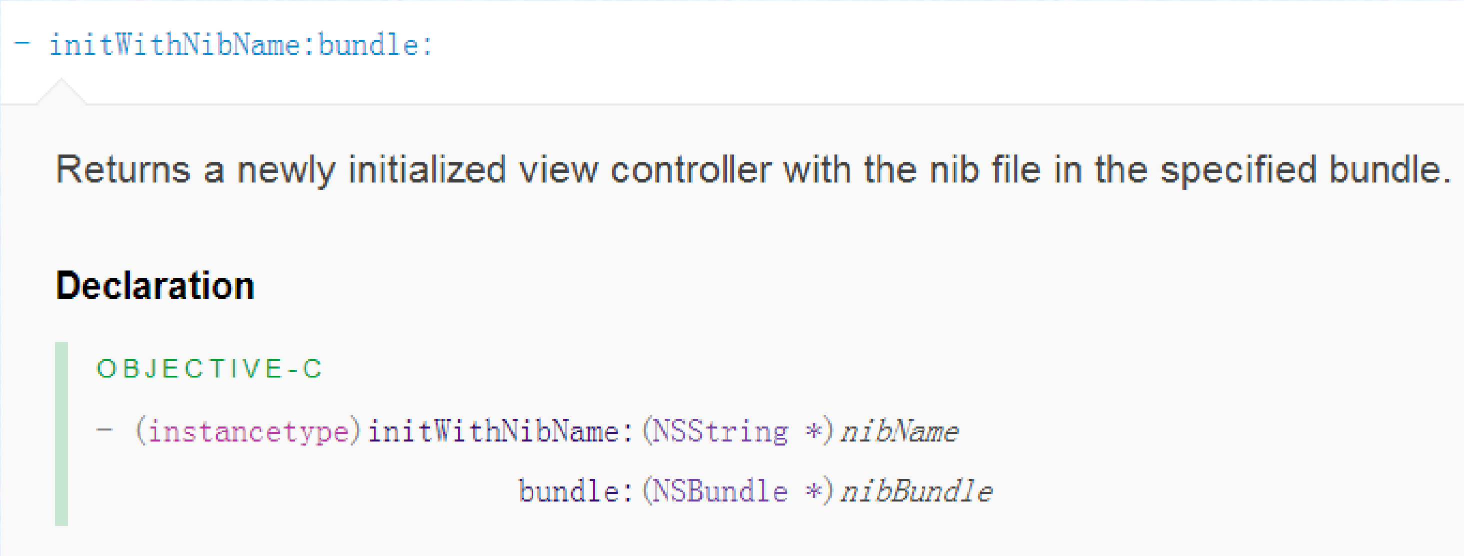
在IOS中，视图控制器对应的类是UIViewController。它是视图和数据间的桥梁。



```
NSObject
  UIResponder
    UIViewController
      UIActivityViewController
      UIAlertController
      UICollectionViewController
      UIDocumentMenuViewController
      UIDocumentPickerExtensionViewControll
      UIDocumentPickerViewController
      UIInputViewController
      UINavigationController
        UIImagePickerController
        UIVideoEditorController
      UIPageViewController
      UIReferenceLibraryViewController
      UISearchController
      UISplitViewController
      UITabBarController
      UITableViewController
```

2、视图控制器的创建和初始化

- 方法一：利用代码生成



- initWithNibName:bundle:

Returns a newly initialized view controller with the nib file in the specified bundle.

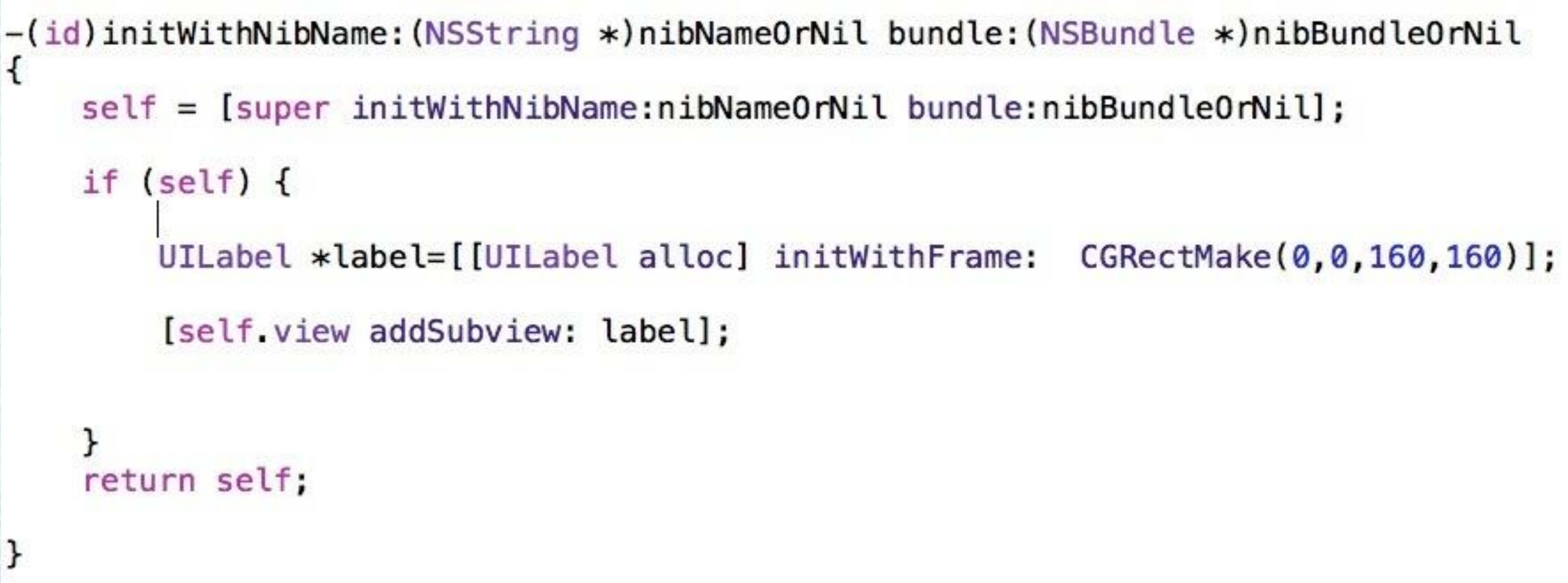
Declaration

OBJECTIVE-C

```
- (instancetype)initWithNibName:(NSString *)nibName
                         bundle:(NSBundle *)nibBundle
```

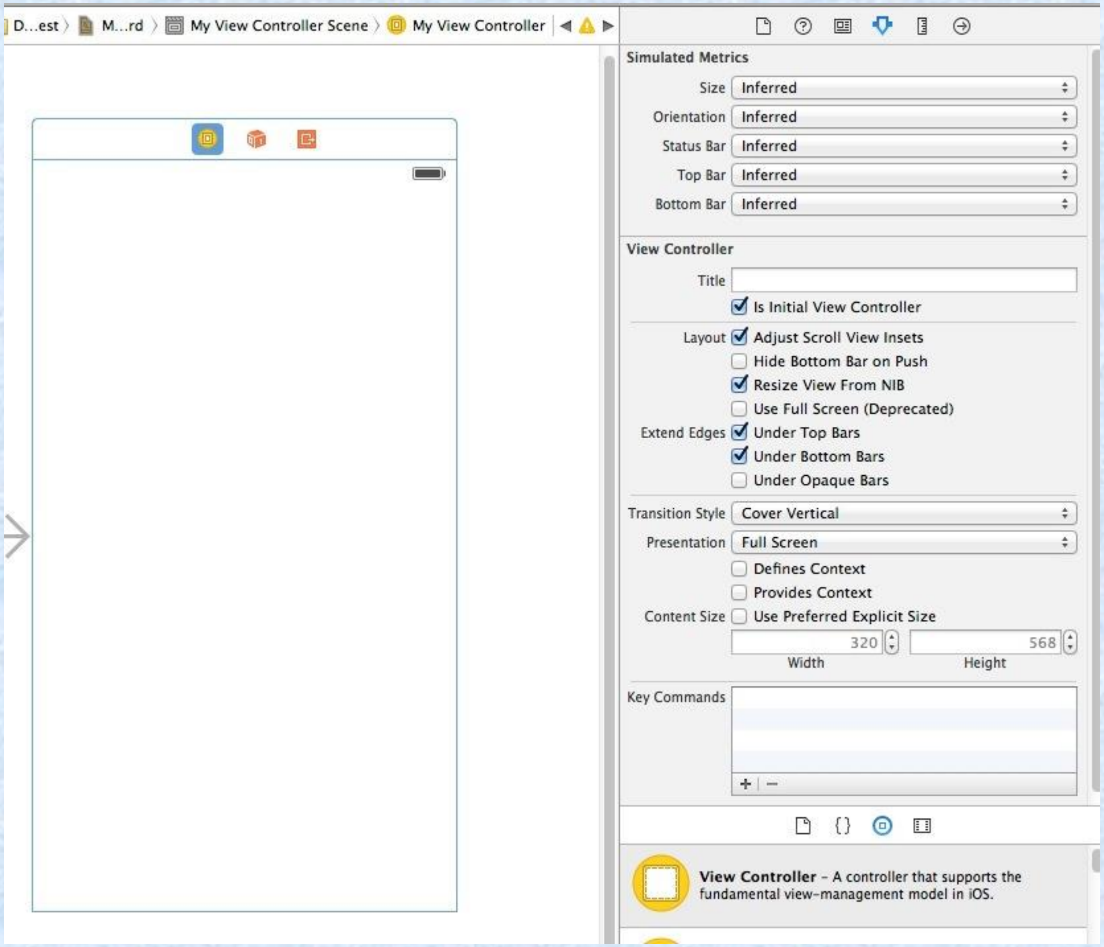
Xcode5后不推荐使用initWithNibName方法实例化

可以覆盖initWithNibName方法，添加自己的初始化代码。



```
-(id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];

    if (self) {

        UILabel *label=[[UILabel alloc] initWithFrame:  CGRectMake(0,0,160,160)];

        [self.view addSubview: label];


    }
    return self;

}
```

- 方法二：在Storyboard利用IB创建



D…est 〉 M…rd 〉 My View Controller Scene 〉 My View Controller

Simulated Metrics

Size: Inferred
Orientation: Inferred
Status Bar: Inferred
Top Bar: Inferred
Bottom Bar: Inferred

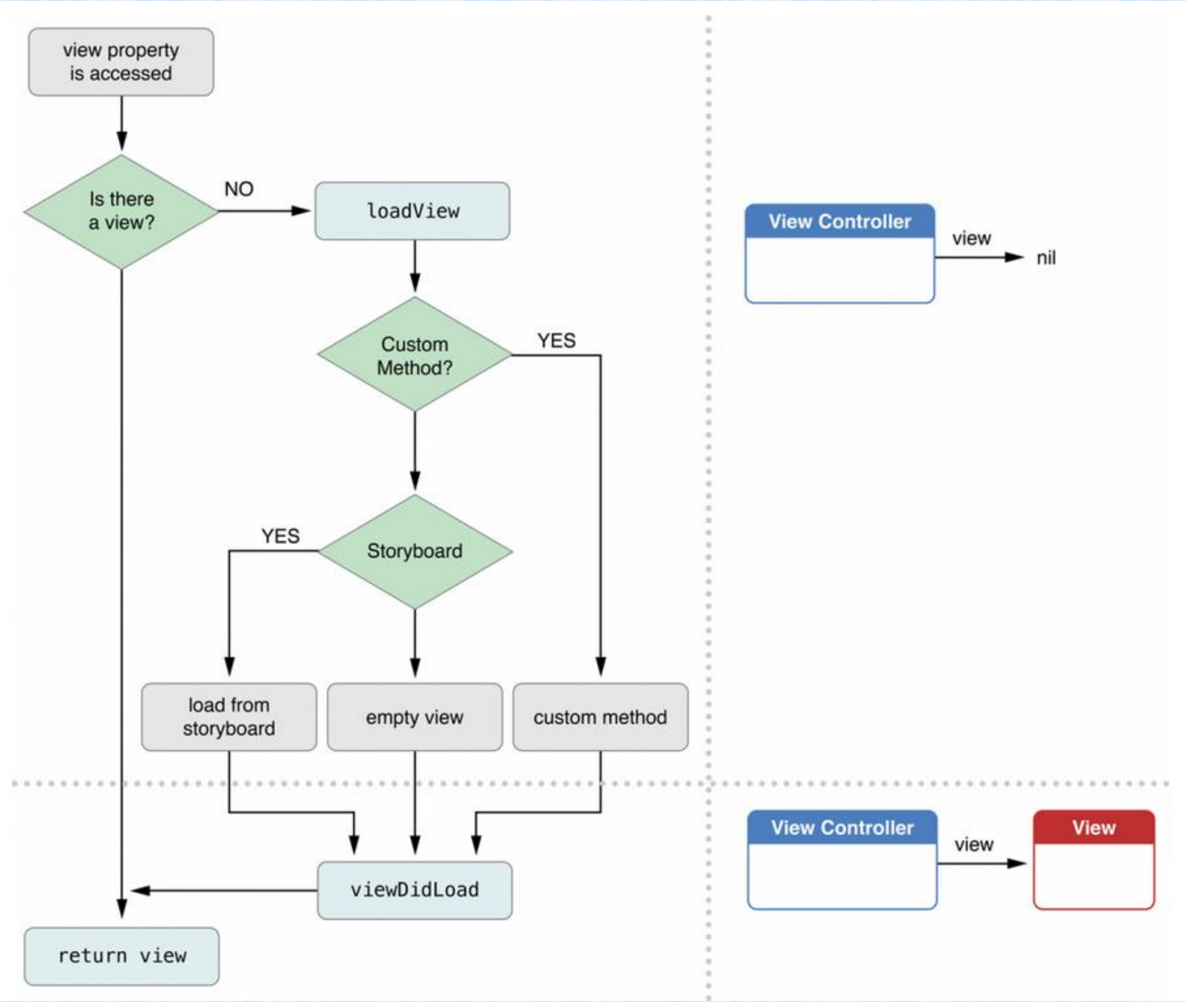
View Controller

Title:
☑ Is Initial View Controller
Layout: ☑ Adjust Scroll View Insets
□ Hide Bottom Bar on Push
☑ Resize View From NIB
□ Use Full Screen (Deprecated)
Extend Edges: ☑ Under Top Bars
☑ Under Bottom Bars
□ Under Opaque Bars
Transition Style: Cover Vertical
Presentation: Full Screen
□ Defines Context
□ Provides Context
Content Size: □ Use Preferred Explicit Size
320 Width
568 Height
Key Commands
+ −

View Controller – A controller that supports the fundamental view-management model in iOS.

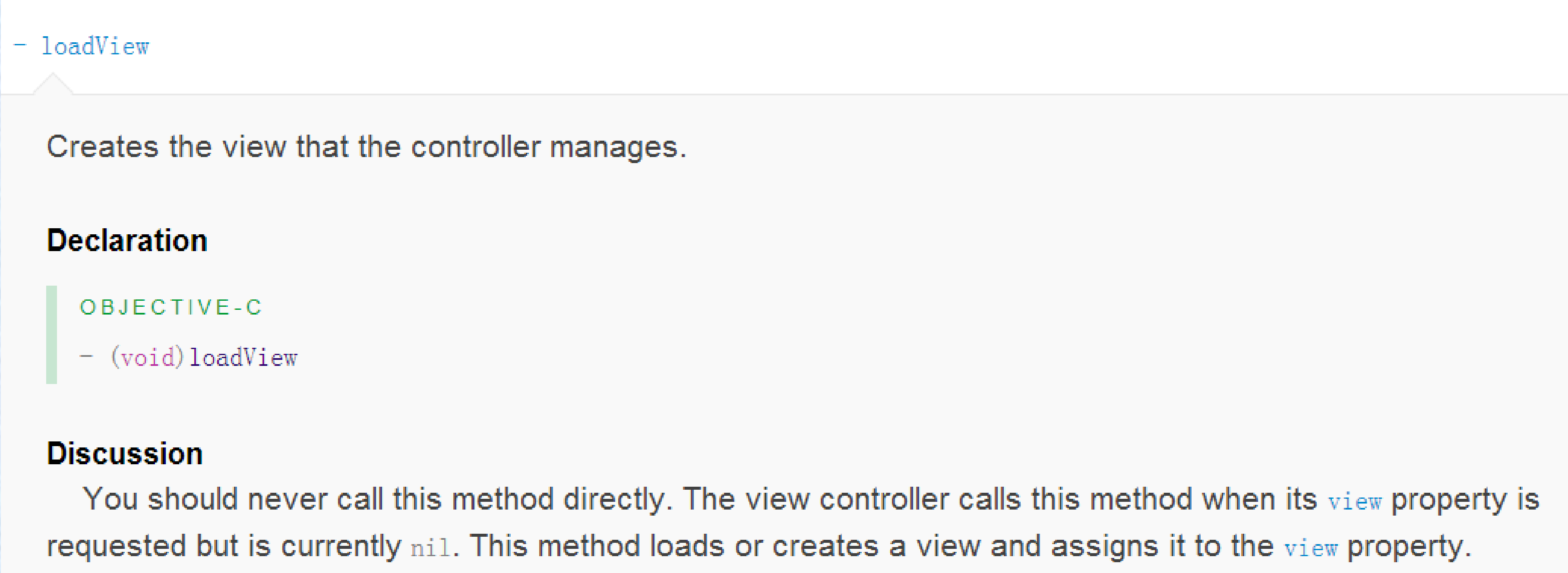
推荐使用Storyboard进行创建

2、视图控制器的生命周期



- loadView

当由视图控制器管理的视图为nil时触发。



- loadView

Creates the view that the controller manages.

Declaration

OBJECTIVE-C

```
- (void)loadView
```

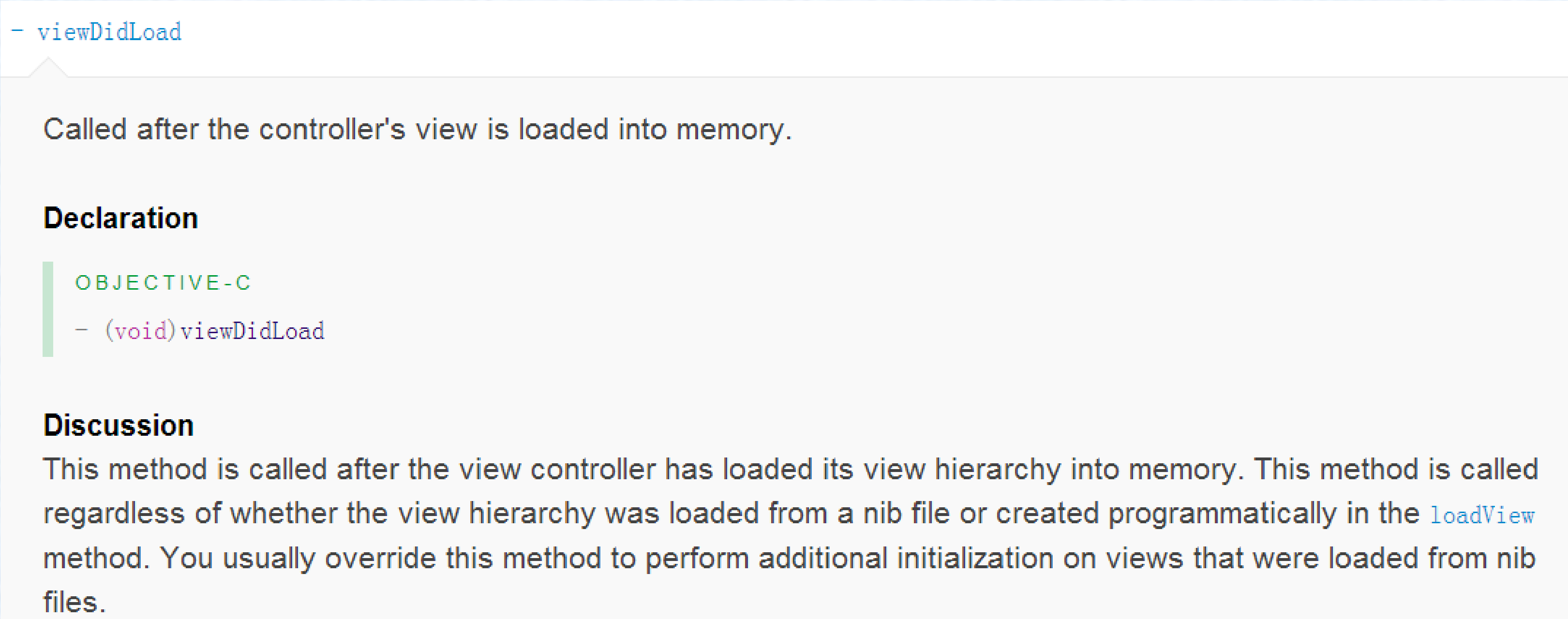
Discussion

You should never call this method directly. The view controller calls this method when its `view` property is requested but is currently `nil`. This method loads or creates a view and assigns it to the `view` property.

当视图控制器由Storyboard创建时，无需考虑。
此方法可用于以代码的方式创建视图时用到。

- viewDidLoad

在由Storyboard实例化,后且所有的outlet设置完成后触发。



- viewDidLoad

Called after the controller's view is loaded into memory.

Declaration

OBJECTIVE-C

```
- (void)viewDidLoad
```

Discussion

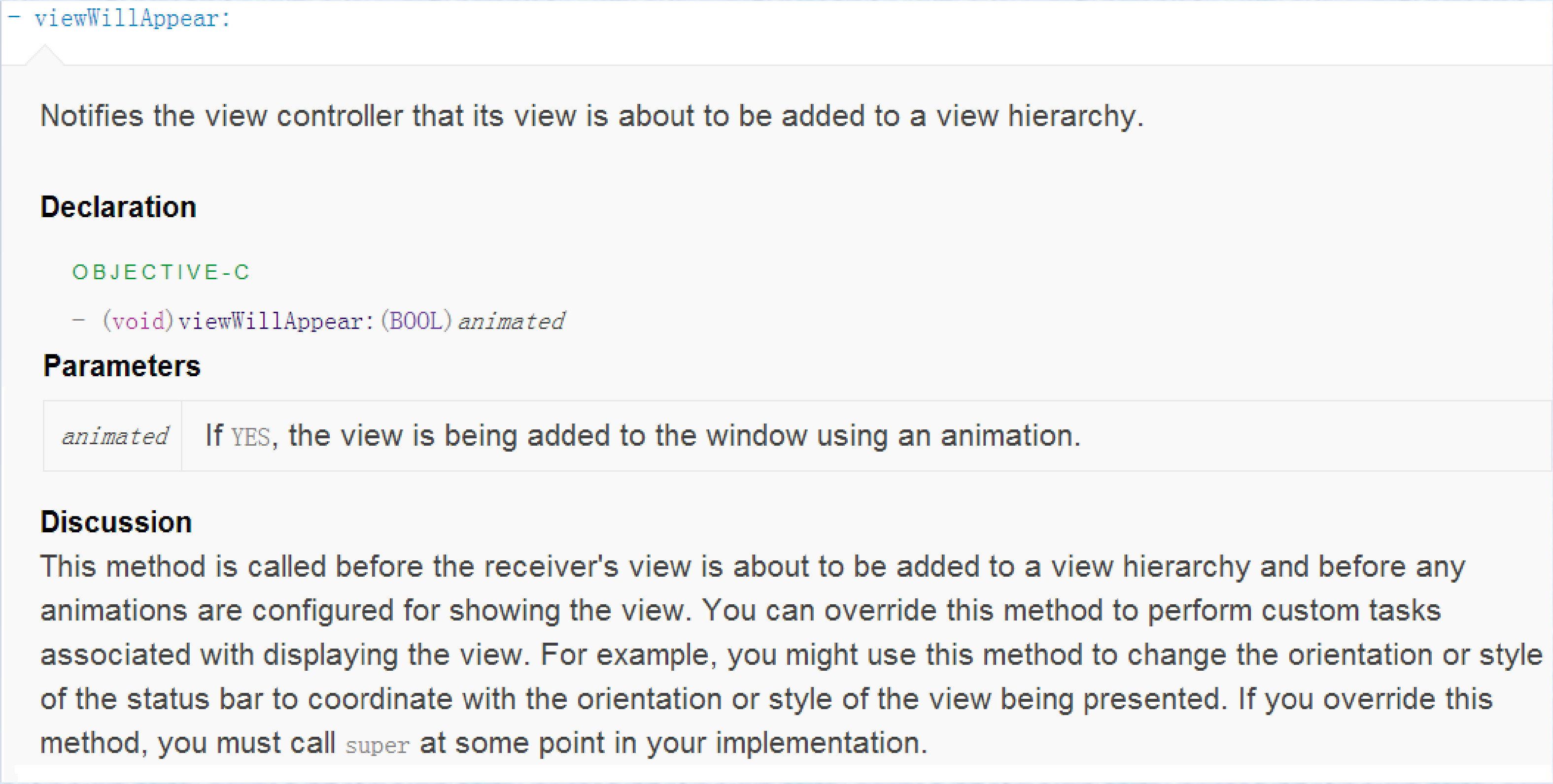
This method is called after the view controller has loaded its view hierarchy into memory. This method is called regardless of whether the view hierarchy was loaded from a nib file or created programmatically in the `loadView` method. You usually override this method to perform additional initialization on views that were loaded from nib files.

注意：●如果视图控制器是由代码方式创建，也会触发该方法

●不要修改视图几何信息

- viewWillAppear

在视图显示在屏幕前前触发。



`- viewWillAppear:`

Notifies the view controller that its view is about to be added to a view hierarchy.

Declaration

OBJECTIVE-C

```
- (void)viewWillAppear:(BOOL)animated
```

Parameters

animated	If `YES`, the view is being added to the window using an animation.

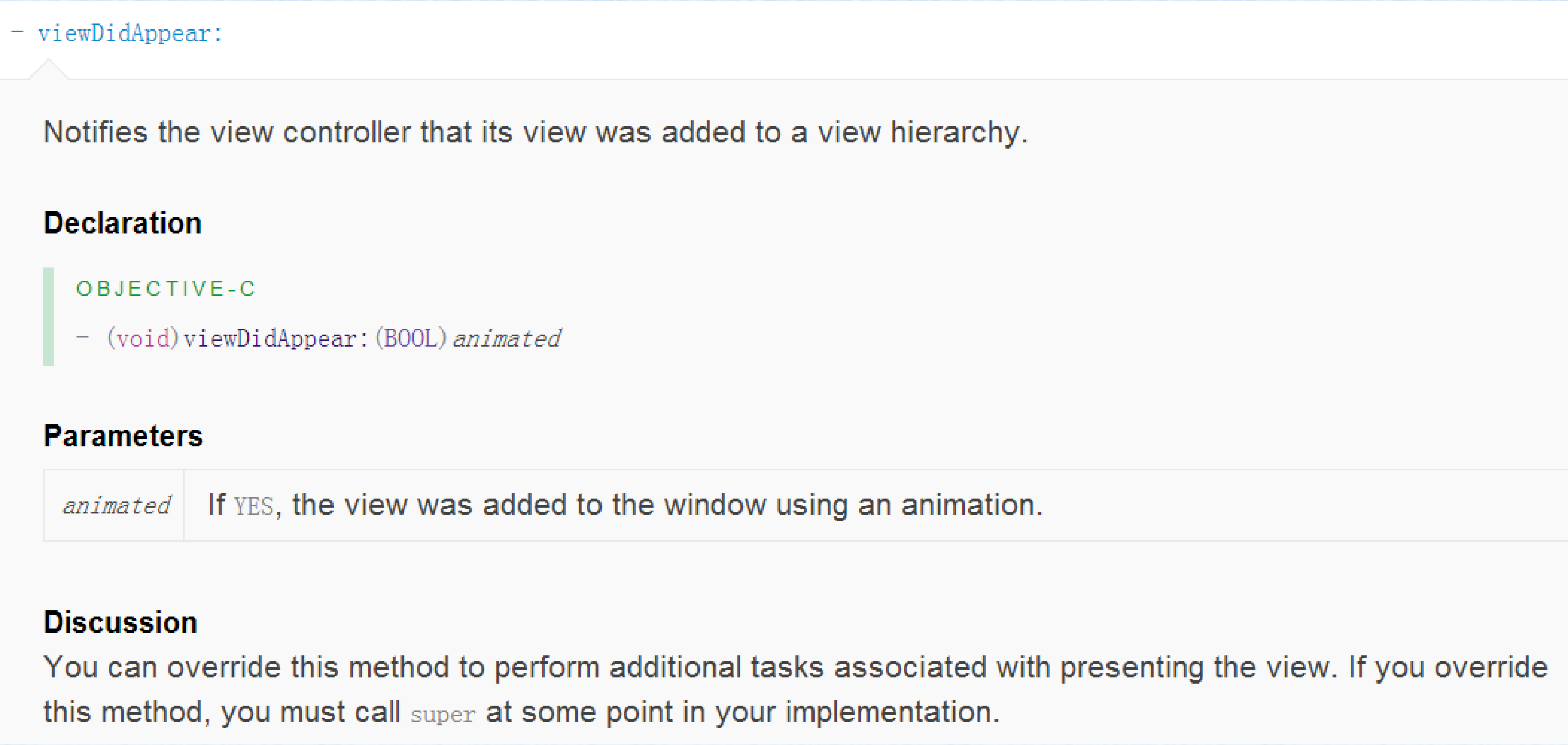
Discussion

This method is called before the receiver's view is about to be added to a view hierarchy and before any animations are configured for showing the view. You can override this method to perform custom tasks associated with displaying the view. For example, you might use this method to change the orientation or style of the status bar to coordinate with the orientation or style of the view being presented. If you override this method, you must call `super` at some point in your implementation.

比较viewWillAppear和viewDidLoad

- viewDidAppear

  在视图在屏幕上显示后触发。



- viewDidAppear:

Notifies the view controller that its view was added to a view hierarchy.

Declaration

OBJECTIVE-C

```
- (void)viewDidAppear:(BOOL)animated
```

Parameters

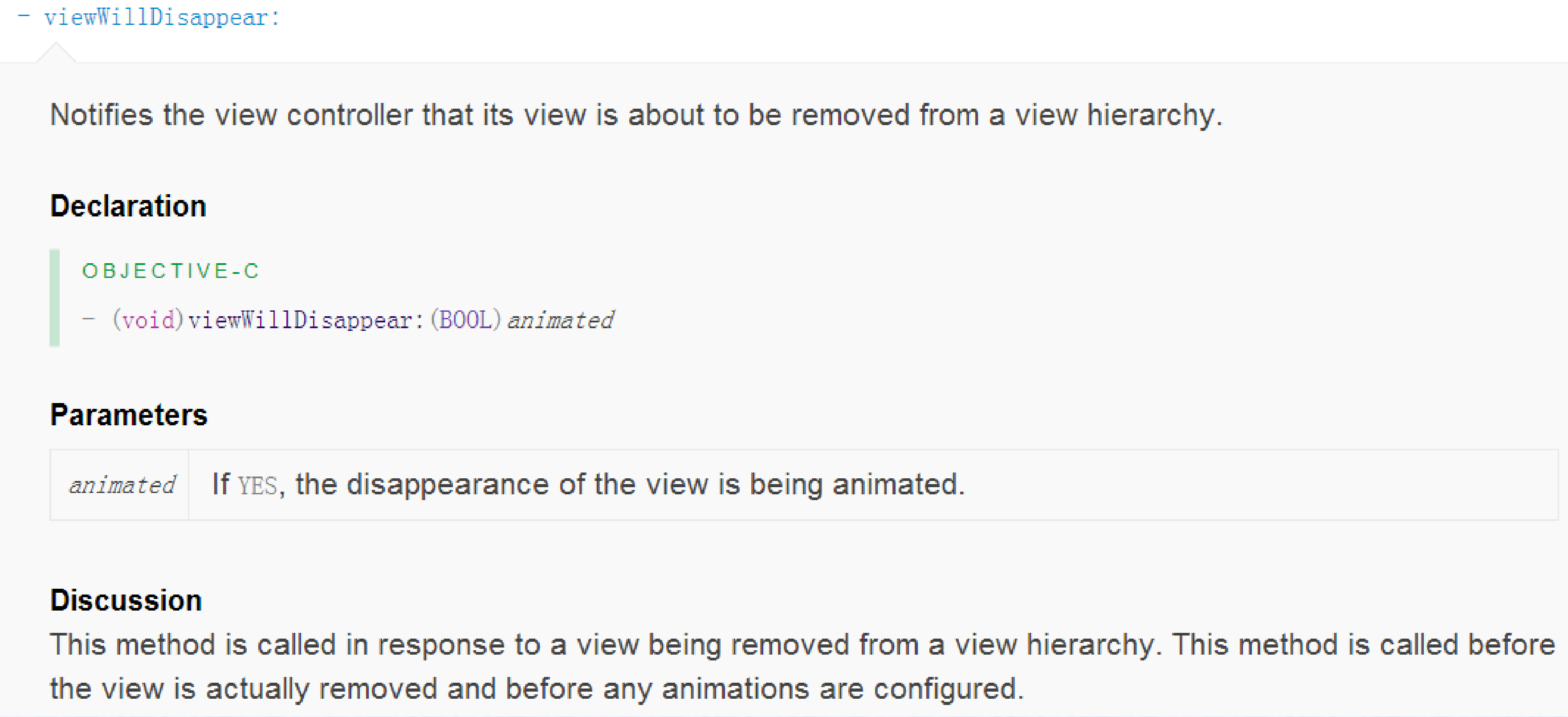
animated	If `YES`, the view was added to the window using an animation.

Discussion

You can override this method to perform additional tasks associated with presenting the view. If you override this method, you must call `super` at some point in your implementation.

- viewWillDisappear

  在视图离开屏幕前触发。



- viewWillDisappear:

Notifies the view controller that its view is about to be removed from a view hierarchy.

Declaration

OBJECTIVE-C

```
- (void)viewWillDisappear:(BOOL)animated
```

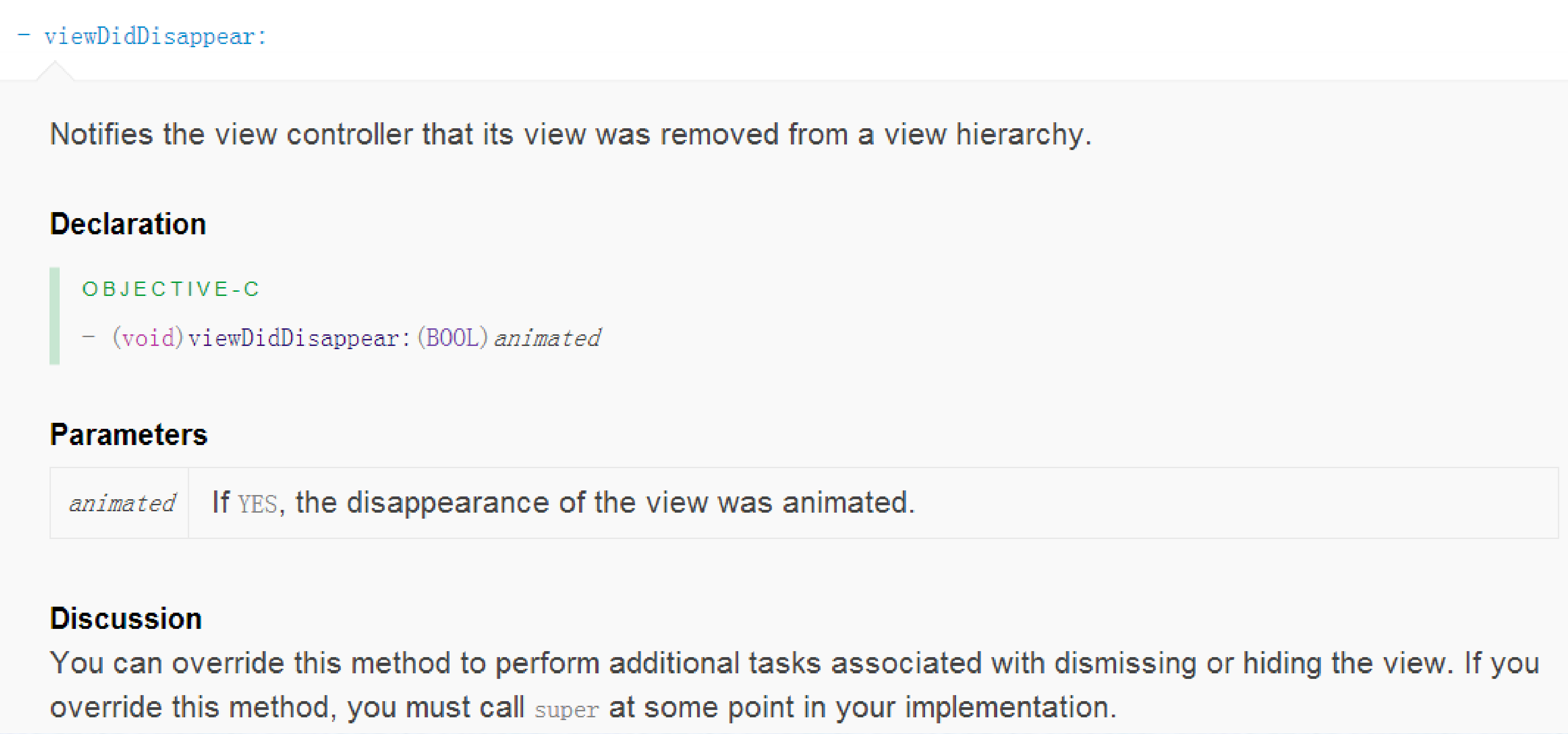
Parameters

animated	If YES, the disappearance of the view is being animated.

Discussion

This method is called in response to a view being removed from a view hierarchy. This method is called before the view is actually removed and before any animations are configured.

- viewDidDisappear
  在视图离开屏幕后触发。



- viewDidDisappear:

Notifies the view controller that its view was removed from a view hierarchy.

Declaration

OBJECTIVE-C

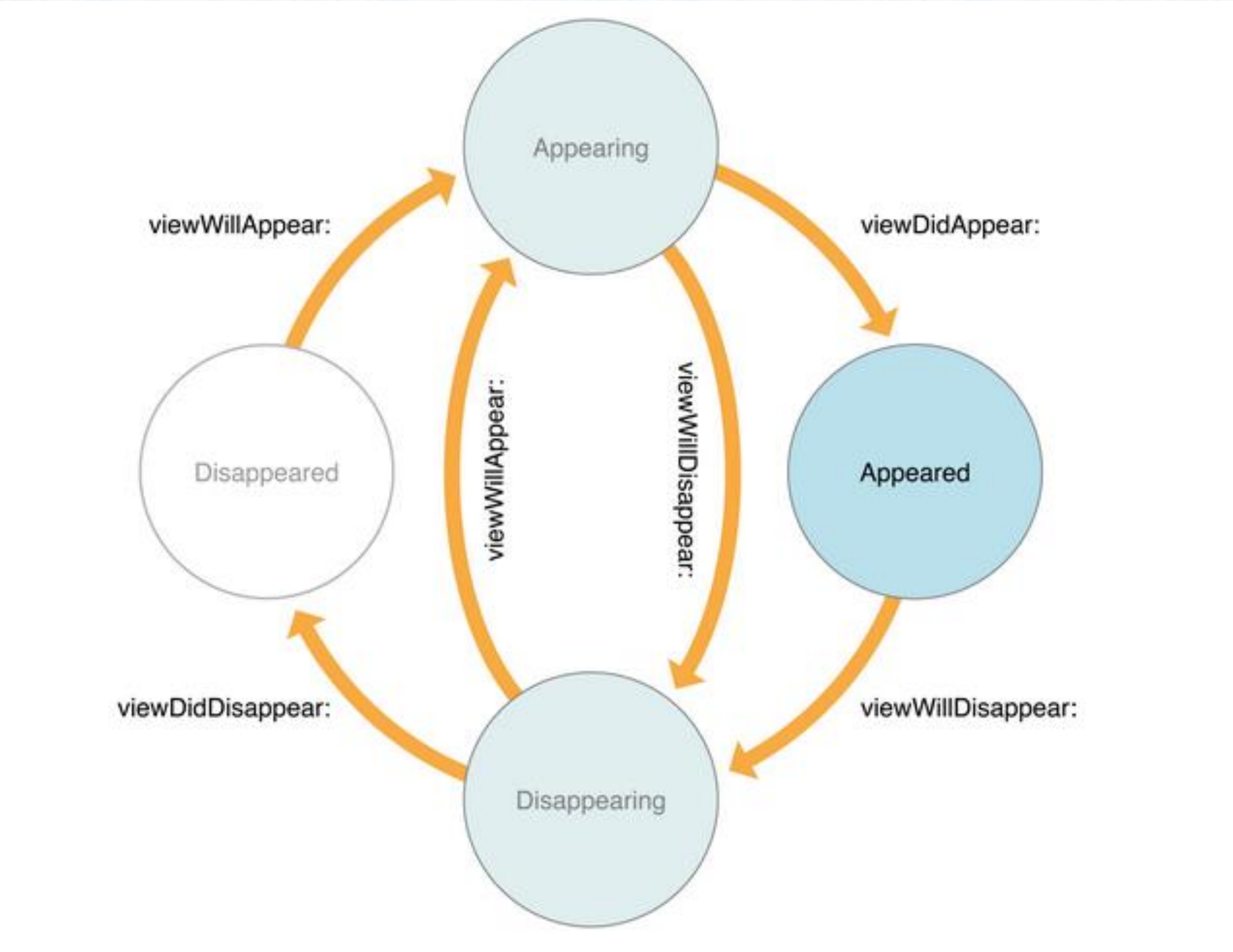
```
- (void)viewDidDisappear:(BOOL)animated
```

Parameters

animated	If `YES`, the disappearance of the view was animated.

Discussion

You can override this method to perform additional tasks associated with dismissing or hiding the view. If you override this method, you must call `super` at some point in your implementation.


Appearing
viewWillAppear:
viewDidAppear:
Disappeared
viewWillAppear:
viewWillDisappear:
Appeared
viewDidDisappear:
viewWillDisappear:
Disappearing

其它一些常用方法

自动布局：

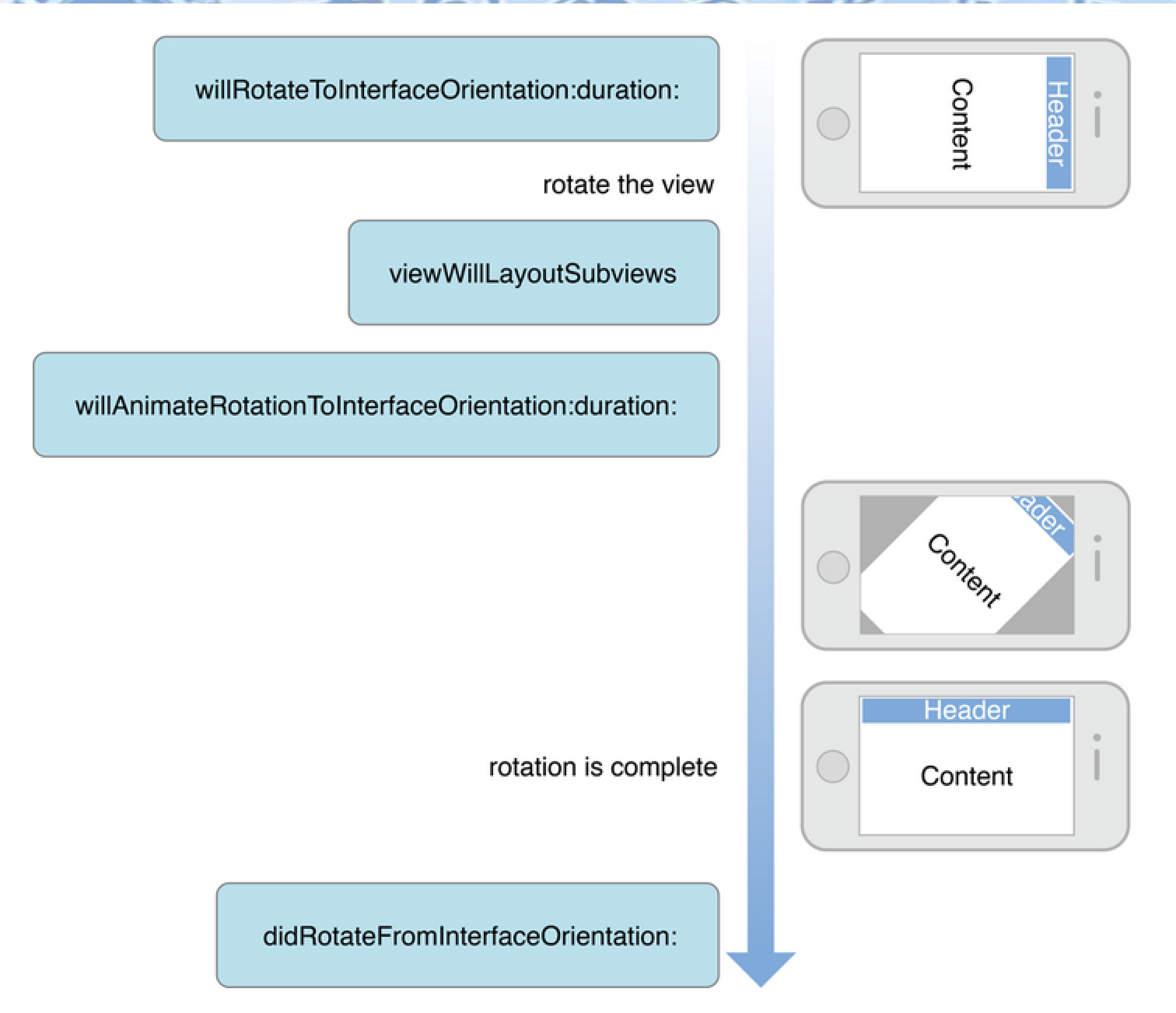
- - (void)viewWillLayoutSubviews
- - (void)viewDidLayoutSubviews

旋转相关

- - (void) willRotateToInterfaceOrientation: (UIInterfaceOrientation) anOrientation
  duration:(NSTimeInterval) seconds
- - (void) willAnimateRotationToInterfaceOrientation: (UIInterfaceOriention) orient
  duration:(NSTimeInterval)seconds
- - (void) didRotateFromInterfaceOrientation:(UIInterfaceOrientation)anOrientation

- - (BOOL) shouldAutorotate

- - (NSUInteger) supportedInterfaceOrientations



willRotateToInterfaceOrientation:duration:

rotate the view

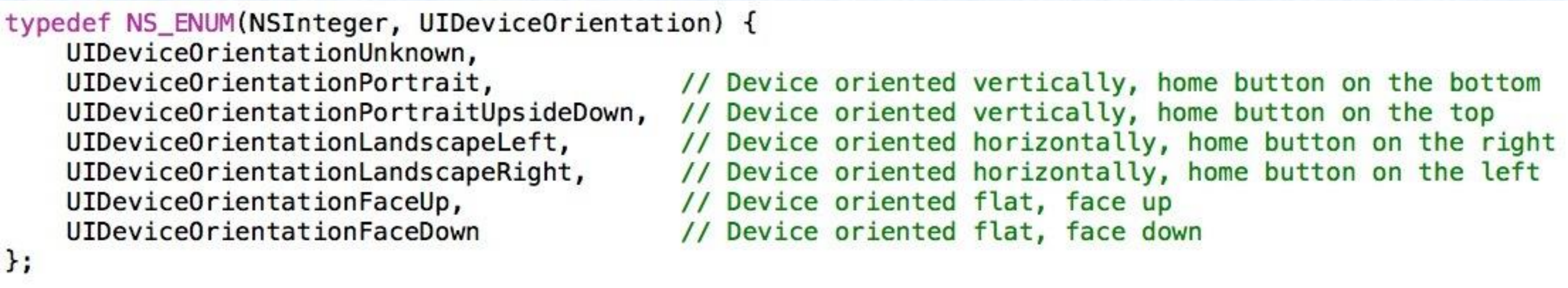
viewWillLayoutSubviews

willAnimateRotationToInterfaceOrientation:duration:

rotation is complete

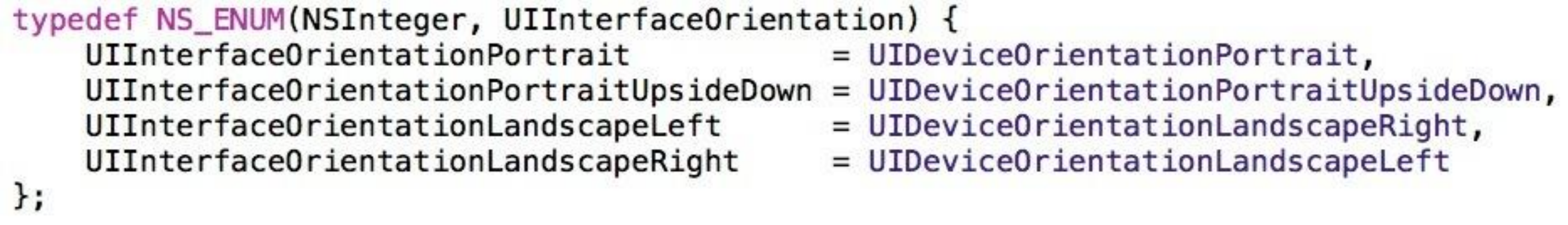
didRotateFromInterfaceOrientation:

设备方向



```
typedef NS_ENUM(NSInteger, UIDeviceOrientation) {
    UIDeviceOrientationUnknown,
    UIDeviceOrientationPortrait,             // Device oriented vertically, home button on the bottom
    UIDeviceOrientationPortraitUpsideDown,   // Device oriented vertically, home button on the top
    UIDeviceOrientationLandscapeLeft,        // Device oriented horizontally, home button on the right
    UIDeviceOrientationLandscapeRight,       // Device oriented horizontally, home button on the left
    UIDeviceOrientationFaceUp,               // Device oriented flat, face up
    UIDeviceOrientationFaceDown              // Device oriented flat, face down
};
```

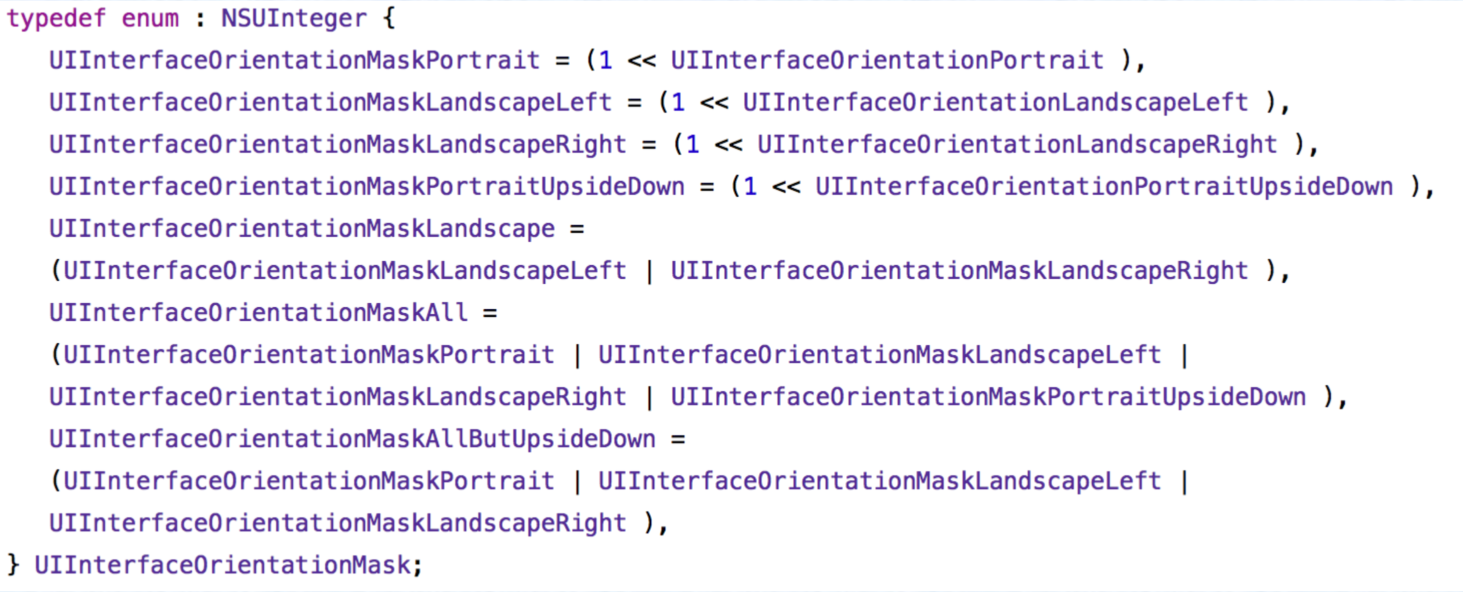
界面方向



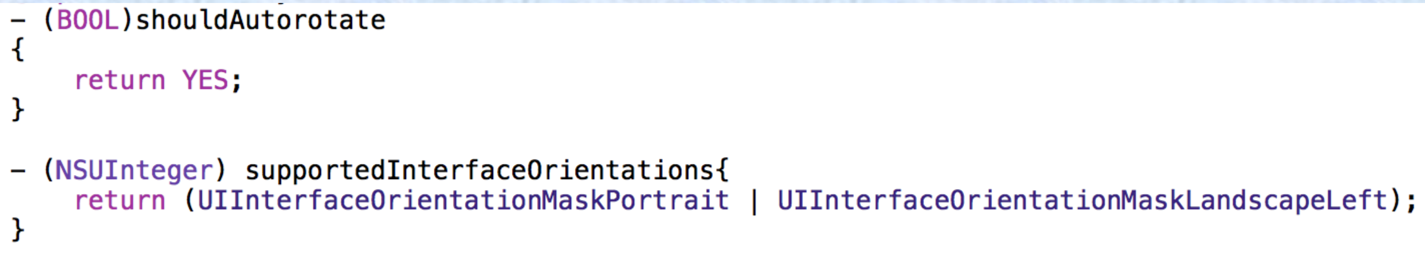
```
typedef NS_ENUM(NSInteger, UIInterfaceOrientation) {
    UIInterfaceOrientationPortrait           = UIDeviceOrientationPortrait,
    UIInterfaceOrientationPortraitUpsideDown = UIDeviceOrientationPortraitUpsideDown,
    UIInterfaceOrientationLandscapeLeft      = UIDeviceOrientationLandscapeRight,
    UIInterfaceOrientationLandscapeRight     = UIDeviceOrientationLandscapeLeft
};
```

UIInterfaceOrientationMask

These constants are mask bits for specifying a view controller’s supported interface orientations

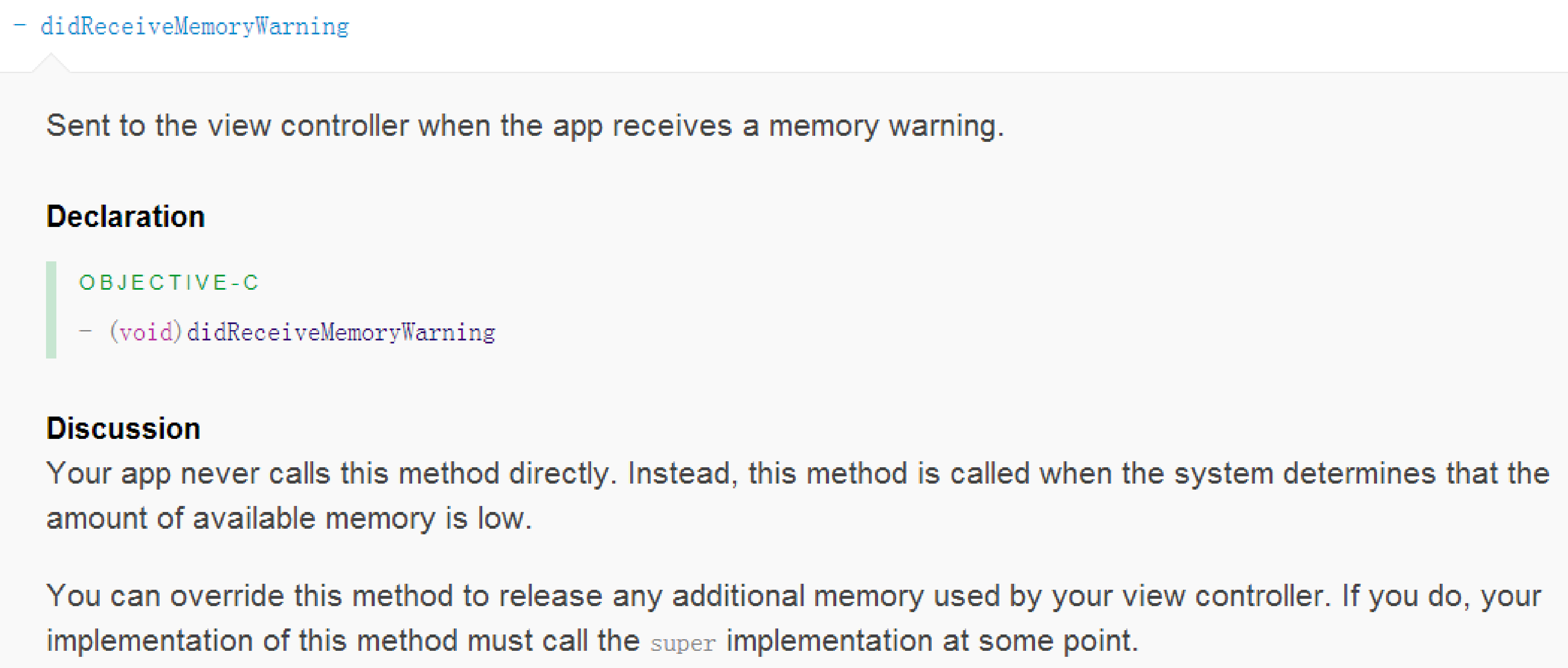


```
typedef enum : NSUInteger {
   UIInterfaceOrientationMaskPortrait = (1 << UIInterfaceOrientationPortrait ),
   UIInterfaceOrientationMaskLandscapeLeft = (1 << UIInterfaceOrientationLandscapeLeft ),
   UIInterfaceOrientationMaskLandscapeRight = (1 << UIInterfaceOrientationLandscapeRight ),
   UIInterfaceOrientationMaskPortraitUpsideDown = (1 << UIInterfaceOrientationPortraitUpsideDown ),
   UIInterfaceOrientationMaskLandscape =
   (UIInterfaceOrientationMaskLandscapeLeft | UIInterfaceOrientationMaskLandscapeRight ),
   UIInterfaceOrientationMaskAll =
   (UIInterfaceOrientationMaskPortrait | UIInterfaceOrientationMaskLandscapeLeft |
   UIInterfaceOrientationMaskLandscapeRight | UIInterfaceOrientationMaskPortraitUpsideDown ),
   UIInterfaceOrientationMaskAllButUpsideDown =
   (UIInterfaceOrientationMaskPortrait | UIInterfaceOrientationMaskLandscapeLeft |
   UIInterfaceOrientationMaskLandscapeRight ),
} UIInterfaceOrientationMask;
```



```
- (BOOL)shouldAutorotate
{
    return YES;
}

- (NSUInteger) supportedInterfaceOrientations{
    return (UIInterfaceOrientationMaskPortrait | UIInterfaceOrientationMaskLandscapeLeft);
}
```

- didReceiveMemoryWarning



- didReceiveMemoryWarning

Sent to the view controller when the app receives a memory warning.

Declaration

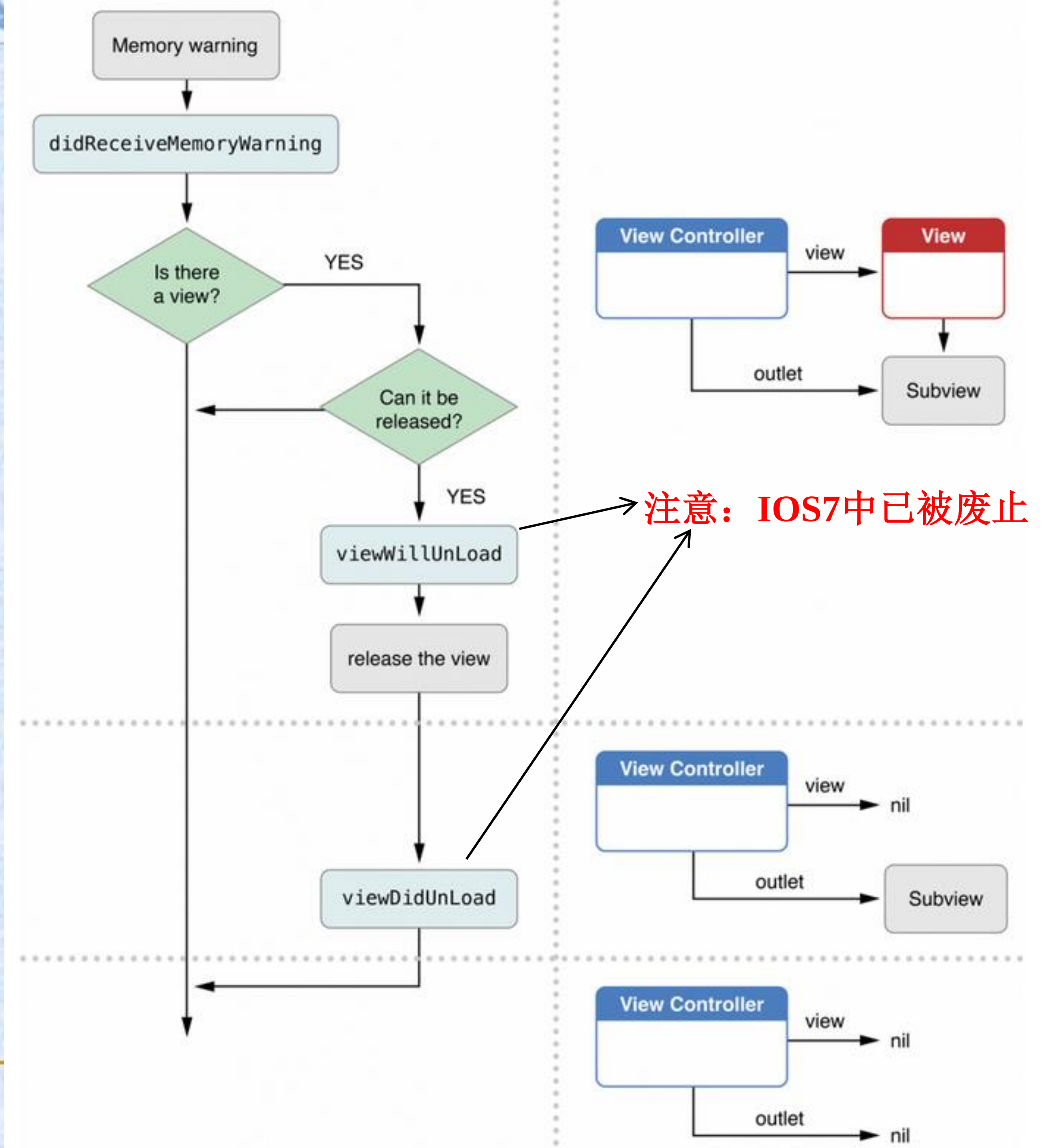
OBJECTIVE-C

```
- (void)didReceiveMemoryWarning
```

Discussion

Your app never calls this method directly. Instead, this method is called when the system determines that the amount of available memory is low.

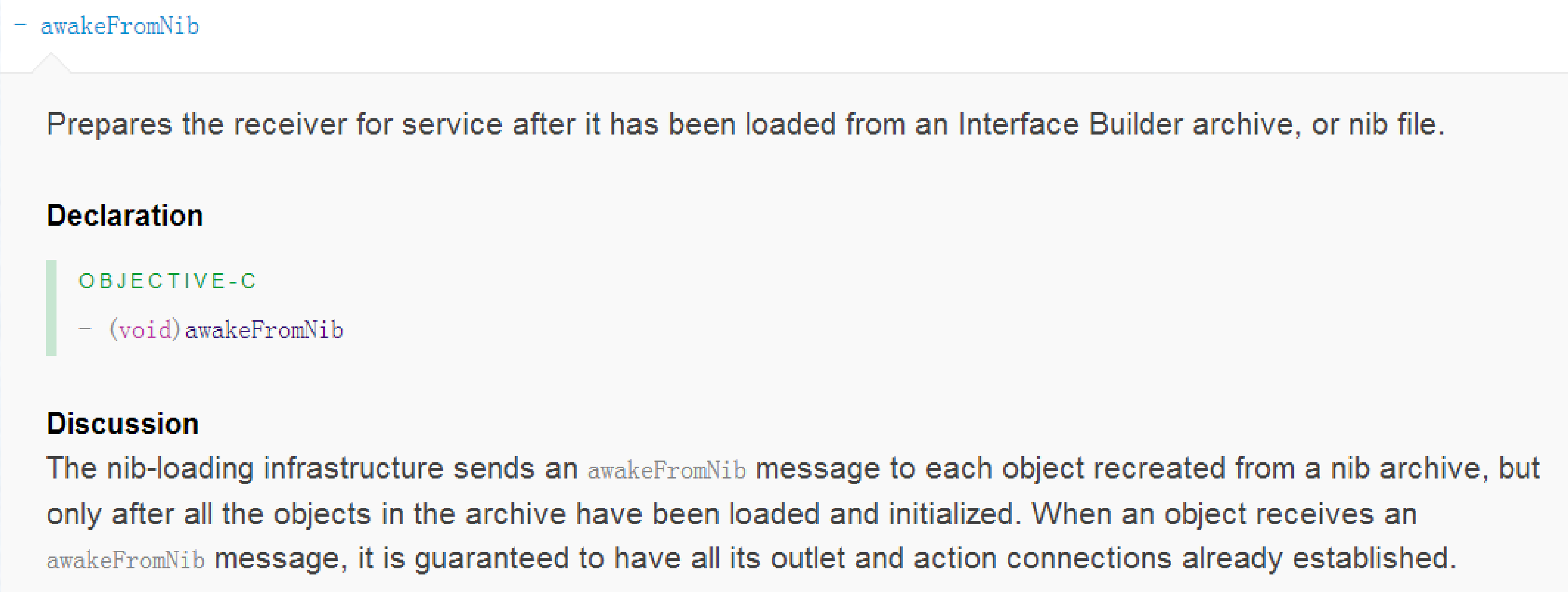
You can override this method to release any additional memory used by your view controller. If you do, your implementation of this method must call the `super` implementation at some point.



注意：IOS7中已被废止

- awakeFromNib

当.nib文件或IB archive 被加载的时，会发送一个awakeFromNib的消息到个对象，每个对象都可以定义自己的 awakeFromNib函数来响应这个消息，执行一些必要的操作。



- awakeFromNib

Prepares the receiver for service after it has been loaded from an Interface Builder archive, or nib file.

Declaration

OBJECTIVE-C

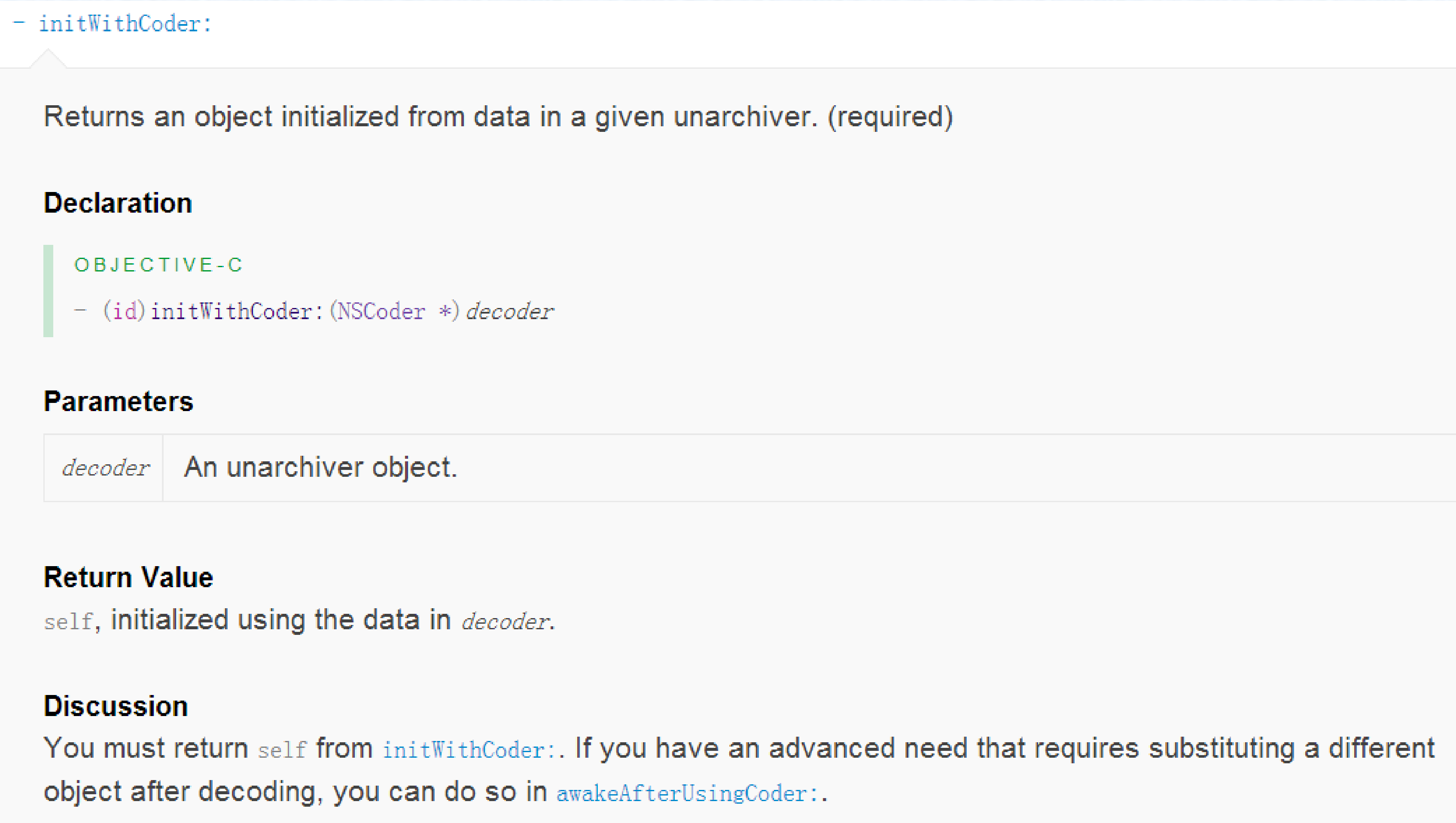
```
- (void)awakeFromNib
```

Discussion

The nib-loading infrastructure sends an `awakeFromNib` message to each object recreated from a nib archive, but only after all the objects in the archive have been loaded and initialized. When an object receives an `awakeFromNib` message, it is guaranteed to have all its outlet and action connections already established.

- initWithCoder

当视图从.nib文件或IB archive 会加载对象实例时， 使用 initWithCoder初始化这些实例对象。



`- initWithCoder:`

Returns an object initialized from data in a given unarchiver. (required)

Declaration

OBJECTIVE-C

```
- (id)initWithCoder:(NSCoder *)decoder
```

Parameters

decoder	An unarchiver object.

Return Value

`self`, initialized using the data in *decoder*.

Discussion

You must return `self` from `initWithCoder:`. If you have an advanced need that requires substituting a different object after decoding, you can do so in `awakeAfterUsingCoder:`.

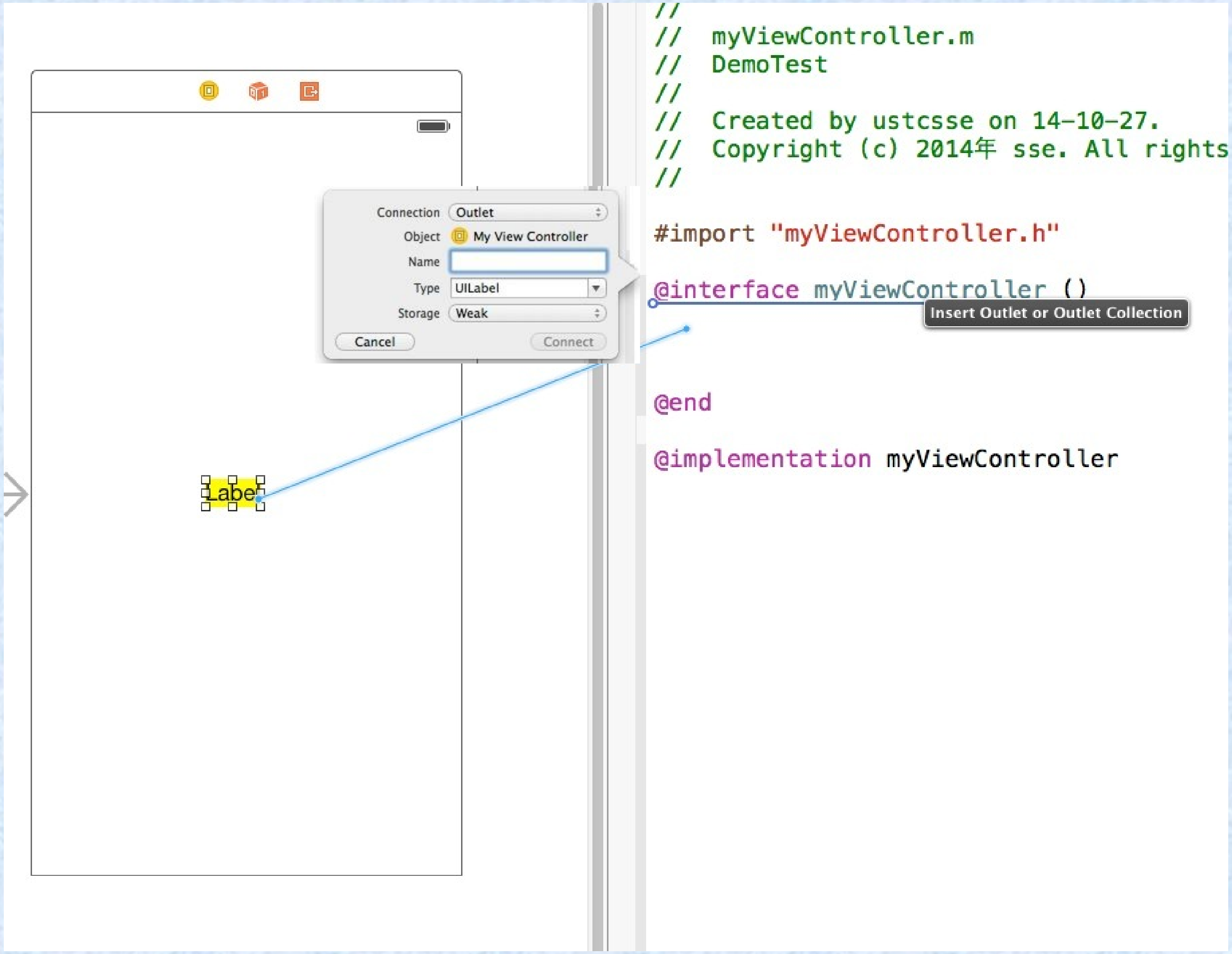
六、Outlet和Action

当在Storyboard中使用Interface Builder以图形化的方式建立用户界面后，必须设置Outlet和Action，将Interface Builder中放置的控件与代码之间建立连接。

使用Outlet，以定义变量的方式将控件链接到代码，该变量代表控件。

使用Action，以定义方法的方式将控件链接到代码，该方法在用户与控件进行交互时运行。

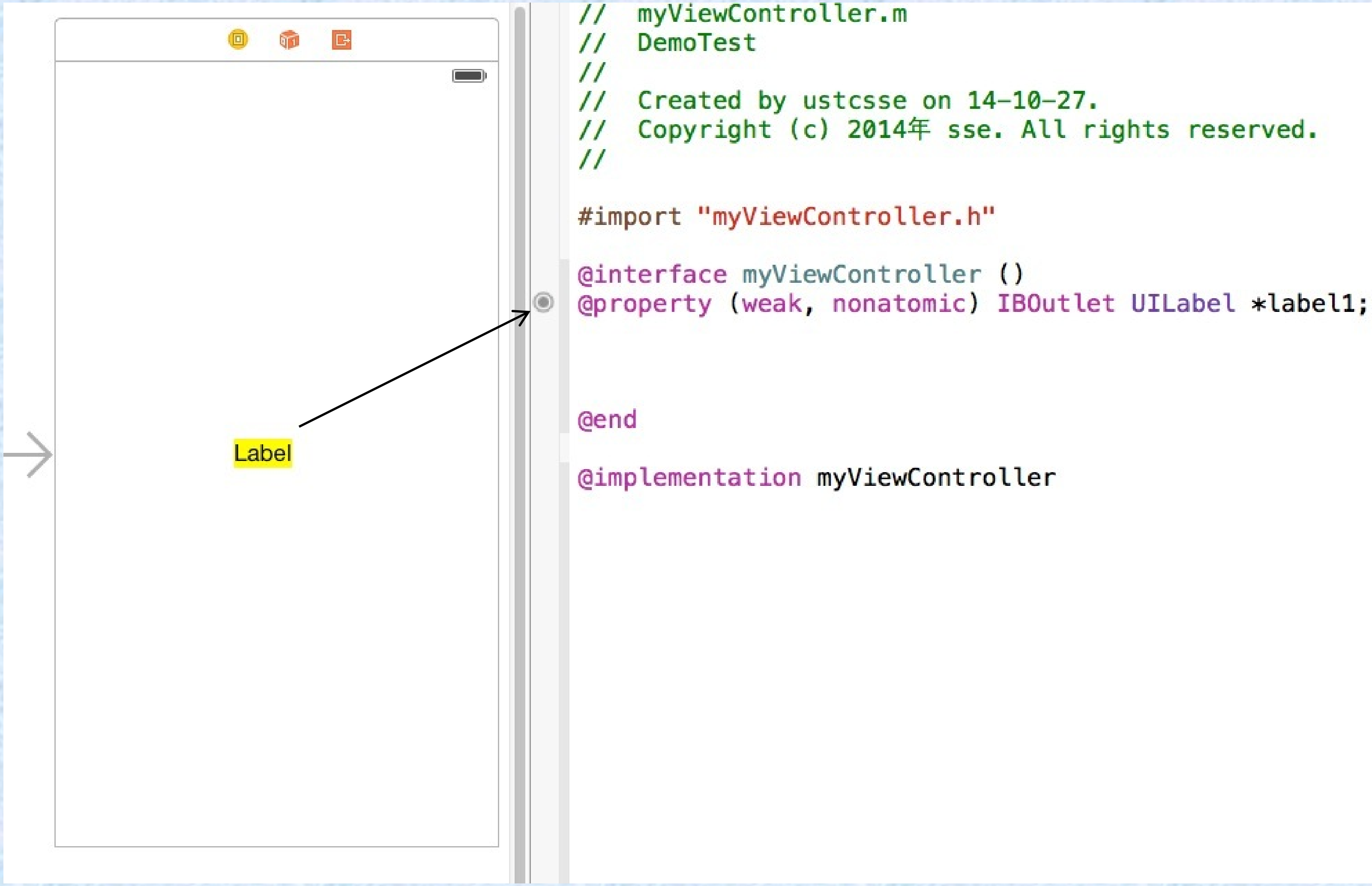
设置方法



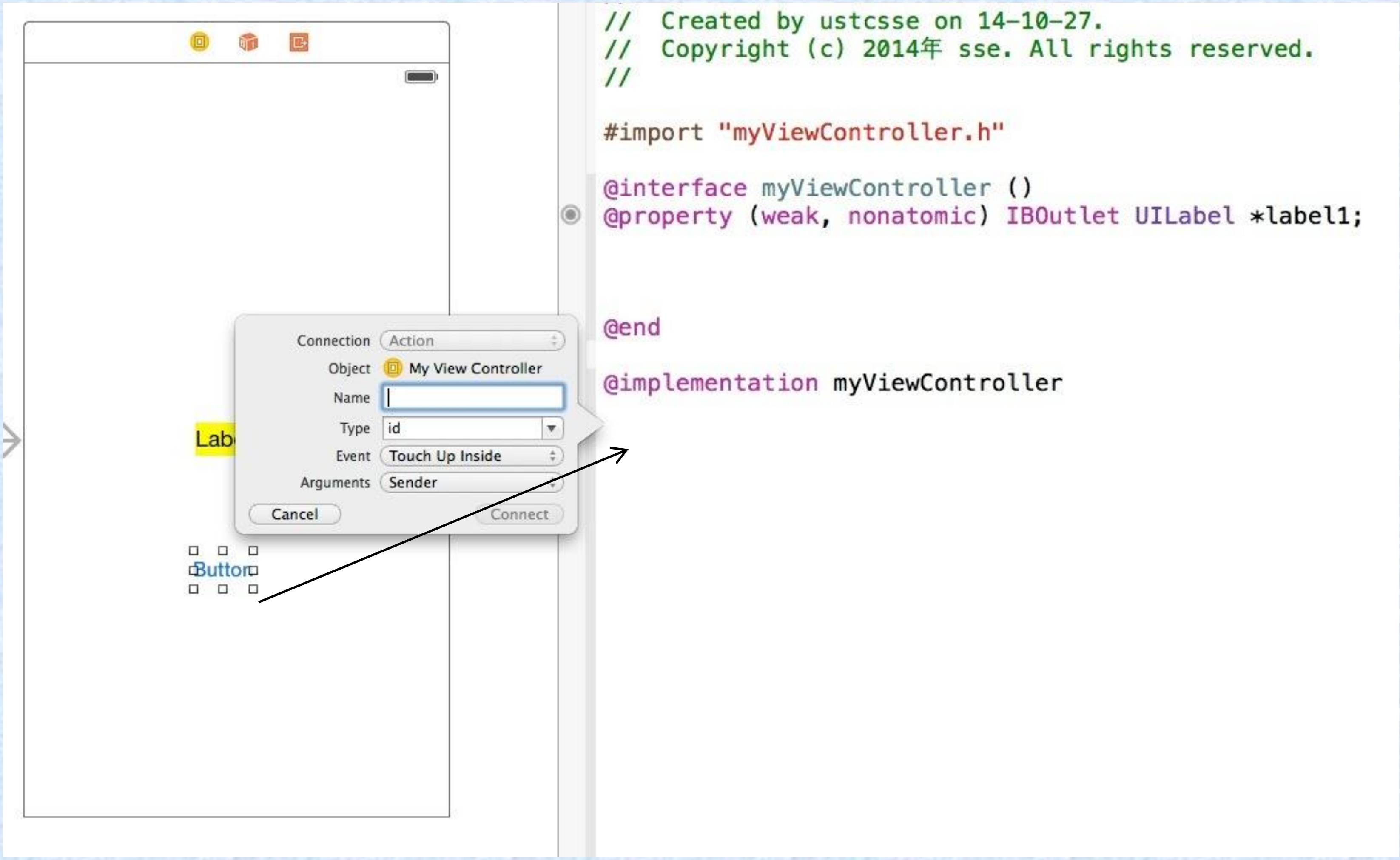
Connection: Outlet
Object: My View Controller
Name:
Type: UILabel
Storage: Weak
Cancel Connect

```
//
//  myViewController.m
//  DemoTest
//
//  Created by ustcsse on 14-10-27.
//  Copyright (c) 2014年 sse. All rights
//

#import "myViewController.h"

@interface myViewController ()


@end

@implementation myViewController
```

Ctrl + drag



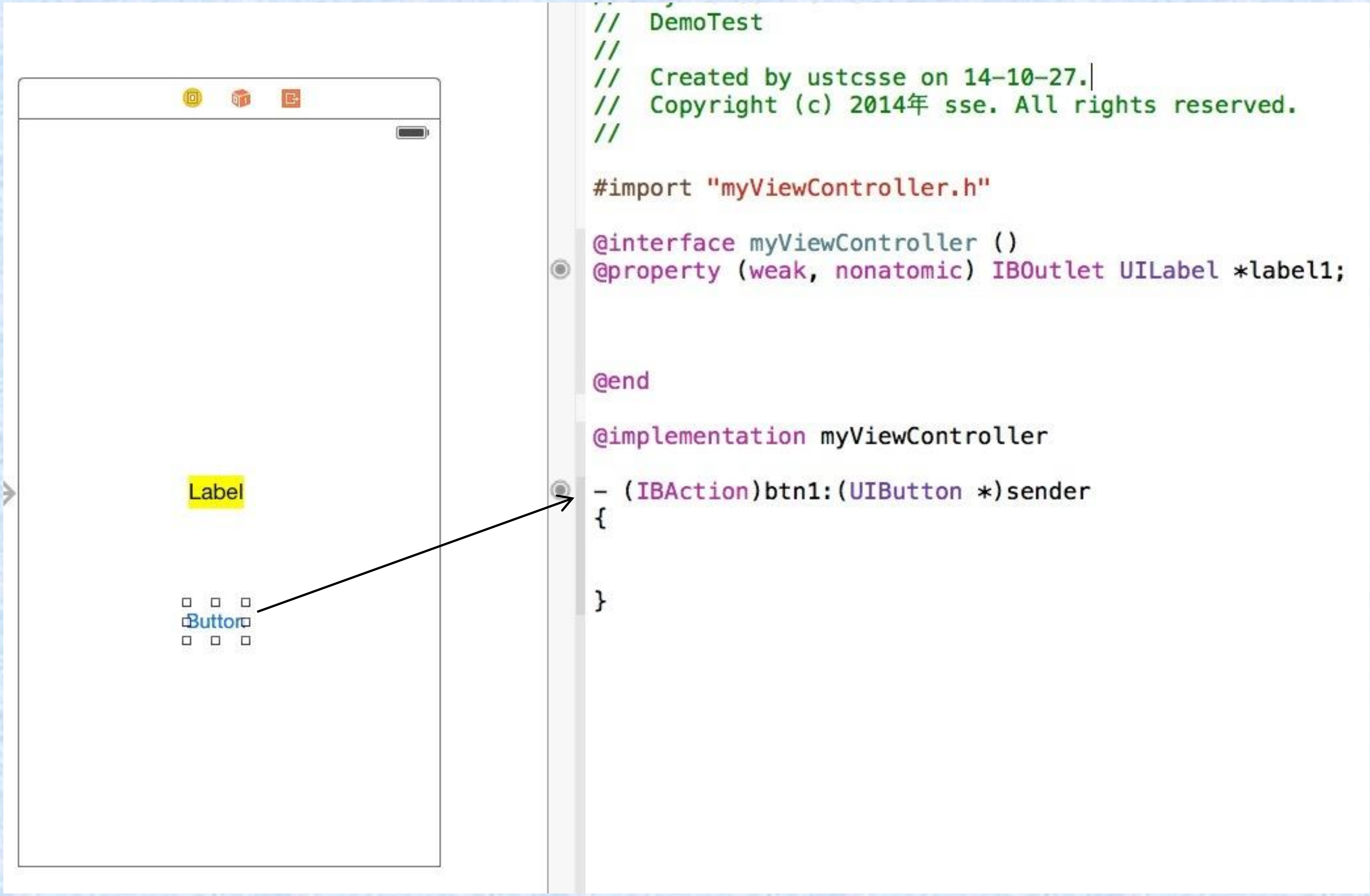
```
//  myViewController.m
//  DemoTest
//
//  Created by ustcsse on 14-10-27.
//  Copyright (c) 2014年 sse. All rights reserved.
//

#import "myViewController.h"

@interface myViewController ()
@property (weak, nonatomic) IBOutlet UILabel *label1;



@end

@implementation myViewController
```



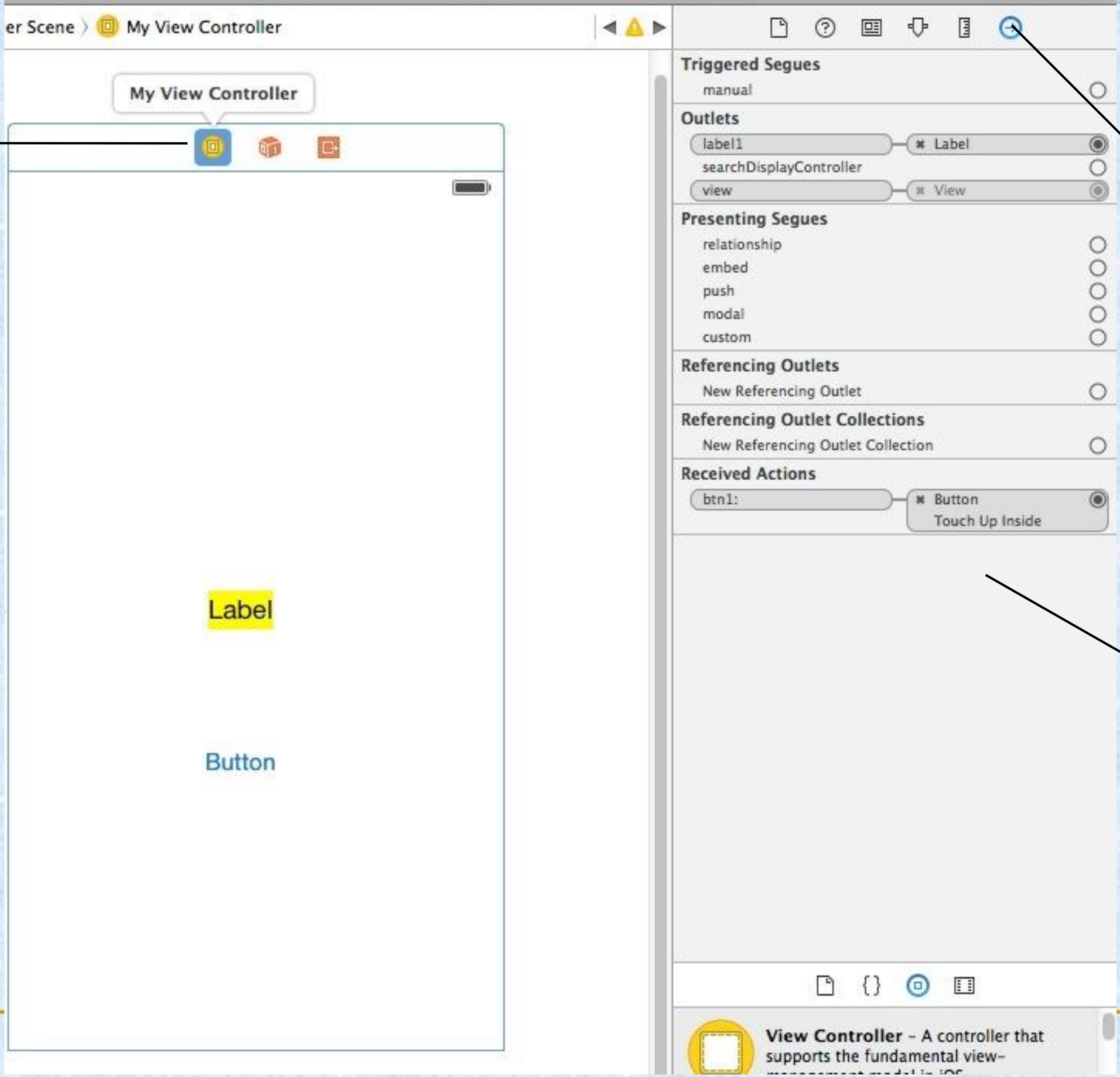
```
//  Created by ustcsse on 14-10-27.
//  Copyright (c) 2014年 sse. All rights reserved.
//

#import "myViewController.h"

@interface myViewController ()
@property (weak, nonatomic) IBOutlet UILabel *label1;



@end

@implementation myViewController
```

Connection: Action
Object: My View Controller
Name:
Type: id
Event: Touch Up Inside
Arguments: Sender

Ctrl + drag



```
//  DemoTest
//
//  Created by ustcsse on 14-10-27.
//  Copyright (c) 2014年 sse. All rights reserved.
//

#import "myViewController.h"

@interface myViewController ()
@property (weak, nonatomic) IBOutlet UILabel *label1;



@end

@implementation myViewController

- (IBAction)btn1:(UIButton *)sender
{


}
```



er Scene 〉 My View Controller

My View Controller

Label

Button

Triggered Segues
manual ○

Outlets
label1 — Label ●
searchDisplayController ○
view — View ●

Presenting Segues
relationship ○
embed ○
push ○
modal ○
custom ○

Referencing Outlets
New Referencing Outlet ○

Referencing Outlet Collections
New Referencing Outlet Collection ○

Received Actions
btn1: — Button Touch Up Inside ●

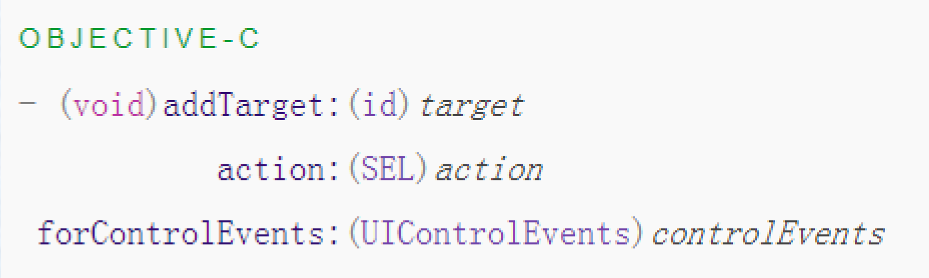
View Controller – A controller that supports the fundamental view-

1、选中视图控制器

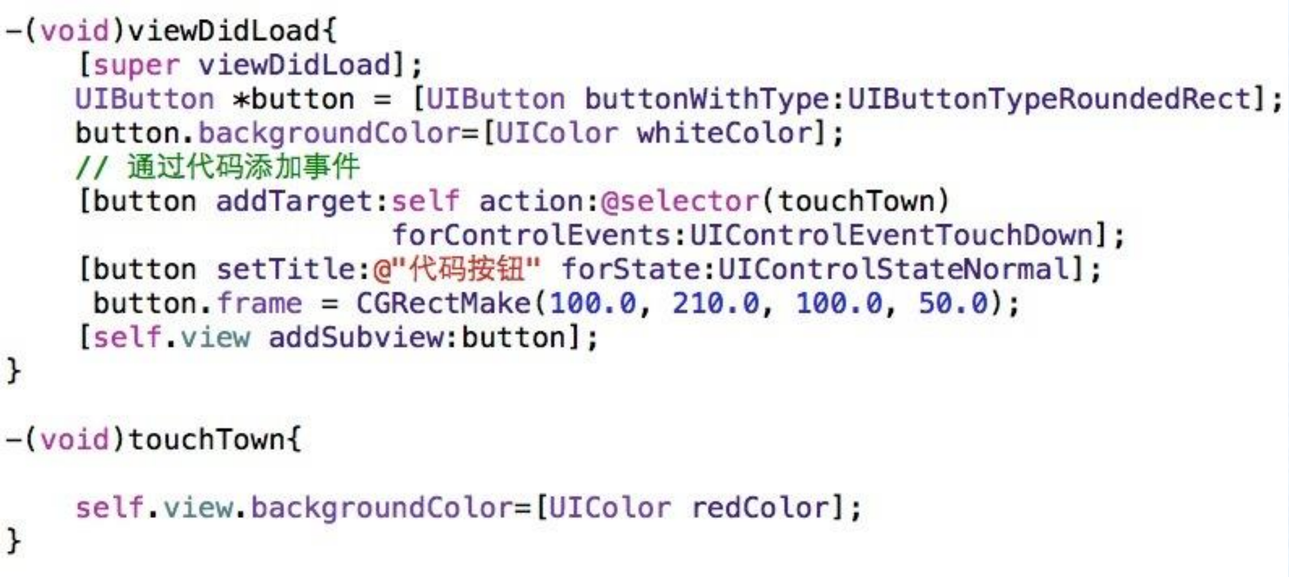
2、再选中连接检查器

查看连接情况

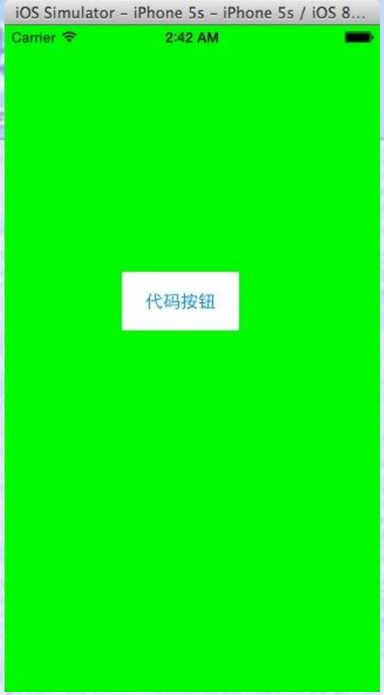
代码生成的UIControl类视图如何添加事件？

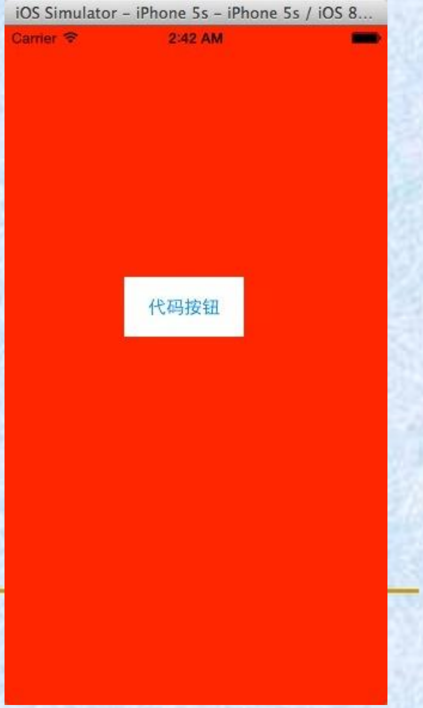


```
OBJECTIVE-C
- (void)addTarget:(id)target
           action:(SEL)action
 forControlEvents:(UIControlEvents)controlEvents
```

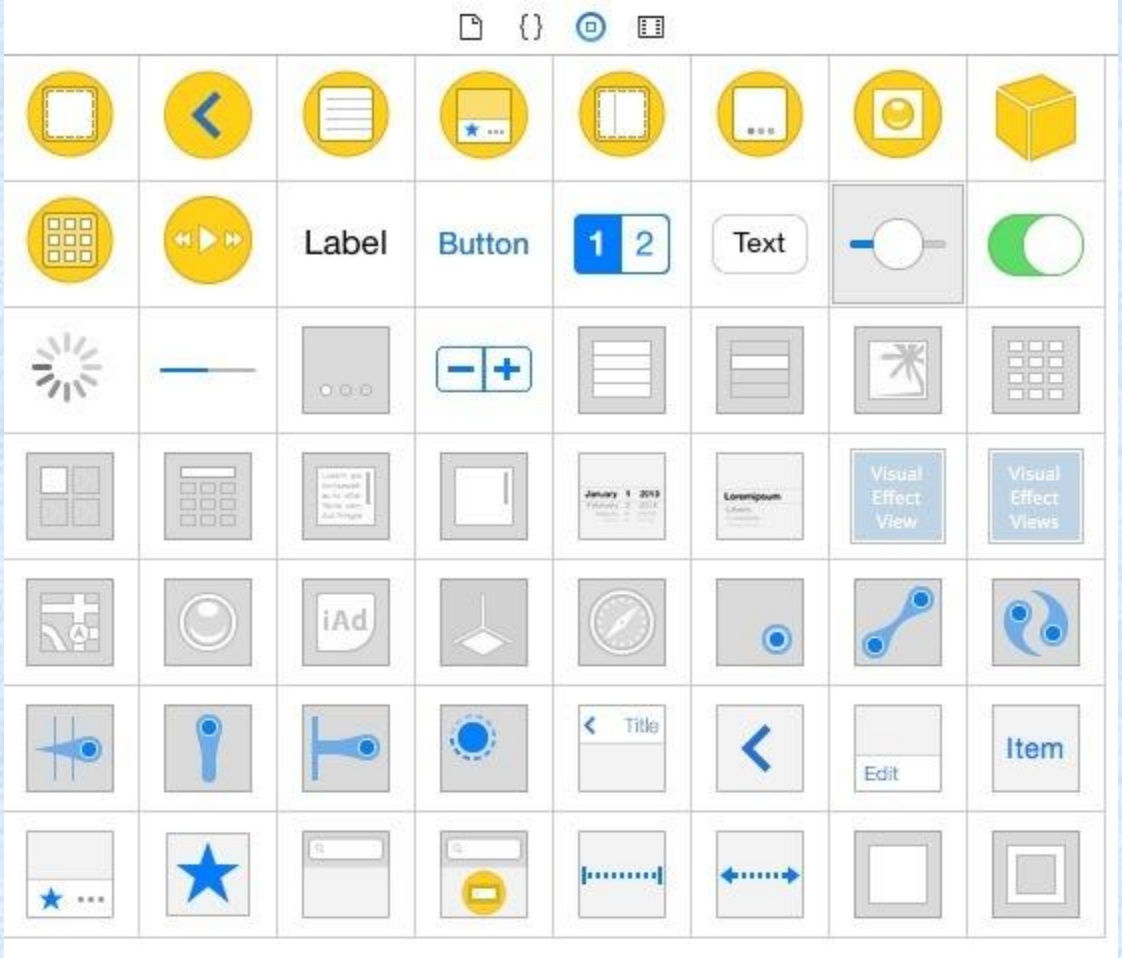


```
-(void)viewDidLoad{
    [super viewDidLoad];
    UIButton *button = [UIButton buttonWithType:UIButtonTypeRoundedRect];
    button.backgroundColor=[UIColor whiteColor];
    // 通过代码添加事件
    [button addTarget:self action:@selector(touchTown)
                      forControlEvents:UIControlEventTouchDown];
    [button setTitle:@"代码按钮" forState:UIControlStateNormal];
     button.frame = CGRectMake(100.0, 210.0, 100.0, 50.0);
    [self.view addSubview:button];
}

-(void)touchTown{

    self.view.backgroundColor=[UIColor redColor];
}
```





七、UIKIT框架中常用的视图


Label
Button
1
2
Text
January 1 2013
February 2 2014
Visual Effect View
Visual Effect Views
iAd
Title
Edit
Item