

iPhone Application Development

Lesson 9

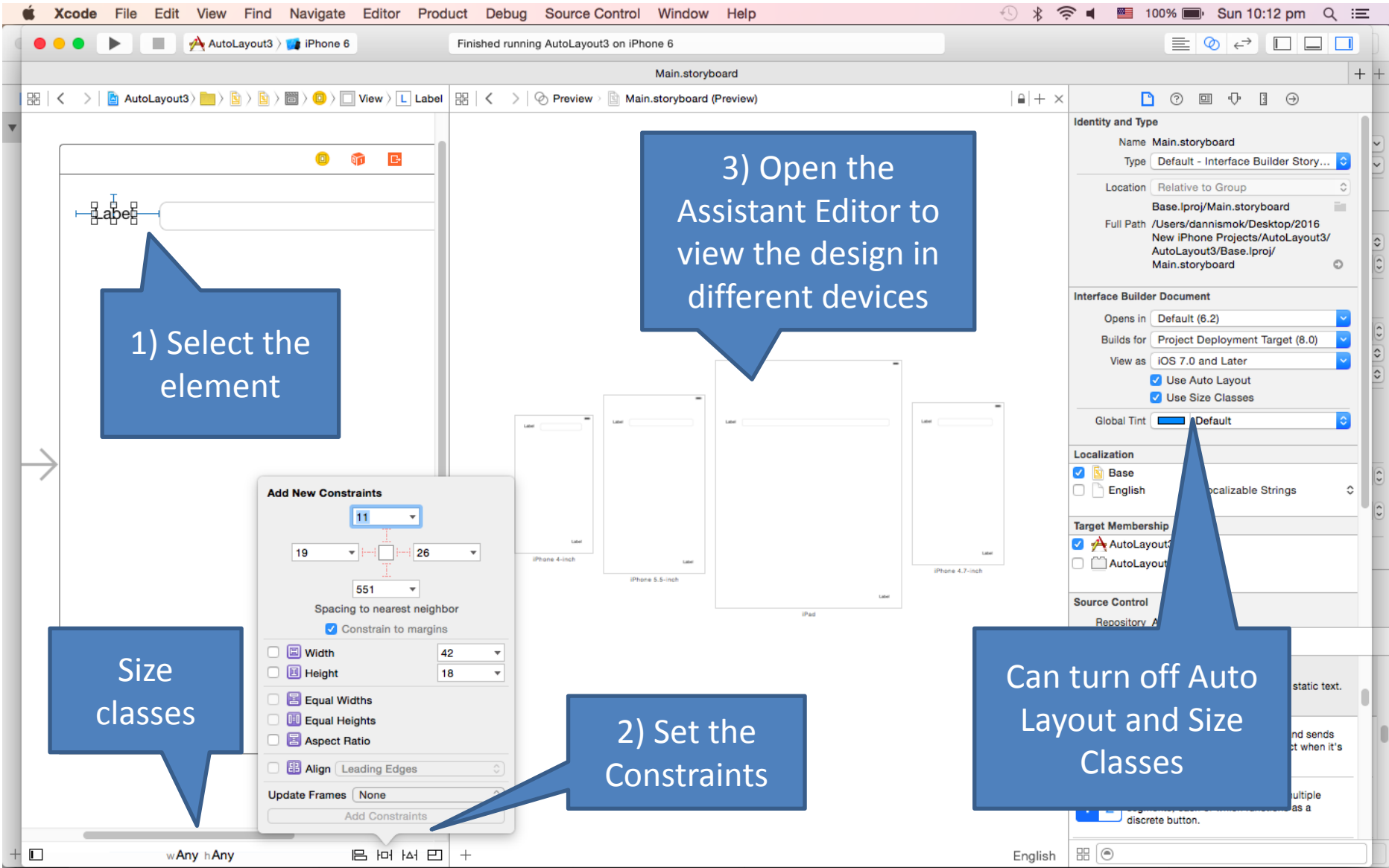
By Dannis Mok

Auto Layout

- Auto Layout helps to add the constraints onto the UI elements and make them to adapt to different devices' sizes and resolutions.
- Size Classes

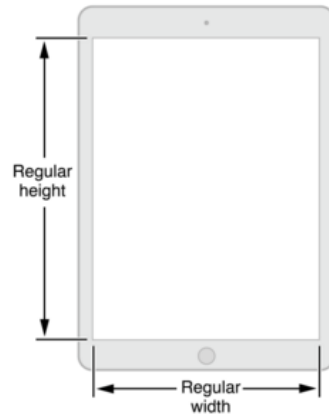
iOS defines two size classes: regular and compact. The *regular* size class is associated with expansive space and the *compact* size class is associated with constrained space.

Auto Layout

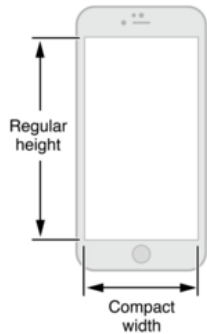
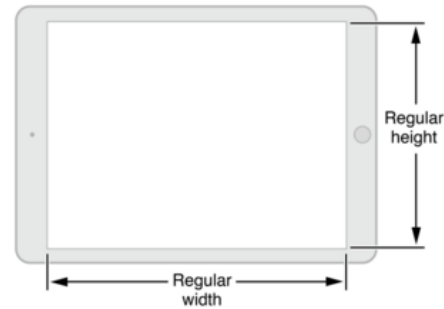


Size Classes

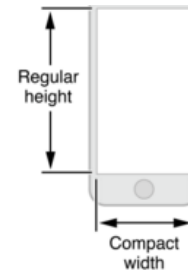
The size classes of iPad in portrait



The size classes of iPad in landscape

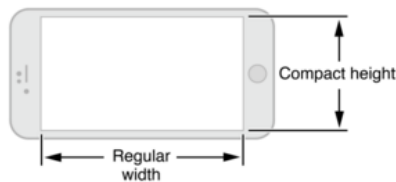


In portrait, iPhone 6 Plus uses the compact horizontal and regular vertical size classes.

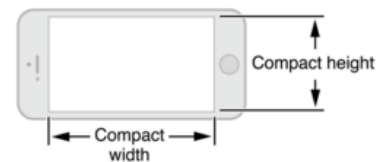


In portrait, iPhone 6, iPhone 5, and iPhone 4s use the compact horizontal and regular vertical size classes.

In landscape, iPhone 6 Plus uses the regular horizontal and compact vertical size classes.



In landscape, these devices use the compact size class in both the horizontal and vertical dimensions.



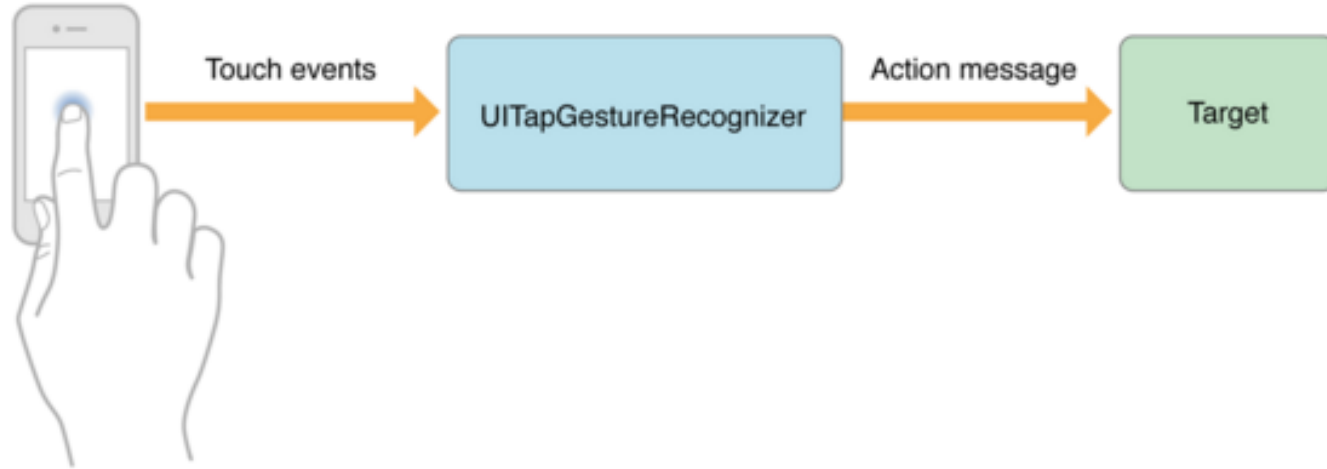
Gesture Processing

The UIKit framework provides predefined gesture recognizers that detect common gestures

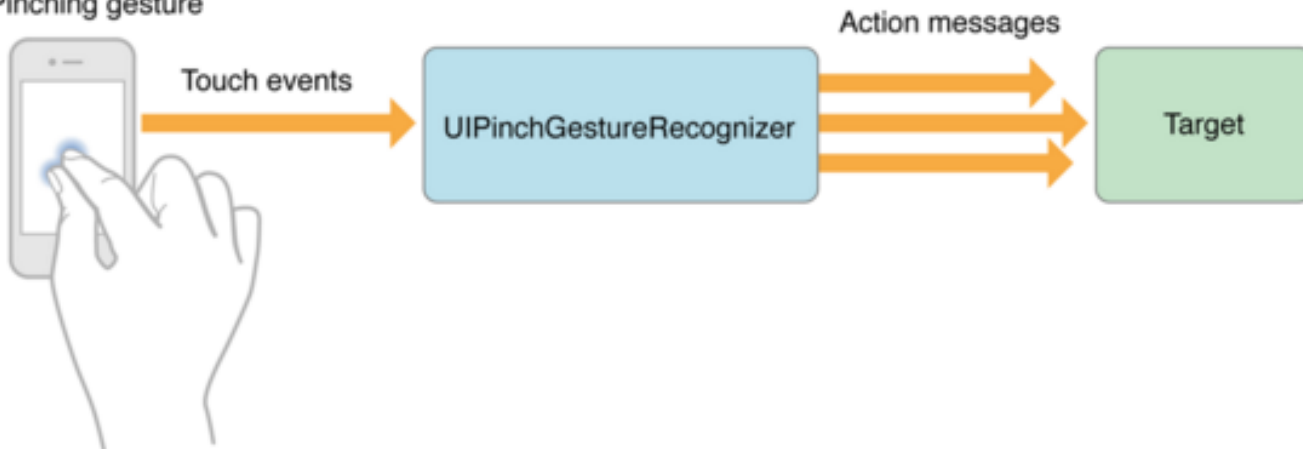
Gesture	UIKit class
Tapping (any number of taps)	<code>UITapGestureRecognizer</code>
Pinching in and out (for zooming a view)	<code>UIPinchGestureRecognizer</code>
Panning or dragging	<code>UIPanGestureRecognizer</code>
Swiping (in any direction)	<code>UISwipeGestureRecognizer</code>
Rotating (fingers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

Gesture Processing

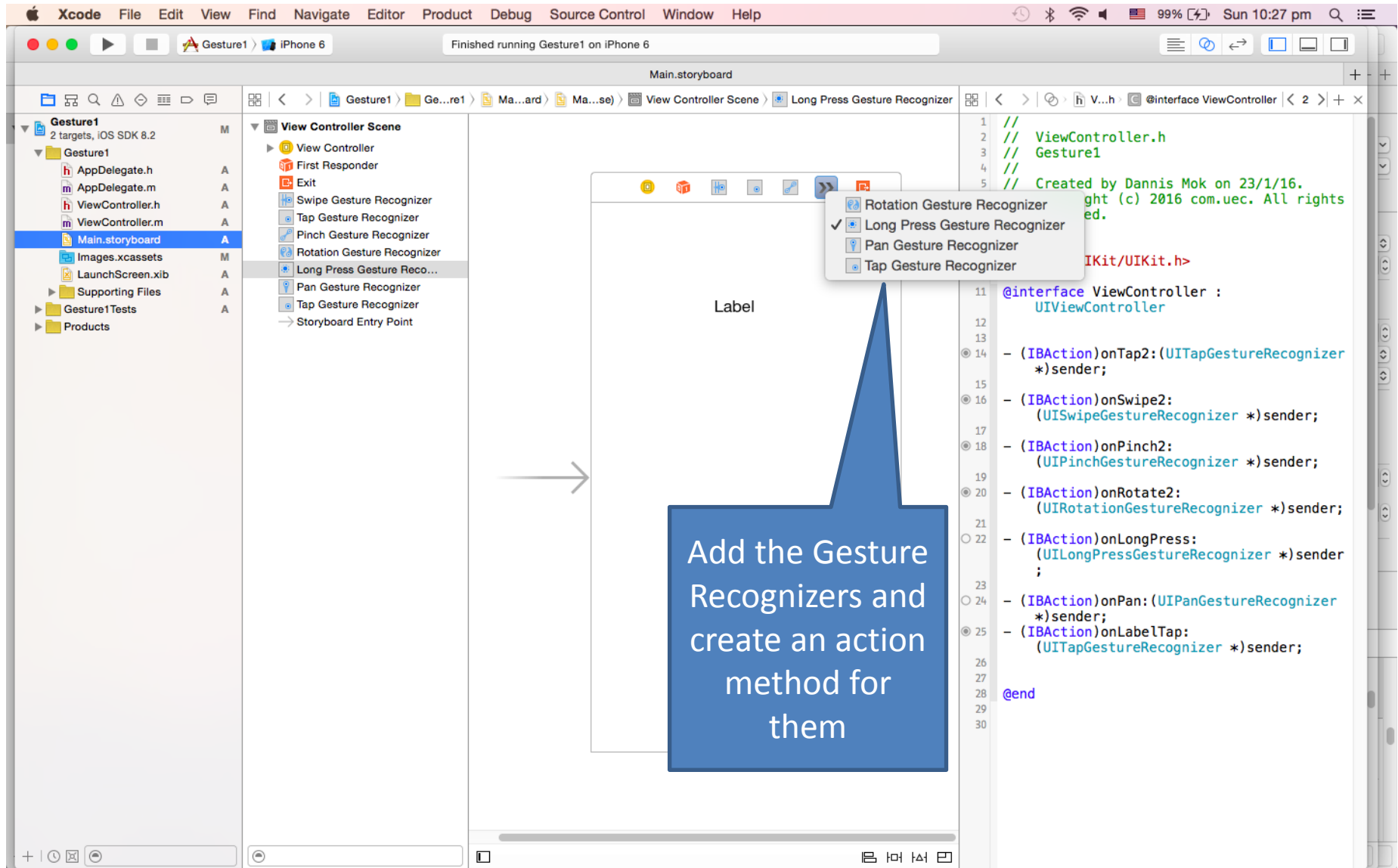
Tapping gesture



Pinching gesture



Gesture Processing



Gesture Processing

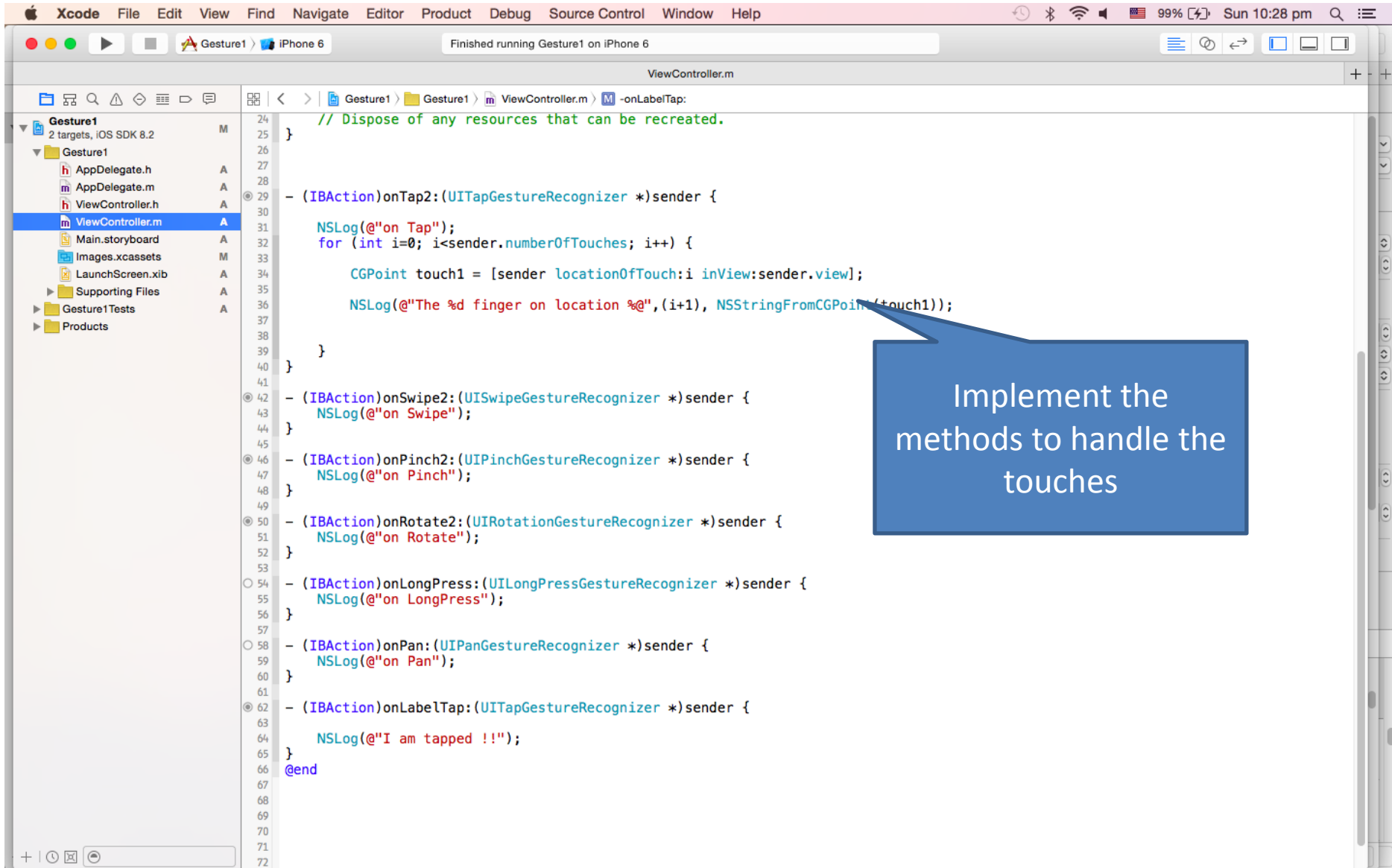


Photo Taking



Photo Taking

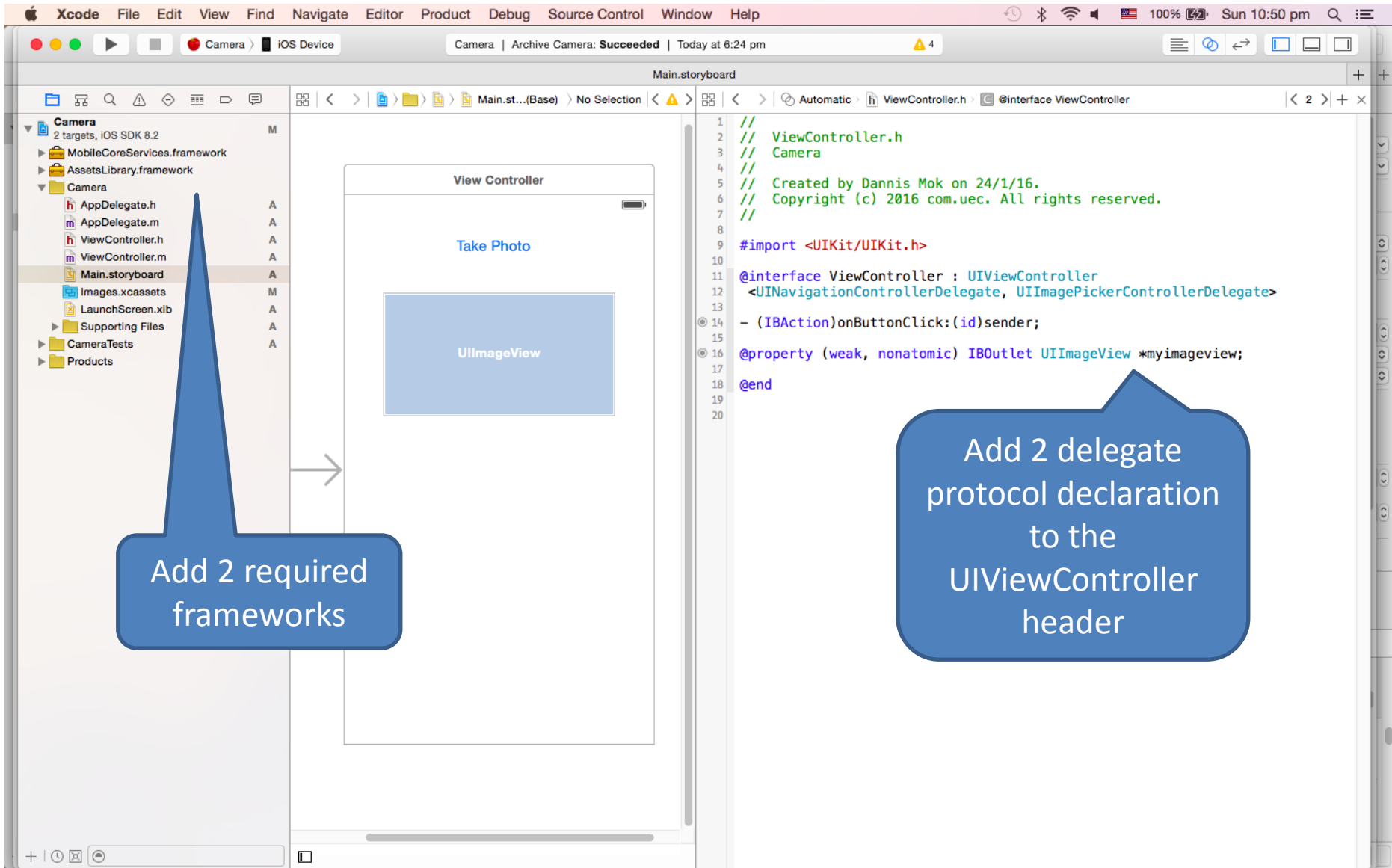


Photo Taking

The screenshot shows the Xcode IDE with the 'Camera' project selected. The left sidebar displays the project structure, including 'AppDelegate.h', 'AppDelegate.m', 'ViewController.h', 'ViewController.m', 'Main.storyboard', 'Images.xcassets', 'LaunchScreen.xib', 'Supporting Files', 'CameraTests', and 'Products'. The main editor window shows the 'ViewController.m' file with the following code:

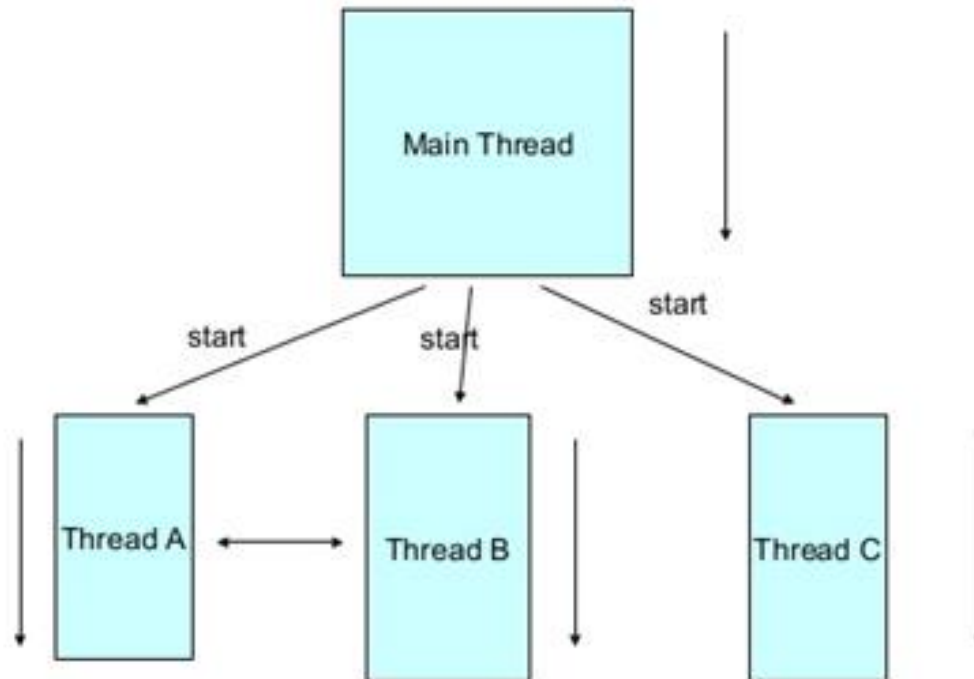
```
26 - (IBAction)onButtonClick:(id)sender {
27
28
29
30     if([UIImagePickerController isSourceTypeAvailable:UIImagePickerControllerSourceTypeCamera]) {
31
32         UIImagePickerController *imagePicker = [[UIImagePickerController alloc] init];
33
34         imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
35
36         imagePicker.delegate = self;
37
38         [self presentViewController:imagePicker animated:YES completion:nil];
39
40     } else {
41
42         UIImagePickerController *imagePicker = [[UIImagePickerController alloc] init];
43
44         imagePicker.sourceType = UIImagePickerControllerSourceTypeSavedPhotosAlbum;
45
46         imagePicker.delegate = self;
47
48         [self presentViewController:imagePicker animated:YES completion:nil];
49
50     }
51 }
52
53
54 -(void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary *)info {
55
56     UIImage *image = [info valueForKey:UIImagePickerControllerOriginalImage];
57
58     UIImageWriteToSavedPhotosAlbum(image, nil, nil, nil);
59
60     [self.myimageview initWithImage:image];
61
62     [self dismissViewControllerAnimated:YES completion:nil];
63
64 }
65
66 -(void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {
67
68
69     [self dismissViewControllerAnimated:YES completion:nil];
70
71 }
72
73 @end
74
```

Three blue callout boxes provide additional context:

- Use UIImagePickerController to take the photo**: Points to the code block starting at line 30, where the UIImagePickerController is initialized and presented.
- Method call after accepting the photo**: Points to the code block starting at line 54, where the didFinishPickingMediaWithInfo: method is called to handle the selected image.
- Method call after canceling**: Points to the code block starting at line 66, where the didFinishCancel: method is called to handle the cancellation.

Multi-Threading

A Multithreaded Program



Threads may switch or exchange data/results

Multi-Threading

```
-(void)run1:(NSTimer *)timer {  
    static int n = 0;  
    if(n == 10) {  
        [timer invalidate];  
    }  
    NSLog(@"%d",n++);  
}  
  
-(void)thread1 {  
    for(int i=0;i<5;i++) {  
        NSLog(@"%d",i);  
    }  
}  
  
-(void)thread2 {  
    for(int i=100;i<110;i++) {  
        NSLog(@"%d",i);  
    }  
}
```

NSTimer has a built-in thread. It will be scheduled to run this method and display the n value till it reaches 10

This thread method will print 0 to 5 to the log file

This thread method will print 100 to 110 to the log file

Multi-Threading

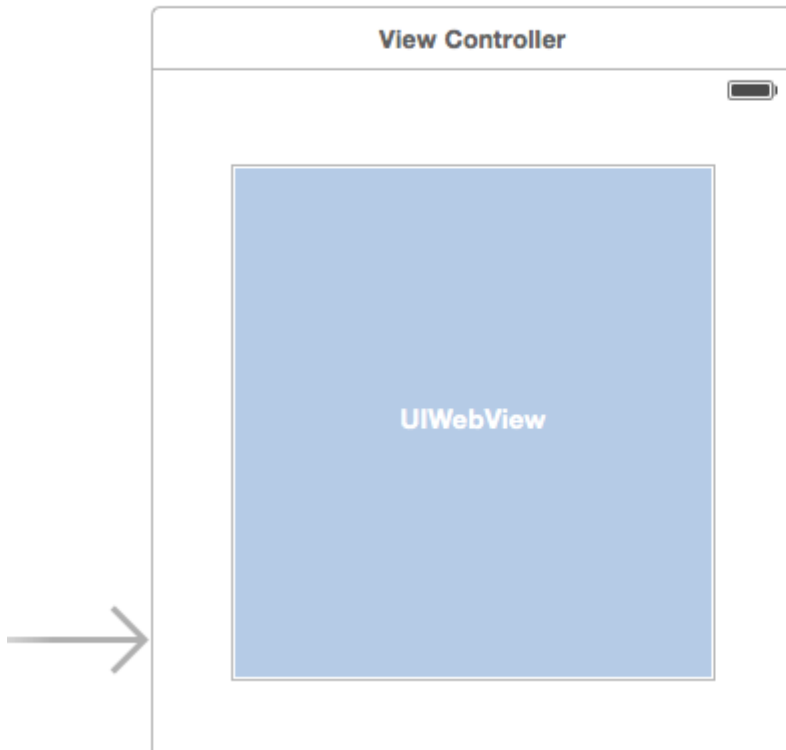
```
- (void)viewDidLoad {  
    [super viewDidLoad];  
    // Do any additional setup after loading the view, typically from a nib.  
  
    NSLog(@"A");  
  
    [NSThread sleepForTimeInterval:3.0];  
  
    NSLog(@"B");  
  
    [NSTimer scheduledTimerWithTimeInterval:1.0 target:self selector:@selector(run1:) userInfo:  
        nil repeats:YES];  
  
    NSThread *t1 = [[NSThread alloc] initWithTarget:self selector:@selector(thread1) object:nil];  
    NSThread *t2 = [[NSThread alloc] initWithTarget:self selector:@selector(thread2) object:nil];  
  
    [t1 start];  
    [t2 start];  
}
```

1) Make the main thread sleep for 3 seconds

2) Use NSTimer to run the method

3) Create 2 threads to run the 2 different methods

Web View



```
1 //  
2 // ViewController.h  
3 // Network1  
4 //  
5 // Created by Dannis Mok on 24/1/16.  
6 // Copyright (c) 2016 com.uec. All rights  
7 // reserved.  
8 //  
9 #import <UIKit/UIKit.h>  
10  
11 @interface ViewController : UIViewController  
12  
13  
14 @property (weak, nonatomic) IBOutlet  
15     UIWebView *myWebView;  
16  
17  
18 @end  
19  
20
```

Web View

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Print the HTML code in the WebView

    NSString *html = @"<h1>iPhone Programming</h1><h2>from Apple</h2><img src='apple.jpg' />";

    [myWebView loadHTMLString:html baseURL:nil];

    // Print the file content in the WebView

    NSURL *url = [NSURL URLWithString:@"http://localhost:8888/hello.html"];
    NSURLRequest *request = [NSURLRequest requestWithURL:url];
    [myWebView loadRequest:request];

    // Create an Image from the local file and then display it

    UIImageView *newimage = [UIImageView new];

    [newimage setImage:[UIImage imageNamed:@"apple.jpg"]];

    CGPoint point = CGPointMake(0,0);
    [newimage setCenter:point];

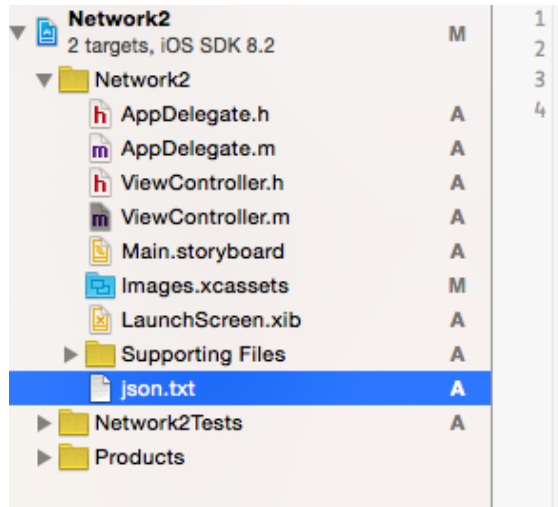
    [self.view addSubview:newimage];
}
```

Use the Web View
to display HTML

Use the Web View
to show web page
content

Create and Image and
show it dynamically

JSON & Network



```
1 [{"id": "1", "name": "Peter", "phone": "99434939", "email": "flower1.jpeg"},  
2   {"id": "2", "name": "Mary", "phone": "29349349", "email": "flower2.jpeg"},  
3   {"id": "3", "name": "Sam", "phone": "9239432", "email": "flower3.jpeg"},  
4   {"id": "4", "name": "David", "phone": "39493943", "email": "flower4.jpeg"}]
```

Can create a local JSON
format file



JSON & Network

```
- (void)viewDidLoad {
    [super viewDidLoad];

    // Read data from JSON and then read the image from the network

    NSString *path = [[NSBundle mainBundle]pathForResource:@"json" ofType:@"txt"];
    NSData *data = [NSData dataWithContentsOfFile:path];

    NSArray *jsonarray = [NSJSONSerialization JSONObjectWithData:data options:
        NSJSONReadingMutableContainers error:nil];

    for(NSDictionary *p in jsonarray) {

        NSString *sid = [p objectForKey:@"id"];
        NSString *name = [p objectForKey:@"name"];
        NSString *tel = [p objectForKey:@"phone"];
        NSString *photo = [p objectForKey:@"email"];

        NSLog(@"%@ %@ %@ %@", sid, name, tel, photo);

        NSMutableString *path = [NSMutableString stringWithString:@"http://localhost:8888/"];
        path = [[path stringByAppendingString:photo]mutableCopy];
        NSLog(@"%@", path);

        NSURL *url = [NSURL URLWithString:path];
        NSURLRequest *request = [NSURLRequest requestWithURL:url];

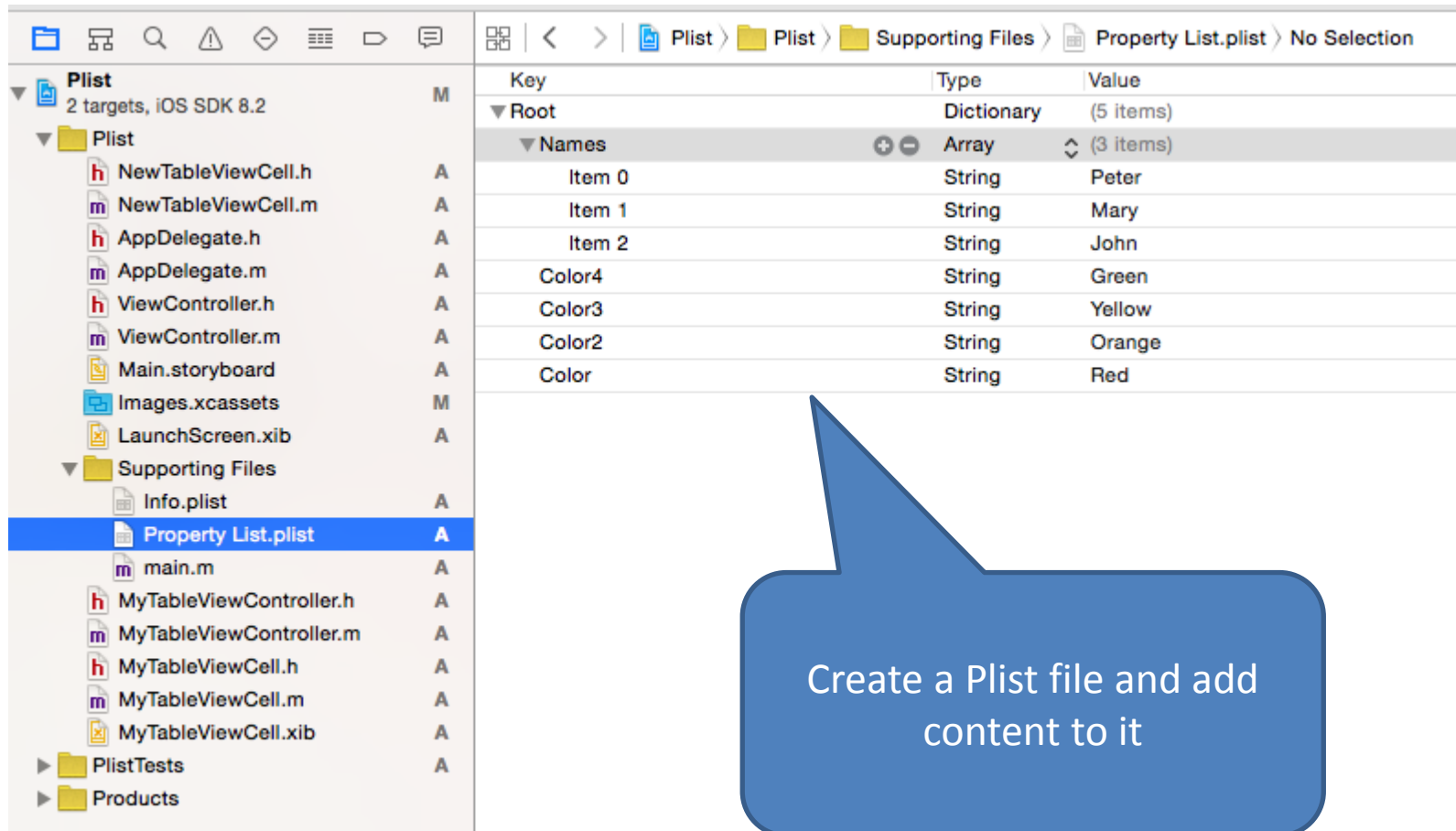
        [NSURLConnection sendAsynchronousRequest:request queue:[NSOperationQueue mainQueue]
            completionHandler:^(NSURLResponse *response, NSData *data, NSError *connectionError)
            {
                UIImage *image = [UIImage imageWithData:data];
                NSLog(@"%@", response, image);
            }];
    }
}
```

Read the file
content into
the NSData
object

Read the data and the
use the image path to
download the image

Use asynchronous
thread to download the
image

Plist file



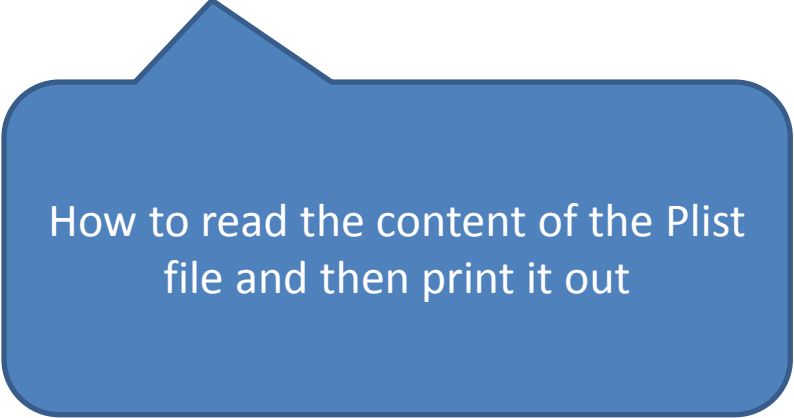
The screenshot shows the Xcode interface. On the left, the Project Navigator displays a project named 'Plist' with two targets, iOS SDK 8.2. The 'Supporting Files' folder is expanded, showing 'Property List.plist' selected. On the right, the Properties pane shows the contents of 'Property List.plist'.

Key	Type	Value
Root	Dictionary	(5 items)
Names	Array	(3 items)
Item 0	String	Peter
Item 1	String	Mary
Item 2	String	John
Color4	String	Green
Color3	String	Yellow
Color2	String	Orange
Color	String	Red

Create a Plist file and add content to it

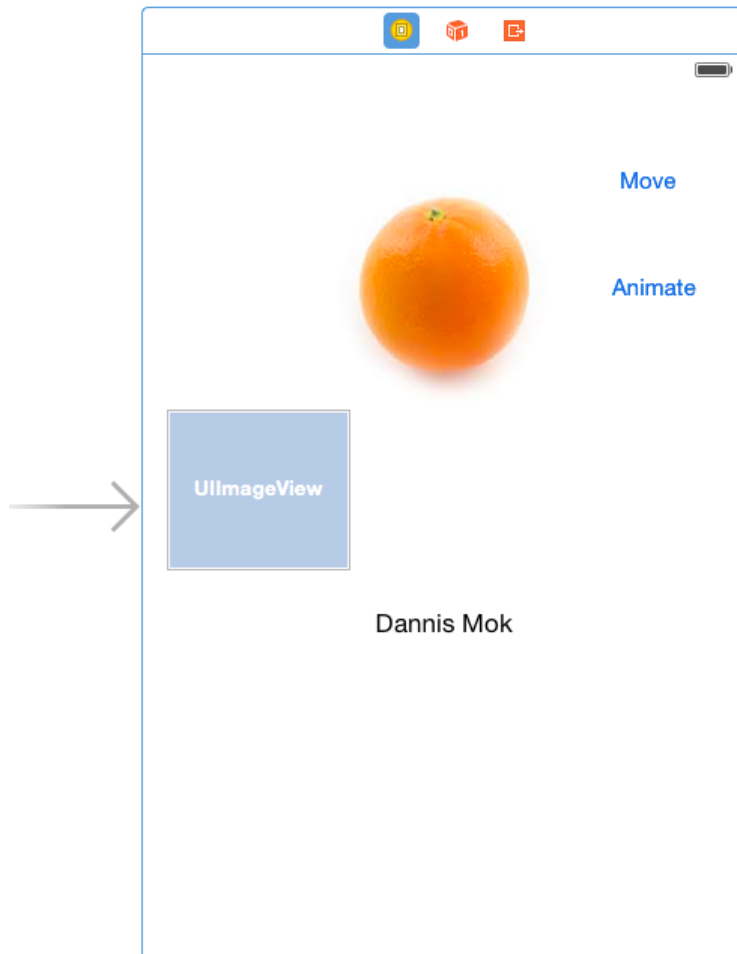
Plist file

```
- (void)viewDidLoad {  
    [super viewDidLoad];  
    // Do any additional setup after loading the view, typically from a nib.  
  
    NSString *plistpath = [[NSBundle mainBundle]pathForResource:@"Property List" ofType:@"plist"  
        ];  
  
    NSMutableDictionary *plist = [NSMutableDictionary dictionaryWithContentsOfFile:plistpath];  
  
    NSString *color = [plist objectForKey:@"Color"];  
  
    NSLog(@"The color is %@",color);  
  
}
```



How to read the content of the Plist
file and then print it out

UIView Properties



```
1 //
2 // ViewController.h
3 // UIViewTest
4 //
5 // Created by Dannis Mok on 20/1/16.
6 // Copyright (c) 2016 com.uec. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10
11 @interface ViewController : UIViewController
12
13
14 @property (weak, nonatomic) IBOutlet UIImageView *myOrange;
15
16
17 @property (weak, nonatomic) IBOutlet UILabel *myName;
18
19 - (IBAction)onClick:(id)sender;
20 @property (weak, nonatomic) IBOutlet UIImageView *myOrange2;
21 ;
22 - (IBAction)onClick2:(id)sender;
23
24 @end
25
```

Can use the UIView
animation properties

UIView Properties

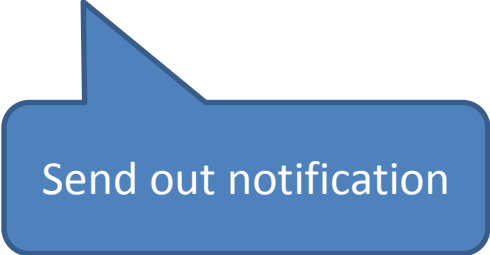
```
- (IBAction)onClick:(id)sender {  
    float x1 = myName.center.x + 10;  
    float y1 = myName.center.y - 10;  
  
    CGPoint newPoint = CGPointMake(x1,y1);  
    myName.center = newPoint;  
  
    NSLog(@"%f",myName.center.x);  
    NSLog(@"%f",myName.center.y);  
  
    [UIView beginAnimations:nil context:nil];  
    [UIView setAnimationRepeatCount:2];  
    [UIView setAnimationDuration:2];  
  
    myOrange.bounds = CGRectMake(0, 0, 100, 100);  
    myOrange.backgroundColor = [UIColor redColor];  
    myName.backgroundColor = [UIColor blueColor];  
    myName.textColor = [UIColor yellowColor];  
  
    myOrange.alpha = 0.5;  
    myOrange.transform = CGAffineTransformMakeRotation(30);  
    myName.transform = CGAffineTransformMakeScale(1.5, 1.5);  
    myOrange.transform = CGAffineTransformMakeTranslation(20, 0);  
  
    [UIView commitAnimations];  
  
    UIImage *orange = [UIImage imageNamed:@"orange.jpg"];  
    UIImageView *newImageView = [[UIImageView alloc] initWithImage:orange];  
  
    newImageView.frame = CGRectMake(0, 0, 100, 100);  
    newImageView.center = CGPointMake(50,50);  
}
```

Move the UI element
by moving its center

Wrap the property
changes in the animation
block

Notification

```
- (IBAction)onPinch:(UIPinchGestureRecognizer *)sender {  
    if(sender.state == UIGestureRecognizerStateBegan) {  
  
        NSLog(@"Start");  
    } else if(sender.state == UIGestureRecognizerStateChanged) {  
        NSLog(@"Changed");  
    } else if(sender.state == UIGestureRecognizerStateEnded) {  
        NSLog(@"Ended");  
  
        UILocalNotification *ln = [UILocalNotification new];  
  
        ln.alertBody = @"Message I send";  
        ln.applicationIconBadgeNumber = 2;  
        ln.fireDate = [NSDate dateWithTimeIntervalSinceNow:5.0];  
  
        [[UIApplication sharedApplication] scheduleLocalNotification:ln];  
    }  
}
```



Send out notification

Notification

```
#import "AppDelegate.h"

@interface AppDelegate ()

@end

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.

    if ([[UIApplication instancesRespondToSelector:@selector(registerUserNotificationSettings:)])
        {
            [application registerUserNotificationSettings:[UIUserNotificationSettings
                settingsForTypes:UIUserNotificationTypeAlert|UIUserNotificationTypeBadge|
                UIUserNotificationTypeSound categories:nil]];
        }
    return YES;
}
```



Add this to the AppDelegate's
applicationDidFinishLaunchingWith
Options to grant the permission to
accept notification (iOS 8 later)