iPhone Boot Camp NYC

Introduction to iOS Development

Introduction Slide option 1 of 4.

Greetings.
Welcome to _____ class.
My name is _____.
Today, we will introduce you _____.
This class will discuss _____.

Instructor Introduction

- James Eberhardt
- james@echomobile.ca
- 647.402.9051

This slide is used as a placeholder for the instructor introduction.

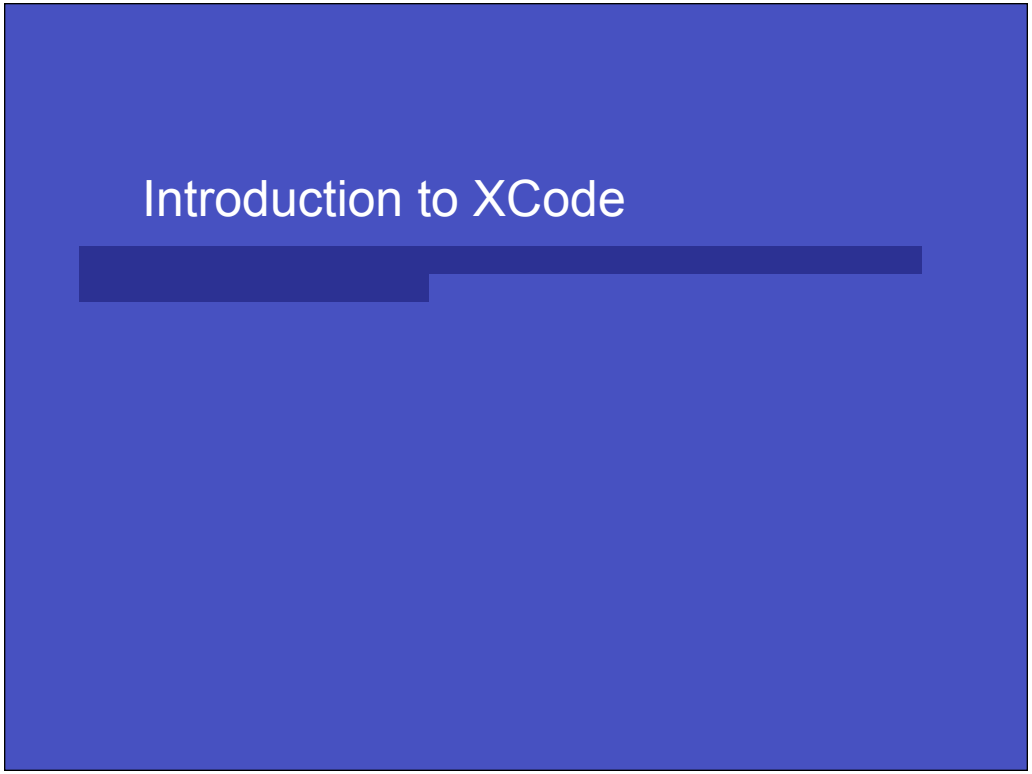
Student Introduction

- Name
- Experience
- Training and Certifications
- Expectations

This slide is used as a placeholder for student introductions.

Use a flipchart, white board or chalk board to write down the students' expectations of the class.

Questions



When Module Titles on the Module Divider pages wrap to two lines and left justify.

Module Objectives

Upon completion of this module, you will be able to:

- create projects using XCodes templates
- understand the files that make up an XCode project
- understand the components that make up XCode
- run an app in the simulator
- navigate the XCode user interface

Page numeration resumes on the Module Objectives page.

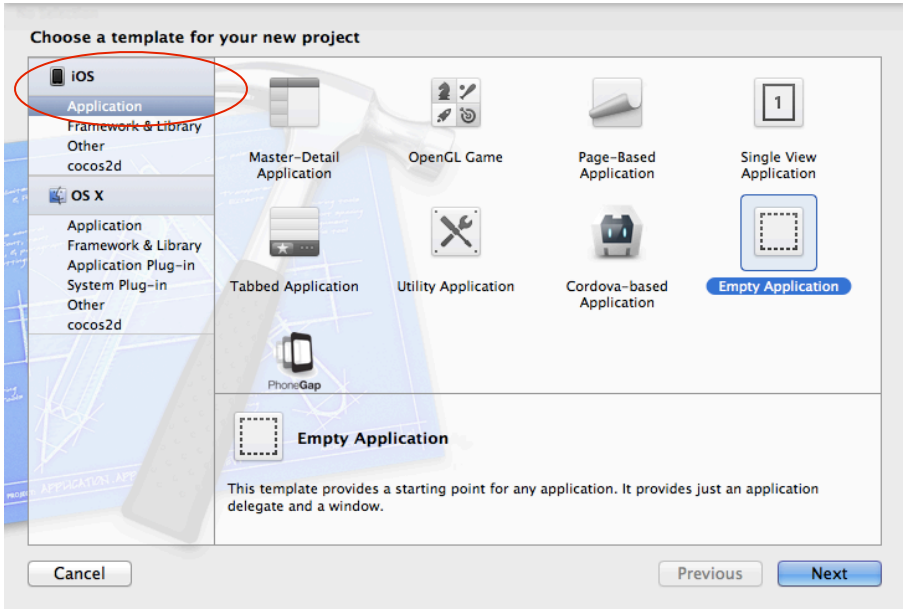
Module Content

The main components of XCode are:

- XCode IDE
- XIB and Storyboard Editor (Interface Builder)
- iOS Simulator
- Instruments

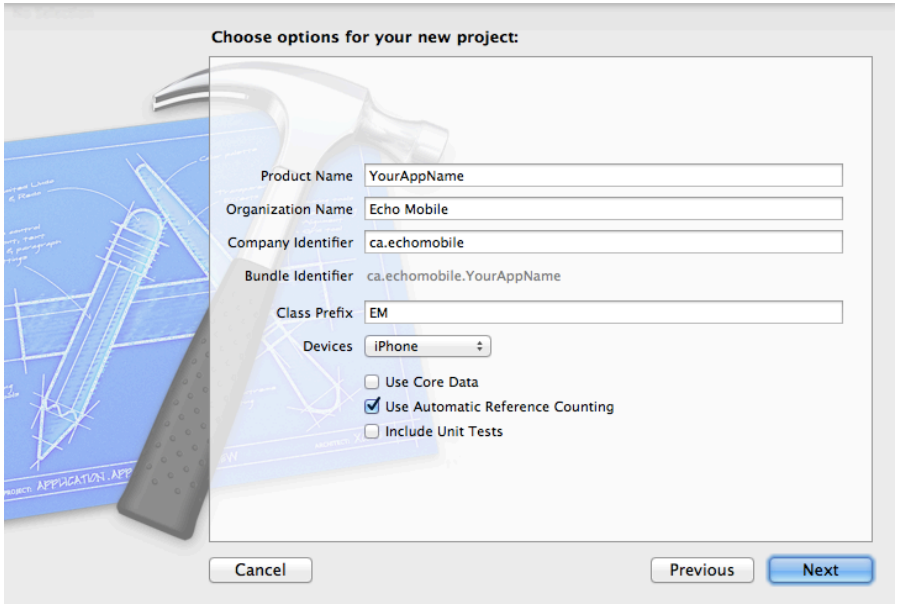
This slide can be a combination of bullets, graphs, animations, etc...

XCode: Project Templates



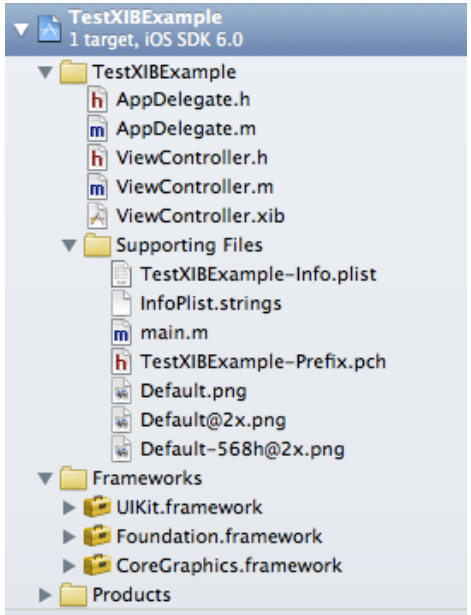
This slide can be a combination of bullets, graphs, animations, etc...

XCode: Project Templates



This slide can be a combination of bullets, graphs, animations, etc...

XCode: Project Files



XCode: Product Menu

Product Menu

- Run

  opens app in simulator or connected device

- Build

  compiles the code without running the app, useful for checking for errors and warnings

- Clean

  removes any pre-compiled code or bundled assets;

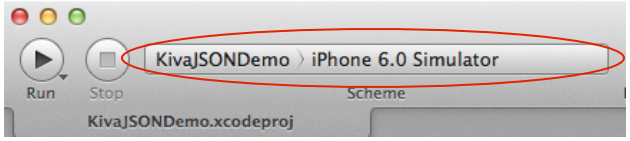
  useful if there seems to be “old” assets or code in the app running on the simulator

  NOTE: The following folder is automatically generated and can be deleted to do a ‘full’ clean.

  /Users/*username*/Library/Developer/Xcode/DerivedData/*ProjectName*-id/

This slide can be a combination of bullets, graphs, animations, etc... The module summary should highlight the main points (module objectives) that the student was expected to takeaway from this module.

XCode: iOS Simulator



- rotate phone - useful for testing the various orientations of the phone
- Change the device hardware and os version
- Simulate Memory Warning
- Save Screen Shot
- Reset Content and Settings - removes all user files from the phone. Those files can be found here:

  ~/Library/Application Support/iPhone Simulator/

This slide can be a combination of bullets, graphs, animations, etc...

Instruments

Instruments

- excellent tool for testing and debugging applications
- we'll look at instruments more another day

Some uses of Instruments

Leaks - Useful in tracking down objects that never released.

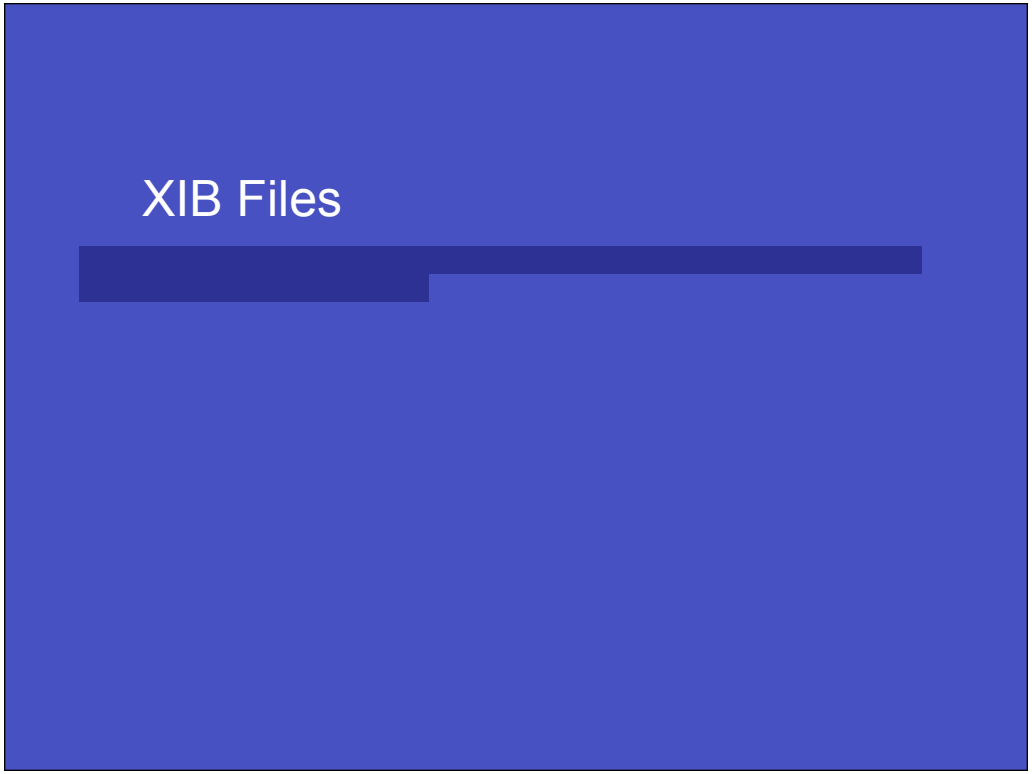
Allocations - Allows you to see the amount of memory being used by the app.

Zombies - Allows you to see if you are referencing objects that have already been released.

Creating a new project

Task:

Create a new project in XCode and run it in the simulator.



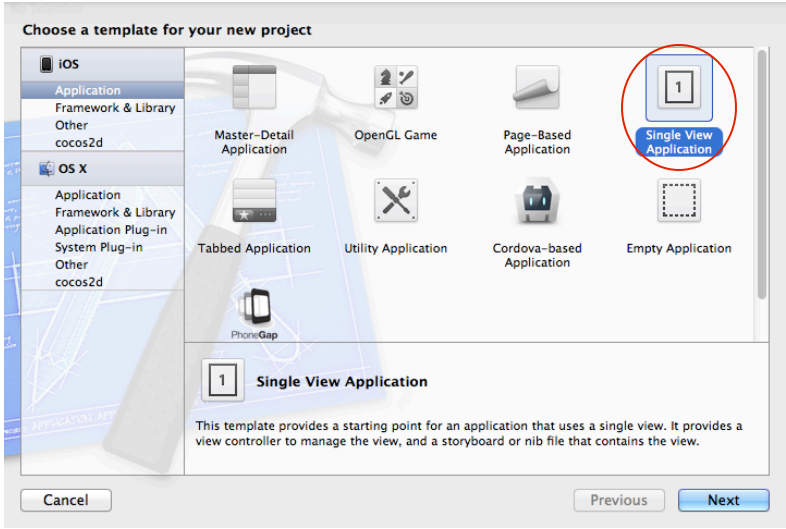
When Module Titles on the Module Divider pages wrap to two lines and left justify.

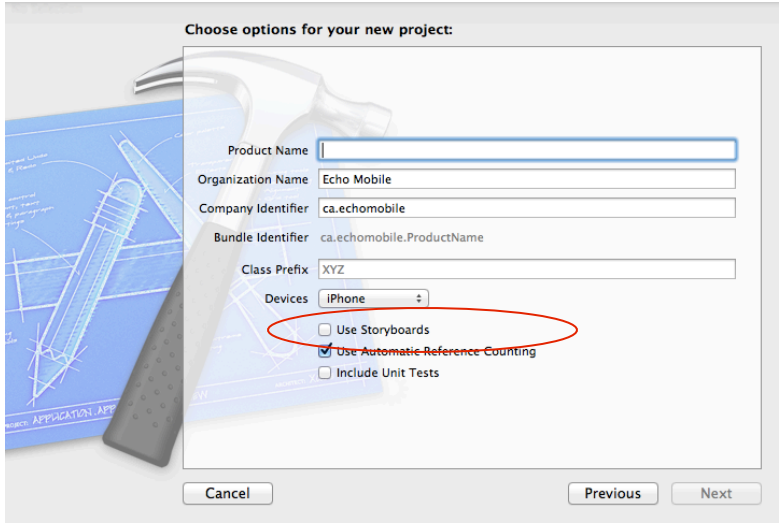
XIB Files: Module Objectives

Upon completion of this module, you will be able to:

- Add a XIB file to the project
- Add UIKit objects to XIB files
- Implement outlets and actions on XIB files
- Show ViewControllers that use XIB files

Page numeration resumes on the Module Objectives page.


Choose a template for your new project
iOS
Application
Framework & Library
Other
cocos2d
OS X
Application
Framework & Library
Application Plug-in
System Plug-in
Other
cocos2d
Master-Detail Application
OpenGL Game
Page-Based Application
Single View Application
Tabbed Application
Utility Application
Cordova-based Application
Empty Application
PhoneGap
Single View Application
This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.
Cancel
Previous
Next


Choose options for your new project:
Product Name
Organization Name
Echo Mobile
Company Identifier
ca.echomobile
Bundle Identifier
ca.echomobile.ProductName
Class Prefix
XYZ
Devices
iPhone
Use Storyboards
Use Automatic Reference Counting
Include Unit Tests
Cancel
Previous
Next

XIB Files: Task

- Create a new project using the “Single View Application” template
- Be certain **NOT** to use Storyboard.
- Select the “MainStoryboard.storyboard” file to display the XIB Editor
- Drag some UI objects onto the screen
- Run the app and see your UI objects in the simulator

NOTE: Your app doesn’t have any functionality because no code has been added to the project.

XIB Files: Actions and Outlets

Actions:

Connecting the event from a UI element on the stage to a selector in the class.

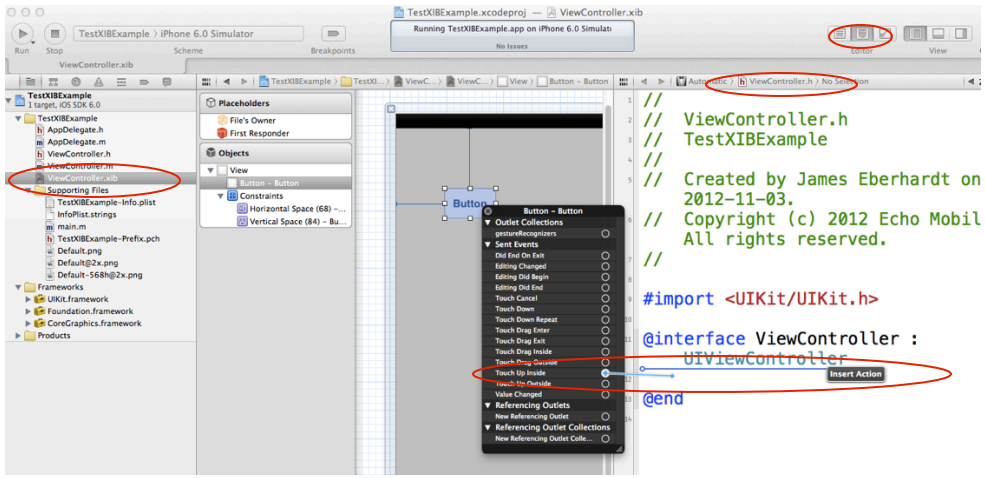
ie: the “touchUpInside” event from a UIButton connects to a selector.

Outlets:

Connecting the instance of a UI element on the stage to a property in the class, allowing the instance to be manipulated via the code by accessing the property name.

ie: a property named *myLabel* could be connected to a label on the screen and *myLabel.text* would be used to set the text of the label.

XIB Files: Actions and Outlets



XIB Files: Tutorial

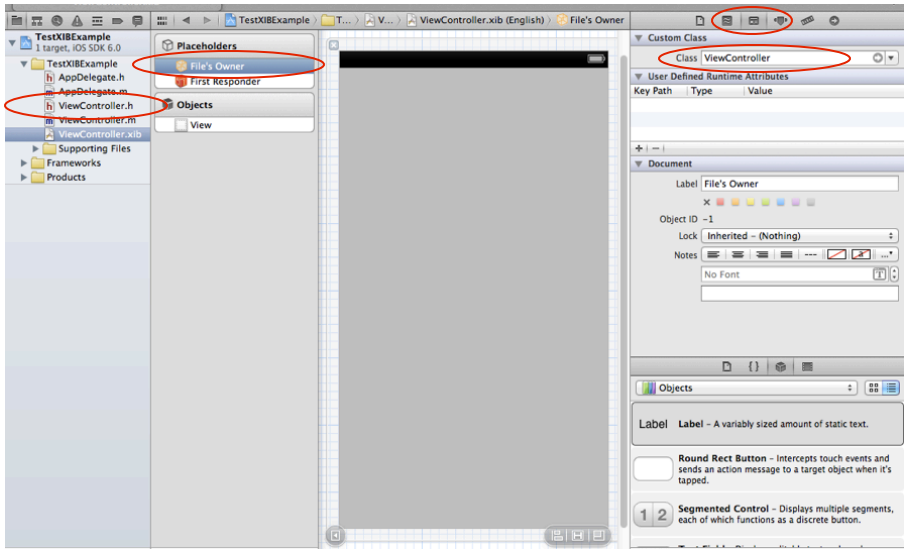
Tutorial:

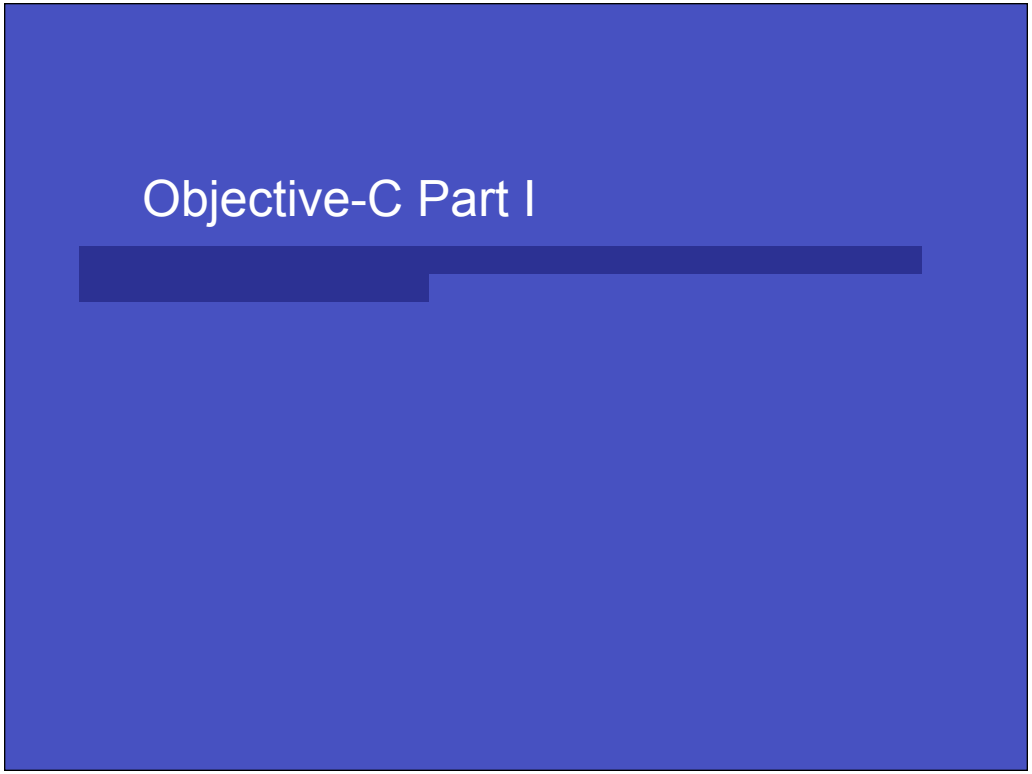
/Tutorial/XIBTutorial.pdf

NOTE:

This tutorial will start by using an “Empty Project” template and add the XIB file manually.

XIB Files: Setting the class





When Module Titles on the Module Divider pages wrap to two lines and left justify.

Module Objectives

Upon completion of this module, you will be able to:

- Compare Objective-C structure with structure of ECMAScript Languages
- Create UIKit Objects with code
- Understand the purpose of the Application Delegate
- Understand the structure of UIViewControllers

Page numeration resumes on the Module Objectives page.

Objective-C

- Object Oriented --- car
  - properties --- km / litre
  - methods --- start
  - events --- gasTankIsLow

Objective-C: .h and .m

Interface (.h)

- define the code
- NO EXECUTING CODE
- define
- define
- define

Implementation (.m)

- execute the code
- uses what has been defined in the interface

- a one-to-one relationship between these files is the easiest to manage

Objective-C Part I: NSLog

Code:

```
NSLog(@"Hello World");
NSLog(@"A number: %i", 5);
NSLog(@"A string: %@", @"my string");
NSLog(@"Some object: %@", someObject);
```

See it working:

/Examples/NSLogExample

Notes:

- A C function
- Only accepts a string
- Use string formatting to convert non-strings to strings

Further Reading:

www.roseindia.net/tutorial/iphone/examples/nslog/

This slide can be a combination of bullets, graphs, animations, etc...

if statements

ECMAScript

```
if (a == b) {

} else {

}
```

Obj-C

```
if (a == b) {

} else {

}
```

switch statements

ECMAScript

```
switch (a) {
   case 1:
      // do something
      break;
   default:
      // do something
      break;
}
```

Obj-C

```
switch (a) {
   case 1:
      // do something
      break;
   default:
      // do something
      break;
}
```

Creating Objects

ECMAScript

```
Button mybutton = new Button();
```

Obj-C

```
UIButton *mybutton = [[UIButton alloc] init];
```

Calling Methods

ECMAScript

```
mybutton.doSomething();
```

Obj-C

```
[mybutton doSomething];
```

Accessing Properties

ECMAScript

```
mybutton.myProperty = 5;
```

Obj-C

```
[mybutton setMyProperty:5];
mybutton.myProperty = 5;
```

Objective-C: Selectors

METHODS = SELECTORS

Usage:

```
[mstring replaceOccurrencesOfString: @" "
withString: @""options: 0 range: 10];
```

Name:

```
replaceOccurrencesOfString: withString: options: range:
```

Common Name:

```
replaceOccurrencesOfString
```

SELECTORS ARE UNIQUE

replaceOccurrencesOfString: withString: options: range:

replaceOccurrencesOfString: withString: options:

replaceOccurrencesOfString: withString:

Objective-C: Selectors

SELECTORS:

```
myObject1 = [self a:value1 param2:value2];

myObject2 = [myObject1 withString:@"someString"];

myObject2 = [[self withParams:value1 param2:value2] withString:@"someString"];
```

Objective-C: Events

WITH CODE:

- Register to listen to the event

```
[btn addTarget:self action:@selector(buttonClick:)
forControlEvents:UIControlEventTouchUpInside];
```

- Receive the event

```
-(IBAction) buttonClick:(id)sender{
    NSLog(@"The button was clicked!");
}
```

WITH INTERFACE BUILDER:

- Open the XIB file.
- Open the Assistant Editor
- Select the object to send the event (ex: UIButton)
- Navigate to the Connections Inspector
- Select the event to send (ex: Touch Up Inside)
- Drag the Pick Whip to the Interface that was opened in the assistant

Objective-C Part I: Objects from Code

Basic Initializers:

```
myObject = [[UIView alloc] init];
```

Custom Initializers:

```
myLabel = [[UILabel alloc] initWithFrame:CGRectMake(50., 150., 150., 50.)];
```

Convenience Initializers: (Static class selectors)

```
myButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
```

Tutorial:

/Tutorials/UIKitFromCode.pdf

This slide can be a combination of bullets, graphs, animations, etc...

Initializing Objects

```
- (id)init {
    if (self = [super init]) {
        // Initialization code
    }
    return self;
}
```

Further Reading:
http://www.cocoawithlove.com/2009/04/what-does-it-mean-when-you-assign-super.html

This slide can be a combination of bullets, graphs, animations, etc...

Objective-C: Pointers

```
NSString *firstName = self.textField.text;
```

The firstName variable becomes a pointer to the NSString object that holds the contents of text field.

That firstName variable is now the owner of that string object.

The * indicates it is a pointer.

Primitive data types are not objects, and do not have pointers.

- int
- float
- bool

Further Reading:

http://www.drdobbs.com/mobile/225700236

Objective-C: Variables (ivar)

Instance Variables are defined in the .h (header) file of a class or in the @interface section in the .m (implementation) file.

MyClass.h

```
@interface MyClass : NSObject {

  NSString *_firstName;

}

@end
```


This slide can be a combination of bullets, graphs, animations, etc...

Objective-C: Properties

Properties are defined in the header file of a class.

These properties are the externally available properties of an object.

MyClass.h

```
@interface MyClass : NSObject

@property (strong, nonatomic) NSString *firstName;

@end
```


This slide can be a combination of bullets, graphs, animations, etc...

Objective-C: Synthesizing Properties

MyClass.h

```
@interface MyClass : NSObject {
  NSString *_firstName;
}
@property (strong, nonatomic) NSString *firstName;
@end
```

Accessing a property of an object actually makes a call to a selector of the object.

Getters and Setters:

MyClass.m

```
-(NSString *) firstName{
    return _firstName;
}

-(void) setFirstName:(NSString *)firstName{
    _firstName = firstName;
}
```


This slide can be a combination of bullets, graphs, animations, etc...

Objective-C: Synthesizing Properties

Use of the @synthesize directive tells the compiler to automatically create the getter and setter selectors.

The following example will pair the property on the left with the ivar on the right.

MyClass.m

```
@synthesize firstName = _firstName;
```

The following example assumes an ivar with the same name as the property.

MyClass.m

```
@synthesize firstName;
```


This slide can be a combination of bullets, graphs, animations, etc...

Objective-C: Synthesizing Properties

NEW!!!! in XCode 4.4

The compiler automatically calls @synthesize by default for unimplemented @properties.

The only thing needed now is:

MyClass.h

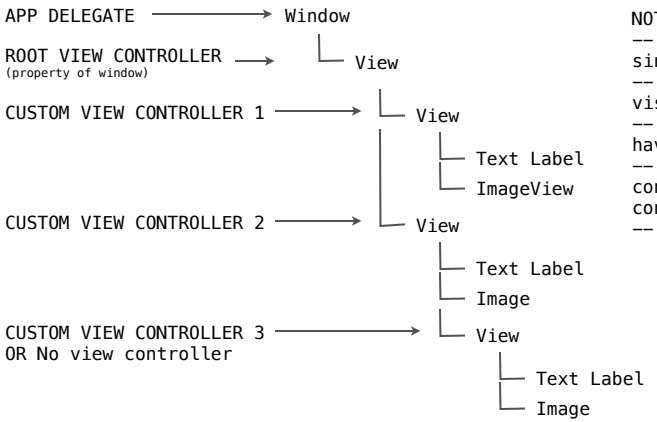
```
@interface MyClass : NSObject

@property (strong, nonatomic) NSString *firstName;

@end
```


This slide can be a combination of bullets, graphs, animations, etc...

Objective-C: A common error.

```
Terminating app due to uncaught exception
'NSInvalidArgumentException', reason: '-[ViewController
buttonClick:]: unrecognized selector sent to instance
0x91256f0'
*** First throw call stack:
```

Objective-C: Structure of an application



NOTES:
-- application must have a single root view controller
-- view controllers are not visible elements
-- each view controller must have a single child view
-- children of the view controller's view do not need controllers (but can have them)
-- unlimited nesting of views

Objective-C: AppDelegate Selectors

```
- (BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions

- (void)applicationWillResignActive:(UIApplication *)application

- (void)applicationDidEnterBackground:(UIApplication *)application

- (void)applicationWillEnterForeground:(UIApplication *)application

- (void)applicationDidBecomeActive:(UIApplication *)application

- (void)applicationWillTerminate:(UIApplication *)application
```

Objective-C: UIViewController Selectors

```
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
- (void)loadView       (NOTE: Must assign self.view)
- (void)viewDidLoad
- (void)viewDidUnload
- (void)viewWillAppear:(BOOL)animated
- (void)viewDidAppear:(BOOL)animated
- (void)viewWillDisappear:(BOOL)animated
- (void)viewDidDisappear:(BOOL)animated

- (BOOL)shouldAutorotateToInterfaceOrientation:
        (UIInterfaceOrientation)interfaceOrientation
- (void)didReceiveMemoryWarning
```

Objective-C: RootViewController

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen
mainScreen] bounds]];
    // Override point for customization after application launch.
    self.viewController = [[ViewController alloc]
initWithNibName:@"ViewController" bundle:nil];
    self.window.rootViewController = self.viewController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

Failure to set the *rootViewController* property of the window will result in the following runtime warning:

```
Application windows are expected to have a root view
controller at the end of application launch
```

Objective-C: Compiler Directives

Compiler directives give the compiler instructions. These commands are executed at compile time, not at runtime. They are designated by a @.

Examples:

```
@implementation

@end

@synthesize
```

Further Reading:

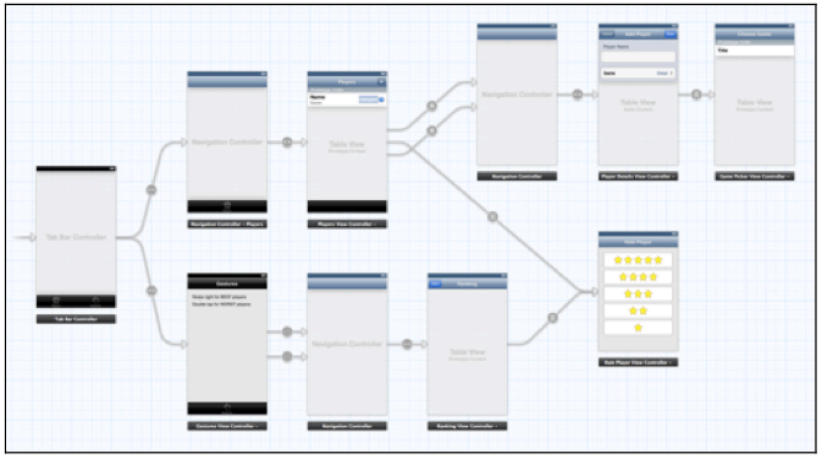
http://www.learn-cocos2d.com/2011/10/complete-list-objectivec-20-compiler-directives/

Frameworks

- UIKit
- MKMapKit
- CGCoreGraphics

When Module Titles on the Module Divider pages wrap to two lines and left justify.

Storyboards



Storyboards: Tutorial

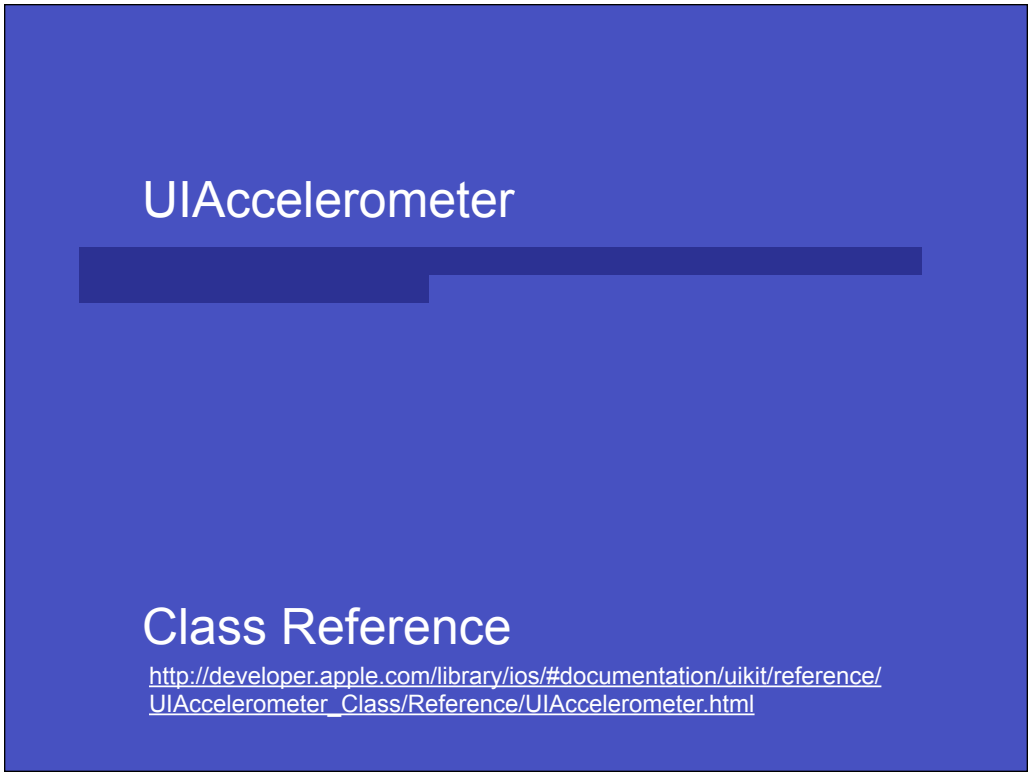
Tutorial:

/Tutorials/StoryboardsTutorial.pdf

Storyboards: Tutorial

Tutorial:

/Tutorials/ThingsProject.pdf



When Module Titles on the Module Divider pages wrap to two lines and left justify.

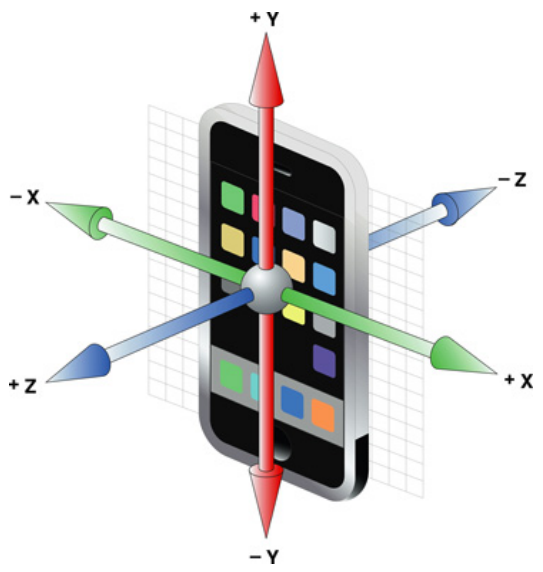
Module Objectives

Upon completion of this module, you will be able to:

- initialize and manage the UIAccelerometer object
- analyze the values of the accelerometer object
- animate objects on the screen using the accelerometer
- test apps using the simulator

Page numeration resumes on the Module Objectives page.

UIAccelerometer: Basics



Page numeration resumes on the Module Objectives page.

UIAccelerometer: Basics

Initializing Object:

```
UIAccelerometer* myAccelerometer =
    [UIAccelerometer sharedAccelerometer];
```

Setting Properties:

```
myAccelerometer.updateInterval = 0.05;
myAccelerometer.delegate = self;
```

Handling Delegate Callbacks:

```
-(void)accelerometer:(UIAccelerometer*)accelerometer
didAccelerate:(UIAcceleration*)acceleration;
{
      NSLog(@"Accleration X: %f", acceleration.x);
      NSLog(@"Accleration Y: %f", acceleration.y);
      NSLog(@"Accleration Z: %f", acceleration.z);
}
```


This slide can be a combination of bullets, graphs, animations, etc...

UIAccelerometer: High and Low Pass Filtering

Problem:

Every small movement of the device is tracked by the accelerometer and can cause “jittery” or “erratic” movements.

Solution:

Create a filter so that only certain changes in acceleration are taken into consideration.

A filter is merely a way of modifying the values the come in from the accelerometer.

High Pass Filtering:

Ignore *small* changes in the acceleration and respond only to the *large* changes.

Low Pass Filtering:

Ignore *large* changes in the acceleration and respond only to the *small* changes.

This slide can be a combination of bullets, graphs, animations, etc...

UIAccelerometer: Filtering

Low Pass Filtering Explained:

```
#define kFilteringFactor 0.1
accelX = (acceleration.x * kFilteringFactor) +
       (accelX * (1.0 - kFilteringFactor));

accelX =  10% of new value +
       90% of old value;
```

High Pass Filtering Explained:

```
#define kFilteringFactor 0.1
previousAccelY = (acceleration.y * kFilteringFactor) +
(previousAccelY * (1.0 - kFilteringFactor));
accelY = acceleration.y - self.previousAccelY;

accelY =  the actual acceleration value minus the value as
calculated by the low pass filter;
```


This slide can be a combination of bullets, graphs, animations, etc...

UIAccelerometer: Calibration

Problem:

The player wants to play a game that is controlled by the accelerometer in a position that is perfectly oriented to the ground.

Solution:

When the game first starts, store the value of the accelerometer so any future calculations remove the original value.

```
calibrationX = acceleration.x;

accelX = acceleration.x - calibrationX;
```

Further Tutorials:

http://iphonedevsdk.com/forum/iphone-sdk-tutorials/39833-tutorial-accelerometer-calibration-optimizations.html

Testing Accelerometer

- The accelerometer does not work in the simulator
- Accelerometer data can be passed from an actual device to the simulator via WiFi.
- Two apps are available to do this:
  - iSimulate (http://www.vimov.com/isimulate/)
  - AccSim (http://www.brianhpratt.net/cms/?page=accsim)

See it working: (Using AccSim)

/Examples/AccelerationExample

This slide can be a combination of bullets, graphs, animations, etc...

UITouch and UIGestureRecognizer

Class Reference

http://developer.apple.com/library/ios/#documentation/uikit/reference/UITouch_Class/Reference/Reference.html

http://developer.apple.com/library/ios/#documentation/uikit/reference/UIGestureRecognizer_Class/Reference/Reference.html

When Module Titles on the Module Divider pages wrap to two lines and left justify.

Module Objectives

Upon completion of this module, you will be able to:

- respond to a touch on the screen
- respond to multiple touches on the screen
- animate an object on the screen to follow a touch
- recognize the built in gestures

Page numeration resumes on the Module Objectives page.

UITouch: Basics

View Properties

```
self.view.userInteractionEnabled = TRUE;
self.view.multipleTouchEnabled = TRUE;
```

Handling Events (within ViewController class):

```
- (void) touchesBegan:(NSSet*)touches withEvent:(UIEvent*)event;
- (void) touchesMoved:(NSSet*)touches withEvent:(UIEvent*)event;
- (void) touchesEnded:(NSSet*)touches withEvent:(UIEvent*)event;
```


This slide can be a combination of bullets, graphs, animations, etc...

UITouch: Touches

Handling Events:

```
- (void) touchesMoved:(NSSet*)touches withEvent:(UIEvent*)event;
```

First Touch Object

```
[touches anyObject]
```

Number of Touches

```
[touches count]
```

Touch Location

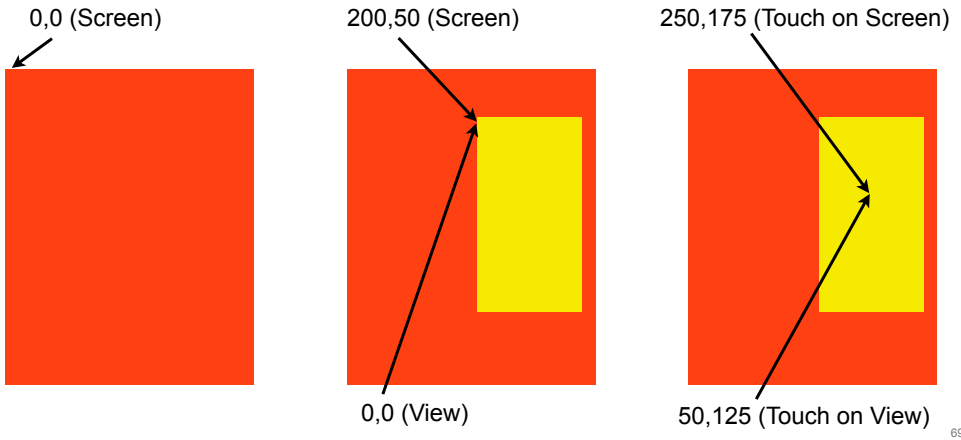
```
[[touches anyObject] locationInView:self.view]
```


This slide can be a combination of bullets, graphs, animations, etc...

UITouch: locationInView

Code:

```
- (CGPoint)locationInView:(UIView *)view
[[touches anyObject] locationInView:self.view]
```



This slide can be a combination of bullets, graphs, animations, etc...

UITouch: Examples

See it working:

/Examples/TouchesExample

/Examples/TouchesMultiExample

This slide can be a combination of bullets, graphs, animations, etc...

UIGestureRecognizer: Basics

Gestures built into iOS

- UITapGestureRecognizer
- UIPinchGestureRecognizer
- UIRotationGestureRecognizer
- UISwipeGestureRecognizer
- UIPanGestureRecognizer
- UILongPressGestureRecognizer

This slide can be a combination of bullets, graphs, animations, etc...

UIGestureRecognizer: Basics

Create gesture recognizer

```
UITapGestureRecognizer *doubleTapGesture =
[[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(handleDoubleTapGesture:)];
```

Set gesture recognizer properties

```
[doubleTapGesture setNumberOfTapsRequired:2];
```

Add the gesture recognizer onto a view

```
[self.view addGestureRecognizer:doubleTapGesture];
```

Respond to the gesture

```
- (void)handleDoubleTapGesture:(UIGestureRecognizer
*)gestureRecognizer {
    NSLog(@"Double tap 2");
}
```


This slide can be a combination of bullets, graphs, animations, etc...

UIGestureRecognizer: Task

See it working:

/Examples/GestureRecognizerExample

Challenge:

- Open the “GestureRecognizerExample” project
- add the necessary code so the view responds to a rotation gesture
- be sure to identify the value of the rotation

This slide can be a combination of bullets, graphs, animations, etc...

UIGestureRecognizer: Notes

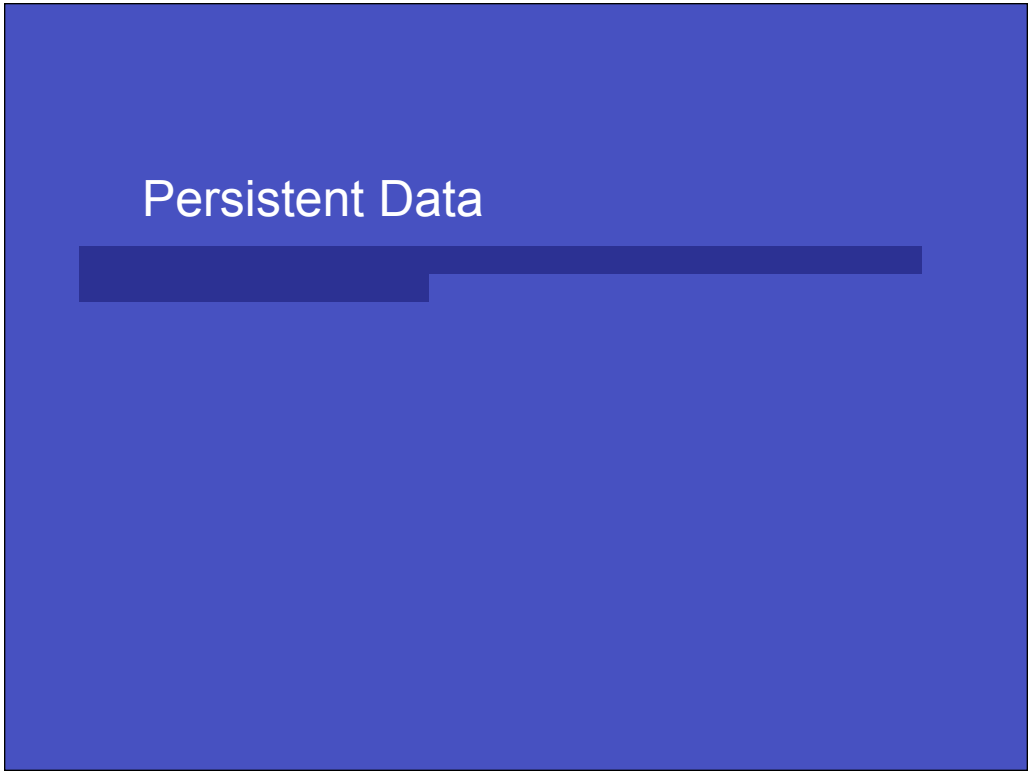
Comments:

- multiple gesture can co-exist on a single view, but only one gesture can be active at a time
- it is possible to subclass the UIGestureRecognizer class to enable custom gestures

Further Tutorials:

http://www.raywenderlich.com/6567/uigesturerecognizer-tutorial-in-ios-5-pinches-pans-and-more

This slide can be a combination of bullets, graphs, animations, etc...



When Module Titles on the Module Divider pages wrap to two lines and left justify.

Module Objectives

Upon completion of this module, you will be able to:

- save user preferences (NSUserDefaults)
- application settings (Settings Panel)
- save files to the documents directory

Page numeration resumes on the Module Objectives page.

NSUserDefaults: Code

Accessing the NSUserDefaults object (a singleton object):

```
self.prefs = [NSUserDefaults standardUserDefaults];
```

Setting a value:

```
[self.prefs setObject:@"red" forKey:@"selectedColour"];
[self.prefs synchronize];
```

Getting a value:

```
[self.prefs stringForKey:@"selectedColour"];
```

Checking to see if a key is present:

```
[self.prefs stringForKey:@"selectedColour"] == nil;
```

Default values:

```
appDefaults = [NSDictionary
dictionaryWithObjects: [NSArray arrayWithObjects:
   @"blue", @"YES", @"james", nil]
forKeys: [NSArray arrayWithObjects:
   @"selectedColour", @"setting2", @"login", nil]];

[self.prefs registerDefaults:appDefaults];
```


This slide can be a combination of bullets, graphs, animations, etc...

NSUserDefaults: Notes

Types of values that can be stored:

- NSData
- NSString
- NSNumber
- NSDate
- NSArray
- NSDictionary

NOTE: Can not store UIImageData, int, float, etc... must convert to one of the above types (ie: convert an int to NSNumber, or convert UIImage to NSData)

This slide can be a combination of bullets, graphs, animations, etc...

NSUserDefaults: Notes

- all data is deleted from the device if the app is removed from the device
- data is stored in a sandbox and not available to other apps
- store changes to values as they happen (ie: when a user changes a tab)
- store changes to values when **applicationDidEnterBackground:** is called

This slide can be a combination of bullets, graphs, animations, etc...

NSUserDefaults: Example

See it working:

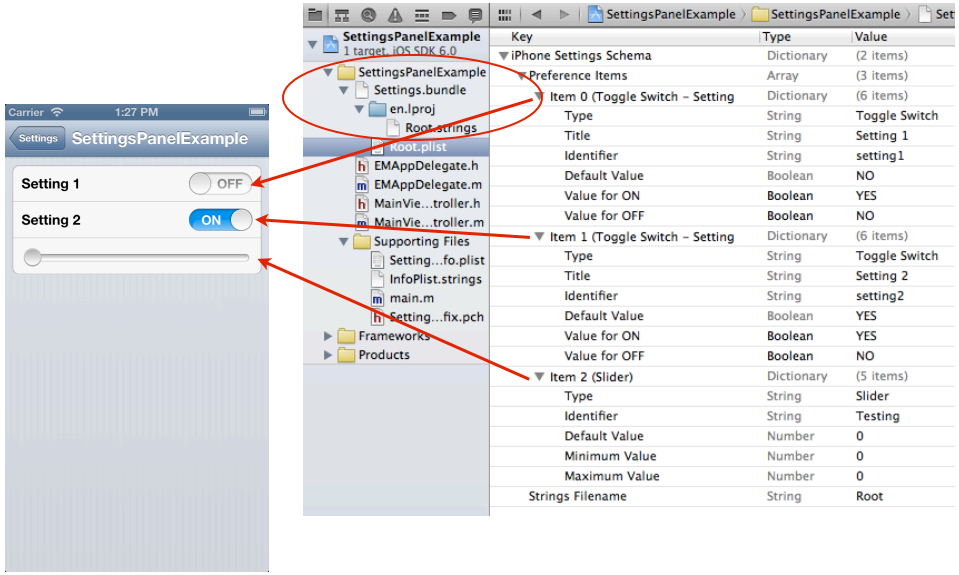
/Examples/SavingUserPreferencesExample

Challenge:

Add a UISwitch to the screen of example above and store it's value in the NSUserDefaults.

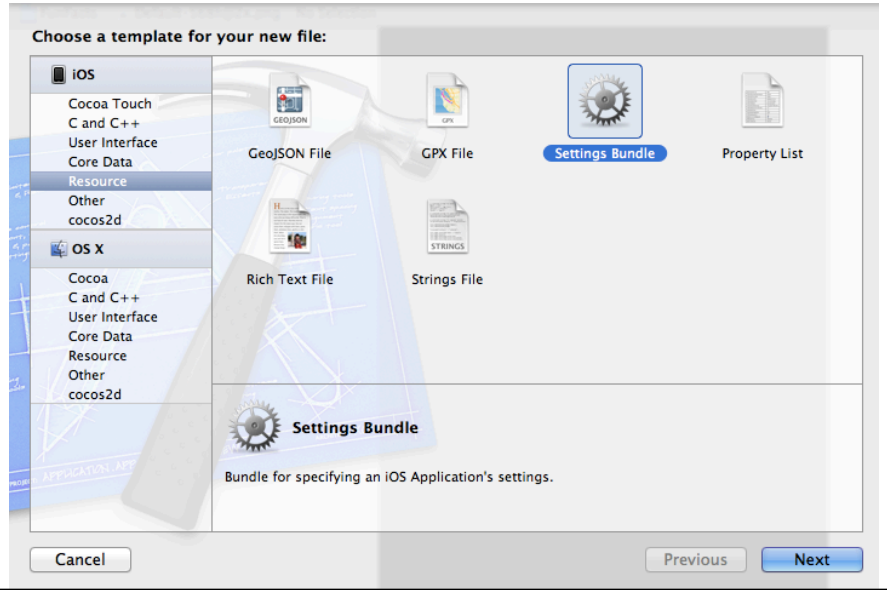
This slide can be a combination of bullets, graphs, animations, etc...

NSUserDefaults: Settings Panel



This slide can be a combination of bullets, graphs, animations, etc...

NSUserDefaults: Settings Panel



This slide can be a combination of bullets, graphs, animations, etc...

NSUserDefaults: Settings Panel Code

Accessing the NSUserDefaults object (a singleton object):

```
defaults = [NSUserDefaults standardUserDefaults];
```

Setting a value:

Setting a value is only done in the Settings Panel for the application.

Getting a value:

```
setting1.text = [defaults stringForKey:@"setting1"];
```

Checking to see if a key is present:

```
[self.prefs stringForKey:@"selectedColour"] == nil;
```


This slide can be a combination of bullets, graphs, animations, etc...

NSUserDefaults: Settings Panel Example

See it working:

/Examples/SettingsPanelExample

Further Tutorials:

NSUserDefaults: automatically register defaults from Settings.bundle

http://ijure.org/wp/archives/179

This slide can be a combination of bullets, graphs, animations, etc...

Documents Directory: Code Basics

Accessing the path to sandbox:

```
NSArray *paths = NSSearchPathForDirectoriesInDomains
(NSDocumentDirectory, NSUserDomainMask, TRUE);
```

Root for User Documents Directory:

```
NSString *documentsDirectory = [paths objectAtIndex:0];
```

Creating a file path:

```
NSString *filePath = [NSString stringWithFormat:@"%@/%@",
documentsDirectory,@"filename.jpg"];
```

Writing data to a file (NSData operation):

```
[receivedData writeToFile:filePath atomically:YES];
```

NSFileManager:

```
[[NSFileManager defaultManager] fileExistsAtPath:filePath];
[[NSFileManager defaultManager] copyItemAtPath:filePath
                toPath:filePath2 error:&error];
```


This slide can be a combination of bullets, graphs, animations, etc...

NSFileManager: Example

See it working:

/Examples/DownloadedAndSaveImageWithProgressBarExample

This slide can be a combination of bullets, graphs, animations, etc...

Various API Examples

Class Reference

http://developer.apple.com/library/ios/#documentation/Foundation/Reference/NSJSONSerialization_Class/Reference/Reference.html

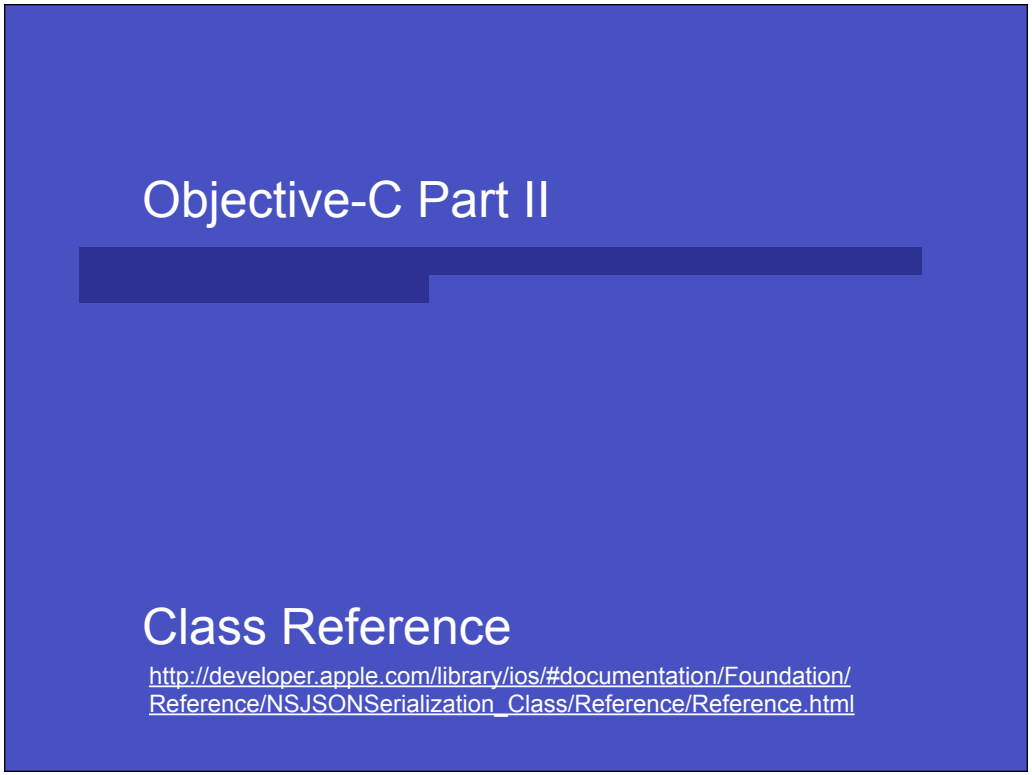
When Module Titles on the Module Divider pages wrap to two lines and left justify.

Downloading Data

See it working:

/Examples/DownloadedAndSaveImageWithProgressBarExample

Page numeration resumes on the Module Objectives page.



When Module Titles on the Module Divider pages wrap to two lines and left justify.

Module Objectives

Upon completion of this module, you will be able to:

- Blocks
- &error
- dispatch_async
- NSNotification
- adding Frameworks

Page numeration resumes on the Module Objectives page.

Objective-C: Blocks ^

```
return_type (^block_name)(param_type, param_type, ...){

}
```

At its core, a Block is a chunk of code that can be executed at some future time.

Sometimes called “closures” in other languages (Python, Lisp, etc...)

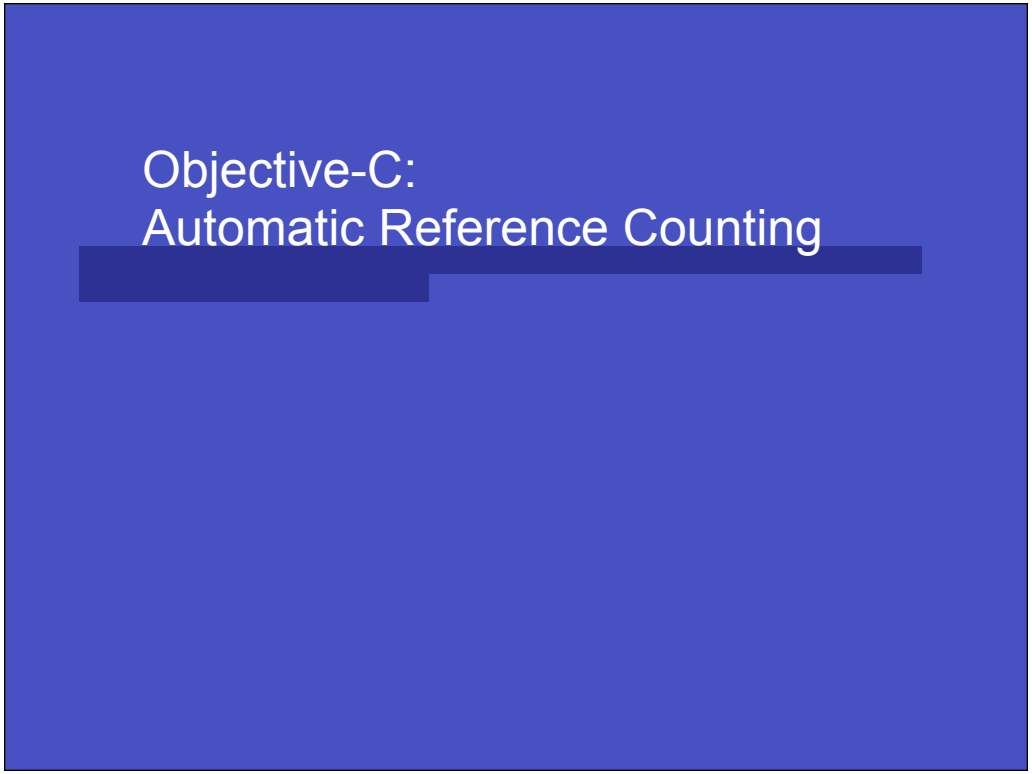
Blocks are touted as the future, replacing delegates, NSNotification, and other scenarios where callbacks are used.

See it working:

/Examples/BlocksExample

/Examples/TouchAnimationExample

This slide can be a combination of bullets, graphs, animations, etc...



When Module Titles on the Module Divider pages wrap to two lines and left justify.

Manual Memory Management

```
@property (nonatomic, retain) UIView *myView;

myView = [[UIView alloc] init] ;

[myView release];
```

ARC evaluates the lifetime requirements of your objects and automatically inserts appropriate memory management calls for you at compile time.

no more
retain, release, autorelease

http://developer.apple.com/library/mac/#releasenotes/ObjectiveC/RN-TransitioningToARC/Introduction/Introduction.html

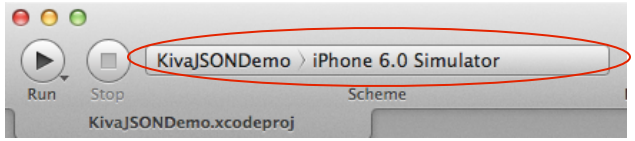
ARC: Convert Old Projects

First thing: Convert all old projects to ARC

Open the following project:
/iOS 6 By Tutorials/2 Programming in Modern Objective-C/Starter.zip

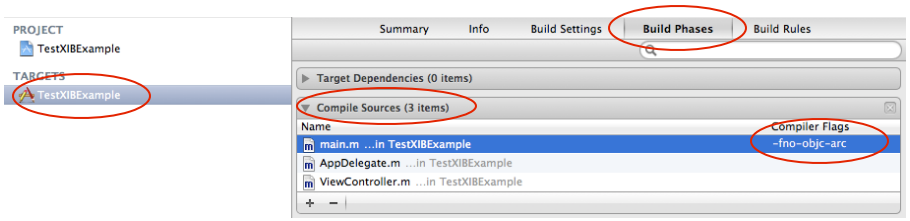
Edit --> Refactor --> Convert to Objective-C ARC

Be sure the iPhone 6.0 Simulator is selected.



ARC: Excluding Files from ARC

- Click on your Project in the Xcode project tree
- Click on the Target
- Select the Build Phases tab
- Expand the Compile Sources section
- Select one or more files you want to exclude from ARC
  (this should include both .h and .m files!)
- Press the return key (a small pop-up box will appear with a text box)
- Type -fno-objc-arc
- Press the return key again
- Each file selected now has a -fno-objc-arc compiler flag set and will be excluded from ARC



Weak vs. Strong

Strong - has ownership of object
Weak - does not have ownership of object

```
__weak NSString *str = [[NSString alloc] initWithFormat:. . .];
NSLog(@"%@", str); // will output "(null)"
```

ARC: IBOutlet

MMR

```
@property (nonatomic, retain) IBOutlet UITableView *tableView;
@property (nonatomic, retain) IBOutlet UISegmentedControl *segmentedControl;

- (void)viewDidUnload {
   [super viewDidUnload];
   self.tableView = nil;
   self.segmentedControl = nil;
   }
```

ARC (in the .m file!)

```
@property (nonatomic, weak) IBOutlet UITableView *tableView;
@property (nonatomic, weak) IBOutlet UISegmentedControl
*segmentedControl;
```

NEW Project - EMPTY

APPDELEGATE.m

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen
mainScreen] bounds]];
    // Override point for customization after application launch.

    // Do something here to set the rootViewController.
    self.window.rootViewController = [[MyViewController alloc] init];

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}
```

NEW Project - XIB

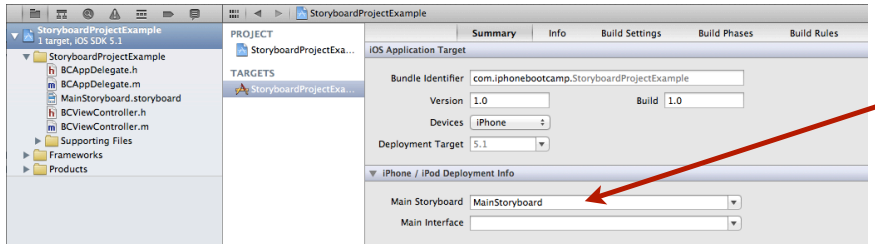
APPDELEGATE.m

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen
mainScreen] bounds]];
    // Override point for customization after application launch.
    self.viewController = [[BCViewController alloc]
initWithNibName:@"BCViewController" bundle:nil];
    self.window.rootViewController = self.viewController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

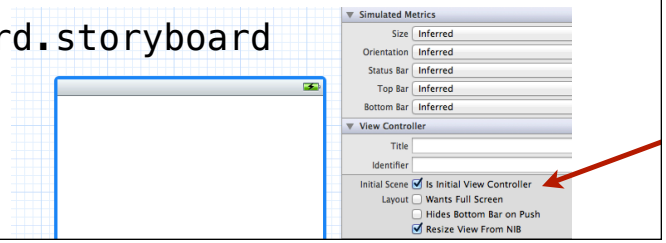
NEW PROJECT - STORYBOARDS

APPDELEGATE.m

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



MainStoryboard.storyboard



PList: Device Specific

~iphone or **~ipad**

UIMainStoryboardFile (generic fallback property)

UIMainStoryboardFile~iphone

UIMainStoryboardFile~ipad