Lesson 5: Cocoa Touch and Design Patterns

INTRODUCTION

Cocoa and **Cocoa Touch** are application development environments built from Objective-C. Cocoa is based on Cocoa API (for the Mac operating system) and includes a wide assortment of frameworks which makes application development much easier. Cocoa Touch is the version used for mobile devices and it focuses on touch interaction.

LESSON OBJECTIVES

By the end of this lesson, the student will be able to:

1. Identity the three frameworks included in almost every iOS application.
2. Explain the inclusion of Core Graphics framework in most applications.
3. Identify the different iOS architecture layers.
4. Identify the characteristics of the Model-View-Controller design pattern.
5. Identify the characteristics of the target-action design pattern.
6. Demonstrate using outlets and actions in an app.
7. Demonstrate using text fields for user input.
8. Demonstrate using label for user output.
9. Demonstrate changing the class of a UI object.
10. Demonstrate the use of FirstResponder in an app.
11. Demonstrate setting up a keyboard for number input.
12. Demonstrate changing the class of a view (UIView) to a subclass (UIControl).

LEARNING SEQUENCE

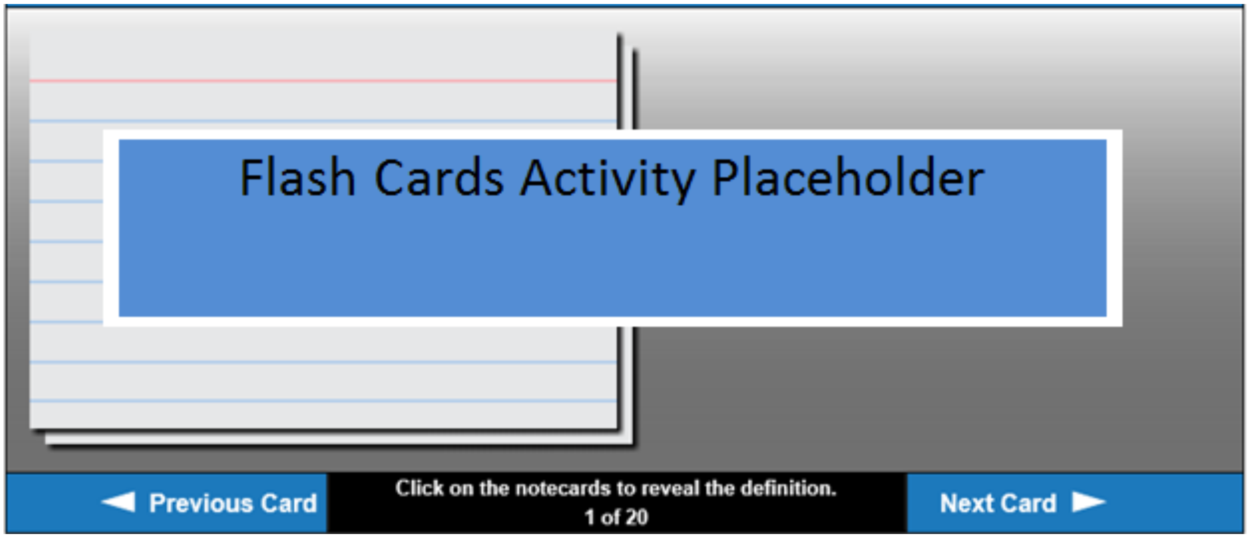
Required Reading	Read the following: • Lesson 5: Cocoa Touch and Design Patterns • iOS Developer Library > iOS icon > Getting Started > Start Developing iOS Apps Today
Resources	Review the following: • iPhone Development Wiki > Frameworks • Implementing a delegate chain system in Objective-C • iOS Developer Library > iOS icon > Release Notes • iOS Developer Library > iOS icon > Sample Code • iOS Developer Library > iOS icon > Technical Notes

Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands

Assignments	Complete the following: 1. Practice Exercise – Convert Inch 2. Quiz 5

KEY TERMS

As you read your lesson, pay close attention to the key terms and phrases listed throughout the lesson. These terms and concepts are important to your understanding of the information provided in the lesson.



Click on the notecards to reveal the definition.
1 of 20

INSTRUCTION

Frameworks

A **framework** is a set of libraries. The code accesses the capabilities of a framework through the application programming interface (API). The library is dynamically shared, so multiple apps can access the framework code and resources at the same time. There are three basic frameworks used in almost every project. When a template is created in single view, the Foundation, UIKit, and Core Graphics frameworks are automatically added.

The **Foundation framework** takes much of the Core Foundation Framework (C Framework) by wrapping it in Objective-C. The Foundation framework determines how all of the components interact with each other and includes:

- Collection data types
- Strings, dates, data blocks, URL and streams management
- Thread and run loop management
- Ports communication

Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands

- Internationalization

UIKit framework is required for interface creation and interaction. Some of the included features are:

- User interface management
- Touch and motion based events
- Accessibility support
- Device interaction support such as printing, multitasking, push notifications and pdf creation

The **Core Graphics Framework** consists of the interfaces for drawing 2D images and includes support for:

- Drawing paths
- Creating gradient fills and transformations
- Images
- Colors
- Pdf document creation and display

A list of iOS Frameworks can be viewed at the iPhone Development Wiki website. This reference provides the version that a framework was first available in to help in planning the design for a particular iOS device. The description provides a brief explanation of what each framework contains. Links for each framework display in-depth, language reference information.

Architecture

There are four layers that make up an application and Figure 1 shows the iOS architecture. The fundamental services and technologies are part of the lower layers. The upper layers build upon the lower layers and provide more pre-built items such as the frameworks and API's.

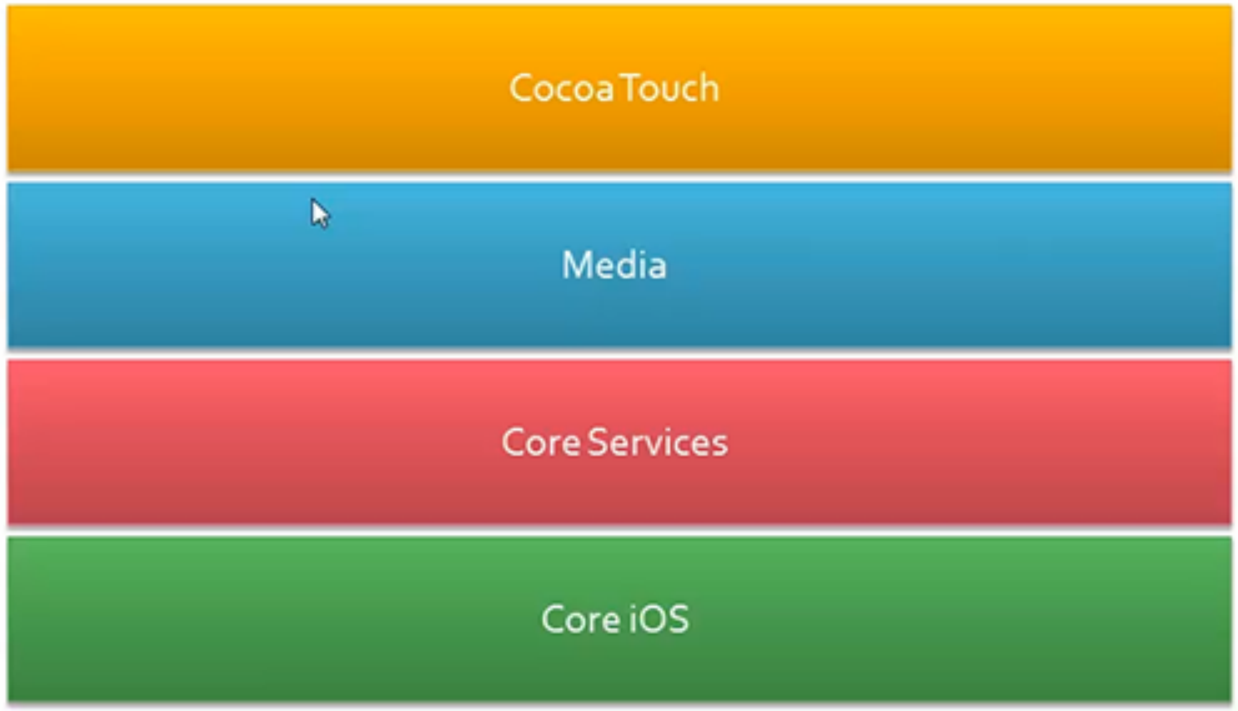


Figure 1: iOS Architecture includes the Cocoa Touch layer, Media layer
Core Services layer, and Core iOS layer

The Cocoa Touch layer is where most of the work occurs. This layer has all of the touch capabilities that make touch possible on the device. The Cocoa Touch layer includes the following:

- UIKit framework
- Map Kit
- Message UI
- Twitter framework
- Gesture recognizers
- Local notifications
- Printing
- Document support

The Media layer contains graphics, audio and video technologies and includes the following:

- Core Graphics
- Core Animation
- OpenGL ES and GLKit
- Core image
- Audio frameworks
- Media Player framework
- Core Media

Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands

- Core Audio
- AirPlay

The Core Services layer contains fundamental system services used by all applications and includes the following:

- Address Book
- Core Data (a visual interface for dealing with data and databases)
- Core Foundation
- Core Location
- Core Media
- Foundation framework
- Event Kit framework
- SQLite
- iCloud Storage

The Core OS layer is probably the least worked with layer. It is mostly used by higher level frameworks, so a programmer would rarely work with this layer directly. It includes the following:

- Core Bluetooth
- Acceleration frameworks
- Security framework
- System

Design Patterns

Design Patterns are not code. Understanding design patterns helps a programmer use frameworks more effectively and write apps that are usable and easier to change.

Model-View-Controller (MVC): Objects in an app are assigned a role: model, view, or controller. Models handle the data; views display the interface and make up the content; and controllers handle the interaction between views and models. Notice that views and models do not directly interact with each other; controllers serve as the middle man for model and view communications.

Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands

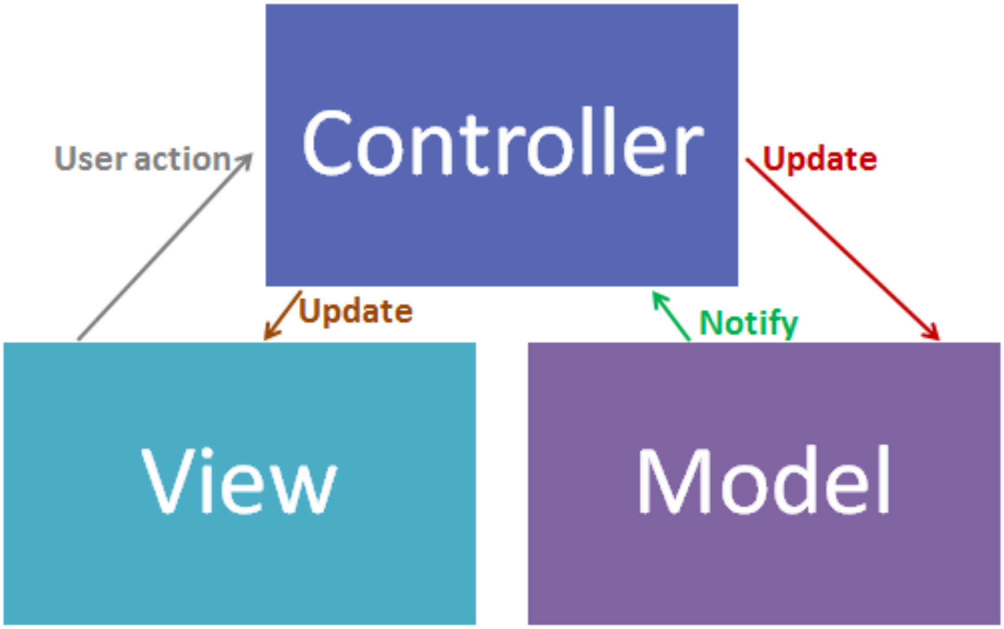


Figure 2: MVC Design Pattern

Delegation: A simple pattern that has one object, the delegating object, acting with another object, the delegate. One class has another class carry out specific actions on its behalf. The delegating object keeps a reference to the delegate and will send a message to the delegate when necessary. Recall that **protocols** are a set of methods that do a certain job or perform a certain role. Protocols are often delegates for the class that uses them. Implementing a delegate chain system in Objective-C provides an example of how delegation works.

Target-Action: An object sends a message to a second object when an event occurs. For example, say a user interacts with an object which triggers an event. Target-Action is the connection between the object and the action that occurs. Outlets allow a controller to access properties of an interface object. Actions allow a specific interface object to access a specific method.

Notifications: The **Notification Center** is part of the Foundation framework. Its function is to broadcast a message to any object that is setup to watch for messages.

Notifications can be used to notify multiple objects of an event, but a notification does not return a value. Take, for example, an app that is going into the background. A notification will be sent; anything listening for that notification hears it and then runs whatever code has been set up specifically when that event occurs.

Key-Value Observing: One object sees the property of another object. This is used to synchronize changes between related objects. A controller observes properties of models. Views observe properties of models through controllers. This means that an object is notified of changes to specified properties of other objects.

Resources

This course is only scratching the surface. Due to the depth of this topic, it is important to become familiar with several references. Start out by reviewing the resources at the iOS Developer Library.

1. Go the https://developer.apple.com.
2. Click on the iOS icon.
3. Click on *Release Notes* under Resource Types to read breaking news and information on what has changed.
4. Click on *Sample Code* to see examples of iOS features.
5. Click *Technical Notes* to view supplementary documentation.
6. Click *Getting Started*. In the right-hand pane, select *Start Developing iOS Apps Today.* The explanations and tutorials will help tie together the concepts presented so far in these lessons.

SUMMARY

This lesson introduced Cocoa Touch and the frameworks used for developing apps that run on iOS devices. This lesson also looked at design patterns which guide a programmer in developing apps that are usable and easier to change.

ASSIGNMENTS

1. Practice Exercise – Convert Inch
2. Quiz 5

Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands