




Copyright © 2011 IGATE Corporation (a part of Capgemini Group). All rights reserved.

No part of this publication shall be reproduced in any way, including but not limited to photocopy, photographic, magnetic, or other record, without the prior written permission of IGATE Corporation (a part of Capgemini Group).

IGATE Corporation (a part of Capgemini Group) considers information included in this document to be confidential and proprietary.

Document History


Date	Course Version No.	Software Version No.	Developer / SME	Change Record Remarks
Nov - 2011	1.0	Max OS X-10.6.4 IOS 4 Xcode 3.25	Anil Patil	New

 **Capgemini**
REVEALING TECHNOLOGICAL OPPORTUNITIES


March 16, 2016 | Proprietary and Confidential | - 2 -

Course Goals and Non Goals

- **Course Goals**
 - To get introduce to Mac OS X
 - Working with Mac OS X
 - To get introduce to iOS, its architecture, tools for development iOS based application and iOS quick development start
 - To get introduce to Cocoa Touch framework and various frameworks available in iOS based devices
- **Non Goals**
 - These are overview sessions. Details of each topics is not covered.



March 16, 2016 | Proprietary and Confidential | - 3 -



Mac OS X, iOS and Cocoa Touch Framework Overview

Pre-requisites


- **Programming experience preferably C, C++**
- **Basic Knowledge about Mobile device**

March 16, 2016 Proprietary and Confidential - 4 -




Intended Audience

- **Developers interested in developing mobile based application**
- **Windows users interested in learning Mac OS X**



© 2008 Apple Computer, Inc. All rights reserved.



March 16, 2008 Proprietary and Confidential - 5 -

Day Wise Schedule

> Day 1

- Lesson 1: Introduction to Mac OS X
- Lesson 2: Basic of working of Mac OS X
- Lesson 3: Introduction to iOS
- Lesson 4: Introduction to Cocoa Touch framework

Table of Contents	
➤ Lesson 1: Introduction to Mac OS X	
– 1.1: What is Mac OS	
– 1.2: History of Mac OS X	
– 1.3: Mac OS X Benefits	
– 1.4 : Mac OS X Architecture	
➤ Lesson 2: Basic of working of Mac OS X	
– 2.1: Logging in to a Mac OS X Station	
– 2.2: Changing Your Password	
– 2.3: Navigating the Mac OS X Environment	
– 2.4: Opening Programs	
– 2.5: Installing Mac OS X Applications	
– 2.6: Working with the System Preferences Application	
– 2.7: A Command Line with the Mac OS	
– 2.8: Closing Programs	
– 2.9: Handling Unresponsive Programs	
– 2.10: Logging Out of a Mac OS X Station	
– 2.11: Shutting Down the MacHibernate Object Life Cycle	

March 16, 2016 | Proprietary and Confidential | - 2 -





Table of Contents

- **Lesson 3: Introduction to iOS**
 - 3.1: What is iOS
 - 3.2: The iOS Architecture
 - 3.3: Tools for iPhone OS Development
 - 3.4: XCode
 - 3.5: Interface Builder
 - 3.6: Instruments
 - 3.7: iOS Development Quick Start
- **Lesson 2: Basic of working of Mac OS X**
 - 4.1: What Is Cocoa?
 - 4.2: Cocoa History
 - 4.3: Cocoa class libraries
 - 4.4: How Cocoa Fits into Mac OS X
 - 4.5: Cocoa and Cocoa Touch
 - 4.6: How Cocoa Fits into iOS?
 - 4.7 Frameworks available in iOS-based devices

March 16, 2016 | Proprietary and Confidential | - 8 -

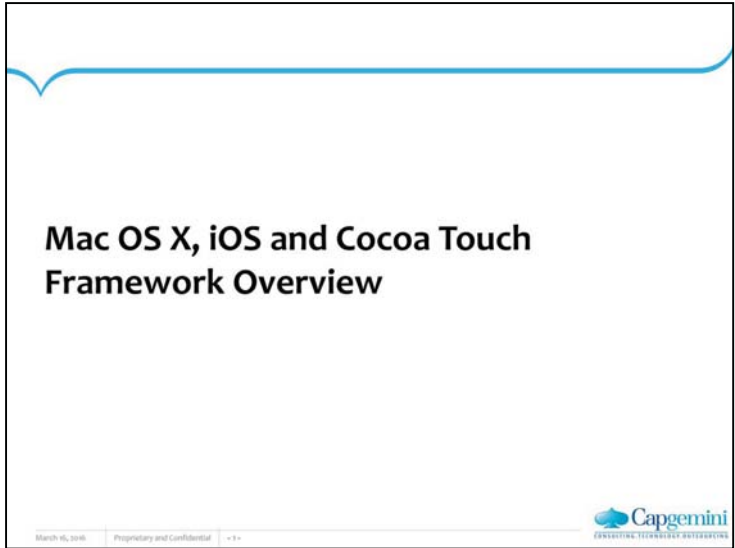


References

➤ Reference URL

- <http://developer.apple.com/devcenter/ios/index.action>
- <http://www.apple.com/macosex/>
- <http://www.uwec.edu/help/macosex/mac-basics.htm>
- <http://www.apple.com/support/mac101/work/>
- http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/GS_iPhoneGeneral/_index.html
- http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/iphone_development/100-iOS_Development_Quick_Start/development_quick_start.html#//apple_ref/doc/uid/TP40007959-CH3-SW1
- http://www.youtube.com/watch?v=oesNwgHn1ws&feature=player_embedded





Lesson Objectives

Learn

- What is Mac OS X
- History of Mac OS X
- Mac OS X Benefits
- Mac OS X Architecture



Mac OS X, iOS and Cocoa Touch Framework Overview

Topics

- What is Mac OS X
- History of Mac OS X
- Mac OS X Benefits
- Mac OS X Architecture

March 16, 2011 Proprietary and Confidential 1/3



Add the notes here.

What is Mac OS X

- Mac OS X is a series of Unix-based operating systems and graphical user interfaces developed, marketed, and sold by Apple Inc.
- Mac OS X, whose X is the Roman numeral for 10 and is a prominent part of its brand identity, is a Unix-based graphical operating system, built on technologies developed at NeXT between the second half of the 1980s and Apple's purchase of the company in late 1996. From its sixth release, Mac OS X v10.5 "Leopard" and onward, every release of Mac OS X gained UNIX 03 certification while running on Intel processors.

History of Mac OS X

- The first version released was Mac OS X Server 1.0 in 1999, and a desktop-oriented version, Mac OS X v10.0 "Cheetah" followed on March 24, 2001. Releases of Mac OS X are named after big cats: for example, Mac OS X v10.7 is usually referred to by Apple and users as "Lion".
- The server edition, Mac OS X Server, is architecturally identical to its desktop counterpart, and includes tools to facilitate management of workgroups of Mac OS X machines, and to provide access to network services. These tools include a mail transfer agent, an LDAP server, a domain name server, and others. It is pre-loaded on Apple's Xserve server hardware, but can be run on almost all of Apple's current selling computer models.
- Apple also produces specialized versions of Mac OS X for use on its consumer devices. iOS, which is based on Mac OS X, runs on the iPhone, iPod Touch, iPad, and the 2nd generation Apple TV.

History of Mac OS X

- Mac OS X is based upon the Mach kernel. Certain parts from FreeBSD's and NetBSD's implementation of Unix were incorporated in NeXTSTEP, the core of Mac OS X.
- NeXTSTEP was the object-oriented operating system developed by Steve Jobs' company NeXT after he left Apple in 1985. While Jobs was away from Apple, Apple tried to create a "next-generation" OS through the Taligent, Copland and Gershwin projects, with little success.
- Eventually, NeXT's OS, then called OPENSTEP, was selected to be the basis for Apple's next OS, and Apple purchased NeXT outright. Steve Jobs returned to Apple as interim CEO, and later became CEO, shepherding the transformation of the programmer-friendly OPENSTEP into a system that would be adopted by Apple's primary market of home users and creative professionals. The project was first known as Rhapsody and was later renamed to Mac OS X.

History of Mac OS X

- Mac OS X's core is a POSIX compliant operating system (OS) built on top of the XNU kernel, with standard Unix facilities available from the command line interface.
- Apple has released this family of software as a free and open source operating system named Darwin.
- On top of Darwin, Apple layered a number of components, including the Aqua interface and the Finder, to complete the GUI-based operating system which is Mac OS X.

History of Mac OS X

- Mac OS X architecture implements a layered design. The layered frameworks aid rapid development of applications by providing existing code for common tasks.
- Mac OS X includes its own software development tools, most prominently an integrated development environment called Xcode. Xcode provides interfaces to compilers that support several programming languages including C, C++, Objective-C, and Java. For the Apple-Intel transition, it was modified so that developers could build their applications as a universal binary, which provides compatibility with both the Intel-based and PowerPC-based Macintosh lines.

Mac OS X Benefits

- **Stability and reliability:** It is very stable because the operating system has been designed using modern architectural principles.
- **Speed:** The OS is optimized for maximum performance on Mac hardware.
- **Beauty:** Because of the advanced graphics subsystem, the images, fonts, icons, and other graphic elements of the operating system are very pleasing to look at.
- **Multiple user support**
- **Security**

March 16, 2006 Proprietary and Confidential 9



Following are some of the highlights as to why Mac OS X is a very good thing:

- **Stability and reliability:** Because the operating system has been designed using modern architectural principles, it is very stable. When an application does crash or hang, only that application is affected. The system manages its resources much more effectively than previous versions of the OS did. The result is that Mac OS X keeps working without those annoying crashes that were far too common with previous versions. Mac OS X is as stable as a rock.
- **Speed:** The OS is optimized for maximum performance on Mac hardware. It also takes advantage of other modern Mac hardware features such as faster memory, modern data buses, and so on. All operations under Mac OS X are much faster than under previous versions; these improvements in speed have continued in version 10.3. Mac OS X flies.
- **Beauty:** Because of the advanced graphics subsystem, the images, fonts, icons, and other graphic elements of the operating system are very pleasing to look at. The new interface design uses color and other graphic effects in a visually stunning way. Mac OS X looks very, very good.
- **Multiple user support:** Mac OS X is designed to facilitate many people using the same machine. Unlike previous versions of the Mac OS, this support is native to the OS rather than being an add-on. Mac OS X is meant to be shared.
- **Security:** Mac OS X has many security features you can employ to protect your machine and its data from other people who use it, as well as from those who share the same network as you, and even from Internet attacks through its built-in firewall. Mac OS X makes the digital life a secure life.

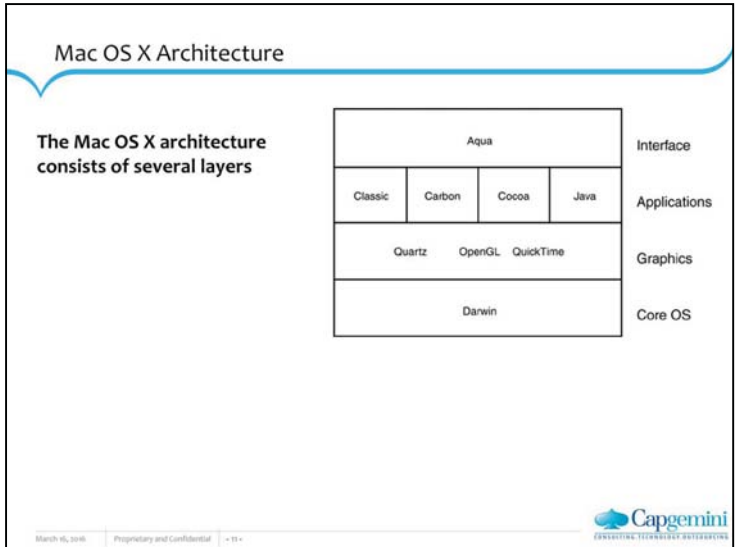
Mac OS X Benefits

- **Compatibility:** With its Classic environment, Mac OS X can use most applications that are written for earlier versions of the Mac OS.
- **Power:** Mac OS X is a very powerful OS. Its multiple layers provide this power in many areas, such as graphics, the Internet, and so on.
- **Network-readiness:** Mac OS X provides support for all sorts of networks, from those containing all Macintosh computers to those composed of Windows PCs.
- **High-technology support:** Mac OS X supports many advanced technologies, including Bluetooth, that enable the OS to interact with wireless devices, such as cell phones and PDAs.
- **Ease of use**
- **Customizability**

March 16, 2005 Proprietary and Confidential | 10 |



- **Compatibility:** With its Classic environment, Mac OS X can use most applications that are written for earlier versions of the Mac OS.
- **Power:** Mac OS X is a very powerful OS. Its multiple layers provide this power in many areas, such as graphics, the Internet, and so on. Its standards-based networking architecture enables you to connect to any system, anywhere. And you have much greater, direct access to system processes than ever before. You can access this power at many levels, from the GUI to using Unix text commands. Mac OS X has all the power you need.
- **Network-readiness:** Mac OS X's networking system is powerful, flexible, and relatively easy to configure. With its Rendezvous technology, Mac OS X Macs can automatically seek out and configure other Rendezvous devices with which they can communicate. From LANs to WANs, Mac OS X has been built to connect.
- **High-technology support:** Mac OS X supports many advanced technologies, including Bluetooth, that enable the OS to interact with wireless devices, such as cell phones and PDAs. The Ink system provides Mac OS X with handwriting recognition so that you can provide input with graphics tablets and other devices in all your Mac OS X applications.
- **Ease of use:** Although power and ease of use are usually conflicting terms, Mac OS X provides both. Its interface features the tools and techniques that have made the Mac OS traditionally the most intuitive and easiest-to-use operating system.
- **Customizability:** It wouldn't be a Mac OS if you couldn't tweak the interface to suit your preferences. Mac OS X is fully customizable, and you can adjust and tweak it to your heart's content.




The base level of the operating system is its Unix core, which is called Darwin. Moving "up" through the layers, the next layer is the graphics subsystem, which consists of three parts: Quartz, OpenGL, and QuickTime. Then comes the application layer, which has four components, those being Classic, Carbon, Cocoa, and Java. Finally, the top layer is the user interface, which is called Aqua.

Mac OS X is built on a Unix core; the Darwin core is based on the Berkeley Software Distribution (BSD) version of Unix. The heart of the Darwin core is called Mach. This part of the operating system performs the fundamental tasks, such as data flow into and from the CPU, memory use, and so on.

Mac OS X Architecture

- **The Core OS: Darwin**
- **Mac OS X is built on a Unix core; the Darwin core is based on the Berkeley Software Distribution (BSD) version of Unix. The heart of the Darwin core is called Mach. This part of the operating system performs the fundamental tasks, such as data flow into and from the CPU, memory use, and so on.**

March 06, 2008 Proprietary and Confidential - 12 - 

Some major features include the following:

- **Protected memory:** Mach provides a separate memory area in which each application can run. It ensures that each application remains in its own memory space and so does not affect other applications. Therefore, if a running application crashes or hangs, other applications aren't affected. You can safely shut down the hung application and continue working in the others.
- **Advanced virtual memory:** The Mach core uses a virtual memory system that is always on. It manages the virtual memory use efficiently so that virtual memory is used only as necessary to ensure maximum performance. Darwin also provides the input/output services for Mac OS X and easily supports three key characteristics of modern devices: plug-and-play, hot-swapping, and power management.

Mac OS X Architecture

➤ The Graphics Subsystem

➤ Mac OS X includes an advanced graphics subsystem, which has three main components: Quartz Extreme, OpenGL, and QuickTime.

- Quartz Extreme : It is graphics subsystem that handles 2D graphics
- OpenGL : This component of the graphics subsystem provides 3D graphics support for 3D applications
- QuickTime provides support for many types of digital media, such as digital video, and is the primary enabler of video and audio streaming under Mac OS X

March 16, 2006 Proprietary and Confidential - 19 -



The Graphics Subsystem

Mac OS X includes an advanced graphics subsystem, which has three main components: Quartz Extreme, OpenGL, and QuickTime.

Quartz Extreme is the name of the part of the graphics subsystem that handles 2D graphics. Quartz provides the interface graphics, fonts, and other 2D elements of the system, as well as on-the-fly rendering and antialiasing of images. Under Mac OS X, the Portable Document Format (PDF) is native to the OS. This means you can create PDF versions of any document without using a third-party application.

The OpenGL component of the graphics subsystem provides 3D graphics support for 3D applications. OpenGL is an industry standard that is also used on Windows and Unix systems. The Mac OS X implementation of OpenGL provides many 3D graphics functions, such as texture mapping, transparency, antialiasing, atmospheric effects, other special effects, and more.

QuickTime provides support for many types of digital media, such as digital video, and is the primary enabler of video and audio streaming under Mac OS X. QuickTime enables both viewing applications, such as the QuickTime Player, and creative applications, such as iMovie, iTunes, and many more. QuickTime is also an industry standard, and QuickTime files can be used on Windows and other computer platforms.

Mac OS X Architecture

➤ The Application Subsystem

➤ Mac OS X provides the Classic environment to enable it to run Classic applications. It also includes three application development environments: Carbon, Cocoa, and Java.

- The Classic environment enables Mac OS X to run applications that were written for previous versions of the OS without modification.
- The Carbon environment enables developers to port existing applications to use Carbon application program interfaces (APIs).
- The Cocoa environment offers developers object-oriented application development environment. Most of the applications included with Mac OS X are Cocoa versions.
- The Java environment enables you to run Java applications, including pure Java applications and Java applets.

March 16, 2005 Proprietary and Confidential 14



The Application Subsystem

Mac OS X provides the Classic environment to enable it to run Classic applications. It also includes three application development environments: Carbon, Cocoa, and Java.

- The Classic environment enables Mac OS X to run applications that were written for previous versions of the OS without modification. This provides access to thousands of existing applications that will run under Mac OS X.
- The Carbon environment enables developers to port existing applications to use Carbon application program interfaces (APIs). The Carbon environment offers the benefits of Darwin for Carbonized applications, such as protected memory and preemptive multitasking. Carbonizing an application is significantly less work than creating a new application from scratch, which enabled many applications to be delivered near the release of Mac OS X.
- The Cocoa environment offers developers a state-of-the-art, object-oriented application development environment. Cocoa applications are designed for Mac OS X from the ground up and take the most advantage of Mac OS X services and benefits. Most of the applications included with Mac OS X are Cocoa versions; as time passes, more and more Cocoa applications will become available and will eventually be the dominant type under Mac OS X.
- The Java environment enables you to run Java applications, including pure Java applications and Java applets. Java applications are widely used on the Web because they enable the same set of code to be executed on various platforms. You can also develop Java applications under Mac OS X.

Mac OS X Architecture

➤ The User Interface

- The Mac OS X user interface, called Aqua, provides Mac OS X's great visual experience as well as the tools you use to interact with and customize the interface to suit your preferences.
- From the drop shadows on open windows to the extensive use of color and texture to the extremely detailed icons, Aqua provides a user experience that is both pleasant and efficient.

March 16, 2006 Proprietary and Confidential - 15 -



The User Interface

The Mac OS X user interface, called Aqua, provides Mac OS X's great visual experience as well as the tools you use to interact with and customize the interface to suit your preferences. From the drop shadows on open windows to the extensive use of color and texture to the extremely detailed icons, Aqua provides a user experience that is both pleasant and efficient.

Summary

➤ **The topics covered..**

- What is Mac OS X
- History of Mac OS X
- Mac OS X Benefits
- Mac OS X Architecture



Add the notes here.

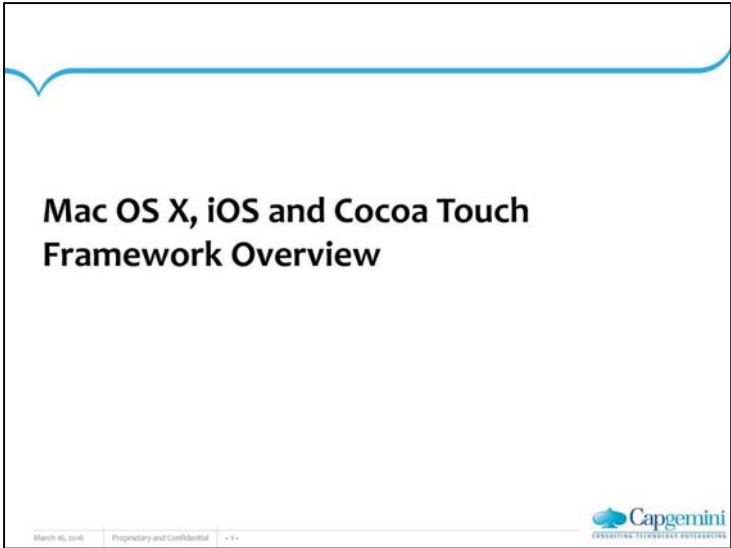
Review Question

- **Question 1: What are three main Mac OS advanced graphics?**
- **Question 2: Which of the following os was the object-oriented operating system developed by NeXT.**
 - NeXTSTEP
 - Cheetah
 - Lion
- **Question 3:**
 - Mac OS X is a based graphical operating system



Add the notes here.

Mac OS X, iOS and Cocoa Touch Framework Overview



Mac OS X, iOS and Cocoa Touch Framework Overview

Lesson Objectives

Learn

- Logging in to a Mac OS X Station
- Changing Your Password
- Navigating the Mac OS X Environment
- Opening Programs
- Installing Mac OS X Applications
- Working with the System Preferences Application
- A Command Line with the Mac OS
- Closing Programs
- Handling Unresponsive Programs
- Logging Out of a Mac OS X Station
- Shutting Down the Mac



Mac OS X, iOS and Cocoa Touch Framework Overview

Topics

- Logging in to a Mac OS X Station
- Changing Your Password
- Navigating the Mac OS X Environment
- Opening Programs
- Installing Mac OS X Applications
- Working with the System Preferences Application
- A Command Line with the Mac OS
- Closing Programs
- Handling Unresponsive Programs
- Logging Out of a Mac OS X Station
- Shutting Down the Mac

March 16, 2010 Proprietary and Confidential | 3 |



Add the notes here.

Logging In to a Mac OS X Station

- Mac OS X is truly a multiuser operating system. What you can see and do depends on the settings for the account you use to log in to the system.
- Each user account can have its own set of preferences and system resources tailored to that user. Many preferences are stored individually for each user account, so that aspect of the OS is unique to each user.
- User accounts also provide system security and control which parts of the machine a particular user can access.
- Each user account has Home folder or directory.

March 16, 2010 Proprietary and Confidential | 4 |



Mac OS X is truly a multiuser operating system. This offers many benefits to you, but it also means that when you use the OS, you have to log in as a particular user. When you do so, what you can see and do depends on the settings for the account you use to log in to the system.

Each user account can have its own set of preferences and system resources tailored to that user. Many preferences are stored individually for each user account, so that aspect of the OS is unique to each user. A simple example is the desktop picture, which can be different for each user account. Other customizable aspects of the desktop, such as the Dock, are also specific to user accounts. Many applications also store preferences specific to each user account.

User accounts also provide system security and control which parts of the machine a particular user can access. One of the most important aspects of a user account is its Home folder or directory. Under Mac OS X, the terms directory and folder are basically synonymous. Typically, non-GUI operating systems use the term directory, whereas GUI operating systems use the term folder.

Logging In to a Mac OS X Station

➤ Understanding the Administrator Account

- An administrator for your Mac can do the following:
 - Create other user accounts
 - Change global system preferences
 - Configure access to files and folders
 - Install applications

➤ **When you attempt to perform an action that requires an administrator you will see an Authentication dialog box.**

➤ **To log in, you simply select or enter the name of the user account, enter the password, and press Return. You then move to the desktop for that user account.**

March 16, 2010 Proprietary and Confidential 15



When you installed Mac OS X, you created the first user account an administrator account. A user who logs in as an administrator for your Mac can do the following:

- **Create other user accounts:** An administrator for your Mac can create additional user accounts. By default, these user accounts have more limited access to the Mac than does an administrator account, but you can allow other accounts to administer the Mac as well (in effect creating multiple administrator accounts).
- **Change global system preferences:** The administrator can change global system settings for your Mac; other user accounts can't. For example, to change the network settings on your Mac, you must be logged in as the administrator (or you must authenticate yourself as an administrator).
- **Configure access to files and folders :** An administrator can configure the security settings of files and folders to determine who can access those items and which type of access is permitted.
- **Install applications:** Applications you install under Mac OS X require that you be logged in as an administrator or that you authenticate yourself as one.

When you attempt to perform an action that requires an administrator, such as those in the previous list, you will see an Authentication dialog box. To authenticate yourself, you enter a valid administrator account username and password and click OK (if you are currently logged in as an administrator, the username is filled in automatically). This enables you to perform that action.

In areas where you need to be authenticated to perform an action, you will see the Lock icon. When the Lock is "open," you are authenticated; when the Lock is "closed," you can click it to open the Authentication dialog box.

To change your password

Steps

- 1. From the Apple menu, select System Preferences...
- 2. In the System section, click ACCOUNTS
 - The Accounts dialog box appears.
- 3. In the Accounts section, select your username
- 4. Verify that the Password tab is selected
- 5. Click CHANGE PASSWORD ...
 - A new dialog box appears.



To change your password

Steps

- 6. In the Old Password text box, type your current password
- 7. Press [tab]
- 8. In the New Password text box, type your desired new password
- 9. Press [tab]
- 10. In the Verify text box, type your new password again
- 11. Click CHANGE PASSWORD
- Your password is now changed.
- NOTE: This is the password you will use when you log in.
- 12. From the System Preferences menu, select Quit System Preferences

Navigating the Mac OS X Environment

➤ **There are several ways to get around in Mac OS X in order to find files or open programs.**

- Navigating with the Dock
 - The Dock (located at the bottom of the screen) allows you to open and move between applications with just one click.
- Navigating with the Apple Menu
 - The Apple menu in Mac OS X provides a short list of computer options such as accessing Recent Items, System Preferences, and shutting down the Mac, while also serving as one of several navigation options.
 - To use the Apple menu, single click the Apple button in the top-left corner of your screen.
- Navigating with Finder
 - The Finder feature, located both on the Dock and in the Apple menu, is a navigational tool.

March 16, 2010 Proprietary and Confidential - 8 -



There are several ways to get around in Mac OS X in order to find files or open programs.

- Navigating with the Dock
 - The Dock is located at the bottom of the screen and functions similar to a toolbar or taskbar in Windows. The Dock allows you to open and move between applications with just one click. When you want to move between different applications, simply click the application you want to work on and it will come to the front.
- Navigating with the Apple Menu
 - The Apple menu in Mac OS X provides a short list of computer options such as accessing Recent Items, System Preferences, and shutting down the Mac, while also serving as one of several navigation options.
 - To use the Apple menu, single click the Apple button in the top-left corner of your screen.
- Navigating with Finder
 - The Finder feature, located both on the Dock and in the Apple menu, is a navigational tool. Opening Finder will allow you to locate and open the programs and files that you want to use.
 - 1. On the Dock, click FINDER
 - 2. From the File menu, select New Finder Window
 - You can then navigate to the desired location.

Opening Programs

- Programs are represented by icons, which may be located on the desktop, in folders, or on different drives. If the program icon appears on the Dock at the bottom of the screen, single click the icon to launch the program.

- Follow these steps if the program you want does not appear on the Dock:
 - 1.Double click the hard drive icon on the desktop
 - 2.Click APPLICATIONS
 - 3.Double click the icon of the program you want to use

March 4, 2010 Proprietary and Confidential - 9 -



There are many ways to launch Mac OS X applications, including the following:

- Select the application in a Finder window and select the Finder's Open command (-O).
- Double-click the application's icon.
- Single-click an application's icon on the Dock.
- Open an alias to the application, such as one stored in your Favorites directory.
- Open a document of the file type that the application is set to open.
- Drag and drop a document onto an application's icon (or an alias's icon).
- Select an application's icon or alias and press -down arrow.
- Launch the application from within another application. (For example, you can launch a Web browser by clicking a URL in an email program.)
- Add the application to the Login Items window so it is launched automatically when you log in.

Installing Mac OS X Applications

- Under Mac OS X, the two basic strategies by which applications are installed are
 - Using Drag and Drop
 - Using Installer
- Because Mac OS X is designed as a multiuser OS, where you install Mac OS X applications is an important consideration.
- The two locations in which you should install Mac OS X applications are
 - The Applications folder- application accessible to everyone who uses your Mac
 - The Home folder- if you don't want everyone to access your application

March 16, 2010 Proprietary and Confidential 10



Under Mac OS X, the two basic strategies by which applications are installed are

- Drag and drop: Under this method, you simply drag the application files (usually just one file or folder) from one location to the location in which you want to install them (usually the Applications folder).
- Installer: Some applications use an installation program to install the application and related files for you. Most applications use the standard Mac OS X Installation application as their installation mechanism. These applications are provided as package files, which have the file extension .pkg.

Because Mac OS X is designed as a multiuser OS, where you install Mac OS X applications is an important consideration. The two locations in which you should install Mac OS X applications are

- The Applications folder: If you want the application to be accessible to everyone who uses your Mac, you should install it in the Applications folder. To do this, you must be logged in as an administrator. Most applications that use an installer are installed in the Applications folder, and you usually don't have the option to install them elsewhere.
- The Home folder: You can sometimes install applications in a user's Home directory (primarily applications that have drag-and-drop installation). You should install applications in a user's Home directory only if you don't want everyone who uses your Mac to be able to use that application. Because users can access only areas to which they have been granted permission through their security settings, you need to ensure that everyone who needs to use the application can access the location in which it is stored.

Installing Mac OS X Applications

➤ Using Drag and Drop

- The general process to install an application provided in a .smi or .dmg file is the following:
 1. Download and uncompress the .smi or .dmg file.
 2. If the file isn't mounted on your Mac automatically, double-click the file (which is likely a .dmg file). Its volume is mounted on your desktop, just like any other volumes, such as a CD, DVD, or volume on a hard drive.
 3. Open the resulting volume and drag the application's folder or file to the appropriate directory on your Mac.
 4. Unmount the mounted volume by selecting it and pressing -E (or select File, Eject).
 5. Discard the .smi or .dmg files if you won't need to install the application again.
- However, in most cases, I recommend that you keep the original file in the event you need to reinstall the application (for example, you can copy it to a CD or DVD).

March 06, 2008 Proprietary and Confidential | 11



The only difference between the behavior of .smi and .dmg files is that .smi files automatically mount on your desktop when you launch them. Disk image files use Apple's Disk Utility software to mount because this application is installed on your Mac by default, these files behave quite similarly and you probably won't notice any difference between them. However, you could use a .smi file even if Disk Utility wasn't installed on your machine, whereas you can't use a .dmg file without the Disk Copy application.

Installing Mac OS X Applications

➤ Using Installer

- The general process to install and use .pkg files is the following:
 - 1.Download and prepare the file containing the application you want to install.
 - 2.Mount the disk image and open it.
 - 3.Double-click the .pkg file.
 - 4.Work through the steps in the installer application.

Working with the System Preferences Application

- The System Preferences application enables you to set preferences for many areas of Mac OS X.
- You will use the System Preferences application so frequently, it is worthy of a more detailed look.
- You can open the System Preferences utility in various ways, including the following:
 - Select Apple, System Preferences.
 - Click the System Preferences icon on the Dock.
 - Open the Applications folder and then open System Preferences using its icon.



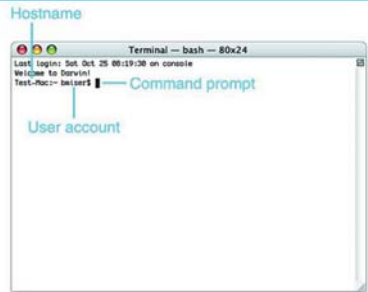
March 16, 2008 Proprietary and Confidential - 19 -



The System Preferences application provides a window with a toolbar at the top and a series of icons or buttons in the bottom part. To access the controls for a specific area, you click its icon. The System Preferences window changes to show the pane for that area.

A Command Line with the Mac OS

- You use the Terminal application (Applications/Utilities) to enter Unix commands in the command-line interface.
- The Terminal window is simple; all you see are your last login date and time, a Welcome message, the hostname, the user account under which you are logged in, and the command prompt.



March 4, 2010 Proprietary and Confidential 14



As you learned earlier in the book, Mac OS X is running on top of a version of the Unix operating system. This means that Mac OS X can use many of the Unix applications. It also means you can enter Unix commands directly in the command-line interface to manipulate your system. In fact, in some situations, using a Unix command might be the only way you can accomplish a task (such as deleting a rogue file that you can't delete by dragging it to the Trash).

Closing Programs

- You should always close programs before logging out to help protect your data.
- The following general instructions will work with most Mac OS X programs.
 - 1.If necessary, save your work
 - 2.Press [command] + [Q]
 - OR
 - From the program's menu, select Quit



Handling Unresponsive Programs

➤ When a program stops working, or “freezes,” try to quit the program by following these steps:

- 1. Press [command] + [option] + [esc]
 - OR
 - From the Apple menu, select Force Quit...
 - The Force Quit Applications dialog box appears.
- 2. From the scroll box, select the name of the unresponsive program
- 3. Click FORCE QUIT
 - NOTE: Any unsaved changes in your document(s) will be lost. For this reason, it is always a good idea to save your work often.
- 4. To close the Force Quit Applications window, click the red button in the top-left corner of the window

Mac OS X, iOS and Cocoa Touch Framework Overview

Logging Out of a Mac OS X Station

- **Remember to log out from your workstation when you are finished using it.**
 - 1.From the Apple menu, select Log Out...
 - 2.In the dialog box that appears, click LOG OUT
 - The Mac OS X dialog box appears.

March 06, 2008 Proprietary and Confidential - 17 -



Failure to log out can result in unauthorized access and modifications to your data and files.

Shutting Down the Mac

- Always shut down your Mac correctly using the Shut Down feature.
- This tells the Mac to do some small yet important housekeeping chores before it turns off.
- To Shut down the Mac
 - 1.From the Apple menu, select Shut Down...
 - 2.In the dialog box that appears, click SHUT DOWN
 - Mac OS X will then shut down.

March 16, 2010 Proprietary and Confidential - 18 -



WARNING: If you shut down the Mac with the switch, you will lose the contents of the disk cache and perhaps damage any open files.

Mac OS X, iOS and Cocoa Touch Framework Overview

Summary

➤ **The topics covered..**

- Logging in to a Mac OS X Station
- Changing Your Password
- Navigating the Mac OS X Environment
- Opening Programs
- Installing Mac OS X Applications
- Working with the System Preferences Application
- A Command Line with the Mac OS
- Closing Programs
- Handling Unresponsive Programs
- Logging Out of a Mac OS X Station
- Shutting Down the Mac



March 16, 2010 Proprietary and Confidential | 19

 **Capgemini**
ENJOYING THE BEST OF BOTH WORLDS

Add the notes here.

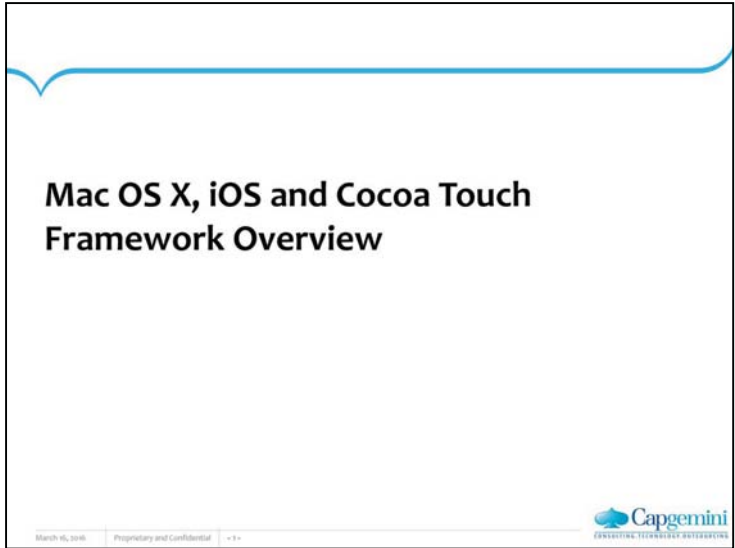
Mac OS X, iOS and Cocoa Touch Framework Overview

Review Question

- Question 1: The System Preferences, shutting down the Mac options are available in ----- menu.
- Question 2: To open program that does not appear on the Dock, you will have to single click the icon of the program you want to use..
 - True or False
- Question 3: Which file type provide the drag and drop installation ?
 - .pkg
 - .smi
 - .dmg
- Question 4: To open Force Quit Applications dialog box appears, which of the following key sequence will you use?
 - 1.Press [command] + [option] + [esc]
 - 2.Press [command] + [option] + [F]
 - 3.Press [command] + [option] + [D]



Mac OS X, iOS and Cocoa Touch Framework Overview



Mac OS X, iOS and Cocoa Touch Framework Overview

Lesson Objectives

➤ Learn

- What is iOS
- The iOS Architecture
- Tools for iPhone OS Development
 - XCode
 - Interface Builder
 - Instruments

➤ iOS Development Quick Start



Mac OS X, iOS and Cocoa Touch Framework Overview

Topics

- What is iOS
- The iOS Architecture
- Tools for iPhone OS Development
- XCode
- Interface Builder
- Instruments
- iOS Development Quick Start

March 16, 2010 Proprietary and Confidential 1/3



Add the notes here.

Mac OS X, iOS and Cocoa Touch Framework Overview

What is iOS

- iOS is the operating system at the heart of iPhone, iPod touch, and iPad devices.
- The iOS Software Development Kit (SDK) provides everything you need to get started creating iOS applications.

March 16, 2010 Proprietary and Confidential | 4



iOS is the operating system at the heart of iPhone, iPod touch, and iPad devices. The iOS platform was built using the knowledge that went into the creation of Mac OS X, and many of the tools and technologies used for development on the platform have their roots in Mac OS X as well. Despite its similarities to Mac OS X, iOS does not require experience developing Mac OS X applications. The iOS Software Development Kit (SDK) provides everything you need to get started creating iOS applications.

Mac OS X, iOS and Cocoa Touch Framework Overview

Getting the iOS SDK

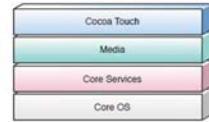
- The iOS SDK contains the tools needed to design, create, debug, and optimize software for iOS. It also contains header files, sample code, and documentation for the platform's technologies.
- You can download the iOS SDK from the members area of the iOS Dev Center, which is located at:
 - <http://developer.apple.com/devcenter/ios>

March 16, 2010 Proprietary and Confidential 15



The iOS Architecture

- The implementation of iOS technologies can be viewed as a set of layers, which are shown in Figure.
- At the lower layers of the system are the fundamental services and technologies on which all applications rely
- Higher-level layers contain more sophisticated services and technologies.



March 4, 2010 Proprietary and Confidential - 6 -



The iOS architecture is similar to the basic architecture found in Mac OS X. At the highest level, iOS acts as an intermediary between the underlying hardware and the applications that appear on the screen, as shown in Figure. The applications you create rarely talk to the underlying hardware directly. Instead, applications communicate with the hardware through a set of well-defined system interfaces that protect your application from hardware changes. This abstraction makes it easy to write applications that work consistently on devices with different hardware capabilities.

What's in the iOS SDK?

- The iOS SDK comes with all of the interfaces, tools, and resources needed to develop iOS applications from your Intel-based Macintosh computer.
- Apple delivers most of its system interfaces in special packages called frameworks.
- To use frameworks, you link them into your application project just as you would any other shared library.
- Some other key components of the SDK include:
 - Xcode Tools—the tools that support iOS application development
 - Xcode—an integrated development environment that manages your application projects
 - Instruments—a runtime performance analysis and debugging tool.
 - iOS Simulator—a Mac OS X application that simulates the iOS technology stack
 - iOS Developer Library

March 16, 2010 Proprietary and Confidential 17



The iOS SDK comes with all of the interfaces, tools, and resources needed to develop iOS applications from your Intel-based Macintosh computer.

Apple delivers most of its system interfaces in special packages called frameworks. A framework is a directory that contains a dynamic shared library and the resources (such as header files, images, helper applications, and so on) needed to support that library. To use frameworks, you link them into your application project just as you would any other shared library. Linking them to your project gives you access to the features of the framework and also lets the development tools know where to find the header files and other framework resources.

Some other key components of the SDK include

- Xcode Tools—the tools that support iOS application development, including the following key applications
- Xcode—an integrated development environment that manages your application projects and lets you edit, compile, run, and debug your code. Xcode.
- Instruments—a runtime performance analysis and debugging tool. You can use Instruments together information about your application's runtime behavior and identify potential problems.
- iOS Simulator—a Mac OS X application that simulates the iOS technology stack, allowing you to test iOS applications locally on your Intel-based Macintosh computer.
- iOS Developer Library—the reference and conceptual documentation that teaches you all about iOS technologies and the application-development process.

What Can You Create?

- iOS supports the development of two types of applications:
 - Native applications
 - Web applications
- The iOS SDK supports the creation of native applications that appear on the device's Home screen only.
- Web applications run inside the Safari web browser
- Native applications, on the other hand, are installed directly on the device and can run without the presence of a network connection.



March 06, 2008 Proprietary and Confidential - 8 -

What Can You Create?

iOS supports the development of two types of applications:

- Native applications
- Web applications

The iOS SDK supports the creation of native applications that appear on the device's Home screen only. It

does not support the creation of other types of code, such as drivers, frameworks, or dynamic libraries. If you

want to integrate code from a framework or dynamic library into your application, you should link that code

statically into your application's executable file when building your project.

Web applications use a combination of HTML, cascading style sheets (CSS), and JavaScript code to implement

interactive applications that live on a web server, are transmitted over the network, and run inside the Safari

web browser. Native applications, on the other hand, are installed directly on the device and can run without

the presence of a network connection.

Tools for iPhone OS Development

- To develop applications for iPhone OS, you need a Mac OS X computer running the Xcode tools.
- Xcode is Apple's suite of development tools that provide support for project management, code editing, building executables, source-level debugging, source-code repository management, performance tuning, and much more.

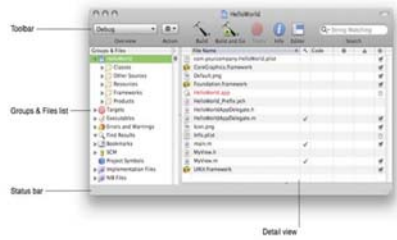
March 16, 2008 Proprietary and Confidential - 9 -



To develop applications for iPhone OS, you need a Mac OS X computer running the Xcode tools. Xcode is Apple's suite of development tools that provide support for project management, code editing, building executables, source-level debugging, source-code repository management, performance tuning, and much more. At the center of this suite is the Xcode application itself, which provides the basic source-code development environment.

XCode

- Xcode is an integrated development environment (IDE)
- To create a new iPhone application, you start by creating a new project in Xcode.



March 16, 2010 Proprietary and Confidential 10



The focus of your development experiences is the Xcode application. Xcode is an integrated development environment (IDE) that provides all of the tools you need to create and manage your iPhone projects and source files, build your code into an executable, and run and debug your code either in iPhone simulator or on a device. To create a new iPhone application, you start by creating a new project in Xcode. A project manages all of the information associated with your application, including the source files, build settings, and rules needed to put all of the pieces together. The heart of every Xcode project is the project window, shown in Figure. This window provides quick access to all of the key elements of your application. The Groups and Files list is where you manage the files in your project, including your source files and the build targets that are created from those source files. The toolbar provides access to commonly used tools and commands while the details pane provides a configurable space for working on your project. Other aspects of the project window provide you with contextual information about your project.

Interface Builder

- Interface Builder is the tool you use to assemble your application's user interface visually.
- Using Interface Builder, you assemble your application's window by dragging and dropping preconfigured components onto it.
- The components include standard system controls such as switches, text fields, and buttons, and also custom views to represent the views your application provides.
- After you've placed the components on the window's surface, you can position them by dragging them around, configure their attributes using the inspector, and establish the relationships between those objects and your code.
- When your interface looks the way you want it, you save the contents to a nib file, which is a custom resource file format.

March 16, 2010 Proprietary and Confidential | 11



Interface Builder is the tool you use to assemble your application's user interface visually. Using Interface Builder, you assemble your application's window by dragging and dropping preconfigured components onto it. The components include standard system controls such as switches, text fields, and buttons, and also custom views to represent the views your application provides. After you've placed the components on the window's surface, you can position them by dragging them around, configure their attributes using the inspector, and establish the relationships between those objects and your code. When your interface looks the way you want it, you save the contents to a nib file, which is a custom resource file format.

Interface Builder

- Using Interface Builder saves a tremendous amount of time when it comes to creating your application's user interface.
- Interface Builder eliminates the custom code needed to create, configure, and position the objects that make up your interface.
- Because it is a visual editor, you get to see exactly what your interface will look like at runtime.



March 16, 2010 Proprietary and Confidential - 10 -



The nib files you create in Interface Builder contain all the information that the UI Kit needs to recreate the same objects in your application at runtime. Loading a nib file creates runtime versions of all the objects stored in the file, configuring them exactly as they were in Interface Builder. It also uses the connection information you specified to establish connections between the newly created objects and any existing objects in your application. These connections provide your code with pointers to the nib-file objects and also provide the information the objects themselves need to communicate user actions to your code.

Overall, using Interface Builder saves a tremendous amount of time when it comes to creating your application's user interface. Interface Builder eliminates the custom code needed to create, configure, and position the objects that make up your interface. Because it is a visual editor, you get to see exactly what your interface will look like at runtime.

Instruments

- Instruments environment lets you analyze the performance of your iPhone applications while running in the simulator or on a device.
- Instruments gathers data from your running application and presents that data in a graphical display called the timeline.
- You can gather data about your application's memory usage, disk activity, network activity, and graphics performance.
- The timeline view can display all of the different types of information side by side, letting you correlate the overall behavior of your application, not just the behavior in one specific area.
- To get even more detailed information, you can also view the detailed samples that Instruments gathers.

iOS Development Quick Start

➤ Essential Development Tasks

- 1. Create your project.
- 2. Design the user interface.
- 3. Write code.
- 4. Build and run your application.
- 5. Measure and tune application performance.



March 06, 2018 Proprietary and Confidential 14

The iOS-application development process is divided into these major steps:

1. Create your project.

Xcode provides several project templates that get you started. You choose the template that implements the type of application you want to develop.

2. Design the user interface.

The Interface Builder application lets you design your application's user interface graphically and save

those designs as resource files that your application loads at runtime. If you do not want to use Interface

Builder, you can layout your user interface programmatically.

3. Write code.

Xcode provides several features that help you write code fast, including class and data modeling, code

completion, direct access to documentation, and refactoring. for details.

4. Build and run your application.


You build your application on your computer and run it in the iOS simulation environment or on your device.

5. Measure and tune application performance.


Mac OS X, iOS and Cocoa Touch Framework Overview

iOS Development Quick Start

- Demo on using Xcode to develop helloworld application



March 16, 2010 Proprietary and Confidential - 15 -



Add the notes here.

Mac OS X, iOS and Cocoa Touch Framework Overview

Summary

➤ The topics covered..

- What is iOS
- The iOS Architecture
- Tools for iPhone OS Development
- XCode
- Interface Builder
- Instruments
- iOS Development Quick Start



March 16, 2010 Proprietary and Confidential | 16 |

 **Capgemini**
ENSAUVE. TROUVEZ VOS SOLUTIONS.

Add the notes here.

Mac OS X, iOS and Cocoa Touch Framework Overview

Review Question

- Question 1: iOS is the at the heart of iPhone, iPod touch, and iPad devices.
- Question 2: The iOS SDK contains the tools needed to design, create, debug, and for iOS.
- Question 3: The implementation of iOS technologies can be viewed as a set of layers, arrange them from bottom to top. (mention 1st at the bottom).
 - Cocoa Touch b) Media c) Core Services d) Core OS
- Question 4: Which of tool following tool support iOS application development
 - a) iTunes b) iPhone c) iPhone Simulator d) Xcode
- Question 5: Application's user interface is developed using

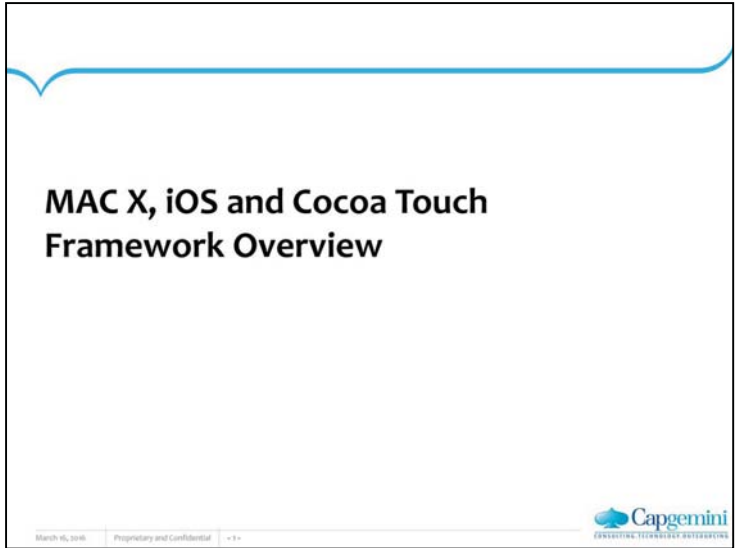


March 06, 2018

Proprietary and Confidential | - 17 -

Capgemini
INSPIRING TRANSFORMATIONS

MAC X, iOS and Cocoa Touch Framework Overview



MAC X, iOS and Cocoa Touch Framework Overview

Lesson Objectives

➤ Learn

- What Is Cocoa?
- Cocoa History
- Cocoa class libraries
- How Cocoa Fits into Mac OS X
- Cocoa and Cocoa Touch
- How Cocoa Fits into iOS?
- Frameworks available in iOS-based devices



MAC X, iOS and Cocoa Touch Framework Overview

Topics

- What Is Cocoa?
- Cocoa History
- Cocoa class libraries
- How Cocoa Fits into Mac OS X
- Cocoa and Cocoa Touch
- How Cocoa Fits into iOS?
- Frameworks available in iOS-based devices

March 16, 2010 Proprietary and Confidential | 3 |



Add the notes here.

MAC X, iOS and Cocoa Touch Framework Overview

What Is Cocoa?

- Cocoa is an application environment for both the Mac OS X operating system and iOS.
- It consists of a suite of object-oriented software libraries, a runtime system, and an integrated development environment.
- Most of the applications you see in Mac OS X and iOS, including Mail and Safari, are Cocoa applications.
- An integrated development environment called Xcode supports application development for both platforms.

March 16, 2010 Proprietary and Confidential | 4 |



Cocoa is a set of object-oriented frameworks that provides a runtime environment for applications running in Mac OS X and iOS. Cocoa is the preeminent application environment for Mac OS X and the only application environment for iOS. (Carbon is an alternative environment in Mac OS X, but it is a compatibility framework with procedural programmatic interfaces intended to support existing Mac OS X code bases.) Most of the applications you see in Mac OS X and iOS, including Mail and Safari, are Cocoa applications. An integrated development environment called Xcode, supports application development for both platforms. The combination of this development environment and Cocoa makes it easy to create a well-factored, full-featured application.

MAC X, iOS and Cocoa Touch Framework Overview

What Is Cocoa?

- You can use several programming languages when developing Cocoa software, but the essential, required language is Objective-C.
- Objective-C is a superset of ANSI to support object-oriented programming.
- The few added conventions are easy to learn and use.
- You can freely intermix straight C code with Objective-C code.
- Your code can call functions defined in non-Cocoa programmatic interfaces, such as the BSD library interfaces in `/usr/include`
- You can mix C++ code with your Cocoa code and link the compiled code into the same executable.

MAC X, iOS and Cocoa Touch Framework Overview

Cocoa History

- Apple acquired NeXT Software (as it was then called) in 1997, also bringing back Steve Jobs to the company he co-founded
- Much of the work that went into developing OpenStep was applied to the development of Mac OS X, and the application runtime environment was finally renamed to Cocoa.
- Because of its history all Cocoa classes begin with the acronym "NS" (NSObject, NSString, NSArray, etc.)

March 4, 2010 Proprietary and Confidential - 6 -



When Steve Jobs left Apple in 1985 he started a new company called NeXT Computer that developed and manufactured computer workstations intended for higher education and business markets. NeXT Computer developed and released version 1.0 of NeXTSTEP in 1989. In its early phase, NeXTSTEP was more than an application environment, the term referred to the entire operating system. Sales of NeXT computers were limited, but its innovative object-oriented NeXTSTEP operating system was very influential.

Around 1993 the OpenStep initiative took form. OpenStep was a collaboration between Sun and NeXT to port the higher levels of NeXTSTEP to Solaris. The official OpenStep API was published in September of 1994.

Apple acquired NeXT Software (as it was then called) in 1997, also bringing back Steve Jobs to the company he co-founded. Much of the work that went into developing OpenStep was applied to the development of Mac OS X, and the application runtime environment was finally renamed to Cocoa. Because of its history all Cocoa classes begin with the acronym "NS" (NSObject, NSString, NSArray, etc.)

MAC X, iOS and Cocoa Touch Framework Overview

Cocoa class libraries

- **The most important Cocoa class libraries come packaged in two core frameworks for each platform:**
 - Foundation and AppKit for Mac OS X, and
 - Foundation and UIKit for iOS.
- **Both platforms additionally support the Core Data framework, which is as important and useful as the core frameworks**
- **Mac OS X also ships with several other frameworks that publish Cocoa programmatic interfaces, such as the WebKit and Address Book frameworks**

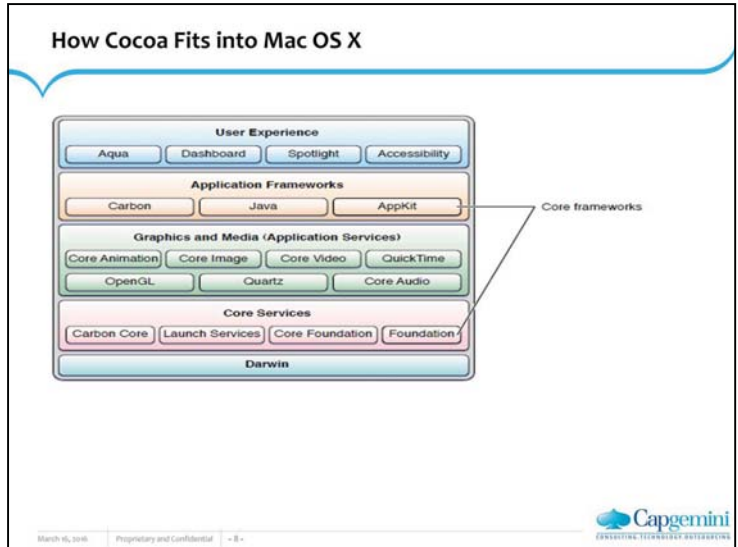
March 16, 2010 Proprietary and Confidential | 77



The most important Cocoa class libraries come packaged in two core frameworks for each platform: Foundation and AppKit for Mac OS X, and Foundation and UIKit for iOS. As with all frameworks, these contain not only a dynamically sharable library (or sometimes several versions of libraries required for backward compatibility), but header files, API documentation, and related resources. The pairing of Foundation with AppKit or UIKit reflects the division of the Cocoa programmatic interfaces into those classes that are not related to a graphical user interface and those that are. For each platform, its two core frameworks are essential to any Cocoa project whose end product is an application. Both platforms additionally support the Core Data framework, which is as important and useful as the core frameworks.

Mac OS X also ships with several other frameworks that publish Cocoa programmatic interfaces, such as the WebKit and Address Book frameworks; more Cocoa frameworks will be added to the operating system over time.

MAC X, iOS and Cocoa Touch Framework Overview



Architecturally, Mac OS X is a series of software layers going from the foundation of Darwin to the various application frameworks and the user experience they support. The intervening layers represent the system software largely (but not entirely) contained in the two major umbrella frameworks, Core Services and Application Services. A component at one layer generally has dependencies on the layer beneath it.

The Cocoa frameworks consist of libraries, APIs, and runtimes that form the development layer for all of Mac OS X. By developing with Cocoa, you will be creating applications the same way Mac OS X itself is created. Your application will automatically inherit the great behaviors and appearances of Mac OS X, with full access to the underlying power of the UNIX operating system. Using Cocoa with the Xcode IDE is simply the best way to create native Mac applications

MAC X, iOS and Cocoa Touch Framework Overview

How Cocoa Fits into Mac OS X

- In Mac OS X, Cocoa has two core Objective-C frameworks :
 - AppKit: It provides the objects an application displays in its user interface and defines the structure for application behavior, including event handling and drawing.
 - Foundation. This framework, in the Core Services layer, defines the basic behavior of objects, establishes mechanisms for their management, and provides objects for primitive data types, collections, and operating-system services.
- Foundation is essentially an object-oriented version of the Core Foundation framework

MAC X, iOS and Cocoa Touch Framework Overview

Cocoa and Cocoa Touch

- Cocoa is commonly referred to as the combination of the Foundation and AppKit frameworks, while Cocoa Touch is the combination of the Foundation and UIKit frameworks.
- Cocoa and Cocoa Touch sit on top of other collections of frameworks to create the API stacks. The other layers are Media, Core Services and Core OS.
- The main difference between Cocoa and Cocoa touch is that the UI classes and APIs aren't the same as Mac OS X, so instead of NSTextField, you have UITextField.
- Many of the classes share the same functionality and can be ported quite easily by simply changing the class name, though most will require some more changes, but usually nothing too heavy.

March 06, 2008 Proprietary and Confidential | 10



How Cocoa Fits into iOS?

➤ The following summarizes some of the frameworks found at each layer of the iOS stack, starting from the foundation layer.

- Core OS : This level contains the kernel, the file system, networking infrastructure, security, power management, and a number of device drivers.
- Core Services : Provide core services, such as string manipulation, collection management, networking, URL utilities, contact management, and preferences. They also provide services based on hardware features of a device, such as the GPS, compass, accelerometer, and gyroscope.
- Media: Provide graphical and multimedia services to the Cocoa Touch layer. They include Core Graphics, Core Text, OpenGL ES, Core Animation, AVFoundation, Core Audio, and video playback.
- Cocoa Touch. The frameworks in this layer directly support applications based in iOS. They include frameworks such as Game Kit, Map Kit, and iAd.

March 06, 2008

Proprietary and Confidential | 11

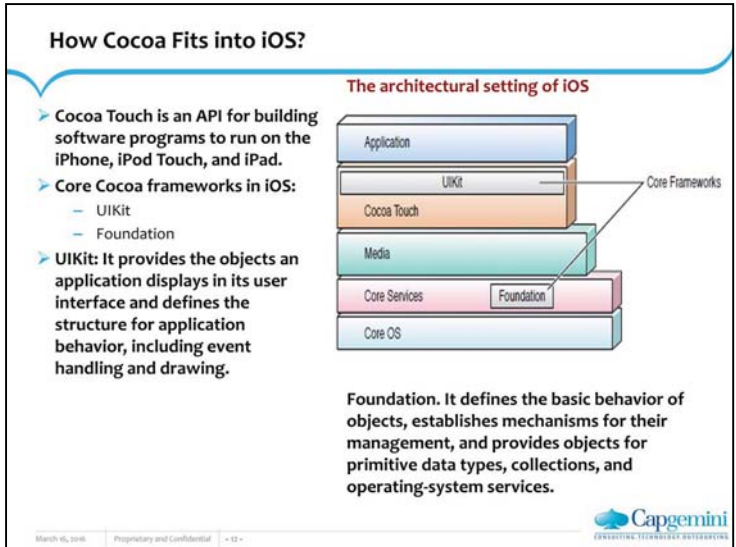


The application-framework layer of iOS is called Cocoa Touch. Generally, the system libraries and frameworks of iOS that ultimately support UIKit are a subset of the libraries and frameworks in Mac OS X.

The following summarizes some of the frameworks found at each layer of the iOS stack, starting from the foundation layer.

- Core OS. This level contains the kernel, the file system, networking infrastructure, security, power management, and a number of device drivers.
- Core Services. The frameworks in this layer provide core services, such as string manipulation, collection management, networking, URL utilities, contact management, and preferences. They also provide services based on hardware features of a device, such as the GPS, compass, accelerometer, and gyroscope. Examples of frameworks in this layer are Core Location, Core Motion, and System Configuration. This layer includes both Foundation and Core Foundation, frameworks that provide abstractions for common data types such as strings and collections. The Core Frameworks layer also contains Core Data, a framework for object graph management and object persistence.
- Media. The frameworks and services in this layer depend on the Core Services layer and provide graphical and multimedia services to the Cocoa Touch layer. They include Core Graphics, Core Text, OpenGL ES, Core Animation, AVFoundation, Core Audio, and video playback.
- Cocoa Touch. The frameworks in this layer directly support applications based in iOS. They include frameworks such as Game Kit, Map Kit, and iAd.

MAC X, iOS and Cocoa Touch Framework Overview



The application-framework layer of iOS is called Cocoa Touch. Generally, the system libraries and frameworks of iOS that ultimately support UIKit are a subset of the libraries and frameworks in Mac OS X.

The following summarizes some of the frameworks found at each layer of the iOS stack, starting from the foundation layer.

- **Core OS.** This level contains the kernel, the file system, networking infrastructure, security, power management, and a number of device drivers. It also has the libSystem library, which supports the POSIX/BSD 4.4/C99 API specifications and includes system-level APIs for many services.
- **Core Services.** The frameworks in this layer provide core services, such as string manipulation, collection management, networking, URL utilities, contact management, and preferences. They also provide services based on hardware features of a device, such as the GPS, compass, accelerometer, and gyroscope. Examples of frameworks in this layer are Core Location, Core Motion, and System Configuration. This layer includes both Foundation and Core Foundation, frameworks that provide abstractions for common data types such as strings and collections. The Core Frameworks layer also contains Core Data, a framework for object graph management and object persistence.
- **Media.** The frameworks and services in this layer depend on the Core Services layer and provide graphical and multimedia services to the Cocoa Touch layer. They include Core Graphics, Core Text, OpenGL ES, Core Animation, AVFoundation, Core Audio, and video playback.
- **Cocoa Touch.** The frameworks in this layer directly support applications based in iOS. They include frameworks such as Game Kit, Map Kit, and iAd.

Frameworks available in iOS-based devices

- `AddressBook.framework 2.0 AB` : Contains functions for accessing the user's contacts database directly.
- `AddressBookUI.framework 2.0 AB` :Contains classes for displaying the system-defined people picker and editor interfaces.
- `AssetsLibrary.framework 4.0 AL`: Contains classes for accessing the user's photos and videos.
- `AudioToolbox.framework 2.0 AU,Audio` :Contains the interfaces for handling audio stream data and for playing and recording audio.
- `AVFoundation.framework 2.2 AV`:Contains Objective-C interfaces for playing and recording audio and video.
- `CFNetwork.framework 2.0 CF`:Contains interfaces for accessing the network via Wi-Fi and cellular radios.

March 16, 2010 Proprietary and Confidential - 19 -



The slide gives some of the frameworks and its use along with available from which version of iOS and prefix for the classes.

Frameworks available in iOS-based devices

- **CoreData.framework 3.0 NS:**Contains interfaces for managing your application's data model.
- **CoreFoundation.framework 2.0 CF:**Provides fundamental software services, including abstractions for common data types, string utilities, collection utilities, resource management, and preferences.
- **CoreGraphics.framework 2.0 CG:**Contains the interfaces for Quartz 2D.
- **CoreImage.framework 5.0 CI:**Contains interfaces for manipulating video and still images.
- **CoreLocation.framework 2.0 CL:**Contains the interfaces for determining the user's location.
- **CoreMedia.framework 4.0 CM:**Contains low-level routines for manipulating audio and video.
- **CoreMotion.framework 4.0 CM :**Contains interfaces for accessing accelerometer and gyro data.

March 16, 2010 Proprietary and Confidential | 14



The slide gives some of the frameworks and its use along with available from which version of iOS and prefix for the classes.

MAC X, iOS and Cocoa Touch Framework Overview

Frameworks available in iOS-based devices

- **CoreTelephony.framework 4.0 CT:** Contains routines for accessing telephony-related information.
- **CoreText.framework 3.2 CT :**Contains a text layout and rendering engine.
- **CoreVideo.framework 4.0 CV :**Contains low-level routines for manipulating audio and video. Do not use this framework directly.
- **EventKit.framework 4.0 EK:**Contains interfaces for accessing a user's calendar event data.
- **EventKitUI.framework 4.0 EK :** Contains classes for displaying the standard system calendar interfaces.
- **Foundation.framework 2.0 NS:**Contains interfaces for managing strings, collections, and other low-level data types.
- **GMKit.framework 3.0 MK:** Contains classes for embedding a map interface into your application and for reverse-geocoding coordinates.

March 16, 2010 Proprietary and Confidential - 15 -



The slide gives some of the frameworks and its use along with available from which version of iOS and prefix for the classes.

MAC X, iOS and Cocoa Touch Framework Overview

Frameworks available in iOS-based devices

- MediaPlayer.framework 2.0 MP: Contains interfaces for playing full-screen video.
- OpenGLES.framework 2.0 EAGL, GL : Contains the interfaces for OpenGL ES, which is an embedded version of the OpenGL cross-platform 2D and 3D graphics rendering library.
- QuartzCore.framework 2.0 CA: Contains the Core Animation interfaces.
- System Configuration.framework 2.0 SC: Contains interfaces for determining the network configuration of a device.
- UIKit.framework 2.0 UI: Contains classes and methods for the iOS application user interface layer.

March 16, 2010 Proprietary and Confidential | 16 |



The slide gives some of the frameworks and its use along with available from which version of iOS and prefix for the classes.

MAC X, iOS and Cocoa Touch Framework Overview

How Cocoa Fits into iOS?

- Demo on developing simple helloworld application using cocoa touch



March 16, 2010 Proprietary and Confidential - 17 -



Add the notes here.

MAC X, iOS and Cocoa Touch Framework Overview

Summary

➤ The topics covered..

- What Is Cocoa?
- Cocoa History
- Cocoa class libraries
- How Cocoa Fits into Mac OS X
- Cocoa and Cocoa Touch
- How Cocoa Fits into iOS?
- Frameworks available in iOS-based devices



March 16, 2010 Proprietary and Confidential | 18 |

 **Capgemini**
ENJOYING THE BEST OF BOTH WORLDS

Add the notes here.

MAC X, iOS and Cocoa Touch Framework Overview

Review Question

- **Question 1: Cocoa consists of a suite of object-oriented,,, and an integrated development environment.**
- **Question 2: Following core frameworks are used for iOS application development.**
 - a) Foundation b) AppKit c)UIKit d)Core Foundation
- **Question 3: Which framework Contains interfaces for accessing accelerometer and gyro data.**
 - a)CoreMotion.framework
 - b)CoreLocation.framework
 - c)CoreFoundation.framework

