Cocoa Core Competencies

[원본 링크]

문서 읽기 순서

- 빨간색
  - 초보에게 꼭 필요한 개념입니다. 이해하기 쉬운 편입니다.
- 녹색
  - 초보에게 꼭 필요한 개념입니다. 이해하기 어려운 편입니다.
- 주황색
  - 특정 상황에만 필요하거나 초보에게 당장은 필요하지는 않은 개념입니다. 이해하기 쉬운 편입니다.
- 파란색
  - 특정 상황에만 필요하거나 초보에게 당장은 필요하지는 않은 개념입니다. 이해하기 어려운 편입니다.
- 갈색
  - 지금 당장 볼 필요는 없고, 필요한 경우에 찾아올 수 있도록 이런 내용이 있다는 것만 알아두면 됩니다.

Accessibility

> 접근성이 있는 애플리케이션은 장애와 관계없이 모든 사람이 사용할 수 있다고 설명합니다. 시각, 청각, 상호작용 또는 학습 장애가 있는 사람의 관점에서 앱의 사용자 경험을 고려하라고 설명합니다.
> **파트 2 H.I.G. 에서 소개하는 내용입니다.**

- Working with VoiceOver
  - 시각장애인 사용자는 운영체제에 내장된 화면 읽기 기술인 `VoiceOver` 를 사용하여 애플리케이션과 상호작용 하는 부분을 설명합니다.

Accessor method

> 접근자 메서드는 객체의 프로퍼티값을 가져오거나 설정하는 인스턴스 메서드 입니다. 코코아의 용어에서 객체의 프로퍼티 값을 가져오는 메서드를 `getter` 메서드 또는 `getter` 라고 하고 객체의 프로퍼티 값을 변경하는 메서드

> 를 `setter` 메서드 또는 `setter` 라고 합니다. 그리고 접근자 메소드가 제공하는 두 가지 이점을 설명합니다.

- Naming Conventions
  - 이름짓기 규칙에 관해 설명합니다.

App ID

> 애플리케이션 ID는 하나 이상의 애플리케이션을 식별하는데 사용되는 부분으로 구성된 문자열입니다. 문자열은 `팀 ID(Team ID)` 와 `번들 ID(Bundle ID search string)` 로 구성되며, `팀 ID(Team ID)` 와 `번들 ID(Bundle ID search string)` 에 관해 설명 합니다.

- An Explicit App ID Matches a Single App
  - 애플리케이션 ID의 일치해야 하는 경우를 설명합니다.
- Wildcard App IDs Match Multiple Apps
  - `팀 ID(Team ID)` , `번들 ID(Bundle ID search string)` 의 마지막 부분으로 별표(*) 문자를 포함하는 경우를 설명합니다.

Application Code Signing

> 애플리케이션에 서명하면 시스템에서 애플리케이션에 서명한 사람을 식별하고 서명 된 이후 수정되지 않았음을 확인할 수 있습니다. 앱 스토어에 제출하는 데 필요합니다. 이 챕터는 서명에 관해 설명합니다.

Block object

> 블록 객체를 선언하는 방법과 사용하는 방법에 관해 설명합니다.
> Objective-C 언어의 블록에 관한 설명입니다. Swift 언어에는 블록을 대체할 클로저(Closure)가 있습니다. Swift 언어의 클로저 문서를 참고하세요.

- Declaring a Block
  - 블록을 선언하는 방법을 설명합니다.
- Creating a Block
  - `(^)` 연산자를 사용해 블록을 만드는 방법을 설명합니다.
- Block-Mutable Variables
  - `__block` 에 관해 설명합니다.
- Using Blocks

    - 블록을 사용하는 방법을 설명합니다.

- Comparison Operations
    - 코코아 환경에서 블록을 사용하여 일반적으로 작업하는 내용을 설명합니다.

Bundle

> 번들은 이미지와 사운드 같은 리소스들을 한곳에 그룹화하는 파일 시스템의 디렉터리 입니다. 번들을 이용해 리소스들을 관리하는 방법에 관해 설명합니다.

- Structure and Content of Bundles
    - iOS, macOS 번들의 구조와 내용에 관해서 설명합니다.
- Accessing Bundle Resources
    - 애플리케이션을 시작하면 즉시 필요로 하는 번들 리소스들을 메모리로 로드하는 방법을 설명합니다.
- Loadable Bundles
    - 로드 가능한 번들을 이용해 여러 방법으로 애플리케이션의 설계를 할 수 있다고 설명합니다.

Category

> 카테고리를 사용해서 기존 클래스에 추가 메서드를 서브 클래싱 필요 없이 정의를 할 수 있습니다. 카테고리를 이용해 수행할 수 있는 부분을 설명합니다.
> Objective-C 언어의 카테고리에 관한 설명입니다. Swift 언어에는 카테고리를 대체할 익스텐션(Extension)이 있습니다. Swift 언어의 익스텐션 문서를 참고하세요.

- Declaration
    - 카테고리 인터페이스의 선언 방법을 설명합니다.
- Implementation
    - 카테고리 구현 방법을 설명합니다.

Class cluster

> 클래스 클러스터 아키텍처에 관해 설명하고 클래스트 클러스트를 이점과 고려해야 할 사항에 관해 설명합니다.

- Benefits
    - 클래스 클러스터의 이점에 관해 설명합니다.

- Considerations
    - 클래스 클러스터의 고려해야할 사항에 관해 설명합니다.

Class definition

> 클래스의 정의와 구현에 관해 설명합니다.
> Objective-C 언어의 클래스 관한 설명입니다. Swift 언어의 클래스는 Swift 언어의 클래스 문서를 참고하세요.

- Interface
    - 클래스 인터페이스에 관해 설명하고 어떻게 동작하는지 설명합니다.
- Implementation
    - 클래스를 구현하는 방법에 관해 설명합니다.

Class method

> 클래스 메서드의 선언 및 구현에 관해 설명합니다. 그리고 클래스 메서드의 선언 및 구현부분의 시작 부분에 `(+)` 기호로 표시합니다.
> Objective-C 언어의 메서드 관한 설명입니다. Swift 언어의 메서드는 Swift 언어의 메서드 문서를 참고하세요.

- Subclasses
    - 서브 클래스에 관해 설명합니다.
- Instance Variables
    - 인스턴스 변수에 관해 설명합니다.
- self
    - 클래스 메서드 내에서 `self` 에 관해 설명합니다.

Cocoa (Touch)

> 코코아 및 코코아 터치 프레임워크를 설명합니다.
> **파트 1에서 소개합니다.**

- The Frameworks
    - 여러 프레임워크에 관해 설명합니다.
- The Language

- 코코아 및 코코아 터치에서 애플리케이션을 개발할 때 사용하는 언어에 설명합니다.

Coding conventions

> 코딩 규칙은 API 사용의 효율성과 일관성을 보장하는 데 도움이 되는 일련의 지침입니다. 코딩 규칙을 사용하는 방법을 설명합니다.
> Objective-C 언어의 코딩 컨벤션에 관한 설명입니다. Swift 언어의 코딩 컨벤션은 Swift 언어의 API Design Guidelines 문서를 참고하세요.

Collection

> 객체를 저장하는 것이 주된 역할인 컬렉션 (파운데이션 프레임워크 객체)에 관해 설명합니다.
> Objective-C 언어의 콜렉션 타입에 관한 설명입니다. Swift 언어의 콜렉션 타입은 Swift 언어의 콜렉션 타입 문서를 참고하세요.

- Collection Classes
  - 컬렉션 클래스의 종류와 여러 기능에 관해 설명합니다.
- Ordering Schemes
  - `NSArray`, `NSSet`, `NSDictionary`에 관해 설명합니다.

Controller object

> 컨트롤러 객체는 하나 이상의 뷰 객체와 하나 이상의 모델객체 사이의 중계자 역할을 합니다. 코코아 프레임워크의 세 가지 기본 컨트롤러 유형에 관해 설명합니다.
> **파트 1에서 소개합니다.**

- Coordinating Controllers
  - 코디네이팅 컨트롤러의 기능을 설명합니다.
- View Controllers
  - UIKit 및 AppKit 프레임워크의 뷰 컨트롤러 클래스가 서로 다른 특성을 가진것에 관해 설명합니다.
- Mediating Controllers (macOS)
  - 조정 컨트롤러의 기능에 관해 설명합니다.

Declared property

> 프로퍼티를 선언을 시작할 때는 `@property` 키워드롤 사용합니다. 그리고 프로퍼티를 선언하는 다른 방법에 관해 설명합니다.
> Objective-C 언어의 프로퍼티에 관한 설명입니다. Swift 언어의 프로퍼티는 Swift 언어의 프로퍼티 문서를 참고하세요.

Delegation

> 델리게이트는 한 객체가 다른 객체를 대신하여 동작하는 간단하고 강력한 패턴입니다. 코코아 프레임워크 내에서 델리게이트 객체가 어떻게 사용되는지 설명합니다.
> **파트 2에서 소개합니다.**

- Delegation and the Cocoa Frameworks
  - 코코아 프레임워크에서 델리게이트 객체의 예를 설명합니다.
- Delegation and Notifications
  - 델리게이트와 옵저버(관찰자)의 관계를 설명합니다.
- Data Source
  - 데이터 소스는 델리게이트와 거의 같지만 차이점이 있습니다. 그 차이점에 관해 설명합니다.

Dynamic binding

> 동적 바인딩은 컴파일시간이 아닌 실행시점에 호출할 메서드를 결정합니다. Objective-C에서는 모든 메서드가 실행시점에 동적으로 확인됩니다. 동적 바인딩의 간단한 예제를 보여주며 설명합니다.

Dynamic typing

> Objective-C는 `id` 데이터 형식을 사용하여 객체 종류를 지정하지 않고 변수를 나타냅니다. 이를 동적 타이핑이라고 합니다. 정적과 동적 간의 차이점과 간단한 예제를 보여주며 설명합니다.

- The isa Pointer
  - `isa` 인스턴스 변수에 관해 설명합니다.

Enumeration

> 열거에 관해 설명하고 사용하는 방법을 설명합니다.
> Objective-C 언어의 열거형에 관한 설명입니다. Swift 언어의 열거형은 Swift 언어의 열거형 문서를 참고하세요.

- NSEnumerator

    - `NSEnumerator`의 예제를 보면서 설명합니다.
- Fast Enumeration
    - 빠른 열거의 예제를 보면서 설명합니다.

Exception handling

> 예외처리는 프로그램 실행의 정상적인 흐름을 방해하는 비정상적인 이벤트를 관리하는 프로세스입니다. 예외 유형과 예외처리 지시문에 관해 설명합니다.
> Objective-C 언어의 오류처리에 관한 설명입니다. Swift 언어의 오류처리는 Swift 언어의 오류처리 문서를 참고하세요.

- Types of Exception
    - 예외가 발생하는 유형을 설명합니다.
- Handling Exceptions Using Compiler Directives
    - 예외처리를 지원하는 컴파일러 지시자를 설명합니다. (`@try` , `@cathc()` , `@finally` , `@throw`)
- Signaling Errors
    - 보통 프로그래밍 흐름을 제어하거나 오류를 나타내기 위해 예외처리를 많이 사용하지만 코코아 프레이워크 환경에서는 메서드의 리턴값을 사용을 권장한다고 설명합니다.

Framework

> 프레임워크는 nib파일, 이미지 파일, 코드 파일과 같은 관련 리소스와 함께 동적 공유 라이브러리를 포함하는 번들(구조화된 디렉터리)입니다. 프레임워크의 동작 구조에 관해 설명합니다.

Information property list

> 정보 프로퍼티 리스트는 애플리케이션 또는 다른 번들 실행파일의 세부정보를 지정하는 구조화된 텍스트입니다.키-값을 이용해 표현을 하고 파일의 이름은 `Info.plist`로 이루어져 있습니다. 그리고 간단한 예제를 통해 설명하고 있습니다.

Initialization

> 초기화는 새롭게 할당된 객체를 상태를 적절한 초기값으로 설정하여 사용 가능하게 만드는 객체생성 단계 입니다. 초기화의 선언 형식과 구현하는 방법을 설명합니다.
> Objective-C 언어의 인스턴스 초기화에 관한 설명입니다. Swift 언어의 인스턴스 초기화는 Swift 언어의 인스턴스 초기화 문서를 참고하세요.

- The Form of an Initializer Declaration
    - 규칙에 따라 이니셜라이저 이름은 항상 `init` 으로 시작한다고 설명합니다.
- Implementing an Initializer
    - 이니셜라이저 메서드를 구현하기 위한 기본단계를 설명합니다.

Internationalization

> 국제화는 하드웨어 변경 없이 다양한 언어 및 지역에 적용 할 수 있도록 소프트웨어 애플리케이션을 설계하는 프로세스라고 설명합니다. 코코아 프레임워크에서 애플리케이션 국제화를 위한 여러 아키텍처를 제공한다고 설명합니다.

Introspection

> 내부 특성은 객체에 특정 메시지를 보내면 객체에 대한 질문을 Objective-C 실행시점에서 사용자에게 제공할 수 있습니다. 내부 특성은 프로그램을 효율적이고 강력하게 만들어 주기 때문에 중요한 코딩 도구라고 설명합니다. Objective-C 언어의 특성에 관한 설명입니다. Swift 언어에서는 타입캐스팅으로 유사하게 구현할 수 있습니다. Swift 언어의 타입캐스팅 문서를 참고하세요.

- Types of Introspection Information
    - 내부 특성의 유형에 관해 설명합니다.

Key-value coding

> 키-값 코딩(KVC)은 문자열 식별자를 이용해 객체의 프로프터 및 관계에 간접적으로 액세스 하기 위한 매커니즘 이라고 설명합니다. 키-값 코딩을 준수하는 클래스를 만드는 방법도 설명합니다.

- Object Properties and KVC
    - 객체의 프로퍼티와 키-값 코딩의 관계에 관해 설명합니다.
- Making a Class KVC Compliant
    - 키-값 키딩을 준수하는 클래스를 만드는 방법을 설명합니다.

Key-value observing

> 키-값 옵저빙(KVO)은 다른 객체의 프로퍼티가 변경 될때 객체에 직접 통보 할 수 있는 매커니즘 이라고 설명합니다. KVO를 구현하는 방법을 설명합니다.

- Implementing KVO

    - KVO를 구현하기 위한 방법을 설명합니다.
- KVO Is an Integral Part of Bindings (macOS)
    - 코코아 바인딩 기술을 설명합니다.

Memory management

> 메모리 관리는 객체의 수명주기를 관리하고 더이상 필요 없는 객체를 해제하는 프로그래밍 분야라고 설명합니다. 메모리를 관리하는 규칙과 메모리를 관리하면서 필요한 개념을 설명합니다.
> Objective-C 언어에서의 메모리 관리에 관한 설명입니다. Swift 언어에서의 메모리 관리는 Swift 언어의 ARC 문서를 참고하세요.

- Memory-Management Rules
    - 메모리를 관리하는 데 도움이 되는 규칙을 설명합니다.
- Aspects of Memory Management
    - 메모리를 관리하는데 도움이 되는 개념을 설명합니다.

Message

> 메시지가 하는 역할과 개념을 설명합니다.
> Objective-C 언어의 메세지에 관한 설명입니다. Swift 언어의 메서드 문서를 참고하세요.

Method overriding

> 메서드를 재정의할 때 주의할 점과 간단한 예제를 통해 설명합니다.
> Objective-C 언어의 오버라이딩에 관한 설명입니다. Swift 언어의 오버라이딩은 Swift 언어의 상속 문서를 참고하세요.

Model object

> 모델 객체의 개념에 관해 설명하고 모델 클래스를 설계 시 고려해야 할 점을 알려줍니다.
> **파트 1에서 소개합니다.**

- A Well-Designed Model Class
    - 모델 클래스를 설계할시 고려해야 할 사항을 설명합니다.

Model-View-Controller

> MVC 디자인 패턴에 관해 설명하고 각각의 객체 (모델, 뷰, 컨트롤러)의 역할을 설명합니다.
> **파트 1에서 소개합니다.**

- Model Objects
    - 모델객체가 애플리케이션에서의 역할을 설명합니다.
- View Objects
    - 뷰 객체가 애플리케이션에서의 역할을 설명합니다.
- Controller Objects
    - 컨트롤러 객체가 애플리케이션에서의 역할을 설명합니다.

Multiple initializers

> 다중 이니셜라이저를 정의하는 방법과 편의성에 관해 설명합니다.
> Objective-C 언어의 초기화 메서드 관한 설명입니다. Swift 언어의 초기화 메서드는 Swift 언어의 초기화 문서를 참고하세요.

- The Designated Initializer
    - 지정된 이니셜라이저를 호출하는 방식을 설명합니다.

Nib file

> Nib파일은 iOS, macOS 애플리케이션의 사용자 인터페이스를 저장하는데 사용하는 특별한 유형의 리소스 파일이라고 설명합니다.

Notification

> 노티피케이션은 하나 이상의 옵저빙(관찰)객체에 보내는 메시지라고 설명합니다. 노티피케이션을 옵저빙(관찰)하는 방법과 노티피 케이션 객체에 관해 설명합니다.
> **파트 5에서 소개합니다.**

- The Notification Object
    - 노티피케이션 객체에 관해 설명합니다.
- Observing a Notification
    - 노티피케이션 옵저빙(관찰) 예제로 설명합니다.
- Posting a Notification

- 노티피케이션을 게시하기 전에 정의하는 방법을 설명합니다.

Object archiving

> 아카이빙은 객체 그룹을 애플리케이션 간에 저장하거나 전송할 수 있는 양식으로 변환하는 프로세스라고 설명합니다.

- Keyed and Sequential Archivers
  - 아카이브 클래스의 두 가지 방법을 설명합니다.
- Creating and Decoding Keyed Archives
  - 아카이브를 생성하고 디코딩 하는 방법을 설명합니다.

Object comparison

> 객체를 비교할 시 각 클래스별 비교 논리를 구현하여 해당 인스턴스의 동등성을 결정한다고 설명합니다. 파운데이션 프레임워크의 일부 클래스에서는 다른 방식의 비교메서드를 구현해서 사용한다고 설명합니다.
> Objective-C 언어의 객체비교에 관한 설명입니다. Swift 언어의 기본 연산자 문서의 Comparison Operators를 참고하세요.

- Implementing Comparison Logic
  - `isEqual` 메서드를 재정의해서 구현하는 방법을 예제로 설명합니다.

Object copying

> 객체를 복사를 해야 하는 이유와 객체 복사를 이용해 메모리상의 이점이 무엇인지에 관해 설명합니다.
> Objective-C 언어의 객체 복사에 관한 설명입니다. Swift 언어의 메모리 안전 문서도 참고해 볼 수 있습니다.

- Requirements for Object Copying
  - 객체 복사를 하려는 요구사항을 설명합니다.
- Memory-Management Implications
  - 객체 복사를 사용 시 메모리 관리에 관해 설명합니다.

Object creation

> 객체가 메모리에 할당되는 과정을 설명하고 객체 생성을 표현하는 형식과 메모리 관리 시점에 관해 설명합니다.
> Objective-C 언어의 객체 생성과정에 관한 설명입니다. Swift 언어의 인스턴스 초기화 문서를 참고하세요.

- The Form of an Object-Creation Expression

    - 객체 생성을 표현하는 형식을 설명합니다.
- Memory-Management Implications
    - 객체를 메모리상에서 관리하는 시점을 설명합니다.
- Factory Methods
    - 클래스 메서드에 관해 설명합니다.

Object encoding

> 객체를 인코딩 및 디코딩하는 방법을 설명하고 인코딩한 객체를 한 프로세스에서 다른 프로세스로 전송하는 방법을 설명합니다.

- How to Encode and Decode Objects
    - `NSCoding` 프로토콜을 이용해 인코딩 및 디코딩하는 방법을 설명합니다.
- Keyed Versus Sequential Archiving
    - `NSKeyedArchiver, NSKeyedUnarchiver` 와 `NSArchiver, NSUnarchiver` 에 관해 비교 설명합니다.

Object graph

> 객체 그래프라는 용어에 관해 설명합니다.

Object life cycle

> 객체가 생성되고 소멸할 때까지의 과정을 설명합니다.
> Objective-C 언어의 객체 생성과 소멸과정에 관한 설명입니다. Swift 언어의 인스턴스 초기화 문서와 Swift 언어의 인스턴스 소멸 문서를 참고하세요.

Object modeling

> 객체 또는 클래스를 설계하는 방법을 설명합니다.

Object mutability

> 객체의 값이 변경할 수 있는 경우와 변경할 수 없는 경우에 관해 설명합니다.

- Receiving Mutable Objects

    - 메서드를 호출하고 반환되는 객체를 변경할 수 있는지에 관해 설명합니다.

- Storing Mutable Objects
    - 서브 클래스를 설계 시 객체의 값이 변경되어야 하는 상황과 변경되지 말아야 할 상황을 비교 설명합니다.

Object ownership

> 객체의 소유권의 개념을 설명하고 강한 참조(strong reference), 약한 참조(weak references)에 관해 설명합니다. Objective-C 언어에서의 메모리 관리에 관한 설명입니다. Swift 언어에서의 메모리 관리는 Swift 언어의 ARC 문서를 참고하세요.

Objective-C

> Objective-C의 언어에 관해 설명합니다.

Property list

> 프로퍼티 리스트의 개념과 여러 유형에 관해 설명합니다. 프로퍼티 리스트를 사용 시 효율성에 관해서도 설명합니다.

- Property List Types and Objects
    - 프로퍼티 리스트에서 사용 유형에 관해 설명합니다.
- Best Practices for Property Lists
    - 프로퍼티 리스트를 사용함으로 효율적인 면에 관해 설명합니다.
- Best Practices for Property Lists
    - 프로퍼티 리스트를 직렬화 및 비 직렬화를 하는 방법을 설명합니다.

Protocol

> 프로토콜의 개념과 프로토콜의 종류에 관해 설명합니다.
> Objective-C 언어에서의 프로토콜에 관한 설명입니다. Swift 언어에서의 프로토콜은 Swift 언어의 프로토콜 문서를 참고하세요.

- Formal and Informal Protocols
    - 두 종류의 프로토콜에 관해 설명합니다.
- Adopting and Conforming to a Formal Protocol

    - 프로토콜 규약을 채택하는 방법을 설명합니다.
- Creating Your Own Protocol
    - 자신만의 프로토콜을 만들 수 있다고 설명합니다.

Root class

> 루트 클래스의 개념을 설명하고 Objective-C의 모든 클래스의 루트 클래스인 `NSObject` 에 관해서도 설명합니다.

Selector

> 셀렉터의 기본 개념과 셀렉터가 하는 역할에 관해 설명합니다. 셀렉터를 사용하는 방법도 설명합니다.

- Getting a Selector
    - 셀렉터를 가져오는 방법을 설명합니다.
- Using a Selector
    - 셀렉터를 사용하는 방법을 설명합니다.

Singleton

> 싱글톤 클래스의 개념과 사용 시 유의할 점에 관해 설명합니다.
> **파트 2에서 소개합니다.**

Uniform Type Identifier

> UTI의 개념과 다른 유형에 비해 몇 가지 장점을 소개하고 규칙이나 구조에 관해 설명합니다.

- UTIs Use the Reverse Domain Name System Convention
    - UTI의 도메인 이름 규칙을 설명합니다.
- Uniform Type Identifiers Are Declared in a Conformance Hierarchy
    - UTI계층 구조를 설명합니다.
- macOS Applications Add New UTIs by Defining Them in an Application Bundle
    - macOS 애플리케이션에서 UTI를 정의, 추가하는 방법을 설명합니다.

Value object

> 값 객체의 종류와 특성에 관해 설명합니다.
> Objective-C 언어에서의 값 타입에 관한 설명입니다. Swift 언어에서의 값 타입에 관한 설명은 Swift 언어의 클래스와 구조체 문서를 참고하세요.

- NSValue
  - `NSValue` 에 관해 설명합니다.