수요일 (09.28)

- [x] ~~TIL 만들기~~
- [x] ~~좋은 동료가 되기 위한 방법 `hue` 님 의견 정리~~

☑️ App Lifecycle

- 앱의 생명주기
- App launching 부터 App termination 까지

앱의 상태가 변경될 때 UIKit 은 적절한 Delegate 메서드를 호출한다

→ 앱의 상태에 따라 앱 작동의 제약이 생긴다. 상태마다 어떤 제약이 생기는지 제대로 알고 있자.

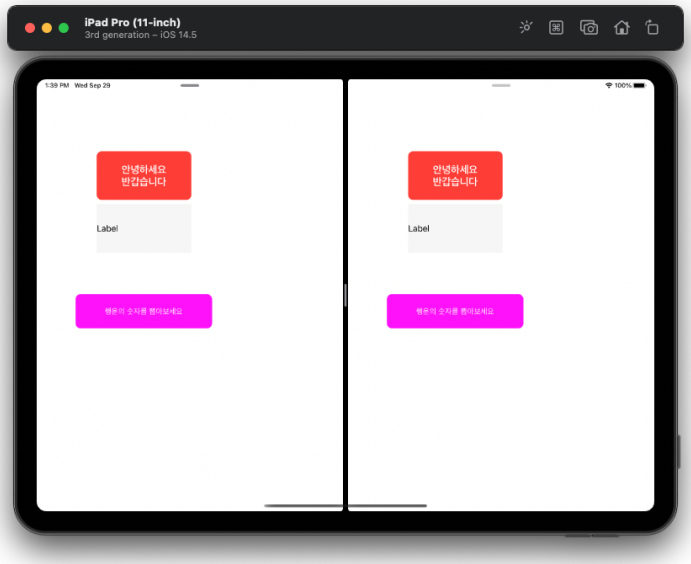
→ iOS 13 이상 버전 부터는 UISceneDelegate 오브젝트의 메서드를 우선 호출한다.

- iOS 12 이하 버전은 UIAppDelegate 의 메서드를 호출한다.

좀 더 자세히 🔎

- 하나의 Scene 은 기기에서 실행되고 있는 하나의 앱 UI 인스턴스를 가리킨다.
- iOS 13 이상을 지원하는 앱은 기본적으로 `Scene-based` 앱이다.
- info.plist 에 `Application Scene Manifest` 키를 찾을 수 있다. (문서 링크)
- 이 키의 내부 키중 `Enable Multiple Windows` 값을 조정하여 `Scene-support` 앱을 만들 수 있다. (선택사항, 해줘도 되고 안해줘도 된다.)
  - 이 때 iPhone 만 지원되는 앱에서 이 값의 설정은 의미가 없다. (확인해 봄)
- `Scene-support` 앱은 하나의 기기에서 같은 앱의 UI 인스턴스를 여러 개 생성할 수 있다. (iPhone 은 안댐)
  - 이 때, 두 씬이 동시에 공유 자원에 접근할 때 발생할 수 있는 문제를 처리해주어야 한다. (race condition)

        - serial queue 등을 사용하여 동기화
    - **Scene 이라는 개념을 구체화한 타입은 `UIWindowScene` 클래스 이다.**
    - 하나의 UIApplication 인스턴스 위에 N 개의 UIWindowScene 인스턴스가 존재하면서 동작하는 것 같다. (이 때 N은 씬의 개수)



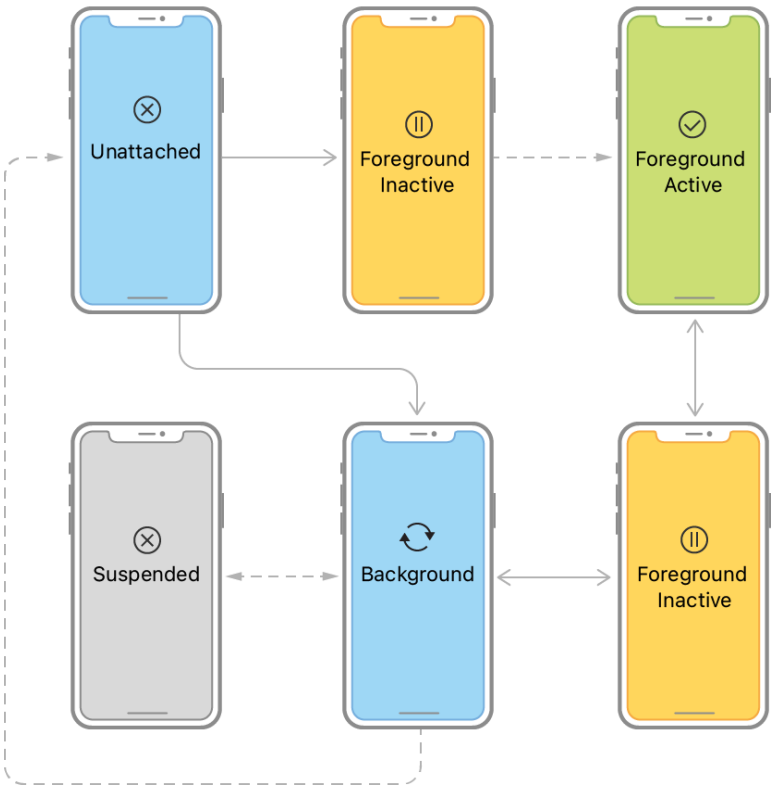
- `Scene Support` 에 대해 간단히 작성한 글이 있다 (블로그 링크)

- 앱이 Launching 되면 UIKit 은 우선 새로운 Scene 인스턴스를 생성하고 `Unattached` 상태로 만든다.
    - 또한 하나의 씬은 하나의 씬 세션 (UISecenSession) 과 연결된다.
    - 씬 세션은 씬을 추적하고, 씬의 id 와 configuration 상세 정보를 담고 있는 객체이다.
    - 씬 세션은 직접 생성하지 않고, UIKit 이 사용자 상호작용에 반응하여 만들어준다.
    - 씬 세션은 씬이 종료될 때 같이 해제된다. (보통 앱 스위처에서 지울 때)

- 시스템이 시작한 앱은 `Background` 상태로 진입하고, 구현된 작업을 한다. (예) 위치 변경에 따른 처리)
- 사용자가 시작한 앱은 `Foreground - Inactive` 상태로 진입한 후, 곧바로 Active 상태에 진입한다.

- Background 와 Foreground 의 가장 큰 차이는 화면에 보이는지
  - 시스템 자원 활용 제약 또한 차이가 있다.
  - 구체적인 수치를 알고 싶다!!!!!!!!!

- 사용자가 앱 UI 를 Dismiss 하면, (종료과 구분 !!! 홈 화면으로 이동하거나 앱 스위처를 통해 다른 앱을 화면으로 가져올 때 Dismiss 라 표현하는 것 같음)
  - Dismiss 된 씬은 backgorund 상태로 만든 후 결국 suspended 상태가 된다.
  - UIKit 은 위 두 가지 상태의 씬을 `Unattached` 상태로 만들어 연결 해제 할 수 있다. (시스템 자원 회수를 위해서)
  - `unattached` 상태는 앱과 연결되어있지 않은 상태인 것 같음



관련 문서

1. 씬이 `foreground-active` 상태에 진입하기 전에 준비해야 할 것들 : UI 업데이트, 사용자 상호작용 준비 (네트워크 서비스, 리소스 가져오기 등)

   → sceneWillEnterForeground(_:) 이후 sceneDidBecomeActive(_:) 가 호출된다.

    - 새로 생성된 씬이나, 기존에 연결되어 있던 씬 모두의 경우에서 호출된다. 링크

2. 씬이 `foreground-active` 에서 벗어날 때 해야할 동작들: 데이터를 저장하고, 중요한 작업을 마치고 가능한한 많은 메모리를 해제한다. 왜? 메모리 경고 받으면 강제종료 됨. 링크

3. UISceneDelegate 가 모든 앱 생명주기 관련 이벤트를 처리하지는 않는다.

   나머지 부분은 UIApplicationDelegate 메서드를 구현해야한다. 링크

Life-cycle 이외의 중요한 이벤트

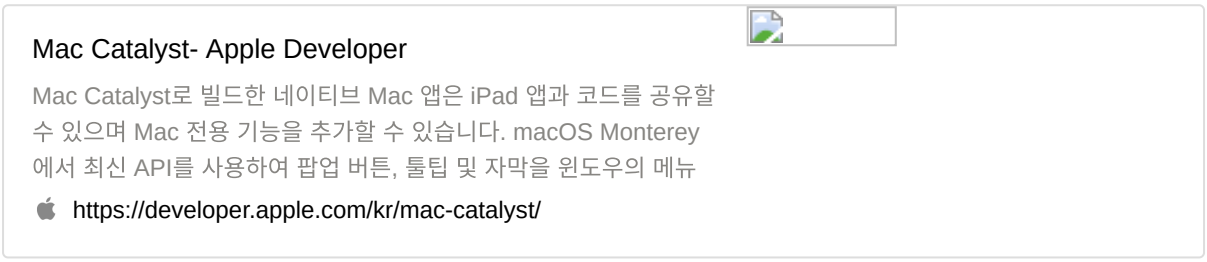
`UIApplicationDelegate` 에서 처리한다.

Memory warnings, Proteted data become available(사용자가 휴대폰을 잠금하거나 잠금해제했을 때)

Handoff tasks, Time Changes, Open URLs (실행중인 앱이 앱이 다른 자원을 열어보려고 할 때)

☑️ Mac Catalyst

iPadOS 로 개발한 코드 베이스를 사용하여 네이티브 Mac 앱을 빌드해주는 도구



☑️ Cocoa Touch Framework

iOS, iPadOS, watchOS, tvOS 에서 실행되는 앱을 빌드하기 위/한 개발 환경 (Foundation + UIKit)

네이티브 언어 : Objective-C / Swift

Foundation

- 기본적으로 사용하는 데이터 형식 (Numbers, Data 등), 앱의 기본 객체와 기반 기술을 제공하는 역할
- 파일 및 데이터 관리, 네트워킹 시스템 등이 포함

UIKit

유저 인터페이스 오브젝트들을 제공

UIKit 의 핵심 객체들은 Objective-C 로 구현되어 있다.

→ 스위프트는 Objective-C API에 대한 완전한 접근 권한이 제공되므로 적용이 용이하다.

SwiftUI

모든 Apple 플랫폼 개발 가능한 UI 개발 도구 (iOS 13 이상)

→ Combine 과 더불어 언젠가 주력으로 사용해보고 싶다.

✅ WWDC21: Build interfaces with style

영상 링크

Xcode 13 부터

새로운 버튼 스타일들이 생겼다.

CornerRadius 도 Predefined 된 여러 옵션들이 생겼다.

Symbol Multiper Layer

자세한 내용은 "What's new in SF Symbols" 비디오 보기

RenderMode

- Heirarchical : 메인 레이어의 색을 하나 정하면 나머지 서브레이어는 알파값 조정으로 뎁스가 나타난다.
- Palette : 심볼 내부를 여러 레이어로 나누어 색을 다양하게 할 수 있다.

☑️ UIViewController

- 앱의 뷰 계층의 일부를 관리 하는 객체
- 기본적으로 MVC 패턴에서 View 와 Model 사이의 상호작용을 중재하는 역할을 한다.
- ViewController 의 역할은 두가지로 나뉜다. (관련 글 링크)
  - Container View Controller
  - Content View Controller
- 이 클래스를 상속한 여러 타입의 서브클래스들이 있다.
  - UITabBarController, UINavigationController
  - UITableViewController, UICollectionViewController 등
- 기본적으로 하나의 최상위 뷰를 가지고 있다. (Content View)

☑️ UIView

- 기본적으로 컨텐츠를 보여주는 역할을 한다.
  - 컨텐츠는 주로 모델 레이어에서 가져온다.
- ViewController 가 보통 사용자 인터렉션 처리를 Delegate 해준다.
- ViewController 의 최상위 뷰 아래 계층에 추가되어 사용된다.
- 이 클래스를 상속한 다양한 종류의 뷰들이 있다.
  - UILabel, UITextField, UITextView, UIImageView

☑️ Utility

SFSymbol = 샌프란시스코 심볼

유용한 사이트

- https://feathericons.com/
- https://colorhunt.co/
- https://www.palettable.io/C1E3D0
- https://icons8.com/illustrations

- https://appicon.co/

Image Asset 의 App Icon

→ iPhone 용 앱이라도 iPad 에서 돌아가기 때문에 iPad 용 이미지도 모두 채워넣어주어야 한다.

@available, #available

```
@available(iOS 13.0, *)

#available(iOS 13.0, *)

// * 는 왼쪽 플랫폼 버전 이상의 모든 미래의 버전을 포함할 것이라는 의미
```

→ `@available` 을 입력한 함수, 클래스 , 프로토콜 앞에 붙음. 플랫폼 지원 버전을 컴파일 타임에 선언할 수 있다.

→ `#availiable` 은 런타임에 구동기기의 플랫폼을 확인하는 기능, Bool 타입으로 반환

글로벌 앱의 경우 현지 상황 (예: 통신환경이 좋지 않은 곳) 에 따라 용량을 줄이기 위해 코드로만 UI 를 구현하는 방향으로 많이 적용하는 경우가 있다.

좋은 동료

커뮤니케이션이 가능해야 한다.

서비스를 만들 때 어떤 기술 스택을 사용할지에 대한 커뮤니케이션이 필수적이다.

이 때 가장 익숙한 기술을 사용할지, 트렌디하고 효율적인 기술을 적용할지 등 여러 관점에서 논의가 된다.

각각의 기술의 대한 경험이 있고, 기술적 특성에 대해 잘 이해하고 있으면 팀원들을 이해시키고 설득할 수 있는 좋은 동료가 될 수 있다.

→ 역시 배경지식, 경험과 그것을 논리적으로 말할 수 있는 기술이 필요하다.