Week 1.
iOS Seminar

iOS 기초, Xcode 사용과 UI 구성하기, 화면 전환 및 데이터 전달 기초

이번 세미나에서 다룰 주제는요!

01 iOS 기초

02 Xcode 사용하기

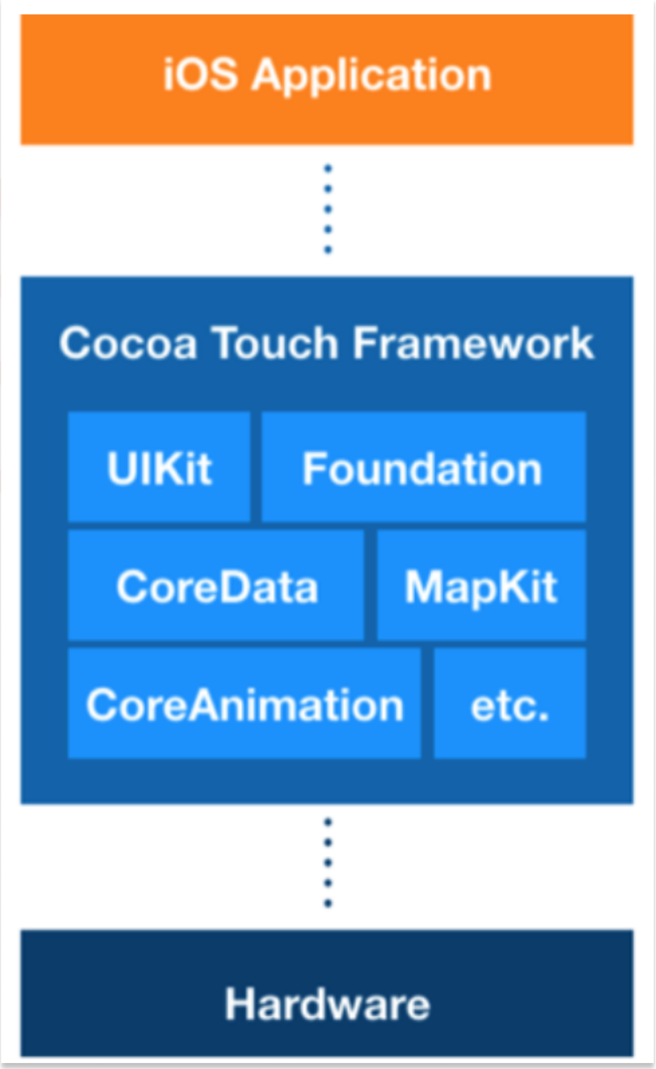
03 UI 구성하기

04 화면 전환

05 데이터 전달 기초

01

iOS 기초

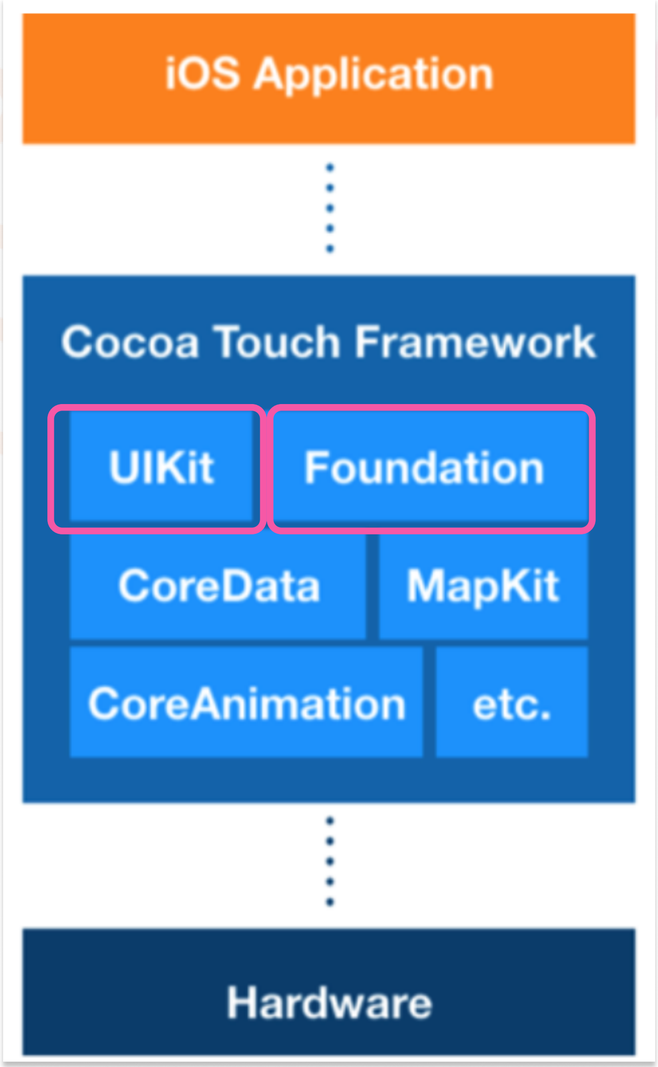


📌 *Cocoa Touch Framework 란?

- iOS 어플리케이션 개발 환경
- 어플리케이션의 다양한 기능 구현에 필요한
  여러 *프레임워크를 포함하는 최상위 레벨의 프레임 워크
- 핵심 프레임워크인 UIKit과 Foundation을 포함

*Cocoa 프레임워크는 macOS 애플리케이션 제작에 사용하는 프레임워크
*프레임워크는 소프트웨어의 구체적인 부분에 해당하는 설계와 구현을 재사용이 가능하게 끔 일련의 협업화된 형태로 클래스들을 제공하는 것

iOS 기초



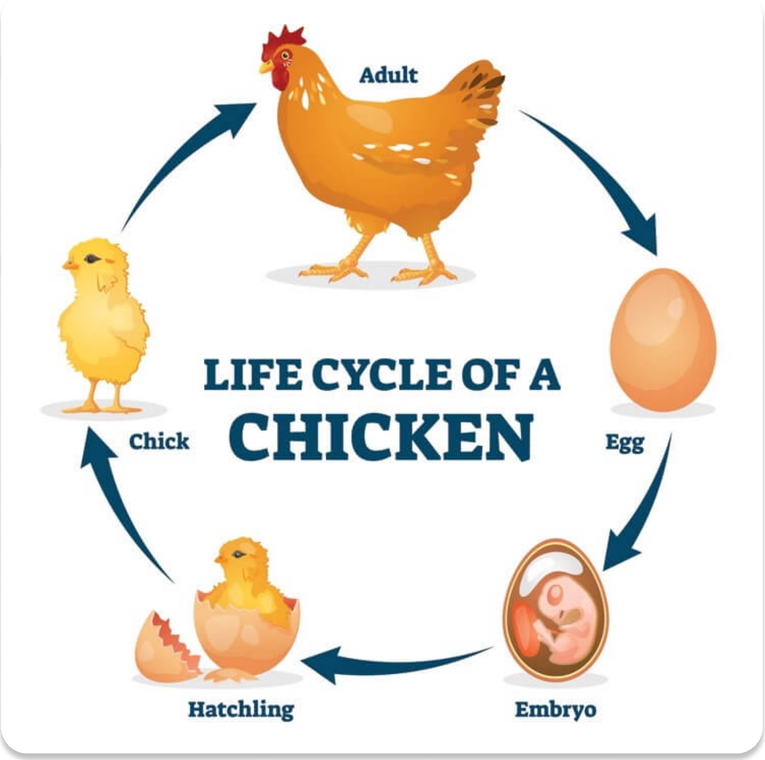
📌 UIKit

- 사용자의 인터페이스를 관리하고 이벤트를 처리하는게 주 목적인 프레임 워크
- 제스처 처리, 애니메이션, 그림 그리기, 이미지 처리, 텍스트 처리 등의 이벤트 처리
- 테이블뷰, 슬라이더, 버튼, 텍스트 필드 등 어플리케이션의 화면을 구성하는 요소가 포함

📌 Foundation

- iOS 어플리케이션의 운영체제 서비스와 기본 기능을 포함하는 프레임 워크
- 원시 데이터 타입(String, Int, Double), 컬렉션 타입(Array, Dictionary, Set) 및 운영체제 서비스를 사용해 어플리케이션의 기본적인 기능을 관리하는 프레임워크

Life Cycle(생명주기)



태어나고 죽기까지 하나의 연결된 과정

Life Cycle(생명주기)

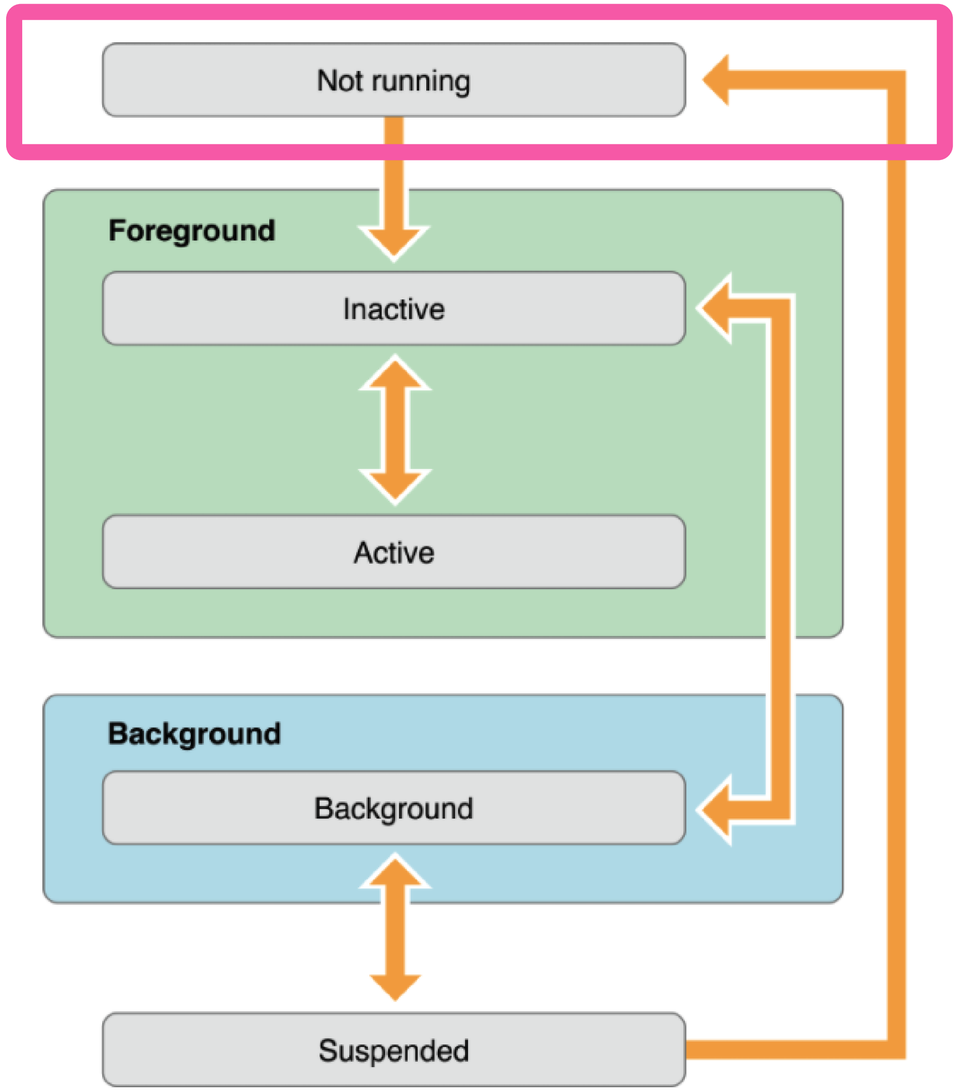
우리가 이제 개발할 iOS APP 또한 생명주기가 있습니다.

사용자가 볼 때는 단순히 화면이 바뀌고,
앱이 켜졌다가 꺼지는 것처럼 보이지만

그 뒤에는 유기적으로 연결된 수 많은 과정들이 반복되고 있습니다.

계란에서 갑자기 닭이 될 수는 없겠죠?
처음이라 낯설겠지만, 전체적인 생명주기에 대해 먼저 보겠습니다!

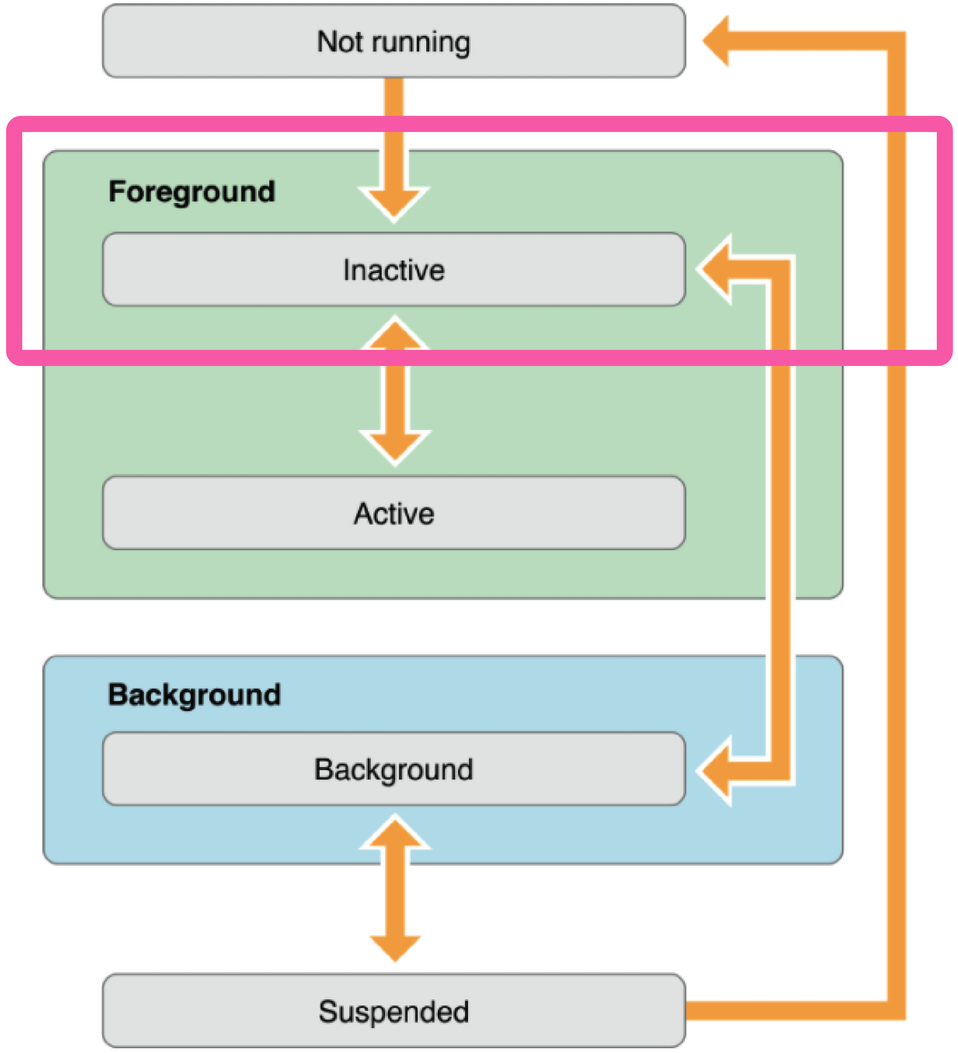
앱의 상태(App State) 변화



📍 Not running

- 앱이 실행되지 않거나 종료된 상태

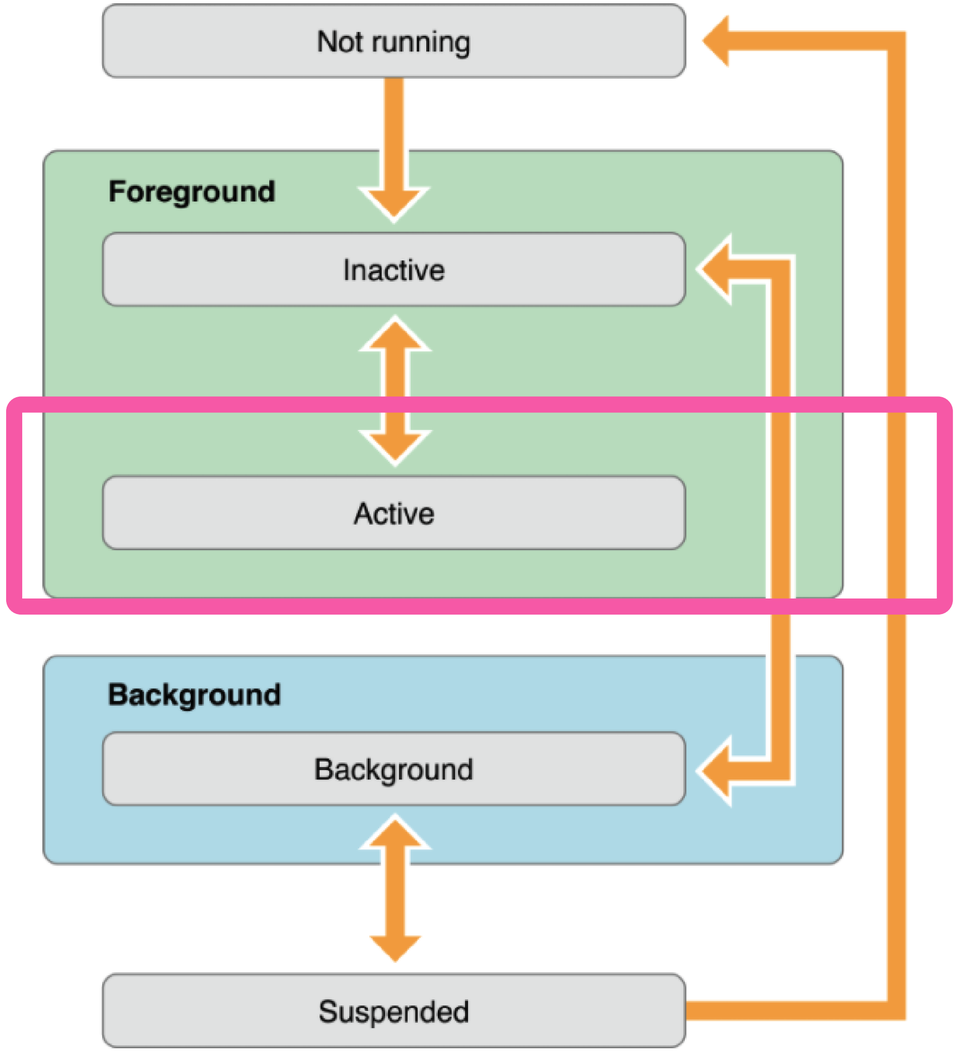
앱의 상태(App State) 변화



📍 Inactive

- 짧은 상태로 앱이 전면에서 실행 중이지만,
  다른 기능을 수행하고 있으며 사용자 입력이나
  이벤트를 받아들일 준비가 되어 있지 않은 상태
- 전화를 걸었을 때, SMS 메세지를 받았을 때의 짧은 순간

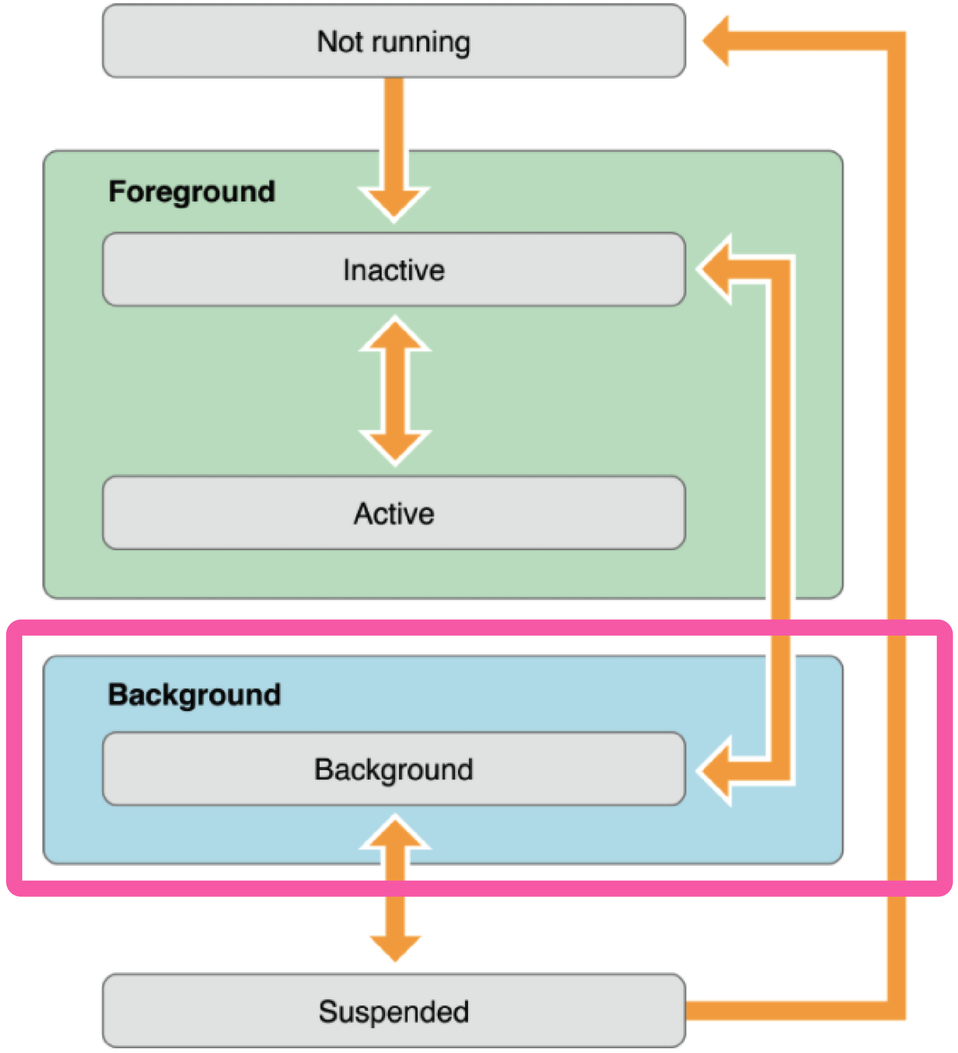
앱의 상태(App State) 변화



📍 Active

- 앱이 전면에서 실행중이며 이벤트를 받을 수 있는 상태

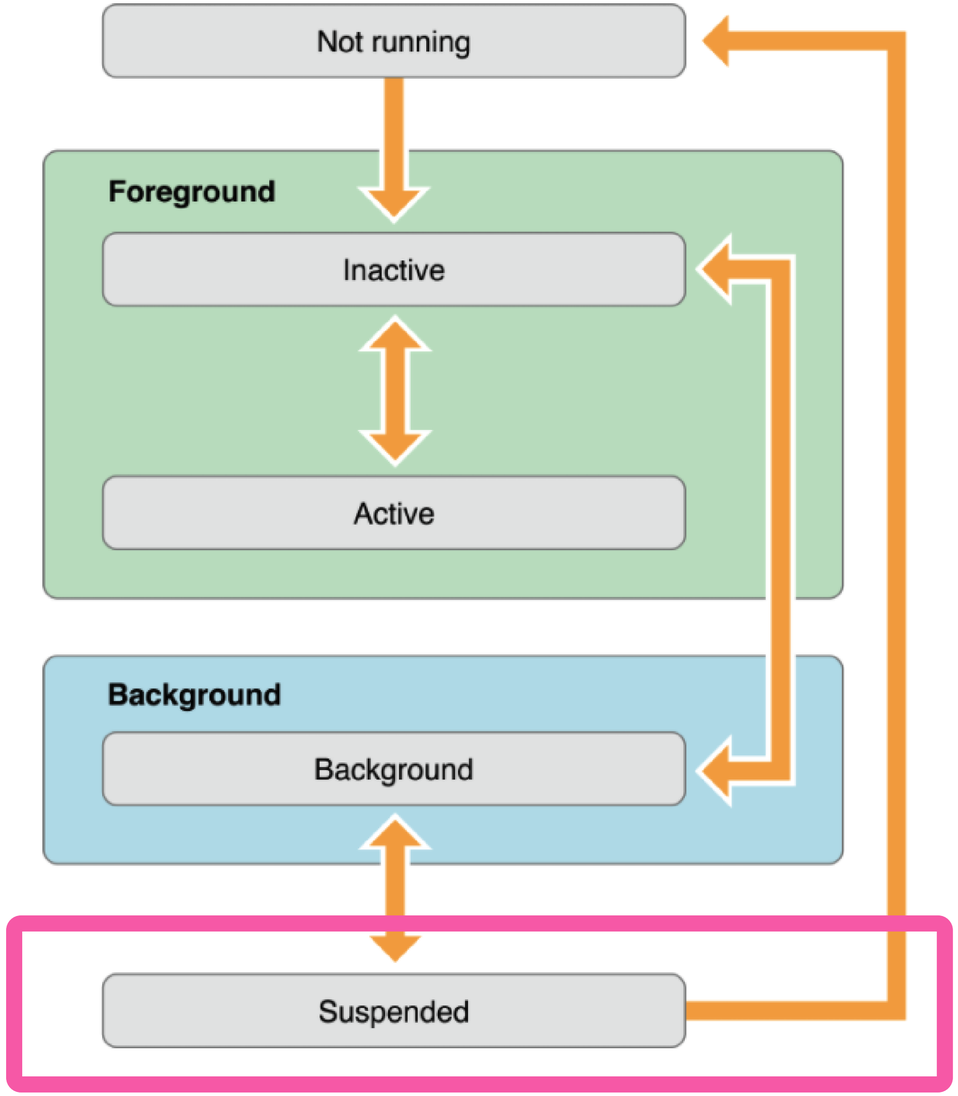
앱의 상태(App State) 변화



📍 Background

- 앱이 백그라운드에 있지만 여전히 코드가 실행되고 있는 상태
- 파일 다운로드나, 업로드, 연산처리 등 여분의 실행 시간이 필요한 앱일 경우 특정 시간 동안 이 상태로 남아있게 되는 경우가 있음

앱의 상태(App State) 변화

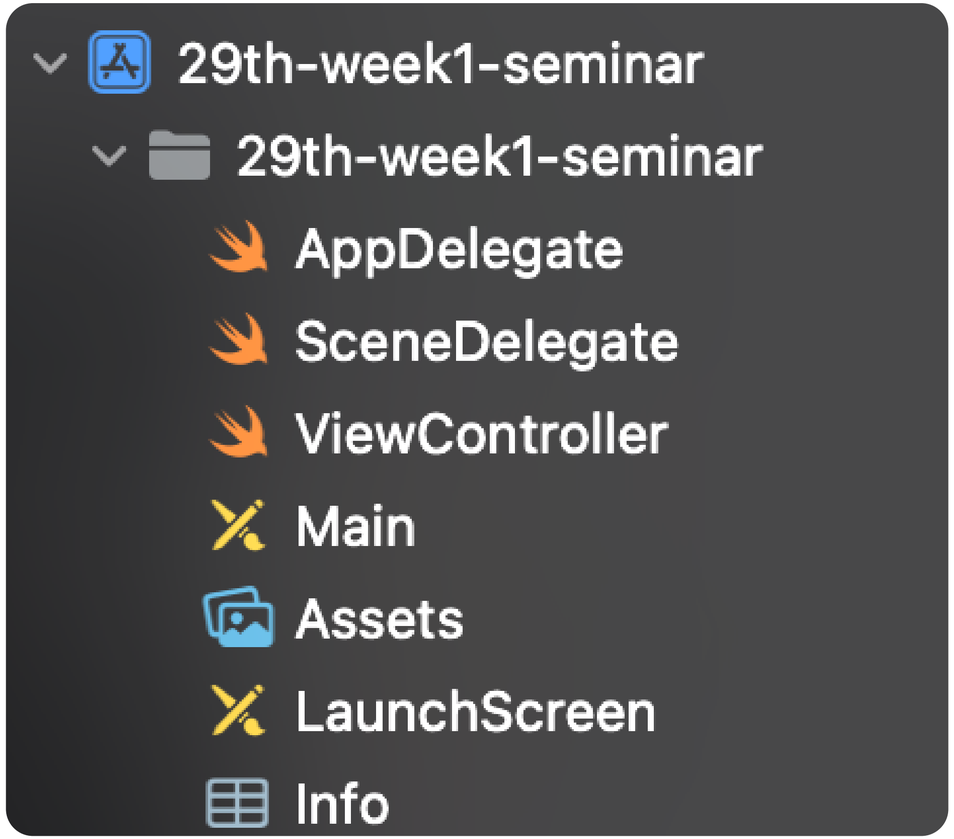


*suspend
- (공식적으로)유예하다, 중단하다
- (공식적으로)연기하다, 유보하다

📍 Suspended

- 백그라운드 상태에서 활동을 멈춘 상태
- 빠른 재실행을 위하여 메모리에 적재된 상태지만 실질적으로 동작하고 있지는 않음.
- 메모리가 부족할 때 비로소 시스템이 강제종료함.

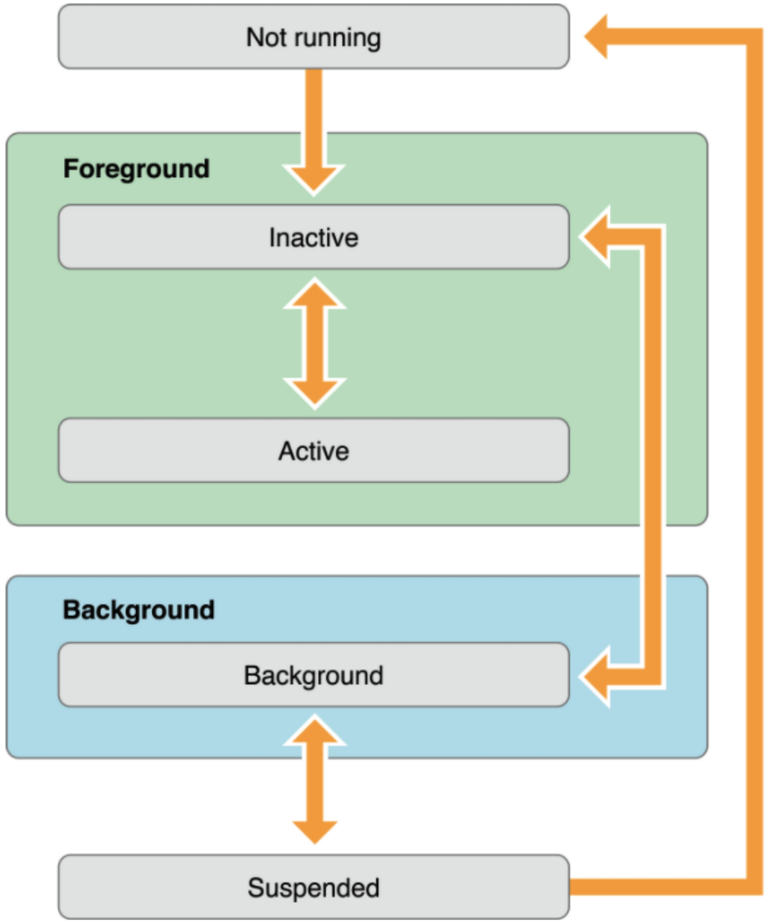
App Life-Cycle



iOS App Project를 생성하면
AppDelegate와 SceneDelegate가
기본으로 있습니다

지금 당장 둘을 완전히 이해하겠다라는 느낌보다는
이 class들이 어떤 역할을 하는구나~
정도로 들어주시면 좋을 것 같아요!

AppDelegate.swift

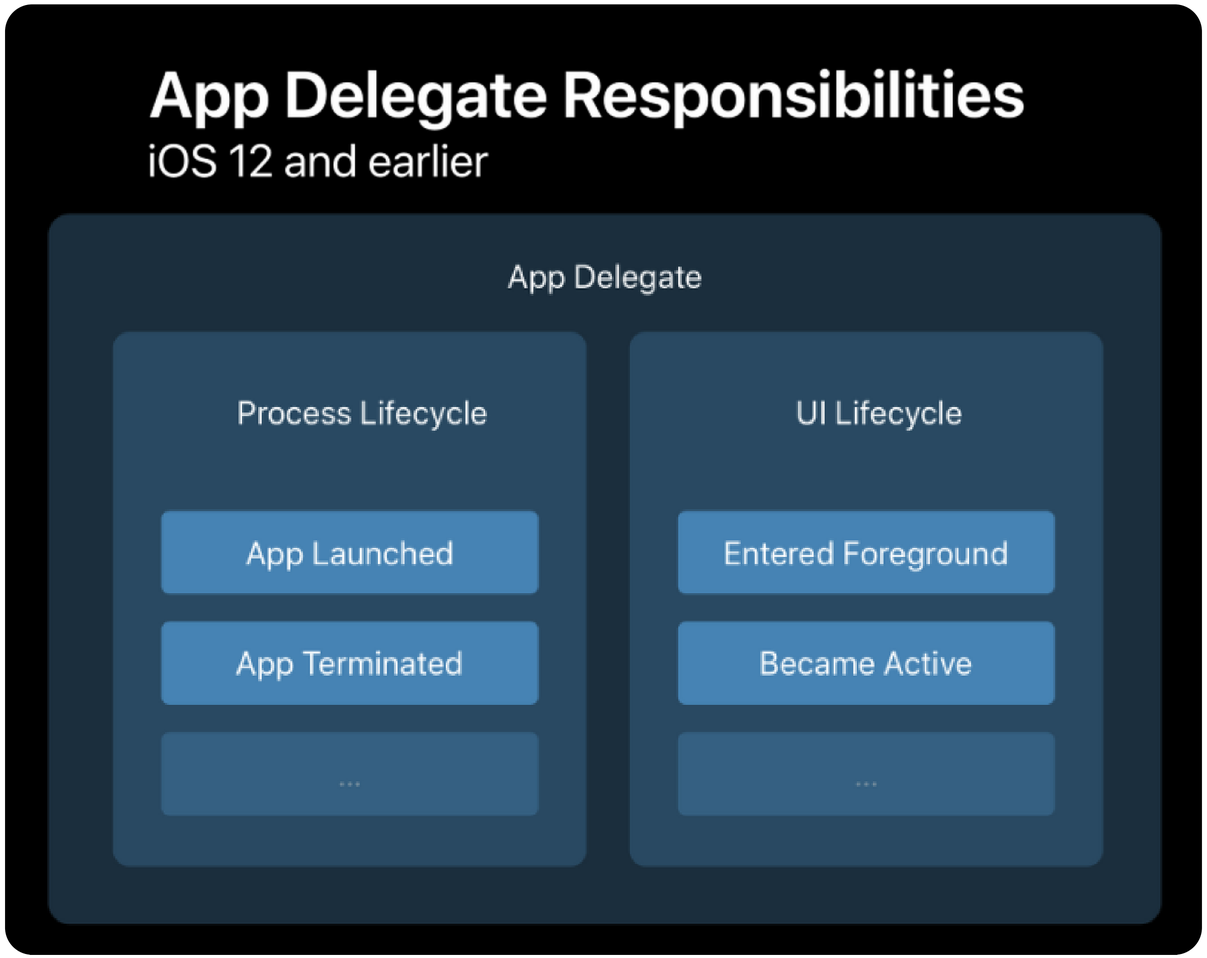


1. **func application(_:didFinishLaunchingWithOptions:)**
   앱이 처음 시작될때, 앱이 사용자에게 화면으로 표시되기 직전에 호출됨
2. **func applicationDidBecomeActive(UIApplication)**
   앱이 active될 때 실행
3. **func applicationWillResignActive(UIApplication)**
   앱이 active에서 inactive로 이동 될 때 실행
4. **func applicationDidEnterBackground(UIApplication)**
   앱이 background 상태일 때 실행 (언제든 suspend 상태가 될 수 있음)
5. **func applicationWillEnterForeground(UIApplication)**
   앱이 background 에서 foreground로 이동 될 때 실행
6. **func applicationWillTerminate(UIApplication)**
   앱이 종료될 때 실행 (suspend 상태에서 종료될 때에는 호출되지 않음)

AppDelegate.swift

Process Lifecycle

응용 프로그램에
프로세스 수준
이벤트를 알리는 것

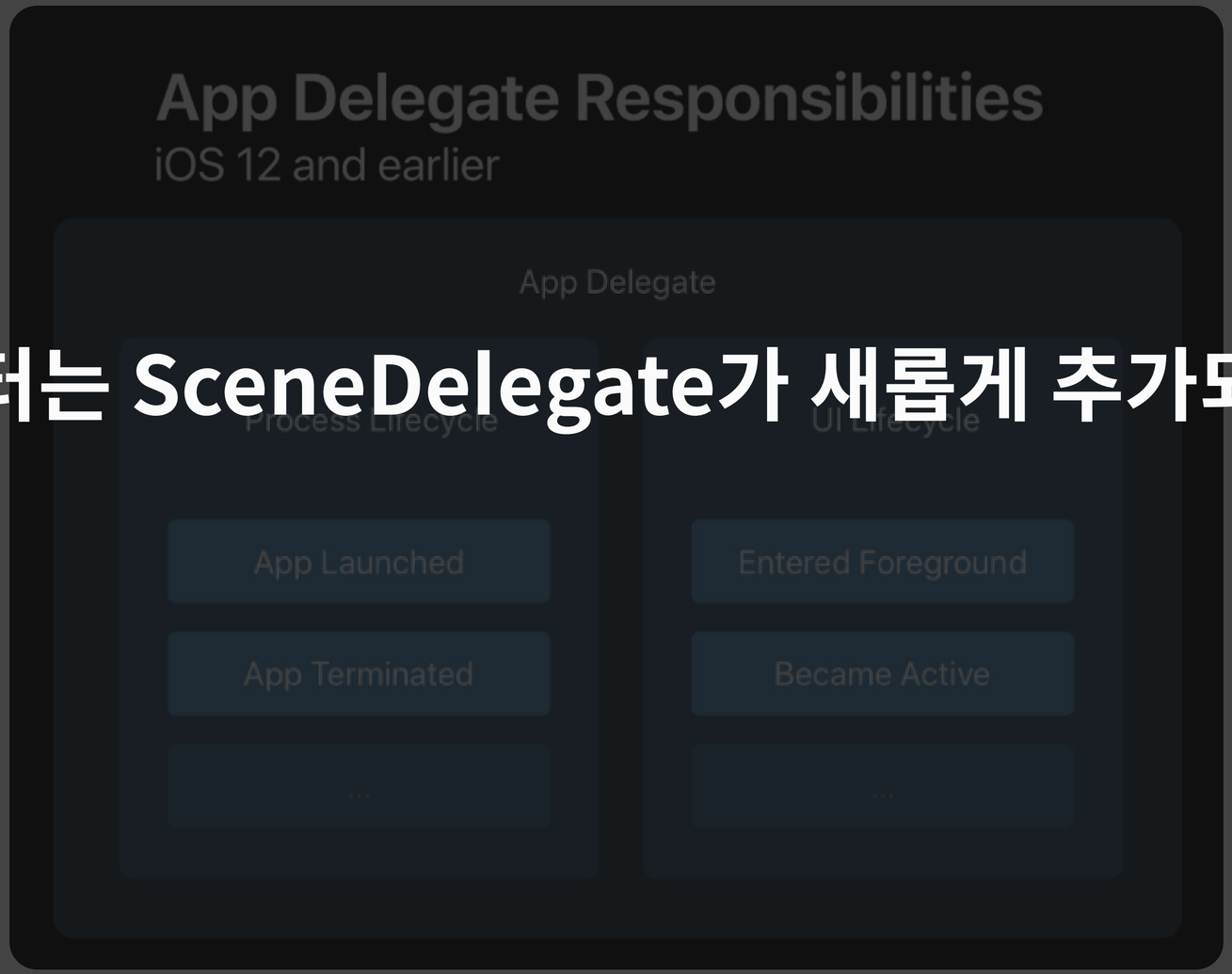


UI Lifecycle

응용 프로그램에
UI의 상태를 알리는 것

앱은 오직 "하나의 프로세스"와 그에 맞는 "하나의 UI"만 가지기 때문

AppDelegate.swift



Process Lifecycle

응용 프로그램에
프로세스 수준
이벤트를 알리는 것

UI Lifecycle

응용 프로그램에
UI의 상태를 알리는 것

iOS 13부터는 SceneDelegate가 새롭게 추가되었습니다

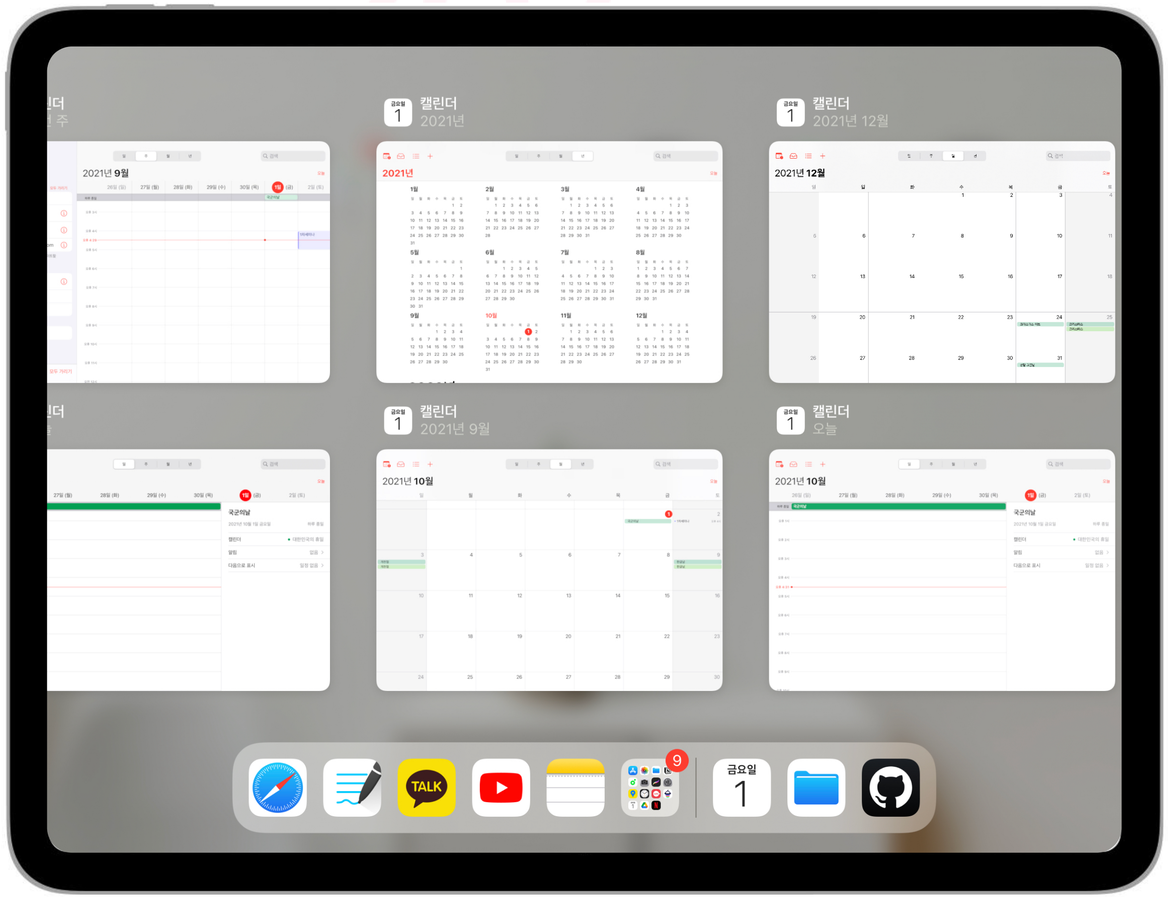
iOS 기초

✅ iOS13 이상과 iPadOS에서 멀티윈도우 지원

✅ 하나의 앱에 하나의 화면(window) 였지만

iOS 13부터는 window의 개념이 scene으로 대체

✅ 하나의 앱에서 여러 개의 scene을 가질 수 있게 됨

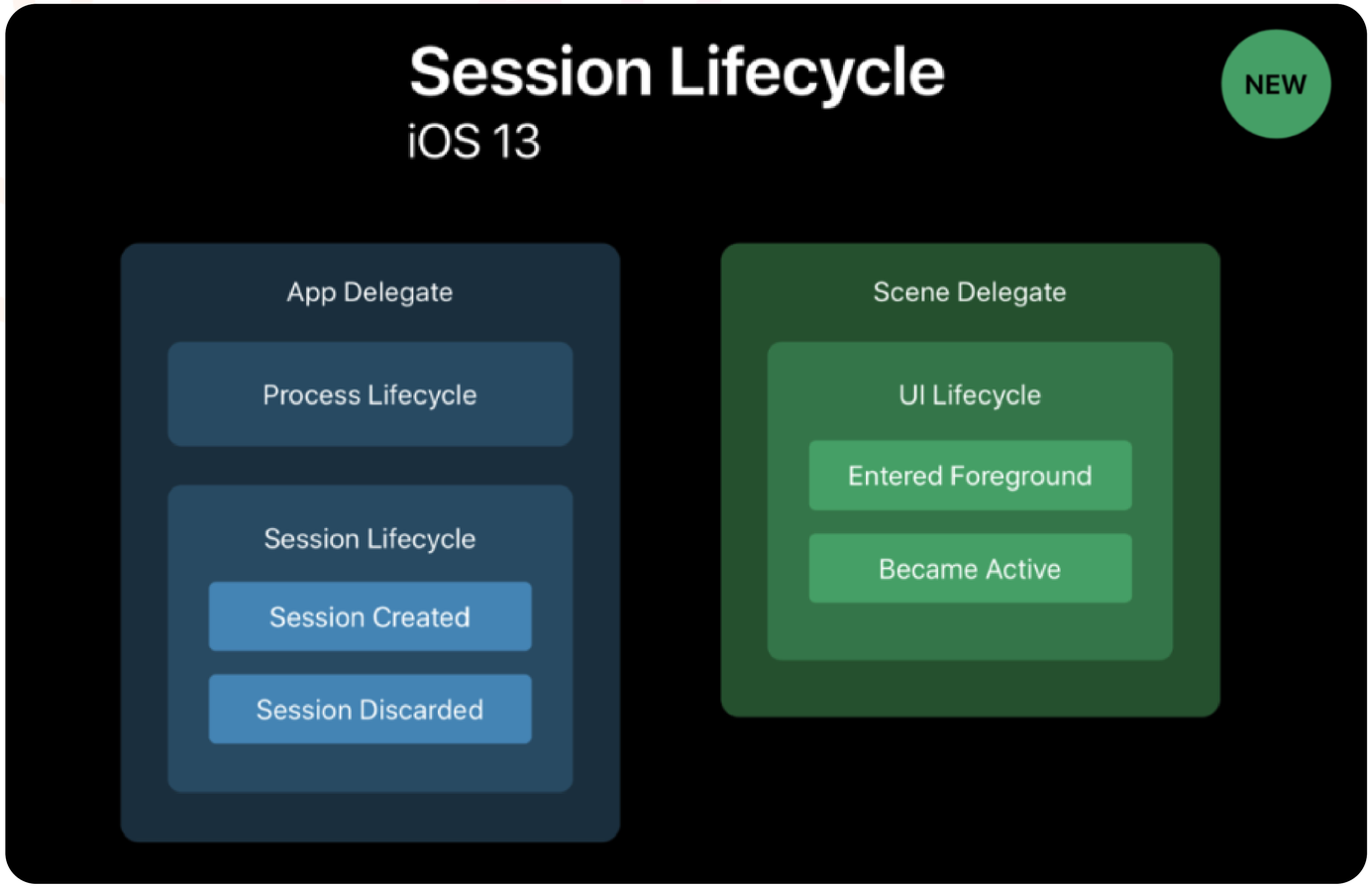


하나의 캘린더 앱이지만

여러 개의 화면 상태로 들어갈 수 있고

어떤 날짜에 이벤트를 추가하면

모든 화면에도 적용되어야겠죠?



📌 **SceneDelegate**

UI Lifecycle을 관리

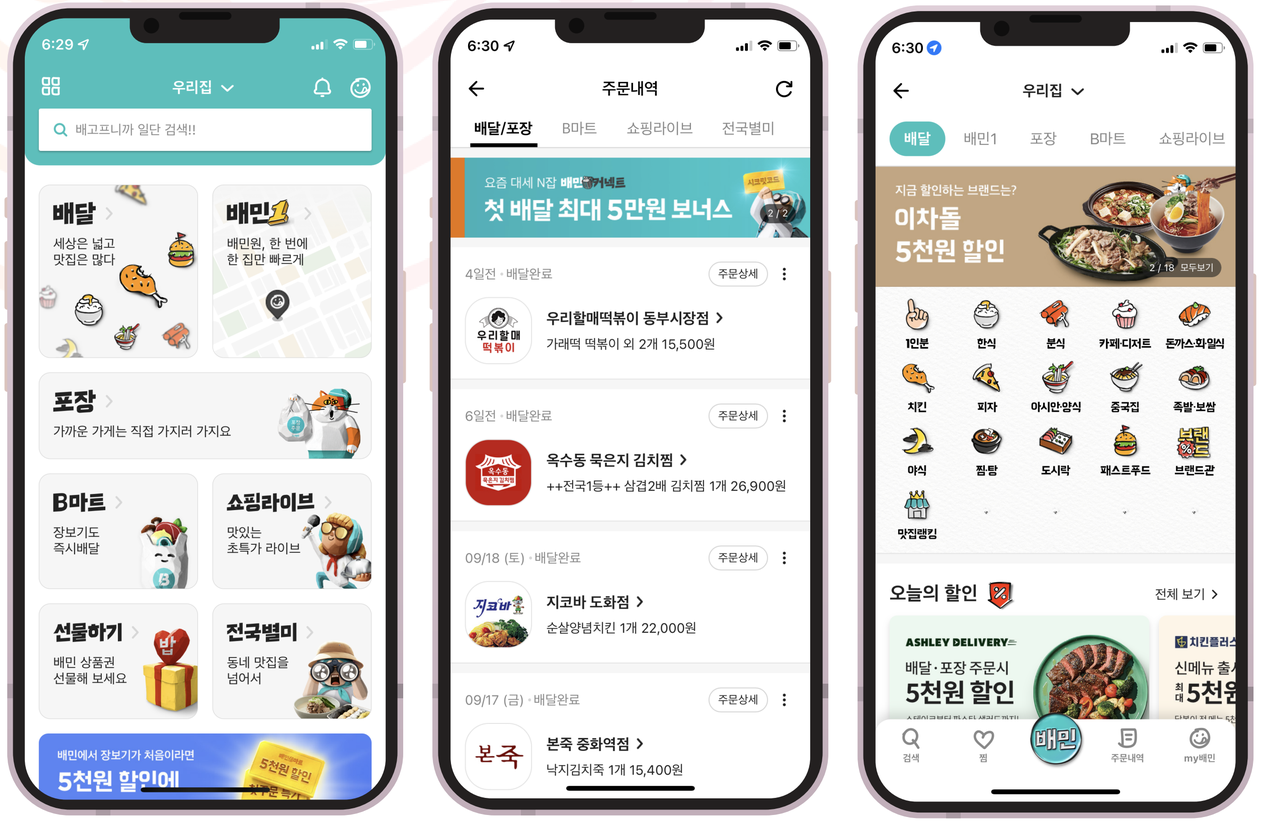
📌 **AppDelegate**

새롭게 생겨난 Session
관리하는 역할이 추가

UIApplicationDelegate	UISceneDelegate
application:willEnterForeground	scene:willEnterForeground
application:didEnterBackground	scene:didEnterBackground
application:willResignActive	scene:willResignActive
application:didBecomeActive	scene:didBecomeActive

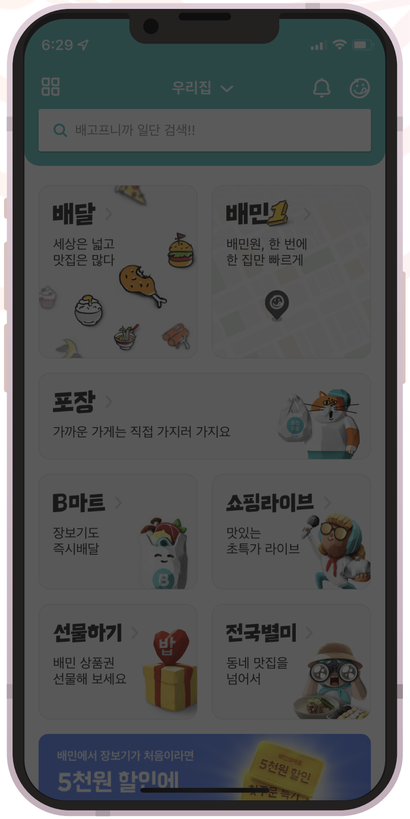
✅ **AppDelegate에서 하던 역할을 SceneDelegate에서 수행**

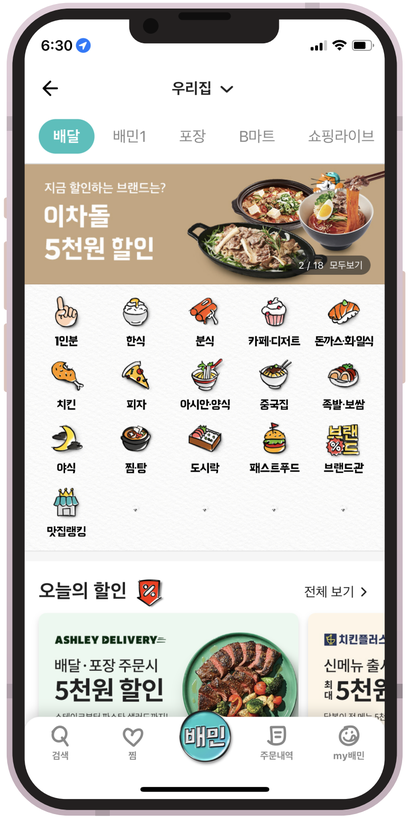
iOS 기초



어플리케이션들은 보통 하나 이상의 ‘화면’으로 구성.

📍이 화면의 단위를 보통 View라고 이야기 함

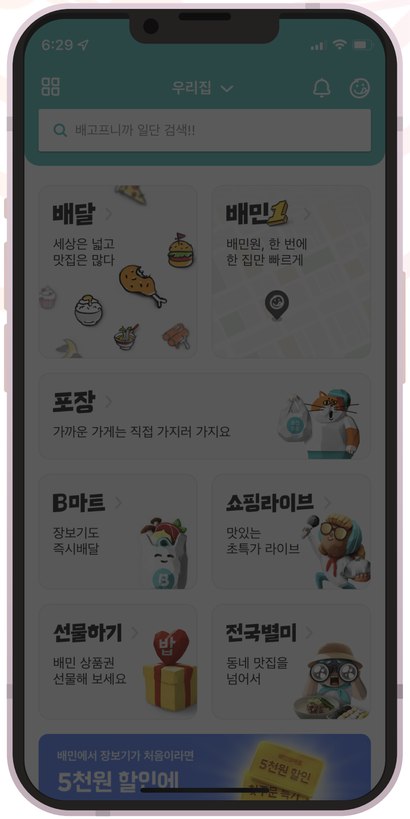


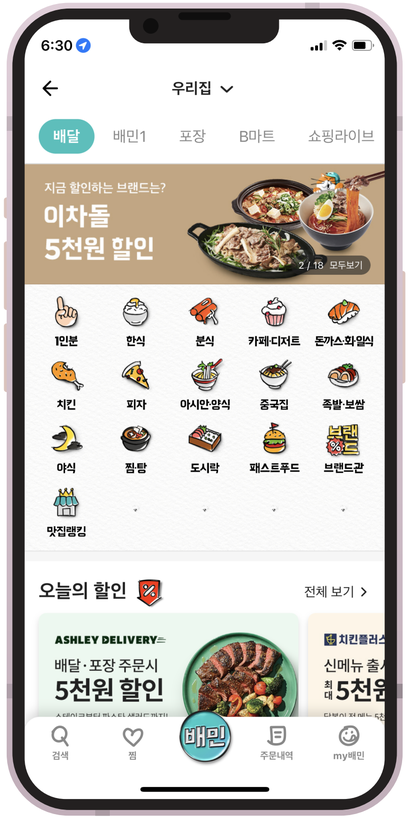


한 화면에서 다른 화면으로 전환될 때

각 화면(View)마다

나타났다 사라지는 생명주기가 있습니다





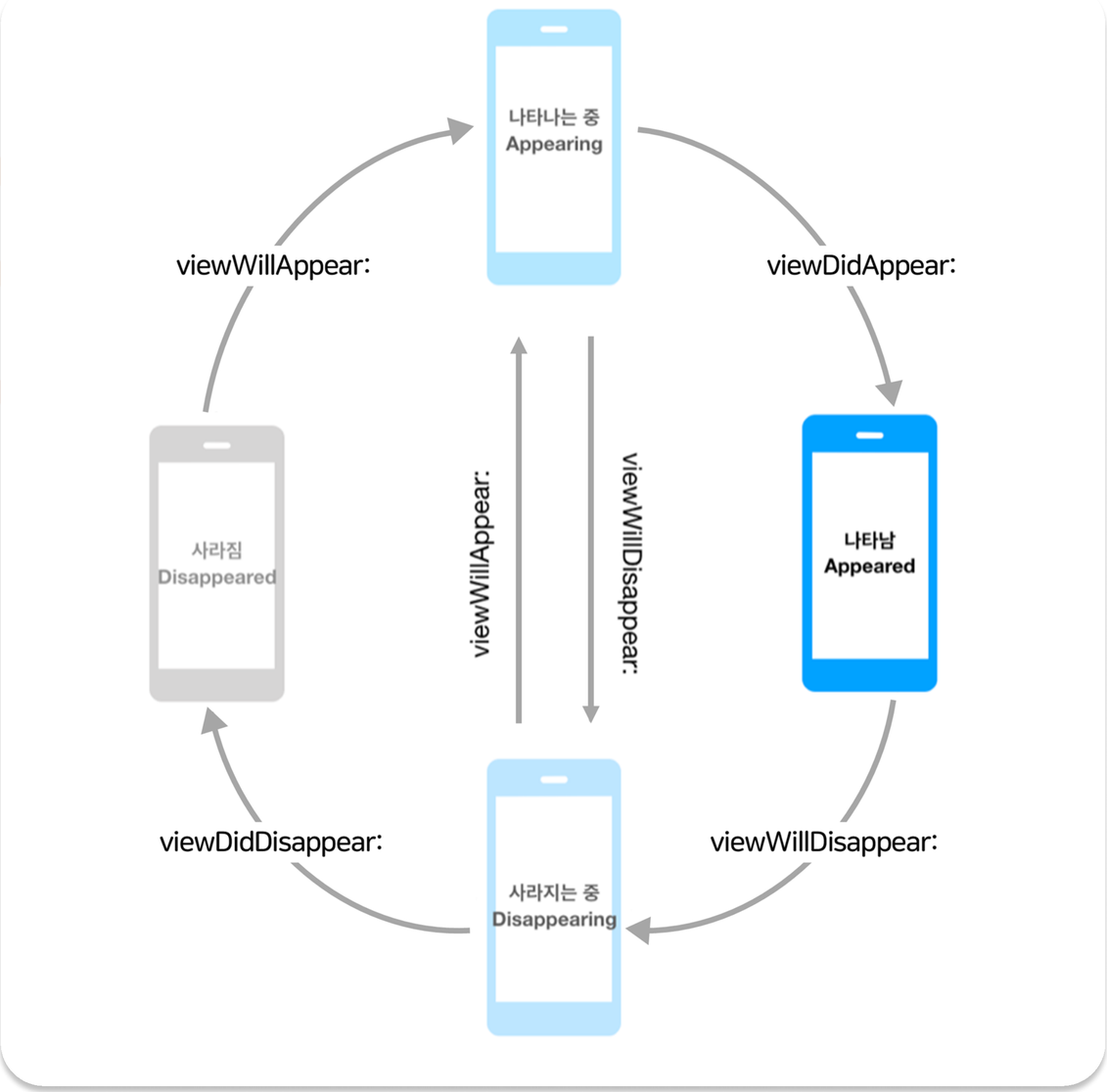
한 화면에서 다른 화면으로 전환될 때

각 화면(View)마다

나타났다 사라지는 생명주기가 있습니다

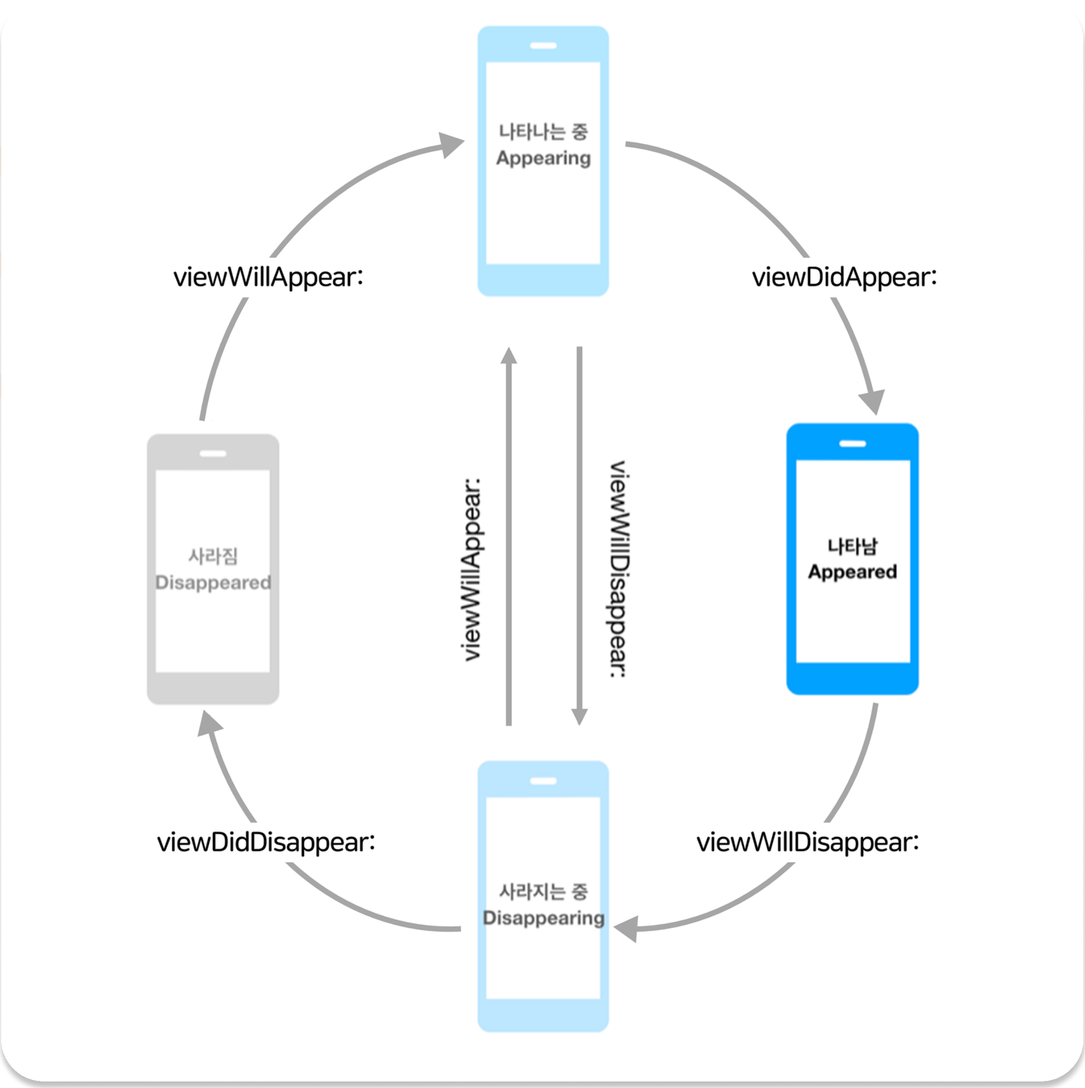
이러한 View들을 관리하는 것이

ViewController



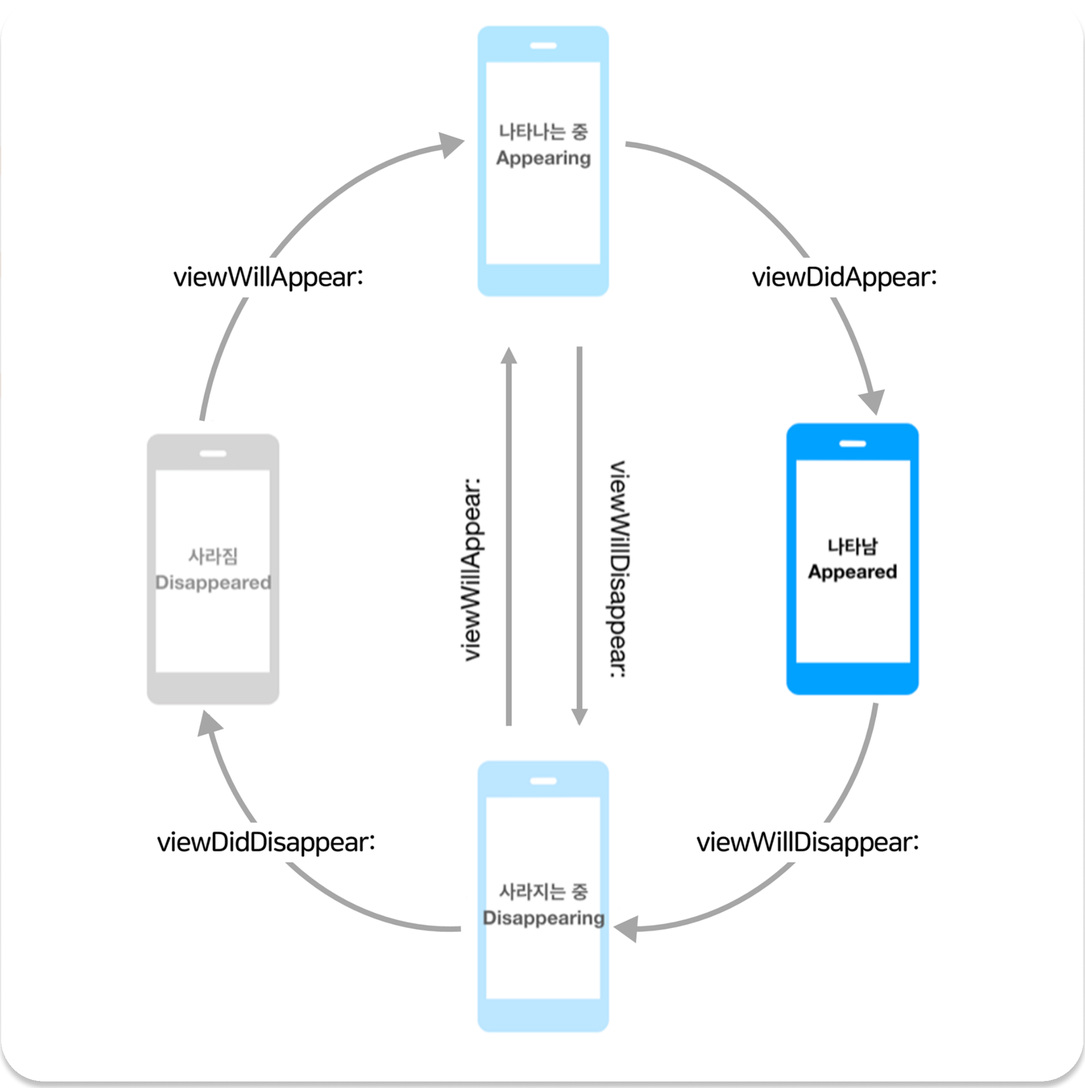
📌 ViewController

- 컨텐츠를 화면에 표시하는 뷰를 관리하고, 해당 뷰에 대한 참조를 유지
- 이 외에도 화면 내에 다양한 기능들을 담당
- View의 Lifecycle에 대한 상태 변화 감지



📌 뷰의 상태 변화 감지 메서드

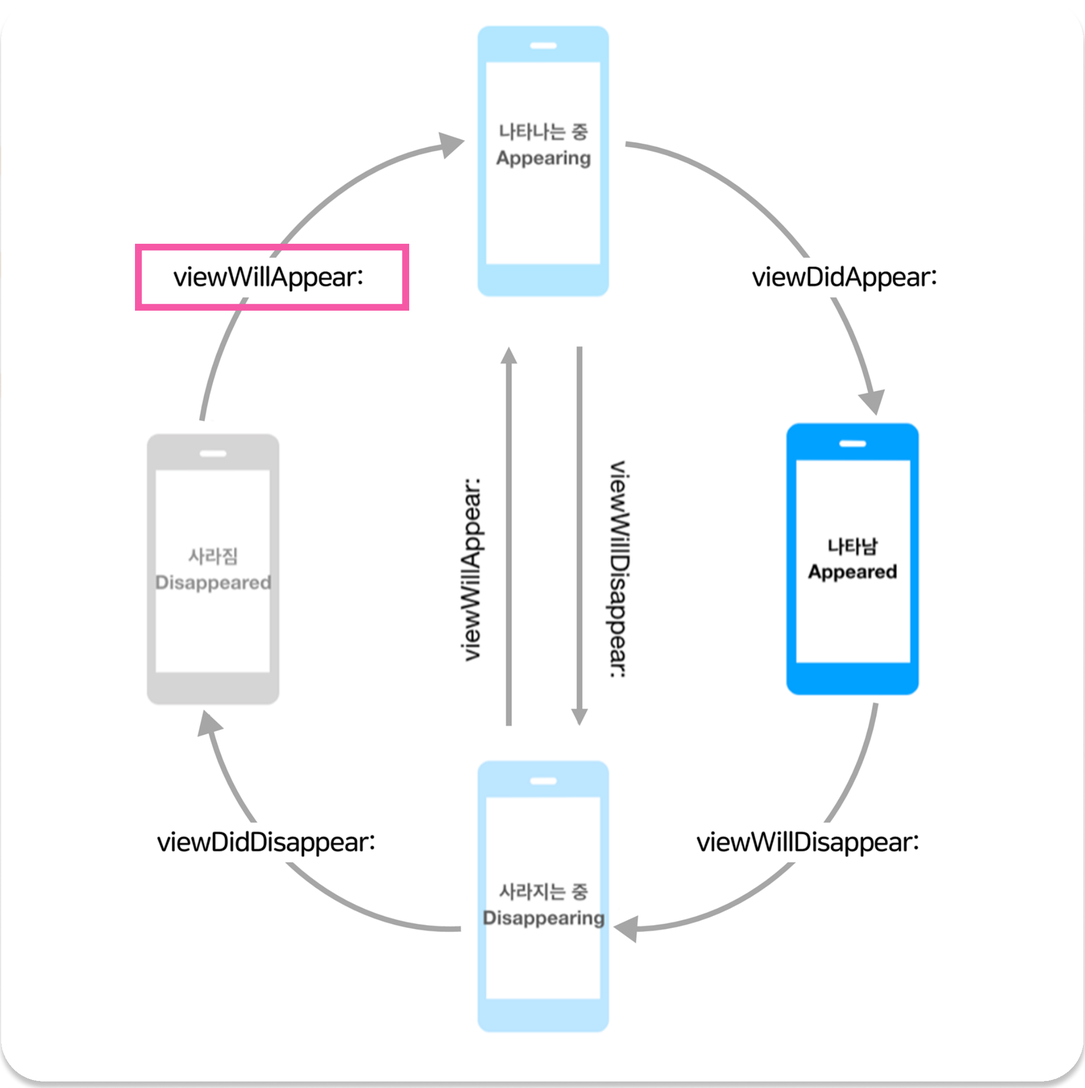
```
func viewDidLoad()
func viewWillAppear(_ animated: Bool)
func viewDidAppear(_ animated: Bool)
func viewWillDisappear(_ animated: Bool)
func viewDidDisappear(_ animated: Bool)
```



📌 뷰의 상태 변화 감지 메서드

func viewDidLoad()

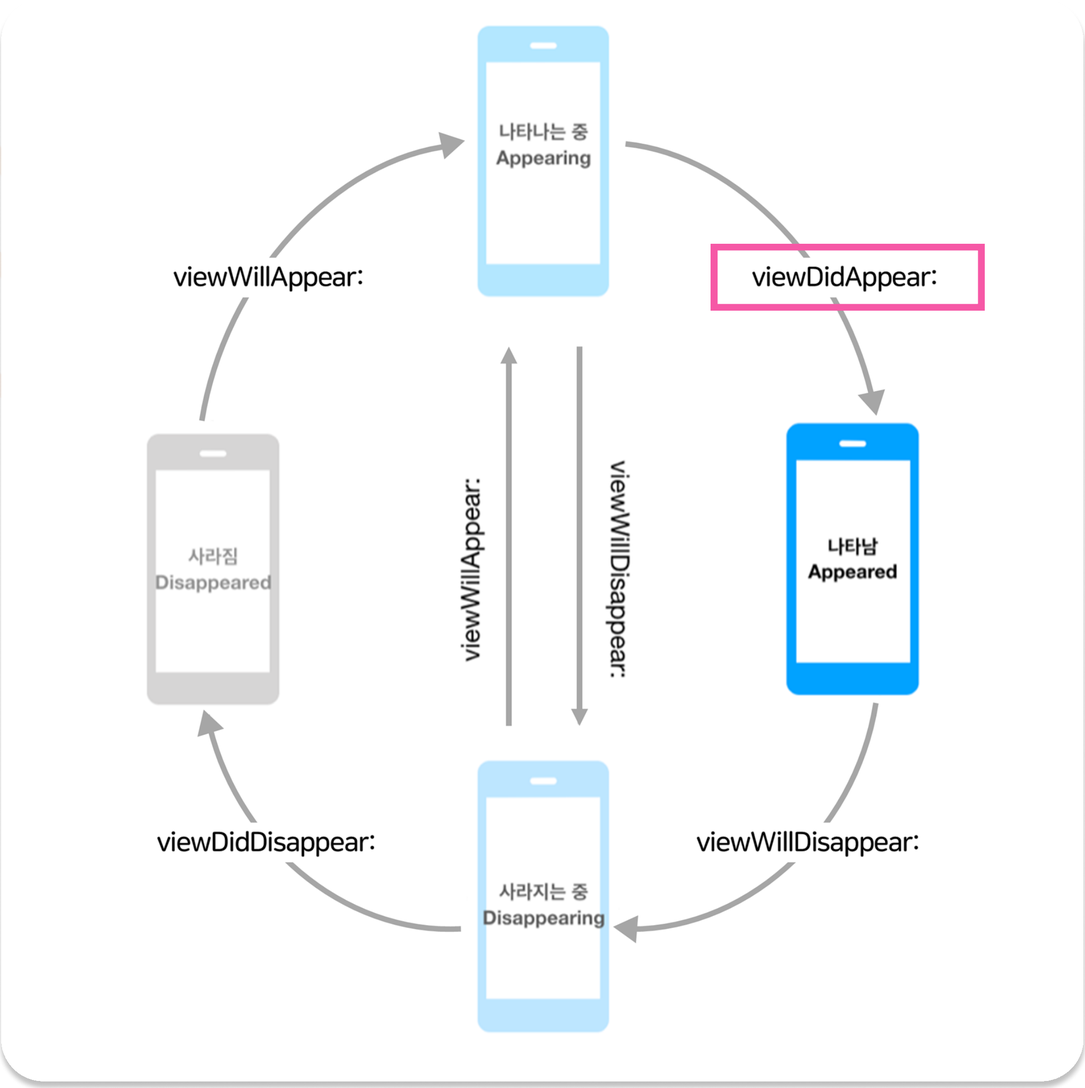
- 뷰 계층이 메모리에 로드된 직후 호출되는 메서드
- 뷰의 추가적인 초기화 작업을 하기 좋은 시점
- 메모리에 처음 로딩 될때 1회 호출되는 메서드로,
- 메모리 경고로 뷰가 사라지지 않는 이상 다시 호출되지 않음



📌 뷰의 상태 변화 감지 메서드

func viewWillAppear(_ animated: Bool)

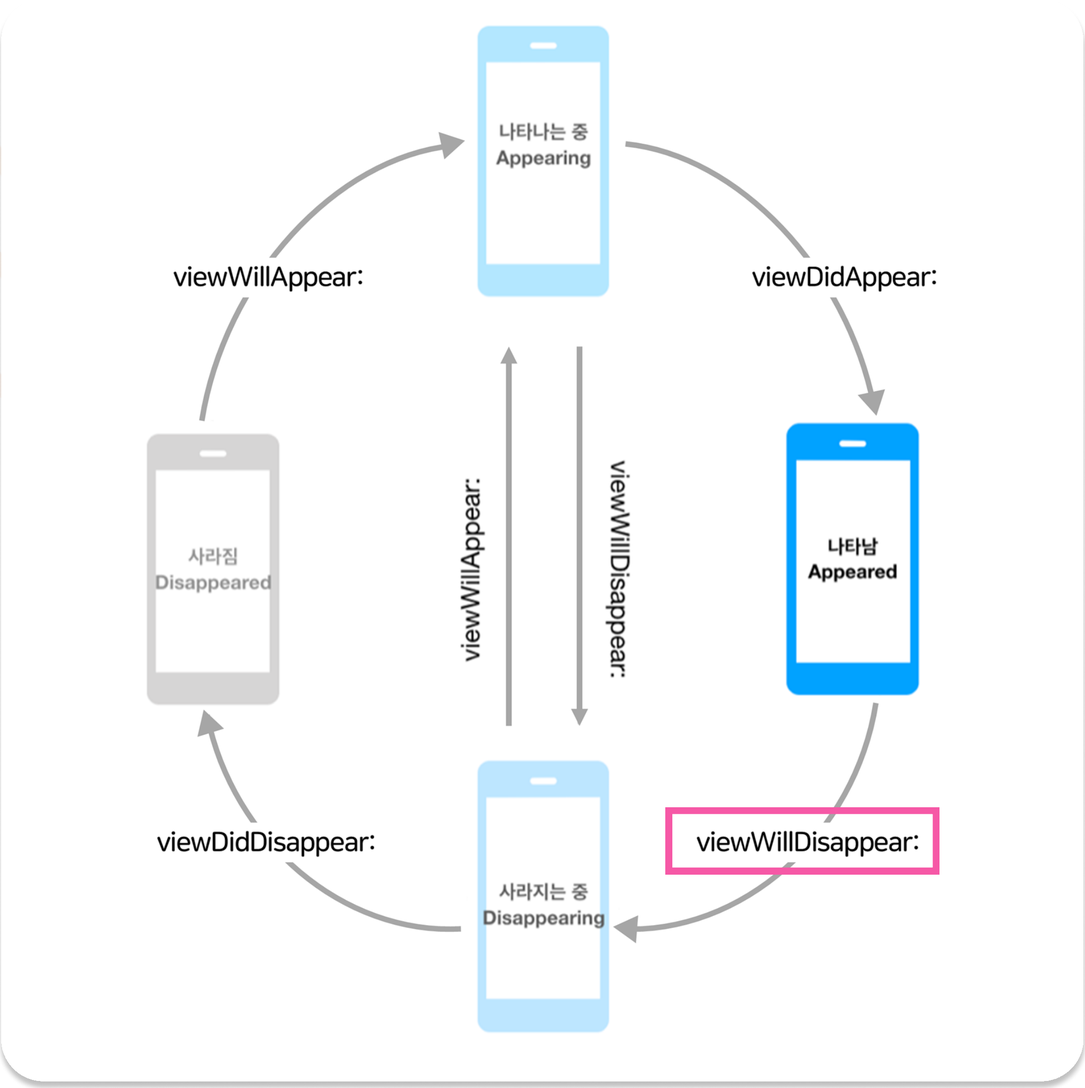
- 뷰가 뷰 계층에 추가되고 화면이 표시되기 직전에 호출되는 메서드
- 뷰의 추가적인 초기화 작업을 하기 좋은 시점
- 다른 뷰로 이동했다가 되돌아오면 재호출되는 메서드로,
  화면이 나타날때마다 수행해야하는 작업을 하기 좋은 시점



📌 뷰의 상태 변화 감지 메서드

func viewDidAppear(_ animated: Bool)

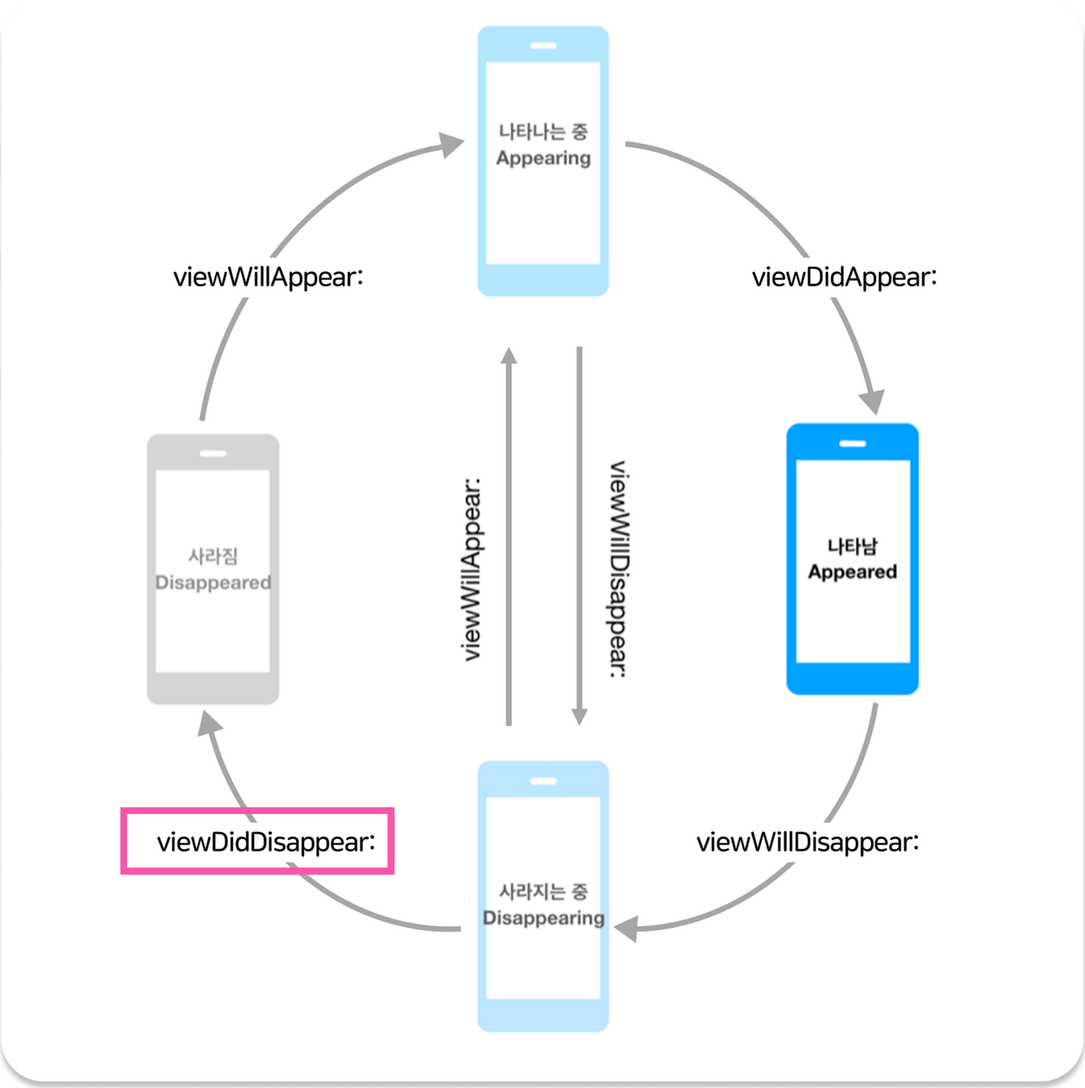
- 뷰가 뷰 계층에 추가되어 화면이 표시되면 호출되는 메서드
- 뷰를 나타내는 것과 관련된 추가적인 작업을 하기 좋은 시점



📌 뷰의 상태 변화 감지 메서드

func viewWillDisappear(_ animated: Bool)

- 뷰가 뷰 계층에서 사라지기 직전에 호출되는 메서드
- 뷰가 생성된 뒤 발생한 변화를 이전상태로 되돌리기 좋은 시점

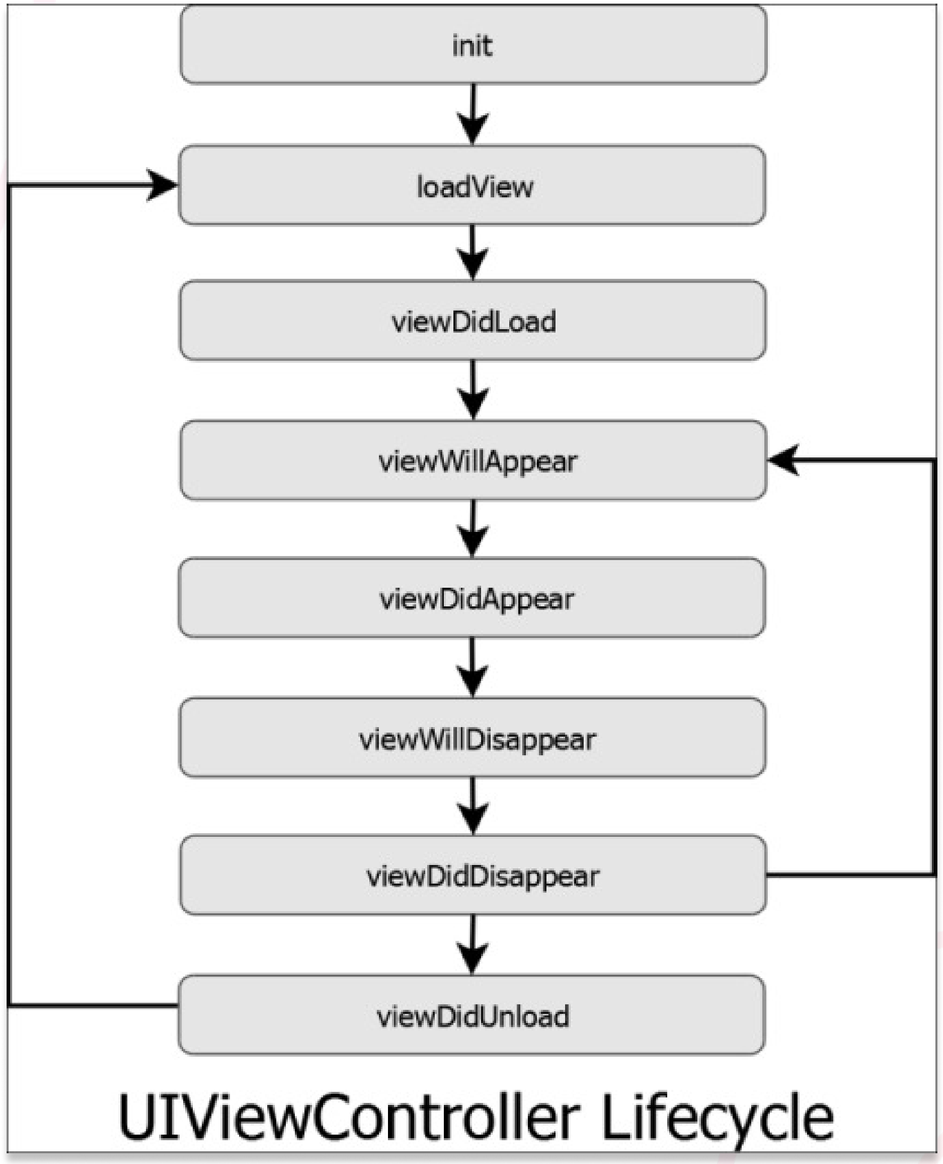


📌 뷰의 상태 변화 감지 메서드

func viewDidDisappear(_ animated: Bool)

- 뷰가 뷰 계층에서 사라진 후 호출되는 메서드
- 뷰를 숨기는 것과 관련된 추가적인 작업을 하기 좋은 시점
- 시간이 오래 걸리는 작업은 하지 않는 것이 좋음

✅ 메모리에 뷰가 없을 경우



✅ 메모리에 뷰가 있을 경우

02

Xcode 사용하기

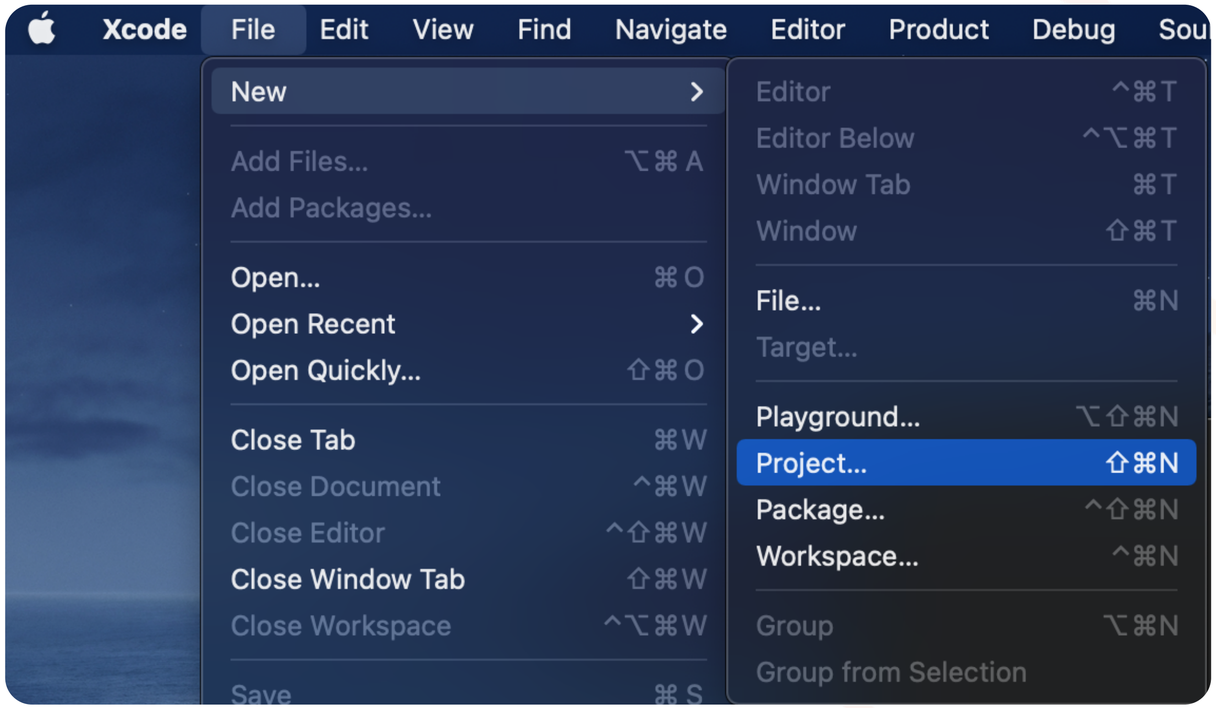
Xcode란?

- Mac X OS, iOS를 위한 IDE
(Integrated Development Environment)

- iOS 개발을 위한 필수 환경

Xcode 사용하기 | 프로젝트 생성

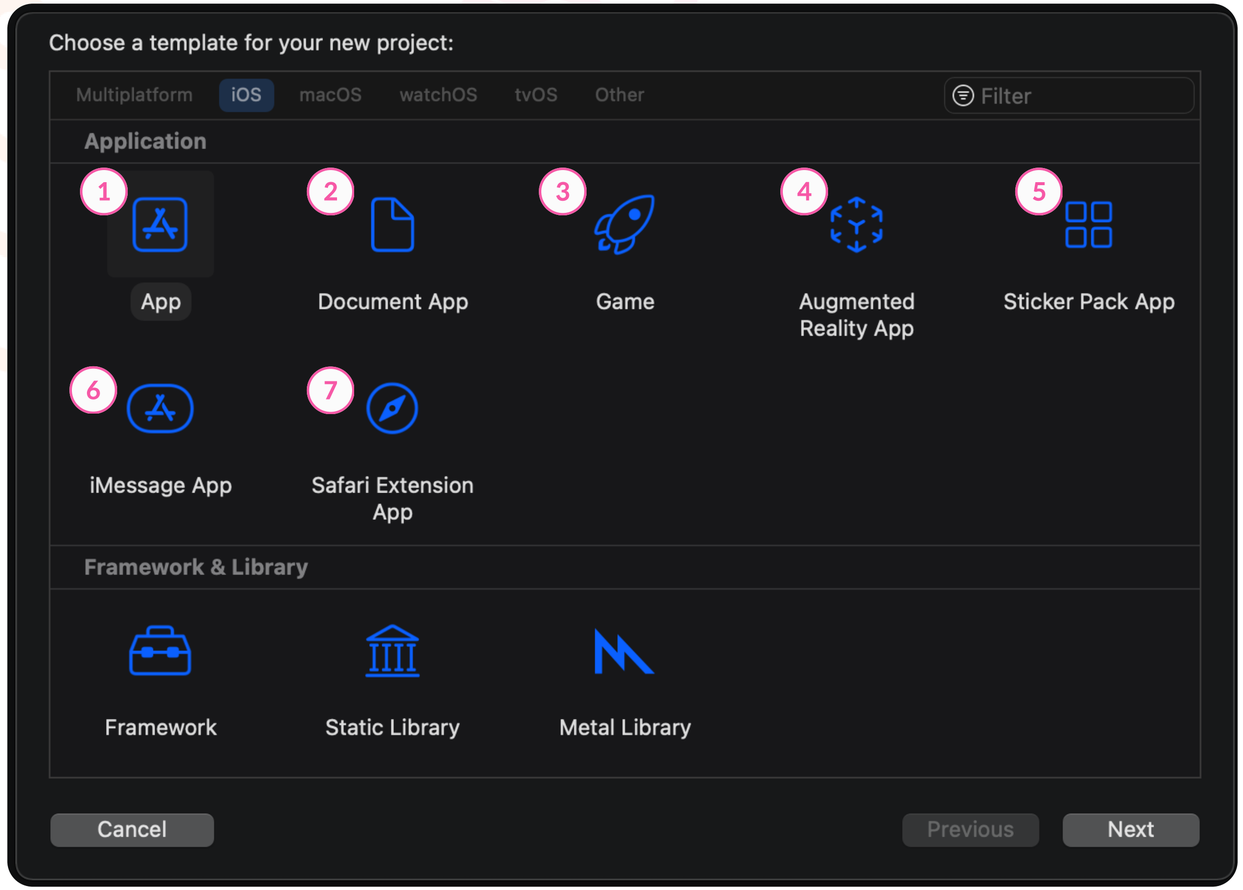
새로운 프로젝트를 생성하려면
Create a new Xcode Project를 눌러주세요



왼쪽 상단 File >> New >> Project 를 누르거나

shift (⇧) + command(⌘) + N 눌러도 생성이 가능합니다!

Xcode 사용하기 | 프로젝트 생성



1. App :
뷰를 사용하여 앱을 개발 할 때 쓰는 템플릿.
일반적으로 제일 많이 사용합니다.

2. Document App
데이터를 저장 할 수 있는 문서 기반의 앱을
개발할 때 사용하는 템플릿

3. Game
게임 앱을 개발 할 때 사용하는 템플릿,
그래픽 처리를 위한 OpenGL 게임 뷰를 생성합니다.

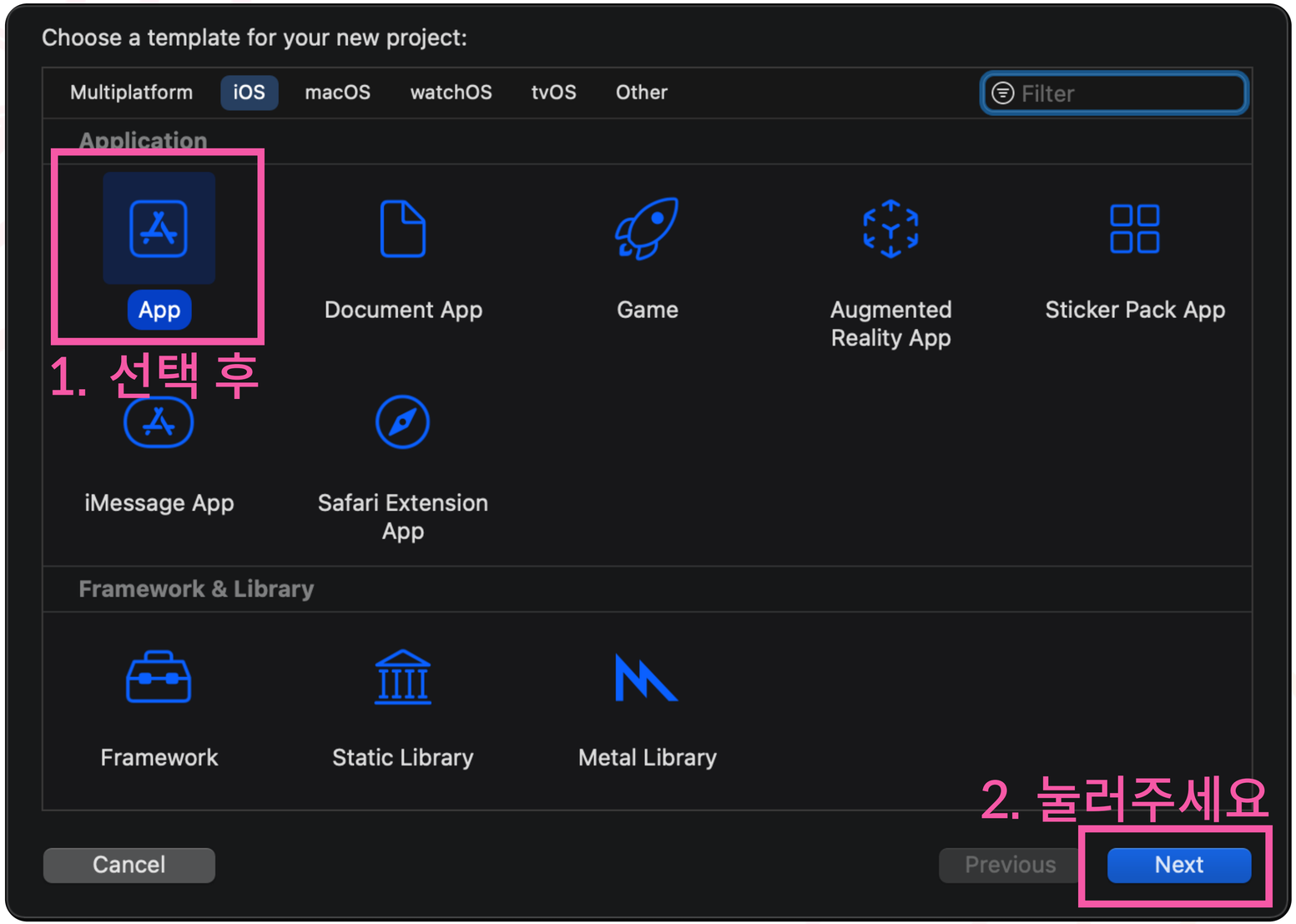
4. Argumented Reality App
증강현실(AR) 앱을 개발할 때 사용하는 템플릿

5. Sticker Pack App
스티커 팩 앱을 개발할 때 사용하는 템플릿

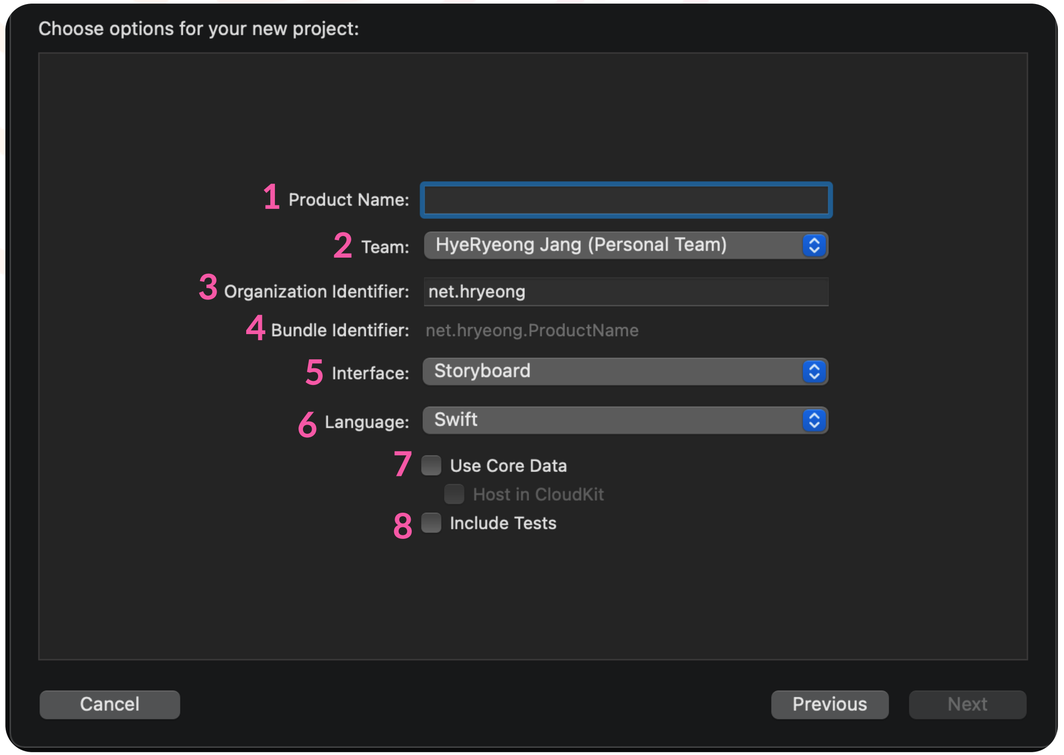
6. iMessageApp
iMessageApp을 개발할 때 사용하는 템플릿

7. Safari Extension App
사파리 확장 app을 개발할 때 사용하는 템플릿

Xcode 사용하기 | 프로젝트 생성



Xcode 사용하기 | 프로젝트 생성



1. Product Name

프로젝트 이름은 앱의 고유 번호인 'Bundle Identifier'를 만드는 요소로 사용 되므로,
실제 프로젝트에서는 적절한 이름을 넣는 것을 추천드립니다. :)

2. Team

애플 개발자 계정으로 생성된 인증서를 선택하는 부분입니다.
앱 스토어에 빌드 된 앱을 등록하거나 배포할 앱을 빌드하기 위해서는
그에 맞는 인증서가 필요합니다. 애플 개발자 사이트에서 발급받은 인증서를 선택하여
빌드하면 목적에 맞게 서명된 앱을 만들어 낼 수 있습니다.
계정이 없다면 우선 None이나 개인계정을 넣어주세요

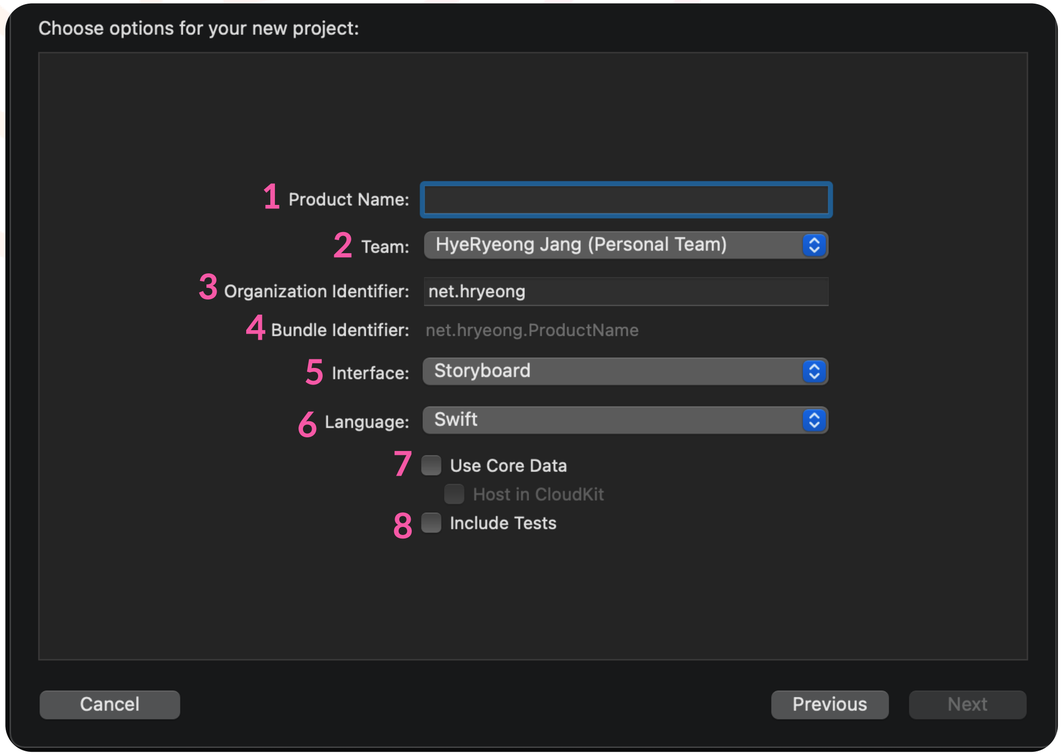
3. Organization Identifier

프로젝트 운영 기관 또는 조직의 식별 값을 입력합니다.
대체로 소속 회사나 해당 프로젝트의 도메인에서 www를 뺀 나머지 역순으로 사용합니다.
이 값은 Product Name 항목과 연결되어 'Bundle Identifier'항목이 완성됩니다.

4. Bundle Identifier

앱의 고유 식별 번호입니다.
Organization Identifier + Product Name의 조합으로 사용됩니다.

Xcode 사용하기 | 프로젝트 생성



5. Interface

Interface를 구현할 때, 사용되는 방법을 선택합니다.
Storyboard와 SwiftUI 중 Storyboard 방식을 선택해주세요:)

6. Language

앱을 개발할 때 사용할 언어를 선택합니다.
Objective-C와 Swift 두 개중 Swift를 선택해주세요 :)

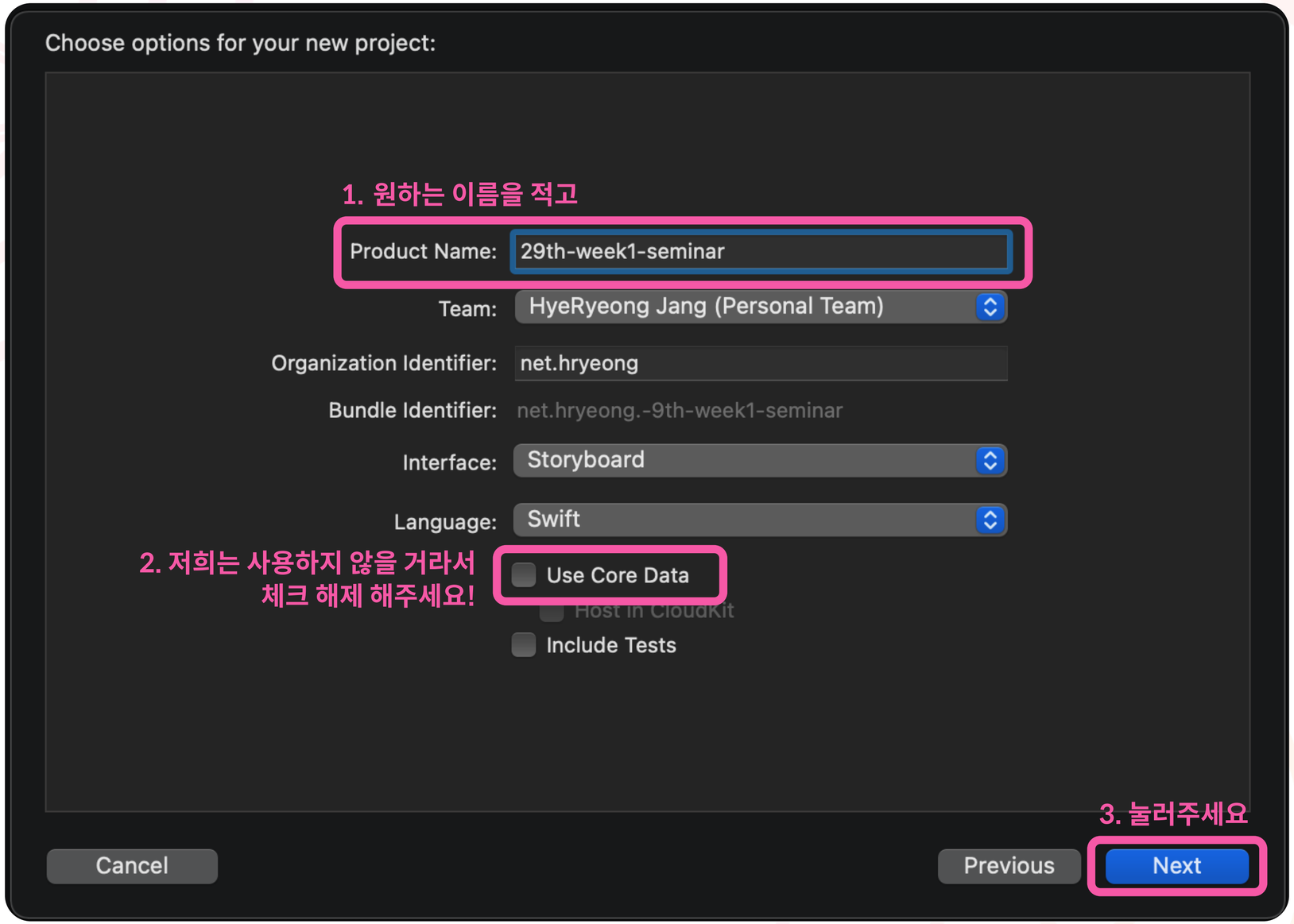
7.Use Code Data

기기 내부에 데이터를 저장해야 할 경우가 있을텐데요,
이 때, 데이터 저장을 지원하는 코어 데이터를 사용할지 묻는 옵션입니다.

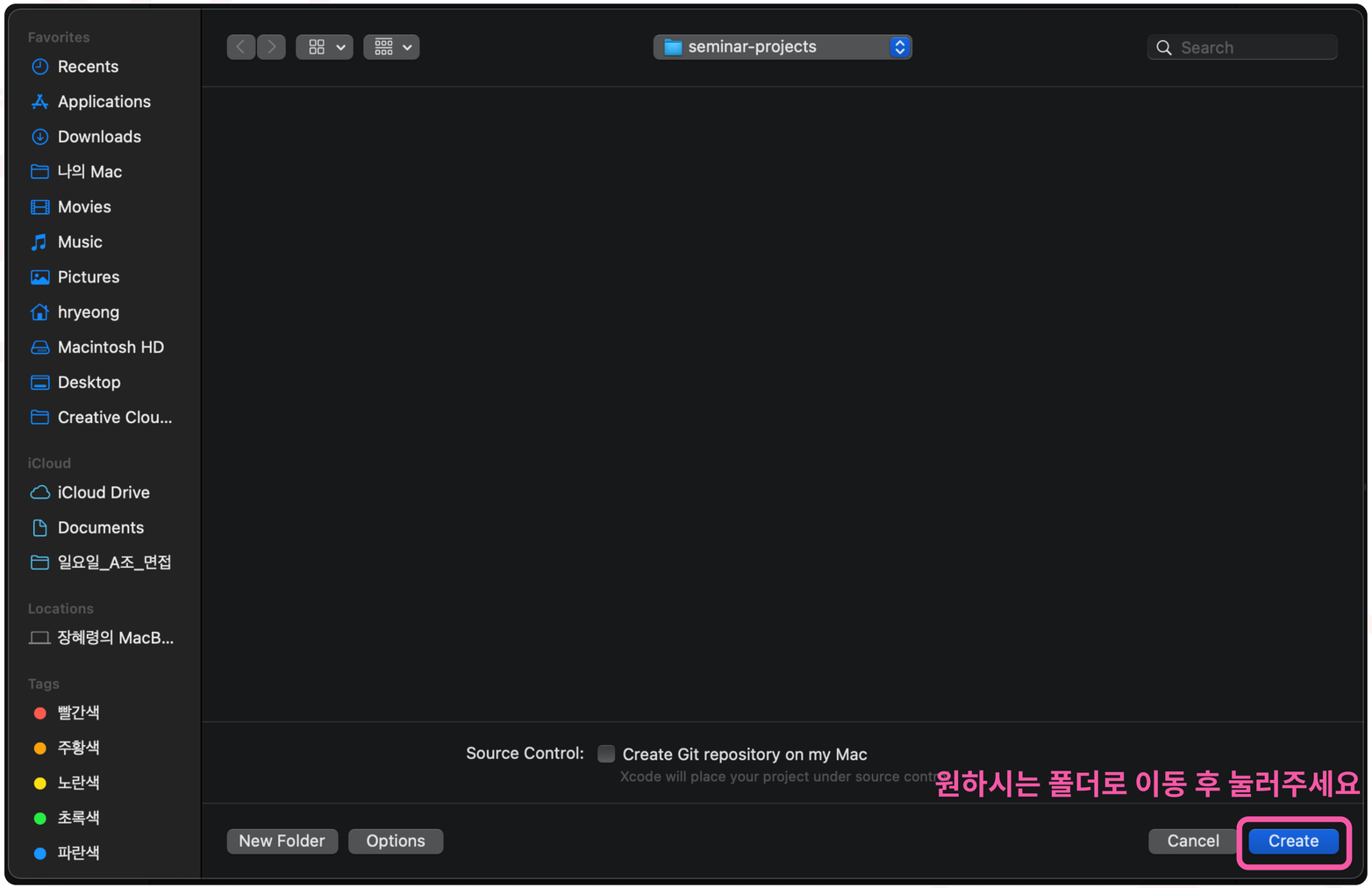
8.Include Tests

개발한 소프트웨어의 품질을 테스트 결과를 확인 할 수 있는 단위 테스트 기능과
UI 요소에 따른 테스트를 할 수 있습니다.

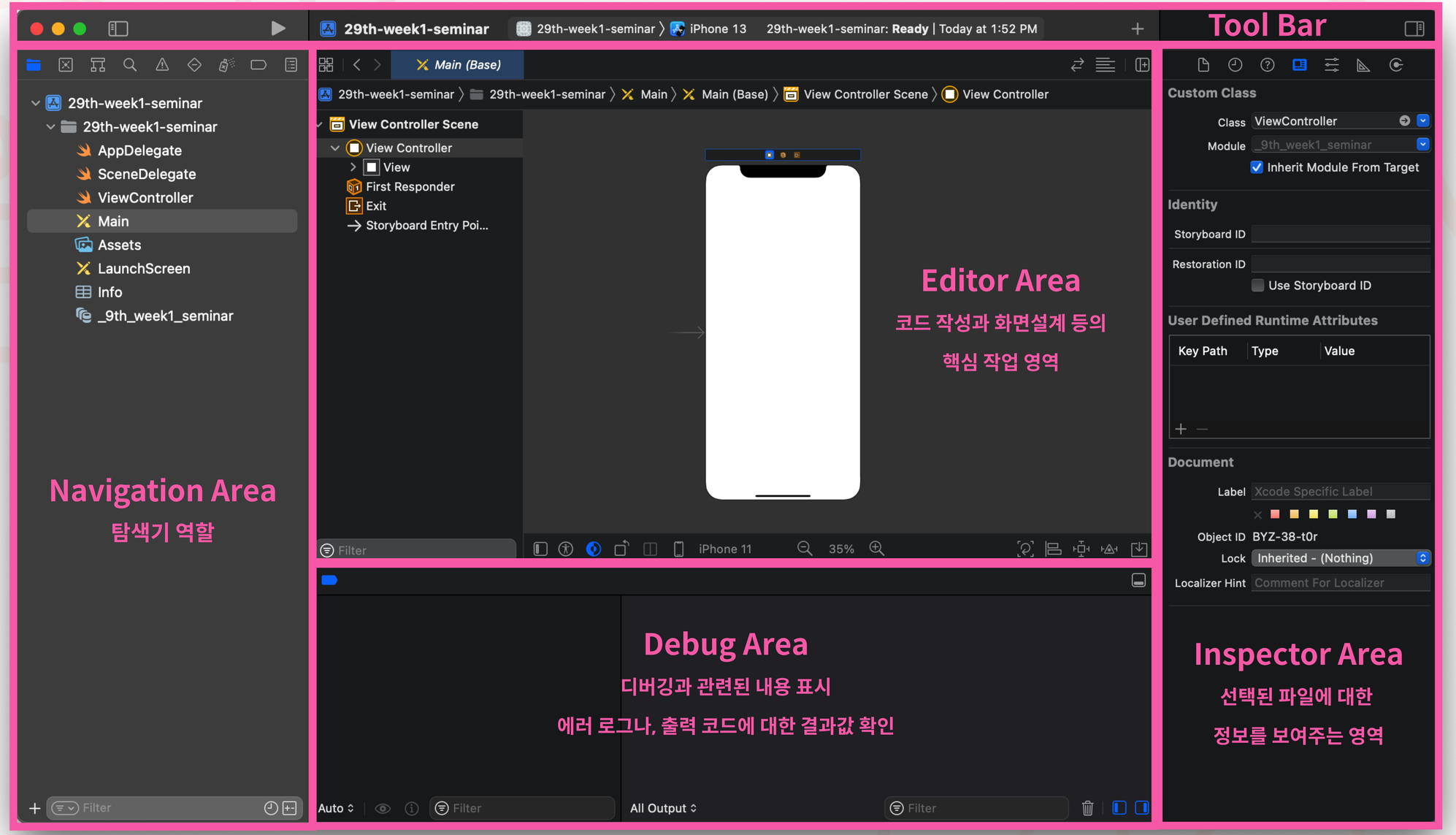
Xcode 사용하기 | 프로젝트 생성



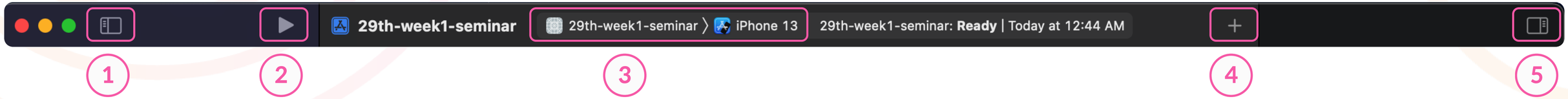
Xcode 사용하기 | 프로젝트 생성



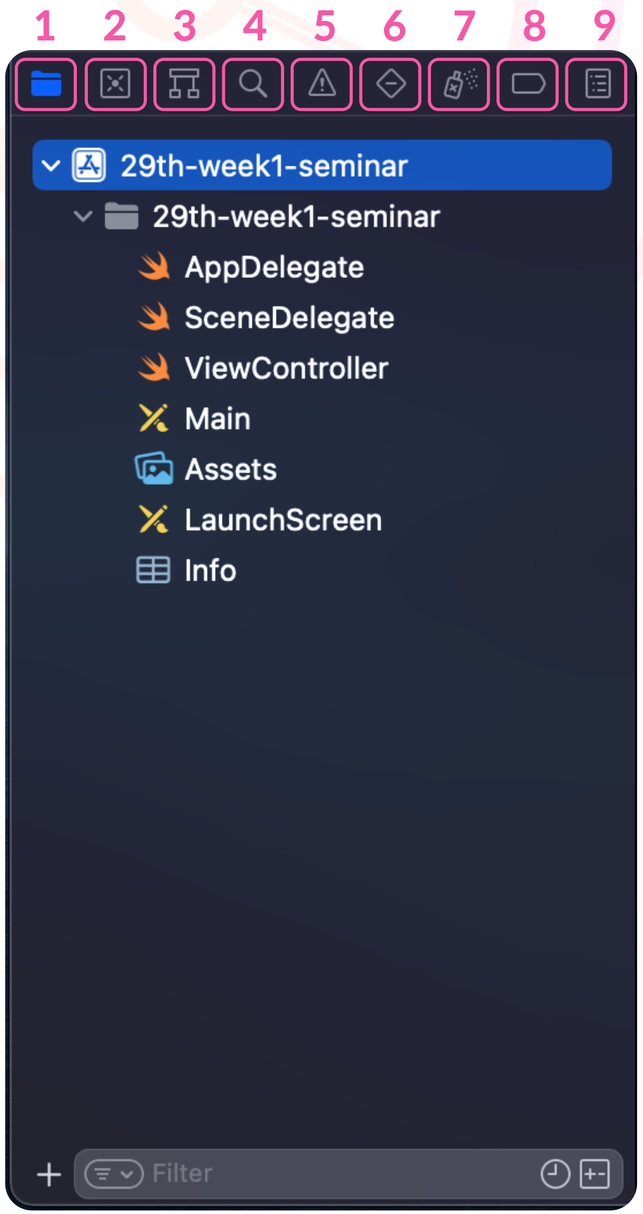
Xcode 사용하기 | 프로젝트 살펴보기



Tool Bar



1. Navigator Area Show / Hidden (command(⌘) + 0)
2. 실행 : 어플리케이션 실행하는 버튼 (실행을 하면 정지버튼 나타남)
3. Scheme Button : 앱을 실행할 시뮬레이터, 기기 등을 선택할 수 있는 버튼
4. Object Library Pane: 우리가 선택할 UI 요소들을 Palette (pane) 형태로 보여줌
5. Inspector Area Show / Hidden (command(⌘) + option(⌥) + 0)

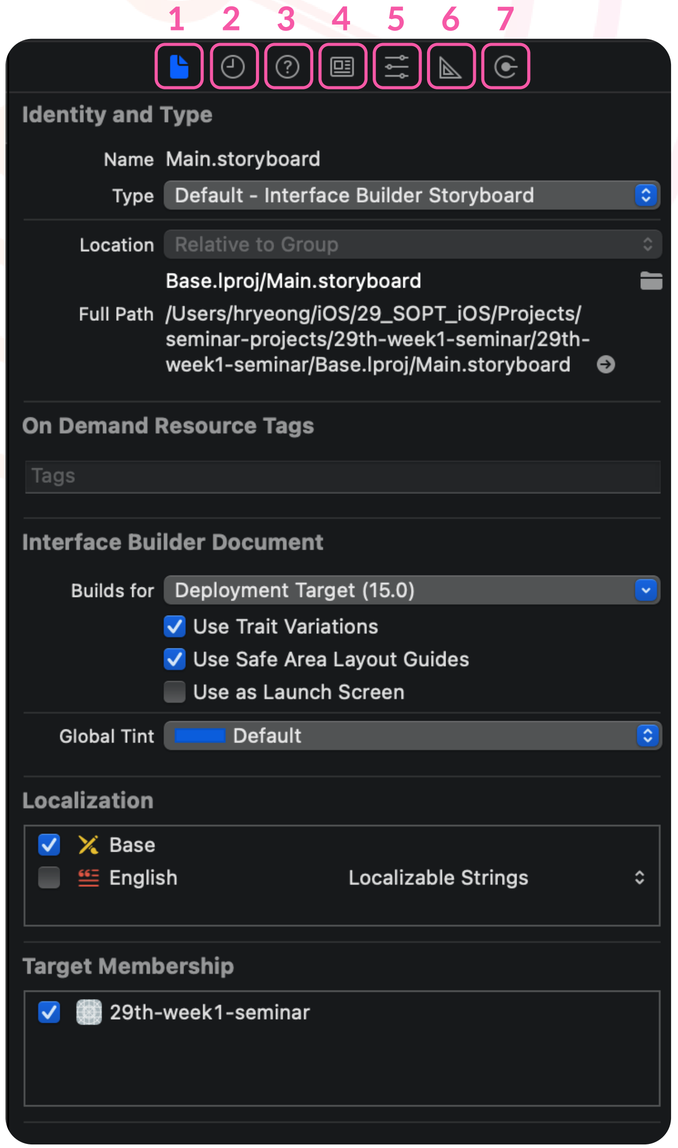


Navigator Area

1. **Project navigator** : 프로젝트 트리를 보여줍니다. 가장 통상적으로 사용됩니다.
2. **Source Control navigator** : git과 연결되어 버전관리에 대한 부분들을 보여줍니다.
3. **Symbol navigator** : 프로젝트의 클래스와 메서드 기준으로 정리해 보여줍니다.
4. **Find navigator** : 프로젝트 안 텍스트를 검색하거나 교체할 때 사용합니다.
5. **Issue navigator** : 에러나 경고 메시지를 모아서 보여줍니다.
6. **Test navigator** : 테스트 프로젝트 항목을 정리해서 보여줍니다.
7. **Debug navigator** : 디버그 관련 메시지를 모아서 보여줍니다.
8. **Breakpoint navigator** : 소스 코드에 설정된 브레이크 포인트 관련 내용을 모아서 보여줍니다.
9. **Report navigator** : 프로젝트 실행이나 빌드, 테스트 등 프로젝트 실행 내역을 정리해서 보여줍니다.

* command(⌘) + 1~9

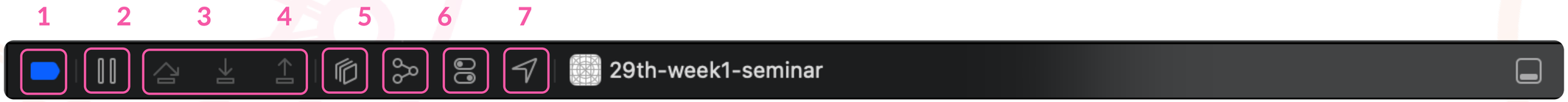
Inspector Area



1. **File Inspector** : 선택한 파일의 타입 정보를 표시하거나 타깃 운영체제 버전 등을 설정
2. **History Inspector** : 버전 관리 도구를 사용하고 있다면 선택한 파일의 이력을 한눈에 확인 가능
3. **Quick Help Inspector** : 선택한 개체와 관련된 간단한 도움말 표시. 해당 개체의 클래스 정보 등을 알려줌
4. **Identity Inspector** : 선택한 개체의 클래스 정보를 표시하거나 설정
5. **Attributes Inspector** : 선택한 개체의 세부 정보를 표시하거나 설정. ex. 텍스트 크기, 캡션, 색상 등
6. **Size Inspector** : 선택한 개체의 위치나 크기를 표시, 설정함
7. **Connections Inspector** : 선택한 개체의 코드 연결 정보나 동작 정보를 표시하거나 설정

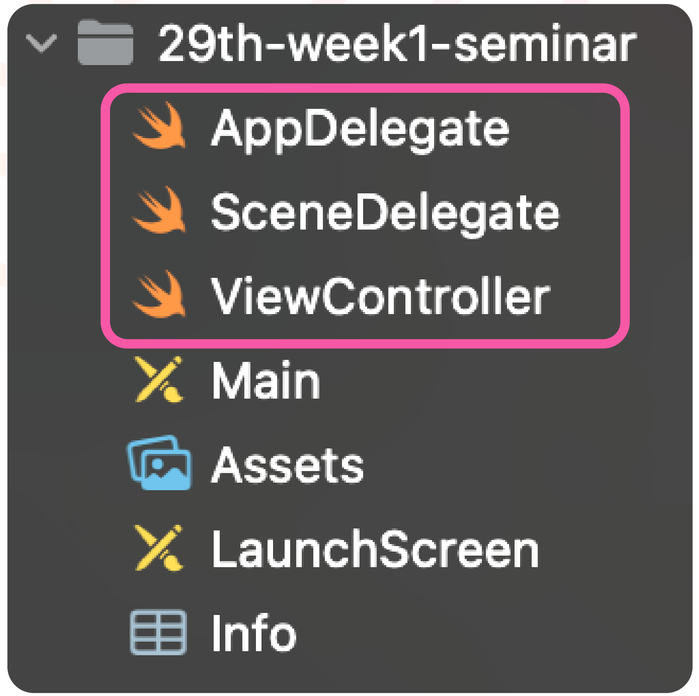
* command(⌘) + option(⌥) + 1~7

Debug Area



1. **Breakpoint Activation button.** : 앱의 모든 중단점을 동시에 비활성화 및 활성화하는 토글 역할
2. **Continue/Pause button** : 일시 중지 버튼을 클릭하여 앱 실행을 일시 중지시키거나 다시 실행시킴
3. **Step controls** : 스텝 오버, 스텝 인, 스탭 아웃을 사용하며 단계별로 코드 실행
   step over : 한줄을 실행합니다. 함수가 있어도 실행 후 다음으로 넘어감
   step in : 함수 내부로 들어감
   step out : 함수를 끝까지 실행시키고 호출시킨 곳으로 되돌아 감
4. **Debug View Hierarchy button** : 3D 렌더링과 디버그 탐색기의 계층 목록을 보여줌
5. **Debug Memory Gragh** : 앱이 사용하는 메모리 영역과 각 영역의 크기를 보여줌.
   그래프의 노드는 개체, 힙 할당 또는 메모리 매핑된 파일을 나타냄.
6. **Environment Override** : 디버거에서 직접 이러한 공통 사용자 인터페이스 설정을 쉽게 재정의 할 수 있는 도구 모음
7. **Simulate Location button** : 기기 또는 시뮬레이터가 앱이 실행될 때 해당 위치를 앱에 보고하도록 함

Xcode 사용하기 | 파일 살펴보기

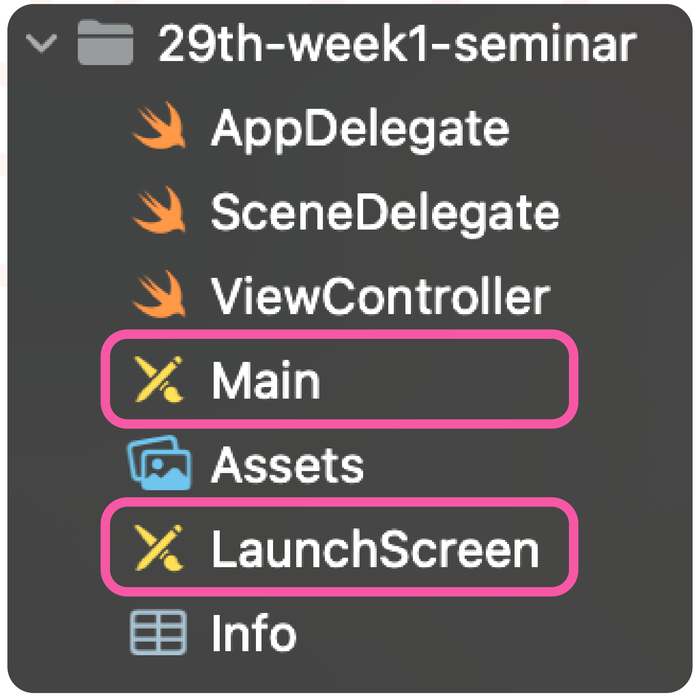


📌 *.swift

- Swift 클래스 파일로, 앱의 소스 코드를 구성하는 역할

- AppDelegate 와 SceneDelegate는 앱 전체의 생명주기를 관리하는 역할

- 프로젝트를 생성하면 기본으로 ViewController가 생성이 되는데,
  일반적으로 화면의 개수만큼 ViewController가 필요함



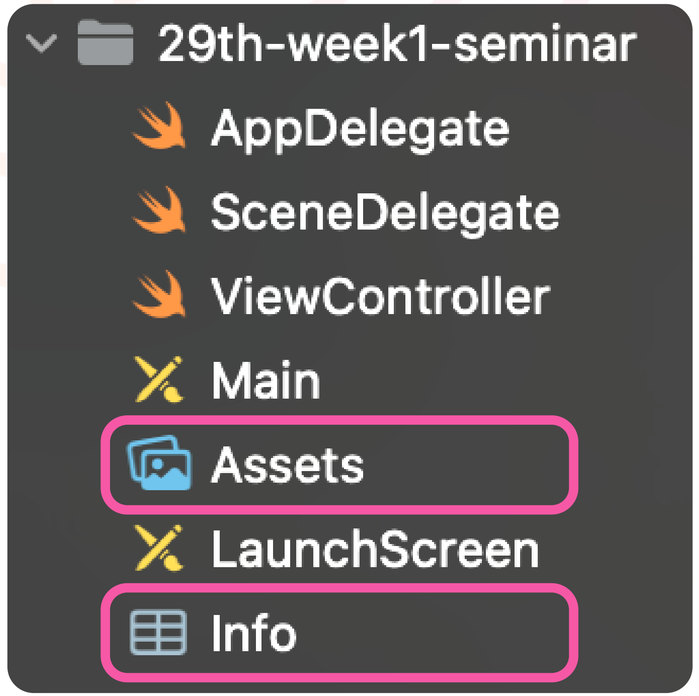
📌 *.storyboard

- 유저 인터페이스를 종합적으로 구현하는 역할

- 프로젝트 생성 시 기본으로 생성되는 Main.storyboard는
  앱의 사용자 인터페이스 설계를 담당

- LaunchScreen.storyboard 파일은 앱을 실행하면
  처음 나타나는 시작 화면을 구성하는 데에 사용 (Splash 라고도 함)

Xcode 사용하기 | 파일 살펴보기



📌 Assets.xcassets

- 이미지 등의 리소스를 관리

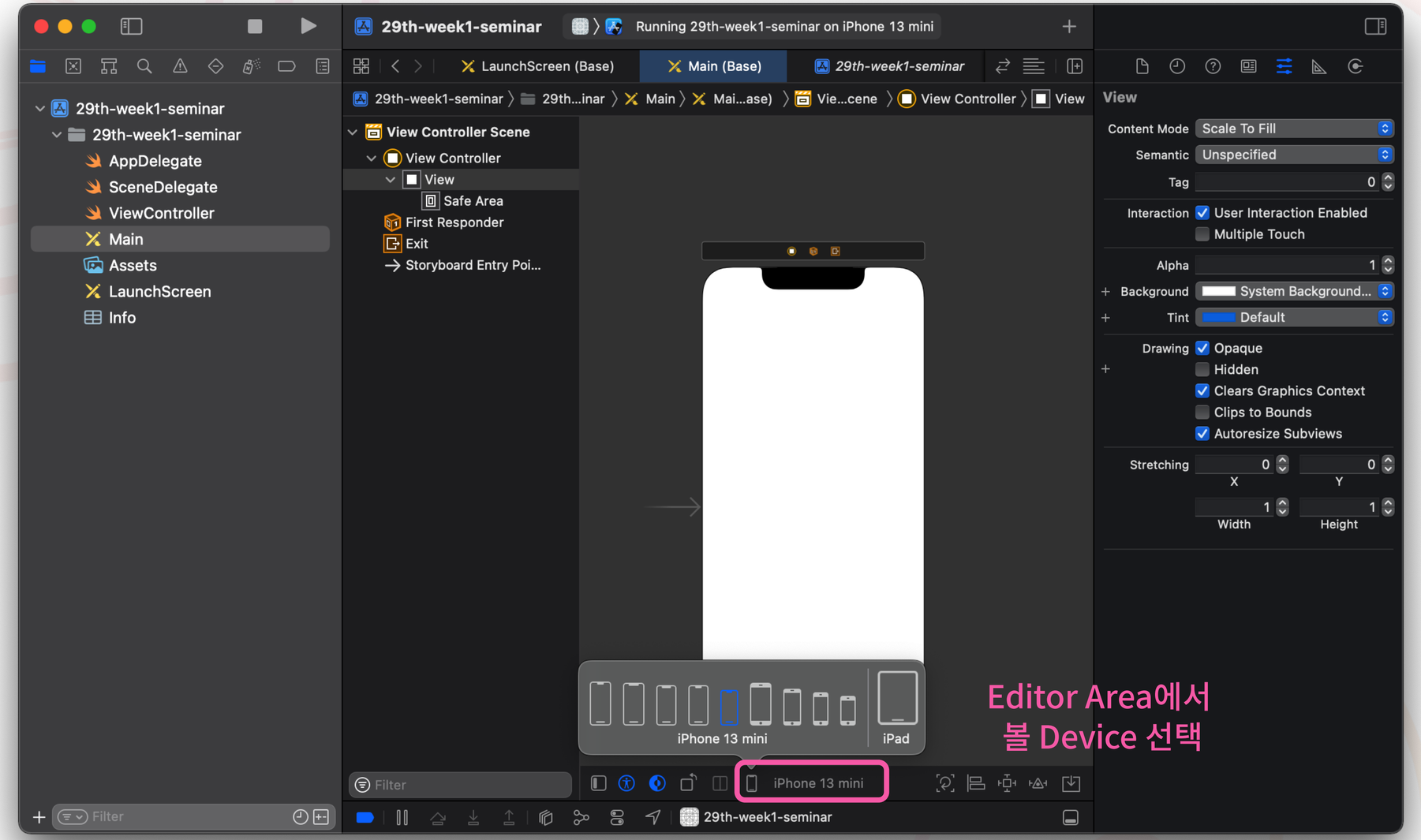
📌 Info.plist

- 프로젝트 설정을 담당한다.
  (ex 네트워크 접근 허용, 카메라 접근 허용, 시작화면 설정)

03

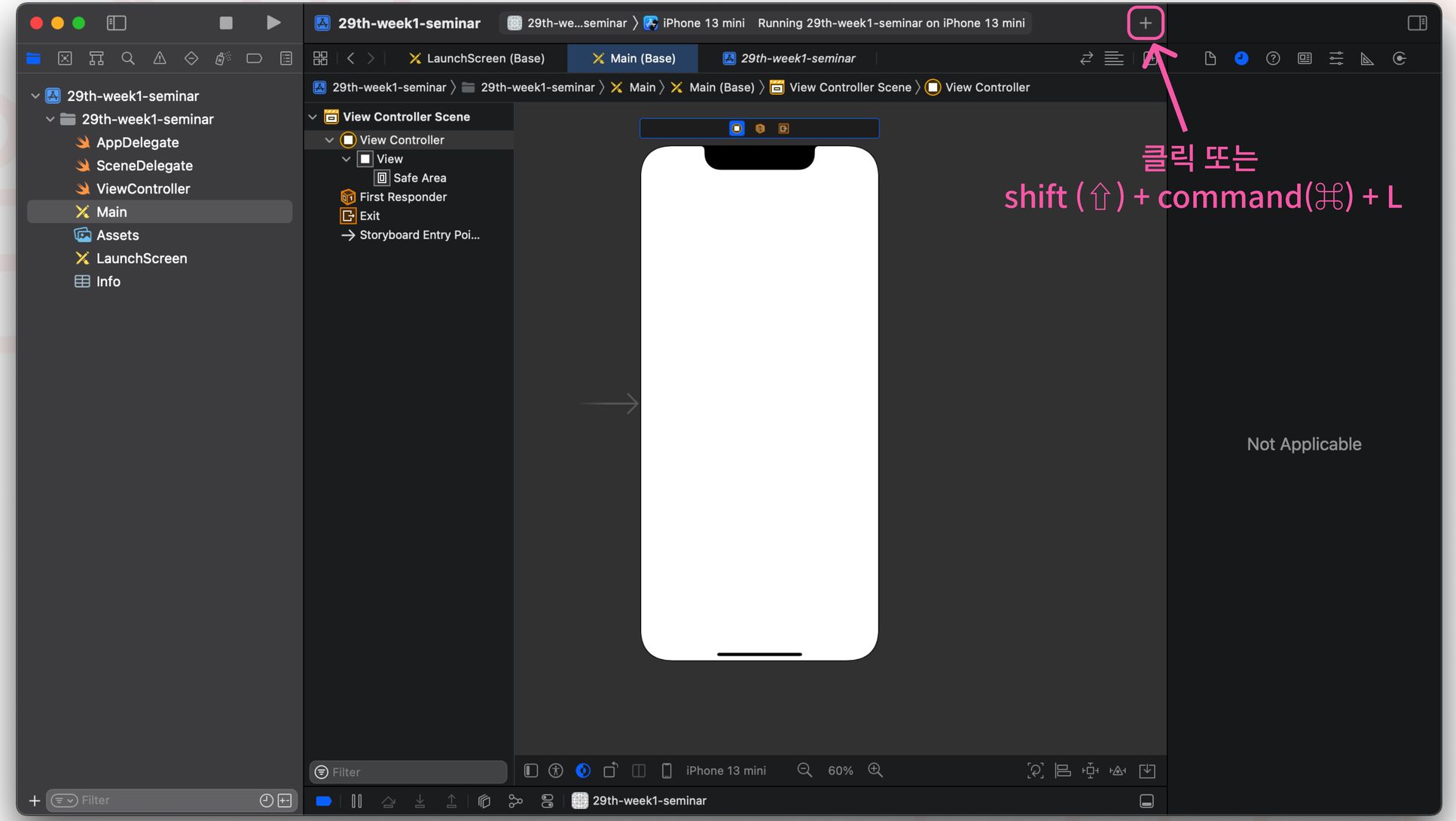
UI 구성하기

UI 구성하기 | 실습

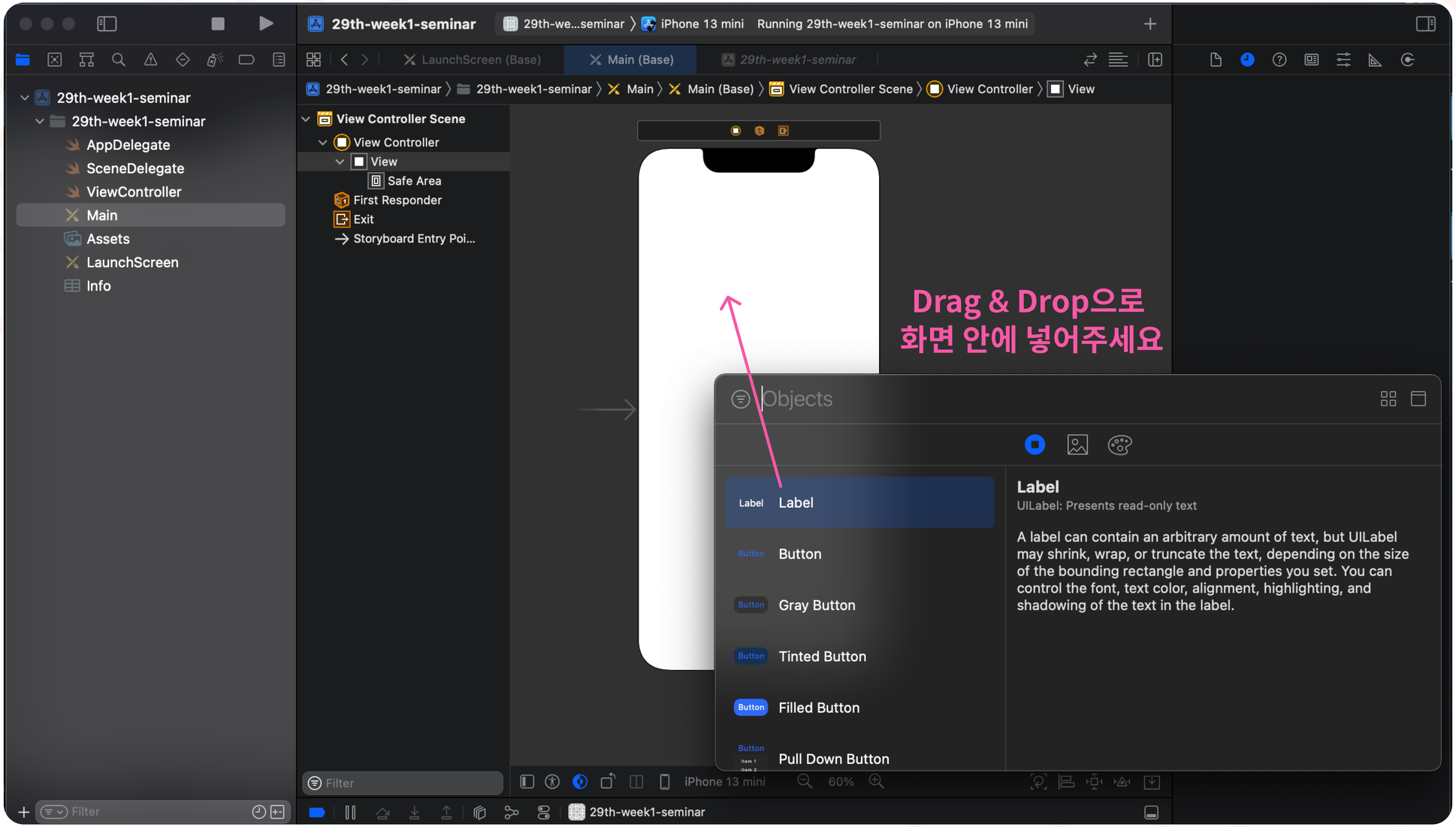


Editor Area에서
볼 Device 선택

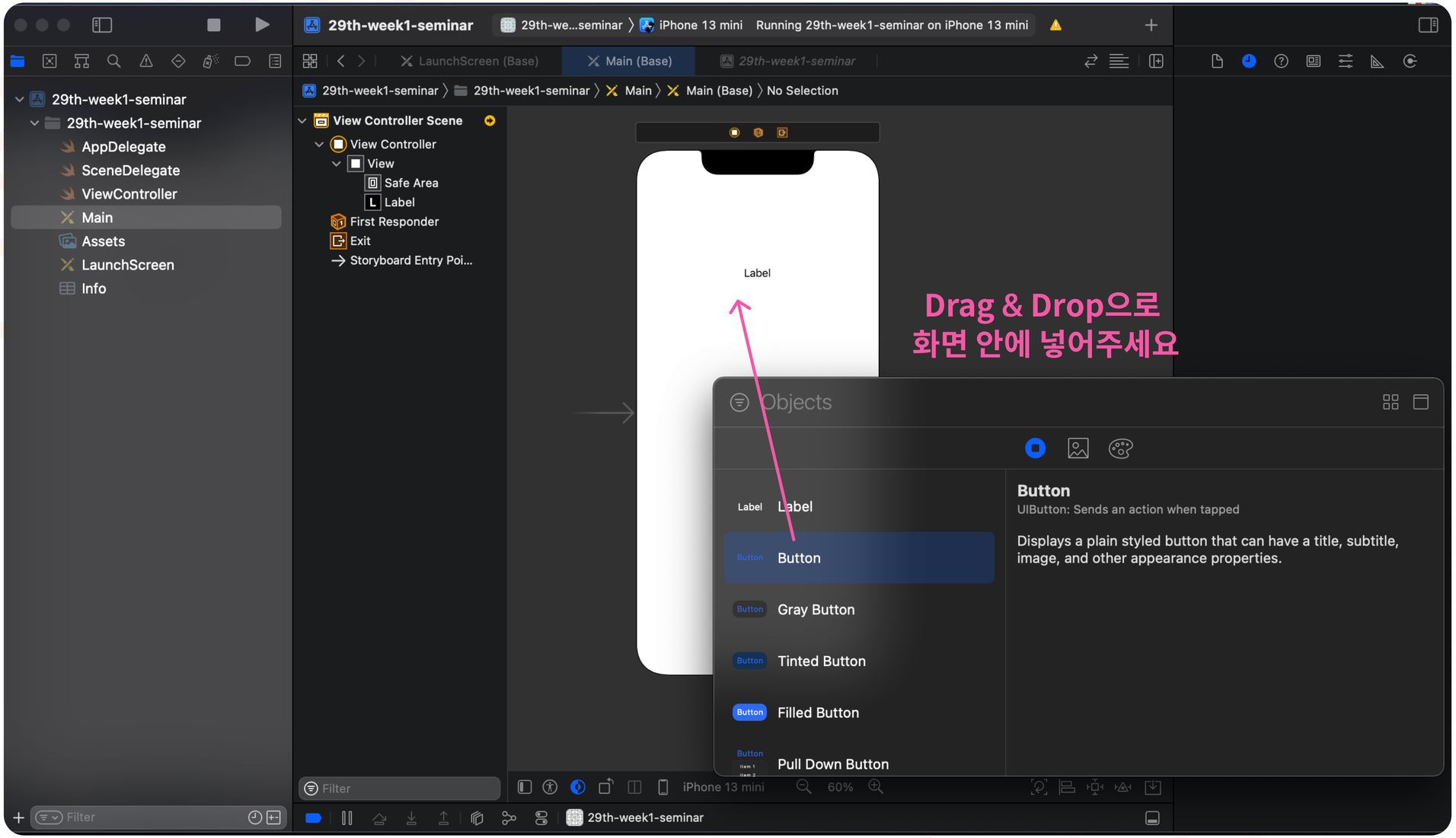
UI 구성하기 | 실습


29th-week1-seminar
29th-we...seminar
iPhone 13 mini
Running 29th-week1-seminar on iPhone 13 mini
LaunchScreen (Base)
Main (Base)
29th-week1-seminar
29th-week1-seminar
29th-week1-seminar
Main
Main (Base)
View Controller Scene
View Controller
29th-week1-seminar
AppDelegate
SceneDelegate
ViewController
Main
Assets
LaunchScreen
Info
View Controller Scene
View Controller
View
Safe Area
First Responder
Exit
Storyboard Entry Poi...
클릭 또는
shift (⇧) + command(⌘) + L
Not Applicable
Filter
iPhone 13 mini
60%
29th-week1-seminar

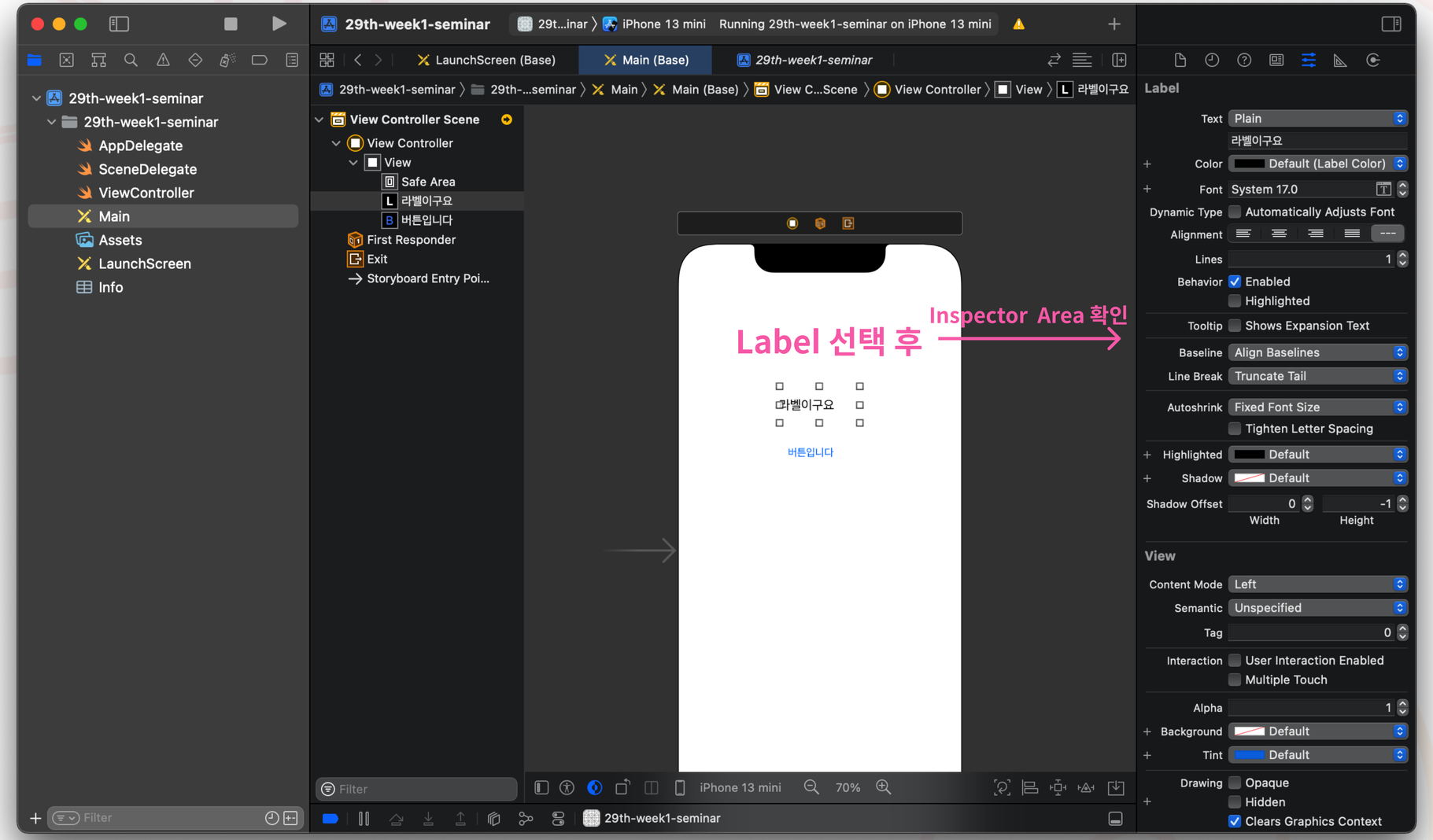
UI 구성하기 | 실습


29th-week1-seminar
29th-we...seminar
iPhone 13 mini
Running 29th-week1-seminar on iPhone 13 mini
LaunchScreen (Base)
Main (Base)
29th-week1-seminar
29th-week1-seminar
29th-week1-seminar
AppDelegate
SceneDelegate
ViewController
Main
Assets
LaunchScreen
Info
Main
Main (Base)
View Controller Scene
View Controller
View
View Controller Scene
View Controller
View
Safe Area
First Responder
Exit
Storyboard Entry Poi...
Drag & Drop으로
화면 안에 넣어주세요
Objects
Label
Button
Gray Button
Tinted Button
Filled Button
Pull Down Button
Label
UILabel: Presents read-only text
A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the label.
Filter
iPhone 13 mini
60%
29th-week1-seminar

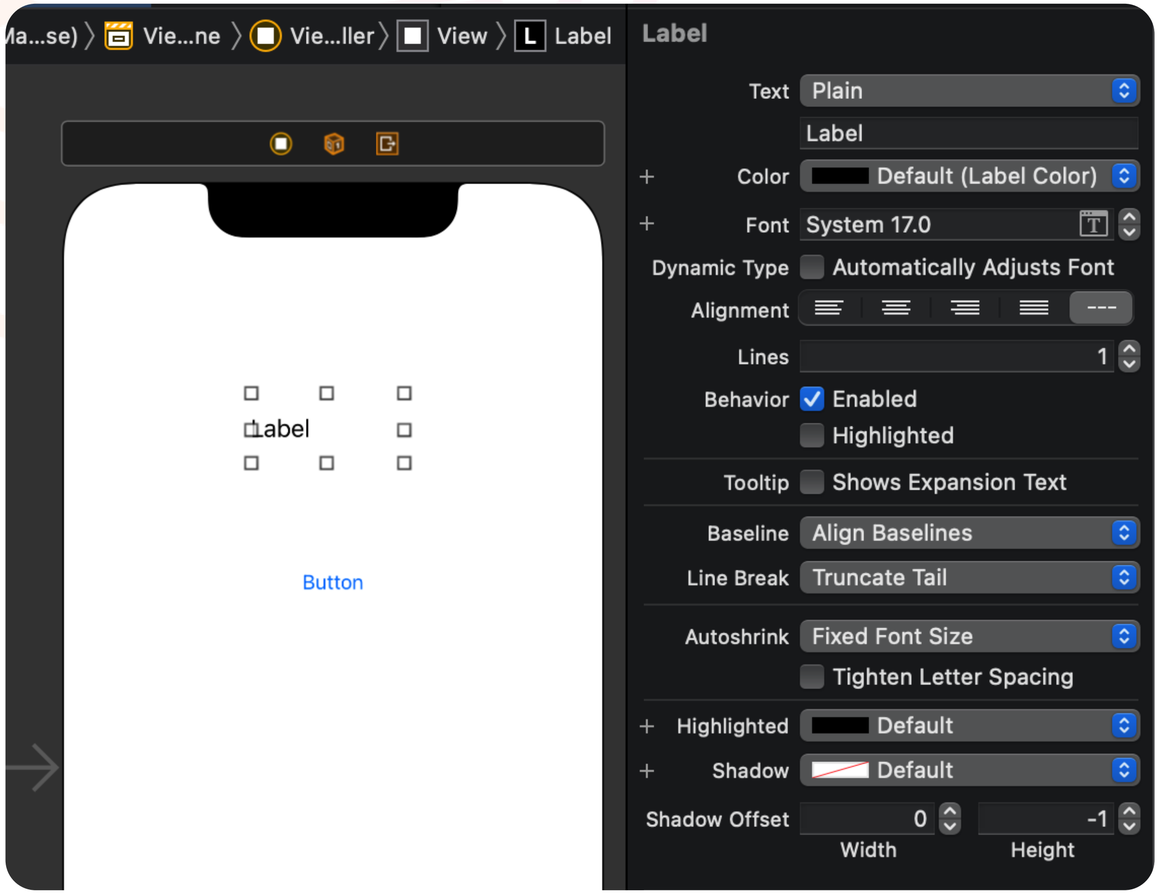
UI 구성하기 | 실습


29th-week1-seminar
29th-we...seminar
iPhone 13 mini
Running 29th-week1-seminar on iPhone 13 mini
LaunchScreen (Base)
Main (Base)
29th-week1-seminar
29th-week1-seminar
29th-week1-seminar
AppDelegate
SceneDelegate
ViewController
Main
Assets
LaunchScreen
Info
View Controller Scene
View Controller
View
Safe Area
Label
First Responder
Exit
Storyboard Entry Poi...
Label
Drag & Drop으로
화면 안에 넣어주세요
Objects
Label
Button
Gray Button
Tinted Button
Filled Button
Pull Down Button
Button
UIButton: Sends an action when tapped
Displays a plain styled button that can have a title, subtitle, image, and other appearance properties.
iPhone 13 mini
60%
29th-week1-seminar

UI 구성하기 | 실습


29th-week1-seminar
iPhone 13 mini
Running 29th-week1-seminar on iPhone 13 mini
LaunchScreen (Base)
Main (Base)
29th-week1-seminar
29th-week1-seminar
AppDelegate
SceneDelegate
ViewController
Main
Assets
LaunchScreen
Info
View Controller Scene
View Controller
View
Safe Area
라벨이구요
버튼입니다
First Responder
Exit
Storyboard Entry Poi...
Label 선택 후
Inspector Area 확인
라벨이구요
버튼입니다
Label
Text Plain
라벨이구요
Color Default (Label Color)
Font System 17.0
Dynamic Type Automatically Adjusts Font
Alignment
Lines 1
Behavior Enabled
Highlighted
Tooltip Shows Expansion Text
Baseline Align Baselines
Line Break Truncate Tail
Autoshrink Fixed Font Size
Tighten Letter Spacing
Highlighted Default
Shadow Default
Shadow Offset 0 -1
Width Height
View
Content Mode Left
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context

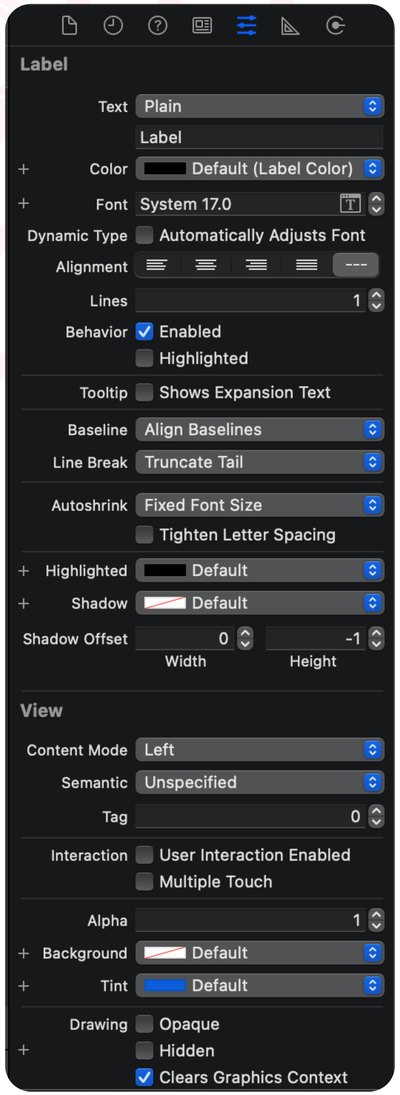
UI 구성하기 | 실습



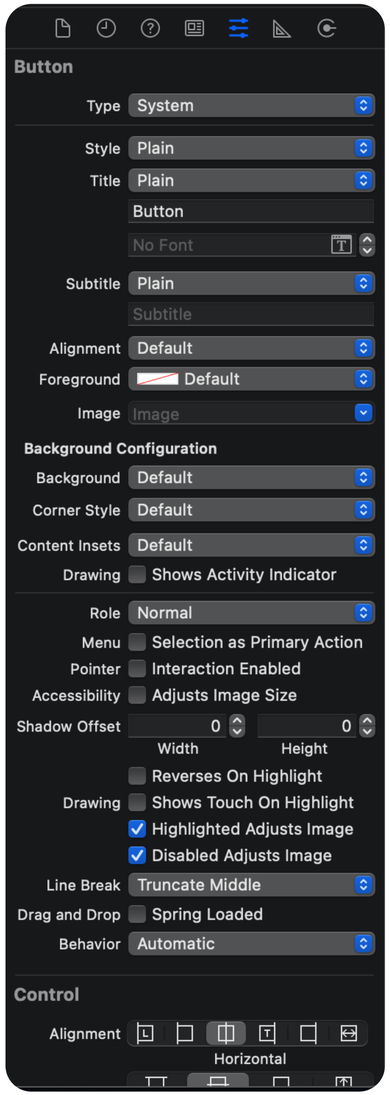
- Text Style : 텍스트 스타일을 정의. Attributed 를 선택하면 세밀한 스타일 조정이 가능
- Text : 원하는 텍스트 내용을 넣을 수 있음.
- Color : 텍스트 컬러 변경
- Font : 폰트 변경
- Dynamic Type : 사용자가 글씨 크기를 조정했을 때, 동적으로 크기 조절 유무
- Alignment : 글자 정렬 (LtR,RtL,중앙정렬등) 방향
- Lines : 최대로 들어갈 줄 수 (제한을 두고 싶지 않다면, 0)

UI 구성하기 | 실습

Label



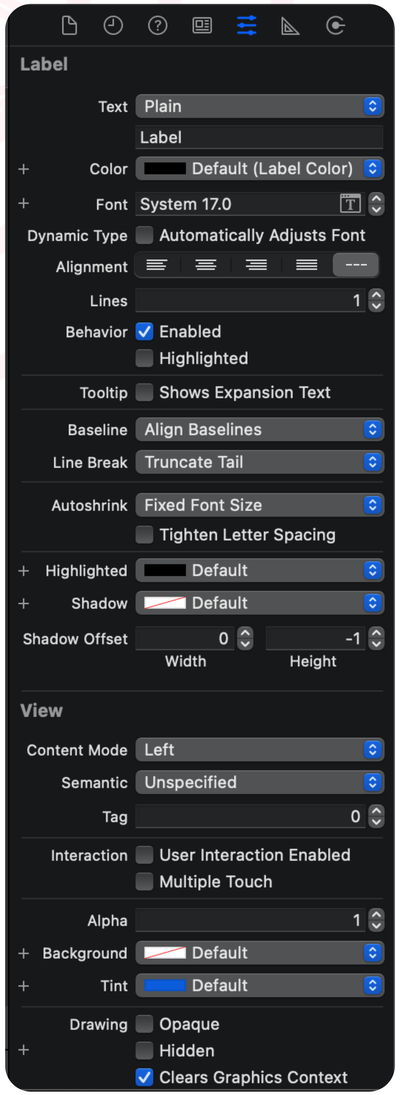
Button



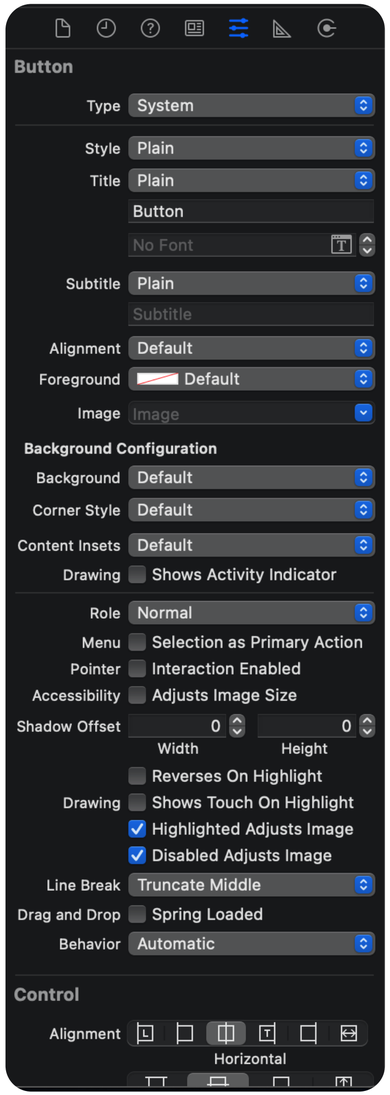
자유롭게 Label과 Button
텍스트, 색상, 정렬, 폰트 등 바꾸어봅시다!

UI 구성하기 | 실습

Label



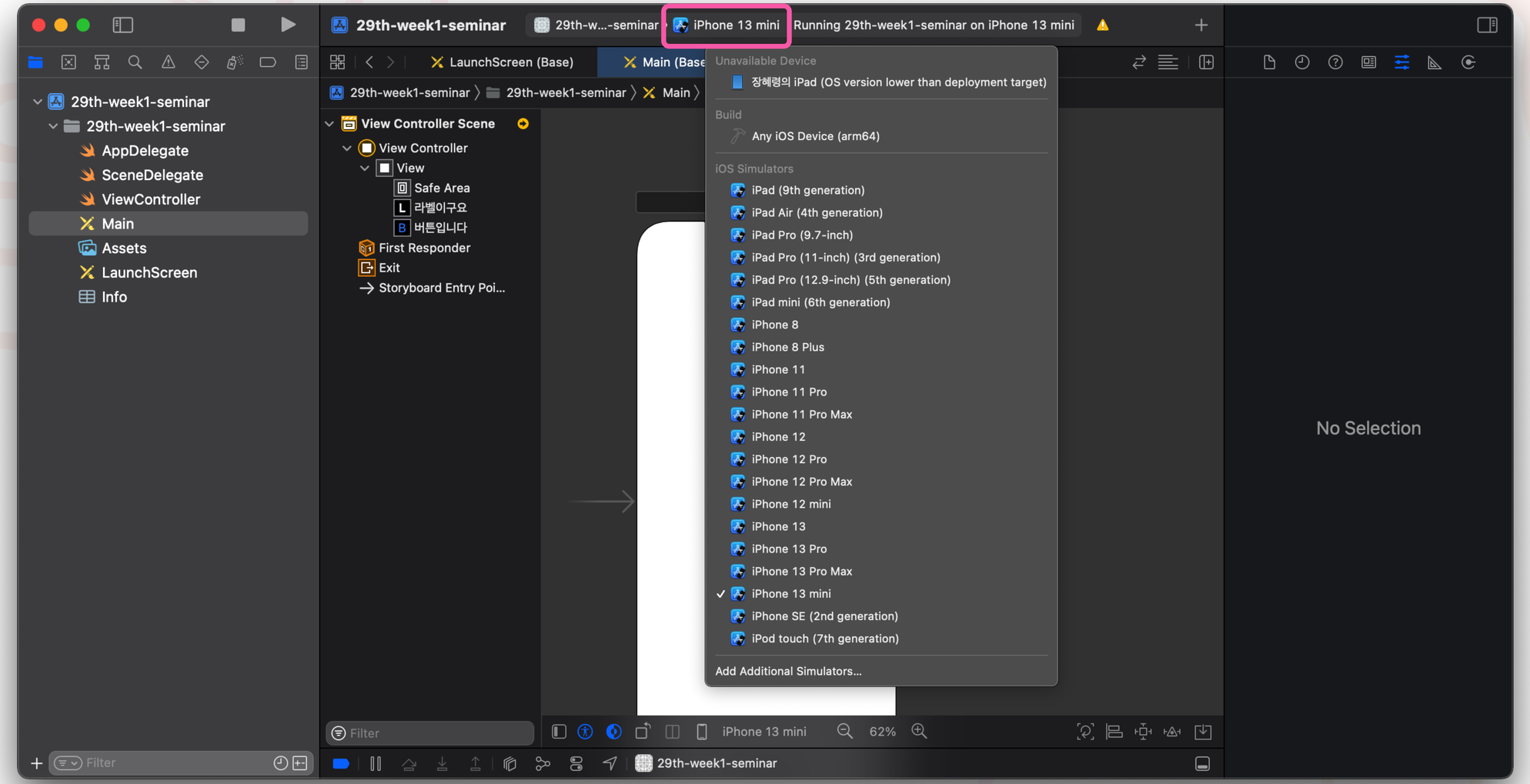
Button



자유롭게 Label과 Button
텍스트, 색상, 정렬, 폰트 등 바꾸어봅시다!

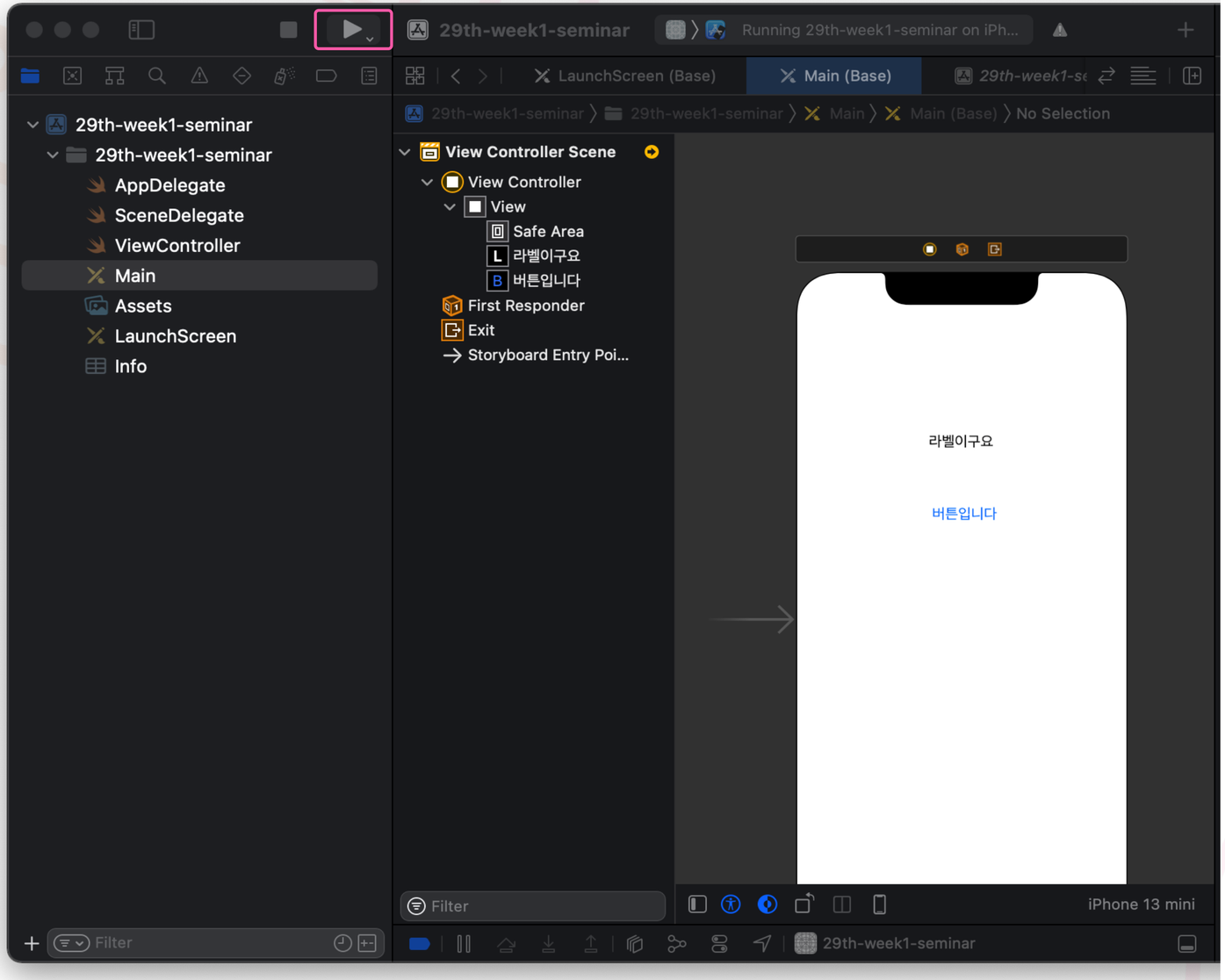
UI 구성하기 | 실습

눌러서 원하는 시뮬레이터를 선택해주세요!

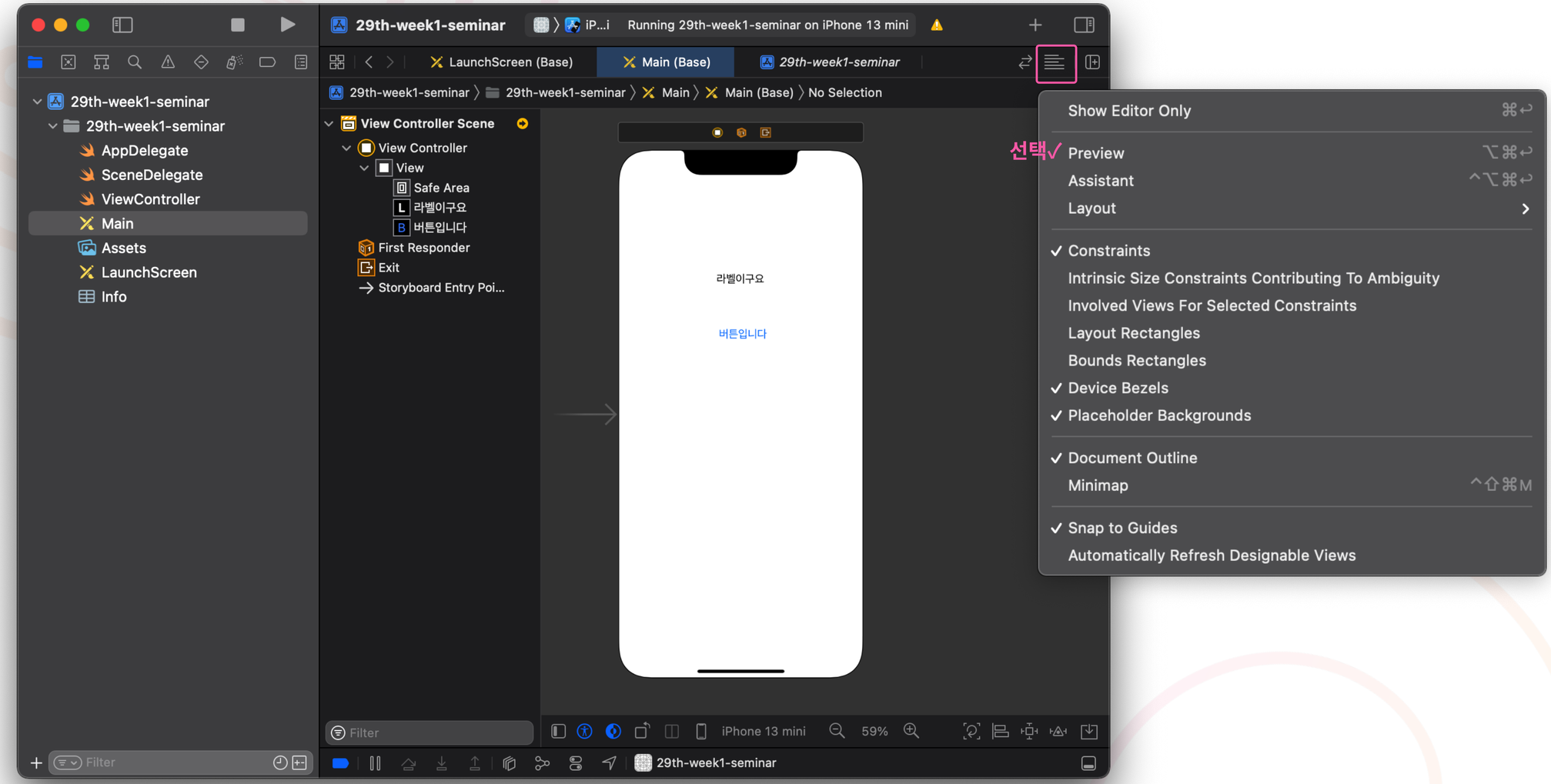


UI 구성하기 | 실습

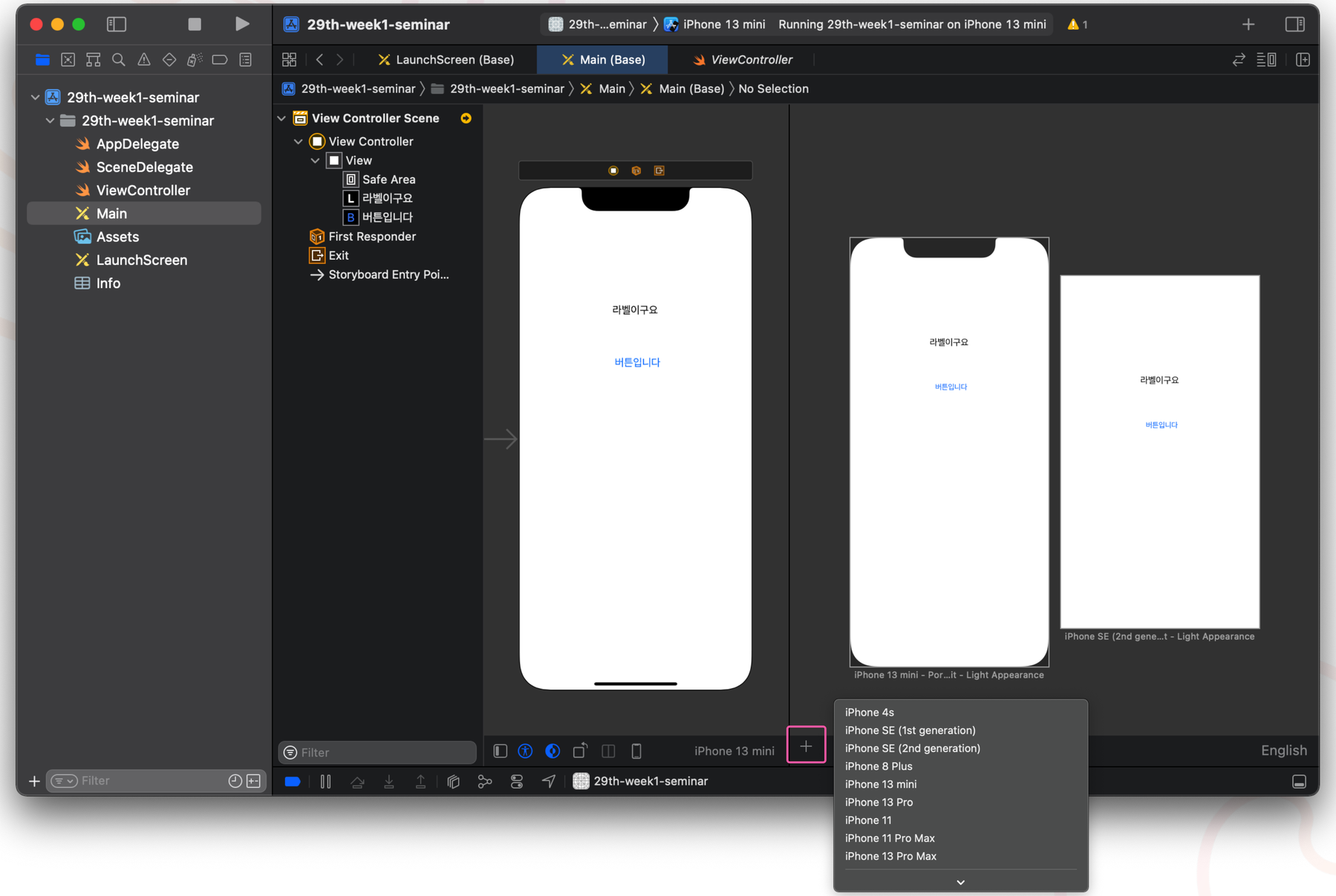
버튼을 누르거나 command(⌘) + R을 눌러주세요



UI 구성하기 | 실습


29th-week1-seminar
iP...i
Running 29th-week1-seminar on iPhone 13 mini
LaunchScreen (Base)
Main (Base)
29th-week1-seminar
29th-week1-seminar 〉 29th-week1-seminar 〉 Main 〉 Main (Base) 〉 No Selection
29th-week1-seminar
29th-week1-seminar
AppDelegate
SceneDelegate
ViewController
Main
Assets
LaunchScreen
Info
Filter
View Controller Scene
View Controller
View
Safe Area
라벨이구요
버튼입니다
First Responder
Exit
Storyboard Entry Poi...
라벨이구요
버튼입니다
iPhone 13 mini
59%
29th-week1-seminar
선택✓
Show Editor Only
Preview
Assistant
Layout
Constraints
Intrinsic Size Constraints Contributing To Ambiguity
Involved Views For Selected Constraints
Layout Rectangles
Bounds Rectangles
Device Bezels
Placeholder Backgrounds
Document Outline
Minimap
Snap to Guides
Automatically Refresh Designable Views

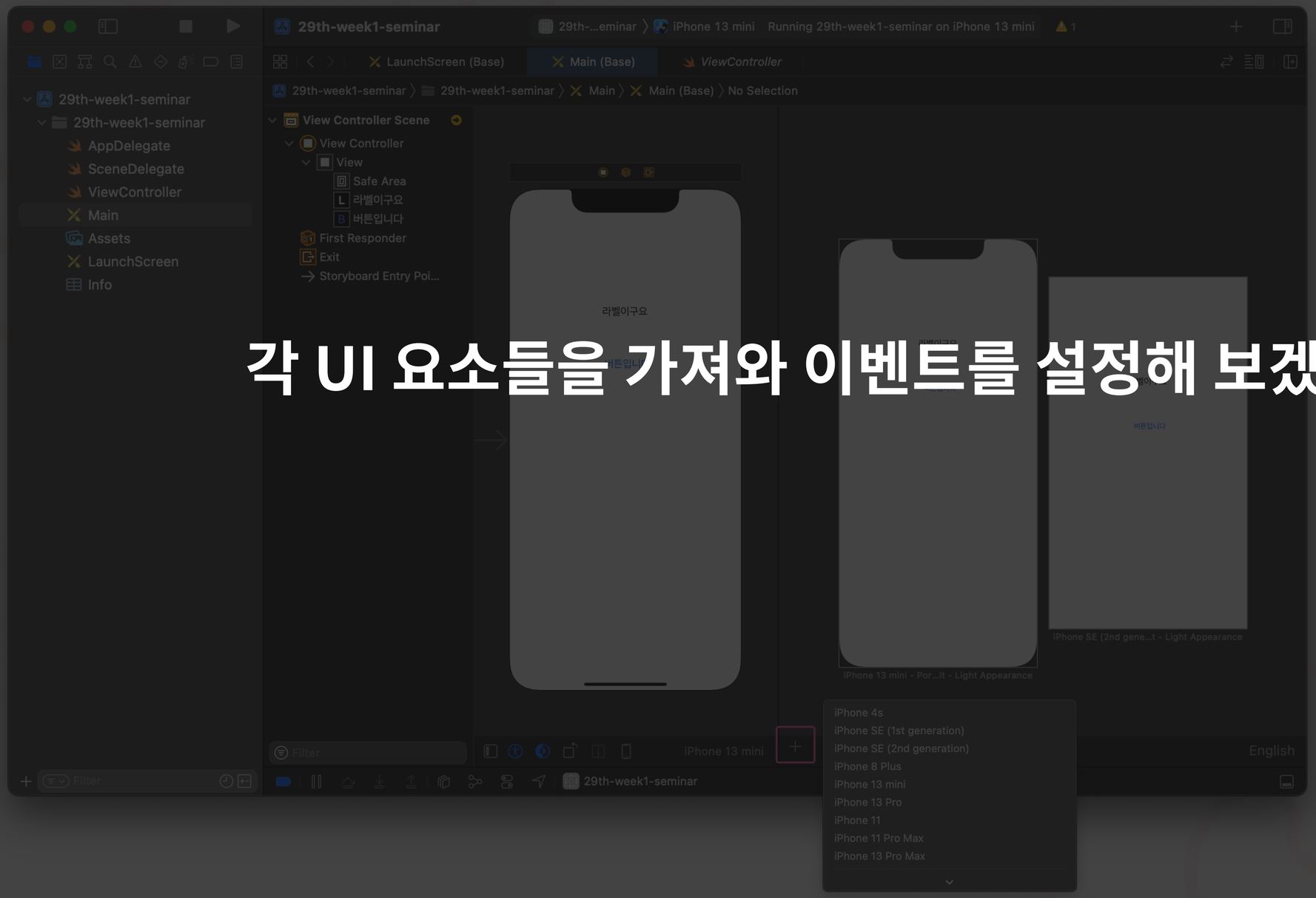
UI 구성하기 | 실습



시뮬레이터를
빌드하지 않더라도

Preview 기능을 활용해
각 기기에서 어떻게 보이는지
빠르게 파악이 가능

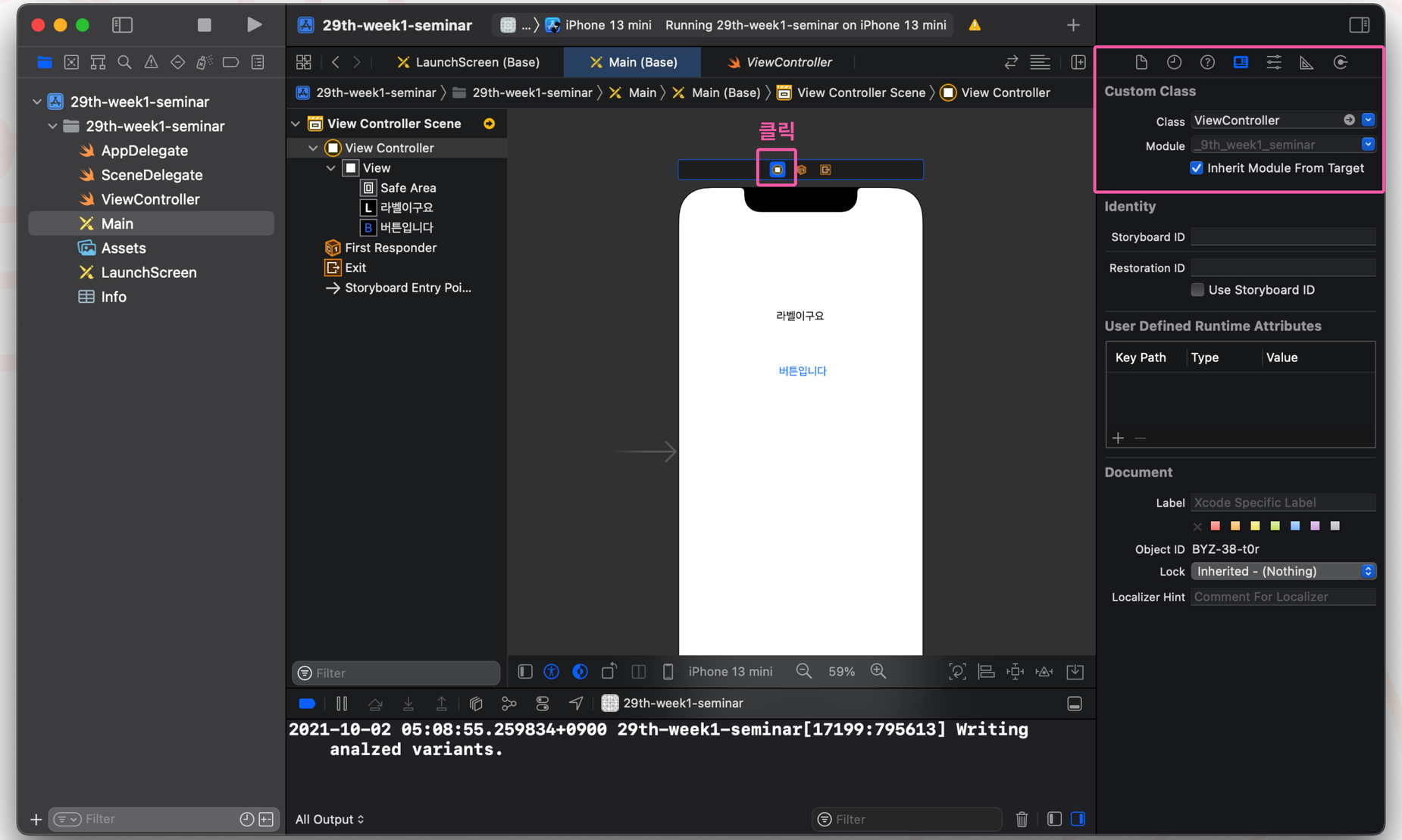
UI 구성하기 | 실습



각 UI 요소들을 가져와 이벤트를 설정해 보겠습니다

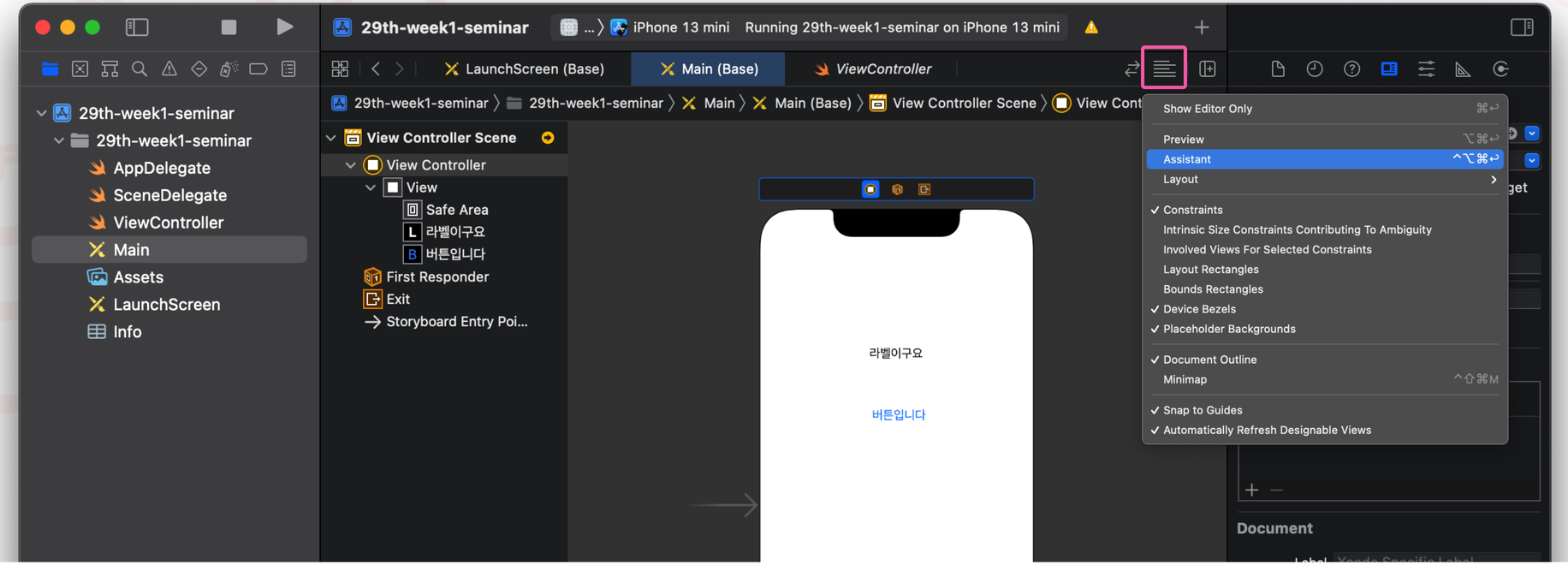
시뮬레이터를
빌드하지 않더라도
Preview 기능을 활용해
각 기기에서 어떻게 보이는지
빠르게 파악이 가능

UI 구성하기 | 실습



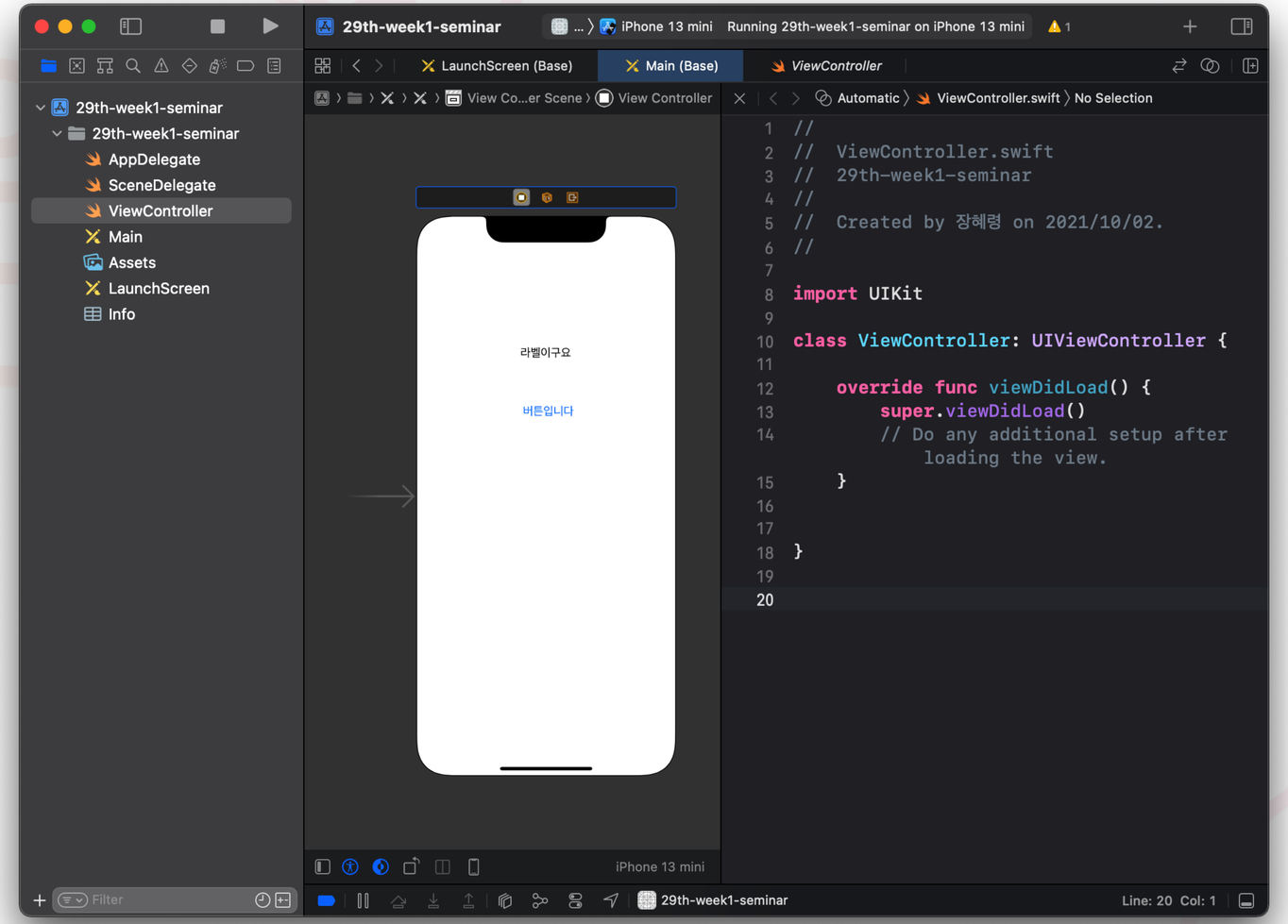
연결된
ViewController 확인

UI 구성하기 | 실습



Assistant Editor(control(^) + option(⌥) + command(⌘) + enter)

UI 구성하기 | 실습



화면과 연결된
ViewController를 편집할 수
있는 창이 나타남

화면 상 요소 제어하기 위한 2가지 타입 (Interface Builder Annotation)

@IBOutlet

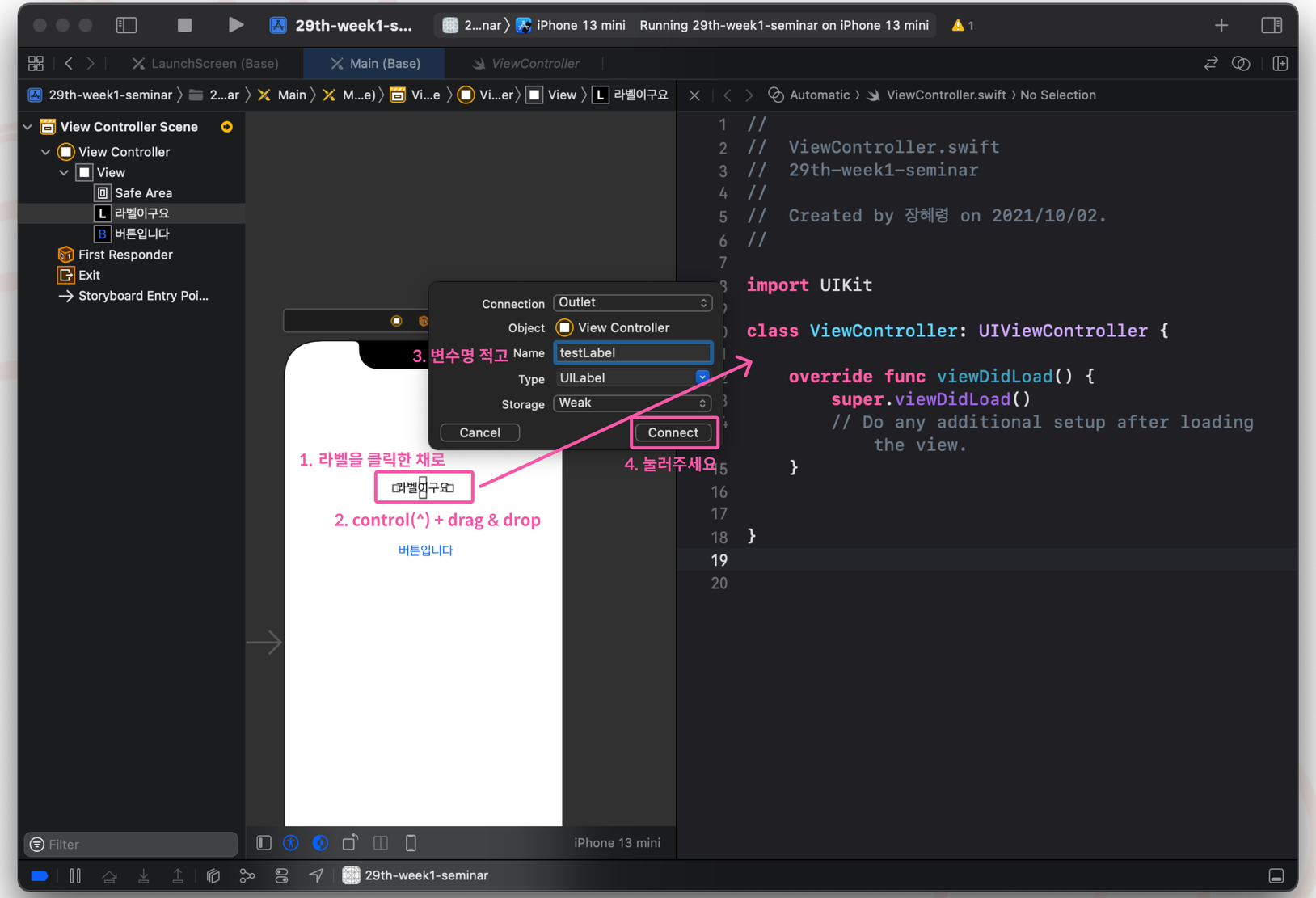
- 처리 결과를 View에 알리고 원하는 동작을 이끔
- View에 존재하는 요소와 Cotroller의 연결하기 위한 **변수**

@IBAction

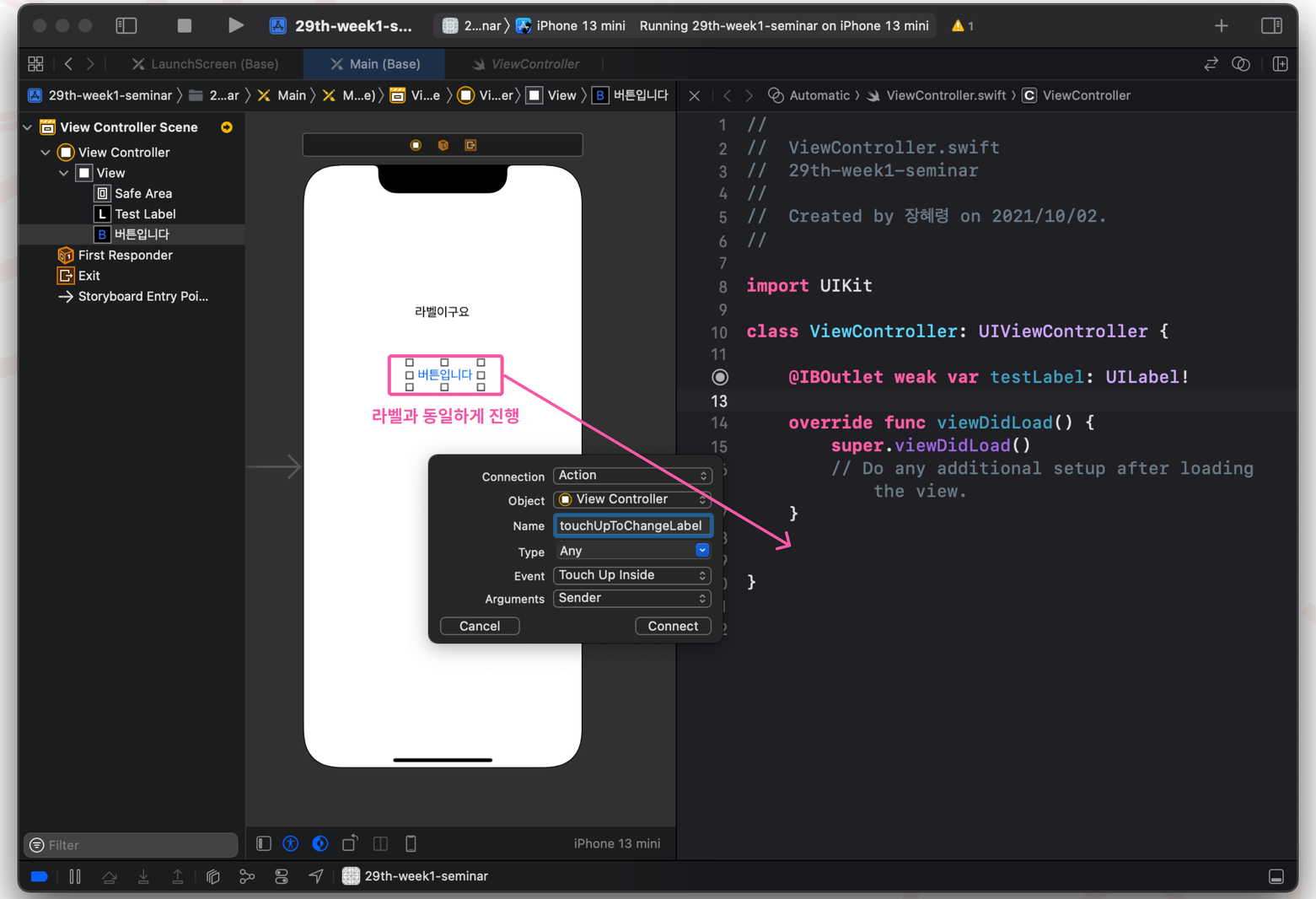
- 유저의 특정 이벤트를 감지해서 Controller에 알리는 역할
- 특정 이벤트 발생 시 실행 될 **동작**들을 정의

* IB 는 Interface Builder의 약자

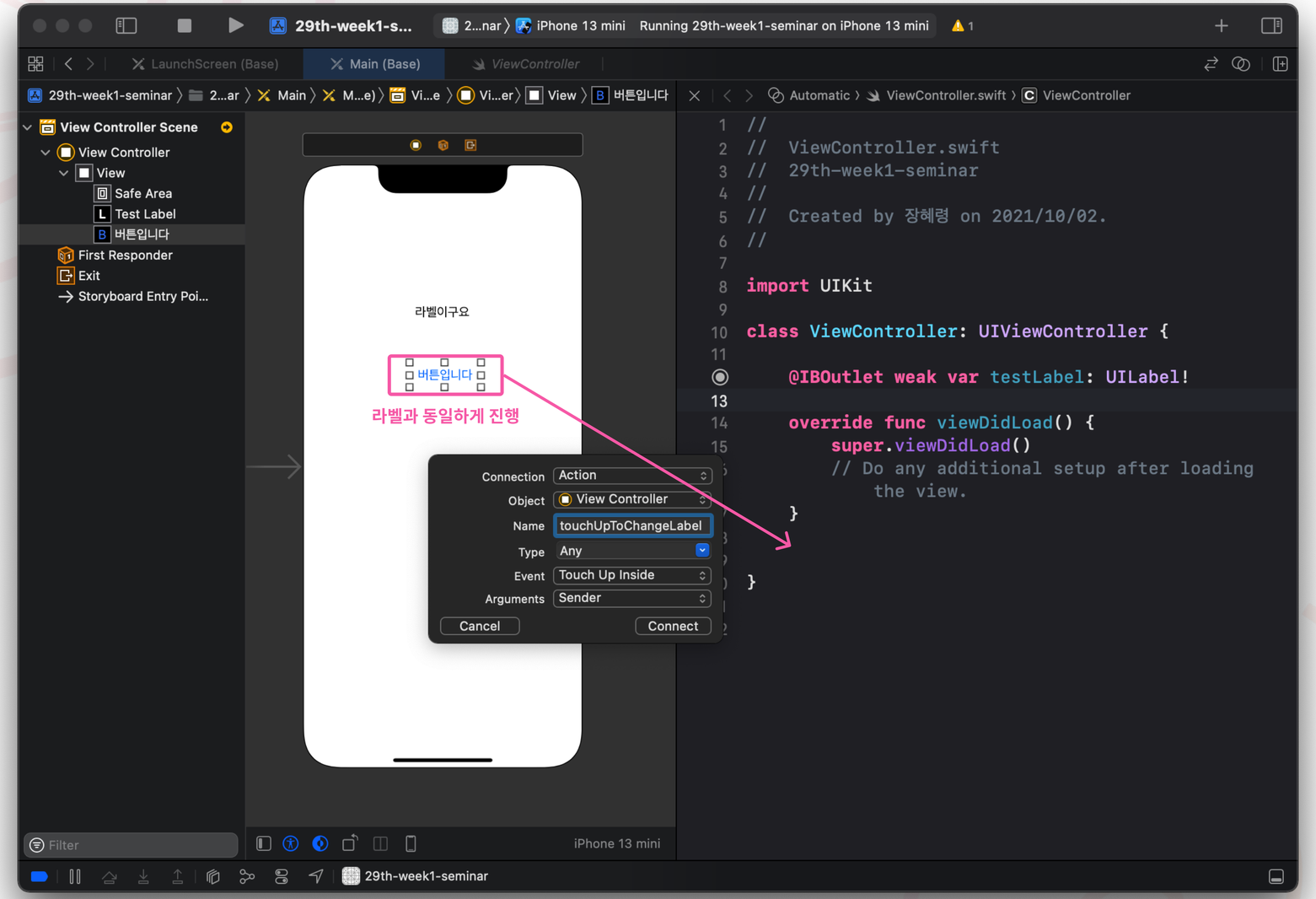
UI 구성하기 | 실습


29th-week1-s...
iPhone 13 mini
Running 29th-week1-seminar on iPhone 13 mini
LaunchScreen (Base)
Main (Base)
ViewController
29th-week1-seminar 〉 2...ar 〉 Main 〉 M...e) 〉 Vi...e 〉 Vi...er 〉 View 〉 라벨이구요
Automatic 〉 ViewController.swift 〉 No Selection
View Controller Scene
View Controller
View
Safe Area
라벨이구요
버튼입니다
First Responder
Exit
Storyboard Entry Poi...
Connection Outlet
Object View Controller
3. 변수명 적고 Name testLabel
Type UILabel
Storage Weak
Cancel
Connect
4. 눌러주세요
1. 라벨을 클릭한 채로
라벨이구요
2. control(^) + drag & drop
버튼입니다
iPhone 13 mini
Filter
29th-week1-seminar
//
// ViewController.swift
// 29th-week1-seminar
//
// Created by 장혜령 on 2021/10/02.
//
import UIKit
class ViewController: UIViewController {
override func viewDidLoad() {
super.viewDidLoad()
// Do any additional setup after loading the view.
}
}

UI 구성하기 | 실습


29th-week1-s...
iPhone 13 mini
Running 29th-week1-seminar on iPhone 13 mini
LaunchScreen (Base)
Main (Base)
ViewController
29th-week1-seminar
Main
View
버튼입니다
Automatic
ViewController.swift
ViewController
View Controller Scene
View Controller
View
Safe Area
Test Label
버튼입니다
First Responder
Exit
Storyboard Entry Poi...
라벨이구요
버튼입니다
라벨과 동일하게 진행
Connection
Action
Object
View Controller
Name
touchUpToChangeLabel
Type
Any
Event
Touch Up Inside
Arguments
Sender
Cancel
Connect
Filter
iPhone 13 mini
29th-week1-seminar
//
// ViewController.swift
// 29th-week1-seminar
//
// Created by 장혜령 on 2021/10/02.
//
import UIKit
class ViewController: UIViewController {
@IBOutlet weak var testLabel: UILabel!
override func viewDidLoad() {
super.viewDidLoad()
// Do any additional setup after loading the view.
}
}

UI 구성하기 | 실습


29th-week1-s...
iPhone 13 mini
Running 29th-week1-seminar on iPhone 13 mini
LaunchScreen (Base)
Main (Base)
ViewController
29th-week1-seminar
Main
View
버튼입니다
View Controller Scene
View Controller
View
Safe Area
Test Label
버튼입니다
First Responder
Exit
Storyboard Entry Poi...
라벨이구요
버튼입니다
라벨과 동일하게 진행
Connection
Action
Object
View Controller
Name
touchUpToChangeLabel
Type
Any
Event
Touch Up Inside
Arguments
Sender
Cancel
Connect
iPhone 13 mini
Automatic
ViewController.swift
ViewController
//
// ViewController.swift
// 29th-week1-seminar
//
// Created by 장혜령 on 2021/10/02.
//
import UIKit
class ViewController: UIViewController {
@IBOutlet weak var testLabel: UILabel!
override func viewDidLoad() {
super.viewDidLoad()
// Do any additional setup after loading the view.
}
}
29th-week1-seminar

UI 구성하기 | 실습

```
8   import UIKit
9
10  class ViewController: UIViewController {
11
13      @IBOutlet weak var testLabel: UILabel!
14
14      override func viewDidLoad() {
15          super.viewDidLoad()
16          // Do any additional setup after loading the view.
17      }
18
20      @IBAction func touchUpToChangeLabel(_ sender: Any) {
20      }
21
22  }
```

Storyboard에 넣었던 Label과 연결

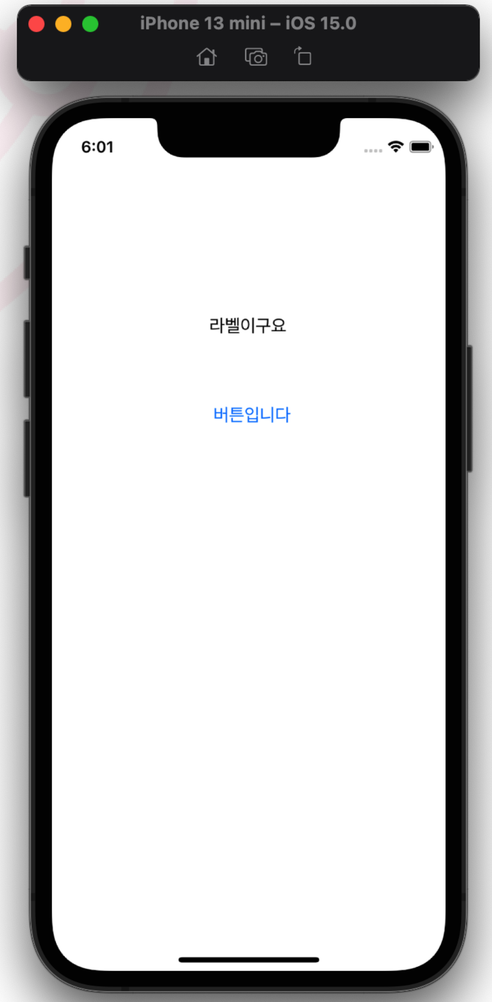
Storyboard에 넣었던 Button의 touchUpInside라는 이벤트 감지

* touchUpInside : 터치한 컴포넌트에서 손을 뗐을 때 이벤트 발생

```
@IBAction func touchUpToChangeLabel(_ sender: Any) {
    testLabel.text = "WE SOPT 1차 세미나"
    testLabel.sizeToFit() // 라벨 크기를 글자에 맞게 변경해주는 메소드
}
```

- 버튼에 touchUpInside 이벤트가 발생하면
- 버튼과 연결한 touchUpToChangeLabel(_) 함수가 불림
- 이 함수 내부에서 실행 될 동작을 코드로 정의

UI 구성하기 | 실습


iPhone 13 mini – iOS 15.0
6:01
라벨이구요
버튼입니다

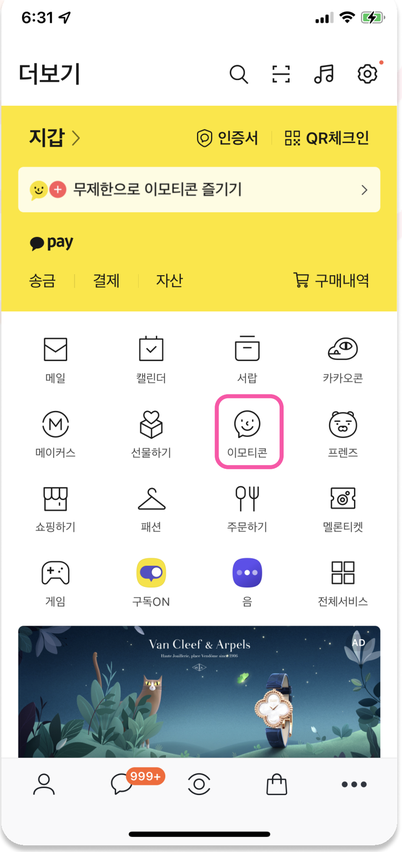
iPhone 13 mini – iOS 15.0
6:02
WE SOPT 1차 세미나
버튼입니다

04

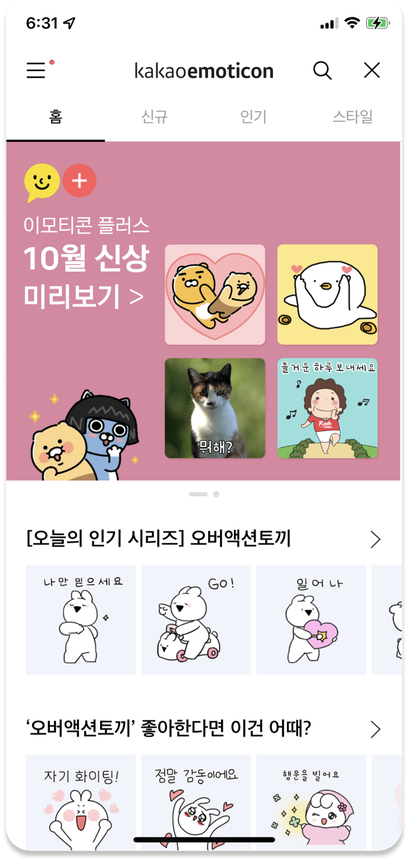
화면 전환

화면 전환

수많은 APP들을 사용하면서 다양한 화면을 보게되는데
현재 보던 화면에서 다른 화면으로 이동하기 위해서는 **화면 전환**이 필요합니다!



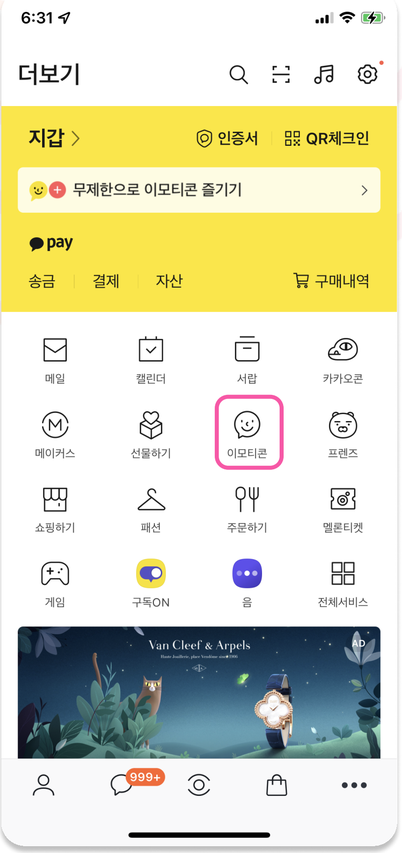
위로 쇽 올라오면서 하단에 있던
친구, 채팅 아이콘들이 사라지고
새로운 화면이 나왔습니다



이번에는 오른쪽으로 쇽 화면이 생기고
또 다른 화면이 나왔습니다

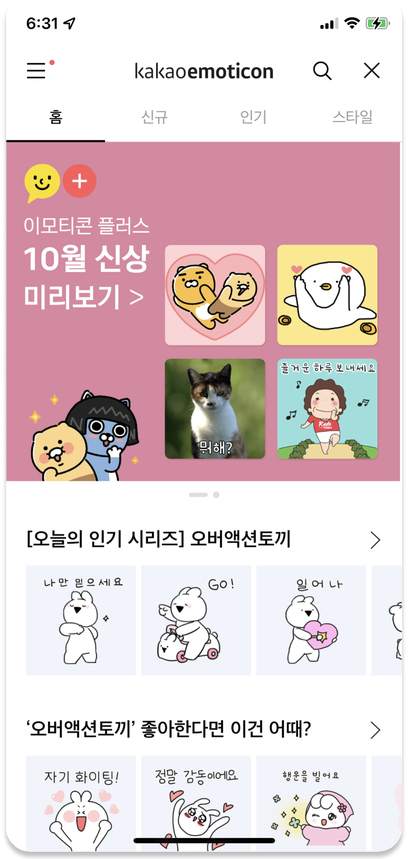
화면 전환

수많은 APP들을 사용하면서 다양한 화면을 보게되는데
현재 보던 화면에서 다른 화면으로 이동하기 위해서는 **화면 전환**이 필요합니다!



모달(Modal)

위로 쇽 올라오면서 하단에 있던
친구, 채팅 아이콘들이 사라지고
새로운 화면이 나왔습니다



네비게이션 (Navigation)

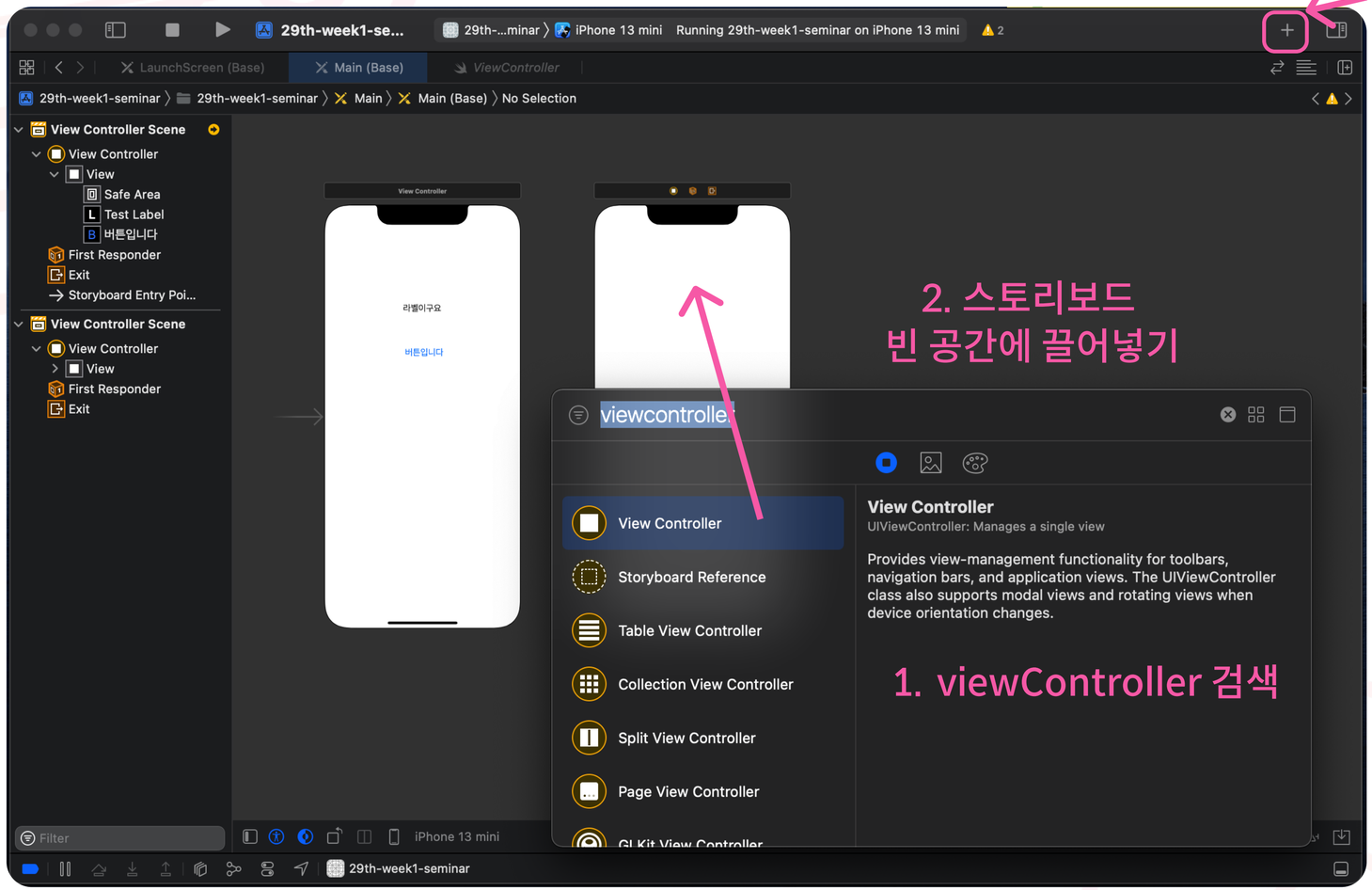
이번에는 오른쪽으로 쇽 화면이 생기고
또 다른 화면이 나왔습니다

📍모달(Modal)

- 사용자의 이목을 끌기 위해 사용하는 화면 전환 기법
- 화면을 전환한다기 보다는 이목을 집중해야 하는 화면을 다른 화면 위로 띄워(Presenting) 표현 하는 방식
- 흐름을 이어지는 컨텐츠를 담기보다는 흐름이 끊어져서 딱 눈에 들어와야 하는 컨텐츠를 담는데 사용

화면 전환 | 모달(Modal)

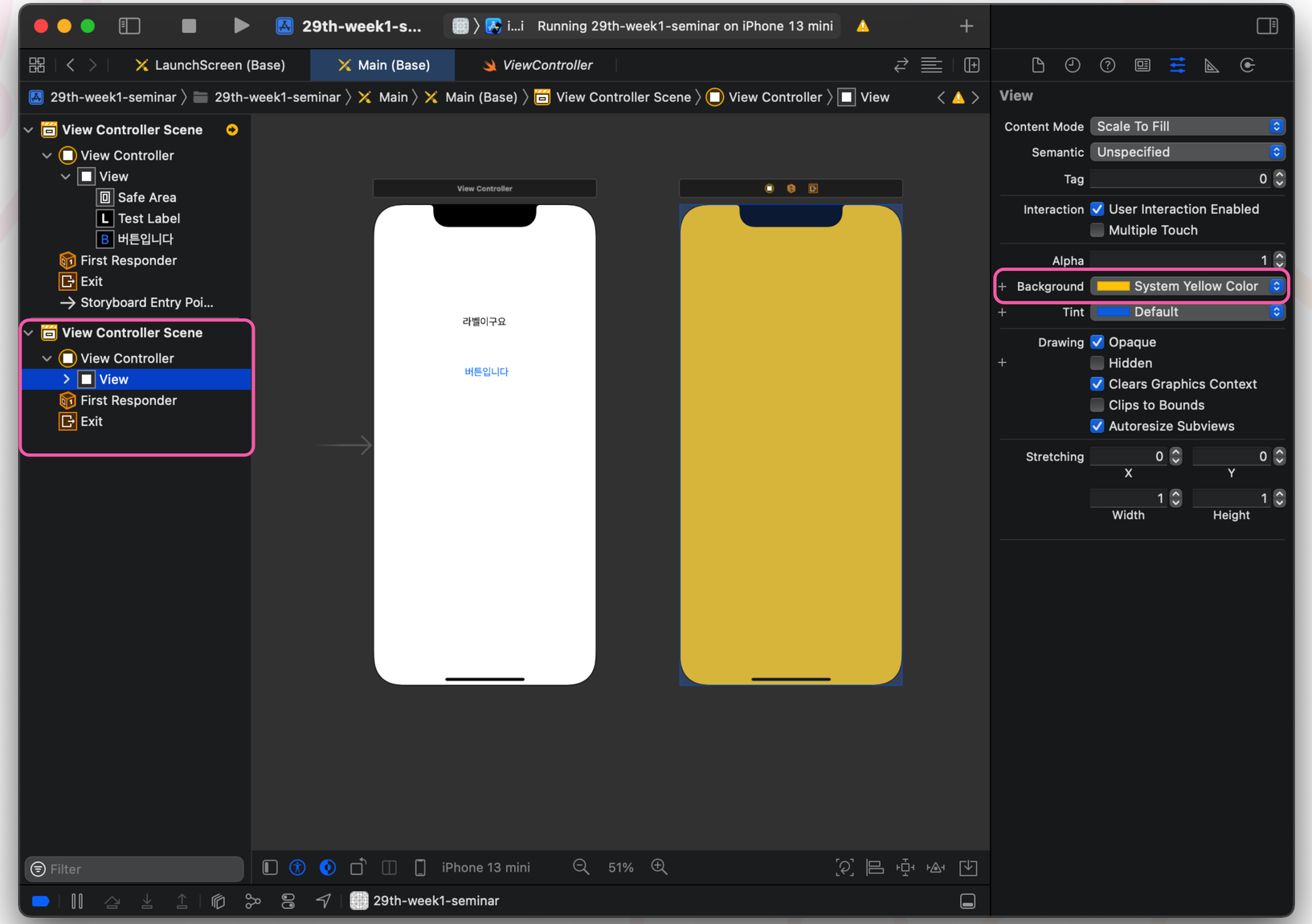
0. 클릭 또는
shift (⇧) + command(⌘) + L



화면 전환 | 모달(Modal)

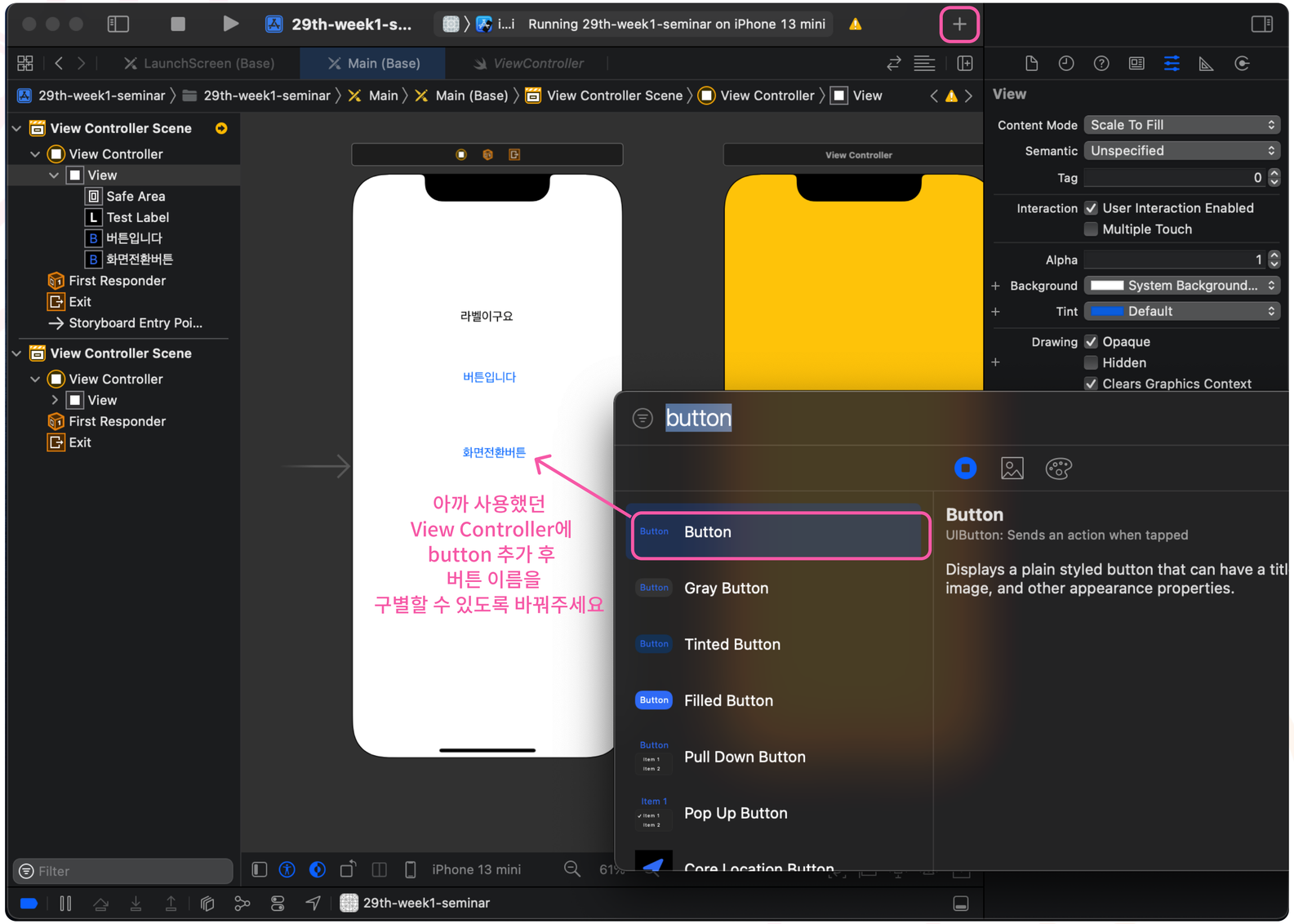
화면전환이 눈에 잘 띄도록

방금 만든 Veiw controller에서
View를 선택한 후
우측 Inspactor 메뉴에서
Background 속성을 변경해주세요
(색상은 자유롭게)

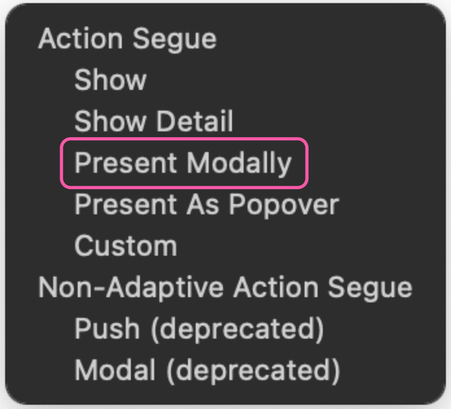


화면 전환 | 모달(Modal)

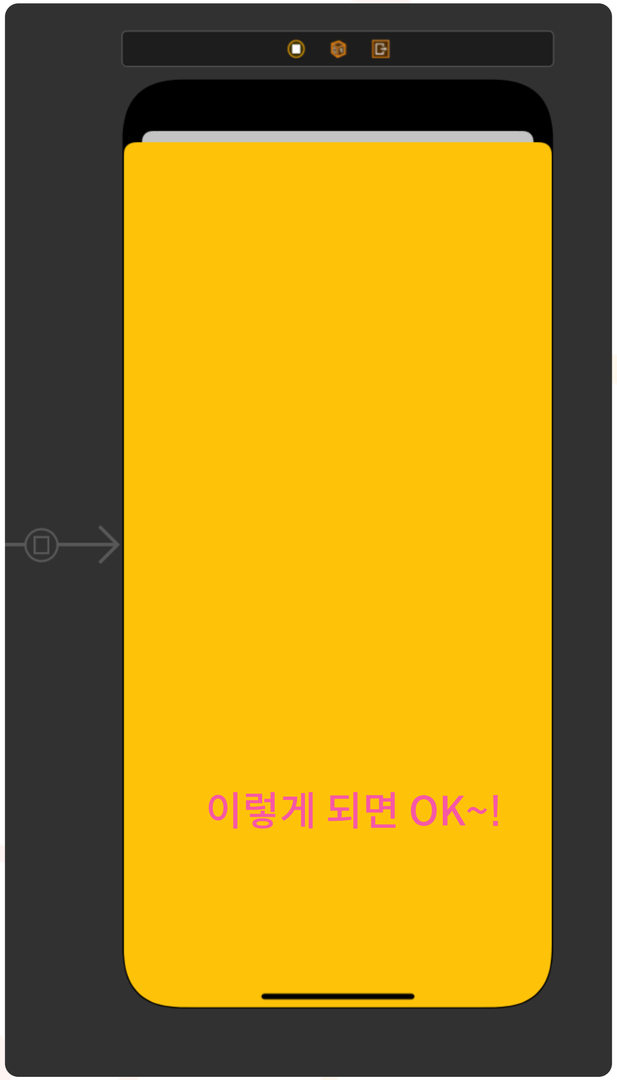
클릭 또는
shift (⇧) + command(⌘) + L



화면 전환 | 모달(Modal)



3. Present Modally 클릭



화면 전환 | 모달(Modal)

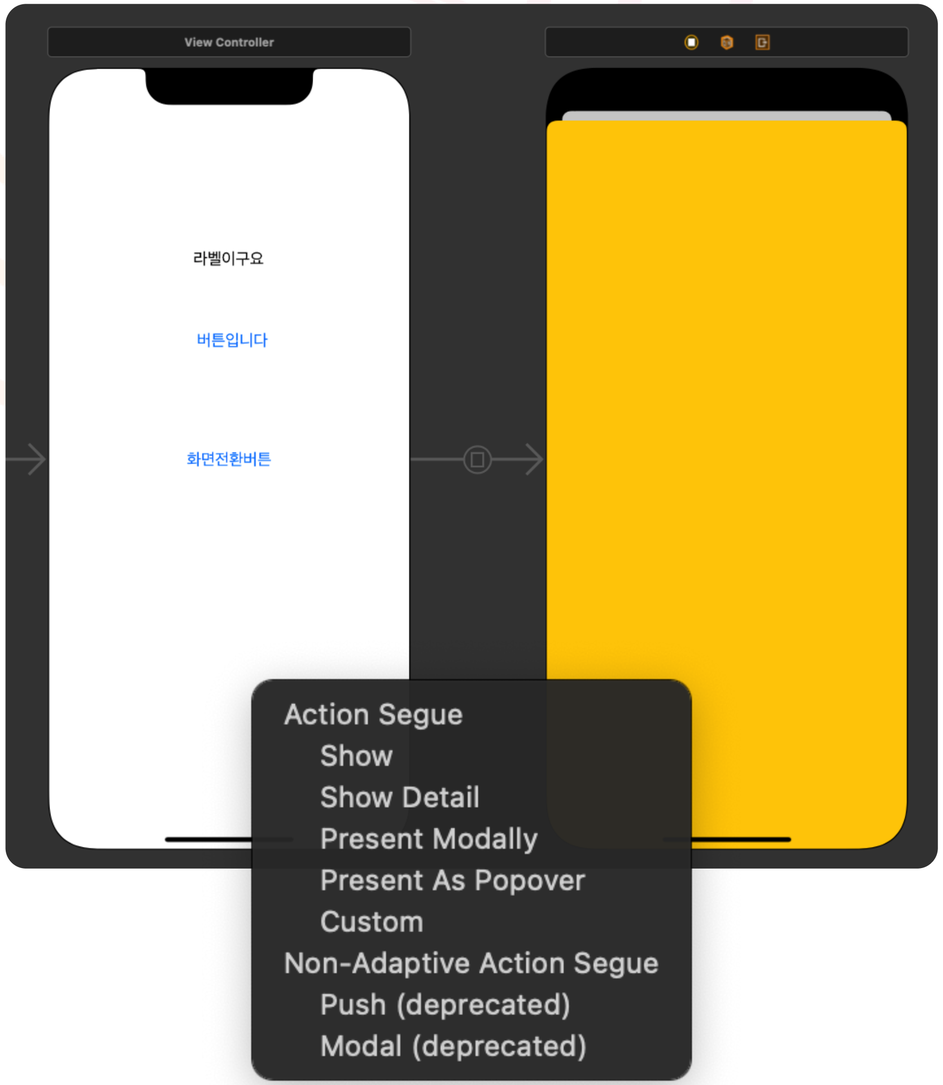
빌드해서 확인해보세요! (command + R)

화면 전환 | 모달(Modal)

세그웨이(Segue)로 모달을 구현한 방식

* 세그웨이(Segue)
iOS에서 화면 전환 방식에 사용되는 방법
스토리보드를 통해 출발지와 목적지를 직접 지정하는 방식
을 세그웨이(segue)을 이용한 화면 전환이라고 하고,
두 개의 ViewController 사이 연결된
화면 전환 객체를 세그웨이라고 합니다.

화면 전환 | 모달(Modal)



세그웨이(Segue)로 모달을 구현한 방식

* 세그웨이(Segue)
iOS에서 화면 전환 방식에 사용되는 방법
스토리보드를 통해 출발지와 목적지를 직접 지정하는 방식을
세그웨이(segue)을 이용한 화면 전환이라고 하고,
두 개의 ViewController 사이 연결된 화면 전환 객체를 세그웨이라고 함.

1. Show
Navigation Controller가 연결되어 있다면 Push 처리
아닌 경우 Present Modally 처리

2. Show Detail
아이패드에서 사용하는 방식
Push가 아닌 Replace 방식을 사용

3. Present Modally
Present 방식 전환
Modal Style, Transition Style을 변경해서 다양한 방식이 가능

4. Popover Presentation
아이패드에서 사용되는 전환 방식
작은 Popup 형태의 뷰를 띄움.

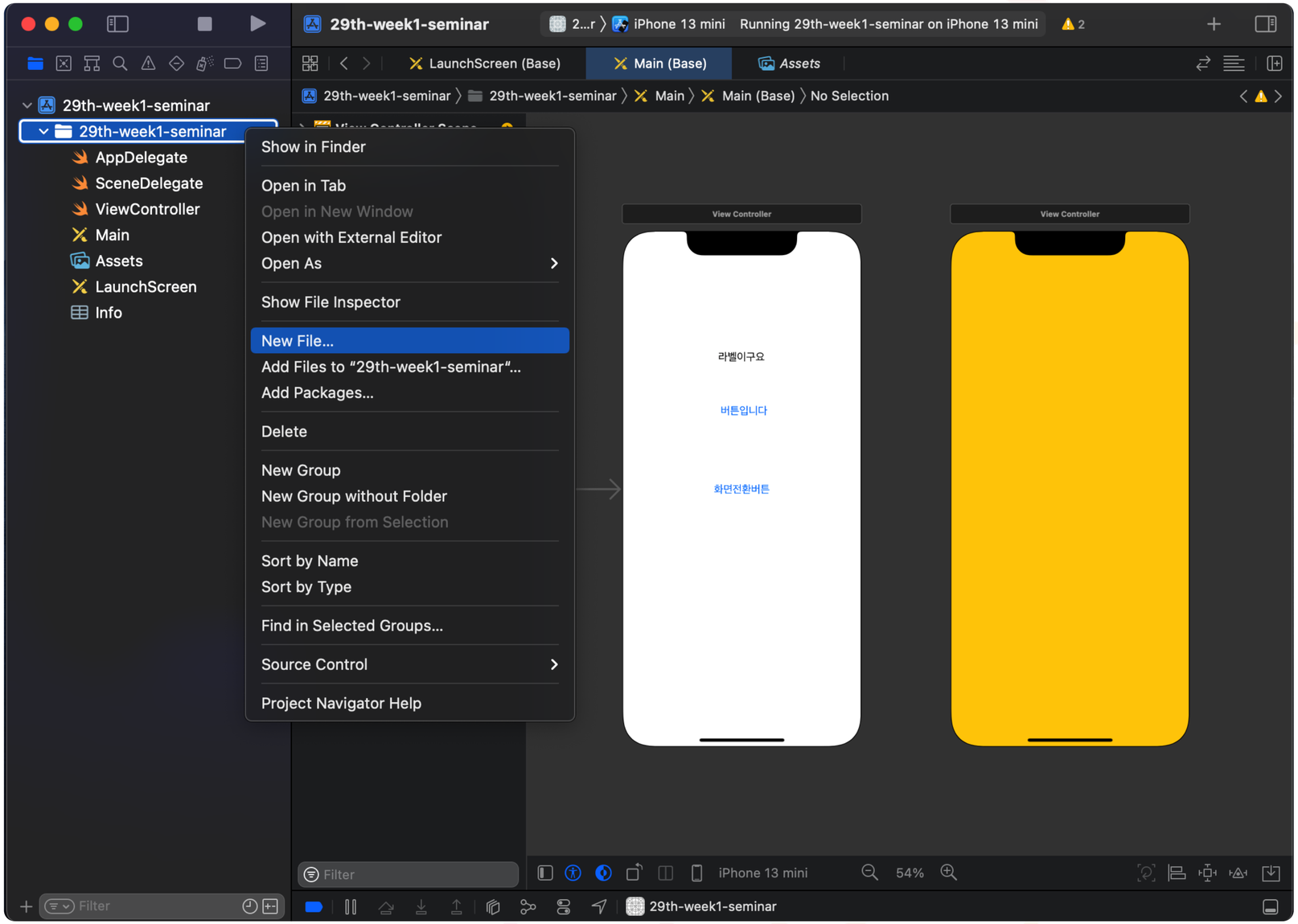
화면 전환 | 모달(Modal)

코드로 모달 구현하기

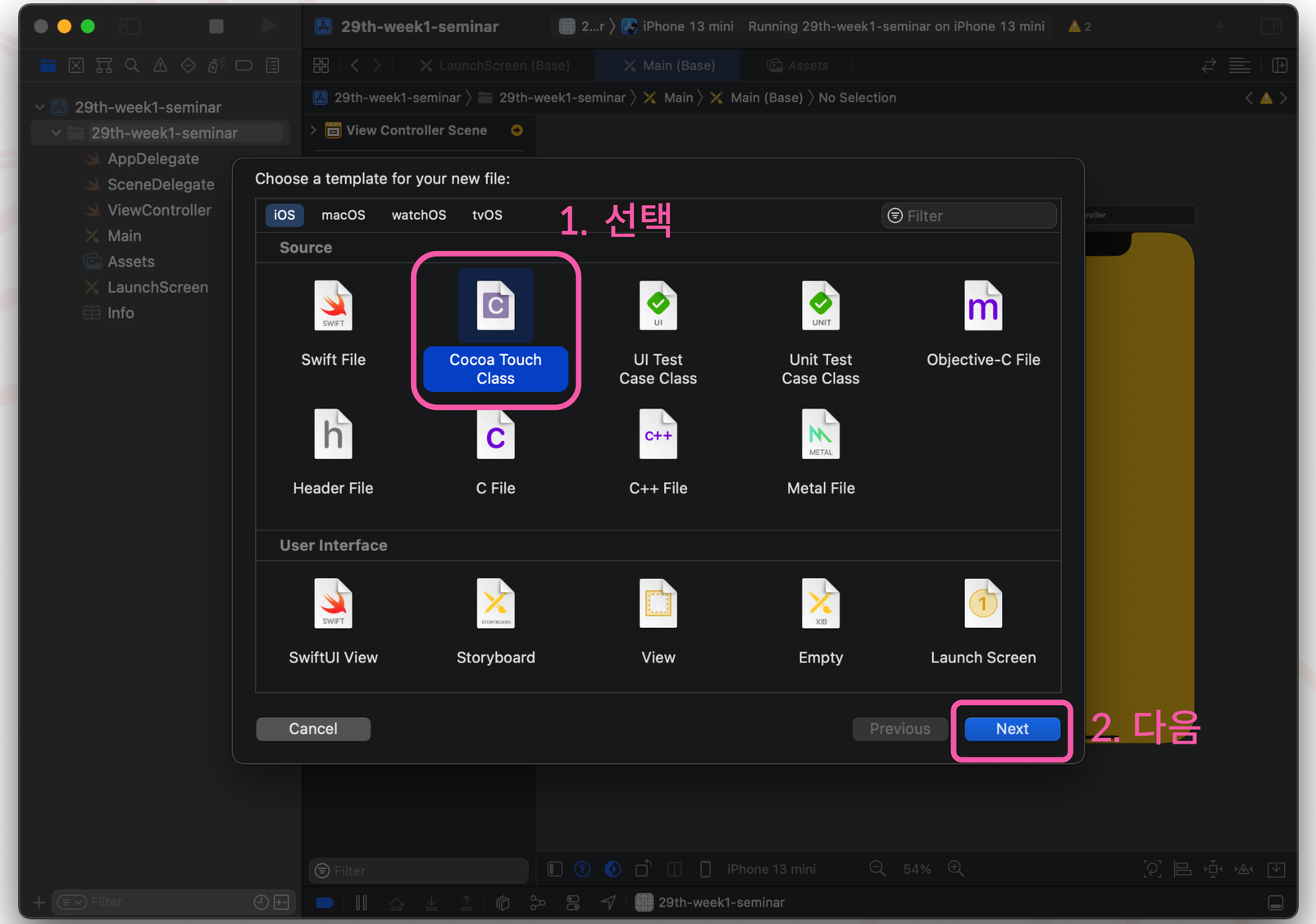
하기 전에는 먼저 세그를 삭제해주세요!
코드방식과 세그 방식을
동시에 사용하면 에러가 발생합니다

화면 전환 | 모달(Modal)

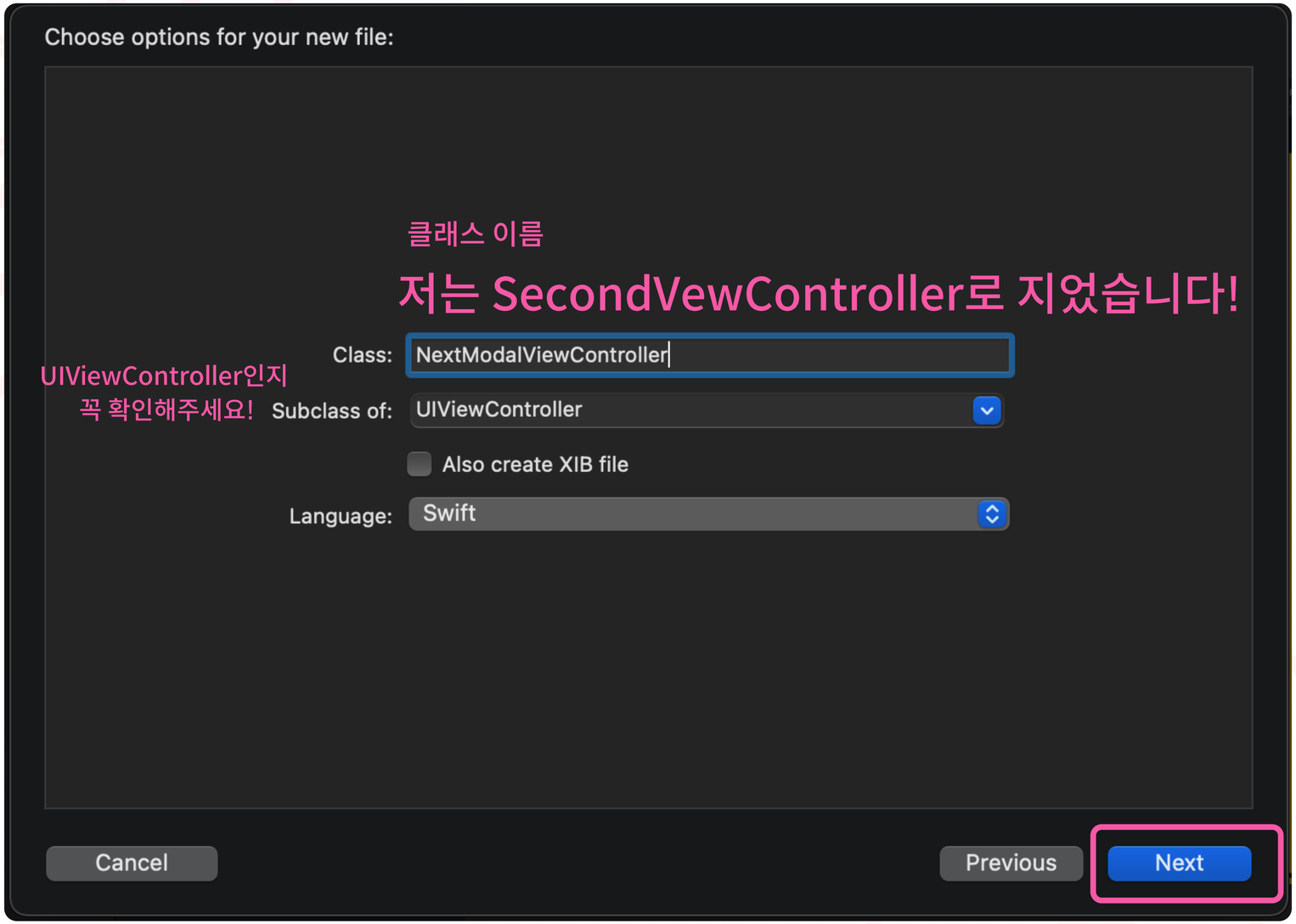
프로젝트 폴더를 선택하여
마우스 오른쪽 버튼을 누르거나,
command + N을 눌러
새로운 파일을 생성해주세요



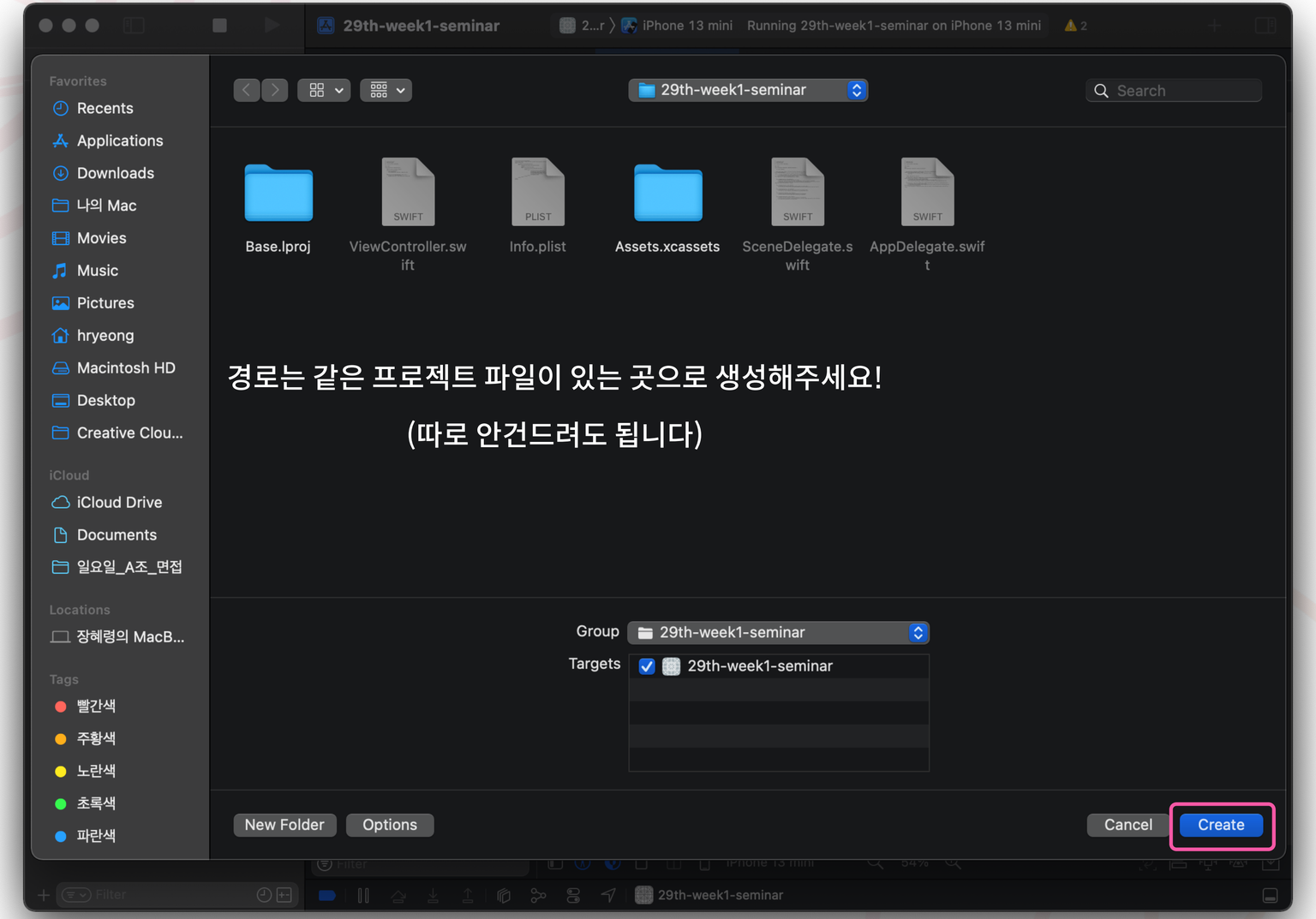
화면 전환 | 모달(Modal)



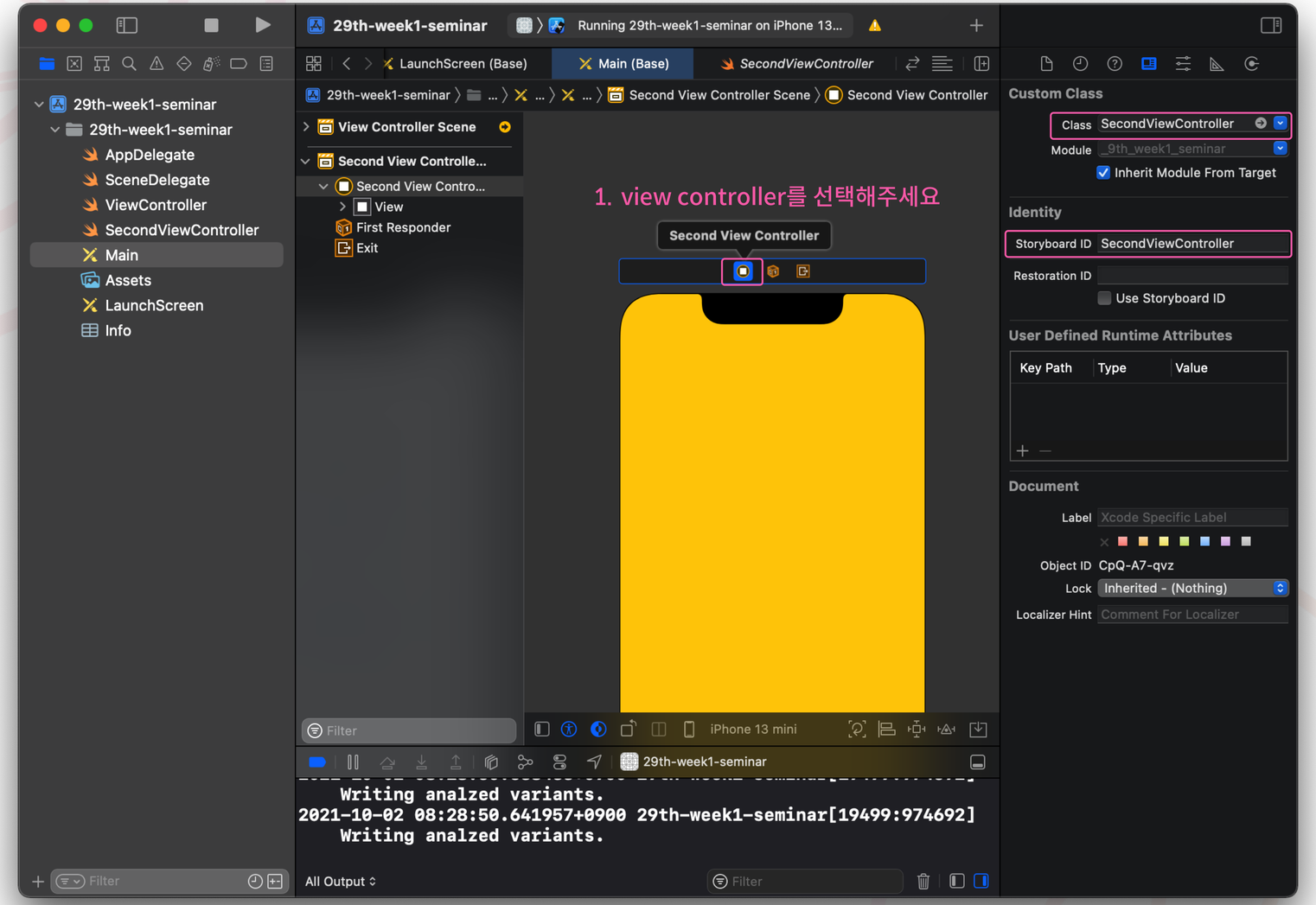
화면 전환 | 모달(Modal)



화면 전환 | 모달(Modal)



화면 전환 | 모달(Modal)



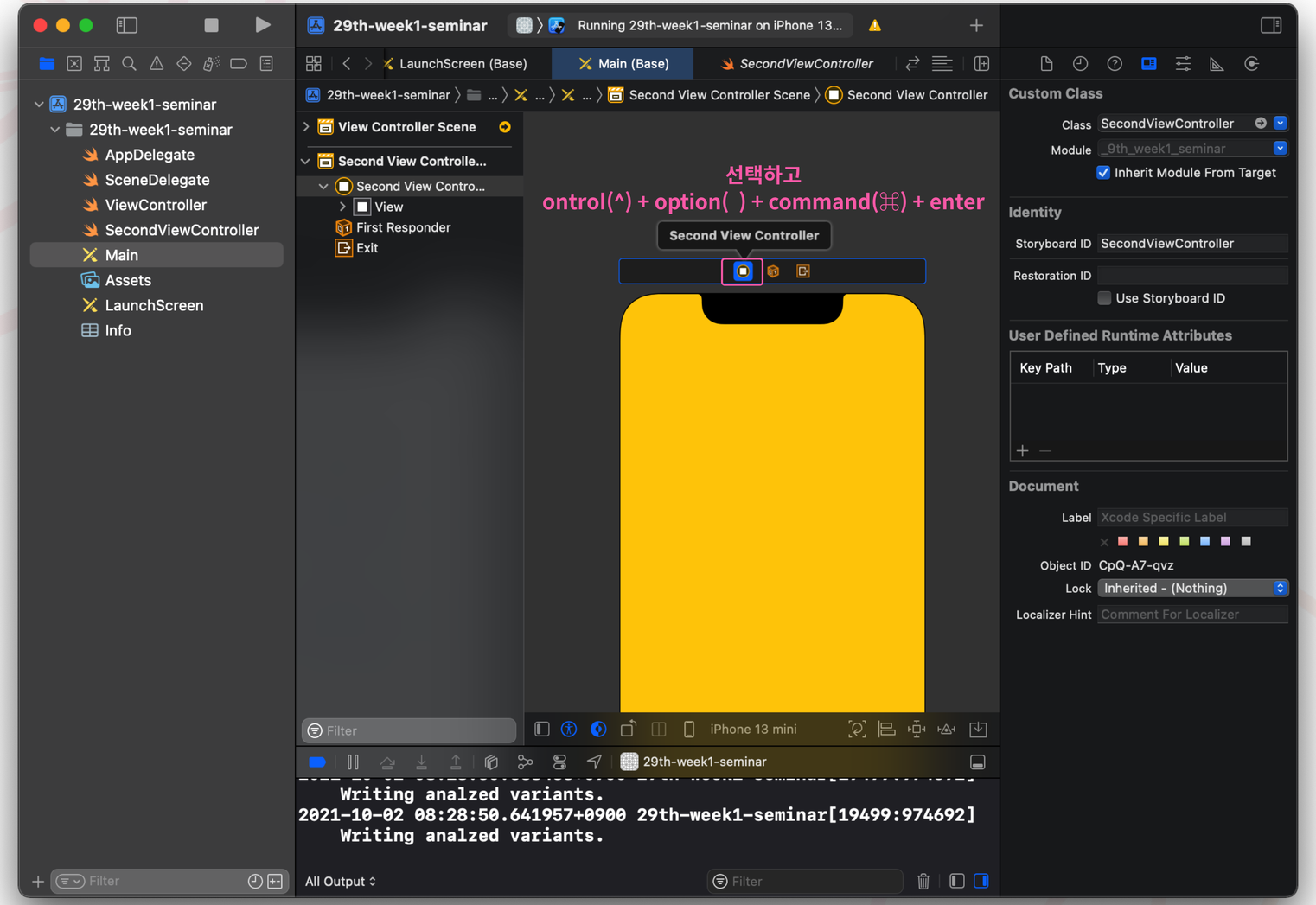
2. Class 이름을 아까 생성해둔 파일이름으로 지정해주세요!

3. identifier를 아까 생성해둔 파일 이름과 동일하게 넣어주세요

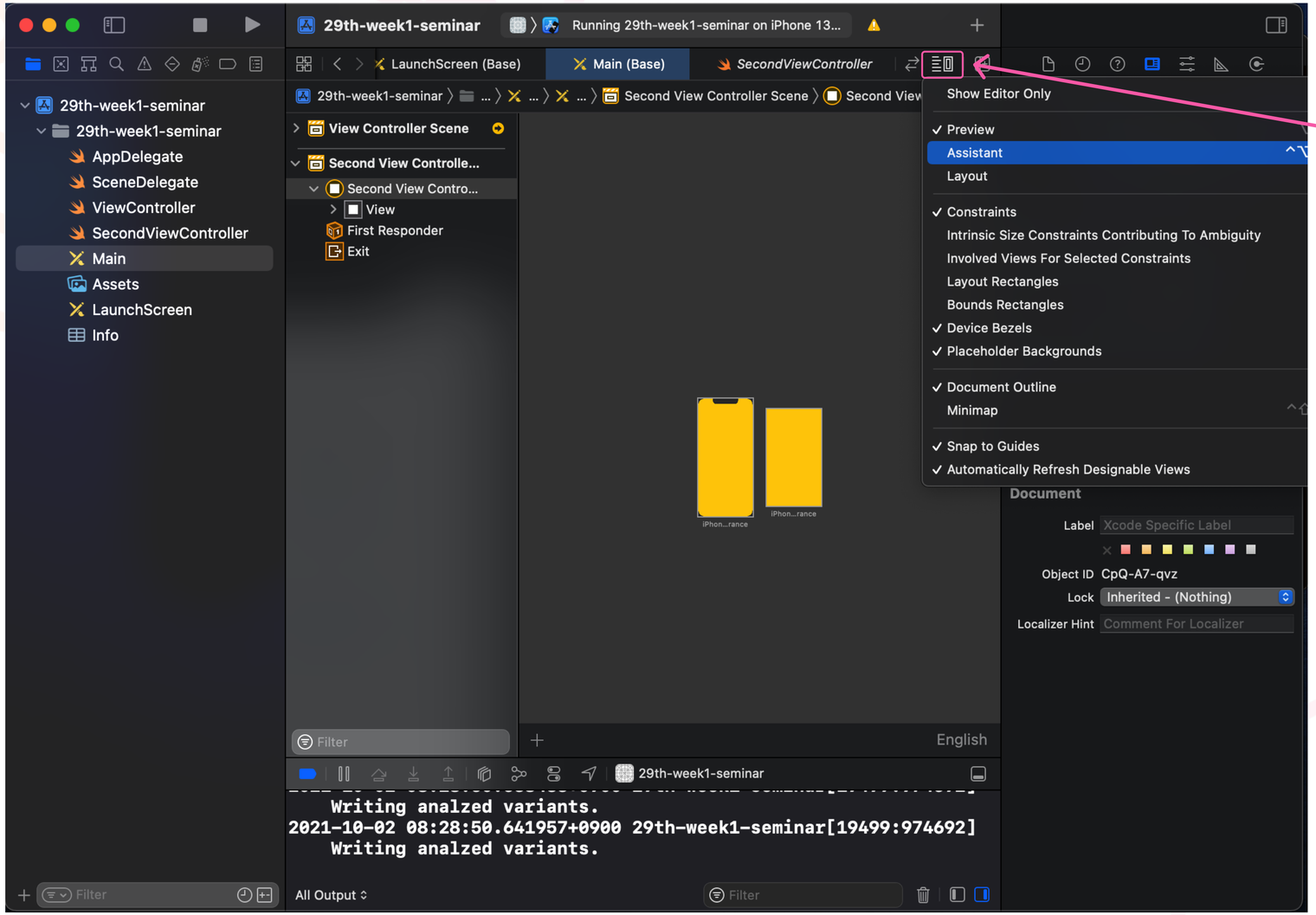
storyBoard ID는 해당 뷰컨의 identifier로 사용됩니다.

클래스 명하고 여기하고 동일하게 맞추는 편입니다 :)
다르게 사용해도 상관 없습니다!!

화면 전환 | 모달(Modal)



화면 전환 | 모달(Modal)



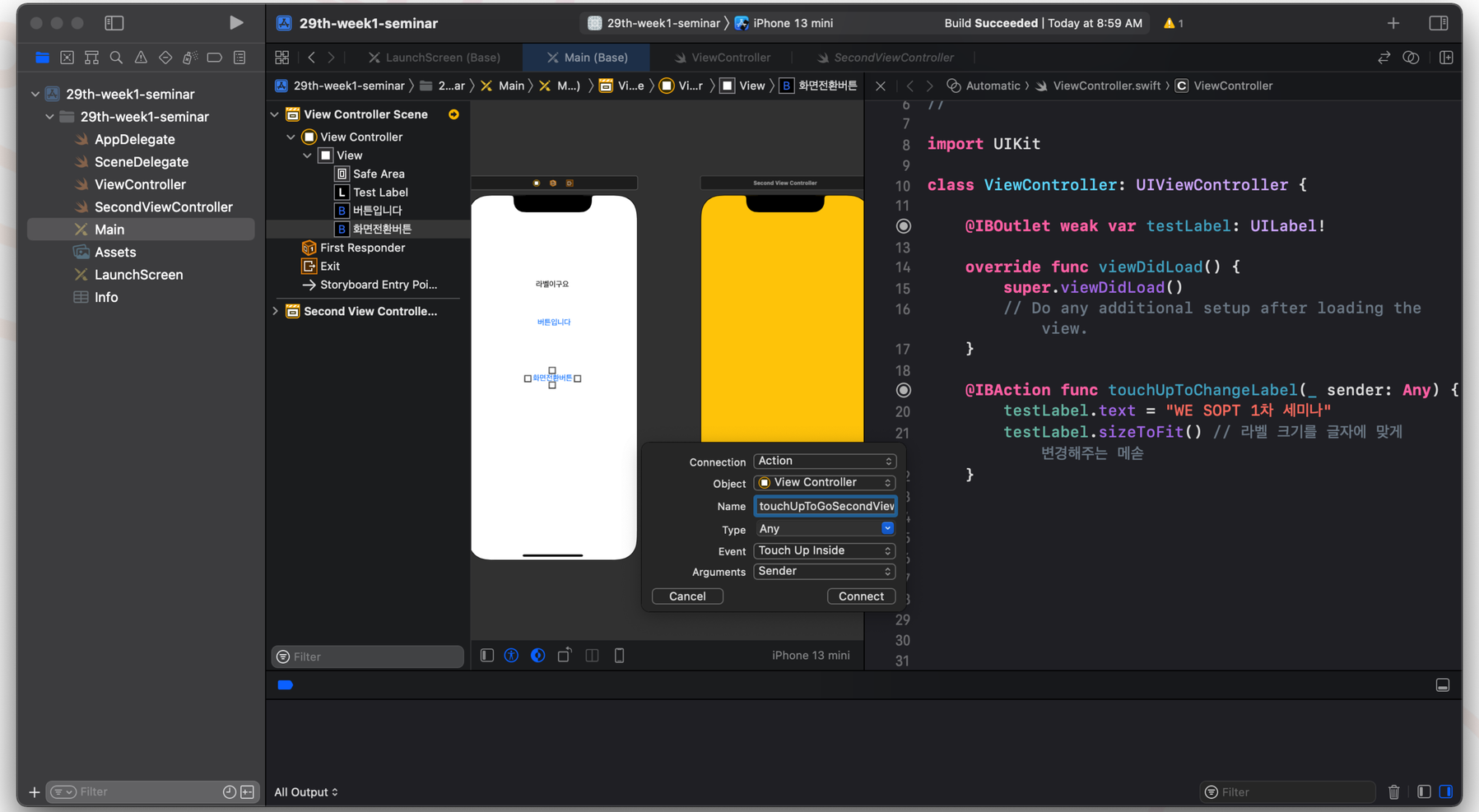
바로 어시스턴스 에디터가 열리지 않을 때!

어시스턴스 데이터가 아닌 다른 화면 나온다면
눌러서 assistant를 선택합니다

파일 연결이 잘 되었는지 확인한다.
SecondViewController.swift 에 연결되었는지 확인합니다.

그래도 잘 안된다면?
한 번 껐다가 켜주세요...😥

화면 전환 | 모달(Modal)



화면 전환버튼을 끌어다
Action을 만들어주세요!

화면 전환 | 모달(Modal)

```
@IBAction func touchUpToGoSecondView(_ sender: Any) {
    guard let nextVC = self.storyboard?.instantiateViewController(withIdentifier: "SecondViewController")
        else {return}

    self.present(nextVC, animated: true, completion: nil)
}
```

```
guard (조건) else { 조건이 false 일 때 실행되는 부분}
// else 안에는 break나 return과 같이 종료 구문이 있어야 함
```

optional인 storyboard에서
SecondViewController라는 identifier를 가진
View controller instance를 생성해서
nextVC 상수에 담아줘

이를 실행하다가 무엇하나라도 잘못되면 else문에 걸린다면 return 되어
해당 touchUpToSecondView함수가 종료됩니다.

guard 구문을 빠져나온 nextVC 는 확실한 instance

화면 전환 | 모달(Modal)

```
@IBAction func touchUpToGoSecondView(_ sender: Any) {
    guard let nextVC = self.storyboard?.instantiateViewController(withIdentifier: "SecondViewController")
        else {return}

    self.present(nextVC, animated: true, completion: nil)
}
```

모달을 띄우는 present 메서드를 통해 nextVC를 띄웁니다.

화면 전환 | 모달(Modal)

빌드해서 확인해보세요! (command + R)

화면 전환 | 모달(Modal)

```
@IBAction func touchUpToGoSecondView(_ sender: Any) {
    guard let nextVC = self.storyboard?.instantiateViewController(withIdentifier: "SecondViewController")
        else {return}

    nextVC.modalPresentationStyle = .pageSheet
    nextVC.modalTransitionStyle = .crossDissolve
    self.present(nextVC, animated: true, completion: nil)
}
```

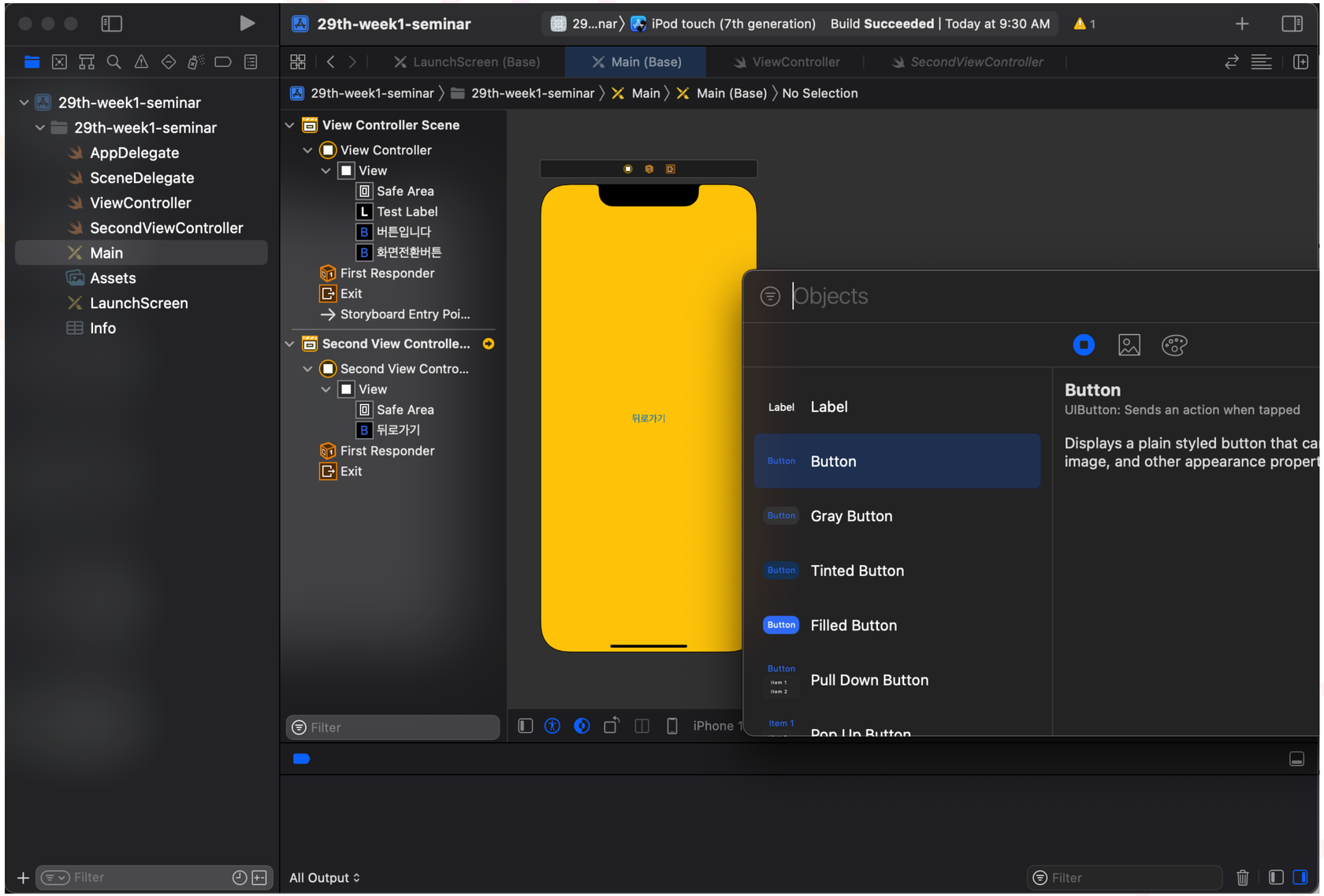
modalPresentationStyle
새로운 모달창을 어떻게 보여줄지

- automatic : 시스템에서 정한 기본 옵션 (현재 하는 방식)
- fullScreen : 위를 남기지 않고 새로운 모달로 창을 덮습니다.
- overCurrentContext : 새로운 모달창이 투명한 경우,
  아래에 깔려있는 이전 뷰도 함께 보임

madalTranstisionStyle
새로운 모달창을 어떻게 전환할지

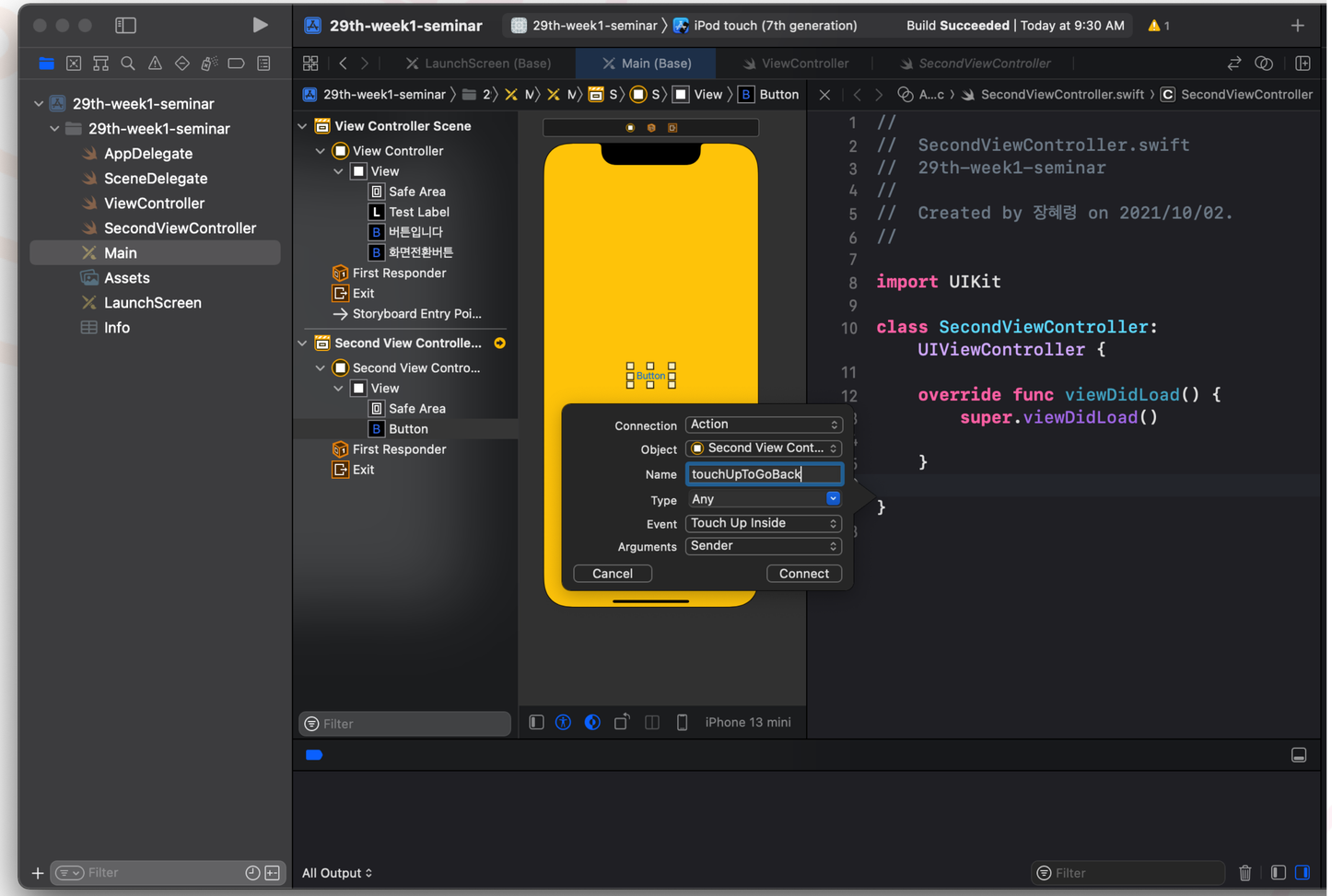
- overVertical : 우리가 처음 본 것 처럼,
  아래에서 나와 뷰를 올리는 형식
- crossDissolve : 뷰가 교차되면서 전환됨.

화면 전환 | 모달(Modal)



SecondViewController에 뒤로가기 버튼을 추가해주세요

화면 전환 | 모달(Modal)



추가한 버튼에
IBAction도 추가해주세요:)

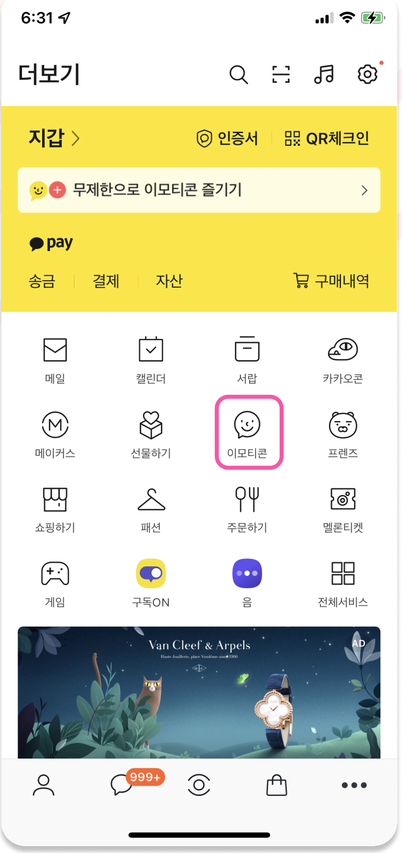
화면 전환 | 모달(Modal)

```
import UIKit

class SecondViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

    }

    @IBAction func touchUpToGoBack(_ sender: Any) {
        self.dismiss(animated: true, completion: nil)
    }
}
```

dismiss 메서드는
현재 present로 띄워진 뷰를
사라지게 하는 메서드

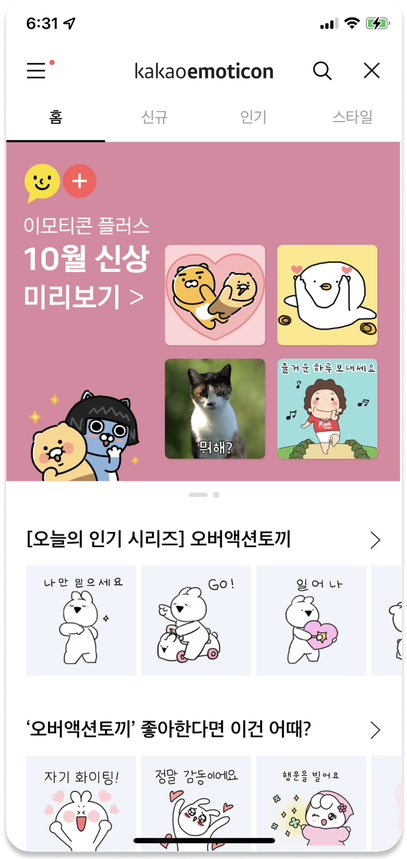
화면 전환

수많은 APP들을 사용하면서 다양한 화면을 보게되는데
현재 보던 화면에서 다른 화면으로 이동하기 위해서는 **화면 전환**이 필요합니다!



모달(Modal)

위로 쇽 올라오면서 하단에 있던
친구, 채팅 아이콘들이 사라지고
새로운 화면이 나왔습니다

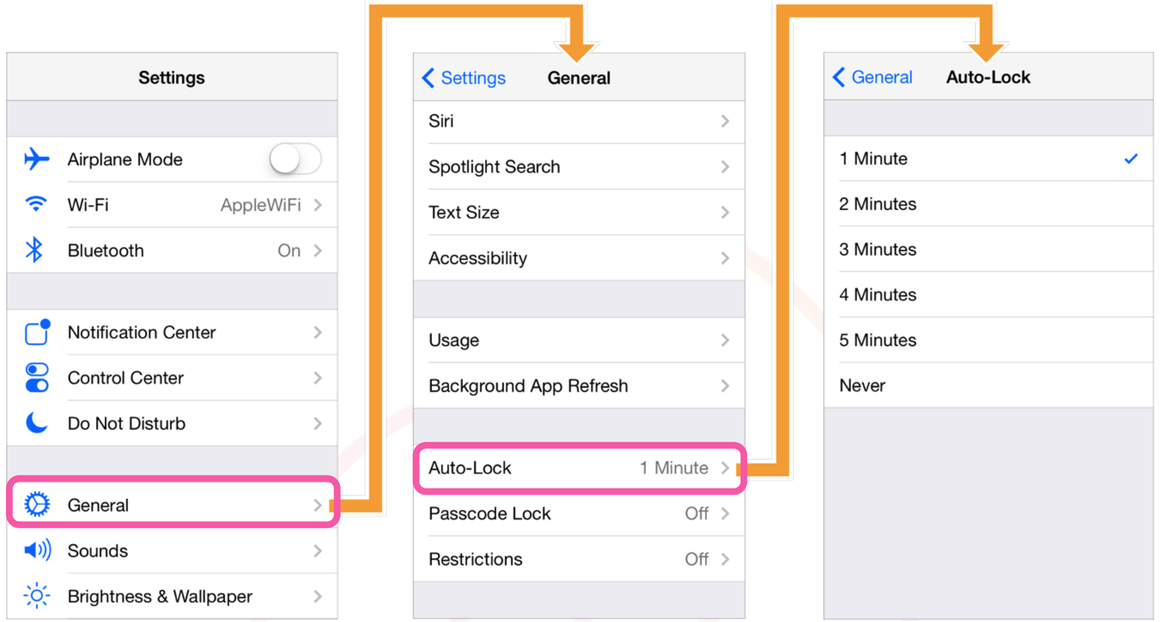


네비게이션 (Navigation)

이번에는 오른쪽으로 쇽 화면이 생기고
또 다른 화면이 나왔습니다

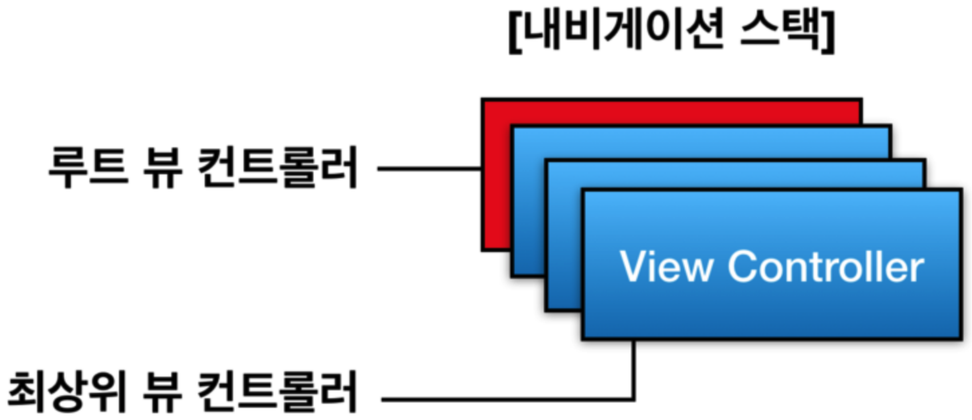
📌 네비게이션 인터페이스

- iOS 화면 전환을 위해 사용 되는 기법 중 가장 대표적인 방법
- 위 이미지와 같이 계층적 구조의 뷰 워크플로우를 구현하기 위해 사용
- UINavigationController에 의해 관리



📌 Navigation Controller 란?

- ViewController 사이 계층 구조를 탐색할 수 있게 해주는 객체
- Present 방식과는 다르게 자식 뷰는 Stack으로 관리
- UIViewController를 담을 수 있는 Container ViewController
- 두 개의 뷰를 화면에 표시
  1. Navigation 스택에 포함된 최상위 ViewController
  2. 직접 관리하는 Navigation bar, Tool bar



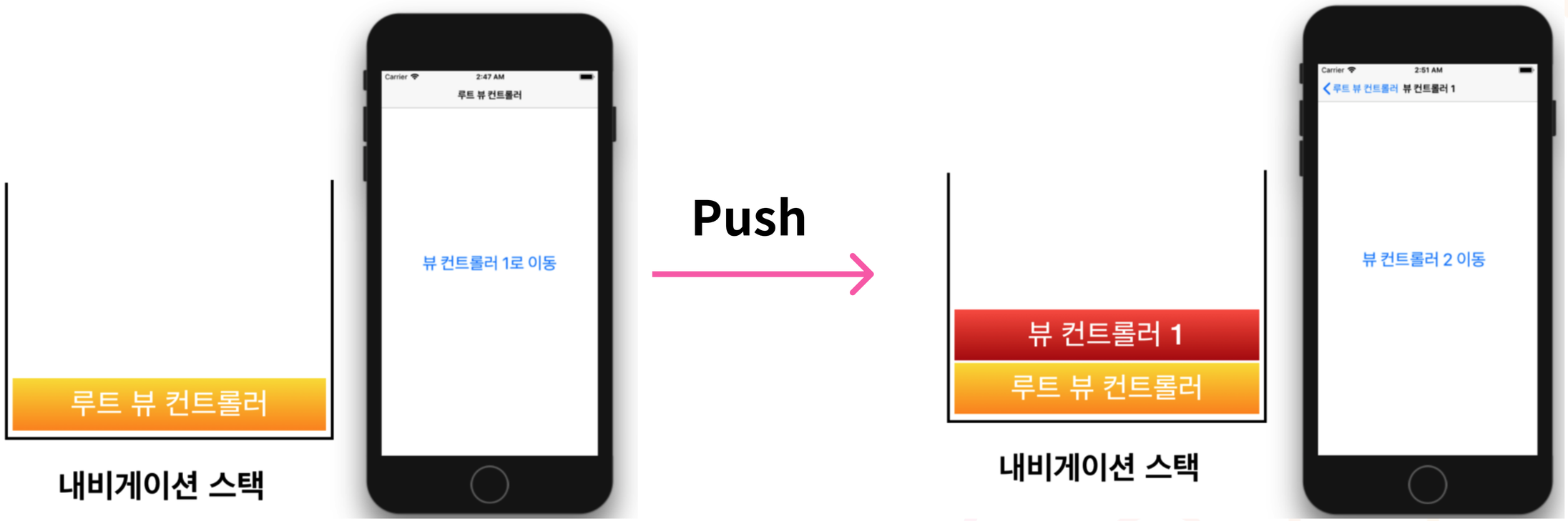
📌 Navigation Controller 란?

📍Navigation Stack

- ViewController를 담을 수 있는 배열
- 가장 먼저 스택에 추가 된 ViewController가 '루트' ViewController
- 가장 늦게 들어간 ViewController 가 '최상위' ViewController가 된다.

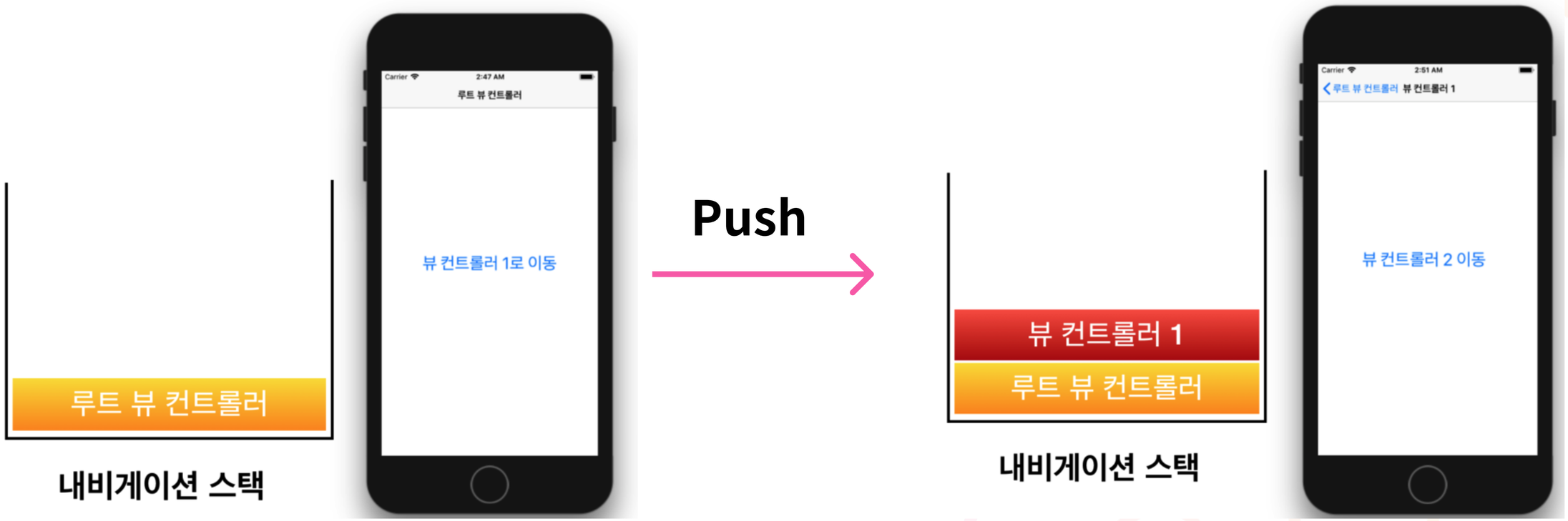
📌 Navigation Controller 란?

📍 Navigation Stack



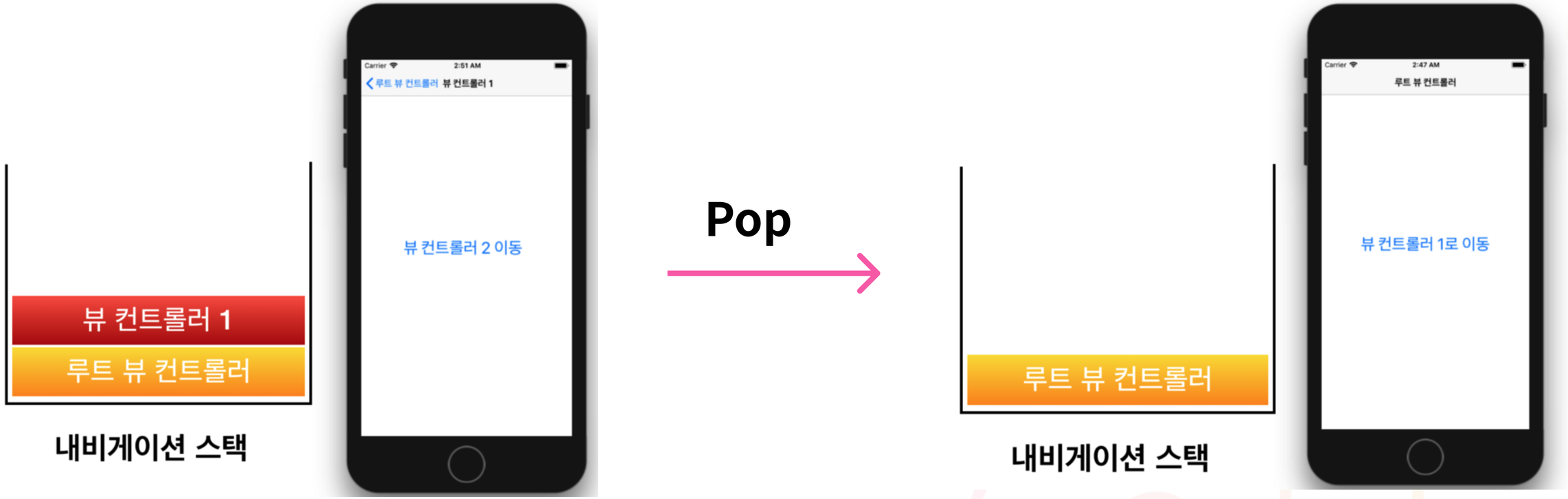
📌 Navigation Controller 란?

📍 Navigation Stack



📌 Navigation Controller 란?

📍 Navigation Stack



📌 Navigation Controller 란?

📍 Navigation Stack

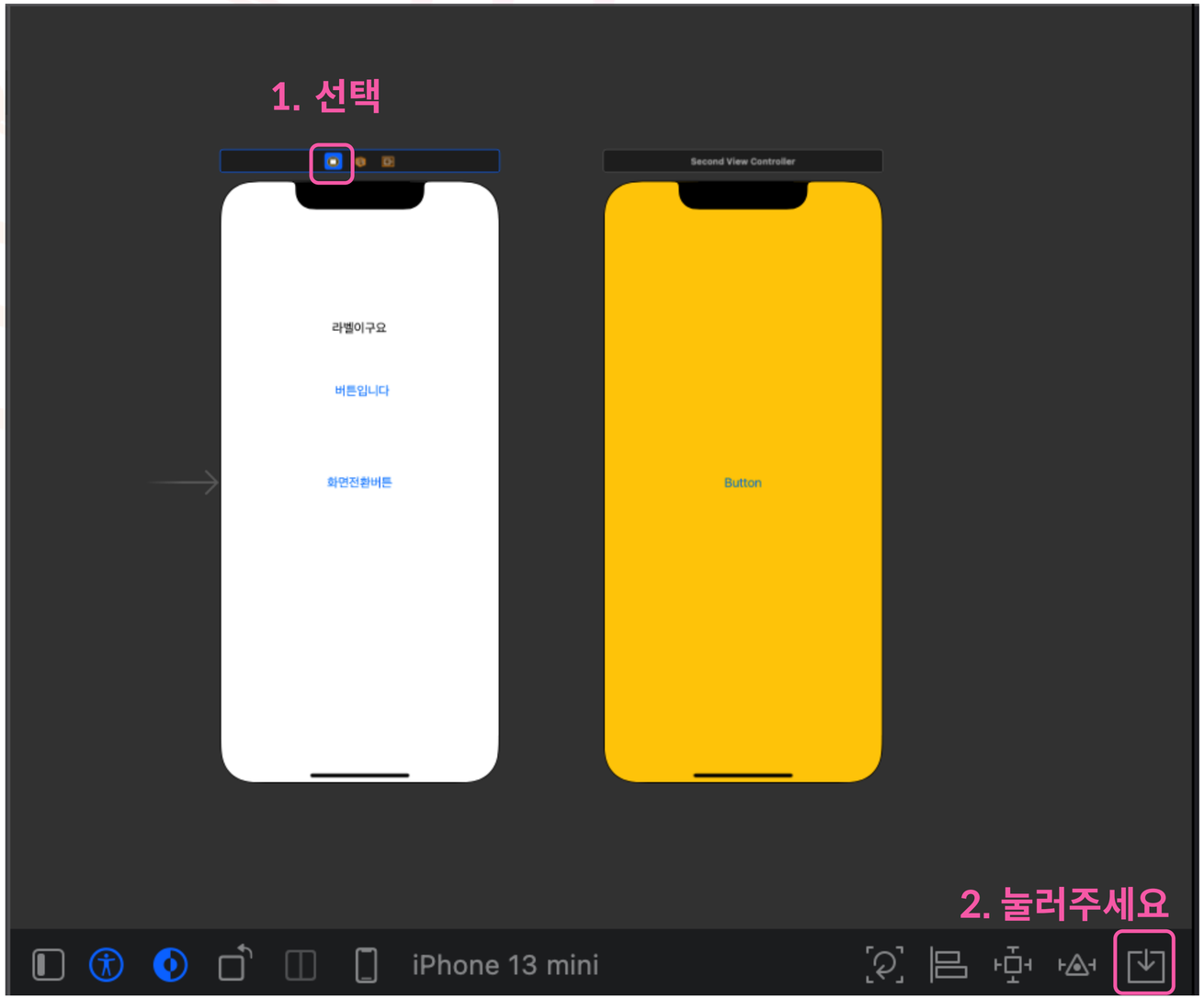
화면 전환 | Navigation

```
class ViewController: UIViewController {

    @IBOutlet weak var testLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func touchUpToChangeLabel(_ sender: Any) {
        testLabel.text = "WE SOPT 1차 세미나"
        testLabel.sizeToFit() // 라벨 크기를 글자에 맞게 변경해주는 메솓
    }


    @IBAction func touchUpToGoSecondView(_ sender: Any) {
        guard let nextVC =
            self.storyboard?.instantiateViewController(withIdentifier:
            "SecondViewController")
            else {return}

//          nextVC.modalPresentationStyle = .fullScreen
//          nextVC.modalTransitionStyle = .crossDissolve
//          self.present(nextVC, animated: true, completion: nil)
    }
}
```

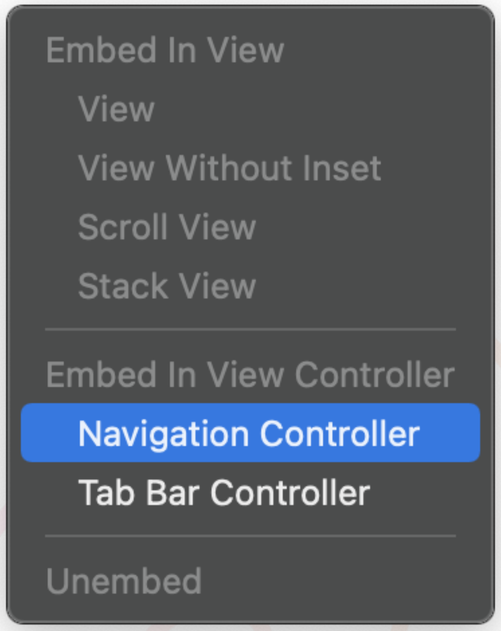
모달로 띄웠던 코드를
먼저 주석처리 해주세요

드래그 한 후 한 번에
command + / 하면
주석처리 됩니다:)

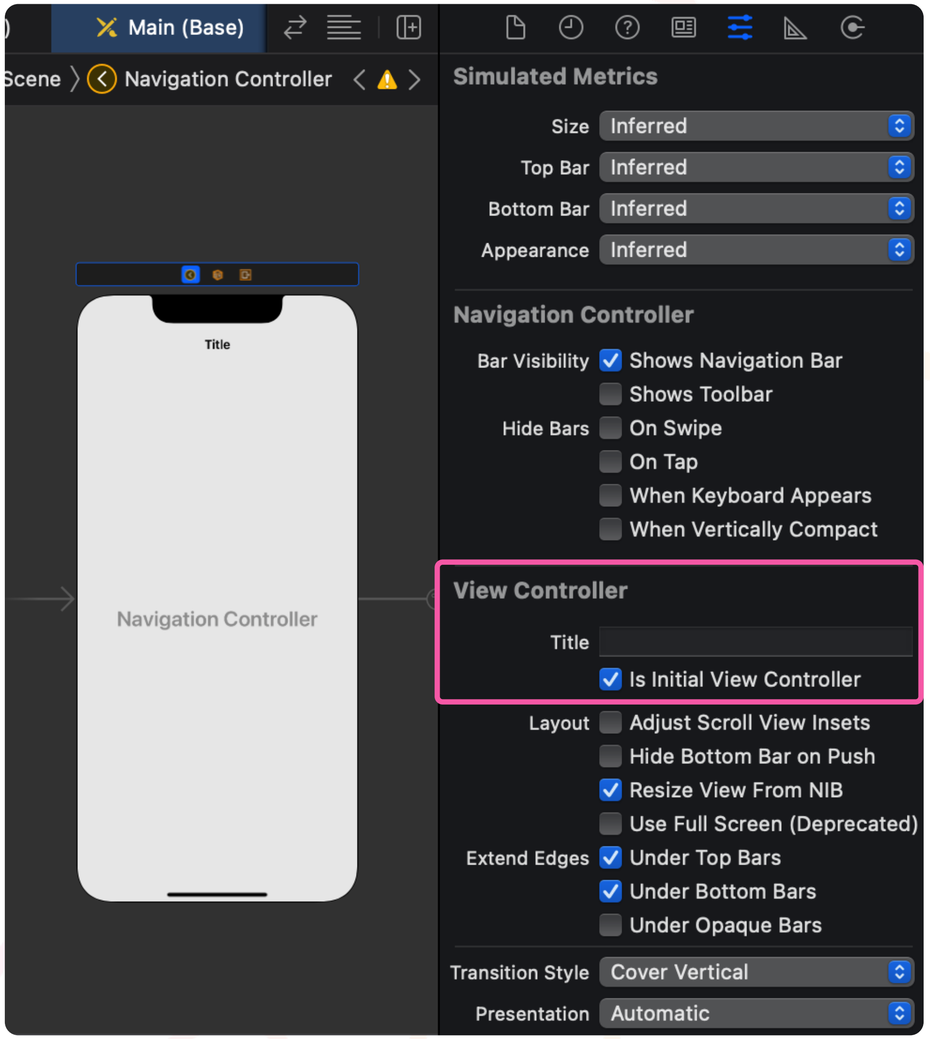
화면 전환 | Navigation



3. 네이게이션 컨트롤러를 선택해주세요



화면 전환 | Navigation



Is initial View Controller

: 스토리 보드의 진입 지점이 되는 뷰 컨트롤러

ViewController.swift

```
@IBAction func touchUpToGoSecondView(_ sender: Any) {
    guard let nextVC =
        self.storyboard?.instantiateViewController(withIdentifier:
        "SecondViewController")
        else {return}

    self.navigationController?.pushViewController(nextVC, animated: true)
}
```

SecondViewController.swift

```
@IBAction func touchUpToGoBack(_ sender: Any) {
    //self.dismiss(animated: true, completion: nil)
    self.navigationController?.popViewController(animated: true)
}
```

05

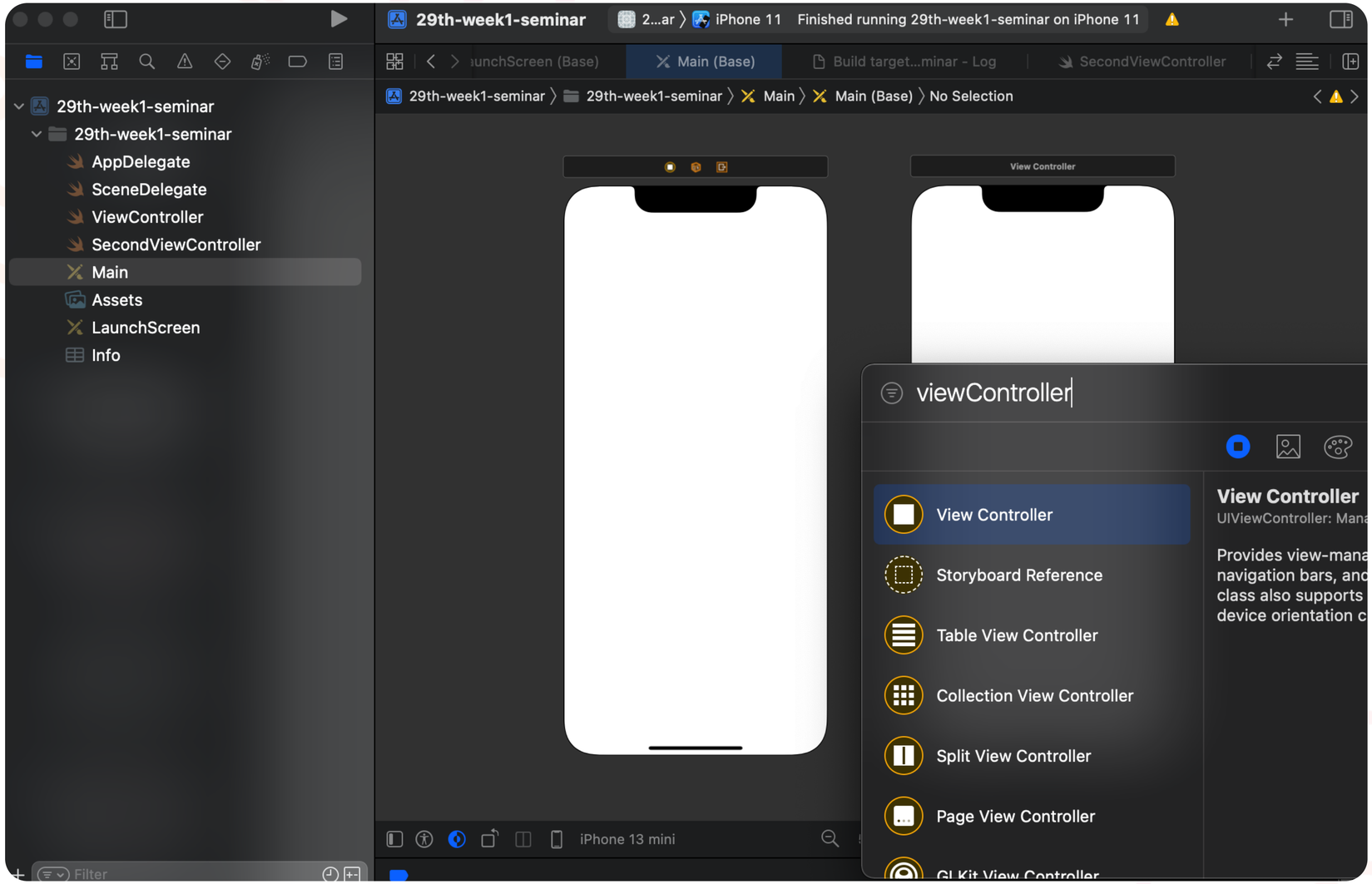
데이터 전달 기초

데이터 전달 기초

이번에는 프로퍼티를 이용한
기초적인 뷰간 데이터 전달을 알아보려 합니다!

✅3주차 다양한 데이터 전달
✅ 7주차 데이터 전달 심화

데이터 전달 기초



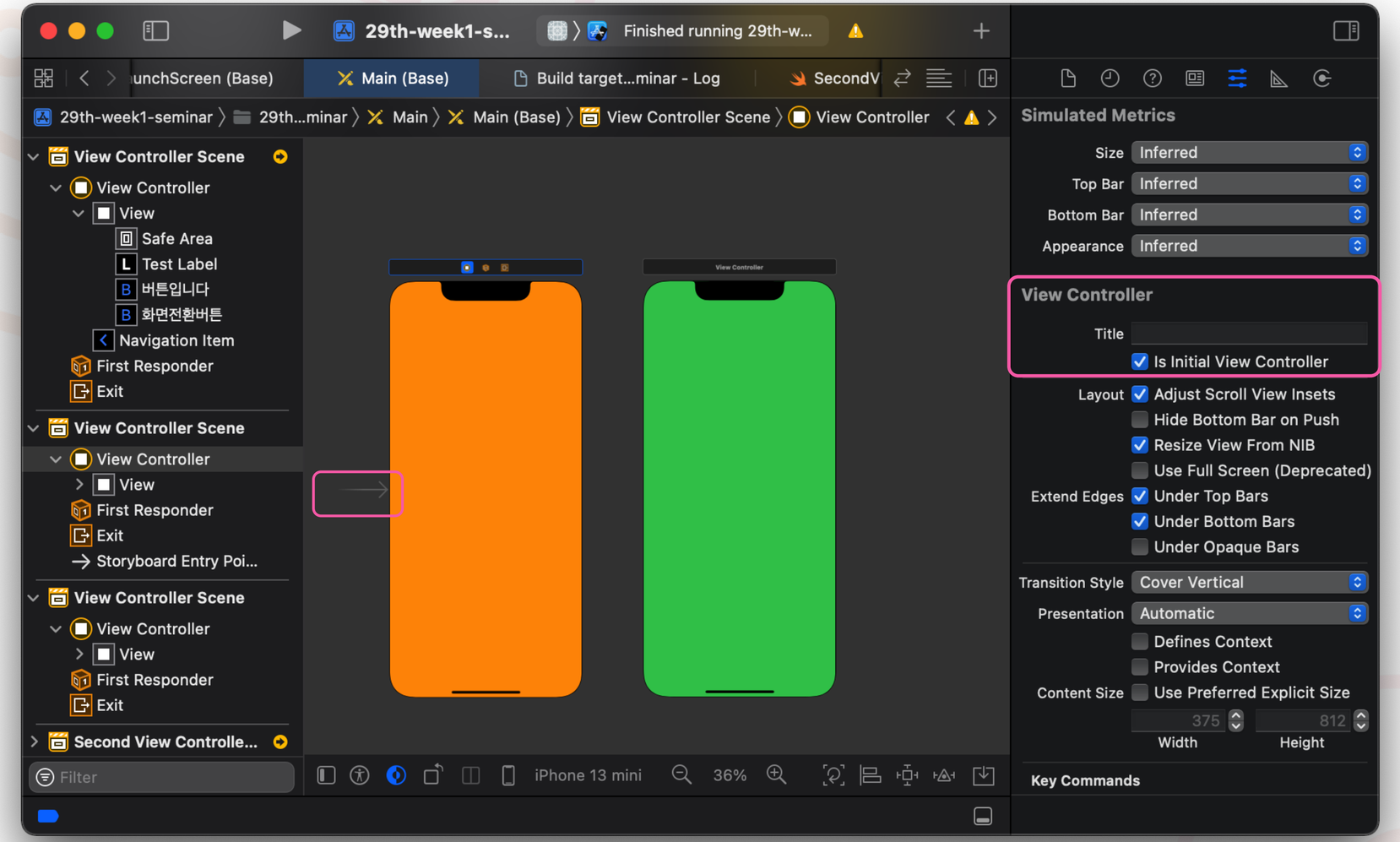
이제 조금씩 익숙해 지시고 계신가요??

cmd + shift + L을 눌러

View Controller 2개를

스토리보드에 놓아주세요!

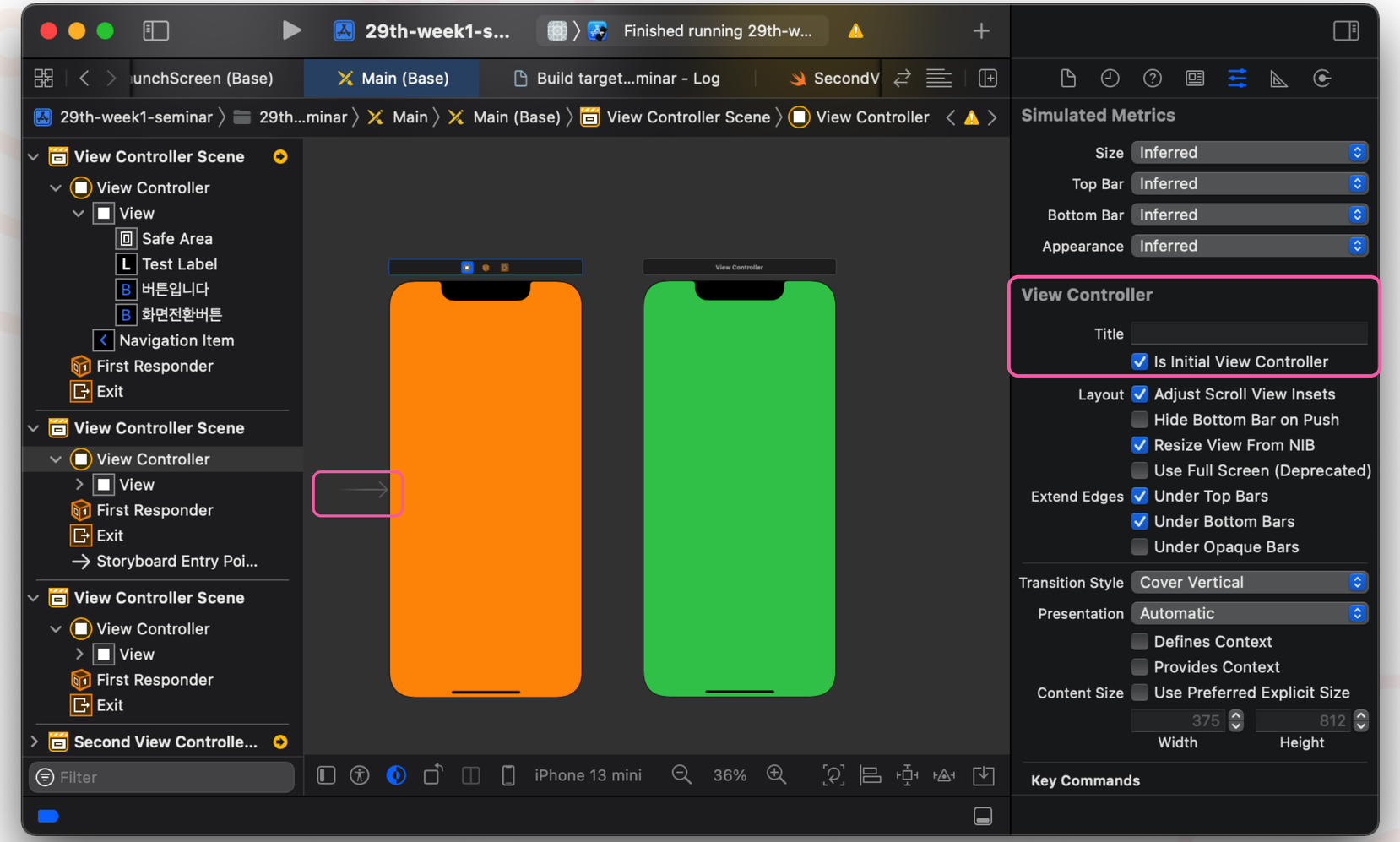
데이터 전달 기초



구분하기 쉽도록
각 view controller마다
view를 눌러 background를 변경해주세요!

그 후 스토리보드상에
왼쪽에 있는 ViewController를
Initial View Controller로 지정해주세요:)

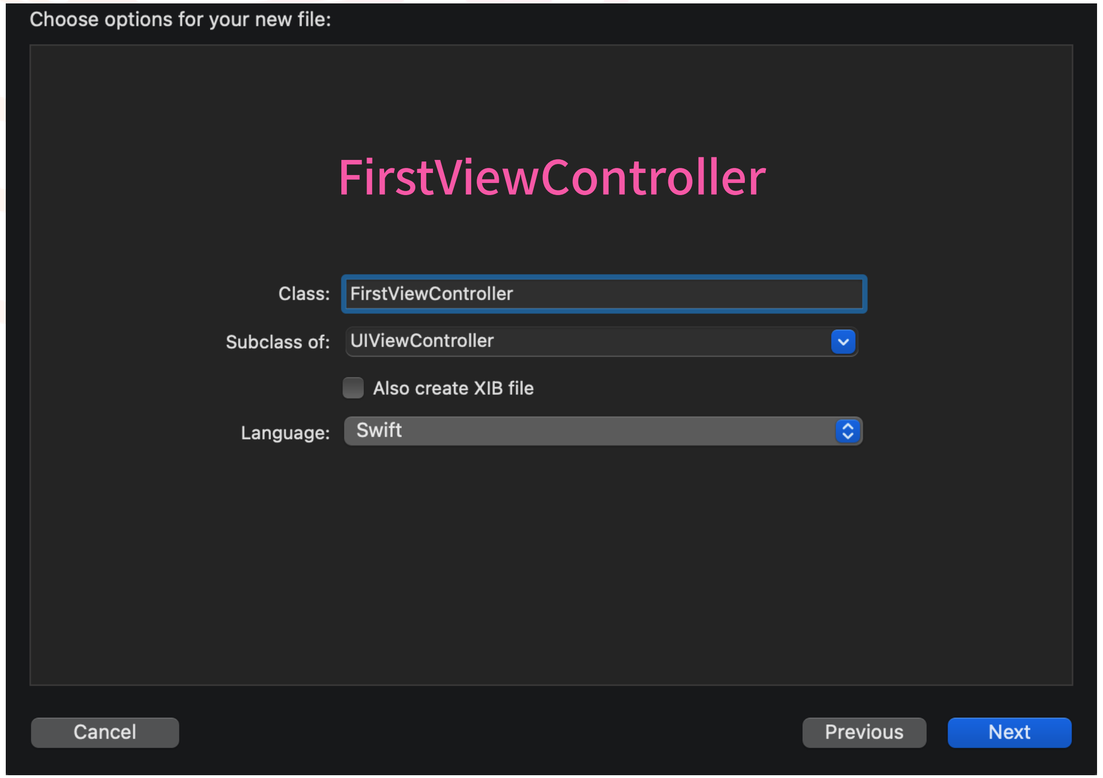
데이터 전달 기초

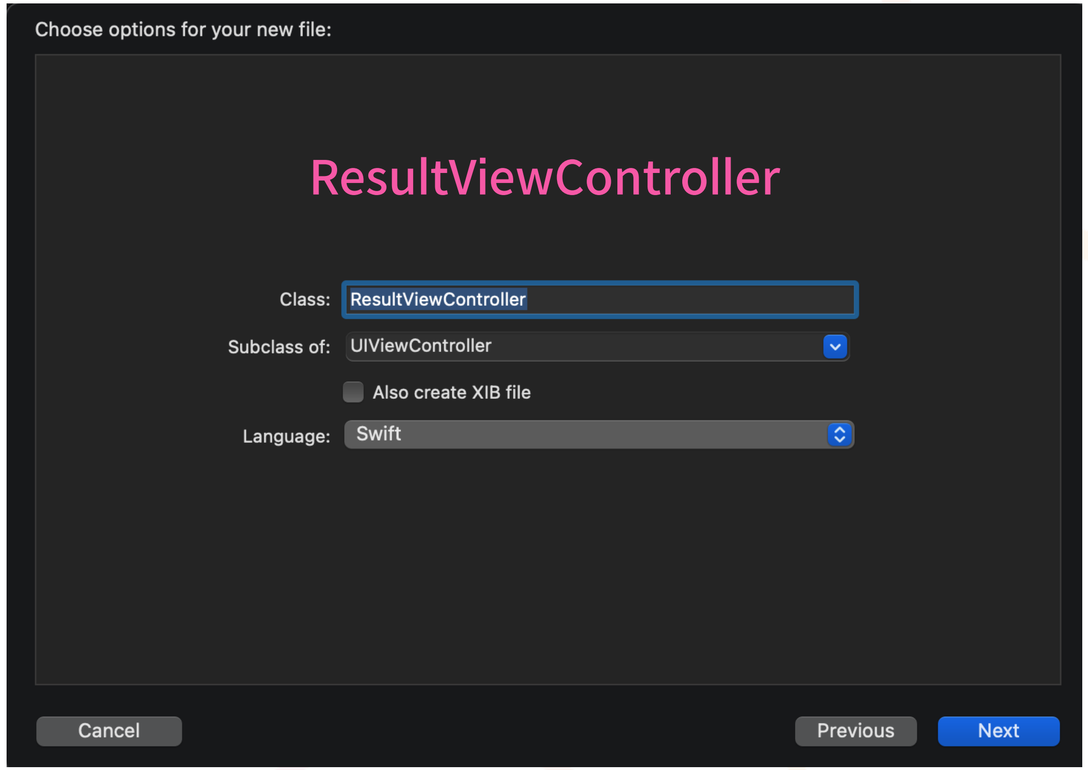


구분하기 쉽도록
각 view controller마다
view를 눌러 background를 변경해주세요!

그 후 스토리보드상에
왼쪽에 있는 ViewController를
Initial View Controller로 지정해주세요:)

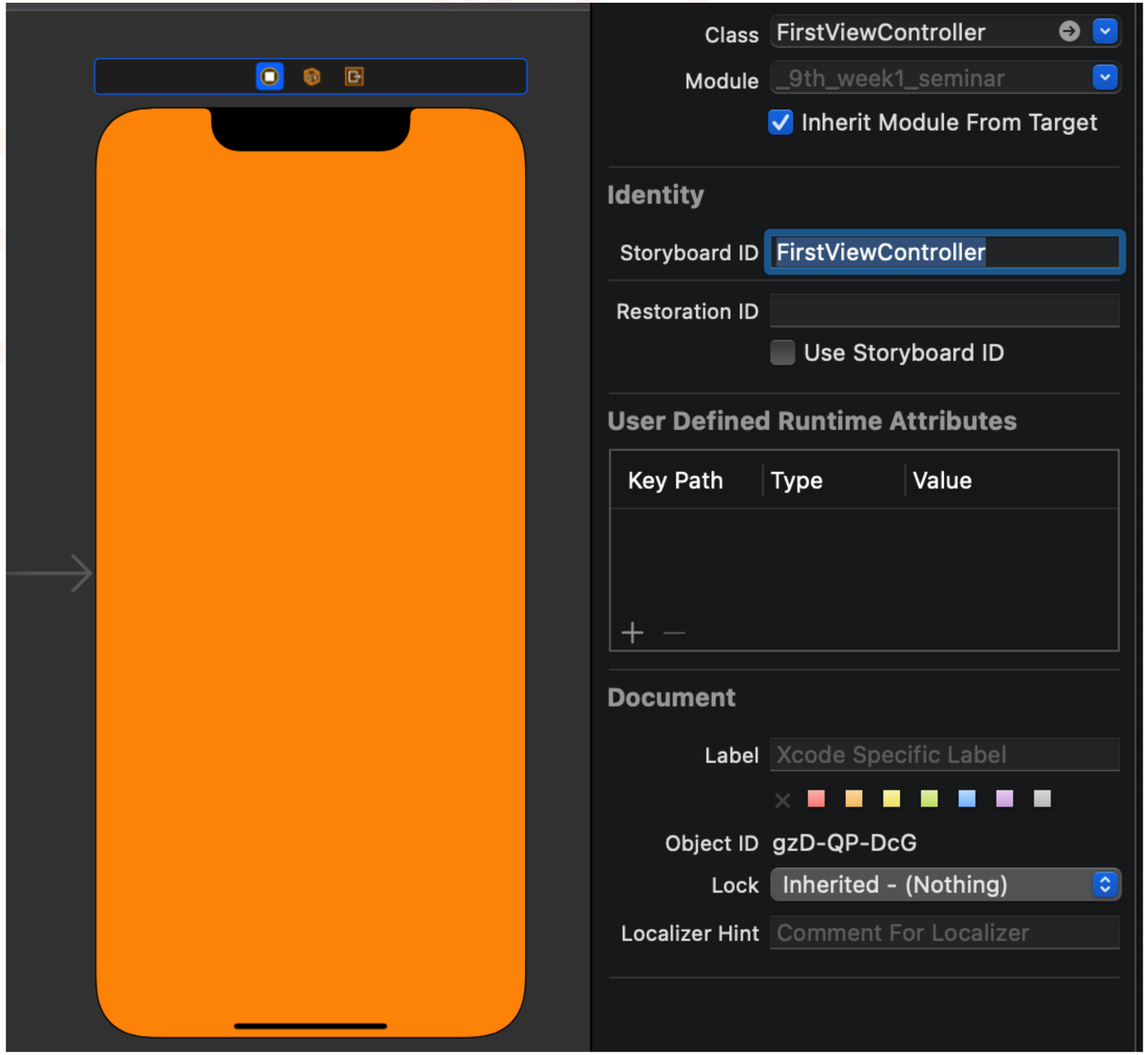
데이터 전달 기초


Choose options for your new file:
FirstViewController
Class: FirstViewController
Subclass of: UIViewController
Also create XIB file
Language: Swift
Cancel
Previous
Next

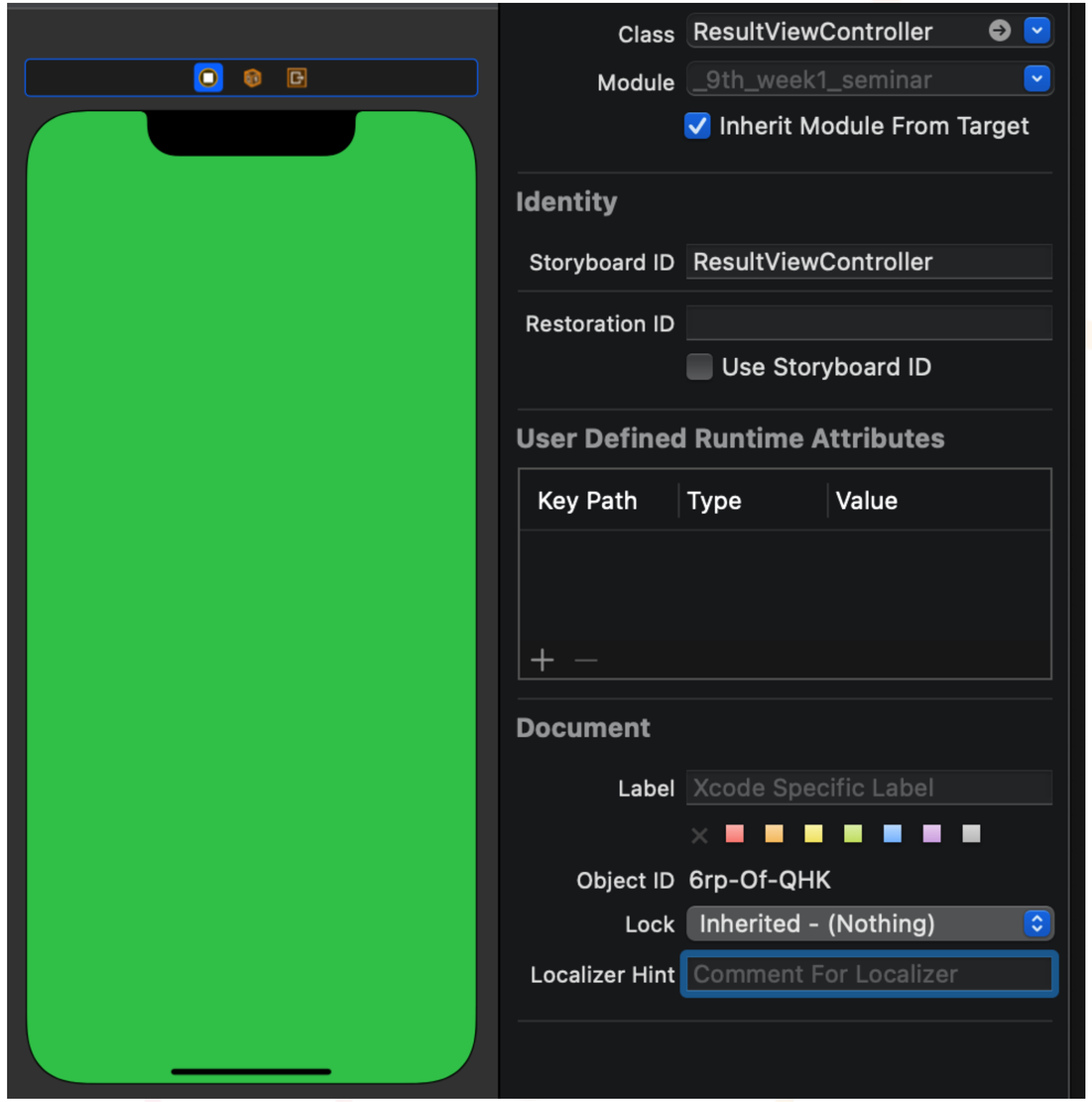

Choose options for your new file:
ResultViewController
Class: ResultViewController
Subclass of: UIViewController
Also create XIB file
Language: Swift
Cancel
Previous
Next

데이터 전달 기초

FirstViewController



ResultViewController



데이터 전달 기초

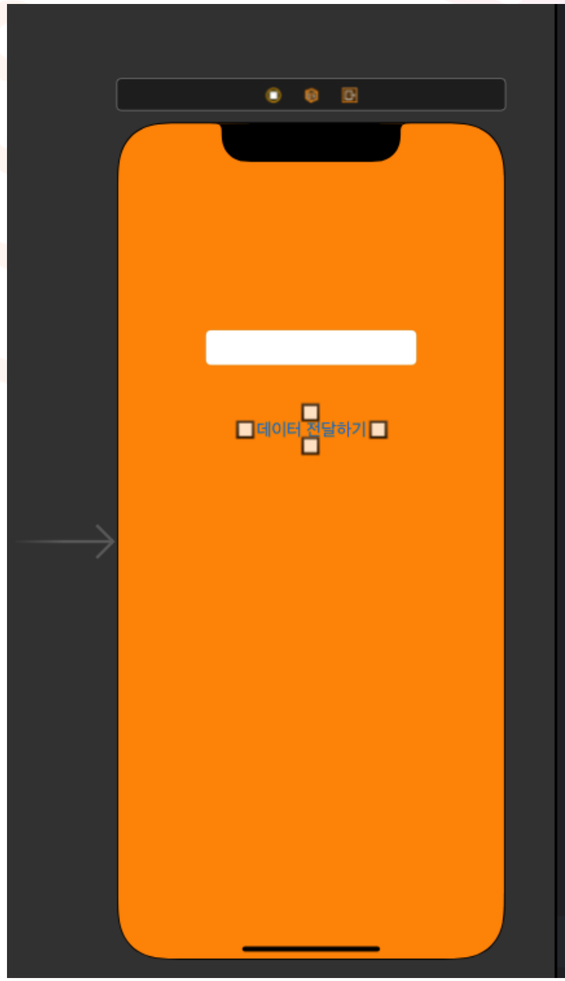
전달할 Text를 넣을 TextField

화면전환 버튼

전달받은 Text 나타낼 Label

뒤로가기 화면전환 버튼

데이터 전달 기초




```
import UIKit

class FirstViewController: UIViewController {

    @IBOutlet weak var dataTextField:
        UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()

    }

    @IBAction func touchUpToSendData(_
        sender: Any) {
    }

}
```

textField는 IBOutlet으로

button은 IBAction으로

연결해주세요:)

데이터 전달 기초

```
5   //  Created by 장혜령 on 2021/10/02.
6   //
7
8   import UIKit
9
10  class ResultViewController: UIViewController {
11
13      @IBOutlet weak var dataLabel: UILabel!
13
14      override func viewDidLoad() {
15          super.viewDidLoad()
16      }
17
19      @IBAction func touchUpToGoBack(_ sender: Any) {
19      }
20
21  }
22
```

label은 IBOutlet으로

button은 IBAction으로

연결해주세요:)

데이터 전달 기초

```
//

import UIKit

class ResultViewController: UIViewController {

    @IBOutlet weak var dataLabel: UILabel!
    var message: String?

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func touchUpToGoBack(_ sender: Any) {
    }

}
```

- 이전 화면에서 전달되는 값을 받기 위해 **프로퍼티를 추가**하는 작업
- IBOutlet 변수는 외부에서 값을 직접 대입할 수 없는 변수
- IBOutlet 변수는 외부에서 직접 참조할 수 없음.
- 그렇기 때문에, 두번째 화면으로 값을 대입하기 위해
  따로 값을 지정해 주어야 함.

데이터 전달 기초

```
class ResultViewController: UIViewController {

    @IBOutlet weak var dataLabel: UILabel!
    var message: String?

    override func viewDidLoad() {
        super.viewDidLoad()
        setMessageInLabel()
    }

    func setMessageInLabel(){
        if let msg = message {
            dataLabel.text = msg
            dataLabel.sizeToFit()
        }
    }

    @IBAction func touchUpToGoBack(_ sender: Any) {
        self.dismiss(animated: true, completion: nil)
    }
}
```

옵셔널 바인딩으로
안전하게 값을 꺼낸 뒤
dataLabel 의 text로 넣습니다

데이터 전달 기초

```
@IBAction func touchUpToSendData(_ sender: Any) {
    guard let nextVC =
        self.storyboard?.instantiateViewController(withIdentifier:
        "ResultViewController") else {return}
```

Summary

No overview available.

Declaration

```
let nextVC: UIViewController
```

Declared In

FirstViewController.swift

nextVC의 타입 : UIViewController

```
@available(iOS 5.0, *)
open class UIStoryboard : NSObject {


    public /*not inherited*/ init(name: String, bundle
        storyboardBundleOrNil: Bundle?)


    open func instantiateInitialViewController() -> UIViewController?


    open func instantiateViewController(withIdentifier identifier: String)
        -> UIViewController
}
```

storyboard의 instantiateViewController의 return값이 ViewController 이기 때문입니다!

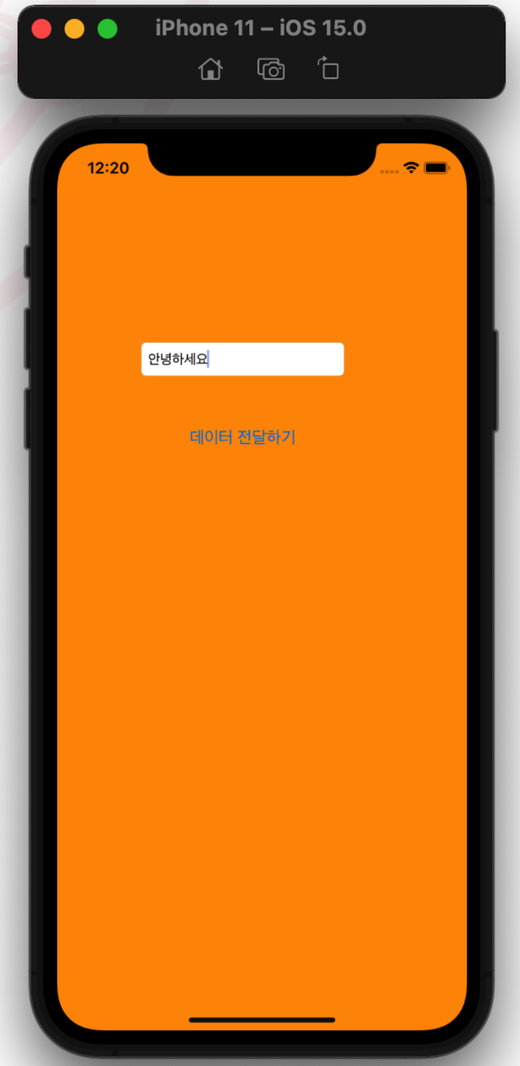
데이터 전달 기초

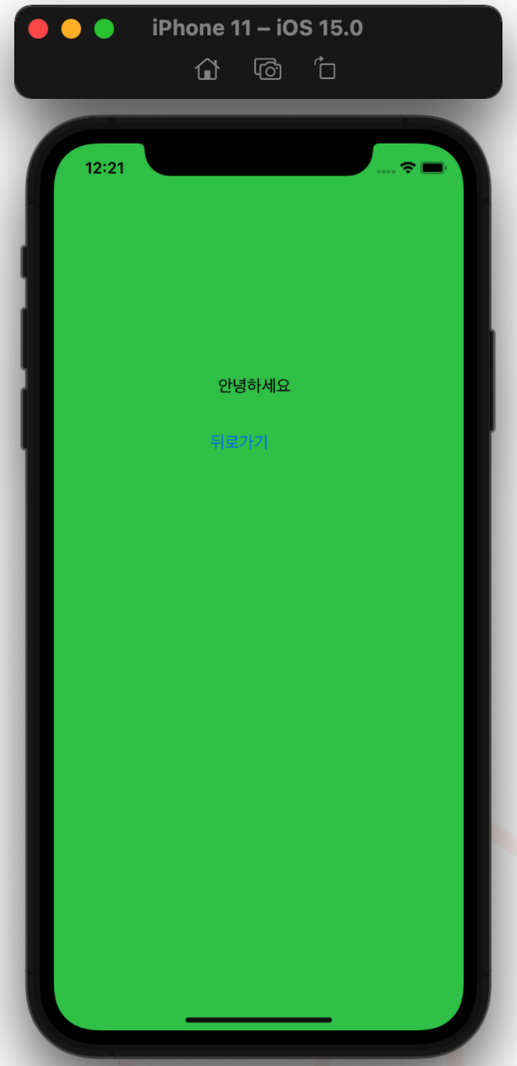
```
@IBAction func touchUpToSendData(_ sender: Any) {
    guard let nextVC = self.storyboard?.instantiateViewController(withIdentifier:
        "ResultViewController") else {return}

    nextVC.message                    ⊗ Value of type 'UIViewController' has no member 'message'
}
```

우리는 ViewController를 상속받은 ResultViewController에 message란 프로퍼티를 넣었는데
부모인 ViewController에게 자식의 프로퍼티를 묻는 것과 동일합니다.

```
@IBAction func touchUpToSendData(_ sender: Any) {
    guard let nextVC = self.storyboard?.instantiateViewController(withIdentifier:
        "ResultViewController") as? ResultViewController else {return}

    nextVC.message = dataTextField.text
    nextVC.modalPresentationStyle = .fullScreen
    self.present(nextVC, animated: true, completion: nil)
}
```

데이터 전달 기초


iPhone 11 – iOS 15.0
12:20
안녕하세요
데이터 전달하기


iPhone 11 – iOS 15.0
12:21
안녕하세요
뒤로가기

다음주, 2차 세미나에서 만나요!

긴 시간동안 너무 고생많으셨습니다!
맛있는 저녁 드세요!

1차 세미나 피드백

더 좋은 iOS 파트를 위해 소중한 의견 꼭 부탁드리겠습니다 ❤️