

# iOS Seminar 0

Wafflestudio 2024





# 세미나 Overview

## 세미나장 소개

- 최유림
- Wafflestudio 19.5 iOS
- 컴공 19 / 시각디자인





## iOS 개발의 장단점

- 장점
  - 예쁘다! (끝)
    - 기본 UI Component만 잘 짜맞춰도 낫배드
    - iOS를 위한 코드 하나만 작성해도 iPad, Macbook에서 설치 가능
- 단점
  - 매년 WWDC가 진행됨에 따라 새롭게 공부할 것들이...
    - 올해 2월에 개설된 따끈한 YouTube

4월 3일 오후 10:06  
수치보기귀차나

 최유림 4월 3일 오후 10:07  
자꾸 그러면 안드 더 못생겨져

 1 

4월 3일 오후 10:07  
진짜못생긴거보여줘?

 1 



## 세미나 일정

- 0회: 9월 12일 (목)
- 1회: 9월 26일 (목)
- 2회: 10월 10일 (목)
- 3회: 10월 31일 (목)
- 4회: 11월 14일 (목)
- 5회: 11월 28일 (목)

## 통과 기준

- 전 세미나 대면 출석
  - 학교 수업과 겹치면 실시간 비대면도 인정
- 모든 과제 제출&통과

## 과제

- 과제 마감 : 다음 세미나 당일 오전 8시까지
  - ex) 과제2 마감: 10월 31일 목 오전 8시
- Grace Day (기본)
  - 와플스튜디오에서 진행하는 코모(코딩 모임) 2회 참여 시 1일 추가
- Grace Day (iOS Seminar 단독)
  - 기본적으로 2일 제공
  - 슬랙 #ios-질문 채널에서 질문 혹은 답변 시 1일 추가
    - 최대 1회
- 각 과제의 **Bonus** 스펙을 구현하시는 경우 2개 당 1일 추가
  - 제한 없음
- 모든 Grace Day는 한 과제에 몰아서 사용하셔도 됩니다!



## 무엇을 다루나요?

- UIKit, SwiftUI를 이용한 UI 구현
- Alamofire를 이용한 네트워크 통신
- Concurrency
- Design Pattern
- Life Cycle
- 그 외

## 어떻게 공부하면 좋을까요?

- UIKit 공식 문서
- SwiftUI 공식 문서
- (그 외 수많은 구글링)
  - 단, 작성 일자에 주의
  - 갑자기 Objective-C를 만날 수도...



Storyboard

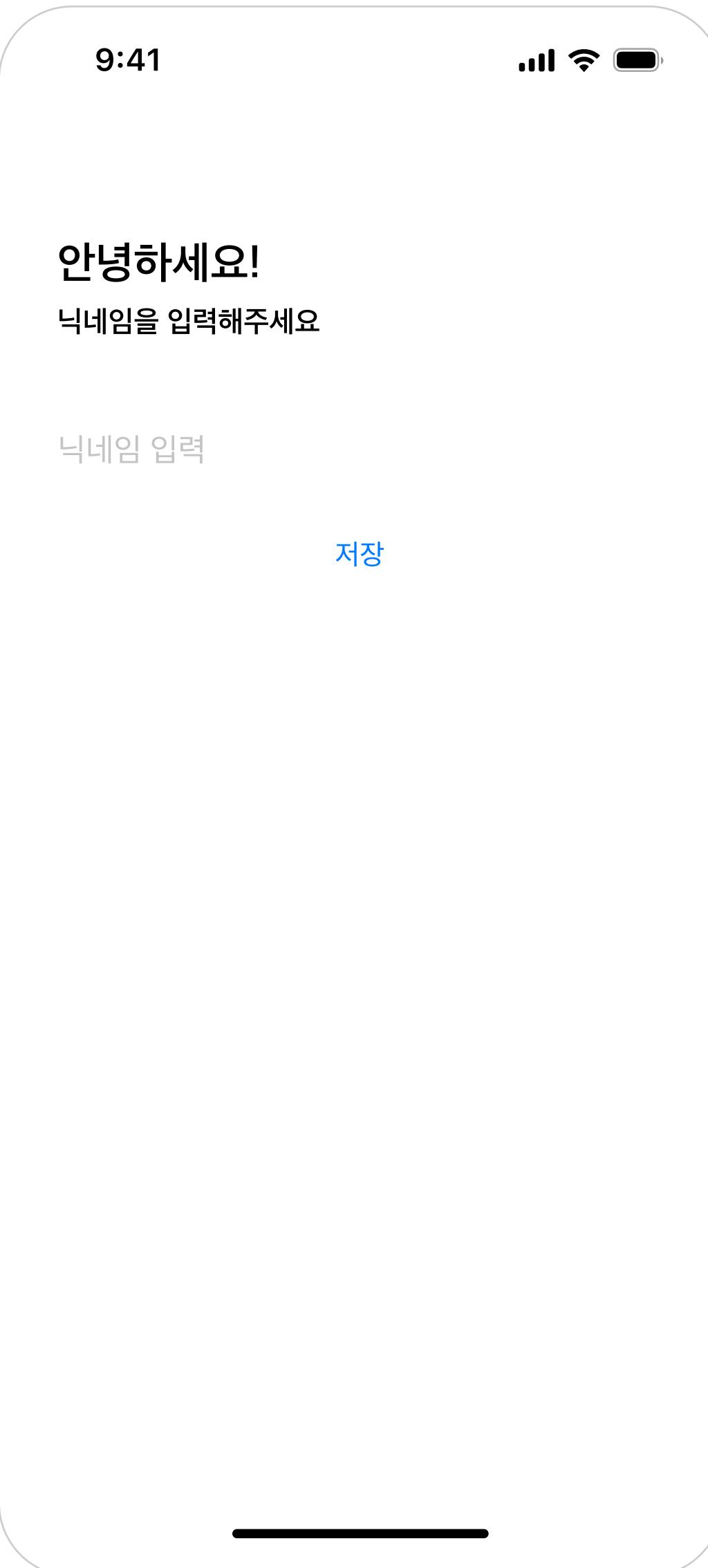
# Storyboard

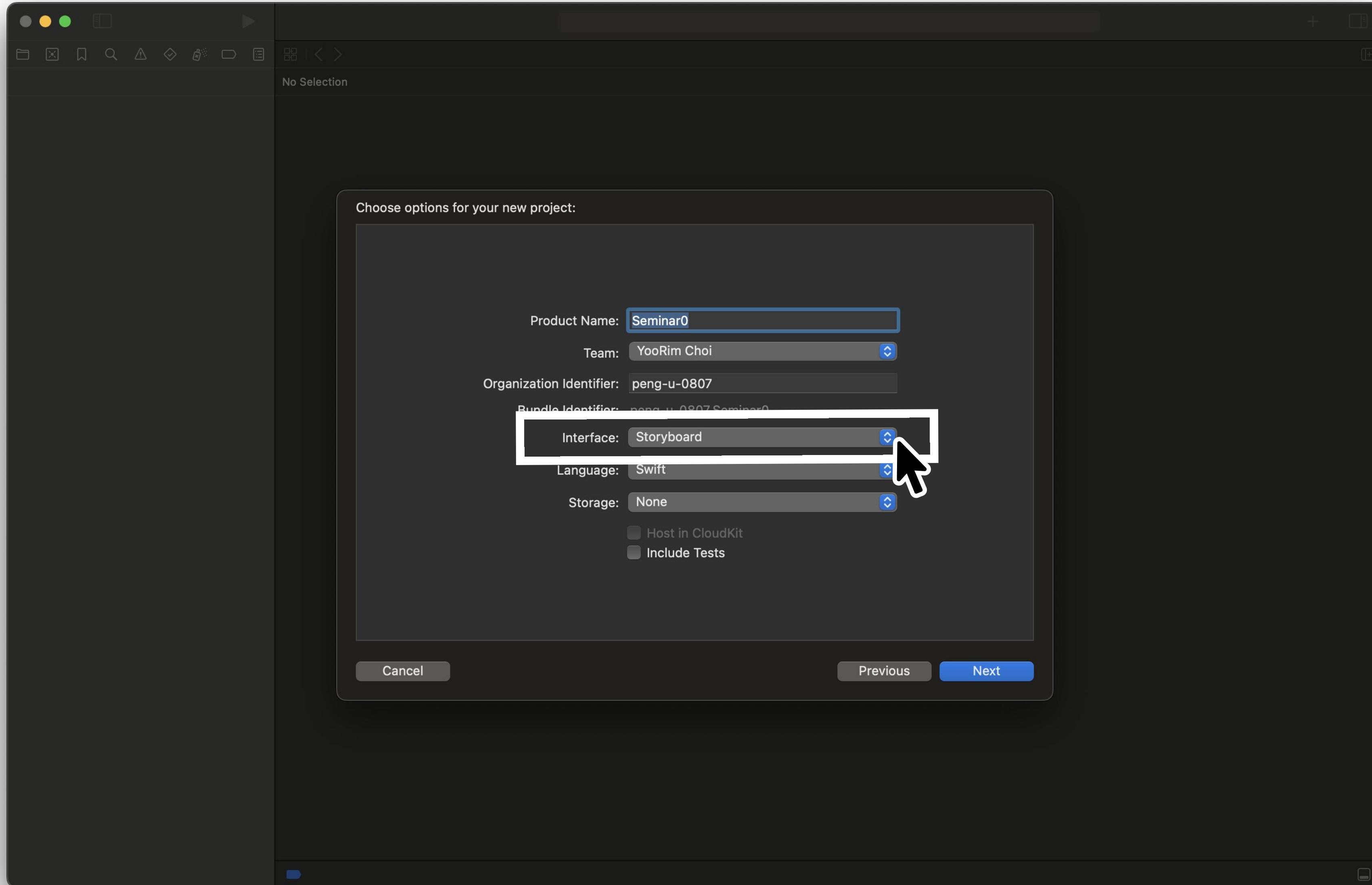


## Storyboard

# iOS 개발에서 UI를 구현하기 위한 방법

- UIKit
  - Storyboard 有 / 無
  - SwiftUI
  - 세미나에서는 UIKit+No Storyboard 위주로 진행
    - (+) SwiftUI 맛보기

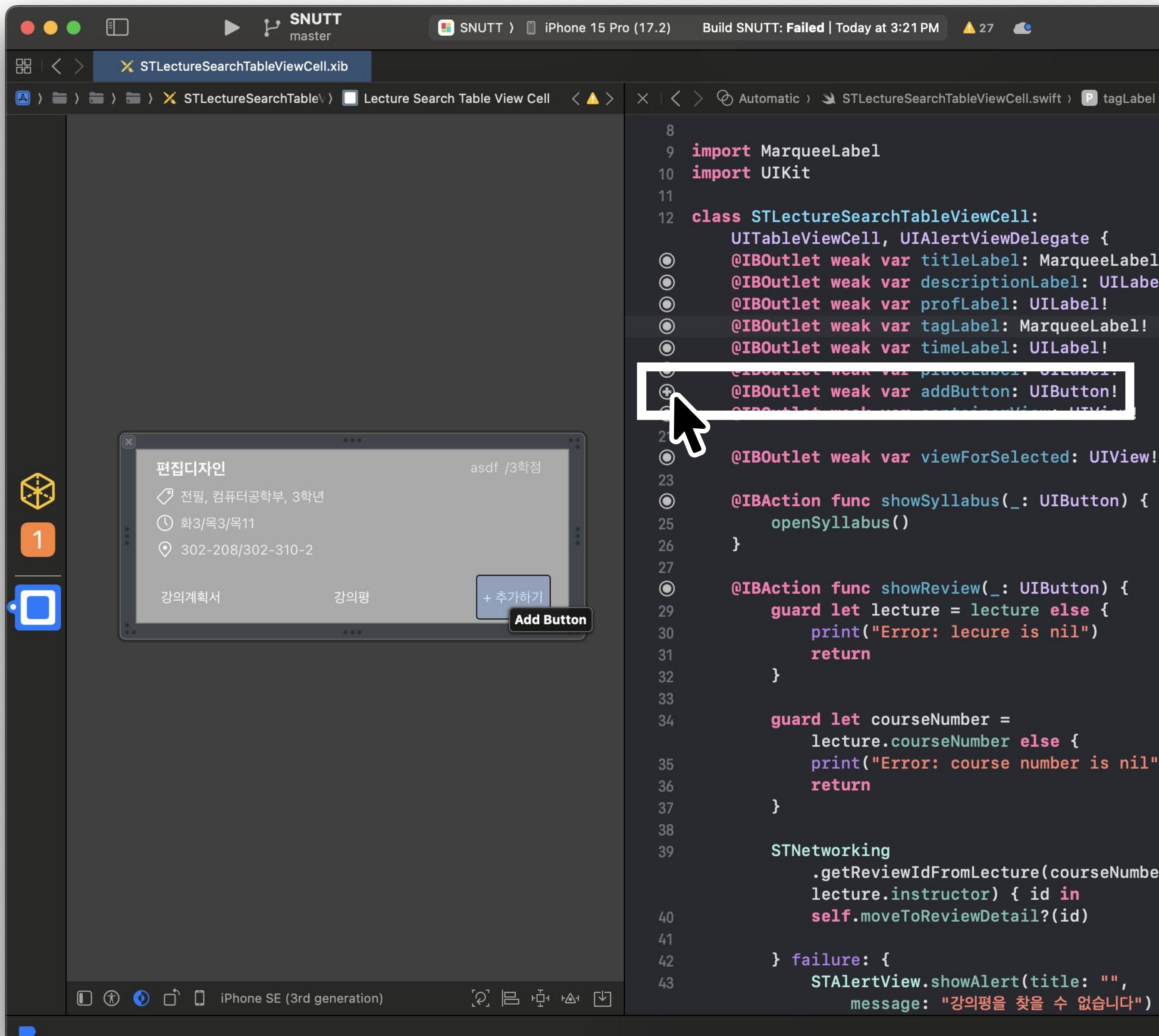


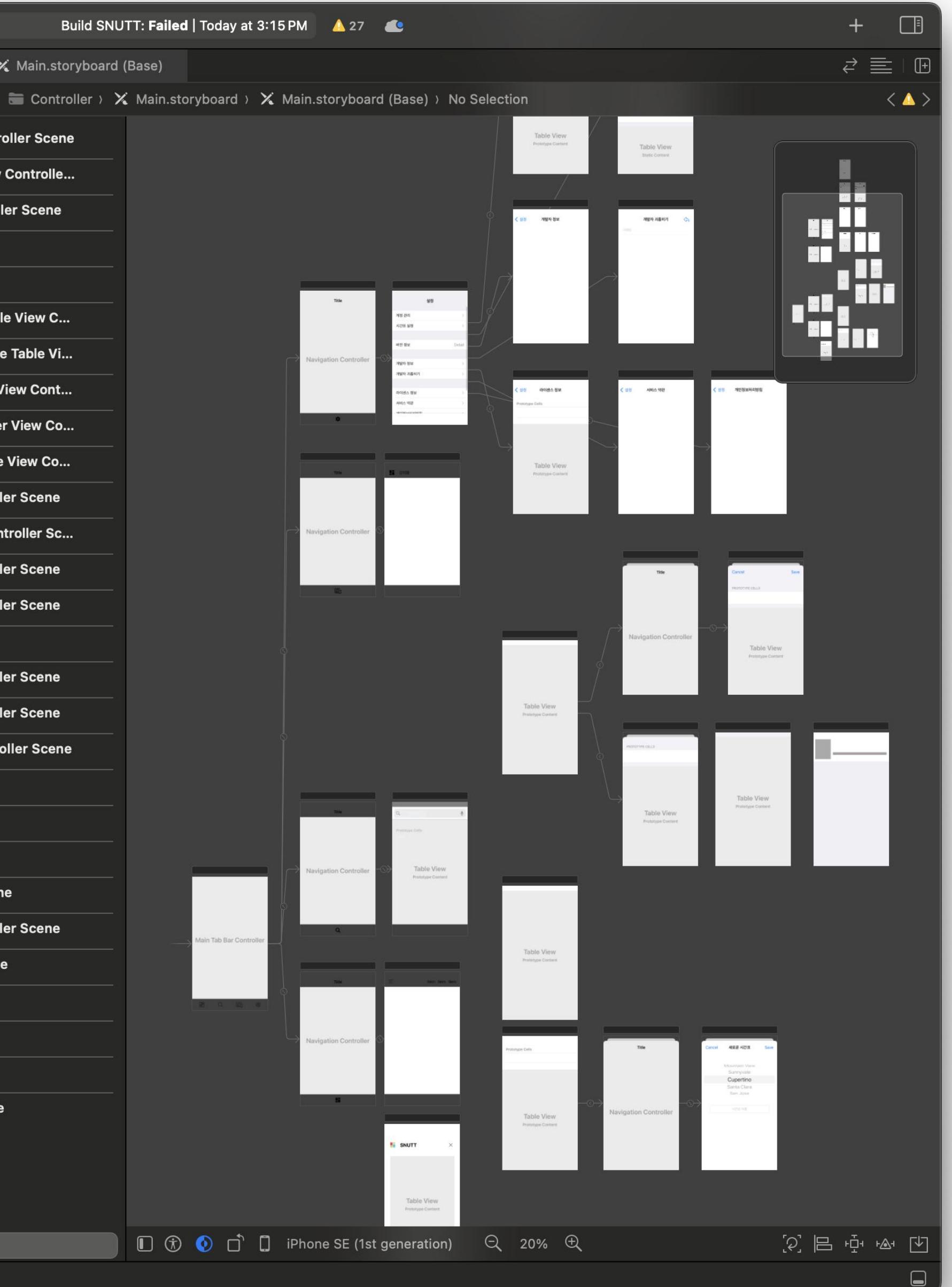
 **Storyboard**

# Storyboard

## Storyboard 좋긴한데...

- 직관적인 인터페이스
  - 추가할 수 있는 UIView의 목록을 확인
  - Drag&Drop으로 UIView 추가
  - Hover Action으로 연결된 UIView 확인





그러나..

- 프로젝트의 규모가 커질수록 관리가 어려워짐
- 수정 사항 반영 및 실행에 오랜 시간이 걸림



# (1) Main.storyboard 삭제

The screenshot shows two parts of the Xcode interface. On the left, the Project Navigator displays a project named 'RemoveStoryboard' with a target 'RemoveStoryboard'. The 'Info' tab is selected in the target settings. A table shows target properties, with 'Main storyboard file base name' set to 'Main'. On the right, the File Inspector shows the 'example' group of the 'Info.plist' file. The 'Storyboard Name' key is set to 'Main'.

Key	Type	Value
Bundle name	String	\$(PRODUCT_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Main storyboard file base name	String	Main
Bundle version	String	\$(CURRENT_PROJECT_VERSION)
Launch screen interface file base name	String	LaunchScreen
Executable file	String	\$(EXECUTABLE_NAME)
Application requires iPhone environment	Boolean	YES
Supported interface orientations (iPhone)	Array	(3 items)

Key	Type	Value
Information Property List	Dictionary	(1 item)
Application Scene Manifest	Dictionary	(2 items)
Enable Multiple Windows	Boolean	NO
Scene Configuration	Dictionary	(1 item)
Window Application Session Role	Array	(1 item)
Item 0 (Default Configuration)	Dictionary	(3 items)
Configuration Name	String	Default Configuration
Delegate Class Name	String	\$(PRODUCT_MODULE_NAME).SceneDelegate
Storyboard Name	String	Main





## (2) SceneDelegate 변경

```
10 class SceneDelegate: UIResponder, UIWindowSceneDelegate {  
11  
12     var window: UIWindow?  
13  
14  
15     func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions:  
16                 UIScene.ConnectionOptions) {  
17         // Use this method to optionally configure and attach the UIWindow `window` to the provided  
18         // UIWindowScene `scene`.  
19         // If using a storyboard, the `window` property will automatically be initialized and attached to  
20         // the scene.  
21         // This delegate does not imply the connecting scene or session are new (see  
22             // `application:configurationForConnectingSceneSession` instead).  
23         guard let _ = (scene as? UIWindowScene) else { return }  
24     }  
25 }
```

```
10 class SceneDelegate: UIResponder, UIWindowSceneDelegate {  
11  
12     var window: UIWindow?  
13  
14  
15     func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions:  
16                 UIScene.ConnectionOptions) {  
17         guard let windowScene = (scene as? UIWindowScene) else { return }  
18         window = UIWindow(frame: windowScene.coordinateSpace.bounds)  
19         window?.windowScene = windowScene  
20         window?.rootViewController = UINavigationController(rootViewController: ViewController())  
21         window?.makeKeyAndVisible()  
22     }  
23 }
```

- SceneDelegate에 있는 다른 메서드들이 궁금하다면...
  - 참고 링크



Auto Layout

# Auto Layout

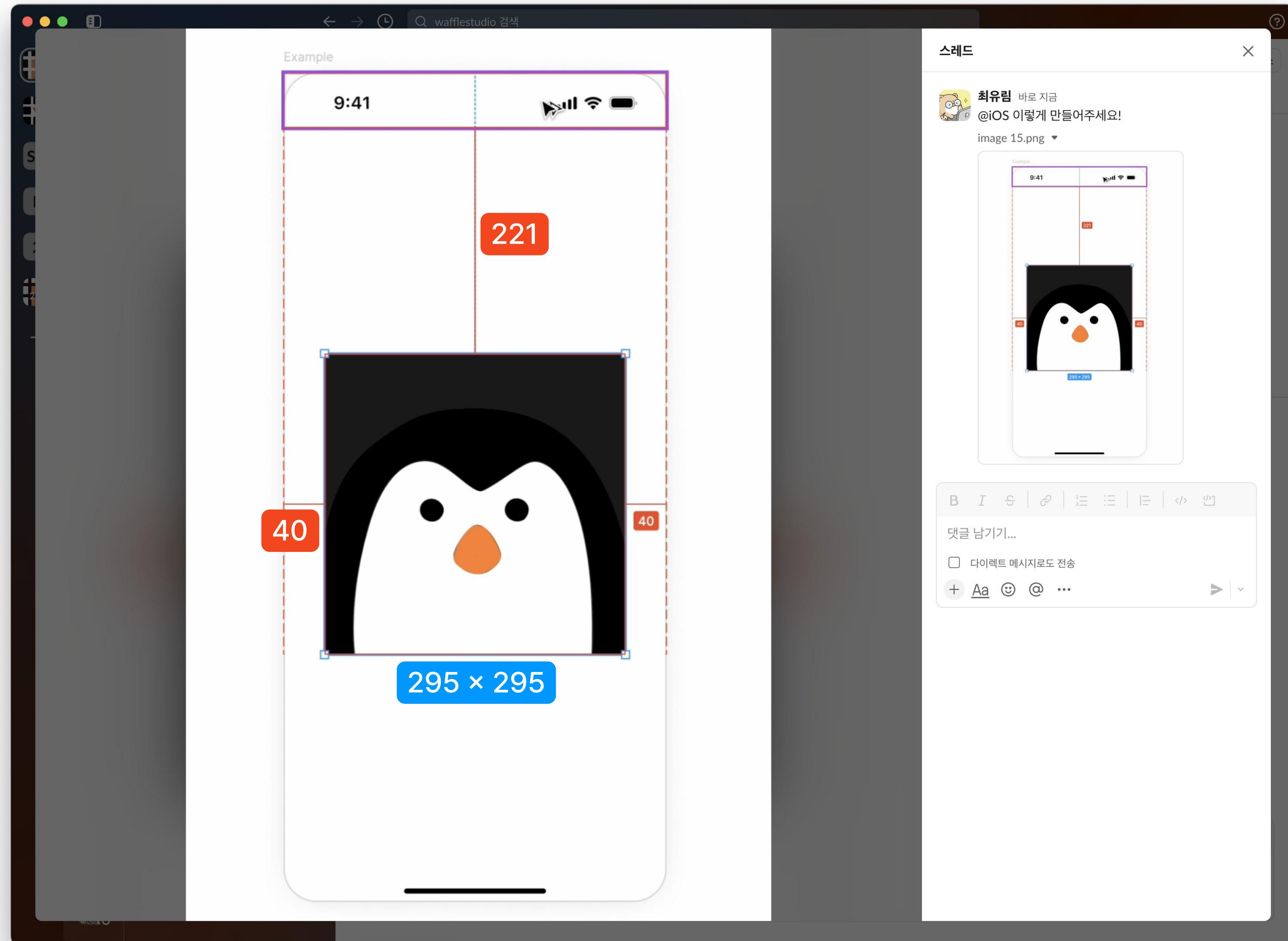


# Auto Layout

- 다른 View를 기준으로 일정한 Constraint를 주어서 View를 상대적으로 그리는 방법
  - Auto Layout dynamically calculates the size and position of all the views in your view hierarchy, based on constraints placed on those views. (공식문서)
  - 만약 Auto Layout를 적용하지 않는다면?



# Auto Layout





## Auto Layout

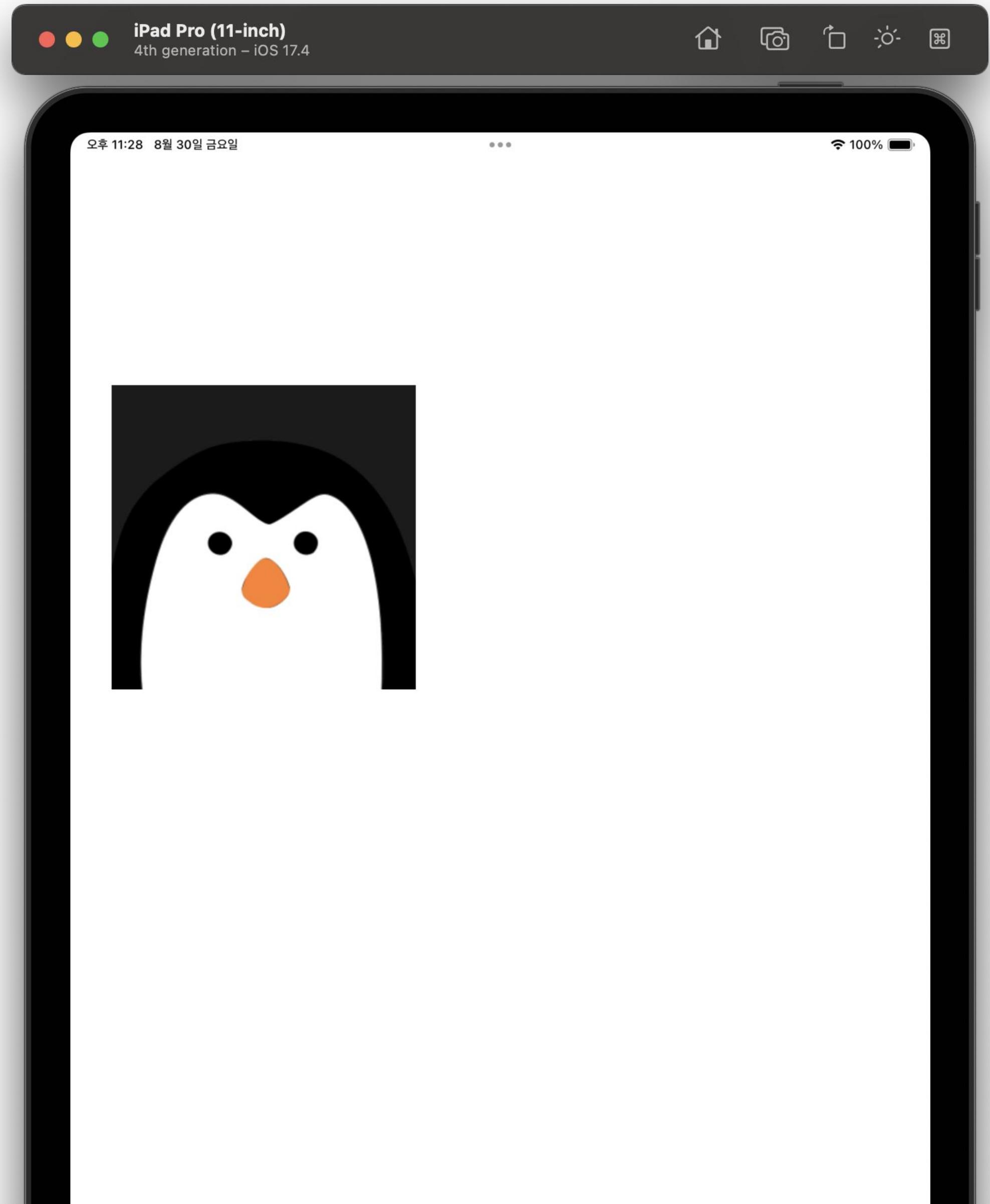
```
8 import UIKit
9
10 class ViewController: UIViewController {
11
12     private lazy var penguinImageView: UIImageView = {
13         let image = UIImage(named: "Penguin")
14         return .init(image: image)
15     }()
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19         view.backgroundColor = .white
20         view.addSubview(penguinImageView)
21         penguinImageView.frame = .init(x: 40, y: 221, width: 295, height: 295)
22     }
23 }
```



## Auto Layout



대부분의 경우,  
이런 결과를 의도하진 않았을 것





## Auto Layout

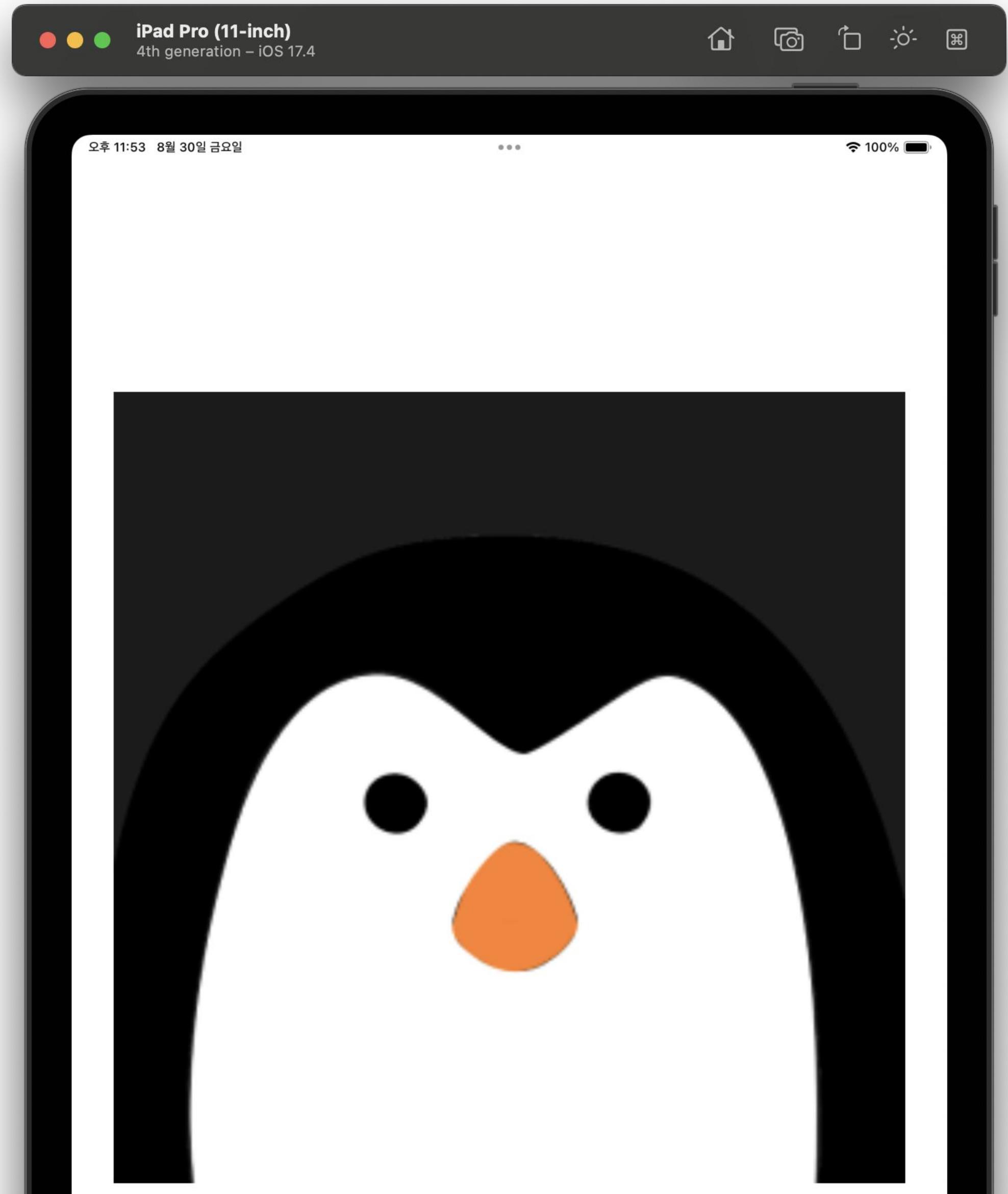
```
8 import UIKit
9
10 class ViewController: UIViewController {
11
12     private lazy var penguinImageView: UIImageView = {
13         let image = UIImage(named: "Penguin")
14         return .init(image: image)
15     }()
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19         view.backgroundColor = .white
20         view.addSubview(penguinImageView)
21         penguinImageView.translatesAutoresizingMaskIntoConstraints = false
22         NSLayoutConstraint.activate([
23             penguinImageView.widthAnchor.constraint(equalTo: penguinImageView.heightAnchor),
24             penguinImageView.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor, constant: 221),
25             penguinImageView.leadingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.leadingAnchor, constant: 40),
26             penguinImageView.trailingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.trailingAnchor, constant: -40)
27         ])
28     }
29 }
```



## Auto Layout



다양한 크기의 디바이스에  
동일한 모습의 View를 유지

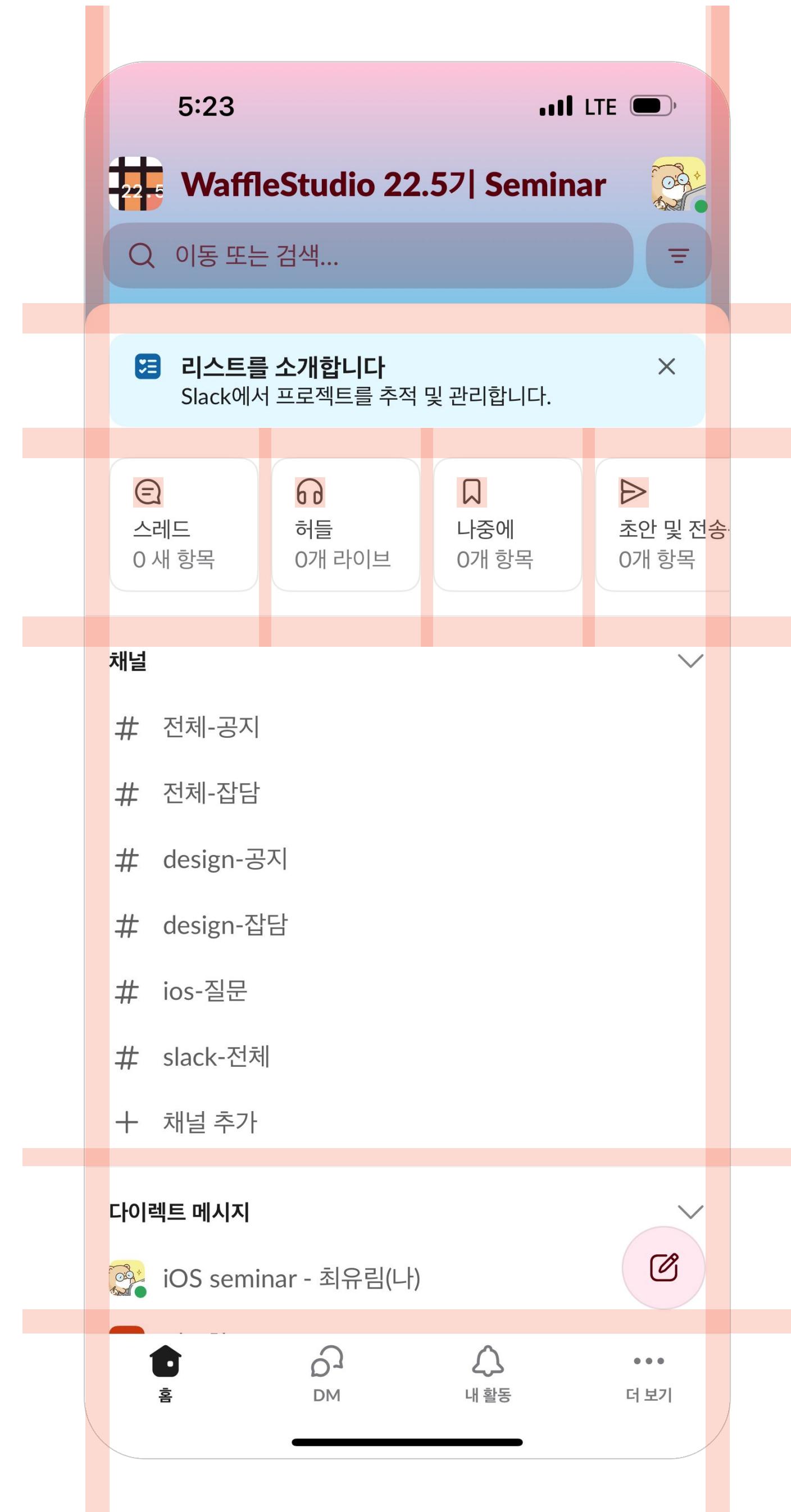




## Auto Layout

# Auto Layout 설정하는 방법

- 디바이스 크기가 달라질 때, 변하는 부분과 변하지 않는 부분 생각하기
  - 예시로, 오른쪽 사진에서의 붉은색 직사각형은 어느 디바이스든 동일
  - 워크스페이스 이름, 검색창 등은 디바이스에 따라 가로의 길이가 달라질 것
- 어느 디바이스에서든 변하지 않는 부분을 constraint로 설정
  - 좀 더 정확하게는, 상위뷰(Superview)의 크기가 바뀌어도 변하지 않는 값을 제약으로 걸고(constraint) 나머지 레이아웃은 자동으로 잡아달라는 것
- topAnchor / bottomAnchor / leadingAnchor / trailingAnchor / leftAnchor / rightAnchor
- centerXAnchor / centerYAnchor
- heightAnchor / widthAnchor





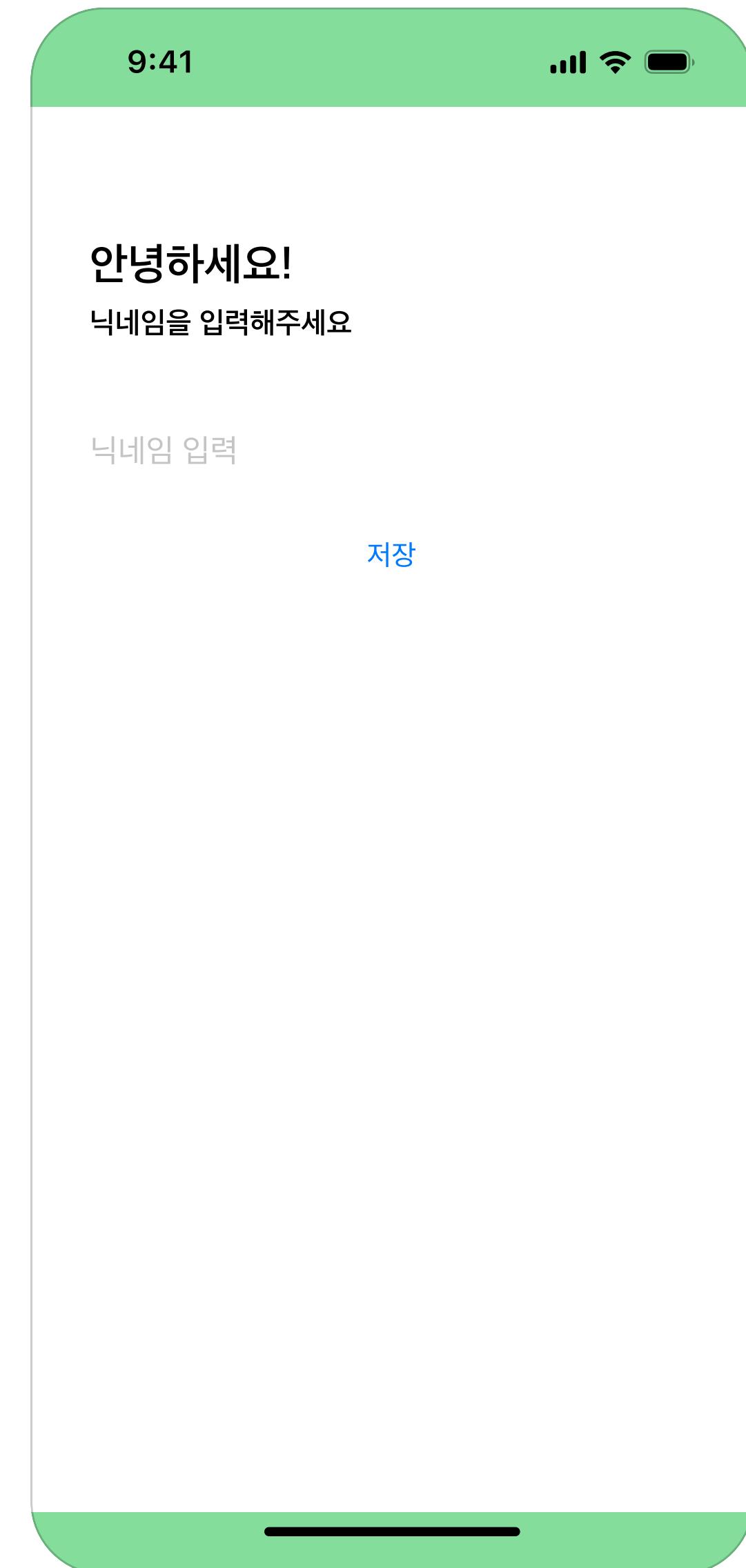
## leadingAnchor & trailingAnchor vs leftAnchor & rightAnchor

- leadingAnchor & trailingAnchor
  - 오른쪽 → 왼쪽으로 읽는 언어에서는 좌 / 우가 뒤집힐 수 있음
- leftAnchor & rightAnchor
  - 절대적인 좌 / 우
- Apple에서는 leading & trailing의 사용을 권장하는 편



## **view.safeAreaLayoutGuide**

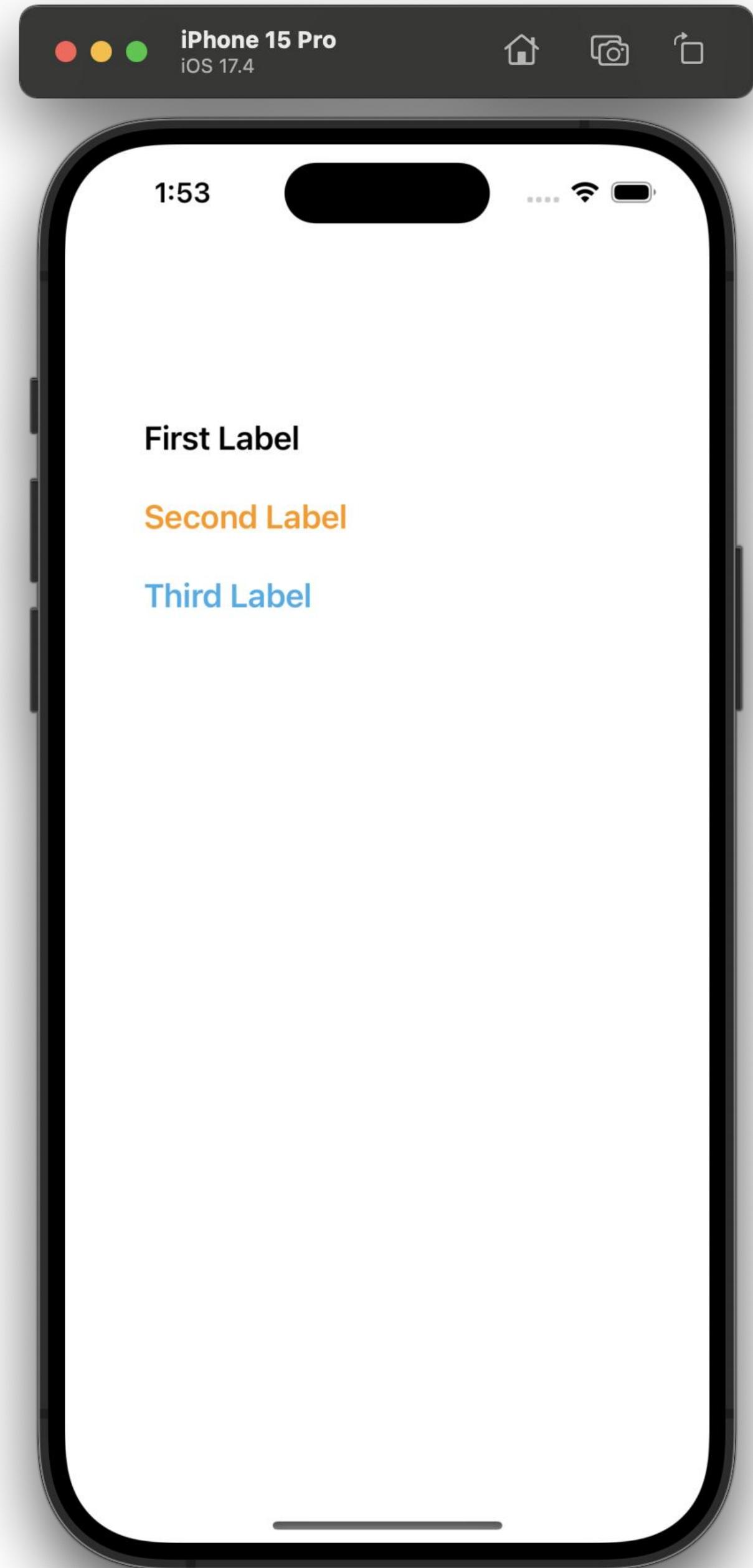
- Navigation Bar, Tab Bar, Toolbar, 그 외의 다른 상위 view에  
가려지지 않는 영역





UIStackView

# UIStackView



## UIStackView

- 일정한 간격으로 View를 나열해야 할 때 유용한 클래스
  - 공식문서
  - 오른쪽 화면을 구현하려면 어떻게 해야 할까요?



# UIStackView

- 이것도 나쁘지는 않지만...

```
35    override func viewDidLoad() {
36        super.viewDidLoad()
37        view.backgroundColor = .white
38        view.addSubview(firstLabel)
39        view.addSubview(secondLabel)
40        view.addSubview(thirdLabel)
41        firstLabel.translatesAutoresizingMaskIntoConstraints = false
42        secondLabel.translatesAutoresizingMaskIntoConstraints = false
43        thirdLabel.translatesAutoresizingMaskIntoConstraints = false
44        NSLayoutConstraint.activate([
45            firstLabel.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor, constant: 108),
46            firstLabel.leadingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.leadingAnchor, constant: 48),
47            firstLabel.trailingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.trailingAnchor, constant: -48),
48
49            secondLabel.topAnchor.constraint(equalTo: firstLabel.bottomAnchor, constant: 24),
50            secondLabel.leadingAnchor.constraint(equalTo: firstLabel.leadingAnchor),
51            secondLabel.trailingAnchor.constraint(equalTo: firstLabel.trailingAnchor),
52
53            thirdLabel.topAnchor.constraint(equalTo: secondLabel.bottomAnchor, constant: 24),
54            thirdLabel.leadingAnchor.constraint(equalTo: firstLabel.leadingAnchor),
55            thirdLabel.trailingAnchor.constraint(equalTo: firstLabel.trailingAnchor),
56        ])
57    }
```



Navigation

# Navigation



## UINavigationController

- 두 개의 UIViewController를 연결해 봅시다
  - 아까 SceneDelegate에서 작성했던 코드의 의미
    - `window?.rootViewController = UINavigationController(rootViewController: ViewController())  
window?.makeKeyAndVisible()`
  - 내부적으로 Stack을 이용하여 다른 UIViewController를 push/pop
    - “< 뒤로” 아이콘이 있는 화면들



Life Cycle

# Life Cycle



```
import UIKit

class ViewController: UIViewController {

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

    override func view
}
```

**M** `viewDidAppear(_ animated:)`

**M** `viewWillAppear(_ animated:)`

**M** `viewIsAppearing(_ animated:)`

**M** `viewDidDisappear(_ animated:)`

**M** `viewWillDisappear(_ animated:)`

**M** `viewDidLayoutSubviews()`

**M** `viewWillLayoutSubviews()`

**M** `viewLayoutMarginsDidChange()`

**M** `viewSafeAreaInsetsDidChange()`

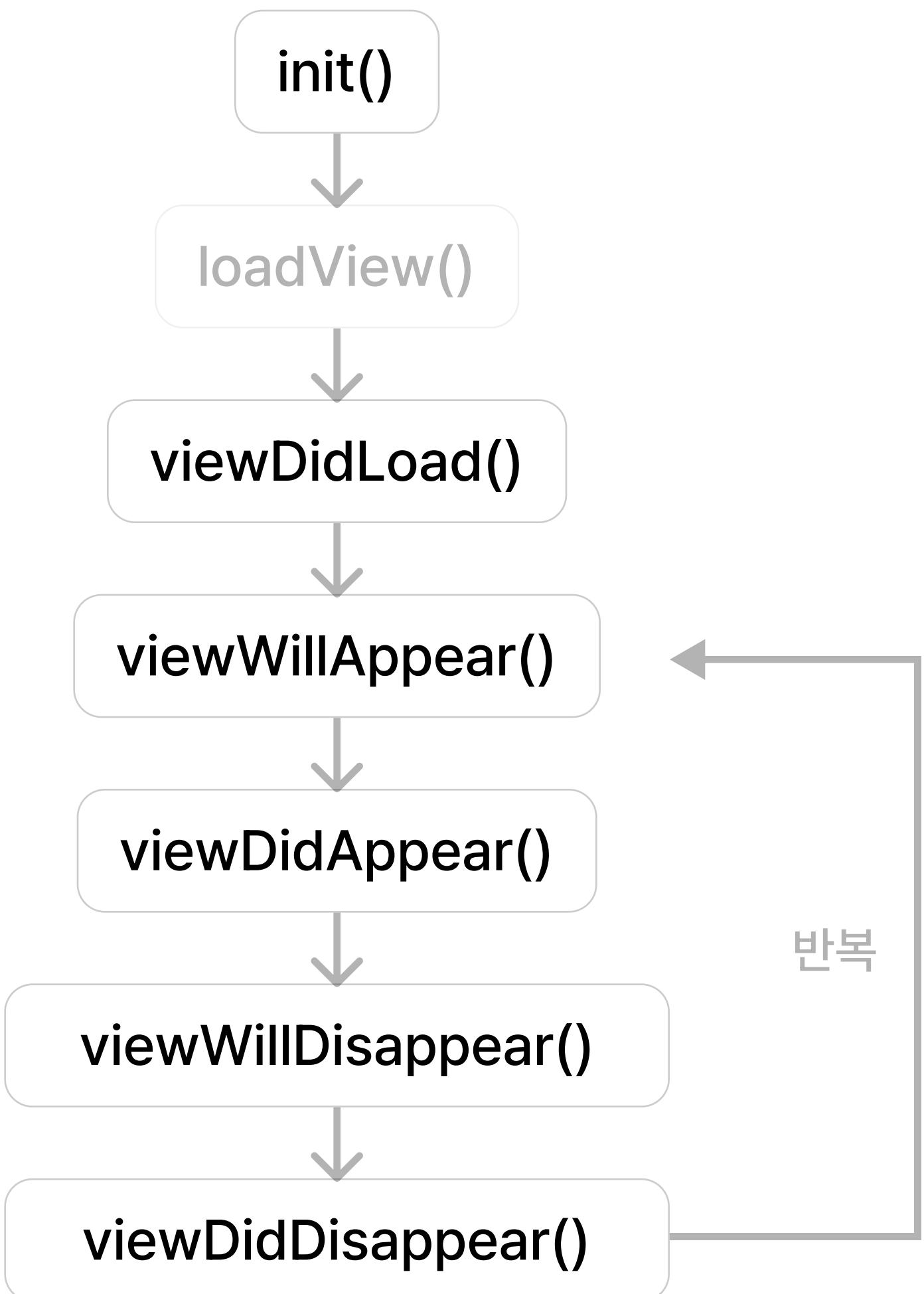
**M** `viewDidAppear(_ animated: Bool)`

Notifies the view controller that its view was added to a view hierarchy.



## UIViewController's Life Cycle

- UIKit에서의 UIViewController
  - 하나의 root view를 관리
    - root view는 하나 이상의 subview로 구성될 수 있음
  - 우리는 viewDidLoad() ~ viewDidDisappear()에 집중
    - viewDidLoad() : view hierarchy가 메모리에 load되었을 때
      - 편의상 딱 한 번만 실행되는 구간이라고 생각해도 된다
      - 어떤 작업을 해야할까?
  - 각 메서드는 자동으로 호출됨
    - 개발자가 할 일: 적당한 메서드를 override하여 원하는 타이밍에 작업 수행

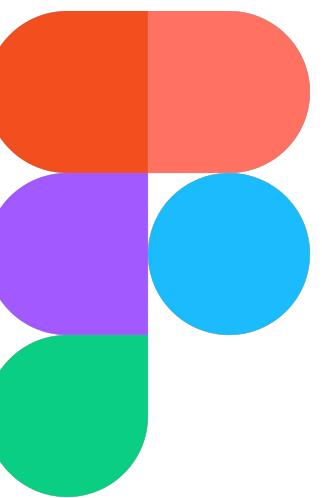




# Figma

# Figma란?

- 대표적인 UI/UX 협업툴 중 하나
- 브라우저 및 데스크톱 앱으로 모두 사용 가능
  - <https://www.figma.com/ko-kr/downloads/>



# 개발자 세미나에서 Figma를 다루는 이유

- 개발자는 혼자 일하지 않습니다
  - 디자이너가 제공하는 스펙에 맞는 UI를 구현
  - QA와 유지보수에 들어가는 비용 줄이기
    - ex. 똑같은 Figma 파일, 서로 다른 안드로이드/iOS 화면

수강신청 빈자리 알림 서비스 창을 들어갔을 때, 수강신청 사이트 버튼이 맨 아래 강좌(6과목 담은 기준) 수강인원 수를 가려서 조금 불편한 것 같아요!

- 세미나의 모든 과제는 Figma 파일을 제공합니다
  - 직접 Layout을 확인하고, 그대로 구현해주세요

빈자리 알림		편집
	지구환경과학부, 3학년	22명/40명
	금 11:00~11:50/금 13:00~16:50	
	500-L311	
지질도학 실험		우주선/3학점
	지구환경과학부, 3학년	15명/40명
	월 09:00~11:50/수 09:00~11:50	
	25-1-406	
대기화학개론 및 실습		박록진/3학점
	지구환경과학부, 4학년	15명/40명
	월 10:00~11:50/수 10:00~11:50	
	500-L310	
대기수치모델링 개론 및 실습		박성수/3학점
	지구환경과학부, 4학년	6명/40명
	화 12:30~14:20/목 12:30~14:20	
	500-L311 / 25-1-418	
대기역학 2		손석우/3학점
	지구환경과학부, 3학년	18명/40명
	화 09:30~10:45/목 09:30~10:45	
	500-L311	
대기물리 2		백종진/3학점
	지구환경과학부, 3학년	25명/40명
	월 12:30~13:45/수 12:30~13:45	
	500-L304	
위성기상기후학		이상무/3학점
	지구환경과학부, 4학년	
	화 15:00~16:50/목 15:00~16:50	
	25-1-418	

수강신청 사이트

# 개발자 세미나에서 Figma를 다루는 이유

- 개발자는 혼자 일하지 않습니다
  - 디자이너가 제공하는 스펙에 맞는 UI를 구현
  - QA와 유지보수에 들어가는 비용 줄이기
    - ex. 똑같은 Figma 파일, 서로 다른 안드로이드/iOS 화면
- 수강신청 빈자리 알림 서비스 창을 들어갔을 때, 수강신청 사이트 버튼이 맨 아래 강좌(6과목 담은 기준) 수강인원 수를 가려서 조금 불편한 것 같아요!
- 세미나의 모든 과제는 Figma 파일을 제공합니다
  - 직접 Layout을 확인하고, 그대로 구현해주세요



# Figma 보는 방법 (view 권한만 있는 경우)

**Black(900)**

**ExtraBold(800)**

**Bold(700)**

**SemiBold(600)**

**Medium(500)**

**Regular(400)**

**Light(300)**

**ExtraLight(200)**

**Thin(100)**

The screenshot shows the Figma Properties panel for a selected text element. The element contains the Korean text "안녕하세요!". The properties listed are:

- Comment**: None
- Properties**: Selected tab
- Export**: Option
- Text**: 안녕하세요!
- Layout**:
  - Width: 329px
  - Height: 29px
  - Top: 450px
  - Left: 32px
  - Border: 1px
- Content**: 안녕하세요!
- Typography**:
  - Font: Apple SD Gothic Neo
  - Weight: 600
  - Size: 24px
  - Line height: 28.8px
- Colors**: Hex #000000
- Borders**: Hex #7FCBEC
  - 1px All sides
  - Outer alignment



Figma

# Figma 보는 방법 (edit 권한이 있는 경우)

- option + 마우스로 확인

The screenshot shows the Figma interface with a mobile component named "Seminar\_Labels". The component has a dark gray background and contains three text labels: "First Label" (black), "Second Label" (orange), and "Third Label" (blue). The Figma properties panel on the right is open, displaying various settings for the selected element.

**Frame:** X: 0, Y: 54, W: 393, H: 764, Rotation: 0°, Transform: Fill, Clip content: Off.

**Auto layout:** Spacing: 24, Gutter: 48, Margin: 48, Horizontal alignment: Left, Vertical alignment: Top.

**Text:** Font: Apple SD Gothic Neo, Weight: SemiBold, Size: 24, Line height: Auto, Text color: #000000, Text transform: None, Text align: Left, Text baseline: Top, Text spacing: 0px, Text direction: Left-to-right, Text orientation: Normal.

**Selection colors:** Primary: #000000, Secondary: #32ACE5, Tertiary: #FF9502.

**Stroke:** Color: #7FCBEC, Width: 1px, Style: Solid.

**Constraints:** Left: 32, Top: 450, Bottom: 29, Right: 29, Angle: 0°.



Assignment 0

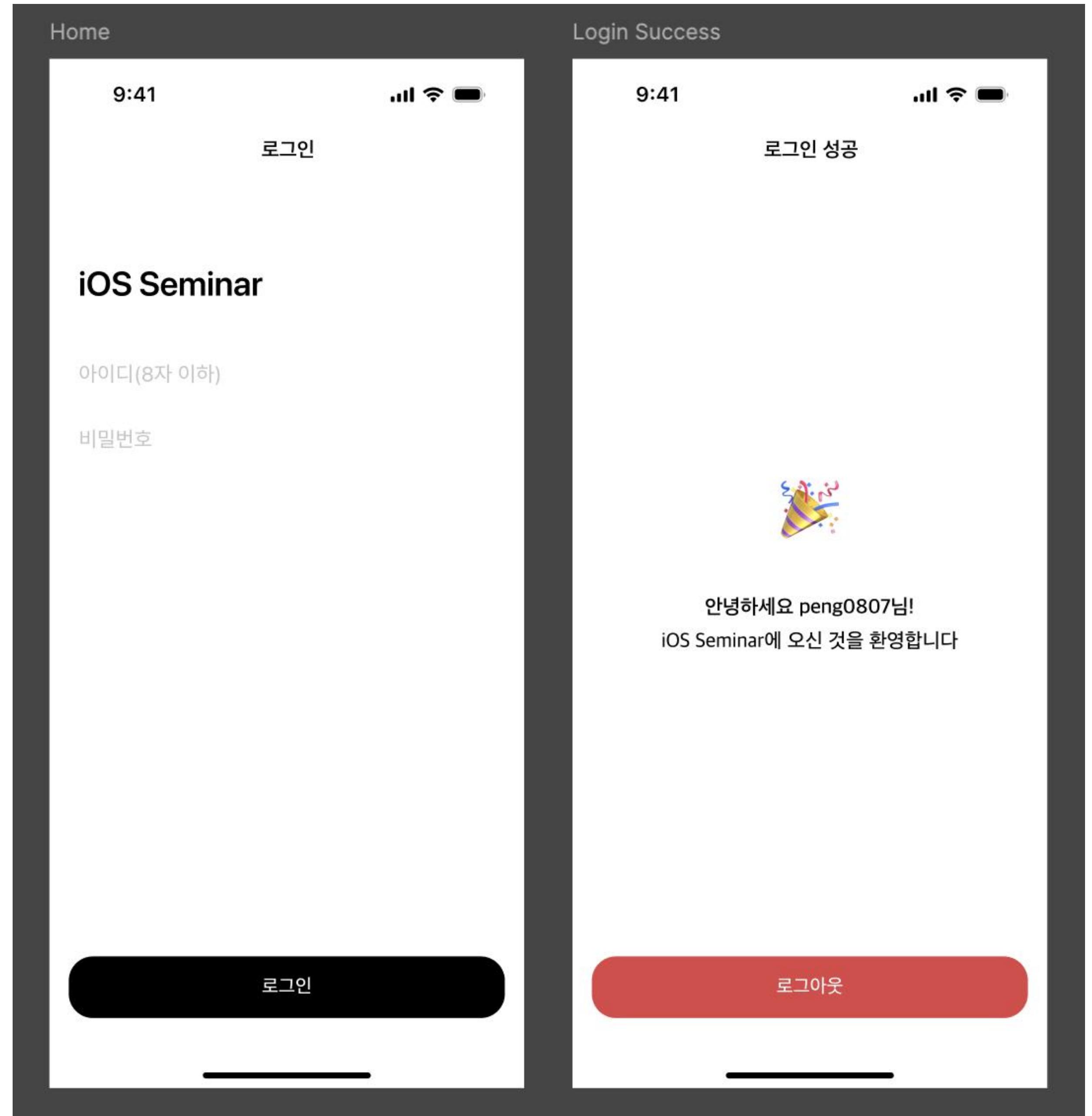
# 과제 0 안내



## Assignment 0

### 간단한 로그인 View 구현

- 과제 0 디자인 링크(Figma)
- 과제 0 스켈레톤 코드 링크(Github)
- 과제 목표
  - No Storyboard
    - Auto Layout
    - UITextField
    - UIButton
    - UIStackView
    - UINavigationController





## 과제 상세 스펙 및 주의사항 (1)

- 전체
  - Storyboard를 사용하지 않습니다 (LaunchScreen.storyboard 제외)
  - 모든 Text, Color, Constraint, Corner Radius 등은 피그마에 주어진 값을 사용합니다
  - Figma에 스펙이 누락된 부분이 있다면, 슬랙에서 질문 부탁드립니다
- 스켈레톤 코드
  - 스켈레톤 코드만으로는 과제를 해결하실 수 없습니다. 적절한 변수와 함수를 추가해 주세요
  - 스켈레톤 코드는 자유롭게 수정 가능하며, 아예 사용하지 않으셔도 괜찮습니다
    - 사용 여부는 과제 채점에 아무런 영향을 미치지 않습니다



## 과제 상세 스펙 및 주의사항 (2)

- Font
  - Figma에 표시되는 폰트명 SF Pro / Apple SD Gothic Neo는 Apple의 시스템 폰트이므로 과제를 진행할 때는 폰트의 size, weight, color만 고려합니다
- UITextField
  - 모든 UITextField의 height는 기본값을 사용합니다 (따로 설정 필요X)
  - 모든 UITextField는 입력하는 동안 오른쪽에 x 버튼이 뜨도록 합니다
  - 비밀번호를 입력하는 UITextField는 입력값이 ●로 마스킹되도록 설정합니다
- UIButton
  - 로그인: id가 조건을 만족하고 password가 비어 있지 않은 경우에만 다음 화면으로 넘어갑니다
  - 로그아웃: 버튼을 눌러 첫 화면으로 되돌아올 때, 모든 상태가 초기화되도록 합니다
    - (아이디 및 비밀번호 UITextField 입력값 등)



## 과제 Bonus 스펙

- 아이디를 입력하는 UITextField에 영어를 입력하면 자동으로 첫 글자가 대문자가 됩니다. 어떻게 해결할 수 있을까요?
- 마지막으로 로그인할 때 사용했던 아이디는 다음으로 앱을 켤 때 자동으로 아이디 UITextField에 채워져 있도록 해봅시다.
  - hint: UserDefaults

## 과제 제출을 위한 세팅

1. 슬랙 "#ios-공지"에 올라온 <https://classroom.github.com/> 링크를 클릭
  - a. 초대를 받으면 자동으로 `ios-assignment-{github name}`라는 이름의 private repository가 만들어짐
2. 과제를 진행할 로컬 위치에 repository를 적절하게 clone
3. 매 과제를 진행할 때는 `main` 브랜치로부터 `assignment0`, `assignment1`, ... 브랜치를 각각 생성하여 작업
4. 완료한 과제는 `main` 브랜치로 `Pull request` 추가하여 제출
  - a. PR 생성 시 반드시 세미나장을 Reviewers로 지정
  - b. 모든 회차의 세미나가 종료되기 전까지는 Merge pull request 누르시면 안됩니다
5. 커밋(commit)은 자잘하게 쪼갤 것을 권장
  - a. "finished assignment 0" << 이렇게 하나로 통치지 말아주세요