

# iOS Seminar 1

Wafflestudio 2024

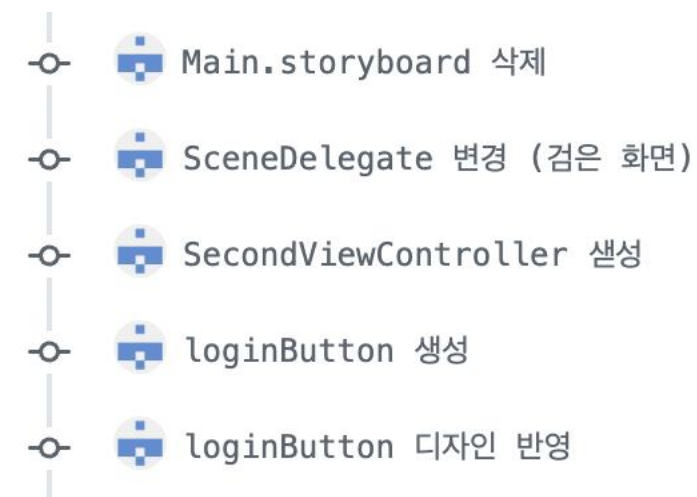




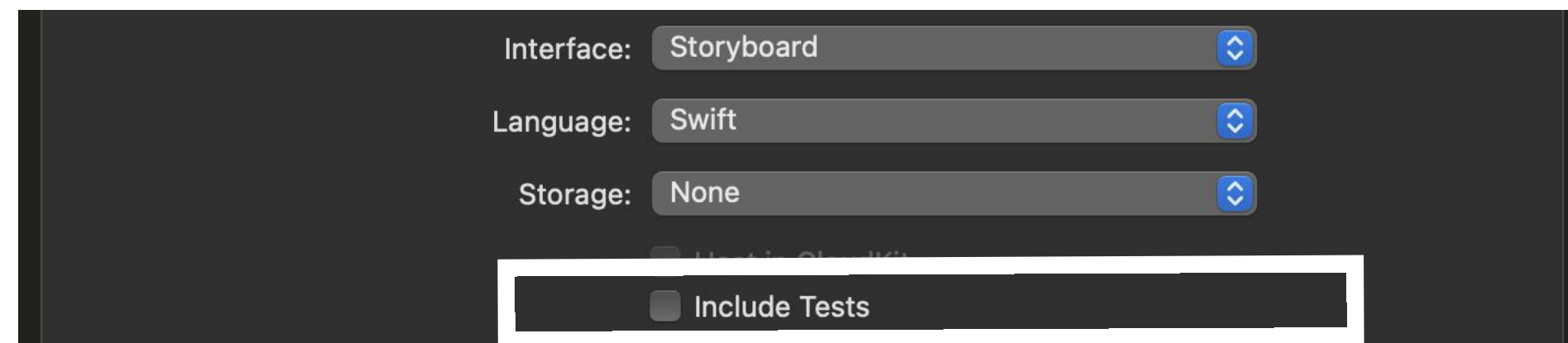
# 과제 0 공통 피드백

## 과제 0 공통 피드백

- AppDelegate, SceneDelegate에 사용하지 않는 메서드는 지우지 말아주세요
- commit 쪼개기



- “Include Tests” 선택 해제



# Delegate Pattern

# Delegate Pattern

- delegate : 위임하다, 대리자
- 기능을 직접적으로 구현 및 수행하지 않고 대리자에게 위임하는 패턴
  - ex) 지난 과제에서 다루려고 했던 UITextFieldDelegate

Protocol






## UITextFieldDelegate

A set of optional methods to manage editing and validating text in a text field object.

- 사용자가 Enter(Return) 키를 눌렀을 때 수행할 action을 지정하고 싶을 때
- 화면에 진입하면 자동으로 TextField에 focus된 상태+키보드를 띄우고 싶을 때

# Delegate Pattern

- Delegate 지정하는 라인을 반드시 넣을 것

**Quick Help**

**delegate**

The text field's delegate.

```
weak var delegate (any UITextFieldDelegate)? get set }
```

**Discussion**

A text field delegate responds to editing-related messages from the text field. You can use the delegate to respond to the text entered by the user and to some special commands, such as when the user taps Return.

**Note**

If the text field is a [UISearchTextField](#), set its delegate to an object that also conforms to the [UISearchTextFieldDelegate](#) protocol.

```
class ViewController: UIViewController {
    private lazy var idTextField: UITextField = {
        let textField = UITextField()
        // TODO: Do something
        return textField
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        idTextField.delegate = self
    }
}

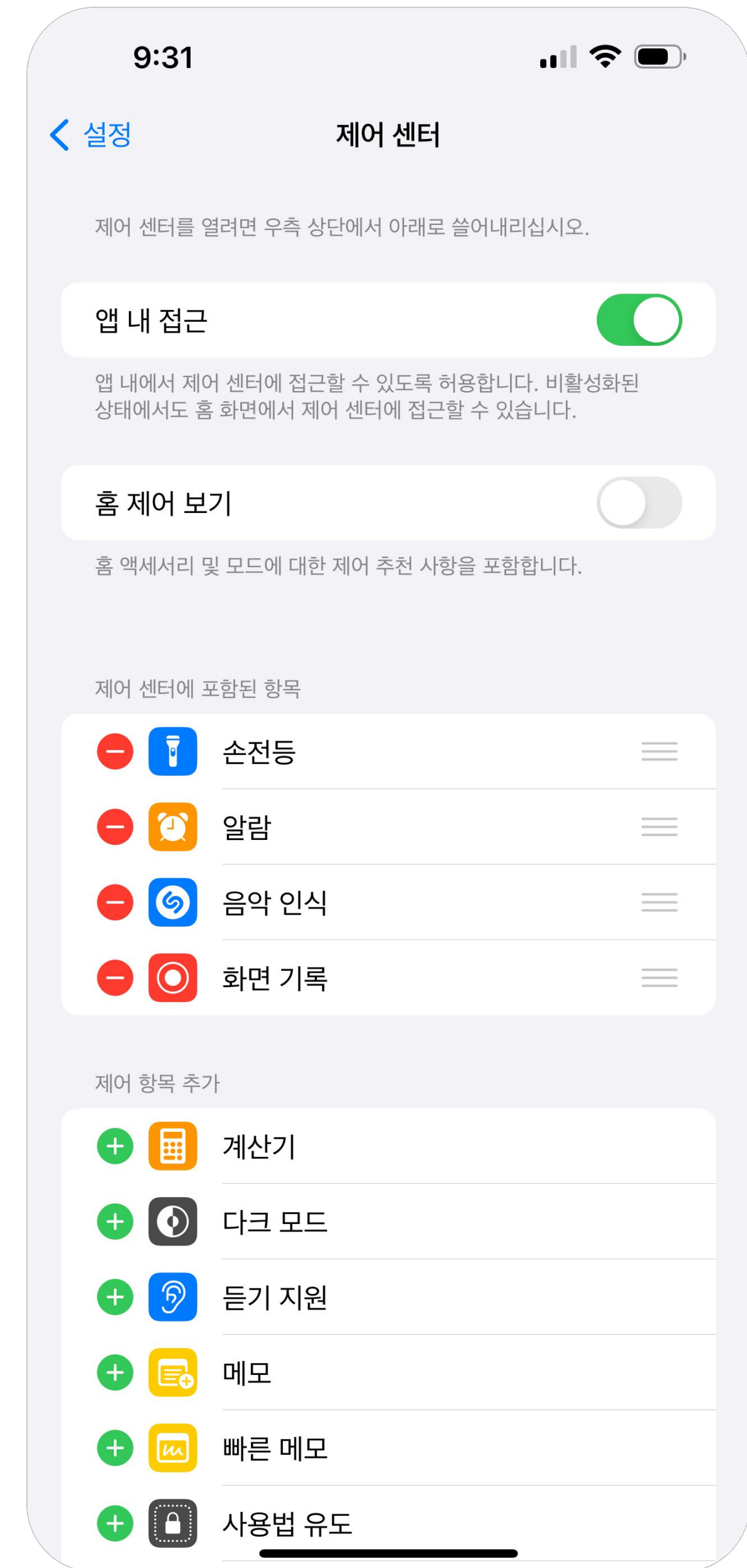
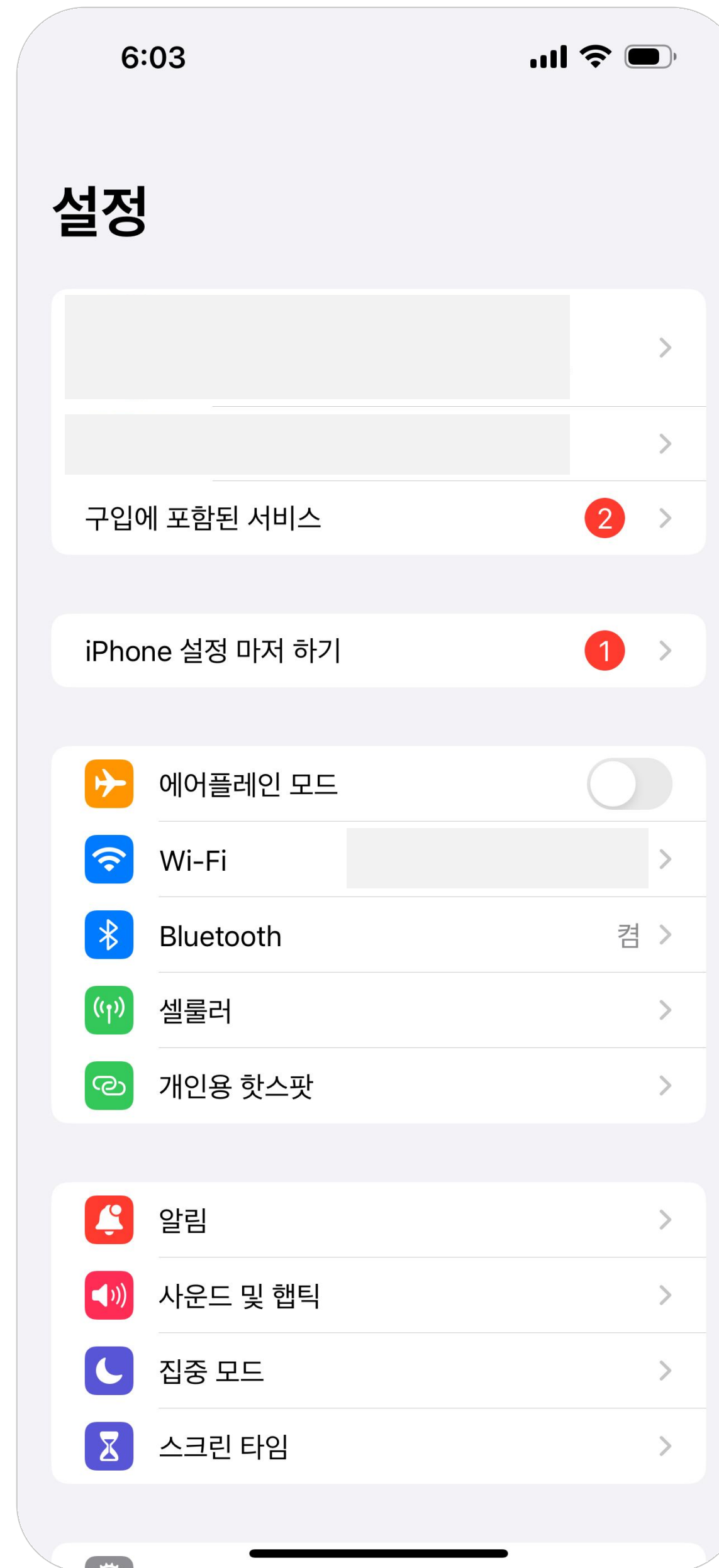
extension ViewController: UITextFieldDelegate {
    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        // TODO: Do something
        return true
    }
}
```

# UITableview



# UITableView (공식문서)

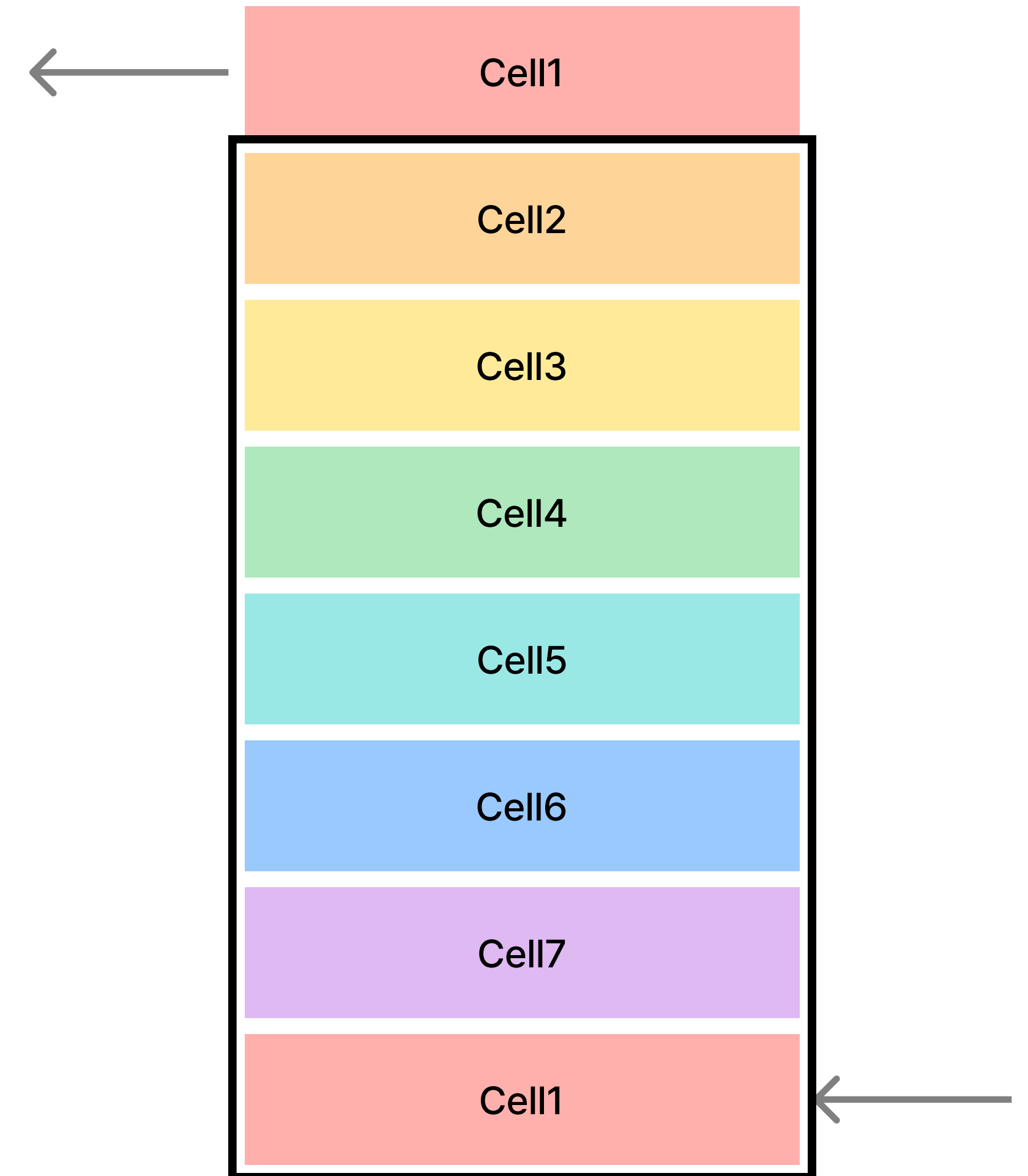
- 하나의 Column으로 상하 스크롤을 이용해 데이터를 보여줄 때 유용한 view
- 아이폰의 "설정" 화면
  - 그 외 카카오톡 친구 목록, SNUTT 검색 결과 등





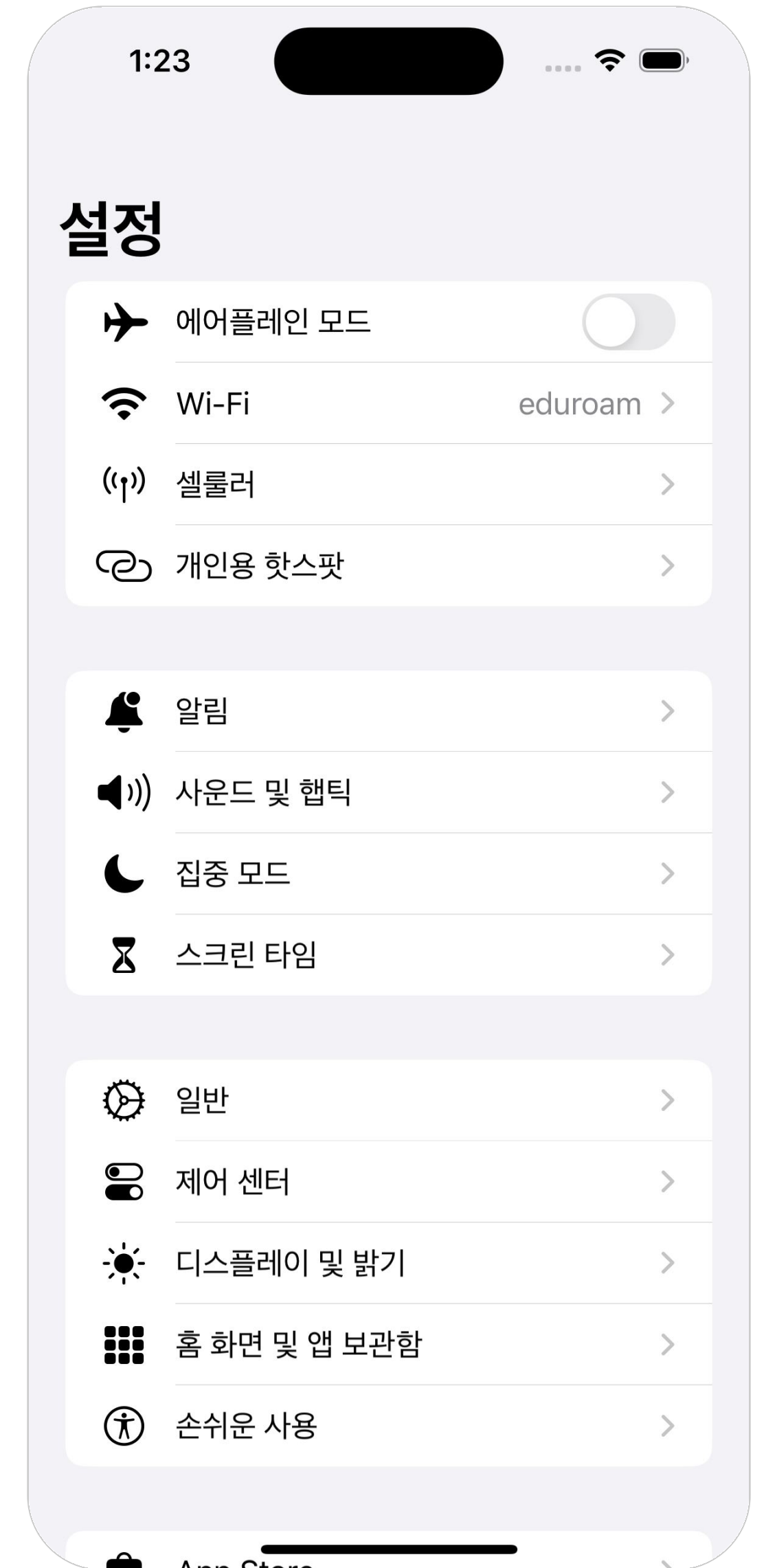
## UITableView에서 Cell을 그리는 방법

- TableView는 화면에서 보이지 않는 cell을 자신의 view hierarchy에서 제거하고, 내부적으로 관리되는 recycling queue에 저장
- 새로 나타나야할 cell에 대해서는 동일한 reuseIdentifier를 사용하는 cell을 dequeue
- Queue-Dequeue 방식
  - 최초에는 init(), 이후에는 prepareForReuse() 호출
  - prepareForReuse() : cell이 재사용될 때를 대비하여, cell의 상태를 원래대로 되돌려놓는 작업 수행
    - 관련 공식문서



## UITableViewCell (공식문서)

- UITableView의 각 row에 들어갈 cell
- 기본 cell에서 디테일을 추가하거나, 아예 새로운 커스텀 cell을 만들 때 모두 사용
- 직접 만들어봅시다🔧



## UITableViewDelegate (공식문서)

- UITableViewDelegate를 통해 할 수 있는 것들
  - 커스텀 header와 footer 생성 및 관리
  - row, header, footer의 높이 지정
  - row가 선택되었을 때, 좌우로 스와이프 되었을 때 등의 상황에서 실행할 action 지정
  - table의 내용 수정
  - 그 외...
- 위와 같은 조작을 위해서는, UITableViewDelegate를 구현하는 object를 UITableView의 delegate로 지정

```
override func viewDidLoad() {  
    // ...  
    customTableView.delegate = self  
}
```

## UITableViewDataSource (공식문서)

- 각 section에 몇 개의 row를 보여줄 것인지
- 각 row에 어떤 data를 담아 어떤 cell을 보여줄 것인지
- 위와 같이 TableView가 보여줄 data와 관련된 조작을 담당하는 프로토콜
  - UITableViewDataSource 프로토콜을 구현하는 object를 TableView의 delegate로 지정해야 한다

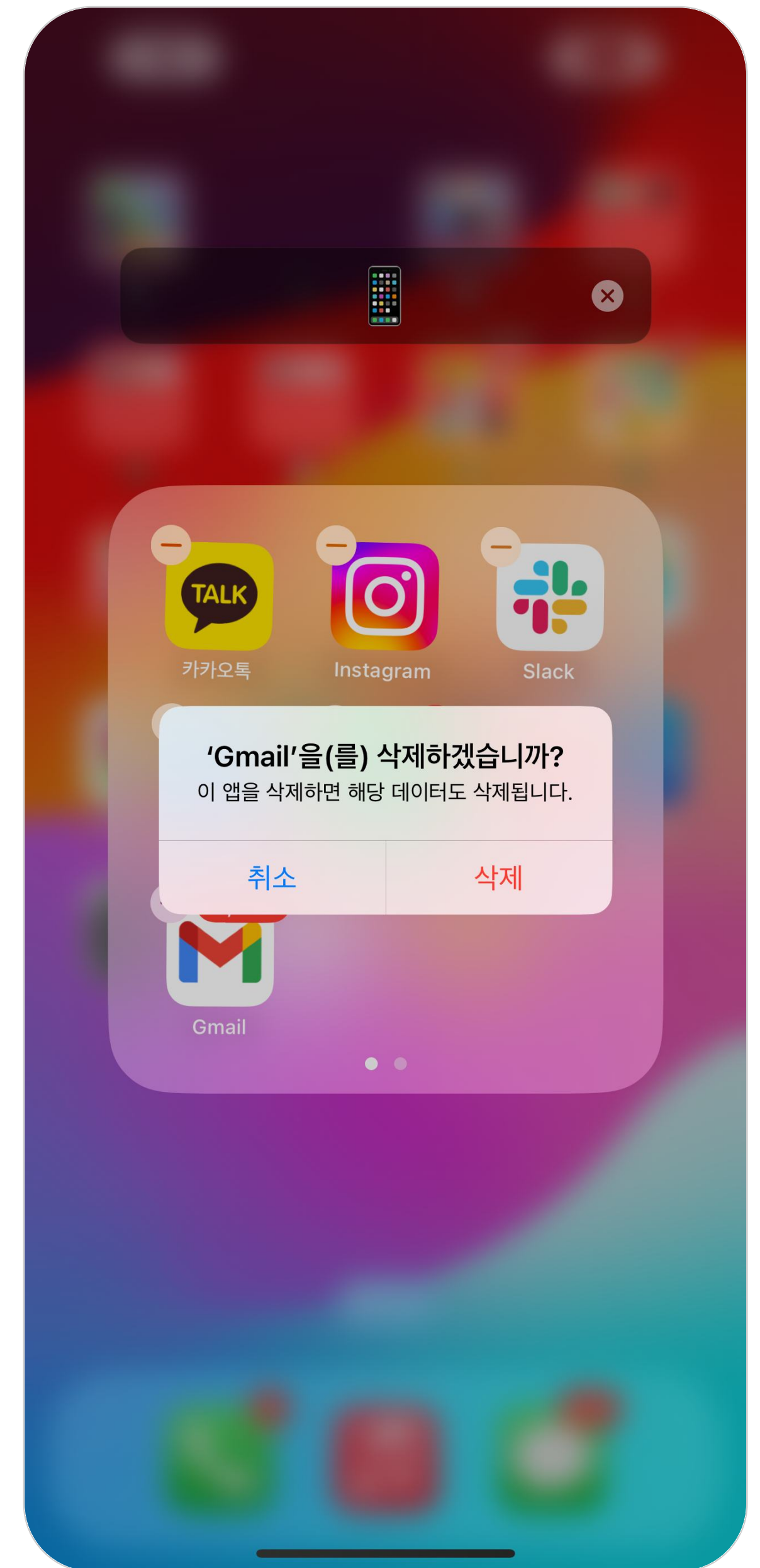
```
// Return the number of rows for the table.
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return 0
}

// Provide a cell object for each row.
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    // Fetch a cell of the appropriate type.
    let cell = tableView.dequeueReusableCell(withIdentifier: "cellTypeIdentifier", for: indexPath)
    return cell
}
```

# UserDefaults

## UserDefaults (공식문서)

- 간단한 캐싱을 위한 클래스
  - key-value pair 이용
  - 앱을 지우기 전까지 유지되는 정보
- 정보를 저장하는 다른 방법들
  - FileManager
  - Core Data

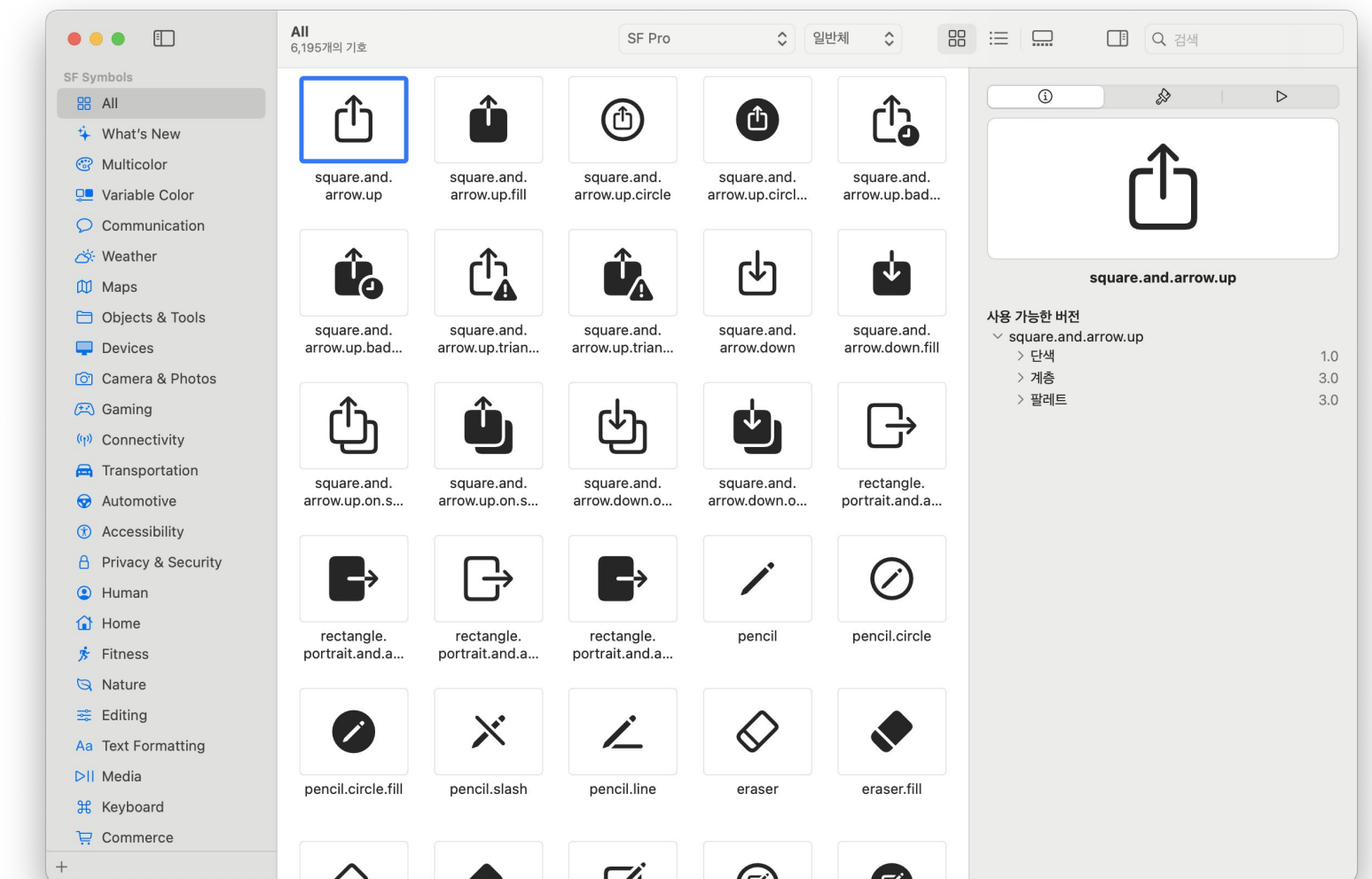




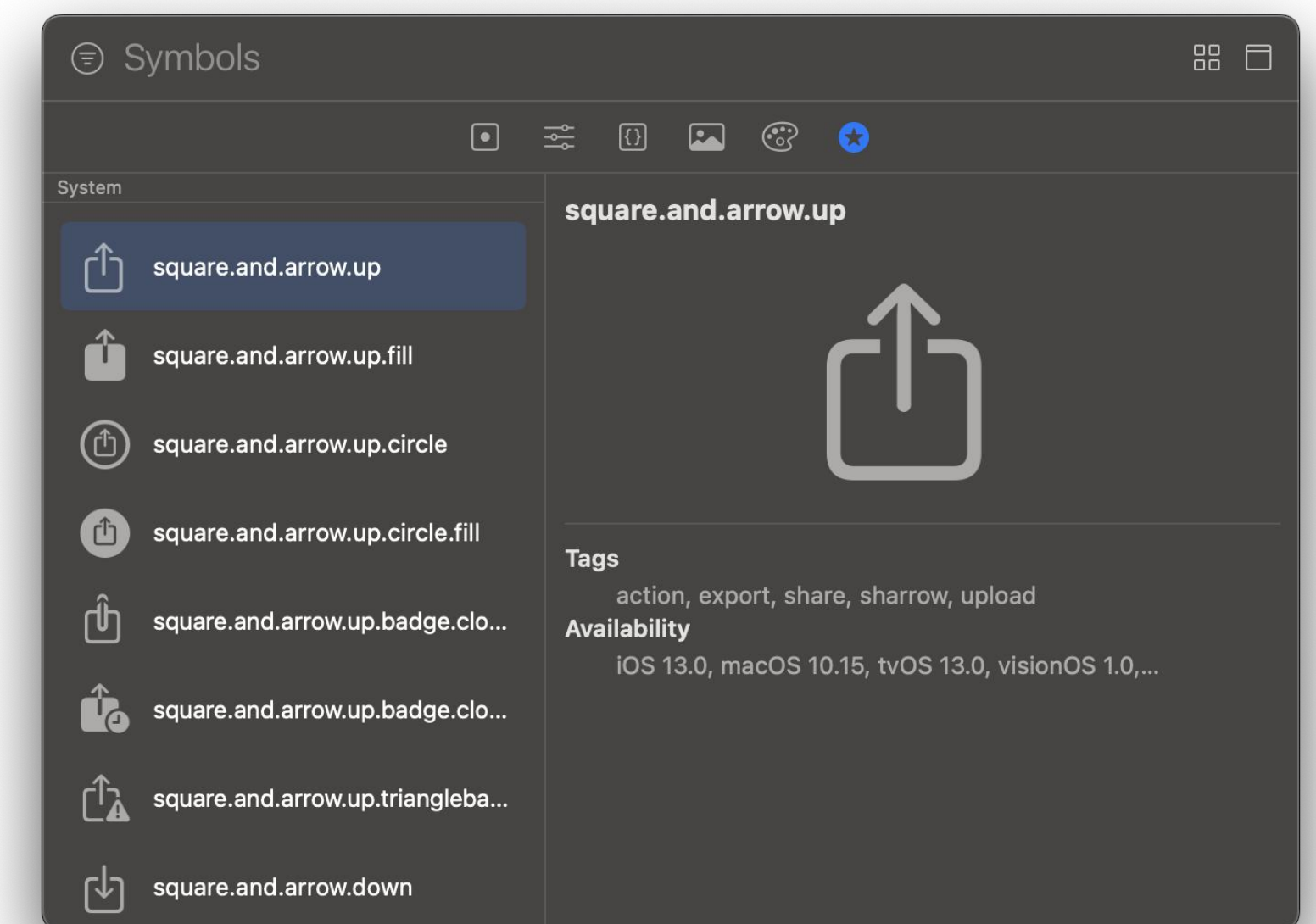
# SF Symbols

## SF Symbols (설치 링크)

- Apple에서 제공하는 디자인 리소스 (아이콘)
  - 디자이너 없이 개발자끼리 있어보이게 앱을 만들 때 매우 유용
- San Francisco 서체(시스템 폰트)와 사용
  - 아이콘이지만 fontSize, weight 설정 가능
- 설치 없이 Xcode에서 바로 이용하는 방법
  - Cmd + Shift + L



SF Symbols



Xcode에서 Cmd + Shift + L

## SF Symbols 실습

- asset을 이용하는 방법과, SF Symbols를 이용하는 방법의 차이

```
// Xcode의 asset catalog에서 이미지를 불러오는 방법 (Seminar 0에서 사용했던 방식)
Image("custom.multiply.circle")

// SF Symbols 이용하는 방법
Image(systemName: "multiply.circle.fill") // ✕
```

```
[
    "airplane",
    "wifi",
    "antenna.radiowaves.left.and.right",
    "personalhotspot"
],
[
    "bell.badge.fill",
    "speaker.wave.3.fill",
    "moon.fill",
    "hourglass"
],
[
    "gear",
    "switch.2",
    "sun.max.fill",
    "square.grid.3x3.fill",
    "accessibility"
],
[
    "bag.fill",
    "creditcard.fill"
]
```

# 과제 1 안내

## 간단한 Todo List 앱 만들기

- 과제 1 디자인 링크(Figma)
- 과제 1 스켈레톤 코드 링크(Github)
- 관련 키워드
  - UITableView + UITableViewCell
  - UITextField
  - UIStackView
  - UINavigationController
  - UINavigationControllerItem
  - UIAlertController

### 과제 1 디자인 (기본 ver)



## 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
- Font
  - SF Pro / Apple SD Gothic Neo는 Apple의 시스템 폰트이므로 과제 진행 시 size, weight, color만 고려합니다
- UIAlertController
  - 별도의 커스텀 디자인이 필요하지 않습니다. 스펙에 맞는 문구만 지정해 주세요



## 과제 상세 스펙 및 주의사항 (2)

- 홈 화면
  - Todo List는 반드시 **UITableView**를 이용하여 구현합니다
  - 각 cell을 누르면 완료 여부를 true / false로 전환할 수 있습니다. 전환 시 달라지는 font 색상에 유의합니다
  - 초기에는 Todo가 없는 빈 TableView를 보여주며, Todo List의 상태는 앱을 껐다 켜도 유지되도록 합니다 (컬러 및 완료 여부)
- Todo 추가 화면
  - “저장” 버튼을 눌렀을 때, Todo의 제목이 비어있는 경우에는 Alert를 띄웁니다
    - Todo의 제목이 비어있지 않다면, 새 Todo를 만들고 이전 화면으로 되돌아갑니다
    - 이때 홈 화면에는 새로 추가된 Todo가 바로 보여야 합니다

## 과제 Bonus 스펙

- 환경설정 화면에서 Color Mode에 대한 Switch를 사용할 수 있도록 합니다
  - 초기 상태는 off이고, 이후 Switch의 on/off 여부는 앱을 껐다 켜도 유지
  - Switch가 on인 경우: 홈 화면의 체크 아이콘 색상이 `.systemOrange`가 됩니다
  - Switch가 off인 경우: 홈 화면의 체크 아이콘 색상이 `.black`이 됩니다
  - hint: UISwitch
- 완료한 Todo의 제목과 설명에 취소선이 그어지도록 합니다
  - 한 번 완료된 Todo가 다시 미완료 상태로 돌아가면 취소선도 사라져야 합니다
  - hint: NSAttributedString

### 과제 1 디자인 (보너스 ver)

