Swift Study 06

2016. 12.03

Swift 문법 & ios

- type casting(형변환)
- double question (??)
- Segue

Type Casting - type check

- is 라는 키워드로 type 확인
- 연산에 대한 return은 Bool (true / false)

형식)

[비교대상] is [비교타입]

=> 비교결과 (true / false)

```
let text = 123

if text is String {
    print("String형입니다.")
} else if text is Int {
    print("Int형입니다.")
} else{
    print("다른 타입입니다.")
}


// 결과는 "Int형입니다."
```

Type Casting - as 키워드

- as라는 키워드로 형변환 (상속 관련 부모-자식 관계타입)
- as뒤에 !, ?으로 옵셔널을 사용가능

형식)

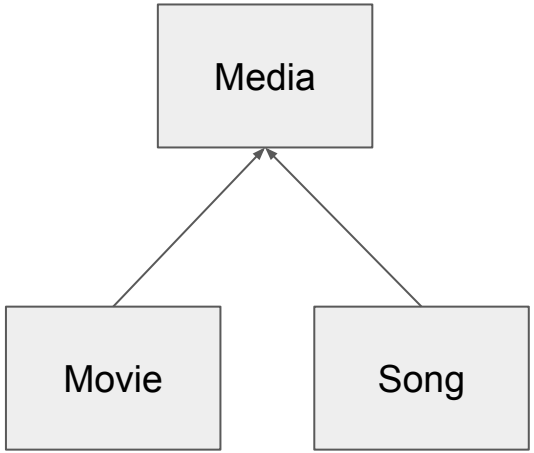
[타입변환 대상] as [변환타입]

as! : 옵셔널 force unwrapping
as?: 옵셔널 타입

```
let text = 123 as Double
// text => Int -> Double

// super: Media, sub: Movie

let movie = media as Movie
// media => Media -> Movie
```

Type Casting - upcasting/downcasting

- as라는 키워드로 형변환 (상속 관련 부모-자식 관계타입)
- upcasting / downcasting 으로 구분



upcasting

```
let media = movie as Media
```

downcasting

```
let song = media as Song
```

Double Question - ??

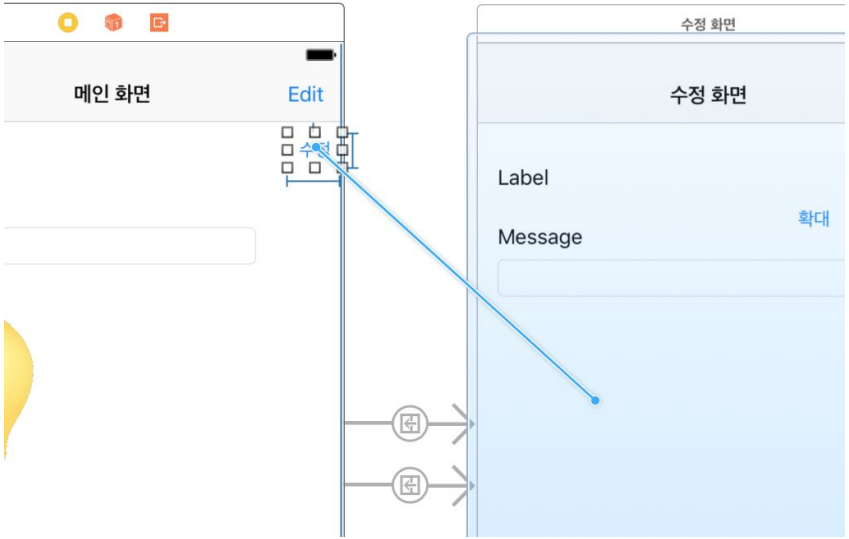
- 옵셔널에 대한 null 처리에 대한 축약 연산자 (3항 연산자의 응용)
- nil 비교 후의 결과값을 (force unwrapping) 하거나 (??뒤의 값)을 적용

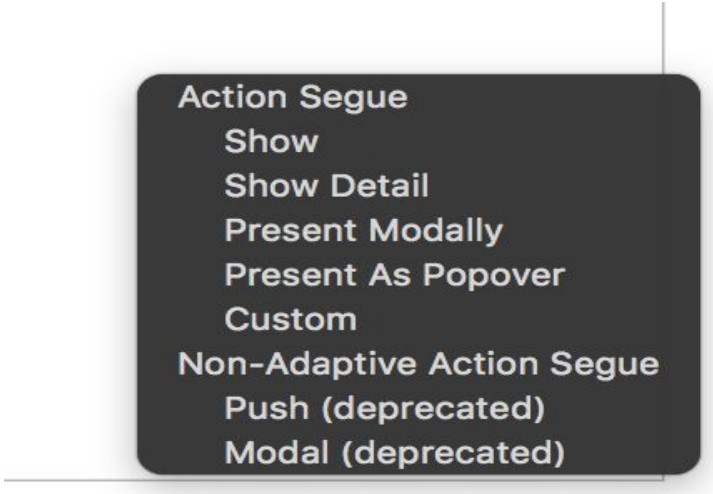
```
let str:String? = "test"
let test = str ?? "empty"

=> str != nil ? str! : “empty”
```

```
let str1:String? = "test"
let test = str1 ?? "empty"
=> “test”

let str1:String? = nil
let test2 = str1 ?? "empty"
=> “empty”
```

Segue

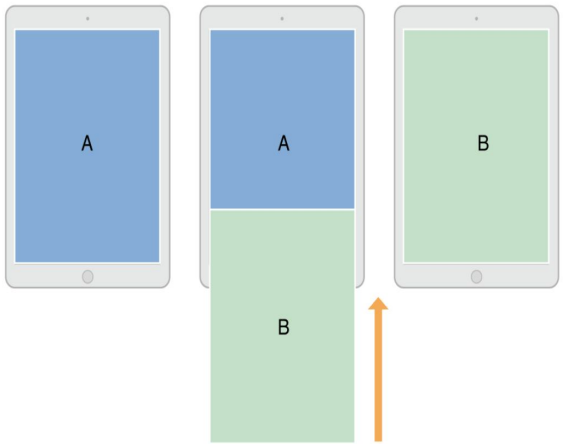
- 화면전환에 대한 식별자 (안드로이드: intent와 유사)
- 화면전환에는 직접 UI에 대한 presentController를 호출하거나 Segue 식별자로 전환방식
- IB(interface builder)에는 segue에 대한 화면전환 action 제공
- UIController 또는 UIControl를 상속받는 대상만 사용가능

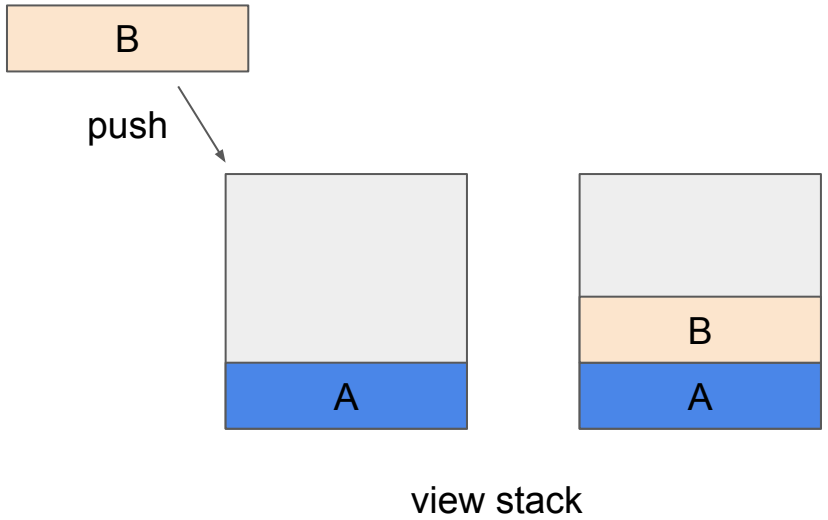




Segue - Show (push)

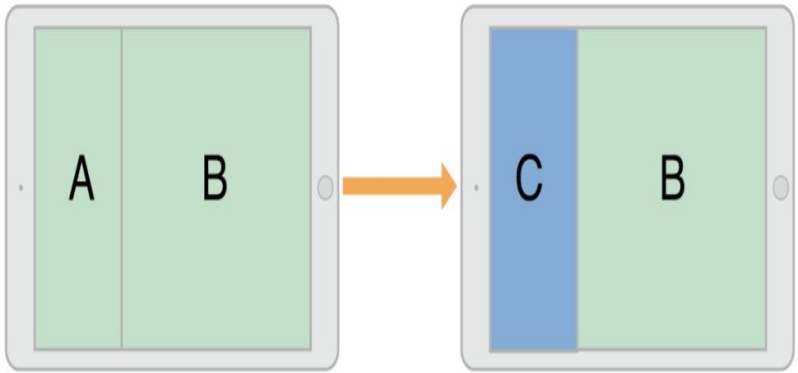
- 전환할 화면을 뷰스택에 쌓으면서 보여주는 화면형태
- 가장 일반적인 화면전환 (ios8부터 push 대신 show 대치됨 / push는 deprecate)





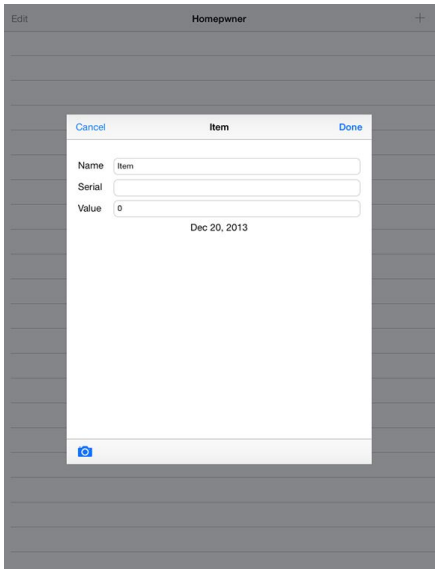
Segue - Show Detail (replace)

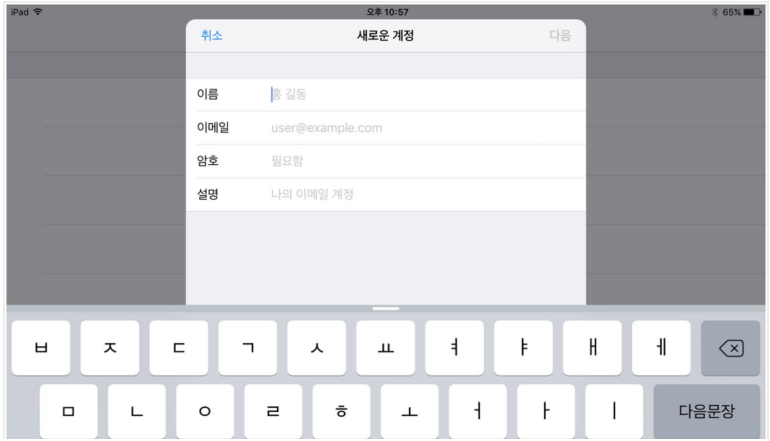
- master와 detail로 나눠지는 화면구성에서 detail 영역을 대치(replace)해서 화면형태
- 태블릿이 지원되는 Universal 앱 경우 show-detail 화면 많이 활용
- view스택에서는 영향이 없음



Segue - Present Modally

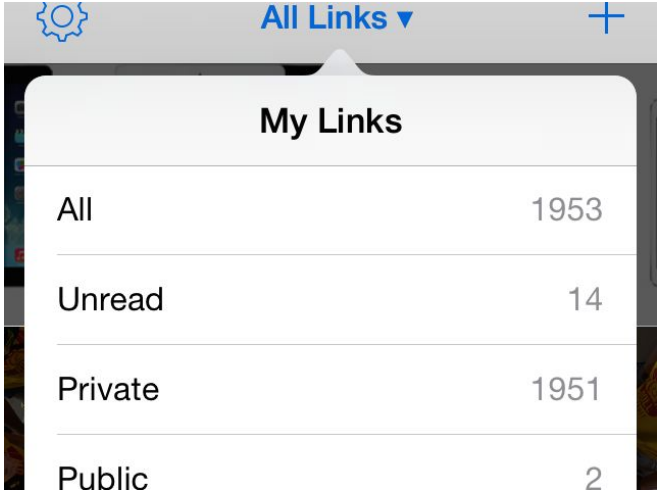
- 기존 화면을 덮으면서 위로 뜨는 화면형태
- ios 8부터 modal 대신 present modally 사용 (modal deprecated)





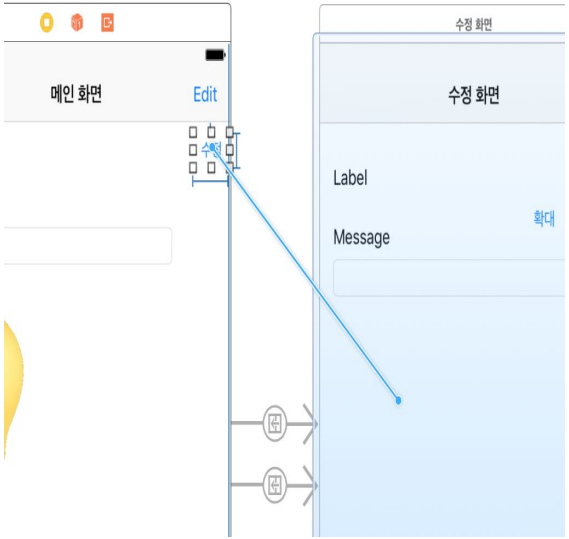
Segue - Present As Popover

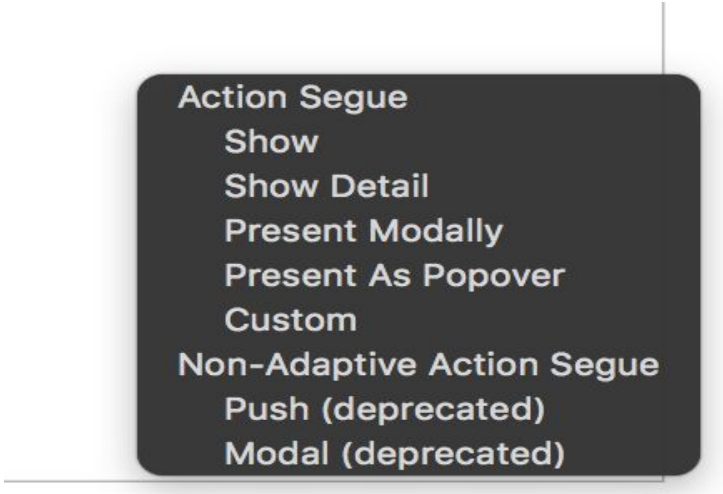
- 작은 팝업형태의 뷰 띄우는 화면형태
- 새로 띄운 뷰의 바깥영역을 터치하면 뷰 사라짐



Segue - 단순 방식

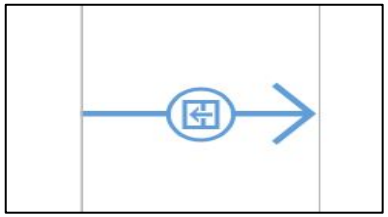
- UIControl를 상속받는 UI의 segue action 추가
- UIViewController 또는 UIControl를 상속받는 UI요소만 segue 설정가능

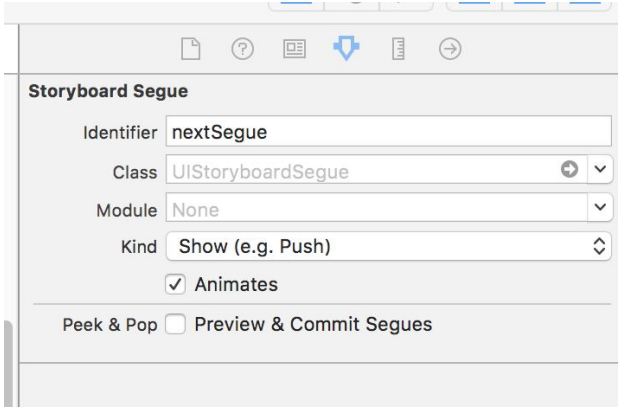




Segue - 제어방법

- segue에 identifier 설정 후 ViewController에서 segue 관련 code정의





func **performSegue**(**withIdentifier** identifier: String, sender: Any?) **//생성된 특정 segue**

func **prepare**(**for segue**: UIStoryboardSegue, sender: Any?) **//segue가 실행전에 초기화 또는 특정기능 설정**

UIViewController 자신.**performSegue**(withIdentifier: "세그 identifier id", sender: 이벤트대상) **//특정이벤트에서 segue 설정**

Segue - 제어방법

```
func prepare(for segue: UIStoryboardSegue, sender: Any?)    //segue가 실행전에 초기화 또는 특정기능 설정
```

ViewController.class

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {

        let editViewController = segue.destination as! EditViewController

        if segue.identifier == "nextSegue" {
            editViewController.textWayValue = "segue : use button"
        }
}
```

Segue - 프로그래밍 방식

- 이벤트 함수에 직접 불러온 ViewController 지정 후 presentController 함수호출

```
@IBAction func segueSend(_ sender: UIButton) {
        let storyboard = UIStoryboard(name: "Main", bundle: nil)
        let view2 = storyboard.instantiateViewController(withIdentifier: "ViewController2") as! ViewController2
        present(view2, animated: true, completion: nil)
}
```