

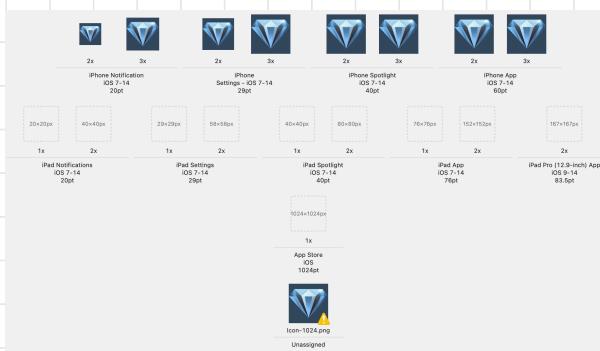
xcode에서 앱 사진을 사용할 때 1x, 2x, 3x가 있다.

배수가 높을 수록 선명한 것이고,

3x는 300 x 300 픽셀이다. 나머지도 배수대로..

제일 높은 스케일의 사진을 사용하면 나머지 이하의 스케일을 사용하는 기종에서도 다 선명하게 보일 것이다.

일러스레이터 사용법을 모르면 app icon generator 사이트에 가서 추출하면 된다.



### 앱 아이콘이 여러개로 있는 이유 속도 때문

- 기종마다 사이즈가 제각각이기 때문에 하나의 큰 아이콘 스케일을 사용하는 것보다 기종에 맞는 아이콘 스케일을 적용하는게 각 기종의 속도면에 도움을 준다.

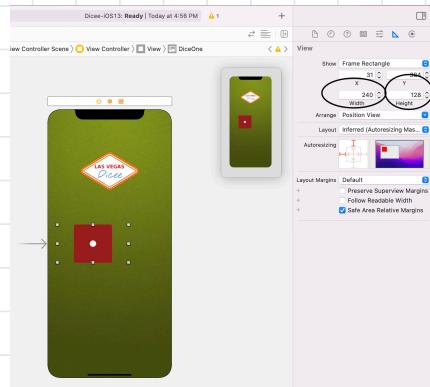
가끔 사이즈를 못찾아 가는 경우도 있는데 (제일 하단의 아이콘) 수동으로 적용시켜 주면 된다.

또한 아이콘 디자인을 쉽게 할 수 있는 canva 사이트에서 디자인을 하고, app icon generator에서 스케일들을 다운받으면된다.

## 배경채우는법

Image view로 화면을 꽉 채운다.

때때로 imageView의 해상도 등에 따라 화면이 꽉 안채워질 경우에는 imageView-View-Content Mode를 scale to fill로 바꾼다. 하지만 이 경우에는 해상도 저하가 올 수 있다. 세 가지 옵션중에 선택하면 된다.



불러온 사진의 사각형 범위가 더 클 때 조정하는 법.

또한 조정 후 저 상태의 사진을 하나 더 생성하고 싶을 때는 옵션키를 누른채로 드래그 해서 복사하면 다시 설정을 할 필요가 없어진다.(모든 세팅값이 그대로 복사된다)

# Who.What = Value

```
@IBAction func askButtonPressed(_ sender: Any) {  
    imageView.image = #imageLiteral(resourceName: "dicee")  
    // 위처럼 이미지 불러오는 코드 :  
    // #imageLiteral(resourceName: "dicee") 까지 치면 사진 선택하는 옵션이 나온다.  
}
```

## • Random

```
Int.random(in: 1...10)
1이상 10이하
```

Random Int

Int.random(in: lower ... upper)

“never mutated”

```
var diceArray = [1, 2, 3, 4, 5, 6]
Variable 'diceArray' was never mutated; consider changing to 'let' constant.
```

• Var이 템플 푸터에 없으면  
기본값은 초기화, diceArray 초기값은 숨겨져  
있으면 let을 사용해 재활용 가능하다.

## - Array에서 Random

```
let diceArray = [1, 2, 3, 4, 5, 6]
```

```
diceImageview1.image = diceArray.randomElement()
```

```
diceImageview2.image = diceArray[Int.random(in: 0...5)]
```

• Int.random은 4가지 가능하지만, diceArray 사용할 때  
Array의 randomElement()은 가능성이 5개다.

## - Random(2)

Random Int

Random Bool

Int.random(in: lower ..< upper)

Bool.random()

Upper 데문트

Random Element from Array

Randomise Array

array.randomElement()

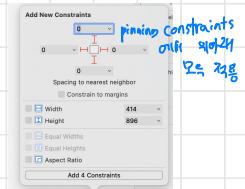
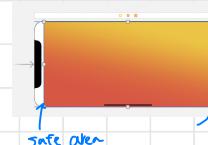
array.shuffle()

## - Code exercise - Randomisation

```
import UIKit
let optional: String? = "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"
var password = ""
for _ in 0...5 {
    password += alphabet.randomElement() ?? "none"
}
print(password)
```

default value

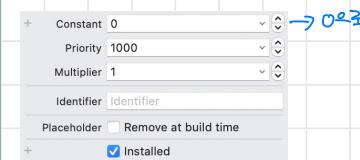
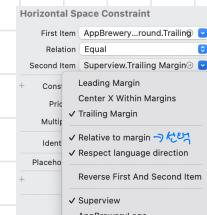
세로 가로 화면 상관 없이 빙글 없애기



## 자유로운 모습

```
AppBreweryBackground.top = top
AppBreweryBackground.leading = Safe Area.leading
AppBreweryBackground.bottom = AppBreweryBackground.bottom
AppBreweryBackground.trailing = Safe Area.trailing
```

## Margin



## 로그 가운데 배치하기

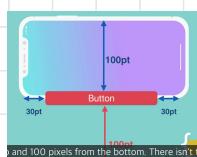
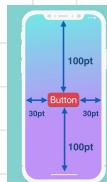
<https://donggooleosori.github.io/2020/12/06/ios-constraint/>



↓  
pinning Constraints를 사용하게 되면,  
지금 보이는 화면 그대로 예상해  
화면을 돌리면 위치는 가운데로 돌아온다.



Alignment Constraints 사용시  
화면의 x,y축을 설정하는 그대로  
모든 화면에 적용되어 줄어들어.

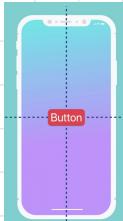


→ pinning Constraints

Pinning Constraint는 거리를 고정하는 방식입니다.

예를 들어 이미지부에 원쪽면을 기준으로 50px의 Pinning Constraint를 설정한다면 세로뷰, 기로부, 디바이스 기종 등에 상관없이 항상 원쪽면에 50px 만큼 떨어져 있게 됩니다.

Alignment Constraint는 몇 그대로 정렬을 사용합니다. 예를 들어 버튼에 Horizontal center로 alignment constraint를 설정한다면 버튼은 어떤 상황에서도 항상 화면 원쪽면과 오른쪽의 중앙에 위치하게 됩니다. vertical center를 설정한다면 항상 화면 위면과 아래면의 중앙에 위치하게 되겠죠.



→ Alignment Constraints

<https://donggooolsori.github.io/2020/12/06/ios-align/>

로고 하단 일정 간격으로 label 배치하기



↑ 다만 설정화면 좌변을 들었을 때  
가장 아래에 위치후면 서로 회전되는  
Constraints로 적용됨. 몰라 확장 봐도  
나가기 때문에 앱 구조와 일정을 먼저 [30:30]  
이 대로 pinning Constraints를 설정해줄까.

# Calculator - auto layout

Goal →



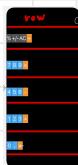
1.



← 처음 모습

일단 주사위화는 단순히 Embedded View를 하지도 않았다.  
이유는 잘 모르겠다.. View에 대해 정확히 더 필요하다.  
어쨌든 별별 부분을 stack view로 만들어 주었다.

2.



→ stack view 적용

모습이 좀 이상하지만 더 활용 시킬 것 같아  
смотрела.

'0'을 Stack View 하지 않은 이유는  
김부겸과 하지면 Display 블록에 그려지겠지 농축된다.

창고로 이 Stack View는 같은 row(가로)의 숫자가 기호로의 Stack View다.  
이제 '0'을 포함한 각 row를 같은 Stack View로 정렬 시킨다.

3.



→ 적용 시킨 모습

하단 세로는 조절 필요 →

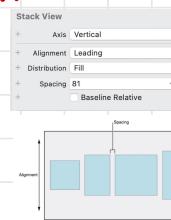
Add New Constraints  
0 -> 0  
Space to nearest neighbor  
Constrain to margins



→ pinning constraints 적용

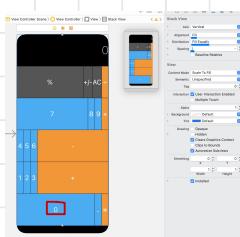
문제점  
1. row stack 간 크기  
2. 숫자, 기호 크기

5.



oi Alignment & spacing  
Alignment: leading 아니 원쪽으로 최우선하고  
Distribution: fill 아니  
Alignment: leading 아니면 정렬은 확장  
Distribution: fill 아니면 row stack 은 view 안에는  
차지하지 않고  
Spacing이 81 아니 raw stack 들의  
간격이 넓은 것이다.  
\* 일정 간격에 따른 설명은 각각입니다.  
공부 필요

6.



Alignment은 fill로 해줄수도  
Stack View 빙대 방향으로 경계를 대로 하지 않고 전체를 채움.  
Distribution은 Fill Equally로 했을 때 row stack의 크기는  
연평균이 됨.

Spacing → 1: 간격 증폭

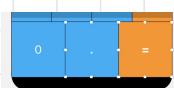
이제, 각 요소들의 크기를 일정하게 해야 한다.  
(맨 앞 0은 제외; 2칸 차지)

7.



↳ 아직 하단 부분 조정은 필요.

8.



또 다른 Stack View 생략해 피로하다.

,=은單 Stack View로 만드는 이유?  
↳ 목표 사진처럼 0이 제일 커야되고 (2칸 차지)

나머지는 위 모든 표소들이 크기가 동일하다.  
그럼 마지막 row stack 만에 2개의 Stack으로 나누기  
그즉 허리는 2칸은 차지하고 (0), 나머지 하나는 또 두개의 표소로  
나누니 그대로 크기 조작을 하면 된다.

↳ 최상의 Stack 안에 두개의 Stack으로 나누어 2칸씩 쓰기 훈련,  
그나 있는 Stack은 Fill Equally로 통해 다시 2로 나눈다

9.



먼저 0 삼자 원쪽으로  
움직이.

→ stack view 적용



→ fill equally 적용

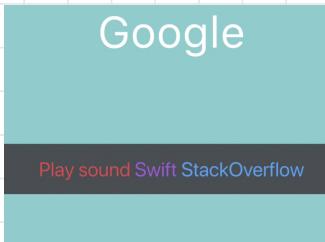


간단히 0 뒤에 온 걸고 올리는 것  
이다. Stack View를 하면 하게 많잖아  
여튼 기울거나 방향으로 압박해  
전부 차지할 때까지  
View를 4등분하여 비율을 잘라주고  
그 View들은 두개로  
pinning constraints를 통해 View의 상용하는  
Constraints를 찾았어 데 반복적으니 양쪽에  
입장 간격은 더해줄 것이다.

# How to get an information from the Internet



Google



## Stack overflow & implement (7월)

```

import UIKit
import AVFoundation

class ViewController: UIViewController {
    var player: AVAudioPlayer?

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

    @IBAction func keyPressed(_ sender: UIButton) {
        playSound()
    }

    func playSound() {
        guard let url = Bundle.main.url(forResource: "C", withExtension: "wav") else { return }
        do {
            try AVAudioSession.sharedInstance().setCategory(.playback, mode: .default)
            try AVAudioSession.sharedInstance().setActive(true)
        } catch {
            print(error.localizedDescription)
        }

        player = try AVAudioPlayer(contentsOf: url, fileTypeHint: AVFileType.mp3.rawValue)

        guard let player = player else { return }

        player.play()
    }
}
  
```

Copied codes ↗

### 2. Instance Method

#### setCategory(\_:)

Sets the audio session's category.

AVAudioSession에서 예상 대체로 보면 Session에서 사용할 수 있는 모드들이 Topics or 게시판에 있다.  
그중 setCategory는 찾았어 글록 (Shared Internet!)는 Session 접근 방식에 나와있음

### 4.

#### Topics

#### Audio Session Categories

```

static let ambient: AVAudioSession.Category
    The category for an app in which sound playback is temporary—that is, your app also
    works with the sound track.

static let multiDevice: AVAudioSession.Category
    The category for making distinct streams of audio data to different output devices at the
    same time.

static let playback: AVAudioSession.Category
    The category for recording input and playback output of audio, such as for a Voice
    Memos app.

static let record: AVAudioSession.Category
    The category for playing recorded music or other sources that are central to the
    successful use of your app.
  
```

이제 우리가 찾던 playback이 나왔다.

Stackoverflow에서 초록색으로

제작자가 된 답변이나 해석 항목 같은 것은 없을 것이다.  
사실은 애초 버전 자체가 있기 때문에 일의 규모 잘  
나타내 보여 준다

## Stack overflow & implement (7월)

## Docs (Apple Developer)

Documentation > AVAudio > AVAudioSession

Class

### AVAudioSession

An object that communicates to the system how you intend to use audio in your app.

Apple Developer

검색

3.

#### func setCategory(\_ category: AVAudioSession.Category) throws

보통은 AVAudioSession.Category에 있는 요소를 통해 선택해  
적이면 되는 것 같다. 클릭.

- 또한 option 키를 누른 채로 코드 위에  
갖다 대입 전부는 정의된 볼 수 있다

- Apple Developer Docs를 적극 활용하면  
간 코드들의 기능과 대체로 알기 쉽고, 전체적인  
흐름을 보는데에 도움을 준다

# Linking Multiple Buttons to the Same IBAction

1.



일단 C 코드는 IBAction으로 연결 했다.  
다른 코드들을 이어서 방식으로 반복하는 것은  
매우 비효율적이다. 그걸 어떻게 할까?

2.



3.

```
@IBAction func keyPressed(_ sender: UIButton) {
    playSound()
}
```

```
① @IBAction func keyPressed(_ sender: UIButton) {
    21     print(sender)
    22     playSound()
    23 }
    24
    25 func playSound() {
    26     let url = Bundle.main.url(forResource: "C", withExtension: "mp3")
    27     player = try! AVAudioPlayer(contentsOf: url!)
    28     player.play()
    29 }
    30
    31 }
    32 }
    33
    34
    35 }
```

<UIButton: 0x7f80e9250700; frame = (25 6; 364 189); opaque = NO>
<CALayer: 0x6000025a0790>>

4.

```
① @IBAction func keyPressed(_ sender: UIButton) {
    21     print(sender.backgroundColor) // Expression implicitly coerced from 'Any'
    22     playSound()
    23 }
    24 }
```

## 5. challenge

이번엔 Button의 title을 구하는 것에 있다.  
처음엔 경지로 안하고 혼자서 해볼 것의 결과다.

```
print(sender.titleLabel) ↘
```

Optional<UILabel>: 0x7fd2ba09f830; frame = (188 31; 28 48); text = "C"; alpha = 1  
Button의 title이 경지로 나와는 빛지 않아,  
필요없는 다른 정보들 솔직히 나와서 이것을 통해 저대로  
Button의 title을 추적하여 그를 없애는 효과적임을 알겠지.



```
@IBAction func keyPressed(_ sender: UIButton) {
```

```
    if let buttonTitle = sender.title(for: .normal) {
        print(buttonTitle)
        playSound()
    }
}
```

다음은 stack overflow이다. 경색한 결과이다.

if let을 optional Binding의 한 종류이다.  
Sender 즉, button의 title이 nil 값일 수 있고 있으니  
사용하는 걸로 생각된다. (혹은 공부 필요)



C button의 title을 없애고  
nil 역시 "nil"을 print하게 됐어야  
실제로 nil이 print 되었지.  
그럼 title 앞 (for: .normal)은 무었인가?



UIControl.State  
General  
Description  
Title  
Title Label  
Insert Subviews  
Insert Subviews  
General  
Title  
Title Label  
Insert Subviews  
Insert Subviews  
print(sender.title(for: UIControl.State)) → 코드의 원형은 뷰 Control(UIKit.Button)의  
State(상태)를 표시하고,

```
print(sender.title(for: .normal)) ↗  
아마 전부 없으니 nil이 아니 같은가  
생각된다. (혹은 그냥 .normal 인자는 모양이다)
```

## 그 외 방법들

```
print(sender.currentTitle)  
print(sender.titleLabel?.text)
```

```

class ViewController: UIViewController {
    var player: AVAudioPlayer!
    override func viewDidLoad() {
        super.viewDidLoad()
    }
    @IBAction func keyPressed(_ sender: UIButton) {
        if let buttonTitle = sender.title(for: .normal) {
            print(buttonTitle)
        } else {
            print("nil")
        }
        print(sender.currentTitle)
        sender.alpha = 0.5 → Sender 즉, 버튼의 opacity(투명도)를 50%로
        playSound(soundName: sender.currentTitle ?? "C")①
        // ! : Don't worry we've already checked
        // playSound(soundName: sender.currentTitle!)
        sender.alpha = 1.0
    }
}

func playSound(soundName: String) {
    let url = Bundle.main.urlForResource: soundName, withExtension: "wav")
    player = try! AVAudioPlayer(contentsOf: url!)
    player.play()
}
}

```

```

//var aYear = Int(readline()!)!
var aYear = 1997

func isLeap(year: Int) {
    let result = (year % 400 == 0) || (year % 4 == 0 && year % 100 != 0) ? "YES" : "NO"
}
    Yes      No

```

```

// var player1Username: String = nil
var player1Username: String? = nil
player1Username = "jackbauerisawesome"
// safety check 깊이 있다고 한다
var unwrappedPUsername = player1Username

playerUsername = nil //> crash
print(player1Username!) // safety check를 해 깊이 unwrapping을 하였는데 깊이 있어서 crash
// how to prevent this happening?
if playerUsername != nil{
    print(player1Username!)
} else {
    print("nil")
}

```

☞ 깊이에서 접근은 가능 X

① sender.currentTitle nil일 때 기본값은 "C"이다.

② stackoverflow에서 가져온 것인데 Delay를 주는 코드야.

\*로 키워 둘 때 있는 코드들은 실행시켜 주는 것이다.  
Sender.Alpha를 기본값으로 넘기 때문에 대신 유저 터치로  
드러워 Button이 누리는 효과를 극대화 한다.

# Egg Timer

```
let hardness = sender.currentTitle!
```

```
print(eggTimes[hardness]) △ Express
```

sender.currentTitle 뒤에 ! (Force-unwrapping)을 쓰지 않으면 오류가 난다. sender.currentTitle이 nil일 수 도 있기 때문인 것 같다. 하지만 이 코드는 내가 직접 작성했고 currentTitle에 분명히 데이터가 들어올 것을 알고있기 때문에 !를 써주었다.

```
switch hardness{
    case "Soft" :
        print(eggTimes["Soft"]!)
    case "Medium" :
        print(eggTimes["Medium"]!)
    case "Hard" :
        print(eggTimes["Hard"]!)
    default :
        print("Wrong access")
}
```

이건 내가 작성한 코드인데, eggTimes dictionary에서 sender.currentTitle로 들어온 값으로 value를 추출해 print를 하는 것이다. 하지만 내가 너무 어렵게 생각했나보다.  
위에 보이는 것처럼 어차피 currentTitle은 eggTimes의 value를 이름으로 들어오기 때문에 바로 eggTimes에 key값을 넣어줘 value를 추출하면 된다.

dictionary에서 optional이 나오는 이유?

-> dictionary has keys which are of string data types.

Let's say that we try to retrieve 5, the value 5 out of our eggTime dictionary.

I could simply provide the key "soft". (eggTimes["Soft"])

Dictionary는 string data type으로 키 key를 가지고 있다.

만약 우리가 위의 dictionary에서 5(Soft)를 추출하고 싶다고 가정해보자. 하지만 "Soft"가 아닌 "soft"로 코드를 작성하게 되면 (eggTimes["soft"]) "soft"라는 key가 없기 때문에 코드를 실행하면 nil이 나온다.

Summary  
No overview available.  
Declaration  
let result: Int?  
Declared In  
ViewController.swift

19 let result = eggTimes[hardness]

Declaration  
let result: Int  
Declared In  
ViewController.swift

18 19 let result = eggTimes[hardness]!

있어도 괜찮았는  
경우에는 사용 (forced-unwrapping)

```
let hardness = sender.currentTitle!
```

```
let result = eggTimes[hardness]!
```

```
print(result)
```

```
print(eggTimes[hardness]) △ Expressio
```

5 eggTimes[hardness]!

Optional(5) eggTimes[hardness]

7

Optional(7)

12

Optional(12)

## Timer

```
class ViewController: UIViewController {

    let eggTimes = ["Soft": 300, "Medium": 420, "Hard": 720]

    var result = 60
    var timer = Timer()

    @objc func updateTimer() {
        if result > 0 {
            print("\(result) seconds.")
            result -= 1
        } else {
            timer.invalidate()
        }
    }

    @IBAction func hardnessSelected(_ sender: UIButton) {
        // ! : 반드시 확신(forced-unwrapping)
        let hardness = sender.currentTitle!

        result = eggTimes[hardness]!

        print("\(hardness) got selected")

        timer.invalidate()
        timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector:
            #selector(updateTimer), userInfo: nil, repeats: true)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

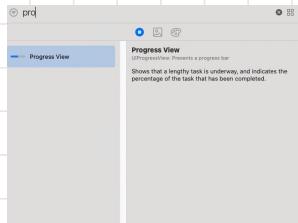
timeInterval - 몇초간격?  
Repeats - true를 해야 timeInterval초 뒤에도 계속 된다.  
Selector - object의 인터페이스 (안에 넣은 function)을 타이머가 불러질때마다 실행된다.

@objc - #selectorobjc의 언어이므로 ()안에 넣은 function을 생성할 때도 @objc를 붙여서 생성해준다.

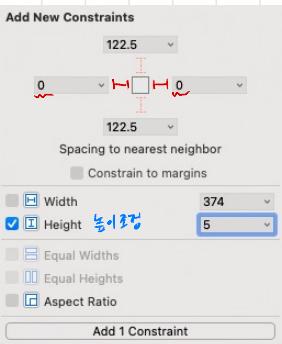
timer.invalidate() -> 타이머 초기화 : 매 버튼이 눌릴때마다 초기화를 하지 않는다면 다른 버튼을 누를때 처음 시작 초기화를 하기 때문에 2배 3배로 초기 줄어든다.

Timer는 잘 작동한다.

하지만 최종 목표인 progressbar와 연동하기 위해서  
몇 가지 작업이 더 필요하다.



Bar 품데  
(Height ≥ 70cm)



## • 내가 작성한 Logic

```

class ViewController: UIViewController {
    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var progressBar: UIProgressView!

    let eggTimes = ["Soft": 3, "Medium": 4, "Hard": 7]

    var result: Float = 0.0
    var totalSeconds: Float = 0.0
    var timer = Timer()

    @objc func updateTimer(){
        if result >= 0 && result < totalSeconds{
            print("(\(totalSeconds) - result) seconds")
            print("totalSeconds: \(totalSeconds), result: \(result)")
            result += 1
            progressBar.setProgress(result/totalSeconds, animated: true)
        } else if result == totalSeconds {
            titleLabel.text = "DONE!"
            timer.invalidate()
        }
    }

    func resetProgressbar(){
        progressBar.setProgress(0, animated: false)
    }

    @IBAction func hardnessSelected(_ sender: UIButton) {
        titleLabel.text = "How do you like your eggs?"
        timer.invalidate()
        resetProgressbar()
    }

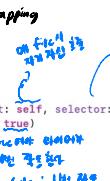
    // ! : 있다고 확신(forced-unwrapping)
    let hardness = sender.currentTitle!
    result = 0.0
    totalSeconds = Float(eggTimes[hardness]!)
    print("\(hardness) got selected") ↳ forced-unwrapping ↳ seconds

    timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector(updateTimer), userInfo: nil, repeats: true)
}

like count-1)의 값 the first element the last element

```

$\downarrow$  Stackoverflow  $\Sigma$   
 $\Sigma$  BT 31 312



result: 초기에  $1/4$  증가한다. `progressBar`의 진행상황을  
시작과 같은 번수

`totalseconds`: `eggsTimes[hardness]` 를 대입시켜 총 시간을 나타냄,

progressBar.setProgress: stackoverflow 에서 같은 코드다.

첫 요소에는  $\text{float: none}$ , 두 번째엔  $\text{float: left}$ 를 넣어준다.

`result / totalSeconds` 는 현재 초 / 전체 초 로 `updateTimer()`안에

있으나 다음에 업데이트 즉, 대체 마우스 사용 되니 `progressbar`의

## 지역사학과 멀치하다

리셋을 해주지 않았더니 다른 차원의 bar가 다른 *entity* 큐에 들어온다.  
두루 같다니 앞으로 가는 한 살에 발상하였다.

```

target: function(value) { ... } | Timeline
target: string | object: selector | selector, context
args: any[] | boolean | Time

Discussion
After 10 seconds have passed, the tree, running the message onSelect on
the target node.

Parameters
ti
The number of seconds between firing of the tree. If it's 0, it is
the result of getDuration. This method changes the non-epoch
value of 8000 seconds instead.

target
The node or selector to target. The tree makes a strong
assumption that the target is a Node or Timeline. It will
silently ignore anything else.

args
An array of arguments to pass to the target function.

onSelect
A function to call when the tree fires. The tree makes a strong
assumption that the onSelect function is a Timeline or Node. It will
silently ignore anything else.

onCreate
The selector used to fire the onCreate signature. Timeline instances
will always be created under the root node. Node instances
will be created under the target node. If the target node is
argued, the tree passes it as the parent argument. If the target node is
a Timeline, the tree passes it as the parent argument.

= onCreate(parent: Timeline | Node, value: any)

useFirst
A boolean to tell the tree that the tree is maintaining a strong
reference to the target node and the onSelect function. This
means that the tree will not release the target node.

remove
F: true, the tree will immediately release target and
deallocate. If false, the tree will wait until ti has
passed.

Return
A Promise that will be resolved when the tree has
been deallocated.

```

# • Udemy Logic

```
import UIKit
import AVFoundation

class ViewController: UIViewController {

    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var progressBar: UIProgressView!

    let eggTimes = ["Soft": 3, "Medium": 4, "Hard": 7]
    var currentTitle: String?
    var totalTime: Double
    var secondsPassed = 0
    var player: AVAudioPlayer!

    @IBAction func hardnessSelected(_ sender: UIButton) {
        timer.invalidate()
        titleLabel.text = "How do you like your eggs?"

        // ! : 00:30 例題(forced-upgrading)
        let hardness = sender.currentTitle!
        totalTime = eggTimes[hardness]!
        print("\(hardness) got selected")

        progressBar.progress = 0.0
        secondsPassed = 0
        titleLabel.text = hardness

        timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector(updateTimer), userInfo: nil, repeats: true)
    }

    @objc func updateTimer() {
        if secondsPassed <= totalTime {
            progressBar.progress = Float(secondsPassed) / Float(totalTime)
            print(Float(secondsPassed) / Float(totalTime))
            secondsPassed += 1
        } else {
            titleLabel.text = "DONE"
            playSound()
            timer.invalidate()
        }
    }

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

    func playSound() {
        let url = Bundle.main.urlForResource: "alarm_sound", withExtension: "mp3"
        player = try! AVAudioPlayer(contentsOf: url!)
        player.play()
    }
}
```

# Using the 5 Step Approach to Debug our App

```
GIFOutlet weak var titleLabel: UILabel!
GIFOutlet weak var progressBar: UIProgressView!
let options = ["Soft": 3, "Medium": 4, "Hard": 7]
var timer: Timer!
var totalTime = 0
var secondsPassed = 0
@IBAction func hardnessSelected(_ sender: UIButton) {
    timer.invalidate()
    titleLabel.text = "How do you like your eggs?"
    // If you like harder eggs, add time
    let hardness = sender.currentTitle!
    totalTime += options[hardness]!
    print("\(hardness) got selected")
    timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector(updateTime), userInfo: nil, repeats: true)
}
objc func updateTime() {
    if secondsPassed < totalTime {
        let percentageProgress = secondsPassed / totalTime
        progressBar.progress = Float(percentageProgress)
        secondsPassed += 1
    } else {
        titleLabel.text = "DONE!"
        timer.invalidate()
    }
}
```



→ progress bar 가 증가하지 않는  
상황 발생



What did you expect  
your code to do?

- 초기 기능에 따라 progressbar도 증가하는 것



What happened  
instead?

- progressbar 비등장



What does your  
expectation depend upon?

•  $\text{percentageProgress} = \frac{\text{secondsPassed}}{\text{totalTime}}$   
 $\text{progressBar.progress} = \text{Float}(\text{percentageProgress})$

secondsPassed & totalTime 간의 계산 결과에  
따라 progressbar는 증가하거나 감소할 것이다.



How can we test the things  
our expectations depend on?

- 로그 찍어보기

```
let percentageProgress = secondsPassed / totalTime
print(percentageProgress)
progressBar.progress = Float(percentageProgress)
print(Float(percentageProgress))
```

↳ 0.0 이 나옴

```
let a = 5
let b = 2
```

```
print(Float(a / b))
```

• 결과 같은 그림처럼 2.5가 나온다.

→ 만약 2를 나눈값 (2)을 그냥 float  
형식으로 꺼내 놓으면 될 것이다.  
(print(Float(1)))

해결 방법

```
let a = 5
let b = 2
print(Float(a) / Float(b))
print(a/b)
```

```
let a: Float = 5
let b: Float = 2
print(a/b)
```



Fix our code to make reality match expectations

```
@objc func updateTimer() {
    if secondsPassed < totalTime {
        progressBar.progress = Float(secondsPassed) / Float(totalTime)
        print(Float(secondsPassed) / Float(totalTime))
        print(Float(secondsPassed) / totalTime))
    }
}
```

```
0.0
0.0
0.33333334
0.0
0.66666667
0.0
```



→ 만약 Float으로 변환을 한 뒤,

계산을 시작함.

Float이 제대로 나오고 progressBar를  
증가하지만 100%가 되지 않는 암울함

### 내가 쓴 코드

```
if secondsPassed <= totalTime {
    progressBar.progress = Float(secondsPassed) / Float(totalTime)
    print(Float(secondsPassed) / Float(totalTime))
    print(Float(secondsPassed) / totalTime))

    secondsPassed += 1
}
• secondsPassed가 Maximum을 놀아놓으면
totalTime은, progressBar는 100%가 되지 않게 함.
```

### Udemy 해설 코드

```
if secondsPassed < totalTime {
    secondsPassed += 1
    progressBar.progress = Float(secondsPassed) / Float(totalTime)
    print(Float(secondsPassed) / Float(totalTime))
    print(Float(secondsPassed) / totalTime))

    • 1을 먼저 더해 줄 경우 마지막 progressBar 칠점에
    증가하지 않는 걸 즐기게 하겠다.
    만약 secondsPassed가 이미 있다면 증가를 했을 때도
    불구하고 totalTime과 같은 증가 회의에 사용할
    수 있음.
}
```

# Swift Deep Dive - Structures, Method and properties

Defining the Structure

```
struct MyStruct{ }
```

Initialising the Structure

```
MyStruct()
```

Struct (구조체)

```
struct Town{
    let name = "HoonLand"
    var citizens = ["Hoon", "Jack Bauer"]
    var resources = ["Grain":100, "Ore": 42, "Wool": 75]

    func fortify(){
        print("Defences increased!")
    }
}

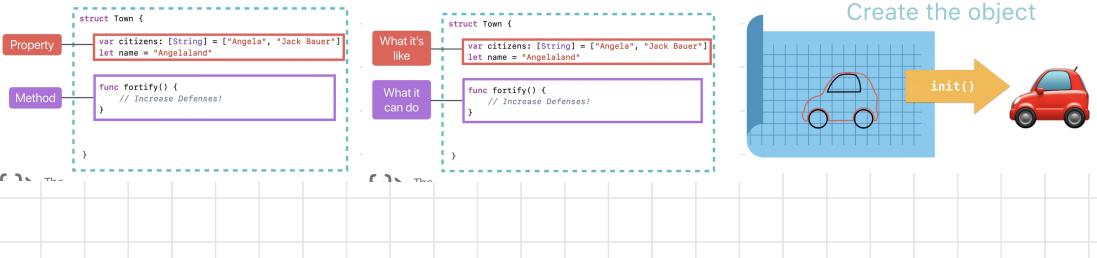
var myTown = Town() // create a new copy of Town

// able to access to properties of Town
print(myTown.citizens)
print("\(myTown.name) has \(myTown.resources["Grain"]!) bags of grain.")

// append
myTown.citizens.append("Keanu Reeves")
print(myTown.citizens.count)

// method
myTown.fortify()
```

Create the object

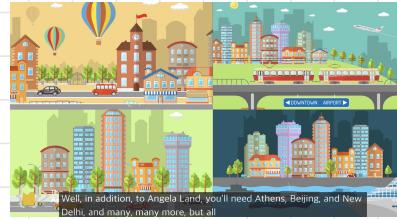


Creating the initialiser

init()로 생성하는건  
object는 만들 때마다  
생성된다.  
왜 이런게 필요한가?

Using the initialiser

```
StructureName()
```



다른 개발자를 예로 들면 여러 마을들을 갖는 것 같아야 하는데 각 마을의 인구수, 토양 등 다른 면에 차이가 많다.

```
struct Town {
    var citizens
    let name
    var resources

    func fortify() {
        print("Defenses increased!")
    }
}
```

그러면 새로운 마을을 만들 때마다  
시작은 예전 마을과 같이  
원래 있는 struct를 blueprint(모형)  
쓰imoto 사용하게 대처한다.

```

struct Town{
    let name: String
    var citizens: [String]
    var resources: [String: Int]

    init(townName: String, people: [String], stats: [String: Int]) {
        name = townName
        citizens = people
        resources = stats
    }

    func fortify(){
        print("Defences increased!")
    }
}

```

```

var anotherTown = Town(
    townName: "New Town",
    people: ["Alice", "Bob", "Charlie"],
    stats: ["Food": 100, "Water": 50, "Metal": 20]
)

```

캐싱 가능한 요소들이 자동 완성으로 나온다.

```

init(name: String, citizens: [String], resources: [String: Int]) {
    self.name = name
    self.citizens = citizens
    self.resources = resources
}

```

internal property, 이를 통해 Struct의 property에 접근

같은 파일 내에서 Struct의 property에 접근할 때 self. 를

필수로

```

func exercise() {

    // Define the User struct here
    struct User{
        let name: String
        var email: String?
        var followers: Int
        var isActive: Bool
    }

    // Initialise a User struct here
    init(name: String, email: String, followers: Int, isActive: Bool){

        self.name = name
        self.email = email
        self.followers = followers
        self.isActive = isActive
    }

    func logStatus(){
        if isActive == true{
            print("\(name) is working hard")
        } else{
            print("\(name) has left earth")
        }
    }
}

// Diagnostic code - do not change this code
print("//Diagnostic code (i.e., Challenge Hint):")
var musk = User(name: "Elon", email: "elon@tesla.com", followers: 2001, isActive: true)
musk.logStatus()
print("Contacting \(musk.name) on \(musk.email) ...")
print("\(musk.name) has \(musk.followers) followers")
// sometime later
musk.isActive = false
musk.logStatus()

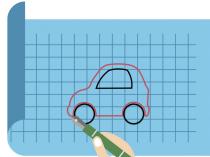
}

exercise()

```

# Swift Deep Dive - Immutability

Struct = Blueprint



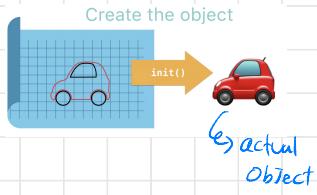
```

struct QuizBrain {
    var questionNumber = 0 ↳ what it's like associated with struct

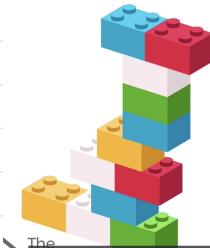
    func checkAnswer(_ userAnswer: String) -> Bool {
        if userAnswer == quiz[questionNumber].answer {
            return true
        } else {
            return false
        }
    }
} ↳ what it can do

```

Create the object

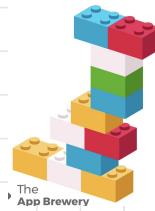


## Immutability



☞ if of 2nd let은  
이전의 값을 몰라  
변경이 불가능 → immutable

Destroy



Rebuild

> Well, instead, you have to destroy the entire structure and create a new one.

Destroy the old copy and  
Create a new copy that encompasses the change

Ex)

```

struct Town {
    let name: String
    var citizens: [String]
    var resources: [String: Int]

    init(citizens: [String], name: String, resources:[String:Int]) {
        self.citizens = citizens
        self.name = name.uppercased()
        self.resources = resources
    }

    mutating func harvestRice() {
        self.resources["Rice"] = 100 ↳ Cannot assign through subscript: 'self' is immutable
    }
}

```

self라는 것은 인자인 걸까요?

- self가 쓰는 코드의 resources 앞에는 항상 self.이 있죠!

생략해도 됩니다.

그럼 error message인 'self' is immutable 라는 걸까요?

이제는 알겠죠.

"self"는 자동으로 사용되는 property라 생각해요.

그리고 self는 let으로 선언한 걸까요.

```

mutating func harvestRice() {
    resources["Rice"] = 100
}

```

mutating은 꼭! 사용하세요

Error는 사용하지.

mutating을 빼면 자동으로 self를 var로 쓸 수 있는 거예요.

# The methods of struct

plain method

: doesn't change  
anything

mutating method

: can change the state of  
structure

```
let myTown = Town(citizens: ["Angela", "Jack Bauer"], name: "Angelaland", resources: ["Wool": 75])  
myTown.citizens.append("Keanu Reeves")  
print("People of \(myTown.name): \(myTown.citizens)")  
myTown.harvestRice()  
print(myTown.resources)
```

- when we use the "let" keyword, our struct and all its properties are immutable and we can't call a mutating function like `harvestRice`

# Quizzler



- Question Text에 케이스 나타나는데

True & False를 퀴즈에 정답을 맞는  
applet. 진정도에 따라 하든 pogressbar가 증가한다.

```
class ViewController: UIViewController {
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var progressBar: UIProgressView!
    @IBOutlet weak var falseButton: UIButton!
    @IBOutlet weak var trueButton: UIButton!

    let quiz = [
        "One + One is equal to Six", "True",
        "Five - Three is greater than One", "True",
        "Three + Eight is less than Ten", "False"
    ]

    var questionNumber = 0

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

    // 사용자 버튼이 눌렀을 때
    @IBAction func answerButtonPressed(_ sender: UIButton) {
        let userAnswer = sender.currentTitle ?? "None"
        let actualAnswer = quiz[questionNumber][1] *
            if userAnswer == actualAnswer {
                print("Right!")
            } else {
                print("Wrong!")
            }
        if questionNumber + 1 < quiz.count {
            questionNumber += 1
        } else {
            questionNumber = 0
        }
        updateUI()
    }

    func updateUI() {
        questionLabel.text = quiz[questionNumber].text
    }
}
```

## 코드 상태:

2D 배열 ([2x1] 배열)로 만든 Question과 Answer를 이욯해진  
array에 따라 Question text를 뜨우고 점을 표시한다.

questionNumber는 quiz 배열의 위치로 도달할 때까지  
answer 배열에 따라 주제와 함께 몇 번째 (question)에  
있는지 알 수 있고 그 인덱스를 quiz 배열에 삽입해  
문제를 표시한다.

quiz 배열이 지정된 채 놓이고 판례가 잘 안되어 때문에

세로운 파일에서 struct를 만든 것이다.

\* 처음 같은 불리는 데도 빙거운 불이 있다.



구조체의 property를 사용해 문제는 오류

```
let quiz = [
    Question(text: "Four + Two is equal to Six", answer: "True"),
    Question(text: "Five - Three is greater than One", answer: "True"),
    Question(text: "Three + Eight is less than Ten", answer: "False")
]
```

```
let actualQuestion = quiz[questionNumber]
let actualAnswer = actualQuestion.answer
```

이후 actualQuestion의 용도는 없거나.

struct를 정의한 후,  
하위에 초기화 가지고 있는 quiz의  
index 높이로 넘기면 Crash 된다. 때문에  
Safety Check을 해야 한다



## .4) Safety Check

```
if questionNumber+1 >= quiz.count {
    print("no more question")
} else {
    questionNumber += 1
    updateUI()
}
```

## Udemys safety check

```
if questionNumber + 1 < quiz.count {
    questionNumber += 1
} else {
    questionNumber = 0
}
updateUI()
```

• 대체적으로 비슷해 보인다.

여전히 빙거운 if로 두고 다음은 두드려 차이다.  
Udemys 책이 가르치는 대체하게 좋은 것 같다.



```

@IBAction func answerButtonPressed(_ sender: UIButton) {
    let userAnswer = sender.currentTitle // True, False
    let actualQuestion = quiz[questionNumber]
    let actualAnswer = actualQuestion.answer

    if userAnswer == actualAnswer {
        sender.backgroundColor = UIColor.green
    } else {
        sender.backgroundColor = UIColor.red
    }
}

```

User Answer이 Actual Answer일 경우  
Button의 backgroundColor가 바뀌는 효과를  
나타내고 있는데요. updateUI에서 정답인 효과니까  
다시 풀어놓은 이유는 그걸 나중에 사용할 예정이  
있기 때문입니다.

```

func updateUI(){
    trueButton.backgroundColor = UIColor.clear
    falseButton.backgroundColor = UIColor.clear
}

```

### • 4th Solution

```

func updateUI() {
    DispatchQueue.main.asyncAfter(deadline: .now() + 0.2) {
        self.trueButton.backgroundColor = UIColor.clear
        self.falseButton.backgroundColor = UIColor.clear
    }
    questionLabel.text = quiz[questionNumber].text
}

```

-은 Xylophone project에서 간접 탐색  
입시적인 효과를 두 DispatchQueues를  
이용하여 0.2초 후에 정답이나 틀리거나  
하였다. 즉, 저 Method 안에서 내가 설정한  
x 초기 단계를 일정한 단계로.  
(물론 위에서 정답이나 틀리거나 오류 발생에  
색깔 효과를 적용하지 않아요.)

### Angelou's Solution

```

@IBAction func answerButtonPressed(_ sender: UIButton) {
    let userAnswer = sender.currentTitle // True, False
    let actualQuestion = quiz[questionNumber]
    let actualAnswer = actualQuestion.answer

    if userAnswer == actualAnswer {
        sender.backgroundColor = UIColor.green
    } else {
        sender.backgroundColor = UIColor.red
    }

    if questionNumber + 1 < quiz.count {
        questionNumber += 1
        questionLabel.text = quiz[questionNumber].text
    }
}

Timer.scheduledTimer(timeInterval: 0.2, target: self, selector: #selector(updateUI), userInfo: nil, repeats: false)

@objc func updateUI() {
    trueButton.backgroundColor = UIColor.clear
    falseButton.backgroundColor = UIColor.clear
    questionLabel.text = quiz[questionNumber].text
}

```

Angelou는 egTimer와 함께 Timer를  
이용하였습니다.

timeInterval을 0.2로 updateUI function이 0.2초마다  
실행되며 그 때 backgroundColor를 초기화하고  
다음 question으로 넘깁니다. 큼직한 번역입니다. 일어나기  
difficult이 repeat은 false입니다.

\* selector는 원래 있는 updateUI Method를 이용합니다.  
그 이유는 우리가 정한 TimeInterval처럼 내용을 Method를  
넘기면서 보내야 굳이 새로 만들 필요없이 기존의 Method를  
이용한 것이고, 앞서 배운 Objective-C의 방법이기 때문에  
① ObjC를 알아 봄이 좋습니다.



# • progress bar

```

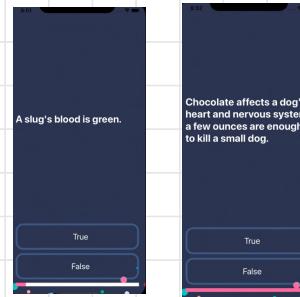
import UIKit
class ViewController: UIViewController {
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var progressBar: UIProgressView!
    @IBOutlet weak var falseButton: UIButton!
    @IBOutlet weak var trueButton: UIButton!
    let quiz = [
        Question(q: "A slug's blood is green.", a: "True"),
        Question(q: "According to one quarter of human bones are in the feet.", a: "True"),
        Question(q: "The surface area of two human lungs is approximately 78 square metres.", a: "True"),
        Question(q: "In West Virginia, USA, if you accidentally hit an elephant while driving through a forest, you are legally allowed to eat it.", a: "True"),
        Question(q: "In London, UK, if you happen to die in the House of Parliament, you are technically entitled to a state funeral, because the building is considered too sacred a place.", a: "False"),
        Question(q: "It is illegal to pee in the Ocean in Portugal.", a: "True"),
        Question(q: "You can lead a cow down stairs but not up stairs.", a: "False"),
        Question(q: "Google was originally called 'Backrub'.", a: "True"),
        Question(q: "Buzz Aldrin's mother's maiden name was 'Moon'.", a: "True"),
        Question(q: "The loudest sound produced by any animal is 188 dB.", a: "False"),
        Question(q: "An ostrich's nest is an African Element.", a: "False"),
        Question(q: "No piece of square dry paper can be folded in half more than 7 times.", a: "False"),
        Question(q: "Chocolate affects a dog's heart and nervous system; a few ounces are enough to kill a small dog.", a: "True")
    ]
    var questionNumber = 0
    override func viewDidLoad() {
        super.viewDidLoad()
        updateUI()
    }
    // 실행하면 button의 기능 구현
    @IBAction func answerButtonPressed(_ sender: UIButton) {
        let userAnswer = sender.currentTitle // True, False
        let actualAnswer = quiz[questionNumber].answer
        let actualQuestion = quiz[questionNumber]
        if userAnswer == actualAnswer {
            sender.backgroundColor = UIColor.green
        } else {
            sender.backgroundColor = UIColor.red
        }
        if questionNumber + 1 < quiz.count {
            questionNumber += 1
        } else {
            questionNumber = 0
            progressBar.progress = 0
        }
        Timer.scheduledTimer(timeInterval: 0.2, target: self, selector:
            #selector(updateUI)), userInfo: nil, repeats: false)
    }
    @objc func updateUI() {
        trueButton.backgroundColor = UIColor.clear
        falseButton.backgroundColor = UIColor.clear
        questionLabel.text = quiz[questionNumber].text
        progressBar.progress = Float(questionNumber+1)/Float(quiz.count)
    }
}

```

— 이번이 eggTimer가 이은 두번째 progressbar 응용인데,  
내가 느낀건 progressbar는 굉장히 간단한 수 있으나,  
현재 진행상황 / 전체의 양 or 깊숙 등 두모로써 대량 개체로된  
이해가 필요 하다는 것이다.

이번에는 현재 QuestionNumber / 전체 Question 개수로  
간단 했다.  
문제를 맞추면 툭리고 Timer의 interval을  
일으키는 updateUI의 코드를 작성해 progressbar가  
상승 하게 하면 된다.

## • 결과

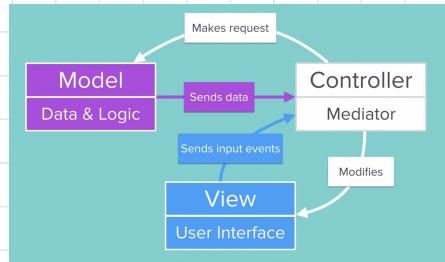
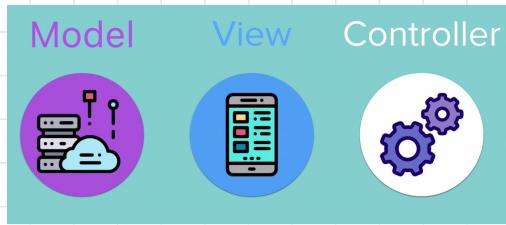


# Quizzer - MVC pattern

현재 Quizzer의 코드는 하루의 파일에만 위치하며 작동한다.

그렇게 되면 가독성이 떨어질 뿐만 아니라 유지보수하기 힘든다.

그러니깐 design pattern들 중 하루인 **MVC pattern**을 사용한 것이다.



## • M - Model

- Application의 정보, data를 나타낸다.  
database, 처음에 정의하는 상수(constant), 초기값, 변수(var) 등을 뜻한다.  
또한 이러한 data 정보들의 가성을 책임지는 component를 말한다.

### - Model의 구조

- 1. 사용자가 편집하기 원하는 모든 data를 가지고 있어야 한다.

즉, 화면 안의 네모박스에 글자나 표령된다면, 네모박스의 화면 위치 정보, 네모박스의 크기정보, 글자내용, 글자의 위치, 글자의 표면 정보 등을 가지고 있어야 하는 것이다.

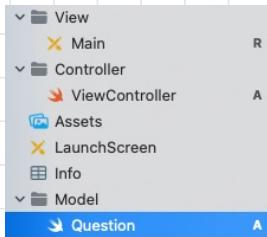
- 2. View나 controller에 대해서 어떤 정보도 알지 않아야 한다

data 변경이 일어났을 때 Model에서 화면 UI를 직접 조정해서 수정할 수 있도록 View를 참조하는 내부 속성값을 가지면 안된다.

- 3. 변경이 일어나면, 변경통지에 대해서 처리방법을 구현 해야 한다.

• Model의 속성 중 text 정보가 변경이 된다면, 이벤트를 발생시키거나 누군가에게 전달해야 하며, 누군가 Model을 변경하도록 요청하는 이벤트를 발생을 때 이를 수신하는 처리방법을 고민해야 한다. 또한 Model은 재사용 가능해야 하여 다른 인터페이스에서도 변하지 않아야 한다.

# Quizler의 Model



```
struct Question{  
    let text: String  
    let answer: String  
    init(q: String, a: String){  
        text = q  
        answer = a  
    }  
}
```

↳ Quizler의 Model은

Question: 문제와 정답 정보를 저장하는 구조체,

QuizBrain: Question의 정보를 두대로  
그리고 사용하는 data 할당,  
정답 정보를 하였다.

모든 parameter 앞에 다른 parameter 이름을 넣거나 \_를 넣을 수 있는데, \_는 외부에서 호출할 때 아무것도 안날거나 아니면 빈거나 넣고 할 수 있다.  
앞에 있는 parameter는 external parameter name이라고 하고 이 예스드 형태에서 쓰이는 parameter name이라고 checkAnswer 메소드 안에서 쓰이는 걸 internal parameter name이라고 한다.

```
var questionNumber = 0  
// : external parameter name, userAnswer : internal parameter name  
func checkAnswer(userAnswer: String) {  
    |  
}
```

- ViewController에 있던 변수와 method는  
QuizBrain struct에 가지고 온 모습.



```
@IBAction func answerButtonPressed(_ sender: UIButton) {  
    let userAnswer = sender.currentTitle // True, False  
    quizBrain.checkAnswer(userAnswer)
```

- 보내기전 checkAnswer에 userAnswer가  
넣었다. 하지만 optional로 오주가 발생했는지  
그 이후는 userAnswer는 optional String이지  
checkAnswer는 actual String을 expect  
하고 있기 때문이다.



```
let userAnswer = sender.currentTitle! // True, False  
quizBrain.checkAnswer(userAnswer)
```

- currentTitle은 force-unwrapping 해주어야  
해결 하였다.



```
@objc func updateUI() {  
    questionLabel.text = quizBrain.getQuestionText()  
    progressBar.progress = quizBrain.getProgress()  
    trueButton.backgroundColor = UIColor.clear  
    falseButton.backgroundColor = UIColor.clear
```

- questionLabel.text = quiz[questionNumber].text

↓  
quizBrain의 Method

progressBar.progress =  
 Float(questionNumber + 1) /  
 Float(quiz.count)

↓  
quizBrain의 Method

```
func nextQuestion(){  
    if questionNumber + 1 < quiz.count {  
        questionNumber += 1  
    } else {  
        questionNumber = 0  
    }  
}
```

↳ mutating func는 H176 (Swift Deep Dive↑)

Left side of mutating operator isn't mutable: 'self' is immutable

Cannot assign to property: 'self' is immutable

```
func getQuestionText() -> String{  
    // quiz[questionNumber].text  
    return quiz[questionNumber].text  
}
```

```
func getProgress() -> Float{  
    Float(questionNumber + 1) / Float(quiz.count)  
}
```

## • V - View

- input text, checkbox 항목 등과 같은 user interface 요소를 나타낸다.  
다시 말해 data의 입력, 그리고 보여주는 출력을 담당한다.  
data를 기반으로 사용자들이 볼 수 있는 화면이다.

### • View의 구조

1. Model이 가지고 있는 정보를  
다른 처리해서는 안된다.



. 화면에 글자를 표시하기 위해 Model이 가지고 있는  
정보를 전달 받게 된 템데, 그 정보로 유지하기 위해서  
임의의 View 내부에 저장하면 안된다.  
단순히 네오박스를 그리라는 명령을 받으면,  
화면에 표시하기만 하고, 그 화면을 그릴 때 필요한  
정보들은 저장하면 안된다.

2. Model이나 controller와 같이  
다른 구성요소를 몰라야 한다.



. Model과 같은 자기 자신의 모든 빼고는  
다른 모스는 참조하거나 어떻게 동작하는지  
알아서는 안된다.  
View는 data를 받으면 화면에 표시하는 역할임.

3. 변경이 일어나면 변경되는 대로  
처리방법을 구현해야만 한다.



. Model과 같이 변경이 일어났을 때 이를  
누군가에게 변경을 알리워야 하는 방법을 구현해야 한다.  
View에서는 화면에서 사용자가 화면에 표시된 내용을 변형하기  
되면 이를 Model에게 전달하여 Model을 변경해야 할 것이다.  
또한, 재사용 가능하게끔 설계를 해야하며 다른 정보들을  
표현할 때 쉽게 설계를 해야 한다.

### • Xcode의 View



- Xcode에서 설계 뷰에서는  
View의 구조들이 크게 상관이  
없는 것 같다.

## • C - Controller

- data와 UserInterface 요소들은 있는 대화역할을 한다.  
즉, 사용자가 data를 클릭하고, 수정하는 것에 대한  
“이벤트”들을 처리하는 부분을 뜻한다.

### • Controller의 규칙

- 1. Model이나 View에 대해서 알고 있어야 한다.
  - Model이나 View는 서로의 존재를 모르고, 변경을 외부로 알리고, 승인하는 방법만 가지고 있는데, 이를 Controller가 중재 하기 위해 Model과 그에 관련된 View에 대해 알고 있어야 한다.
- 2. Model이나 View의 변경을 모니터링 해야 한다.
  - Model이나 View의 변경 통지를 받으면 이를 해석해서 각각의 구성요소에게 통지를 해야 한다.  
또한, application의 Main Logic은 Controller가 담당하게 된다.

# QuizBrain의 Controller

X QuizBrain을 MVC pattern으로 재설계하고  
그 후에 challenge로 선택지가 3개 있는 Question으로  
바꾸는 걸 했는데, 그걸 dataSource를 collectionView로  
challenge 버전으로 처리 한다.

```
class ViewController: UIViewController {
    @IBOutlet weak var scoreLabel: UILabel!
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var progressBar: UIPrgressView!
    @IBOutlet weak var firstButton: UIButton!
    @IBOutlet weak var secondButton: UIButton!
    @IBOutlet weak var thirdButton: UIButton!

    var buttonArray: [UIButton] = [UIButton]() // 이니셜라이저를 사용하여 빈 배열 생성
    var quizBrain = QuizBrain()

    override func viewDidLoad() {
        super.viewDidLoad()
        buttonArray = [firstButton, secondButton, thirdButton]
        updateUI()
    }

    // 실질적인 button의 기능 구현
    @IBAction func answerButtonPressed(_ sender: UIButton) {

        let userAnswer = sender.currentTitle! // True, False
        let userGotItRight = quizBrain.checkAnswer(userAnswer)

        if userGotItRight == true {
            sender.backgroundColor = UIColor.green
        } else {
            sender.backgroundColor = UIColor.red
        }

        quizBrain.nextQuestion()

        Timer.scheduledTimer(timeInterval: 0.2, target: self, selector: #selector(updateUI), userInfo: nil, repeats: false)
    }

    @objc func updateUI() {
        updateAnswerButton()
        questionLabel.text = quizBrain.getQuestionText()
        progressBar.progress = quizBrain.getProgress()
        scoreLabel.text = "Score : \(quizBrain.getScore())"
        clearButton()
    }

    func clearButton(){
        firstButton.backgroundColor = UIColor.clear
        secondButton.backgroundColor = UIColor.clear
        thirdButton.backgroundColor = UIColor.clear
    }

    func updateAnswerButton(){
        for i in 0..<buttonArray.count{
            buttonArray[i].setTitle(quizBrain.quiz[quizBrain.questionNumber].answer[i], for: .normal)
        }
    }
}
```

① 버튼 3개에 각각 선택지를 어떤 방법으로 넣을까 하니까  
array를 이용하게 했지

② viewDidLoad는 즉시 배열이  
button을 설정한다.

③ button의 초기값.  
↳ ;버튼 button  
title(answer) 변경

```

import Foundation

struct Question{
    let text: String
    let answer: Array<String>
    let correctAnswer : String

    init(q: String, a: Array<String>, correctAnswer: String){
        text = q
        answer = a
        self.correctAnswer = correctAnswer
    }
}

import Foundation

struct QuizBrain{
    // BM: WW : control + i
    let quiz = [
        Question(q: "Which is the largest organ in the human body?", a: ["Heart", "Skin", "Large Intestine"], correctAnswer: "Skin"),
        Question(q: "Five dollars is worth how many nickels?", a: ["20", "25", "10"], correctAnswer: "25"),
        Question(q: "What do the letters in the GMT time zone stand for?", a: ["Global Meridian Time", "Greenwich Mean Time", "General Median Time"], correctAnswer: "Greenwich Mean Time"),
        Question(q: "What is the French word for 'hat'?", a: ["Chapeau", "Echarpe", "Bonnet"], correctAnswer: "Chapeau"),
        Question(q: "In past times, what would a gentleman keep in his top pocket?", a: ["Notebook", "Handkerchief", "Watch"], correctAnswer: "Watch"),
        Question(q: "How many sides does a hexagon have?", a: ["4", "5", "6"], correctAnswer: "6"),
        Question(q: "Which of these colours is NOT featured in the logo for Google?", a: ["Green", "Orange", "Blue"], correctAnswer: "Orange"),
        Question(q: "What type of alcohol drink is made from molasses?", a: ["Rum", "Whisky", "Gin"], correctAnswer: "Rum"),
        Question(q: "What animal was Humpty Dumpty?", a: ["Panda", "Ostrilla", "Crocodile"], correctAnswer: "Ostrilla"),
        Question(q: "Where is Tasmania located?", a: ["Indonesia", "Australia", "Scotland"], correctAnswer: "Australia")
    ]

    var questionNumber = 0
    var score = 0

    // - external parameter name, userAnswer: internal parameter name
    mutating func checkAnswer(_ userAnswer: String) -> Bool {
        // userAnswer: internal parameter name, used in this method.
        if userAnswer == quiz[questionNumber].correctAnswer{
            score += 1
            return true
        } else {
            return false
        }
    }

    func getScore() -> Int{
        return score
    }

    mutating func nextQuestion(){
        if questionNumber + 1 < quiz.count {
            questionNumber += 1
        } else {
            questionNumber = 0
            score = 0
        }
    }

    func getQuestionText() -> String{
        return quiz[questionNumber].text
    }

    func getProgress() -> Float{
        let progress = Float(questionNumber+1)/Float(quiz.count)
        return progress
    }
}

```

→ Model  
- Question

→ Model  
- QuizBrain

# Quizzer - Multiple choice Udemy & Me's Code Review

## • Udemy's code

### . Model - Question

#### - Udemy

```
import Foundation

struct Question {
    let text: String

    //Multiple choice questions have multiple answers, an Array of Strings would work for our quiz data.
    let answers: [String]
    //Look at the data in the quiz array, there is a seperate string that is the correctAnswer.
    let rightAnswer: String

    //The initialiser needs to be updated to match the new multiple choice quiz data.
    init(q: String, a: [String], correctAnswer: String) {
        text = q
        answers = a
        rightAnswer = correctAnswer
    }
}
```

#### - Me

```
import Foundation

struct Question{
    let text: String
    let answer: Array<String>
    let correctAnswer : String

    init(q: String, a: Array<String>, correctAnswer: String){
        text = q
        answer = a
        ? self.correctAnswer = correctAnswer
    }
}
```

- 다른 점은 띄히 있다.  
나의 correctAnswer 보다 rightAnswer CT 가정한 이유 같고  
self는 쓸 것 보다 없어보다

# . Model 1 - QuizBrain

## · Udemy

```
import Foundation

struct QuizBrain {
    var questionNumber = 0
    var score = 0

    let quiz = [
        Question(q: "What is the largest organ in the human body?", a: ["Heart", "Skin", "Large Intestine"], correctAnswer: "Skin"),
        Question(q: "Five dollars is worth how many nickels?", a: ["100", "1000"], correctAnswer: "100"),
        Question(q: "What do the letters in the GMT time zone stand for?", a: ["Global Meridian Time", "Greenwich Mean Time"], correctAnswer: "Greenwich Mean Time"),
        Question(q: "The French word for 'hat'?", a: ["Chapeau", "Écharpe", "Bonnet"], correctAnswer: "Chapeau"),
        Question(q: "In past times, what would a gentleman keep in his fob pocket?", a: ["Notebook", "Handkerchief", "Watch"], correctAnswer: "Watch"),
        Question(q: "What is the capital city of Thailand?", a: ["Bangkok", "Phuket", "Koh Samui"], correctAnswer: "Bangkok"),
        Question(q: "What one of these colours is NOT featured in the logo for Google?", a: ["Green", "Orange", "Blue"], correctAnswer: "Orange"),
        Question(q: "What alcoholic drink is made from molasses?", a: ["Rum", "Whisky", "Gin"], correctAnswer: "Rum"),
        Question(q: "What type of animal was Harambe?", a: ["Panda", "Gorilla", "Crocodile"], correctAnswer: "Gorilla"),
        Question(q: "Where is Tasmania located?", a: ["Indonesia", "Australia", "Scotland"], correctAnswer: "Australia")
    ]

    func getQuestionText() -> String {
        return quiz[questionNumber].text
    }

    // Need a way of fetching the answer choices.
    func getAnswers() -> [String] {
        return quiz[questionNumber].answers
    }

    func getProgress() -> Float {
        return Float(questionNumber) / Float(quiz.count)
    }

    mutating func getScore() -> Int {
        return score
    }

    mutating func nextQuestion() {
        if questionNumber + 1 < quiz.count {
            questionNumber += 1
        } else {
            questionNumber = 0
        }
    }

    mutating func checkAnswer(isCorrect: Bool) {
        // Need to change answer to rightAnswers inc.
        if userAnswer == quiz[questionNumber].rightAnswer {
            score += 1
            print("Yay!")
        } else {
            print("Nah!")
        }
    }
}

mutating func checkAnswer(userAnswer: String) -> Bool {
    // Need to change answer to rightAnswers inc.
    if userAnswer == quiz[questionNumber].rightAnswer {
        score += 1
        print("Yay!")
    } else {
        print("Nah!")
    }
}
```

→ d把握은 끝났다.  
answer 빠져는 return 값  
Method는 Model이랑 맞지 않아.  
구조 분석은 끝났다.

## · Me

```
import Foundation

struct QuizBrain {
    let quiz = [
        Question(q: "Which is the largest organ in the human body?", a: ["Heart", "Skin", "Large Intestine"], correctAnswer: "Skin"),
        Question(q: "Five dollars is worth how many nickels?", a: ["100", "1000"], correctAnswer: "100"),
        Question(q: "What do the letters in the GMT time zone stand for?", a: ["Global Meridian Time", "Greenwich Mean Time"], correctAnswer: "Greenwich Mean Time"),
        Question(q: "The French word for 'hat'?", a: ["Chapeau", "Écharpe", "Bonnet"], correctAnswer: "Chapeau"),
        Question(q: "In past times, what would a gentleman keep in his fob pocket?", a: ["Notebook", "Handkerchief", "Watch"], correctAnswer: "Watch"),
        Question(q: "What is the capital city of Thailand?", a: ["Bangkok", "Phuket", "Koh Samui"], correctAnswer: "Bangkok"),
        Question(q: "What one of these colours is NOT featured in the logo for Google?", a: ["Green", "Orange", "Blue"], correctAnswer: "Orange"),
        Question(q: "What is the French word for 'hat'?", a: ["Chapeau", "Écharpe", "Bonnet"], correctAnswer: "Chapeau"),
        Question(q: "In past times, what would a gentleman keep in his fob pocket?", a: ["Notebook", "Handkerchief", "Watch"], correctAnswer: "Watch"),
        Question(q: "How would one say goodbye in Spanish?", a: ["Au Revoir", "Adios", "Salir"], correctAnswer: "Adios"),
        Question(q: "Which of these colours is NOT featured in the logo for Google?", a: ["Green", "Orange", "Blue"], correctAnswer: "Orange"),
        Question(q: "What alcoholic drink is made from molasses?", a: ["Rum", "Whisky", "Gin"], correctAnswer: "Rum"),
        Question(q: "What type of animal was Harambe?", a: ["Panda", "Gorilla", "Crocodile"], correctAnswer: "Gorilla"),
        Question(q: "Where is Tasmania located?", a: ["Indonesia", "Australia", "Scotland"], correctAnswer: "Australia")
    ]

    var questionNumber = 0
    var score = 0

    // : external parameter name, userAnswer : internal parameter name
    mutating func checkAnswer(_ userAnswer: String) -> Bool {
        // userAnswer == quiz[questionNumber].correctAnswer
        if userAnswer == quiz[questionNumber].correctAnswer {
            score += 1
            return true
        } else {
            return false
        }
    }

    func getScore() -> Int {
        return score
    }

    mutating func nextQuestion() {
        if questionNumber + 1 < quiz.count {
            questionNumber += 1
        } else {
            questionNumber = 0
            score = 0
        }
    }

    func getQuestionText() -> String {
        return quiz[questionNumber].text
    }

    func getProgress() -> Float {
        let progress = Float(questionNumber+1)/Float(quiz.count)
        return progress
    }
}
```

## • Controller - View Controller

```

@IBOutlet weak var questionLabel: UILabel!
@IBOutlet weak var progressBar: UIProgressView!

//Added another button and a corresponding outlet.
@IBOutlet weak var choice1: UIButton!
@IBOutlet weak var choice2: UIButton!
@IBOutlet weak var choice3: UIButton!
@IBOutlet weak var choice4: UIButton!

@IBOutlet weak var scoreLabel: UILabel!

var quizBrain = QuizBrain()

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

//New button needs to be linked to this IBAction too.
@IBAction func answerButtonPressed(_ sender: UIButton) {
    let userAnswer = sender.currentTitle!
    let userGetItRight = quizBrain.checkAnswer(userAnswer: userAnswer)

    if userGetItRight {
        sender.backgroundColor = UIColor.green
    } else {
        sender.backgroundColor = UIColor.red
    }

    quizBrain.nextQuestion()
}

Timer.scheduledTimer(timeInterval: 0.2, target: self, selector: #selector(updateUI), userInfo: nil, repeats: false)

objc_func updateUI() {
    questionLabel.text = quizBrain.getQuestionText()

    //Need to fetch the answers and update the button titles using the setTitle method.
    let answerChoices = quizBrain.getAnswers()
    choice1.setTitle(answerChoices[0], for: .normal)
    choice2.setTitle(answerChoices[1], for: .normal)
    choice3.setTitle(answerChoices[2], for: .normal)
    choice4.setTitle(answerChoices[3], for: .normal)

    progressBar.progress = quizBrain.getProgress()
    scoreLabel.text = "Score: \(quizBrain.getScore())"

    choice1.backgroundColor = UIColor.clear
    choice2.backgroundColor = UIColor.clear
    choice3.backgroundColor = UIColor.clear
    choice4.backgroundColor = UIColor.clear
}

//Third button needs to be reset too.
choice3.backgroundColor = UIColor.clear

```

→ 022 Ako/2  
만화는 양지다.

```

class ViewController: UIViewController {

    @IBOutlet weak var scoreLabel: UILabel!
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var progressBar: UIProgressView!
    @IBOutlet weak var firstButton: UIButton!
    @IBOutlet weak var secondButton: UIButton!
    @IBOutlet weak var thirdButton: UIButton!

    var buttonArray: [UIButton] = [UIButton]()
    var quizBrain = QuizBrain()

    override func viewDidLoad() {
        super.viewDidLoad()
        buttonArray = [firstButton, secondButton, thirdButton]
        updateUI()
    }

    // 터치되었을 때 기록
    @IBAction func answerButtonPressed(_ sender: UIButton) {
        let userAnswer = sender.currentTitle! // True, False
        let userGetItRight = quizBrain.checkAnswer(userAnswer)

        if userGetItRight {
            sender.backgroundColor = UIColor.green
        } else {
            sender.backgroundColor = UIColor.red
        }

        quizBrain.nextQuestion()

        Timer.scheduledTimer(timeInterval: 0.2, target: self, selector: #selector(updateUI), userInfo: nil, repeats: false)
    }

    @objc func updateUI() {
        updateUI()
        questionLabel.text = quizBrain.getQuestionText()
        progressBar.progress = quizBrain.getProgress()
        scoreLabel.text = "Score: \(quizBrain.getScore())"
        clearButton()
    }

    func clearButton() {
        firstButton.backgroundColor = UIColor.clear
        secondButton.backgroundColor = UIColor.clear
        thirdButton.backgroundColor = UIColor.clear
    }

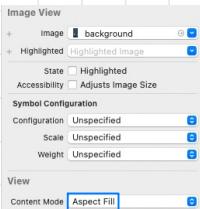
    func updateAnswerButtons() {
        for i in 0..

→ 022 Ako/4  
UI를 그려서 대체해버렸지.  
→ 022 Ako/4  
Controller와 View는 같은 공간에 있는 것 같아요.

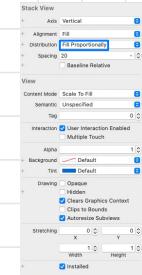

```

# Destini

## 1. Background



- stackView setting



## 4. Apply MVC

### • Model - Story

```
import Foundation

struct Story {
    var title: String
    var story: String
    var choice1: String
    var choice2: String

    init(title: String, choice1: String, choice2: String) {
        self.title = title
        self.choice1 = choice1
        self.choice2 = choice2
    }
}
```

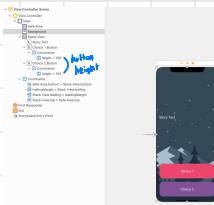
• Story 구조체를 만들어 사용하게 필요할 data Format 정의를 담고 있다.

### • Model - StoryBrain

```
struct StoryBrain {
    let stories = [
        Story(title: "You see a fork in the road.", choice1: "Take a left.", choice2: "Take a right."),
        Story(title: "You see a tiger.", choice1: "Shout for help.", choice2: "Play dead."),
        Story(title: "You find a treasure chest.", choice1: "Open it.", choice2: "Check for traps.")
    ]
}
```

- StoryBrain 구조체에 Story를 이용해 Stories Analysis를 만들었다.

## 2. View & constraints



- button의 간격은 저정도로 설정 하길 원했지만, Stack View는 적용 시키고 button 사이 간격이 조작이 안되서 다시 unembed하고 조작을 다시 Stack View로 만들었다.
- \* StackView 상태에서도 조정 가능성이 있음을.

## 3. IBOutlet - button, label

```
@IBOutlet weak var storyText: UILabel!
@IBOutlet weak var choiceButton: UIButton!
@IBOutlet weak var choice2Button: UIButton!

let story0 = "You see a fork in the road."
let choice0 = "Take a left."
let choice1 = "Take a right."

override func viewDidLoad() {
    storyText.text = story0
    choiceButton.setTitle(choice0, for: .normal)
    choice2Button.setTitle(choice1, for: .normal)
    super.viewDidLoad()
}
```

- button과 label은 연결해 테스트 해 봤다.

## Controller - ViewController

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var storyText: UILabel!
    @IBOutlet weak var choiceButton: UIButton!
    @IBOutlet weak var choice2Button: UIButton!

    var storyBrain = StoryBrain()

    override func viewDidLoad() {
        storyText.text = storyBrain.stories[0].title
        choiceButton.setTitle(storyBrain.stories[0].choice1, for: .normal)
        choice2Button.setTitle(storyBrain.stories[0].choice2, for: .normal)
        super.viewDidLoad()
    }
}
```

- Story와 StoryBrain을 이용해 View와 Model에 연결시키는 logic을 구현했다.

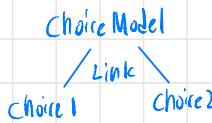


## 5. IBAction



```
@IBAction func choiceMade(_ sender: UIButton) {
    if sender == choice1Button {
        storyText.text = storyBrain.stories[1].title
        choice1Button.setTitle(storyBrain.stories[1].choice1, for: .normal)
        choice2Button.setTitle(storyBrain.stories[1].choice2, for: .normal)
    } else {
        storyText.text = storyBrain.stories[2].title
        choice1Button.setTitle(storyBrain.stories[2].choice1, for: .normal)
        choice2Button.setTitle(storyBrain.stories[2].choice2, for: .normal)
    }
}
```

- 각 button을 누를 때마다 새로운 선택지를 나오게 한다.



· 모식도 (diagram)



## 6. Model (StoryBrain) - Next Story function

```
mutating func nextStory() {
    if storyNumber+1 < stories.count {
        storyNumber += 1
    } else {
        storyNumber = 0
    }
}
```

```
func updateUI() {
    updateUI()
    storyText.text = storyBrain.getStory()
    choice1Button.setTitle(storyBrain.getChoice1(), for: .normal)
    choice2Button.setTitle(storyBrain.getChoice2(), for: .normal)
    print("updateUI")
}
```

View - viewcontroller



- Update UI가 재执시 작동하지 않는다.  
작동 ViewDidLoad를 재执시하는 흐름이 안된다.

- 또한 스크린은 못 짜워지만

choice 1 → storyNumber += 1

choice 2 → storyNumber += 2 가 되기

구성했지만, 첫번째 가지는 작동이 잘되다가  
두번째 클릭 시부터 choice 1 을 누르면 각  
choice 2로 인식한다.

## • Model - Story Brain

```

mutating func nextStory(userChoice: String) {
    if userChoice == stories[storyNumber].choice1 {
        print("choice1 : \(storyNumber)")
        storyNumber += 1
        print("choice1 : \(storyNumber)")
    } else {
        print("choice2 : \(storyNumber)")
        storyNumber += 2
        print("choice2 : \(storyNumber)")
    }
}

func getStory() -> String {
    return stories[storyNumber].title
}

func getChoice1() -> String {
    return stories[storyNumber].choice1
}

func getChoice2() -> String {
    return stories[storyNumber].choice2
}

```



What did you expect your code to do?



What happened instead?



What does your expectation depend upon?



How can we test the things our expectations depend on?

• 무엇이 문제인가?

## 5 step approach to Debug our App

통해 문제를 해결하자

- 누르는 Button에 따라 storyNumber의 증가 수가 다르고, updateUI를 통해 다음 story로 넘어가는가?

- 누르는 button에 따라 story가 달라지니  
button마다 증가하는 storyNumber도 다르다

그것이 바로 반영되지 않은 뿐이며,  
updateUI로 최초 배포를 제외하고는 실행이 안된다

### • Model - Story Brain

```

mutating func nextStory(userChoice: String) {
    if userChoice == stories[storyNumber].choice1 {
        print("choice1 : \(storyNumber)")
        storyNumber += 1
        print("choice1 : \(storyNumber)")
    } else {
        print("choice2 : \(storyNumber)")
        storyNumber += 2
        print("choice2 : \(storyNumber)")
    }
}

```

### Controller ~ ViewController

```

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

@IBAction func choice1(_ sender: UIButton) {
    stories[storyNumber].choice1 = "choice1"
    print(stories[storyNumber].choice1)
}

@IBAction func choice2(_ sender: UIButton) {
    stories[storyNumber].choice2 = "choice2"
    print(stories[storyNumber].choice2)
}

func updateUI() {
    storyText.text = choice1Text
    choice1Text = "choice 1.2 text"
    choice2Text = "choice 2.2 text"
}

```

• updateUI의 의문점:  
storyText, choice1Text가 동시에 업데이트 되어  
처음 내용 그대로 remains 들어오게 되고,  
그것이 nextStory에서 이미 다음 story로 넘어간 내용과  
당연히 다르게 차례로 각줄을 얹어는 것이다.  
*Update UI의 흐름이 괜찮아!*

- 각각의 button과 nextStory에서 증가하는 storyNumber의

로그를 적어보았다.  
여기 updateUI가 호출의 안도니 문제나 생기는 것이 코드를 통해 확인되었다.

5

- ViewController이 타이머를 추가한 모습

- 앞서 한 project Quizzer에서 updateUI를  
어디까지 호출 했는지 살펴본다.

- `Quizzler`에서는 전문가에게 오답률을 누를 때마다 일시정지 버튼을 터치하면 시간을 멈춰 `Time`을 사용했는데, `Time`의 `property` 중 `Selectors`은 개별시간 설정하는 `timeInterval` 초기 후에 실행할 Function을 넣어된다.

그리고 보다자세히 `updateUI`를 넣고,  
Objective-C의 암시적 @objc까지 말해  
붙여 주니 `updateUI`가 정상적으로 반복호흡이 되었다.

- ReadMe에서 여러 Story가 있는 내용을 가져와  
접근시킨 모습이다.

struct 또는 class의 property가 적용된 모습을

볼 수 있는데, Story Number에 증가수를 적용해주는 것이 아닌,

Destination property  $\equiv$  StoryNumber on  $\sqsubseteq$  ChapterBook

더 깊은 편의 `start` 전략이 가능하다.  
만약 `if-else` 문으로 구현되는 `button`에 따라 `start` 메서드가 이동되는 기준의 `logic`을  
사용하면, 코드가 자세로 지나가는 것 뿐만 아니라 여러 가지 사용이 가능하다.

## - Model - Story Brain

```
mutating func nextStory(userChoice: String)
```

```
    if userChoice == stories[storyNumber].choice1 {  
        storyNumber = stories[storyNumber].choice1Destination  
    } else {  
        storyNumber = stories[storyNumber].choice2Destination  
    }  
}
```

- Next Story on Destination property  $\frac{2}{2}$

## 적용시킨 모습

# Destini - Code Review with Angela's

## • Angela - ViewController

```

@IBAction func choiceMade(_ sender: UIButton) {
    storyBrain.nextStory(userChoice: sender.currentTitle!)
    updateUI()
}

func updateUI() {
    storyLabel.text = storyBrain.getStoryTitle()
    choice1Button.setTitle(storyBrain.getChoice1(), for: .normal)
    choice2Button.setTitle(storyBrain.getChoice2(), for: .normal)
}

```

- Timer가 있다.  
또, 41 choiceMade에 있는 sender가 누른다거나 따라  
구분하는 if-else문도 있다.  
여기 사용한 버튼 choiceMade 아래의 sender 구분은 왜이거  
인가.

왜? → 그 기능은 이미 StoryBrain의  
nextStory에서 하고 있기 때문

## • updateUI() 흐름?

- Timer도 쓸 필요가 있다.
- updateUI는 function이다.
- 그냥 호출하면 되는 것이다.
- 만, button 클릭 시마다 내용이 반영되는 것이다.
- IBAction choiceMade에선 흐름은 해석된다.
- 이제 간단한 것이다.
- 금한 수록 간단한 방법부터 천천히 생각해보자.**

## • Angela - StoryBrain

```

mutating func nextStory(userChoice: String) {
    let currentStory = stories[storyNumber]
    if userChoice == currentStory.choice1 {
        storyNumber = currentStory.choice1Destination
    } else if userChoice == currentStory.choice2 {
        storyNumber = currentStory.choice2Destination
    }
}

```

## • Me - StoryBrain

```

mutating func nextStory(userChoice: String) {
    if userChoice == stories[storyNumber].choice1 {
        storyNumber = stories[storyNumber].choice1Destination
    } else {
        storyNumber = stories[storyNumber].choice2Destination
    }
}

```

- 크게 다른점은 없지만, 나는 상수나 변수 선언을  
습관화해야 할 것이다. Angela의 코드가 더 가독성이 있다.

\* 반복되는 코드는 상수, 변수 선언을 하여 효율적으로 코드하자.  
**Make variables or constants when it comes to Repetitive**