

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.

일러스트레이터 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.

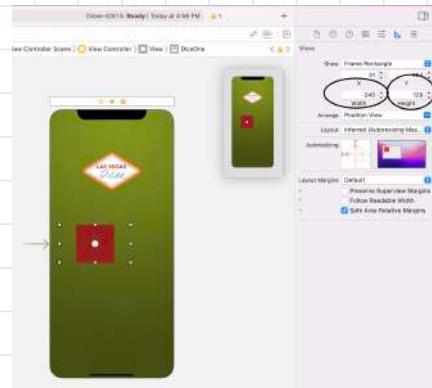
가끔 사이즈를 못찾아 가는 경우도 있는데 (제일 하단의 아이콘) 수동으로 적용시켜 주면 된다.

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: "image_name")  
    // 위처럼 이미지 불러오는 코드 :  
    // #imageLiteral( 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.
Replace 'var' with 'let'

.Var이 템플릿 주제에 맞지 않음
기본형은 정수여서, dieArray처럼 가능하지 않음
값을만 랙을 통해 사용 가능하지;

- 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, diceArray에서는  
Array의 randomElement()은 4가지 사용 가능하다
```

- Random(2)

Random Int

Random Bool

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

Bool.random()

Upper 이정도

Random Element from Array

Randomise Array

array.randomElement()

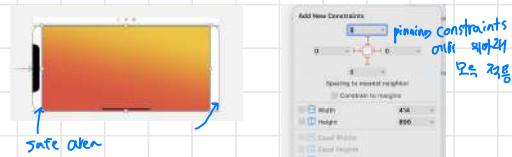
array.shuffle()

- Code exercise - randomisation

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

default value

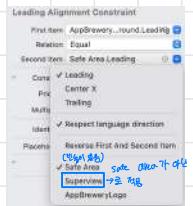
세로 가로 화면 상관 없이 빙글 없애기



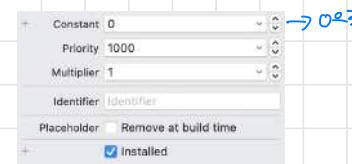
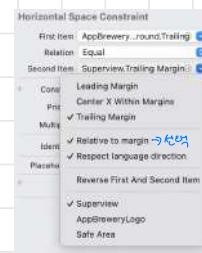
pinning constraints
only 허용
없는 경우

제공된 모습

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



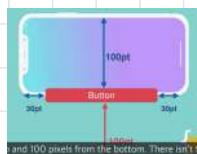
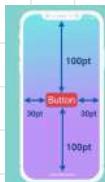
Margin



로고 가운데 배치시키기

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



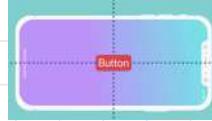
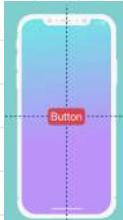


→ pinning Constraints

Pinning Constraint는 거리를 고정하는 방식입니다.

예를 들어 이미지뷰에 왼쪽면을 기준으로 50px의 Pinning Constraint를 설정한다면 세로뷰, 가로뷰, 디바이스 기준 등에 상관없이 항상 왼쪽면이 50px 만큼 멀어져 있게 됩니다.

Alignment Constraint는 말 그대로 정렬을 사용합니다. 예를 들어 버튼은 Horizontal center로 alignment constraint를 설정한다면 버튼은 어떤 상황에서도 항상 화면 왼쪽과 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



Calculator - auto layout

Goal →



1.



↳ 처음 모습

일단 시작위치는 단순히 Embedded View를 하지도 않았다.
이유는 잘 모르겠다.. View에 대해서 정확히 더 필요하다.
어쨌든 빠른 부분으로 Stack View를 만들어 주었다.

2.



↳ stack view 적용

모습이 조금 이상하지만 더 활용 시킬 것 같아
불렀다.

'D'을 Stack View 하지 않은 이유는
김별화하지만 Display 부분은 그전에 있던가 쭉 흐른다.

창고로 이 Stack View는 같은 row(가로)의 숫자가 기호로의 Stack View다.
이제 'D'을 포함한 각 row를 같은 Stack View를 적용 시킨다.

3.



→ 적용 시킨 모습
하단 세로 조절 필요 →



사이즈 조절?

내장한 UI 생략하면 된다.

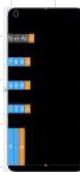
모든 요소들은 같은 Stack View

최종 전체의 Safe Area 까지 확장해 해야

나중에 각 숫자가 기호를 약간은 크기도 조절은

한 공간이 생기면, 다시 만드는 고통 없이 모든 예상에

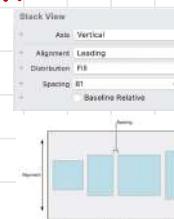
4.



→ pinning constraints 적용

문제점
1. row stack 간 크기
2. 숫자, 기호 크기

5.



oi Alignment & gap
Alignment: leading 아니 원쪽으로 최우선하고
Distribution: fill 아니

Alignment: leading 아니면 정렬은 확장
Distribution: fill 아니면 row stack은 view 안에는
차지하지 않고
Spacing이 8 아니면 raw stack은
간격이 넓은 것이다.

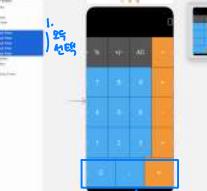
* 일정 복잡해 예상 설명이 잘 안된다.
공부 필요

6.



Alignment은 fill로 해줄으로서
Stack View 빼내 방향으로 경계를 대로 하지 않고 전체를 채움.
Distribution은 Fill Equally로 함으로서 row stack의 크기는
연평균이 됨.
Spacing → 1: 간격 증폭
이제, 각 요소들의 크기를 일정하게 해야 한다.
(맨 앞 0은 제외; 2칸 차지)

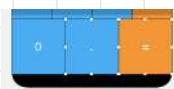
7.



↳ 아직 하단 부분 조작은 필요.



8.



또 다른 Stack View 생략해 피로 하다.

,=은單 Stack View로 만드는 이유?

↳ 목표 사진처럼 0이 제일 커야하고 (2칸 차지)

나머지는 위 모든 요소들이 크기가 동일하다.

그럼 마지막 row stack 만에 2개의 Stack으로 나누기

그즉 허리는 2칸은 차지하고 (0), 나머지 하나는 또 두개의 요소로

나누니 그때 크기 조절은 하면 된다.

..최상의 Stack 안에 두개의 Stack으로 나누어 2칸씩 쓰기 훈련,

, 그나 원하는 Stack은 Fill Equally로 통해 다시 반으로 나눈다

;; stack view 적용



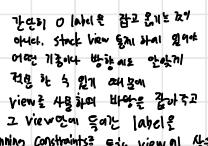
;; fill equally 적용



9.

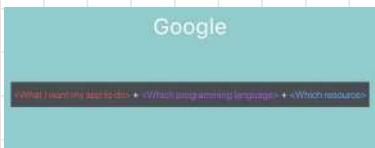
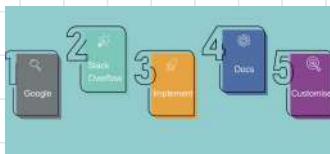


먼저 0 선택 원쪽으로
움직이기.

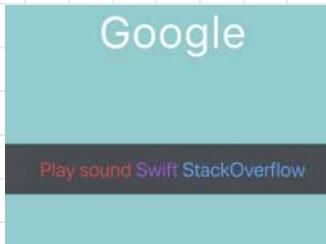


간단히 0 뒤에 온 걸고 옮기는 2번
이다. Stack View를 옮겨 차례로 옮겨
이면 가능하다. 방향대로 옮겼다가
전환하는데 번거워 예전에
View를 4개 헤더에 바꿔놓을 걸까하고
그 View는 두개는 아래로 옮기고
pinning constraints를 통해 View의 상용하는
Constraints를 찾았더니 멀리 끌어내려 망쪽에
입점 강제로 더운 배출 가능하다.

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 keyPress(_ sender: UIButton) {
        playSound()
    }

    func playSound() {
        guard let url = Bundle.main.url(forResource: "C", withExtension: "mp3") 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()

        let rating: Int = 1
        print("Rating: \(rating) \(sender.titleLabel?.text ?? "")")
    }
}
```

Copied codes



StackOverflow에서 코딩으로

제작이 된 답변이나 해석 항목은 있는 것은 아니다.
사실상 앱과 버전 차이가 있기 때문에 일부 코드는 잘
수록 보여 준다.

2. Instance Method

setCategory(_:)

Sets the audio session's category.

AVAudioSession에서 예상 내려온 Session에서
사용할 수 있는 모드들이 Topics or 게시판에 있다.
그중 setCategory는 찾았던 글의 (Share, Initiate!)는
Session 접근 방식에 나와있음

4.

Topics

Audio Session Categories

Topics for audio | Apple Developer Documentation

The category for use in which sound playback is expected—either to your own user or with the sound itself.

Topics for playback | Apple Developer Documentation

The category for sending direct streams of audio to other devices at the same time.

Topics for playback | Apple Developer Documentation

The category for recording direct and playback instead of audio such as for a video camera.

Topics for playback | Apple Developer Documentation

The category for implementing file reads or other sources that are internal to the recorded use of your app.

이제 우리가 찾던 Playback이 나왔다.

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 모든 타입 선택해

적이면 되는 것 같다. 즉, 기본.

• 또한 option 키를 누른 채로 코드 위에

갖다 대입 전부는 정의된 볼 수 있다

• Apple Developer Docs를 적극 활용하면

간 코드들의 기능과 예상은 알기 어렵고, 전체적인
흐름을 보는데에 도움을 준다

Linking Multiple Buttons to the Same IBAction

1.



일단 C 코드는 IBAction으로 연결 했다.
다른 코드들을 이용해 별도로 연결해도 되는 것은
매우 비효율적이다, 그럼 어떻게 할까?

2.



3.

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

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

func playSound() {
    let ucl = Bundle.main.url(forResource: "C", withExtension: "mp3")
    player = try! AVAudioPlayer(contentsOf: ucl!)
    player.play()
}

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

5. challenge

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

```
print(sender.titleLabel)
```



Optional<UILabel> Optional<UILabel> 0x7fd2ba0ff9d8: frame = (188 31; 28 48); text = "C"; alpha = 1
Button의 title이 경사로로 나와는 빛지 않아,
필요없는 다른 정보들 속에 걸쳐 나와서 이것을 통해 저대로
Button의 title을 추적하여 그를 찾는데 효과적일까 모르겠다.



C button의 title을 없애고
nil 역시 "nil"을 print하기 했어야
실제로 nil이 print되었다.
그럼 title 앞 (for: .normal)은 무었인가?

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

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

다음은 stack overflow이다. 경색한 결과이다.
if let은 optional Binding의 한 종류이다.
Sender 즉, button의 title이 nil 값일 수 있고 있으니
사용하는 걸로 생각된다. (혹은 공부 필요)



→ 코드의 원형은 뷰 Control(IBOutlet Button)의
state(상태)를 표시하고,

```
print(sender.title(for: .normal))  
이 애 진짜 없어서 .normal이지 같은가  
생각된다. (왜 그냥 .normal 인지는 모르겠다)
```

그 외 방법은

```
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, UIButton의 opacity(투명도)를 50%로
        playSound(soundName: sender.currentTitle ?? "C")①
        // I : Don't worry we've already checked
        // playSound(soundName: sender.currentTitle!)
        ② DispatchQueue.main.asyncAfter(deadline: .now() + 0.2) {
            print("start")
            print("end")
            sender.alpha = 1.0
        }
    }
    func playSound(soundName: String) {
        let url = Bundle.main.url(forResource: soundName, withExtension: "wav")
        player = try! AVAudioPlayer(contentsOf: url!)
        player.play()
    }
}

```

```

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

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

```

```

// var playerUsername: String = nil
var playerUsername: String? nil
playerUsername = "jackbauerisawesome"
    ↪ 즉시 터트려 48X
// safety check 토해 보고 싶어
var unwrappedUsername = playerUsername

playerUsername = nil // > crash
print(playerUsername!) // safety check를 해야만 unwrapping을 하겠는데 값이 없어서 crash
// how to prevent this happening?

if playerUsername != nil {
    print(playerUsername!)
} else {
    print("nil")
}

```

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

② stackoverflow에서 가져온 것인데 Delay를 주는 코드야.

*로 풀어 봄에 있어 있는 코드들은 실행시켜 주는 것이다.
Sender.Alpha를 기본값으로 넘기 때문에 대신 유저 터치로
드러워 Button이 누리는 효과를 극대화 시킨다.

Egg Timer

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

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

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

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

이건 내가 작성한 코드입니다, eggTimes dictionary에서 sender.currentTitle로 들어온 값으로 value를 추출해 print를 하는 것이다. 하지만 내가 너무 어렵게 생각했나보다.

위에 보이는 것처럼 어짜피 currentTitle은 eggTimes의 value이므로 들어오기 때문에 바로 eggTimes의 key값을 냇아줘 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 of our eggTime dictionary.

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

Dictionary는 string data type으로 된 key를 가지고 있다.

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



```
let hardness = sender.currentTitle!
let result = eggTimes[hardness]!
print(result)
print(eggTimes[hardness]) ⚠️ Expression
```

```
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: Int?
    var timer: Timer?

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

    @IBAction func hardnessSelected(_ sender: UIButton) {
        // I : 强制解包(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 - 몇초간의?
Repeats - true를 택할 timeInterval로 뒤에도 계속 한다.

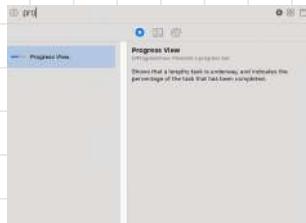
Selector - object의 인라이브로 ()안에 넣은 function을 설정할 때도 @objc를 넣어서 생성해준다.

(timer.invalidate() -> 타이머 초기화) : 타이머가 놀랄때마다 초기화를 하지 않는다면 다른 버튼을 누를때 처음 시작 하는 바꿔지만 그때로 초기화된다.

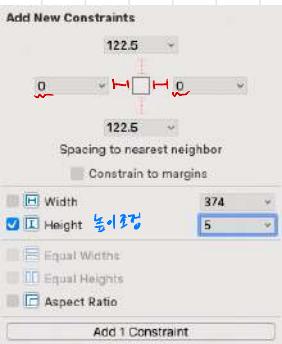
Timer는 잘 작동한다.

하지만 최종 목표인 progressbar와 연동하기 위해서

몇가지 작업이 더 필요하다.



Barिंडी
(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()
    }

    // ! : 뒤집고 확장(forced-upwrapping)
    let hardness = sender.currentTitle!
    result = 0.0
    totalSeconds = Float(eggTimes[hardness]!)
    print("\(hardness) got selected")
    timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector(updateTimer), userInfo: nil, repeats: true)
}

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

```

Stackoverflow 예제

result : 초기에 1씩 증가한다. progressbar의 진행상황을
시각화한 변수

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

progressBar.setProgress: stackoverflow와 같은 에러.

처음에는 H_2O_2 에 대해, 두 번째엔 Br_2 에接触을 넓혀온다.

`result / totalSeconds` 은 현재 초 / 전체 초 로 `updateTimer()` 한다.

이어 다음에 언급하는 그 이후 매우 심해 되니 *progressbar*의

지금 사는 곳 이하하단

리셋을 해주지 않았더니 다 채워진 box가 다른 egg²로 긁적⁴되니
두루 갑다가 앞으로 가는 현상이 발생하였다.

forked - unmappping
 sess())
 on forked
 self selector
 seconds
 interval: 1, target: self, selector:
 : nil, repeats: True
 true not selector
 false self
 false : 116 248

Time class function

Initialization
The constructor of the class takes the following parameters:
- `host`: string - host, where the application will be deployed.
- `port`: integer - port, where the application will be deployed.
- `username`: string - login for the database.
- `password`: string - password for the database.

Deployment
After the application has started, the user can send the message via `Socket` to the server:
For example:
The number of incoming bytes: length of the file. It is possible to receive a file in 4 KB blocks, then the message `length` is sent.
target
The object to which we want to send the message specified by a string. The string must be unique, otherwise the message will be ignored.
addListener
The message to which we want to add the listener specified by a string. The string must be unique, otherwise the message will be ignored.
receive
Method. Returns a `String`, which indicates that the method was called successfully. If the method was not called successfully, it will return the error message.
receiveFile
This method can be used to read the file. This method returns a `String`, which contains the file content. The parameter `path` is the path to the file, from which the file will be read.
receiveFileWithListener
This method is similar to the previous one, but it is possible to add a listener to the file. If the file is received successfully, the method will return the message `OK`.
receiveFileWithTarget
This method can be used to read the file. This method returns a `String`, which contains the file content. The parameter `path` is the path to the file, from which the file will be read. The parameter `target` is the target to which the file will be sent. If the file is received successfully, the method will return the message `OK`.

• Udemy Logic

```
import UIKit
import AVFoundation

class ViewController: UIViewController {

    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var progressbar: UIProgressView!
    let eggLines = ["Soft": 3, "Medium": 4, "Hard": 7]
    var timer: Timer?
    var totalTime = 0
    var secondsPassed = 0
    var player: AVAudioPlayer?

    @IBAction func hardnessSelected(_ sender: UIButton) {
        timer?.invalidate()
        titleLabel.text = "How do you like your eggs?"
        if let button = sender as UIButton {
            let hardness = button.currentTitle!
            totalTime = eggLines[hardness]!
            print("hardness: \(hardness)")
            progressbar.progress = 0.0
            secondsPassed = 0
            titleLabel.text = hardness
            timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector(timerUpdate), userInfo: nil, repeats: true)
        }
    }

    @objc func updateTimer() {
        if secondsPassed < totalTime {
            progressbar.progress = Float(secondsPassed) / Float(totalTime)
            secondsPassed += 1
        } else {
            titleLabel.text = "DONE!"
            player?.stop()
            player?.removeFromSuperview()
        }
    }

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

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

Using the 5 Step Approach to Debug our App

```
#IBDesignable weak var titleLabel: UILabel!
@IBDesignable weak var progressbar: UIProgressView!
let eggs = ["Soft", "Medium", "Hard"]
var timer = Timer()
var totalTime = 0
var secondsPassed = 0
var selectedEggIndex = 0
@IBAction func normalizeSelected(_ sender: UIButton) {
    timer.invalidate()
    titleLabel.text = "How do you like your eggs?"
    // You can see the progressview
    let percentageProgress = secondsPassed / totalTime
    progressbar.setProgress(Float(percentageProgress))
    secondsPassed += 1
}
@IBAction func start(_ sender: UIButton) {
    timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector(timerUpdate), userInfo: nil, repeats: true)
}
@objc func timerUpdate() {
    if secondsPassed > totalTime {
        let percentageProgress = secondsPassed / totalTime
        progressbar.setProgress(Float(percentageProgress))
        secondsPassed += 1
    }
    else {
        titleLabel.text = "Time"
        timer.invalidate()
    }
}
```



→ progress bar 가 증가하지 않는
상황 발생



What did you expect
your code to do?

- 초기 기능에 따라 progressbar 증가하는 것



What happened
instead?

- progressbar 비동작



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 progressbar 가 증가해야 한다.



How can we test the things
our expectations depend on?

- 로그 찍어보기

```
let percentageProgress = secondsPassed / totalTime
print("percentageProgress")
print(secondsPassed)
print(totalTime)
print(Float(percentageProgress))
```

↳ 0.0 이 나옴

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

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

- 결과 같은 그림처럼 2.5를 나온다.

→ 만약 2를 나온값 (2)을 그냥 float
형식으로 꺼내 놓여진 것 앤.
(print (Float(1)))

해결 방법

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

```
8.8
8.8
0.33333334
0.0
0.66666667
0.8
```



→ 만약 Float으로 변환을 한 뒤,

계산을 시작함.

Float이 제대로 나오고 progressbar로
증가하지만 100%가 되지 않는 암울함

내가 써놓은 코드

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

secondsPassed += 1
// secondsPassed의 Maximum을 높여놓으니
// totalTime은, progressbar는 100%가 되도록 있게 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 때서 마지막 progress 초기화
    정지되거나 원하는 증가수가 되잖아.
    만약 secondsPassed가 이미 있다면 증가를 했을 때도
    뒤그하고 totalTime과 같은 증가 회수와 사용할
    수 있겠지.
```

Swift Deep Dive - Structures, Method and properties

Defining the Structure

```
struct MyStruct{ }
```

Initialising the Structure

```
MyStruct()
```

Struct (구조체)

```
struct Town{
    let name = "HoontLand"
    var citizens = ["Hoon", "Jack Bauer"]
    var resources = ["Grain":100, "Ore": 42, "Wool": 76]

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

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

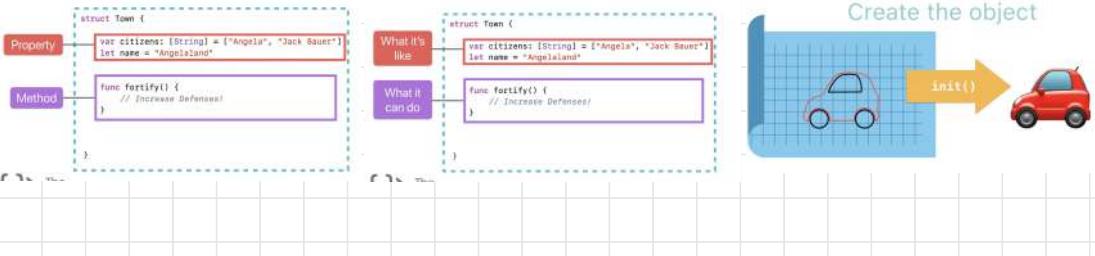
// able to access to properties of Town
print(myTown.citizens)
print("\(myTown.name) has \(myTown.resources["Grain"]!) bags of grain.")

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

// method
myTown.fortify()
```

한국어 번역

Create the object



Creating the initializer

init()으로 생성자 초기화
Object는 만들 때 초기화
할 수 있다.
왜 init() 필요 한가?

Using the initializer

```
StructureName()
```



D를 개발할 때도 둘째 머리가운도는 결론이다.
하는 데 각각은 외 인구수, 통장 등 다른점에 차이가
많다.

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

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

그러면 새로운 이름을 만들 때마다
Struct은 어떤 점은 같다 하니
원래 있는 Struct를 blueprint(모형)
쓰면 유용하게 사용된다.

```

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]
)

```

캐비어한 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 이름의 Struct의 property에 접근할 때 self.을

붙여야 한다

```

func exercise() {

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

    // Initialize 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: 2000, 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()

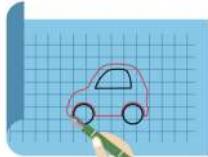
}

exercise()

```

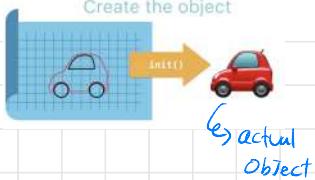
Swift Deep Dive - Immutability

Struct = Blueprint

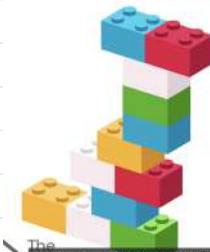


```
struct QuizBrain {
    property var questionNumber = 0 ↳ what it's like associated with struct
    method func checkAnswer(_ userAnswer: String) -> Bool {
        if userAnswer == quiz[questionNumber].answer {
            return true
        } else {
            return false
        }
    } ↳ what it can do
}
```

Create the object

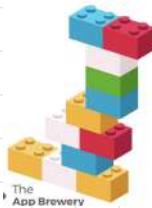


Immutability



*↳ if of 2nd let은
이전의 값을 몰라
변경할 때 → immutable*

Destroy



Rebuild

*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
    }
}
```

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

self라는 것은 인자인 걸까?

- self가 쓸 코드의 resources 앞에는 서야 self.이

생략하기 위해,

그냥 error message 온 'self' is immutable 오류!

나타나.

"self"는 자동으로 사용되는 property이기 때문에 더러운,

자신이 self를 뒤에 붙인 것 같아.

mutating은 꼭! 주의!

Error는 사용금지.

mutating을 끝으로서 self를 variables 쓸 수 있는 것이다.

The methods of struct

Plain method

: doesn't change
anything

Mutating method

: can change the state of
structure

```
myTown = Town(citizens: ["Angela", "Jack Bauer"], name: "Angelaland", resources: ["Wool": 75])  
myTown.citizens.append("Keanu Reeves")  
print("People of \u2028(myTown.name): \u2028(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 퀴즈가 나타나며

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

```
class ViewController: UIViewController {
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var progressView: UIProgressView!
    @IBOutlet weak var trueButton: UIButton!
    @IBOutlet weak var falseButton: UIButton!
    let size = 10
    let quiz = [
        "Four + Two 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()
    }
    // MARK: Button Tap
    @IBAction func answerButtonPressed(_ sender: UIButton) {
        let userAnswer = sender.currentTitle! == "True" ? true : false
        let actualAnswer = quiz[questionNumber]!.answer
        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:

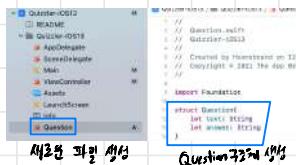
2D 배열 (2차원 배열)로 만든 Question & Answer를 이용해
array에 따라 Question text를 뜨우고 점을 표시한 것이다.

questionNumber는 quiz 배열의 위치로 도달할 때까지
answer 배열에 따라 퀴즈에 현재 몇 번째 (question)에
있는지 알 수 있고 그 Int값은 Quiz 배열에 삽입해
문제를 표시하는 것이다.

Quiz 배열이 지정된 채 놓이고 편리한 걸 알았기 때문에

세로운 파일에서 struct를 만든 것이다.

* 처음 같은 불리는 데도 빙거운 불이 있다.



서로운 파일 만들기

Question.swift

```
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")
]
```

퀴즈의 property를 사용해 정답은 오른

```
let actualQuestion = quiz[questionNumber]
let actualAnswer = actualQuestion.answer
// 실제 actualQuestion의 용도는 모른다.
```

struct를 정리를 했지.
하지만 초기 가지고 있는 Quiz의
index 높여놓은 넘기면 Crash 된다. 때문에
Safety Check을 해야 한다



.4) Safety Check

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

Udemy's safety check

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

• 대체적으로 비슷해 보인다.

마지막 부분은 if로 두고 다음은 두거나의 차이다.
Udemy 책이 가능성이 미리하게 좋은 것 같다.



```

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

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

```

User Answer과 Actual Answer가 같은 경우
 Button의 background color가 바뀌는 결과를
 확인하고자 한다. updateUI에서 발생한 문제는
 단지 초기화된 상태에서 그 결과가 바로 적용
 되어야 하며, 대신,



• Heej Solution

```

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

```

-은 Xylophone project에서 간접 탐색
 일시적인 효과를 주 DispatchQueue를
 이용하여 0.2초 후에 실제 내용을 처리하게
 하였다. 즉, 저 Method 안에서 내용이 적용
 x 초기 단계로 일정을 적용하였다.
 (물론 위에서 정답이나 오류는 별개)
 색깔 효과를 주기 위해서

Angelaa's Solution

```

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

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

    if questionLabel.text == quizQuestions[questionNumber].text {
        questionLabel.text = quizQuestions[questionNumber + 1].text
    }
}

func updateUI() {
    DispatchQueue.main.async {
        self.updateUI()
    }
}

```

Angelaa는 egTimer에 0.2 Timer를 사용해
 적용하였다.

timeInterval을 0.2로 updateUI function이 0.2로
 실행되면서 그에 background color를 초기화해주고
 다음 question으로 넘긴다. 즉, 한 번당 일정에는 일정이
 설정이 repeat은 false이다.

X selector는 원래 있는 updateUI Method를 이용한다.
 그 이유는 우리가 정한 interval delay 내용은 Method를
 넣어주면 되니까 굳이 새로 만들 필요없이 기존의 Method를
 이용한 것이고, 앞서 정한 Objective-C의 방법이기 때문이다.
 ① ObjC를 알아 봄이 중요하다.



• progress bar

```

Aspect UIKit
class ViewController: UIViewController {
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var trueButton: UIButton!
    @IBOutlet weak var falseButton: UIButton!
    @IBOutlet weak var resultLabel: UILabel!
    @IBOutlet weak var progressBar: UIProgressView!
}

let quiz = [
    Question("A slug's blood is green.", at: "True",
        QuestionText: "A slug's blood is green.", is: "True"),
    Question("The surface area of two human lungs is approximately 70 square meters.", at: "True",
        QuestionText: "The surface area of two human lungs is approximately 70 square meters.", is: "True"),
    Question("In West Virginia, USA, if you accidentally hit an animal with your car, you must stop and help it to safety.", at: "True",
        QuestionText: "In West Virginia, USA, if you happen to drive by some house of an animal and accidentally smacking the animal, you must stop and help it to safety.", is: "True"),
    Question("It is illegal to pee in the Ocean in Portugal.", at: "True",
        QuestionText: "It is illegal to pee in the Ocean in Portugal.", is: "True"),
    Question("You can lead a cow down stairs but not up stairs.", at: "False",
        QuestionText: "You can lead a cow down stairs but not up stairs.", is: "False"),
    Question("Bees are originally yellow." at: "False",
        QuestionText: "Bees are originally yellow.", is: "False"),
    Question("A full moon is never seen in the day time.", at: "False",
        QuestionText: "A full moon is never seen in the day time.", is: "False"),
    Question("The loudest sound produced by any animal is 188 decibels.", at: "False",
        QuestionText: "The loudest sound produced by any animal is 188 decibels.", is: "False"),
    Question("Chocolate affects a dog's heart and nervous system; a few ounces are enough to kill a small dog.", at: "True",
        QuestionText: "Chocolate affects a dog's heart and nervous system; a few ounces are enough to kill a small dog.", is: "True")
]

var questionNumber = 0

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

@IBAction func answerButtonTapped(_ sender: UIButton) {
    let userAnswer = sender.currentTitle // True, False
    let actualAnswer = quiz[questionNumber].is
    let actualAnswer = actualAnswer == userAnswer ? "Correct" : "Incorrect"
    if userAnswer == actualAnswer {
        sender.backgroundColor = UIColor.green
    } else {
        sender.backgroundColor = UIColor.red
    }

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

Timer.scheduledTimer(timersInterval: 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 = Double(questionNumber)/Float(quiz.count)
}

```

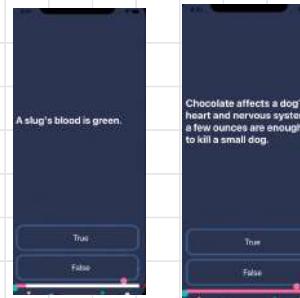
이제 막 문제 개수 5개 까지
Progress 2개

다음은 7개 문제
Progress 3개

— 이번이 EggTimer가 이용 두번째 progressbar 응용인데,
내가 느낀건 progressbar는 굉장히 간단한 수 있으나,
현재 진행상황 / 정답의 양 or 개수 등 두 가지에 대응 가능해진다.
아래가 뜰 모 하다는 것이다.

이번에는 현재 QuestionNumber / 전체 Question 개수로
간단 했다.
문제를 맞추면 투리고 Timer의 정답 Interval로
일어나는 updateUI의 코드를 작성해 progressbar가
상승 하게 하면 된다.

• 결과

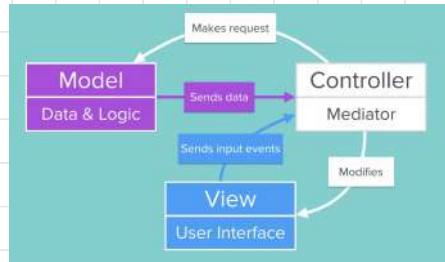
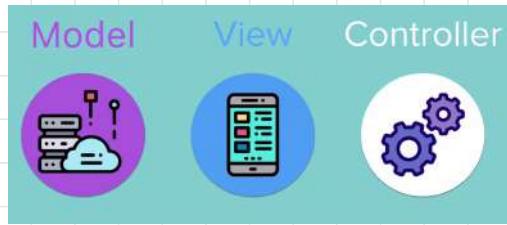


Quizzer - MVC pattern

현재 Quizzer의 코드는 하나의 파일에만 위치하며 작동한다.

그렇게 되면 가독성이 떨어질 뿐만 아니라 유지보수하기 힘든다.

그러니 사용되는 design pattern들 중 하나인 **MVC pattern**을 사용해 보자.



• M - Model

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

- Model의 구조

1. 사용자가 편집하기 원하는 모든 data를 가지고 있어야 한다.

즉, 화면 안의 네모박스에 글자를 표시된다면, 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

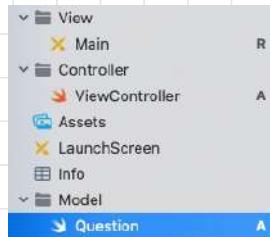
data 변경이 일어났을 때 Model에서 화면 UI를 직접 조정해서 수정할 수 있도록 View를 참조하는 내부 속성값을 가지면 안된다.

3. 변경이 일어나면, 변경 통지에 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 변경하지 않아야 한다.

Quizler의 Model

~~Quizler의 Model~~



↳ Quizler의 Model은

Question: 문제와 정답 정보를 저장하는 구조체,

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



```
//# parameter 외에 다른 parameter 이름을 넣거나 _ 를 넣을 수 있다. _ 는 외부에서 프로퍼티의 아무것도 만들거나 아니면 아무거나 넣고 할 수 있다.
//# _ : parameter // external parameter name이라고 하고 이 레스Than 명에서 쓰는 parameter name이고 checkAnswer 메소드 안에서 쓰이는 걸
internal parameter name이 된다.
```

```
var questionNumber = 0
```

```
// _ : external parameter name, userAnswer : internal parameter name
func checkAnswer(_ userAnswer: String) {
    if userAnswer == quiz[questionNumber].text {
        print("Correct!")
    } else {
        print("Incorrect!")
    }
}
```

- ViewController() 있던 변수와 method는

QuizBrain struct에 가지고 온 모습.



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

- 보내자면 checkAnswer에 userAnswer가
넣었으나, 하지만 optional 오류가 발생할수는 있어
그 이유는 userAnswer는 optional String이기
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
    }
}
```

```
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 저장해서는 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 알려 주어야 하는 방법을 구현해야 한다.
View에서는 화면에서 사용자가 화면에 표시된 내용을 변경하기
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 규칙들이 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 collectionView에
challenge 버전으로 처리 한다.

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

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

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

    // 설정적인 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)
        }
        ↳; 각각 button의 title을 question에 맞게 변경
    }
}
```

① 버튼 3개에 같은 선택지를 어떤 방법으로 넣을까 하니까
Array를 이용하게 됐다

② View가 load되는 즉시 배열의
button을 설정한다.

③ button의 초기값.

```

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
    }
}

```

→ Model
- Question

```

import Foundation
struct QuizBrain{
    var quiz: [Question]
    var score: Int = 0

    // An array of internal question type, questioner : internal parameter name
    // and also has overriden, questioner: string) => void {
    // questioner: internal parameter name, used in this method.
    // If userAnswer == quizquestionnumber.correctAnswer
    // return true
    // else {
    // return false
    // }

    func getScore() -> Int{
        return score
    }

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

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

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

```

→ 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 initializer 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 띄어 둘까.
나의 correctAnswer 보다 rightAnswer CT 가정한 이유 같고
self는 쓸 것 같아 없애보자

Model 1 - QuizBrain

Udemy

```

import Foundation

struct Question {
    var questionNumber = 0
    var score = 0
}

let quiz = [
    Question(id: "Q1", question: "Which is the largest organ in the human body?", answers: ["Heart", "Skin", "Large Intestine"], correctAnswer: "Skin"),
    Question(id: "Q2", question: "What is the capital of France?", answers: ["Paris", "London", "Berlin"], correctAnswer: "Paris"),
    Question(id: "Q3", question: "What is the letter in the 9th line name stand for?", answers: ["General Aviation Time", "General Marine Time"], correctAnswer: "General"),
    Question(id: "Q4", question: "The French word for 'heart', as ["Coeur", "Cœur"], correctAnswer: "Cœur"),
    Question(id: "Q5", question: "In past times, what would a gentleman never do to his pocket?", answers: ["Turn it inside out", "Turn it outside in"], correctAnswer: "Turn it outside in"),
    Question(id: "Q6", question: "What is the French word for 'water', as ["Eau", "Wasser", "Wasser"], correctAnswer: "Eau"),
    Question(id: "Q7", question: "What is the English word for 'water', as ["Water", "Wasser", "Wasser"], correctAnswer: "Water"),
    Question(id: "Q8", question: "What colour is NOT featured in the logo for Google?", answers: ["Green", "Orange", "Yellow"], correctAnswer: "Orange"),
    Question(id: "Q9", question: "What is the Australian name for kangaroo?", answers: ["Kangaroo", "Kangaroos", "Kangaroos"], correctAnswer: "Kangaroo"),
    Question(id: "Q10", question: "What is the German word for 'Australia', as ["Deutschland", "Australien", "Australien"], correctAnswer: "Australien")
]

var questionNumber = 0
var correctAnswersCount = 0
var totalScore = 0

func getQuestion() -> Question? {
    if questionNumber < quiz.count {
        return quiz[questionNumber]
    } else {
        return nil
    }
}

func calculateScore() {
    if questionNumber + 1 < quiz.count {
        questionNumber += 1
        correctAnswersCount += 1
    } else {
        return false
    }
}

 mutating func checkCorrectness(answer: String) -> Bool {
    if answer == quiz[questionNumber].correctAnswer {
        quiz[questionNumber].score += 1
        return true
    } else {
        return false
    }
}

```

→ d 별은 놓쳤다...
 Answer 뒤에 return 입니다.
 Method는 Model이 아닙니다.
 주석은 뺀 후에 풀입니다.

Me

```

import Foundation

struct QuizBrain {
    // Q1 Q2 ... Q10
    var quiz = [
        Question(id: "Q1", question: "What is the largest organ in the human body?", answers: ["Heart", "Skin", "Large Intestine"], correctAnswer: "Skin"),
        Question(id: "Q2", question: "What is the capital of France?", answers: ["Paris", "London", "Berlin"], correctAnswer: "Paris"),
        Question(id: "Q3", question: "What is the letter in the 9th line name stand for?", answers: ["General Aviation Time", "General Marine Time"], correctAnswer: "General"),
        Question(id: "Q4", question: "The French word for 'heart', as ["Coeur", "Cœur"], correctAnswer: "Cœur"),
        Question(id: "Q5", question: "In past times, what would a gentleman never do to his pocket?", answers: ["Turn it inside out", "Turn it outside in"], correctAnswer: "Turn it outside in"),
        Question(id: "Q6", question: "What is the French word for 'water', as ["Eau", "Wasser", "Wasser"], correctAnswer: "Eau"),
        Question(id: "Q7", question: "What is the English word for 'water', as ["Water", "Wasser", "Wasser"], correctAnswer: "Water"),
        Question(id: "Q8", question: "What colour is NOT featured in the logo for Google?", answers: ["Green", "Orange", "Yellow"], correctAnswer: "Orange"),
        Question(id: "Q9", question: "What is the German word for 'Australia', as ["Deutschland", "Australien", "Australien"], correctAnswer: "Australien"),
        Question(id: "Q10", question: "What is the German word for 'Australia', as ["Deutschland", "Australien", "Australien"], correctAnswer: "Australien")
    ]
}

var questionNumber = 0
var score = 0

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

func getQuestion() -> Question? {
    if questionNumber + 1 < quiz.count {
        questionNumber += 1
        return quiz[questionNumber]
    } else {
        return nil
    }
}

 mutating func calculateScore() {
    if questionNumber + 1 < quiz.count {
        questionNumber += 1
        score += 1
    } else {
        return false
    }
}

 func getQuizScore() -> String {
    return quiz[questionNumber].score
}

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

```

• Controller - View Controller

```

@IBOutlet weak var questionLabel: UILabel!
@IBOutlet weak var progressLabel: UILabel!
//Add another button and a corresponding outlet.
@IBOutlet weak var thridButton: UIButton!
@IBOutlet weak var scoreLabel: UILabel!
@IBOutlet weak var thridLabel: UIButton!
@IBOutlet weak var correctLabel: UILabel!
//New button needs to be linked to this IBAction too.
@IBAction func answerButtonPressed(_ sender: UIButton) {
    let userAnswer = sender.currentTitle!
    let userGetRight = quizBrain.getQuestionUserAnswer(userAnswer)
    if userGetRight {
        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 return the answers and update the button titles using the setTitle method.
    let answerButtons = quizBrain.getAnswers()
    choice1.setTitle(answerButtons[0], for: .normal)
    choice1.setTitle(answerButtons[1], for: .normal)
    choice2.setTitle(answerButtons[2], for: .normal)
    choice3.setTitle(answerButtons[3], for: .normal)
    progressLabel.progress = quizBrain.getProgress()
    scoreLabel.text = "Score: \(quizBrain.getScore())"
    choice1.backgroundColor = UIColor.clear
    choice2.backgroundColor = UIColor.clear
    choice3.backgroundColor = UIColor.clear
}

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

```

→ 022 Ako? 3
만화로 알기.

```

class ViewController: UIViewController {
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var progressLabel: UILabel!
    @IBOutlet weak var firstButton: UIButton!
    @IBOutlet weak var secondButton: UIButton!
    @IBOutlet weak var thirdButton: UIButton!
    @IBOutlet weak var scoreLabel: UILabel!
    @IBOutlet weak var buttonArray: [UIButton]!
    var quizBrain: QuizBrain!
    override func viewDidLoad() {
        super.viewDidLoad()
        buttonArray = [firstButton, secondButton, thirdButton]
        quizBrain = QuizBrain()
        let buttonArray: [UIButton] = [firstButton] // 초기화 버튼을 사용해 전 쪽에
        var quizBrain: QuizBrain()
        override func viewDidload() {
            super.viewDidLoad()
            buttonArray = [firstButton, secondButton, thirdButton]
            quizBrain = QuizBrain()
        }

        // 다음으로 넘어가기 버튼
        @IBAction func answerButtonPressed(_ sender: UIButton) {
            let userAnswer = sender.currentTitle! // True, False
            let userGetRight = quizBrain.checkAnswers(userAnswer)
            if userGetRight == 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() {
            updateQuestion()
            questionLabel.text = quizBrain.getQuestionText()
            progressLabel.progress = quizBrain.getProgress()
            scoreLabel.text = "Score: \(quizBrain.getScore())"
            clearButton()
        }

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

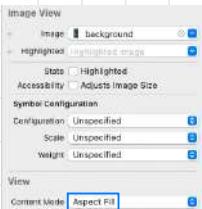
        func updateAnswerButton() {
            for i in 0...buttonArray.count {
                buttonArray[i].setTitle(quizBrain.quizBrain.operationNumber1.answer[i], for: .normal)
            }
        }
    }
}

```

→ Model의 역할은 완전히
Controller에서 처리된다.

Destini

1. Background



- stackView setting

• Distribution: Fill Proportionally
 비율을 맞춰 StackView를 채워주는 것.
 Stack View OK는 유연성이 따라 사용 가능한
 공간을 활용다. (View's intrinsic content size)
 • spacing: 20
 간격

4. Apply MVC

• Model - Story

```

import Foundation

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

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

• Story 구조체를 만들어 사용하게
필요한 data format 정의 담고 있다.

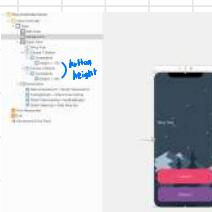
• Model - StoryBrain

```

enum StoryBrain {
  case story1
  case story2
  case story3
  case story4
  case story5
  case story6
}
  
```

- StoryBrain 구조체에 Story를 이용해 Stories Analysis를 만들었다.

2. View & constraints



• button의 간격은 저정도로 설정 하길 원했지만, Stack View는
 적용 시기고 button 사이 간격이
 조작이 안되서 다시 unembed
 하고 조작을 다시 stack view
 를 만들었다.
 * StackView 상태에서도 조작
 가능성이 있음을.



3. IBoutlet - button, label

```

@IBOutlet weak var storyText: UILabel!
@IBOutlet weak var choiceButtons: UIButton!
@IBOutlet weak var choice2Buttons: UIButton!

let story1 = "You see a rock in the road."
let choice1 = "Take a left."
let choice2 = "Take a right."

override func viewDidLoad() {
  storyText.text = story1
  choiceButtons.setTitle(choice1, for: .normal)
  choice2Buttons.setTitle(choice2, for: .normal)
  super.viewDidLoad()
}
  
```

• button과 label은 연결해 테스트 해보자.

Controller - View Controller

```

import UIKit

class ViewController: UIViewController {
  @IBOutlet weak var storyText: UILabel!
  @IBOutlet weak var choiceButtons: UIButton!
  @IBOutlet weak var choice2Buttons: UIButton!

  var storyBrain = StoryBrain()

  override func viewDidLoad() {
    storyText.text = storyBrain.story1
    choiceButtons.setTitle(storyBrain.choice1, for: .normal)
    choice2Buttons.setTitle(storyBrain.choice2, for: .normal)
    super.viewDidLoad()
  }
}
  
```

- StoryBrain 구조체 이용해 View와 Model에 연결시키는 logic을 구현했다.



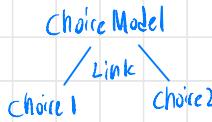
5. IBAction



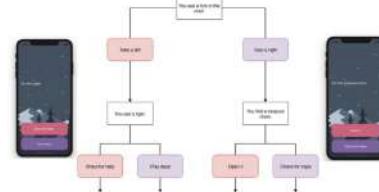
```
IBAction func choiceModel(sender: UIButton) {
    if sender == choice1Button {
        storyText.text = storyBrain.stories[1].title
        choice1Button.setTitle(storyBrain.stories[1].choice1, for: .normal)
        choice2Button.setTitle(storyBrain.stories[1].choice2, for: .normal)
    } else {
        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 선택지가 나오게 한다.

↳



· 모식도 (diagram)



6. Model (StoryBrain) - Next Story function

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

```
func nextStory() {
    guard storyNumber < stories.count else {
        return
    }
    let story = stories[storyNumber]
    self.storyNumber += 1
    self.storyText.text = story.title
    self.choice1Text.text = story.choice1
    self.choice2Text.text = story.choice2
}
```

View - Viewcontroller



- 다음 스토리로 넘어가기 위해 StoryBrain에서 새로운 method를 만들었다.

처음에 .count 셀프인 애를 만들었지만 .count는 앞 그대로 계속해서 사용하는 걸로 대체해 봄에 시도한다.

· Update UI가 재渲을 잘 동하지 않는다.
적용 viewDidLoad를 재渲하고는 초기화 안된다.

.또한 스크린은 못 짜워지면

choice 1 → storyNumber += 1

choice 2 → storyNumber += 2 가 되기

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

• Model - StoryBrain

```

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
}

```

• 무엇이 문제인가?

5 step approach to Debug our App

통해 문제를 해결하자

What did you expect
your code to do?

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



What happened
instead?

- 누르는 button에 따라 story가 달라지니
button마다 증가하는 storyNumber도 다르다

그것이 기대로 반영되지 않은 뿐더러,
updateUI로 최초 배포를 제외하고는 실행이 안된다



What does your
expectation depend upon?

• Model - StoryBrain

```

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

```

@IBAction func increment(_ sender: UITapGestureRecognizer) {
    if userChoice == stories[storyNumber].choice1 {
        print("choice1 : \(storyNumber)")
        storyNumber += 1
        print("choice1 : \(storyNumber)")
    } else {
        print("choice2 : \(storyNumber)")
        storyNumber += 2
        print("choice2 : \(storyNumber)")
    }
}

@IBAction func updateUI(_ sender: UIButton) {
    storyText.text = choice1_2_text // 선택된 텍스트 출력
    // 내용은 대체로 sender에 들어오게 되어,
    // 그걸 nextStory에서 이미 다음 story로 넘기면 내용과
    // 당연히 다르게 출력될 가능성이 있음을 알수는 있다.
    // Update UI의 조건이 꽤다.
}

```



How can we test the things
our expectations depend on?

- 각각의 button과 nextStory에서 증가하는 storyNumber의

로그를 적어보았다.
여기 updateUI가 호출의 안도와 문제를 생기는 것이 코드를 통해
확인되었다.



- View controller에 타이머를 추가한 모습

```
    else if (new_cookbook == cookbook, newer < older) {
        if (older == null) {
            old_cookbook = new_cookbook;
            older = new_cookbook;
        } else {
            old_cookbook = new_cookbook;
            older = new_cookbook;
        }
    }
}

public void test(String name, String file, String type, String value) {
    new TestCookbook(name, file, type, value);
}
```

- 앞서 한 project Quizzer에서 updateU2를
이용해 흐름 했는지 살펴봤다.

- `Quizzler`에서는 전달해온 오답률을 누울 때마다 일시저장을 사용해 효과를 주기 위해 `TimeIntervel`을 사용했는데, `TimeIntervel`의 `property` 중 `Selectors`는 개별마다 설정하는 `timeInterval` 초기 후에 실행할 Function을 넣어줫다.

그리하여 보다やすく `updateUI`를 넣고,
`Objeactive-C`의 문제에 @objc까지 앞에
붙여 주어 `updateUI`가 정상적으로 반응할 수 있어 `도입` ct.

- ReadMe에서 여러 Story가 있는 내용을 가져와 적용시킨 모습이다.

Struct에는 2개의 property가 적용된 모습을

볼 수 있는데, StoryNumber이 증가수를 걸친 대체주는 *But* 이런,

Destination property 를 StoryNumber 으나 바로 대체하기로 했지만
더 깔끔하고 편한 storyboard 진행이 가능하다.
`if`-`else`문으로 클릭되는 button에 따라 storyboard 뷰사이드 이동되는 기준의 logic 을
사용하면, 코드가 자세히는 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 버튼은 choiceMade에서의 sender 구분은 없어
있다.

왜? → 그 기능은 이미 StoryBrain의
nextStory에서 하고 있기 때문이다

• updateUI() 헷갈리?

- Timer도 쓸 필요가 있다.
updateUI는 function이다.
그냥 호출하면 되는 것이다.
만, button 클릭 시마다 내용이 반영되는 것이다.
IBAction choiceMade에서 호출은 해주면 된다.
이제 간단한 것처럼,
굳이 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 있지만, 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