

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.

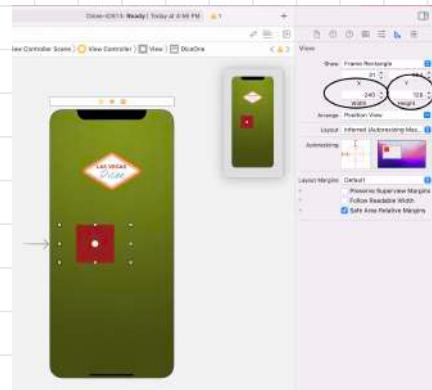
가끔 사이즈를 못찾아 가는 경우도 있는데 (제일 하단의 아이콘) 수동으로 적용시켜 주면 된다.

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 let을 사용해 가능하는 것임;

- Array에서 Random

```
let diceArray = [1, 2, 3, 4, 5, 6]  
  
diceImageView1.image = diceArray.randomElement()  
diceImageView2.image = diceArray[Int.random(in: 0...5)]  
  
.Int.random() 가능하지만, diceArray 사용  
Array의 randomElement()은 가능하지 않음
```

- 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  
  
let optional: 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 += optional.randomElement() ?? "none"  
}  
  
print(password)
```

default value

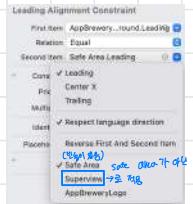
세로 가로 화면 상관 없이 빙글 없애기



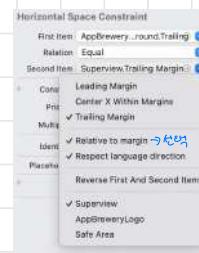
pinning constraints  
only 원하는  
위치 적용

제공된 모습

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



Margin



로고 가운데 배치시키기

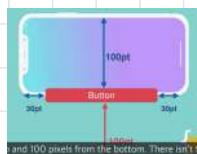
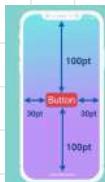
<https://donggooleosori.github.io/2020/12/06/ios-constraint/>



↓  
pinning constraints를 사용하게 되면,  
지금 보이는 화면 그대로에 예상해  
화면을 돌리면 위치는 가운데로 돌아온다.



Alignment Constraints 사용시  
화면의 X-Y축을 설정하는 그대로  
모든 화면에 적용되어야 한다.

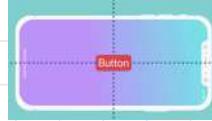
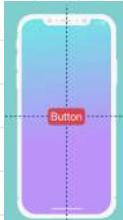


→ pinning Constraints

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  
김별하나 하지면 Display 부분에 그려지겠지 놓쳐된다.

참고로 이 Stack View는 같은 row(가로)의 숫자가 기본값의 Stack View이다.

이제 'D'을 포함한 각 row를 간 Stack View를 적용 시킨다.

3.



→ 적용 시킨 모습  
하단 세로는 조절 필요 →



사이즈 조절?

내장한 UI 생략하면 된다.

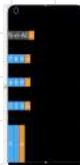
모든 요소들은 같은 row Stack View

최대한 전체의 Safe Area 까지 확장해 해야

나중에 각 숫자가 기준을 맞았을 때 크기 조절은

한 번 공연이 생기면, 다시 만드는 고통 없이 모듈화 예상해

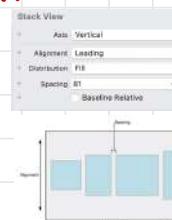
4.



→ pinning constraints 적용

문제점  
1. row Stack View 크기  
2. 숫자 크기를 조정

5.



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

Alignment: leading 아니면 정렬은 확장  
Distribution: fill 아니면 row Stack View의 view는  
최우선이고  
Spacing이 8 아니면 raw Stack View의  
간격이 넓어질 것이다.

\* 일정 복잡해 예상 설명이 잘 안된다.  
공부 필요

6.

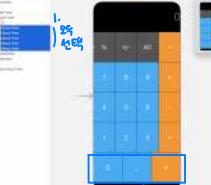


Alignment은 fill로 해줄으로서  
Stack View 뒷배경창으로 경계를 대로 하지 않고 전체를 채움.  
Distribution은 Fill Equally로 함으로서 row Stack View의 크기는  
연동되게 됨.

Spacing → 1: 간격 증폭

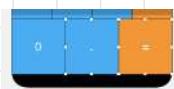
이제, 각 요소들의 크기를 일정하게 해야 한다.  
(맨 앞 0은 제외; 2칸 차지)

7.



↳ Fill Equally 적용  
↳ 아직 하단 부분 조정은 필요.

8.



또 다른 Stack View 생략해 피로하다.

,=은 단 Stack View로 만드는 이유?

↳ 목표 사진처럼 0이 제일 커야하고 (2칸 차지)

나머지는 위 모든 요소들이 크기가 동일하다.

그럼 마지막 row Stack View는 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

9.



먼저 0 선택 원쪽으로  
움직이기.

;;= stack view 적용

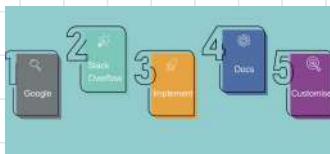


;;= fill equally 적용

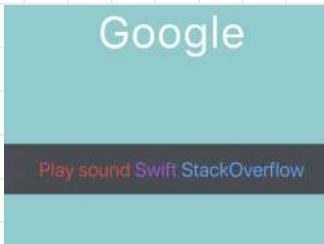


간단히 0 뒤에 0은 합쳐지는 2칸  
이다. Stack View를 하면 차이 있잖아  
이면 기울거나 방향으로 압축해  
줄을 두 칸 줄여 예전에  
View를 4개 줄여 비율을 잘라주고  
2 View로 만들었지만  
pinning constraints를 통해 View의 상용화는  
Constraints를 찾았어 예전에 필요해서 양쪽에  
인정 가능성을 더해 해줄 거 같다.

# 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: "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 = arc4random_uniform(100)
        print("Rating: \(rating)")
    }
}
```

Copied codes



### 2. Instance Method

#### setCategory(\_:)

Sets the audio session's category.

AVAudioSession에서 사용할 수 있는 모드는 Topics 및 게시판 모드이다.  
그중 setCategory() 활용 예제 (shareDucker())는 Session 접근 방식과 나와있음

## 4.



이제 우리가 찾던 Playback이 나왔다.

Stackoverflow에서 초록색으로

제작자가 된 답변이나 해석 항목 같은 것은 없을 것이다.  
사실은 애플 버전 자체가 있기 때문에 일의 글도 잘  
수록 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.

3.

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

보통은 AVAudioSession.Category에 있는 모든 타입을 선택해

적어놓은 되는 것 같다. 즉, 기본.

- 또한 option 키를 넣을 때는 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 이러한 방식으로 반복하는 것은  
매우 비효율적이다 그걸 어떻게 할까?

2.



3.

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

```
    @IBAction func keyPressed(_ sender: UIButton) {
        if sender.tag == 1 {
            print("sender")
            playSound()
        }
    }

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

## 5. challenge

이번엔 Button의 title을 구하는 것이였다.  
처음에 걸세을 막하고 혼자서 해볼 것의 결과다.

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

Button의 title이 절과격으로 나외는 했지만,  
밀보없는 다른 정보들 속에 같이 나와서 이것을 통해 저네로  
Button의 title을 측정하여 그를 없애는데 효과적일지 모르겠다.

Button의 title을 주목하여 그를 하는데 효과적입니다.



```
    C button의 title을 없애고  
    nil 일시 "nil"을 Print하게 했는데  
    산출은 nil이 print 되었다.  
  
그럼 title 은 (for: nameLabel)은 무엇인가
```

A screenshot of an iPhone X simulator displaying a vertical stack of colored buttons. The buttons are red, orange, yellow, green, purple, blue, and pink, each labeled with a letter (D through H) in white. To the right of the simulator is a Swift code editor window. The code defines a class 'ColorfulStack' with a 'init' method that creates a stack of buttons. It uses 'UIStackView' with horizontal alignment 'center' and vertical alignment 'top'. Each button's title is set to its corresponding letter. The code also includes a 'main' function that creates an instance of 'ColorfulStack' and runs it on the main thread.

4

```
⑥ @IBAction func keyPressed(_ sender: UIButton) {  
21  
22     print(sender.backgroundColor) △ Expression implicitly coerced fr...  
23     playSound()  
24 }
```

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

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

```
    }  
    public void printButton(UIButton button) {
```

```
    playSound()
```

Copyright © Stack Overflow 2024. All Rights Reserved.

```
UiControl.State  
+ STATE_UP(0x00000000)  
  
Methods:  
+ void SetState(UiControl.State state);  
+ void SetState(UiControl.State state, int duration);  
+ void SetState(UiControl.State state, int duration, int delay);  
+ void SetState(UiControl.State state, int duration, int delay, int repeat);  
  
Properties:  
+ void SetImage(UIImage image);  
+ UIImage GetImage();  
+ void SetImage(UIImage image, int duration);  
+ UIImage GetImage(int duration);  
+ void SetImage(UIImage image, int duration, int delay);  
+ UIImage GetImage(int duration, int delay);  
+ void SetImage(UIImage image, int duration, int delay, int repeat);  
+ UIImage GetImage(int duration, int delay, int repeat);
```

→ 코드의 원형을 보니 Control (선택 Button)의 state (상태)를 표기하고,

```
sender.title(for: .normal))
```

→ 아마 전통 것이 없으니 natural 이지 않은가 생각한다. (왜 그냥 .natural 인지는 모르겠다)

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, UIButton의 opacity(투명도)를 0.5로 맞추기
        playSound(soundName: sender.currentTitle ?? "C")①
        // I : Don't worry we've already checked
        // playSound(soundName: sender.currentTitle!)
        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 = "jackbaeuerizmesome"
// safety check 문제 되고 있음
var unwrappedUsername = playerUsername

playerUsername = nil // => crash
print(playerUsername!) // safety check를 하지 않았을 때 값이 빈값이면 crash
// how to prevent this happening?

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

```

☞ 짐작해보니 문제는 nil이 있는 부분입니다.

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 코드인데, eggTimes dictionary에서 sender.currentTitle로 들어온 값으로 value를 추출해 print를 하는 것이다. 하지만 내가 너무 어렵게 생각했나보다.  
위에 보이는 것처럼 어짜피 currentTitle은 eggTimes의 value이므로 이름으로 들어오기 때문에 바로 eggTimes의 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 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.

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

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

Declaration  
let result: Int  
Declared In  
ViewController.swift

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

있어도 괜찮았는지  
정작에는 사용 (force-unwrapping)

```
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-upcasting)
        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()
    }
}
```

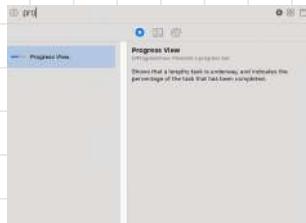
timeInterval - 몇초간의?  
Repeats - true를 택할 timeInterval로 뒤에도 계속 한다.

Selector - object의 인터페이스 ()번의 function을 설정할 때도 @objc를 붙여서 생성해준다.

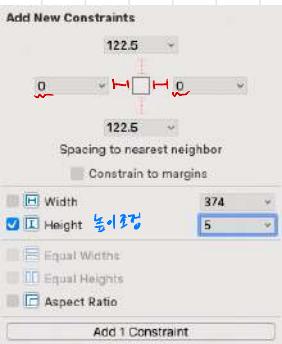
(timer.invalidate() -> 타이머 초기화) : 타이머가 놀랍게마다 초기화를 하지 않는다면 다른 버튼을 누를때 처음 시작 하는 바꿔지만 그때로 초기화된다.

Timer는 잘 작동한다.

하지만 최종 목표인 progressbar와 연동하기 위해서  
몇가지 작업이 더 필요하다.



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 >= 0.8 & 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()
    hardness = sender.currentTitle!
    resetProgressbar()
}

// (! : 뒤집고 쪼신(forced-upwrapping)
let hardness = sender.currentTitle!
    ^ forced - upwrapping
result = 0.0
totalSeconds = Float(eggTimes[hardness]!)
    ^ seconds
print("(hardness) got selected")
    ^ seconds
timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector:
    #selector(updateTimer), userInfo: nil, repeats: true)
    ^ selector
    ^ time interval
override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
}
```

**result**: 초기에 1/4씩 증가한다. progressBar의 진행상황을 시각화한 변수

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

progressBar.setProgress: stackoverflow와 같은 에러.

첫 요소에는 float 러테, 두 번째엔 3에 정식을 넣어 줄다.

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

이유나 다가와 함께하는 즐거움이나 실현되는 *progressive*의

## 지혜사학과 인천하다

리셋을 해주지 않았더니 다른 배우가 다른 *emoji*를 클릭해  
두른 것이다. 앞으로 가는 헌살에 박상하였다.

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:  
- `l`: the length of the string.  
- `ch`: the character to be repeated.  
- `reps`: the number of times the character is to be repeated.  
- `key`: a key used for encryption.  
- `method`: the encryption method to be used.

**Encryption:**  
The `Encrypt` method takes the message specified by `m` and returns the encrypted message. It uses the `key` parameter to generate a random sequence of numbers which are used to index the `charList` array.

**Decryption:**  
The `Decrypt` method takes the encrypted message specified by `m` and returns the original message. It uses the `key` parameter to generate a random sequence of numbers which are used to index the `charList` array.

**Example:**  
The user enters the message "Hello World" and the key "SECRET". The program outputs the encrypted message "Hglo Wrdl".

# • 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 2.5가 나온다.

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

해결 방법

```
let a = 5
let b = 2
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))
    }
}
```

```
0.8
0.8
0.33333334
0.6
0.66666667
0.8
```



→ 만약 Float으로 변환을 한 뒤,

계산을 시작함.

Float이 제대로 나오고 progressbar가  
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
// secondsPassed가 Maximum을 놓아놓으니
// 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 progress가 1.0이에  
증가하지 않는 걸로 증가하지 않는다.  
만약 secondsPassed가 1일 때 있다면 증가를 했을 때도  
부구하고 totalTime과 같은 경기 회피와 사용을  
시킬 수 있다.
```

# 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

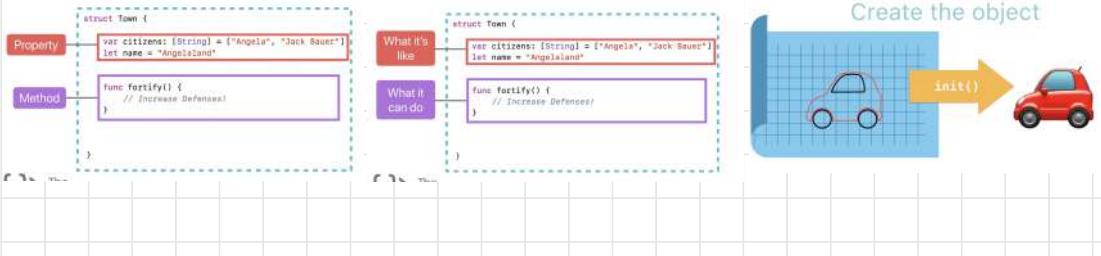
// able to access to properties of Town
print(myTown.citizens)
print("\(myTown.name) has \(myTown.resources["Grain"]!) bags of grain.")

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

// method
myTown.fortify()
```

한국어 번역

Create the object



Creating the initializer

init()으로 생성하는  
Object 만들 때는  
설정하기.  
왜这样说 필요 있나?

Using the initializer

```
StructureName()
```



D를 개발할 때도 둘째 머리가운도는 결론이야  
하는데 각각은 외연적, 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 고지 해야  
유저가 있는 Struct를 blueprint(모형)  
쓰임 용도 하기 때문이다.

```

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, "Gold": 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()

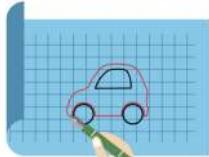
}

exercise()

```

# Swift Deep Dive - Immutability

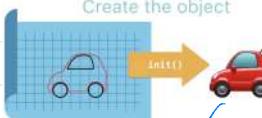
Struct = Blueprint



```
struct QuizBrain {
    property var questionNumber = 0
    method func checkAnswer(_ userAnswer: String) -> Bool {
        if userAnswer == quiz[questionNumber].answer {
            return true
        } else {
            return false
        }
    }
}
```

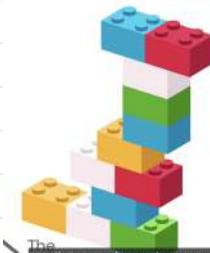
↳ what it's like  
Associated with struct  
↳ what it can do

Create the object



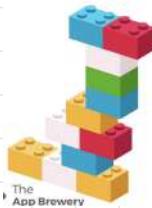
(actual object)

## Immutability



☞ if of 2nd let<sup>ez</sup>  
한번이 되었을 때 이는 끝 불변화  
변경 x 끝 → immutable

Destroy



Rebuild

> Well, instead, you have to destroy the entire structure and create

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
    }
}
```

↳ resources var이 mutable 하기 때문에 immutable 선언 x.

self라는 유형의 값을 짜?

- self가 쓸 코드의 resources 앞에는 self로

생략해도 괜찮아.

그렇지 않으면 error message on 'self' is immutable 오류!

나중에 알게.

"self"는 자동으로 사용되는 properties라서 괜찮아요,

하지만 self는 let으로 만들 수 있음.

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

mutating은 꼭 할 것처럼

error는 아니지만.

mutating은 블록에서 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:

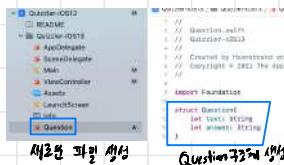
2D 배열 (2차원 배열)로 만든 Question & Answer를 이용해  
array에 따라 Question text를 뜨우고 점을 표시한 것이다.

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 복습이 있다.



서로운 파일 만들기

```
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")
]
```

구조체의 property를 사용해 정답은 오류

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

만일 actualQuestion의 용도는 끝나면.

struct로 정리를 했지.  
하지만 초기 가지고 있는 Quiz의  
index 높여놓은 놈이가면 Crash 된다는 것이다.  
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 비슷한 보았다.

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 backgroundColor가 바뀌는 결과를  
 확인하고자 한다. updateUI에서 정답인 경우에만  
 텍스트가 바뀌도록 하기 위해서는 그 전에 나온  
 텍스트를 초기화하는 코드를 넣어야 한다.



### • 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
    }
}

```

-党风 xylophone project에서 간접 탐색  
 일시적인 효과를 주 DispatchQueue를  
 이용하여 0.2초 후에 새내기 했을 때  
 하겠다. 즉, 저 Method 안에서 새내기 설정.  
 x 이후 단계별 일들을 작성하였습니다.  
 (물론 위에서 정답이나 오류 분별에  
 색깔 효과를 주기 때문이다.)

```

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

```

### 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
    }

    questionLabel.text = quizQuestions[questionNumber].text
}

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

```

Angelaa는 egTimer에 0.2 Timer를 사용해  
 적용하였다.

timeInterval을 0.2로 updateUI function이 0.2로  
 실행되며 그에 backgroundColor를 초기화해 주고  
 다음 question으로 넘긴다. 큼직한 번역 일어에는 일어나  
 셀프에 repeat은 false이다.

X selector는 원래 있는 updateUI Method를 이용한다.  
 그 이유는 우리가 정한 interval value 나 value method를  
 넣어주면 되니까 굳이 새로 만들 필요없이 기존의 Method를  
 이용한 것이고 앞에서 Objective-C의 방법이기 때문에  
 ① ObjC를 알아 봄이 좋다.

