



Code로 UI개발하기

AutoLayout helper Library 소개

이정우 IBM Japan

저는…



- 일본에서만 13년차 일하고 있는 외노자 입니다. 😇
- Backend, frontend 여러 가지 합니다만, Java는 안 합니다. (싫어합니다.) 😥
- iOS는 약 10년 정도
- 현 IBM Japan 근무중

오늘 할 얘기

- iOS App 개발의 중요한 부분인 UI관련 개발할때
Storyboard를 사용하지 않는 방법에 대해.....😊

Agenda

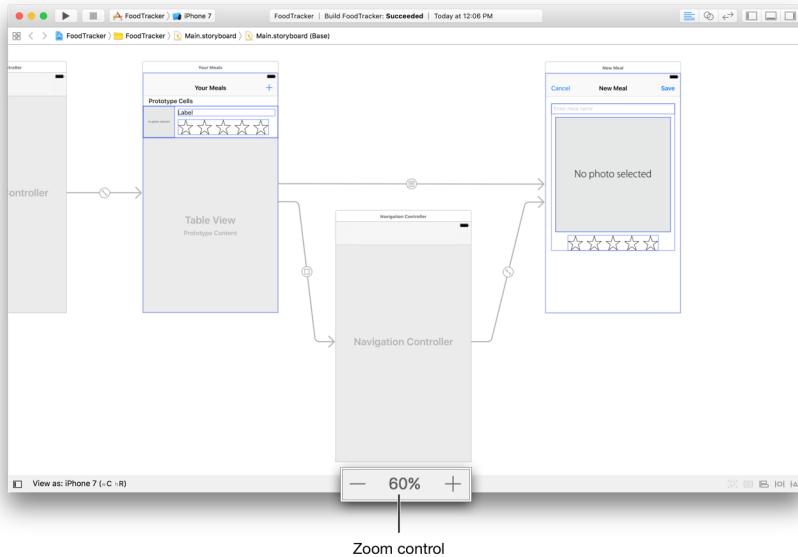
- Storyboard vs Code Base
- Storyboard로 개발할때의 장단점
- Code Base로 개발할때의 장단점
- Code Base로 개발 단점 극복하기

Let



Storyboard vs Code Base

Storyboard vs Code base



VS

A screenshot of the Xcode interface showing the same storyboard and rating control view controller code as the previous image. The code in the rating control view controller is highlighted in red in the Xcode editor. The console output shows the message "Button pressed" with a thumbs-up emoji.

```
func ratingButtonTapped(button: UIButton) {
    print("Button pressed")
}
```

Let



Storyboard로 개발할때의 장단점

Storyboard로 개발의 장점

1. WYSIWYG

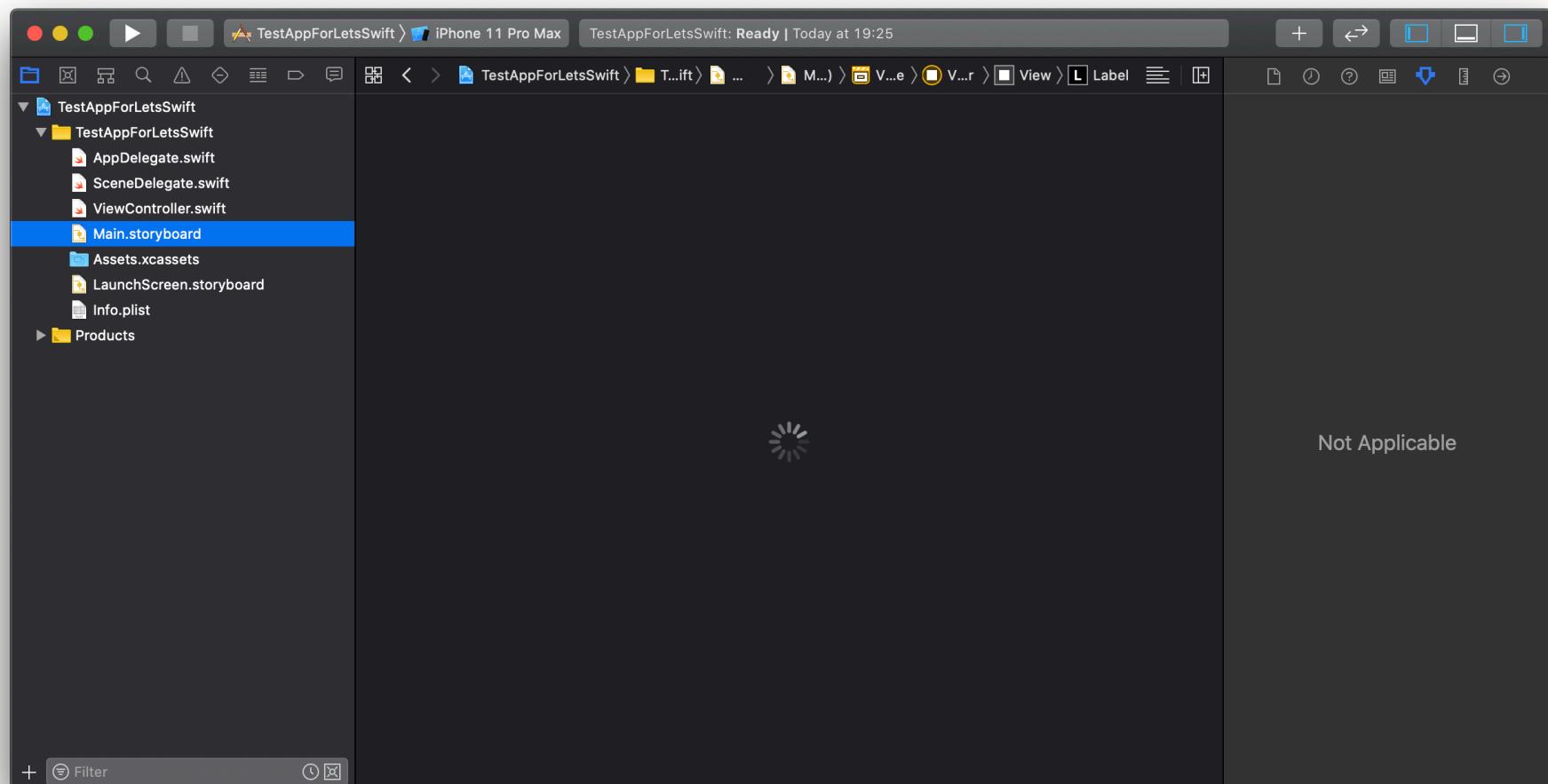
1. 결과물에 관해 예측하기 쉬움

2. 속성확인가능

3. 소스코드를 일일이 기억하지 않아도 됨

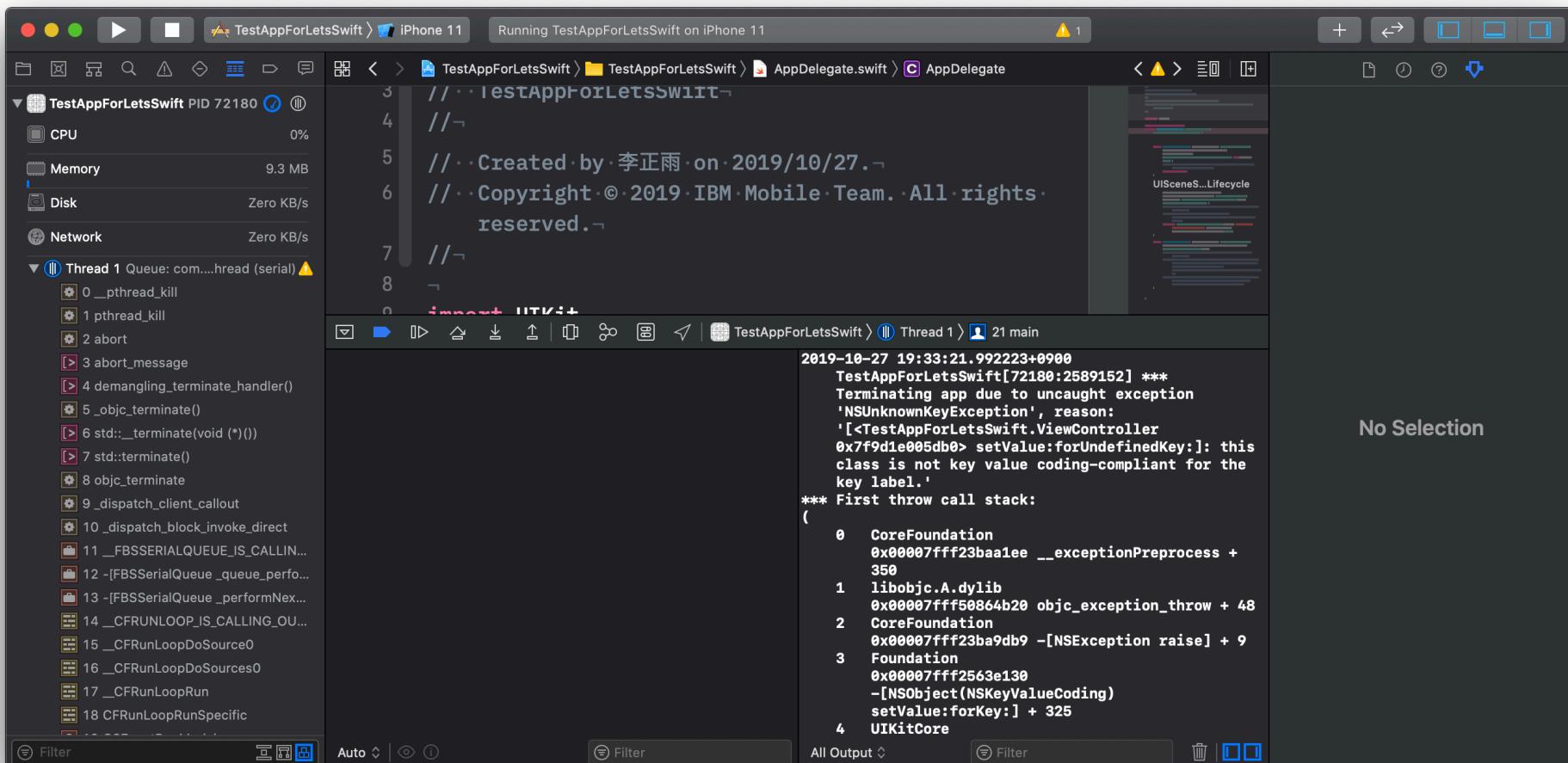
Storyboard로 개발의 단점

1. 무거움 😞



Storyboard로 개발의 단점

2. 링크가 끊어졌을 때 알기힘듬... 🤷



The screenshot shows the Xcode interface during a debug session for a project named "TestAppForLetsSwift" running on an iPhone 11. The top navigation bar indicates the app is running on iPhone 11. The left sidebar displays various monitoring tools: CPU (0%), Memory (9.3 MB), Disk (Zero KB/s), and Network (Zero KB/s). Below these, the Thread 1 Queue (com....hread (serial)) is expanded, showing a list of symbols from 0 to 18, including pthread_kill, abort, and various Objective-C and C++ symbols.

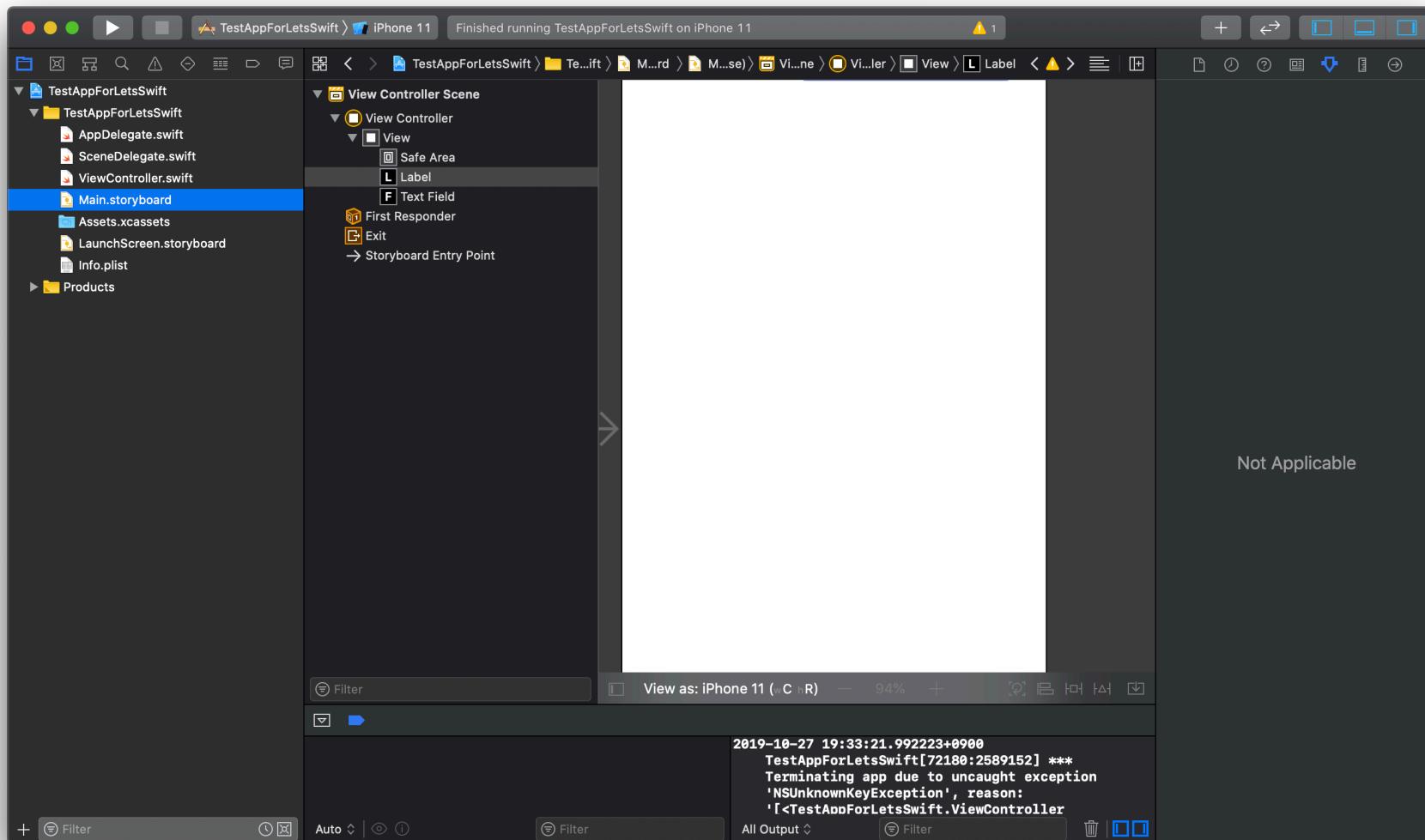
The main area shows the code editor for `AppDelegate.swift` with some initial comments. The bottom half of the screen is the Console tab, which outputs the following crash log:

```
2019-10-27 19:33:21.992223+0900
TestAppForLetsSwift[72180:2589152] ***
Terminating app due to uncaught exception
'NSUnknownKeyException', reason:
'[<TestAppForLetsSwift.ViewController
0x7f9d1e005db0> setValue:forUndefinedKey:]: this
class is not key value coding-compliant for the
key label.'
*** First throw call stack:
(
    0   CoreFoundation
        0x00007fff23baa1ee __exceptionPreprocess + 350
    1   libobjc.A.dylib
        0x00007fff50864b20 objc_exception_throw + 48
    2   CoreFoundation
        0x00007fff23ba9db9 -[NSException raise] + 9
    3   Foundation
        0x00007fff2563e130
        -[NSObject(NSKeyValueCoding) setValue:forKey:] + 325
    4   UIKitCore
```

The right side of the interface has a message "No Selection".

Storyboard로 개발의 단점

3. 작은 모니터로 개발이 짜증남... 😞



Storyboard로 개발의 단점

4. Diff로는 개발 내용을 알기힘듬 -> 리뷰가...



```
$ git diff
diff --git a/TestAppForLetsSwift.xcodeproj/project.xcworkspace/xcuserdata/JungwooLee.xcuserdatad/UserInterfaceState.xcus
index d7c20d4..08d816f 100644
Binary files a/TestAppForLetsSwift.xcodeproj/project.xcworkspace/xcuserdata/JungwooLee.xcuserdatad/UserInterfaceState.xc
diff --git a/TestAppForLetsSwift/Base.lproj/Main.storyboard b/TestAppForLetsSwift/Base.lproj/Main.storyboard
index 89f9f43..7456be7 100644
--- a/TestAppForLetsSwift/Base.lproj/Main.storyboard
+++ b/TestAppForLetsSwift/Base.lproj/Main.storyboard
@@ -49,6 +49,15 @@
    <view key="view" contentMode="scaleToFill" id="hVf-II-4wX">
        <rect key="frame" x="0.0" y="0.0" width="414" height="842"/>
        <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES" />
+
+       <subviews>
+           <label opaque="NO" userInteractionEnabled="NO" contentMode="left" horizontalHuggingPriority
+               <rect key="frame" x="186" y="115" width="42" height="78"/>
+               <autoresizingMask key="autoresizingMask" flexibleMaxX="YES" flexibleMaxY="YES"/>
+               <fontDescription key="fontDescription" type="system" pointSize="17"/>
+               <nil key="textColor"/>
+               <nil key="highlightedColor"/>
+           </label>
+       </subviews>
+       <color key="backgroundColor" systemColor="systemBackgroundColor" cocoaTouchSystemColor="whiteCo
+           <viewLayoutGuide key="safeArea" id="Ezq-ba-8YL"/>
    </view>
@@ -56,7 +65,7 @@
            </viewController>
            <placeholder placeholderIdentifier="IBFirstResponder" id="2B3-se-e61" userLabel="First Responder" custo
        </objects>
-
-       <point key="canvasLocation" x="862" y="135" />
+
+       <point key="canvasLocation" x="907.24637681159425" y="134.59821428571428" />
    </scene>
</scenes>
</document>
lines 1-32/32 (END)
```

Storyboard로 개발의 단점

4.1 Conflict... 😰

The screenshot shows the Xcode interface with the storyboard code editor open. The code is annotated with red numbers (33-40) and arrows pointing to specific elements. A large yellow box highlights the button and its associated rect element. A smaller yellow box highlights the label at the bottom.

```
< > Pro Storyboard Techniques > Main-Unwind.storyboard
33 <button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="K9c-cV-U5T">
34 </button>
35 <label opaque="NO" userInteractionEnabled="NO" contentMode="left" horizontalHuggingPriority="251" verticalHuggingPriority="251" text="Welcome to App" textAlignment="natural" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraints="NO" id="gwA-KU-loY">
36 <rect key="frame" x="146" y="405" width="82" height="30"/>
37 <state key="normal" title="Tap for Fun!" />
38 </button>
39 <label opaque="NO" userInteractionEnabled="NO" contentMode="left" horizontalHuggingPriority="251" verticalHuggingPriority="251" text="Welcome to App" textAlignment="natural" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO" id="gwA-KU-loY">
40 <rect key="frame" x="146" y="405" width="82" height="30"/>
```



Code Base로 개발할때의 장단점

Code Base로 개발 장점

1. 따로 무거워지지 않음
 2. Code만 보이면 모니터 크기는 그렇게 중요하지 않음
 3. 상대적으로 Diff를 알아보기 쉬움
 4. Conflict 발생 가능성이 상대적으로 낮음
- > Storyboard의 단점이 반대로

Code Base로 개발 단점

1. 해당 Component의 속성을 어느정도 숙지하고 있어야함
-> Document를 자주 봅시다!(Cmd + Shift + 0)
2. Source Code이기 때문에 어떤 화면이 만들어질지 알아보기 힘듬
3. Code가 상당히 길어짐

-> 2와 3을 해결해 봅시다!🤔



Code Base 단점 극복하기

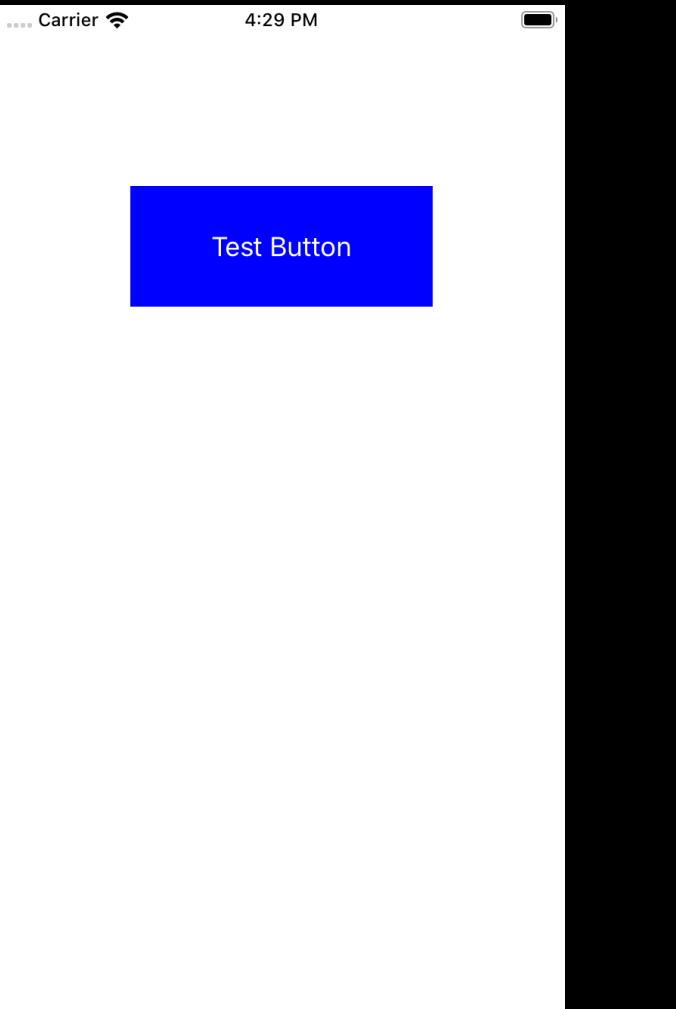
Code Base로 개발 전재

- 당연히 Auto Layout 사용
 - Autoresizing은 논외... 😳
- Auto Layout의 구현 방법
 1. NSLayoutConstraint를 일일이 작성
 2. Visual Format Language를 사용
 3. Anchor 사용

NSConstraint

```
let button = UIButton()
button.setTitle("Test Button", for: .normal)
button.backgroundColor = .blue

view.addSubview(button)
button.translatesAutoresizingMaskIntoConstraints = false
NSLayoutConstraint.activate([
    NSLayoutConstraint(item: button,
        attribute: .centerX,
        relatedBy: .equal,
        toItem: view,
        attribute: .centerX,
        multiplier: 1.0,
        constant: 0.0),
    NSLayoutConstraint(item: button,
        attribute: .width,
        relatedBy: .equal,
        toItem: nil,
        attribute: .height,
        multiplier: 1.0,
        constant: 200.0),
    NSLayoutConstraint(item: button,
        attribute: .top,
        relatedBy: .equal,
        toItem: view,
        attribute: .topMargin,
        multiplier: 1.0,
        constant: 100.0),
    NSLayoutConstraint(item: button,
        attribute: .height,
        relatedBy: .equal,
        toItem: nil,
        attribute: .height,
        multiplier: 1.0,
        constant: 80.0),
])
```



The screenshot shows an iPhone X simulator interface. At the top, there's a status bar with signal strength, 'Carrier', and a battery icon. The time '4:29 PM' is also visible. The main screen displays a single blue rectangular button in the center. The button has a white border and contains the text 'Test Button' in a black sans-serif font.

Visual Format Language

```
NSLayoutConstraint.activate(  
    [  
        NSLayoutConstraint.constraints(withVisualFormat: "H:[button(200)]",  
            options: .alignAllCenterX,  
            metrics: nil,  
            views: ["button": button]),  
        NSLayoutConstraint.constraints(withVisualFormat: "V:[sv]-(<=1.0)-[button(80)]",  
            options: .alignAllCenterX,  
            metrics: nil,  
            views: ["sv": view!, "button": button]),  
        NSLayoutConstraint.constraints(withVisualFormat: "V:|-100-[button(80)]",  
            options: .alignAllTop,  
            metrics: nil,  
            views: ["button": button])  
    ].flatMap({ $0 })  
)
```

- 독특한 문법
- 가운데 정렬이 매우 귀찮음
- 가독성이 아주 뛰어나다고 하기 힘듬

NSLayoutAnchor

```
NSLayoutConstraint.activate([
    button.centerXAnchor.constraint(equalTo: view.centerXAnchor),
    button.widthAnchor.constraint(equalToConstant: 200.0),
    button.topAnchor.constraint(equalTo: view.topAnchor, constant: 100.0),
    button.heightAnchor.constraint(equalToConstant: 80.0)
])
```

- 앞의 방법보다 가독성도 뛰어나고
- Priority의 설정이 귀찮음
 - isActive

Lot



아직 쉽진 않네요.. 😞🤔

생각해봅시다.



```
NSLayoutConstraint.activate([
    button.centerXAnchor.constraint(equalTo: view.centerXAnchor),
    button.widthAnchor.constraint(equalToConstant: 200.0),
    button.topAnchor.constraint(equalTo: view.topAnchor, constant: 100.0),
    button.heightAnchor.constraint(equalToConstant: 80.0)
])
```

equal... greaterThan...

```
let constraint = view1.anchor ==|<=|>=| view2.anchor (*|/|) 1.0 (+|-) 10.0
```

Return

Relation

Multiplier

Constant

Source Code

- <https://github.com/doil6317/SuperEasyLayout>



좀 더 편하고 읽기 쉽게…

- Anchor + Operation!!

```
button.centerX == view.centerX
```

- Constant

```
button.top == view.topMargin + 100.0
```

- Multiplier

```
button.centerX == view.centerX / 2.0 // or view.centerX * 0.5
```

- Constant + Multiplier

```
button.centerX == view.centerX / 2.0 + 20.0
```

- Dimension

```
button.width == 200.0
```

좀 더 편하고 읽기 쉽게…

- 부등호

```
label.right <= view.right - 20.0  
label.top >= view.top - 20.0  
label.width <= 200.0
```

- Anchor Priority

```
button.centerX != .defaultHigh == view.centerX
```

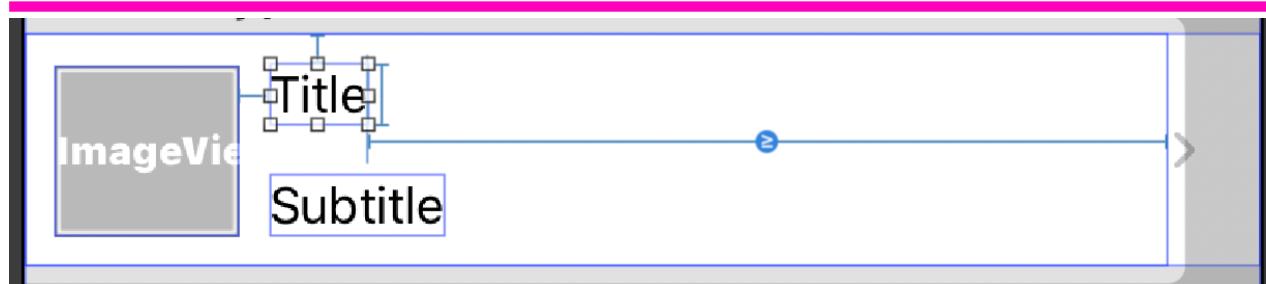
- huggingPriority

```
button.huggingHorizontalPriority == .defaultHigh
```

- compressionResistancePriority

```
button.compressionResistanceVerticalPriority == .defaultLow
```

이렇게..



```
imageView.left == superview.left + 10.0
imageView.width == 60.0
imageView.top == superview.top + 10.0
imageView.bottom == superview.bottom - 10.0

titleLabel.left == imageView.right + 10.0
titleLabel.right <= superview.right - 10.0
titleLabel.top == superview.top + 10.0
titleLabel.height == 21.0

subtitleLabel.left == imageView.right + 10.0
subtitleLabel.right <= superview.right - 10.0
subtitleLabel.height == 21.0
subtitleLabel.bottom == superview.bottom - 10.0
```

참 쉽죠?! 😊

Let



QnA