Mysteries of Auto Layout, Part 2

데릭

보리입니다 데스

바드

1~ 50 페이지 보리
50~98 페이지 바드
98~154 페이지 데릭

D2Coding으로 할 것

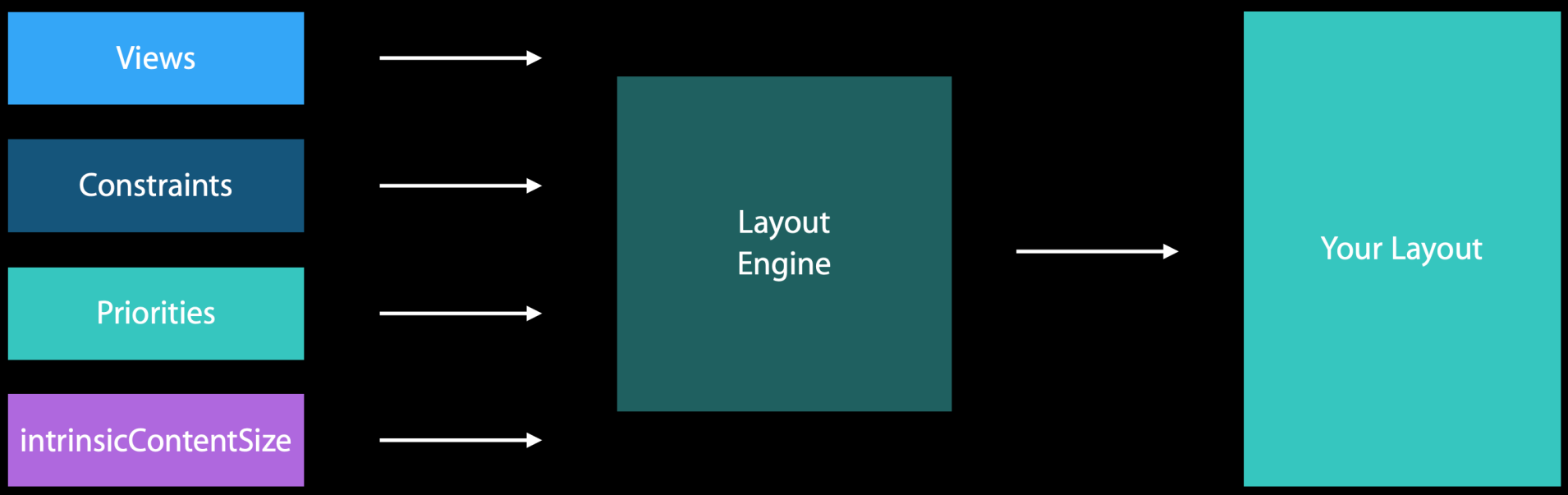
보리

The Layout Cycle

Mystery #7

우리는 사용자 인터페이스를 구성하는 방법은 알고 있지만 자동레이아웃 부분은 여전히 블랙박스 일 수 있다.

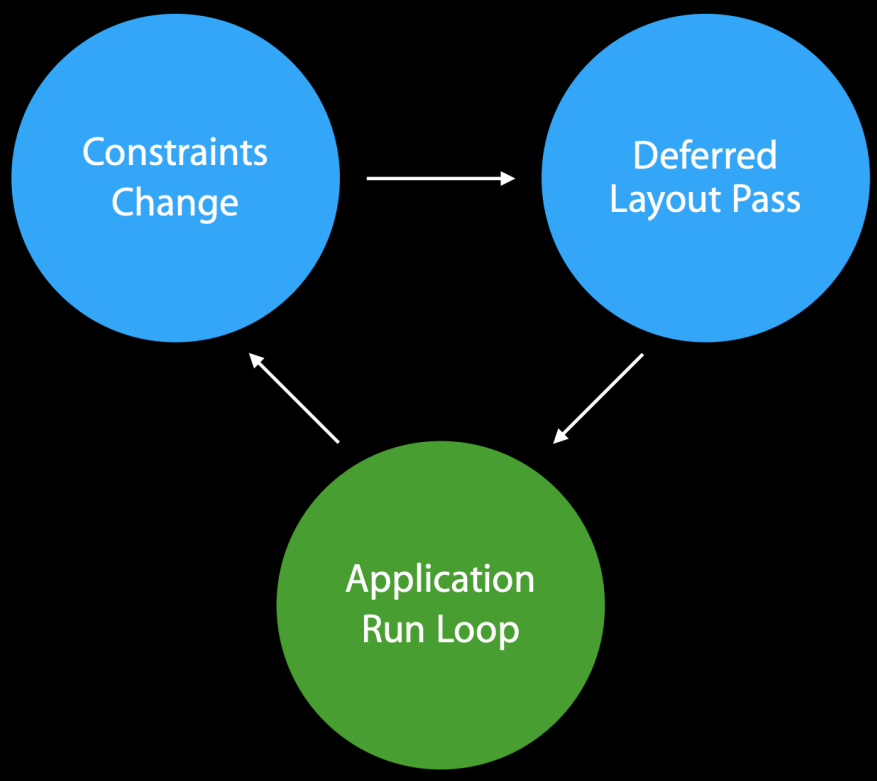
Inside the Black Box



뷰, 제약 조건 등등을 구성하고,
엔진이 이를 계산하고
실제 레이아웃을 얻는다.
이 레이아웃이 우리가 원하는것일 수 있지만 아닐수도있음

이번 챕터에서는 중간에 어떤 일이 일어나는지 알아보고,
뷰에대한 제약조건에서 해당 뷰에 할당된 프레임을 갖는것으로 실제로 어떻게 이동하는지 살펴볼것

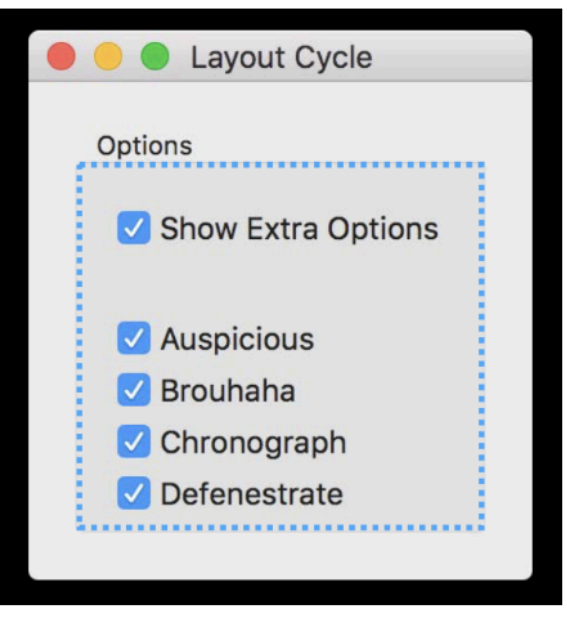
The Layout Cycle

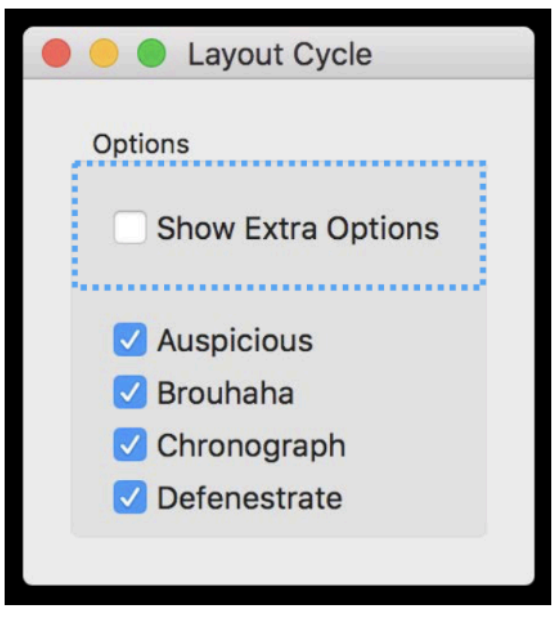


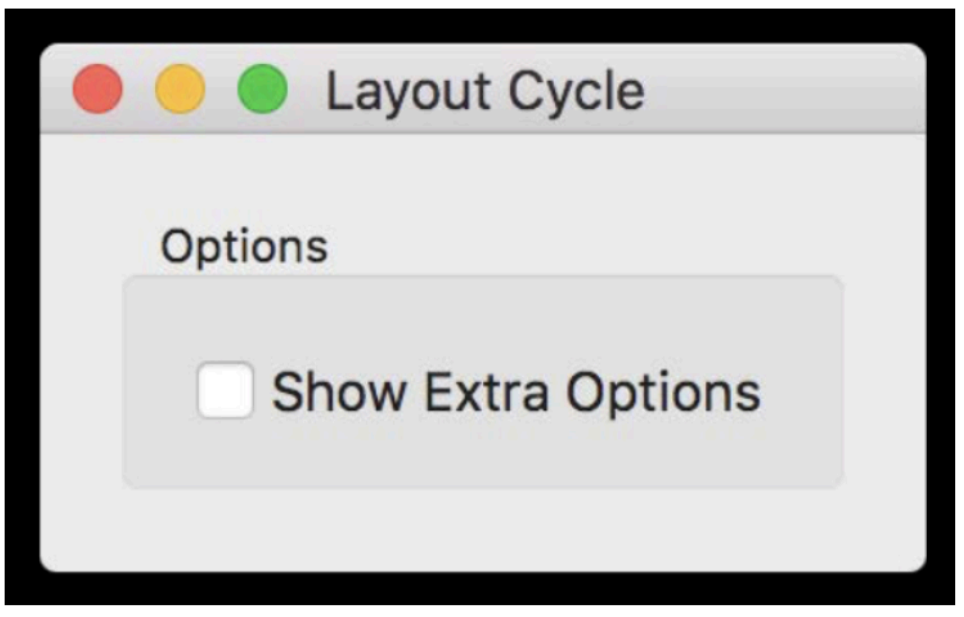
이 사이클은 레이아웃엔진에서 일어나는 작업을 High-level로 표현한것.

중요한건 레이아웃 엔진의 변화와 뷰 계층에서의 변화에는 딜레이가 존재한다는 점이다.

프레임을 즉시 업데이트 하는 대신, 오토레이아웃은 가까운 미래에 layoutPass를 예약한다.
Layout Pass가 완료되면 View 계층 구조를 보고 view의 모든 프레임이 업데이트 된다.







다음과 같은 체크박스로 제약조건이 변경되는 예시를 들었다.

1. show Extra Option 체크 박스를 해제
2. 체크박스에는 제약조건을 변경하는 코드가 존재
3. 레이아웃 엔진이 레이아웃을 다시 계산
   >> Views call superview.setNeedsLayout()
   >> 이는 비동기적 메서드, updatecycle이 왔을 때 작동함.
4. 내 superview가 다시 레이아웃이 필요하다고 표시(mark)
   (== superview.setNeedsLayout()) == deferred layout pass가 예약됨

분명히 나는 Show Extra Options를 체크 해제했지만 아직 view의 크기가 변하지 않음.
여기까지는 레이아웃엔진에서 프레임이 실제로 변경되었지만, view 계층 구조에서는 아직 변경되지 않은 것입니다.

이제 업데이트 사이클이 왔을경우 pass 처리

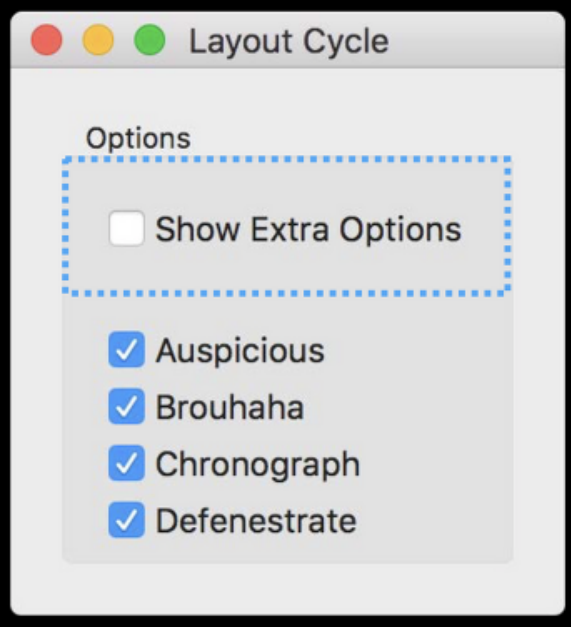
[deferred layout pass]
- layout의 constraints를 업데이트 한 다음, >> update Pass
- view 계층 구조의 모든 view에 대한 프레임을 계산합니다. >> Layout Pass

5. deferred layout pass가 완료되면, 마지막 그림처럼 올바르게 변경됨.

Deferred Layout Pass

`updateConstraints`

Request via `setNeedsUpdateConstraints()`



Often not needed

- Initial constraints in IB
- Separate logic is harder to follow

Implement it when

- Changing constraints in place is too slow
- A view is making redundant changes

[Update pass 과정] > constraint를 업데이트 한다.

1. deferred layout pass시작
2. layout의 constraints를 업데이트 해야함
3. setNeedsUpdateConstraints() 호출 >> 예약하고
4. near future에 updateViewConstraints()가 호출됨 >> 업데이트

다만 문제가 있는데 이런 일련의 과정이 실제로는 그다지 필요하지 않다는 점임.

1. 모든 초기 constraints 설정은 Interface Builder(스토리보드, xib, nib 등) 내에서 발생
2. constraints을 프로그래밍 방식으로 할당해야하는 경우에는 viewDidLoad같은 위치가 훨씬 좋음.

<but>

- 제약 조건 변경이 너무 느리거나
- 뷰가가 중복적인 변경을 수행할 경우.

updateViewConstraints()를 override하여 구현하는것이 도움이 될 수 있다. 이는 updateViewConstraints() 내에서 Constraints를 변경하는 것이, 다른 시간에 constraints를 변경하는 것보다 실제로 더 빠르기 때문

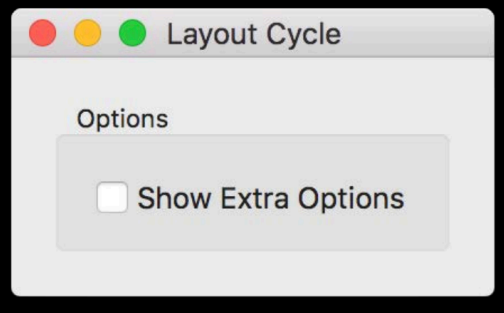
이유 : 엔진이 이 패스에서 발생하는 모든 Constraint 변경사항을 배치로 처리할 수 있기 때문.

Deferred Layout Pass

`layoutSubviews` aka `layout`

Traverse the view hierarchy, top-down

- Call `layoutSubviews()` (or `layout()` on OS X)



Position the view's subviews

- Copy subview frames from the layout engine

Override `layoutSubviews()` for custom layout

- … but be careful!

이제 레이아웃의 constraint는 모두 최신 상태이며(Update pass 완료) view를 그리기만 하면 된다.

[layout pass진입]
1. 시스템은 viewController와 view계층을 위에서 아래로 탐색하며 layoutSubviews()를 호출
>> 이는 리시버의 위치가 아닌 서브뷰의 위치를 재조정 하기 위함.
>> 레이아웃 엔진으로부터 서브뷰의 프레임을 읽은 뒤 할당하는 방식임
2. 레이아웃이 필요하다!고 mark된 모든 view에서 layoutSubviews를 호출.

이때 Constraints를 이용하여 표현할 수 없는 layout을 필요한 경우에만 layoutSubviews() 를 override 한다.
override를 할 경우, 어떠한 view들은 이미 배치되어져 있지만, 또다른 view들은 그렇지 않을 수 있다.
배치되지 않은 view도 아마 곧 배치될 것이므로 약간 섬세한 순간(?)이다.

Deferred Layout Pass

Overriding `layoutSubviews`

Override when constraints are insufficient

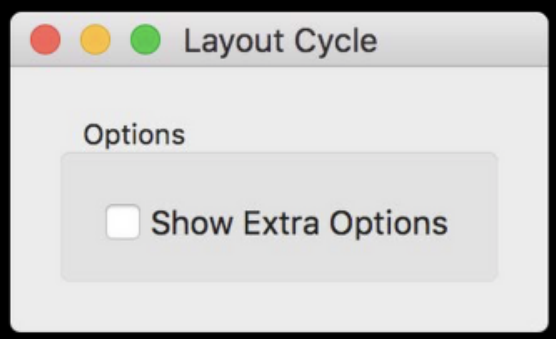
Some views have already been laid out

DO

- Invoke `super.layoutSubviews()`
- Invalidate layout within your subtree

DON'T

- Call `setNeedsUpdateConstraints()`
- Invalidate layout outside your subtree
- Modify constraints indiscriminately



그래서 이 설정한 제약조건대로 뷰를 배치하는 layoutSubviews를 오버라이딩할 때는 몇가지 주의점이 필요하다.

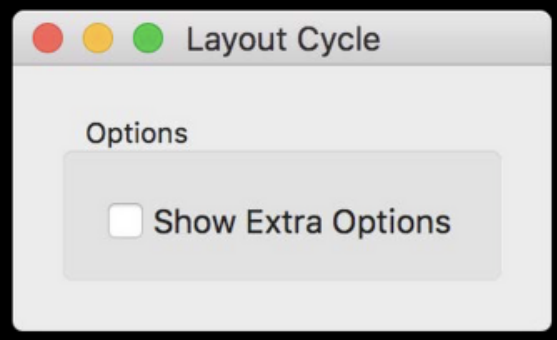
[Do]
super.layoutSubviews() 호출해라
subtree 내에서 view의 레이아웃을 무효화(invalidate)하는 것도 좋다. 하지만 super호출 전에 해야함.

[Don't]
setNeedsUpdateconstraints() 를 호출하지 말아라. > 그럼 다음 레이아웃 패스로 넘어가서 중복이 일어나기 때문같음
하위 트리 외부의 layout을 무효화하지 말아라.
Layout feedback loop가 발생하며 무한 루프에 빠질 수 있다.t
무차별적으로 constraints를 수정하지 말아라.

The Layout Cycle

Remember

- Don't expect frames to change immediately
- Proceed with caution when overriding `layoutSubviews()`



1. **Constraints를 수정 할 때 view프레임이 즉시 변경될 것으로 기대하지 말것.**

2. **layoutSubviews를 override해야하는 경우, 디버깅이 어려울 수 있으므로 레이이아웃 피드백 루프를 피하도록 주의 할 것.**
>> subtree 외부에서 레이아웃 무효화(invalidate)는 하면 안된다 -> 레이아웃 피드백 루프가 발생할 수 있음.
>> layoutSubviews내부에서 Constraints를 수정하는 경우가 있음. 이거 괜찮기는 한데..주의해야 함. 이는 Constraint를 수정할 때 계층 구조의 다른 view가 영향을 받을 수 있는 사항을 예측하기 어려울 수 있기 때문
따라서 Constraints를 변경하는 경우, 실수로 subtree외부의 레이아웃을 무효화(invalidate)하기 매우 쉬움.

Interacting with Legacy Layout

Mystery #8

다음 챕터는 오토 레아아웃이 레거시 레이아웃 시스템과 상호작용하는 방향에 대해 이야기 해본다.

Interacting with Legacy Layout

Positioning by frame versus constraints

Sometimes you need to set the frame

- e.g., if you're overriding `layoutSubviews()`

```
var translatesAutoresizingMaskIntoConstraints: Bool
```

전통적으로 프레임을 설정하여 뷰를 배치한 다음, 슈퍼뷰의 크기가 변경될 때 뷰의 크기를 조정하는 방법을 지정하는 autoResizingMask가 있다.

사실 서브프레임은 작동하지도 않는다

다만 우리는 프레임을 직접 지정해야하는 경우가 있는데LayoutSubviews를 오버라이딩 할 경우 이다.

이것에 대한 flag가 있는데 바로 translateAutoresizingMaskIntoConstraints라는 프로퍼티 이다.

View의 프레임을 이 프로퍼티와 같이 설정하게 되면 프레임워크는 constraints를 만들고 layout engine에서 frame를 enforce한다.

이게 의미하는 바는 frame를 설정할 때 Auto layout를 count하고 뷰를 내가 넣을 곳에 유지한다.
게다가 이 constraints는 autoresizingmask과 유사하게 동작한다.
그리고 Auto layout engine 이용해서 뷰와 상대적인 위치를 파악해서 뷰의 frame를 설정해준다.
만약에 engine에게 어디에서 이 레이아웃이 필요하고 위치가 어딘지 말해주지 않는다면 시스템은 정확하게 파악할 수 없다.

translatesAutoresizingMaskIntoConstraints

Setting the frame automatically generates constraints

- Set the frame with gleeful abandon!
- Constraints implement the autoresizingMask
- Other views can be constrained to it

Set to false when using constraints

- Beware–defaults to true for programmatically created views

프레임을 설정하면 자동으로 제약조건을 설정함.

- 프레임을 원하는 만큼 자주 설정할 수 있고, 오토레이아웃을 사용하여 원하는 위치에 뷰를 유지시킬 수 있다.
- 이러한 일련의 과정은 autoResizingMask의 동작을 구현한다.
  >> 어플리케이션의 일부가 오토레이아웃으로 업데이트 안되어있고 autoResizingMask에 의존하는 경우에도 여전히 정상 작동 해야함.
- 오토레이아웃 엔진을 사용하여 설정한 프레임을 적용함으로써 해당 뷰에 상대적인 다른 뷰들과의 제약조건을 걸고 배치가 가능
  >> 오토레이아웃 엔진으로 프레임을 적용했기 때문에 다른뷰들에 대해서 상대적인 제약조건이 가능하다는 이야기 같음

프레임이 설정되었기 때문에 뷰를 자체적으로 이동시킬 수는 없음. 다만 레이아웃 엔진에 이 뷰가 있어야할 위치를 알려주지 않으면 제약조건으로 참조하자마자 문제가 발생할 수 있음.

그리고 인터페이스 빌더(스토리보드 방식)를 쓸때는 알아서 잘 처리해주지만 코드베이스로 짜면 기본적으로 모두 true임.

```
2015-05-08 09:41:27.668 WWDC 2015[4107:226949] Unable to simultaneously
satisfy constraints:
(
    "<NSAutoresizingMaskLayoutConstraint:0x6100000810e0 h=--& v=--& H:|-(0)-
[NSButton:0x618000140160'Button']   (Names: '|':NSView:0x618000120460 )>",
    "<NSLayoutConstraint:0x6180000828a0 H:|-(10)-[NSButton:
0x618000140160'Button'](LTR)   (Names: '|':NSView:0x618000120460 )>"
)
Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x6000000825d0 H:|-(10)-[NSButton:0x600000140c60'Button']
(LTR)   (Names: '|':NSView:0x6000001203c0 )>
```

코드로 작성할 때 frame을 직접 설정하는 경우, translateAutoresizingMaskIntoContraints를 false로 주지 않으면 resizingMask와 contraints 세팅한 것이 충돌나서 제대로 레이아웃이 세팅되지 않는다!
이 예제에서는 좌표값이 0,0과 10,10이 동시에 존재하게 만드는것이 문제임

이렇게 아예 잡히지 않게 됨.

translatesAutoresizingMaskIntoConstraints

Remember

- Use when setting the frame directly
- Otherwise, don't forget to turn this off!

결론

translatesAutoresizingMaskIntoConstraints 는 프레임을 직접 설정할 때 사용해라
일반적인 경우엔 이 플래그는 필요하지 않긴하지만..

그리고 인터페이스 빌더를 사용하여 제약조건이 있는 항목을 배치할 경우 끄는것을 잊지말라.

바드

Constraint Creation

Mystery #9

이번에는 제약조건 생성에 대해 애기해볼 것

우리가 방금 화면에 표시한 코드, 특히 우리가 이러한 제약을 만들고 있는 마지막 부분을 보는 것만으로도 우리는 제약조건 생성을 가장 쉽게 할 수 있음

Layout Constraint Creation

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

    let b = NSButton()
    b.bezelStyle = .RoundedBezelStyle
    b.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(b)

    NSLayoutConstraint(item:b, attribute:.Top, relatedBy:.Equal, toItem:view,
       attribute:.Top, multiplier:1, constant:10).active = true
    NSLayoutConstraint(item:b, attribute:.Leading, relatedBy:.Equal,
       toItem:view, attribute:.Leading, multiplier:1, constant:10)
       .active = true
}
```

이것은 오토 레이아웃 초기부터 사용해온 제약 공장 방식과 동일하며, 완벽하게 효과적이지만 사용하기에는 약간 어색할 수 있음

코드가 꽤 장황하고 읽기가 조금 어려움
여기서 우리가 정말 표현하고자 하는 것은 단지 버튼을 위에서 10포인트, 왼쪽에서 10포인트 위치시키고 싶다는 것임

하지만 이것을 이해하기 위해서는 이 코드를 주의 깊게 읽고 하나로 봐야함
따라서 OS X 및 iOS의 새로운 릴리스에서는 제약 조건을 생성하기 위한 보다 간결한 새로운 구문을 소개

Layout Constraint Creation

Layout anchors

NEW

```
[NSLayoutConstraint constraintWithItem:b attribute:NSLayoutAttributeTop
    relatedBy:NSLayoutRelationEqual toItem:self.view
    attribute:NSLayoutAttributeTop multiplier:1 constant:10];
[NSLayoutConstraint constraintWithItem:b
    attribute:NSLayoutAttributeLeading relatedBy:NSLayoutRelationEqual
    toItem:self.view attribute:NSLayoutAttributeLeading multiplier:1
    constant:10];

[b.topAnchor constraintEqualToAnchor:self.view.topAnchor constant:10];
[b.leadingAnchor constraintEqualToAnchor:self.view.leadingAnchor constant:10];
```

레이아웃 앵커는 특정 뷰의 특정 속성을 나타내며 앵커 객체는 다양한 형태의 제약 조건을 생성하기 위한 다양한 공장 방법을 노출함

일곱 줄에서 두 줄로 줄였음
그래서 이 새로운 구문은 여전히 우리의 모든 규칙을 준수하지만
훨씬 더 많은 표현을 읽으며, 코드의 의도를 훨씬 더 쉽게 볼 수 있게 해줌

이 구문을 사용하여 모든 유효한 형식의 제약 조건을 만들 수 있으며
실제로 많은 유효하지 않은 형식의 제약 조건에 대해 컴파일러 오류가 발생할 수도 있음

objective-C에서도 오류가 발생하지만, swift에서도 오류가 발생함

Layout Constraint Creation

Layout anchors

NEW

Cannot set a location equal to a constant

```
[v1.leadingAnchor constraintEqualToConstant:100];
   // Error: may not respond to method
```

Cannot relate a location to a size

```
[v1.leadingAnchor constraintEqualToAnchor:v2.widthAnchor];
   // Error: incompatible pointer type
```

예를 들어, 뷰의 leading anchor가 100이어야 한다고 말하는 것은 말이 안됨
왜냐하면 100을 해석할 것이 없기 때문임
따라서 위치 anchor에서 이 메서드를 사용할 수 없다는 오류가 표시됨
마찬가지로, 뷰의 leading edge가 다른 뷰의 너비와 같다고 말하는 것은 말이 안됨

위치 및 크기는 기본적으로 자동 레이아웃에서 호환되지 않는 유형이므로 호환되지 않는 포인터 유형을 얻을 수 있음

이전에는 이러한 것들은 여전히 오류였지만, 실행 시에만 나타남
그래서그것들을 컴파일 시간 오류로 만드는 것은 우리 모두가 처음에 우리의 제약을 바로 잡을 수 있도록 도와줄 뿐만 아니라, 더 읽기 쉽고, 더 유지 가능한 코드를 작성하는 데 도움이 될 것

Constraining Negative Space

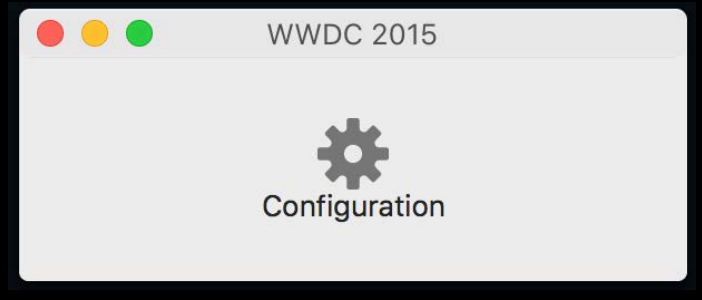
Mystery #10

그래서 다음으로 저는 부정적인 공간을 제한하는 것에 대해 말할것임
몇 가지 다른 종류의 레이아웃이 때때로 나타나는데,
이 레이아웃을 달성하는 방법이 바로 명확하지 않음

Constraining Negative Space

Equal spacing between buttons

Centering a group



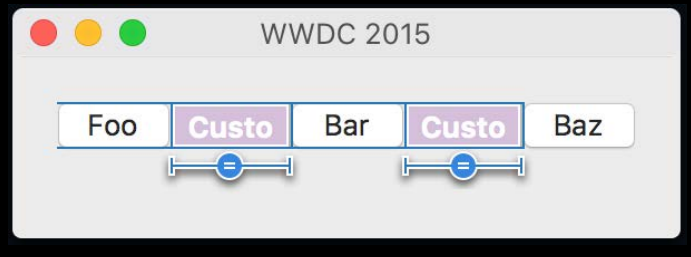
여기 몇 가지 예가 있음
여기서 첫 번째 경우에는 window 크기를 조정할 때 이러한 버튼 사이의 공간이 동일하게 유지되도록 하는 것이 목표

그리고 맨 아래에는 이미지와 레이블이 있음
그리고 우리는 그들을 각각의 컨텐츠를 개별적으로 중앙에 배치하는 것이 아니라 그룹으로 배치하려고 함

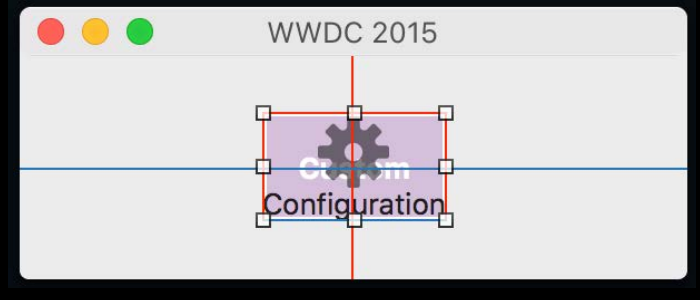
이 레이아웃 문제에 대한 해결책은 더미 뷰를 사용하는 것임

Constraining Negative Space

Equal spacing between buttons



Centering a group



우리는 실제로 빈 뷰를 할당하고 버튼 사이의 공간을 채우도록 제한함
뷰가 있으면 동일한 너비 제약 조건을 사용하여 창 크기가 변경될 때와 동일하게 유지되도록 할 수 있음

그리고 맨 아래의 경우에도 똑같이 할 수 있음
빈 뷰를 사용하고 이미지와 레이블의 가장자리로 제한함
그런 다음 콘텐츠 뷰 자체보다는 빈 뷰에 중심을 지정할 수 있음

예전부터 이러한 layout 문제들을 해결함
하지만 비효율적임
특히 iOS에서는 모든 뷰에 레이어가 연결되어 있습니다.

NSLayoutGuide / UILayoutGuide

NEW

`UILayoutGuide` represents a rectangle in the layout engine

Constrain just like a view

```
let guide = UILayoutGuide()
view.addLayoutGuide(guide)
```

새로운 릴리즈에서는 레이아웃 가이드를 위한 새로운 클래스를 보여줌

UILayoutGuide는 레이아웃 엔진에서 직사각형을 나타냄
UILayoutGuide는 사용하기 매우 쉬움
이들을 할당한 다음 소유 뷰에 추가하면 뷰와 마찬가지로 제한할 수 있음

이들은 앵커 객체를 노출하므로 새 제약 조건 생성 구문으로 작동하지만 기존 제약 조건 팩토리 메서드로 전달할 수도 있음
그래서 그들은 시각적 형식 언어와 같은 것들로 작업할 것

NSLayoutGuide / UILayoutGuide

Layout anchors are not available for margins

UIView now exposes `layoutMarginsGuide`

```
var layoutMarginsGuide: UILayoutGuide
```

우리는 이것들을 내부적으로 사용하기 위해 기존 layout guide를 변환하고 있고 좋은 예가 있음

UIView는 실제로 여백 속성에 대한 레이아웃 앵커를 노출하지 않음
대신 UIView에 새 layout margin guide가 있음

이 layout guide는 여백 내부의 보기 영역만 냄

따라서 레이아웃 가이드는 근본적으로 새로운 동작을 가능하게 하지 않음
View를 사용하여 이러한 모든 작업을 수행할 수 있음

그러나 이러한 문제들을 훨씬 더 가볍게 해결할 수 있고 실제로 그릴 필요가 없는 View로 View 계층을 혼란스럽게 하지 않아도 됨

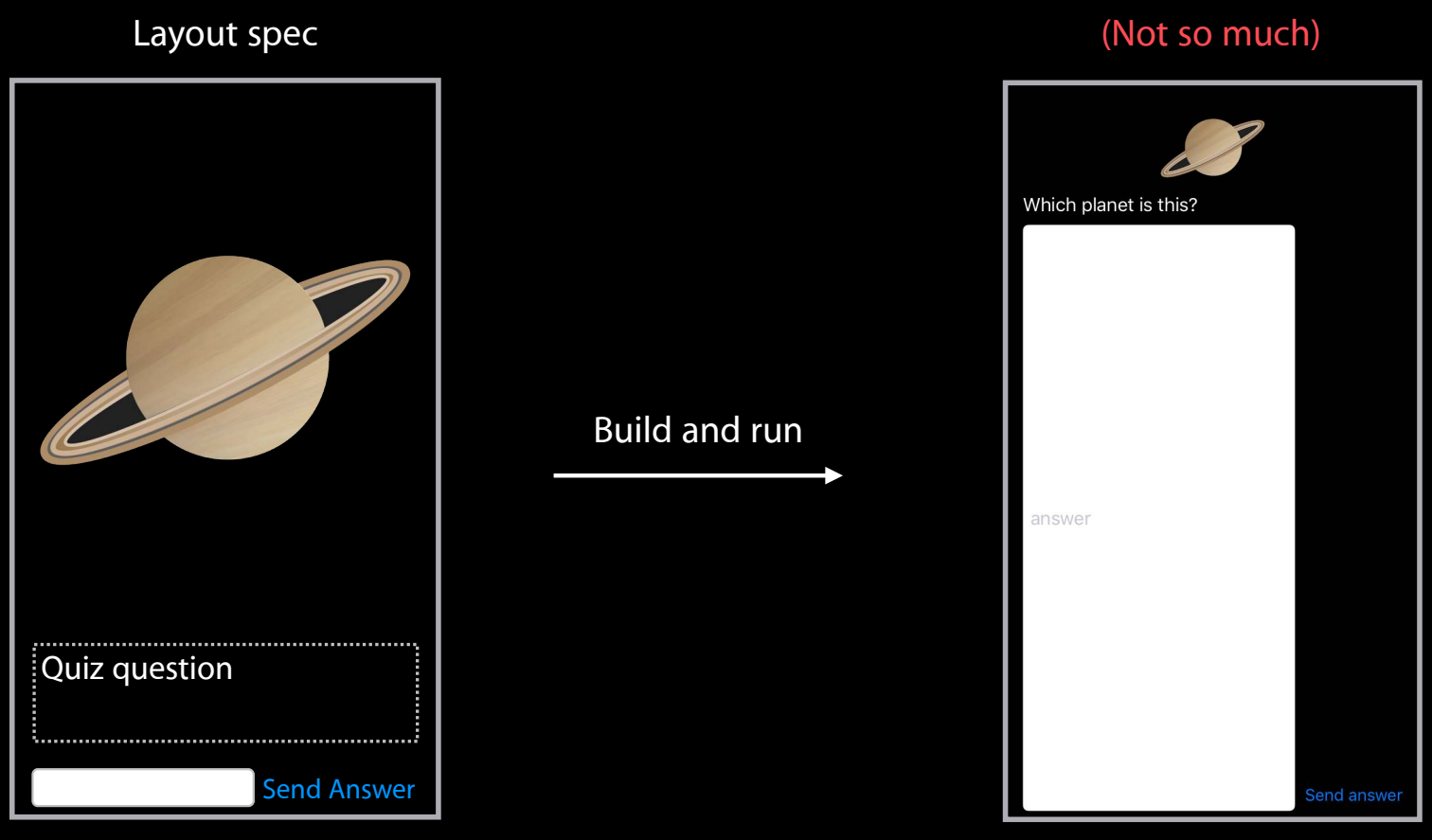
Debugging Your Layout

Mysteries of Auto Layout, part 2

Kasia Wawer iOS Keyboards Engineer

이번에는 layout에 문제가 있을 때 디버깅하는 것에 대해 애기해볼 것

Has This Ever Happened to You?



뷰를 작업하다 자신이 원하는 대로 안나온 적이 있을 것

Has This Ever Happened to You?

```
2015-05-25 16:01:39.543 DebuggingAutoLayout[12208:1048406] Unable to simultaneously satisfy constraints.
      Probably at least one of the constraints in the following list is one you don't want.
(
    "<_UILayoutSupportConstraint:0x7ffe9ad11a80 V:[_UILayoutGuide:0x7ffe9ad10650(20)]>",
    "<_UILayoutSupportConstraint:0x7ffe9ad10ba0 V:|-(0)-[_UILayoutGuide:0x7ffe9ad10650]   (Names: '|':UIView:0x7ffe9c81b720 )>",
    "<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width == 1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905460 'imageHorizontal' saturn.leading == UIView:0x7ffe9c81b720.leadingMargin   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905aa0 'verticalLayout' V:[_UILayoutGuide:0x7ffe9ad10650]-(NSSpace(8))-[saturn]   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905b40 'verticalLayout' V:[saturn]-(NSSpace(8))-[UILabel:0x7ffe9c903d10'Which planet is this?']
(Names: saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c906050 'labelToTop' V:|-(100)-[UILabel:0x7ffe9c903d10'Which planet is this?']   (Names: '|':UIView:
0x7ffe9c81b720 )>",
    "<NSLayoutConstraint:0x7ffe9c905920 'imageMiddle' saturn.centerX == UIView:0x7ffe9c81b720.centerX   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9aca0130 'UIView-Encapsulated-Layout-Width' H:[UIView:0x7ffe9c81b720(375)]>"
)

Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width == 1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>
```

디버거 콘솔 창에서 이런 것을 볼 수 있음

만은 글이 나올것이고 조금 위협적일 수 있음
하지만 실제로는 정말 유용한 로그임

그리고 이것은 만족할 수 없는 제약 오류에 부딪힐 때 발생함
엔진은 우리가 부여한 일련의 제약조건들을 살펴보았고 레이아웃을 실제로 해결할 수 없다고 결정함
왜냐하면 무언가 다른 제약과 충돌하기 때문
해결하기 위해서는제약조건 중 하나를 깨야함

그래서 그것이 무엇을 했는지 말해주는 오류를 던짐
그리고 나서 여러분은 가서 추가적으로 경쟁적인 제약조건을 찾아야 함

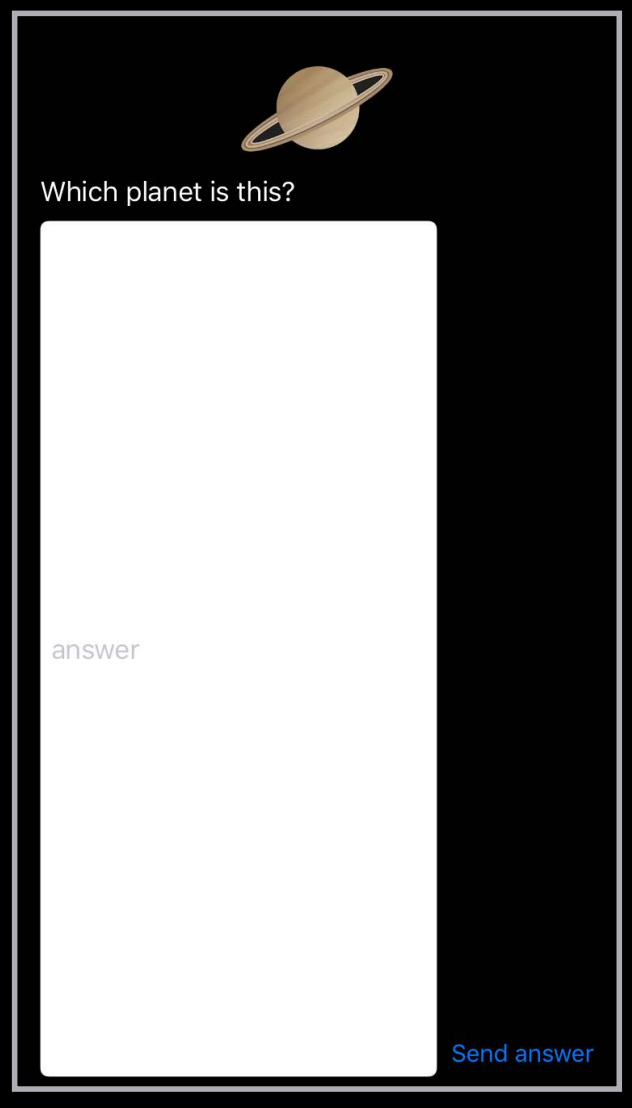
Unsatisfiable Constraints

Mystery #11

Understanding the Log

```
(
    "<_UILayoutSupportConstraint:0x7ffe9ad11a80 V:[_UILayoutGuide:
0x7ffe9ad10650(20)]>",
    "<_UILayoutSupportConstraint:0x7ffe9ad10ba0 V:|-(0)-[_UILayoutGuide:
0x7ffe9ad10650]   (Names: '|':UIView:0x7ffe9c81b720 )>",
    "<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width ==
1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905460 'imageHorizontal' saturn.leading
== UIView:0x7ffe9c81b720.leadingMargin   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7fedd3423ae0 'imageHorizontal' UIView:
0x7fedd3607b90.trailingMargin == saturn.trailing>",   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905b40 'verticalLayout' V:[_UILayoutGuide:0x7ffe9ad10650] (NSSpace(8))-[saturn]>",   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905b40 'verticalLayout' V:[saturn]-
(NSSpace(8))-[UILabel:0x7ffe9c903d10'Which planet is this?']   (Names:
saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c906050 'labelToTop' V:|-(100)-[UILabel:
0x7ffe9c903d10'Which planet is this?']   (Names: '|':UIView:0x7ffe9c81b720
)>",
    "<NSLayoutConstraint:0x7ffe9aca0130 'UIView-Encapsulated-Layout-Width'
H:[UIView:0x7ffe9c81b720(375)]>"
)

Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width ==
1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>
```



이러한 뷰를 짰고 많은 로그를 가짐
그냥 읽으면 어려우니 일단 노란색 부분을 보자

이 부분은 실제로 깨진 조건임

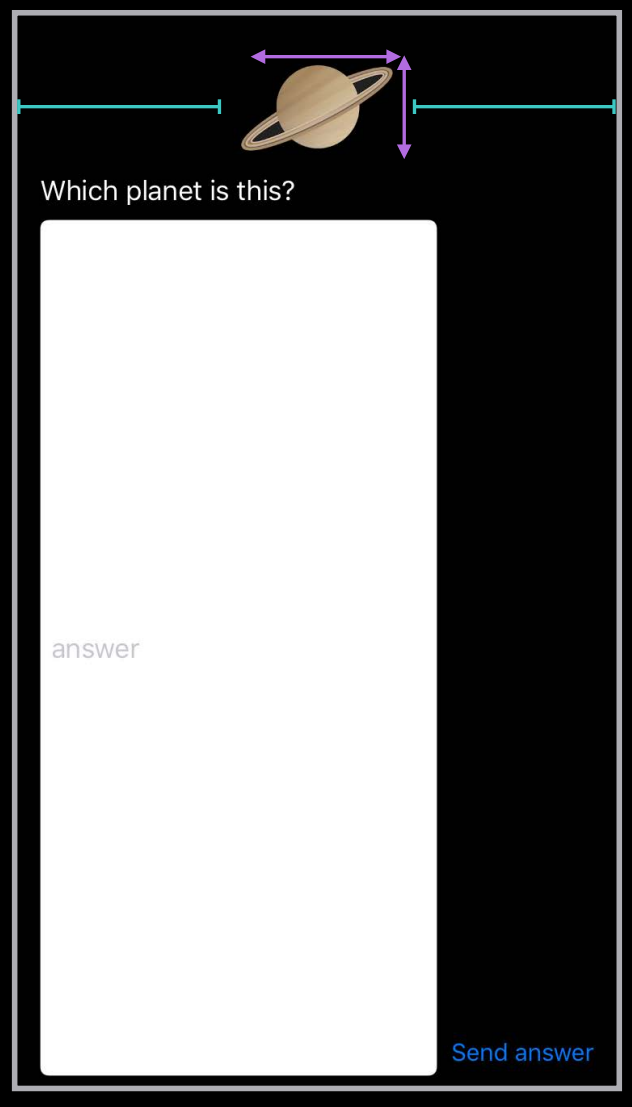
이는 반드시 문제를 일으키는 제약 조건이 아니라 레이아웃을 해결하기 위해 엔진이 고장 나야 하는 제약 조건이기 때문에 시작하기에 정말 좋은 곳임

일단 translatesAutoResizingMask를 확인하는 것이 좋음
현재 경우 Saturn의 aspect ratio가 깨졌음

Understanding the Log

```
(
    "<_UILayoutSupportConstraint:0x7ffe9ad11a80 V:[_UILayoutGuide:
0x7ffe9ad10650(20)]>",
    "<_UILayoutSupportConstraint:0x7ffe9ad10ba0 V:|-(0)-[_UILayoutGuide:
0x7ffe9ad10650]   (Names: '|':UIView:0x7ffe9c81b720 )>",
    "<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width ==
1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905460 'imageHorizontal' saturn.leading
== UIView:0x7ffe9c81b720.leadingMargin   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7fedd3423ae0 'imageHorizontal' UIView:
0x7fedd3607b90.trailingMargin == saturn.trailing>”,   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905aa0 'verticalLayout' V:
[_UILayoutGuide:0x7ffe9ad10650]-(NSSpace(8))-[saturn]   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905b40 'verticalLayout' V:[saturn]-
(NSSpace(8))-[UILabel:0x7ffe9c903d10'Which planet is this?']   (Names:
saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c906050 'labelToTop' V:|-(100)-[UILabel:
0x7ffe9c903d10'Which planet is this?']   (Names: '|':UIView:0x7ffe9c81b720
)>",
    "<NSLayoutConstraint:0x7ffe9aca0130 'UIView-Encapsulated-Layout-Width'
H:[UIView:0x7ffe9c81b720(375)]>"
)

Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width ==
1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>
```



이러한 뷰를 짰고 많은 로그를 가짐
그냥 읽으면 어려우니 일단 노란색 부분을 보자

이 부분은 실제로 깨진 조건임

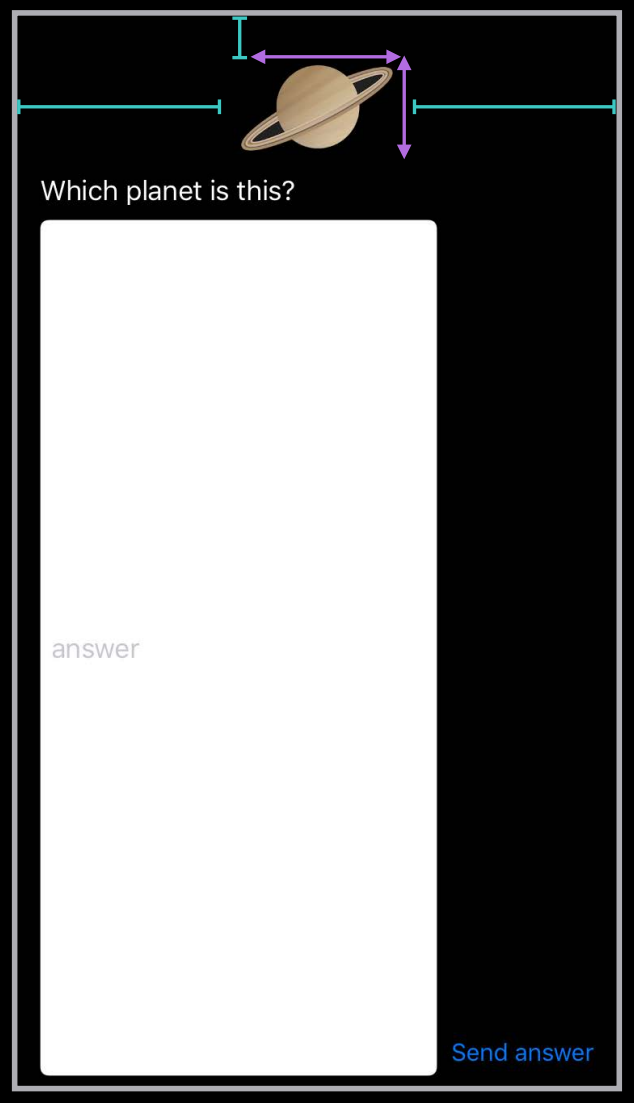
이는 반드시 문제를 일으키는 제약 조건이 아니라 레이아웃을 해결하기 위해 엔진이 고장 나야 하는 제약 조건이기 때문에 시작하기에 정말 좋은 곳임

일단 translatesAutoResizingMask를 확인하는 것이 좋음
현재 경우 Saturn의 aspect ratio가 깨졌음

Understanding the Log

```
(
    "<_UILayoutSupportConstraint:0x7ffe9ad11a80 V:[_UILayoutGuide:
0x7ffe9ad10650(20)]>",
    "<_UILayoutSupportConstraint:0x7ffe9ad10ba0 V:|-(0)-[_UILayoutGuide:
0x7ffe9ad10650]   (Names: '|':UIView:0x7ffe9c81b720 )>",
    "<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width ==
1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905460 'imageHorizontal' saturn.leading
== UIView:0x7ffe9c81b720.leadingMargin   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7fedd3423ae0 'imageHorizontal' UIView:
0x7fedd3607b90.trailingMargin == saturn.trailing>”,   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905aa0 'verticalLayout' V:
[_UILayoutGuide:0x7ffe9ad10650]-(NSSpace(8))-[saturn]   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905b40 'verticalLayout' V:[saturn]-
(NSSpace(8))-[UILabel:0x7ffe9c903d10'Which planet is this?']   (Names:
saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c906050 'labelToTop' V:|-(100)-[UILabel:
0x7ffe9c903d10'Which planet is this?']   (Names: '|':UIView:0x7ffe9c81b720
)>",
    "<NSLayoutConstraint:0x7ffe9aca0130 'UIView-Encapsulated-Layout-Width'
H:[UIView:0x7ffe9c81b720(375)]>"
)

Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width ==
1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>
```



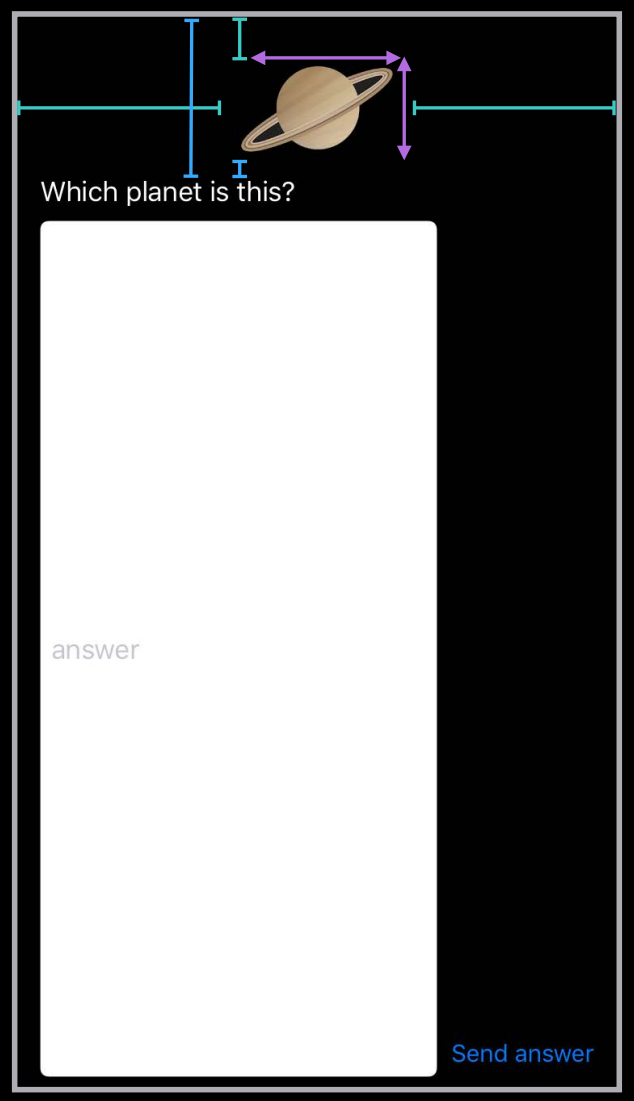
슈퍼 뷰의 leading과 trailing에 이어져 있고 하나는 탑에, 그 밑에 레이블 뷰가 있는 것을 볼 수 있음

현재까지는 충돌하는게 없어보임

Understanding the Log

```
(
    "<_UILayoutSupportConstraint:0x7ffe9ad11a80 V:[_UILayoutGuide:
0x7ffe9ad10650(20)]>",
    "<_UILayoutSupportConstraint:0x7ffe9ad10ba0 V:|-(0)-[_UILayoutGuide:
0x7ffe9ad10650]   (Names: '|':UIView:0x7ffe9c81b720 )>",
    "<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width ==
1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905460 'imageHorizontal' saturn.leading
== UIView:0x7ffe9c81b720.leadingMargin   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7fedd3423ae0 'imageHorizontal' UIView:
0x7fedd3607b90.trailingMargin == saturn.trailing>”,   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905aa0 'verticalLayout' V:
[_UILayoutGuide:0x7ffe9ad10650]-(NSSpace(8))-[saturn]   (Names: saturn:
0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c905b40 'verticalLayout' V:[saturn]-
(NSSpace(8))-[UILabel:0x7ffe9c903d10'Which planet is this?']   (Names:
saturn:0x7ffe9acb8cb0 )>",
    "<NSLayoutConstraint:0x7ffe9c906050 'labelToTop' V:|-(100)-[UILabel:
0x7ffe9c903d10'Which planet is this?']   (Names: '|':UIView:0x7ffe9c81b720
)>",
    "<NSLayoutConstraint:0x7ffe9aca0130 'UIView-Encapsulated-Layout-Width'
H:[UIView:0x7ffe9c81b720(375)]>"
)

Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x7ffe9acbef60 'saturnWidth ' saturn.width ==
1.5*saturn.height   (Names: saturn:0x7ffe9acb8cb0 )>
```



이제 볼 것은 label과 관련된 뷰임

그래서 이 꼬리표는 Saturn의 bottom과 같은 구속조건을 가지고 있고, 다음 구속조건은 그것을 superView의 top에 묶는 것

그리고 이것이 문제가 되는 이유는 Saturn의 높이가 100 이상이어야 하기 때문
이 제약조건은 Saturn이 100이상 되어야 한다고 말하고 있음

Understanding the Log

Make it easier with identifiers

```
    "<_UILayoutSupportConstraint:0x14630d40 V:[_UILayoutGuide:0x14538610(0)]>",
    "<_UILayoutSupportConstraint:0x14627b90 V:|-(0)-[_UILayoutGuide:0x14538610]
(Names: '|':UIView:0x14538470 )>",
    "<NSLayoutConstraint:0x1464b4d0 'photoHeight' UIImageView:0x14644300.height
== 0.6*UIView:0x14538470.height>",
    "<NSLayoutConstraint:0x1464b530 'captionToCenterY' Caption for
photo.centerY <= UIView:0x14538470.centerY   (Names: Caption for photo:
0x14644ab0 )>",
    "<NSLayoutConstraint:0x1464b0e0 'topVerticalArray' V:[_UILayoutGuide:
0x14538610]-(NSSpace(8))-[UIImageView:0x14644300]>",
    "<NSLayoutConstraint:0x1464b150 'topVerticalArray' V:[UIImageView:
0x14644300]-(NSSpace(8))-[Caption for photo]   (Names: Caption for photo:
0x14644ab0 )>"
```

이렇게 보이면 읽기 쉬움
이를 달성하려면 제한조건에 identifier를 추가하기만 하면 됨

몇 가지 쉬운 방법이 있음
명시적 제약 조건을 사용하는 경우 속성일 뿐

Understanding the Log

Adding identifiers

Use constraint identifiers

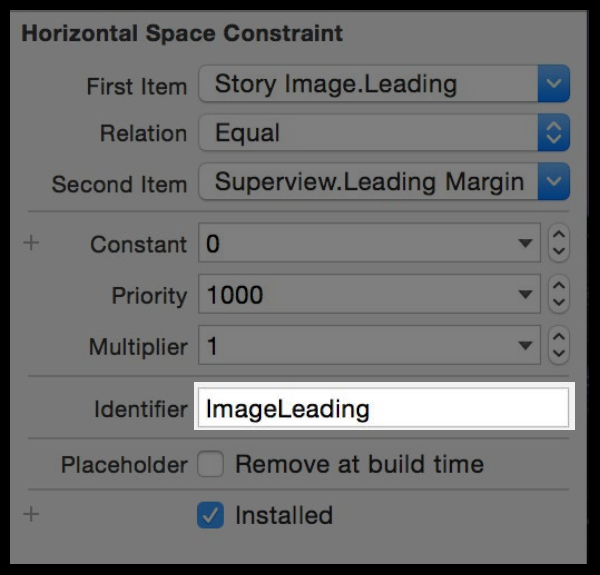
Explicit constraints

```
labelToTop.identifier = @"labelToTop";
```

Constraints using VFL

```
for (NSLayoutConstraint *constraint in verticalLayout)
{
    constraint.identifier = @"verticalLayout";
}
```

Constraints in Interface Builder



identifier의 이름을 코드에서 찾아야 할 경우 나중에 쉽게 찾을 수 있도록 제약 조건의 이름을 지정하는 것과 동일한 것으로 제안함

하지만 원하는 대로 이름을 붙일 수 있으니 맘대로 해도 됨
시각적 형식 언어를 사용하는 경우, 배열을 반환하고 제약 조건을 반환하지 않으므로 해당 배열을 반복하여 모든 제약 조건에 식별자를 설정해야 함

But you can name it anything you want, so go forth and do so. If you are using Visual Format Language, you get an array back, you don't get a constraint back, so you have to loop through that array and set the identifier on every constraint.

배열의 모든 제약 조건에 동일한 식별자를 설정할 수 있고 일반적으로 이 방법이 좋음

여기서 개별 제약 조건을 선택하여 식별자를 설정하고 나중에 배열에서 무언가를 변경하면 순서가 변경되고 identifier 순서도 변경해야 함

인터페이스 빌더는 identifier를 가지고 있기에 더 분별하기 쉬움

Understanding the Log

Tips

Set accessibility identifiers

- Identifies views in logs

Set identifiers on layout guides

Add as you go

View one axis at a time

- `constraintsAffectingLayoutForAxis:` on iOS
- `constraintsAffectingLayoutForOrientation:` on OS X

먼저 뷰에 accesibility identifier를 설정하면 해당 identifier가 해당 뷰와 쌍을 이룬 로그에 표시되므로 찾고 있는 뷰를 찾을 수 있음
이것이 앞에서 본 제약조건에서 Saturn을 얻은 방법
그것은 토성이라고 불리는 accesibility identifier를 가지고 있음

새로운 layout guide에서 identifier를 설정할 수도 있음
단순한 identifier 속성일 뿐 특별한 것은 아님

layout guide를 사용하여 레이아웃을 디버깅하는 것을 매우 쉽게 해줌

하면서 디버깅해라 하나하나 검사하면 오래걸림

뷰가 복잡할 때는 각각의 뷰에 대한 제약조건을 봐라

iOS에서는 Affecting Layout for Axis
OS X에서는 Affecting Layout for Orientation
이걸로 확인하면 한 축 또는 다른 축에서 해당 뷰에 영향을 미치는 제약 조건만 알 수 있음

Understanding the Log

Start from the bottom

Check `translatesAutoresizingMaskIntoConstraints`

Set identifiers

Use `constraintsAffectingLayoutForAxis:`

밑에서 부터 봐라
translatesAutoresizingMastIntoConstraints 다시 한 번 확인해라
identifier 설정해라
constraintsAffectingLayoutForAxis를 활용해라

데릭

Resolving Ambiguity

Mystery #12

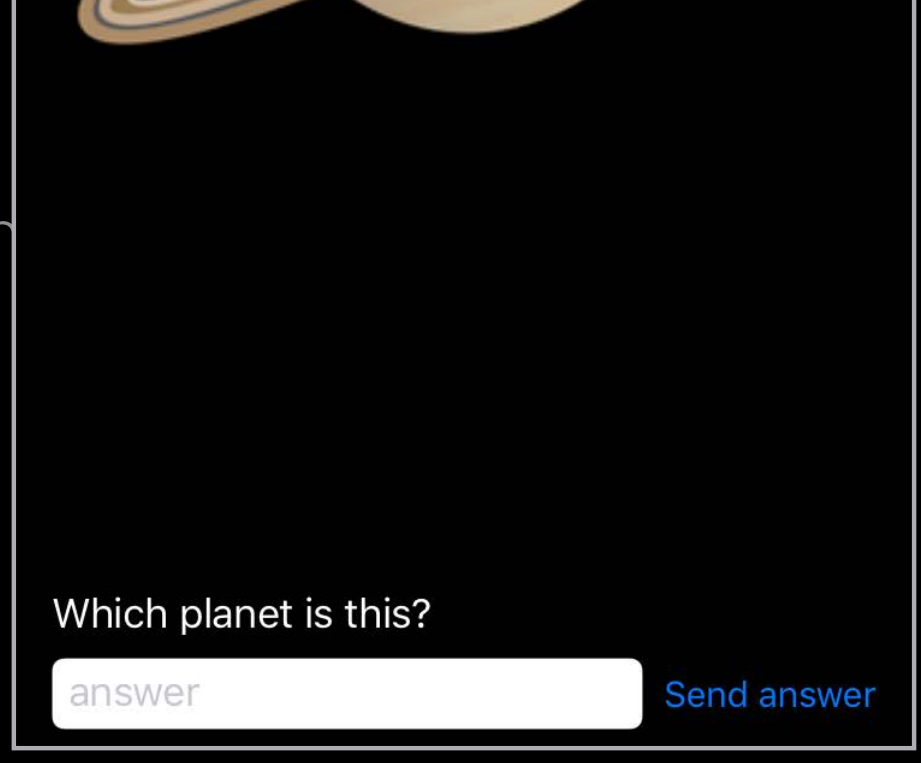
Ambiguity 레이아웃의 몇 가지 원인은 제약 조건이 너무 적기 때문이다.

Ambiguous Layouts

Why doesn't my layout look righ

Possible causes

- Too few constraints
- Conflicting priorities



어디에 둘지 모르면 그냥 어딘가에 둘 것이다.
제약 조건들을 추가해야 한다.

모호한 레이아웃의 또 다른 원인은 충돌하는 우선순위이다.
Hugging 우선 순위가 동일하기 때문에 위의 상황이 발생한다.

Ambiguous Layouts

Why doesn't my layout look right?

Possible causes

- Too few constraints
- Conflicting priorities

answer Send answer

어디에 둘지 모르면 그냥 어딘가에 둘 것이다.
제약 조건들을 추가해야 한다.

모호한 레이아웃의 또 다른 원인은 충돌하는 우선순위이다.

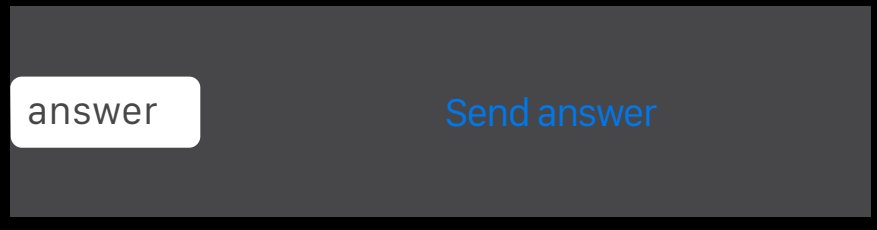
Ambiguous Layouts

Why doesn't my layout look right?

Possible causes

- Too few constraints
- Conflicting priorities

Content hugging priorities



Both:
contentHuggingPriority = 250
compressionResistancePriority = 750

둘다 250으로 우선 순위를 주면 다른 방법이 없다.
엔진이 이러한 뷰의 크기를 수평으로 조정하는 다른 방법을 지시하지 않는다.

텍스트 뷰가 아닌 버튼의 레이블의 Hugging을 조정하고 싶었다.

Ambiguous Layouts

Why doesn't my layout look right?

Possible causes

- Too few constraints
- Conflicting priorities

Content hugging priorities

Both:
contentHuggingPriority = 250
compressionResistancePriority = 750

텍스트 필드의 우선 순위보다 낮게 설정되어 있으면 뷰의 가장자리가 콘텐츠에서 멀어질 수 있다.

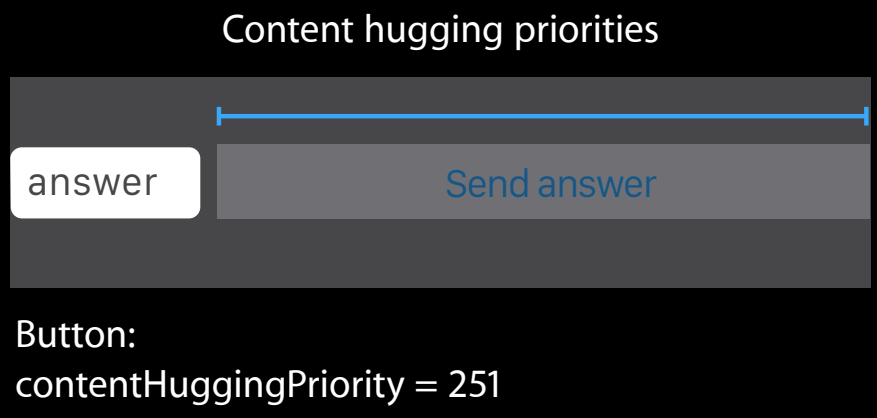
콘텐츠의 Hugging을 더 끌어당기는 것이 덜 중요하다고 엔진에 알릴 경우

Ambiguous Layouts

Why doesn't my layout look right?

Possible causes

- Too few constraints
- Conflicting priorities



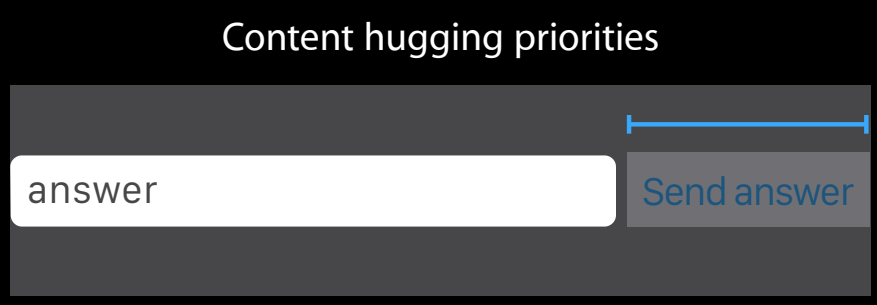
텍스트 뷰의 콘텐츠 Hugging 우선순위를 위처럼 설정하면 이제 버튼이 콘텐츠를 Hugging에 맞춰 텍스트 필드가 늘어난다.

Ambiguous Layouts

Why doesn't my layout look right?

Possible causes

- Too few constraints
- Conflicting priorities

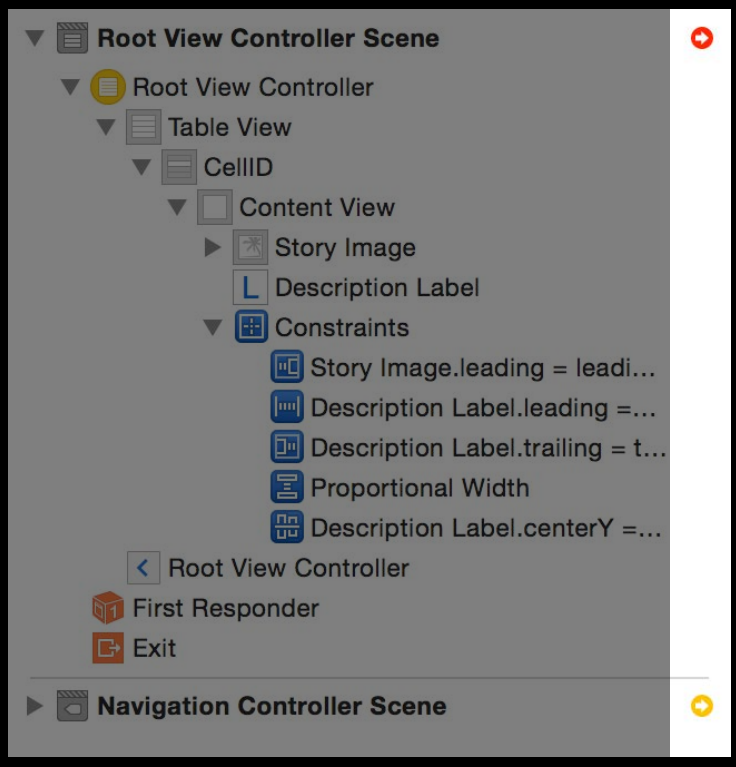


우선순위를 올바르게 설정하면 실행되는 이러한 모호한 레이아웃 중 일부를 해결할 수 있다.

Resolving Ambiguity

Diagnostic tools

Red and yellow icons in IB

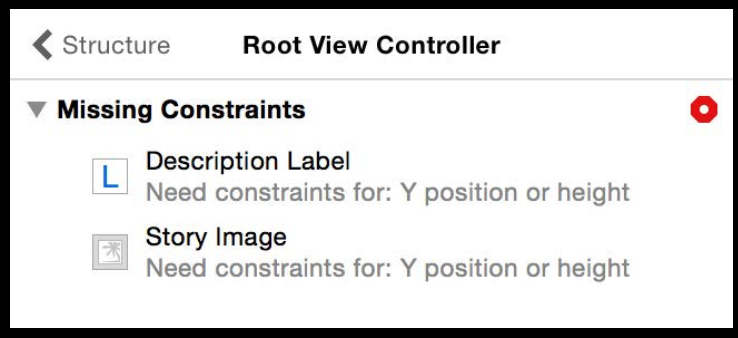


붉은색 아이콘을 클릭하면 엔진이 이해하지 못하는 레이아웃의 상황을 알려준다.

Resolving Ambiguity

Diagnostic tools

Red and yellow icons in IB



Y Position 또는 높이에 대한 제약 조건이 필요하다고 알려준다.

이 문제가 있는 앱을 빌드하고 실행할 때 Y축 어딘가에 이러한 뷰가 표시될 것이다. 여기서 엔진은 정보가 없기 때문에 이동해야 한다고 결정했다.

Resolving Ambiguity

Diagnostic tools

Red and yellow icons in IB

`_autolayoutTrace`

```
(lldb) po [self.view _autolayoutTrace]

UIWindow:0x7fe7434a3fe0
|    •UIView:0x7fe7434a8140
|    |    *_UILayoutGuide:0x7fe7434a84f0
|    |    *_UILayoutGuide:0x7fe7434a90d0
|    |    *Mercury:0x7fe7434a7790
|    |    *Venus:0x7fe743639380
|    |    *Earth:0x7fe74363aae0
|    |    *Mars:0x7fe74363bed0
|    |    *Jupiter:0x7fe74363ce30
|    |    *Saturn:0x7fe74363e220- AMBIGUOUS LAYOUT for Saturn.minX{id: 165}
|    |    *Uranus:0x7fe74363f690
|    |    *Neptune:0x7fe743640d60

Legend:
    * - is laid out with auto layout
    + - is laid out manually, but is represented in the layout engine
because translatesAutoresizingMaskIntoConstraints = YES
    • - layout engine host

(lldb)
```

Interface Builder를 사용하지 않거나 통과했지만 여전히 이 문제를 겪고 있을 때 autolayoutTrace라는 방법이 있다.

뷰의 디버거에서 이 방법을 사용하면 모두 대문자로 알려준다.

-> 레이아웃이 모호한 뷰가 있고 해당 뷰의 문제를 진단할 수 있다.

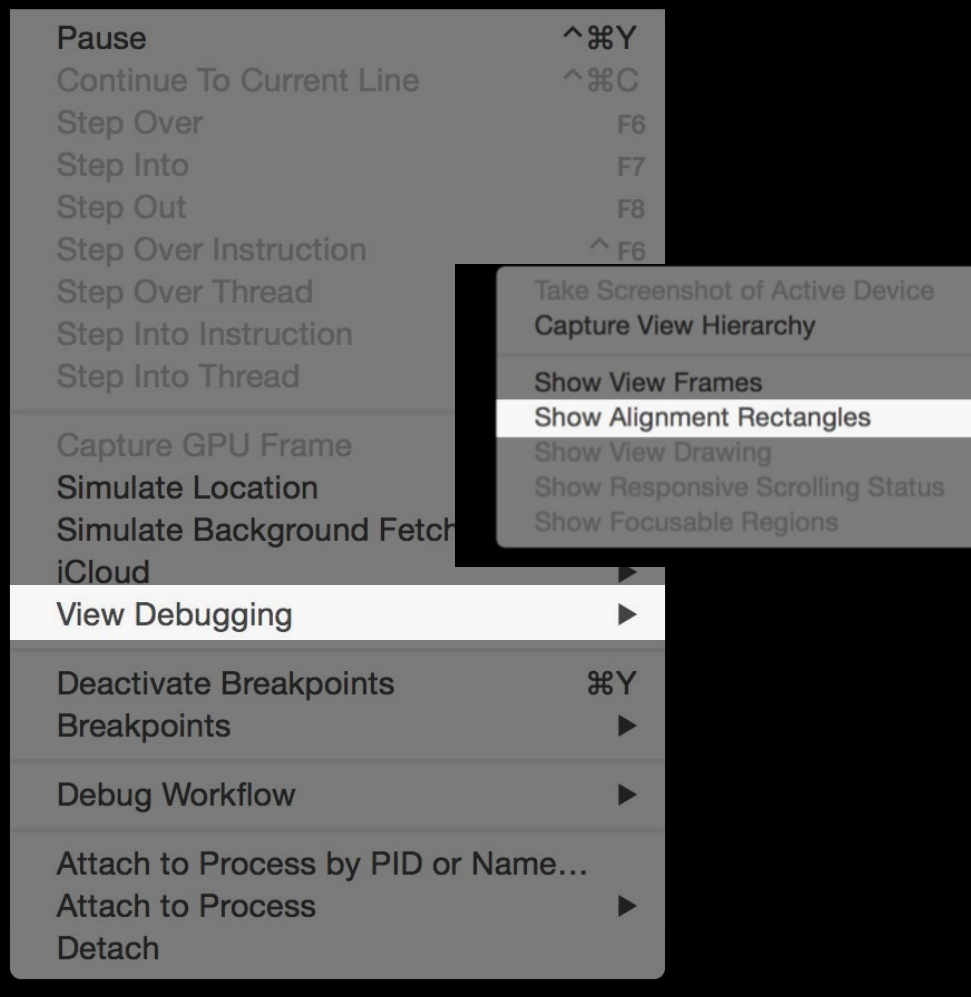
Resolving Ambiguity

Diagnostic tools

Red and yellow icons in IB

`_autolayoutTrace`

Select Debug > View Debugging



또한 디버그 메뉴에 뷰 디버거가 있어 레이아웃 엔진이 뷰에 대해 계산한 프레임 및 정렬 기록을 확인할 수 있다.

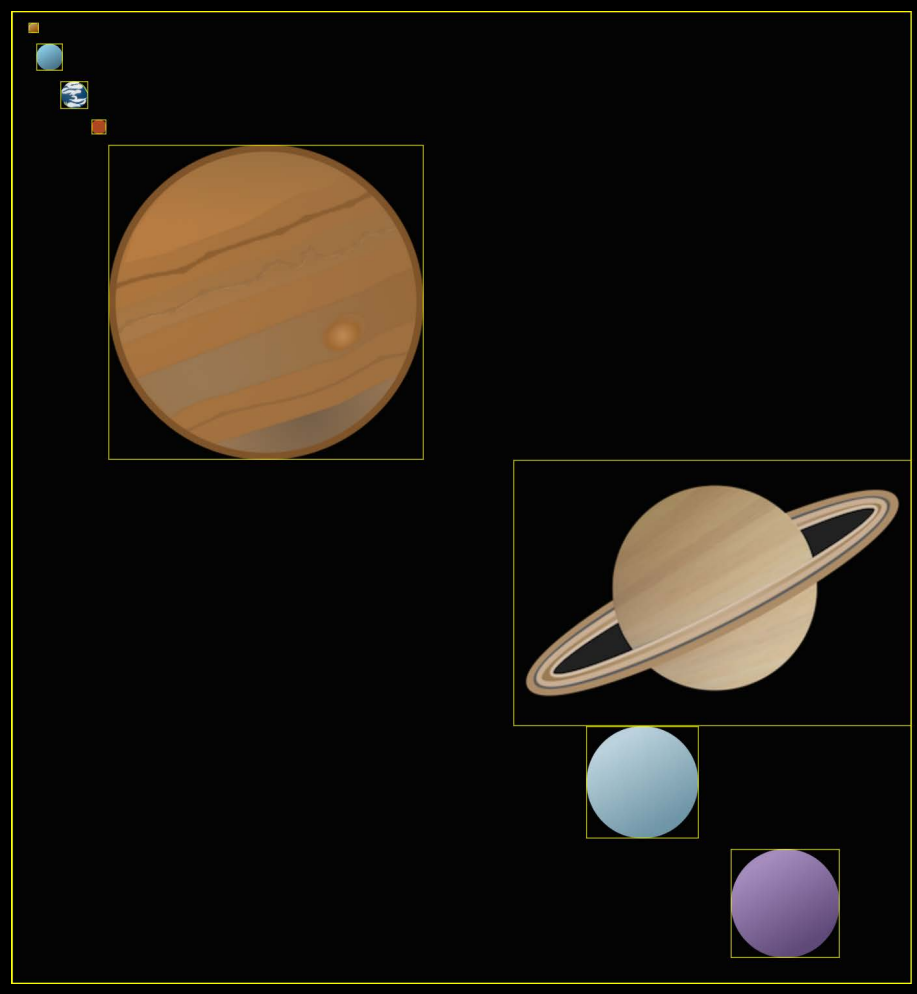
Resolving Ambiguity

Diagnostic tools

Red and yellow icons in IB

`_autolayoutTrace`

Select Debug > View Debugging



내가 원했던 것이 아니기 때문에 문제이다.

Resolving Ambiguity

Diagnostic tools

Red and yellow icons in IB

`_autolayoutTrace`

Select Debug > View Debugging

Look in the view debugger

뷰 디버거를 통해 확인한다.

Resolving Ambiguity

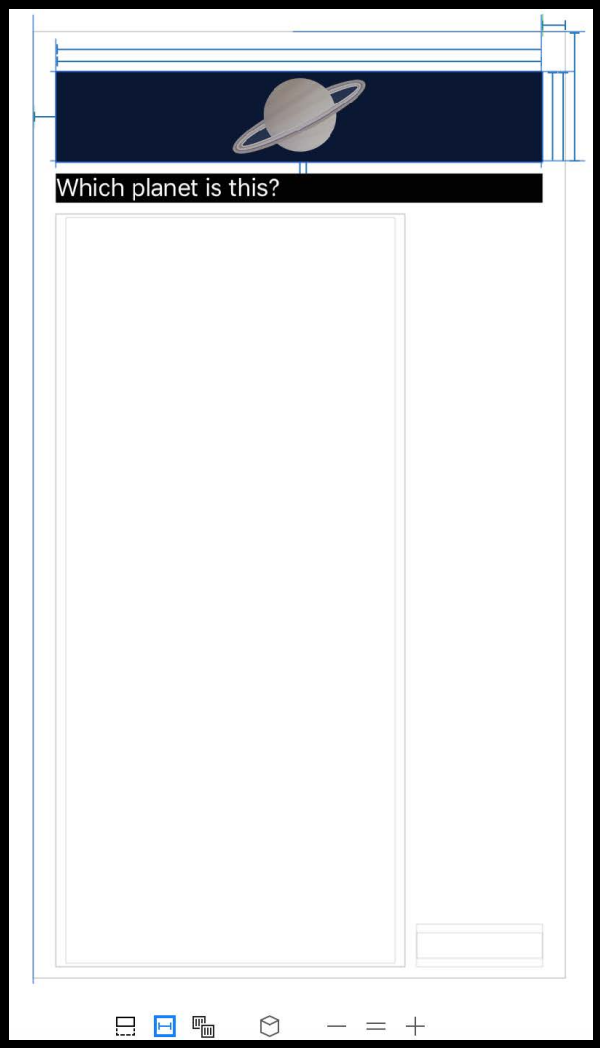
Diagnostic tools

Red and yellow icons in IB

`_autolayoutTrace`

Select Debug > View Debugging

Look in the view debugger



뷰 디버거를 누르면 레이아웃이 표시되고 제약 조건을 볼 수 있다. 또한 뷰의 와이어프레임만 3D로 볼 수 있다.

Resolving Ambiguity

Diagnostic tools

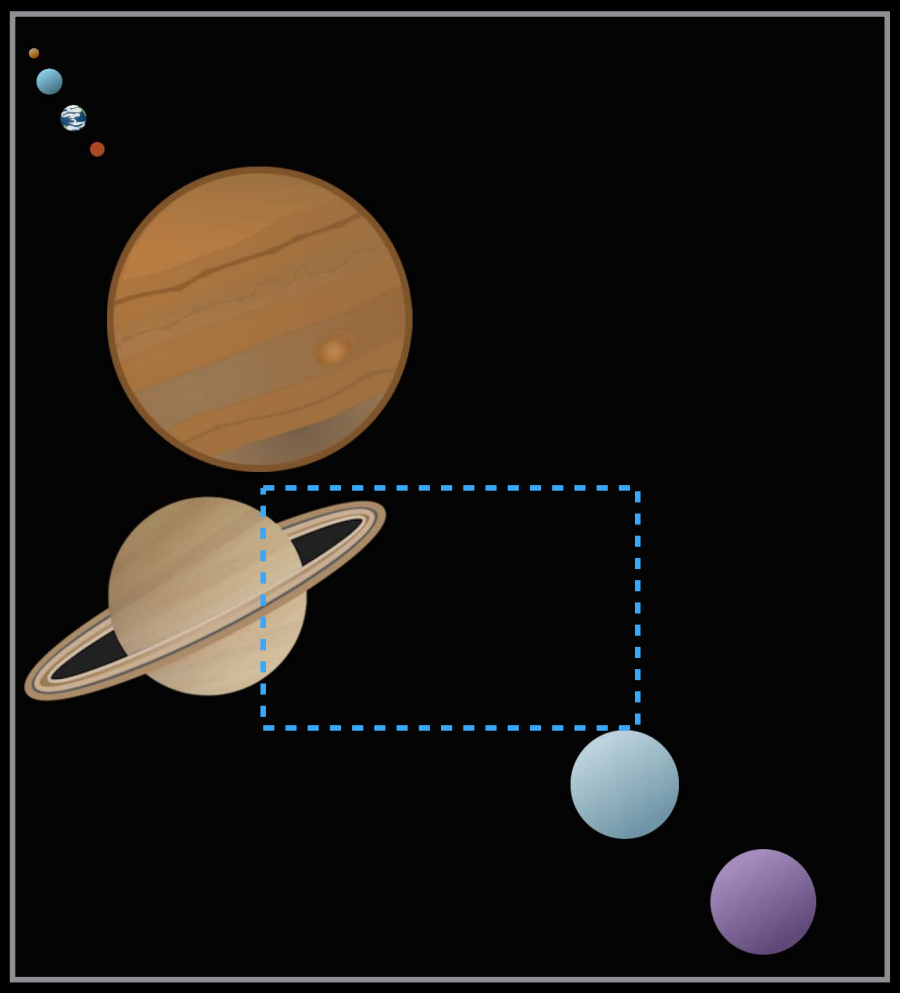
Red and yellow icons in IB

`_autolayoutTrace`

Select Debug > View Debugging

Look in the view debugger

`exerciseAmbiguityInLayout`



모호한 뷰가 있고 디버거의 해당 뷰에서 이 뷰를 실행하고 계속하면 레이아웃 엔진이 다른 솔루션을 표시한다. -> 문제가 있는 위치를 파악하려고 할 때 유용한 단서.

```
(lldb) po [self.view _autolayoutTrace]
```

LLDB를 통해 모호한 뷰가 어떤 것인지 확인 할 수 있다.

Debugging Your Layout

Think about what information the engine needs

Use the logs when constraints are unsatisfiable

- Add identifiers for constraints and views

Check for ambiguity regularly

Use tools to help resolve issues

- Icons in Interface Builder
- View debugger
- Methods in lldb

- 엔진이 필요한 정보에 대해 생각하라
  제약 조건이 충족되지 않으면 로그를 사용하라
  실행에서 반드시 볼 수 있는 것은 아니다
  단위 테스트와 같은 것을 넣고 정기적으로 모든 뷰에서 실행하는 것이 좋다.
  Interface Builder
  뷰 디버거
  LLDB

PPT 페이지 명시할 것 피피티 페이지 링크 걸수 있음
상세한 것은 각 페이지에 주석으로 달기

	궁금한 점	이해 안된 점
보리		

PPT 페이지 명시할 것 피피티 페이지 링크 걸수 있음
상세한 것은 각 페이지에 주석으로 달기

	궁금한 점	이해 안된 점
바드	Visual Format Language	

PPT 페이지 명시할 것 피피티 페이지 링크 걸수 있음
상세한 것은 각 페이지에 주석으로 달기

	궁금한 점	이해 안된 점
데릭		