Interface Builder

- 코드를 작성하지 않고도 전체 user interface를 간단하게 디자인할 수 있음
- windows, buttons, text fields 등의 objects를 끌어다 놓기만 하면 작동하는 user interface를 만들 수 있음

> Storyboards, Assistant, Auto Layout, Preview
> https://developer.apple.com/xcode/interface-builder/

Interface Builder에서 사용되는 속성이라고 인식 하기 위해 내부적으로 변환되어 동작

```
#ifndef IBOutlet
#define IBOutlet
#endif


#ifndef IBAction
#define IBAction void
#endif
```

CGPoint

- A structure that contains a point in a two-dimensional coordinate system.

Declaration

```
struct CGPoint
```

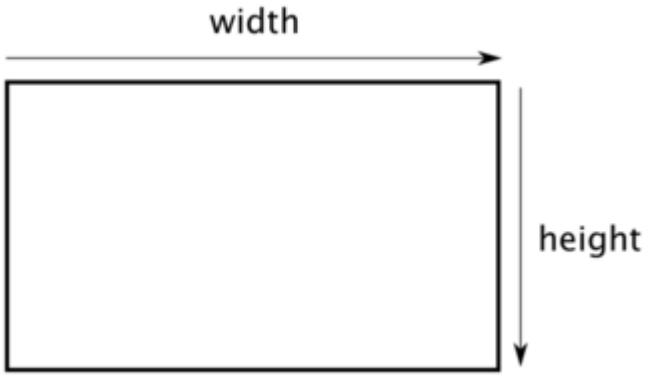
Instance Properties

```
var x: CGFloat // The x-coordinate of the point.
var y: CGFloat // The y-coordinate of the point.


public struct CGPoint {
    public var x: CGFloat
    public var y: CGFloat
    public init()
    public init(x: CGFloat, y: CGFloat)
}
```

CGSize

- A structure that contains width and height values.



Declaration

```
struct CGSize
```

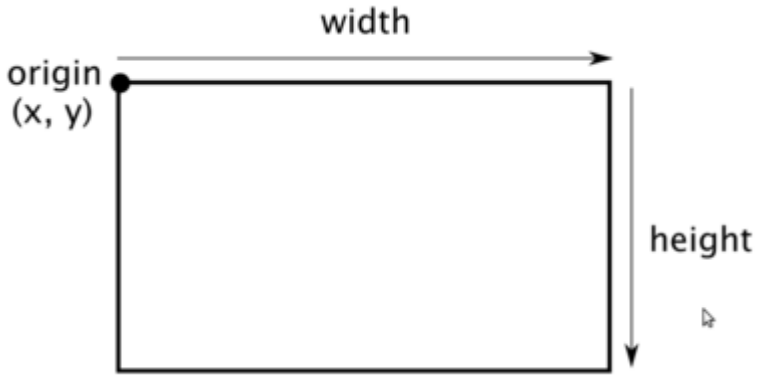
Instance Properties

```
var width: CGFloat // A width value.
var height: CGFloat // A height value.


public struct CGSize {
    public var width: CGFloat
    public var height: CGFloat
    public init()
    public init(width: CGFloat, height: CGFloat)
}
```

CGRect

- A structure that contains the location and dimensions of a rectangle.



Declaration

```
struct CGRect
```

Instance Properties

```
var origin: CGPoint // A point that specifies the coordinates of the rectangl
var size: CGSize // A size that specifies the height and width of the rectang
```

```
public struct CGRect {
    public var origin: CGPoint
    public var size: CGSize
    public init()
    public init(origin: CGPoint, size: CGSize)
}
```

frame

- The frame rectangle, which describes the view's location and size in its **superview's** coordinate system.

Declaration

```
var frame: CGRect { get set }
```

bounds

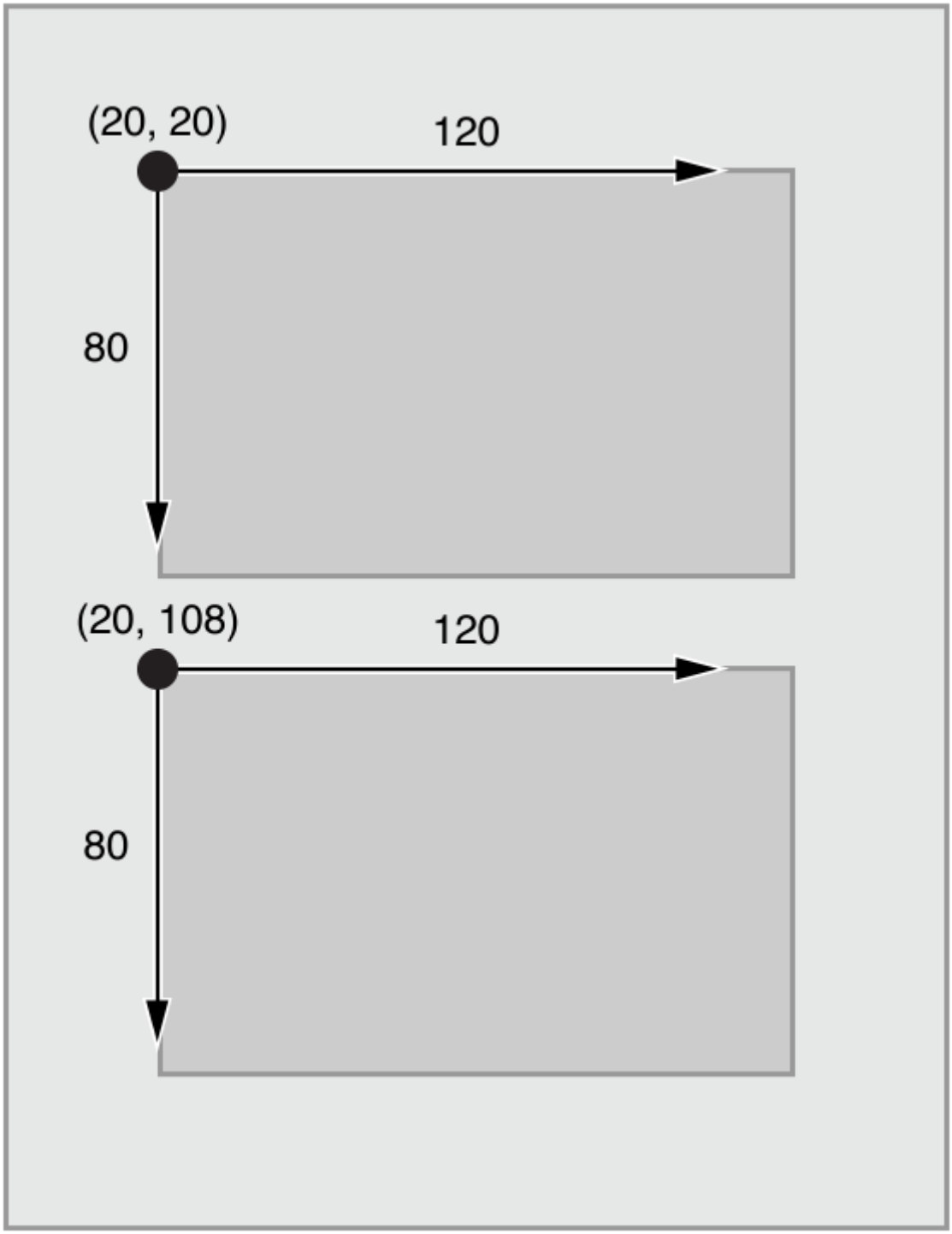
- The bounds rectangle, which describes the view's location and size in **its own** coordinate system.

Declaration

```
var bounds: CGRect { get set }
```

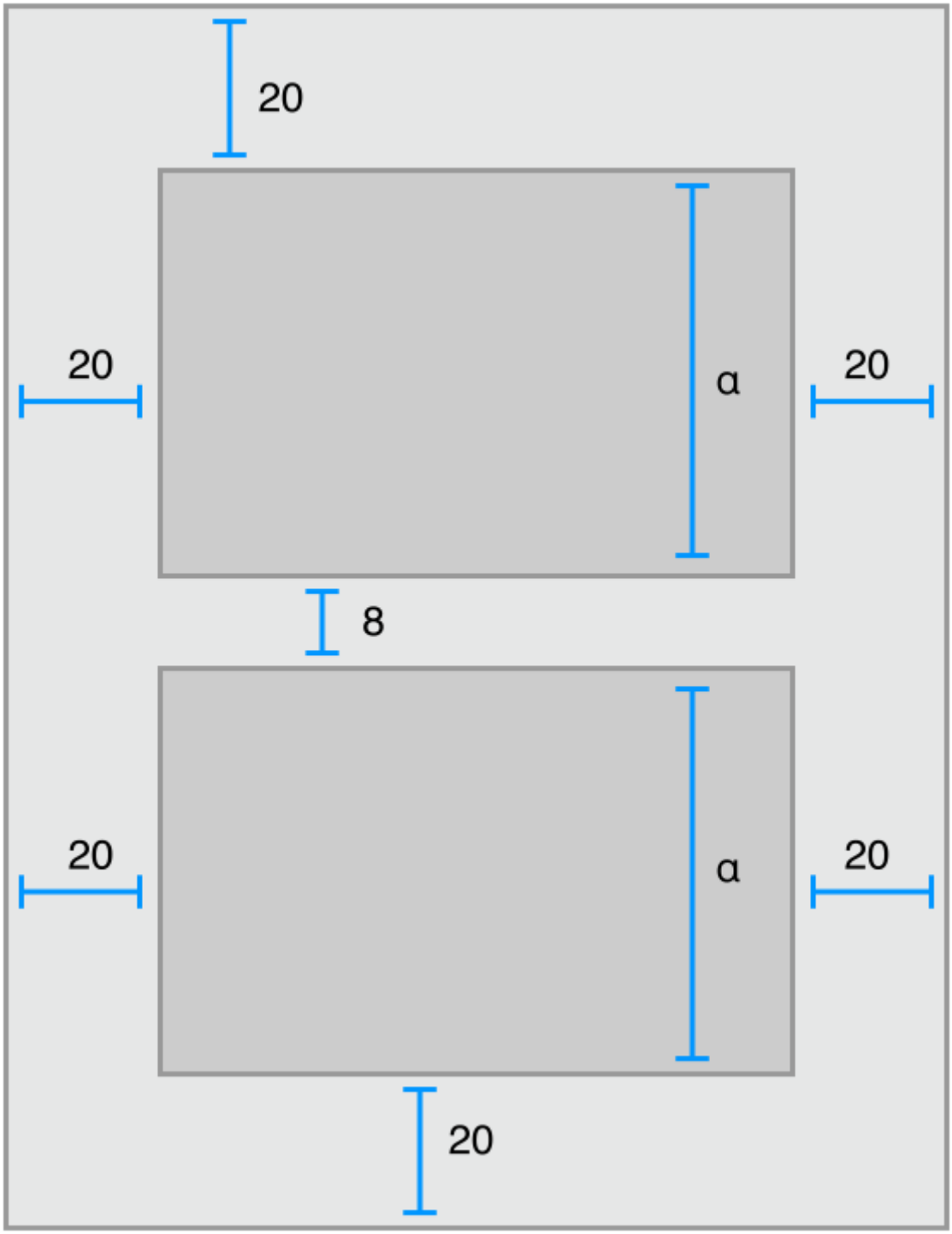
Frame-Based Layout

- 뷰 계층 구조에서 각 뷰마다 frame을 설정하여 user interface를 배치
- 어떤 뷰의 크기나 위치가 바뀌면 영향 받는 뷰들의 frame을 재계산해야 함.
- 간단한 user interface라도 설계, 디버그, 유지관리에 상당한 비용 발생
- origin(x, y) / height / width



Auto Layout

- 뷰의 frame 대신, 뷰들의 relationships을 생각하는 방식
- Auto Layout은 일련의 constraints를 사용하여 user interface를 정의
  - Constraints는 일반적으로 2개의 view 사이의 관계를 타냄
- Constraints에 기반하여 각각의 뷰는 크기와 위치를 계산



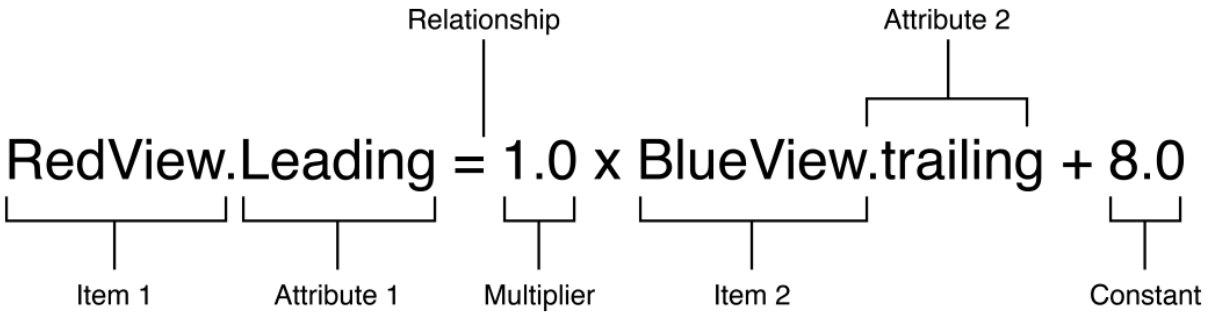
Auto Layout을 왜 쓰는가

1. 화면 크기와 디바이스 방향에 따라 유연하게 업데이트 되는 UI를 비교적 쉽게 구현
2. 향후 새로운 해상도의 디바이스가 출시되더라도 업데이트 없이 일관된 UI를 유지 가능
3. 화면 좌표를 직접 계산하거나 수많은 분기 코드를 작성할 필요가 없음
4. 우선 순위와 활성화 속성을 활용하여 특정 조건에 따라 업데이트 되는 UI를 구현 가능
5. 지역화 문자열을 사용할 때 문자열의 너비에 따라 버튼이나 레이블의 너비가 자동으로 업데이트
6. Content Hugging과 Compression Resistance의 우선 순위를 조절하여 동적인 UI를 더욱 세부적으로 제어 가능
7. 뷰 애니메이션, 모션 이펙트와 함께 사용 가능

8. 동일한 계층구조에 존재하지 않는 뷰 사이의 관계를 설정 가능
9. 스토리보드에서 제약을 쉽게 추가할 수 있으며, 코드를 통해 런타임에 동적으로 추가하거나 제거 가능

Constraint

- 일련의 선형 방정식으로 정의 가능



NSLayoutConstraint

- Autolayout 환경에서 충족되어야할 두개의 user interface 객체 사이의 관계
- NSLaytoutConstraint

fristItem

- The first object participating in the constraint.

```
unowned(unsafe) var firstItem: AnyObject? { get }
```

firstAttribute

- The attribute of the first object participating in the constraint.

```
var firstAttribute: NSLayoutConstraint.Attribute { get }
```

relation

- The relation between the two attributes in the constraint.

```
var relation: NSLayoutConstraint.Relation { get }
```

secondItem

- The second object participating in the constraint.

```
unowned(unsafe) var secondItem: AnyObject? { get }
```

secondAttribute

- The attribute of the second object participating in the constraint.

```
var secondAttribute: NSLayoutConstraint.Attribute { get }
```

multiplier

- The multiplier applied to the second attribute participating in the constraint.

```
var multiplier: CGFloat { get }
```

constant

- The constant added to the multiplied second attribute participating in the constraint.

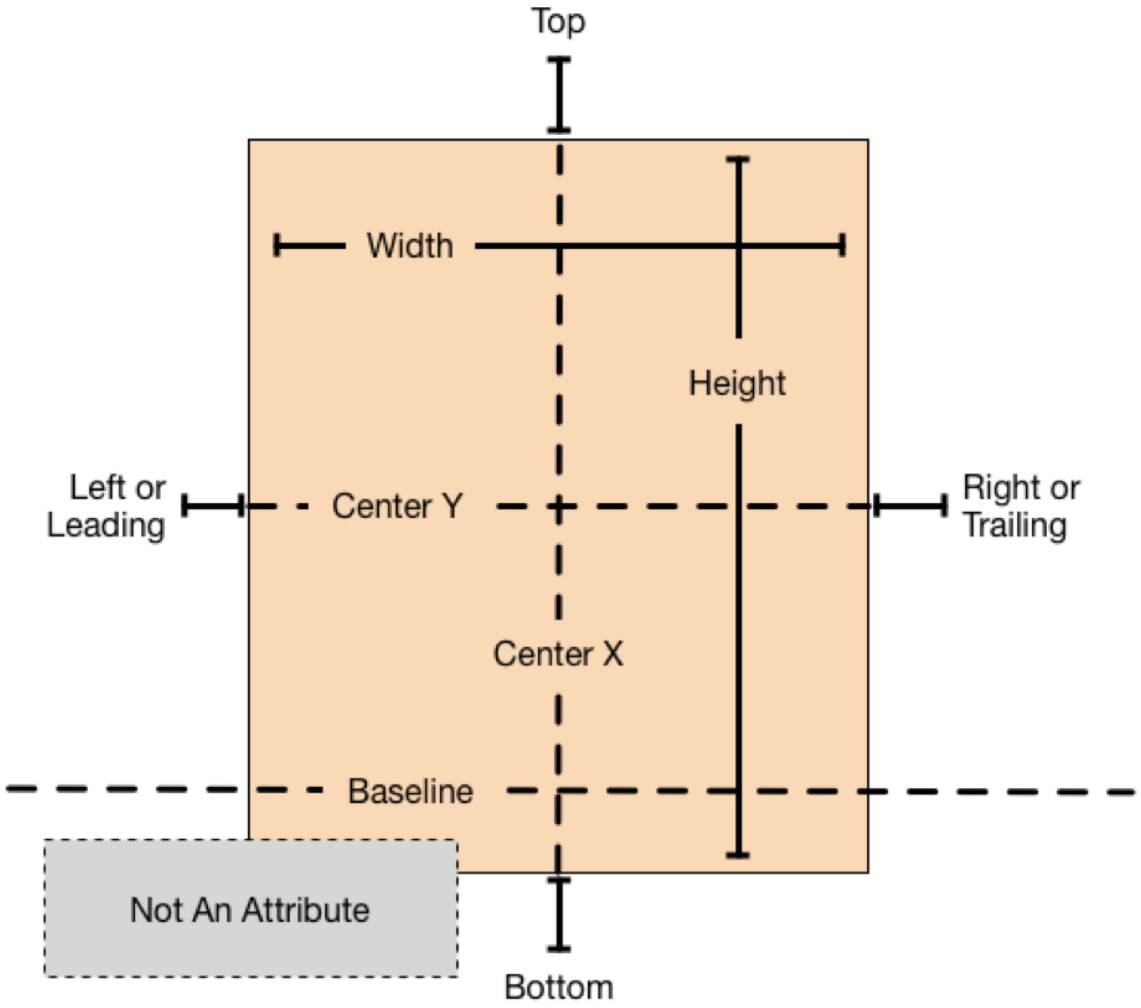
```
var constant: CGFloat { get set }
```

Auto Layout Attributes

- four edges (leading / trailing / top / bottom)
- height, width
- centerX(horizontal), centerY(vertical)
- baseline

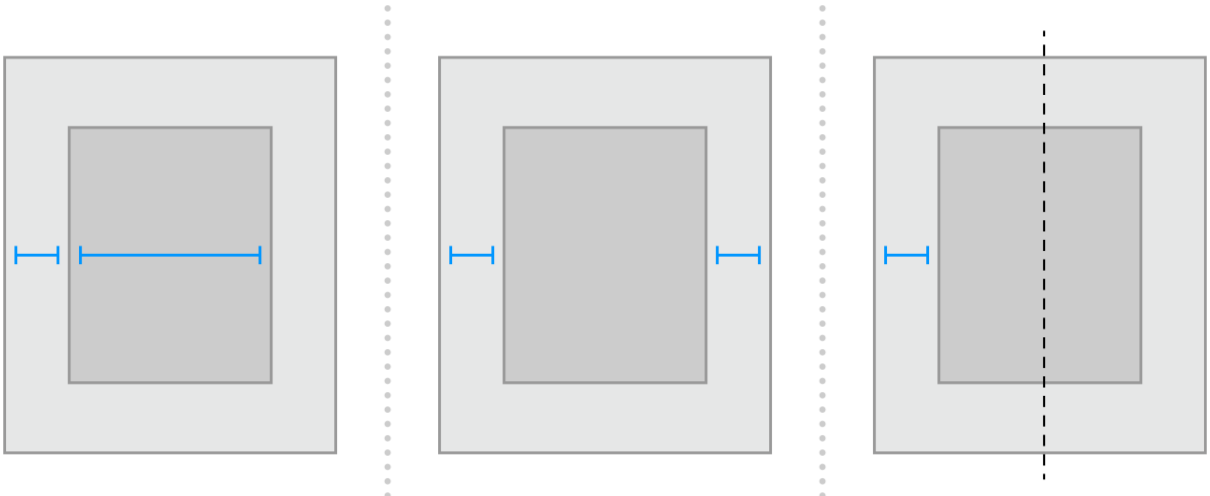
- Not An Attribute

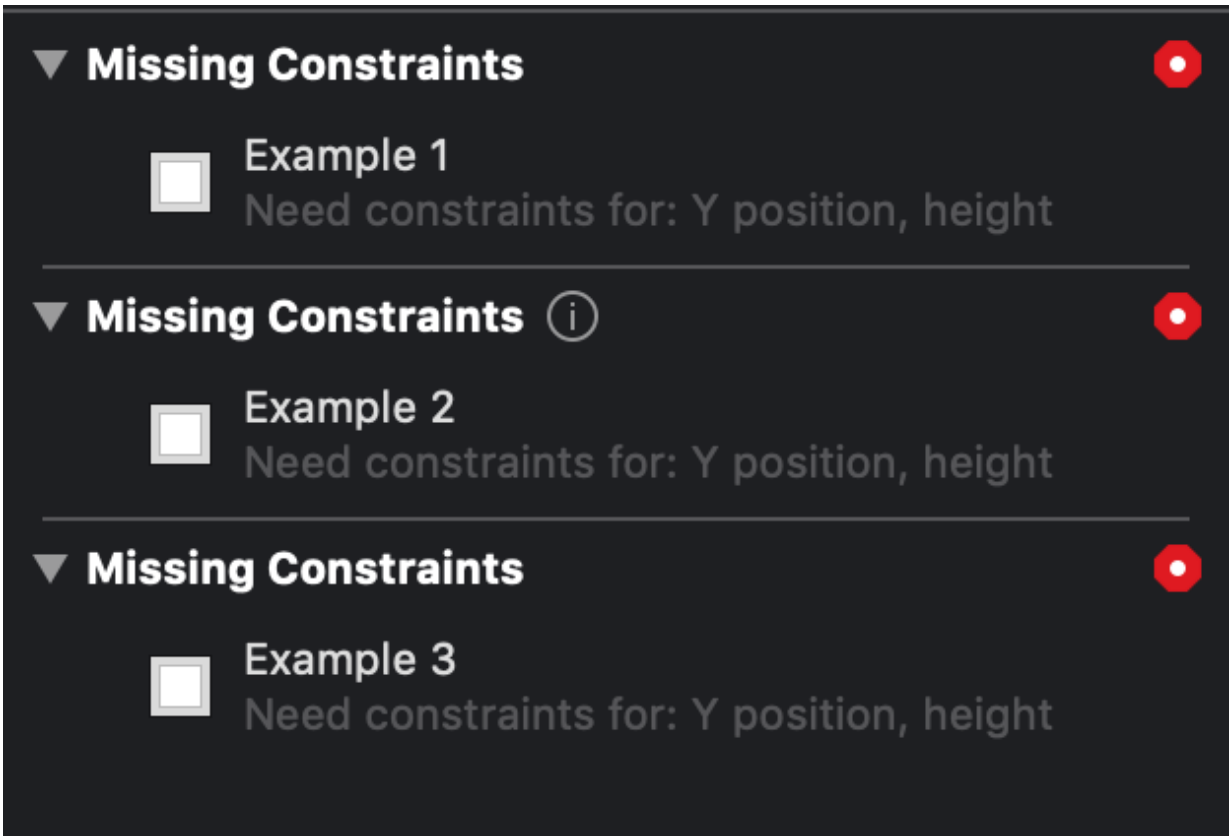
- NSLaytoutAttribute



Creating Nonambiguous, Satisfiable Layouts

- Constraint는 각 뷰의 사이즈와 위치가 정의되어야만 함.



- 
Missing Constraints
Example 1
Need constraints for: Y position, height
Missing Constraints
Example 2
Need constraints for: Y position, height
Missing Constraints
Example 3
Need constraints for: Y position, height