Auto Layout

기존의 Frame-Based Layout**과 다른** View**들 간의 관계를 이용하여**
View**의 위치와 크기를 자동으로 결정하는 레이아웃 시스템**

Constraint

- Equation
- Leading, Top, Trailing, bottom
- Width, Height
- centerX, centerY
- AspectRatio
- Baseline
- Relation
- Multiplier
- Constant
- Priority
- Intrinsic Content Size
- Content Hugging
- Compression Resistance

NSlayoutConstraint

```
func loadDefaultSetting() {
  blueView.translatesAutoresizingMaskIntoConstraints = fa

  let leading = NSLayoutConstraint(item: blueView, attribu
  let top = NSLayoutConstraint(item: blueView, attribute:
  let trailing = NSLayoutConstraint(item: blueView, attri
  let height = NSLayoutConstraint(item: blueView, attribu
  NSLayoutConstraint.activate([leading, top, trailing, he

  topToSuperview = top

  if #available(iOS 11.0, *) {
     topToSafeArea = NSLayoutConstraint(item: blueView, a
  } else {
     topToSafeArea = NSLayoutConstraint(item: blueView, a
  }
}
```

Visual Format Language

```
func loadDefaultSetting() {
  blueView.translatesAutoresizingMaskIntoConstraints = fa

  let horzVfl = "|[blue]|"
  let vertVfl = "V:|[blue(100)]"

  let views: [String: Any] = ["blue": blueView]

  let horzConstraints = NSLayoutConstraint.constraints(wi
  let vertConstraints = NSLayoutConstraint.constraints(wi
  NSLayoutConstraint.activate(horzConstraints + vertConst
}
```

Layout Anchor

```
func loadDefaultSetting() {
  blueView.translatesAutoresizingMaskIntoConstraints = fa

  blueView.leadingAnchor.constraint(equalTo: view.leading
  blueView.trailingAnchor.constraint(equalTo: view.traili
  blueView.heightAnchor.constraint(equalToConstant: 100).

  topToSuperview = blueView.topAnchor.constraint(equalTo:
  topToSuperview?.isActive = true

  if #available(iOS 11.0, *) {
        topToSafeArea = blueView.topAnchor.constraint(equa
  } else {
        topToSafeArea = blueView.topAnchor.constraint(equa
  }
}
```

Autolayout 관련 라이브러리

Snapkit

TinyConstraints

Cartography

Snapkit

```
func loadDefaultSetting() {
  let margin: CGFloat = 10.0

  redView.snp.makeConstraints { target in
     target.leading.equalTo(view).offset(margin)
     target.top.equalTo(view).offset(margin)
     target.trailing.equalTo(blueView.snp.leading).offset
     target.bottom.equalTo(yellowView.snp.top).offset(-ma
  }

  blueView.snp.makeConstraints { target in
     target.top.equalTo(view).offset(margin)
     target.trailing.equalTo(view).offset(-margin)
     target.bottom.equalTo(blackView.snp.top).offset(-mar
     target.size.equalTo(redView)
  }

  // ...
}
```

Auto Layout GuideLines

- View Bounds, Frame, Center 값으로 View 위치를 정하지 마라
- Stack View를 사용하라
- 가급적 View 높이나 너비를 고정하는 것을 피하라
- Auto Layout 사용하는 이유는 동적으로 변화때문에 사용함. 필요하면 최소 크기, 최대 크기를 정하라
- Right, Left 대신 Leading, Trailing Constraint를 사용하라
- 코드로 View를 생성할 경우 translatesAutoresizingMaskIntoConstraints Property를 NO로 설정하라. 자동으로 설정하는 Constraint와 커스터 Constraint가 충돌할 가능성이 생겨 미완성 레이아웃이 될 수 있음

- Horizontal Constraints
  - 시스템은 디바이스와, 앱이 뷰 컨트롤러를 보여 주는 방식을 기반해서 자동으로 거리를 맞춤
  - margin을 포함해 루트 뷰를 채우는 텍스트 객체의 경우 layout margin 대신 가독성 있는 컨텐트 가이드를 사용하라
  - 배경 이미지처럼 루트 뷰를 모두 채우는 아이템의 경우 뷰의 leading과 trailing edge를 사용하라
- Vertical Constraints
  - 뷰가 bar 아래를 확장하는 경우 top과 bottom margin을 사용하라
  - 뷰가 bar 아래를 확장하지 않으면 레이아웃 가이드의 top과 bottom으 로부터 constraint를 만들라

참고

Autolayout Guide

Auto Layout Cookbook

Auto Layout Tutorial in iOS 11: Getting Started

Adaptivity and Layout