# Mysteries of Auto Layout, Part 1

데릭

보리입니다 데스

바드

1~ 79 페이지 보리
80~139 페이지 데릭
140~189 페이지 바드

D2Coding으로 할 것

보리

# Maintainable Layouts

## Mystery #1

# Many Constraints


수많은 제약 사항이 존재함.

누가 여기서 새로 별점 아래에 컨트롤러를 넣어달라한다면?

해당 영역의 컨트롤을 검사하고 제약 조건을 살펴보고, 버튼을 삽입하고, 일부를 부수고, 다시조립해야 한다.

> 이런 경우 StackView를 쓰면 매우 편리하다.
>
> 뷰 들을 선형으로 관리할 때 좋다.

# Stack View

NEW

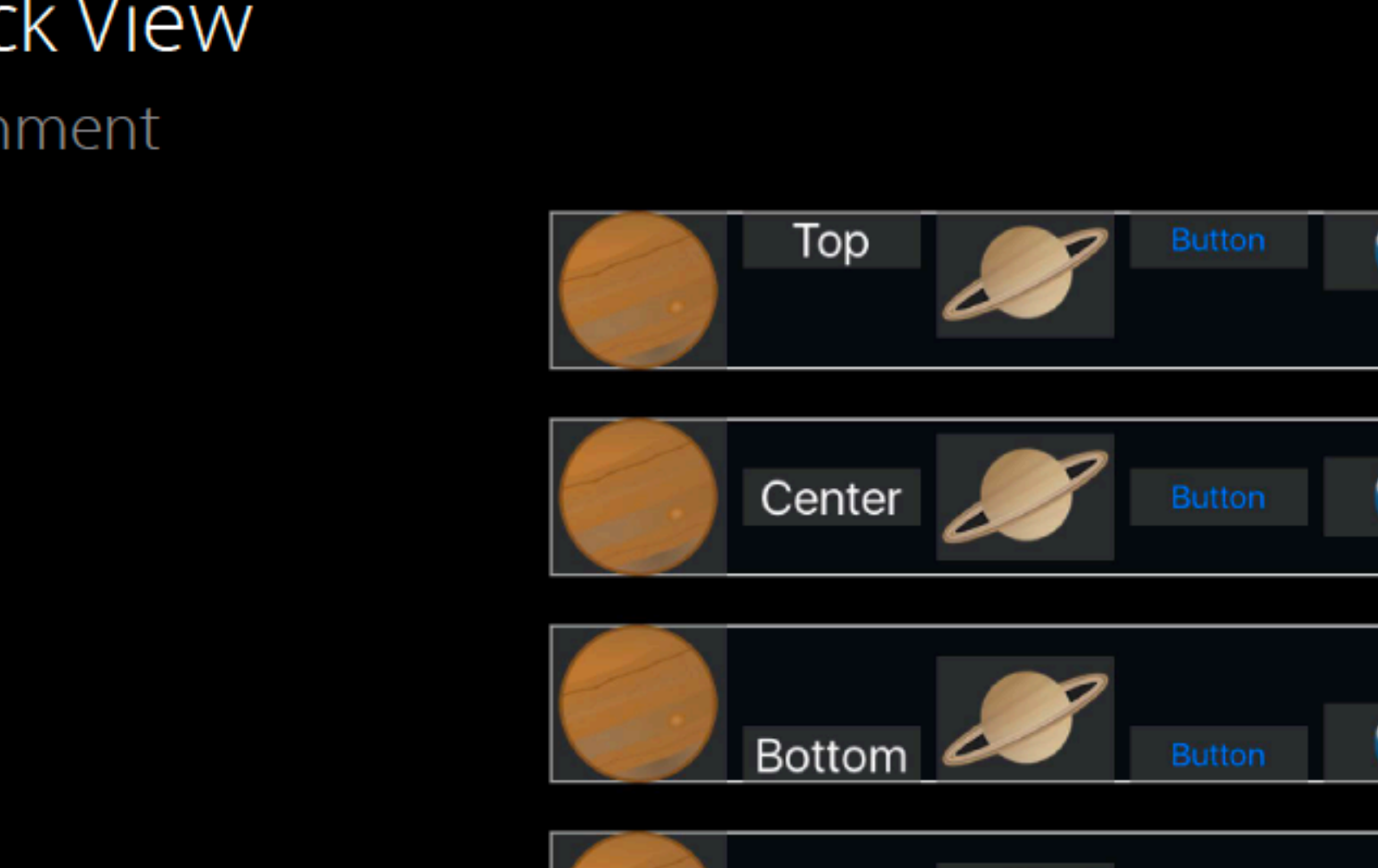
Built with Auto Layout

Manages constraints

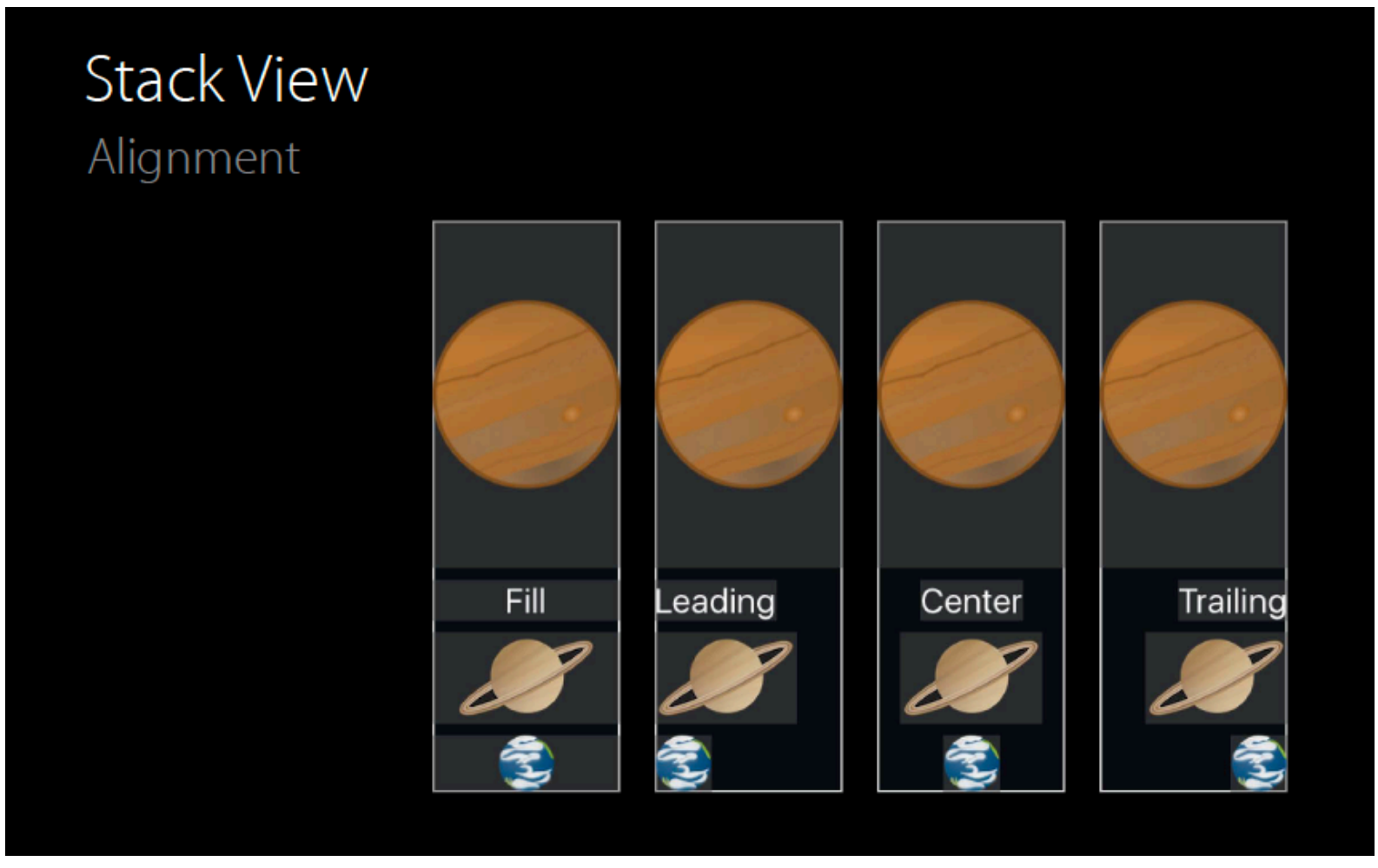
Horizontal or vertical

수평 또는 수직방향이나 축은 물론 정렬과 같은 기타 사용자 정의 가능한 속성을 가질 수 있따.

자식 레이아웃을 알아서 관리하기 때문에 제약을 간소화 할 수 있다.



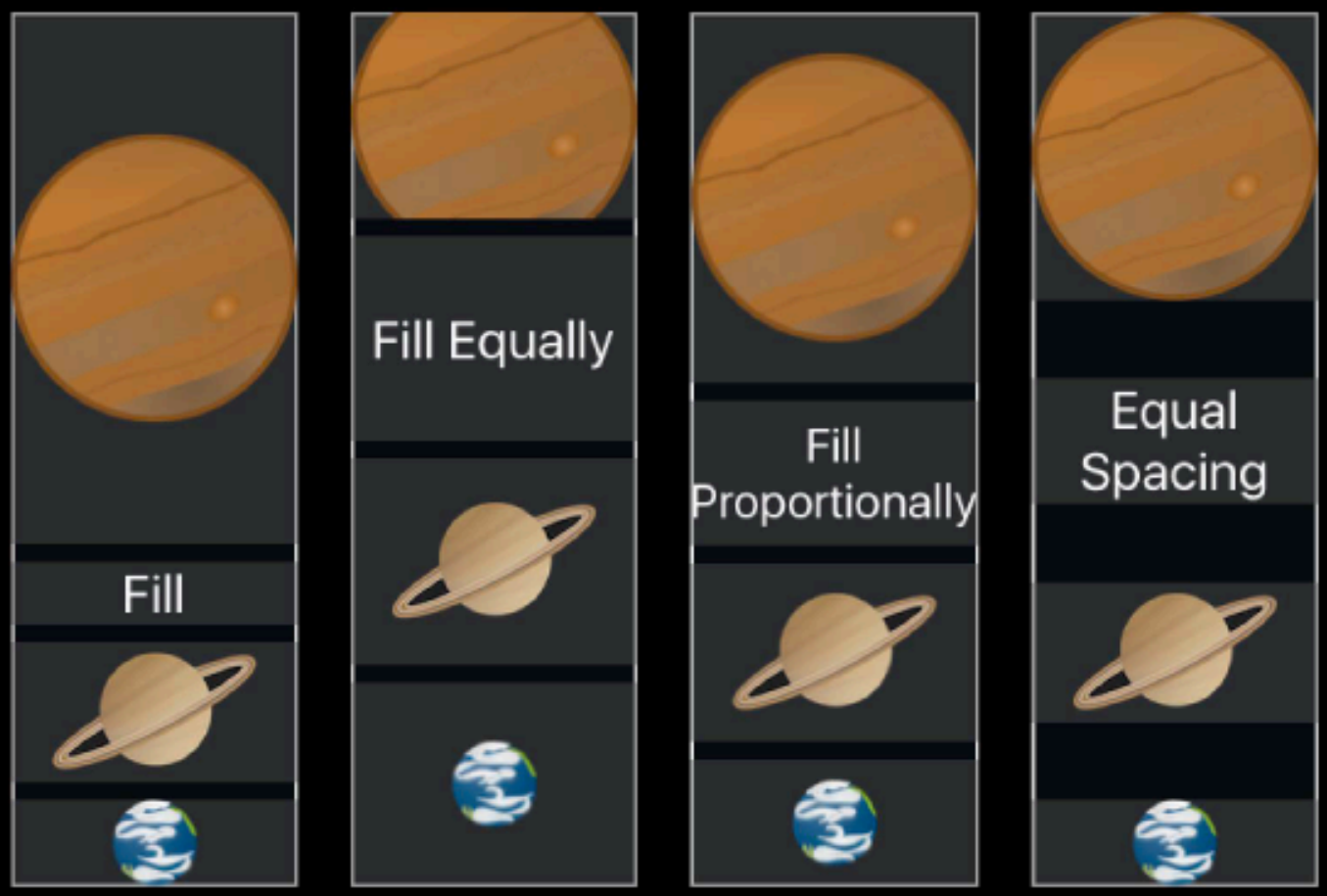

정렬은 내부 자식 요소들이 어떻게 정렬될지에 대한 속성이다.




세로도 마찬가지임.

# Stack View

## Distribution




분포에 관ㄹ한 속성이다. 제약조건을 처리하지 않고도 꽤 복잡한 분포 동작을 수행한다.

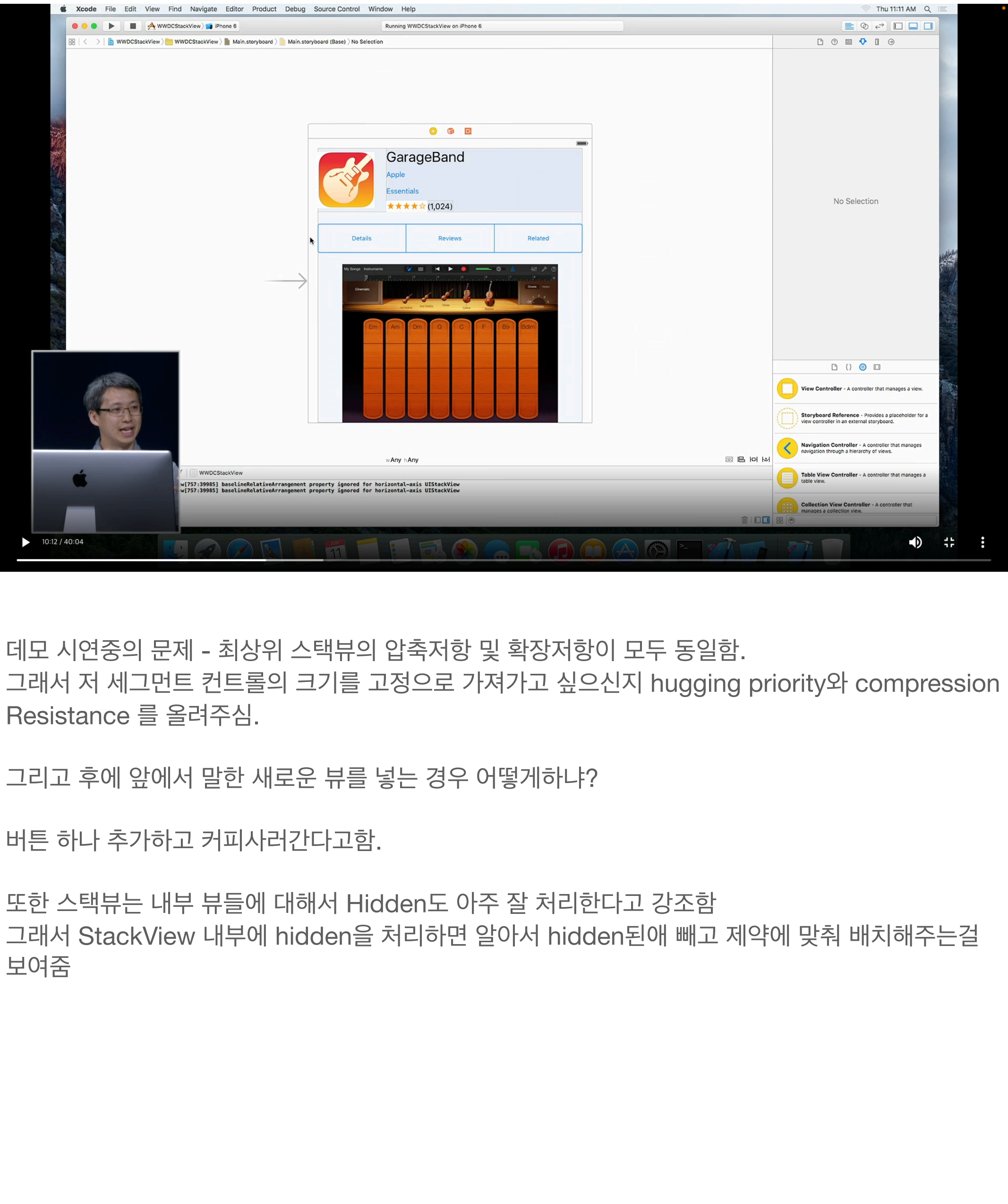
꽉채우고, 동등하게 채우고, 비율에 맞게 채우고, 같은 간격을 주는 등 우리는 스택뷰 내부의 하위뷰에 대해서 분포에 대한 제약을 줄 수 있다.

# *Demo*

## Stack View in Interface Builder

데모 내용

개러지 뷰의 모든 뷰를 최상위에 따로 놔둔 뒤 스토리보드에서 임베드 함
스택뷰로 묶고 나서 축, 정렬, 분포, 간격 과 같은 기본적인 옵션 설명
그리고 나서 전부 스택뷰로 묶음.




데모 시연중의 문제 - 최상위 스택뷰의 압축저항 및 확장저항이 모두 동일함.
그래서 저 세그먼트 컨트롤의 크기를 고정으로 가져가고 싶으신지 hugging priority와 compression Resistance 를 올려주심.

그리고 후에 앞에서 말한 새로운 뷰를 넣는 경우 어떻게하냐?

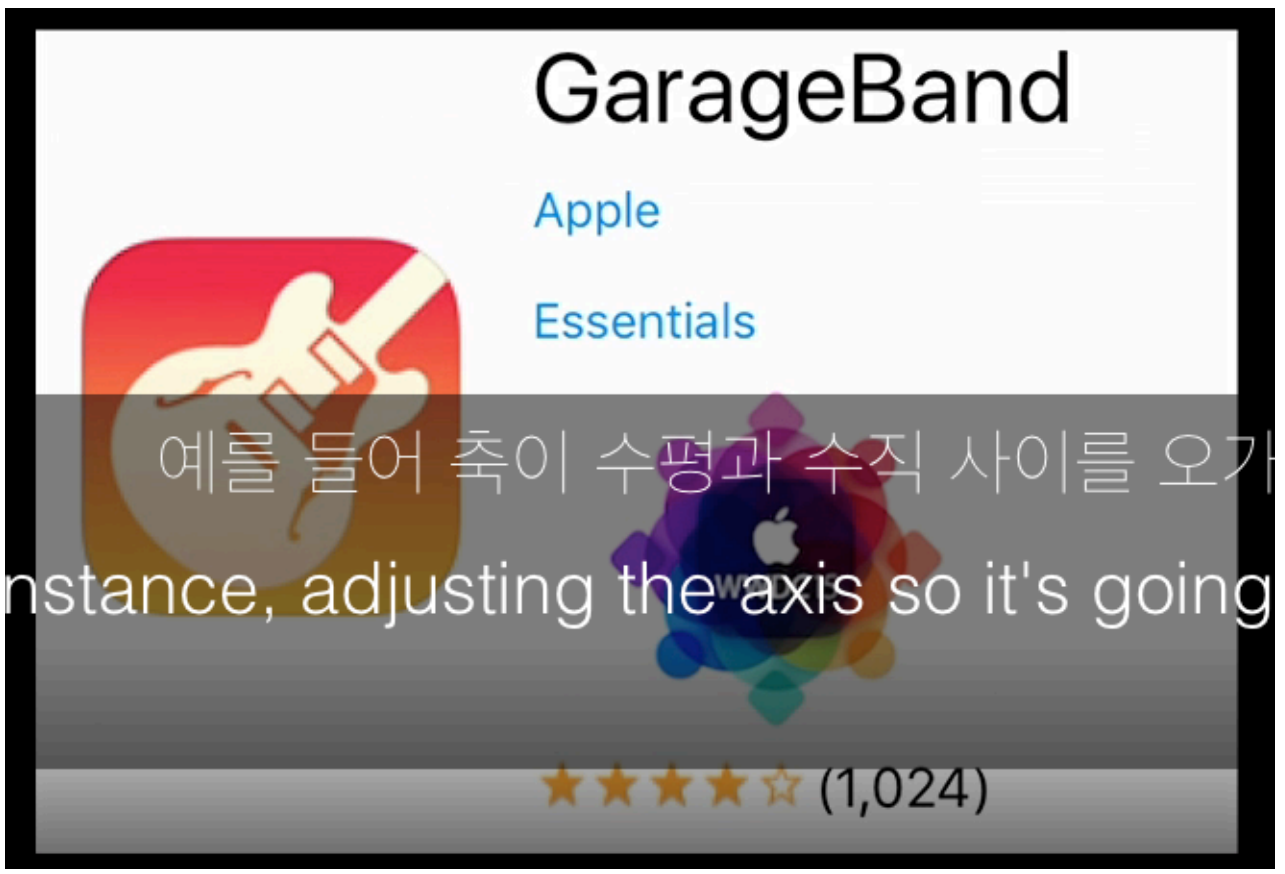
버튼 하나 추가하고 커피사러간다고함.

또한 스택뷰는 내부 뷰들에 대해서 Hidden도 아주 잘 처리한다고 강조함
그래서 StackView 내부에 hidden을 처리하면 알아서 hidden된애 빼고 제약에 맞춰 배치해주는걸 보여줌

# Animate

```
// iOS 9
UIView.animateWithDuration(1.0) { () -> Void in
     self.subviewToHide.hidden = !self.subviewToHide.hidden
}
```


스택뷰는 애니메이션도 잘 처리한다고 강조함.




이것처럼 중간에 hidden되도록 애니메이션 처리를 했는데 무너지지 않고 잘 스르륵 배치가 재 구성되는걸 볼수 있었음

# API

```
// iOS 9
class UIStackView  {
var axis:UILayoutConstraintAxis
var distribution:UIStackViewDistribution
var alignment:UIStackViewAlignment
var spacing:CGFloat
func addArrangedSubview(view: UIView)
var arrangedSubviews:[UIView]
…
}
```

```
// OS X 10.11
class NSStackView  {
var orientation:NSUserInterfaceOrientation
var distribution:NSStackViewDistribution
var alignment:NSLayoutAttribute
var spacing:CGFloat
func addArrangedSubview(view: NSView)
var arrangedSubviews:[NSView]
…
}
```

그래서 API 를 보자

축, 분배, 정렬, 간격 등의 속성이 있고
구성 요소를 추가하기위한 메서드도 있다.

또한 현재 스택뷰에 속한 뷰들도 반환할 수 있는 속성도 존재한다.

# Stack View in Interface Builder

Easy to build

Easy to maintain

Composable Stack Views

Lightweight

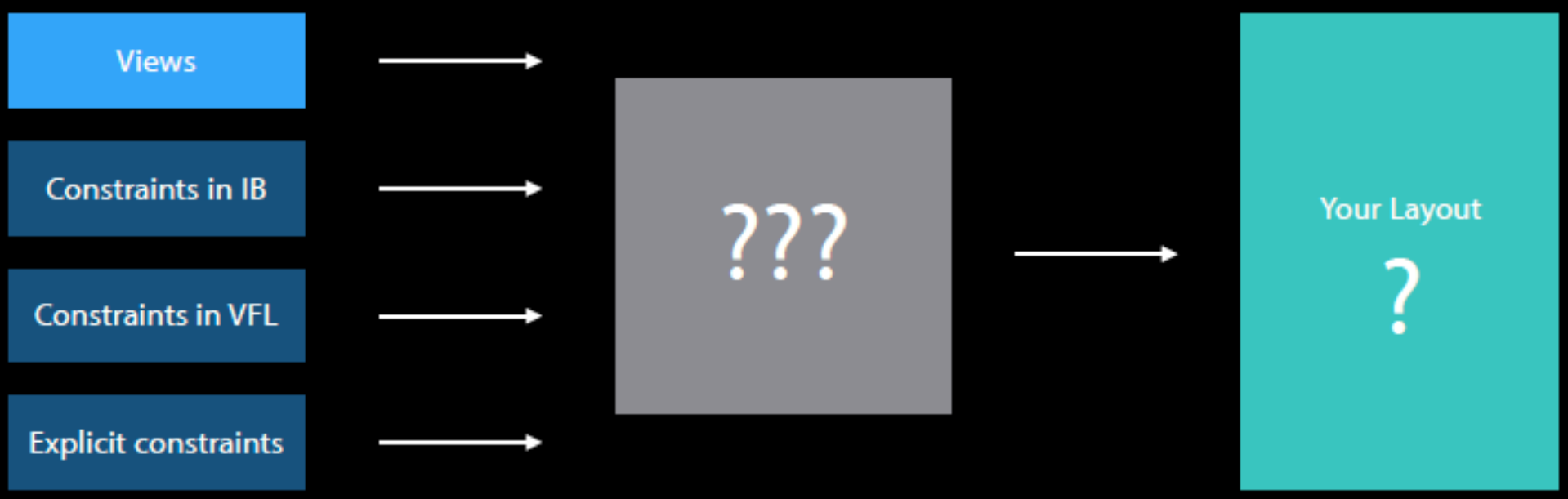
스택뷰에 대해서 정리해보자.

빌드하기 쉽고 유지하기 쉽다.

레이아웃에 관한 것이므로 자체 배경을 렌더링할 필요가 없기 때문에 우리는 약간의  최적화를 해야함.

따라서 StackView를 일반 보기보다 빠르게 실행 하고 성능을 높일 수 있는 자체적으로 렌더링 되지 않는 특별한 transformLayer클래스가 있다. > 결론은 가볍다.

# Getting from Constraints to Layout




우리는 뷰를 짜고 제약조건을 줄 때 다음과같이 한다.

뷰를 선언하고 레이아웃을 설정한다.
그리고 이걸 블랙홀에 던지면 레이아웃이 짜잔 하고 나온다.  근데 이건 내가 바라는건지 아닌지 모름.

그래서 우리는 제약조건을 변경하는것에 대해서 알아볼것임.

# Activate and Deactivate

Constraints find their own container

Adds constraints efficiently

Do not need to own all views

제약조건 변경에 대해서 얘기할 때 우리는 주로 제약 조건 활성화와 비 활성화에 대해서 이야기한다.

하지만 제약조건은 추가와 삭제보다는 > 하지 말라.
활성화와 비활성화가 더 좋다.

자체 컨ㅌ체이너를 찾아서 효율적이다.
레이아웃을 사용하기; 위해 모든 뷰를 소유할 필요가없다.

# Things to Keep in Mind

Never deactivate `self.view.constraints`

- Not all of those constraints belong to you
- Weird things will happen
- Just don't do it!

Keep references to constraints that change

---

따라서 제약 조건 변경에 대해 생각할 때 염두에 두어야 할 몇가지 사항이 있다.

1. self.view.constraints의 모든 항목을 비활성화 하지 마십시오. 누군가 이걸 시도했다는건 정말 이상한 일이 일어나는 것이다.
2. 나중에 변경해야 할 제약 조건에 대한 참조를 제약조건 배열 또는 개별 제약조건으로 유지하여 원하는 방식으로 정확하게 관리할 수 있도록 하는것이다.,

# *Demo*

## Changing constraints

데모에서 대각 정렬인 제약조건이 있고 이건 아이패드앱용임
아이폰 용 세로 정렬로 변경했다가 다시 아이패드용으로 돌렷을 때 제약조건을 "삭제" 했기 때문에  돌아오지 않음.

이를 활성화와 비활성화로 관리하니 슈퍼뷰의 크기가 변경되도 제약조건이 잘 적용됨.

또한 제약조건의 활성화 비활성화에도 애니메이션이 잘 적용되는걸 볼 수 있었음.

# Changing Constraints

Never deactivate **`self.view.constraints`**

Keep references to constraints

Animate changing constraints with view animation

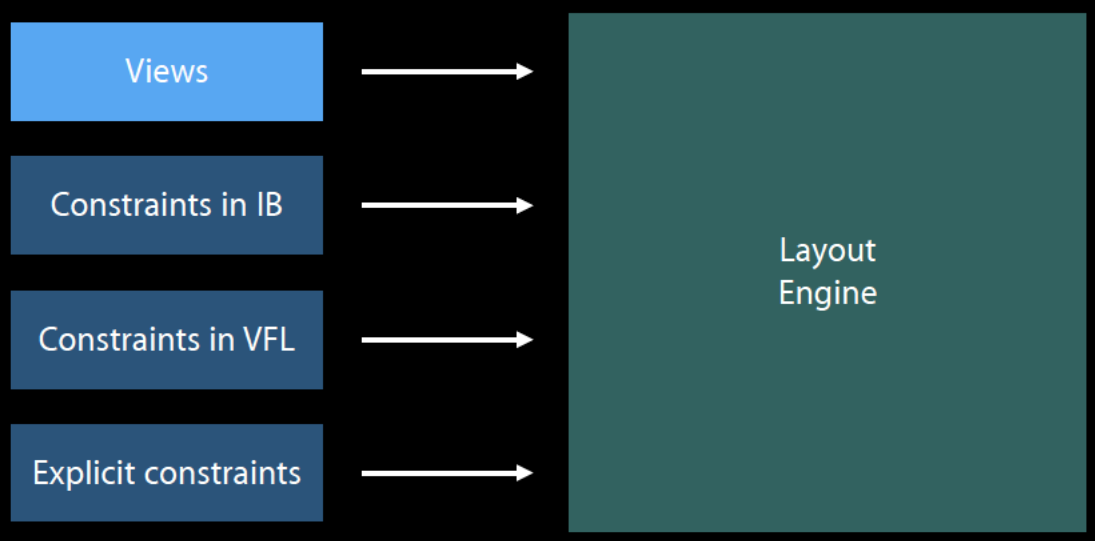
결론

self.view.constraints를 절대 비활성화 하지 ㅏ마라

제약조건 참조를 유지하라

제약조건변경 에 관한 애니메이션은 잘 동작한다.

# Building the Layout




우리는 이렇게 제약조건을 레이아웃 엔진에 배치할 수 있다.

또한 레이아웃엔진에 더 많은 정보를 제공하면 다양한환경에 적응하는 좋은 결과를 얻을 수 있다.

# View Sizing

## Mystery #3

제약 조건에 관해 이야기 해왔고 이제는 뷰의 크기에 대해서 얘기해보자.

# View Size

## Defining a particular view size

Use constraints first

Override **`intrinsicContentSize`** for specific reasons

- If size information does not come from constraints
- If view has custom drawing (sometimes)
- You will be responsible for invalidating

Size can change with size class changes

제약조건을 먼저 설정하되,
우리는 필요한 경우 컨텐츠의 고유 크기를 설정할 수 있음.

1. 제약조건에서 크기를 가져올 수 없는 경우
2. 뷰에서 사용지 자정 드로잉을 사용하는 경우

이거외에는 대부분의 경우에서는 고유 크기를 딱히 지정할 필요가 없음

필요한경우는, 콘텐츠 크기가 재정의 되는 경우(로컬라이제이션 하면서 글자 크기가 변경되는 경우)는
이런 제약조건을 무효해야할 책임이있음

데릭

# Self-Sizing Table View Cells

Mystery #4

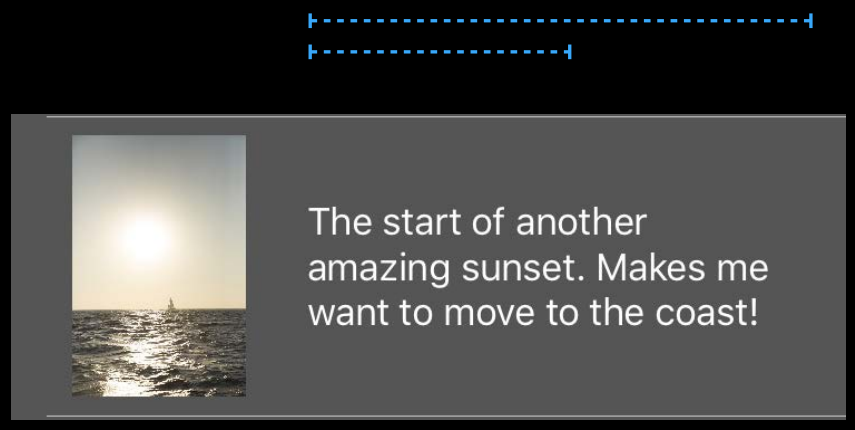
# Self-Sizing Table View Cells

Self-sizing needs size from constraints

Width is defined with table view cells

Constraints must determine height

- Take advantage of proportions




View의 높이는 Default 60이다.
테이블 뷰 셀에 레이블이 커지면 함께 커지는 것이다.

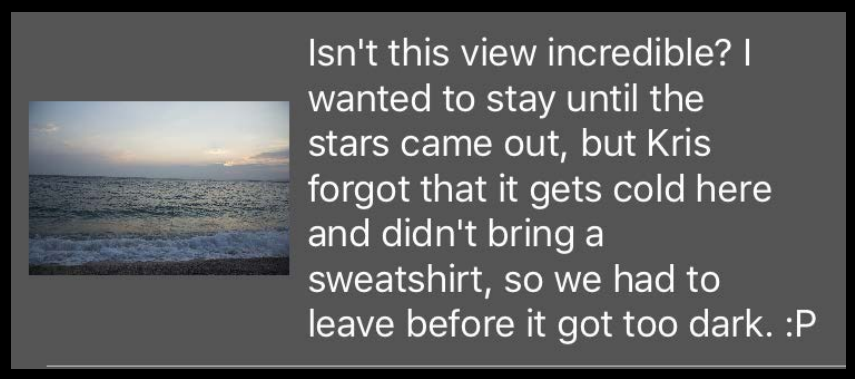
# Self-Sizing Table View Cells

Self-sizing needs size from constraints

Width is defined with table view cells

Constraints must determine height

- Take advantage of proportions




뷰가 고유한 콘텐츠 크기 이상으로 확장될 수 있다는 사실을 활용하고 레이블이 최소한 이미지만큼 커야 한다.

따라서 레이블은 이미지의 높이에서 시작하여 이미지의 높이가 충분히 높아지면 그 이상으로 늘어날 수 있도록 주변에 약간의 추가 패딩을 갖게 됩니다.

따라서 여분의 간격이 표시되지 않고 텍스트가 중앙에 잘 유지됩니다.

# View Sizing

Certain views have an `intrinsicContentSize`

Constraints should define size when possible

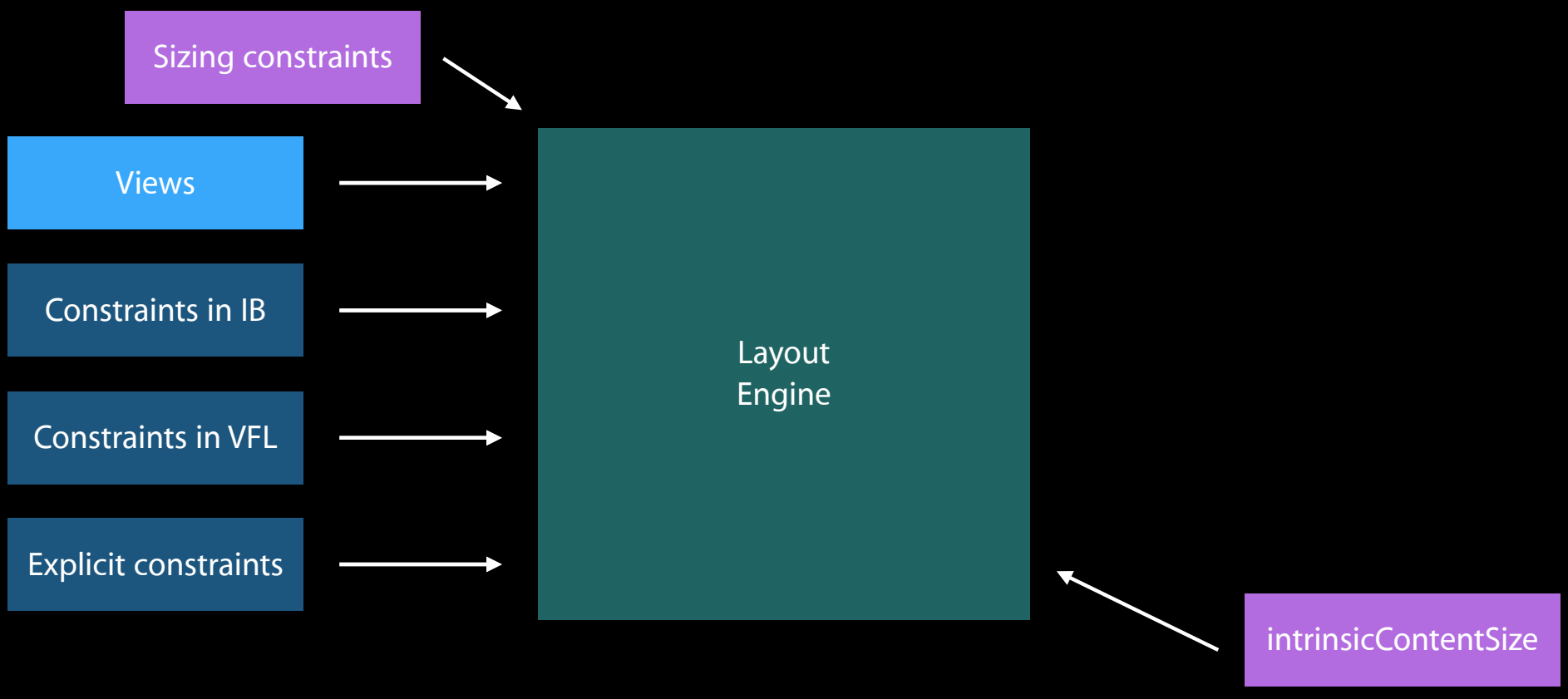
For self-sizing views, define size fully in constraints

제약 조건은 가능한 경우 크기를 정의해야 합니다.

그렇지 않은 경우 `intrinsicContentSize` 재정의할 수 있지만 무효화해야 합니다.

Self-sizing view의 경우 제약 조건에서 크기를 완전히 정의합니다.

# Building the Layout



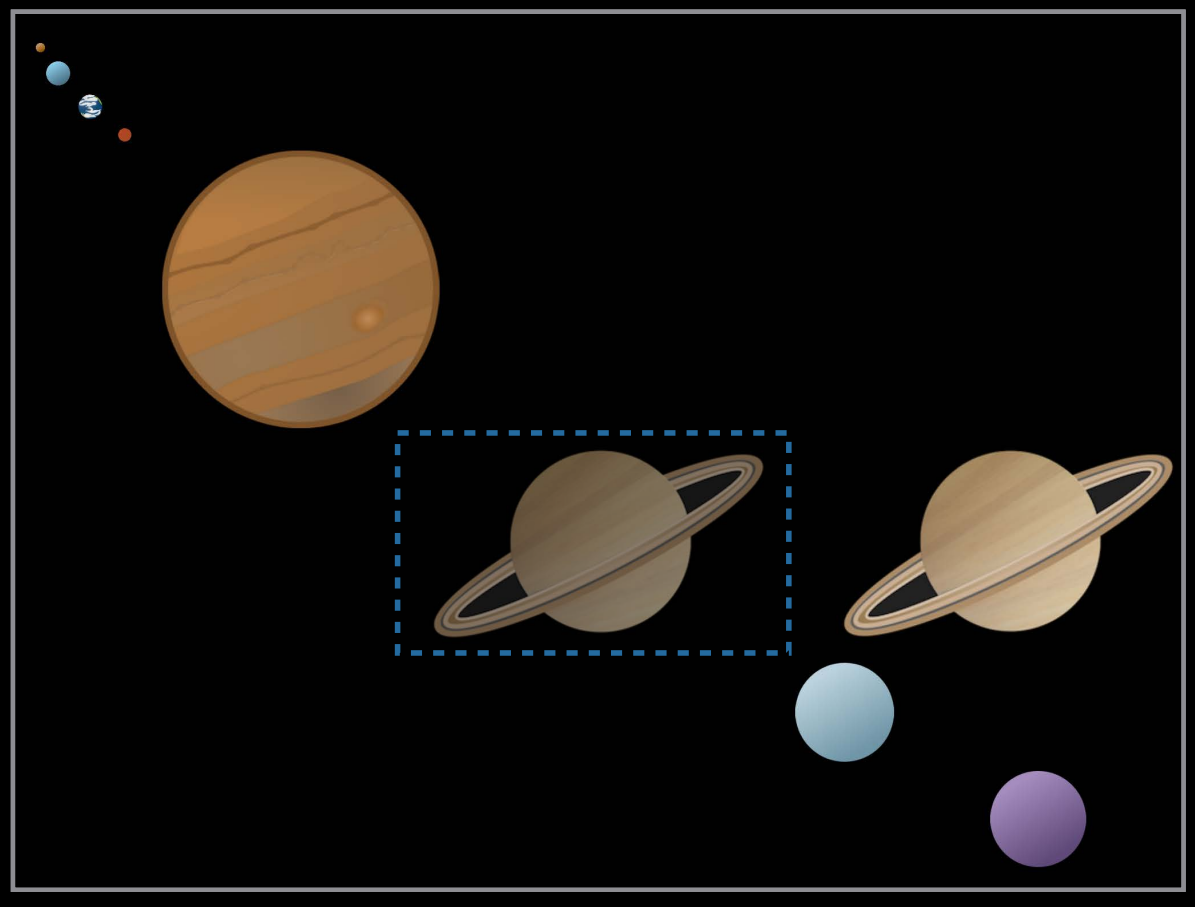
# Ambiguity

## Why does it happen?

More than one layout solution

- Not enough constraints
- Equal, non-required priorities

The engine makes a choice



한동안 Auto Layout을 사용했다면 뷰가 슈퍼뷰와 관련하여 예상한 위치에 도달하지 않는 상황에 도달했을 수 있습니다.

# Constraint Priorities

Priorities go from 1–1000

Required is 1000

DefaultHigh is 750

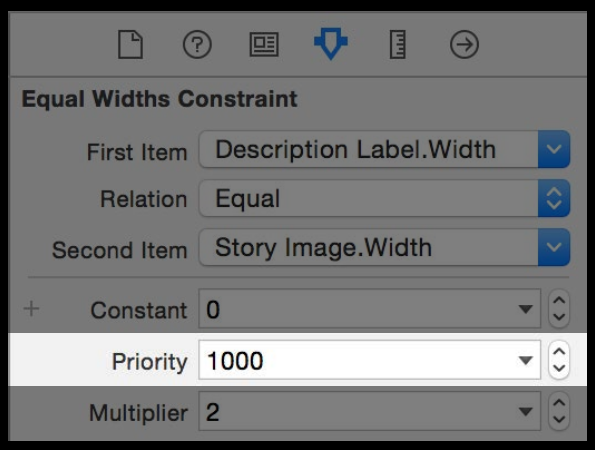
DefaultLow is 250

Highest priority wins

System uses some priorities

- Set around, not equal to

```
@"H:|-[image]-|"
@"V:|-[image]-[caption(==image@751)]-|"
```




```
widthConstraint.priority =
        UILayoutPriorityDefaultHigh + 10;
```

그렇다면 우선 순위는 어떻게 작동합니까?

제약 우선순위
- 뷰가 콘텐츠를 처리하는 방법
- 기본적으로 필요에 따라 설정되지 않습니다.
- 필요에 따라 설정하지 않음
- 만족할 수 없는 제약을 야기할 수 있음
- 동일한 우선 순위는 모호성을 유발할 수 있습니다.
- Type
- Content Hugging
- Compression resistance

# Content Priorities

How a view handles its content

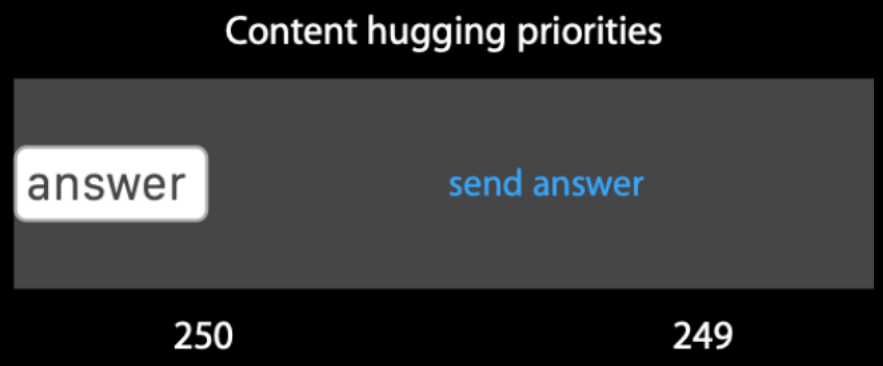
By default, these are not set as required

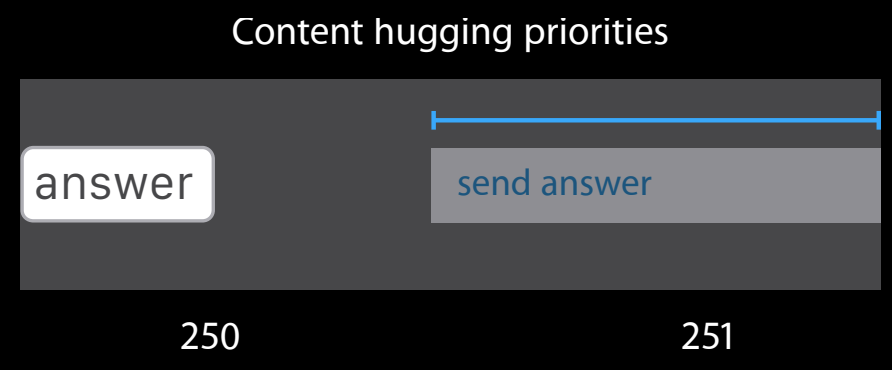
- Do not set as required
- Can cause unsatisfiable constraints

Equal priorities can cause ambiguity

Types

- Content hugging




텍스트뷰 249
→ 엔진은 제약 조건과 우선 순위를 보고 이렇게 말합니다. 텍스트 보기  Content Hugging는 상당히 중요하지만 버튼  Content Hugging 순위는 전혀 중요하지 않으므로 콘텐츠에서 멀리 확장하여 이 내용을 채울 수 있습니다. 여기 보기의 수평 부분입니다.

텍스트뷰 251
→ 엔진에서 "이제 이 버튼 텍스트를 정말 가까이 껴안고 싶습니다. 즉, 레이아웃을 해결하기 위해 텍스트 필드를 늘릴 수 있습니다."라고 말합니다.

# Content Priorities

How a view handles its content

By default, these are not set as required

- Do not set as required
- Can cause unsatisfiable constraints

Equal priorities can cause ambiguity

Types

- Content hugging
- Compression resistance

Compression resistance : 콘텐츠가 찌그러지는 것을 얼마나 저항하는지입니다.

# Content Priorities

How a view handles its content

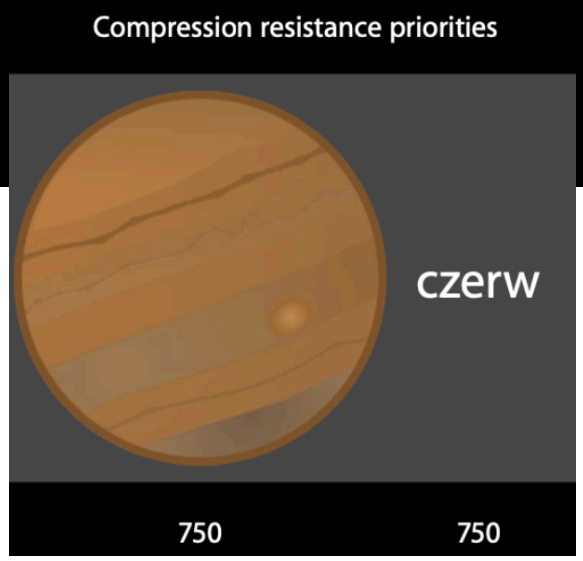
By default, these are not set as required

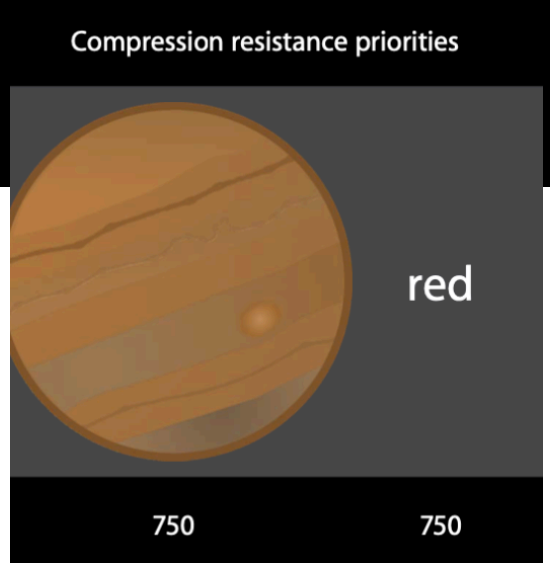
- Do not set as required
- Can cause unsatisfiable constraints

Equal priorities can cause ambiguity

Types

- Content hugging
- Compression resistance


이미지 뷰를 축소하거나 레이블을 자를 수 있습니다.

# Priorities

Can help keep constraints from unsatisfiability

- But look out for competing priorities!

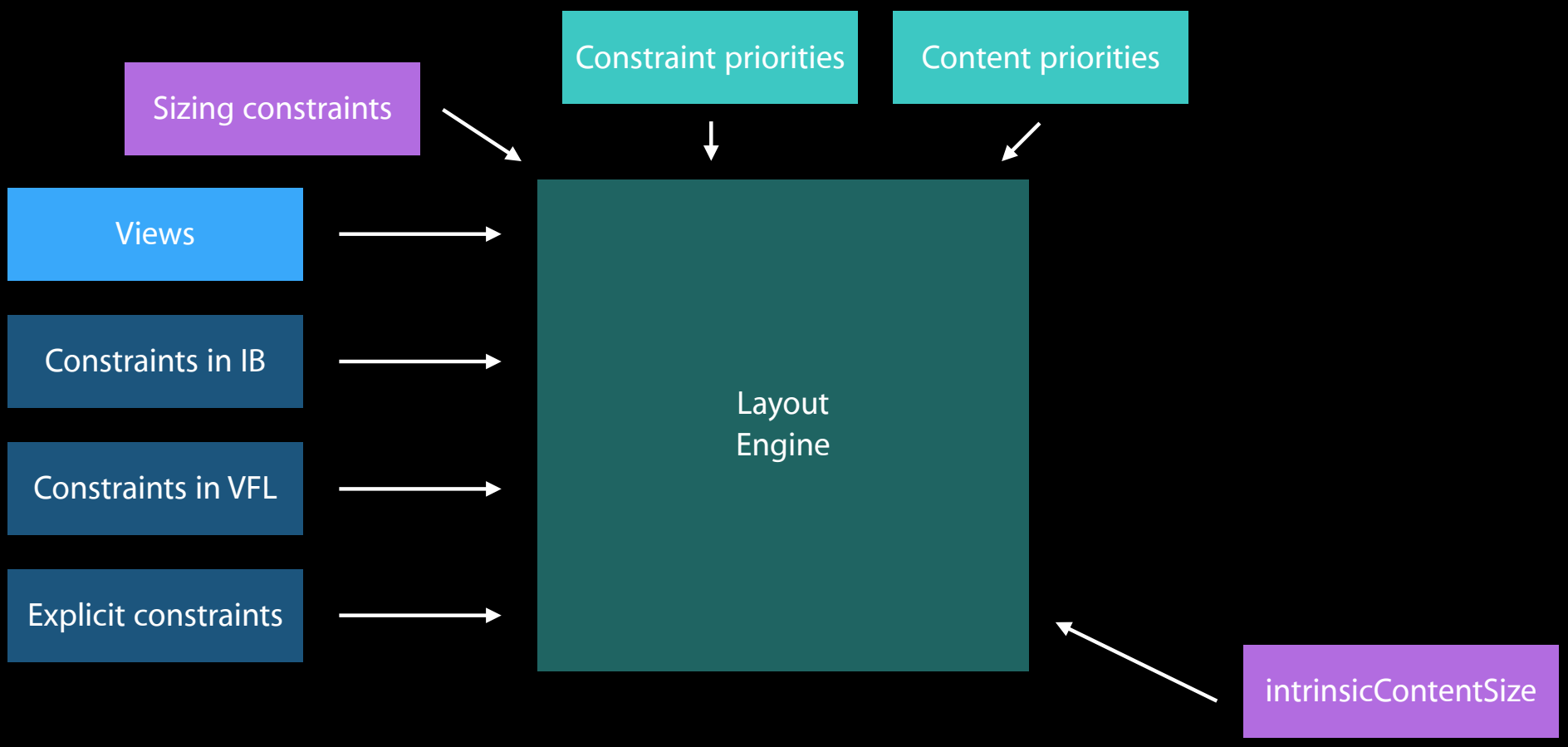
Results are more consistent

Use content priorities to get to the right layout

- Hugging priorities hug content
- Compression resistance resists squishing

- 만족하지 못하는 제약 조건을 유지하는 데 도움이 될 수 있습니다.
  그러나 경쟁 우선 순위를 찾으십시오!
  결과가 더 일관됨
  콘텐츠 우선 순위를 사용하여 올바른 레이아웃을 얻습니다.
  Hugging 우선 순위 Hugging Content
  Compression resistance는 찌그러짐에 저항

# Building the Layout



바드

# Alignment

## Mystery #6

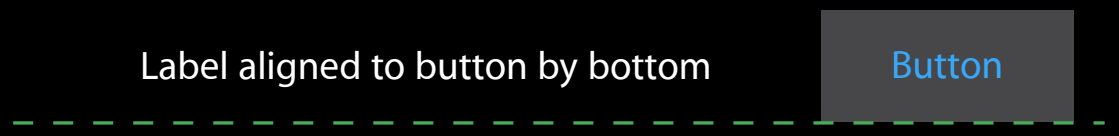
오토레이아웃을 사용한 경우 뷰를 정렬하는데 익숙할 것임
뷰들은 어떤 종류의 수평적인 정렬, 또는 수직적인 정렬이 필요함
그래서 뷰들은 서로 어떠한 관계를 가지고 있는지 알고 있음
이번에는 텍스트 정렬에 대해 구체적으로 애기해볼 것

# Aligning Baselines

Use **`firstBaseline`** and **`lastBaseline`**

Aligns text better than top or bottom

Better control over changing views




먼저 텍스트 뷰들은 기준선들을 가지고 있고
이 기준선은 텍스트의 바로 아래에 있는 선임
텍스트 뷰에는 first baseLine과 last baseLine이 있음
first baseLine은 텍스트의 첫번 째 줄 바로 아래에서 실행되고
last baseLine은 텍스트의 마지막 줄 바로 아래에서 실행이 됨
한 줄짜리 텍스트 뷰는 이 두가지가 서로 같음

많은 상황에서 텍스트를 top이나 bottom보다 더 잘 정렬하고
동적 텍스트 크기 조정과 같은 작업에 도움이 됨

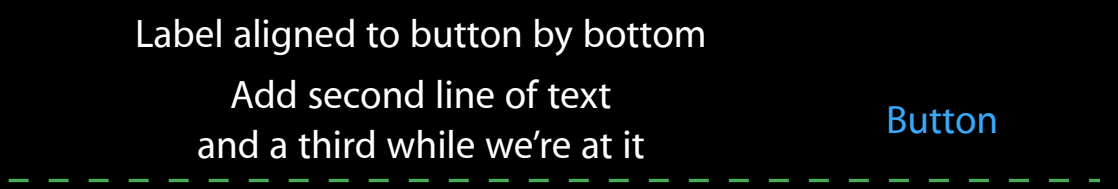
또한 뷰 변경을 통해 커질 때 더 제어를 잘할 수 있음
예를 들어 버튼 옆에 레이블이 있고 버튼이 아래쪽으로 정렬되어 있으면
기존의 공간을 가지고 있기 때문에  버튼의 프레임이 더 큼

# Aligning Baselines

Use `firstBaseline` and `lastBaseline`

Aligns text better than top or bottom

Better control over changing views



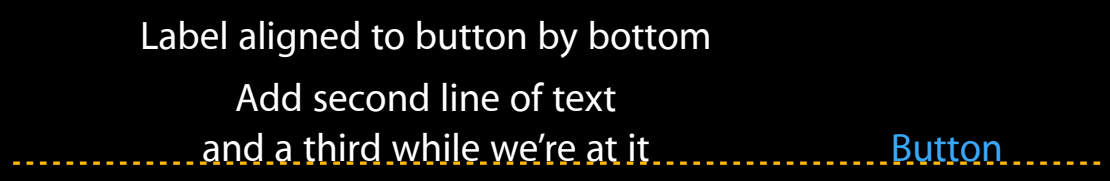

그래서 텍스트 뷰에 몇 라인들을 더 추가하면
버튼은 애매한 영역을 가지게 됨

# Aligning Baselines

Use `firstBaseline` and `lastBaseline`

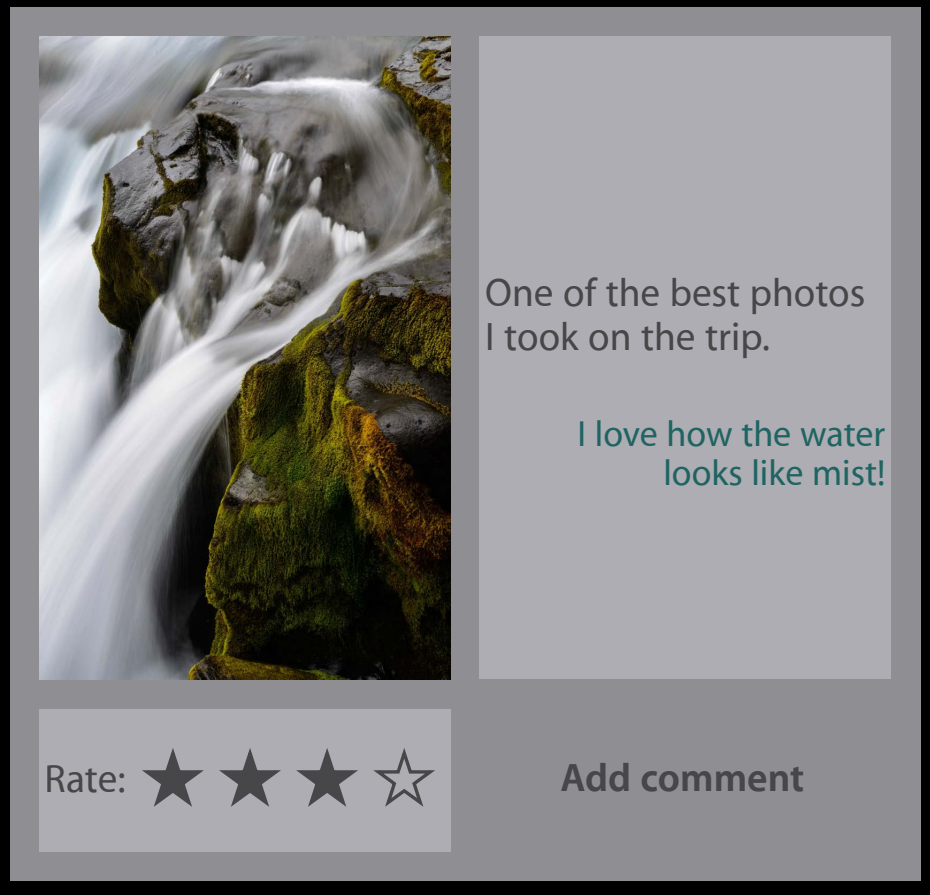
Aligns text better than top or bottom

Better control over changing views

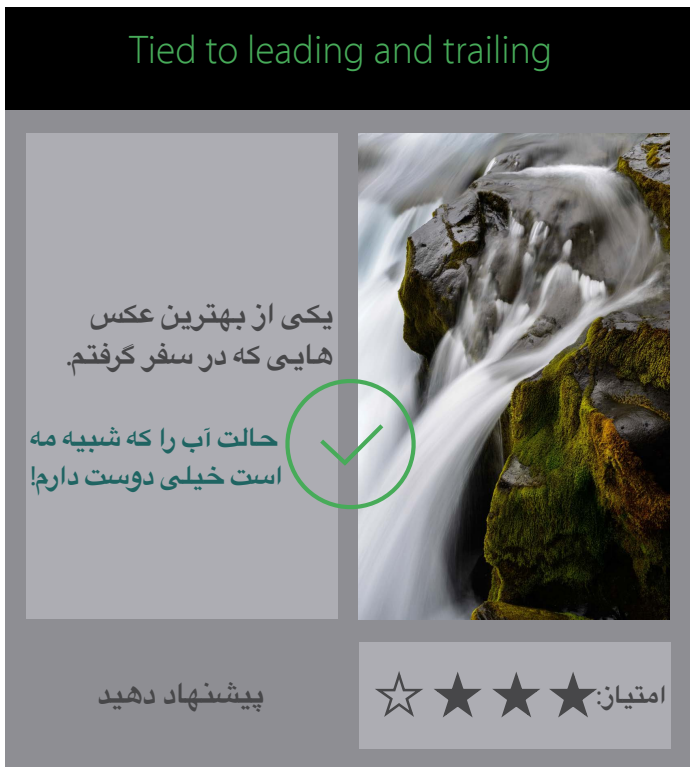


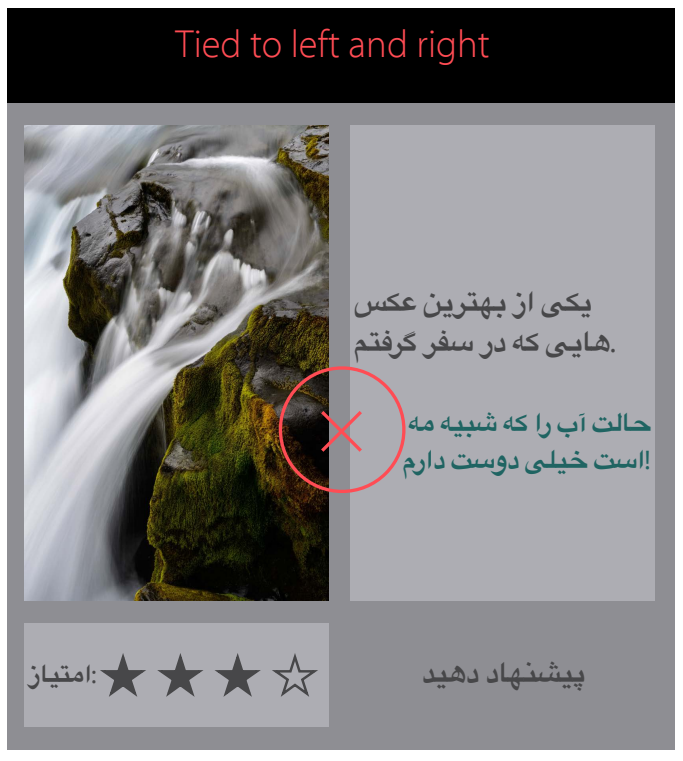

Last baseLine에 맞춰서 정렬하면 텍스트 뷰가 어떻게 되든 상관없이
버튼이 last baseLine과 정렬된 상태를 유지함
물론 first baseLine에 맞출 수도 있고 중앙에 맞출 수도 있음

# Leading and Trailing

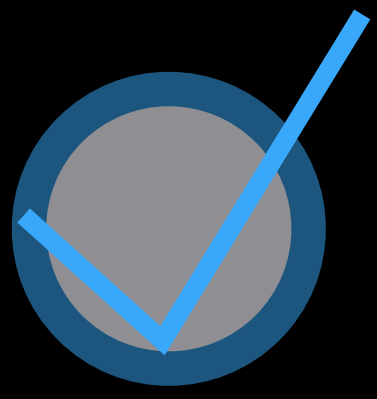



텍스트 정렬의 다른 중요한 부분은 leading과 trailing임
이 둘은 실제로 모든 타입의 뷰에 중요하지만, 큰 용도는 localizing임
앱을 작성할 때 왼쪽에서 오른쪽으로 언어를 사용하고 레이아웃에
leftAnchor와 rightAnchor에 제약조건을 사용하면
오른쪽에서 왼쪽을 사용하는 언어로 현지화되면 이상한 결과를 얻을 수 있음



# Alignment Rects




정렬의 마지막 조각은 alignment Rects임

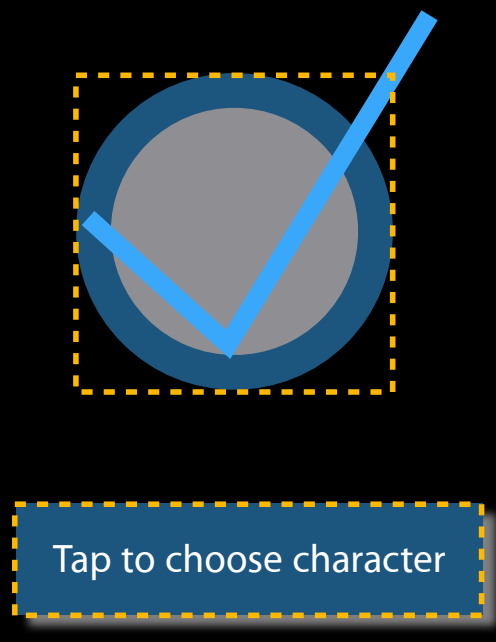
alignment Rects는 엔진이 실제로 계산하기에 중요함
엔진은 사용자의 모든 제한을 계산하여
뷰를 실제로 배치하는데 사용함

# Alignment Rects

Usually (not always) same as frame

Includes the critical content only

Does not change when view is transformed




그렇다면 alignment Rects는 무엇임?
alignment Rects는 보통 뷰의 프레임이지만, 항상 그렇지는 않음

뷰 주위의 프레임이 아니라
뷰가 실질적으로 정렬되기 원하는 주요한 내용이 포함되어 있을 수도 있음
예를 들어 여기 체크박스가 있음
다른 뷰와 centerXAlignment 정렬을 하는 경우 체크박스의 중심이 아닌
원의 중심으로 정렬하고 싶을 수도 있음
그림자가 있는 버튼도 마찬가지임
그림자를 제외하고 내용의 layout에 중심을 맞춤
뷰가 변환될 때도 변하지 않음
그래서 너의 뷰를 배치하고, 어떤것을 강조하거나 강조하고 싶지않을 때
너의 layout이 나머지 부분을 망치지 않을 것

# Alignment Rects

Usually (not always) same as frame

Includes the critical content only

Does not change when view is transformed

Override `alignmentRectInsets` if needed

Find out the calculated rects


만약 이것들을 변경해야 할 필요가 있다면, 거의 할 일은 없겠지만,
alignmentRectInsets를 재정의해서 사용을 해야 함
그리고 이것은 기본적으로 엔진에 손을 대는 것을 의미함
예를 들어 어떤 alignment를 계산하든, 기존 삽입대신 자신이 계산한
alignment를 사용할 수 있음

# Alignment Rects

Usually (not always) same as frame

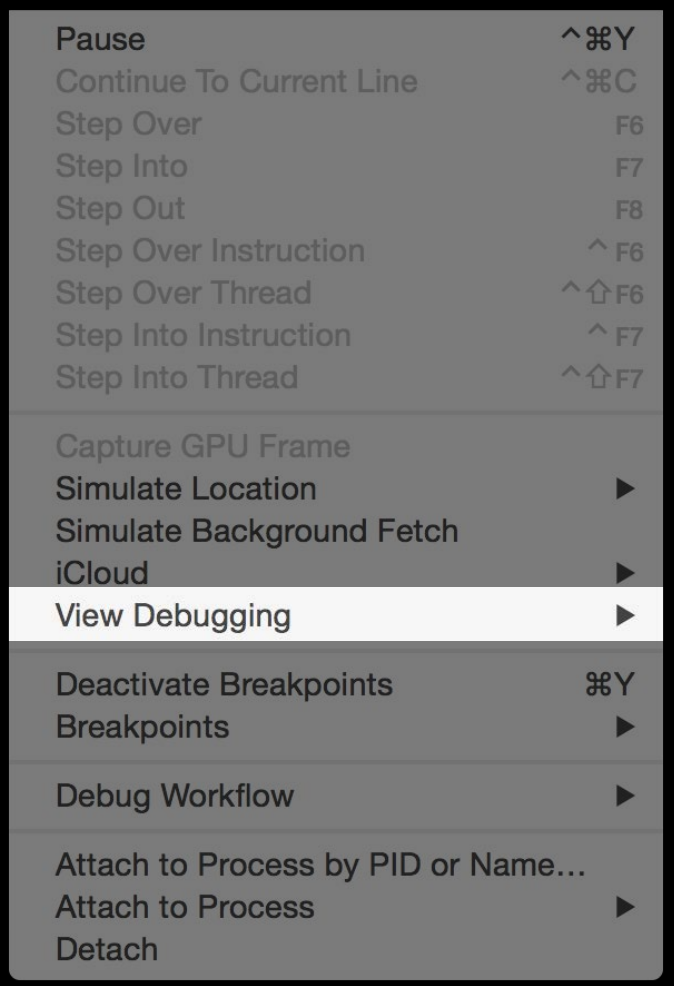
Includes the critical content only

Does not change when view is transformed

Override `alignmentRectInsets` if needed

Find out the calculated rects

- Use Show Alignment Rectangles in Debug menu




엔진이 실제로 계산한 내용을 확인하려면  Xcode의 디버그 메뉴에서
View Debugging 옵션을 누르고 Show Alignment Rectangles를 누르면 됨
그러면 뷰에 노란색 정렬 직사각형이 그려짐
또한 특정 뷰의 프레임에 대해 alignment Rect를 사용하여
디버거에서 가져올 수 있음

# Alignment Rects

Usually (not always) same as frame

Includes the critical content only

Does not change when view is transformed

Override `alignmentRectInsets` if needed

Find out the calculated rects

- Use Show Alignment Rectangles in Debug menu
- Get using `alignmentRectForFrame:`

More in Part 2

Part 2에서 더 얘기해볼 것임

# Alignment

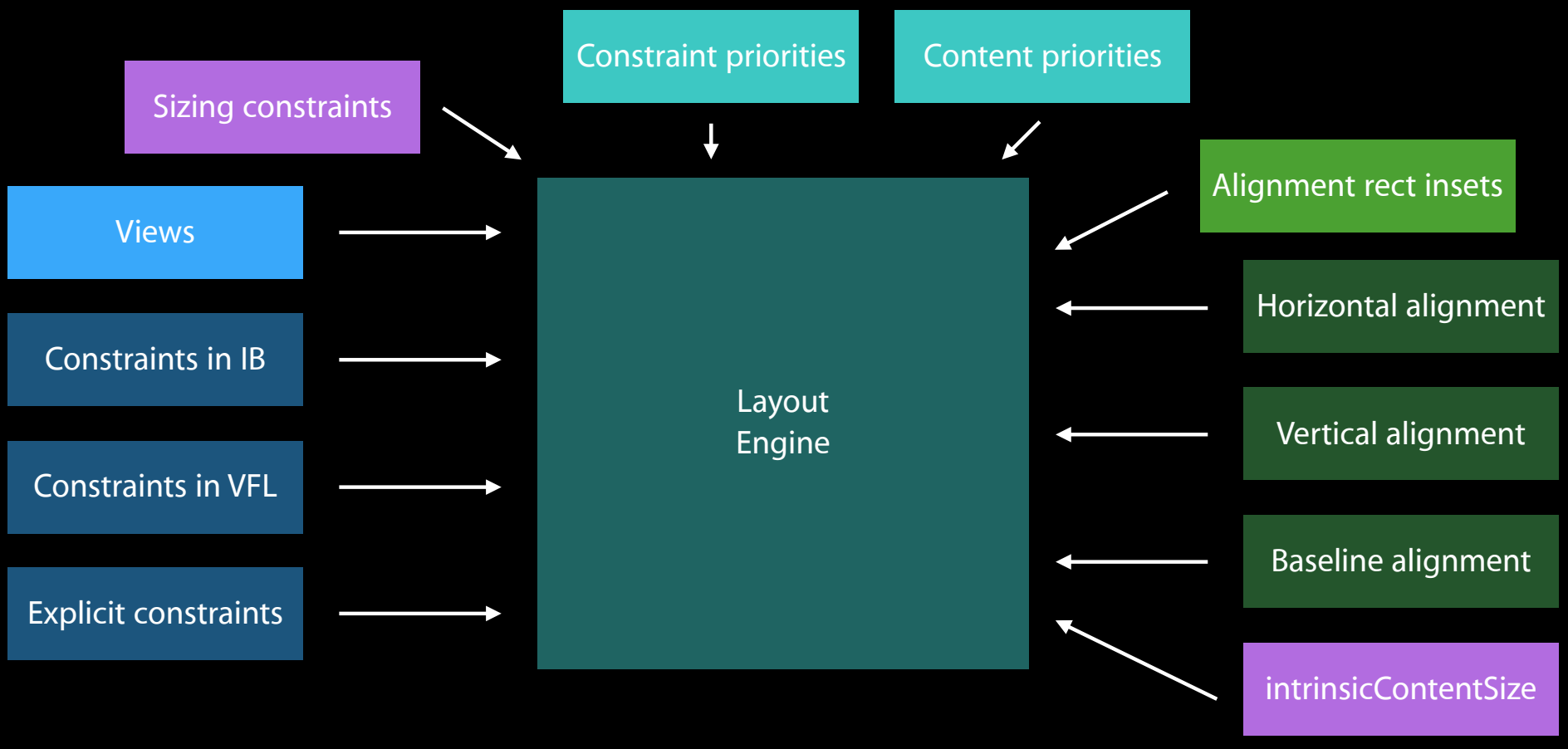
First and last baseline for better aligned text

Leading and trailing instead of left and right

Override `alignmentRectInsets` to adjust alignment rects

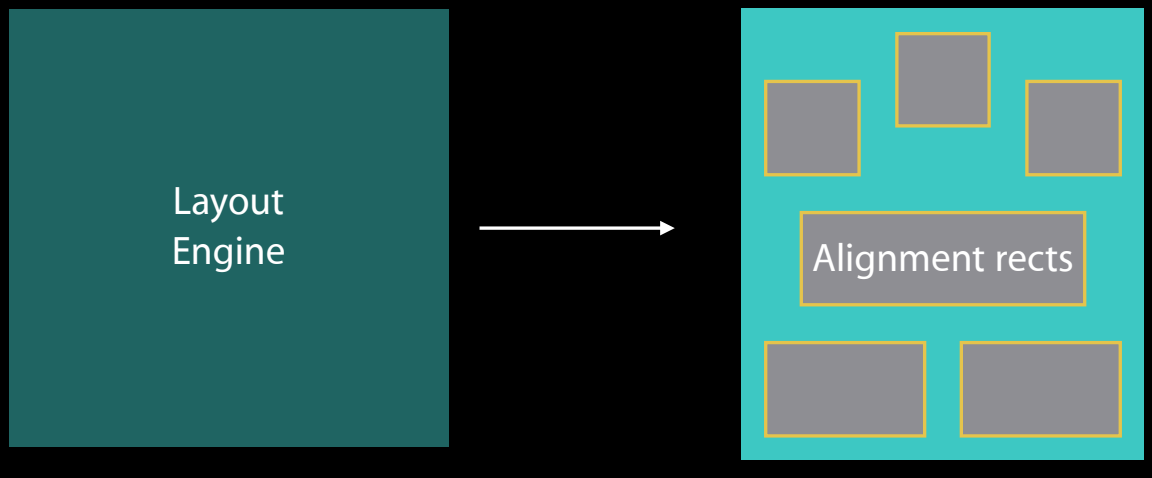
그래서 Alignment에서
우리는 텍스트가 다른 뷰들과는 조금 다르다는 것을 배움
앱을 준비할 때 동적 텍스트와
localizing이 모두 준비되었는지 확인해야 함
오토 레이아웃을 사용하면 이러한 작업을 수행하는 데 큰 도움이 됨
또한 특정한 뷰가 특정 프레임이 있어야 하는 경우
alignmentRectInsets를 재정의해야 함

# Building the Layout



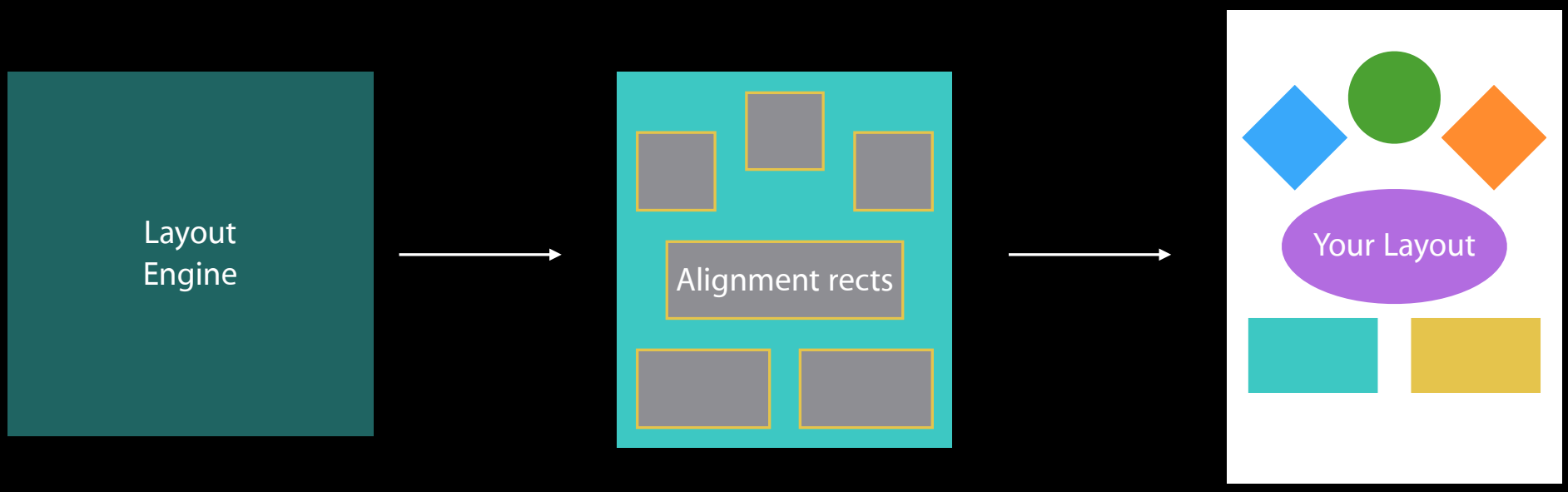

이제 Layout Engine이 자기 역할을 하도록 두어보자

# Building the Layout




Layout Engine은 alignment Rects를 계산한 다음
실제로 layout을 그리는데 사용됨

# Building the Layout




이제 모든 screen 사이즈에 원하는데로 레이아웃이 그려짐

# Summary

Stack Views help build easily maintainable layouts

Use activate and deactivate for constraints

Determine size through constraints

- Override intrinsicContentSize judiciously

Use priorities to properly solve your layout

Alignment goes beyond top, bottom, and center

- Keep localization in mind

스택 뷰는 모든 장치와 크기에서 잘 작동하는 유지관리 가능한
레이아웃을 만드는 좋은 방법
제약 조건에 맞게 활성화 및 비활성화를 적절히 사용하면 모든 화면에 대해
설계하는 것보다 작업이 덜 필요한 매우 역동적인 레이아웃을 얻을 수 있음

특정 크기를 결정할 때는 제약조건을 사용해라
꼭 필요한 경우에만 고유 컨텐츠 크기를 재정의해라
우선순위를 사용하여 마지막 작은 조정들을 수행해라
정렬은 위, 아래 및 가운데 말고도 특히 텍스트를 볼 때
localizing 및 동적 텍스트를 항상 조심해라

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 |
|---|---|---|
| 데릭 | | |
| | | |
| | | |
| | | |

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

| | 궁금한 점 | 이해 안된 점 |
|---|---|---|
| 바드 | | |
| | | |
| | | |
| | | |