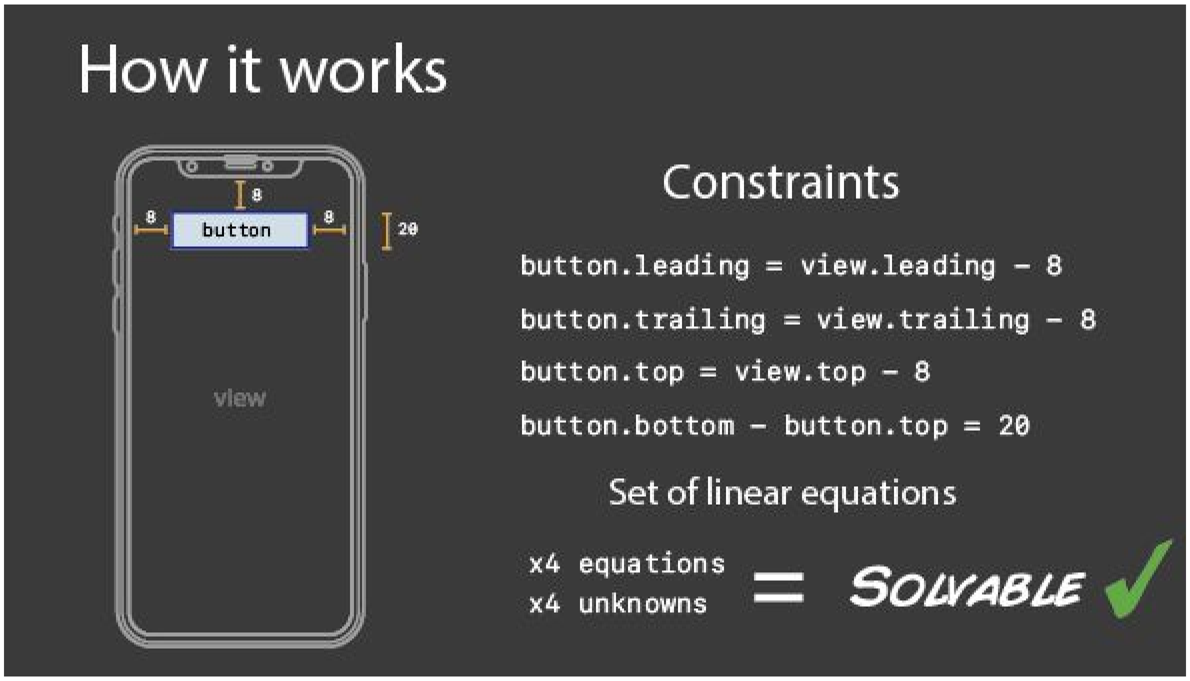
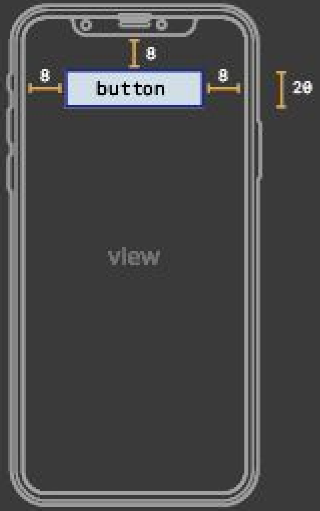
Anchors

Auto Layout



How it works



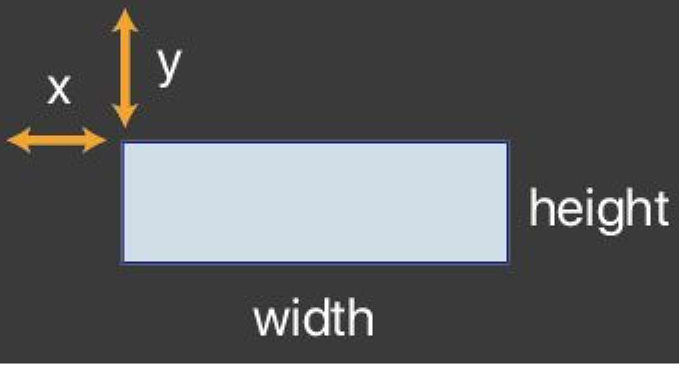
Constraints

```
button.leading = view.leading - 8
button.trailing = view.trailing - 8
button.top = view.top - 8
button.bottom - button.top = 20
```

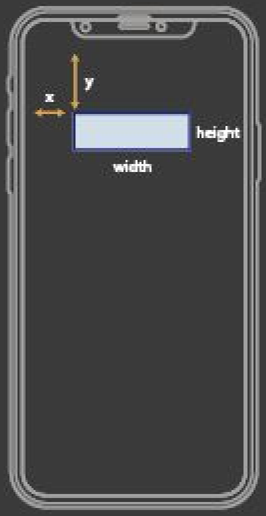
Set of linear equations

```
x4 equations
x4 unknowns
```
= SOLVABLE ✓

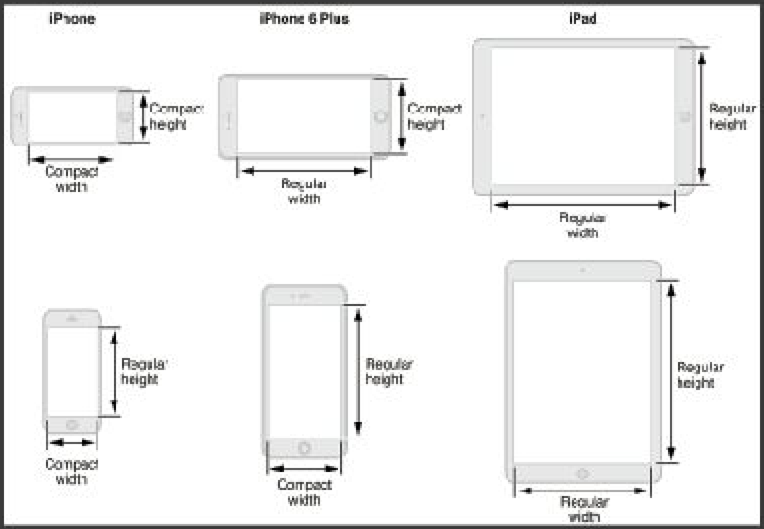
Every control needs



Framebased Layout

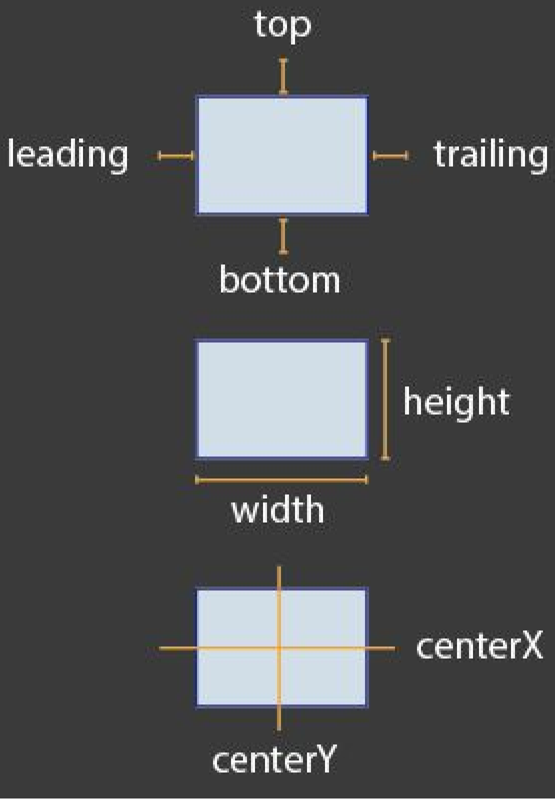


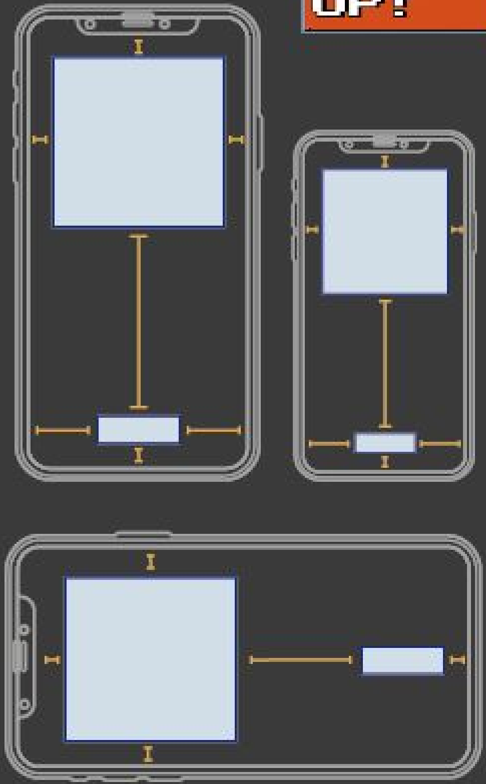
Size Classes

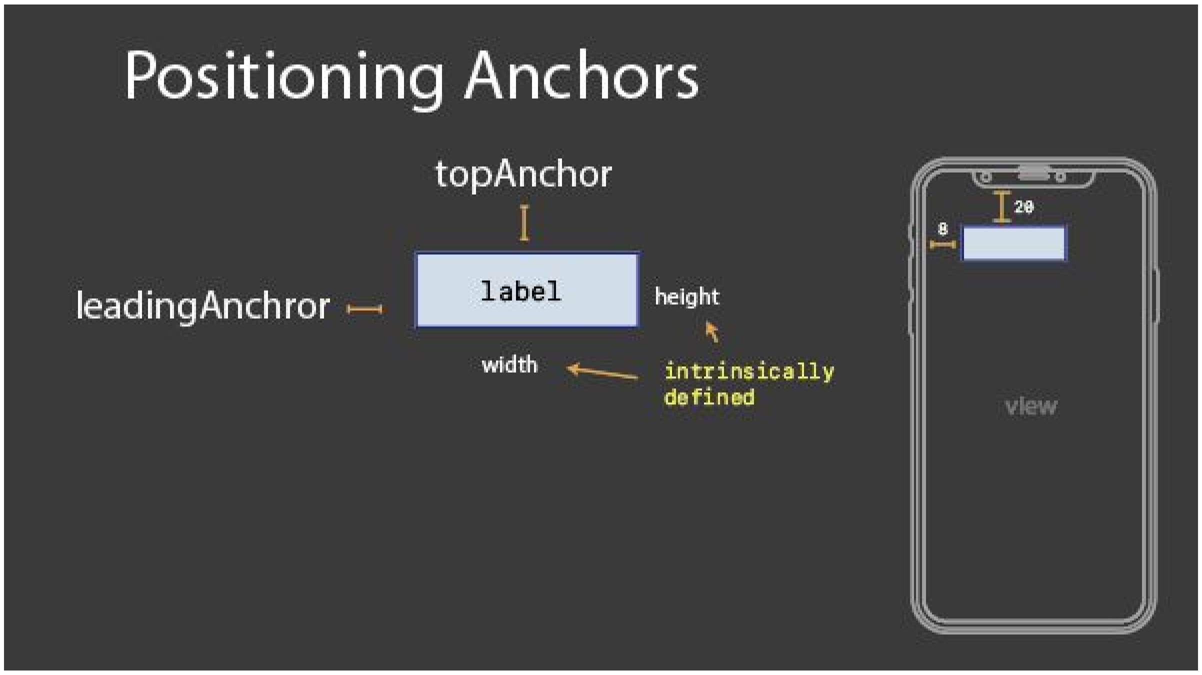


Auto Layout

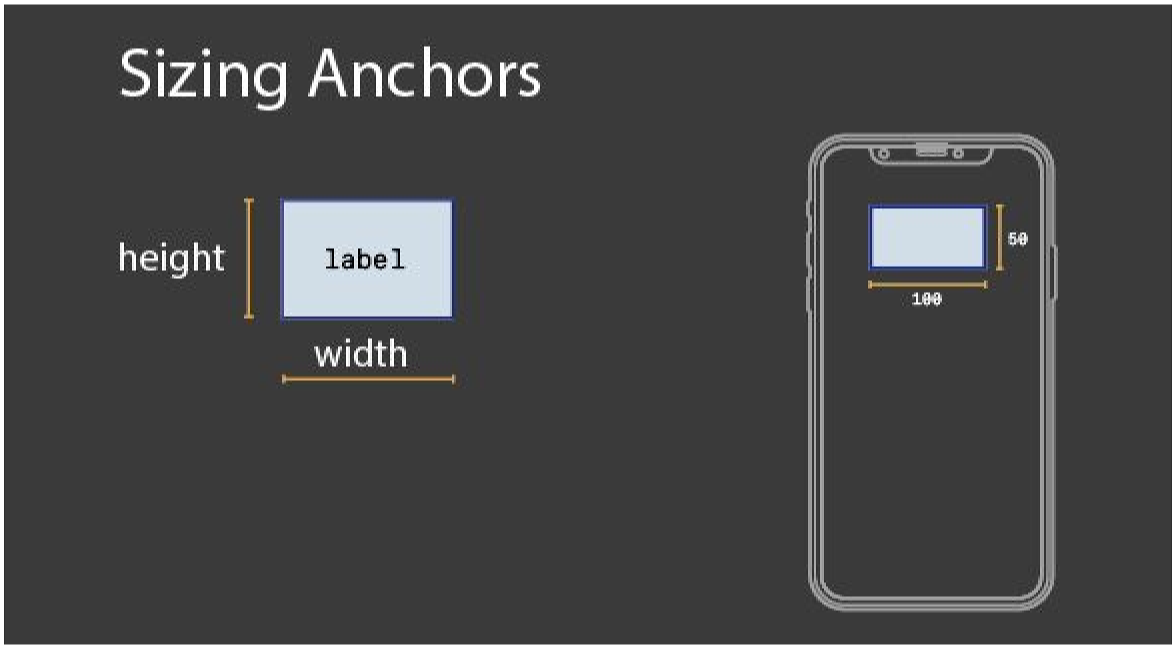
Constraint / Anchor Attributes



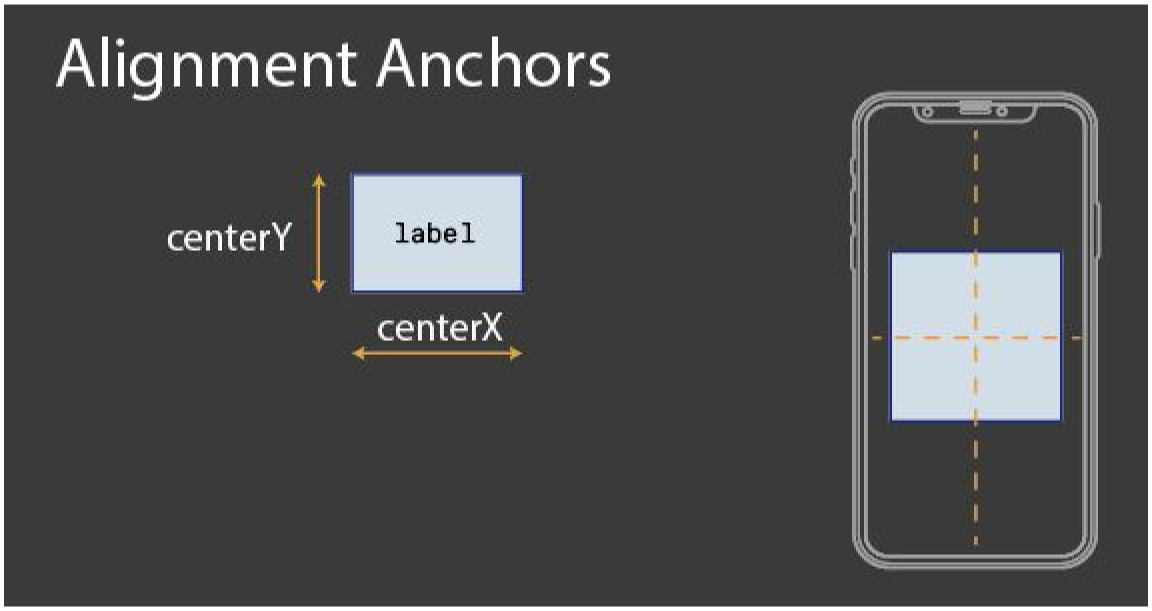




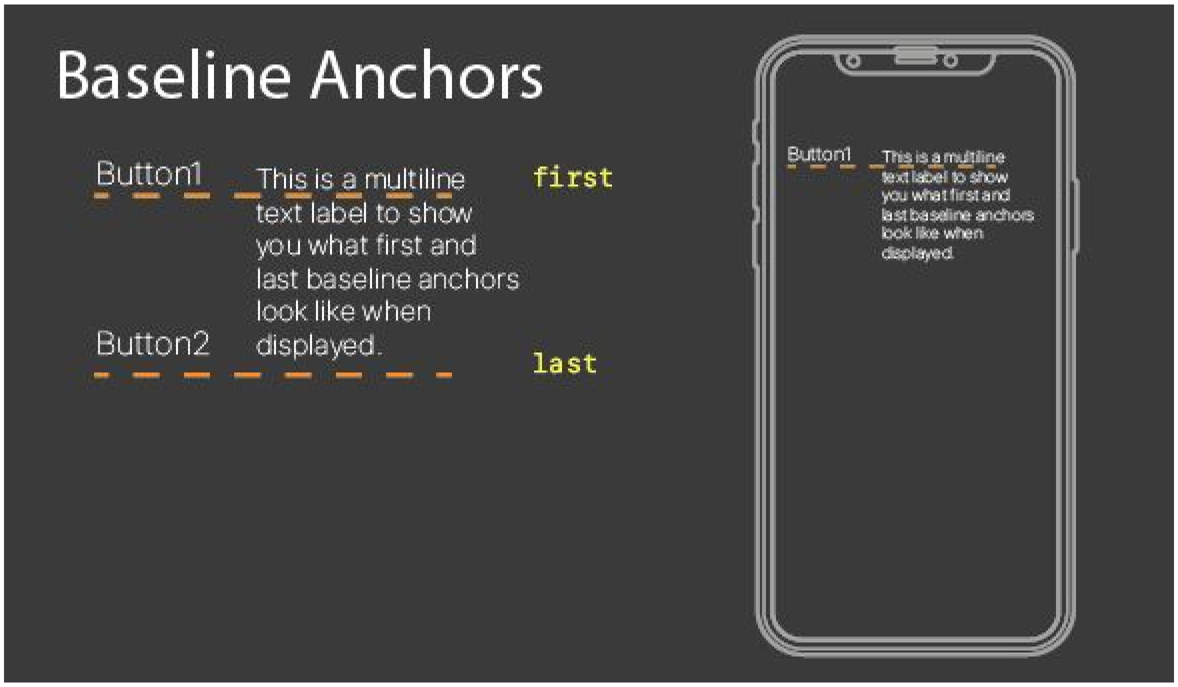
Positioning Anchors



Sizing Anchors

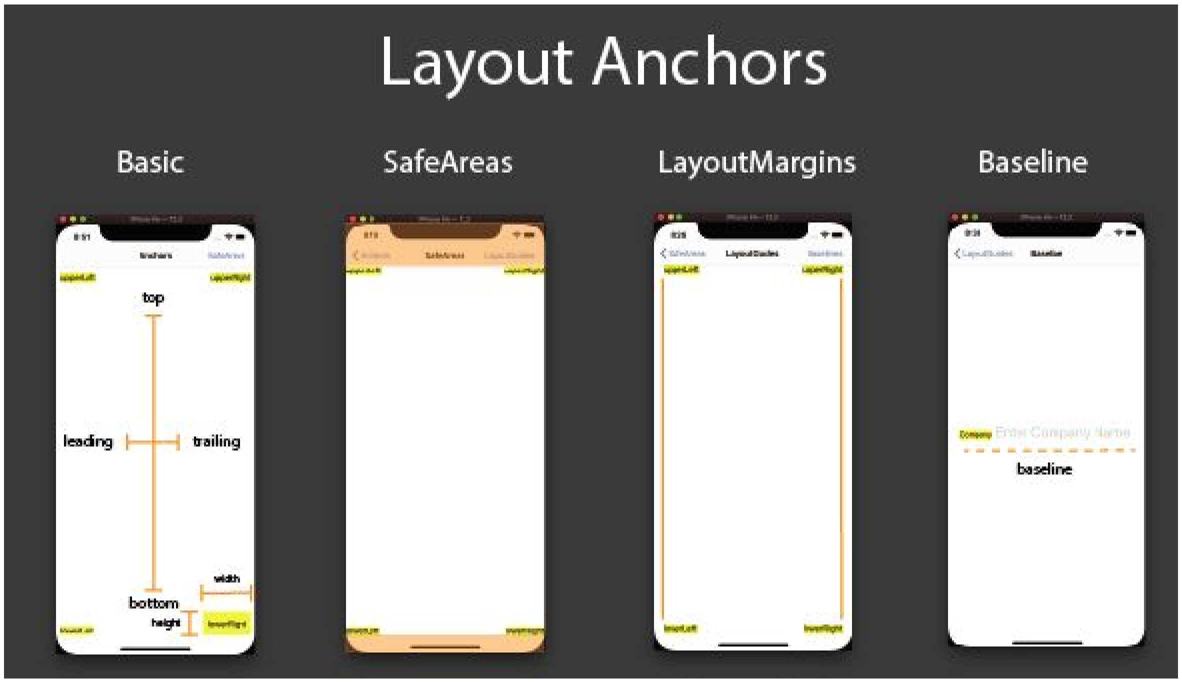


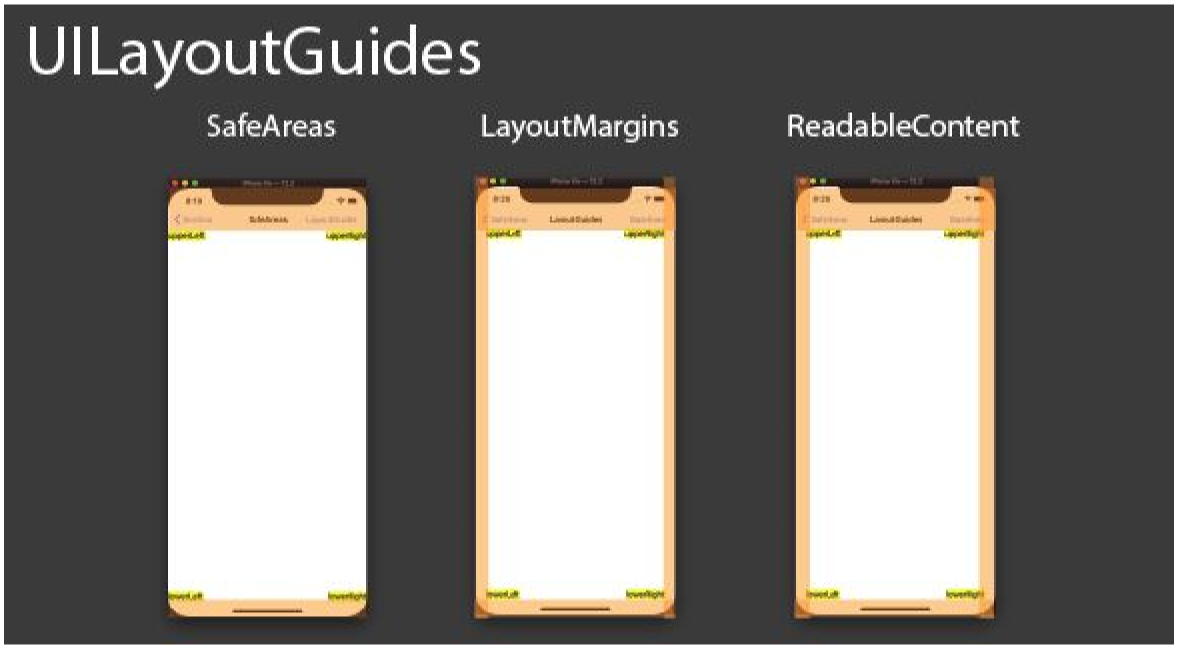
Alignment Anchors



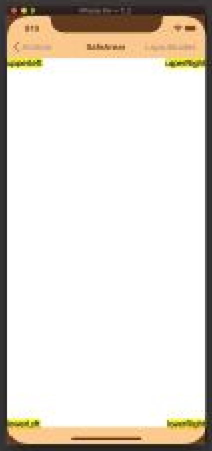
Baseline Anchors

This is a multiline text label to show you what first and last baseline anchors look like when displayed.


Layout Anchors
Basic
SafeAreas
LayoutMargins
Baseline
Anchors
SafeAreas
upperLeft
upperRight
top
leading
trailing
width
bottom
height
lowerRight
LayoutGuides
Baseline
lowerLeft
Company
Enter Company Name
baseline

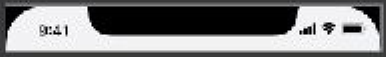

UILayoutGuides
SafeAreas
LayoutMargins
ReadableContent
upperLeft
upperRight
lowerLeft
lowerRight

SafeAreaLayoutGuide

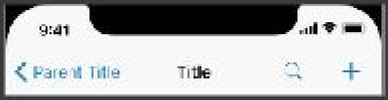


Areas where controls won't be blocked or hidden from

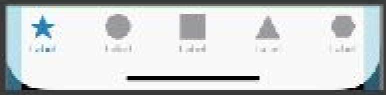
Status bars



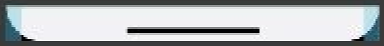
Navigation bars



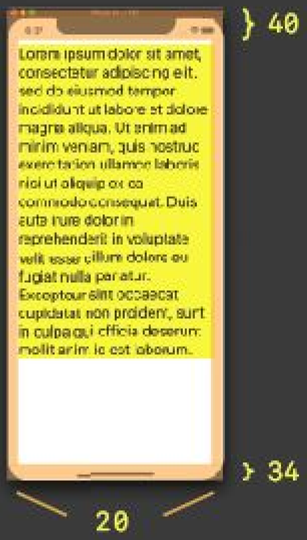
Tab bars



Tools bars



LayoutMarginsGuide



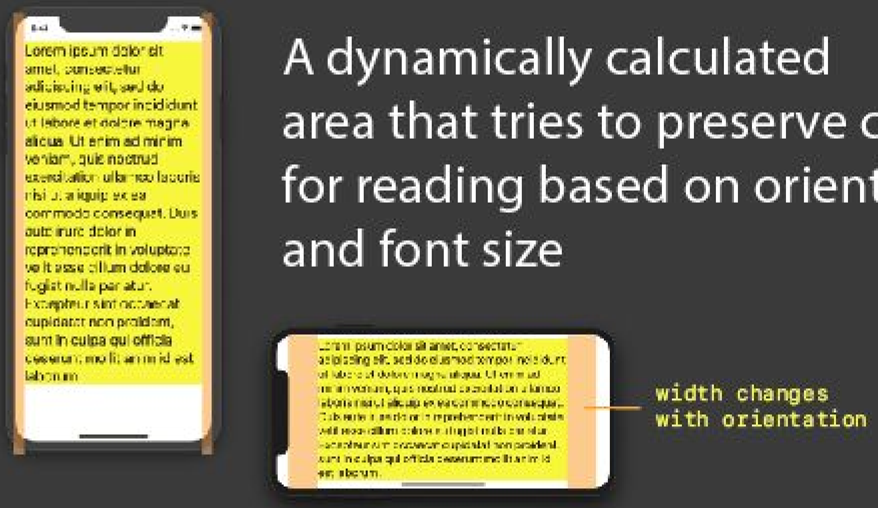
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

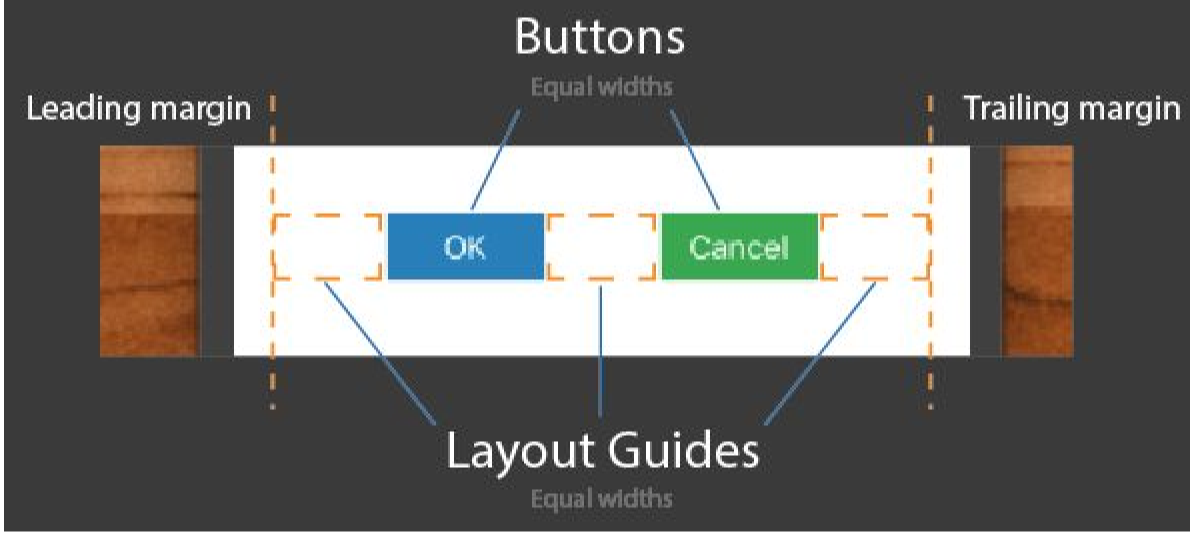
Rectangular layout guide used to provide default margins for spacing as well as custom layout areas for extra space

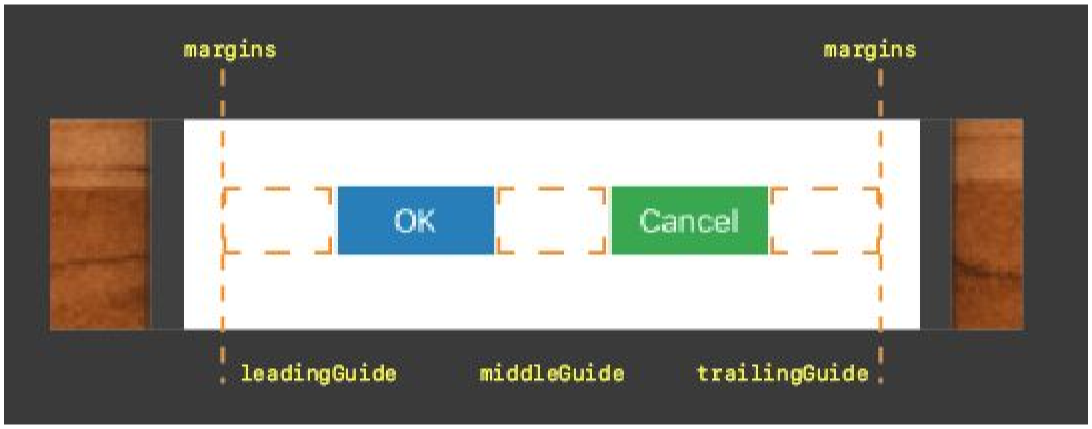
* Default values (subject to change)

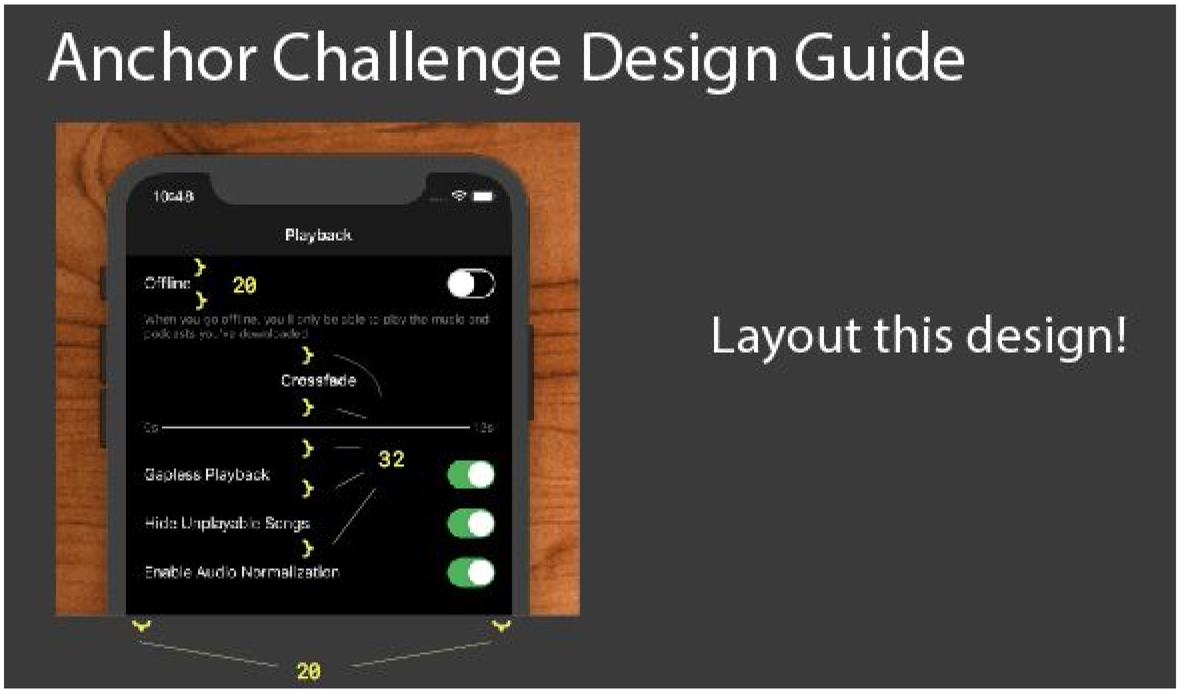
ReadableContentGuide

A dynamically calculated area that tries to preserve content for reading based on orientation and font size


Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
width changes
with orientation







Anchor Challenge Design Guide

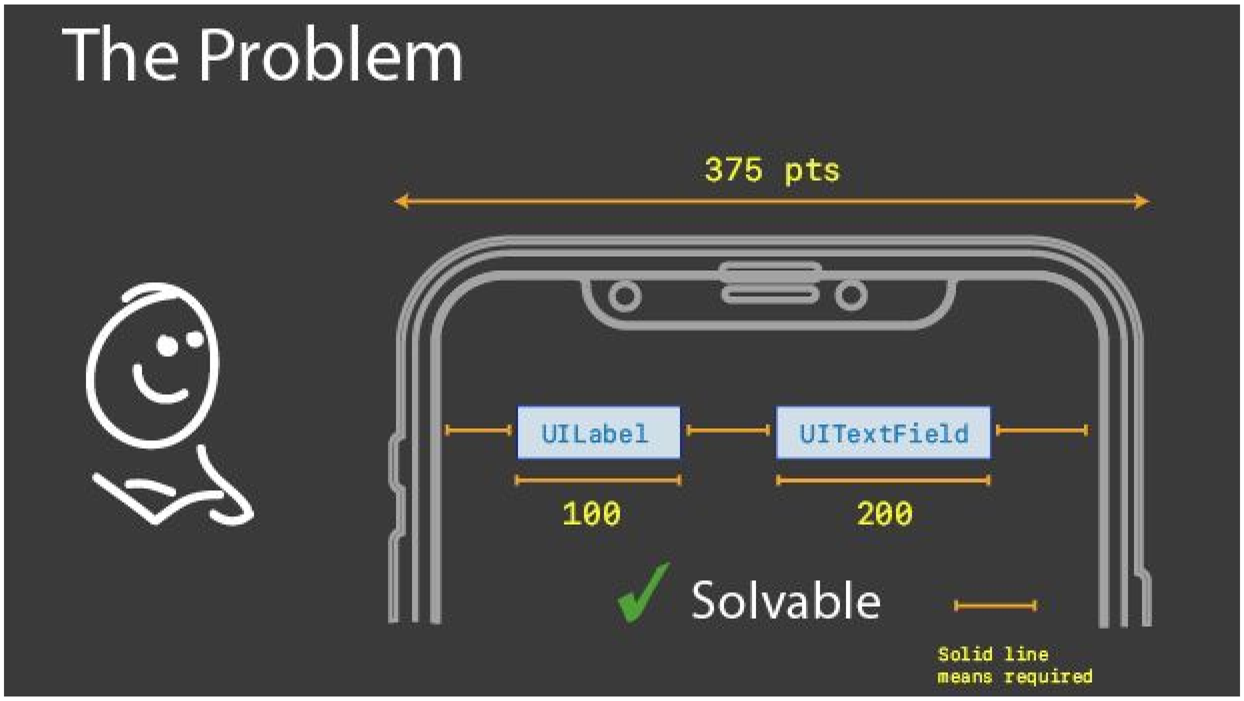
Layout this design!

10:48
Playback
Offline
20
When you go offline, you'll only be able to play the music and podcasts you've downloaded.
Crossfade
0s
12s
32
Gapless Playback
Hide Unplayable Songs
Enable Audio Normalization
20

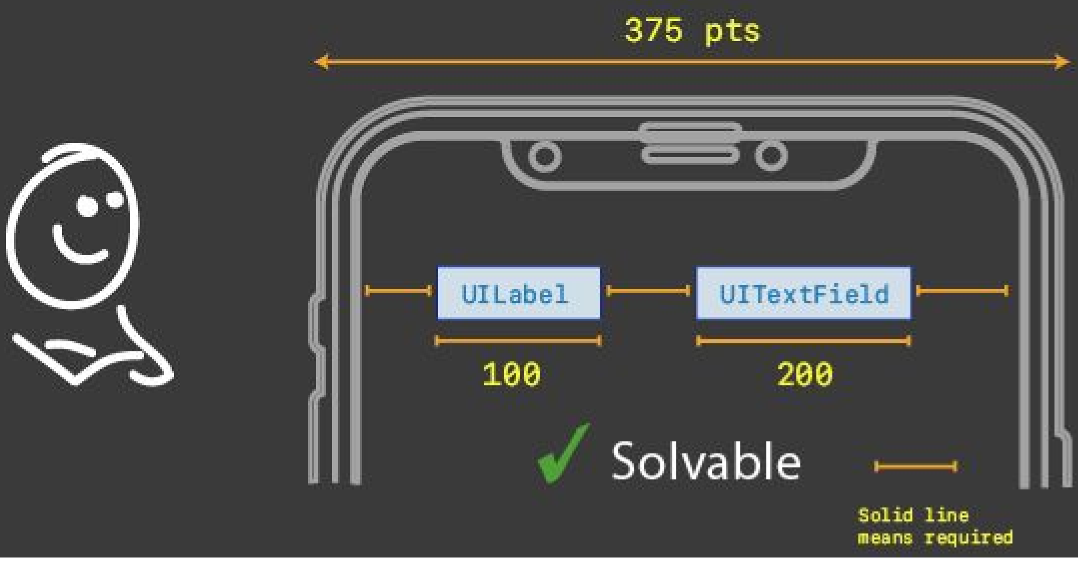
Intrinsic Content Size

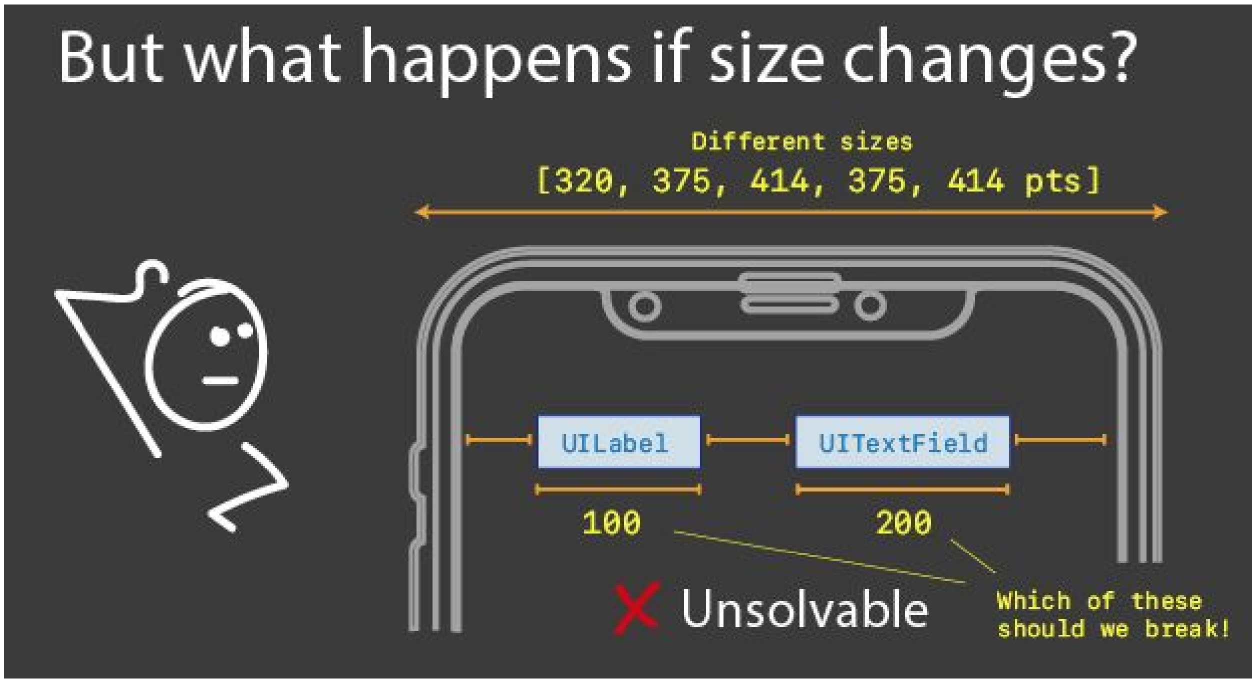
Content Hugging

Compression Resistance



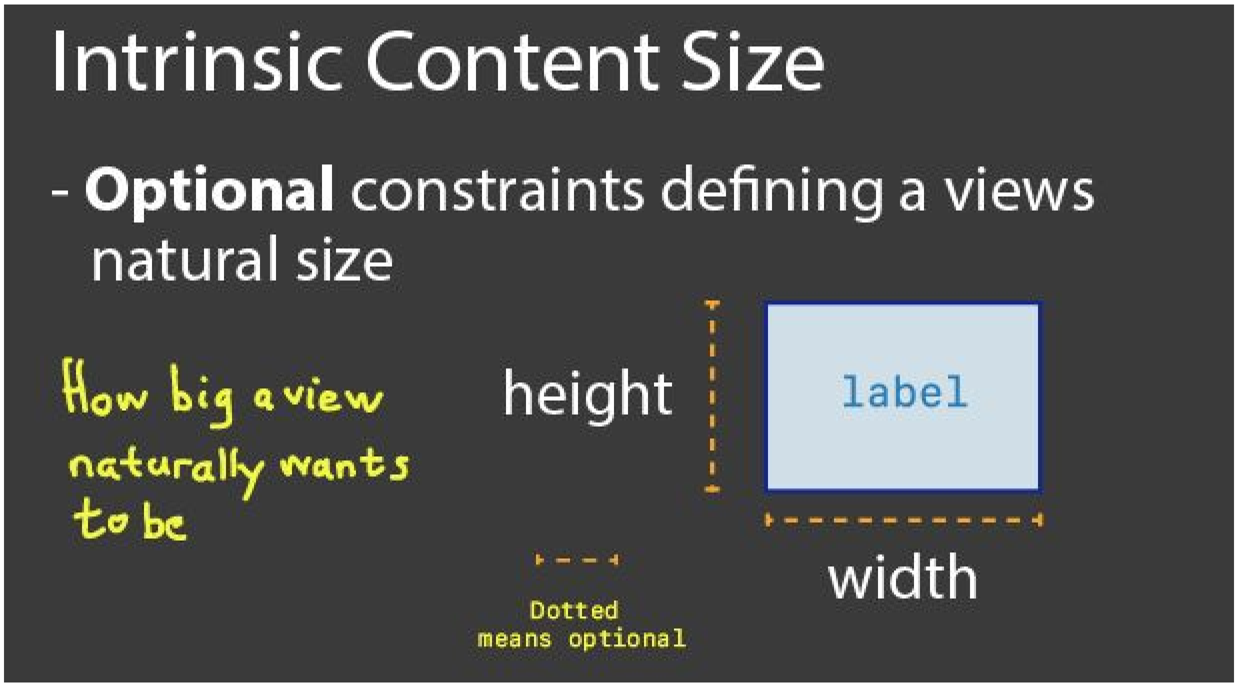
The Problem





But what happens if size changes?

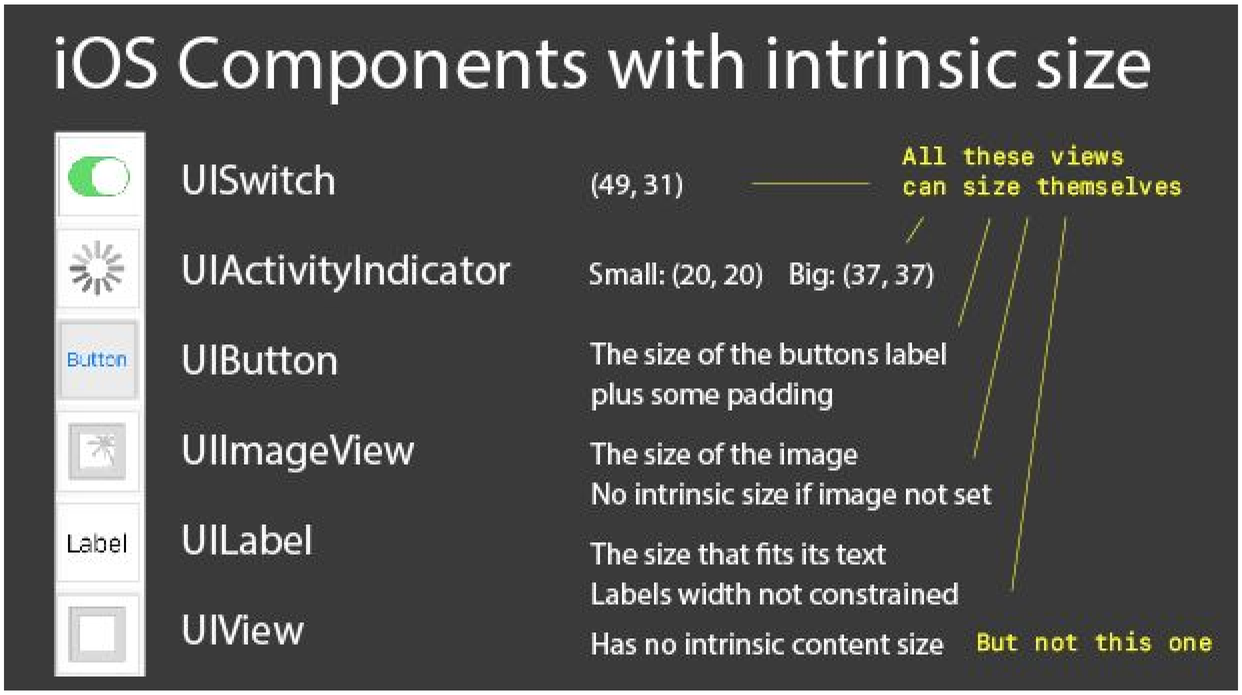
Different sizes
[320, 375, 414, 375, 414 pts]
UILabel
UITextField
100
200
❌ Unsolvable
Which of these should we break!



Intrinsic Content Size

- **Optional** constraints defining a views natural size

How big a view naturally wants to be

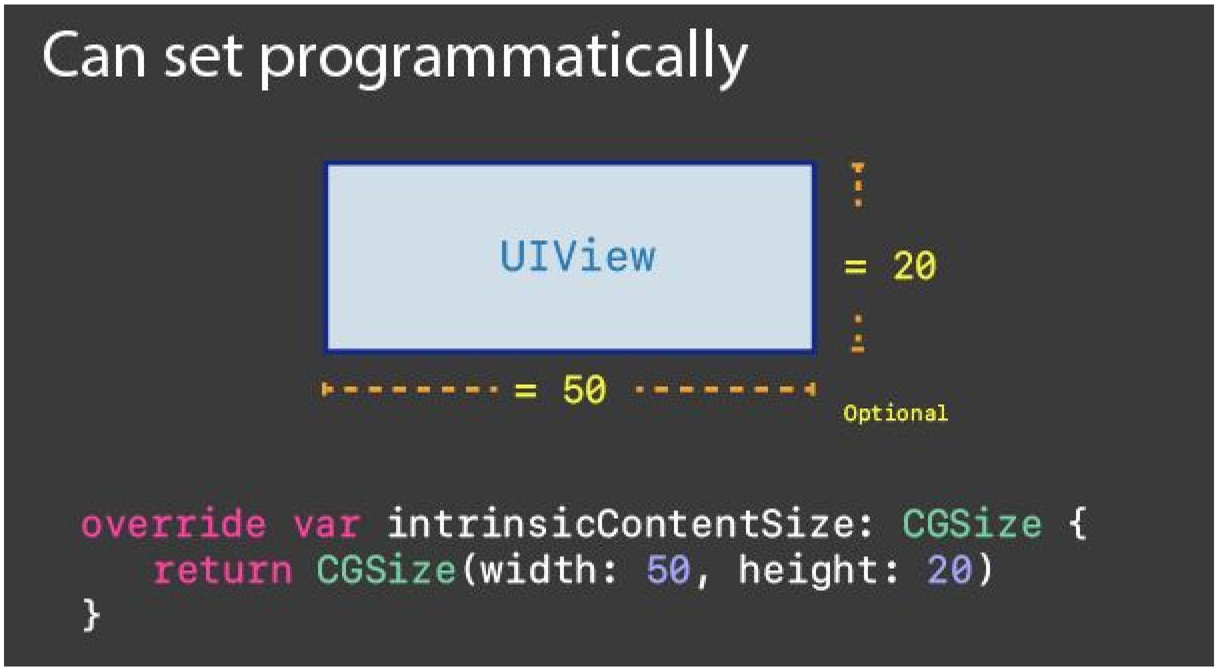


iOS Components with intrinsic size

All these views can size themselves

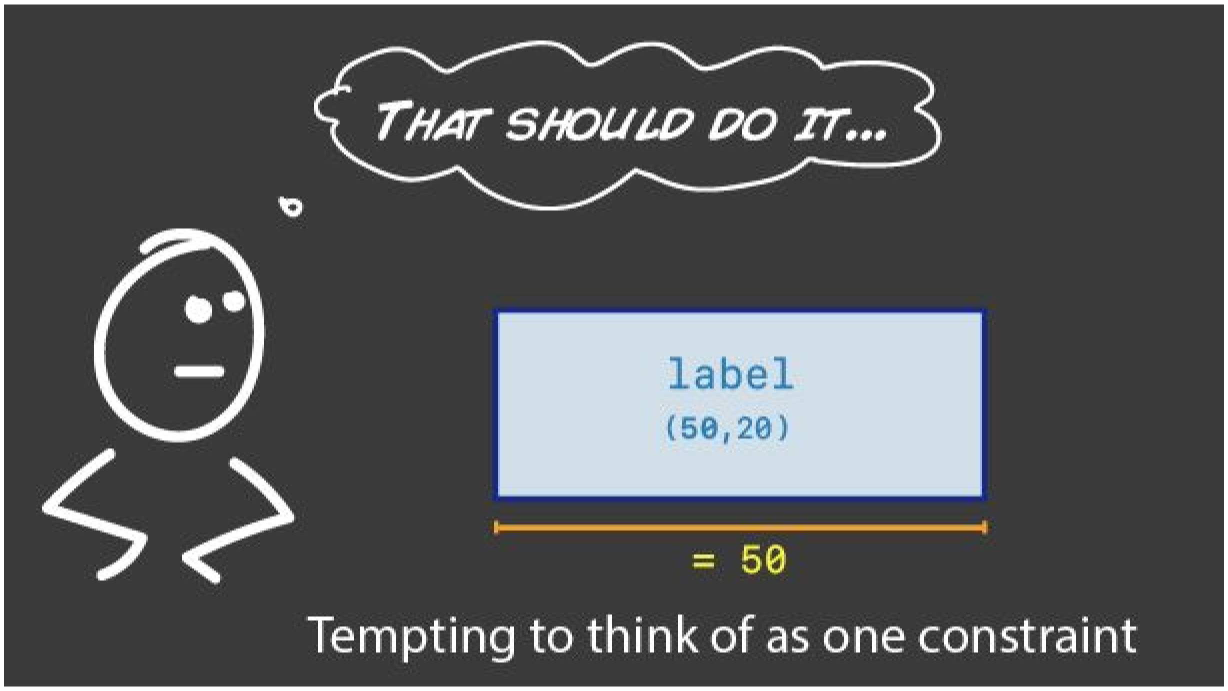
UISwitch	(49, 31)
UIActivityIndicator	Small: (20, 20) Big: (37, 37)
UIButton	The size of the buttons label plus some padding
UIImageView	The size of the image No intrinsic size if image not set
UILabel	The size that fits its text Labels width not constrained
UIView	Has no intrinsic content size

But not this one

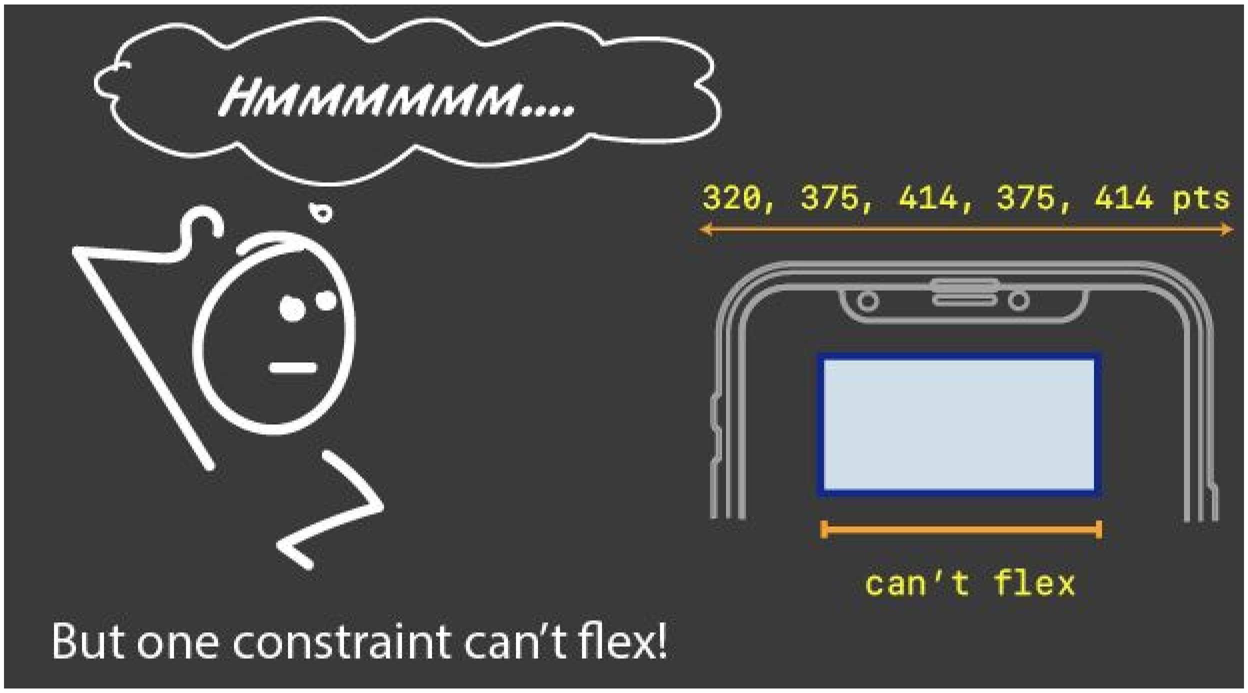


Can set programmatically

```
override var intrinsicContentSize: CGSize {
   return CGSize(width: 50, height: 20)
}
```



Tempting to think of as one constraint



But one constraint can’t flex!

What we need is

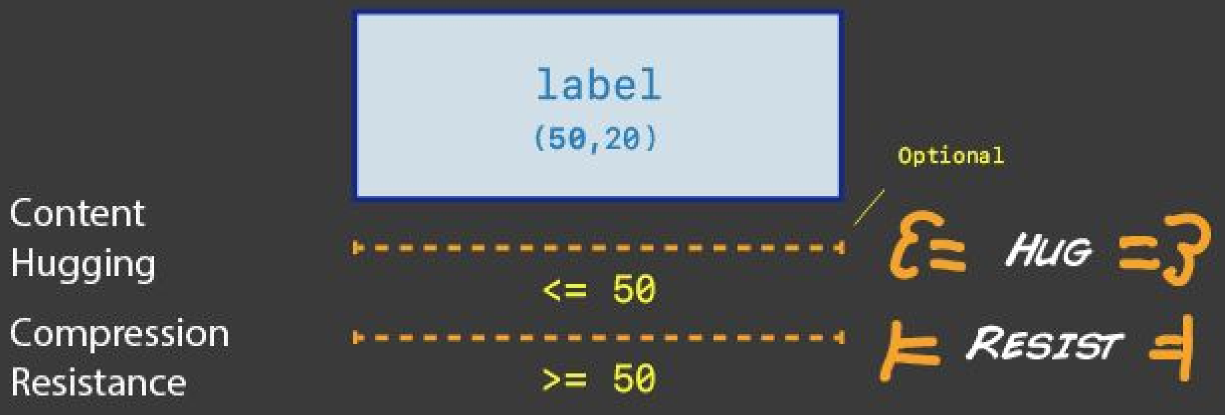
- Away of defining a controls natural size
- Yet can flex

Content Hugging Compression Resistance

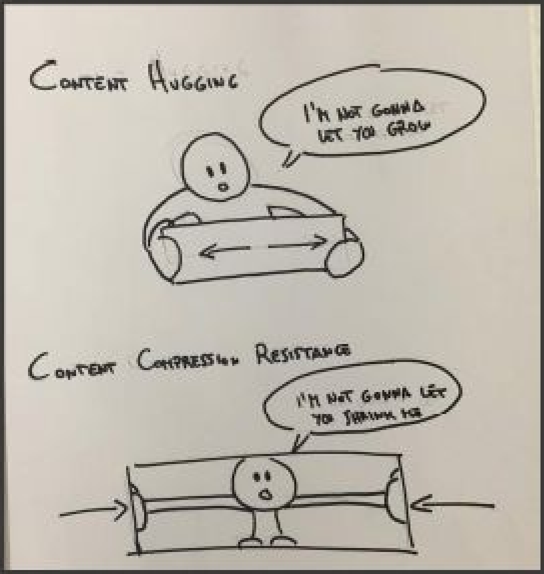
aka CHCR

WHAT GIVES OUR CONTROLS THE ABILITY TO FLEX

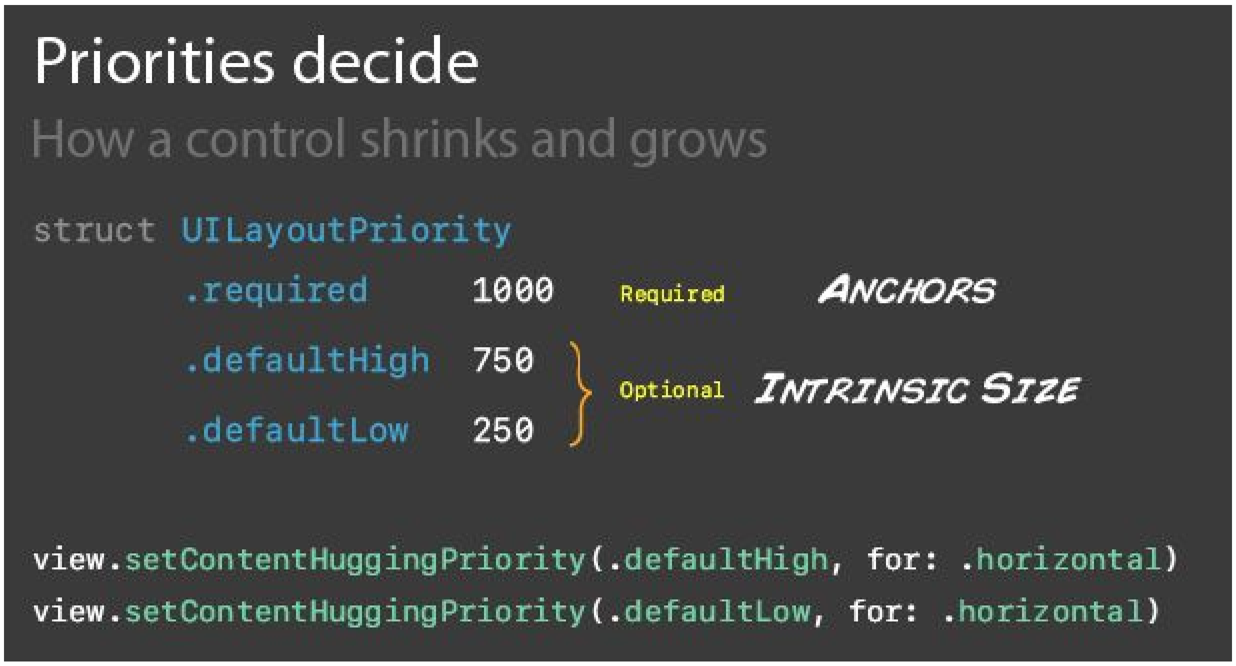
Each dimension requires two contraints



Can shrink and grow


CONTENT HUGGING
I'M NOT GONNA LET YOU GROW
CONTENT COMPRESSION RESISTANCE
I'M NOT GONNA LET YOU SHRINK ME

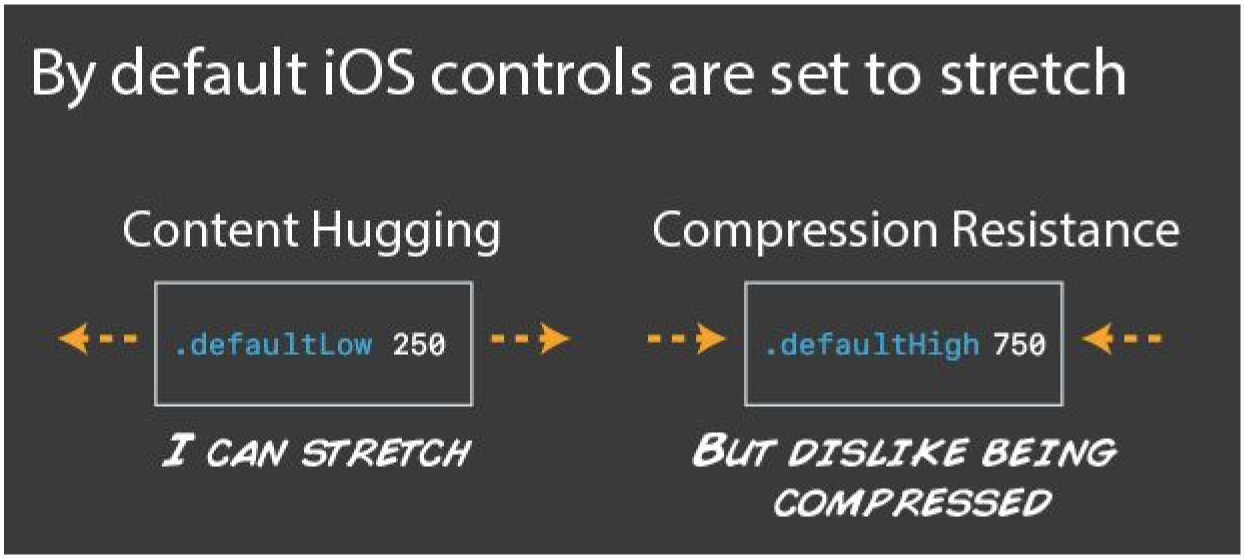
@MOKAGIO



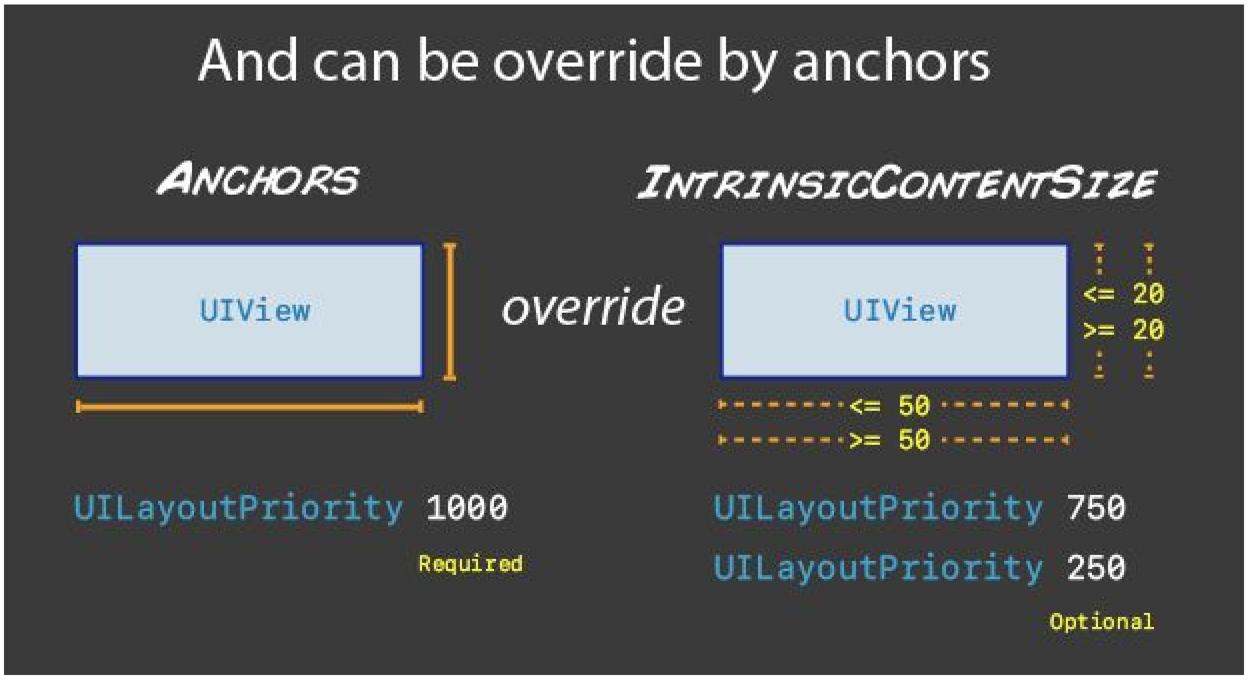
Priorities decide

How a control shrinks and grows

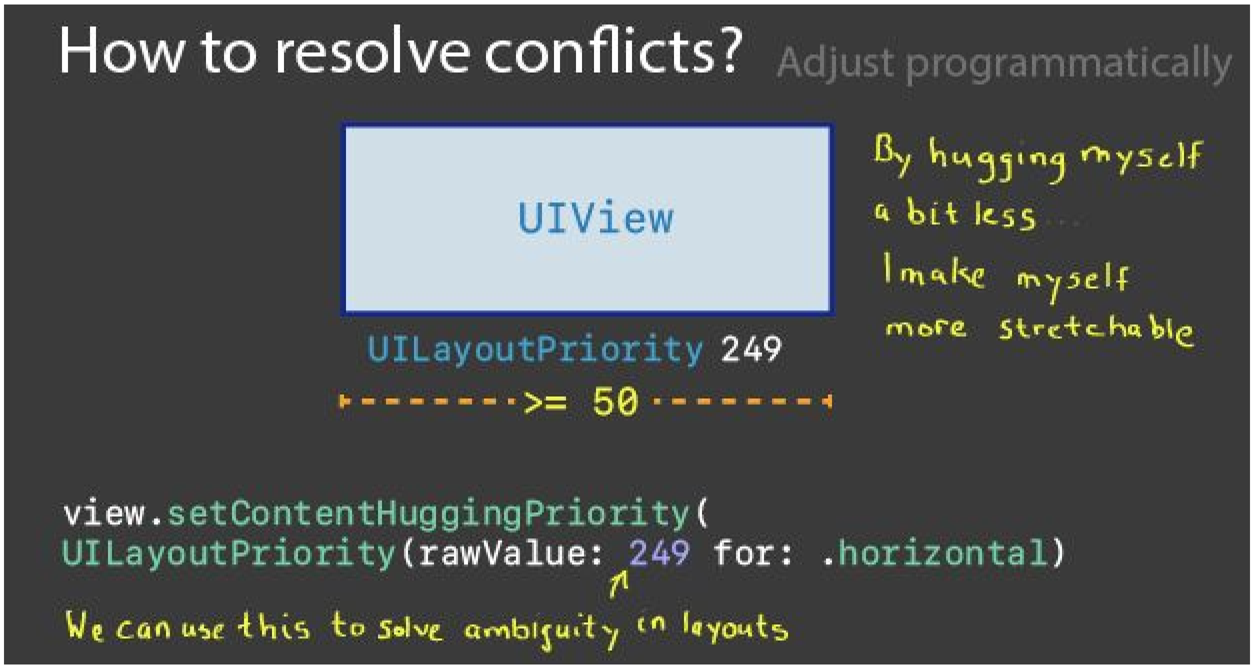
```
struct UILayoutPriority
        .required     1000    Required      ANCHORS
        .defaultHigh  750  }
                           }  Optional   INTRINSIC SIZE
        .defaultLow   250  }


view.setContentHuggingPriority(.defaultHigh, for: .horizontal)
view.setContentHuggingPriority(.defaultLow, for: .horizontal)
```



By default iOS controls are set to stretch



And can be override by anchors



How to resolve conflicts?

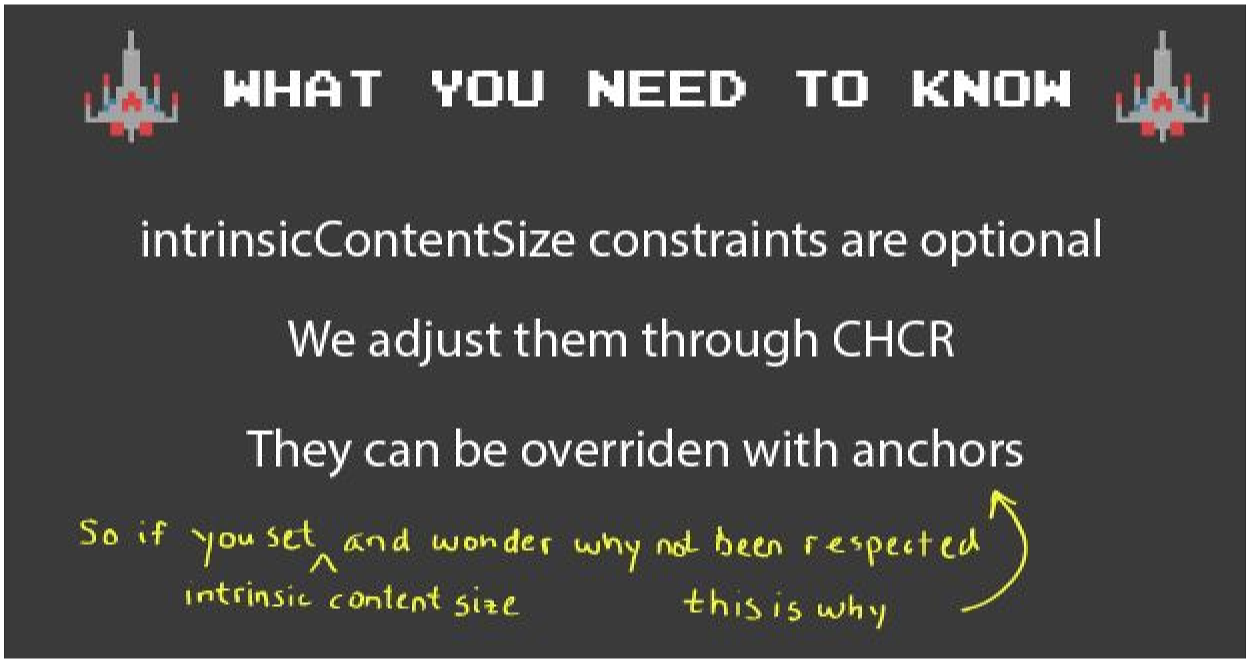
Adjust programmatically

UIView
UILayoutPriority 249
>= 50

By hugging myself a bit less...
I make myself more stretchable

```
view.setContentHuggingPriority(
UILayoutPriority(rawValue: 249 for: .horizontal)
```

We can use this to solve ambiguity in layouts



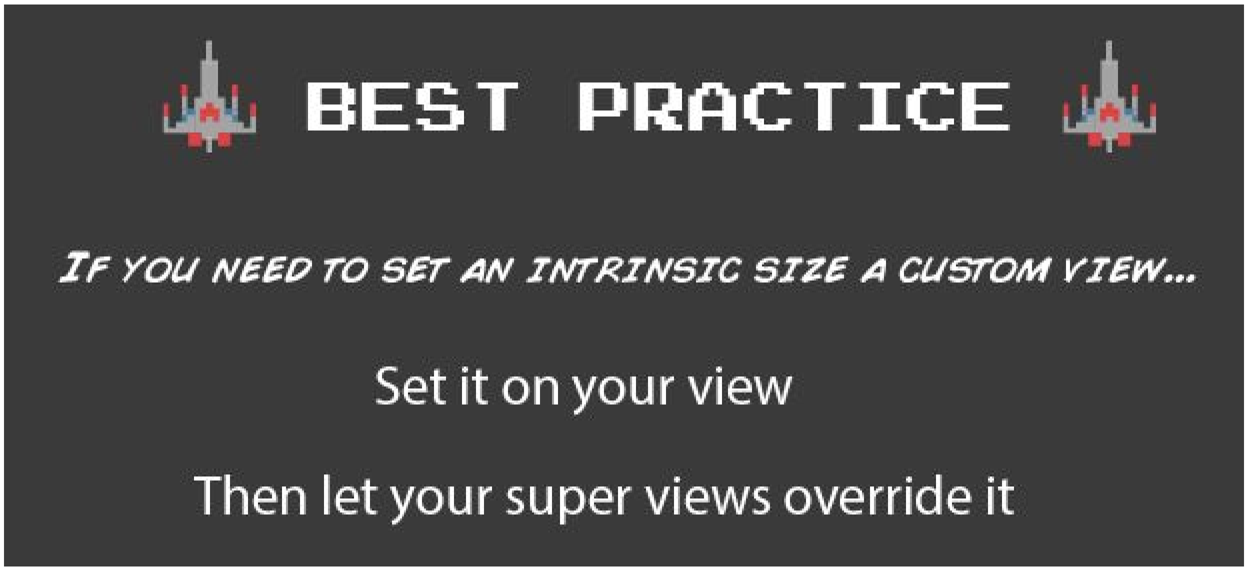
WHAT YOU NEED TO KNOW

intrinsicContentSize constraints are optional

We adjust them through CHCR

They can be overriden with anchors

So if you set ^ and wonder why not been respected
intrinsic content size this is why



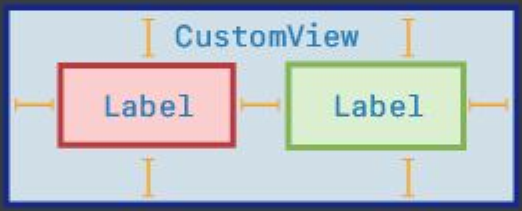
BEST PRACTICE

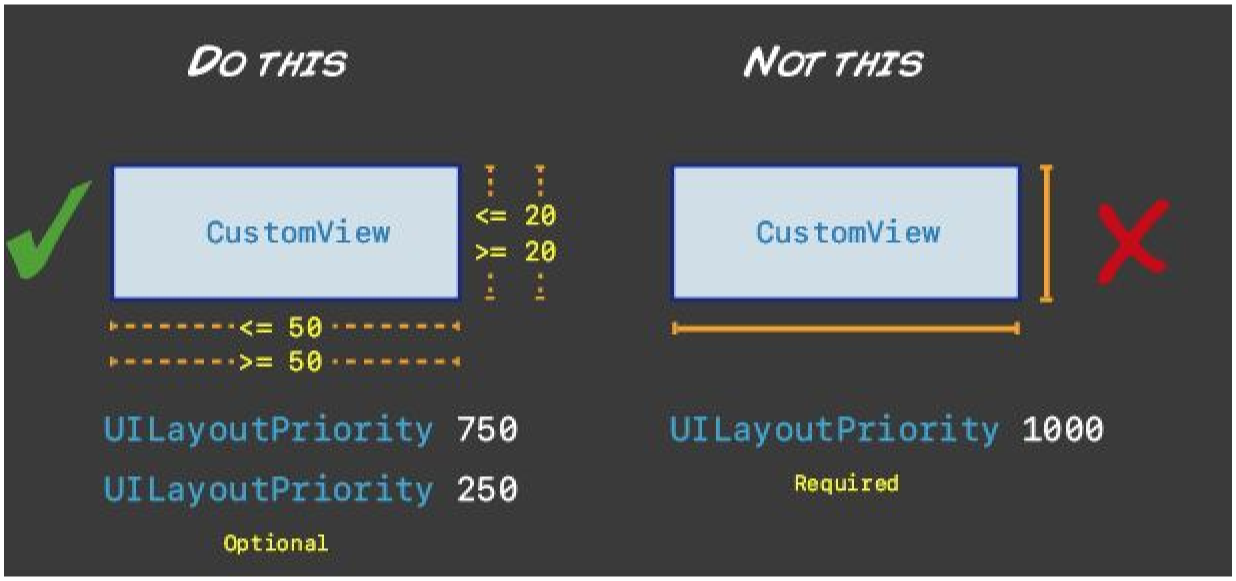
If you need to set an intrinsic size a custom view...

Set it on your view

Then let your super views override it

Most custom views don't require an intrinsicContentSize





Do this

`UILayoutPriority 750`

`UILayoutPriority 250`

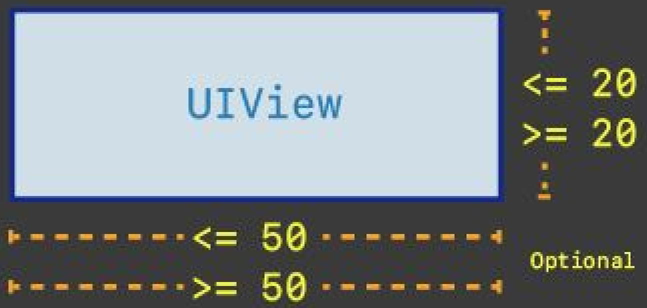
Optional

Not this

`UILayoutPriority 1000`

Required

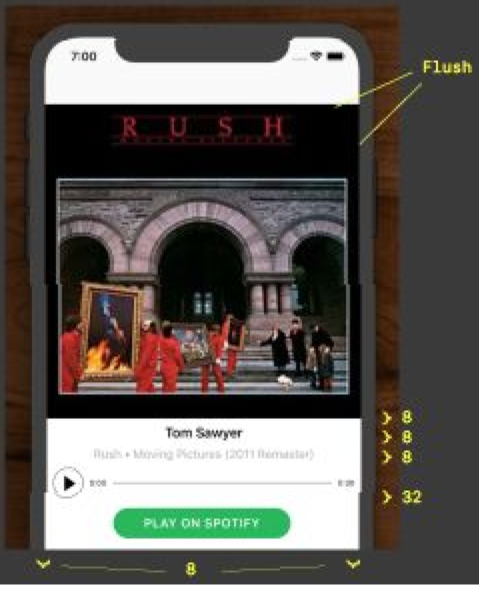
Can set programmatically



```swift
override var intrinsicContentSize: CGSize {
   return CGSize(width: 50, height: 20)
}
```

Suggested size

Preview Screen Design Challenge

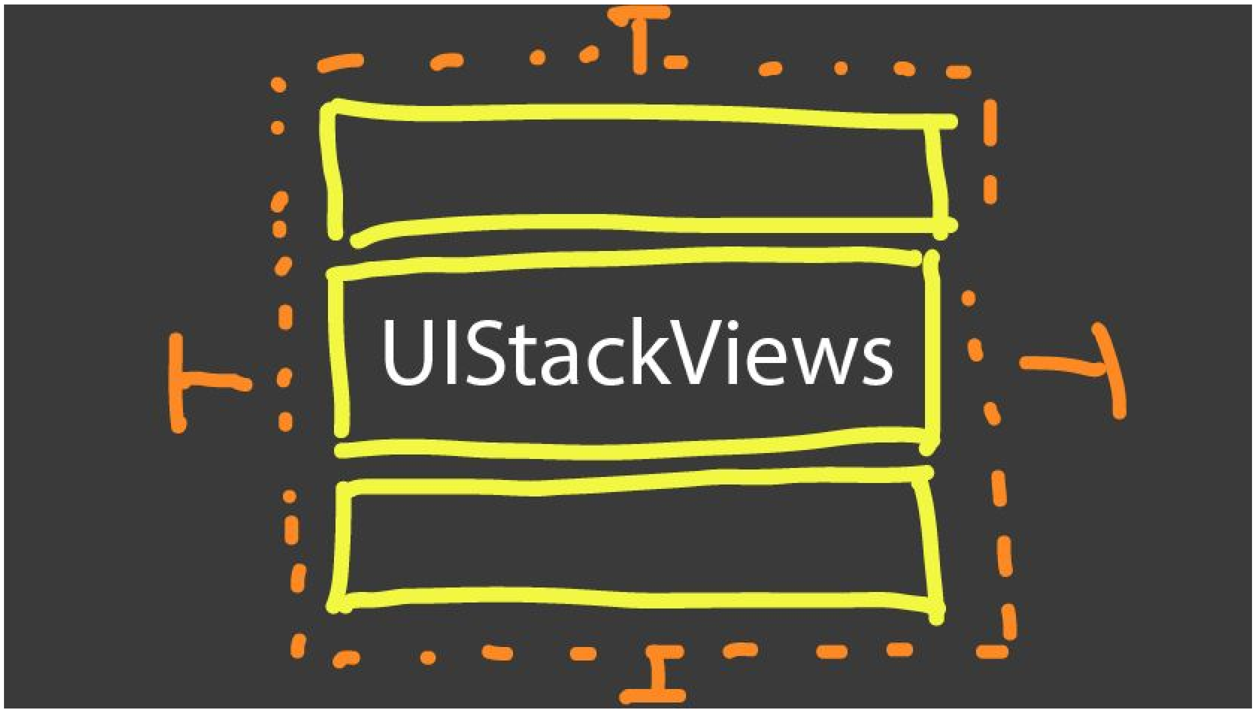


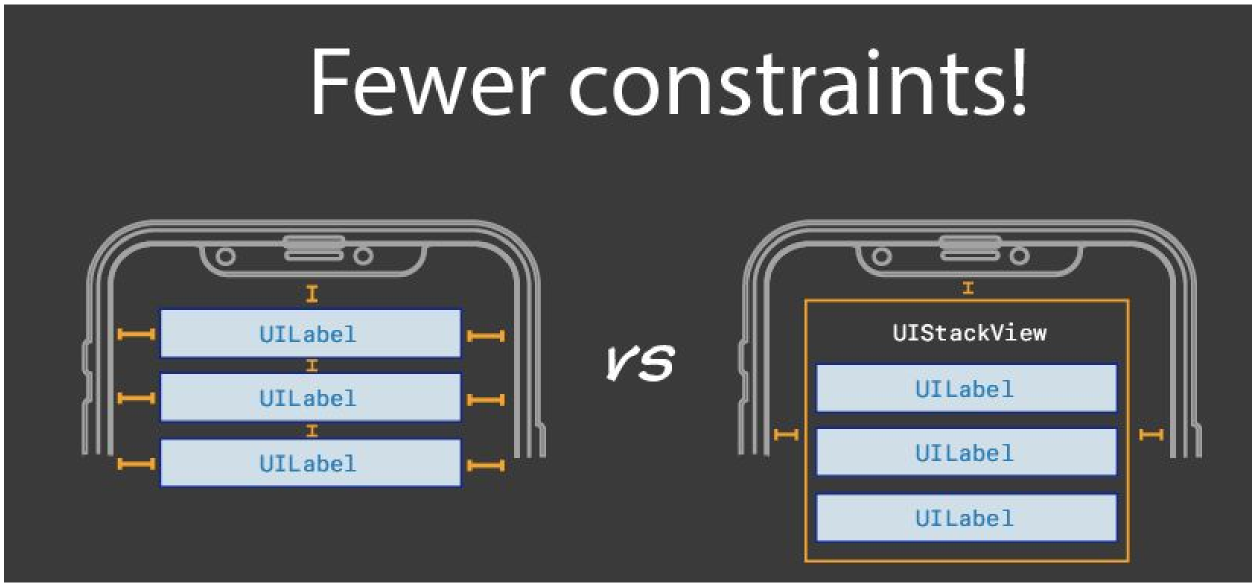
Layout this design!

LEVEL UP

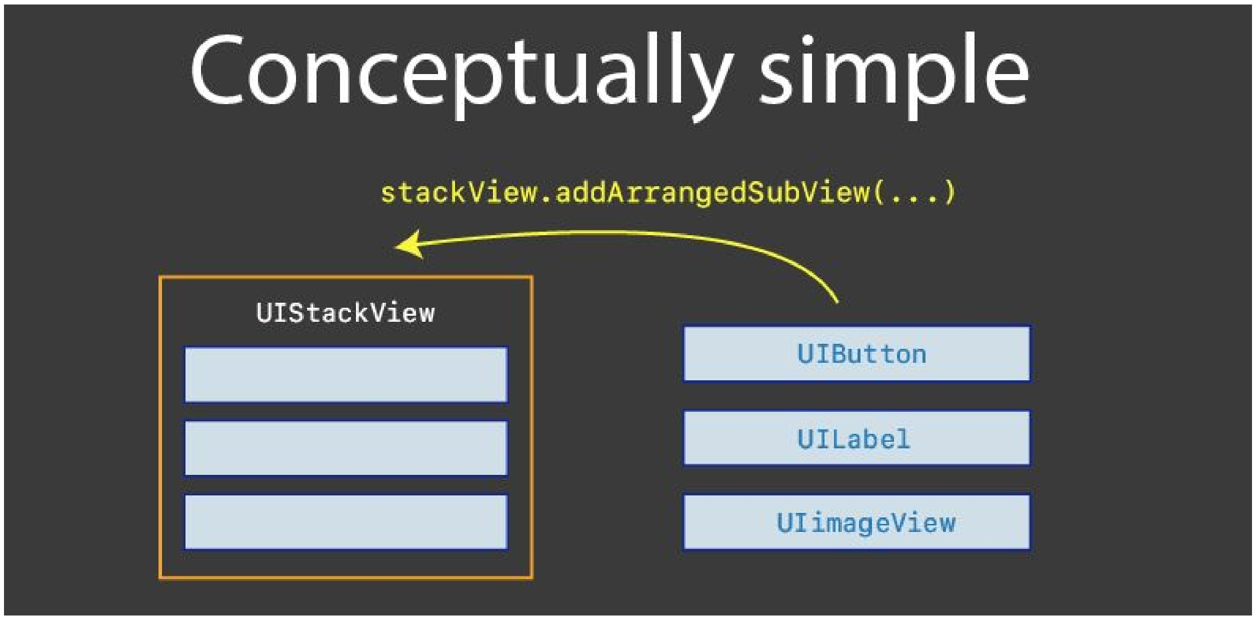
Intrinsic Content Size

- Allows controls to lay themselves out without requiring full constraints
- CHCR is how we resolve ambiguity
- Constraints have priorities
- Need to adapt and be flexible...





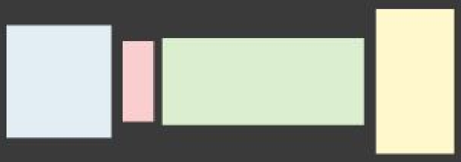
Fewer constraints!

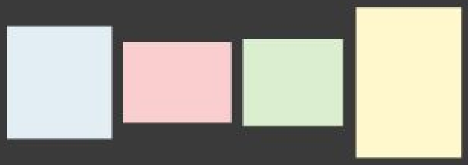


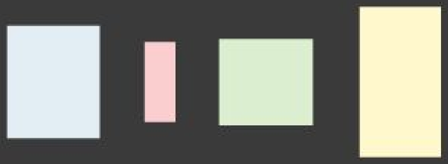
Conceptually simple

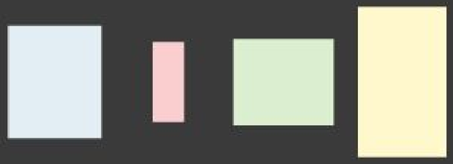
```
stackView.addArrangedSubView(...)
```

Decide on distribution...


fill

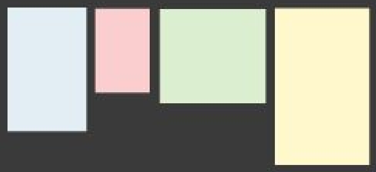

fill equally

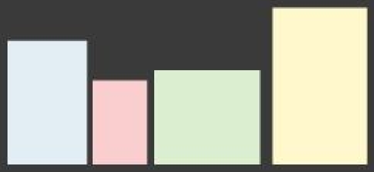

equal spacing

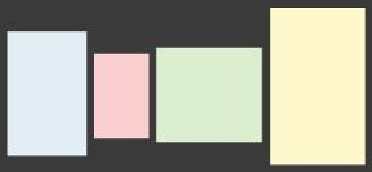

equal centering

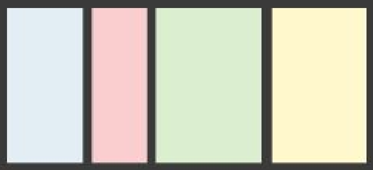
...and alignment

horizontal

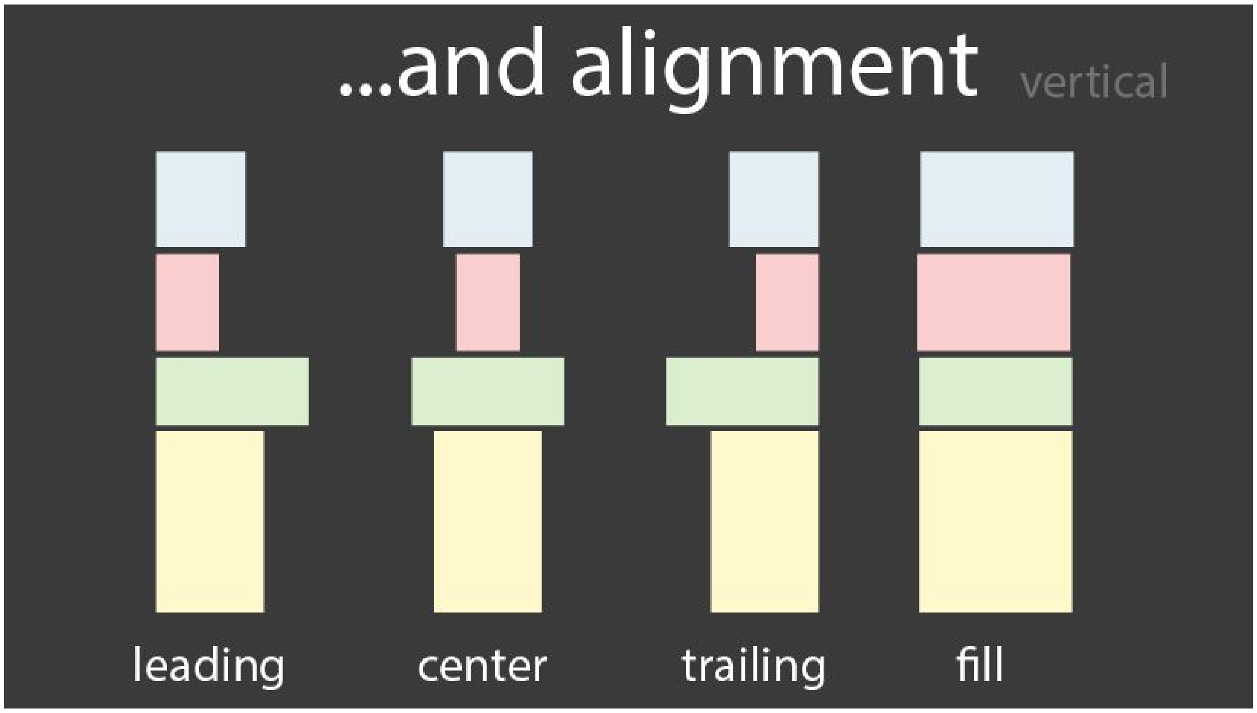

top


bottom

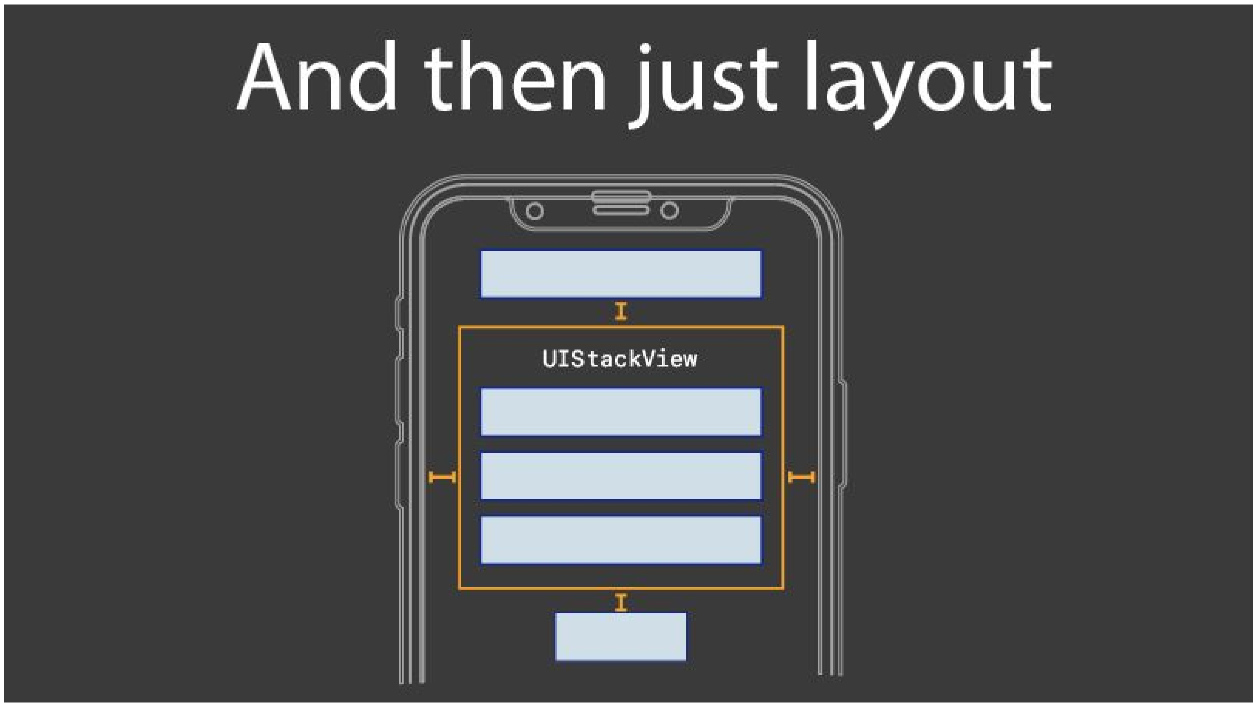

center


fill

...and alignment

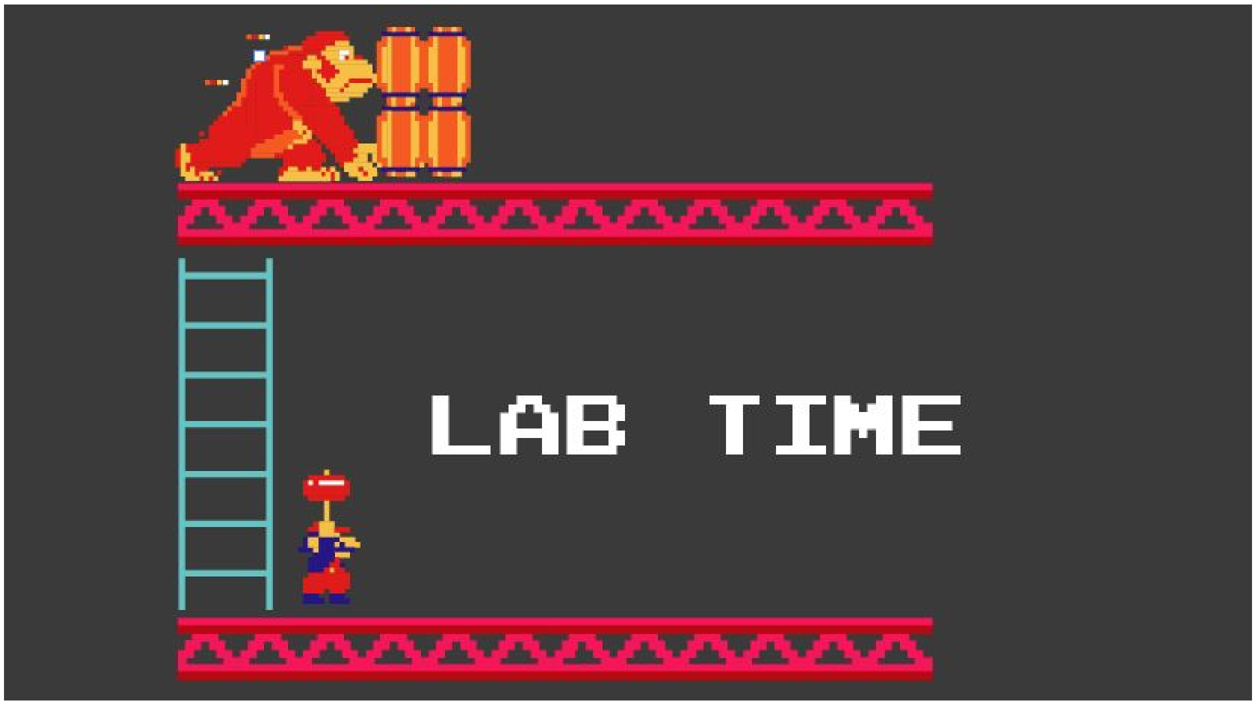


And then just layout

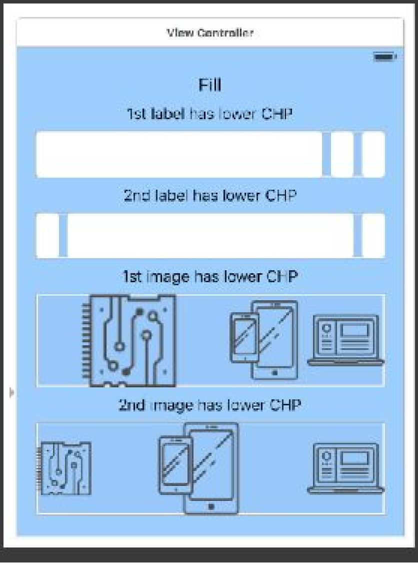


What you need to know

- UIStackView is a container
  - has no intrinsic content size of it's own
- Not all distributions work the same
- Everything inside needs to be intrinsically content sized ↖ Everything needs to be able to size itself



Distribution - Fill



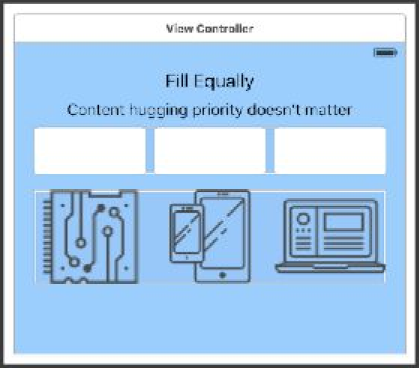
Fills all available space

Default setting

Uses intrinsic content size (CHCR)

If CHCR the same - will complain

Distribution - Fill Equally



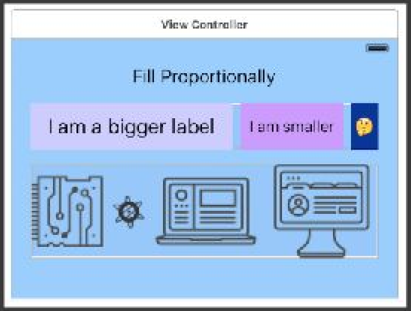
Makes all controls the same size

Only disribution NOT to use intrinsic content size

Remember: intrinsic content size is an recommendation - not a requirement.

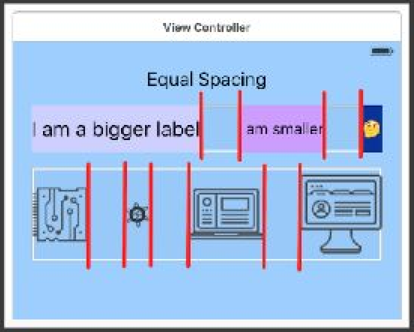
So fill equally will break the optional intrinsic content size in order to fill equally.

Distribution - Fill Proportionally



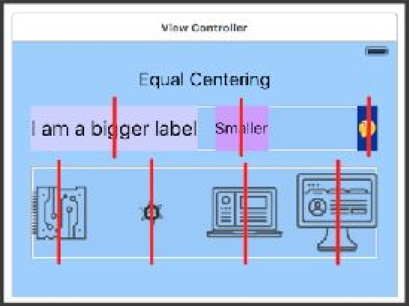
Maintains proportions as layout grows and shrinks

Distribution - Equal Spacing



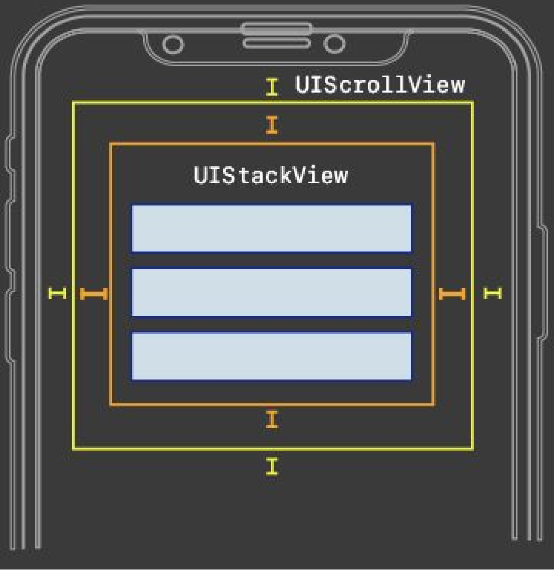
Maintains equal space between each control

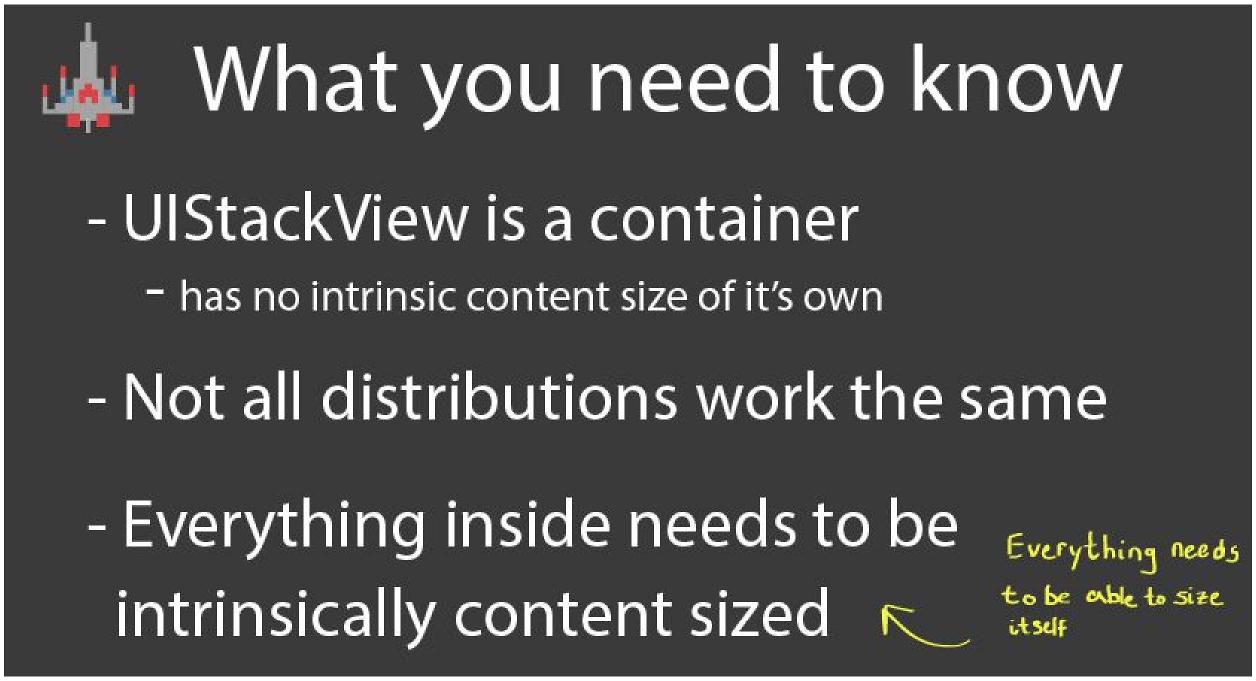
Distribution - Equal Centering



Spaces equally between center of controls

Scrollable StackViews

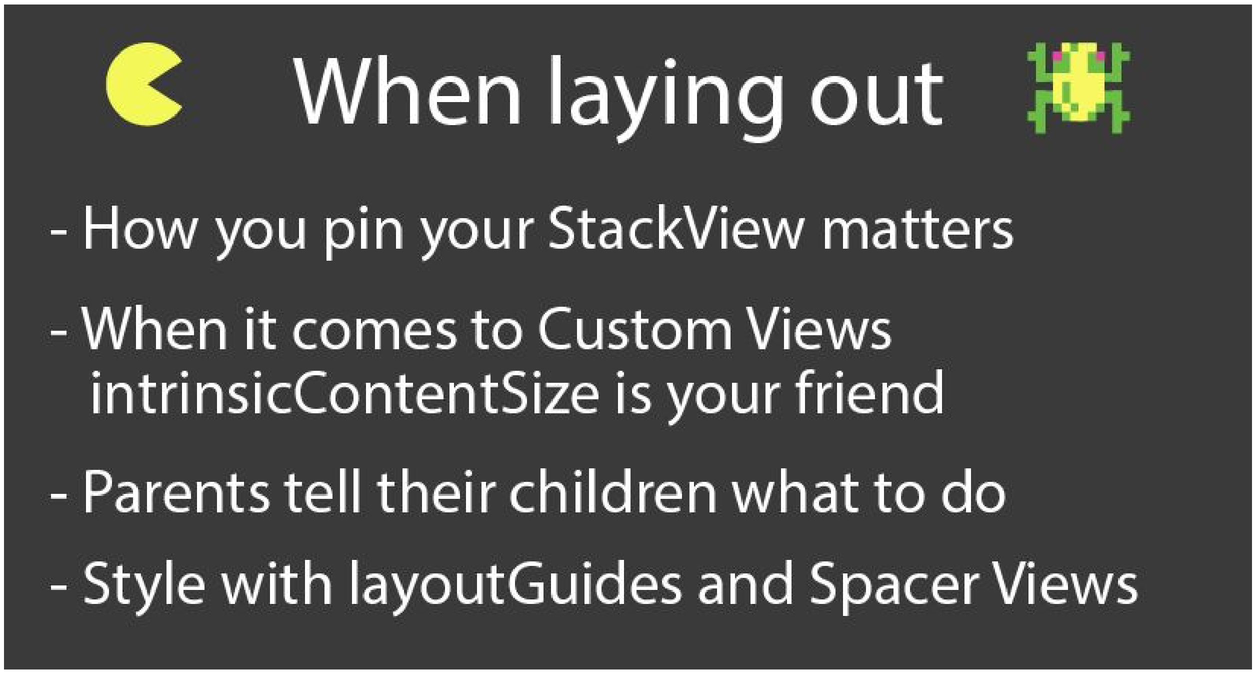




What you need to know

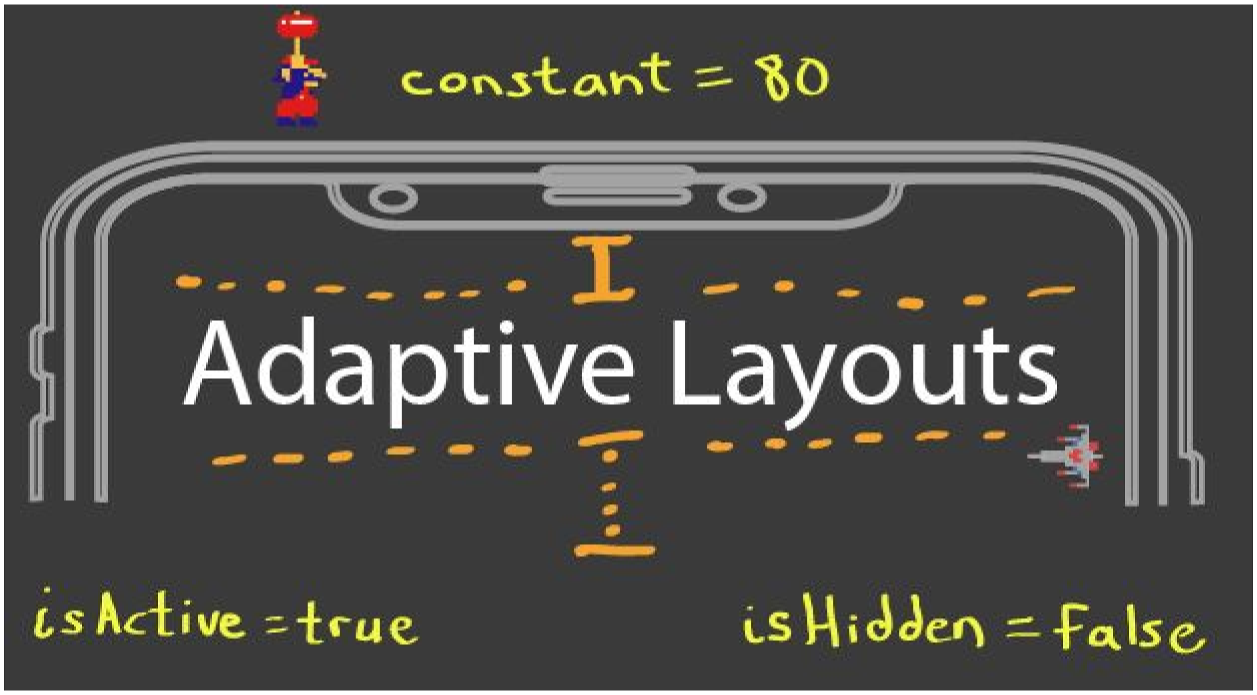
- UIStackView is a container
  - has no intrinsic content size of it’s own
- Not all distributions work the same
- Everything inside needs to be intrinsically content sized

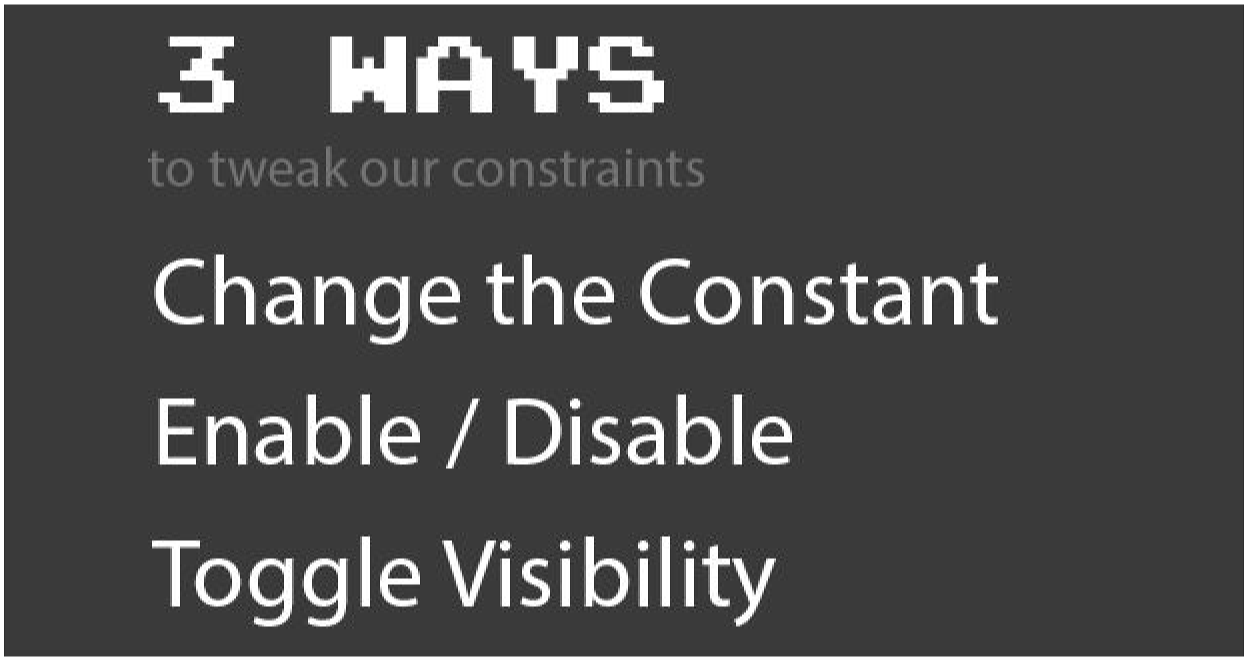
Everything needs to be able to size itself



When laying out

- How you pin your StackView matters
- When it comes to Custom Views intrinsicContentSize is your friend
- Parents tell their children what to do
- Style with layoutGuides and Spacer Views





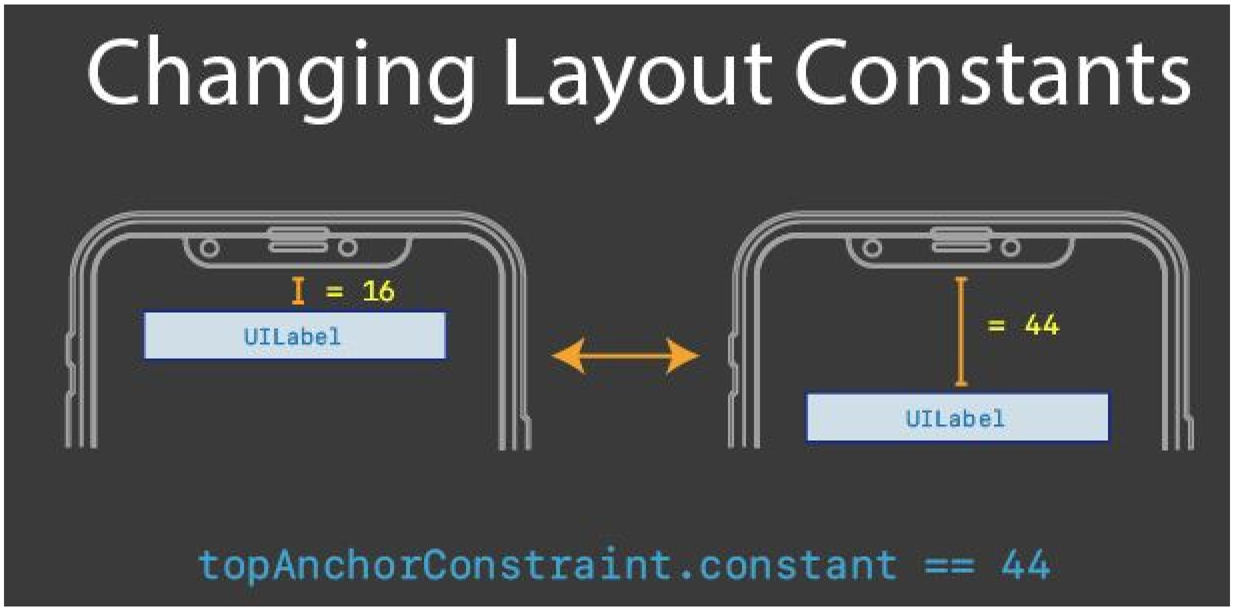
3 WAYS

to tweak our constraints

Change the Constant

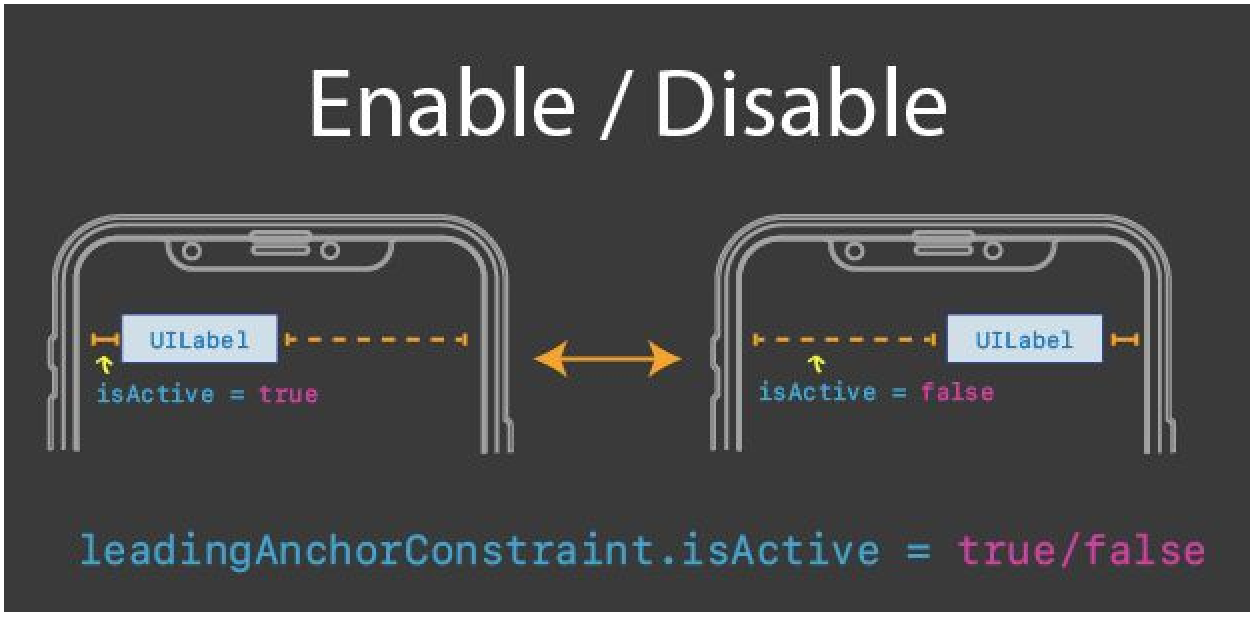
Enable / Disable

Toggle Visibility



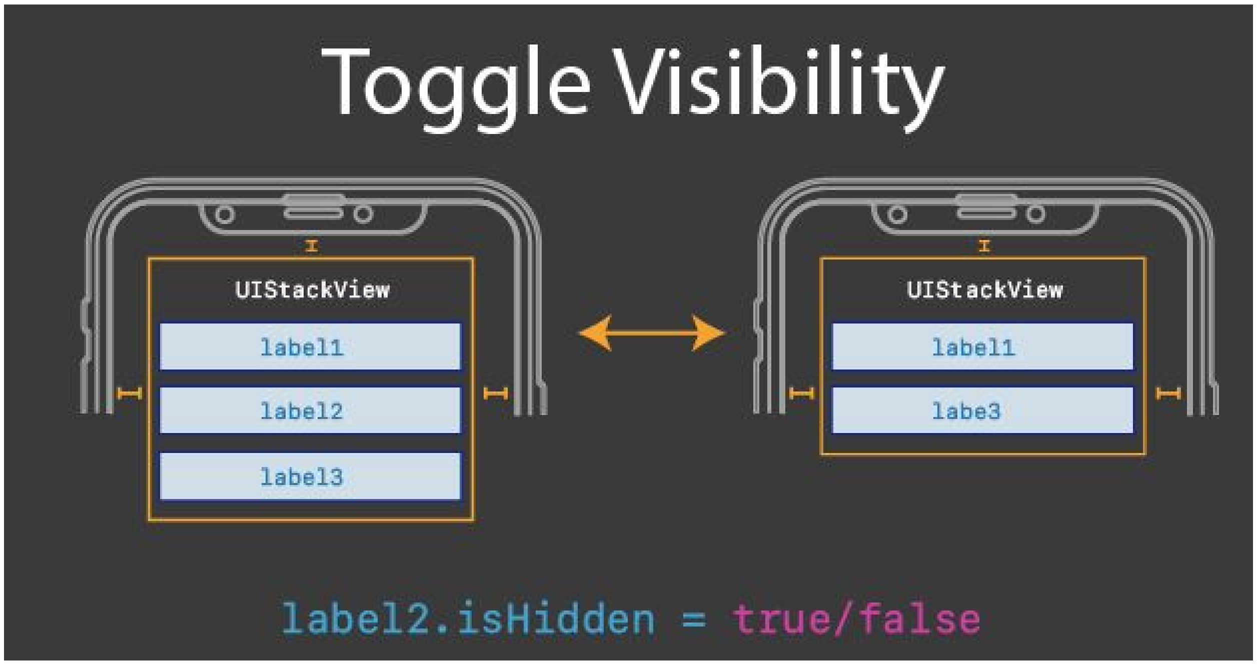
Changing Layout Constants

```
topAnchorConstraint.constant == 44
```



Enable / Disable

```
leadingAnchorConstraint.isActive = true/false
```



```
label2.isHidden = true/false
```