

# 4 - Why conflicting constraint happen and how to solve them

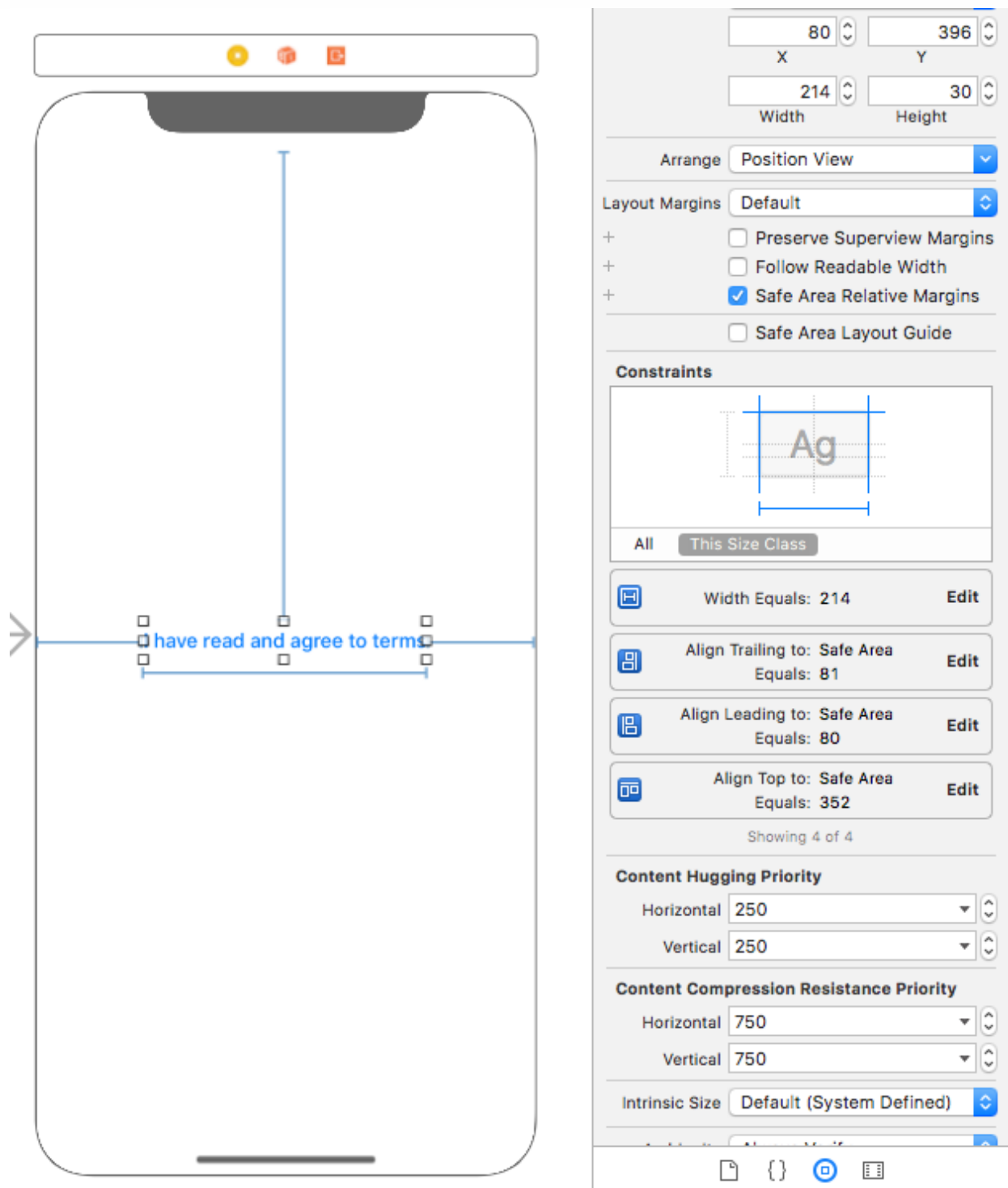
---

This is the fourth part of Making Sense of Auto Layout Series.

Previously, we have explained why missing constraints appear and how to solve them. In this post we will look into why Xcode shows you red lines with numbers when designing layout and how to solve them.

Ever experience that you finally got rid of all the red lines and it looks good on your device, but when you view it on another device in simulator, it look totally off?

This is probably because you have conflicting constraint in the layout, we will use a simple example below. The example contain a vertically centered button.

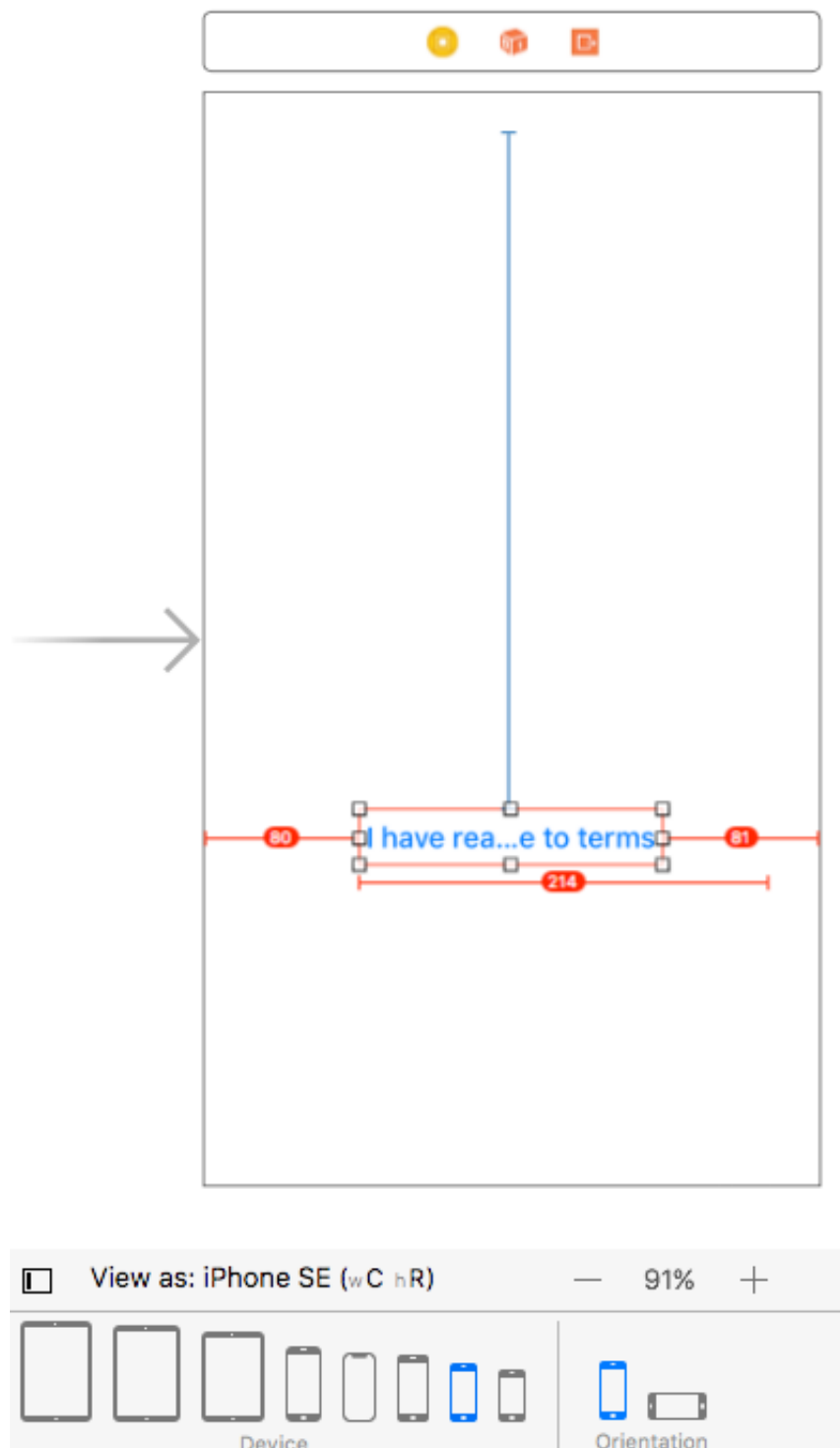


The constraints are :

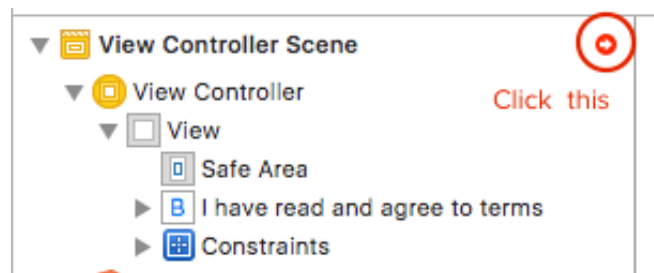
1. Width of the button equals 214 pt
2. Distance from the screen left to the left of the button is 80 pt
3. Distance from the screen right to the right of the button is 81 pt
4. Distance from the screen top to the top of the button is 352 pt

It looks good and all the lines are blue, yay! Enough constraints are defined and Xcode is able to calculate the x position, y position, width and height of the button.

It looks good on iPhone X, but once you switch the view to iPhone SE, there's red lines everywhere with numbers 😱 (this means that there are conflicting constraints), and the button text got squeezed :



Similar to solving missing constraint, we can troubleshoot conflicting constraints by clicking white arrow icon inside red circle button :



And we will see that Xcode complains to us that there's conflicting constraints :



Hmmm... so whats with the conflicting constraints? Remember how Auto Layout calculate size of a view (chapter 2)? We will first discuss why the layout works fine on iPhone X.

## Why it works fine on iPhone X

Since iPhone X has a screen width of 375 point,

From the leading and trailing constraints, Auto Layout Engine can deduce that the width of the button with this calculation :

```
ScreenWidth = 375
LeadingConstraint = 80
TrailingConstraint = 81
widthOfButton = ScreenWidth - LeadingConstraint -
TrailingConstraint
widthOfButton = 375 - 80 - 81
widthOfButton = 214
```

So the width of button is 214 by calculation of leading and trailing constraints we have defined. But remember we also explicitly defined the width of the button to 214.

```
// explicitly defined width constraint
widthOfButton = 214
```

Since the explicitly defined width is equal to the width derived from using leading and trailing constraints, Xcode doesn't complain.

# Why it doesn't work on iPhone SE (and also other screen width)

Well iPhone SE has a screen width of 320 point,

Using the same calculation with leading and trailing constraints :

```
ScreenWidth = 320
LeadingConstraint = 80
TrailingConstraint = 81
widthOfButton = ScreenWidth - LeadingConstraint -
TrailingConstraint
widthOfButton = 320 - 80 - 81
widthOfButton = 159
```

We calculated that the width of the button should be 159 points using the leading and trailing constraint.

But since we have another constraint explicitly mentioned the width of button to 214, Xcode got confused (use 159 as width or 214? they are conflicting 😞) thus showing you red lines with numbers.

When you run the app, you will see some error message in the console like this :

```
2018-03-26 22:22:21.586008+0800 LayoutIntro[33361:29112209] [LayoutConstraints] Unable to simultaneously satisfy constraints.
Probably at least one of the constraints in the following list is one you don't want.
Try this:
(1) look at each constraint and try to figure out which you don't expect;
(2) find the code that added the unwanted constraint or constraints and fix it.
(
    "<NSLayoutConstraint:0x6040000895b0 UIButton:0x7fde59e048d0'I have read and agree to ...'.width == 214 (active)>",
    "<NSLayoutConstraint:0x60800008ca80 UILayoutGuide:0x6080001ac780'UIViewSafeAreaLayoutGuide'.trailing == UIButton:
0x7fde59e048d0'I have read and agree to ...'.trailing + 81 (active)>",
    "<NSLayoutConstraint:0x60800008d390 UIButton:0x7fde59e048d0'I have read and agree to ...'.leading == UILayoutGuide:
0x6080001ac780'UIViewSafeAreaLayoutGuide'.leading + 80 (active)>",
    "<NSLayoutConstraint:0x60800008e100 'UIView-Encapsulated-Layout-Width' UIView:0x7fde59e044e0.width == 320 (active)>",
    "<NSLayoutConstraint:0x60800008d250 'UIViewSafeAreaLayoutGuide-left' H:[-()-]UILayoutGuide:
0x6080001ac780'UIViewSafeAreaLayoutGuide'] (LTR) (active, names: '|':UIView:0x7fde59e044e0 )>",
    "<NSLayoutConstraint:0x60800008d480 'UIViewSafeAreaLayoutGuide-right' H:[UILayoutGuide:
0x6080001ac780'UIViewSafeAreaLayoutGuide']-()-] (LTR) (active, names: '|':UIView:0x7fde59e044e0 )>"
)

Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x6040000895b0 UIButton:0x7fde59e048d0'I have read and agree to ...'.width == 214 (active)>

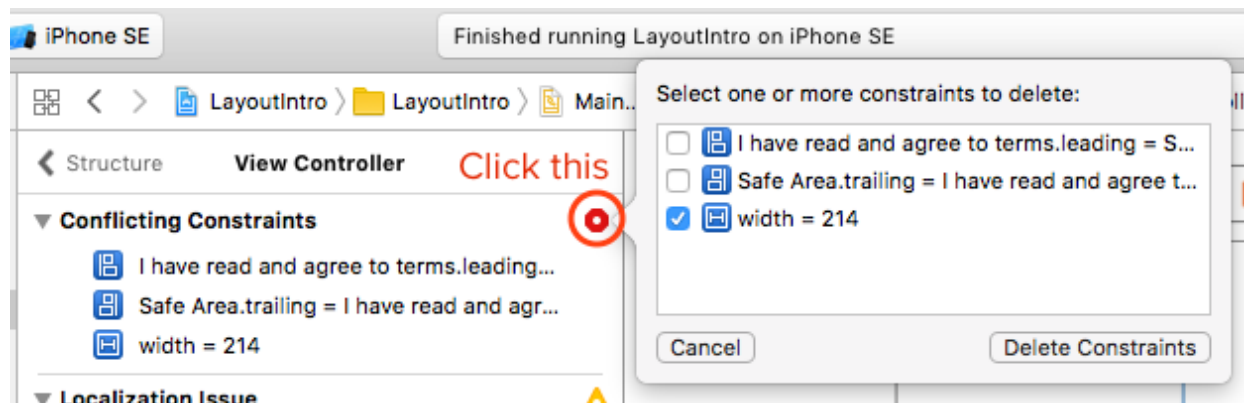
Make a symbolic breakpoint at UIViewAlertForUnsatisfiableConstraints to catch this in the debugger.
The methods in the NSLayoutConstraintBasedLayoutDebugging category on UIView listed in <UIKit/UIView.h> may also be helpful.
```

Auto Layout Engine can't satisfy all of the constraints as it is impossible to set the width of the button to 159 and 214 at the same time. Thus to render the button on screen, Auto Layout Engine will break (ignore) one of the constraints ^[Auto Layout Engine doesn't randomly choose constraint to break, it will choose with a certain order, but I don't really know what's the exact order], in this case, the explicit width 214 constraint will be ignored. After breaking (ignoring) the width 214 constraint, Auto Layout Engine will use the 159 as width of the button. As the width is too short, the text in the button got squeezed.

When there are several conflicting constraints while the app is running, Auto Layout Engine will start to break constraint one by one with a certain order until there is no conflict in the layout calculation (size and position) then only render the layout.

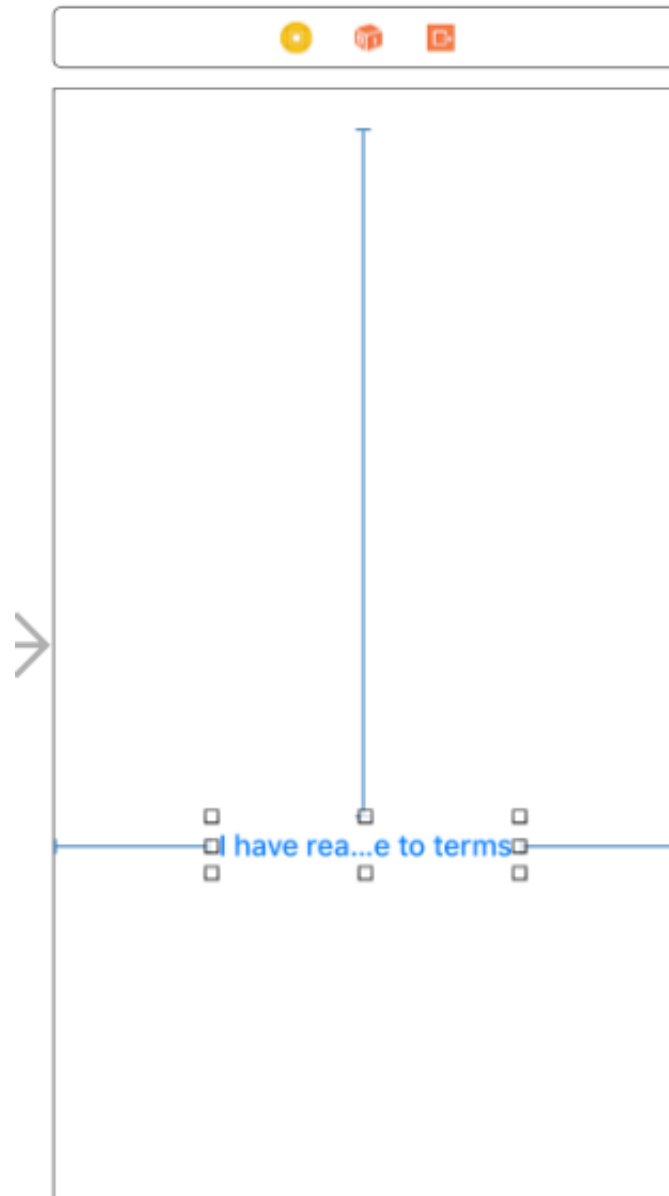
# Fix

To resolve conflicting constraint, click on the red icon with white dot and select the constraint to delete. In this example, we will delete the explicit width 214 constraint so that the width of the button can be dynamic depending on the screen width.



After deleting the conflicting constraint, Xcode doesn't confuse anymore and show us blue lines on the iPhone SE screen (and other screen size too) , yay!





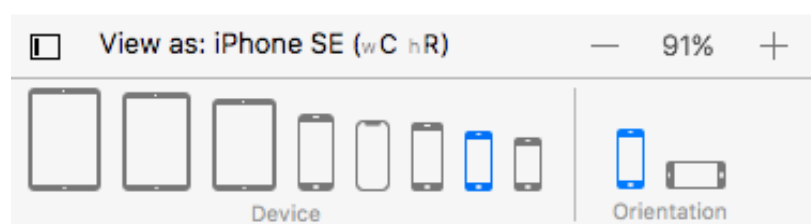
Ahh the text still got squeezed because the width from calculation is too short, we need to reduce the distance from screen left and screen right. This is straightforward so I won't further explain it here.

# Summary

Conflicting constraints usually happen when you have defined too many constraint on an UI element causing Auto Layout Engine unable to satisfy all of them.

I noticed quite some beginner iOS Developer is prone to making conflicting constraints because they usually design UI for only one device type (usually the iPhone model they are using) in the Interface Builder during development. They might unconsciously place excess constraints and Xcode shows blue lines on the device they are designed to. But when they have finished developing the app and release it to friends or beta tester, the layout problem arise as different screen size is used.

To prevent accidentally placing conflicting constraint, be sure to switch between different iPhone model when placing a new UI element / deleting an existing UI element in the Interface Builder.



In the next post, we will look into what is intrinsic content size and the usefulness of it.