GENERAL ASSEMBLY

SCROLL VIEWS

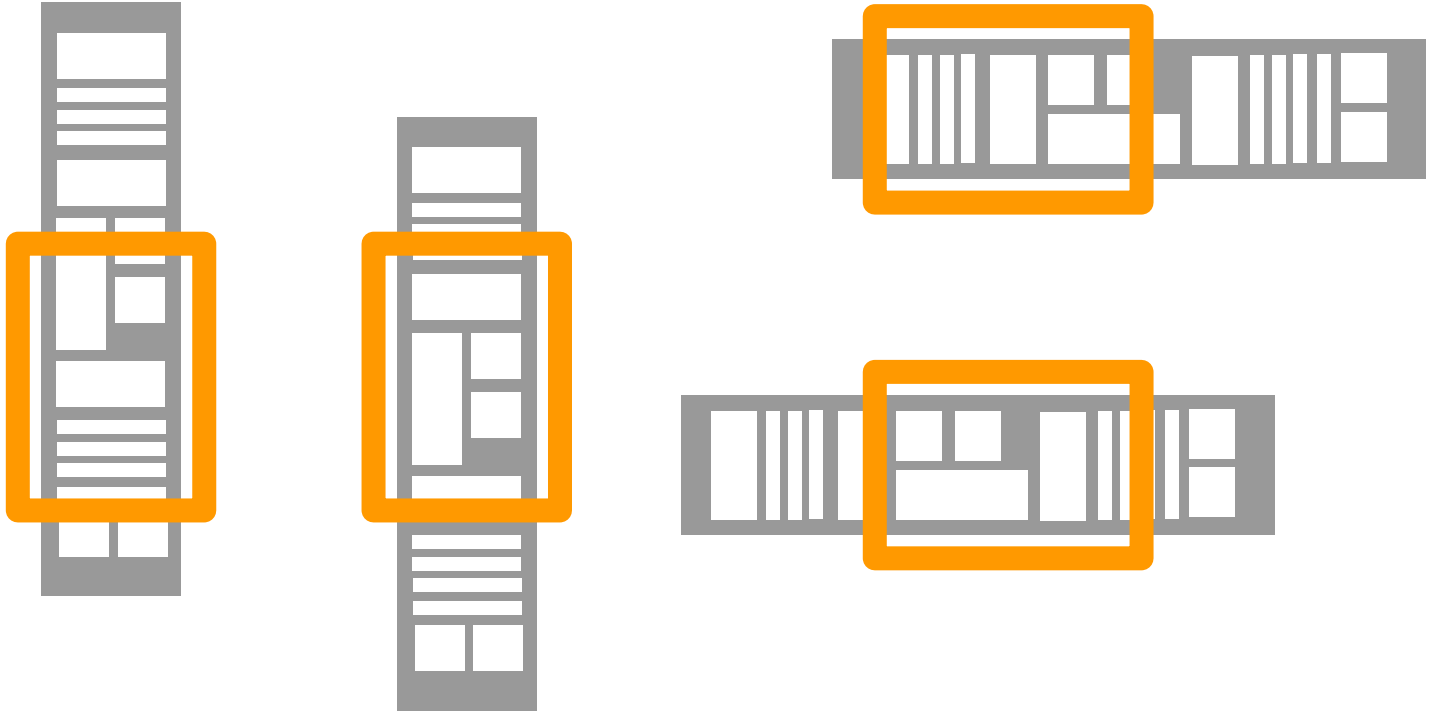
Zeke Abuhoff

Lead iOS Instructor, General Assembly

LEARNING OBJECTIVES

+ Create a scroll view
+ Configure Auto Layout constraints in a scroll view
+ Access a scroll view's content offset in code

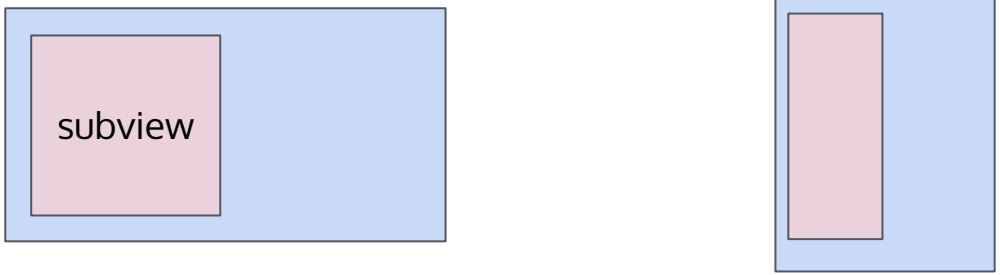
SCROLLING



SUBVIEWS

Any view can have **subviews**.

Subviews reside within their superview, adapting their location on the screen to wherever their superview is. If Auto Layout changes the superview's shape, the subviews change according to their own constraints.



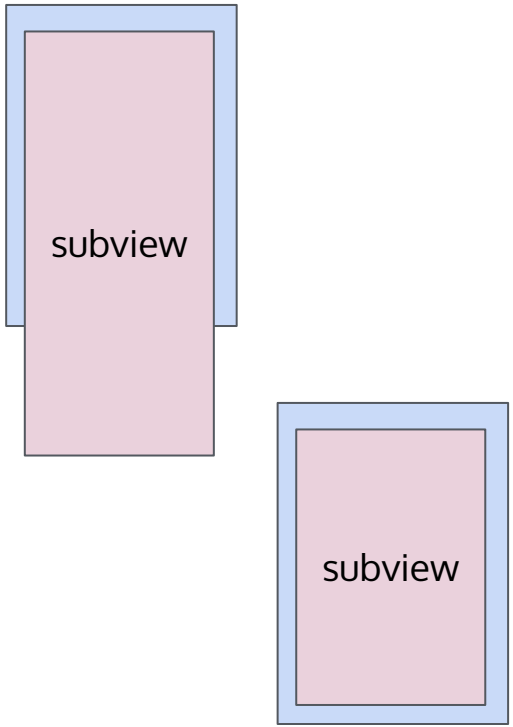
SCROLL VIEWS

A scroll view's subviews act as its scrollable content.

For there to be room to scroll, this content must be larger than the scroll view's own dimensions.

To stay oriented, the scrollable content must also be bound to the dimensions of the scroll view.

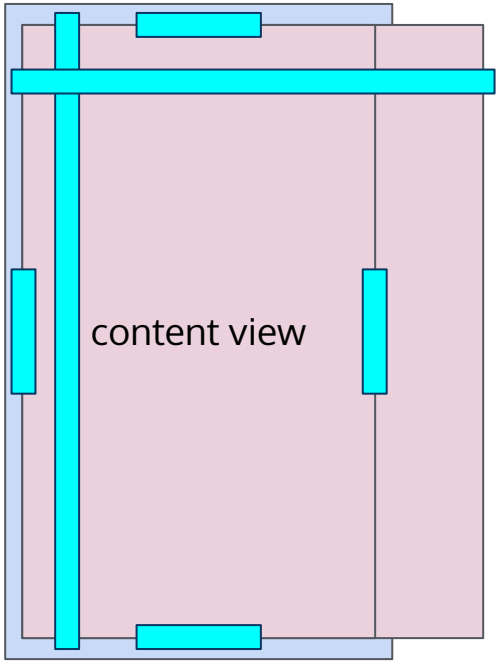
Compared to how Auto Layout normally works, this combination is paradoxical, but for scroll views it's the arrangement that Auto Layout expects.



SCROLL VIEWS

The most straightforward way to configure Auto Layout within a scroll view is to have one view that will be the super view for all other content.

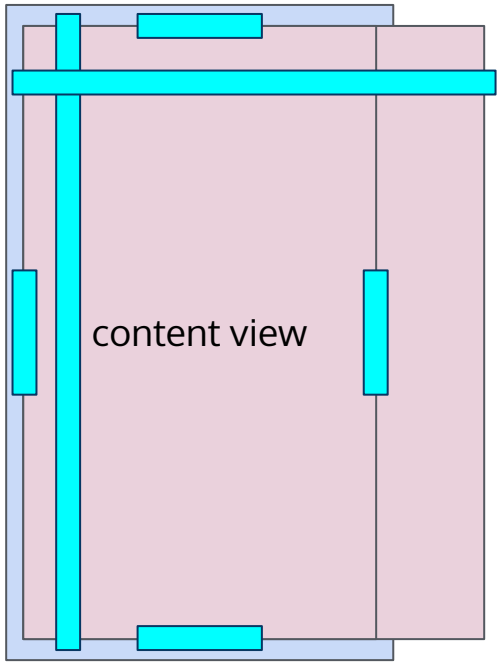
This content view should have constraints specifying 0 pts between it and its container on every side, as well as height and width constraints that specify dimensions greater than the scroll view.



SCROLL VIEWS

Practice:

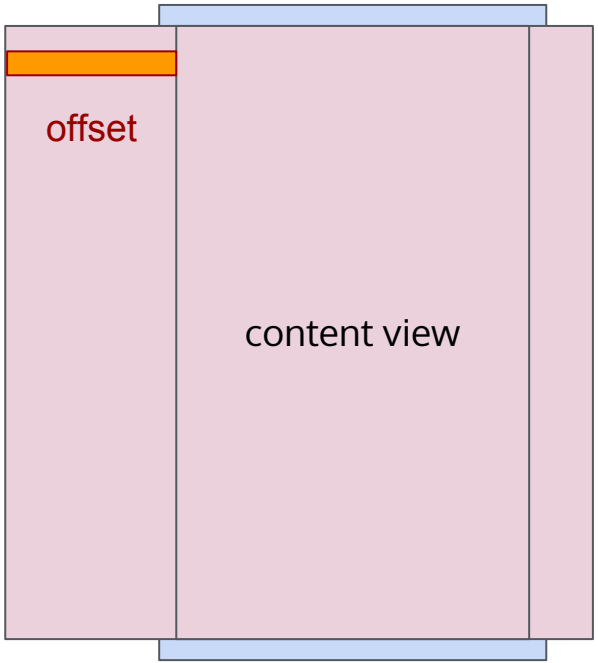
1) Create a scroll view whose content view is twice as wide as its container.
2) Create a scroll view whose content view is 150 pts taller than its container.
3) Create a scroll view whose content view is 50 percent bigger overall than its container.



SCROLL VIEW OFFSET

The **content offset** is a property of scroll views that denotes how far the content is from the edge of the scroll view.

By making an outlet for your scroll view, you can read this property (to get information about where the user has scrolled) or write it (to move the scroll somewhere you want the user to see).




```
myScrollView.contentOffset = CGPoint(x:10.0, y:20.0)
```

SCROLL VIEW DELEGATE

Scroll views use the delegate pattern.

To get more precise control over your scroll view, make a class that conforms to UIScrollViewDelegate and set the scroll view's delegate property to an instance of that class.

```
func scrollViewDidScroll(_ scrollView: UIScrollView) {}
func scrollViewDidZoom(_ scrollView: UIScrollView) {}
// etc...
```

SCROLL VIEW DELEGATE

Practice:

1) Create a new view controller with a new scroll view in it.
2) Create an outlet for that scroll view in the view controller code.
3) Set the view controller as the scroll view's delegate.
4) Use the scroll view's didScroll method to print out the new content offset whenever the user scrolls.

Bonus:

1) Add two views to the scroll view's content. Use Auto Layout constraints to make one stay where it is and one follow the user during scrolling.