Lecture 6

Autolayout

Attendance

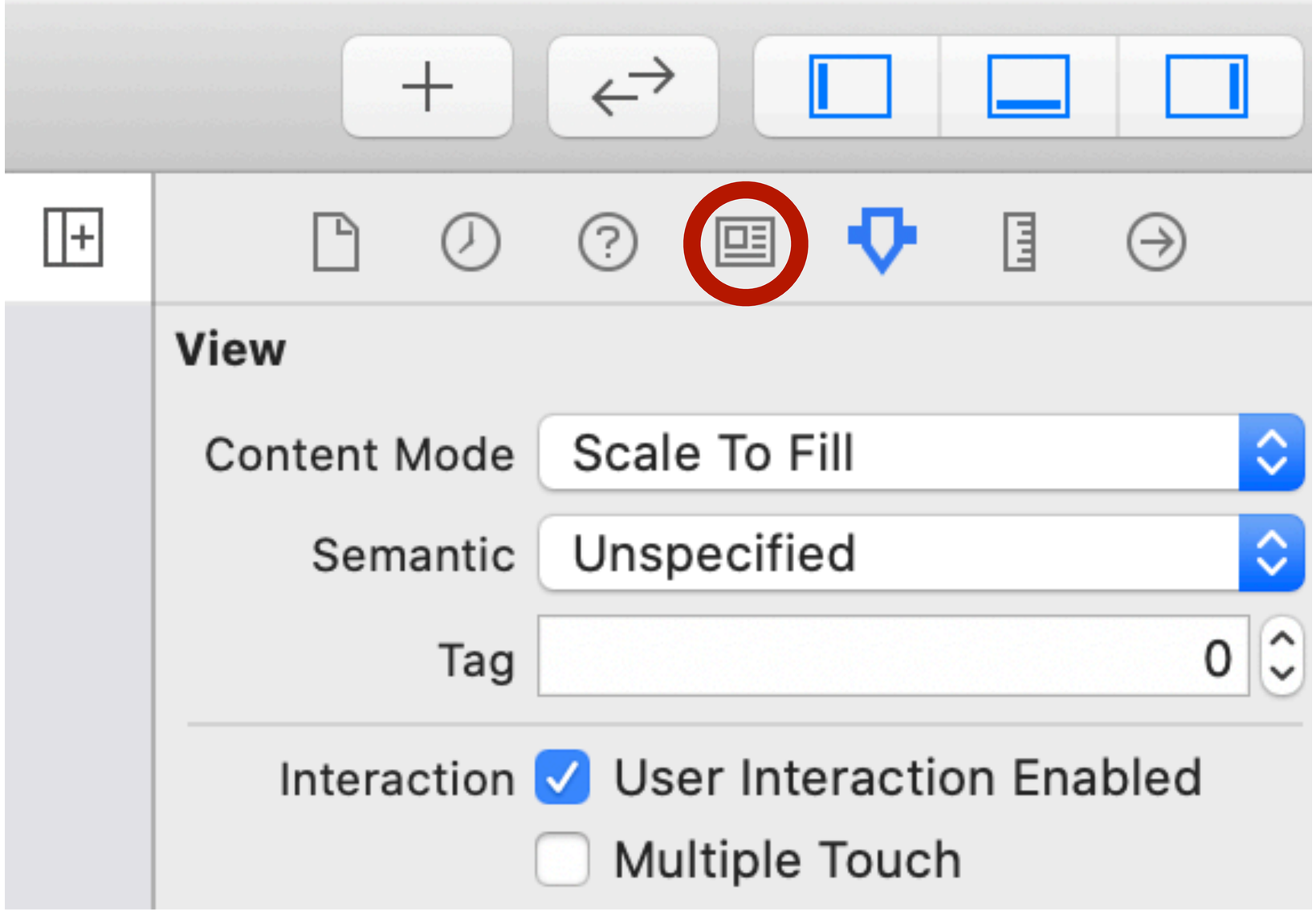
tiny.cc/cis195-lec6

Today

- Swift Topic: Closures { }
- Layout
  - Autolayout and Autoresizing
- Live Demo

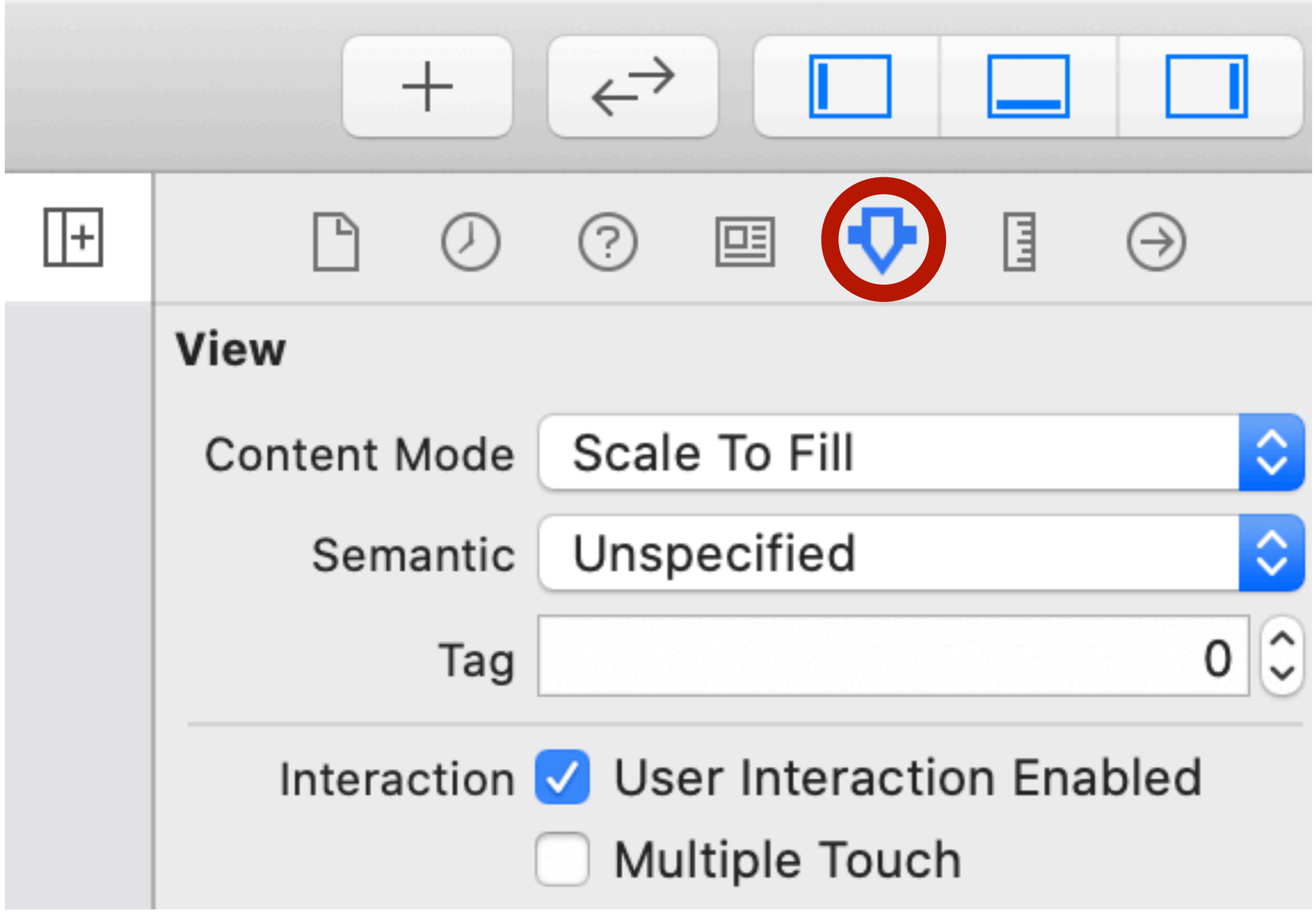
Identity Inspector

Used to set custom classes



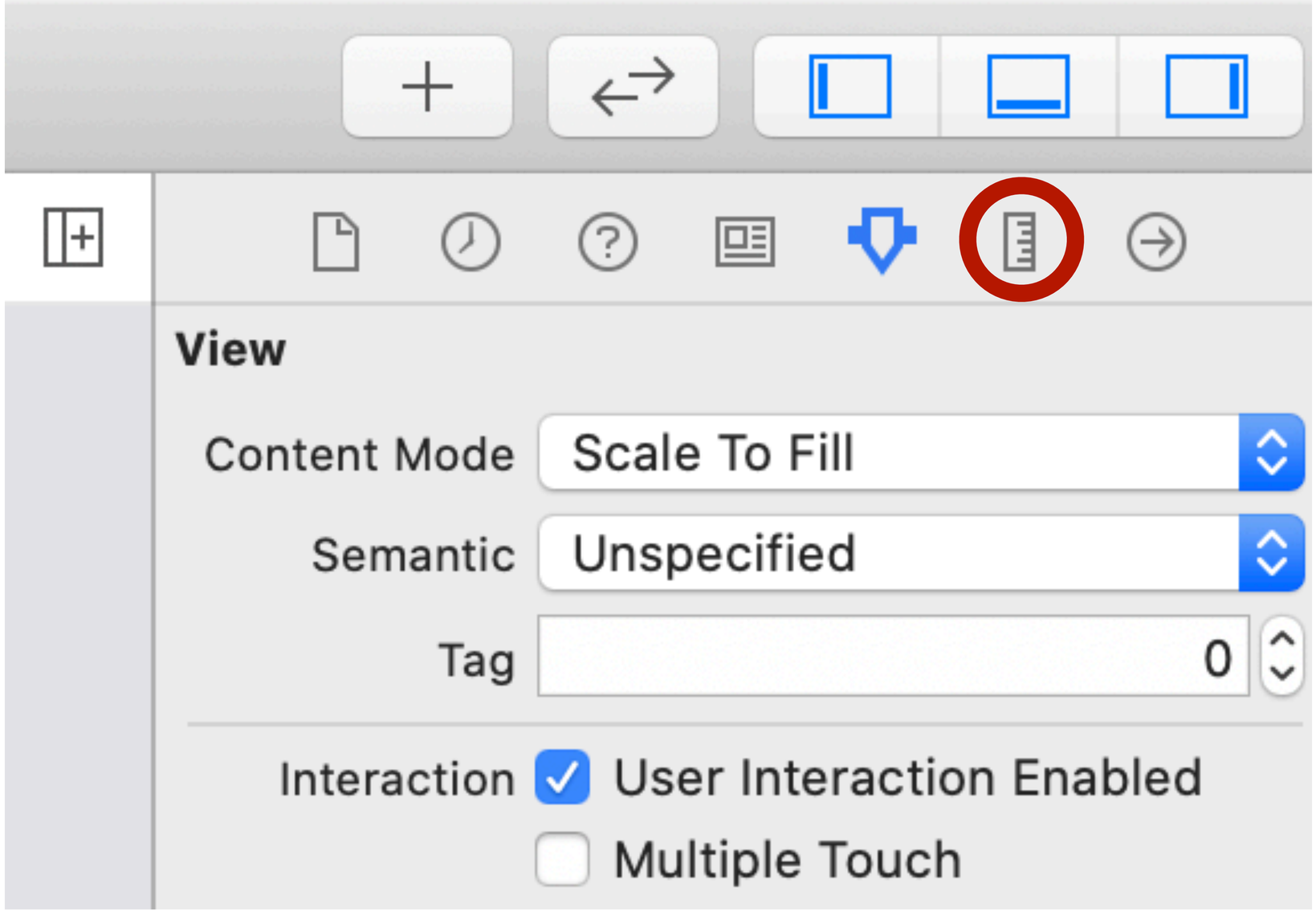
Attribute Inspector

Used to change View properties



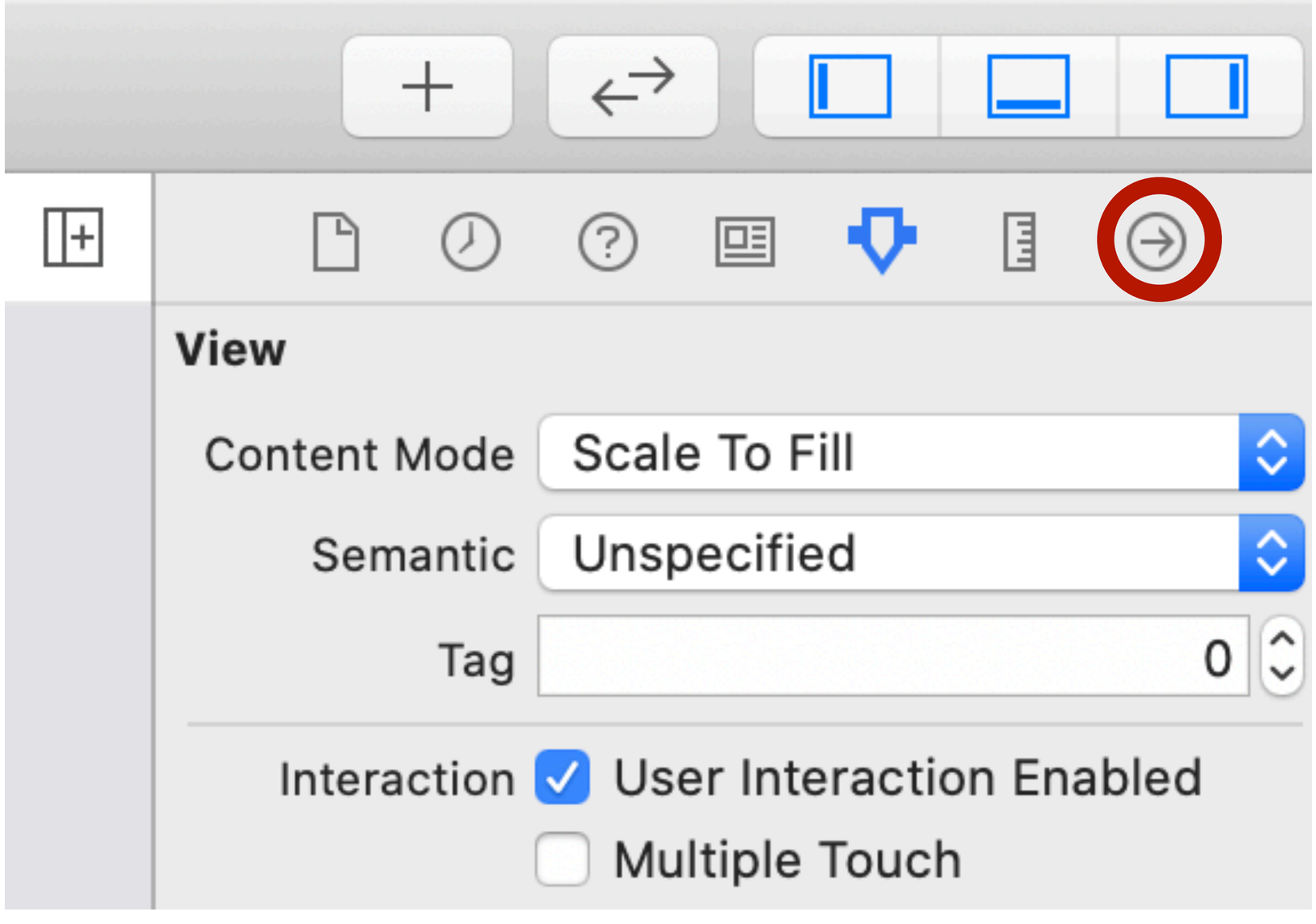
Size Inspector

Used to size, position, and constrain



Connections Inspector

Used to control IBOutlets and IBActions



Closures

Consult the docs

- https://docs.swift.org/swift-book/LanguageGuide/Closures.html
- *“Closures are self-contained blocks of functionality that can be passed around and used in your code.*

  Closures in Swift are similar to blocks in C and Objective-C and to lambdas in other programming languages.”

```swift
let names = ["Chris", "Alex", "Ewa", "Barry", "Daniella"]

func backward(_ s1: String, _ s2: String) -> Bool {
    return s1 > s2
}
var reversedNames = names.sorted(by: backward)
```

```
func sorted(by: (Element, Element) -> Bool) -> [Element]
```

Returns the elements of the sequence, sorted using the given predicate as the comparison between elements.

```
sorted(by: (Element, Element) -> Bool) -> [Element]

sort(by: (Element, Element) -> Bool)
```

```
sorted(by: (Element, Element) -> Bool) -> [Element]

sort(by: (Element, Element) -> Bool)
```

```swift
func backward(_ s1: String, _ s2: String) -> Bool {
    return s1 > s2
}
var reversedNames = names.sorted(by: backward)
```

```
sorted(by: (Element, Element) -> Bool) -> [Element]
```

```swift
func backward(_ s1: String, _ s2: String) -> Bool {
    return s1 > s2
}
var reversedNames = names.sorted(by: backward)
```

```
sorted(by: (Element, Element) -> Bool) -> [Element]
```

What if we don’t want to define a new function every time?

```
sorted(by: (Element, Element) -> Bool) -> [Element]
```

```
reversedNames = names.sorted(by: { (s1: String, s2: String) -> Bool in
    return s1 > s2
})
```

```
sorted(by: (Element, Element) -> Bool) -> [Element]
```

```
reversedNames = names.sorted(by: { (s1: String, s2: String) -> Bool in
    return s1 > s2
})
```

How can we improve this?

```
sorted(by: (Element, Element) -> Bool) -> [Element]
```

```
reversedNames = names.sorted(by: { (s1: String, s2: String) -> Bool in
    return s1 > s2
})
```

How can we improve this? Type Inference!

```
reversedNames = names.sorted(by: { s1, s2 in return s1 > s2 } )
```

```
sorted(by: (Element, Element) -> Bool) -> [Element]
```

Implicit returns from single-line closures

```
reversedNames = names.sorted(by: { s1, s2 in s1 > s2 } )
```

```
sorted(by: (Element, Element) -> Bool) -> [Element]
```

Implicit returns from single-line closures

```
reversedNames = names.sorted(by: { s1, s2 in s1 > s2 } )
```

Shorthand argument names ($0 = arg0)

```
reversedNames = names.sorted(by: { $0 > $1 } )
```

```
sorted(by: (Element, Element) -> Bool) -> [Element]
```

Implicit returns from single-line closures

```
reversedNames = names.sorted(by: { s1, s2 in s1 > s2 } )
```

Shorthand argument names ($0 = arg0)

```
reversedNames = names.sorted(by: { $0 > $1 } )
```

Operator function, b.c. String conforms to Comparable!

```
reversedNames = names.sorted(by: >)
```

```swift
func someFunctionThatTakesAClosure(closure: () -> Void) {
    // function body goes here
}

// Here's how you call this function without using a trailing closure:

someFunctionThatTakesAClosure(closure: {
    // closure's body goes here
})

// Here's how you call this function with a trailing closure instead:

someFunctionThatTakesAClosure() {
    // trailing closure's body goes here
}
```

```
let submitAction = UIAlertAction(title: "Submit", style: .default) { [unowned alert] _ in

        let answer = alert.textFields![0]

}
```

More about Closures

https://docs.swift.org/swift-book/LanguageGuide/Closures.html

Motivation

2:33

3 1 11

Who to follow

Antoine v.d. SwiftLee 🚀 and 2 others follow

John Sundell
@johnsundell
Follow

Creator of @swiftbysundell, co-host of @stacktracepod, Swift developer, avid prototyper and amateur chef 👨‍💻👨‍🍳👨‍🍳
Contact me: swiftbysundell.com/contact

Antoine v.d. SwiftLee 🚀 and 2 others follow

Sean Allen
@seanallen_dev
Follow

iOS Engineer and content creator: youtube.com/seanallen. Basketball Junkie. Star Wars. Game of Thrones.

John Gruber and 2 others follow

Daniel Dale
@ddale8
Follow

Reporter for CNN, fact-checking the president and other politicians.

Show more

Deque Systems @dequesystems
Our developers created axe Pro to h
dev teams get more from accessib
testing without needing to become
accessibility experts

2:33

Lists

CIS 195

- [ ] Revise lecture 5
- [ ] Revise lecture 6
- [ ] Revise lecture 7
- [ ] Revise lecture 8
- [ ] Revise lecture 9
- [ ] Revise lecture 10
- [ ] Tutorial 4
- [ ] Tutorial 5
- [ ] Tutorial 6
- [ ] Tutorial 7
- [ ] Tutorial 8
- [ ] App 2b
- [ ] App 3
- [ ] App 4

New Reminder

2:32
Search

Instagram

hhaurora
Uptown, Minnea
New Posts

38 likes

hhaurora i just ordered a Ricoh RZ1050 film camera and two disposables from @filmwholesale for devon... more

Layout Challenges

External Changes

- The user resizes the window (OS X).
- The user enters or leaves Split View on an iPad (iOS).
- The device rotates (iOS).
- The active call and audio recording bars appear or disappear (iOS).
- You want to support different size classes.
- **You want to support different screen sizes.**

Internal Changes

- The content displayed by the app changes.
- The app supports internationalization.
- The app supports Dynamic Type (iOS).

https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/

Expectation of Quality

- iOS users (in general) have a higher expectation of quality, and lower tolerance for broken apps
- Broken layouts = broken apps

  -> low reviews

  -> fewer subscribers

  -> less proud developers

- Take advantage of iOS's small device ecosystem & make something great!

AutoLayout Constraints

Three Ways to Layout Your Interface

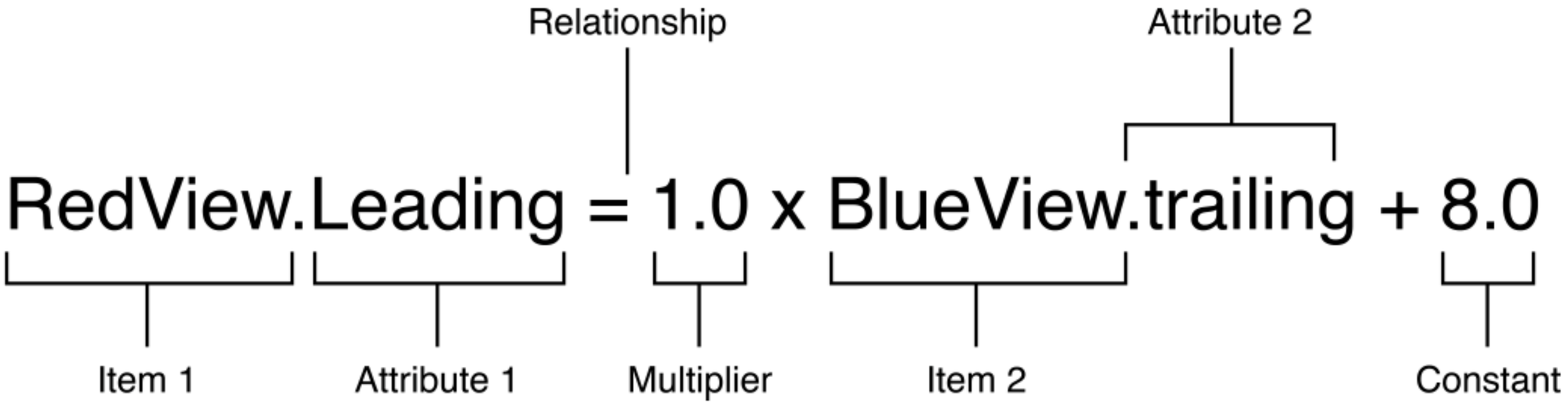
1. Manual (counting pixels)
2. NSConstraint
    - “AutoLayout”
    - Powerful and verbose
3. Autoresizing Masks
    - Less powerful, but easy to use

Three Ways to Layout Your Interface

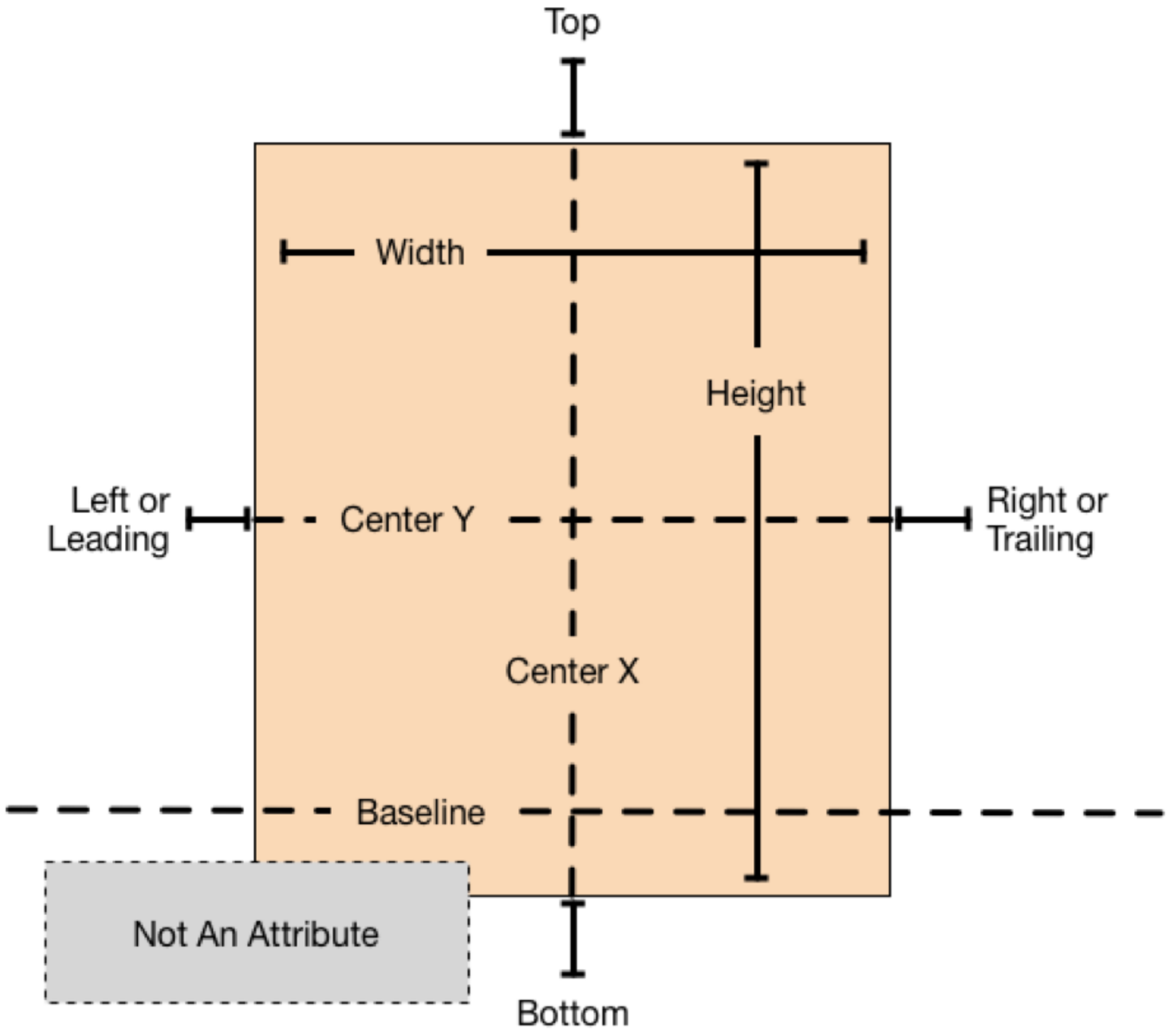
1. Manual (counting pixels)
2. **NSConstraint**
   - "AutoLayout"
   - Powerful and verbose
3. Autoresizing Masks
   - Less powerful, but easy to use

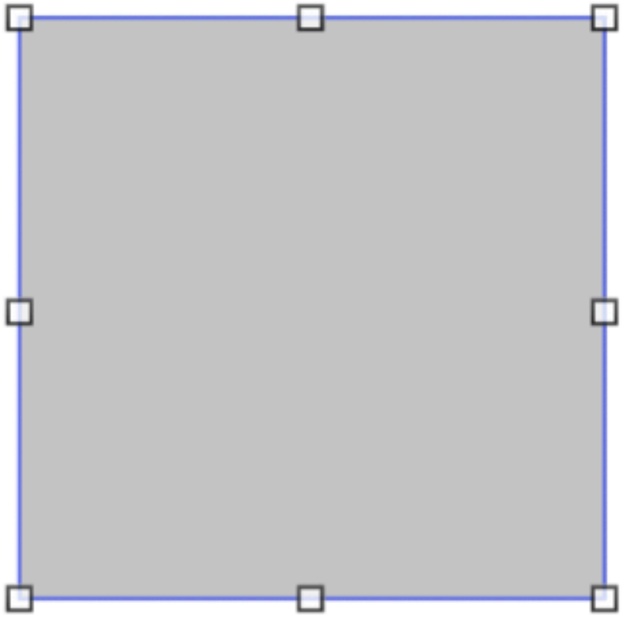
First, we'll discuss NSConstraints (Autolayout)

Anatomy of a Constraint



Attributes


Top
Width
Height
Left or
Leading
Center Y
Right or
Trailing
Center X
Baseline
Not An Attribute
Bottom



Add New Constraints

668

239

46

22

Spacing to nearest neighbor

Constrain to margins

Width 129

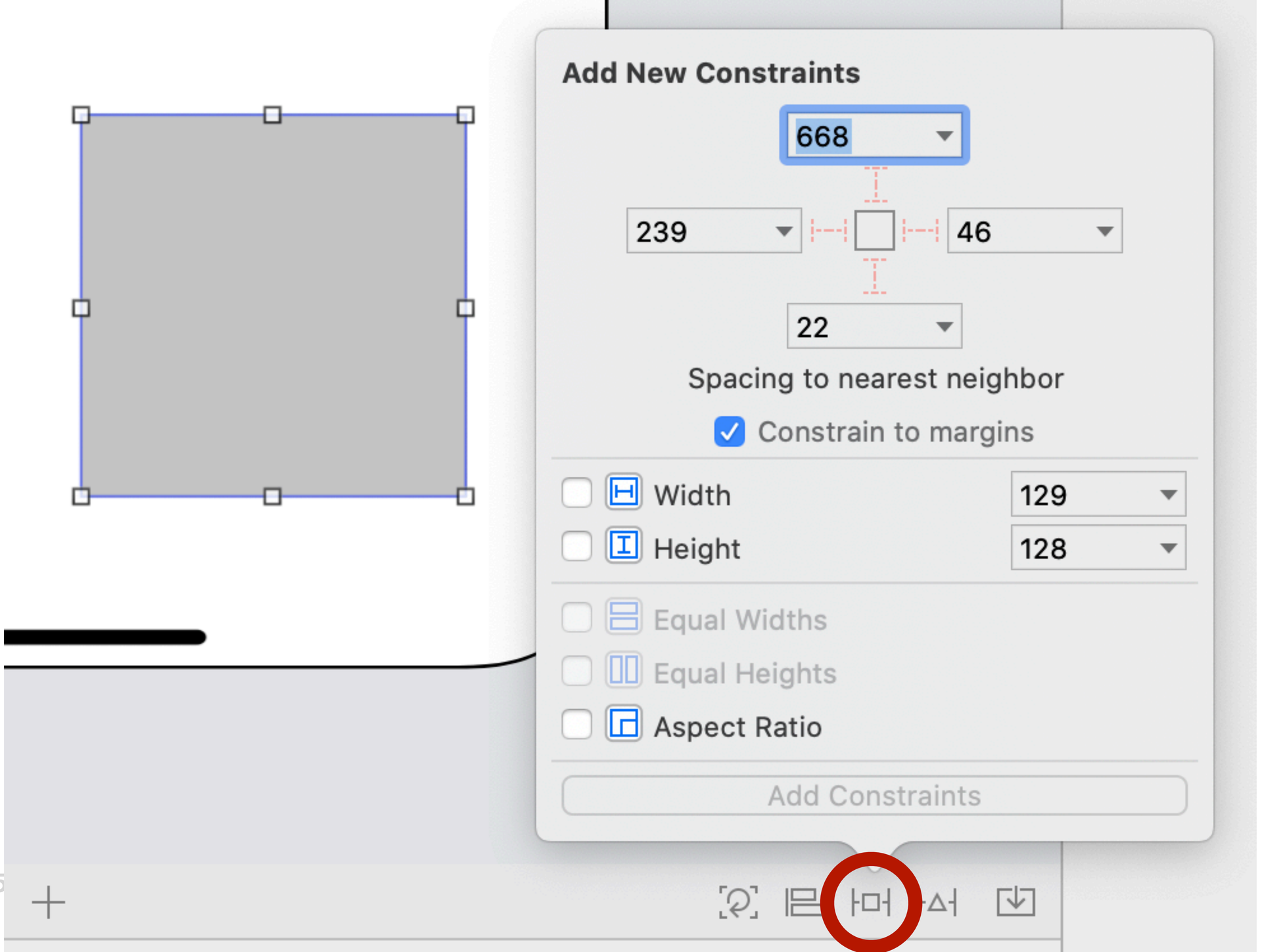
Height 128

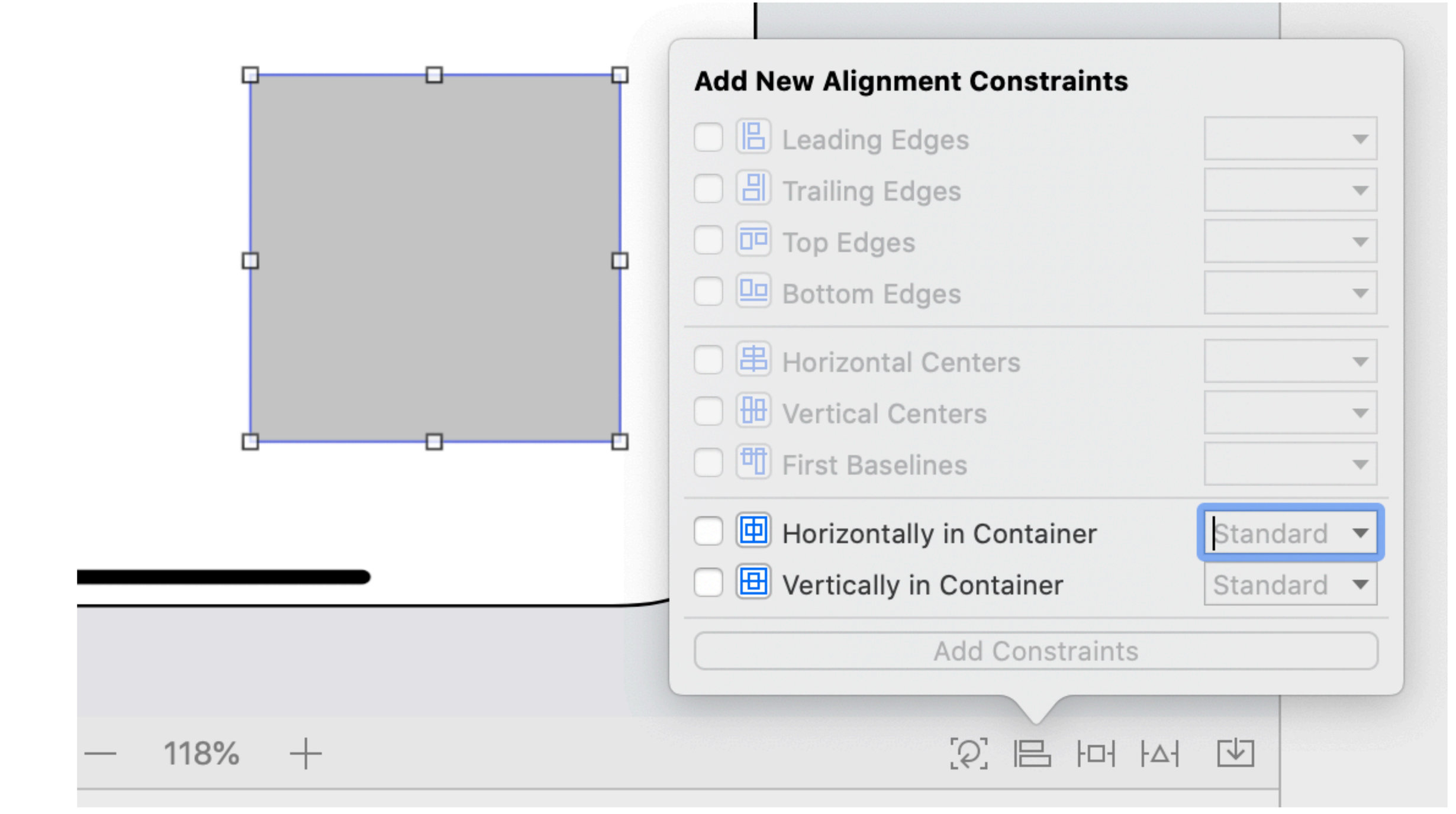
Equal Widths

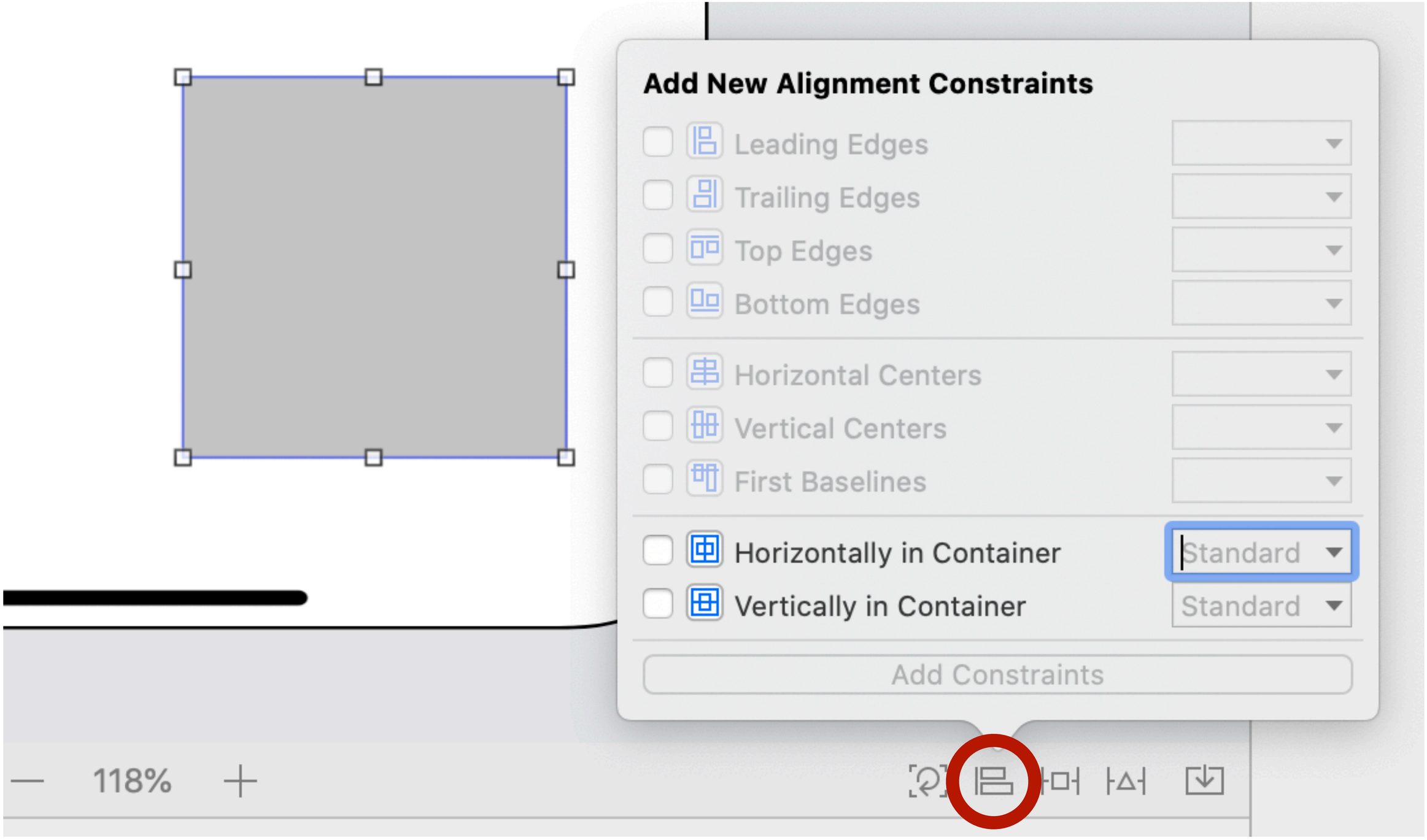
Equal Heights

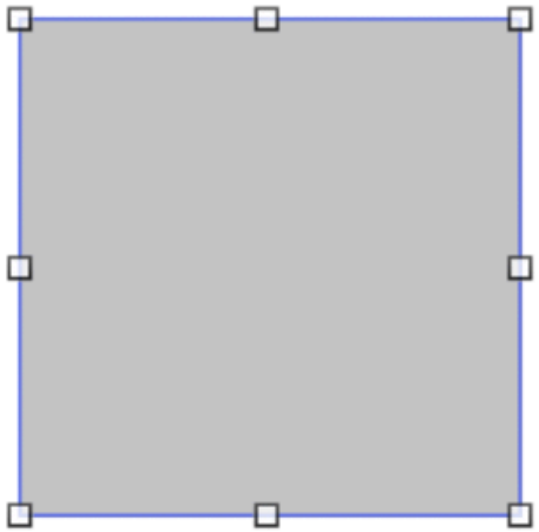
Aspect Ratio

Add Constraints


Add New Constraints
668
239
46
22
Spacing to nearest neighbor
Constrain to margins
Width
129
Height
128
Equal Widths
Equal Heights
Aspect Ratio
Add Constraints


Add New Alignment Constraints
Leading Edges
Trailing Edges
Top Edges
Bottom Edges
Horizontal Centers
Vertical Centers
First Baselines
Horizontally in Container
Standard
Vertically in Container
Standard
Add Constraints
118%


Add New Alignment Constraints
Leading Edges
Trailing Edges
Top Edges
Bottom Edges
Horizontal Centers
Vertical Centers
First Baselines
Horizontally in Container
Standard
Vertically in Container
Standard
Add Constraints
118%



Add New Constraints

668

239 | 46

22

Spacing to nearest neighbor

Constrain to margins

Width 129

Height 128

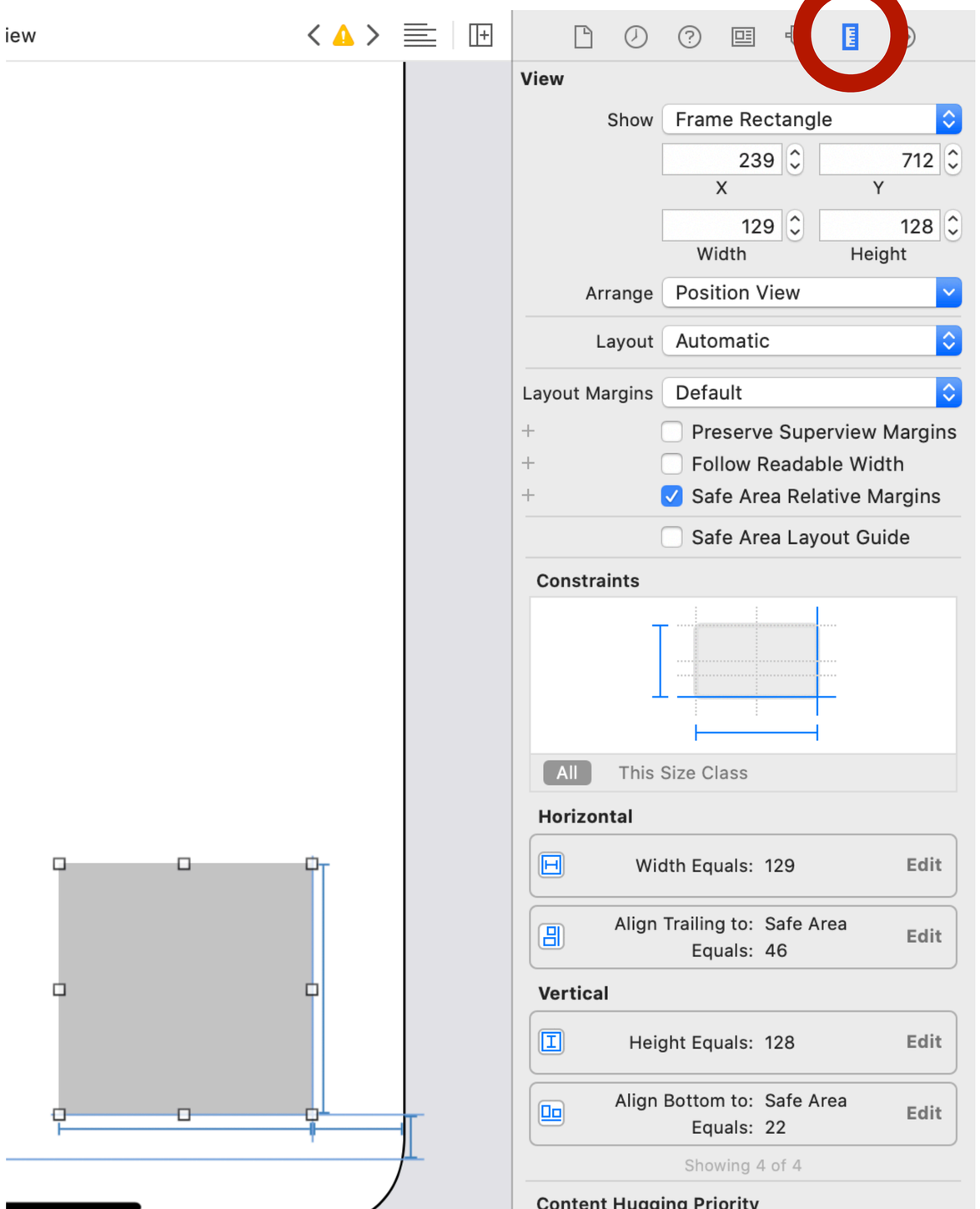
Equal Widths

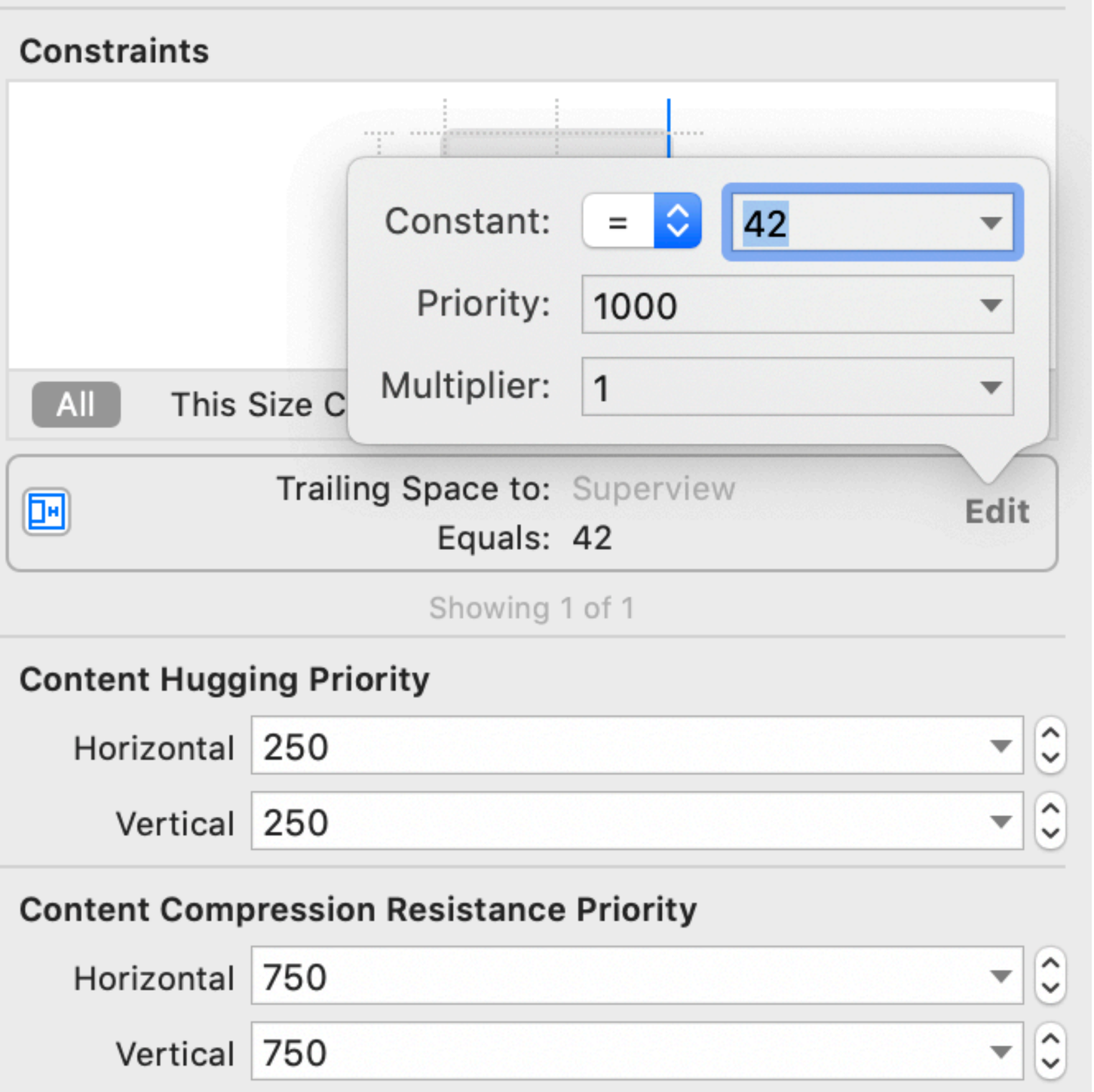
Equal Heights

Aspect Ratio

Add 4 Constraints

10%


iew
View
Show
Frame Rectangle
239
X
712
Y
129
Width
128
Height
Arrange
Position View
Layout
Automatic
Layout Margins
Default
Preserve Superview Margins
Follow Readable Width
Safe Area Relative Margins
Safe Area Layout Guide
Constraints
All
This Size Class
Horizontal
Width Equals: 129
Edit
Align Trailing to: Safe Area
Equals: 46
Edit
Vertical
Height Equals: 128
Edit
Align Bottom to: Safe Area
Equals: 22
Edit
Showing 4 of 4
Content Hugging Priority


Constraints
Constant:
=
42
Priority:
1000
Multiplier:
1
All
This Size C
Trailing Space to: Superview
Equals: 42
Edit
Showing 1 of 1
Content Hugging Priority
Horizontal
250
Vertical
250
Content Compression Resistance Priority
Horizontal
750
Vertical
750

Creating non-ambiguous layouts

- Rule of thumb: we need **2 horizontal** and **2 vertical** constraints per view

Creating non-ambiguous layouts

- Rule of thumb: we need **2 horizontal** and **2 vertical** constraints per view

    - We also want to avoid *over constraining* our system

Creating non-ambiguous layouts

- Rule of thumb: we need **2 horizontal** and **2 vertical** constraints per view
    - We also want to avoid *over constraining* our system
    - **Demo on the board**

Creating non-ambiguous layouts

- Rule of thumb: we need **2 horizontal** and **2 vertical** constraints per view
    - We also want to avoid *over constraining* our system
    - **Demo on the board**
- Exception: some views have *intrinsic content size* and may require fewer. Let's take a look at some examples.

Intrinsic Content Size

- Recall the rule of thumb: we need **2 horizontal** and **2 vertical** constraints per view
- View's with *intrinsic content size* use their content to define one or more dimensions. This means you can fully define them with < 4 constraints (often 3)
    - Put simply, these views are allowed to "Grow" in one or more directions, according to their content
- Examples: UILabel, UIButton, UIStackView

UILabel Intrinsic Content Size

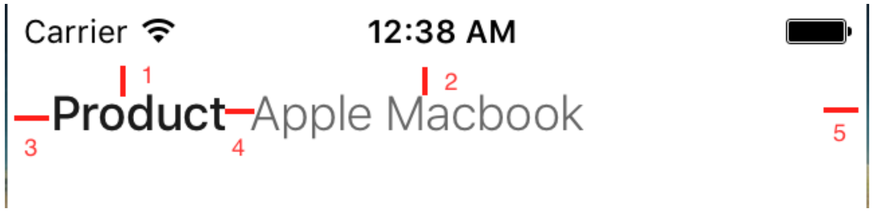
- Recall the rule of thumb: we need **2 horizontal** and **2 vertical** constraints per view

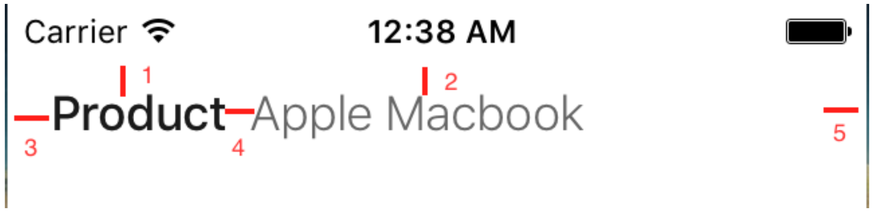
- If the text is left-to-right, we can allow the label to define it's own *width and height* according to its content

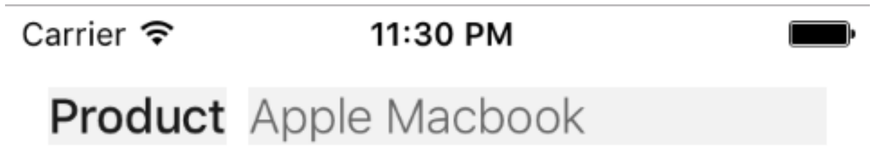
    - We can add an explicit width constraint — the label will then *grow vertically* with it's content

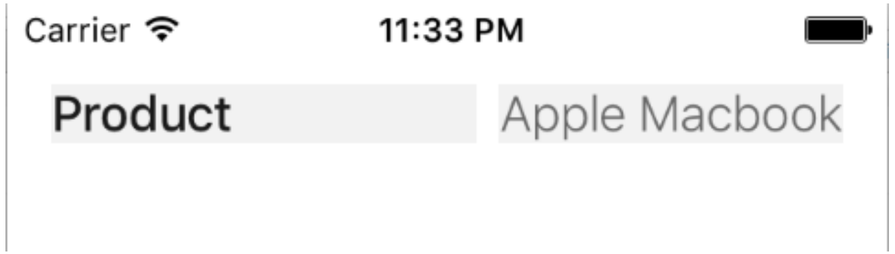
Content Hugging and Compression Resistance

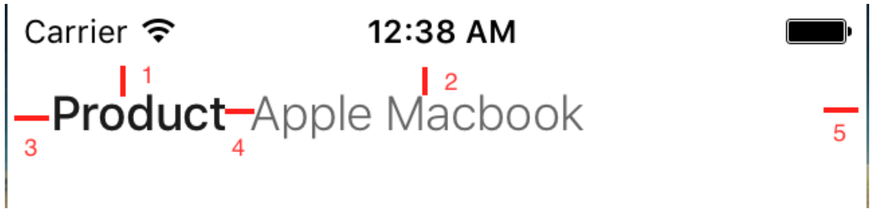
- Don’t worry about understanding this fully. It is an advanced feature of AutoLayout.
    - You should just know it exists so you can solve errors.

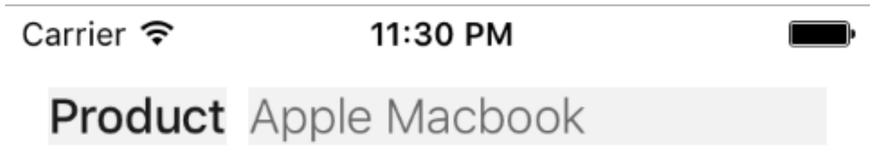

Carrier
12:38 AM
1
2
Product
Apple Macbook
3
4
5

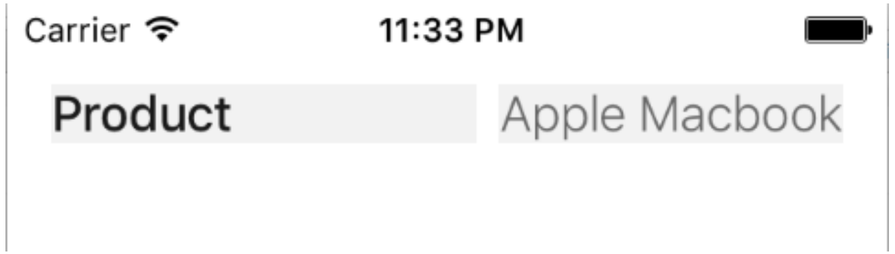

Carrier
12:38 AM
1
2
Product
Apple Macbook
3
4
5

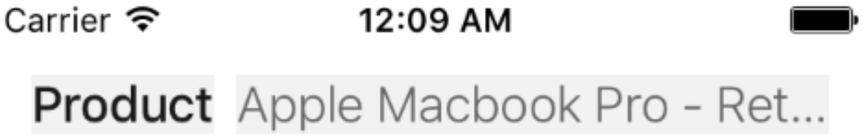

Carrier
11:30 PM
Product
Apple Macbook

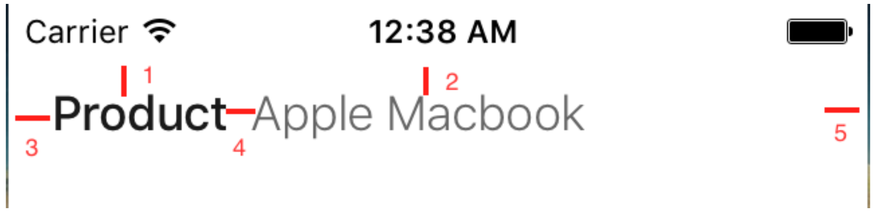

Carrier
11:33 PM
Product
Apple Macbook

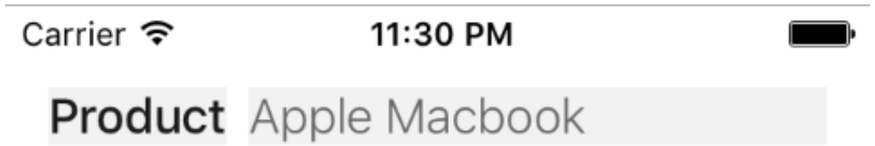

Carrier
12:38 AM
1
2
Product
Apple Macbook
3
4
5

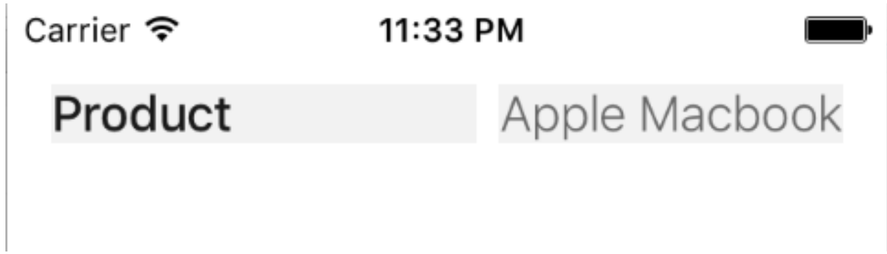

Carrier
11:30 PM
Product
Apple Macbook

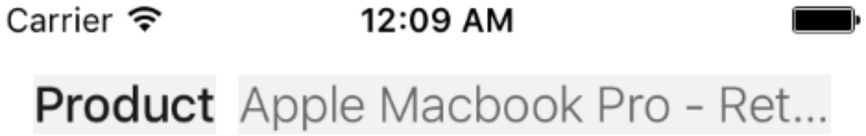

Carrier
11:33 PM
Product
Apple Macbook


Carrier
12:09 AM
Product
Apple Macbook Pro - Ret...



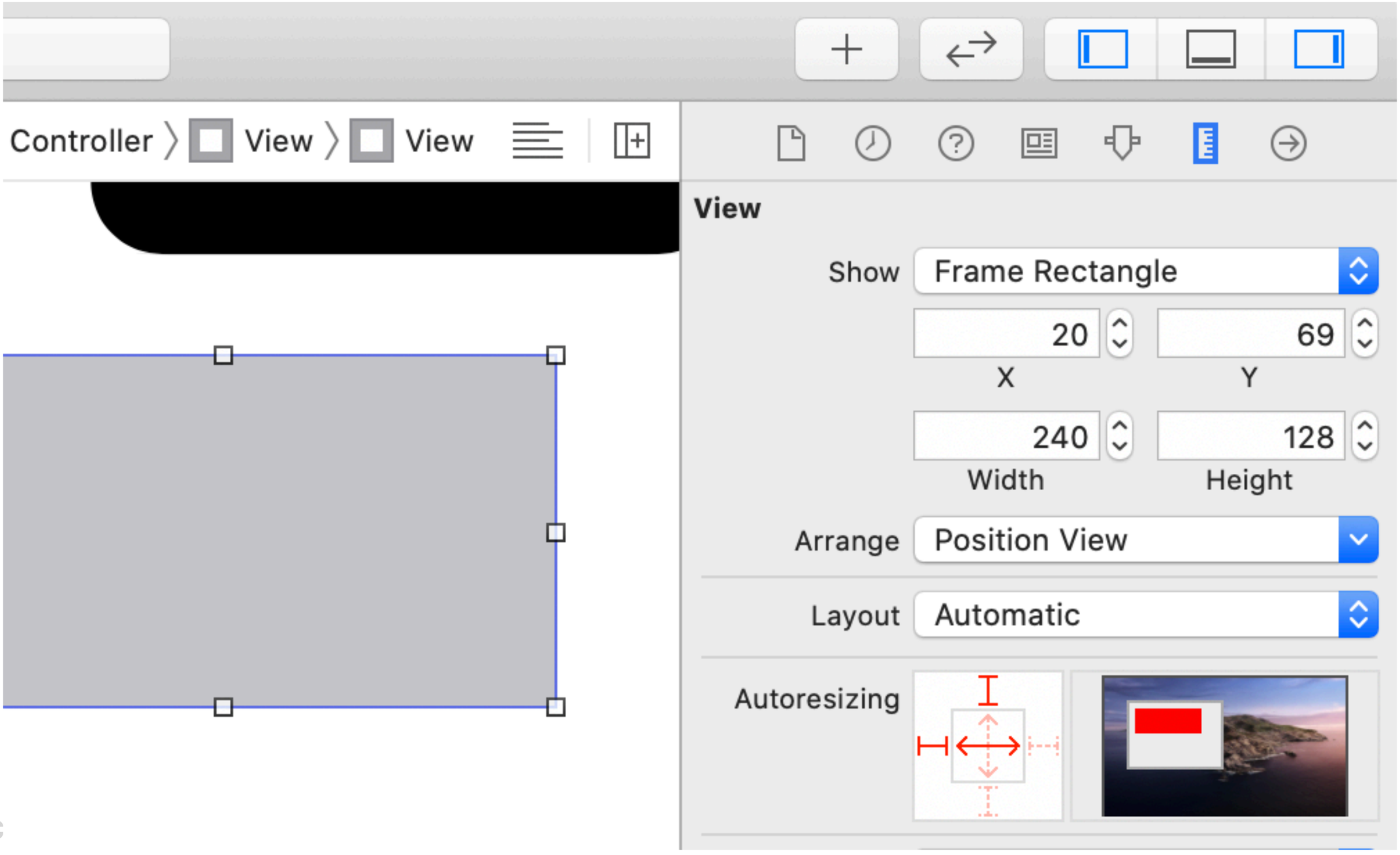






Content Hugging and Compression Resistance

https://medium.com/@dineshk1389/content-hugging-and-compression-resistance-in-ios-35a0e8f19118


Controller
View
View
View
Show
Frame Rectangle
20
X
69
Y
240
Width
128
Height
Arrange
Position View
Layout
Automatic
Autoresizing

Live Demo: Constraints

Due Before Next Class

- App 4: Profile
- Tutorial 4: Segues

Links

- Survey: tiny.cc/cis195-lec6
- Piazza: tiny.cc/cis195-piazza