Mobile Engineering

Day 03

AutoLayout

Controls, Controls, Controls

Tom Crawford

moveablebytes.com

@movebytes @thcrawford

Introductions

Introductions

If you could **visit** anywhere in the world, where would it be? (you have all the time and money you need to make it happen)

Upcoming Meetups

Upcoming Meetups

- DC Thu 10/1 - Cocoaheads
- DC Wed 10/14 - MDC User Acquisition
- DC Tue 10/20 - DCTech Demos
- DC Wed 10/21 - MDC Mixer
- DC Mon 10/26 - UXDC
- DC Thu 11/5 - Cocoaheads
- DC Tue 11/17 - DCTech Demos
- DC Wed 11/18 - MDC Mixer
- DC Thu 12/3 - Cocoaheads
- DC Wed 12/16 - MDC Mixer

Quiz

Homework Review

Properties & Methods

Declaring Properties

Public (.h) or Private (.m)?

Syntax

```
@property (nonatomic,weak) IBOutlet UILabel *nameLabel;
```

Auto-synthesized in Modern Objective C!

Declares property

Memory Management

Nullability

Declaration Object Name

Creates ivar & getter/setter (or just getter if readonly)

Not everything needs a property

(in Modern Objective C)

…only when you need to access it by name

Declaring Methods

Public (.h) or Private (.m)?

Syntax

```
- (void)doSomethingCool;
```

Object to Return

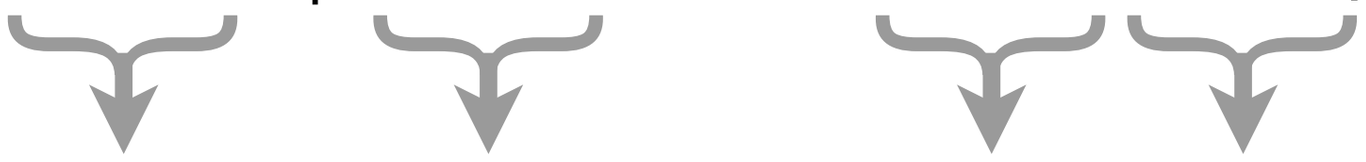
Method Name

Declaring Methods

Public (.h) or Private (.m)?

Syntax

```
- (void)doSomethingCool;
- (float)squareWithNumber:(float)number;
```



(Internal to Method)

Declaring Methods

Public (.h) or Private (.m)?

Syntax

```
- (void)doSomethingCool;
- (float)squareWithNumber:(float)number;
- (IBAction)myButtonPressed:(id)sender;
```

Available to Interface Builder

Method Name

Untyped Object Parameter

Declaring Methods

Public (.h) or Private (.m)?

Syntax

```
- (void)doSomethingCool;
- (float)squareWithNumber:(float)number;
- (IBAction)myButtonPressed:(id)sender;
- (IBAction)myButtonPressed:(UIButton *)button;
```

Specific Object Parameter

Doesn't need to be declared if not public

```
- (void)doSomethingCool {
    NSLog(@"Something Cool Goes Here");
}

- (float)squareWithNumber:(float)number {
    return (number * number);
}
```

Calling Methods & Properties

Reference auto-synthesized properties:

```
_myLabel.text = @"My Label";
[_myLabel setText:@"My Label"];
NSLog(@"Label Text:%@",_myLabel.text);
NSLog(@"Label Text:%@",[_myLabel text]);
```

Reference manually synthesized properties

(self is the current view controller)

```
self.myLabel.text = @"My Label";
[self.myLabel setText:@"My Label"];
NSLog(@"Label Text:%@",self.myLabel.text);
NSLog(@"Label Text:%@",[self.myLabel text]);
```

Life Cycle Methods are called automatically

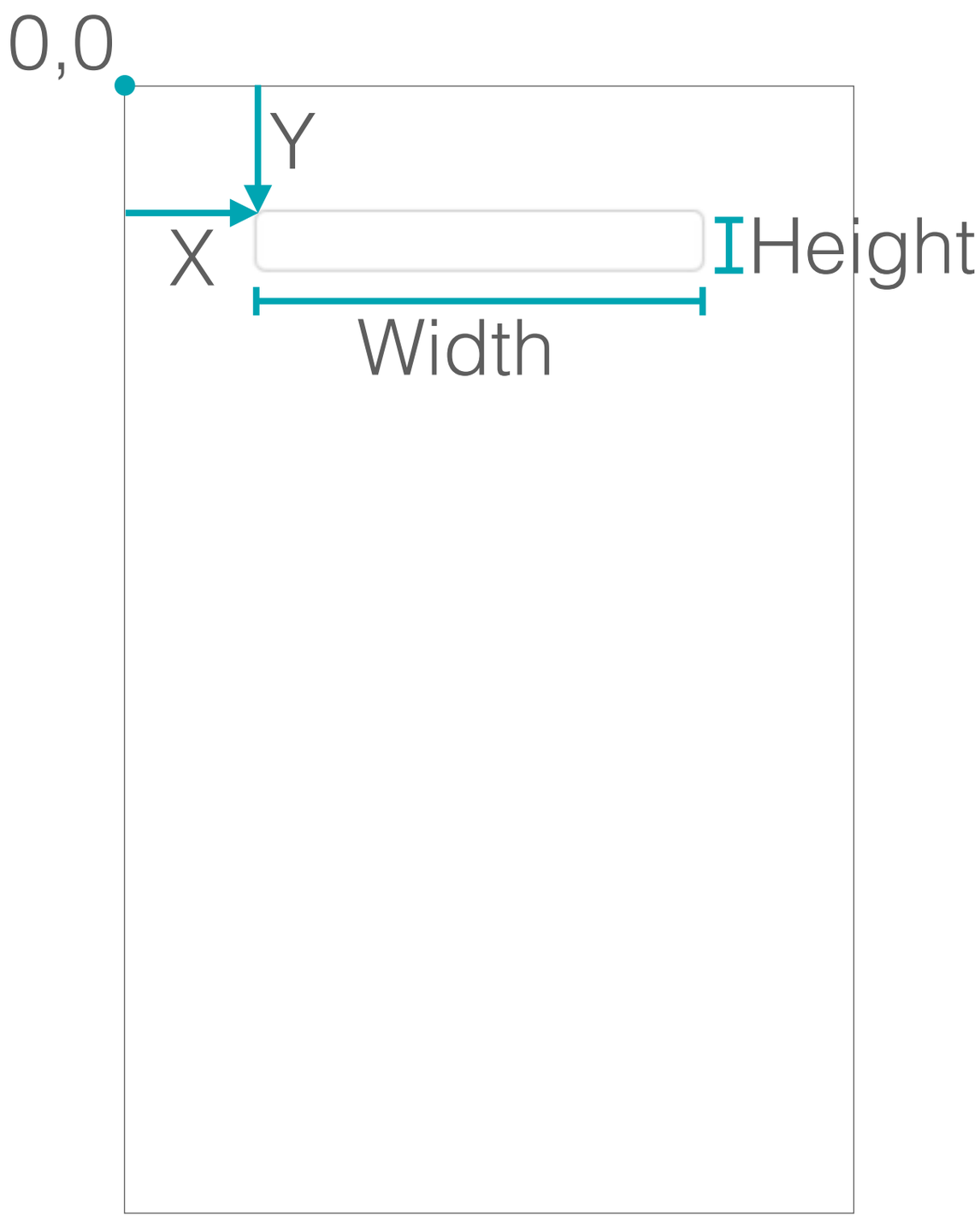
Custom methods need to be called:

```
[self doSomethingCool];
float squaredNumber = [self squareWithNumber:12.0];
[self myButtonPressed:self]; // This COULD be bad
[self myButtonPressed: myButton];
```

AutoLayout

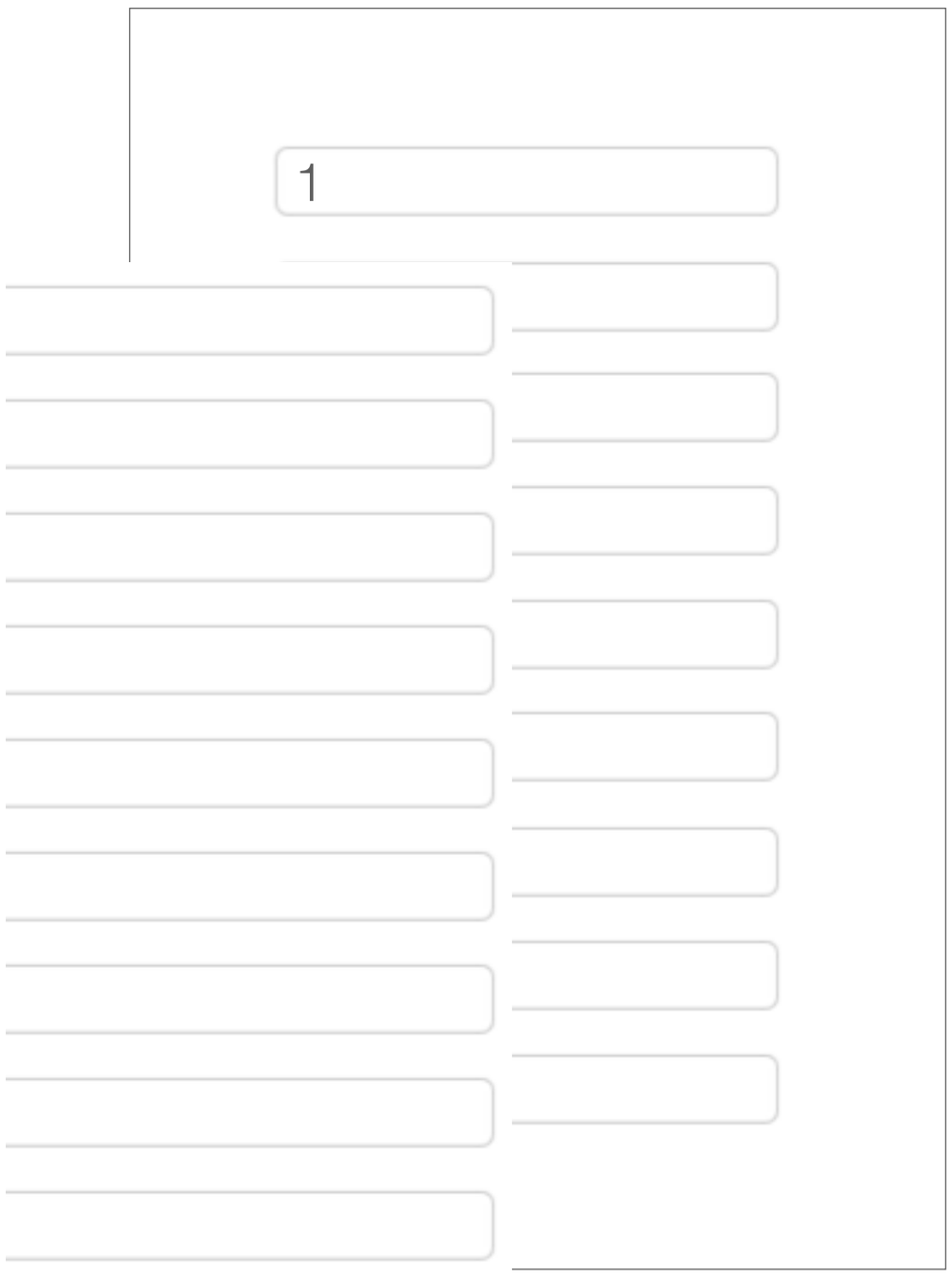
Layout Basics

Position & size UI elements on the screen



Handling Size Changes

But things change…first, it was rotation

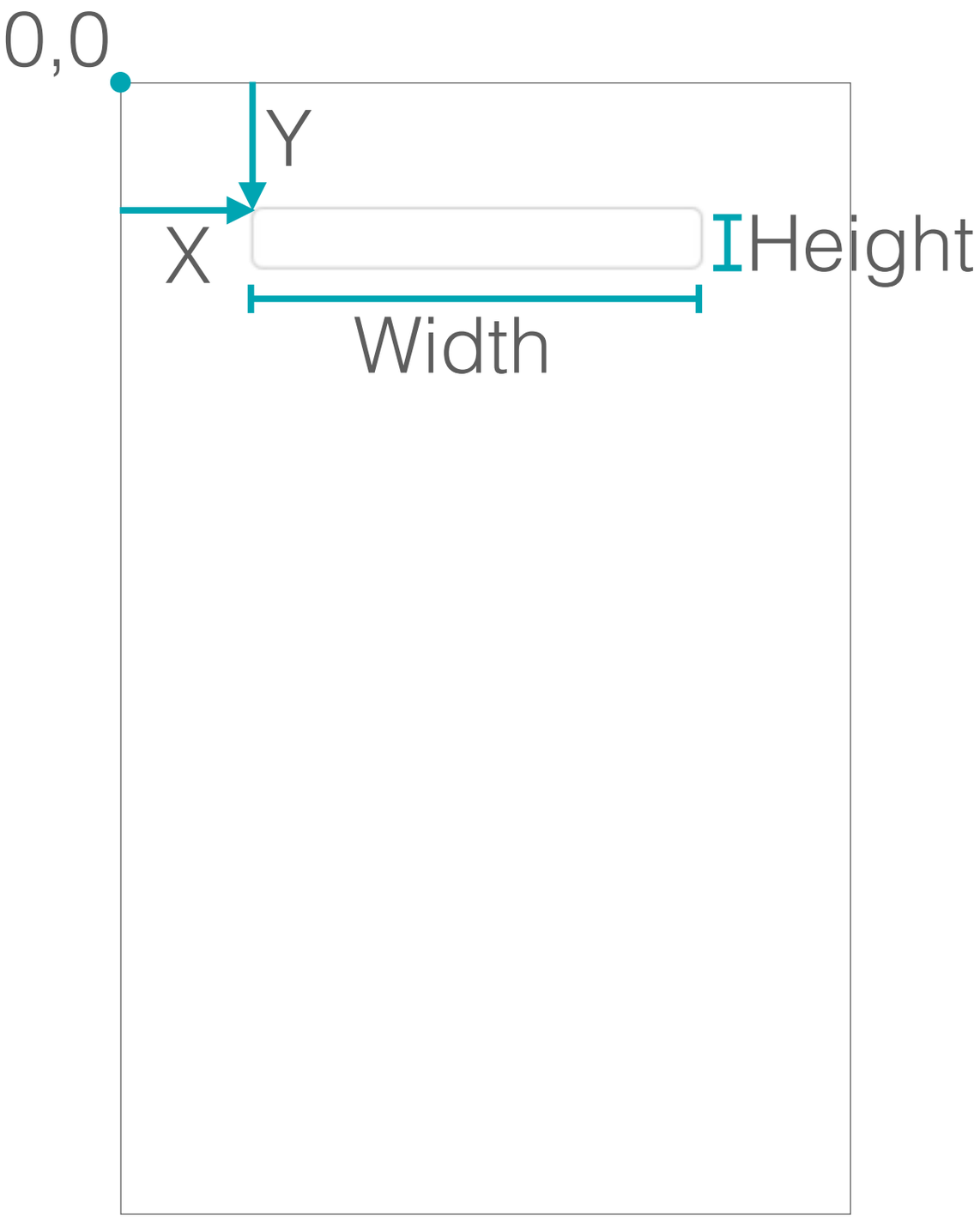


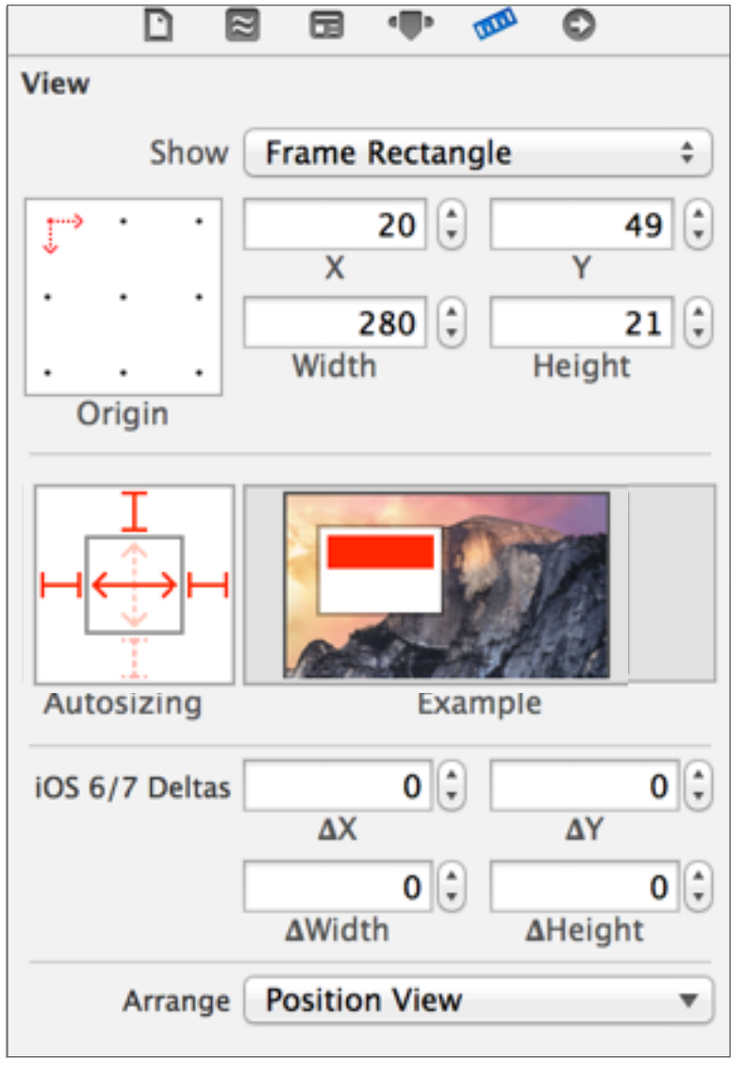
Rotation Tangent

Rotation happens automatically at ~30°

Orientation can be restricted

Springs & Struts (i.e. the Old Way)



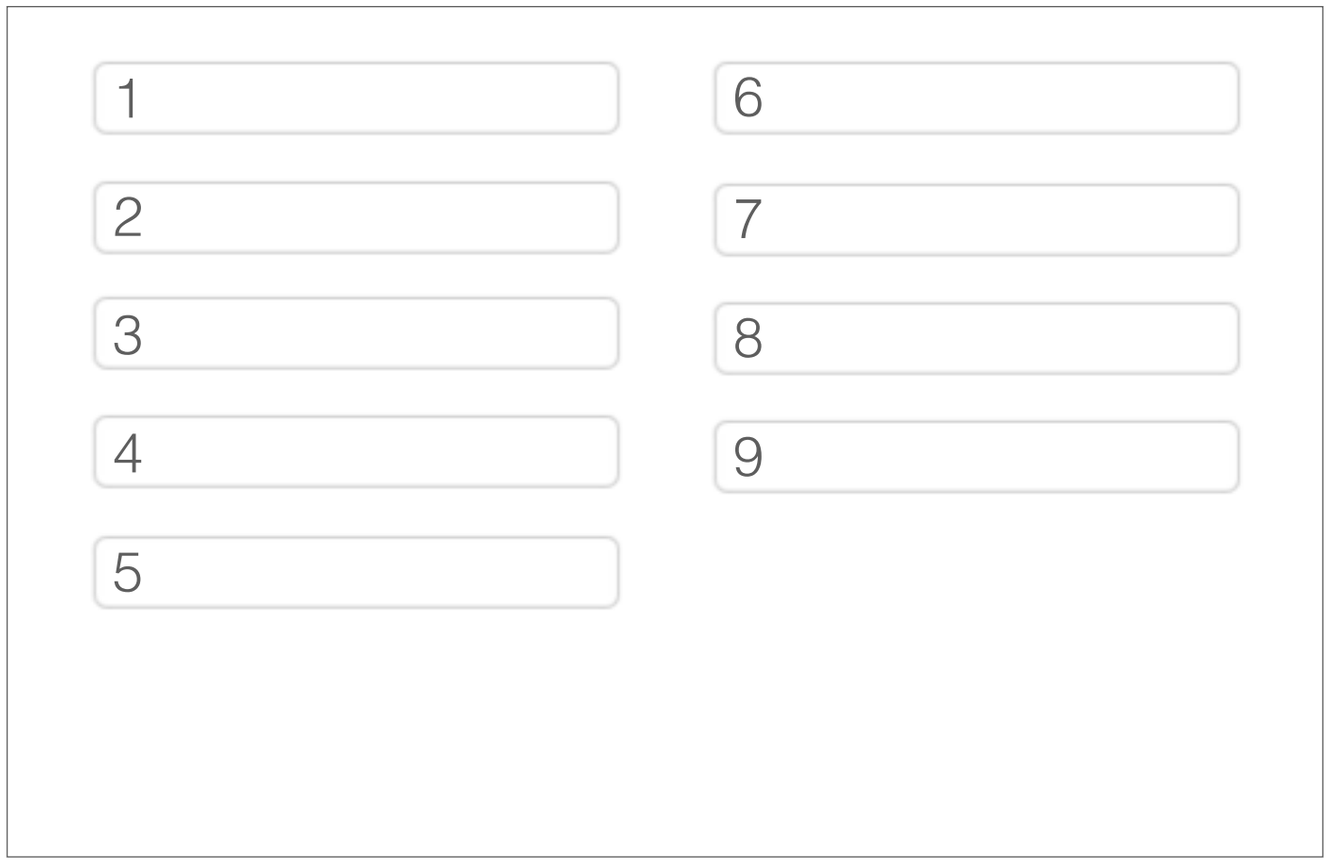


Handling Size Changes

Forces scrolling, which we want to avoid

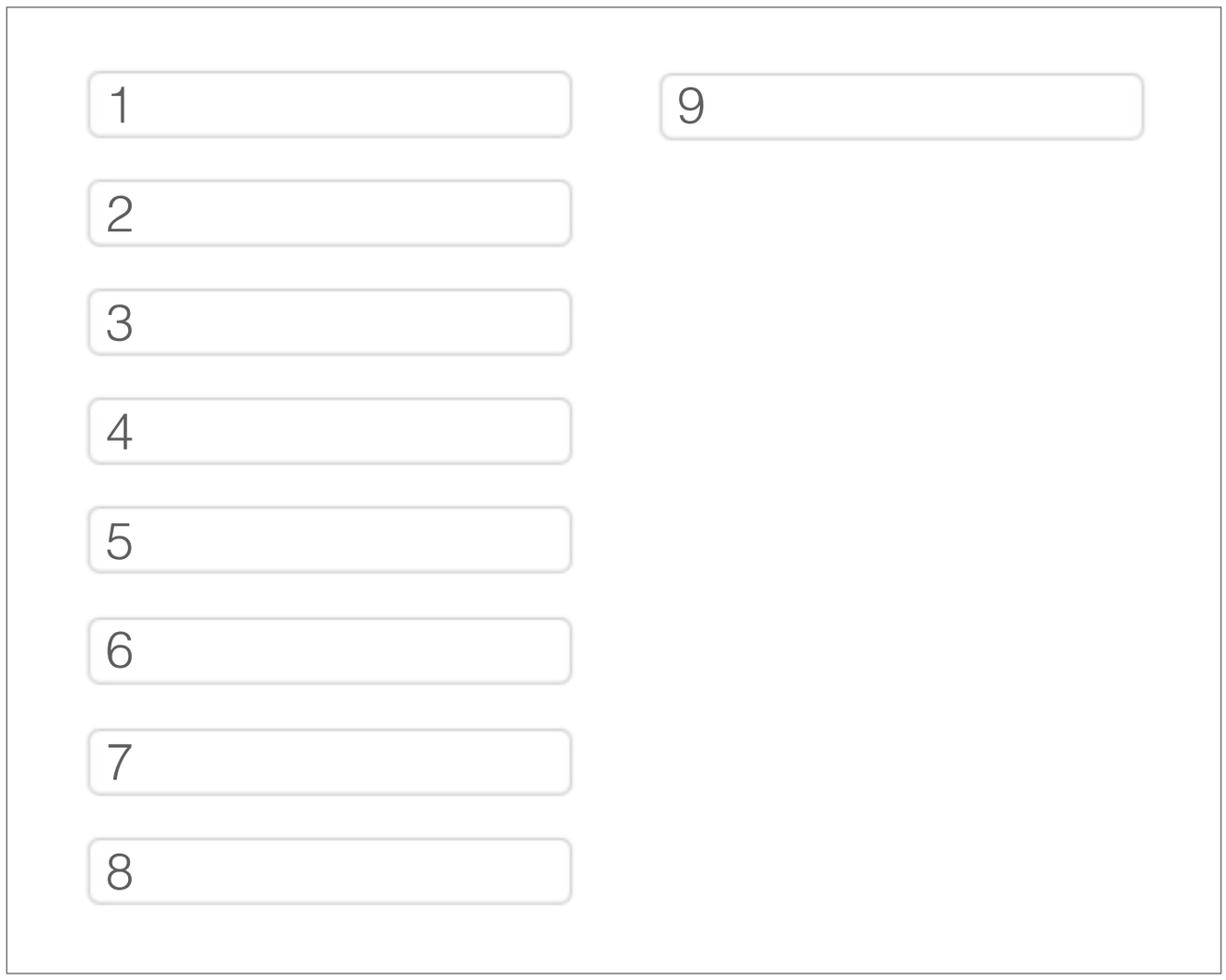
Handling Size Changes

And what if we want a different layout...

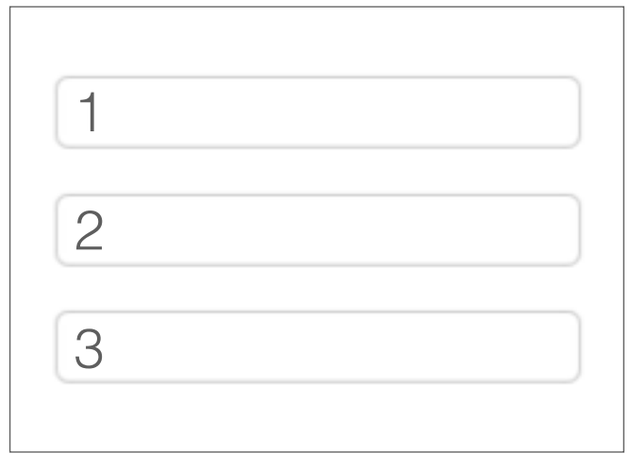


Handling Size Changes

Or what if we have a completely different size or shape...

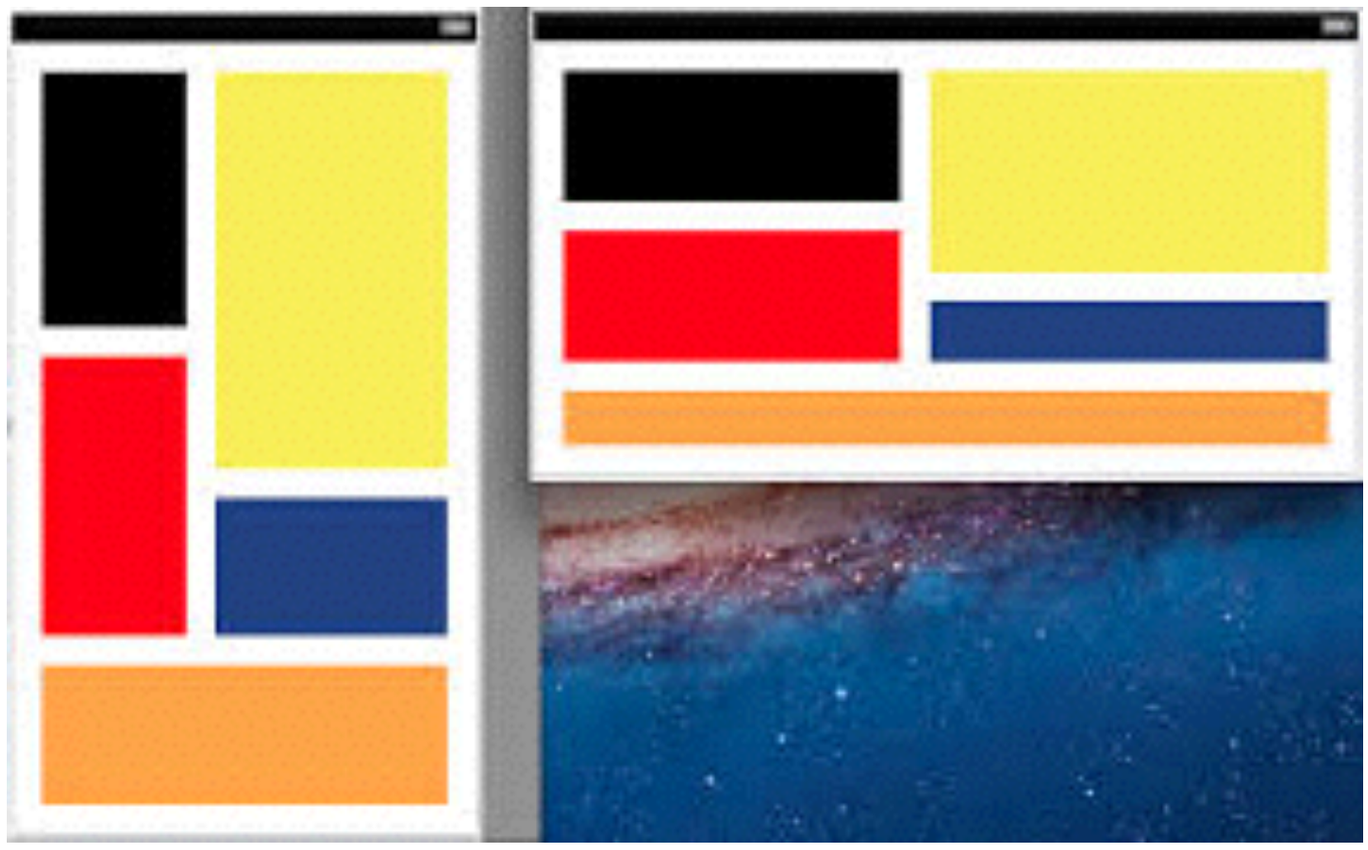


OR



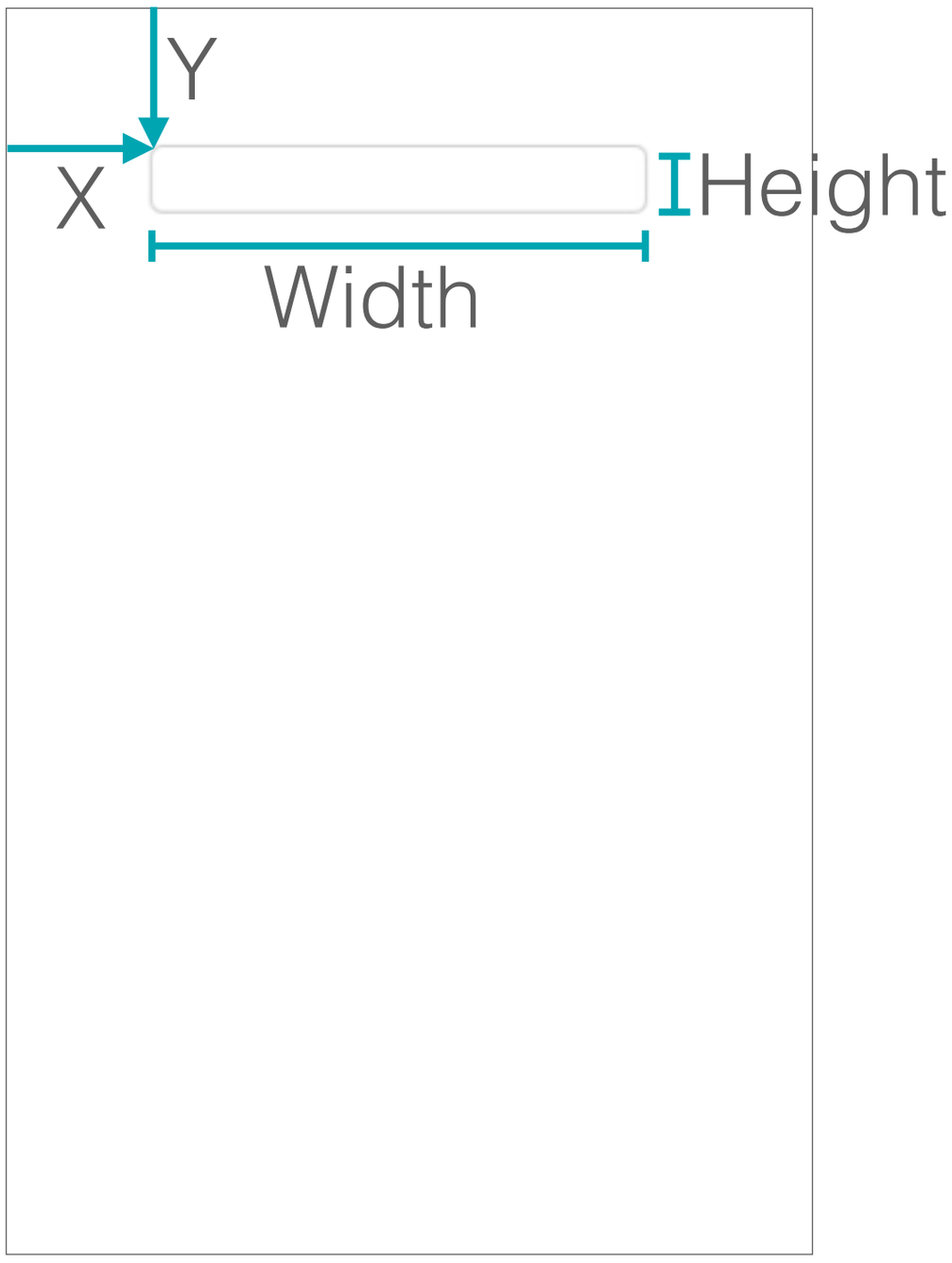
AutoLayout

- How things resize and even reposition in different orientations
- Uses a set of rules (called constraints) for layout
- Generally hated, but getting better and has become essential



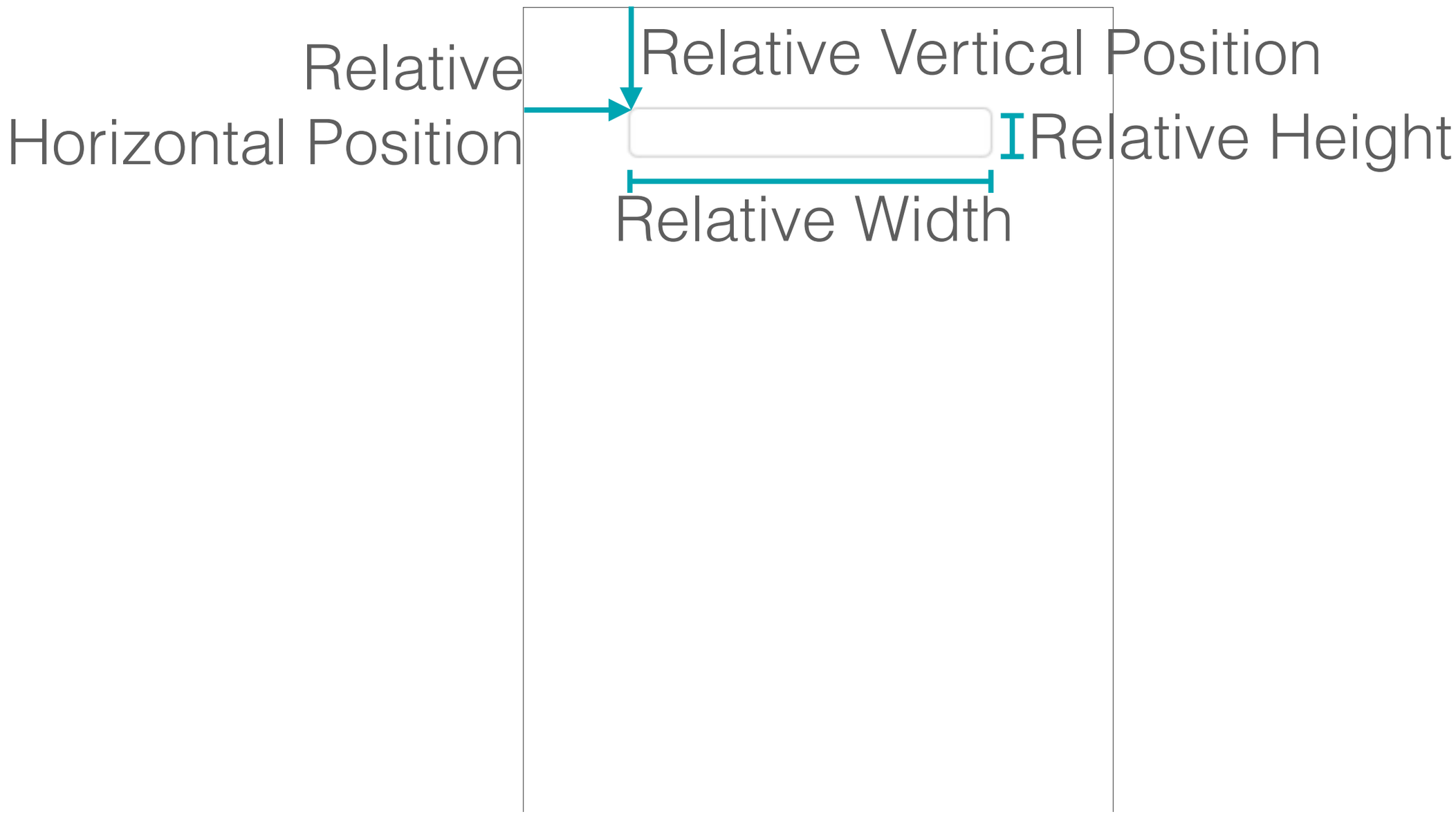
Layout Basics

Still need to know...



AutoLayout Basics

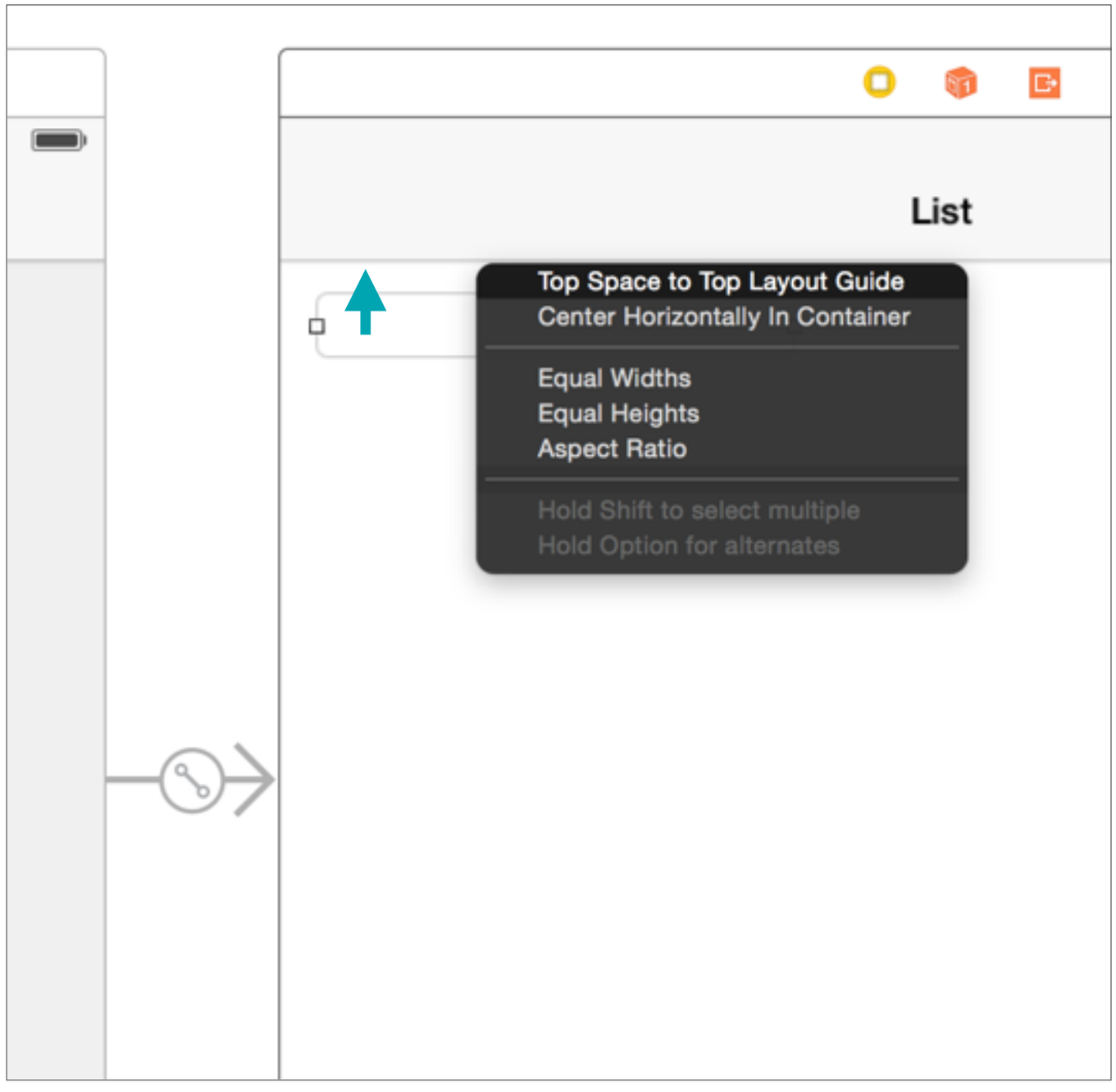
With a slight change...



Everything is positioned relative to something else

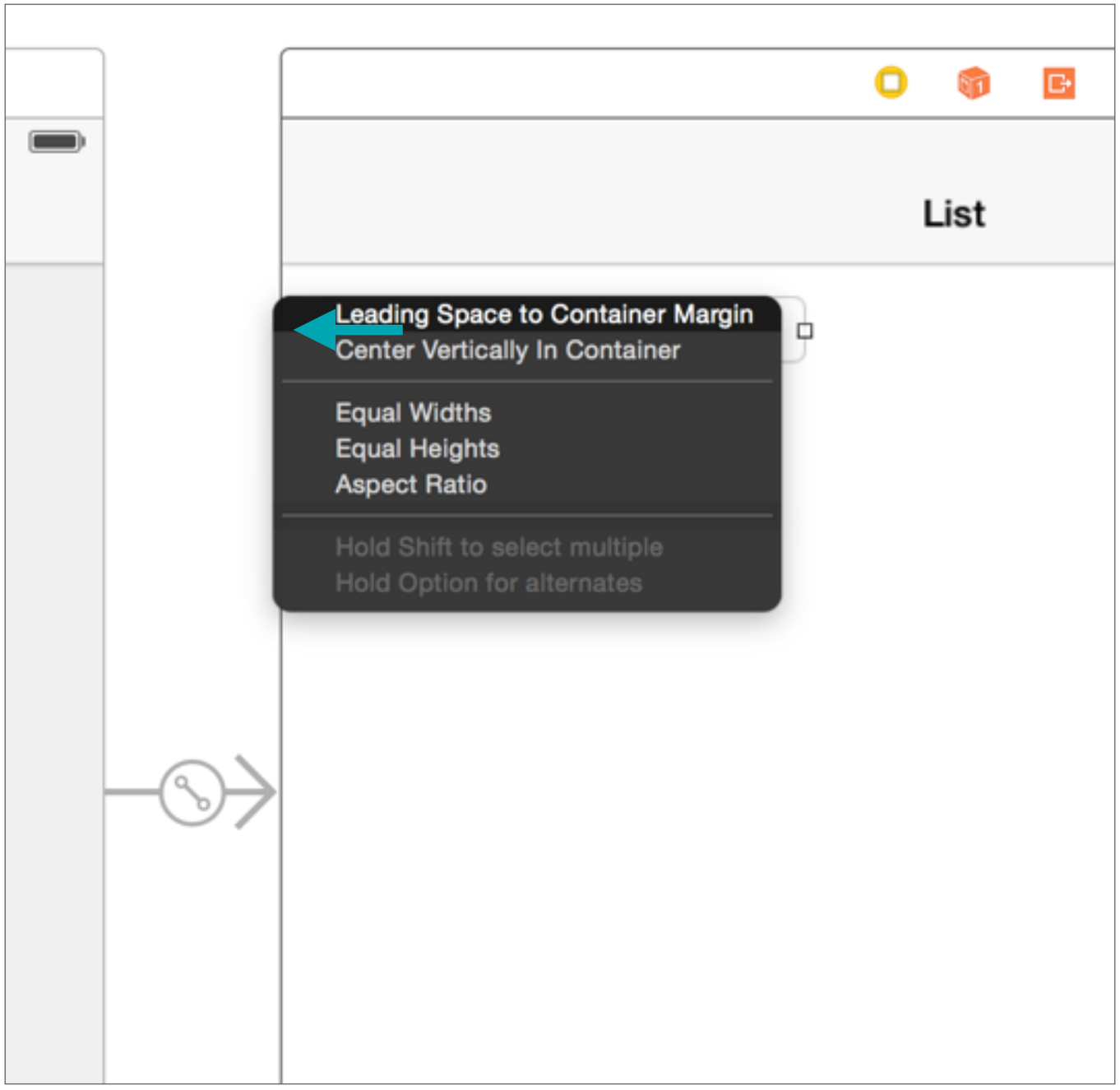
AutoLayout Basics

CTRL-drag from object to top margin or to another object to get Top Space (i.e. Y)



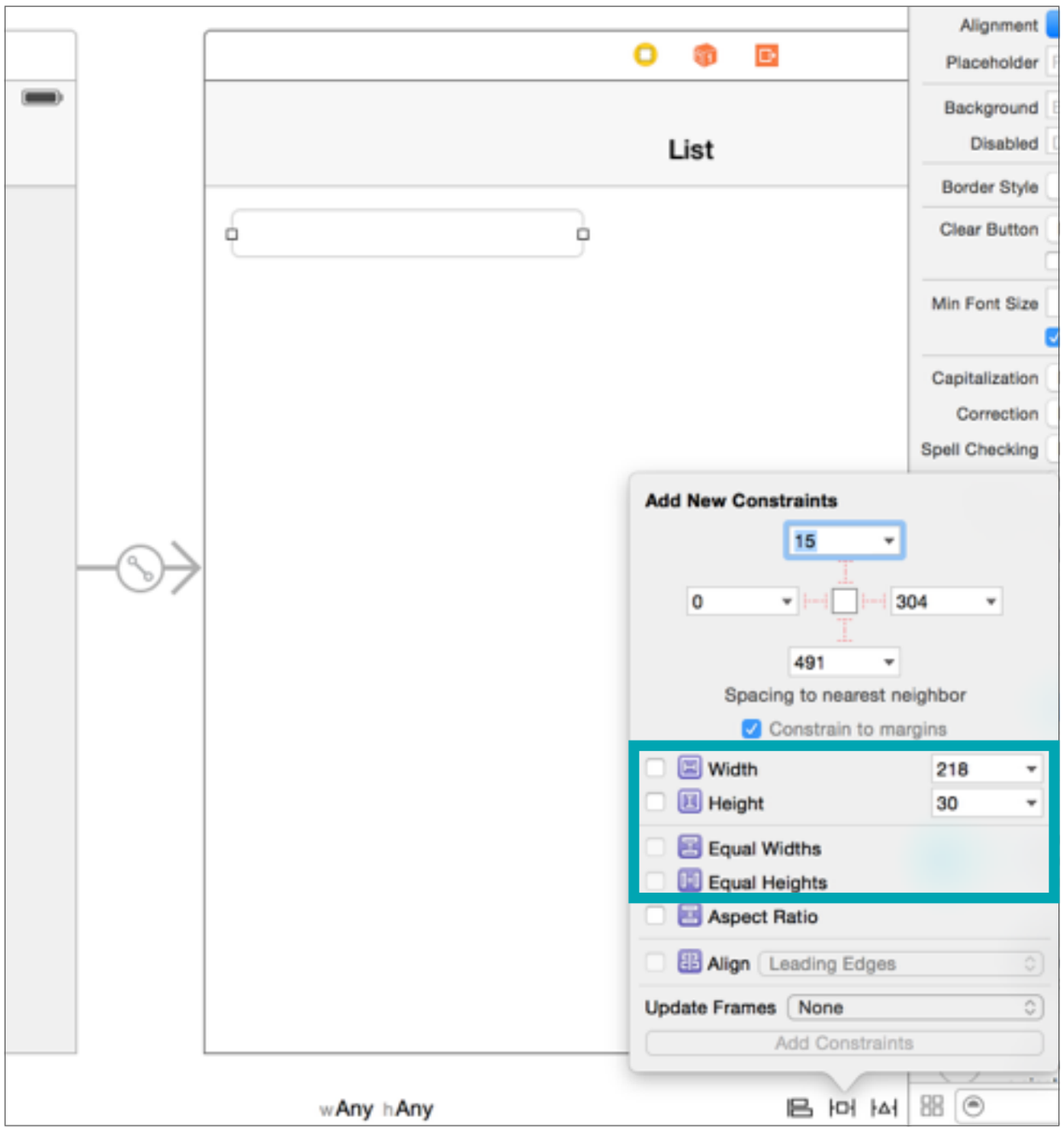
AutoLayout Basics

CTRL-drag from object to left margin or to another object to get Leading Space (i.e. X)



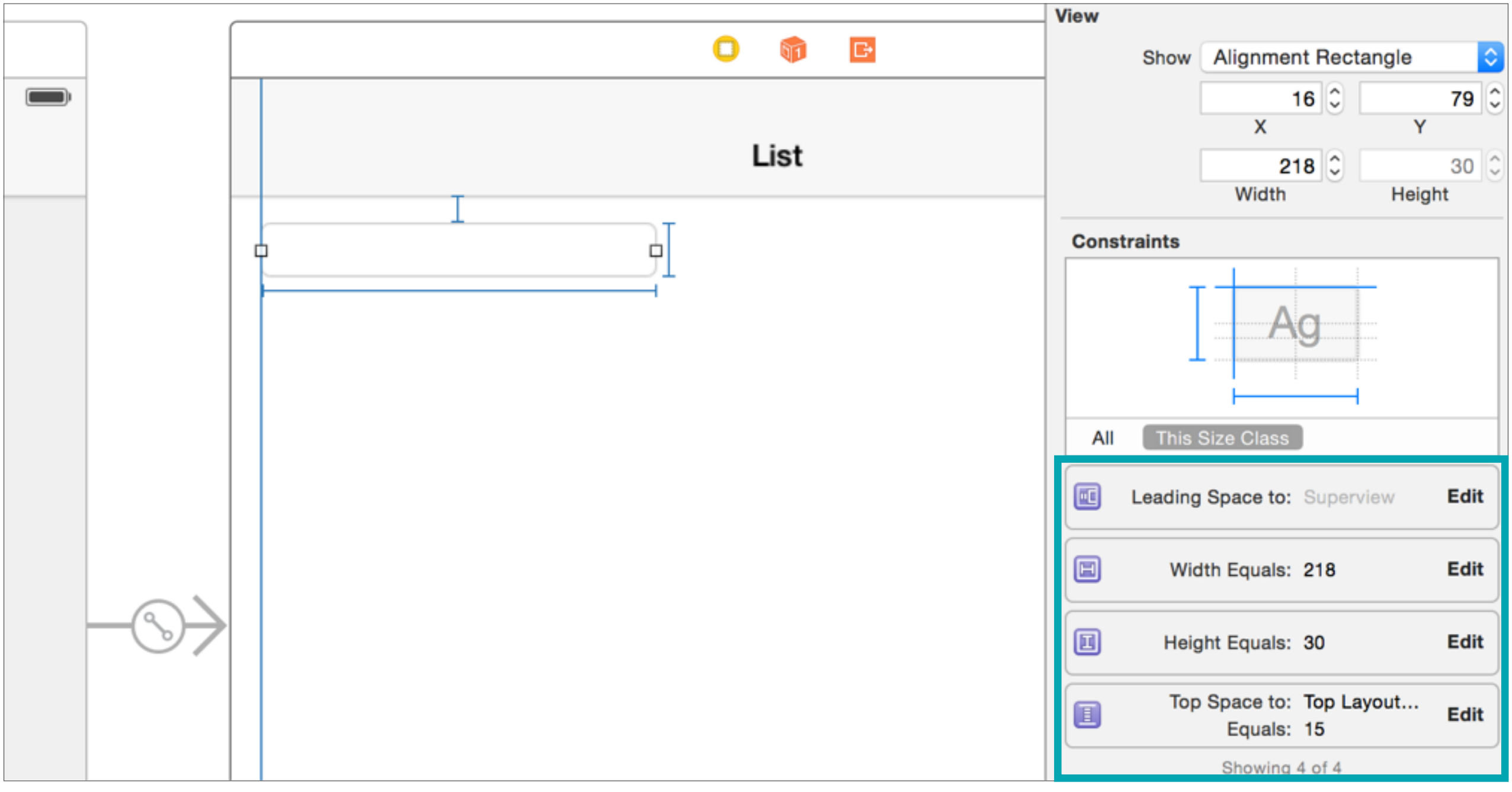
AutoLayout Basics

Set object's specific width & height or make them relative to other objects (equal or percent)



AutoLayout Basics

These create rules for layout (or Constraints, more specifically)

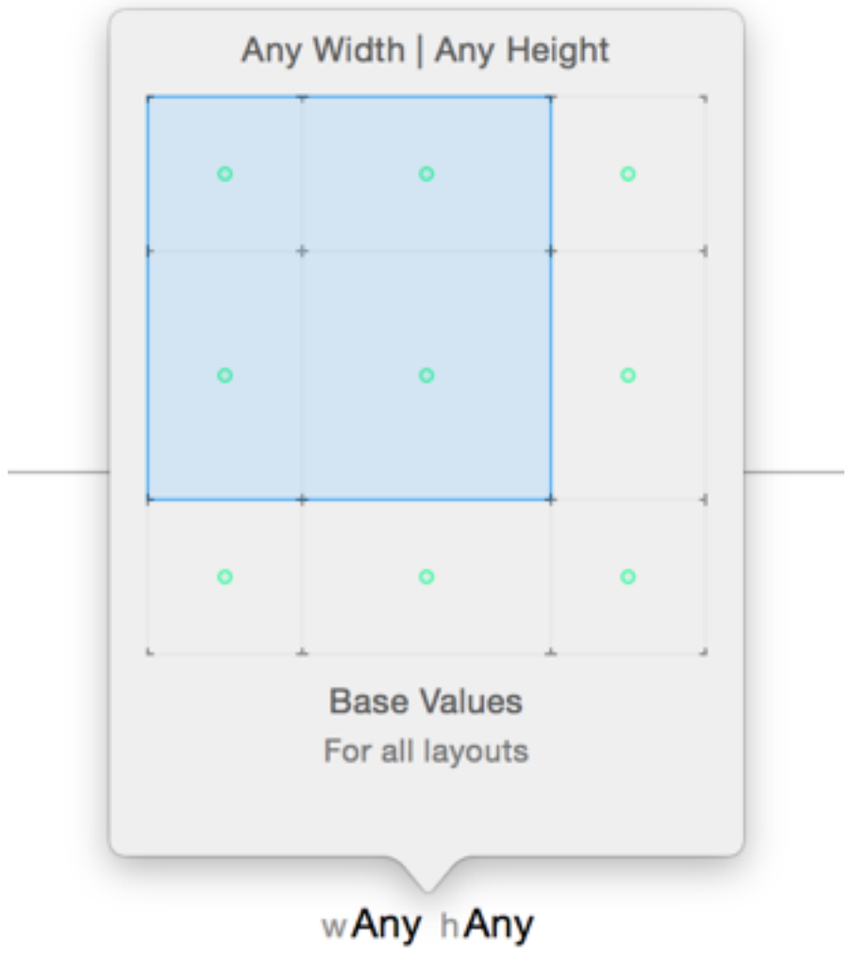


AutoLayout Basics

- Every UI object needs some way to define it's horizontal & vertical placement plus size & width
- These are often derived rather than explicit:
  - Relative to left & right margin gives width
  - Relative to top & bottom margin gives height
  - Equal Width to another control with a width gives width
  - Percentage Width to another control with a width gives width
  - Aspect Ratio with a height gives width

AutoLayout with Size Classes

Size Classes allow different layouts for different sizes of devices or the same device when rotated, using the same (or different) controls



More Controls

Homework