Introduction to Auto Layout for iOS and OS X

Come on in, the water's fine!

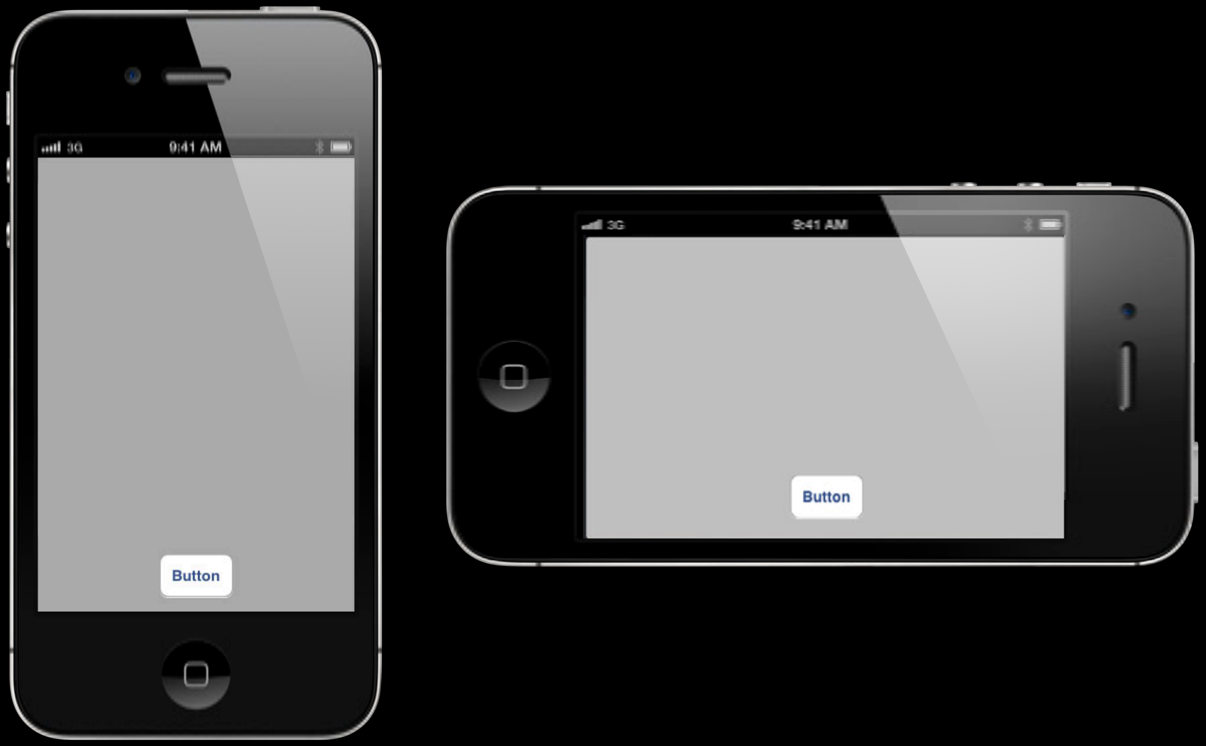
Session 202

Marian Goldeen
iOS Frameworks Engineer

What Is Auto Layout?

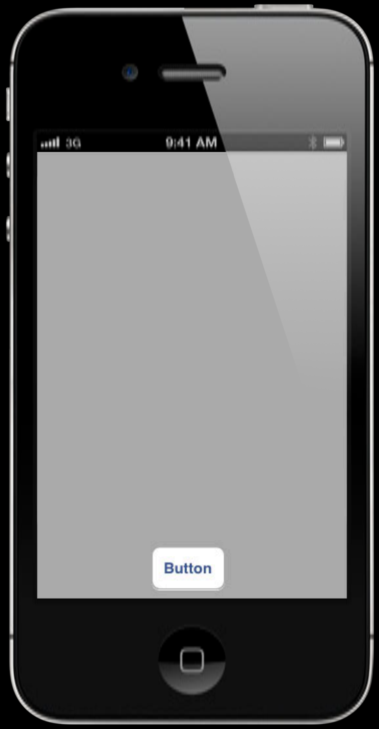
What Is Auto Layout?

What Is Auto Layout?



Auto Layout Is a Constraint-Based, Descriptive Layout System

Hard-Coded Layout



- Button’s frame origin is (124, 396)

Hard-Coded Layout

- Button’s frame origin is (124, 396)

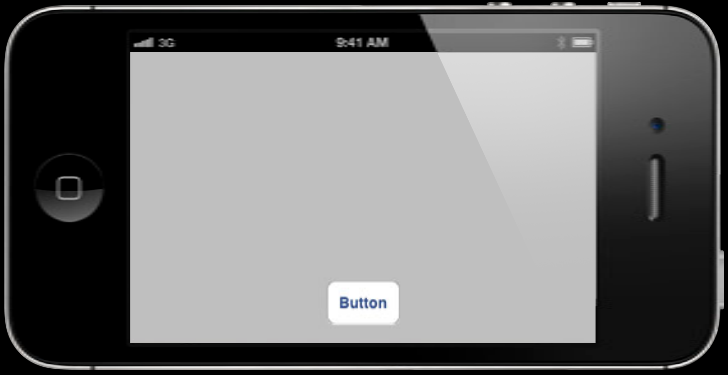
Hard-Coded Layout

- Button’s frame origin is (124, 396)

Hard-Coded Layout

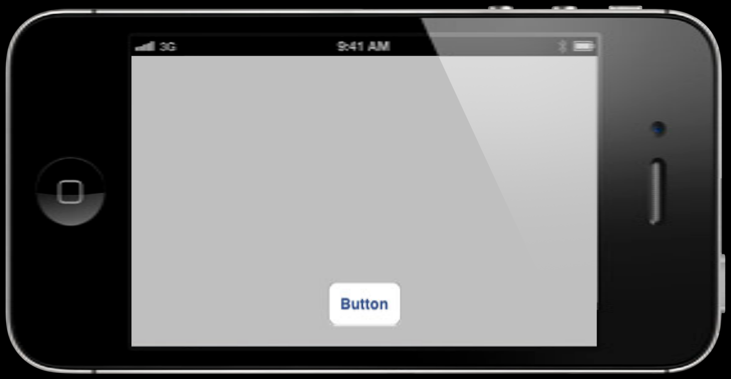
- Button’s frame origin is (124, 396)

Auto Layout

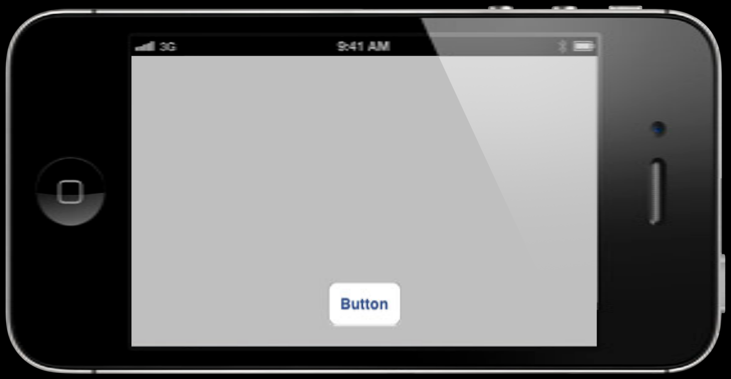


- Button is centered horizontally in its superview
- Button is a fixed distance from the bottom of the superview

- Button's frame origin is (124, 396)



- Button is centered horizontally in its superview
- Button is a fixed distance from the bottom of the superview



- Button.centerX = Superview.centerX
- Button.bottom = Superview.bottom – <padding>

Auto Layout is a constraint-based, descriptive layout system

Auto Layout is a constraint-based, descriptive layout system

Describe the layout with constraints, and frames are calculated automatically.

Agenda

- Setting up Constraint-Based Layout

Agenda

- Setting up Constraint-Based Layout
- Layout Behind the Scenes

Agenda

- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language

Agenda

- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong

Agenda

- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

Agenda

- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

Demo

Setting up layout, part 1

Describe the Layout with Constraints

- Using Interface Builder
- Using code (optional)
    - Step 1—Create your constraints

Describe the Layout with Constraints

- Using Interface Builder ✓
- Using code (optional)
    - Step 1—Create your constraints

Describe the Layout with Constraints

- Using Interface Builder ✓
- Using code (optional)

Describe the Layout with Constraints

- Using Interface Builder ✓
- Using code (optional)
  - Step 1—Create your constraints

NSLayoutConstraint.h

item1.attribute1 = multiplier × item2.attribute2 + constant

NSLayoutConstraint.h

item1.attribute1 = multiplier × item2.attribute2 + constant

NSLayoutConstraint.h

item1.attribute1 = multiplier × item2.attribute2 + constant

```
+ (id)constraintWithItem:(id)item1
               attribute:(NSLayoutAttribute)attribute1
               relatedBy:(NSLayoutRelation)relation
                  toItem:(id)item2
               attribute:(NSLayoutAttribute)attribute2
              multiplier:(CGFloat)multiplier
                constant:(CGFloat)constant;
```

item1.attribute = multiplier × item2.attribute + constant

- Button.centerX = Superview.centerX

```
[NSLayoutConstraint constraintWithItem:button
                             attribute:NSLayoutAttributeCenterX
                             relatedBy:NSLayoutRelationEqual
                                toItem:superview
                             attribute:NSLayoutAttributeCenterX
                            multiplier:1.0
                              constant:0.0]
```

- Button.bottom = Superview.bottom – <padding>

```
[NSLayoutConstraint constraintWithItem:button
                             attribute:NSLayoutAttributeBottom
                             relatedBy:NSLayoutRelationEqual
                                toItem:superview
                             attribute:NSLayoutAttributeBottom
                            multiplier:1.0
                              constant:-padding]
```

Describe the Layout with Constraints in Code

- Step 1—Create your constraints
- Step 2—Add them to a view

Describe the Layout with Constraints in Code

- Step 1—Create your constraints ✓
- Step 2—Add them to a view

Describe the Layout with Constraints in Code

- Step 1—Create your constraints ✓
- Step 2—Add them to a view

NSLayoutConstraint.h

AppKit

UIView.h

UIKit

```
- (void)addConstraint:(NSLayoutConstraint *)constraint;
```

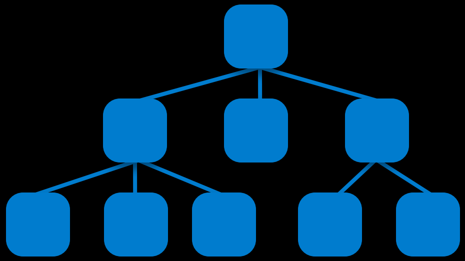
NSLayoutConstraint.h

AppKit

UIView.h

UIKit

```
- (void)addConstraint:(NSLayoutConstraint *)constraint;
```



Q: add them to which view?

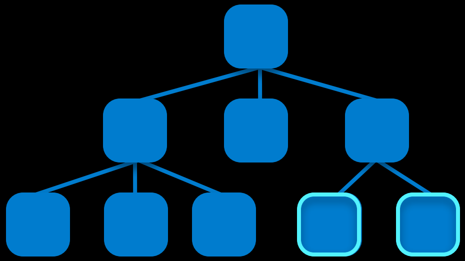
NSLayoutConstraint.h

AppKit

UIView.h

UIKit

```
- (void)addConstraint:(NSLayoutConstraint *)constraint;
```



Q: add them to which view?

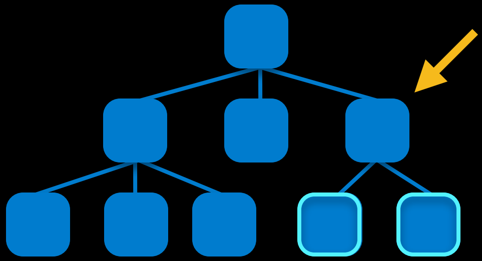
NSLayoutConstraint.h

AppKit

UIView.h

UIKit

```
- (void)addConstraint:(NSLayoutConstraint *)constraint;
```



Q: add them to which view?

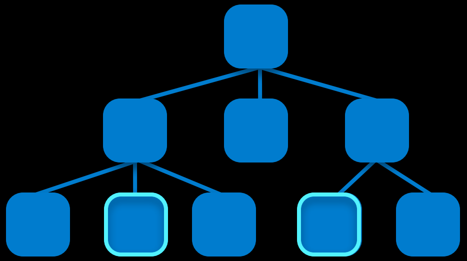
NSLayoutConstraint.h

AppKit

UIView.h

UIKit

```
- (void)addConstraint:(NSLayoutConstraint *)constraint;
```



Q: add them to which view?

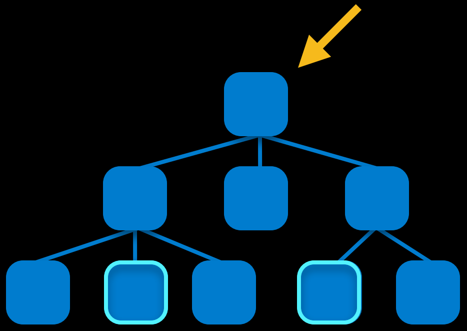
NSLayoutConstraint.h

AppKit

UIView.h

UIKit

```
- (void)addConstraint:(NSLayoutConstraint *)constraint;
```



Q: add them to which view?

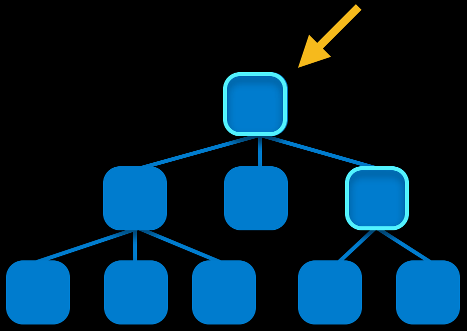
NSLayoutConstraint.h

AppKit

UIView.h

UIKit

```
- (void)addConstraint:(NSLayoutConstraint *)constraint;
```



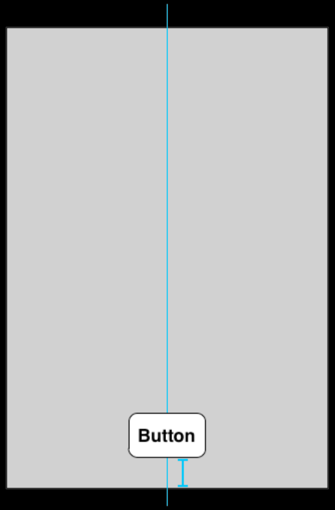
Q: add them to which view?

Demo

Setting up layout, part 2

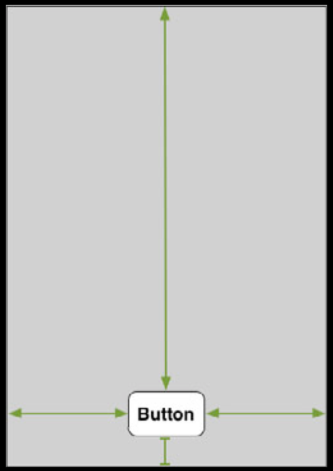
I Can Do That with Springs and Struts!

Constraints



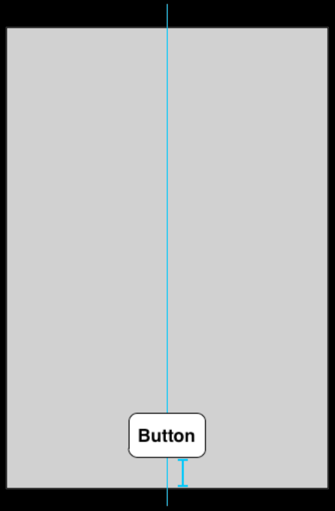
=?

Autoresizing Mask

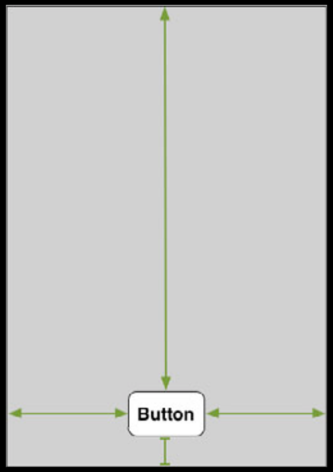


I Can Do That with Springs and Struts!

Constraints



Autoresizing Mask



Constraints

Constraints

- Can apply to any two views, regardless of view hierarchy

Constraints

- Can apply to any two views, regardless of view hierarchy
- Can establish maximums and minimums with inequalities

Constraints

- Can apply to any two views, regardless of view hierarchy
- Can establish maximums and minimums with inequalities
- Can be prioritized

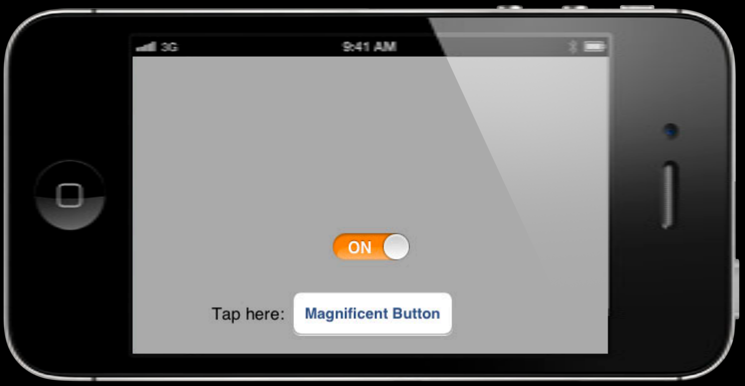
```
          NS
@property   LayoutPriority priority;
          UI
          NS
@property   LayoutPriority priority;
          UI

 NS
   LayoutPriorityRequired = 1000
 UI
```

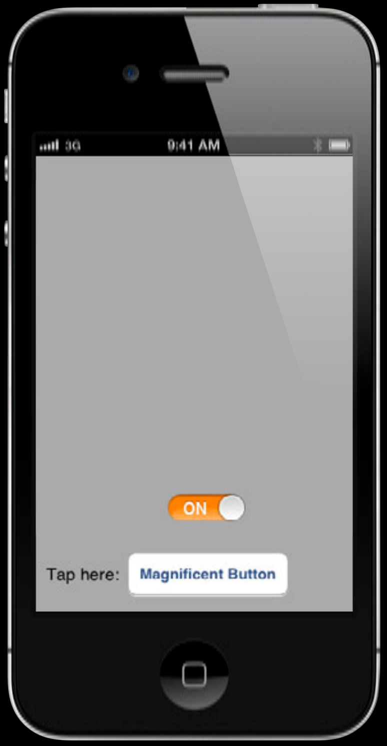
Demo

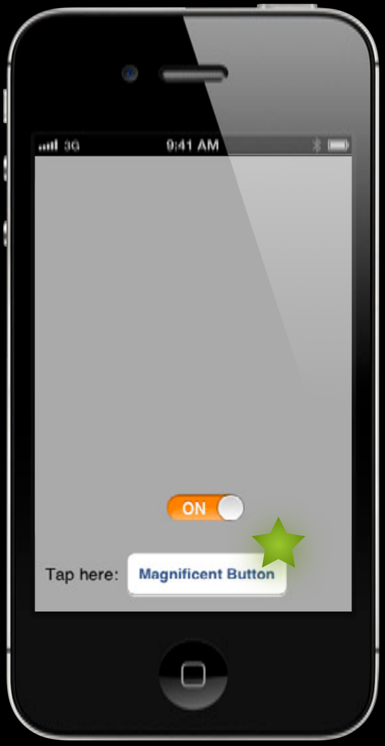
Priorities, inequalities, and cross-view constraints

Agenda

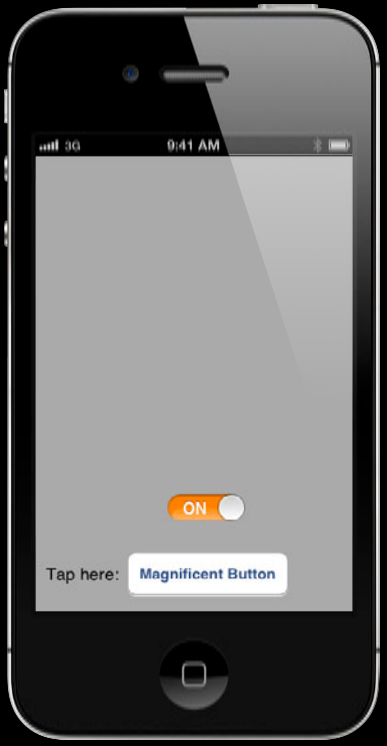
- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

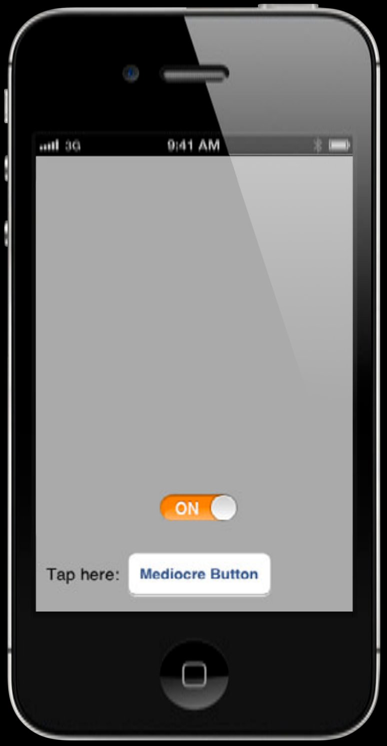

3G
9:41 AM
ON
Tap here:
Magnificent Button


3G
9:41 AM
ON
Tap here:
Magnificent Button

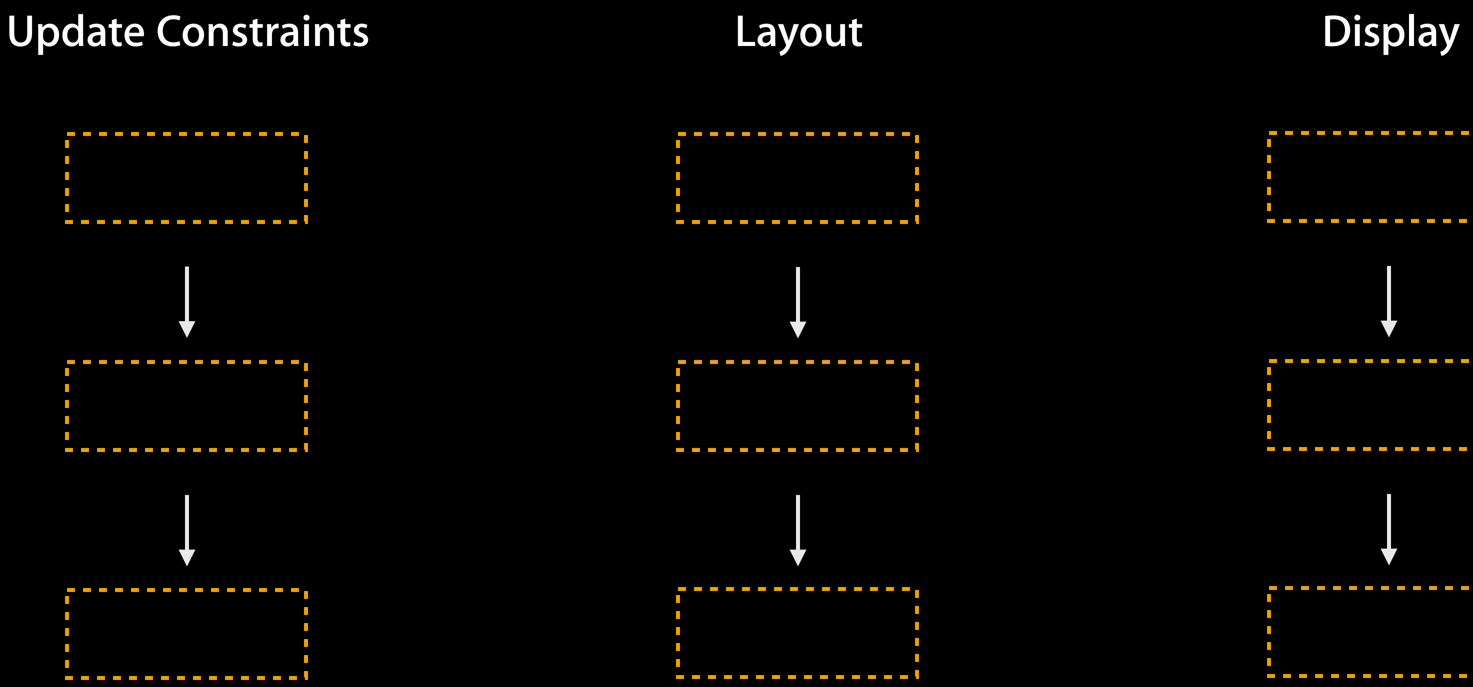


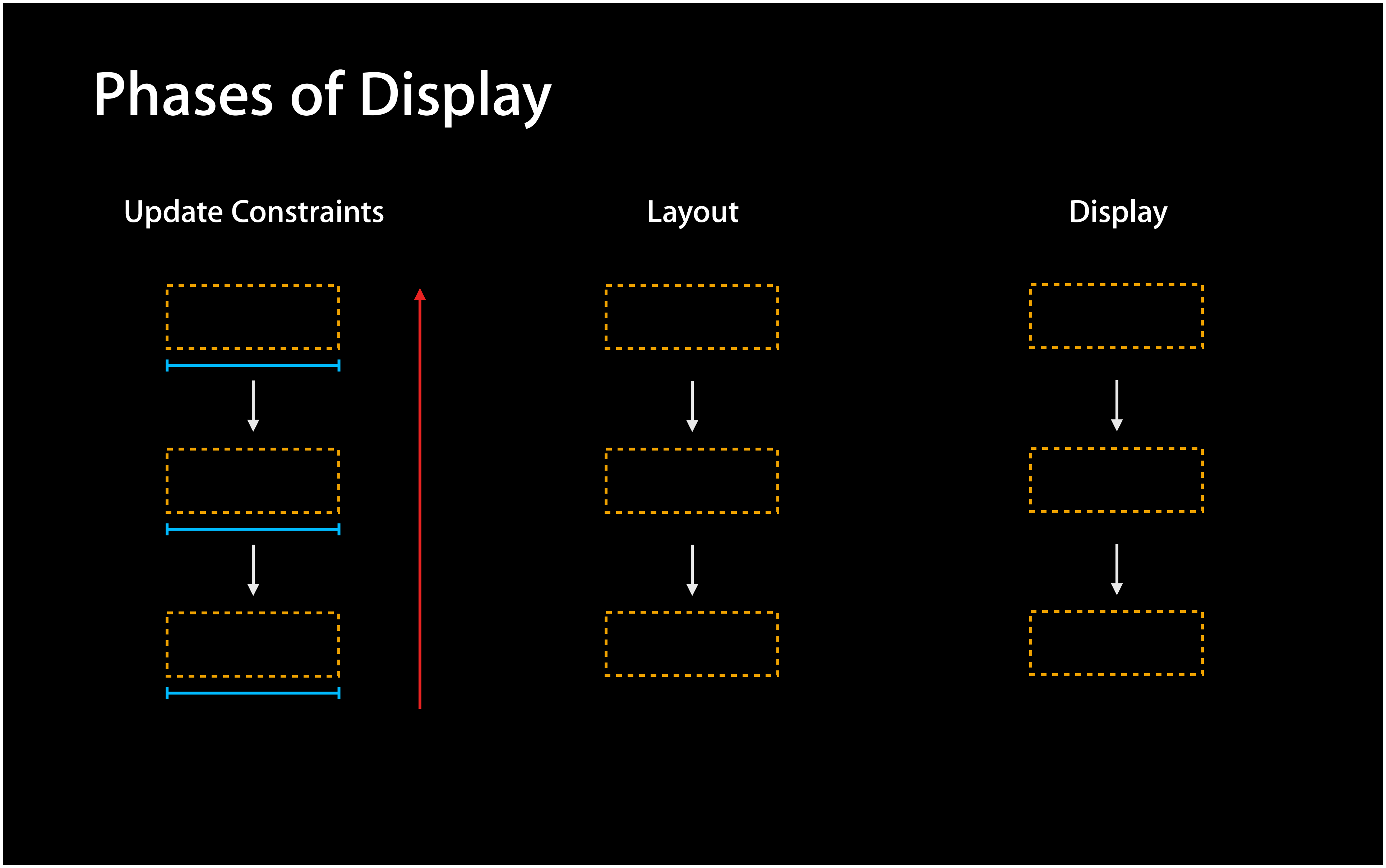
3G
9:41 AM
ON
Tap here:
Magnificent Button


3G
9:41 AM
ON
Tap here:
Magnificent Button

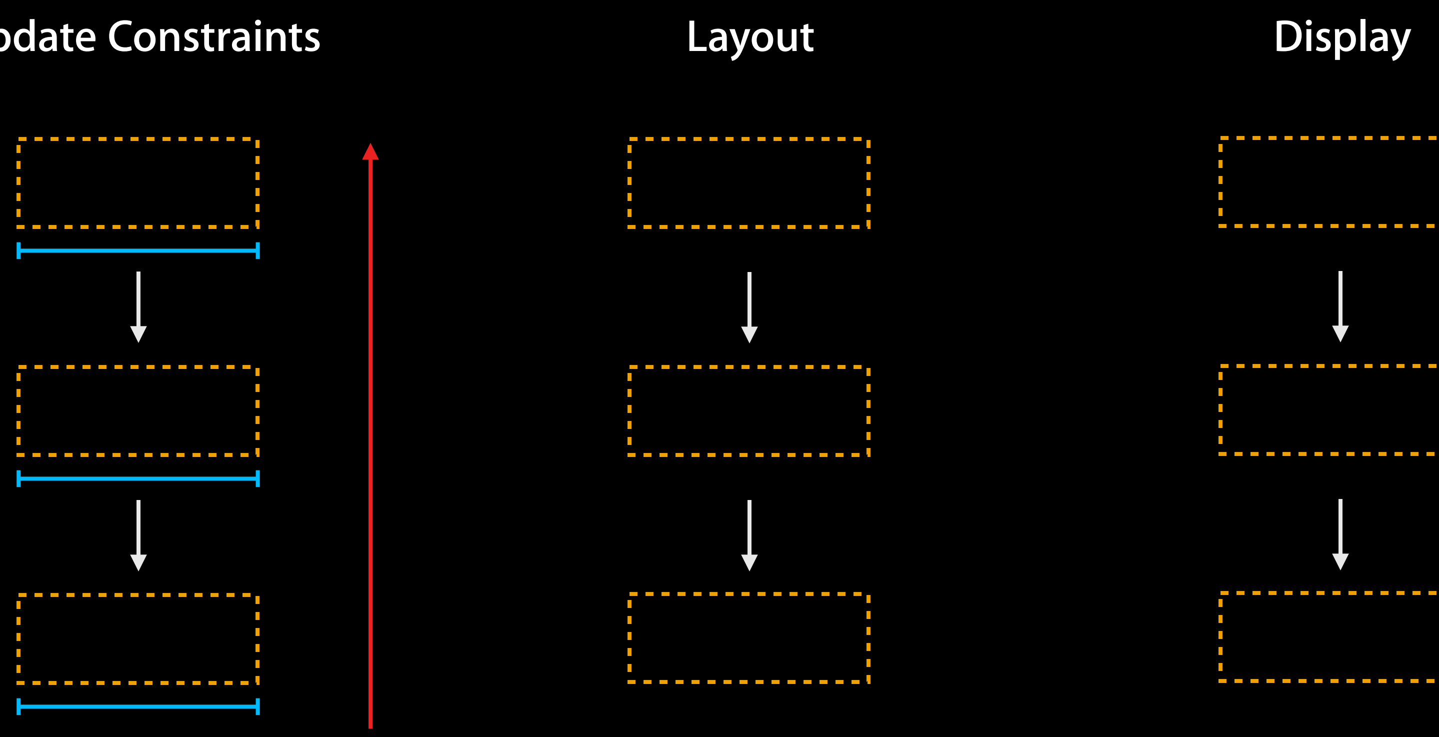

3G
9:41 AM
ON
Tap here:
Mediocre Button

Phases of Display

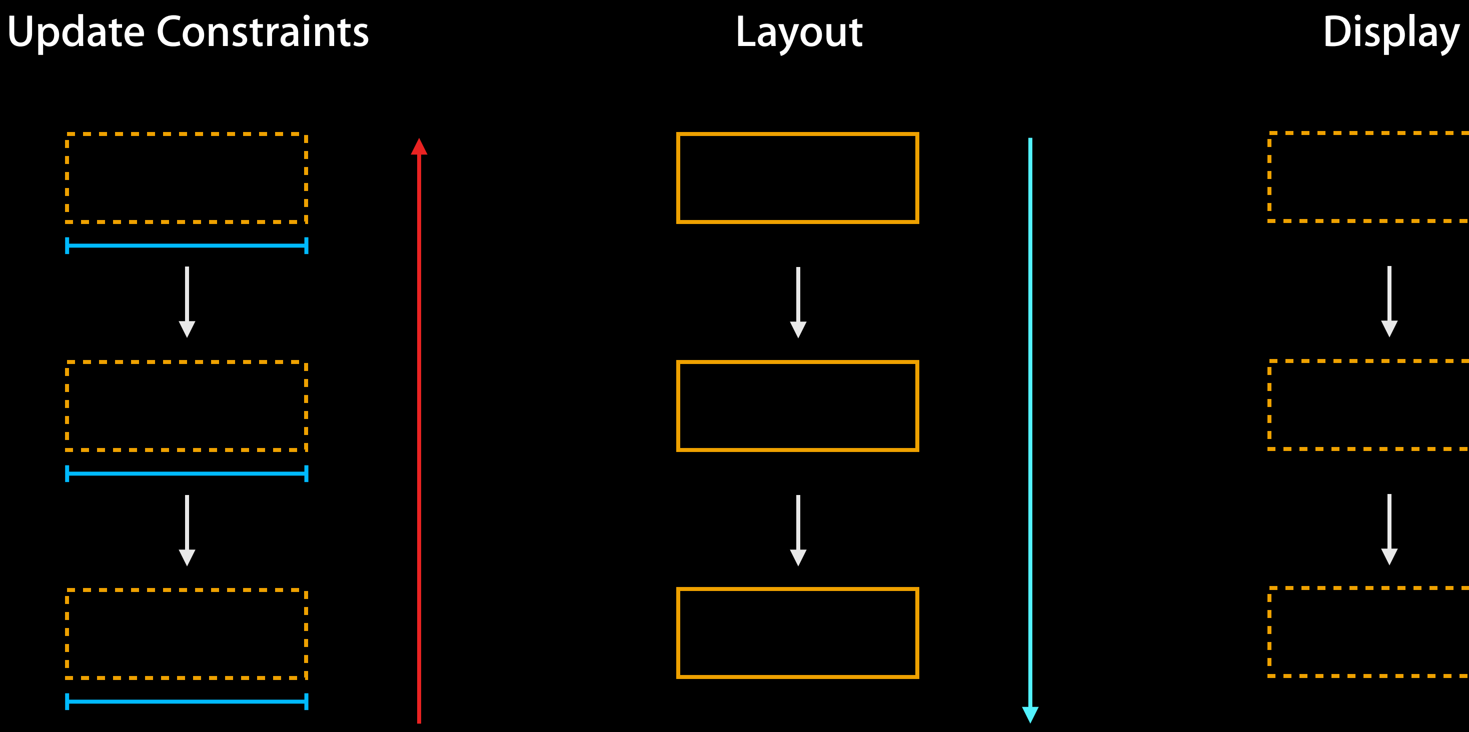


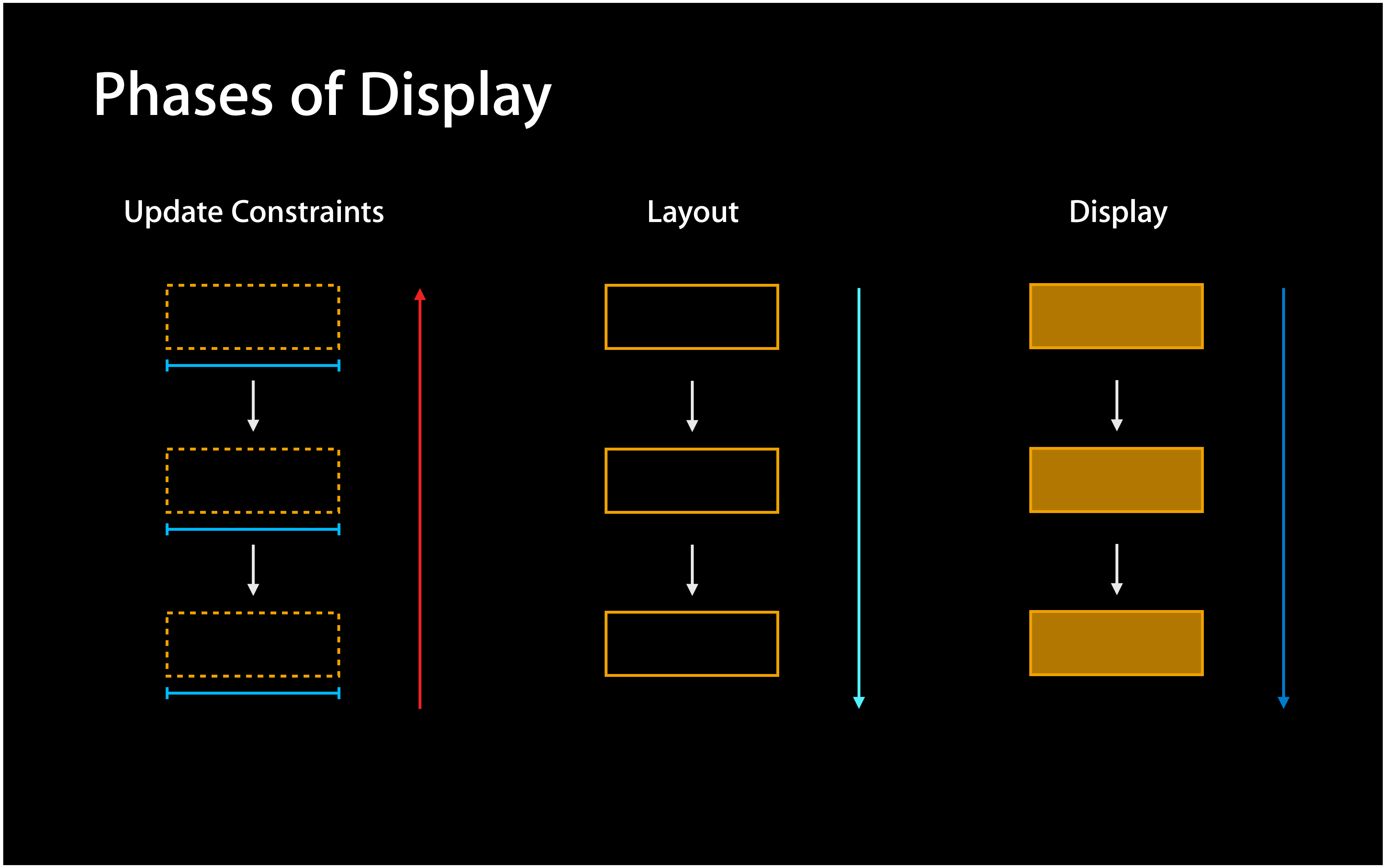


Phases of Display

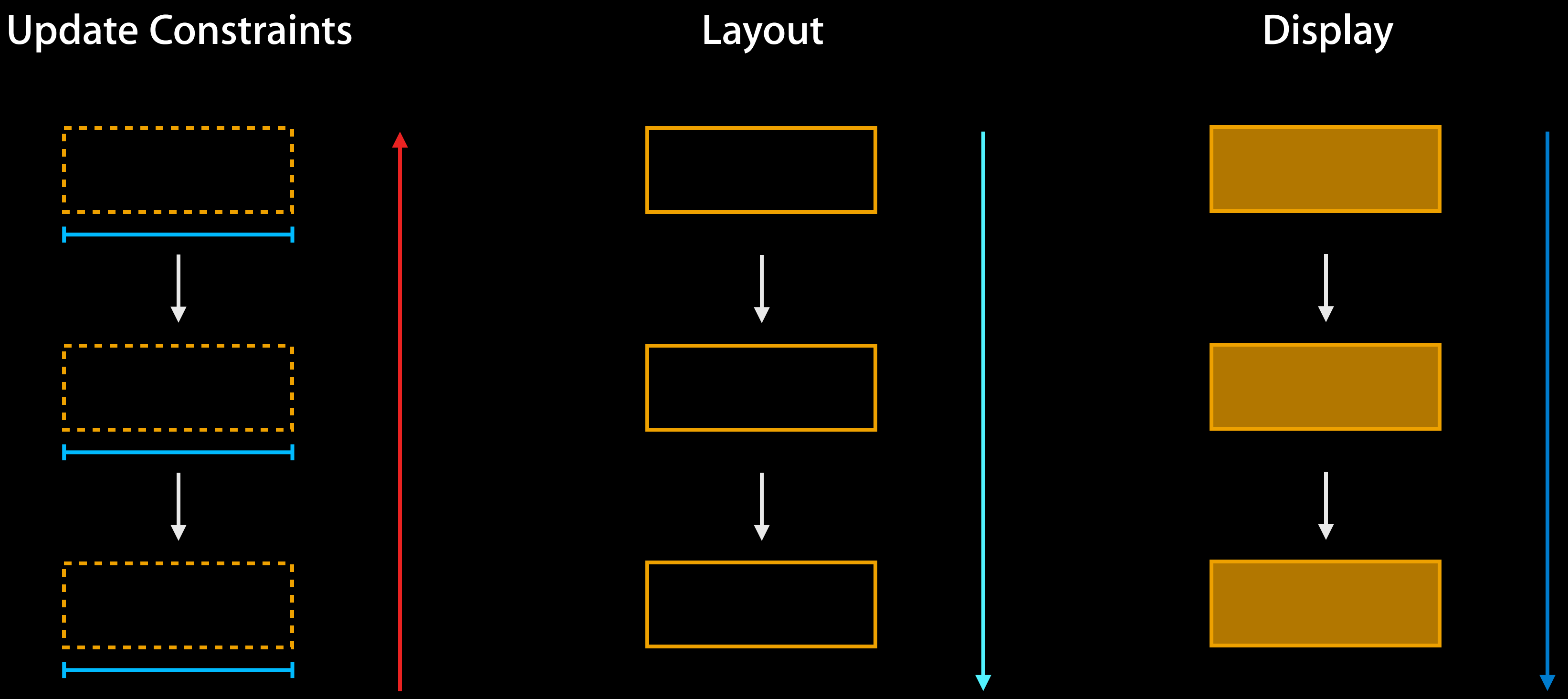


Phases of Display





Phases of Display



NSView	UIView
`-setNeedsDisplay:`	`-setNeedsDisplay`
`-setNeedsLayout:`	`-setNeedsLayout`
`-setNeedsUpdateConstraints:`	`-setNeedsUpdateConstraints`

NSView	UIView
`-setNeedsDisplay:`	`-setNeedsDisplay`
`-setNeedsLayout:`	`-setNeedsLayout`
`-setNeedsUpdateConstraints:`	`-setNeedsUpdateConstraints`

NSView

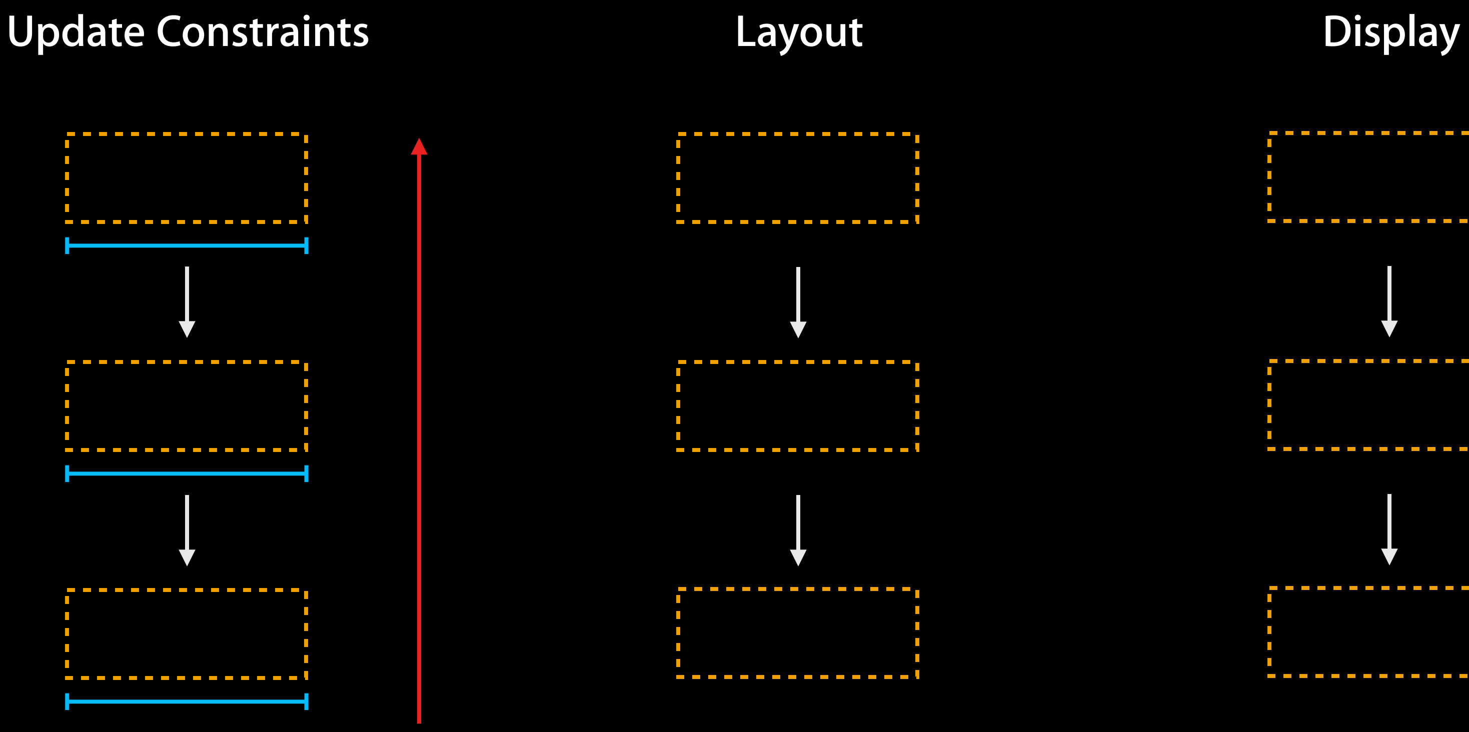
```
-setNeedsDisplay:
-setNeedsLayout:
-setNeedsUpdateConstraints:
```

UIView

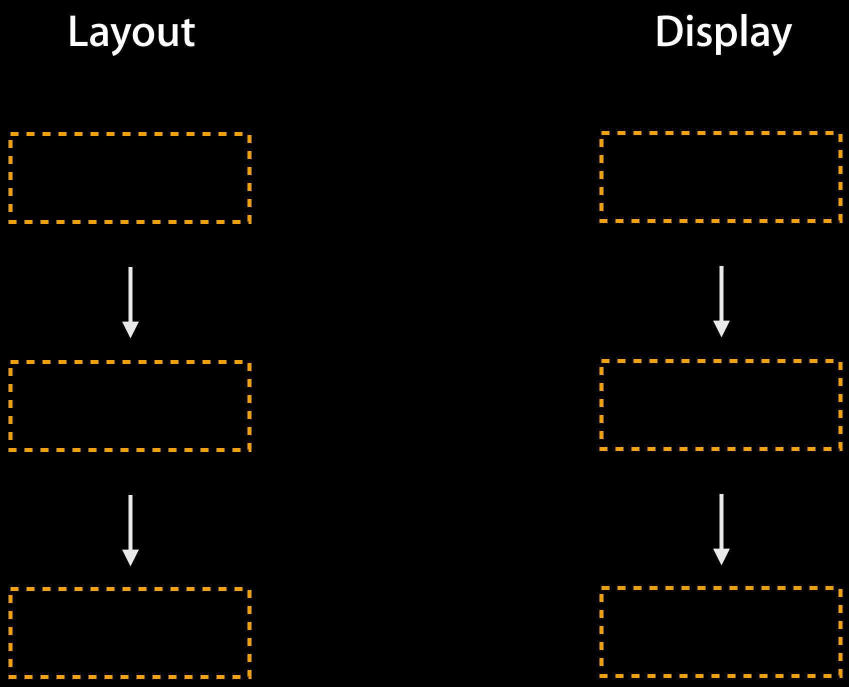
```
-setNeedsDisplay
-setNeedsLayout
-setNeedsUpdateConstraints
```

UIView/UIWindow	NSWindow	NSView
`-layoutIfNeeded`	`-layoutIfNeeded`	`-layoutSubtreeIfNeeded`

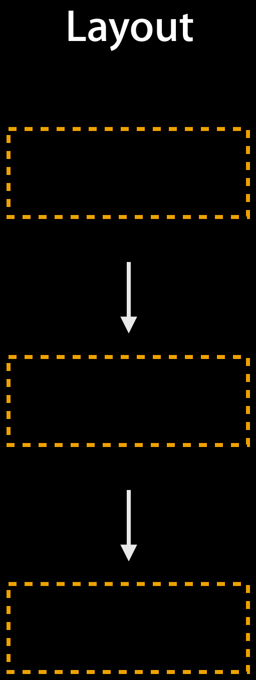
Phases of Display



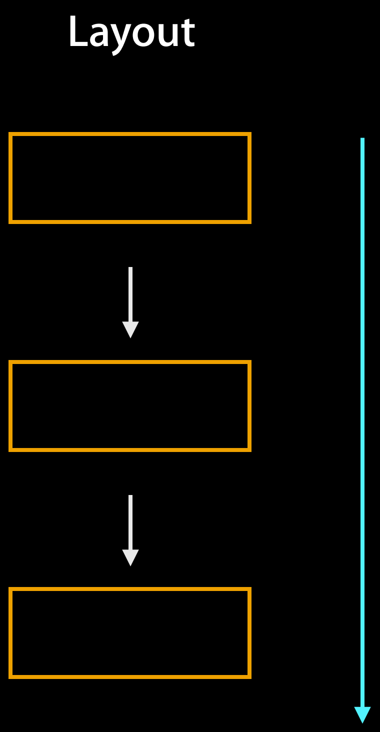
Phases of Display



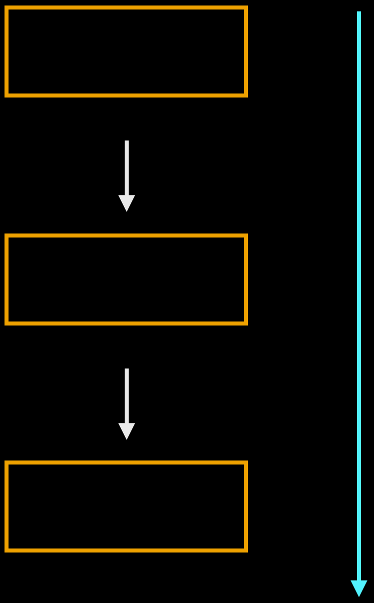
Phases of Display

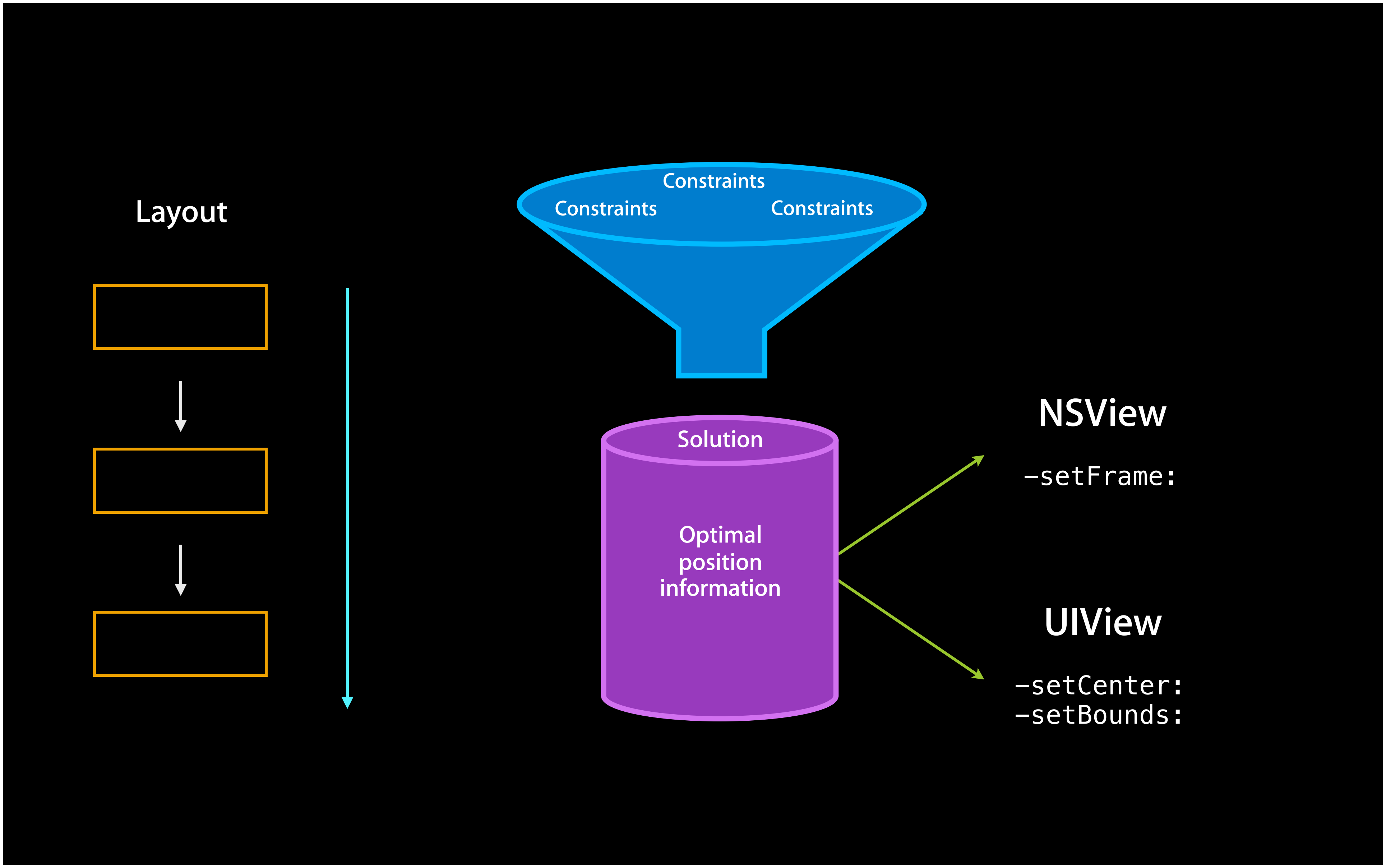


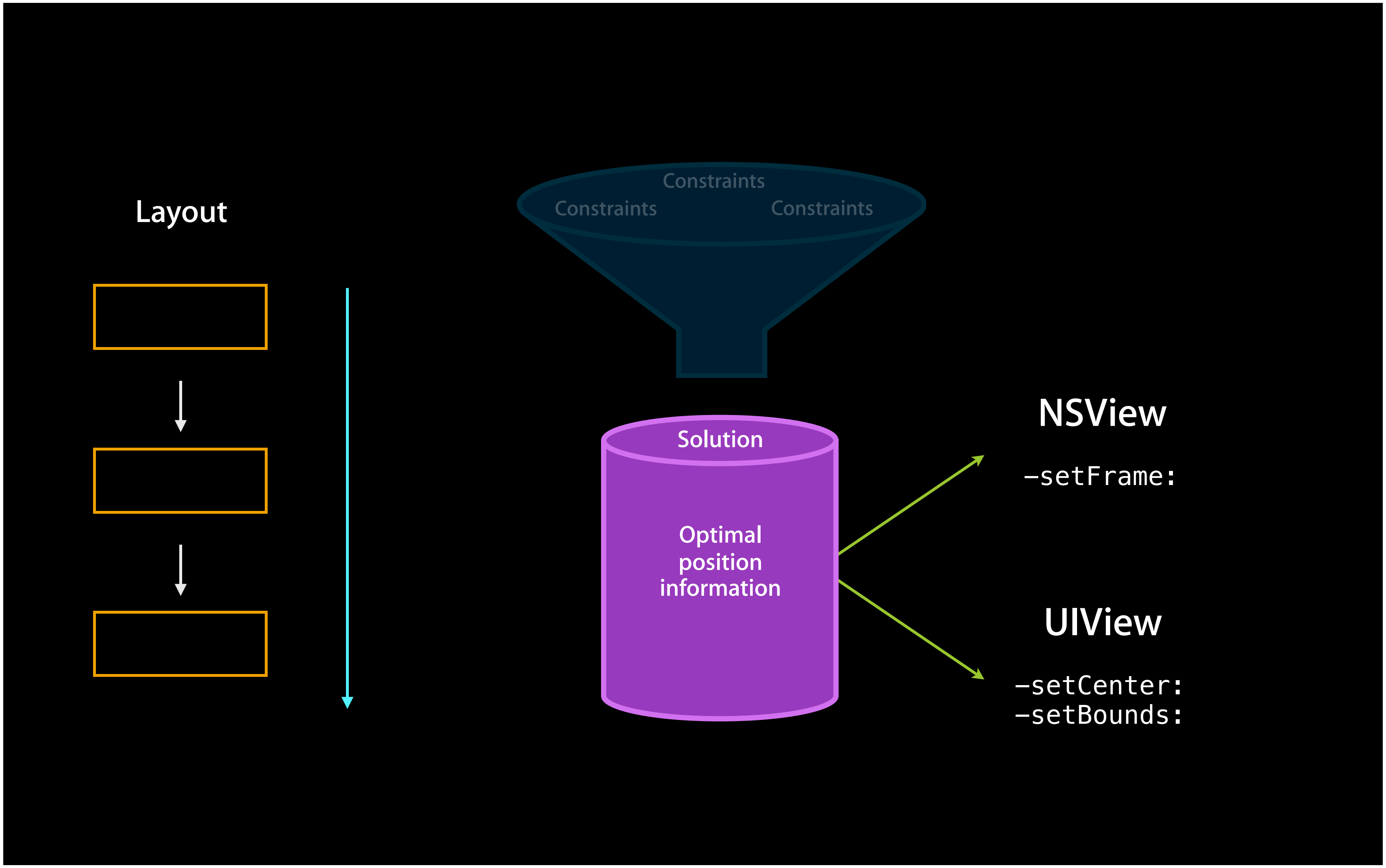
Phases of Display

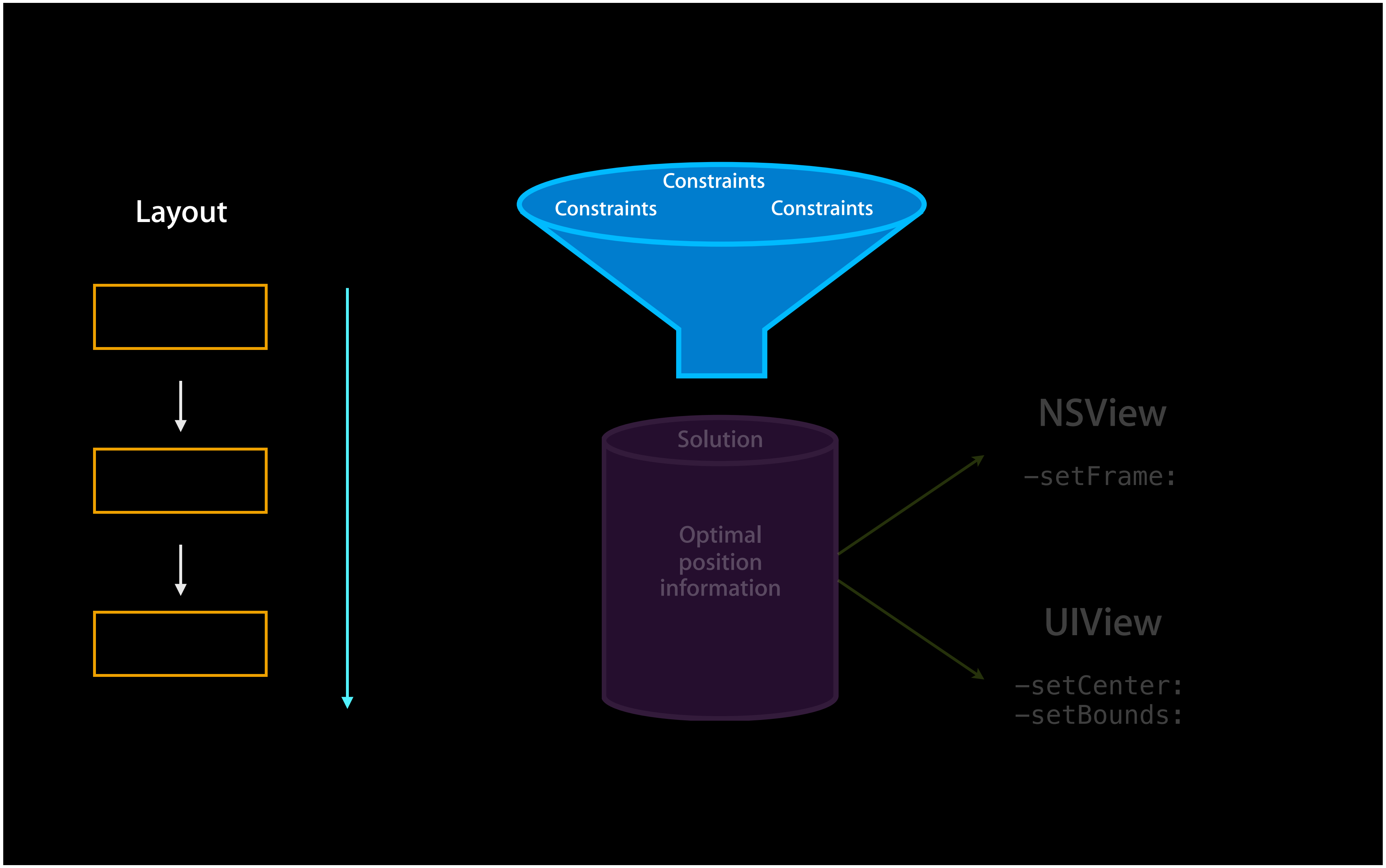


Layout




Layout
Constraints
Constraints
Constraints
Solution
Optimal position information
NSView
-setFrame:
UIView
-setCenter:
-setBounds:


Layout
Constraints
Constraints
Constraints
Solution
Optimal position information
NSView
-setFrame:
UIView
-setCenter:
-setBounds:


Layout
Constraints
Constraints
Constraints
Solution
Optimal position information
NSView
-setFrame:
UIView
-setCenter:
-setBounds:

[redacted]

The Constraints Must Be Sufficient

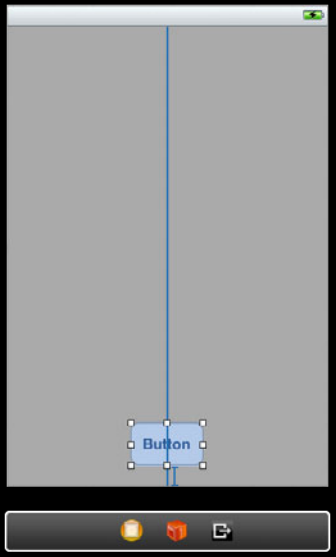
The Constraints Must Be Sufficient

The Constraints Must Not Conflict

What’s Going on Here?

- Button.centerX = Superview.centerX

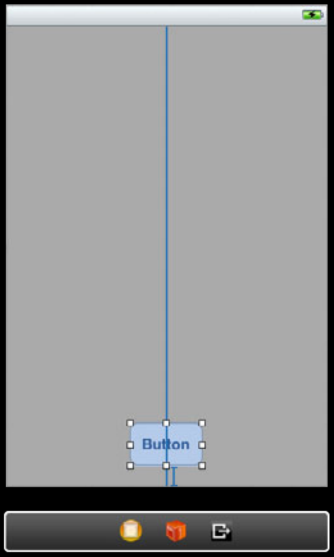
```
[NSLayoutConstraint constraintWithItem:button
                             attribute:NSLayoutAttributeCenterX
                             relatedBy:NSLayoutRelationEqual
                                toItem:superview
                             attribute:NSLayoutAttributeCenterX
                            multiplier:1.0
                              constant:0.0]
```



What’s Going on Here?

- Button.bottom = Superview.bottom – <padding>

```
[NSLayoutConstraint constraintWithItem:button
                             attribute:NSLayoutAttributeBottom
                             relatedBy:NSLayoutRelationEqual
                                toItem:superview
                             attribute:NSLayoutAttributeBottom
                            multiplier:1.0
                              constant:-padding]
```

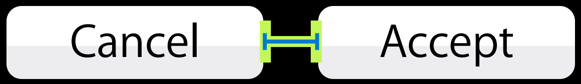


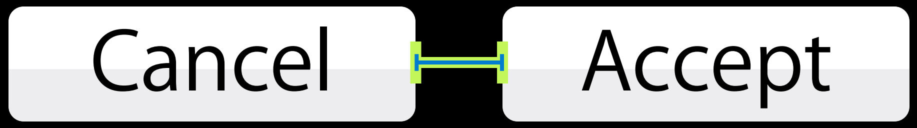
```
-intrinsicContentSize
```

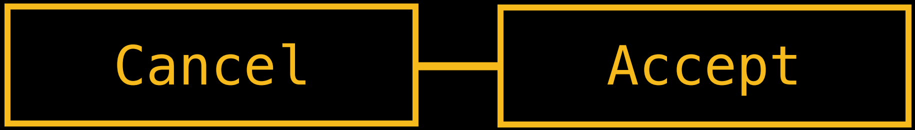
Agenda

- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

```
[NSLayoutConstraint constraintWithItem:acceptButton
                             attribute:NSLayoutAttributeLeft
                             relatedBy:NSLayoutRelationEquals
                                toItem:cancelButton
                             attribute:NSLayoutAttributeRight
                            multiplier:1.0
                              constant:12]
```




Cancel
Accept


Cancel
Accept

```
[cancelButton]-[acceptButton]
```

```
[NSLayoutConstraint constraintsWithVisualFormat:
        @”[cancelButton]-[acceptButton]”
 options:0 metrics:nil views:viewsDictionary];
```

```
[NSLayoutConstraint constraintsWithVisualFormat:
          @”[cancelButton]-[acceptButton]”
 options:0 metrics:nil views:viewsDictionary];
```

```
[NSLayoutConstraint constraintsWithVisualFormat:
        @"[cancelButton]-[acceptButton]"
  options:0 metrics:nil views:viewsDictionary];
```

```
UIButton *cancelButton = …
UIButton *acceptButton = …
viewsDictionary =
   NSDictionaryOfVariableBindings(cancelButton,acceptButton);
```

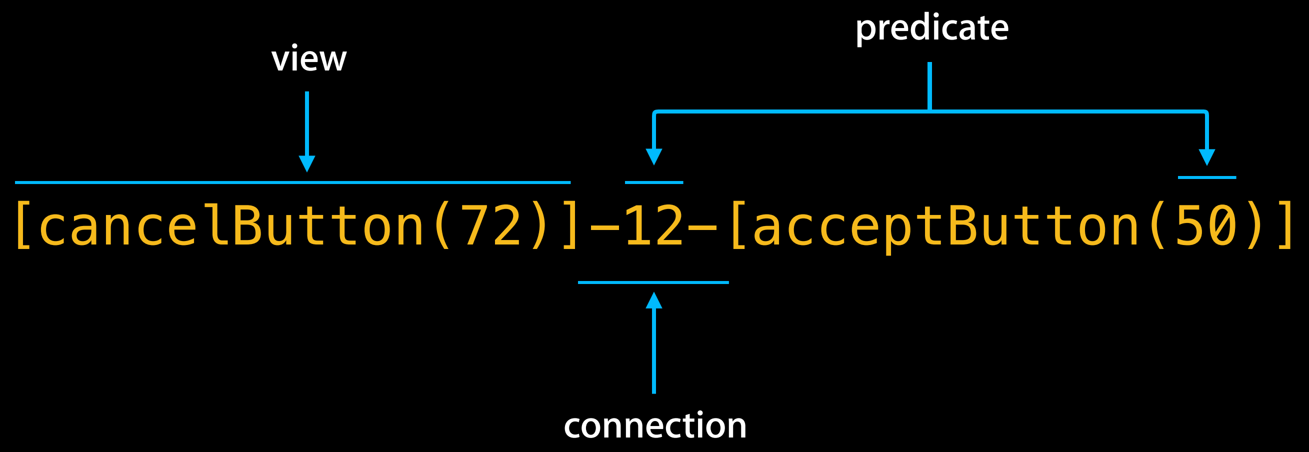
```
(lldb) po viewsDictionary
{
    acceptButton = "<UIButton: 0x4004c0>";
    cancelButton = "<UIButton: 0x4004ab>";
}
```

```
[cancelButton]-[acceptButton]
```

```
[cancelButton    ]-  -[acceptButton    ]
```

```
[cancelButton(72)]-12-[acceptButton(50)]
```

```
[cancelButton(72)]-12-[acceptButton(50)]
```

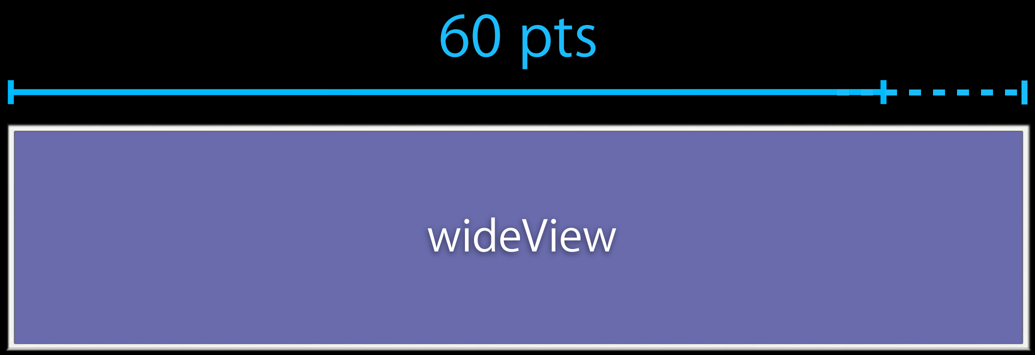


```
[cancelButton(72)]-12-[acceptButton(50)]
```

Visual Format Language Examples

Inequality, Priority	[wideView(>=60@700)]
Vertical: Flush Views, Equal Heights	V:[redBox][yellowBox(==redBox)]
Combination	H:|-[Find]-[FindNext]-[FindField(>=20)]-|

[wideView(>=60@700)]



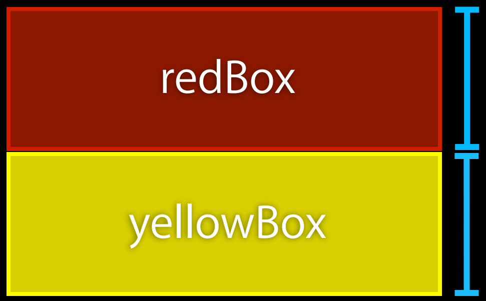
Visual Format Language Examples

Inequality, Priority	[wideView(>=60@700)]
Vertical: Flush Views, Equal Heights	V:[redBox][yellowBox(==redBox)]
Combination	H:|-[Find]-[FindNext]-[FindField(>=20)]-|

Visual Format Language Examples

Inequality, Priority	[wideView(>=60@700)]
Vertical: Flush Views, Equal Heights	V:[redBox][yellowBox(==redBox)]
Combination	H:|-[Find]-[FindNext]-[FindField(>=20)]-|

V:[redBox][yellowBox(==redBox)]



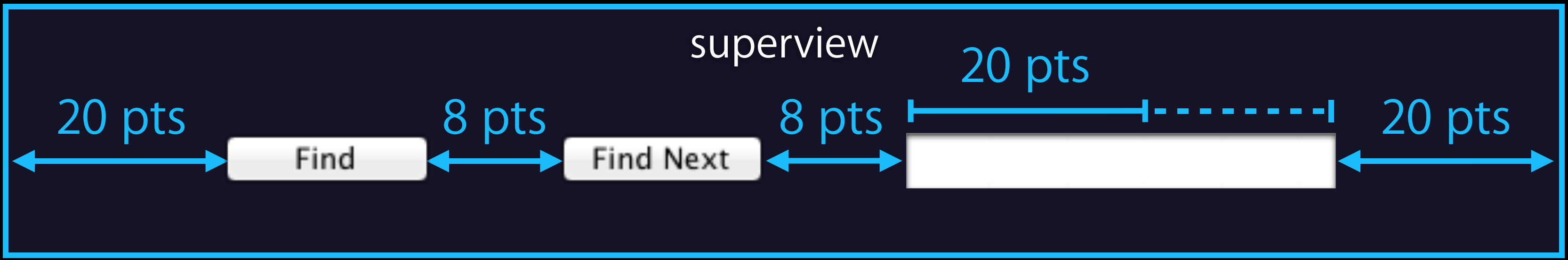
Visual Format Language Examples

Inequality, Priority	[wideView(>=60@700)]
Vertical: Flush Views, Equal Heights	V:[redBox][yellowBox(==redBox)]
Combination	H:|-[Find]-[FindNext]-[FindField(>=20)]-|

Visual Format Language Examples

Inequality, Priority	[wideView(>=60@700)]
Vertical: Flush Views, Equal Heights	V:[redBox][yellowBox(==redBox)]
Combination	H:|-[Find]-[FindNext]-[FindField(>=20)]-|

H:|-[Find]-[FindNext]-[FindField(>=20)]-|



```
[NSLayoutConstraint constraintsWithVisualFormat:
  @”H:|-[Find]-[FindNext]-[FindField(>=20)]-|”
     options:NSLayoutFormatAlignAllBaseline
      metrics:nil views:viewsDictionary];
```

Agenda

- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

The constraints must be sufficient

The constraints must not conflict

✗ Ambiguous Layout

The constraints must not conflict

✗ Ambiguous Layout

✗ Unsatisfiable Constraints

Demo

Ambiguous and unsatisfiable constraints

Agenda

- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

What About Compatibility?

What About Compatibility?

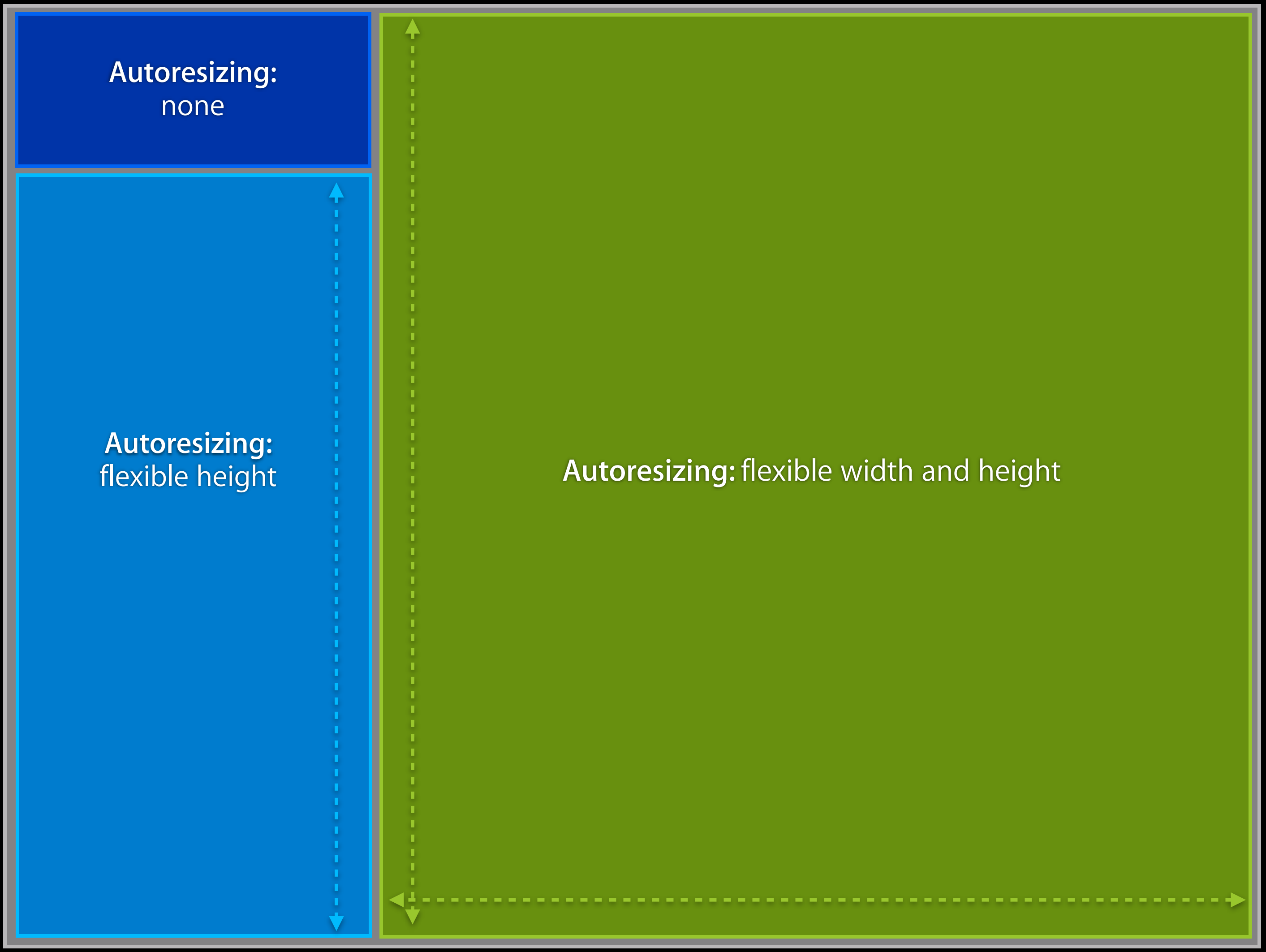
- I’m ready to convert to Auto Layout!

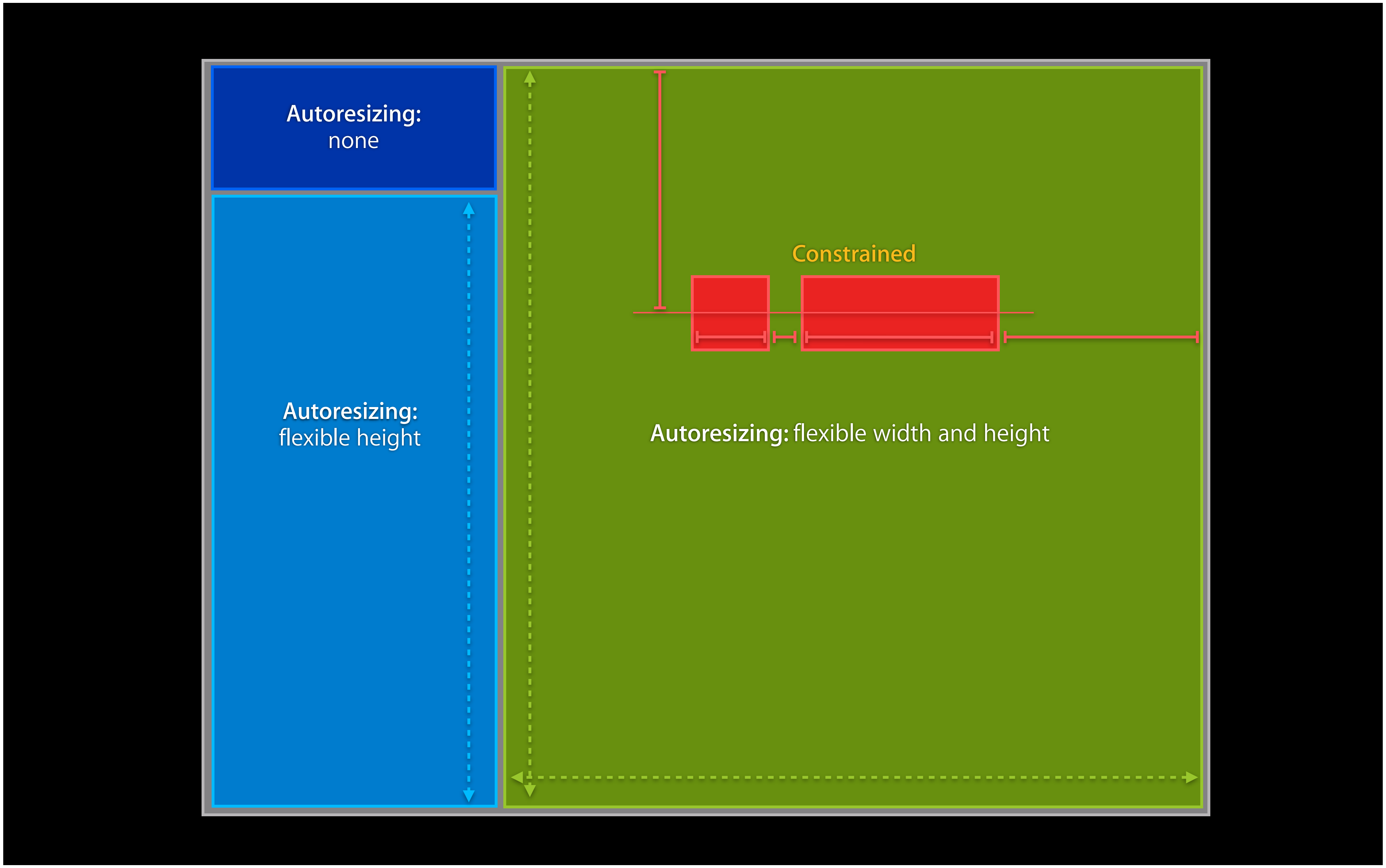
What About Compatibility?

- I’m not ready to convert—of course, I will be soon, but not quite yet

What About Compatibility?

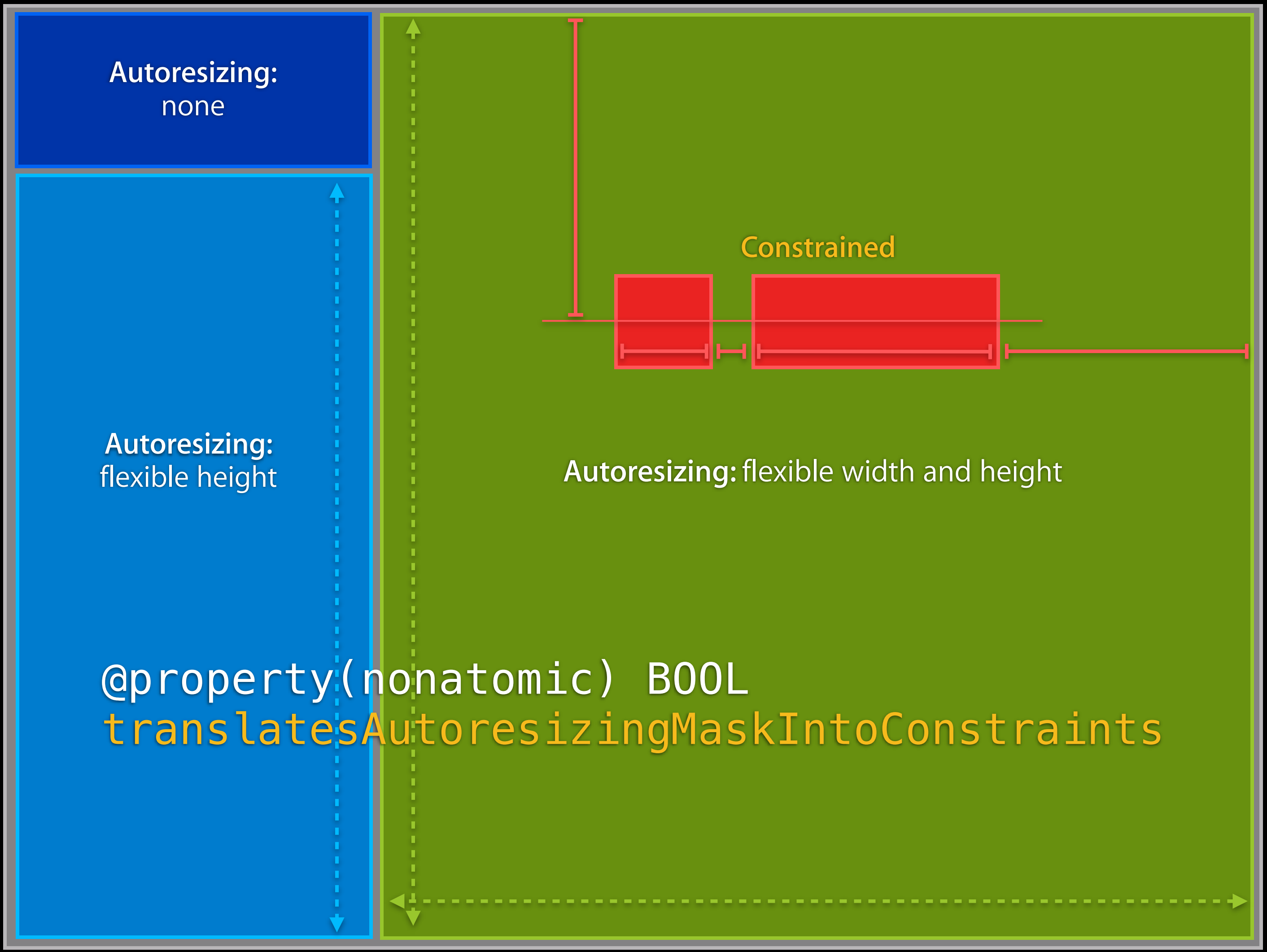
- I want to use Auto Layout in part of my application


Autoresizing: none
Autoresizing: flexible height
Autoresizing: flexible width and height

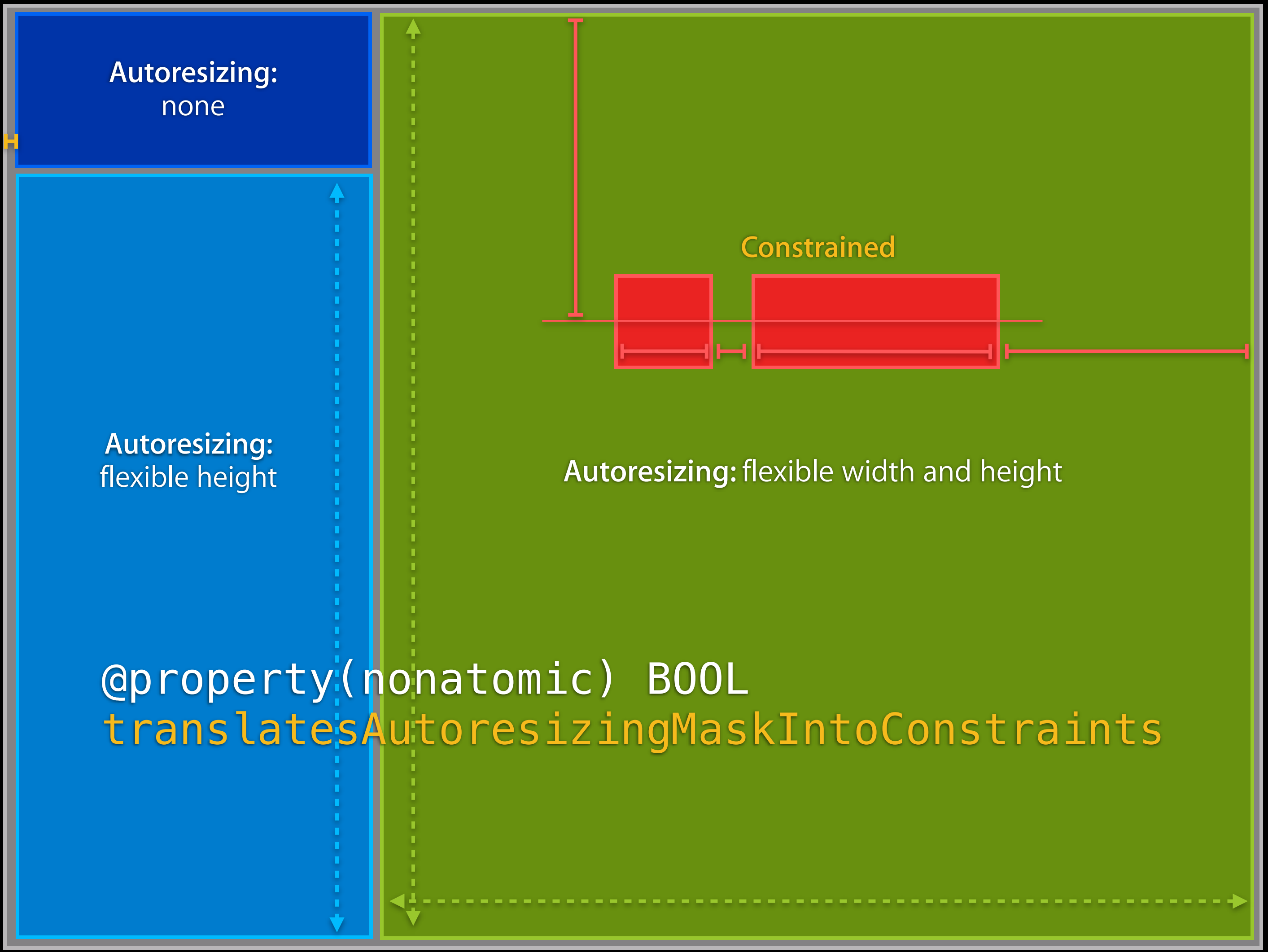

Autoresizing:
none
Autoresizing:
flexible height
Constrained
Autoresizing: flexible width and height

Autoresizing:
none
Constrained

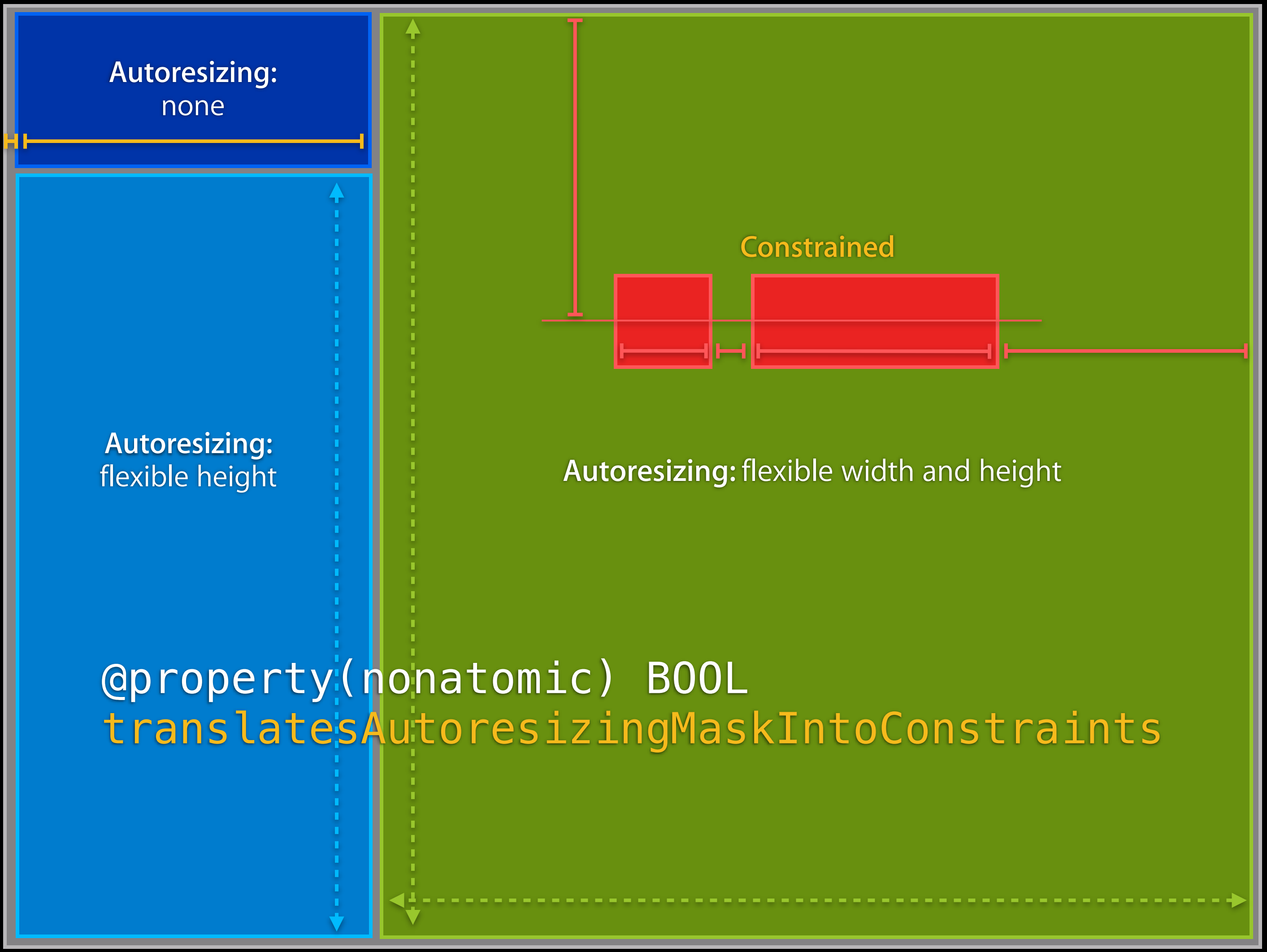
```
@property(nonatomic) BOOL
translatesAutoresizingMaskIntoConstraints
```



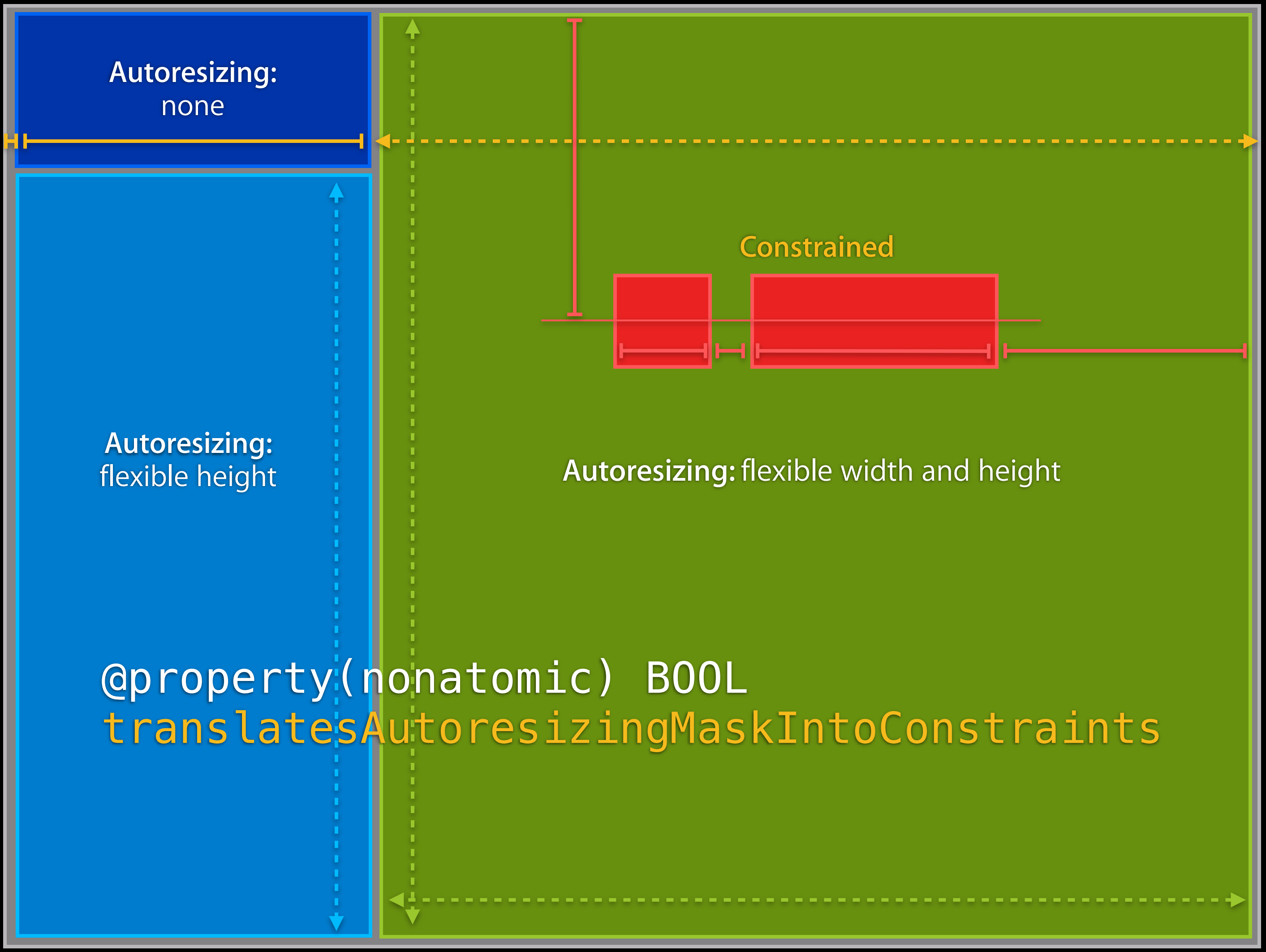
```
@property(nonatomic) BOOL
translatesAutoresizingMaskIntoConstraints
```


Autoresizing: none
Autoresizing: flexible height
Constrained
Autoresizing: flexible width and height

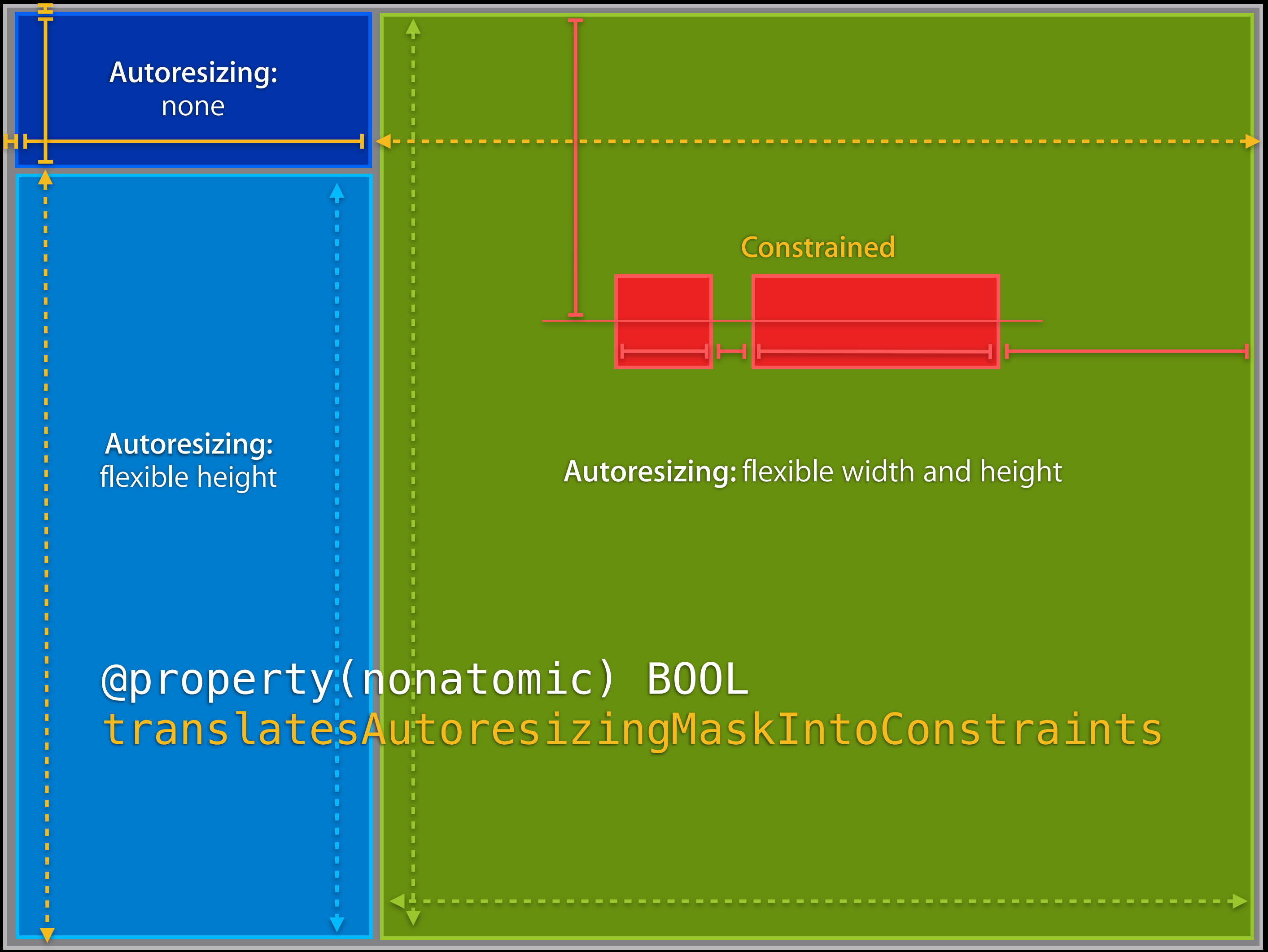
```
@property(nonatomic) BOOL
translatesAutoresizingMaskIntoConstraints
```


Autoresizing: none
Autoresizing: flexible height
Constrained
Autoresizing: flexible width and height

```
@property(nonatomic) BOOL
translatesAutoresizingMaskIntoConstraints
```


Autoresizing: none
Autoresizing: flexible height
Constrained
Autoresizing: flexible width and height

```
@property(nonatomic) BOOL
translatesAutoresizingMaskIntoConstraints
```



```
@property(nonatomic) BOOL
translatesAutoresizingMaskIntoConstraints
```

```
[button setTranslatesAutoresizingMaskIntoConstraints:NO];
```

Interface Builder Takes Care of This for You ✓

```
[button setTranslatesAutoresizingMaskIntoConstraints:NO];
```

Recap

- Setting up Constraint-Based Layout
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

Recap

- Setting up Constraint-Based Layout ✓
- Layout Behind the Scenes
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

Recap

- Setting up Constraint-Based Layout ✓
- Layout Behind the Scenes ✓
- The Visual Format Language
- Things That Can Go Wrong
- Compatibility

Recap

- Setting up Constraint-Based Layout ✓
- Layout Behind the Scenes ✓
- The Visual Format Language ✓
- Things That Can Go Wrong
- Compatibility

Recap

- Setting up Constraint-Based Layout ✓
- Layout Behind the Scenes ✓
- The Visual Format Language ✓
- Things That Can Go Wrong ✓
- Compatibility

Recap

- Setting up Constraint-Based Layout ✓
- Layout Behind the Scenes ✓
- The Visual Format Language ✓
- Things That Can Go Wrong ✓
- Compatibility ✓

Summary

Summary

- Elementary API

Summary

Elementary API

```
+[NSLayoutConstraint constraintWithItem:
                              attribute:
                              relatedBy:
                                 toItem:
                              attribute:
                             multiplier:
                               constant:]

        -[NSView
          UIView addConstraint:]
```

Summary

- Elementary API
- Visual Format Language

Summary

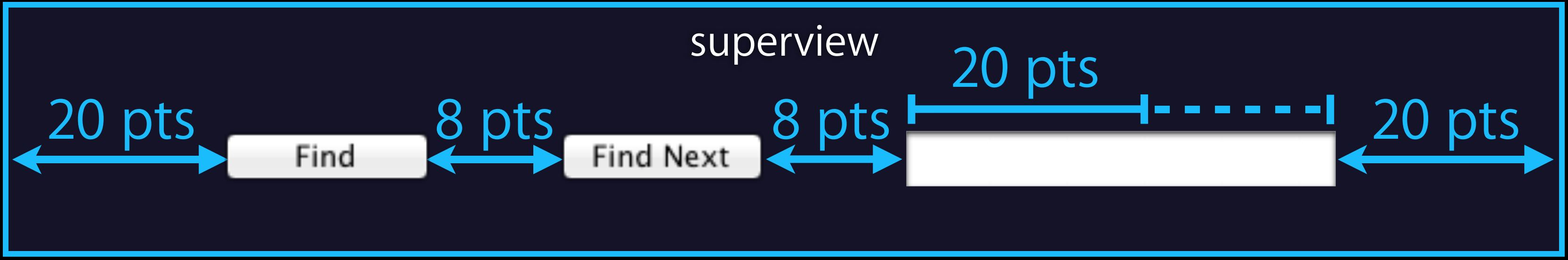
Visual Format Language

```
+[NSLayoutConstraint constraintsWithVisualFormat:
                                          options:
                                          metrics:
                                  viewsDictionary:]
```

Summary

Visual Format Language

H:|-[Find]-[FindNext]-[FindField(>=20)]-|



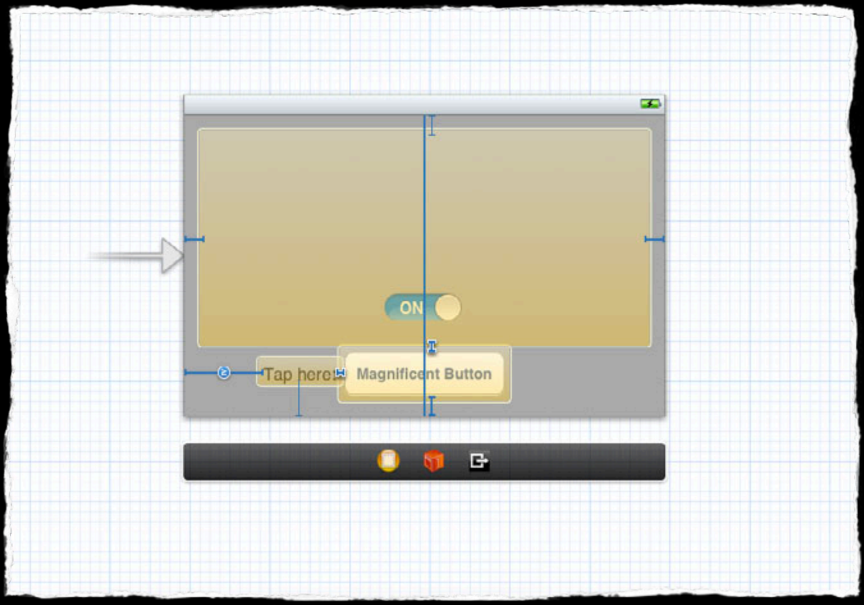
```
+[NSLayoutConstraint constraintsWithVisualFormat:
                                         options:
                                         metrics:
                                  viewsDictionary:]
```

Summary

- Elementary API
- Visual Format Language
- Interface Builder

Summary

Interface Builder gives you power



Summary

- Elementary API
- Visual Format Language
- Interface Builder Gives You Power

Related Sessions

Best Practices for Mastering Auto Layout	Mission Thursday 9:00AM
Auto Layout by Example	Mission Thursday 11:30AM
The Evolution of View Controllers on iOS	Mission Thursday 2:00PM

More Information

Jake Behrens
UI Frameworks Evangelist
behrens@apple.com

Auto Layout Documentation, Sample Code, and Release Notes
Log in and search for "Auto Layout" at developer.apple.com
https://developer.apple.com/search/index.php?q=auto+layout

Cocoa Auto Layout
Session from WWDC 2011
https://developer.apple.com/videos/wwdc/2011/?id=103

Programming with Constraints
The Cassowary Linear Arithmetic Constraint Solving Algorithm
http://www.cs.washington.edu/research/constraints/cassowary/

Apple Developer Forums
http://devforums.apple.com

Labs

Auto Layout Lab	Essentials Lab A Tuesday 2:00PM
Auto Layout Lab	App Services Lab B Thursday 2:00PM

WWDC2012

[redacted]

[redacted]

[redacted]