Auto Layout

Thinking with Constraints

Kyle Sluder

The Omni Group

Twitter: @optshiftk

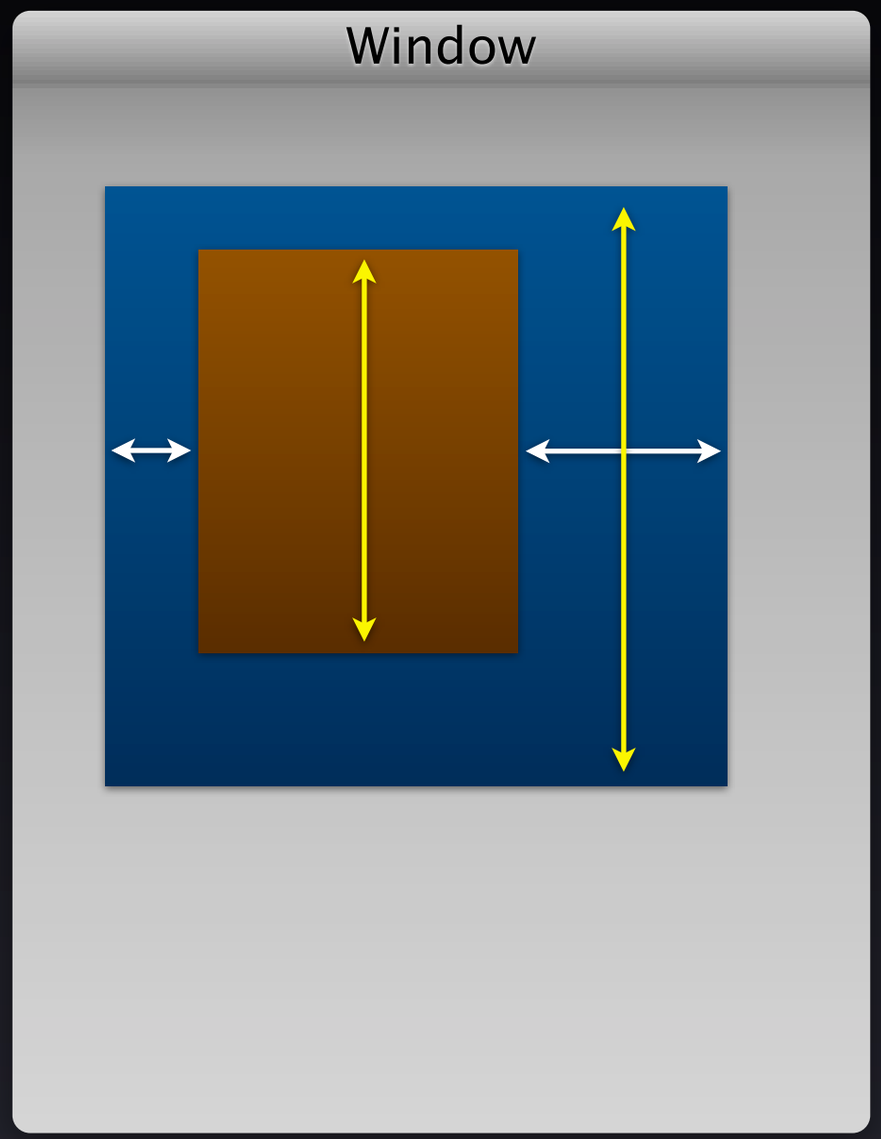
optshiftk@optshiftk.com

What is Auto Layout?

- Better than springs & struts
- Available in 10.7+ and iOS 6
- Defines views' frames by their contents and relationships to each other
- Dynamically recalculates interface
- Encourages single-pass layout phase over ad-hoc layout

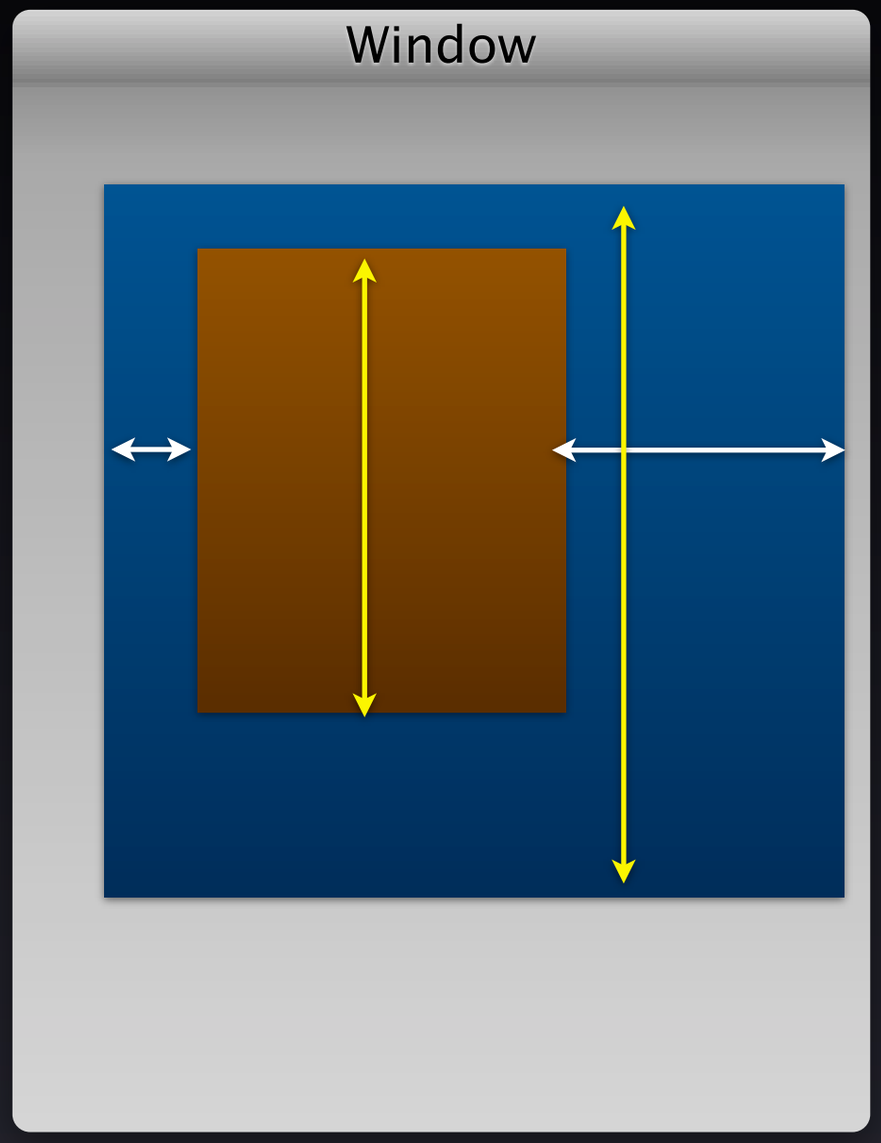
The Old Way: Springs & Struts

- Manually resize views as necessary
- Subviews resize whenever superview resizes
- Autoresizing mask dictates behavior



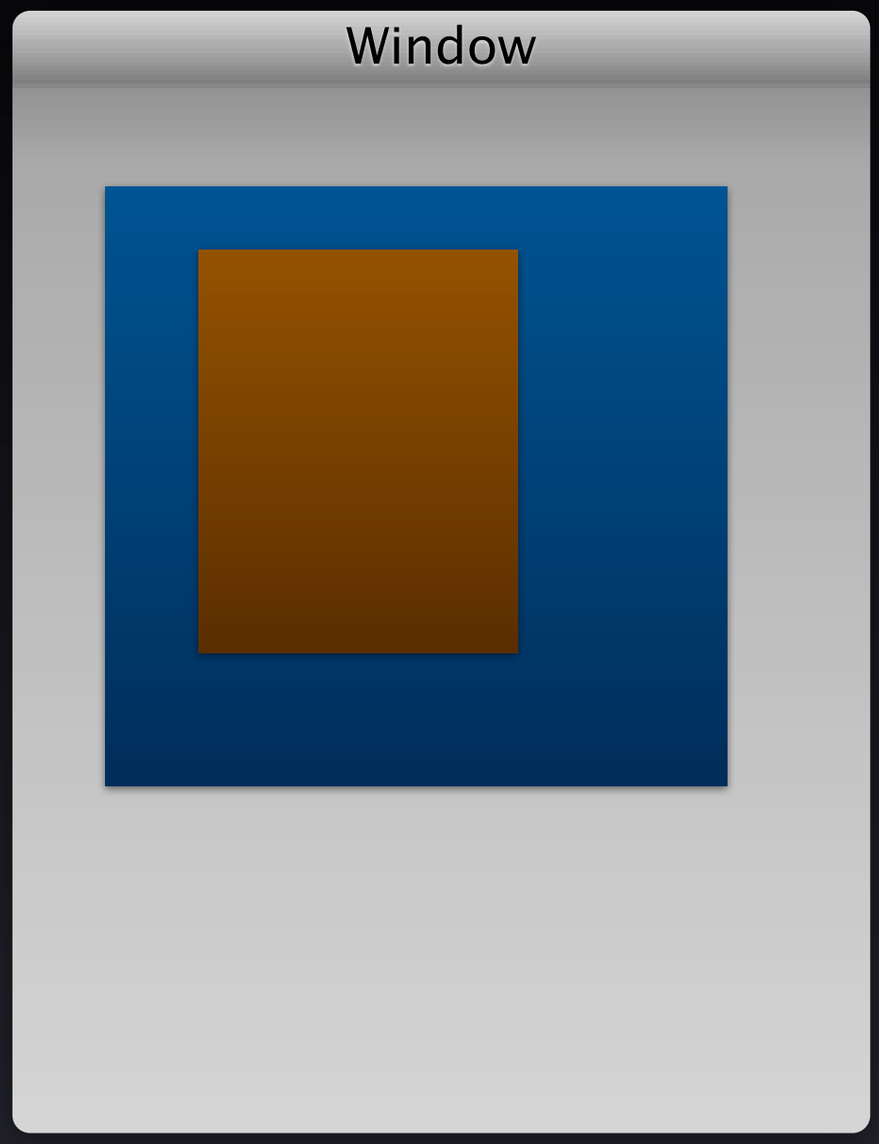
The Old Way: Springs & Struts

- Manually resize views as necessary
- Subviews resize whenever superview resizes
- Autoresizing mask dictates behavior



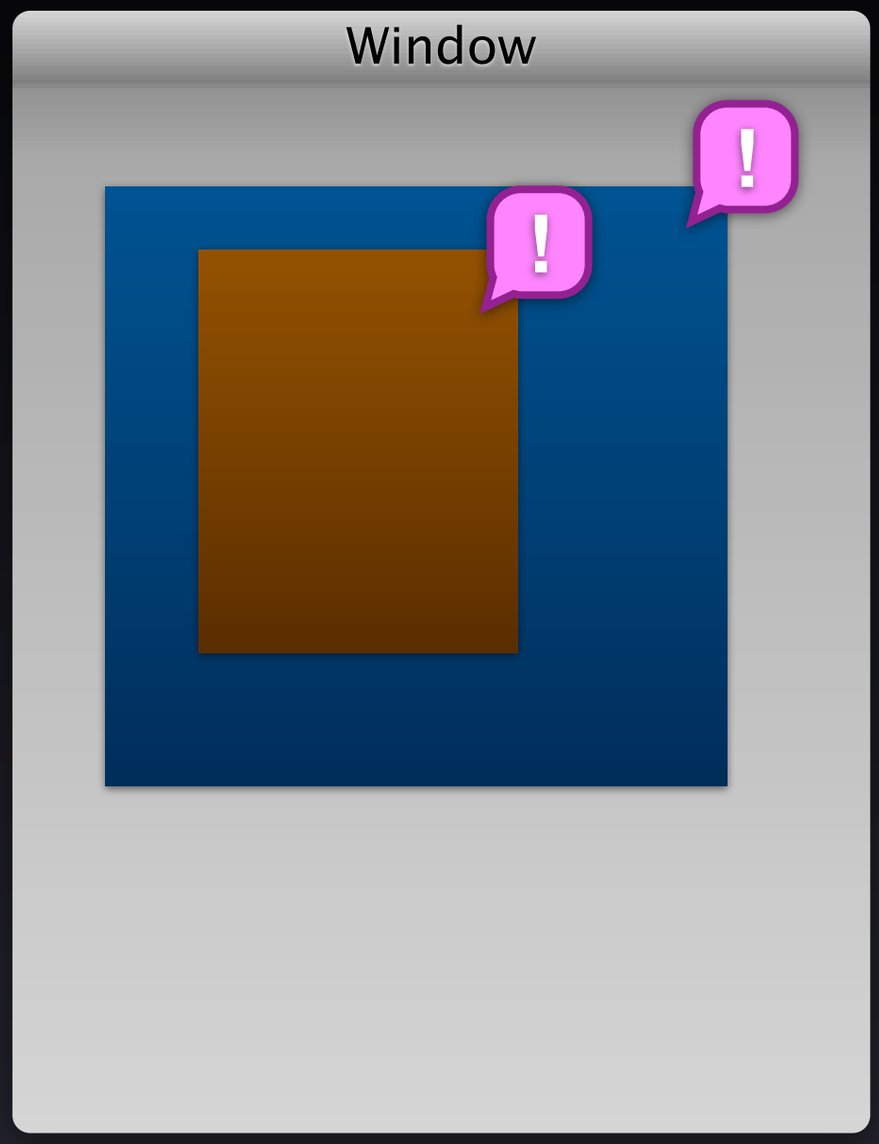
How Auto Layout Works

1) Install constraints on views



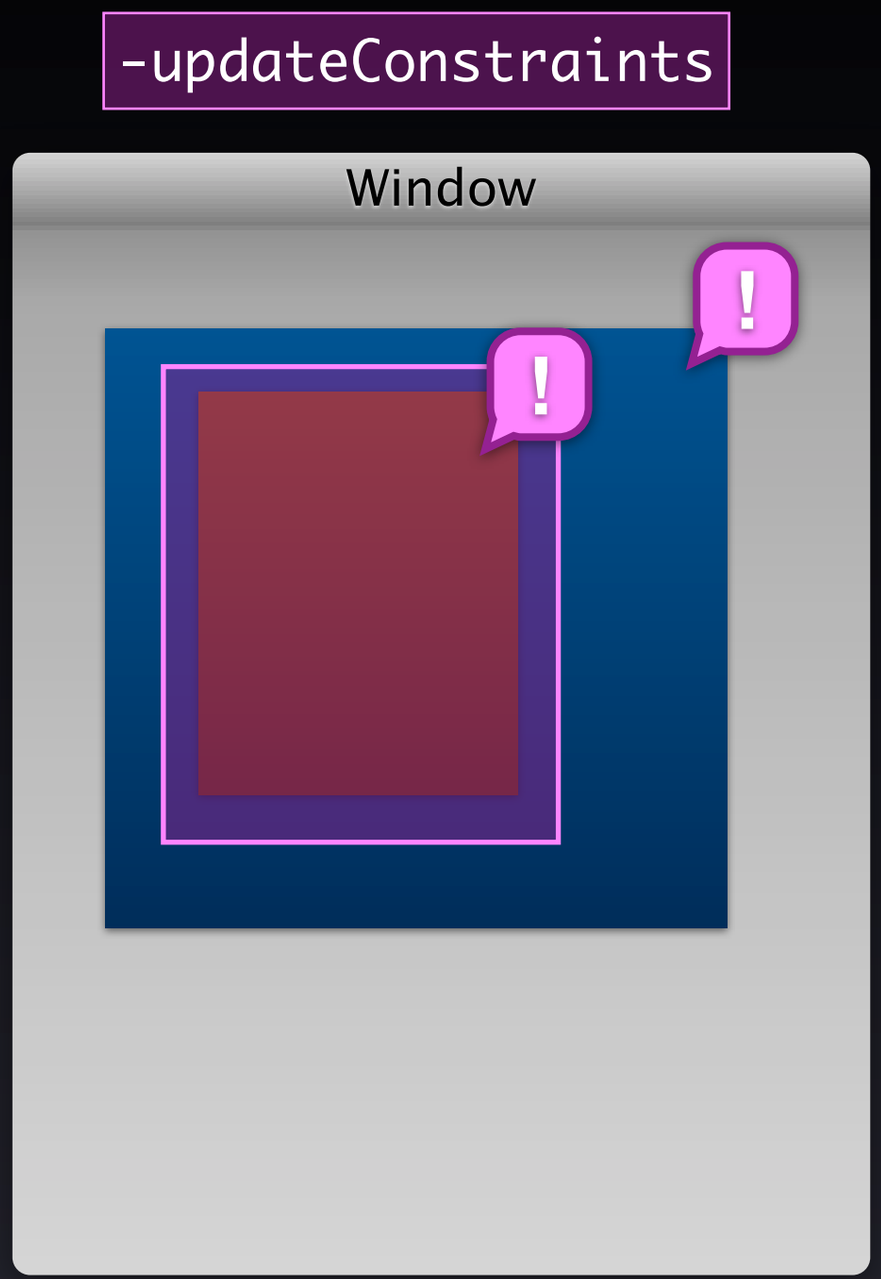
How Auto Layout Works

1) Install constraints on views



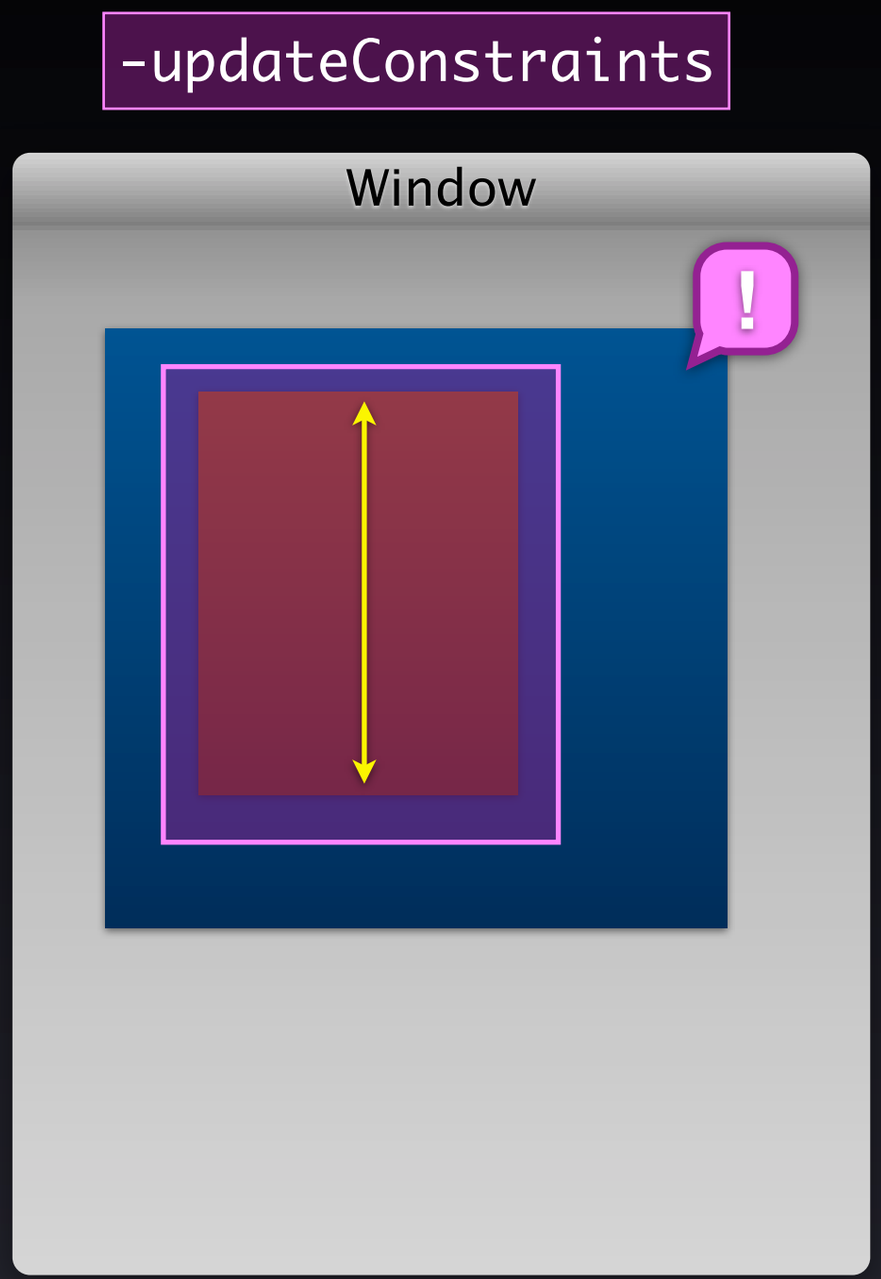
How Auto Layout Works

1) Install constraints on views



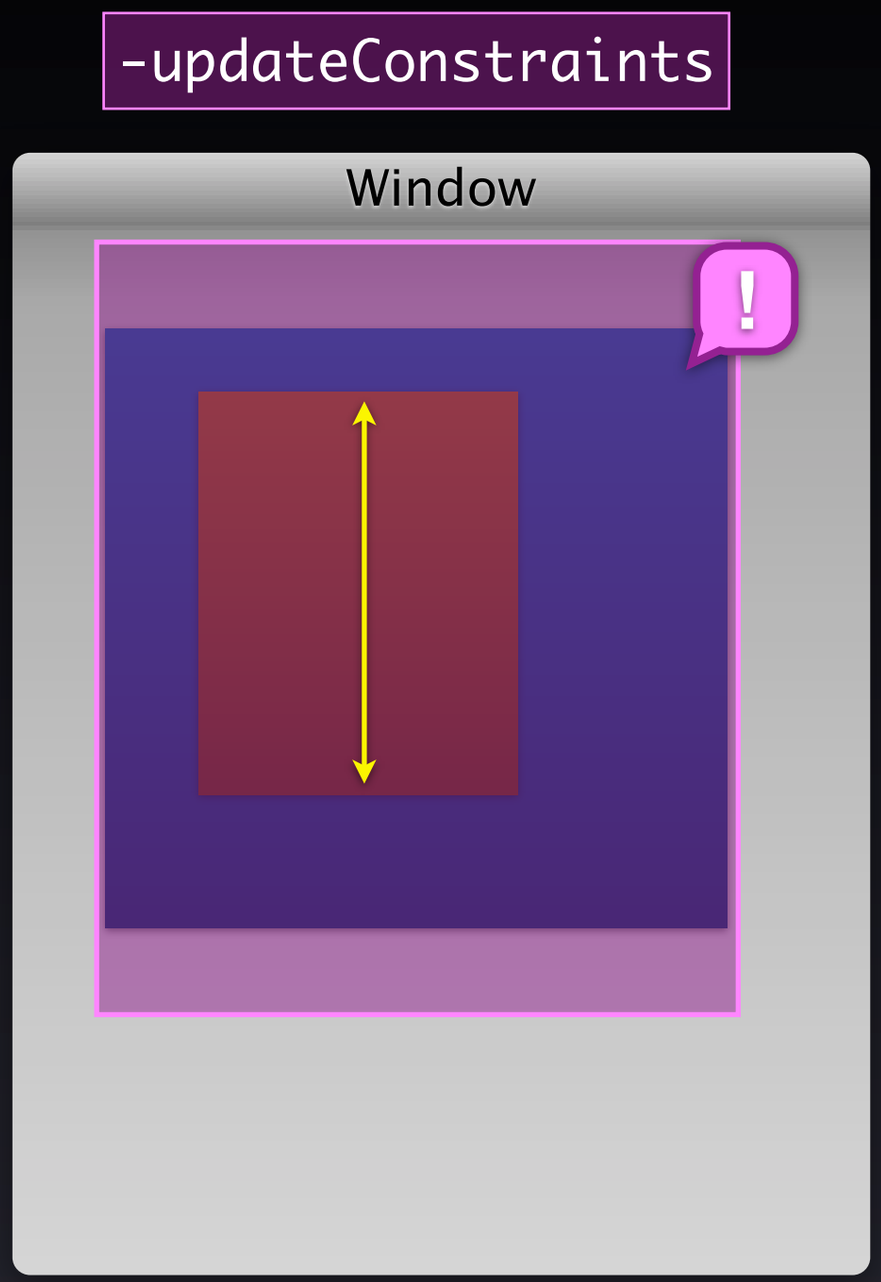
How Auto Layout Works

1) Install constraints on views



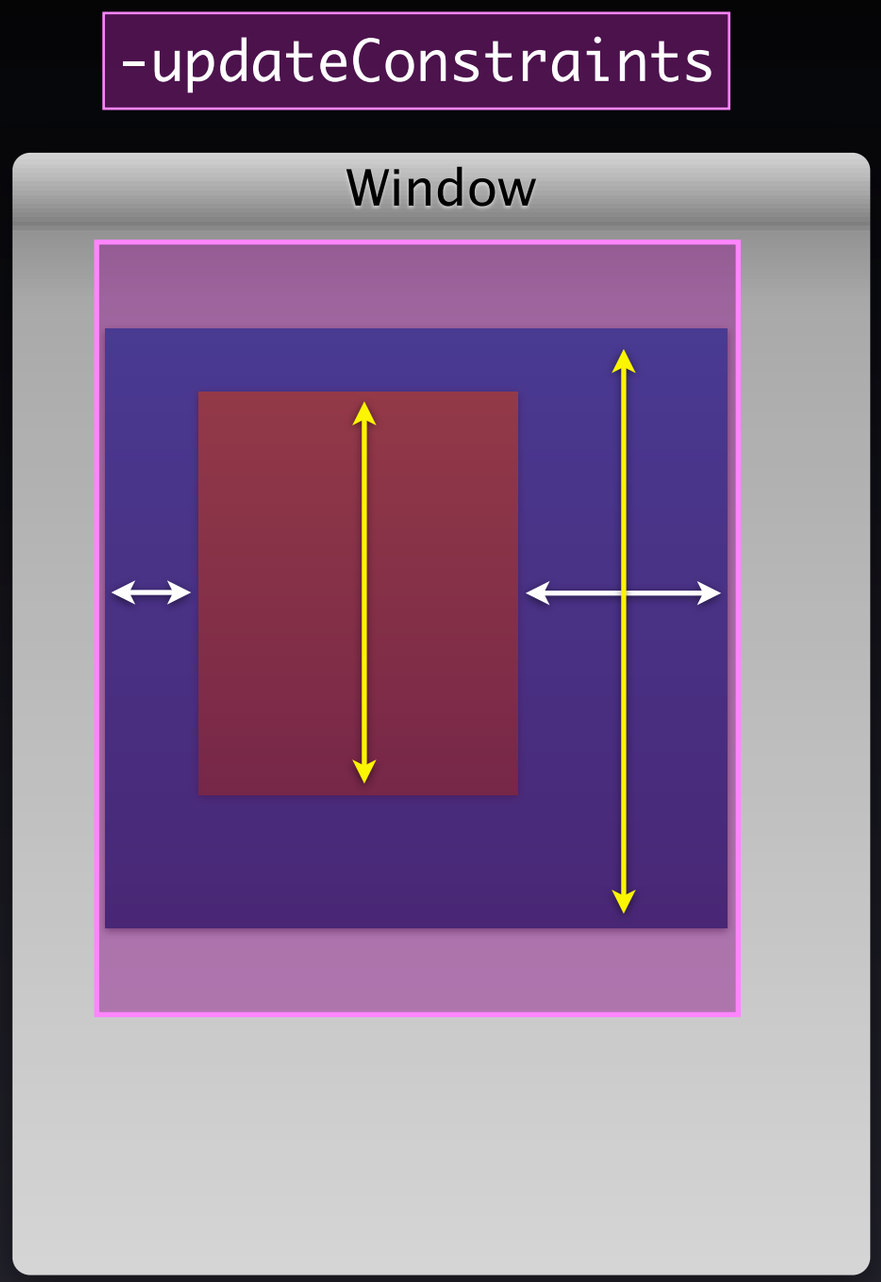
How Auto Layout Works

1) Install constraints on views



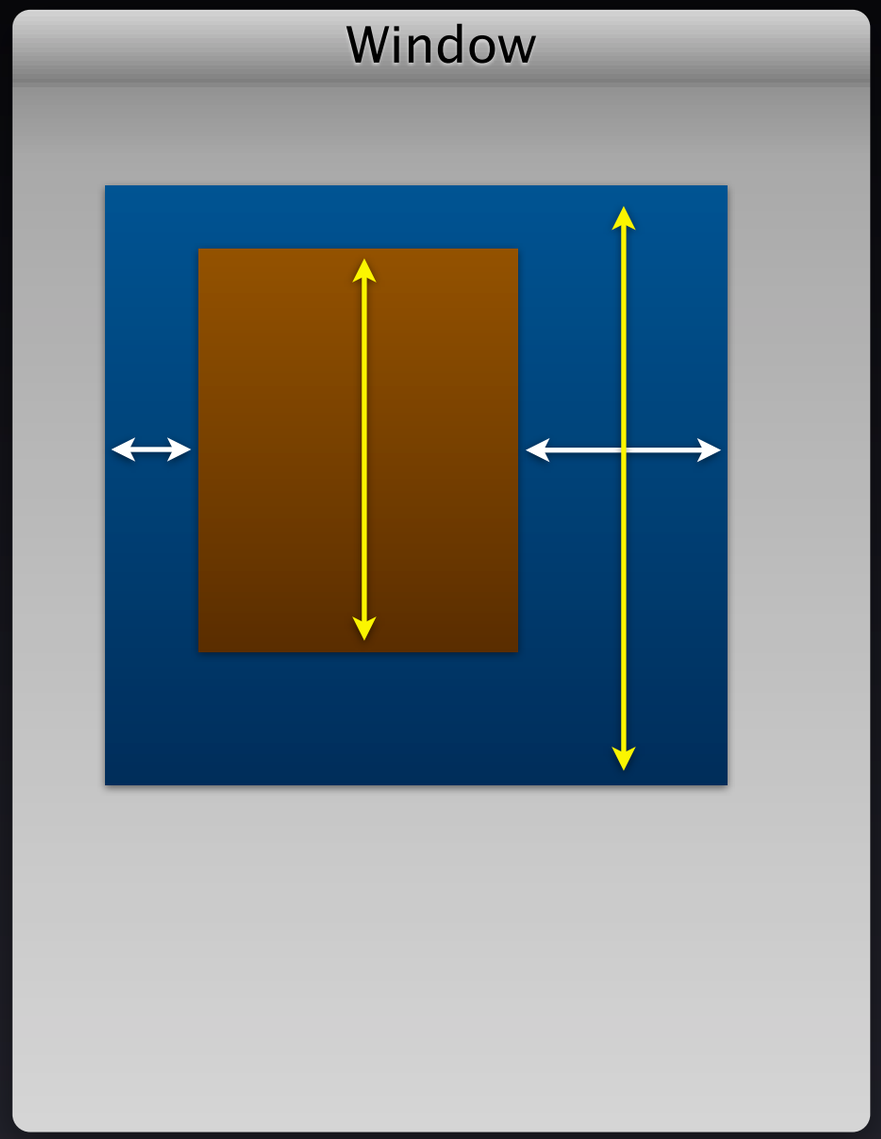
How Auto Layout Works

1) Install constraints on views



How Auto Layout Works

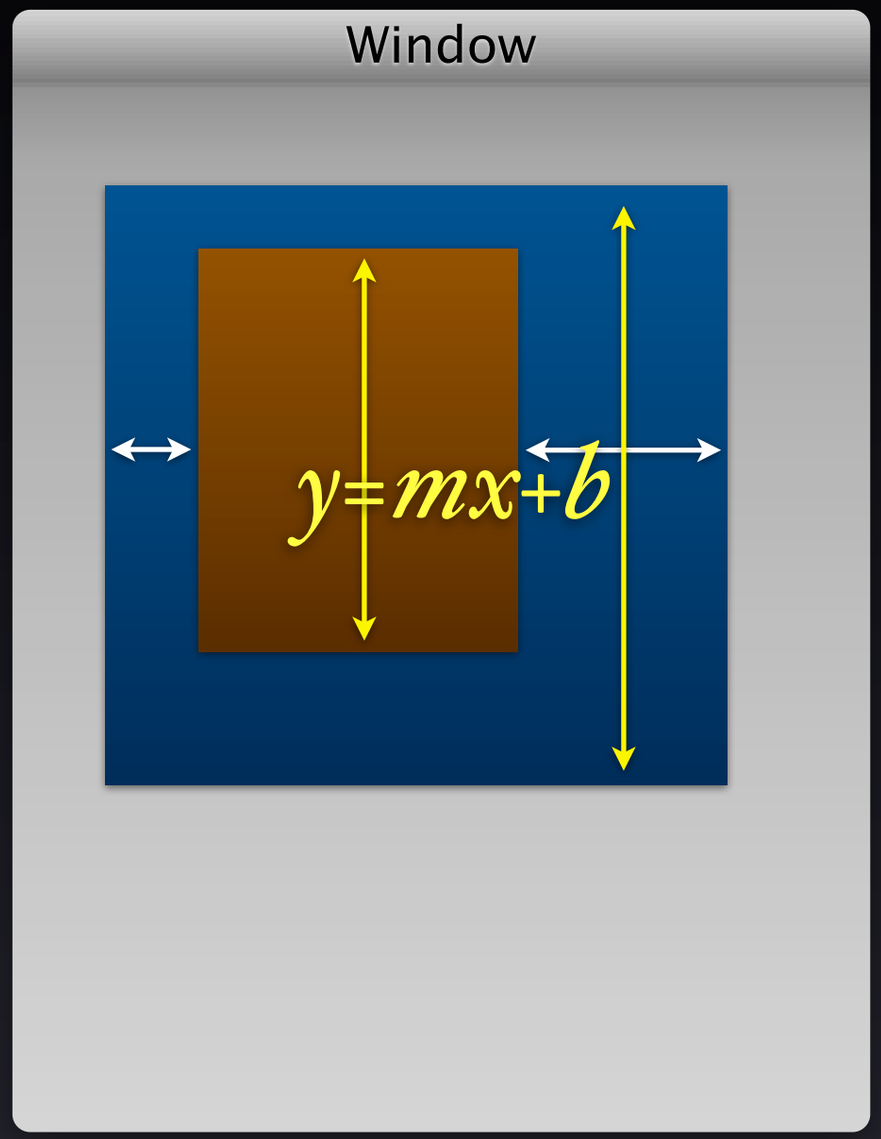
1) Install constraints on views



How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

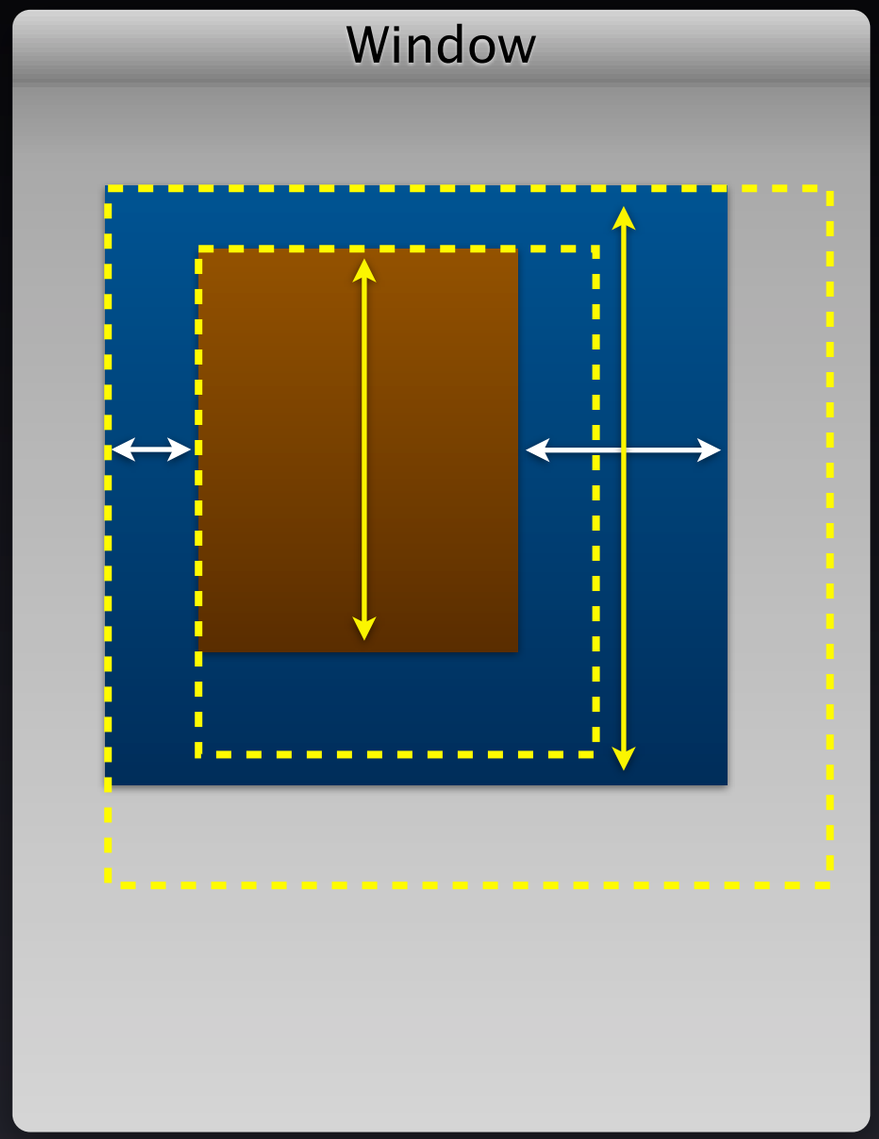


How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

3) System of inequalities is solved all at once



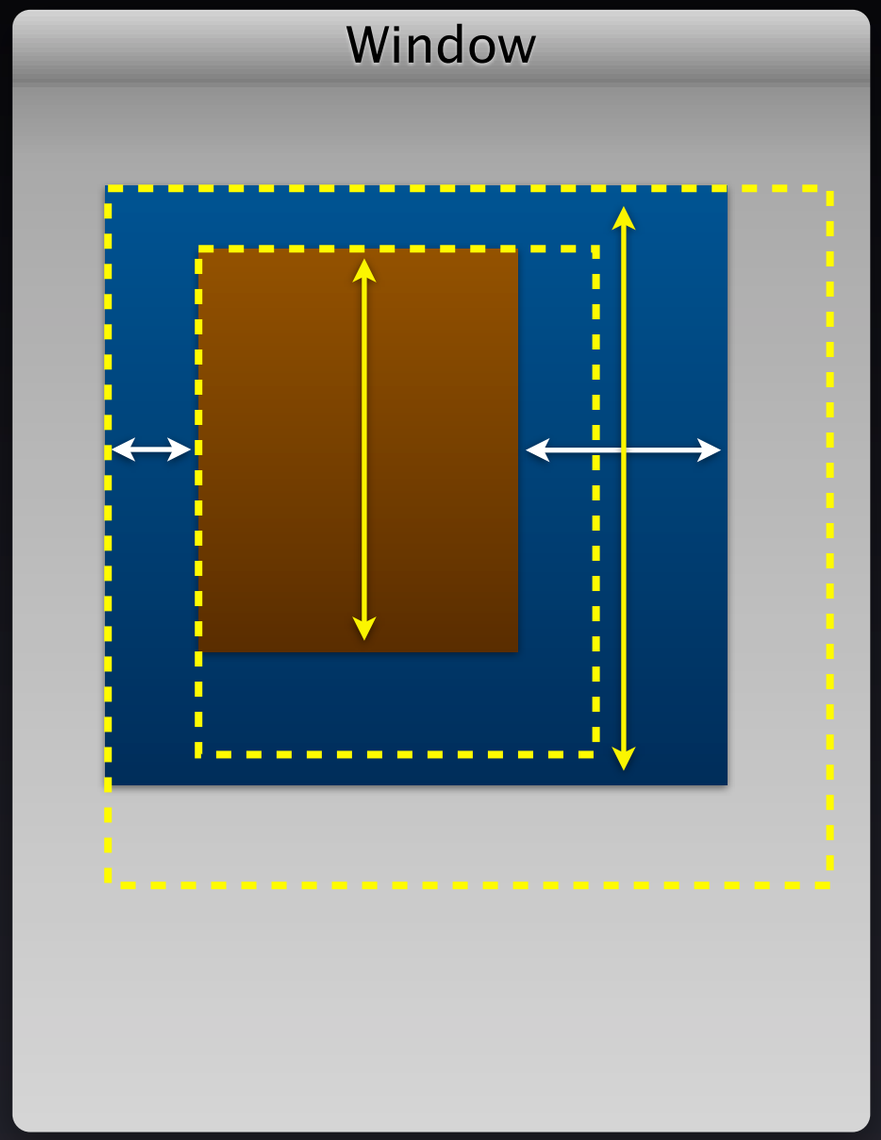
How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

3) System of inequalities is solved all at once

4) Views' frames are updated in -layout



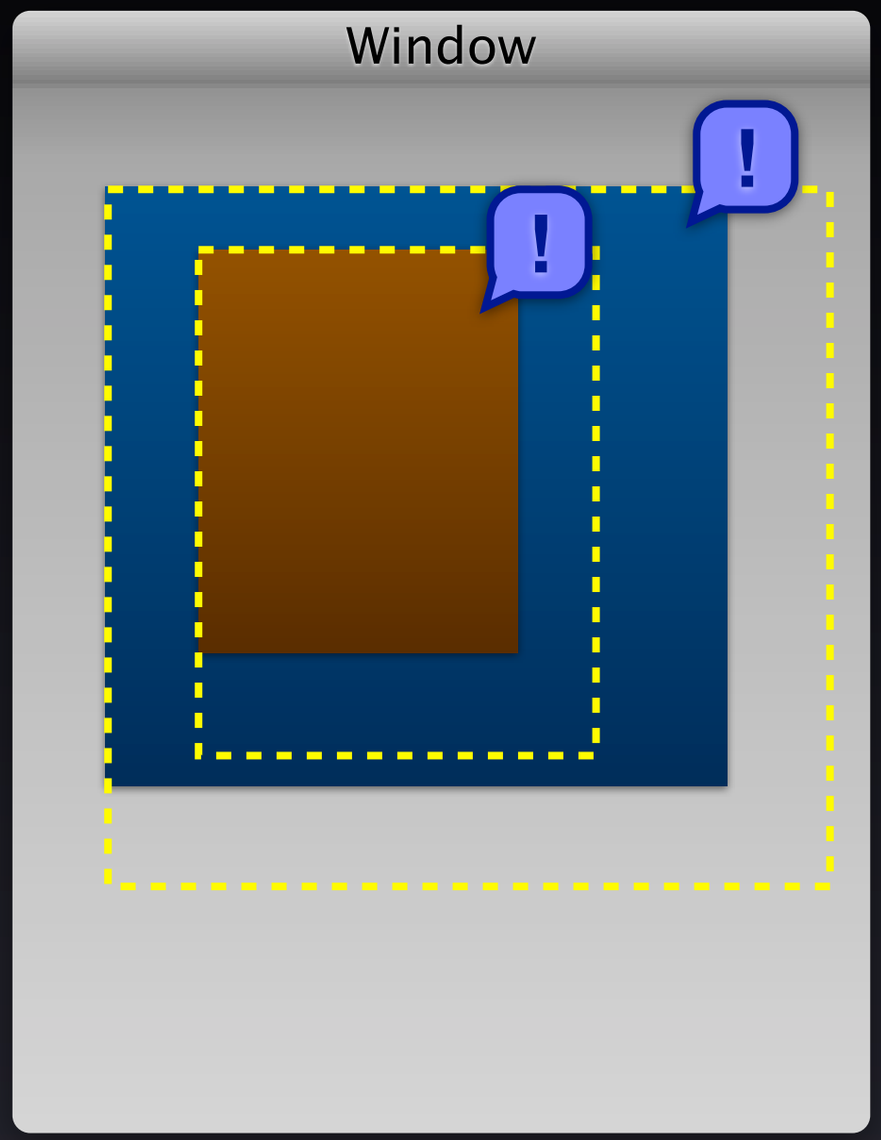
How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

3) System of inequalities is solved all at once

4) Views' frames are updated in -layout



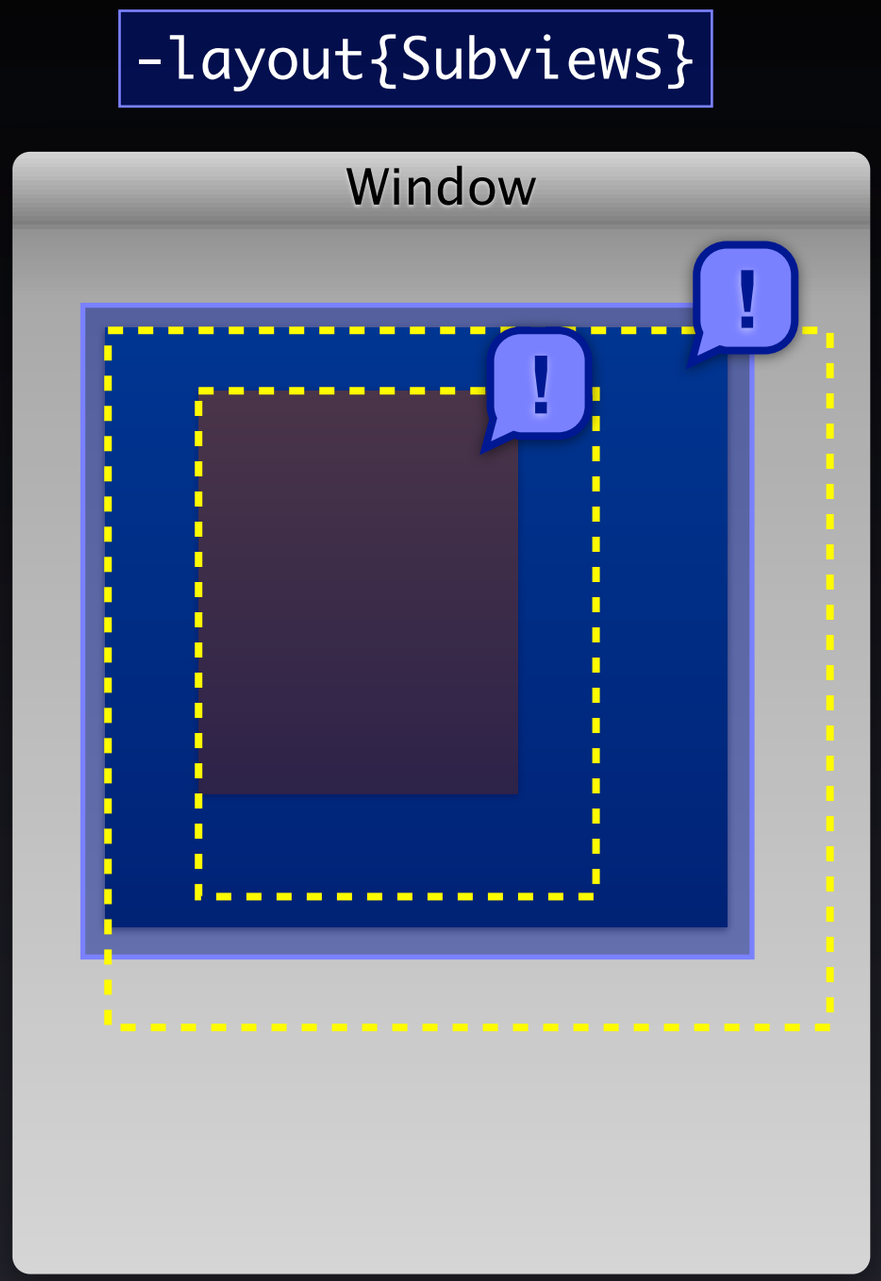
How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

3) System of inequalities is solved all at once

4) Views' frames are updated in -layout



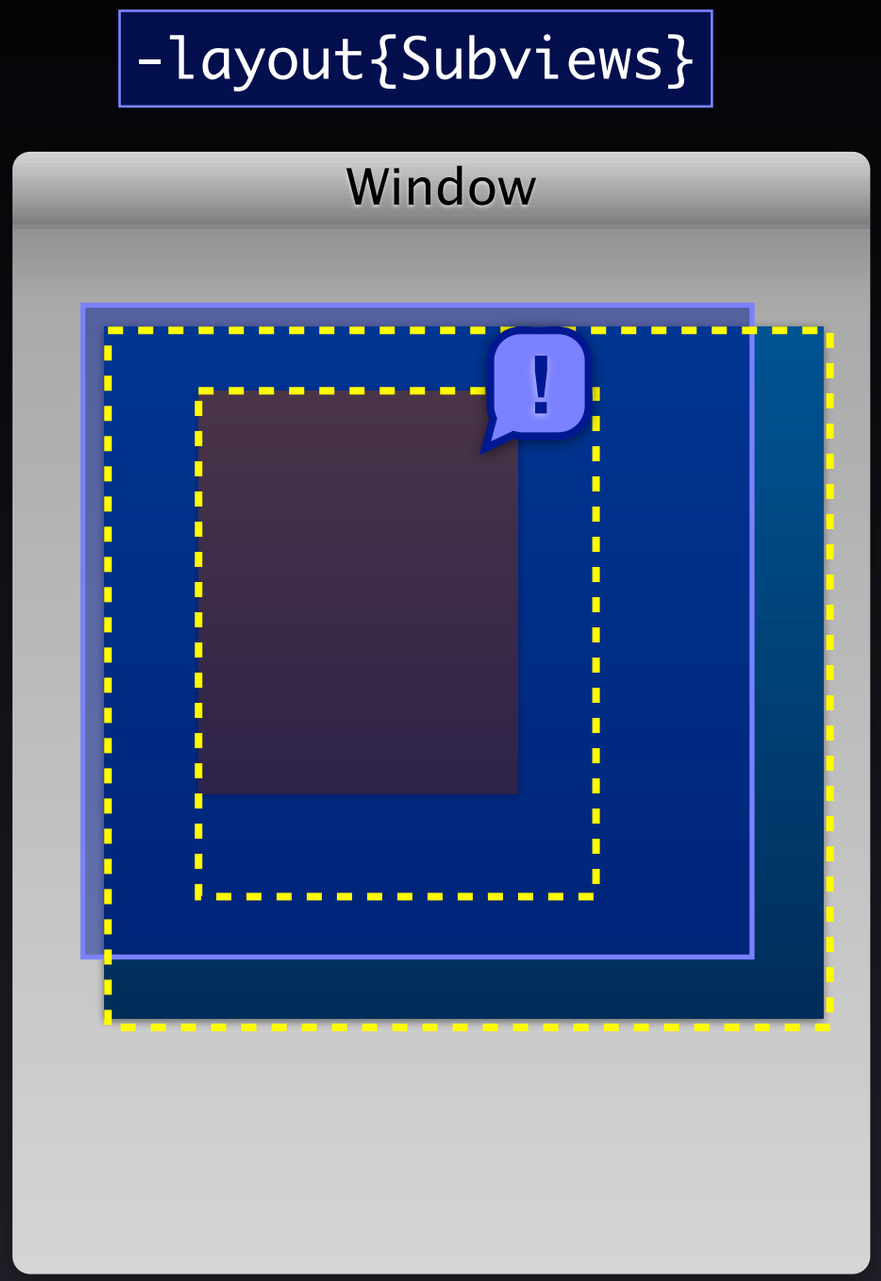
How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

3) System of inequalities is solved all at once

4) Views' frames are updated in -layout



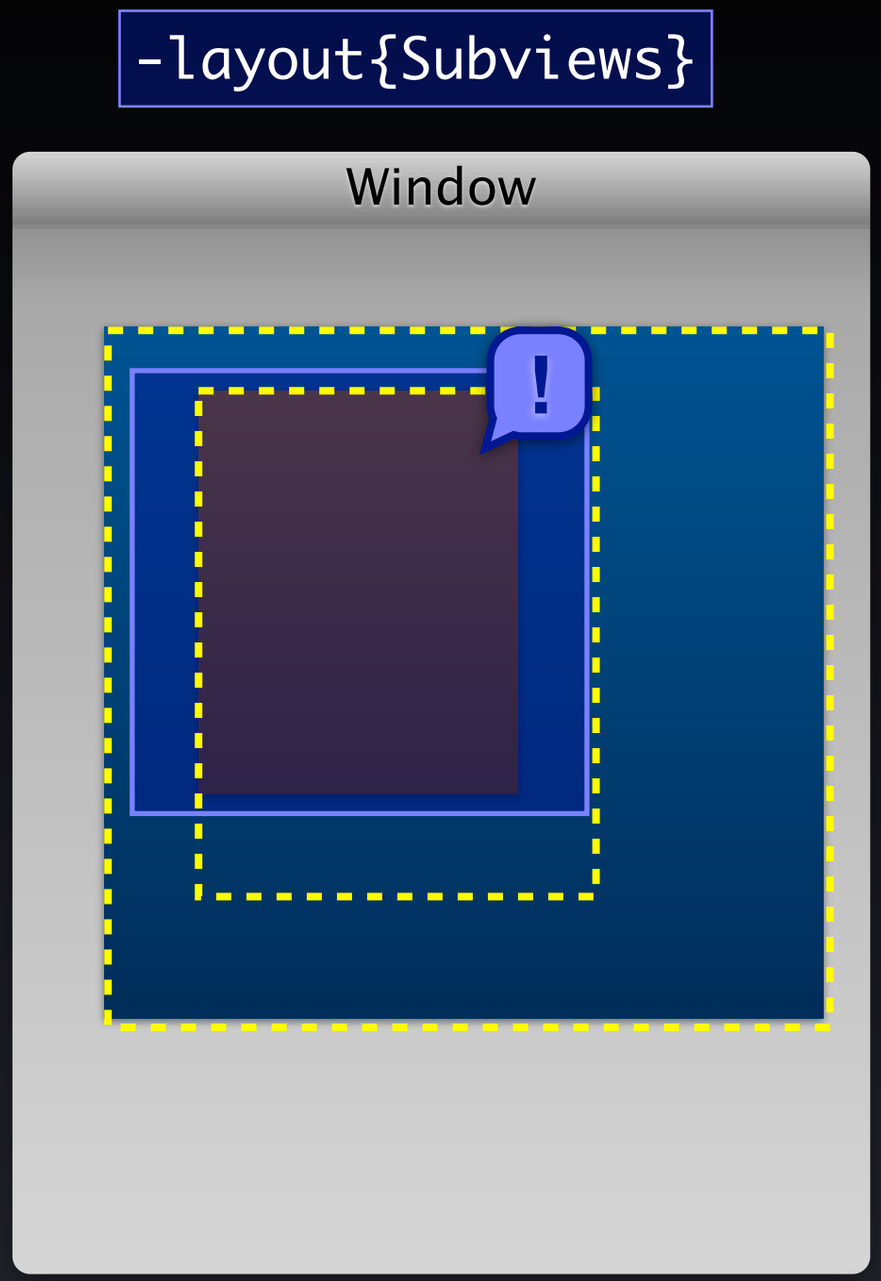
How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

3) System of inequalities is solved all at once

4) Views' frames are updated in -layout



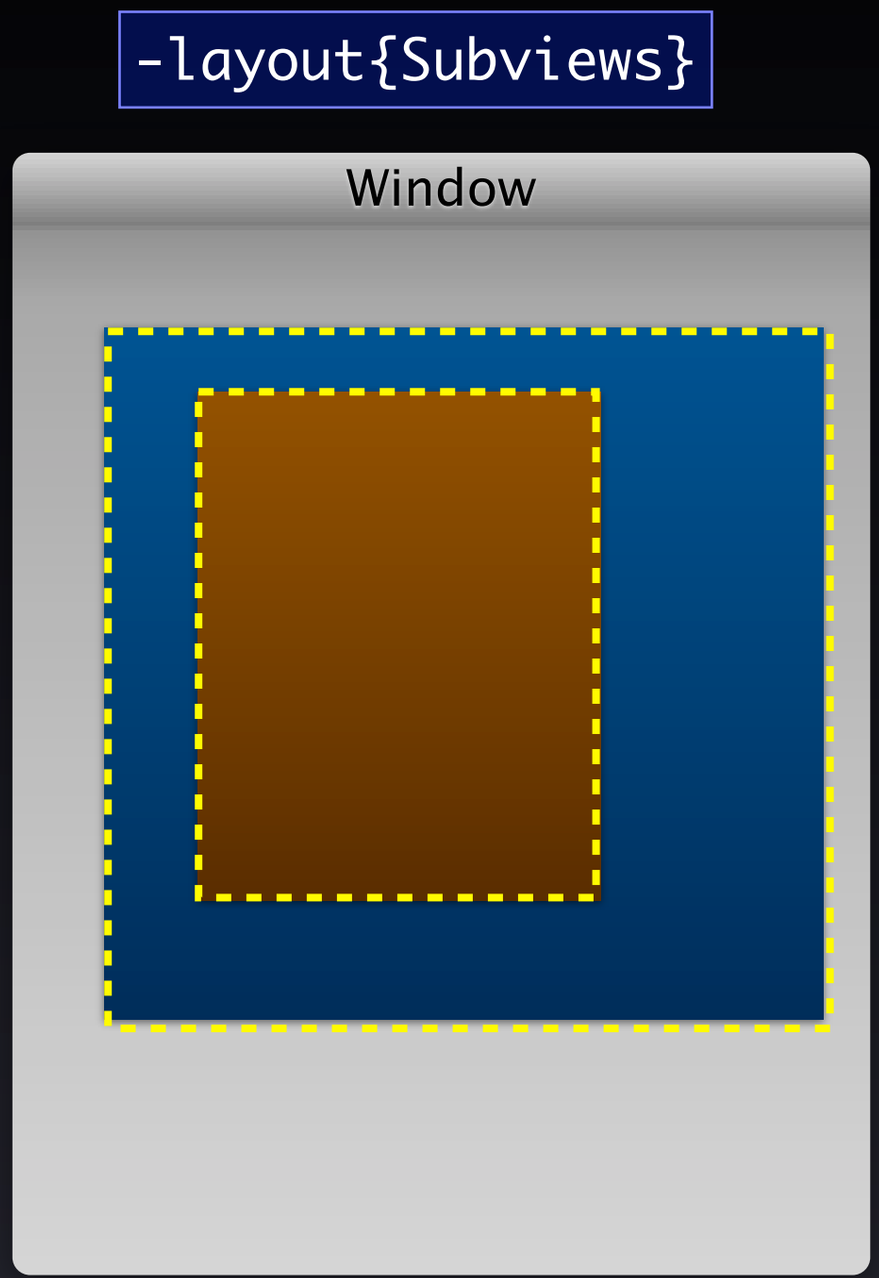
How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

3) System of inequalities is solved all at once

4) Views' frames are updated in -layout



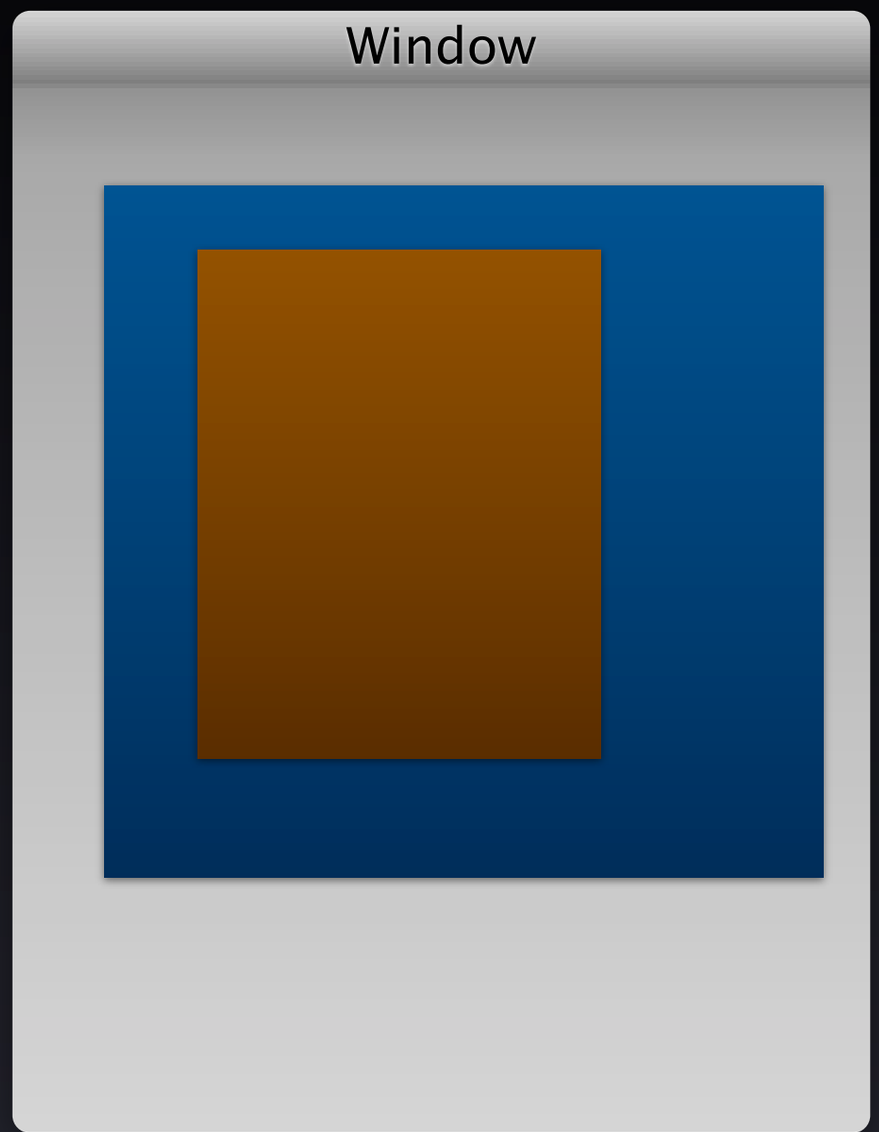
How Auto Layout Works

1) Install constraints on views

2) Constraints are converted into inequalities

3) System of inequalities is solved all at once

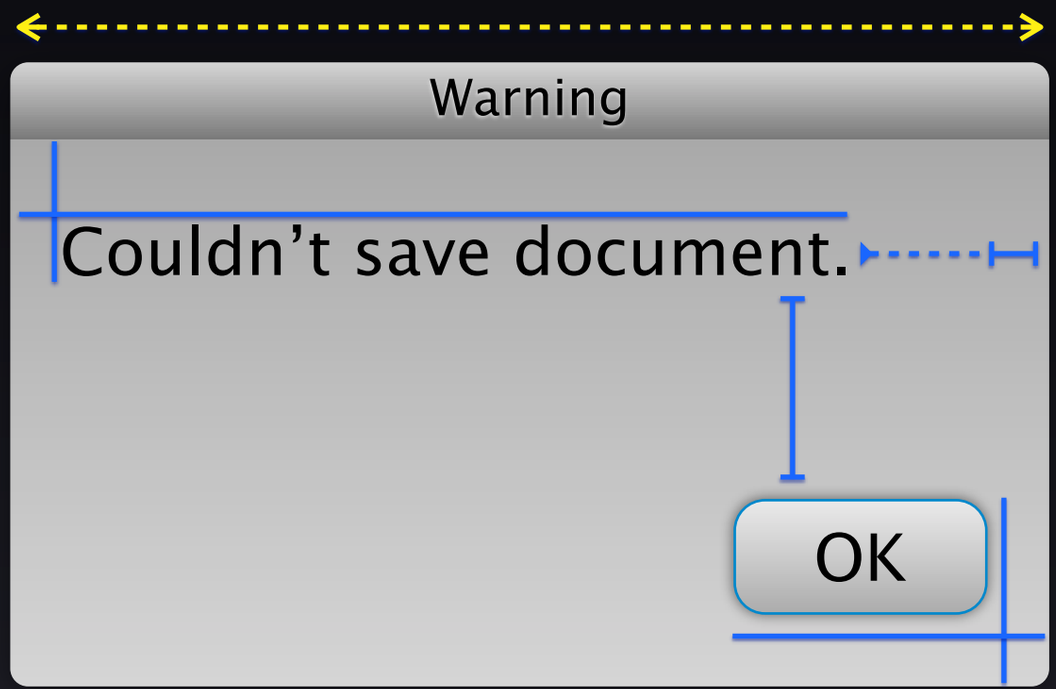
4) Views' frames are updated in -layout



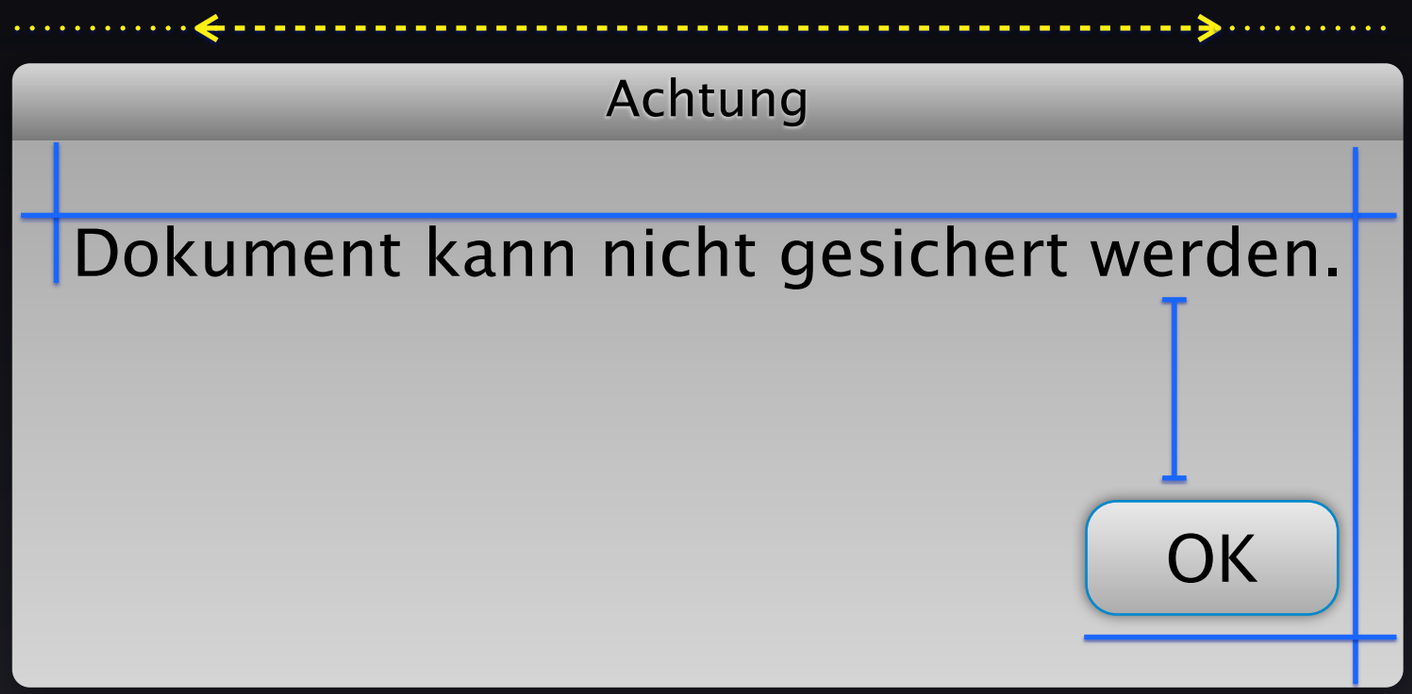
Advantages of Constraints

- Have priorities
- Express inequalities
- Take into account intrinsic sizes
- Have a multiplier and a constant
- Relate to views other than superview

Localization



Localization



Thinking with Constraints

- Layout is descriptive, not reactive
- Think Globally, Act Locally
- Inequalities have infinite solutions
- Your container needs a size too!

Demo

Auto Layout in Interface Builder

- Turn on auto layout in document inspector
- IB refuses to allow ambiguous layout
  - Would be nice if it did *(r.12986107)*
- "Promote to user constraints" (on Size inspector) to edit and persist constraints
  - It might still delete them if you drag views around!

Auto Layout in Code (OS X)

```
- (void) updateConstraints {
    if (!_myConstraints) {
        _myConstraints = [[NSLayoutConstraint
            constraintsWithVisualFormat:…] retain];
        [self addConstraints:_myConstraints];
    }

    [super updateConstraints];
}
```

Auto Layout in Code (iOS)

```
- (void) updateViewConstraints {
    if (!_myConstraints) {
        _myConstraints = [[NSLayoutConstraint
            constraintsWithVisualFormat:…] retain];
        [self.view addConstraints:_myConstraints];
    }

    [super updateViewConstraints];
}
```

Constraints best practices

- Publish your priority constants
- Wait until `-updateConstraints` to add constraints
- Store constraints in strong ivars
  - Would be nice if constraints had identifiers *(r. 12560969)*

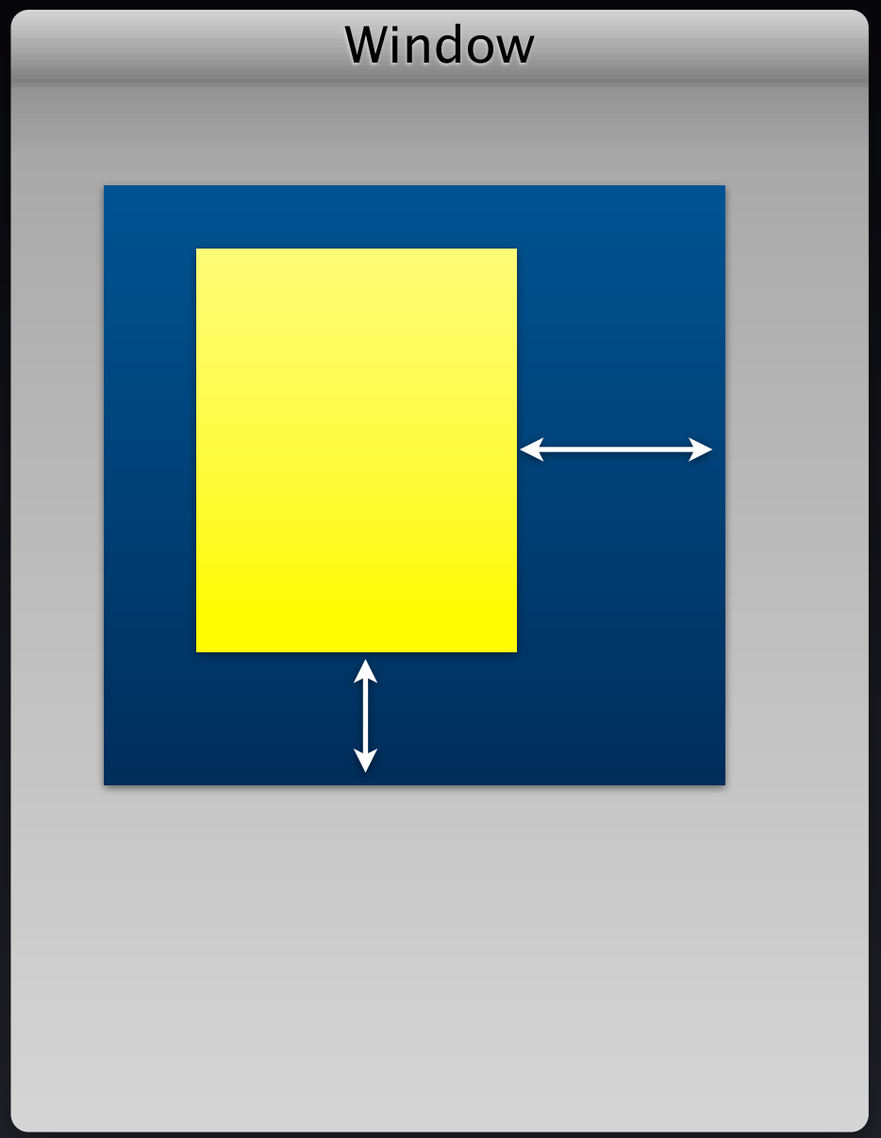
Constraints best practices

- Remove constraints as soon as sensible
- Don’t remove constraints unnecessarily
  - Updating is faster than removing
  - Easy to cause ambiguity
  - Other views might add constraints
  - This means do not do `[self removeConstraints:self.constraints]`

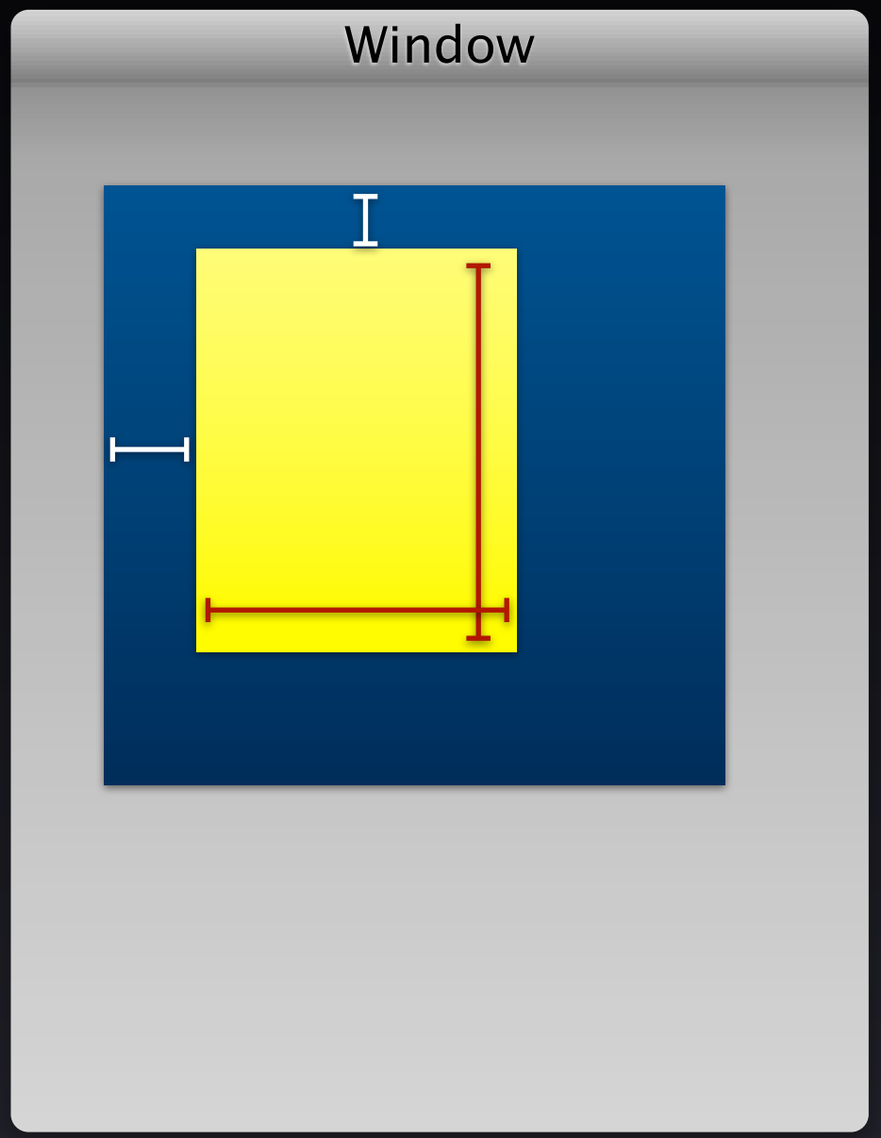
What about existing code?

- `translatesAutoresizingMaskInto-Constraints`
- Managed by the *superview* (or view controller)
- Produces constraints that express your view's frame

translatesAutoresizingMask-IntoConstraints



translatesAutoresizingMask-IntoConstraints



It Just Works™

- If nobody opts into auto layout, it is never enabled for the window
- `-resizeSubviewsWithOldSize` is sent as usual
  - This method is aware of constraints
  - Runs the solver without invoking `-updateConstraints` *(r.12466034)*

It Just Works™

- Translated constraints are installed on the superview (not guaranteed, but logical)
- Not guaranteed when constraints will be updated or installed
  - Installation seems to wait for in `-updateConstraints`
  - Updates seem to happen immediately in `-setFrame:`

Scroll Views

Scroll Views (iOS)

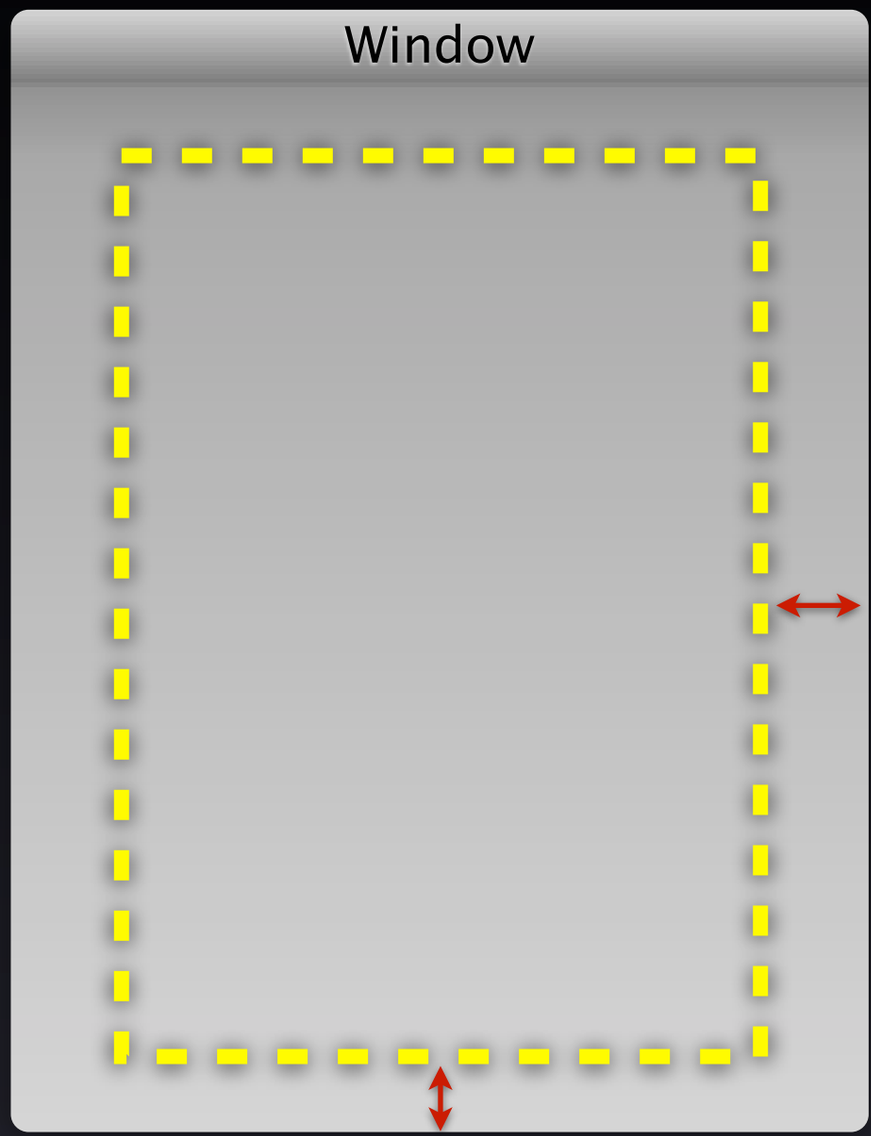
- UIScrollView treats constraints specially
  - Its subview's constraints determine the scroll view's `contentSize`
  - **Do not set the `contentSize` property!**
  - Ensure that constraints *external* to the scroll view sufficiently define its frame
- IB support is a bit confusing at first

Demo

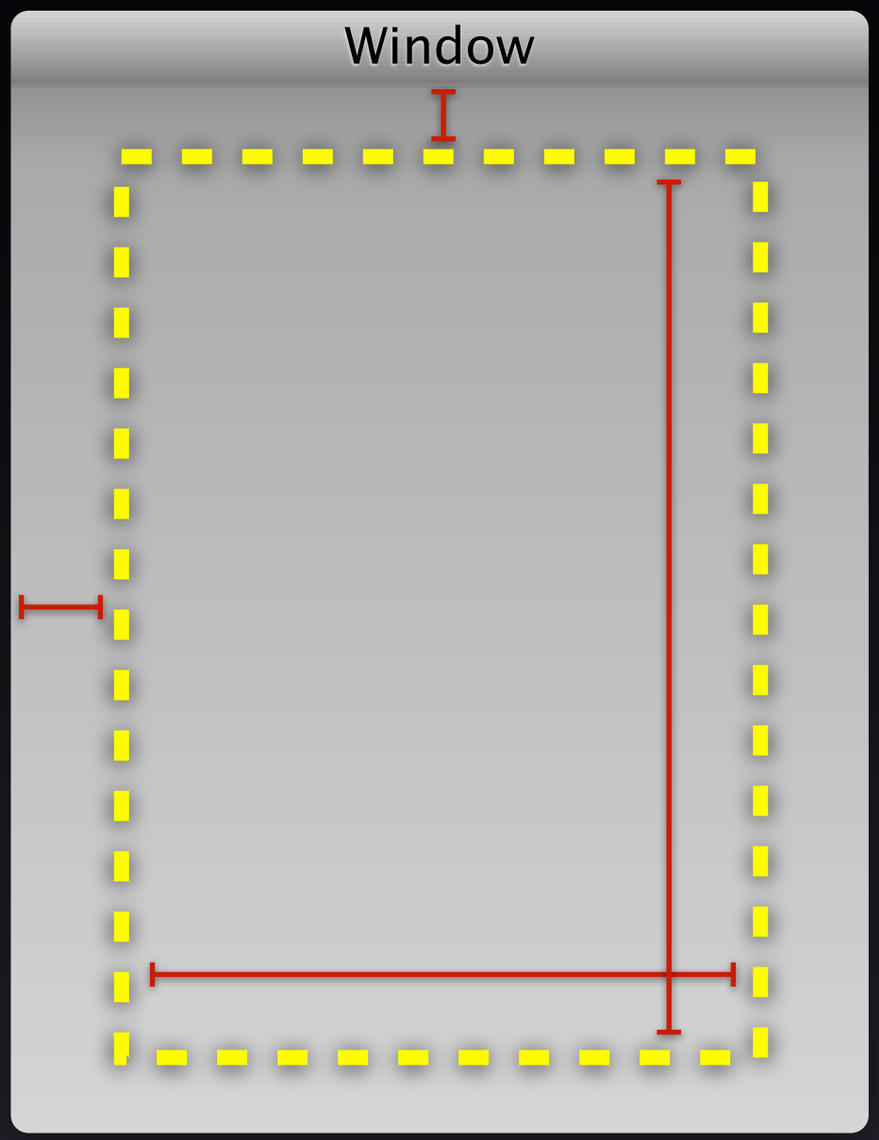
Scroll Views (OS X)

- NSScrollView contains an NSClipView
- Positions this view in `-tile` using `-setFrame:`
- NSClipView transforms its bounds origin
- Naive attempts to fill the scroll view's viewport will resize the scroll view

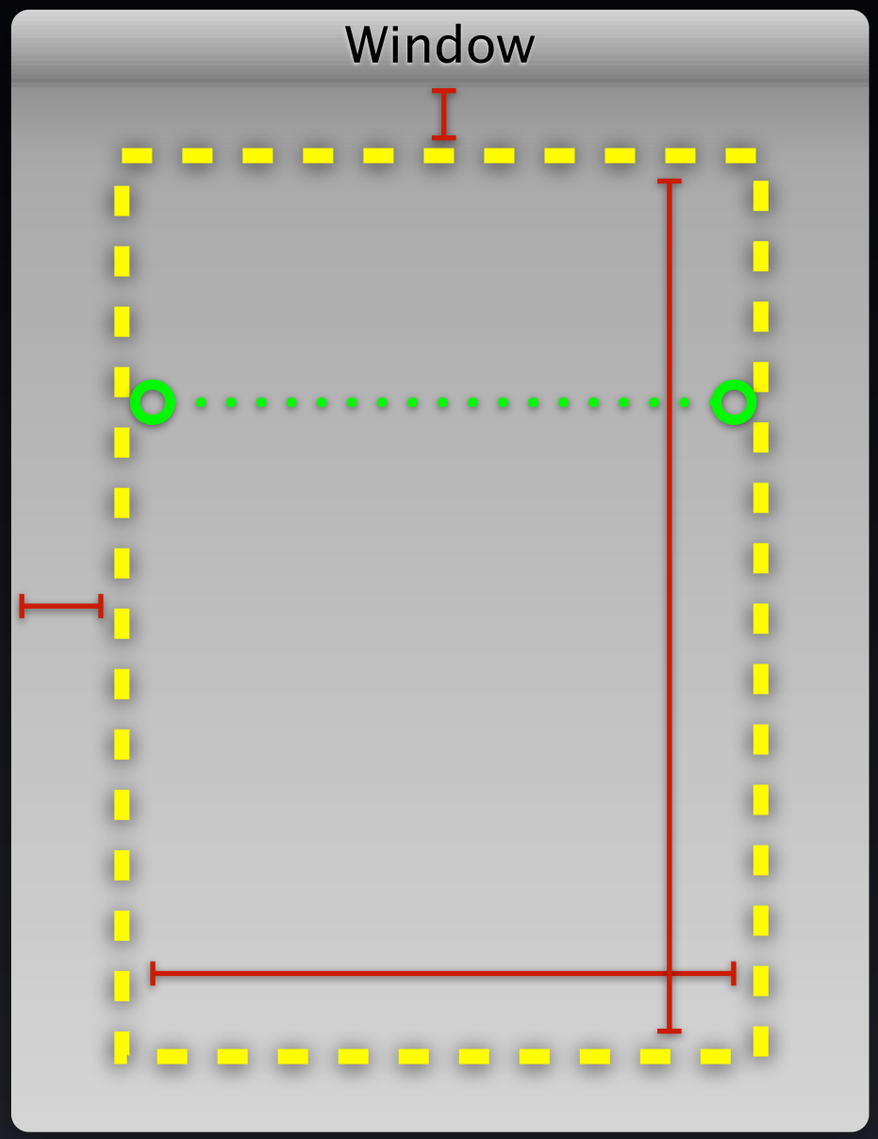
Scroll Views (OS X)



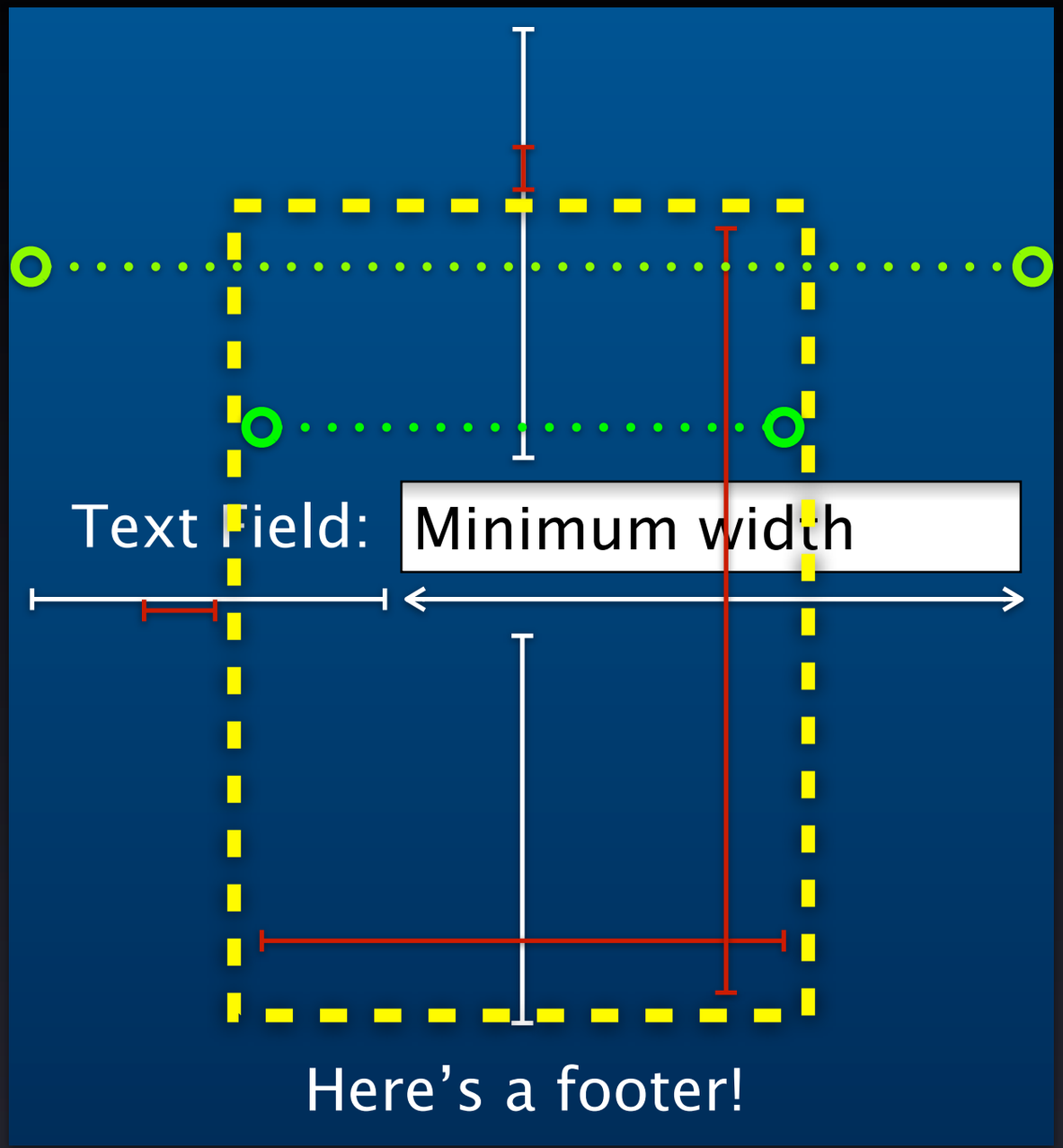
Scroll Views (OS X)



Scroll Views (OS X)



Scroll Views (OS X)



Scroll Views (OS X)

- Solution: install constraints on self to fill available viewport space
- Break our superview's use of `translatesAutoresizingMaskInto-Constraints` by forcing it to `NO`
- This is a **HACK!** Don't do this lightly!

Views that don’t support Auto Layout

- NSSplitView (10.7)
- UICollectionView
- Views that want to fill their scroll view frame
  - NSTableView, NSTextView
  - Can sometimes fake it by overriding `-intrinsicContentSize`

Beyond form fields

- Most interesting interfaces aren’t forms
- Sometimes auto layout can’t express *all* of what we want, but can express *most* of it
- Single-pass layout is beneficial on its own
- Override `-layout`

Rules of custom layout

- Cannot leave `-layout` needing layout
- Cannot leave `-updateConstraints` needing constraints updated
- Call `super` from overrides of both
- Earlier passes should not invoke later ones
- Views' positions must be fully specified

Demo

Debugging

- -[NSWindow visualizeConstraints:]
- -[{NS,UI}View constraintsAffecting LayoutFor{Orientation,Axis}:]
- -[NSView _subtreeDescription]
- -[UIView _autoLayoutTrace]

Summary

- Constraints are more powerful than springs & struts
- Your existing springs & struts code should work with `translatesAutoresizingMaskIntoConstraints`

Summary

- When using auto layout with UIScrollView, don’t try to set its contentSize
- When using it with NSScrollView, you have to hack around to fill your superview frame
- There are some classes that don’t support constraints at all

Other Resources

- WWDC 2012 videos
  - Session 202: Introduction to Auto Layout
  - Session 232: Auto Layout by Example
  - Session 228: Best Practices for Mastering Auto Layout

Other Resources

- Radars
  - 12986107 (IB should allow ambiguous layout at design time)
  - 12560969 (Constraints should have an identifier property)
  - 12466034 (`-resizeWithOldSuperview-Size:` should invoke `-updateConstraints`)

Q&A

Kyle Sluder
The Omni Group

optshiftk.com

@optshiftk

Thank You