AutoLayout

jano@jano.com.es

AutoLayout

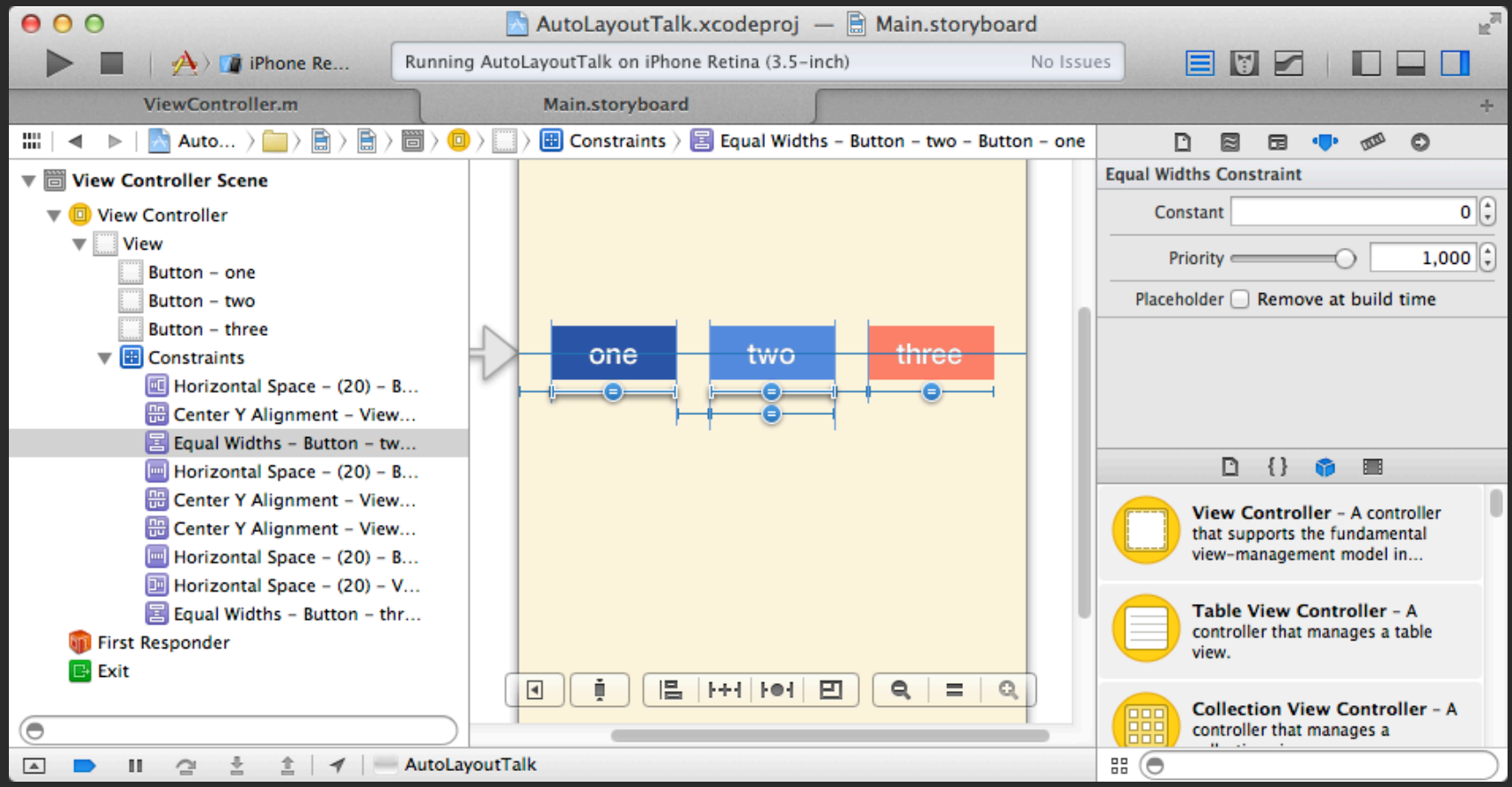
Layout

Layout

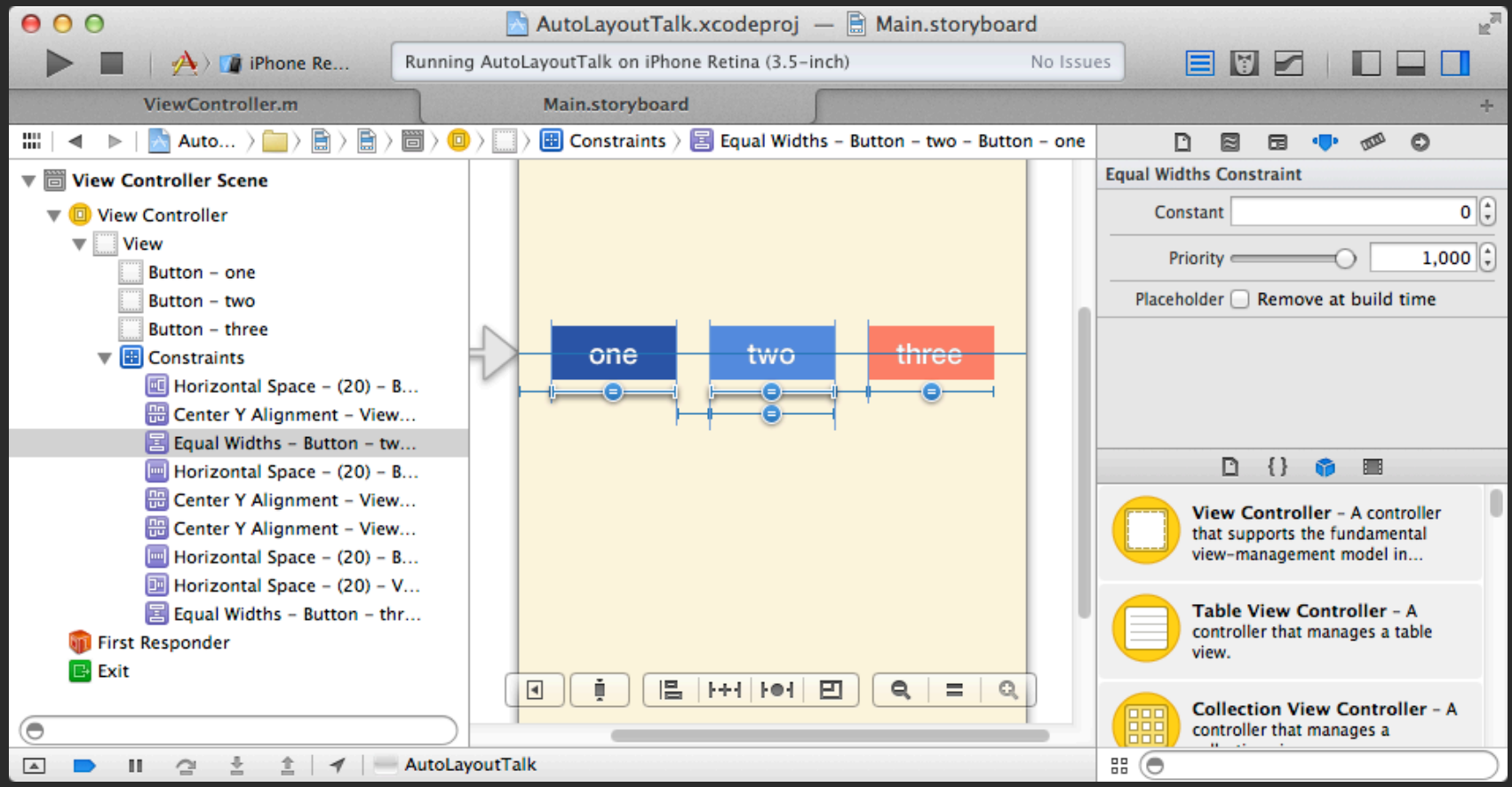
bounds
center
} frame
transform
autoresizingMask

AutoLayout

AutoLayout


AutoLayoutTalk.xcodeproj — Main.storyboard
iPhone Re...
Running AutoLayoutTalk on iPhone Retina (3.5-inch)
No Issues
ViewController.m
Main.storyboard
Auto...
Constraints
Equal Widths - Button - two - Button - one
View Controller Scene
View Controller
View
Button - one
Button - two
Button - three
Constraints
Horizontal Space - (20) - B...
Center Y Alignment - View...
Equal Widths - Button - tw...
Horizontal Space - (20) - B...
Center Y Alignment - View...
Center Y Alignment - View...
Horizontal Space - (20) - B...
Horizontal Space - (20) - V...
Equal Widths - Button - thr...
First Responder
Exit
one
two
three
Equal Widths Constraint
Constant
0
Priority
1,000
Placeholder
Remove at build time
View Controller - A controller that supports the fundamental view-management model in...
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a
AutoLayoutTalk

AutoLayout


AutoLayoutTalk.xcodeproj — Main.storyboard
iPhone Re...
Running AutoLayoutTalk on iPhone Retina (3.5-inch)
No Issues
ViewController.m
Main.storyboard
Auto...
Constraints
Equal Widths - Button - two - Button - one
View Controller Scene
View Controller
View
Button - one
Button - two
Button - three
Constraints
Horizontal Space - (20) - B...
Center Y Alignment - View...
Equal Widths - Button - tw...
Horizontal Space - (20) - B...
Center Y Alignment - View...
Center Y Alignment - View...
Horizontal Space - (20) - B...
Horizontal Space - (20) - V...
Equal Widths - Button - thr...
First Responder
Exit
one
two
three
Equal Widths Constraint
Constant
0
Priority
1,000
Placeholder
Remove at build time
View Controller - A controller that supports the fundamental view-management model in...
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a
AutoLayoutTalk

What do I need to know?

What do I need to know?

UIView properties

- `alignmentRectInsets`
- `constraints`
- `contentSize`
- `compression`
- `hugging`
- `translatesAutoresizingMaskIntoConstraints`

What do I need to know?

UIView properties
Interface Builder > VFL > API

What do I need to know?

UIView properties
Interface Builder > VFL > API
UIViewController lifecycle

What do I need to know?

UIView properties
Interface Builder > VFL > API
Animation

UIView Properties

constraints

Layout	`NSLayoutConstraint`
Content	`NSContentSizeLayoutConstraint`
Autosizing	`NSAutoresizingMaskLayoutConstraint`

constraints

```
view1.attribute1 RELATION multiplier * view2.attribute2 + constant
```

```
NSLayoutAttributeLeft
NSLayoutAttributeRight
NSLayoutAttributeTop
NSLayoutAttributeBottom
NSLayoutAttributeLeading
NSLayoutAttributeTrailing
NSLayoutAttributeWidth
NSLayoutAttributeHeight
NSLayoutAttributeCenterX
NSLayoutAttributeCenterY
NSLayoutAttributeBaseline

NSLayoutAttributeNotAnAttribute
```

```
NSLayoutRelationEqual
NSLayoutRelationGreaterThanOrEqual
NSLayoutRelationLessThanOrEqual
```

constraints

NSLayoutConstraint

Tasks

Creating Constraints

+ constraintsWithVisualFormat:options:metrics:views:
+ constraintWithItem:attribute:relatedBy:toItem:attribute:multiplier:constant:

Accessing Constraint Data

priority *property*
firstItem *property*
firstAttribute *property*
relation *property*
secondItem *property*
secondAttribute *property*
multiplier *property*
constant *property*

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

Controlling Constraint Archiving

shouldBeArchived *property*

constraints

`view1.attribute1 RELATION multiplier * view2.attribute2 + constant`

```
constraint = [NSLayoutConstraint
              constraintWithItem: view
              attribute: NSLayoutAttributeWidth
              relatedBy: NSLayoutRelationEqual
              toItem: nil
              attribute: NSLayoutAttributeNotAnAttribute
              multiplier: 1.0
              constant: 100.0];
[view addConstraint: constraint];

constraint = [NSLayoutConstraint
              constraintWithItem: view
              attribute: NSLayoutAttributeWidth
              relatedBy: NSLayoutRelationEqual
              toItem: nil
              attribute: NSLayoutAttributeNotAnAttribute
              multiplier: 1.0
              constant: 80.0];
[view addConstraint: constraint];
```

size=100x80

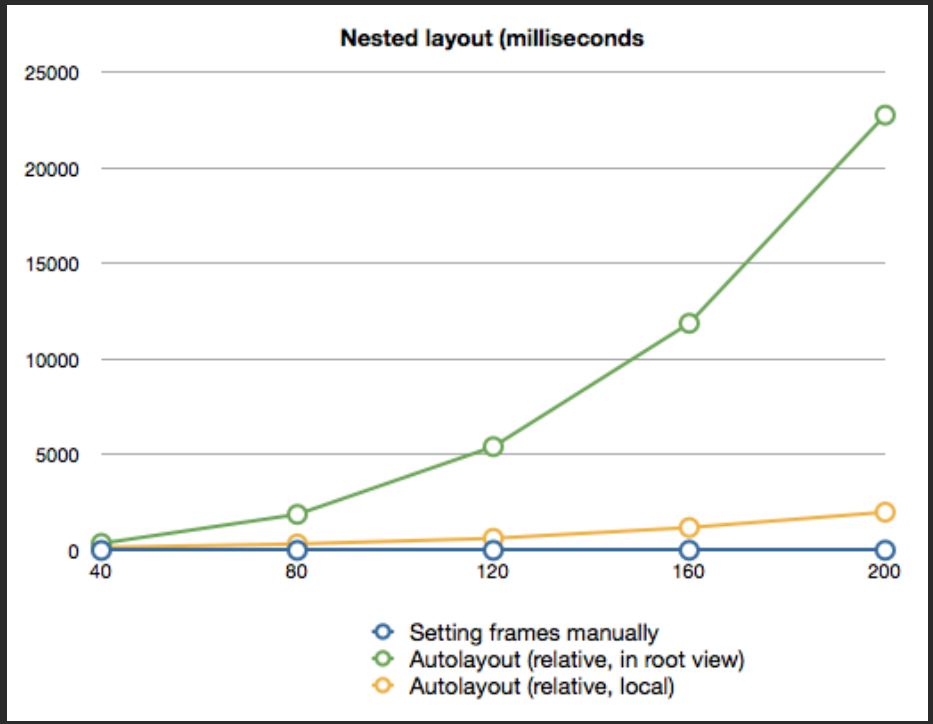
constraints

```
view1.attribute1 RELATION multiplier * view2.attribute2 + constant
```

- linear equations
- Cassowary Linear Arithmetic Constraint Solving Algorithm
- Pro tip: Use local flat hierarchies.

constraints

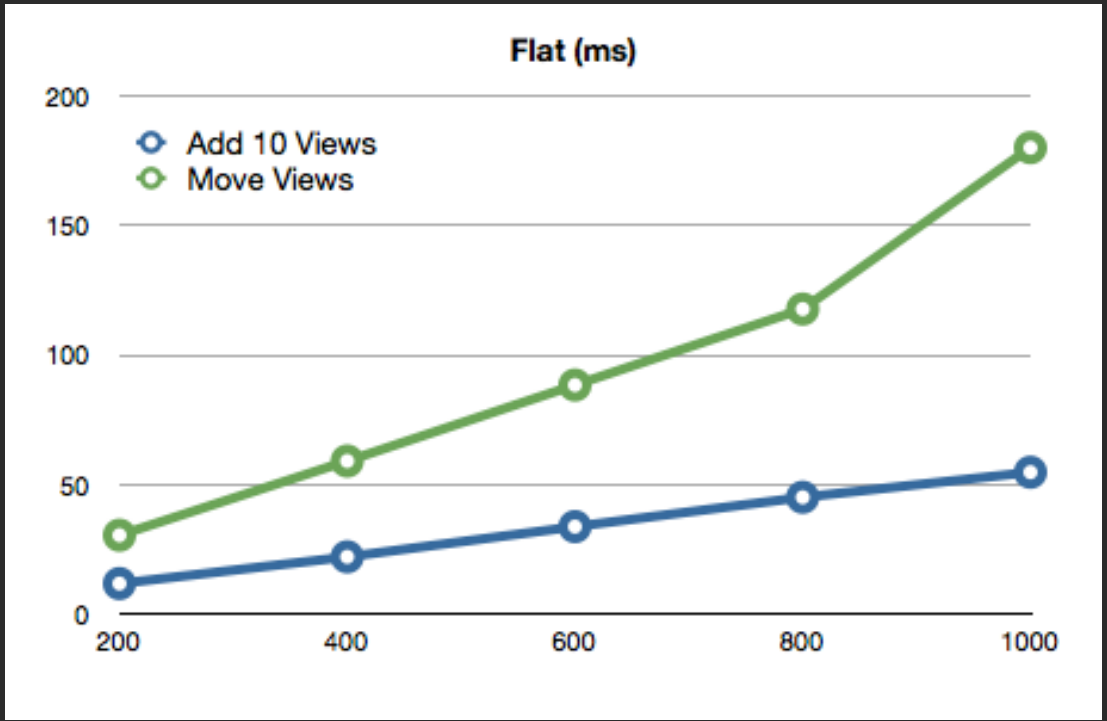
```
view1.attribute1 RELATION multiplier * view2.attribute2 + constant
```



http://pilky.me/view/36

constraints

```
view1.attribute1 RELATION multiplier * view2.attribute2 + constant
```



http://pilky.me/view/36


constraints
Screw-ups

Unsatisfiable

Errors › Errors › ViewController.m › @implementation ViewController

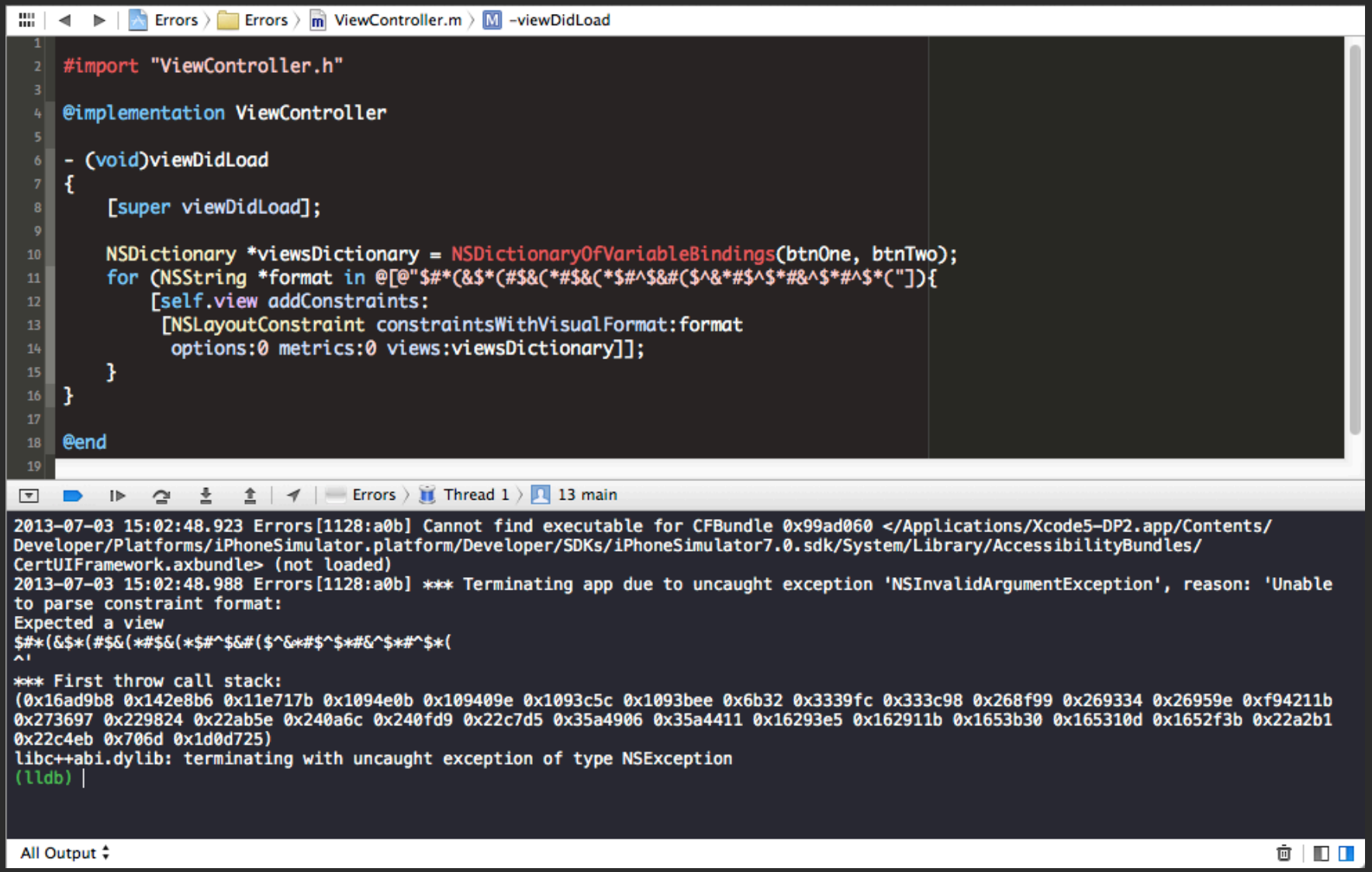
```
#import "ViewController.h"

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];

    NSDictionary *viewsDictionary = NSDictionaryOfVariableBindings(btnOne, btnTwo);
    for (NSString *format in @[@"H:|[btnOne][btnTwo]|",@"H:|[btnTwo][btnOne]|"]){
        [self.view addConstraints:
         [NSLayoutConstraint constraintsWithVisualFormat:format
          options:0 metrics:0 views:viewsDictionary]];
    }
}

@end
```

Errors.app

```
2013-07-03 14:59:25.205 Errors[1050:a0b] Cannot find executable for CFBundle 0xa0b2d70 </Applications/Xcode5-DP2.app/Contents/
Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator7.0.sdk/System/Library/AccessibilityBundles/
CertUIFramework.axbundle> (not loaded)
2013-07-03 14:59:25.287 Errors[1050:a0b] Unable to simultaneously satisfy constraints.
    Probably at least one of the constraints in the following list is one you don't want. Try this: (1) look at each constraint and
try to figure out which you don't expect; (2) find the code that added the unwanted constraint or constraints and fix it. (Note: If
you're seeing NSAutoresizingMaskLayoutConstraints that you don't understand, refer to the documentation for the UIView property
translatesAutoresizingMaskIntoConstraints)
(
    "<NSLayoutConstraint:0xa0a14a0 'IB auto generated at build time for view with fixed frame' H:|-(42)-[UIButton:0xa0c11e0](LTR)
(Names: '|':UIView:0xa0b1530 )>",
    "<NSLayoutConstraint:0xa08c9f0 H:|-(0)-[UIButton:0xa0c11e0]   (Names: '|':UIView:0xa0b1530 )>"
)

Will attempt to recover by breaking constraint
<NSLayoutConstraint:0xa08c9f0 H:|-(0)-[UIButton:0xa0c11e0]   (Names: '|':UIView:0xa0b1530 )>
```

All Output

Invalid


Errors
Errors
ViewController.m
-viewDidLoad
#import "ViewController.h"
@implementation ViewController
- (void)viewDidLoad
{
[super viewDidLoad];
NSDictionary *viewsDictionary = NSDictionaryOfVariableBindings(btnOne, btnTwo);
for (NSString *format in @[@"$#*(&$*(#$&(*#$&(*$#^$&#($^&*#$^$*#&^$*#^$*("]){
[self.view addConstraints:
[NSLayoutConstraint constraintsWithVisualFormat:format
options:0 metrics:0 views:viewsDictionary]];
}
}
@end
Errors
Thread 1
13 main
2013-07-03 15:02:48.923 Errors[1128:a0b] Cannot find executable for CFBundle 0x99ad060 </Applications/Xcode5-DP2.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator7.0.sdk/System/Library/AccessibilityBundles/CertUIFramework.axbundle> (not loaded)
2013-07-03 15:02:48.988 Errors[1128:a0b] *** Terminating app due to uncaught exception 'NSInvalidArgumentException', reason: 'Unable to parse constraint format:
Expected a view
$#*(&$*(#$&(*#$&(*$#^$&#($^&*#$^$*#&^$*#^$*(
^
*** First throw call stack:
(0x16ad9b8 0x142e8b6 0x11e717b 0x1094e0b 0x109409e 0x1093c5c 0x1093bee 0x6b32 0x3339fc 0x333c98 0x268f99 0x269334 0x26959e 0xf94211b 0x273697 0x229824 0x22ab5e 0x240a6c 0x240fd9 0x22c7d5 0x35a4906 0x35a4411 0x16293e5 0x162911b 0x1653b30 0x165310d 0x1652f3b 0x22a2b1 0x22c4eb 0x706d 0x1d0d725)
libc++abi.dylib: terminating with uncaught exception of type NSException
(lldb)
All Output

Ambiguous

```
view.hasAmbiguousLayout
view.exerciseAmbiguityInLayout
```

```
for (UIView *view in self.subviews) {
    if ([view hasAmbiguousLayout]){
        NSLog(@"<%@:0x%0x>", view.description, (int)self);
    }
}
```

instrinsicContentSize

Suggested size for the view.

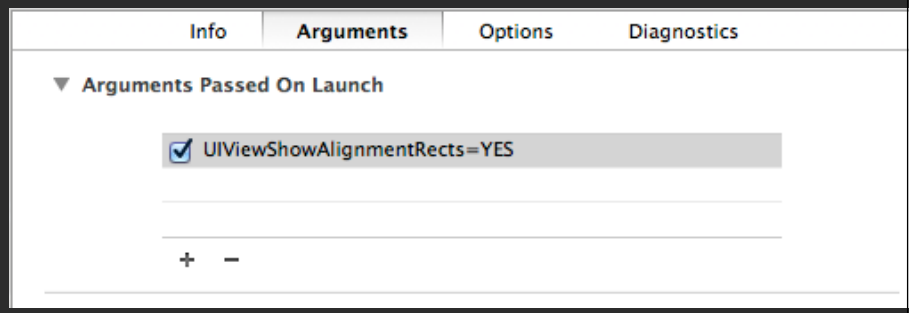
```
- (CGSize) intrinsicContentSize {
    return mySize;
}

[self invalidateIntrinsicContentSize];


UIImage *img = UIImage imageNamed:@"Icon.png"];
UIImageView *iv = [[UIImageView alloc] initWithImage:img];
NSLog(@"%@", NSStringFromCGSize(iv.intrinsicContentSize));
```

Alignment rectangle

UIView

```
- (UIEdgeInsets)alignmentRectInsets

- (CGRect)frameForAlignmentRect:(CGRect)alignmentRect
- (CGRect)alignmentRectForFrame:(CGRect)frame
```

Show rect lines:



Hug & compress

Hugging resists stretching
Compression resists shrinking


```
UILayoutConstraintAxis axis = UILayoutConstraintAxisHorizontal;
UILayoutPriority p = UILayoutPriorityDefaultHigh;
[button setContentCompressionResistancePriority:p forAxis:axis];
[button setContentHuggingPriority:p forAxis:axis];
```

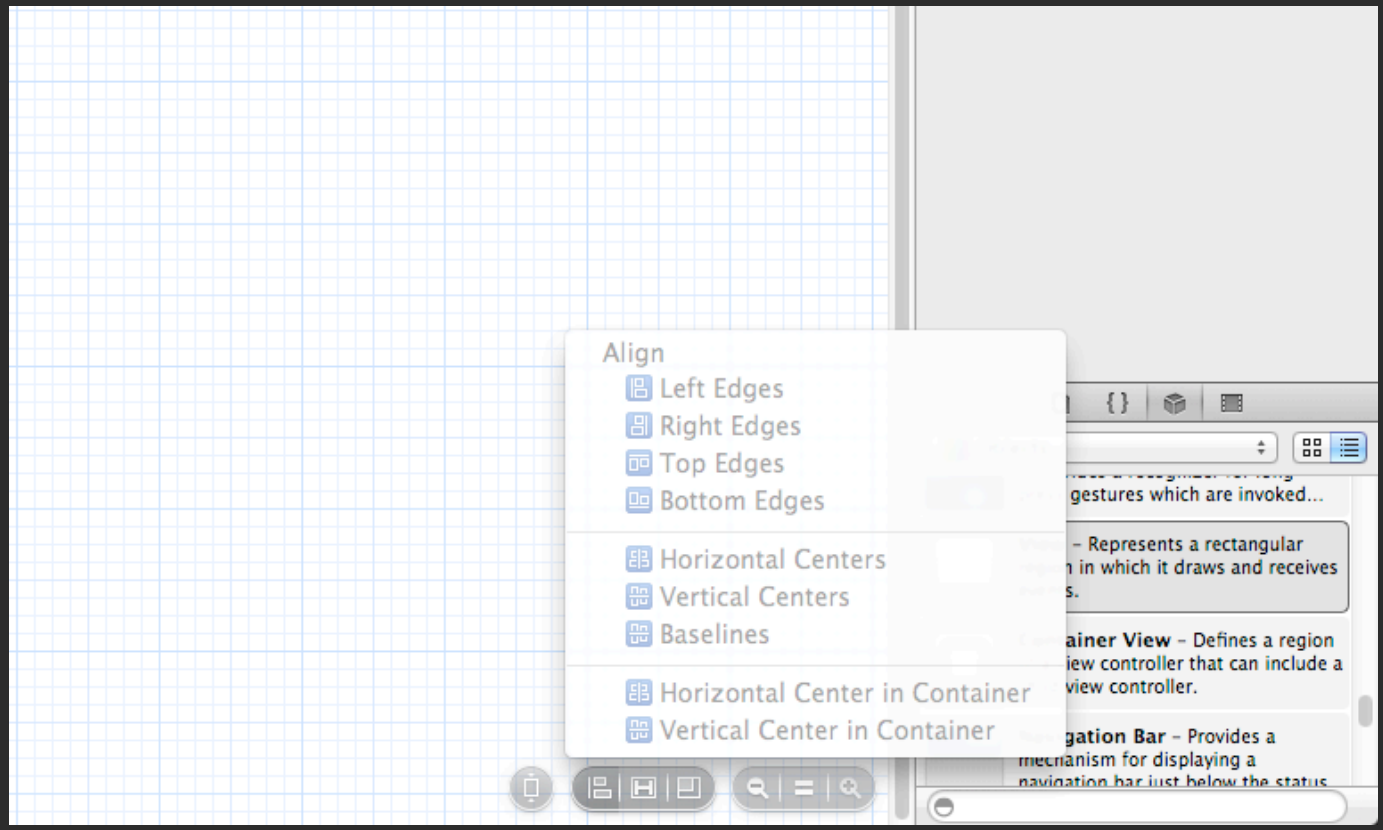
UIView properties

Interface Builder > VFL > API

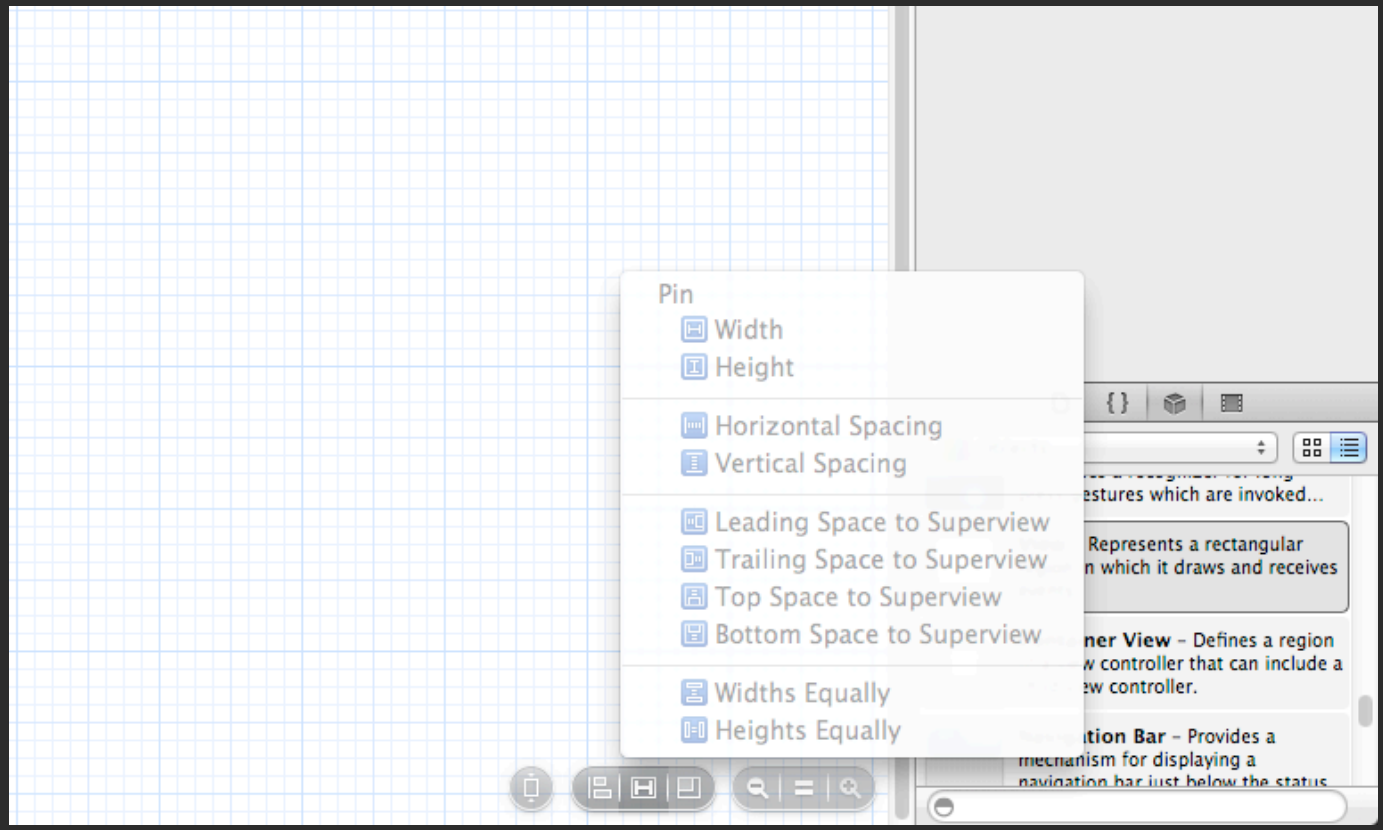
Animation

Interface Builder

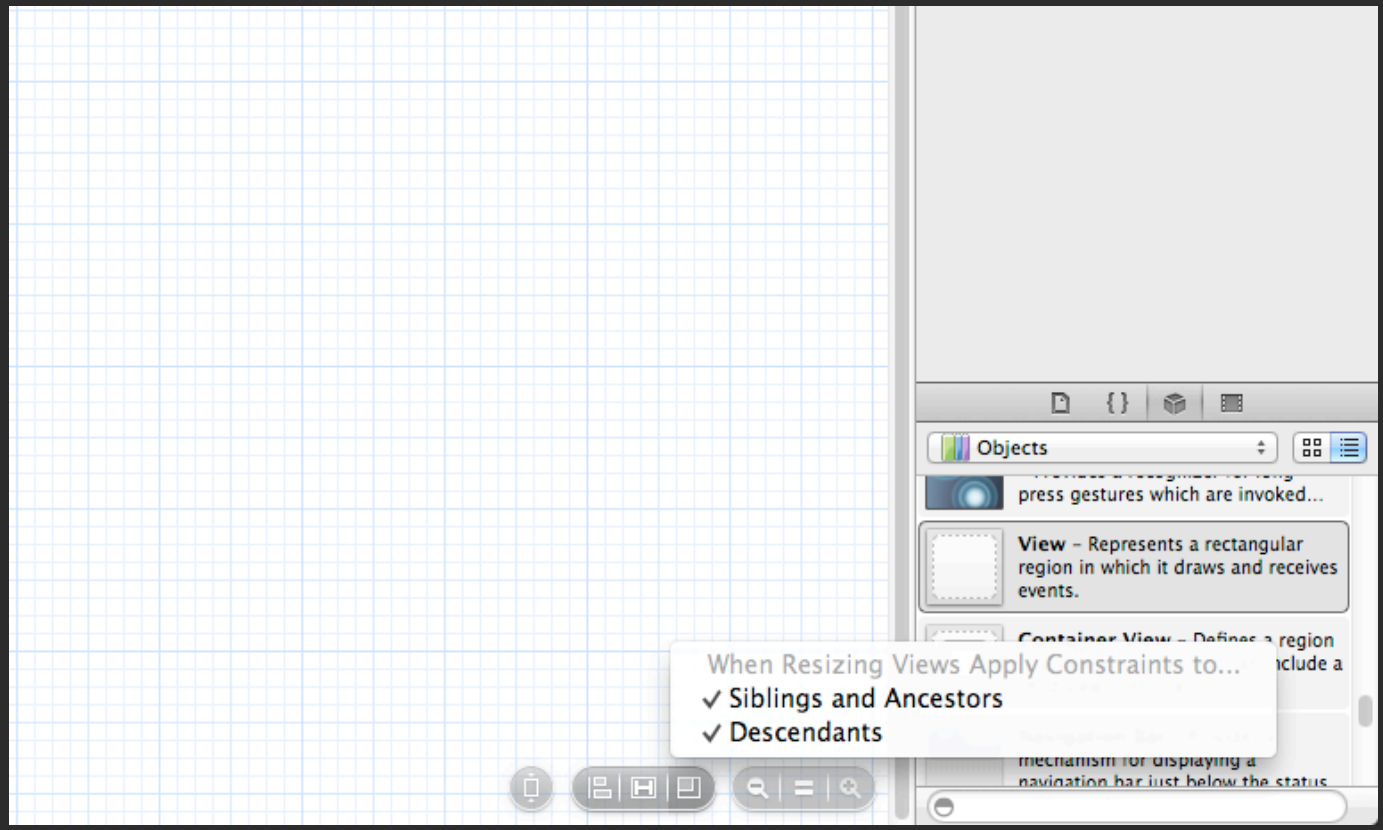
Interface Builder



Interface Builder



Interface Builder



Interface Builder

Interface Builder

IB > VFL > API
Constraints colors
IB can’t create ambiguous layouts
Add a constraint before deleting another
Preserve intrinsic size
Don’t optimize until everything is in place

UIView properties
Interface Builder > VFL > API
Animation

Visual Format Language

VFL

```
[NSLayoutConstraint
    constraintsWithVisualFormat:@"H:|-[buttonA]-|"
    options:0
    metrics:nil
    views:@{ @"buttonA" : buttonA }];
```

VFL

```
[NSLayoutConstraint
    constraintsWithVisualFormat:@"H:|-[buttonA]-|"
    options:0
    metrics:nil
    views:@{ @"buttonA" : buttonA }];
```

VFL

```
[NSLayoutConstraint
    constraintsWithVisualFormat:@"H:|-[buttonA]-|"
    options:0
    metrics:nil
    views:@{ @"buttonA" : buttonA }];
```

VFL

```
[NSLayoutConstraint
    constraintsWithVisualFormat:@"H:|-[buttonA]-distance-|"
    options:0
    metrics: @{ @"distance": @50 }
    views:@{ @"buttonA" : buttonA }];
```

VFL

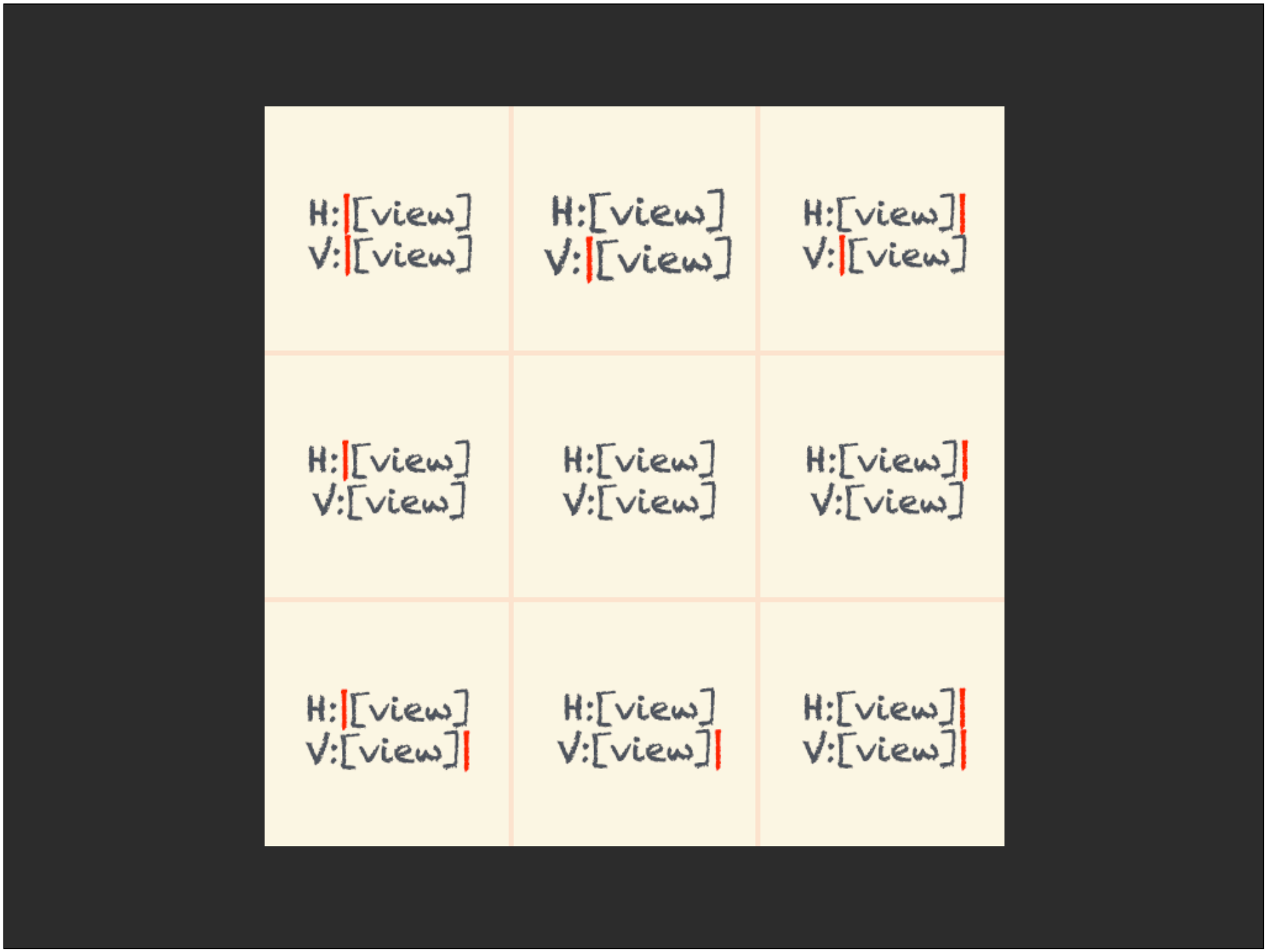
```
[NSLayoutConstraint
    constraintsWithVisualFormat:@"H:|-[buttonA]-distance-|"
    options:0
    metrics: @{ @"distance": @50 }
    views:@{ @"buttonA" : buttonA }];
```

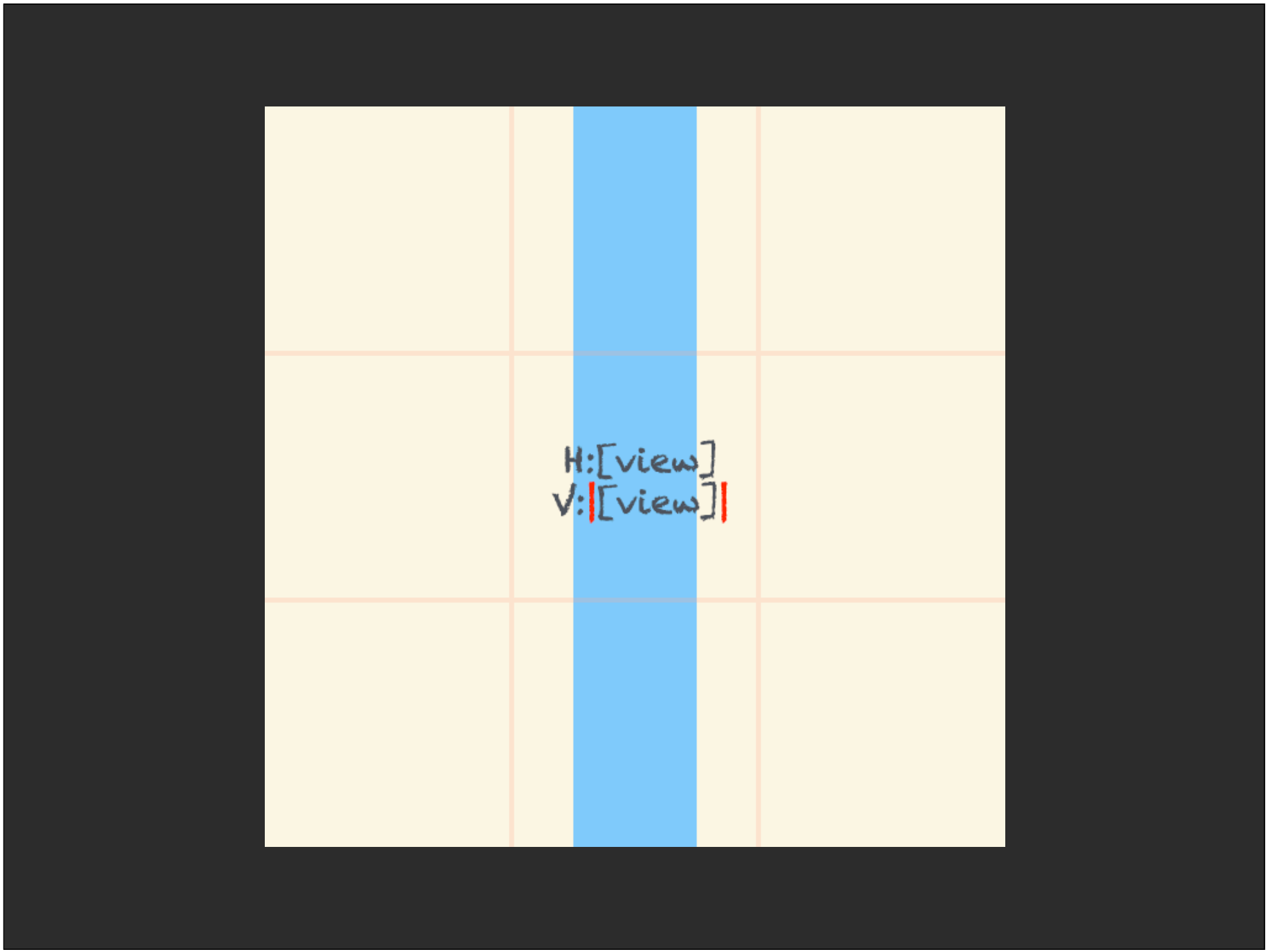
VFL

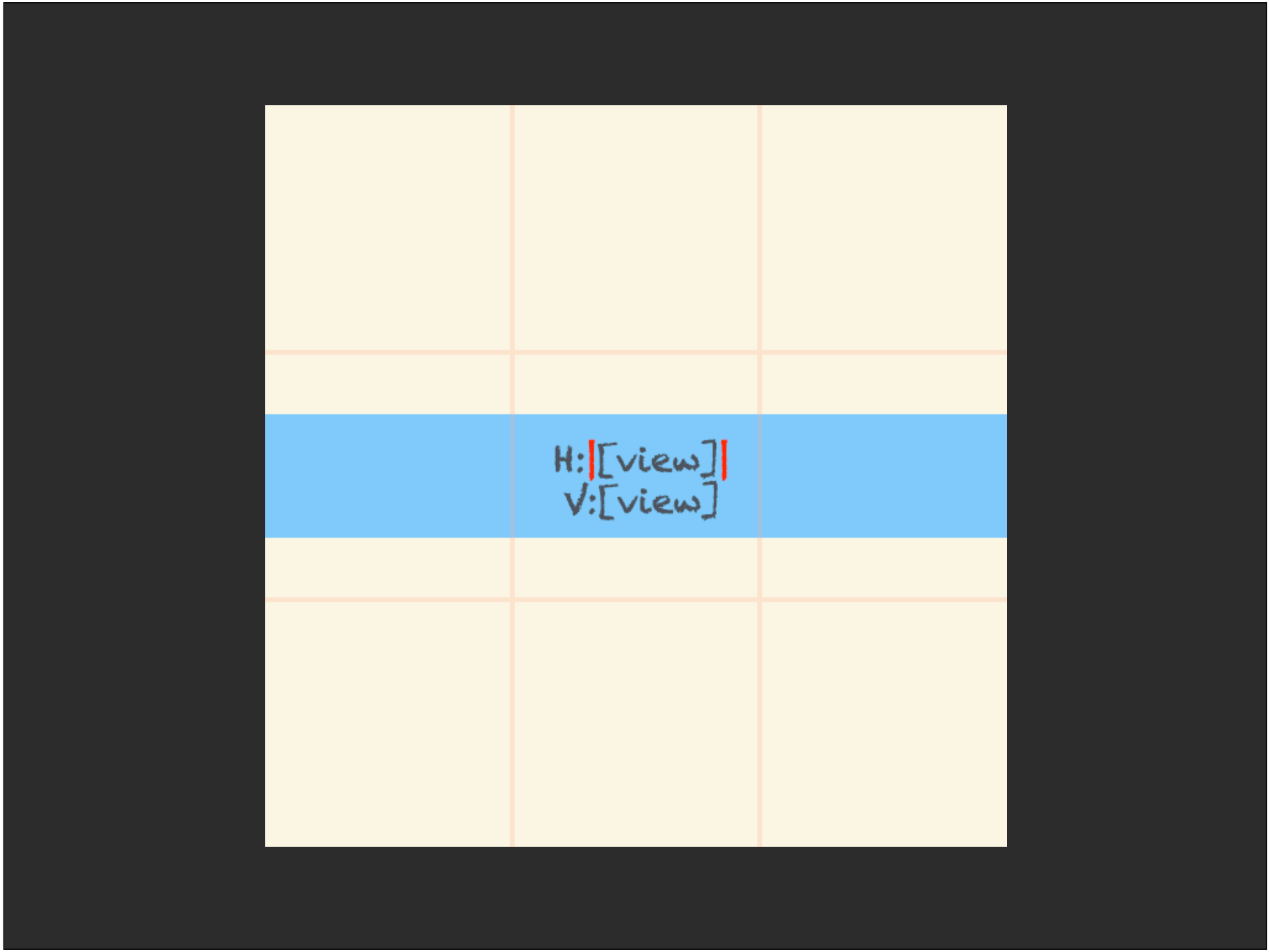
```
[NSLayoutConstraint
    constraintsWithVisualFormat:@"H:|-[buttonA]-distance-|"
    options:0
    metrics: @{ @"distance": @50 }
    views:NSDictionaryOfVariableBindings(buttonA)];
```

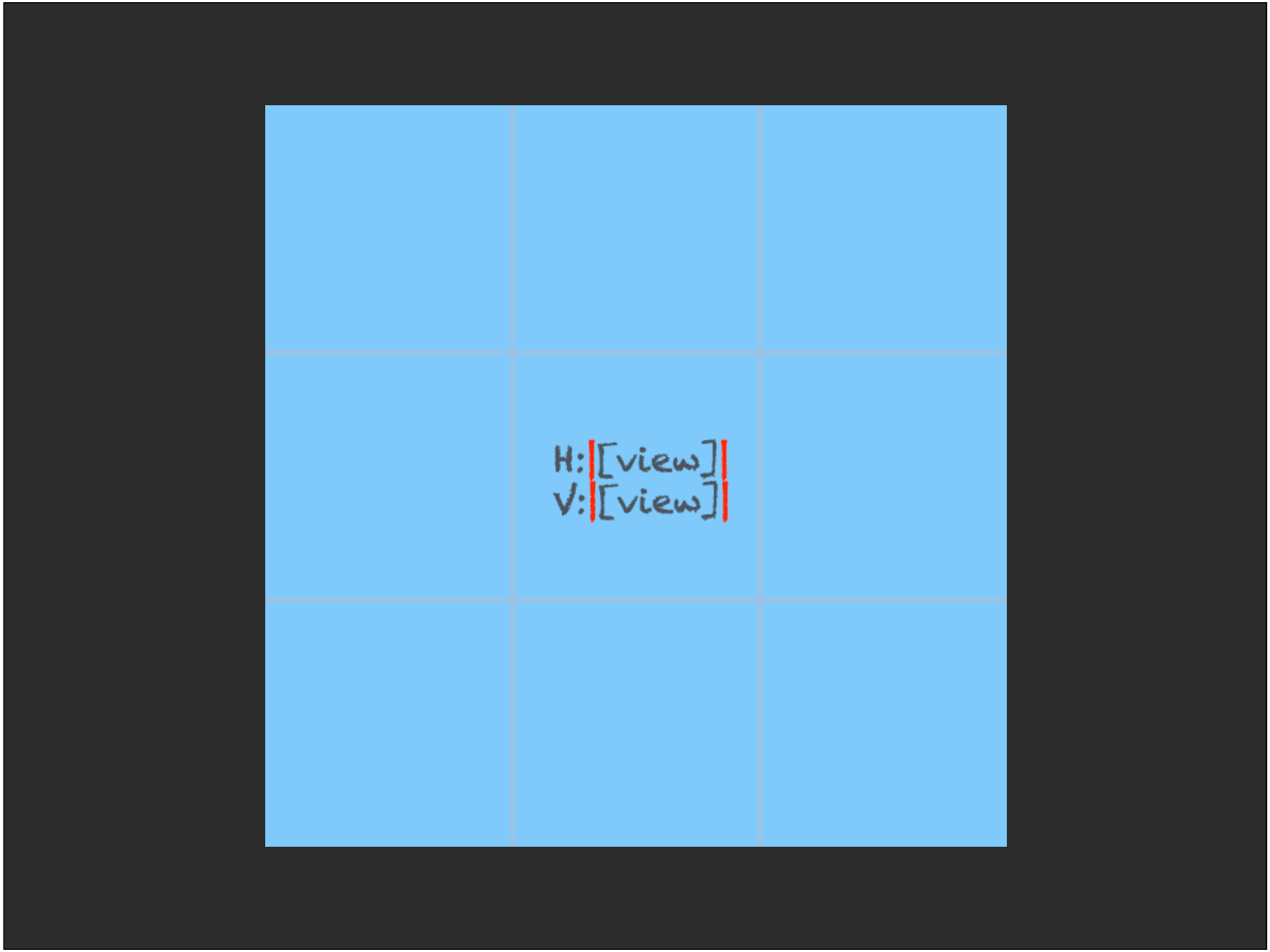
VFL

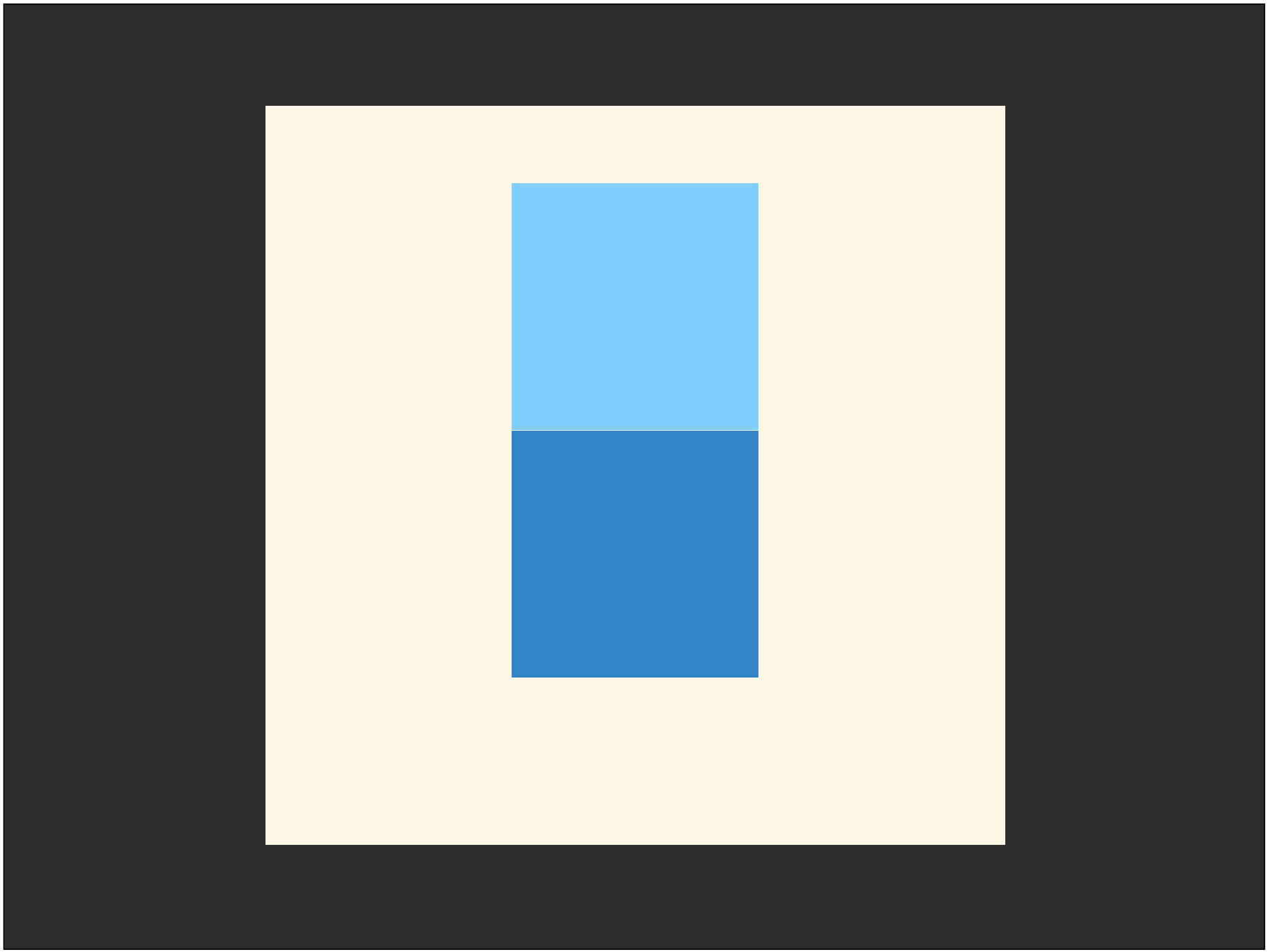
```
H V [view] | - @ ()
```

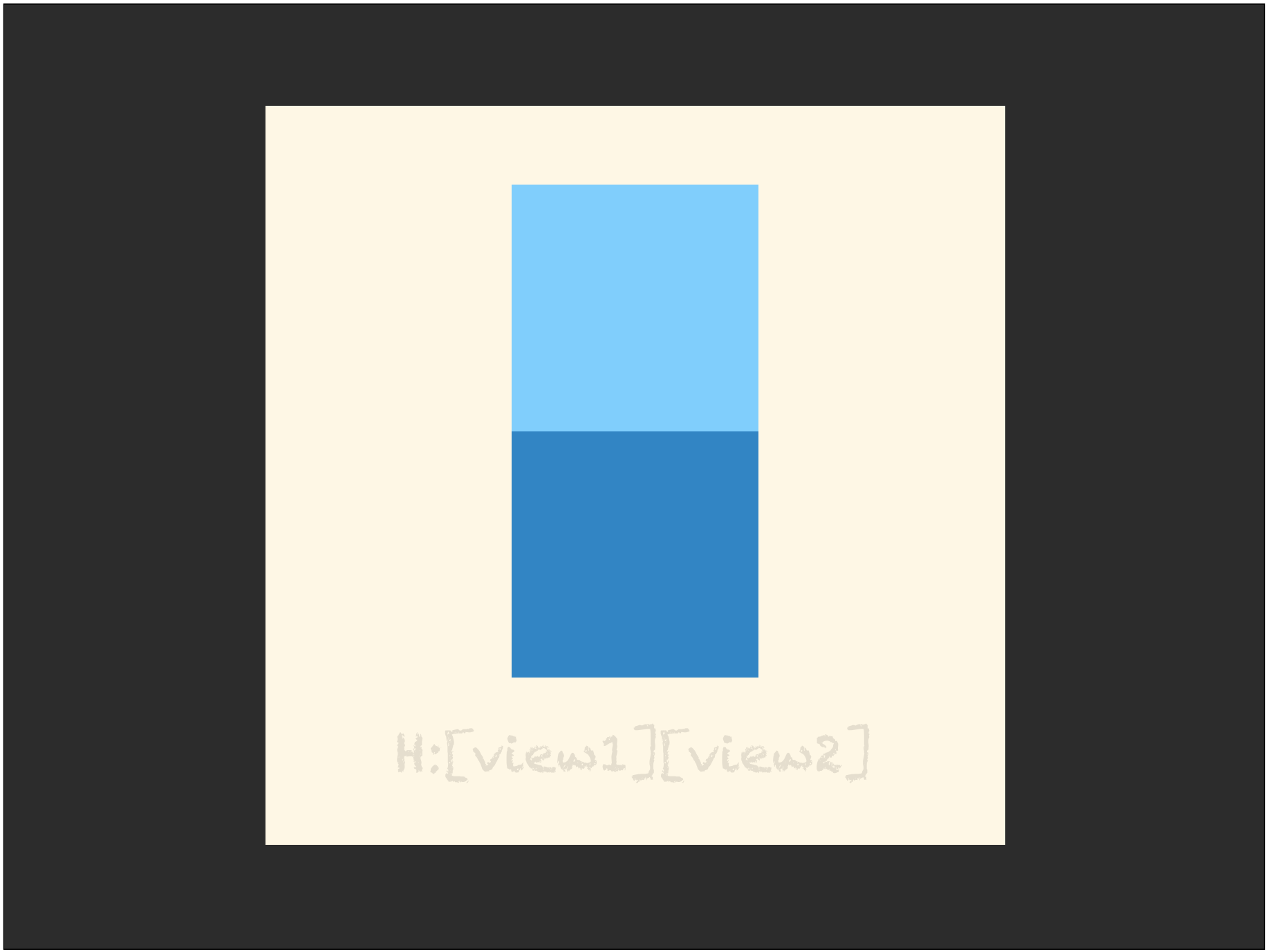

H:|[view]
V:|[view]
H:[view]
V:|[view]
H:[view]|
V:|[view]
H:|[view]
V:[view]
H:[view]
V:[view]
H:[view]|
V:[view]
H:|[view]
V:[view]|
H:[view]
V:[view]|
H:[view]|
V:[view]|

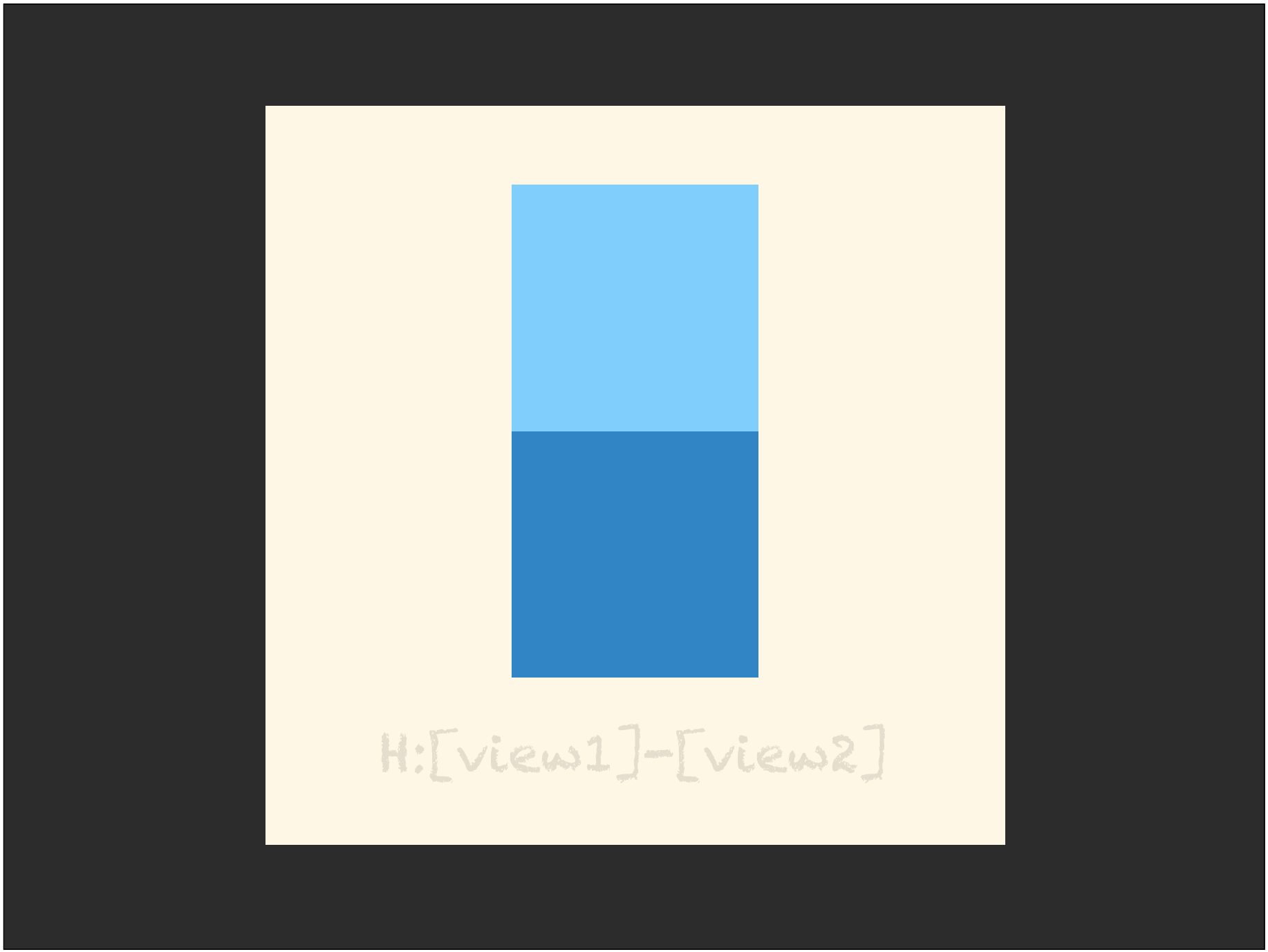

H:[view]
V:|[view]|

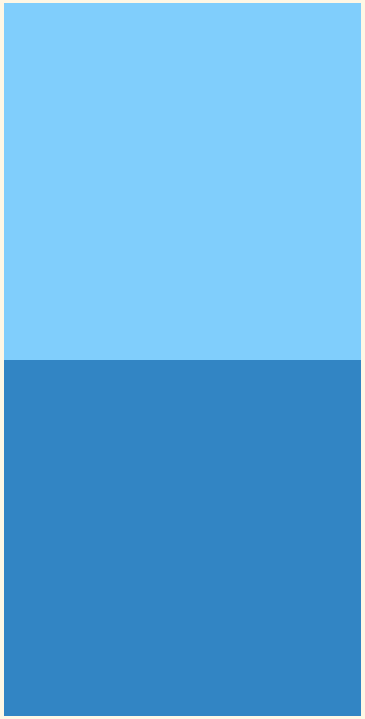

H:|[view]|
V:[view]


H:|[view]|
V:|[view]|



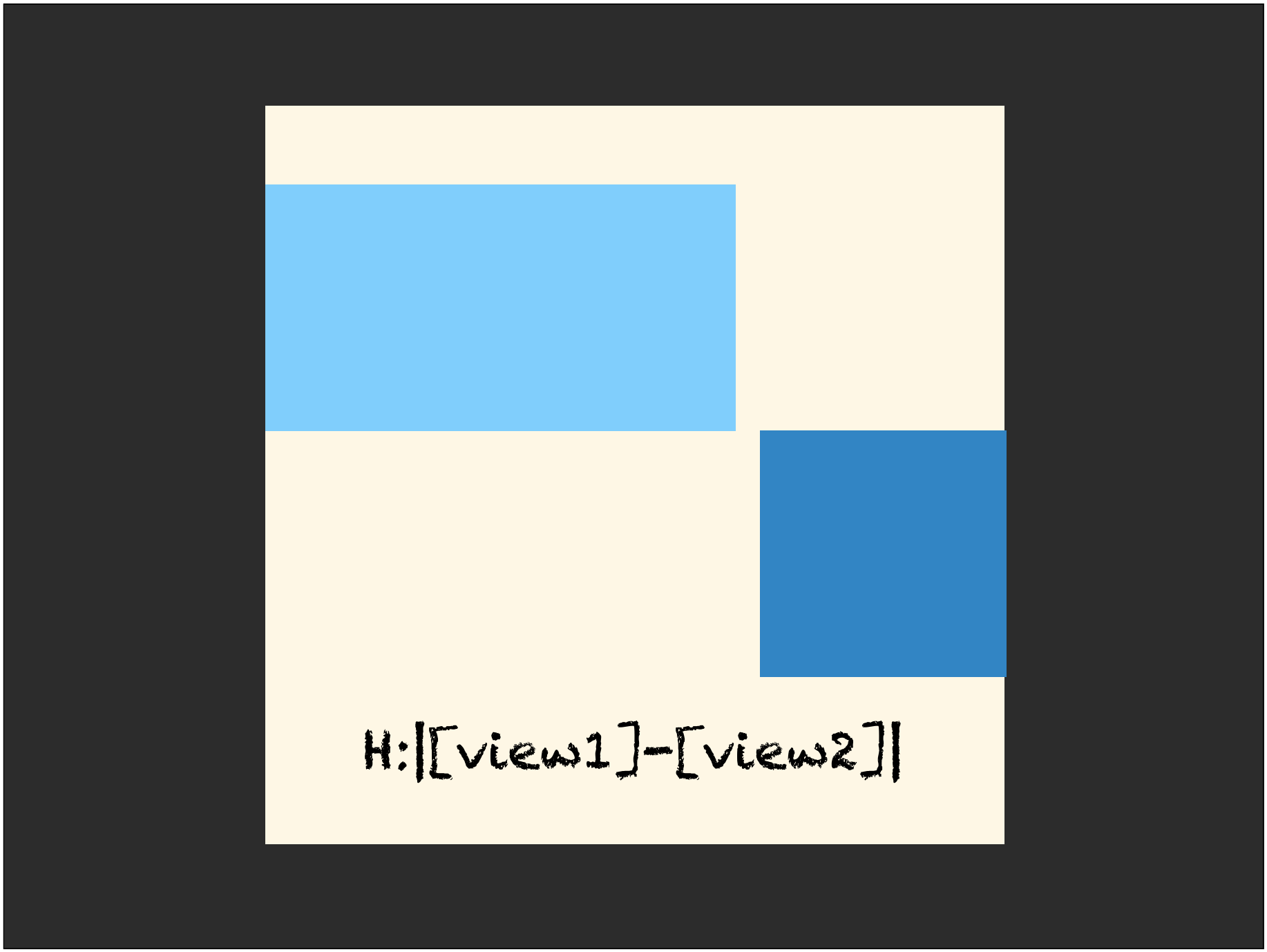

H:[view1][view2]

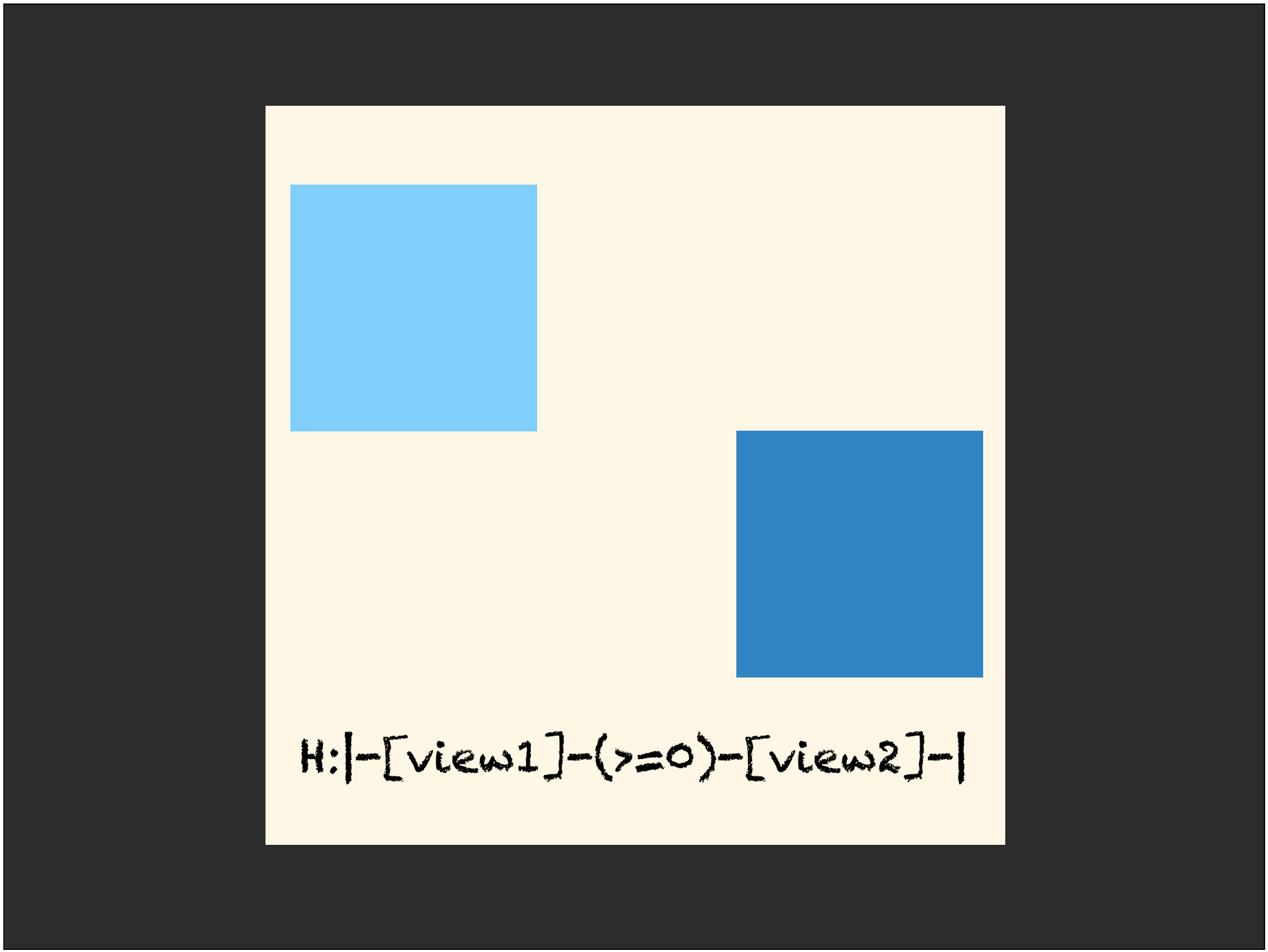

H:[view1]-[view2]

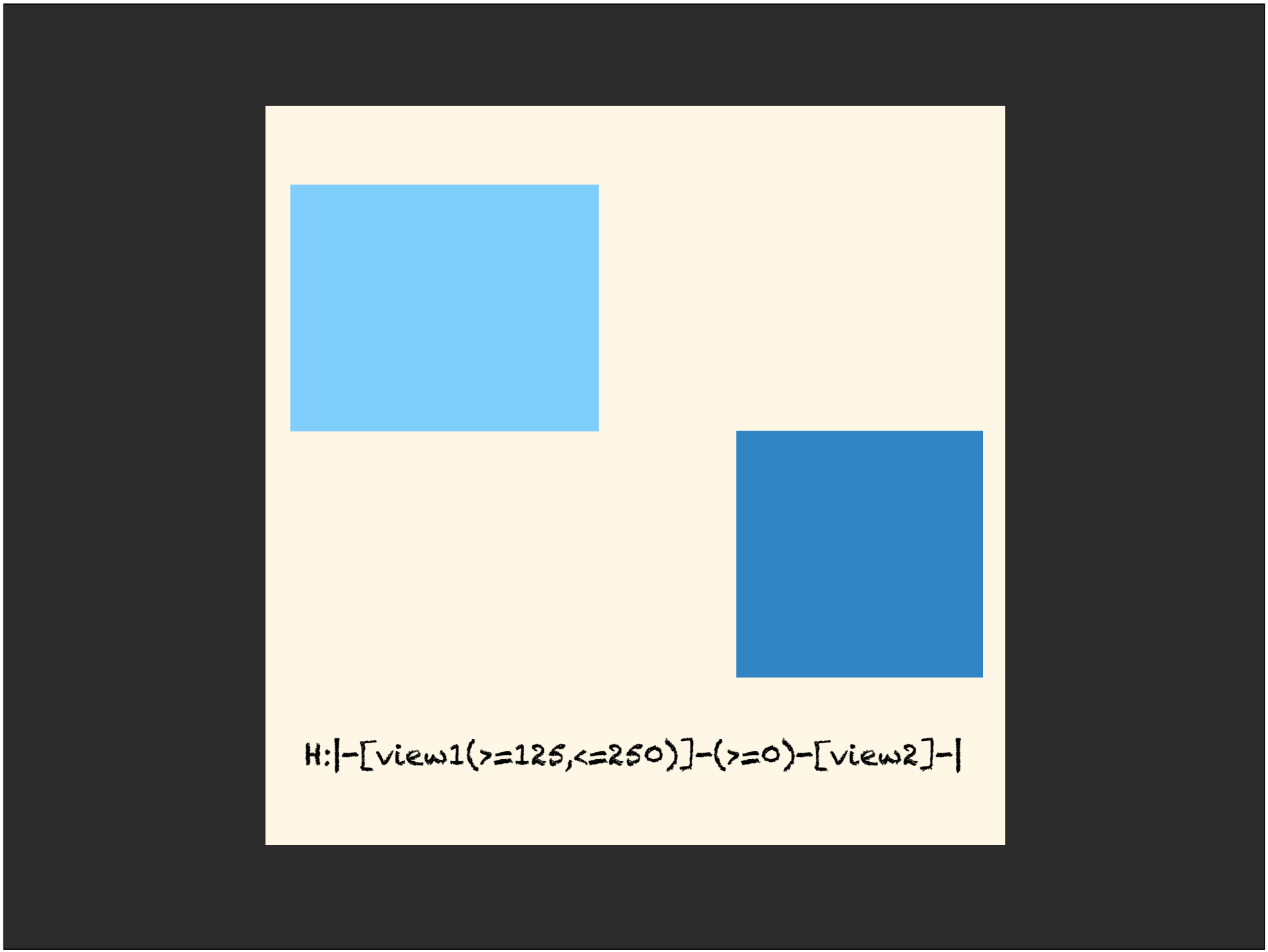


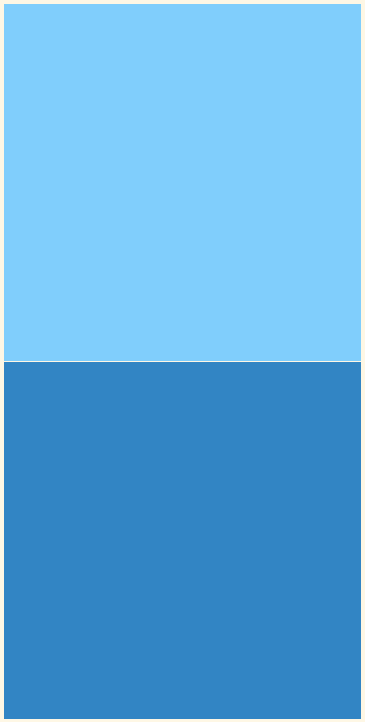
H:[view1]-30-[view2]

H:[view1]-(==30)-[view2]


H:|[view1]-[view2]|


H:|-[view1]-(>=0)-[view2]-|


H:|-[view1(>=125,<=250)]-(>=0)-[view2]-|



H:[view1(>=view2)][view2]

H:[button(100@20)]

H:|[view1]-(>=50@30)-[view2]|

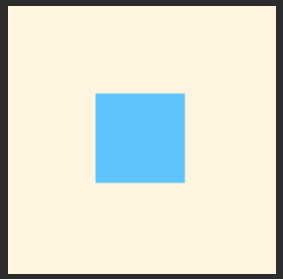
H:|-[view1(==view2)]-[view2]-|

H:[view1(view2)]

...

100x100 Square

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    self.blueView.translatesAutoresizingMaskIntoConstraints = NO;

    [self.blueView setContentHuggingPriority:UILayoutPriorityDefaultHigh
forAxis:UILayoutConstraintAxisHorizontal];
    [self.blueView setContentCompressionResistancePriority:UILayoutPriorityDefaultHigh
forAxis:UILayoutConstraintAxisVertical];

    [self.blueView removeConstraints:self.blueView.constraints];
    [self.blueView.superview removeConstraints:self.blueView.superview.constraints];

    NSArray *constraints = @[ @"H:[blueView(100)]", @"V:[blueView(100)]"];
    NSDictionary *views = @{@"blueView":self.blueView};
    for (NSString *format in constraints)
    {
        [self.view addConstraints:
         [NSLayoutConstraint
          constraintsWithVisualFormat: format
                              options: 0
                              metrics: nil
                                views: views]];
    }
}
```



UIView properties
Interface Builder > VFL > API
Animation

UIView API

UIView API

Opting in to Constraint-Based Layout

```
+ requiresConstraintBasedLayout
– translatesAutoresizingMaskIntoConstraints
– setTranslatesAutoresizingMaskIntoConstraints:
```

Managing Constraints

```
– constraints
– addConstraint:
– addConstraints:
– removeConstraint:
– removeConstraints:
```

Measuring in Constraint-Based Layout

```
– systemLayoutSizeFittingSize
– intrinsicContentSize
– invalidateIntrinsicContentSize
– contentCompressionResistancePriorityForAxis:
– setContentCompressionResistancePriority:forAxis:
– contentHuggingPriorityForAxis:
– setContentHuggingPriority:forAxis:
```

UIView API

Aligning Views with Constraint-Based Layout

- `alignmentRectForFrame:`
- `frameForAlignmentRect:`
- `alignmentRectInsets`
- `viewForBaselineLayout`

Triggering Constraint-Based Layout

- `needsUpdateConstraints`
- `setNeedsUpdateConstraints`
- `updateConstraints`
- `updateConstraintsIfNeeded`

Debugging Constraint-Based Layout

- `constraintsAffectingLayoutForAxis:`
- `hasAmbiguousLayout`
- `exerciseAmbiguityInLayout`

CALayer API

CALayer

- `layoutIfNeeded`

UIViewController

viewDidLoad
- autolayout-
viewDidLayoutSubviews
viewDidAppear

[self.view layoutIfNeeded]

UIView properties

Interface Builder > VFL > API

Animation

Animation

Animation

#238: Animate the constant

constant

```
self.someConstraint.constant = 10.0;
[UIView animateWithDuration:0.25 animations:^{
    [self.view layoutIfNeeded];
}];
```

Animation

#238: Animate the constant.

#238: Call layoutIfNeeded in a block.

layoutIfNeeded

```
- (BOOL)continueTrackingWithTouch:(UITouch *)touch
withEvent:(UIEvent *)event
{
    CGPoint touchPoint = [touch locationInView:self];
    [UIView animateWithDuration:0.1f animations:^(){
        NSLayoutConstraint *constraint =
            [trackView constraintNamed:THUMB_POSITION_TAG];
        constraint.constant = touchPoint.x;
        [trackView layoutIfNeeded];
    }];
    return YES;
}
```

Animation

#238: Animate the constant.

#238: Call layoutIfNeeded in a block.

Animate layers instead of views.

Layer animation

```
// jumpy
[UIView animateWithDuration:0.3 delay:0
options:UIViewAnimationOptionAutoreverse
animations:^{
    v.transform = CGAffineTransformMakeScale(1.1, 1.1);
} completion:^(BOOL finished) {
    v.transform = CGAffineTransformIdentity;
}];

// smooth
CABasicAnimation* ba = [CABasicAnimation
animationWithKeyPath:@"transform"];
ba.autoreverses = YES;
ba.duration = 0.3;
ba.toValue = [NSValue
valueWithCATransform3D:CATransform3DMakeScale(1.1, 1.1, 1)];
[v.layer addAnimation:ba forKey:nil];
```

Animation

#238: Animate the constant.

#238: Call layoutIfNeeded in a block.

Animate layers instead of views.

Drop constraints, use autosizing masks.

Animation

#238: Animate the constant

#238: Call layoutIfNeeded in a block

Animate layers instead of views.

Drop constraints, use autosizing masks.

Use a container view.

Animation

#238: Animate the constant.

#238: Call layoutIfNeeded in a block.

Animate layers instead of views.

Drop constraints, use autosizing masks.

Use a container view.

Use constraints that don’t interfere.

Animation

#238: Animate the constant.
#238: Call layoutIfNeeded in a block.
Animate layers instead of views.
Drop constraints, use autosizing masks.
Use a container view.
Use constraints that don’t interfere.
Set frame in viewDidLayoutSubviews.

Animating Rotations

Fading in/out during rotation

```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)to
duration:(NSTimeInterval)duration
{
    // fade away old layout
    [UIView animateWithDuration:duration animations:^{
        for (UIView *view in @[settingsView, creditsView])
        view.alpha = 0.0f;
    }];
}

- (void) didRotateFromInterfaceOrientation:(UIInterfaceOrientation)from
{
    // update the layout for the new orientation
    [self updateViewConstraints];
    [self.view layoutIfNeeded];

    // fade in the new layout
    [UIView animateWithDuration:0.3f animations:^{
        for (UIView *view in @[settingsView, creditsView])
        view.alpha = 1.0f;
    }];
}
```

Update and animate changes

```
- (void) willAnimateRotationToInterfaceOrientation:
(UIInterfaceOrientation)to
duration:(NSTimeInterval)duration
{
   [UIView animateWithDuration:duration animations:^{
      [self updateViewConstraints];
      [self.view layoutIfNeeded];
   }];
}
```

Calling updates

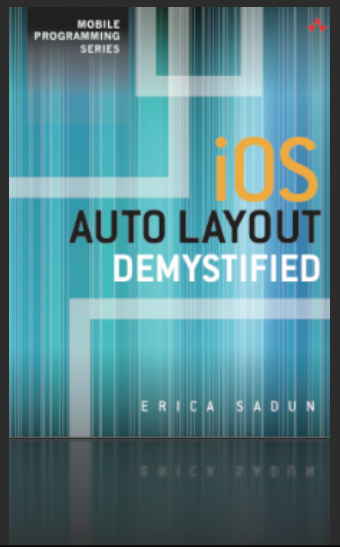
```
UIDeviceOrientation newOrientation;

- (void) updateViewConstraints
{
    [super updateViewConstraints];
    [self.view removeConstraints:self.view.constraints];
    if (newOrientation==UIDeviceOrientationPortrait){
        // ...
    }
}

- (void) willRotateToInterfaceOrientation:
(UIInterfaceOrientation)toInterfaceOrientation
duration:(NSTimeInterval)duration
{
    newOrientation = toInterfaceOrientation;
    [self updateViewConstraints];
}
```

References

#202 WWDC 2012: Introduction to Auto Layout for iOS and OS X
#228 WWDC 2012: Best Practices for Mastering Auto Layout
#232 WWDC 2012: Auto Layout by Example
#406 WWDC '13 Taking Control of Auto Layout in Xcode 5

Cocoa Auto Layout Guide

iOS Auto Layout Demystified



$16