Pace University - PCBT

Advanced iOS Class Spring 2015

AUTO LAYOUT INTRODUCTION

- This lecture will provide a background and context for using the Auto-Layout and Constraints in your iOS Apps.
- We will dicuss the history, purpose, and features.
- This lecture will also discuss many key topics such as adding layout by code as well as with IB/Storyboard..
- The lecture should be accompanied by an XCode exercise creating several projects using Storyboards.
- http://bit.ly/1yYJ4rL

What was wrong before?

- Called Springs and Struts
- Autosizing masks
- For example, with a flexible width the view will become proportionally wider if the superview also becomes wider. And with a fixed right margin, the view's right edge will always stick to the superview's right edge.

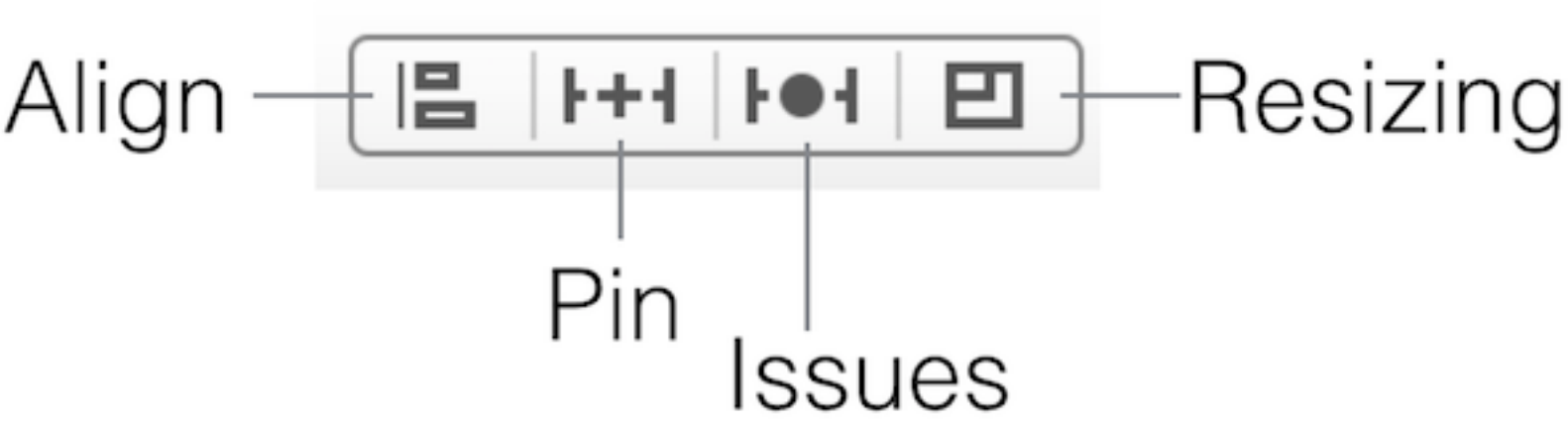
What is a Constraint?

- Represent relationships between views.
- When you drag out an element from the Object Library and drop it on the Interface Builder canvas, it starts out unconstrained.
- If you build and run without adding any constraints to an element, you'll find that Interface Builder fixes the element's width and height, and pins its position relative to the top left corner of the superview.
- To make your interface react correctly to changes in size or orientation, you need to start adding constraints.

Ideas

- For example, you might have a constraint that says: "The right edge of label A is connected to the left edge of button B with 20 points of empty space between them."'
- With Auto Layout you arrange your app's UI using relations between UI elements. These relations are called constraints.

The AutoLayout Tool



The Menu

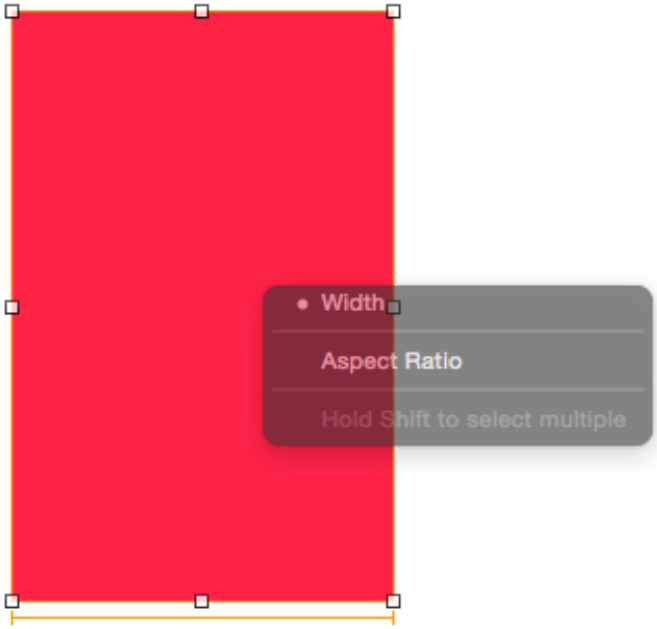
- Align – Create alignment constraints, such as aligning the left edges of two views
- Pin – Create spacing constraints, such as defining the width of a UI control.
- Issues – Resolve layout issues.
- Resizing – Specify how resizing affects constraints.

Intrinsic Content Size

- Leaf-level views such as buttons typically know more about what size they should be than does the code that is positioning them. This is communicated through the intrinsic content size, which tells the layout system that a view contains some content that it doesn't natively understand, and indicates how large that content is, intrinsically.

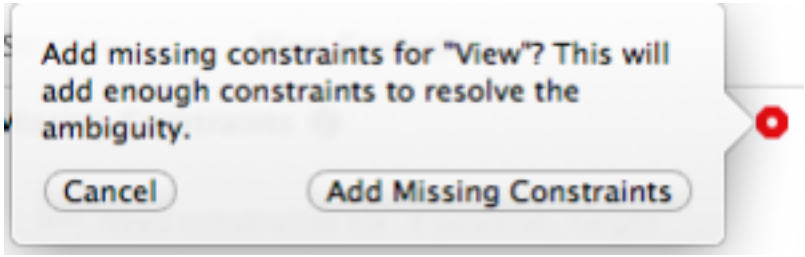
Setup a width constraint

- Hold down control, drag right within your view but don't leave the view.



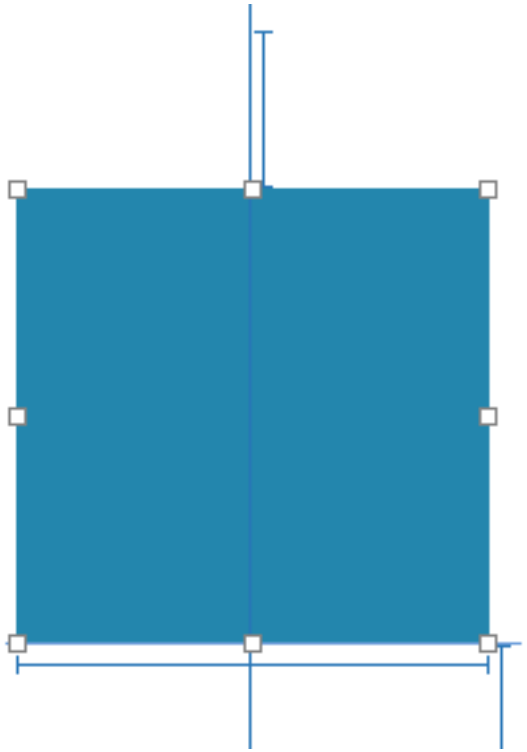
You are missing things.

- Once a constraint is set, Xcode is good about telling you what is missing.



Blue Good, Orange Bad

- Blue means all your constrains are set properly. Orange lines means that something is missing.



Setting up Constraints in Code

```
// 1. Create a dictionary of views
    NSDictionary *viewsDictionary = @{@"redView":self.redView};

    // 2. Define the redView Size
    NSArray *constraint_H = [NSLayoutConstraint constraintsWithVisualFormat:@"V:[redView(100)]"
                                                                    options:0
                                                                    metrics:nil
                                                                      views:viewsDictionary];

    NSArray *constraint_V = [NSLayoutConstraint constraintsWithVisualFormat:@"H:[redView(100)]"
                                                                    options:0
                                                                    metrics:nil
                                                                      views:viewsDictionary];
    [self.redView addConstraints:constraint_H];
    [self.redView addConstraints:constraint_V];
```

NSLayoutConstraint

- A single hyphen denotes the standard Aqua space, so you can also represent the relationship like this:
- [button1]-[button2] “aqua standard ui space”
- Have multipliers!
- V:|-vSpacing-[redView]-vSpacing-| “spacing before and after”
- H:|-hSpacing-[redView]-hSpacing-| “spacing before and after”

End of Lecture

rpascazio@pace.edu