Mobile Engineering

Day 02

Controls, Controls, Controls

Life Cycle

Tom Crawford

moveablebytes.com

@movebytes @thcrawford

Introductions

Introductions

Where do you live in now?
Where do you consider home?

If you could **live** 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

QXiz

Homework Review

More Code Basics

Data Types

int (whole numbers)

```
int myInt = 1;
NSNumber *myNumber = [NSNumber numberWithInt: 1];
NSNumber *myNumber = @1;
```

long (big whole numbers)

float (decimal numbers)

```
float myFloat = 2.345;
NSNumber *myNumber = [NSNumber numberWithFloat:2.345f];
NSNumber *myNumber = @2.345f;
```

double (big decimal numbers)

bool (yes/no, true/false, 1/0)

```
BOOL myBool = YES;
NSNumber *myNumber = [NSNumber numberWithBool:YES];
NSNumber *myNumber = @YES;
```

Operators

Operator	Description	Example
+	Add	x = x + y;
++	Add 1	x++;
+=	Add value to itself	x += y;
-	Subtract	x = x - y;
--	Subtract 1	x--;
-=	Subtract value from itself	x =- y;
*	Multiply	x = x * y;
*=	Multiply itself by value	x *= y;
/	Divide	x = x / y;
/=	Divide itself by value	x /= y;
%	Get remainder from division by value	x = x % y;
%=	Get remainder from division of self by value	x %= y;

The Life Cycle

App Delegate

- Called from main.m

  (generally, you should never touch main.m)
- Every app has one App Delegate (and only one)
- Controls the life cycle of the app as a whole

What's a Life Cycle?

Conceived

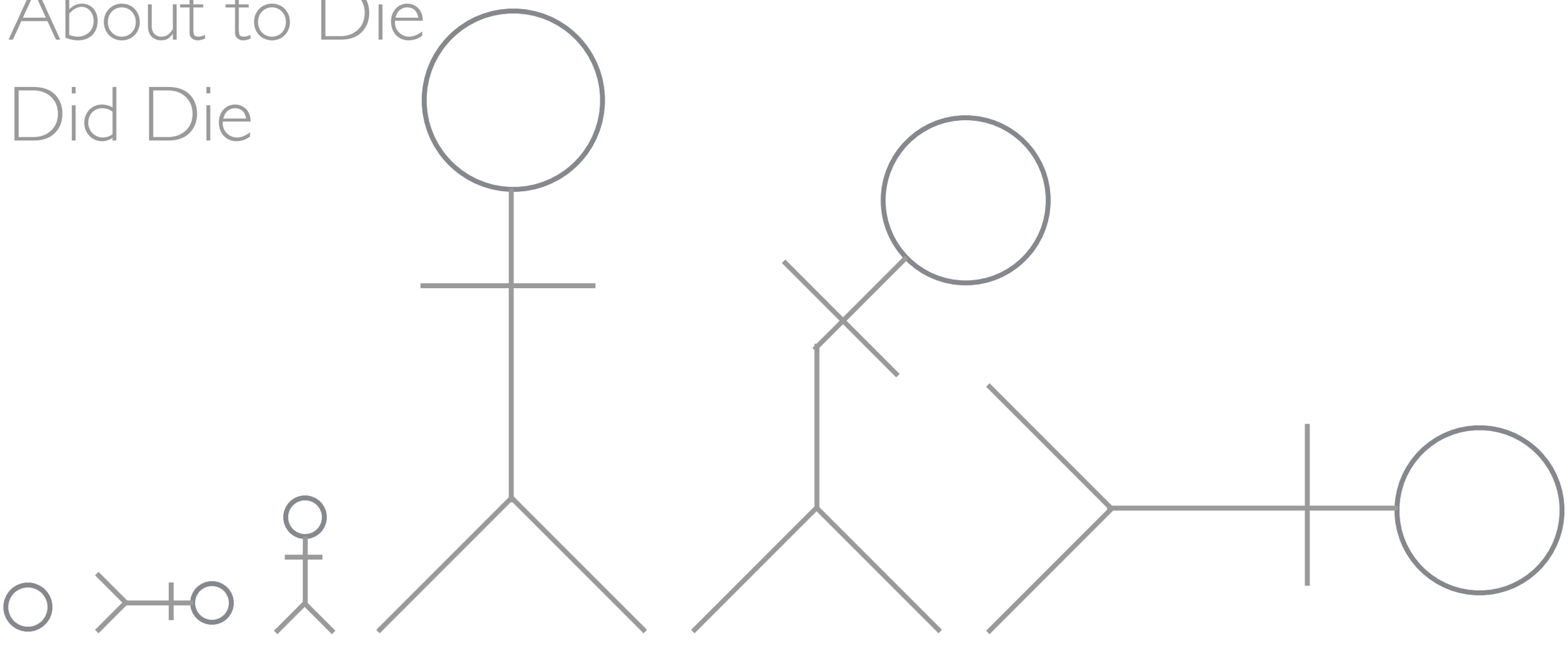
About to be Born

Was Born

Living

About to Die

Did Die



Application Life Cycle

application:willFinishLaunchingWithOptions: *(launch)*
application:didFinishLaunchingWithOptions: *(launch)*
(View Controller Life Cycle)
applicationWillEnterForeground: *(background)*
applicationDidBecomeActive: *(launch & background)*
applicationWillResignActive:
applicationDidEnterBackground:
applicationWillTerminate:

View Controller Class

- Pre-configured with all the basic things View Controllers need to show and do
- Hold all the properties and methods for everything you can see and do currently
- However, it knows nothing about the custom things you want to show and do
- Public (.h) vs. Private (.m)

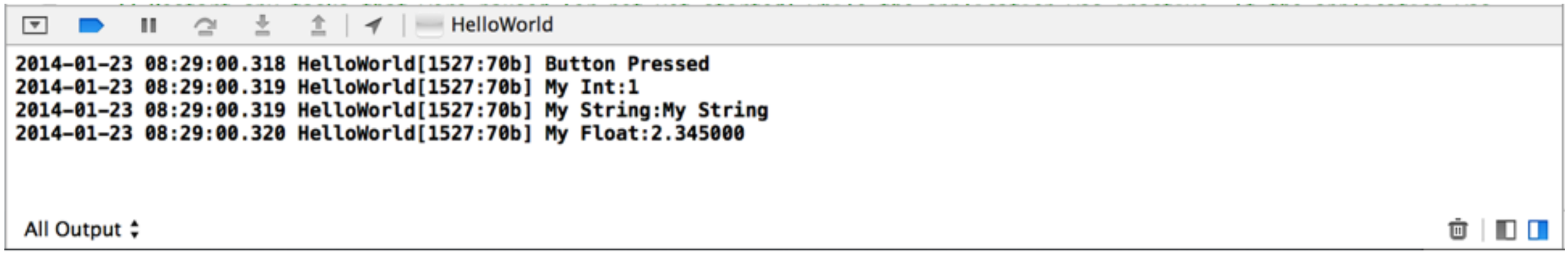
View Controller Life Cycle

viewDidLoad
viewWillAppear
viewDidAppear
(user does stuff)
viewWillDisappear
viewDidDisappear
viewWillUnload *(deprecated in iOS 6)*
viewDidUnload *(deprecated in iOS 6)*

If you override a method, call it's "super" first

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do stuff here
}
```

Debug Area



Super Basic Testing & Debugging:

```
NSLog(@"Button Pressed”);

int myInt = 1;
NSLog(@"My Int:%i",myInt);

NSString *myString = @"My String";
NSLog(@"My String:%@",myString);

float myFloat = 2.345;
NSLog(@"My Float:%0f",myFloat);
```

LifeCycle App

LifeCycle App Instructions

1. Create a new Single View app called LifeCycles
2. Add NSLogs in every Life Cycle stage

    application:willFinishLaunchingWithOptions:
    application:didFinishLaunchingWithOptions:
    applicationWillEnterForeground:
    applicationDidBecomeActive:
    applicationWillResignActive:
    applicationDidEnterBackground:
    applicationWillTerminate:
    viewDidLoad
    viewWillAppear
    viewDidAppear
    viewWillDisappear
    viewDidDisappear

3. Run the app, and play with the simulator
4. Notice the order of the messages

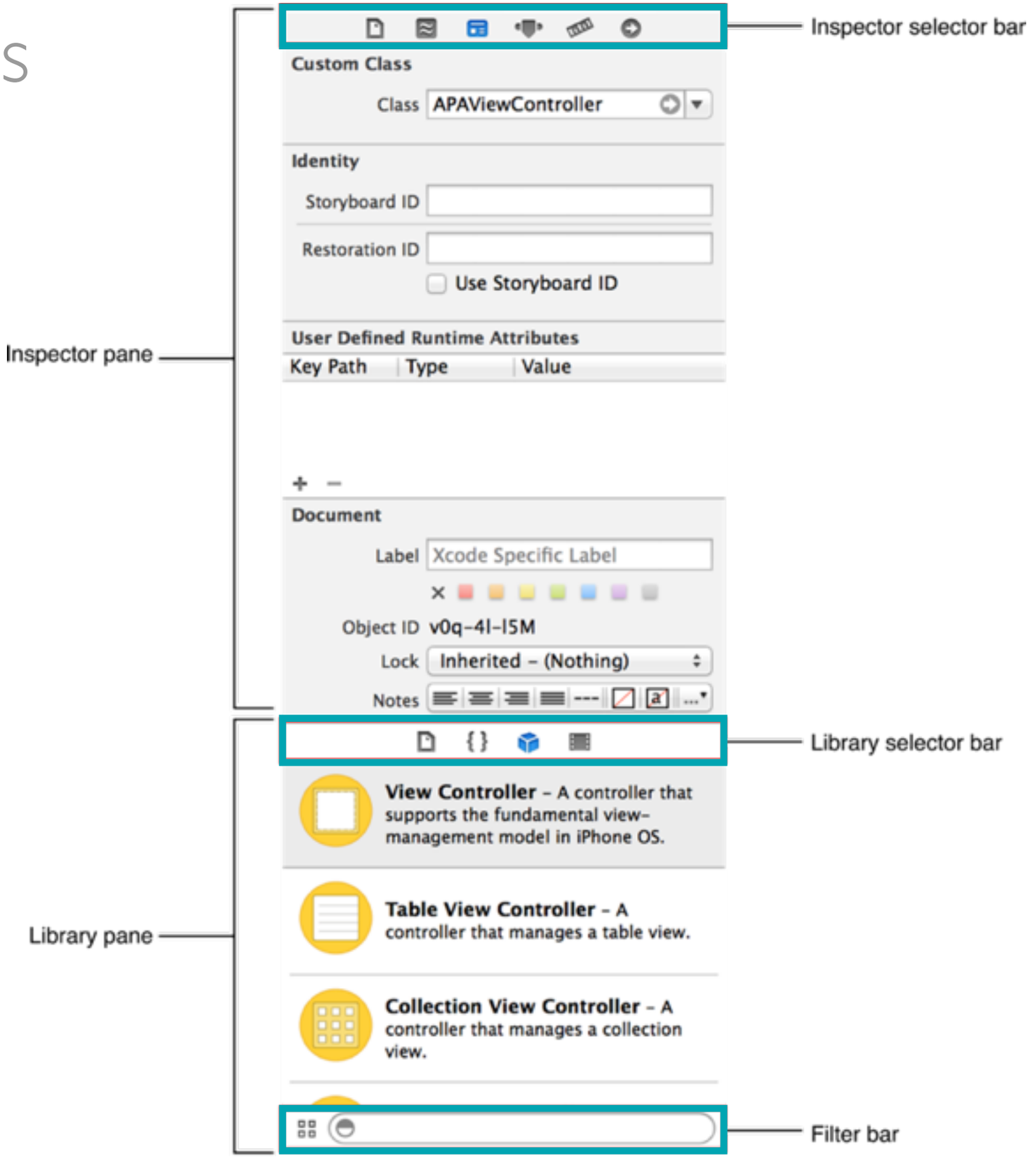
Control Properties

(adjusting in IB and in code)

Utility Area

Object properties

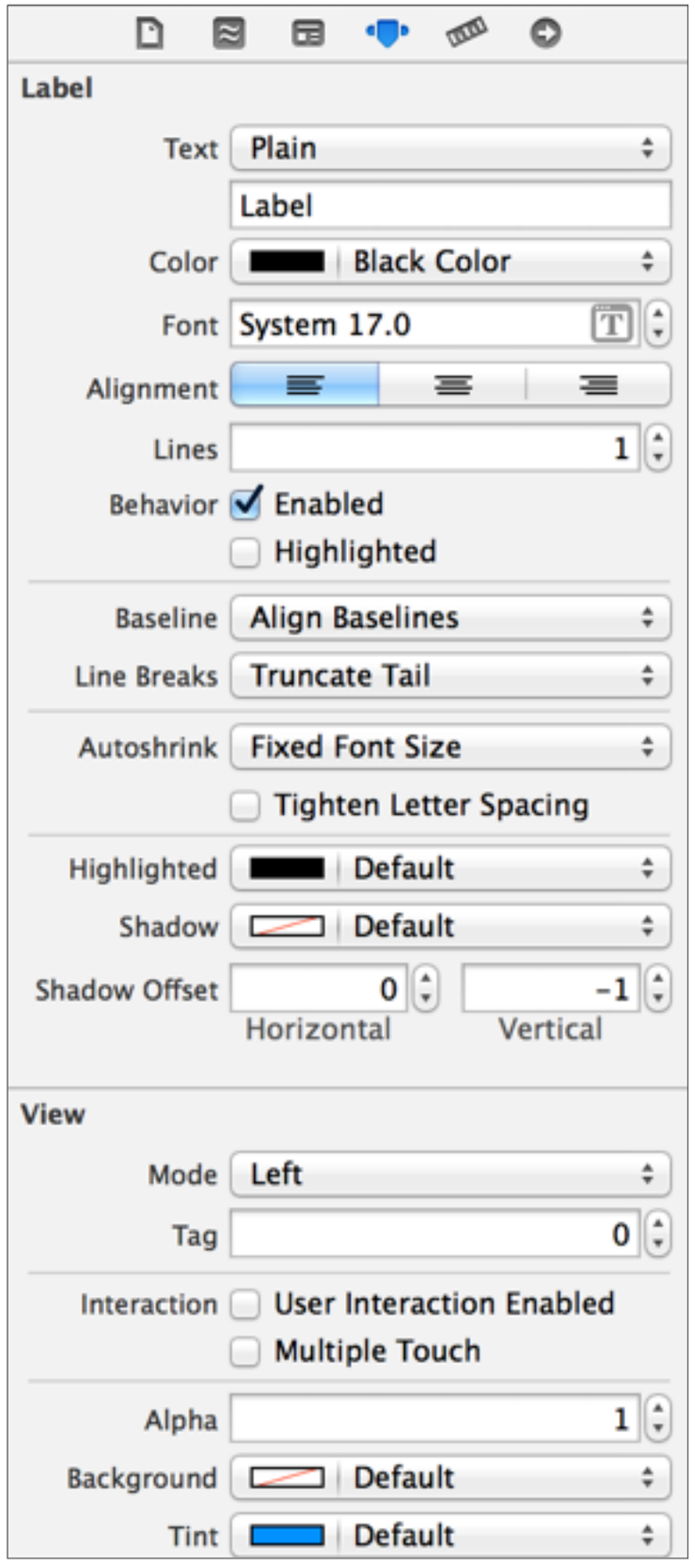
Object selection



IB Inspectors

Attributes Inspector:

- Every class has a different set of attributes
- Each class inherits attributes from it's parent, plus it's own
- So, not all attributes have meaning in all objects :-/
- Will discuss attributes of each of the controls later
- If here, implies availability in code!

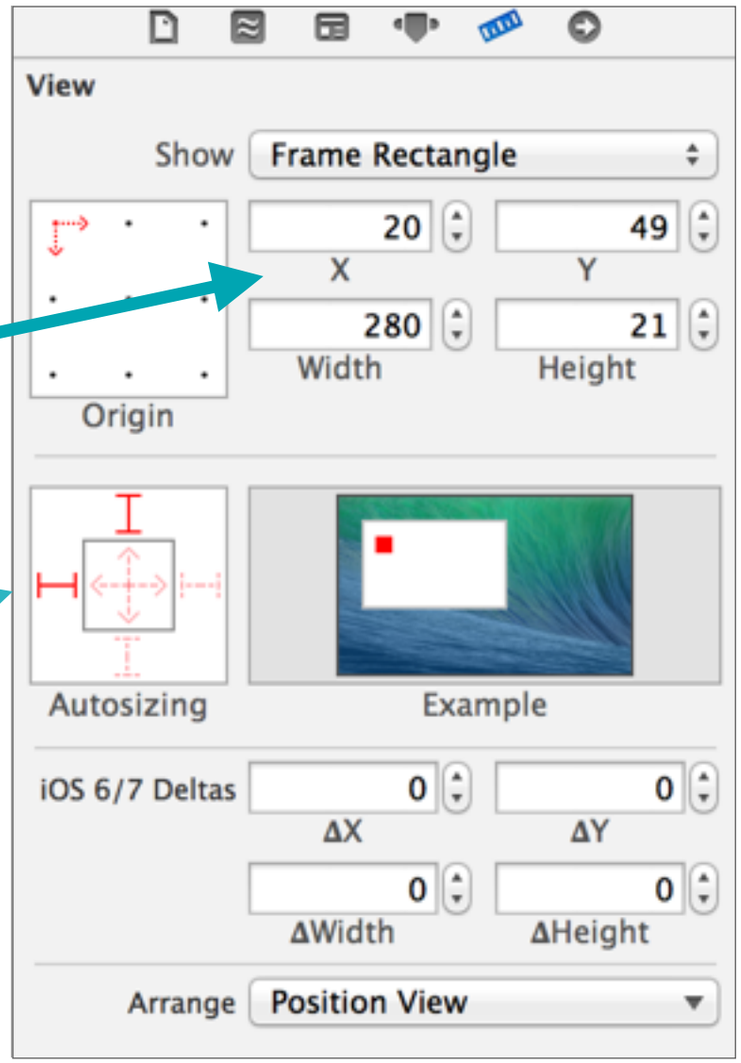


IB Inspectors

Size Inspector (Autosizing):

Determines the position and size of an object

Determines how the object moves and resizes when it's parent changes size (usually on rotation, but there are other cases)

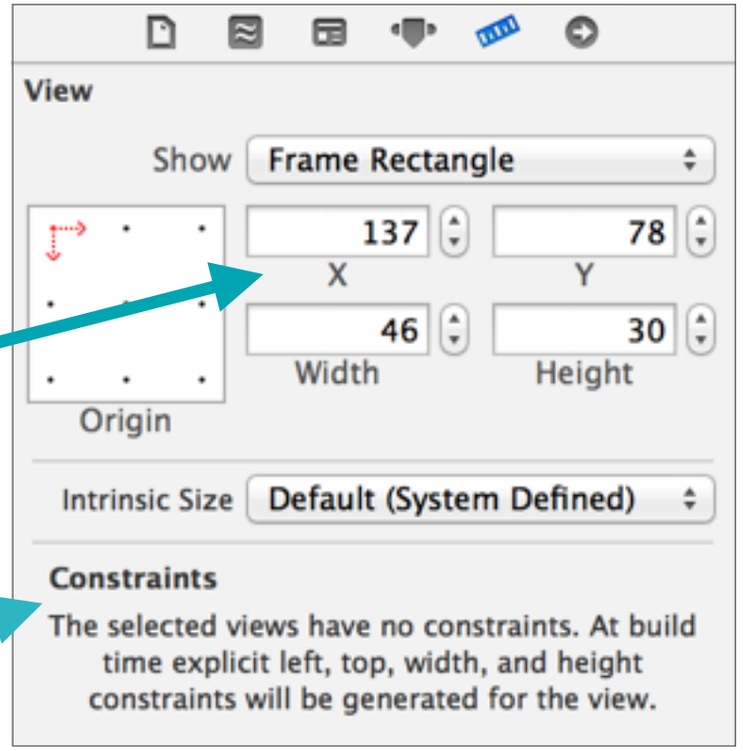


IB Inspectors

Size Inspector (AutoLayout):

Determines the initial position and size of an object, but…

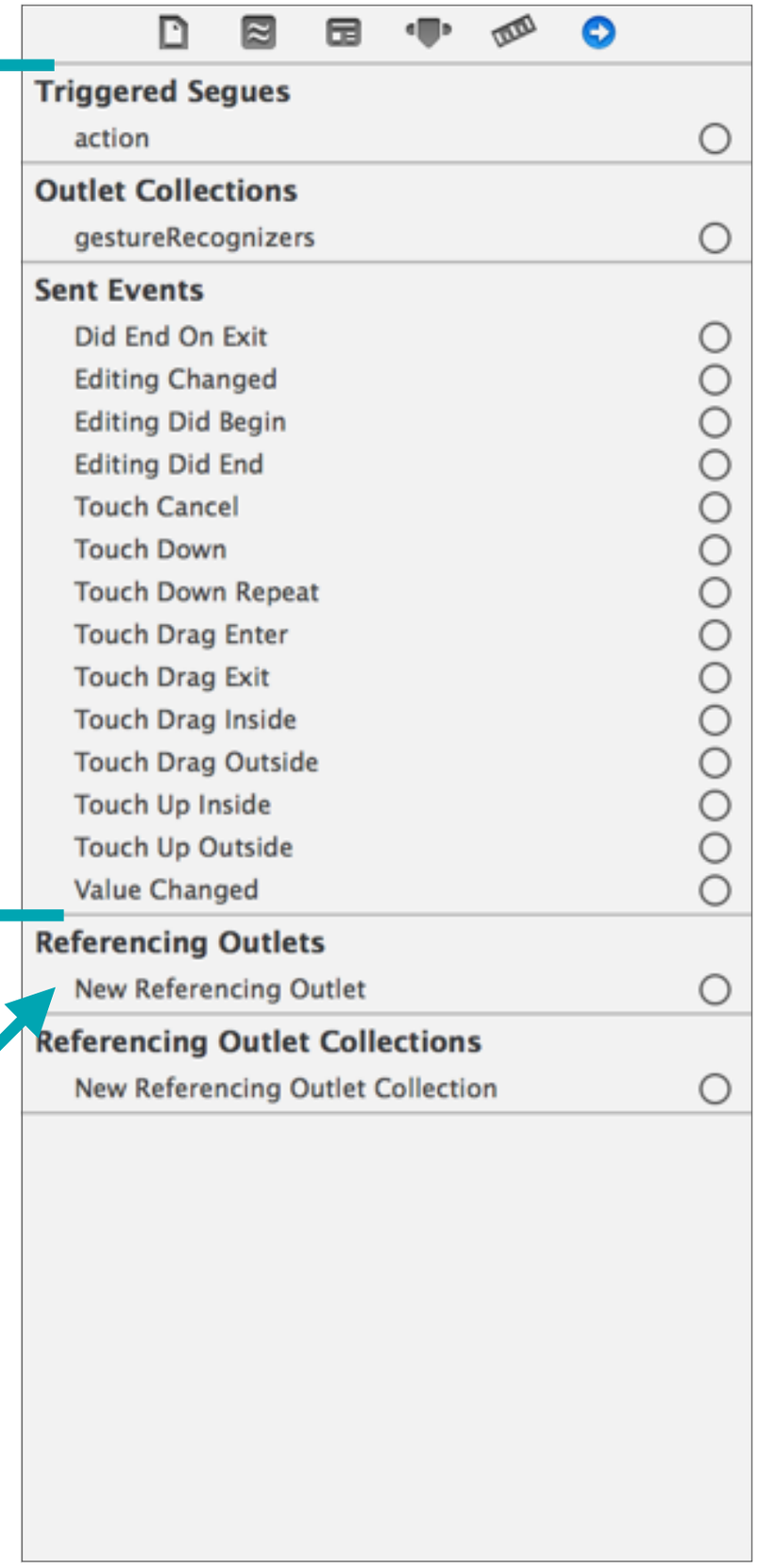
The constraints (and their relationship with other objects and the parent) can (and will) change the values as the views are laid out (and re-laid out) usually on rotation, but there are other cases



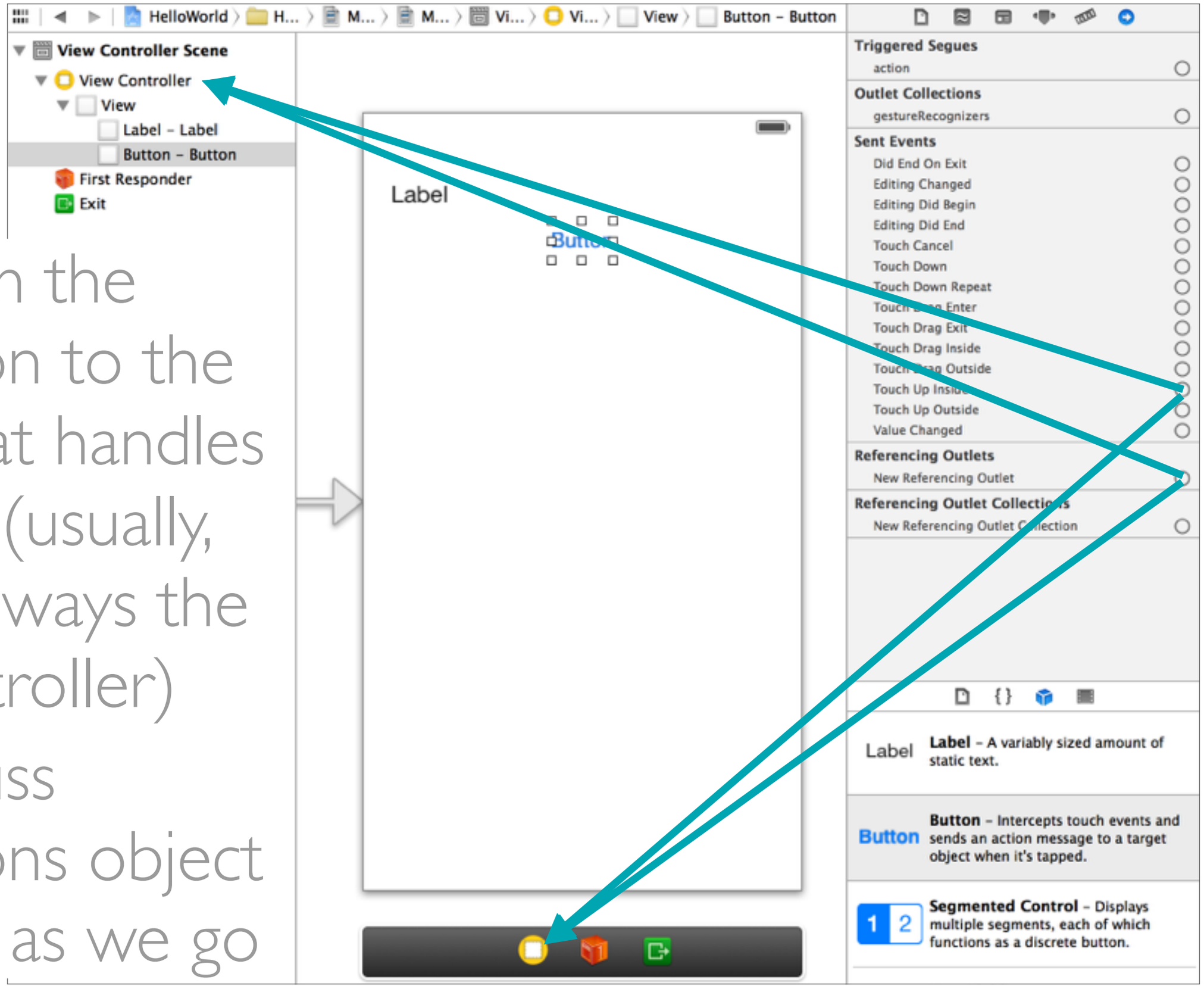
IB Inspectors

Connections Inspector:

- Each class inherits connections from it's parent, plus it's own
- So, not all connections have meaning in all objects or contexts :-/
- Set how an object responds to interactivity
- Set an object's referencing outlet (i.e. name)



IB Inspectors



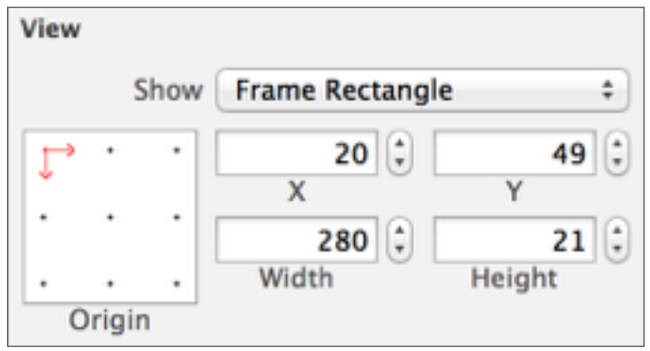
Drag from the connection to the object that handles the code (usually, but not always the view controller)

Will discuss connections object by object as we go

Setting UILabel Properties in IB

Properties

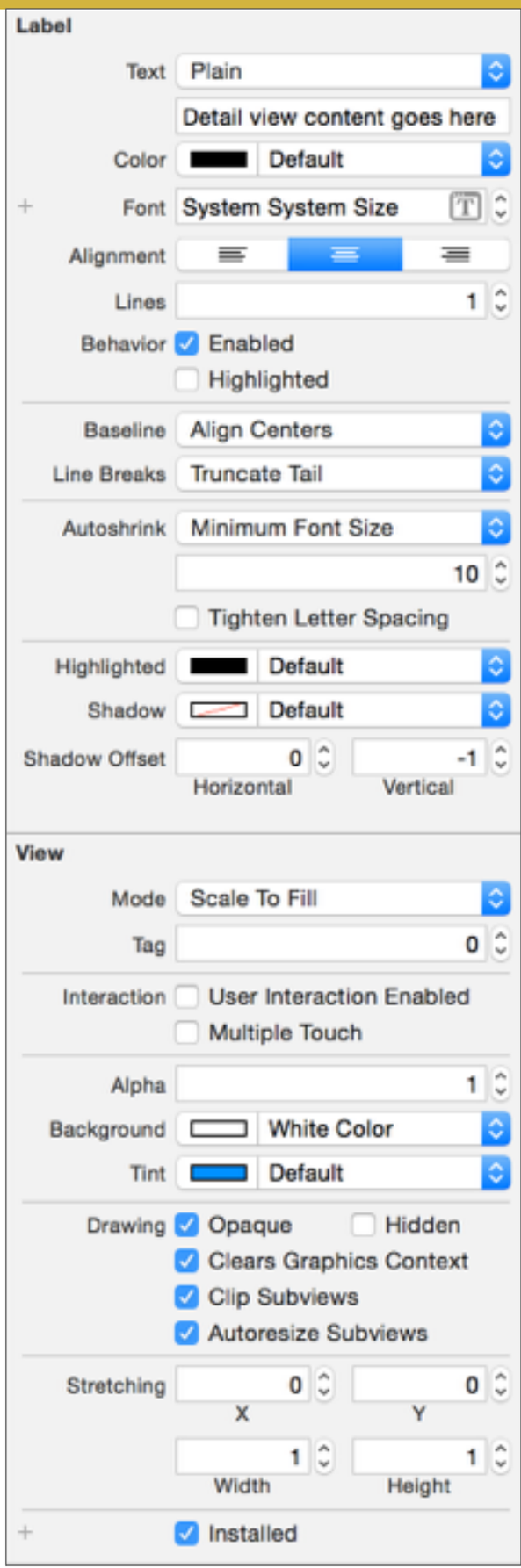
- Text or attributed text
- Color, Font, Alignment
- Number of lines
- Handle text that's too long
- Hidden, Alpha, Background color
- Tag
- Origin, Size



Super Easy, just change them

But if it needs to change? Needs code

- (e.g. be red when bad, green when good, plus some prefer code :-/)



Objective-C Tangent

Object-oriented

Thin layer on top of C++, C

Can use *any* C code

There's almost always more than one way!

History back to Smalltalk and NeXTStep

Cocoa Touch runs on top of Objective-C

Dot vs. Bracket Notation

Generally equivalent

```
myLabel.text = @"Label text";
[myLabel setText:@"Label text”];
```

But not always

```
myButton.titleLabel.text = @"Button Title”;
```

```
[myButton setTitle:@"Button Title”
      forState:UIControlStateNormal];
```

Do what you want. Some prefer consistency.

I mix them. Know and be able to use both.

Setting UILabel Properties in Code

Common properties to set in code

Text

```
_myLabel.text = @"Text goes here";
```

Color, Font, Background color

```
_myLabel.textColor = [UIColor redColor];
_myLabel.backgroundColor = [UIColor clearColor];
_myLabel.font = [UIFont fontWithName:@"Futura" size:17.0];
```

Hidden, Alpha

```
[_myLabel setHidden:YES];
[_myLabel setAlpha:0.0];
```

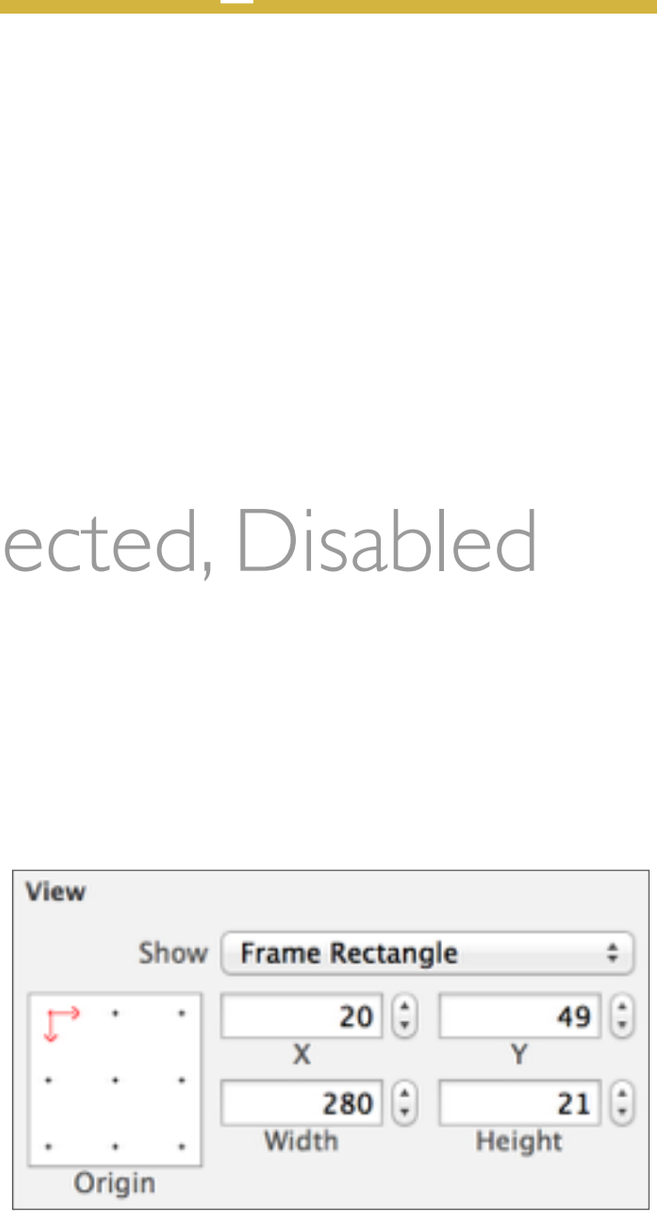
Origin, Size

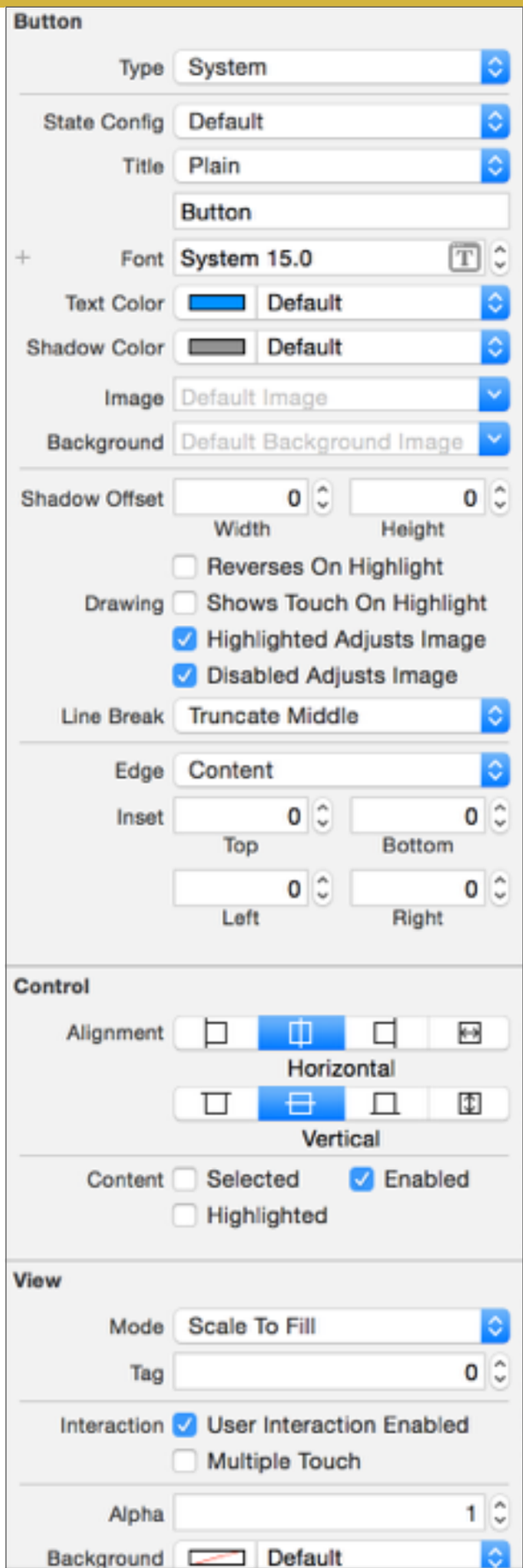
```
_myLabel.frame = CGRectMake(10.0, 10.0, 50.0, 21.0);
```

Setting UIButton Properties in IB

Properties

- Type
- Title or attributed title
- State - Default, Highlighted, Selected, Disabled
- Color, Font, Alignment
- Image, Background image
- Handle text that's too long
- Hidden, Alpha
- Background color
- Tag
- Origin, Size





Setting UIButton Properties in Code

Common properties to set in code

Title

```
[_myButton setTitle:@"Button Title”
    forState:UIControlStateNormal];
```

Color, Font

```
[_myButton setTitleColor:[UIColor greenColor]
    forState:UIControlStateNormal];
[[_myButton titleLabel] setFont:
    [UIFont fontWithName:@"Menlo" size:12.0]];
```

Image

```
[_myButton setImage:[UIImage imageNamed:@“myImage.png"]
    forState:UIControlStateSelected];
```

Hidden, Alpha

```
[_myButton setHidden:YES];
[_myButton setAlpha:0.5];
```

Origin, Size

```
[_myButton setFrame:CGRectMake(10.0, 35.0, 40.0, 30.0)];
```

Setting UIButton Events in Code

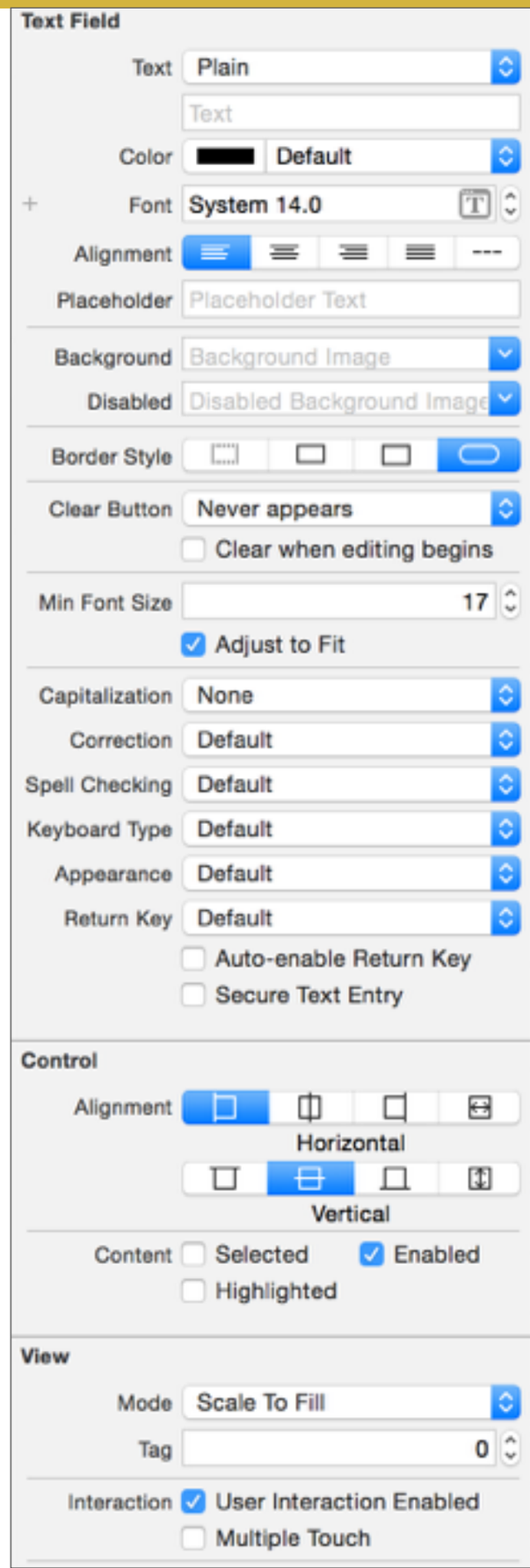
Event

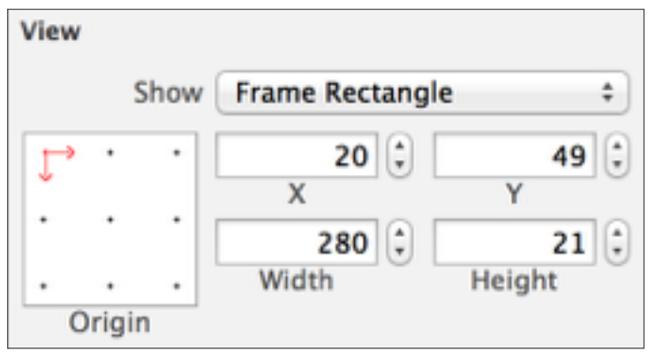
```
[_myButton addTarget:self action:@selector(myButtonPressed:)
    forControlEvents:UIControlEventTouchUpInside];
```

Setting UITextField Properties in IB

Properties

- Title or attributed title
- Color, Font, Alignment
- Placeholder text
- Clear button
- Handle text that's too long
- Capitalization, Correction, Keyboard type,
  Keyboard appearance, Return key, & Security
- Hidden, Alpha
- Tag
- Origin, Size





Setting UITextField Properties in Code

Common properties to set in code

Title

```
_myTextField.text = @"My Text";
```

Color, Font

```
_myTextField.textColor = [UIColor colorWithRed:25.0/255.0
    green:125.0/255.0 blue:225.0/255.0 alpha:1.0];
_myTextField.font = [UIFont systemFontOfSize:17.0];
```

Hidden, Alpha

```
[_myTextField setHidden:YES];
[_myTextField setAlpha:0.75];
```

Origin, Size

```
[_myTextField setFrame:
    CGRectMake(10.0, 55.0, 100.0, 30.0)];
```

HelloWorld +Custom

HelloWorld+Custom Instructions

1. Open the HelloWorld App
2. Use both IB and code to change the colors, fonts, background color, text, and other properties of the buttons, text fields, label

Kill Interface Builder?

Pros to Keeping Interface Builder

1. Quick
2. Easy
3. WYSI(almost)WYG
4. Much, much less code
5. Powerful prototyping tool that easily moves to app development (unlike other prototyping tools)

Pros to Killing Interface Builder

1. Fewer files (if you use Xibs)
2. Easier to collaborate on sites like gitHub
3. Temporary controls don't hold memory
4. Some people prefer code
5. Sometimes, code is the only good option

(but even then it's probably a hybrid)

Creating an Object in Code

All object's created in code need to be Allocated (alloc) and Initialized (init)

(when created in IB, Apple does it for you)

You do them together

```
UILabel *myLabel = [[UILabel alloc] init];
```

Some objects have helper inits that make things easier

```
myLabel.frame = CGRectMake(10.0, 10.0, 40.0, 21.0);
// This helper combines two lines of code
UILabel *myLabel = [[UILabel alloc] initWithFrame:
    CGRectMake(10.0, 10.0, 40.0, 21.0)];
```

In either case, iOS uses this to allocate memory to the object

Creating an Object in Code

If memory is allocated, how is it released?

In the old days (3 years ago), we had to dealloc objects, set them to nil, and a variety of other memory management.

Now **ARC**: **A**utomatic **R**eference **C**ounting, iOS increments and decrements counters for objects. When it goes to zero, its killed. We do nothing

This is awesome! Be thankful! Very thankful!

Creating an Object in Code

We set objects' properties in IB, but we can also do that in code:

```
UILabel *newLabel = [[UILabel alloc]
    initWithFrame:CGRectMake(10.0, 20.0, 40.0, 21.0)];
newLabel.textColor = [UIColor grayColor];
newLabel.font = [UIFont fontWithName:@"Futura" size:16.0];
newLabel.text = @"My Label”;
```

Then all we have to do is add it to the view:

```
[self.view addSubview:newLabel];
```

HelloWorld +Label

HelloWorld+Label Instructions

1. Open the HelloWorld App
2. Add a label to the view in code

Yet More Code Basics

What's Up with the Asterisk?

Ceci n'est pas une pipe.

Magritte

1600 Pennsylvania Avenue NW
Washington, DC, 20500

Ceci n'est pas une lieu

What's Up with the Asterisk?

The Asterisk is a Pointer to an Object

```
NSNumber *myNumber = @1;
```

Rather than passing an object around, the OS passes a pointer to the memory location where the object resides

Strings

String Manipulation

Declare

```
NSString *firstNameString = @"Mary";
NSString *lastNameString = @"Jones";
```

Concatenate (put two strings together)

```
NSString *nameString = [firstNameString
    stringByAppendingString:lastNameString];
// MaryJones
NSString *nameStringCorrect = [firstNameString
    stringByAppendingFormat:@" %@“,lastNameString];
// Mary Jones
NSString *sortNameString = [NSString stringWithFormat:
    @"%@, %@“,lastNameString,firstNameString];
// Jones, Mary
```

String Manipulation

Formatting

```
NSString *nameString = [NSString
    stringWithFormat:@“Last Name:%@ Age:%i Weight:%0.1f“,
    lastNameString,age,weight];
// Last Name:Jones Age:32 Weight:132.4
NSString *nameStringCorrect = [firstNameString
    stringByAppendingFormat:
    @" %@ Age:%i Weight:%0.1f“,lastNameString,age,weight];
// Mary Jones Age:32 Weight:132.4
```

Formats

%@ - Objects

%i - Integer

%f - Float (%0.2f yields 3.14, %2.0f yields 03)

Date/Time is an Object, and has additional formats

String Manipulation

Search

```
NSString *myString = @"Text Goes Here";
NSString *searchString = @" ";
int firstLoc = [myString rangeOfString:searchString].location;
// 4
```

Left

```
NSString *leftString = [myString substringToIndex:firstLoc];
// Text
```

Length

```
int myStringLen = [myString length];
// 14
int searchStringLen = [searchString length];
// 1
```

String Manipulation

Middle

```
int secondLoc = [myString rangeOfString: searchString
      options: NSCaseInsensitiveSearch
      range: NSMakeRange(firstLoc + searchLen,
      myStringLength - firstLoc - searchLen)].location;
// 9
NSString *midString = [myString substringWithRange:
      NSMakeRange(firstLoc + searchLen,
      secondLoc - firstLoc - searchLen)];
// Goes
```

Right

```
NSString *rightString = [myString substringFromIndex:
      secondLoc + searchLen];
// Here
```

String Manipulation

Attributed Strings

```
NSMutableAttributedString *myAttString =
    [[NSMutableAttributedString alloc]
    initWithString:@"Text Goes Here"];
NSRange midRange = NSMakeRange(firstLoc + searchLen,
    secondLoc - firstLoc - searchLen);
[myAttString addAttribute: NSForegroundColorAttributeName
    value:[UIColor redColor] range:midRange];
[myAttString addAttribute:NSFontAttributeName
    value:[UIFont fontWithName:@"Helvetica-Bold" size:20.0]
    range:midRange];
_myTextField.attributedText = myAttString;
```

Homework

Git Workflow

Homework Git Workflow

1. Create a new directory in your Homework folder for this project
2. Find the Homework in the class organization on GitHub
3. Fork the homework to your GitHub account
4. Copy the link from the 'HTTPS clone URL' in GitHub
5. Open SourceTree
    1. Select '+ New Repository'
    2. 'Clone from URL'
    3. Use the URL you copied as the source, the directory you created in the Homework folder as the Destination & click 'Clone'
6. Create a new project in Xcode (**without** Git) and save it in the new directory that you cloned (note: it should have the README.md in it)
7. Commit as major pieces are done
8. Push as you want to have a backup on the server & when done

Homework