Contents

1 Introspections

- `isKindOfClass:` returns whether an object is that kind of class (inheritance included)
- `isMemberOfClass:` returns whether an object is that kind of class (no inheritance)
- `respondsToSelector:` returns whether an object responds to a given method

1.1 Selectors

```
// Using the performSelector: or performSelector:withObject: methods in NSObject
[obj performSelector:shootSelector];
[obj performSelector:shootAtSelector withObject:coordinate];
// Using makeObjectsPerformSelector: methods in NSArray
[array makeObjectsPerformSelector:shootSelector]; // cool, huh
[array makeObjectsPerformSelector:shootAtSelector withObject:target]; // target is an id
// In UIButton,
- (void)addTarget:(id)anObject action:(SEL)action ...;
[button addTarget:self action:@selector(digitPressed:) ...];
```

2 UIFonts

It is best to get a UIFont by asking for the preferred font for a given text style ...

```
UIFont *font = [UIFont preferredFontForTextStyle:UIFontTextStyleBody];
```

Some other styles (see `UIFontDescriptor` documentation for even more styles) ...

`UIFontTextStyleHeadline`, `UIFontTextStyleCaption1`, `UIFontTextStyleFootnote`, etc.

3 NSAttributedString

You can ask an `NSAttributedString` all about the attributes at a given location in the string.

```
(NSDictionary *)attributesAtIndex: (NSUInteger)indexeffectiveRange:(NSRangePointer)range;
```

Sample code:

```
UIColor *yellow = [UIColor yellowColor];
UIColor *transparentYellow = [yellow colorWithAlphaComponent:0.3];
 @{ NSFontAttributeName :
```

```
   [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline],
   NSForegroundColorAttributeName : [UIColor greenColor],
   NSStrokeWidthAttributeName : @-5,
   NSStrokeColorAttributeName : [UIColor redColor],
   NSUnderlineStyleAttributeName : @(NSUnderlineStyleNone),
   NSBackgroundColorAttributeName : transparentYellow }
//You could use transparent colors in other attributes as well.
```

The range is returned and it lets you know for how many characters the attributes are identical.

4 UITextView

Obtain the NSMutableAttributedString representing the text in the UITextView using ...

```
@property (nonatomic, readonly) NSTextStorage *textStorage;
```

`NSTextStorage` is a subclass of `NSMutableAttributedString`.

There is also a property that applies a font to the entire UITextView:

```
@property (nonatomic, strong) UIFont *font;}
```

5 View Controller Lifecycle

5.1 viewDidLoad

After instantiation and outlet-setting, viewDidLoad is called This is an exceptionally good place to put a lot of setup code.

```
- (void)viewDidLoad
{
   [super viewDidLoad]; // always let super have a chance in lifecycle methods
   // do some setup of my MVC
}
```

But be careful because the geometry of your view (its bounds) is not set yet!

At this point, you can't be sure you're on an iPhone 5-sized screen or an iPad or . So do not initialize things that are geometry-dependent here.

5.2 viewWillAppear

Just before the view appears on screen, you get notified (argument is just whether you are appearing instantly or over time via animation)

```
- (void)viewWillAppear:(BOOL)animated;
```

Your view will only get loaded once, but it might appear and disappear a lot. So don't put something in this method that really wants to be in `viewDidLoad`. Otherwise, you might be doing something over and over unnecessarily. Do something here if things you display are changing while your MVC is off-screen.

5.3 viewWillDisappear

And you get notified when you will disappear off screen too This is where you put "remember what's going on" and cleanup code.

```
- (void)viewWillDisappear:(BOOL)animated
{
   [super viewWillDisappear:animated]; // call super in all the viewWill/Did... methods //
      let's be nice to the user and remember the scroll position they were at ...
   [self rememberScrollPosition]; // we'll have to implement this, of course // do some
      other clean up now that we've been removed from the screen
   [self saveDataToPermanentStore]; // maybe do in did instead
```

```
  // but be careful not to do anything time-consuming here, or app will be sluggish
  // maybe even kick off a thread to do what needs doing here (again, we'll cover threads
      later)
}
```

5.4 awakeFromNib

Anything that would go in your Controller's init method would have to go in awakeFromNib too (because init methods are not called on objects that come out of a storyboard).

```
- (void)setup{}; //do something which can't wait until viewDidLoad
- (void)awakeFromNib { [self setup]; }
```

6 NSNotification

Get the default notification center via `[NSNotificationCenter defaultCenter]`. Then send it the following message if you want to listen to a radio station:

```
- (void)addObserver:(id)observer // you (the object to get notified)
          selector:(SEL)methodToInvokeIfSomethingHappens
              name:(NSString *)name // name of station (a constant somewhere)
            object:(id)sender; // whose changes youre interested in (nil is anyones)
```

You will then be notified when there are broadcasts

```
- (void)methodToInvokeIfSomethingHappens:(NSNotification *)notification
{
  notification.name // the name passed above
  notification.object // the object sending you the notification notification.userInfo //
      notification-specific information about what happened
}
```

Be sure to tune out when done listening

```
[center removeObserver:self];
```

or

```
[center removeObserver:self name:UIContentSizeCategoryDidChangeNotification object:nil];
```

Failure to remove yourself can sometimes result in crashers. This is because the NSNotificationCenter keeps an unsafe retained pointer to you. A good place to remove yourself is when your MVCs View goes off screen.

7 UINavigationController

Usually because the user presses the back button (shown on the previous slide). But it can happen programmatically as well with this UINavigationController instance method

```
- (void)popViewControllerAnimated:(BOOL)animated;
```

This does the same thing as clicking the back button. Somewhat rare to call this method. Usually we want the user in control of navigating the stack. But you might do it if some action the user takes in a view makes it irrelevant to be on screen.

Sometimes it makes sense to segue directly when a button is touched, but not always.

For example, what if you want to conditionally segue

You can programmatically invoke segues using this method in UIViewController:

```
- (void)performSegueWithIdentifier:(NSString *)segueId sender:(id)sender;
```

The segueId is set in the attributes inspector in Xcode (seen on previous slide).

The sender is the initiator of the segue (a UIButton or yourself (UIViewController) usually).

The segue offers the source VC the opportunity to prepare the new VC to come on screen. This method is sent to the VC that contains the button that initiated the segue:

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([segue.identifier isEqualToString:@DoSomething]) {
        if ([segue.destinationViewController isKindOfClass:[DoSomethingVC class]]) {
            DoSomethingVC *doVC = (DoSomethingVC *)segue.destinationViewController;
            doVC.neededInfo = ...;
        }
    }
}
```

Sometimes (very rarely) you might want to put a VC on screen yourself (i.e., not use a segue).

```
NSString *vcid = @something;
UIViewController *controller = [storyboard instantiateViewControllerWithIdentifier:vcid];
```

Usually you get the storyboard above from self.storyboard in an existing `UIViewController`. The identifier vcid must match a string you set in Xcode to identify a `UIViewController` there.

Example: creating a `UIViewController` in a target/action method. Lay out the View for a `DoitViewController` in your storyboard and name it doit1.

```
- (IBAction)doit
{
   //Note use of self.navigationController again.
    DoitViewController *doit = [self.storyboard
        instantiateViewControllerWithIdentifier:@doit1];
      doit.infoDoitNeeds = self.info;
      [self.navigationController pushViewController:doit animated:YES];
}
```

8 Views

The hierarchy is most often constructed in Xcode graphically Even custom views are often added to the view hierarchy using Xcode.

But it can be done in code as well

```
- (void)addSubview:(UIView *)aView; // sent to aViews (soon to be) superview
- (void)removeFromSuperview; // sent to the view that is being removed
```

The top of this hierarchy for your MVC is

```
UIViewControllers @property (strong, nonatomic) UIView *view
```

This is the view whose bounds will be changed when autorotation happens. This is the view you would programmatically add subviews to. All your MVCs Views UIViews eventually have this view as their parent (its at the top). It is automatically hooked up for you when you drag out a View Controller in Xcode.

8.1 Initializing a `UIVIew`

You will also want to set up stuff in awakeFromNib This is because initWithFrame: is NOT called for a UIView coming out of a storyboard! But awakeFromNib is. Same as we talked about with UIViewController. Its called awakeFromNib for historical reasons. Typical code ...

```
- (void)setup { ... }
```

```
- (void)awakeFromNib { [self setup]; } - (id)initWithFrame:(CGRect)aRect
{
    self = [super initWithFrame:aRect];
    [self setup];
    return self;
}
```

8.2 View coordinates

Use `CGPointMake, CGSizeMake, CGRectMake` to create a new point / size / rectangle Views have 3 properties related to their location and size

```
@property CGRect bounds; // your views internal drawing spaces origin and size
```

The bounds property is what you use inside your views own implementation. It is up to your implementation as to how to interpret the meaning of bounds.origin.

```
@property CGPoint center; // the center of your view in your superviews coordinate space
@property CGRect frame; // a rectangle in your superviews coordinate space which entirely
    contains your views bounds.size
```

Use frame and center to position the view in the hierarchy. These are used by superviews, never inside your UIView subclasss implementation.

How do you create a UIView in code (i.e. not in Xcode) Just use alloc and initWithFrame: (UIViews designated initializer). Can also use init (frame will be CGRectZero).

```
CGRect labelRect = CGRectMake(20, 20, 50, 30);
UILabel *label = [[UILabel alloc] initWithFrame:labelRect]; label.text = @Hello!;
[self.view addSubview:label]; // Note self.view!
```

Drawing is easy ... create a `UIView` subclass and override 1 method

```
- (void)drawRect:(CGRect)aRect;
```

You can optimize by not drawing outside of aRect if you want (but not required).

NEVER call drawRect:!! EVER! Or else! Instead, let iOS know that your views visual is out of date with one of these UIView methods:

```
- (void)setNeedsDisplay;
- (void)setNeedsDisplayInRect:(CGRect)aRect;
```

It will then set everything up and call drawRect: for you at an appropriate time. Obviously, the second version will call your `drawRect:` with only rectangles that need updates.

8.3 Context

`UIBezierPath` draws into the current context, so you dont need to get it if using that. But if youre calling Core Graphics C functions directly, youll need it (its an argument to them). Call the following C function inside your drawRect: method to get the current graphics context:

```
CGContextRef context = UIGraphicsGetCurrentContext();
```

Create a path with `UIBezierPath`:

- Begin the path.

```
*path = [[UIBezierPath alloc] init];
```

- Move around, add lines or arcs to the path

```
[path moveToPoint:CGPointMake(75, 10)];
[path addLineToPoint:CGPointMake(160, 150)];
[path addLineToPoint:CGPointMake(10, 150]);
```

- Close the path (connects the last point back to the first)

```
  [path closePath];
  // not strictly required but triangle wont have all 3 sides otherwise
```

- Now that the path has been created, we can stroke/fill it. Actually, nothing has been drawn yet, weve just created the UIBezierPath.

```
  [[UIColor greenColor] setFill];
  [[UIColor redColor] setStroke];
  [path fill];
  [path stroke];
```

And draw rounded rects, ovals, etc.

```
UIBezierPath *roundedRect = [UIBezierPath bezierPathWithRoundedRect:(CGRect)bounds
   cornerRadius:(CGFloat)radius];
```

Note: the casts in the arguments are just to let you know the types (i.e. theyre not required).

```
UIBezierPath *oval = [UIBezierPath bezierPathWithOvalInRect:(CGRect)bounds]; [roundedRect
    stroke];
[oval fill];
```

`UIView` also has a `backgroundColor` property which can be set to transparent values. Be sure to set `@property BOOL opaque` to `NO` in a view which is partially transparent. If you dont, results are unpredictable (this is a performance optimization property, by the way). `UIViews @property CGFloat alpha` can make the entire view partially transparent. (you might use this to your advantage in your homework to show a disabled custom view)

Also, you can hide a view completely by setting hidden property

```
@property (nonatomic) BOOL hidden;
myView.hidden = YES; // view will not be on screen and will not handle events
```

What if you wanted to have a utility method that draws something You dont want that utility method to mess up the graphics state of the calling method. Use save and restore context functions.

```
- (void)drawGreenCircle:(CGContextRef)ctxt {
    CGContextSaveGState(ctxt);
    [[UIColor greenColor] setFill];
// draw my circle
    CGContextRestoreGState(ctxt);
}
- (void)drawRect:(CGRect)aRect {
    CGContextRef context = UIGraphicsGetCurrentContext();
    [[UIColor redColor] setFill];
// do some stuff
    [self drawGreenCircle:context];
// do more stuff and expect fill color to be red
}
```

9 UIImageView

Create a UIImage object from a file in your Resources folder

```
UIImage *image = [UIImage imageNamed:@foo.jpg];
```

Now blast the UIImages bits into the current graphics context

```
UIImage *image = ...;
[image drawAtPoint:(CGPoint)p];
[image drawInRect:(CGRect)r];
[image drawAsPatternInRect:(CGRect)patRect;
// p is upper left corner of the image // scales the image to fit in r
// tiles the image into patRect
Aside: You can get a PNG or JPG data representation of UIImage NSData *jpgData =
   UIImageJPEGRepresentation((UIImage *)myImage, (CGFloat)quality);
NSData *pngData = UIImagePNGRepresentation((UIImage *)myImage);
```

By default, when your UIViews bounds change, there is no redraw Instead, the bits of your view will be stretched or squished or moved.

Often this is not what you want ... Use `UIViewContentModeRedraw`.

Default is UIViewContentModeScaleToFill (stretch the bits to fill the bounds)

10 UIGestureRecognizer

Adding a gesture recognizer to a UIView from a Controller

```
- (void)setPannableView:(UIView *)pannableView // maybe this is a setter in a Controller
{
            _pannableView = pannableView;
            UIPanGestureRecognizer *pangr =
                [[UIPanGestureRecognizer alloc] initWithTarget:pannableView
                    action:@selector(pan:)];
            [pannableView addGestureRecognizer:pangr]; // remember to do this
}
```

Note that we are specifying the view itself as the target to handle a pan gesture when it is recognized. Thus the view will be both the recognizer and the handler of the gesture. The UIView does not have to handle the gesture. It could be, for example, the Controller that handles it. The View would generally handle gestures to modify how the View is drawn. The Controller would have to handle gestures that modified the Model.

11 Protocols

11.1 Declaring a @protocol

Looks a lot like @interface (but theres no corresponding @implementation)

```
@protocol Foo
- (void)someMethod;
- (void)methodWithArgument:(BOOL)argument;
@property (readonly) int readonlyProperty; // getter (only) is part of this protocol
@property NSString *readwriteProperty; // getter and setter are both in the protocol -
   (int)methodThatReturnsSomething;
@end
```

All of these methods are required. Anyone implementing this protocol must implement them all. Add keyword @optional / @required to state that the rest of the protocal declaration is optional / required.

```
@protocol Foo <Xyzzy>
```

Now you can only say you implement Foo if you also implement the methods in Xyzzy protocol.

11.2 Where do @protocol declarations go

In header files.

It can go in its own, dedicated header file.

Or it can go in the header file of the class that is going to require its use. Which to do

If the @protocol is only required by a particular classs API, then put it there, else put it in its own header file.

Example: The UIScrollViewDelegate protocol is defined in UIScrollView.h.

11.3 Examples of protocol

11.3.1 ChooseYesOrNoView.h

```
ChooseYesOrNoView.h
#import

@protocol ChooseYesOrNoViewDelegate <NSObject>
- (void) chooseYesOrNoResponse: (BOOL) response;
@end
```

```
@interface ChooseYesOrNoView : UIView

@property (nonatomic, weak) id delegate;

- (ChooseYesOrNoView *) display;

@end
```

11.3.2 ChooseYesOrNoView.m

```
ChooseYesOrNoView.m

- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code
    }
    return self;
}

- (ChooseYesOrNoView *) display
{
    UIView *chooseView = [[UIView alloc]initWithFrame: CGRectMake(50,50,200,100)];
    [self addSubview:chooseView];
    UILabel *pleaseChooseLabel = [[UILabel alloc]initWithFrame: CGRectMake(0,0,260,20)];
    pleaseChooseLabel.text = @"Please choose Yes or No";
    [chooseView addSubview:pleaseChooseLabel];

    UIButton *yesButton = [UIButton buttonWithType:UIButtonTypeSystem];
    yesButton.tag = 1;
    [yesButton addTarget:self
                  action:@selector(buttonTapped:)
        forControlEvents:UIControlEventTouchDown];
    [yesButton setTitle:@"Yes" forState:UIControlStateNormal];
    yesButton.frame = CGRectMake(50, 60, 60, 40);
```

```
    [self addSubview:yesButton];

    UIButton *noButton = [UIButton buttonWithType:UIButtonTypeSystem];
    noButton.tag = 0;
    [noButton addTarget:self
                 action:@selector(buttonTapped:)
       forControlEvents:UIControlEventTouchDown];
    [noButton setTitle:@"No" forState:UIControlStateNormal];
    noButton.frame = CGRectMake(160, 60, 60, 40);
    [self addSubview:noButton];

    return self;
}

- (void) buttonTapped: (UIButton*) sender
{
    BOOL response;
    if(sender.tag==1) response = YES;
    else response = NO;
    [self.delegate chooseYesOrNoResponse:response];
}
```

11.3.3 MainViewController.m

```
MainViewController.m
@implementation MainViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
   // Do any additional setup after loading the view, typically from a nib.
    self.view.backgroundColor = [UIColor grayColor];
    ChooseYesOrNoView *chooseYesOrNoView = [[ChooseYesOrNoView alloc]
                          initWithFrame:CGRectMake(30,160,260,110)];
    chooseYesOrNoView.delegate = self;
    chooseYesOrNoView.backgroundColor = [UIColor whiteColor];
    chooseYesOrNoView = [chooseYesOrNoView display];
```

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


}



- (void) chooseYesOrNoResponse:(BOOL)response
{
    if(response) {
        NSLog(@"YES was chosen");
    }
    else {
        NSLog(@"NO was chosen");
    }
}
```

12 Blocks

12.1 Definition

Here's an example of calling a method that takes a block as an argument.

```
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog("@value for key %@ is %@", key, value);
    if ([@ENOUGH isEqualToString:key]) {
*stop = YES; }
}];
```

Blocks start with the magical character caret

```
^
```

Then (optional) return type, then (optional) arguments in parentheses, then {, then code, then }.

12.2 Creating a type for a variable that can hold a block

Blocks are kind of like objects with an unusual syntax for declaring variables that hold them. Usually if we are going to store a block in a variable, we typedef a type for that variable,

```
typedef double (^unary_operation_t)(double op);
```

This declares a block which takes a double as its only argument and returns a double. Then we could declare a variable, square, of this type and give it a value

```
unary_operation_t square;
square = ^(double operand) { // the value of the square variable is a block
    return operand * operand;
}
```

And then use the variable square like this ...

```
double squareOfFive = square(5.0); // squareOfFive would have the value 25.0 after this
```

(It is not mandatory to typedef, for example, the following is also a legal way to create square ...)

```
double (^square)(double op) = ^(double op) { return op * op; };
```

12.3 Example of block

12.3.1 ChooseYesOrNoView.h

```
ChooseYesOrNoView.h
#import
typedef void (^ChooseYesOrNoCallbackBlock) (BOOL);
@interface ChooseYesOrNoView : UIView
@property (nonatomic, strong) ChooseYesOrNoCallbackBlock callbackBlock;
- (ChooseYesOrNoView *) displayWithCallback:(ChooseYesOrNoCallbackBlock) block;
@end
```

12.3.2 ChooseYesOrNoView.m

```
ChooseYesOrNoView.m
@implementation ChooseYesOrNoView


- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code
    }
```

```
    return self;
}

- (ChooseYesOrNoView *) displayWithCallback:(ChooseYesOrNoCallbackBlock)block
{
    self.callbackBlock = block;
    UILabel *pleaseChooseLabel = [[UILabel alloc]initWithFrame: CGRectMake(0,20,260,20)];
    pleaseChooseLabel.text = @"Please choose Yes or No";
    [self addSubview:pleaseChooseLabel];

    UIButton *yesButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
    yesButton.tag = 1;
    [yesButton addTarget:self
                  action:@selector(buttonTapped:)
        forControlEvents:UIControlEventTouchDown];
    [yesButton setTitle:@"Yes" forState:UIControlStateNormal];
    yesButton.frame = CGRectMake(50, 60, 60, 40);
    [self addSubview:yesButton];

    UIButton *noButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
    noButton.tag = 0;
    [noButton addTarget:self
                  action:@selector(buttonTapped:)
        forControlEvents:UIControlEventTouchDown];
    [noButton setTitle:@"No" forState:UIControlStateNormal];
    noButton.frame = CGRectMake(160, 60, 60, 40);
    [self addSubview:noButton];

    return self;
}
- (void) buttonTapped: (UIButton*) sender
{
    BOOL response;
    if(sender.tag==1) response = YES;
    else response = NO;
    self.callbackBlock(response);
}
```

12.3.3 MainViewController.m

```
MainViewController.m
@implementation MainViewController


- (void)viewDidLoad
{
    [super viewDidLoad];
   // Do any additional setup after loading the view, typically from a nib.
    self.view.backgroundColor = [UIColor grayColor];
    ChooseYesOrNoView *chooseYesOrNoView = [[ChooseYesOrNoView alloc]
                         initWithFrame:CGRectMake(30,160,260,110)];
    chooseYesOrNoView.backgroundColor = [UIColor whiteColor];
    chooseYesOrNoView = [chooseYesOrNoView displayWithCallback:^(BOOL response)
                        {
                            if(response) {
                                NSLog(@"YES was chosen");
                            }
                            else {
                                NSLog(@"NO was chosen");
                            }
                        }];
    [self.view addSubview:chooseYesOrNoView];
}
```