

# Arrays

# Array Concepts

Kind of like a spreadsheet with only 1 column  
“Unlimited” rows  
Numbered starting at 0 (zero)  
Can hold any object  
(even different objects)

0	← @“Vanilla”
1	← @“Chocolate”
2	← @“Strawberry”
3	← @“Rocky Road”
4	← @123
5	← @3.14F
6	← @YES
7	

# NSArray

Traditional Objective-C (i.e. wordy)

```
NSArray *myArray = [NSArray arrayWithObjects:  
    @"Vanilla",@"Chocolate",  
    @"Strawberry",@"Rocky Road", nil];  
NSString *myFlavorString = [myArray objectAtIndex:0];
```

Modern Objective-C (i.e. slightly less wordy)

```
NSArray *myArray = @[@"Vanilla",@"Chocolate",  
    @"Strawberry",@"Rocky Road"];  
NSString *myFlavorString = myArray[0];
```

Initialized with another array, one or more objects

Cannot be changed once it's created :-/

# NSMutableArray

Pretty much the same as NSArray

```
NSMutableArray *myArray = [NSMutableArray arrayWithObjects:  
    @"Vanilla",@"Chocolate",  
    @"Strawberry",@"Rocky Road", nil];  
NSString *myFlavorString = [myArray objectAtIndex:0];
```

Initialized with another array, one or more objects,  
AND with capacity

More importantly can be changed (Mutable):

```
[myMutableArray addObject:@"Chocolate Peanut Butter"];  
[myMutableArray insertObject:@"Superman" atIndex:2];  
[myMutableArray removeObjectAtIndex:1];  
[myMutableArray removeLastObject];  
[myMutableArray removeAllObjects];  
[myMutableArray sortUsingSelector:  
    @selector(localizedCaseInsensitiveCompare:)];
```

# More Code Basics

# Comparison

Comparison	Description	Example
<code>==</code>	Equivalent (equal)	<code>x == y</code>
<code>&gt;</code>	Greater than	<code>x &gt; y</code>
<code>&gt;=</code>	Greater than or equal to	<code>x &gt;= y</code>
<code>&lt;</code>	Less than	<code>x &lt; y</code>
<code>&lt;=</code>	Less than or equal to	<code>x &lt;= y</code>
<code>!=</code>	Not equal to	<code>x != y</code>
<code>!</code>	Not (inverse true/false)	<code>!(x = y)</code>
<code>isEqualToString</code>	String comparison	<code>[x isEqualToString:y]</code>
<code>isEqualToDate</code>	Date comparison	<code>[x isEqualToDate:y]</code>
<code>compare</code>	Date comparison	<code>[x compare:y] == NSOrderedAscending</code>
<code>&amp;&amp;</code>	And	<code>x == y &amp;&amp; a == b</code>
<code>  </code>	Or	<code>x == y    x == z</code>

# Very Common Mistakes

`==` is **NOT** the same as `=`

`==` is comparison, `=` is assign

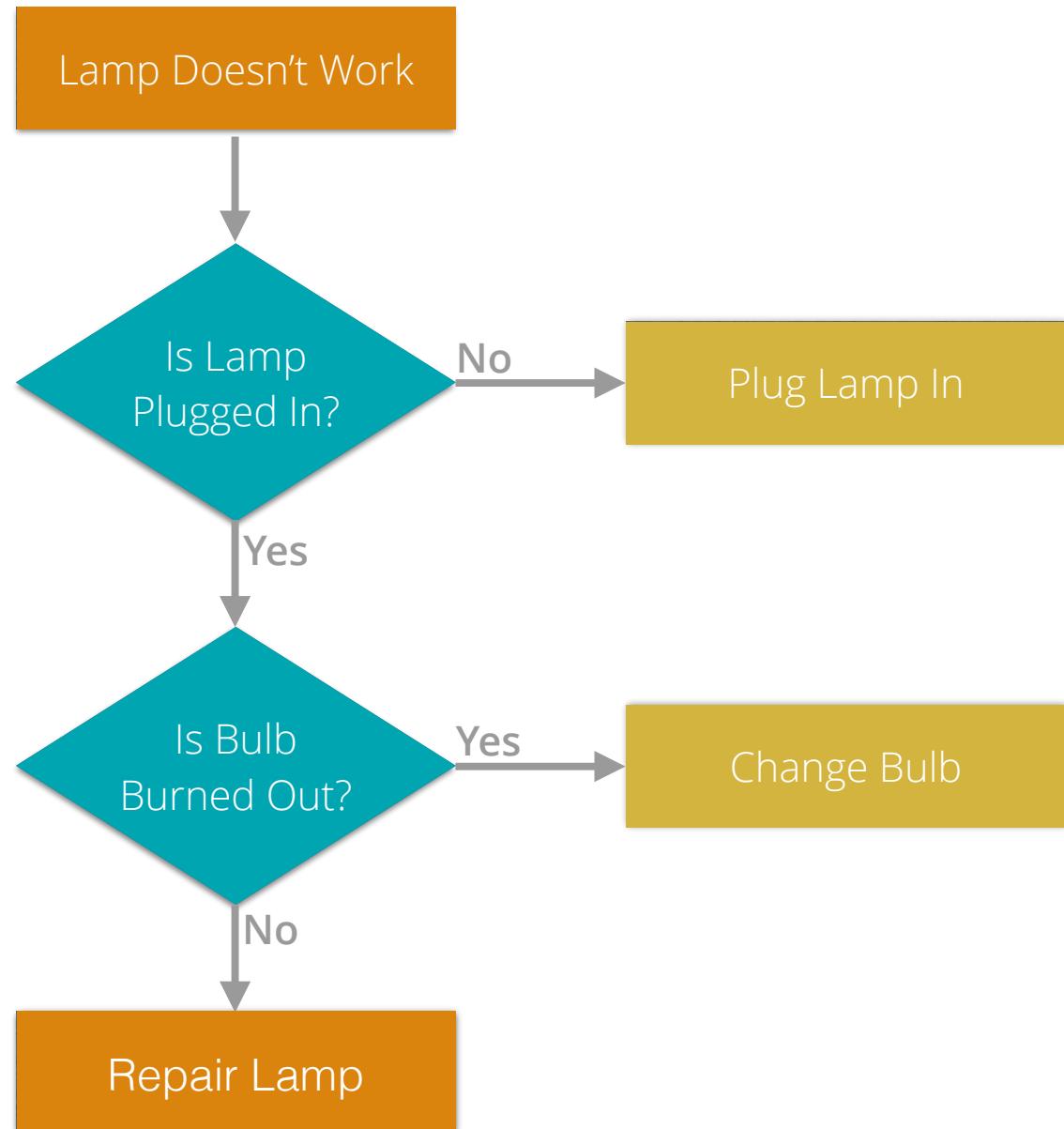
`@“text” == @“text”` will not evaluate correctly

use `isEqualToString` instead

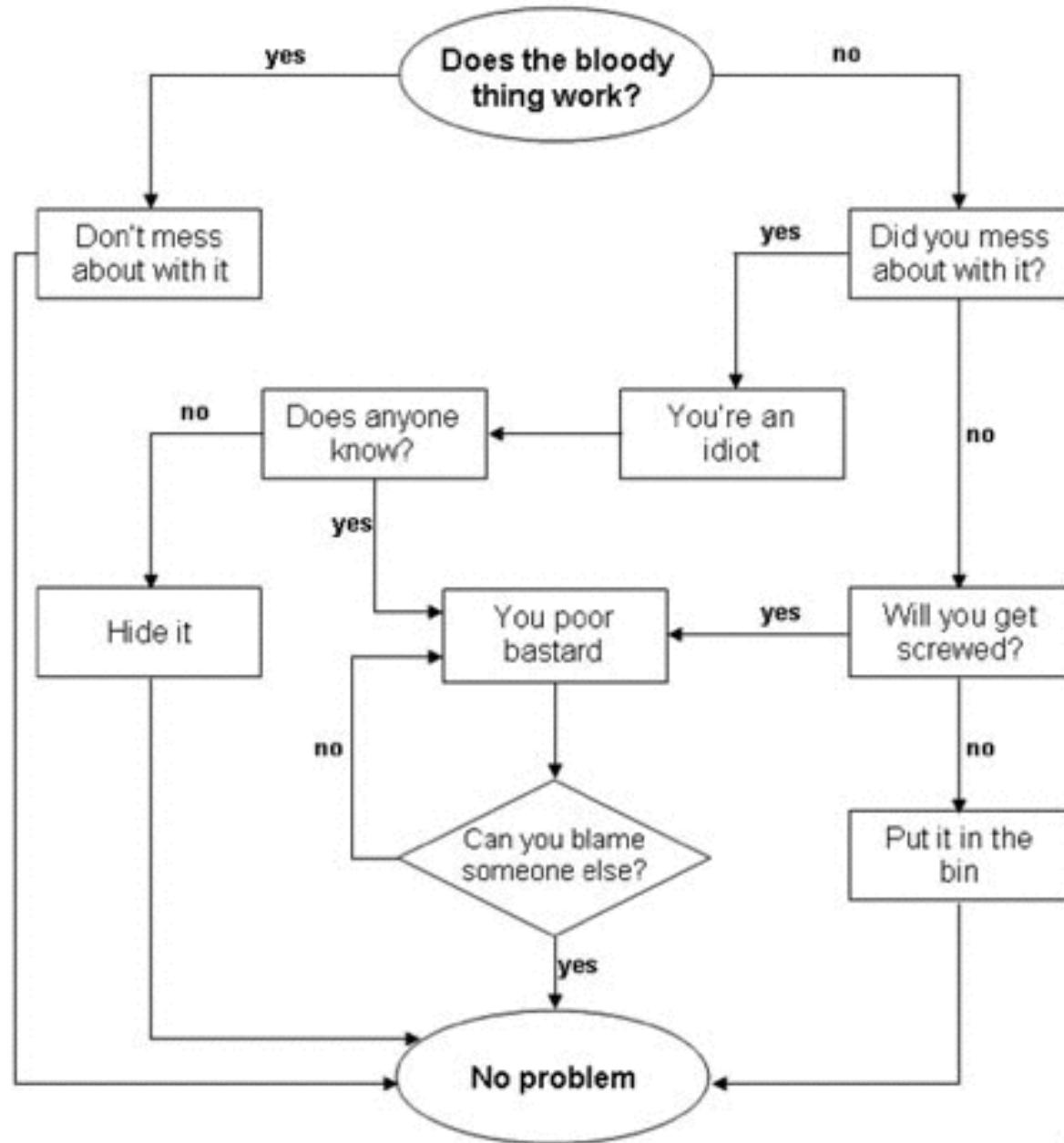
`xDate == yDate` will not evaluate correctly

use `isEqualToDate` instead

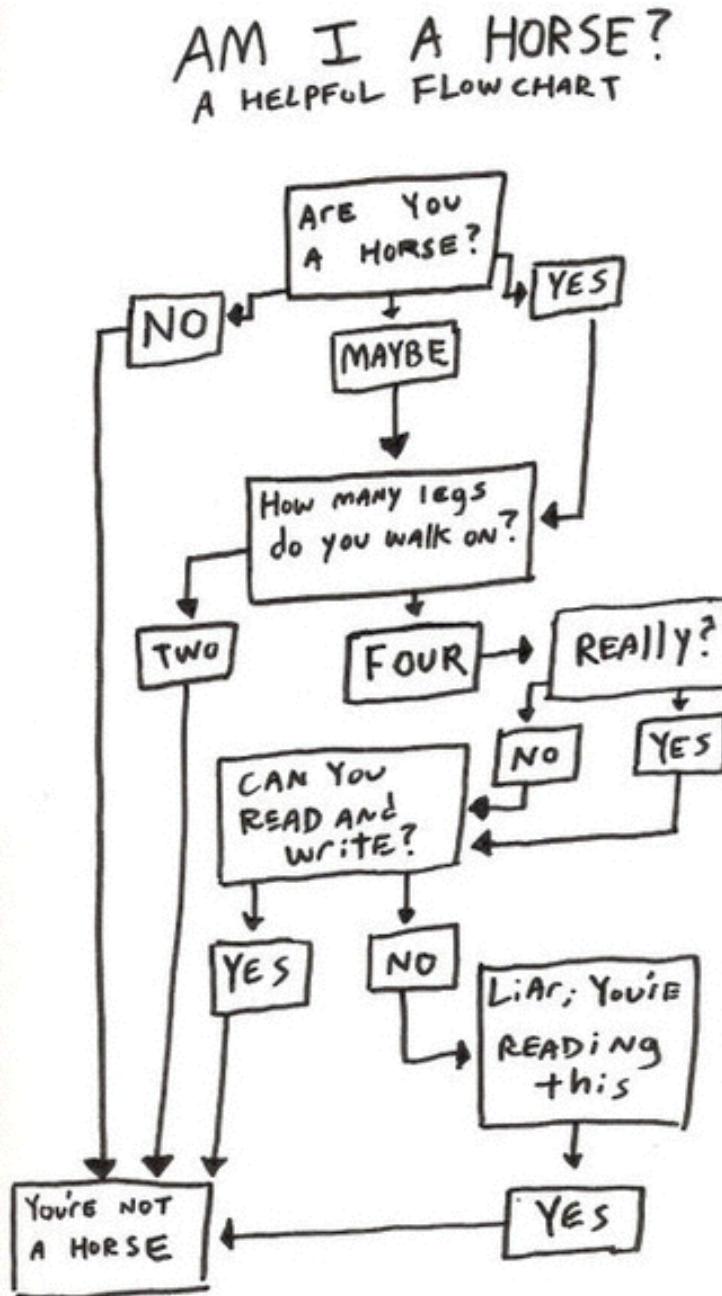
# Conditional Statements



# Conditional Statements



# Conditional Statements



Source: Mental Floss

# Conditional Statements

## If-Then-Else

```
if (myInt == 1) {  
    NSLog(@"Got 1");  
} else {  
    NSLog(@"Didn't Get 1");  
}
```

## Short If-Then-Else

```
if (myInt == 1) NSLog(@"Got 1"); else NSLog(@"Didn't get 1");
```

## Shorter If-Then-Else

```
myInt == 1 ? NSLog(@"Got 1") : NSLog(@"Didn't get 1");
```

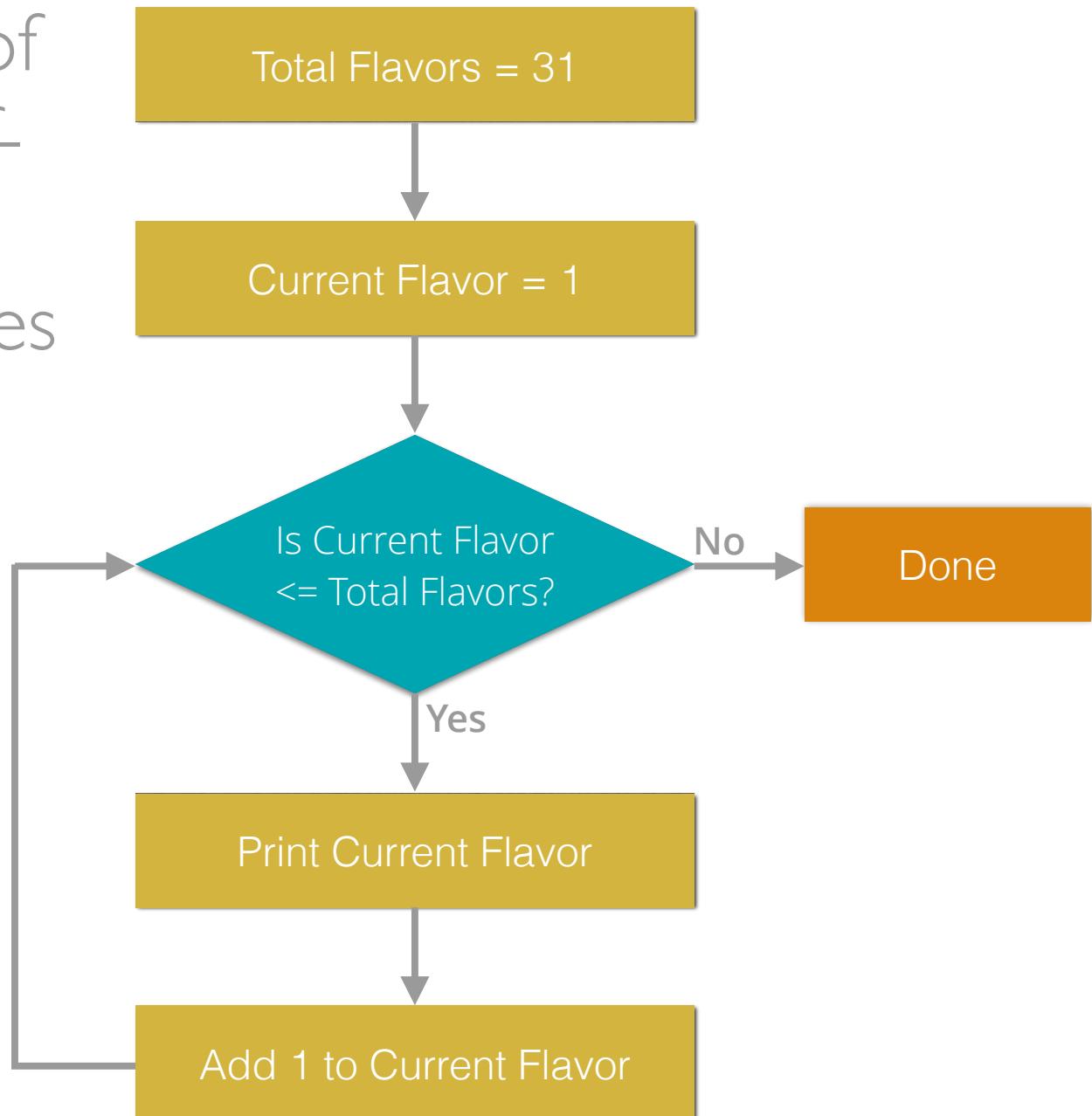
# Conditional Statements

## Case

```
switch (myInt) {  
    case 1: {  
        NSLog(@"Got 1");  
        break;  
    }  
    default: {  
        NSLog(@"Didn't Get 1");  
        break;  
    }  
}
```

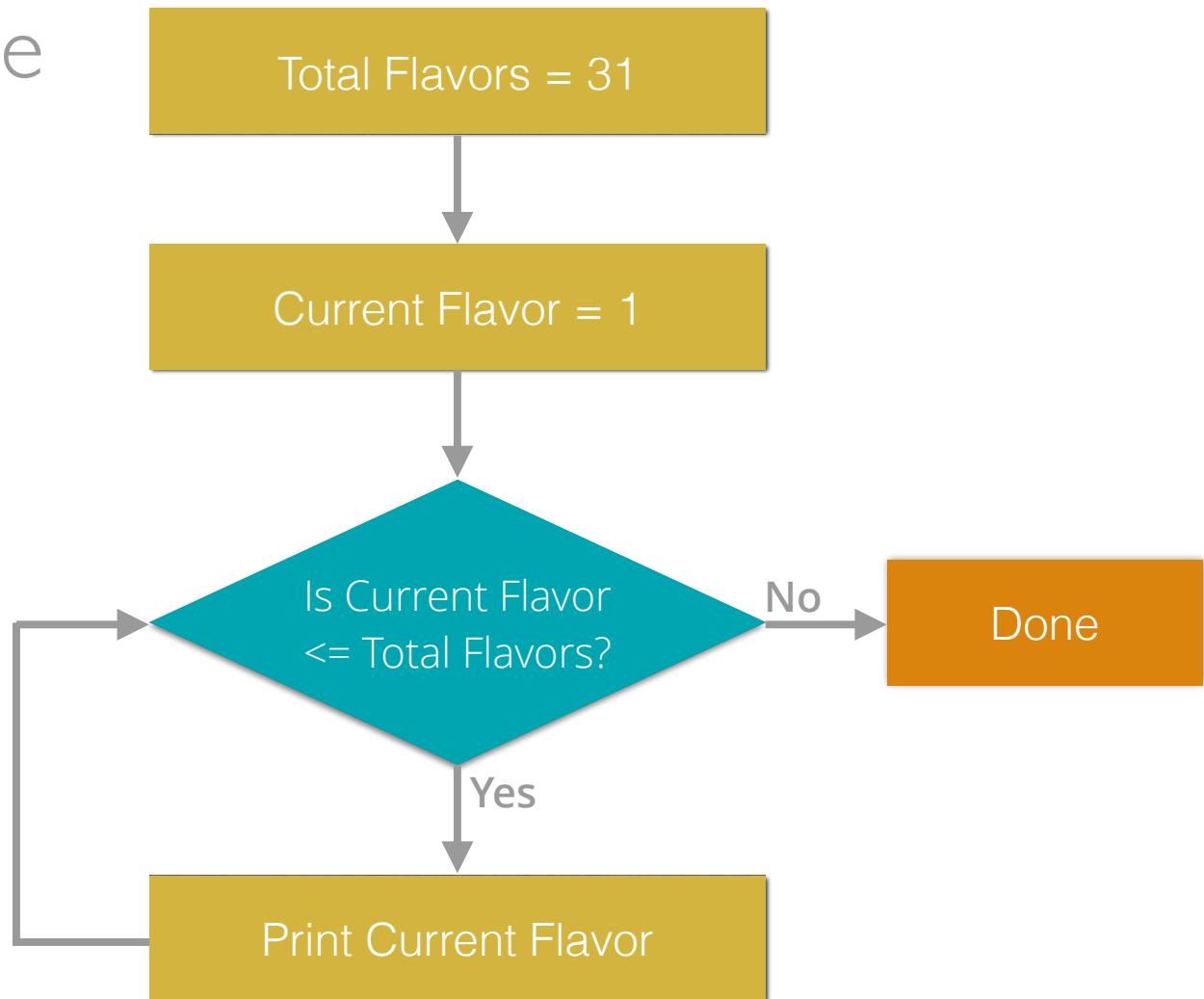
# Loops

Loops are sort of like specialized If-Statements that repeat themselves

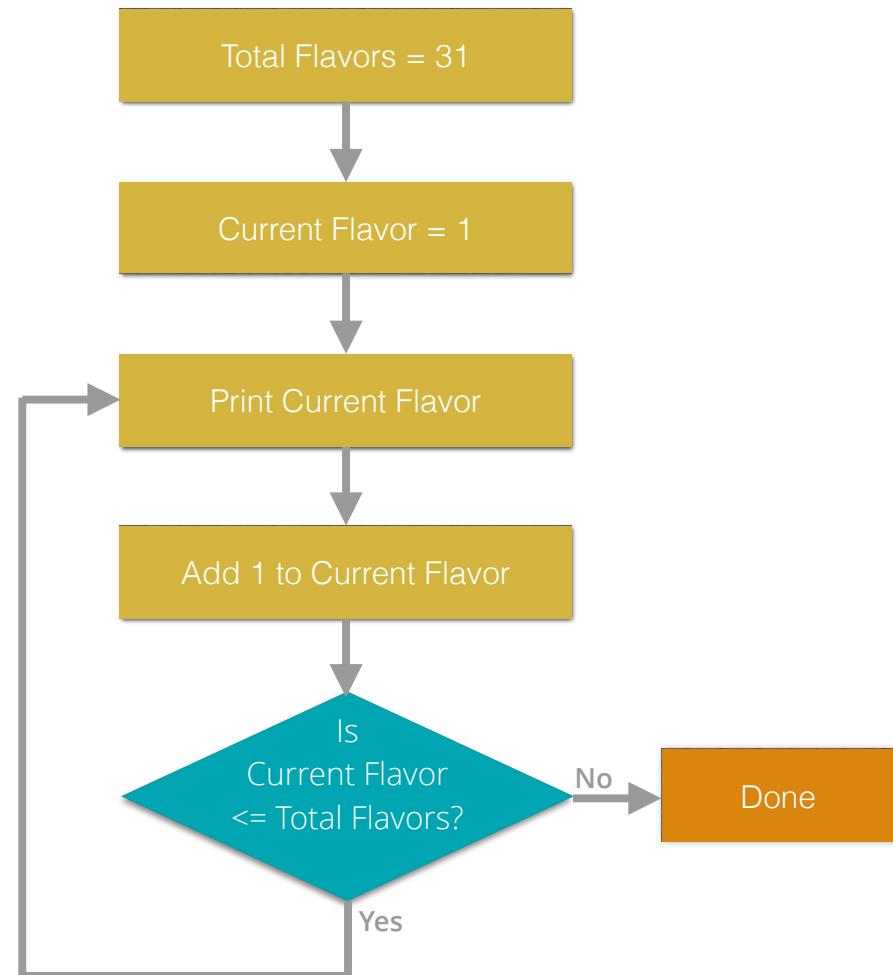
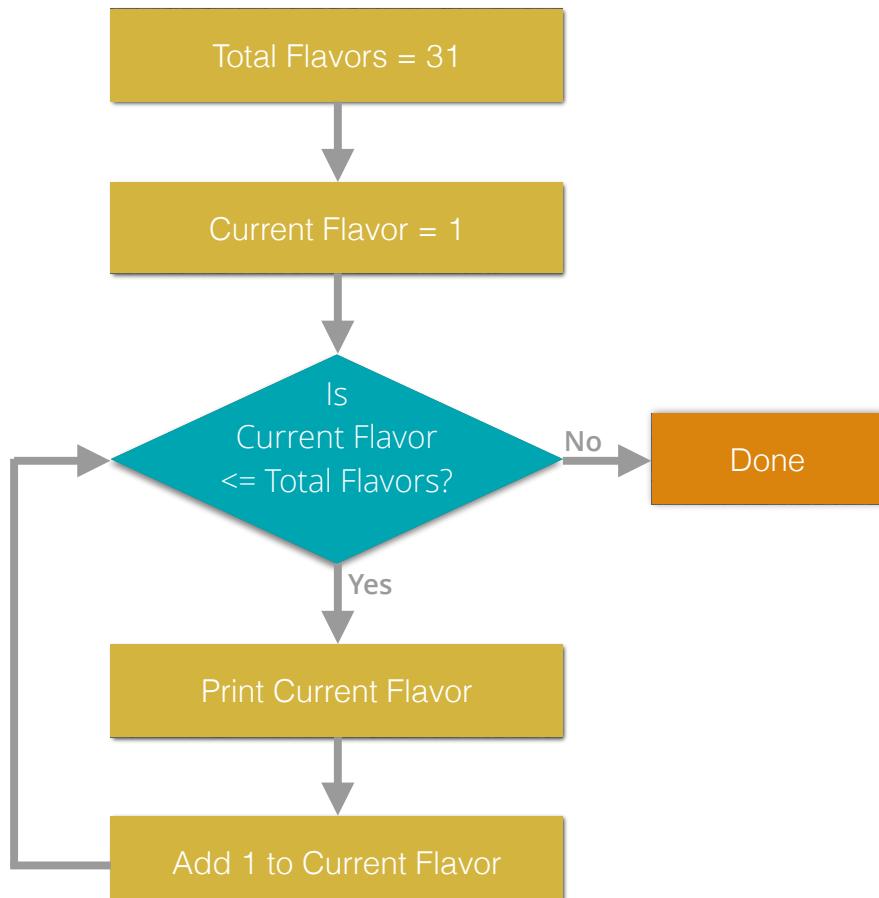


# Loops

Beware of infinite loops



# Loop Testing Location



Loops can have different locations for testing

# Loops

## While

```
int i = 0;
while (i < [myArray count]) {
    NSLog(@"Flavor: %@", myArray[i]);
    i++;
}
```

## Do...While

```
int i = 0;
do {
    NSLog(@"Flavor: %@", myArray[i]);
    i++;
} while (i < [myArray count]);
```

# Loops

For

```
for (int i = 0; i < [myArray count]; i++) {  
    NSLog(@"Flavor: %@", myArray[i]);  
}
```

Fast Enumeration For

```
for (NSString *flavorString in myArray) {  
    NSLog(@"Flavor: %@", flavorString);  
}
```

# Images

# UIImage vs. UIImageView

UIImage

An object that holds image data

UIImageView can

Display one UIImage at a time

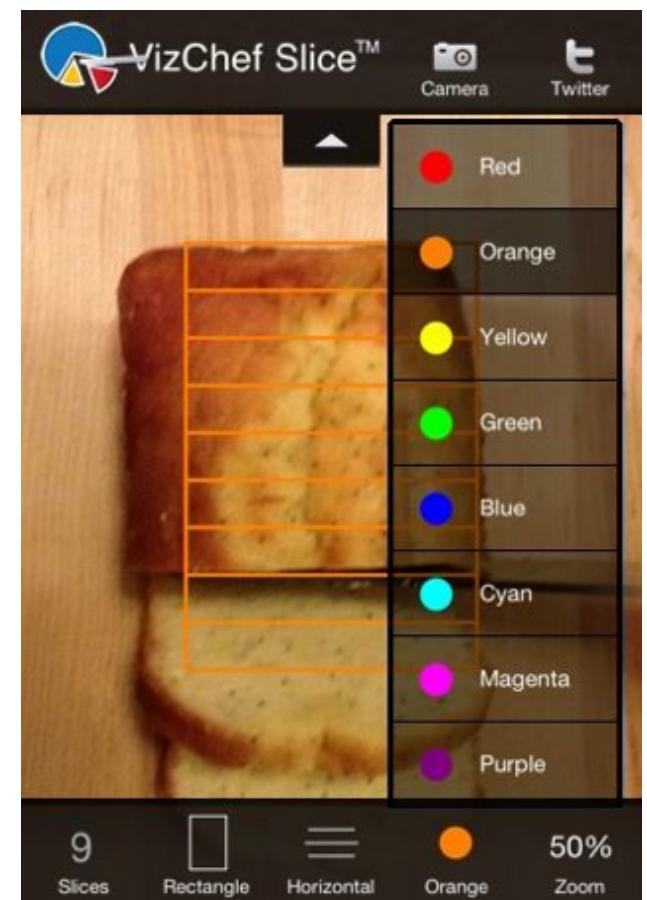
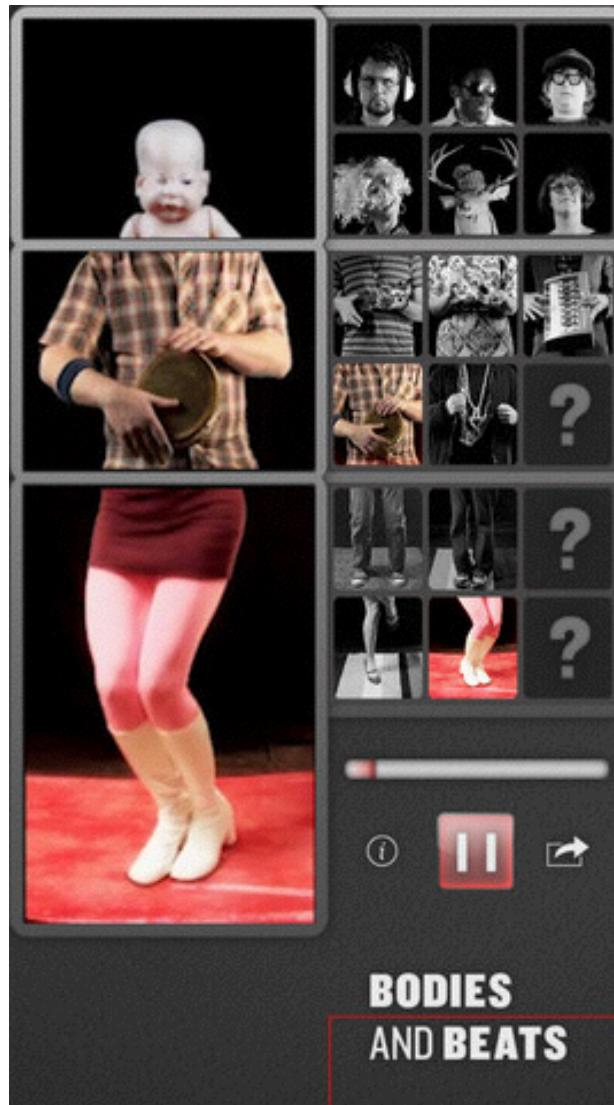
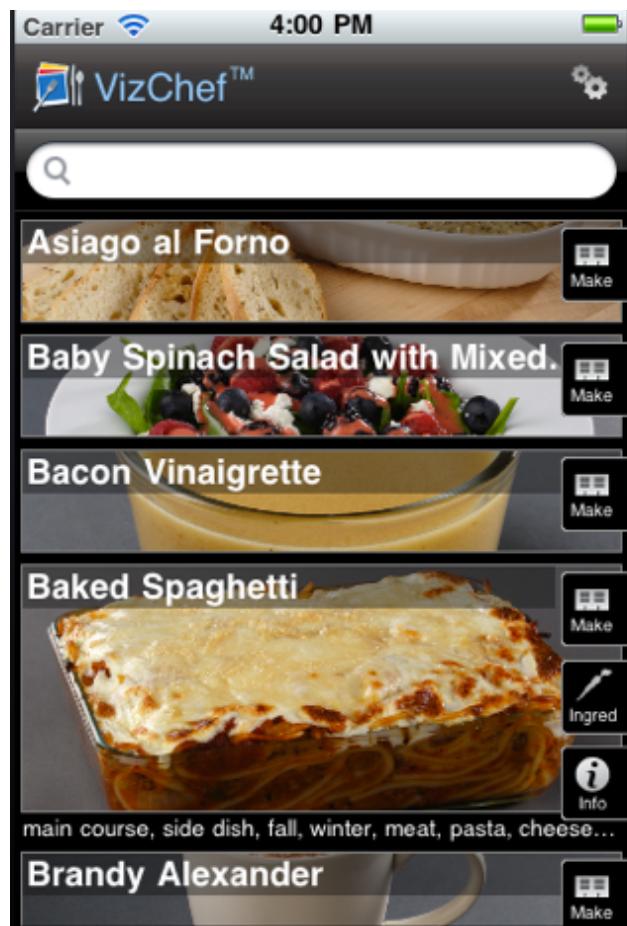
Can animate a series of images like a flip book

Can stretch images (with or without insets)

Can scale (zoom), pan, or rotate a UIImage

(in code or with user interaction)

# Pick the Images



# UIImageViews

Used to display images such as photos, duh

Also used to display interface elements  
(i.e. borders, backgrounds, etc)

Typically use .PNG (preferred) or .JPG

.PNGs can have transparency (use sparingly)

Can resize, but design for the size you need

# Retina Display

Old iPhones are 320x480 pixels

Retina has double! (640x960)

We still lay controls out on 320x480 (or 568)

Apple automatically doubles the numbers

For best results, we provide retina & non-retina images with different names

myimage.png

myimage@2x.png

# Naming Conventions

Type	Example
Non-retina	myimage.png
Retina	myimage@2x.png
Retina HD	myimage@3x.png
Tall Launch Image	myimage-568h@2x.png
iPhone	myimage~iphone.png
iPhone Retina	myimage@2x~iphone.png
iPad	myimage~ipad.png
iPad Retina	myimage@2x~ipad.png

Still only one line of code!

```
[_myImageView setImage:[UIImage imageNamed:@"myimage.png"]];
```

*Tip: use all lower case if there's a chance of going cross platform*

# Retina

Non-Retina



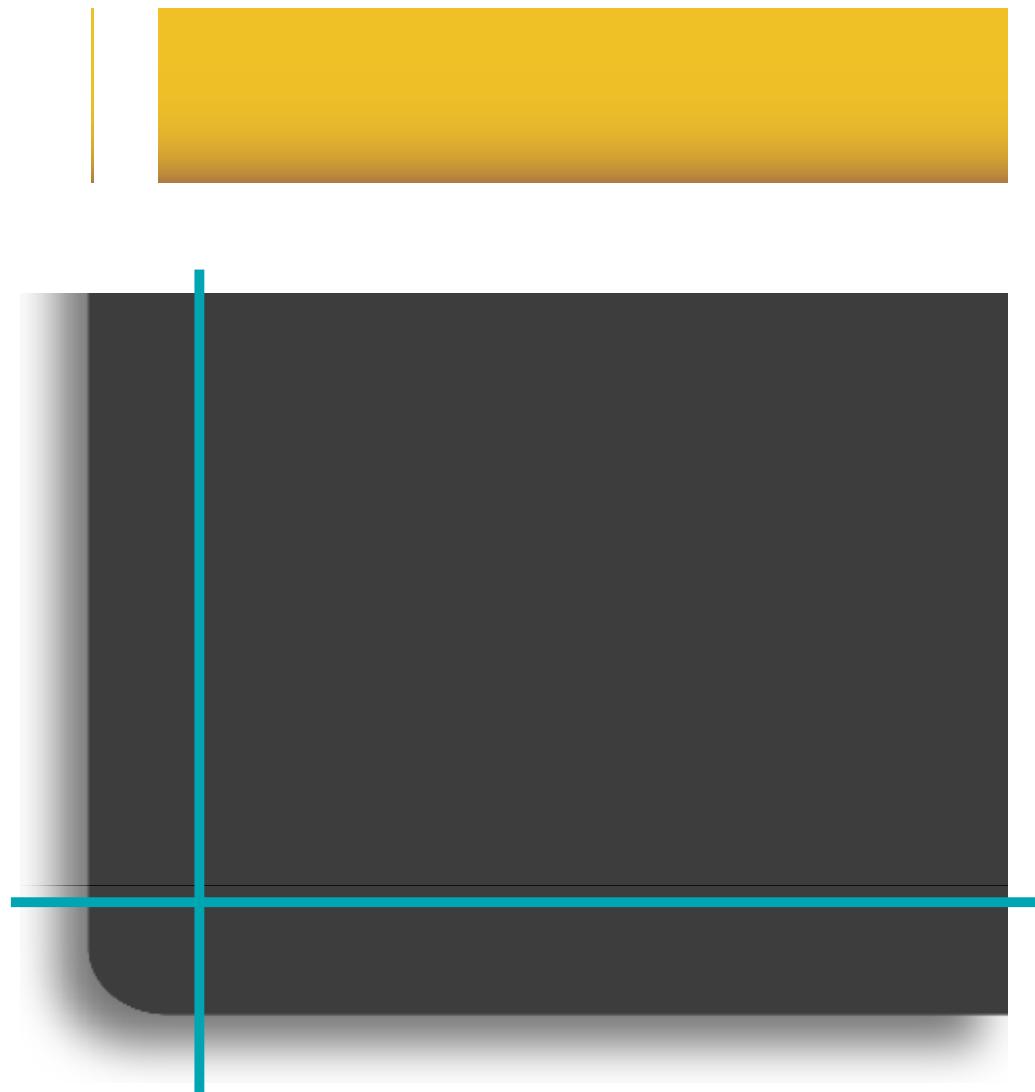
Retina



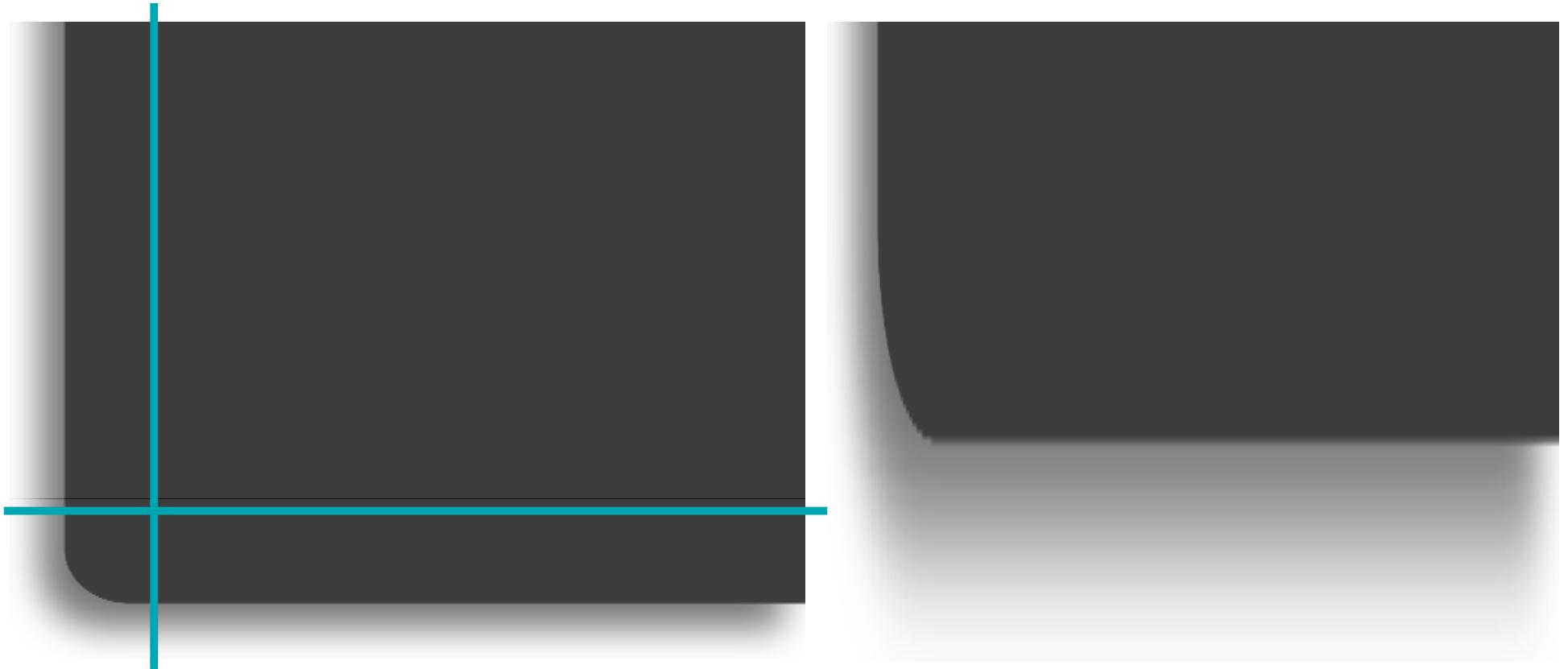
Retina HD



# Stretchable Images



# Image Insets



```
[_myImageView setImage:  
    [[UIImage imageNamed:@"myimage.png"]  
     resizableImageWithCapInsets:  
     UIEdgeInsetsMake(0.0, 10.0, 10.0, 0.0)]];
```

# Adding Images

# Adding Images

Screenshot of Xcode showing the project structure and the Images.xcassets editor.

The project navigation bar shows "HappyHour" selected for the iPhone 6 target.

The left sidebar lists the project structure:

- HappyHour (2 targets, iOS SDK 8.3)
  - HappyHour
    - happyhour.entitlements
    - Storyboard.storyboard
    - VNAppDelegate.h
    - VNAppDelegate.m
    - VNNavigationController.h
    - VNNavigationController.m
    - VNMainViewController.h
    - VNMainViewController.m
    - VNBusinessLoc...ViewController.h
    - VNBusinessLoc...ViewController.m
    - VNLocationDealsViewController.h
    - VNLocationDealsViewController.m
    - VNDealViewController.h
    - VNDealViewController.m
    - AboutViewController.h
    - AboutViewController.m
  - Images.xcassets
  - Categories
  - Annotations
  - Cells
  - Data
  - Supporting Files
  - happyhournow
  - Frameworks
  - Products
  - Pods

The "Images.xcassets" item is currently selected, highlighted with a blue background.

The main area displays the contents of the "Images.xcassets" folder:

  - AppIcon
  - AppIconShaded
  - BackgroundCriteriaBlue
  - BackgroundCriteriaGreen
  - BackgroundCriteriaOrange
  - BackgroundCriteriaPurple
  - BackgroundCriteriaPurpleDark
  - BackgroundCriteriaPurpleLight
  - BackgroundCriteriaRed
  - BackgroundDetailTop
  - BackgroundDistanceSelected
  - BackgroundDistanceUnselected
  - BackgroundFeaturedCell
  - BackgroundTopBlue
  - BackgroundTopDarkPurple
  - BackgroundTopGreen
  - BackgroundTopOrange
  - BackgroundTopPurple
  - BackgroundTopRed
  - ButtonGo
  - FeaturedFooter
  - FeaturedHeader
  - IconBackground
  - IconFixIt
  - IconHappyHour
  - IconMail
  - IconMailSmall
  - IconShare
  - IconTabHappyHour
  - IconTabMap
  - IconTabName
  - IconTypeBeer

The right side of the interface shows three cards with icons and descriptions:

  - View Controller** - A controller that supports the fundamental view-management model in iOS.
  - Navigation Controller** - A controller that manages navigation through a hierarchy of views.
  - Table View Controller** - A controller that manages a table view.

# Adding Images

The screenshot shows the Xcode interface with a project named "HappyHour". The left sidebar displays the project structure, including files like "happyhour.entitlements", "Storyboard.storyboard", and "Images.xcassets". A blue arrow points from the bottom-left towards the "New Image Set" option in a context menu.

The main area shows the "Images.xcassets" folder contents, which include various asset names such as AppIcon, BackgroundCriteriaBlue, and IconTypeBeer. To the right, there are three cards describing View Controller, Navigation Controller, and Table View Controller.

**Context Menu Options:**

- New Image Set
- New App Icon
- New Launch Image
- New OS X Icon
- New Folder
- New Folder From Selection
- Import...
- Import From Project...

**Right-hand Side Cards:**

- View Controller** - A controller that supports the fundamental view-management model in iOS.
- Navigation Controller** - A controller that manages navigation through a hierarchy of views.
- Table View Controller** - A controller that manages a table view.

# Adding Images

The screenshot shows the Xcode Assets Catalog interface. The left sidebar lists various asset names, and the main area displays the 'IconMail' asset set. Inside 'IconMail', there are three image slots labeled '1x', '2x', and '3x'. A large blue arrow points from the text 'Select image' at the bottom of the slide towards the '2x' slot. The right side of the interface shows the 'Image Set' configuration panel with fields for Name (IconMail), Devices (Universal), Width (Any), Height (Any), Scale Factors (Multiple), and Render As (Default). Below this panel, there are three cards: 'View Controller' (A controller that supports the fundamental view-management model in iOS.), 'Navigation Controller' (A controller that manages navigation through a hierarchy of views.), and 'Table View Controller' (A controller that manages a table view.). At the bottom right, there are buttons for 'Show Slicing' and a trash icon.

HappyHour > iPhone 6 HappyHour: Ready | Today at 4:03 PM 1

IconMail

IconMail

1x 2x 3x

Universal

Name IconMail

Devices Universal

Width Any

Height Any

Scale Factors Multiple

Render As Default

View Controller - A controller that supports the fundamental view-management model in iOS.

Navigation Controller - A controller that manages navigation through a hierarchy of views.

Table View Controller - A controller that manages a table view.

Show Slicing

# Adding Images

The screenshot shows the Xcode Assets Catalog interface. The left sidebar lists various asset names with checkboxes, and the main area displays the 'IconMail' image set. The 'IconMail' set contains three 1x icons of an envelope, one 1x iPad icon, and four Apple Watch icons (2x, 38 mm 2x, 42 mm 2x). The right panel shows the 'Image Set' configuration for 'IconMail', which is set to 'Device Specific' and includes checkboxes for iPhone, iPad, Apple Watch, and Mac. Below this are settings for Width, Height, Scale Factors, and Render As. A sidebar on the right lists View Controller, Navigation Controller, and Table View Controller with their descriptions.

Assets:

- AppIcon
- AppIconShaded
- BackgroundCriteriaBlue
- BackgroundCriteriaGreen
- BackgroundCriteriaOrange
- BackgroundCriteriaPurple
- BackgroundCriteriaPurpleDark
- BackgroundCriteriaPurpleLight
- BackgroundCriteriaRed
- BackgroundDetailTop
- BackgroundDistanceSelected
- BackgroundDistanceUnselected
- BackgroundFeaturedCell
- BackgroundTopBlue
- BackgroundTopDarkPurple
- BackgroundTopGreen
- BackgroundTopOrange
- BackgroundTopPurple
- BackgroundTopRed
- ButtonGo
- FeaturedFooter
- FeaturedHeader
- IconBackground
- IconFixit
- IconHappyHour
- IconMail** (selected)
- IconMailSmall
- IconShare
- IconTabHappyHour
- IconTabMap
- IconTabName
- IconTypeBeer

IconMail

1x 2x 3x 1x 2x 2x 38 mm 2x 42 mm 2x

iPhone iPad Apple Watch

**Image Set**

Name: IconMail

Devices: Device Specific

iPhone

Retina 4-inch

iPad

Apple Watch

Mac

Width: Any

Height: Any

Scale Factors: Multiple

Render As: Default

**View Controller** - A controller that supports the fundamental view-management model in iOS.

**Navigation Controller** - A controller that manages navigation through a hierarchy of views.

**Table View Controller** - A controller that manages a table view.

# Adding Files

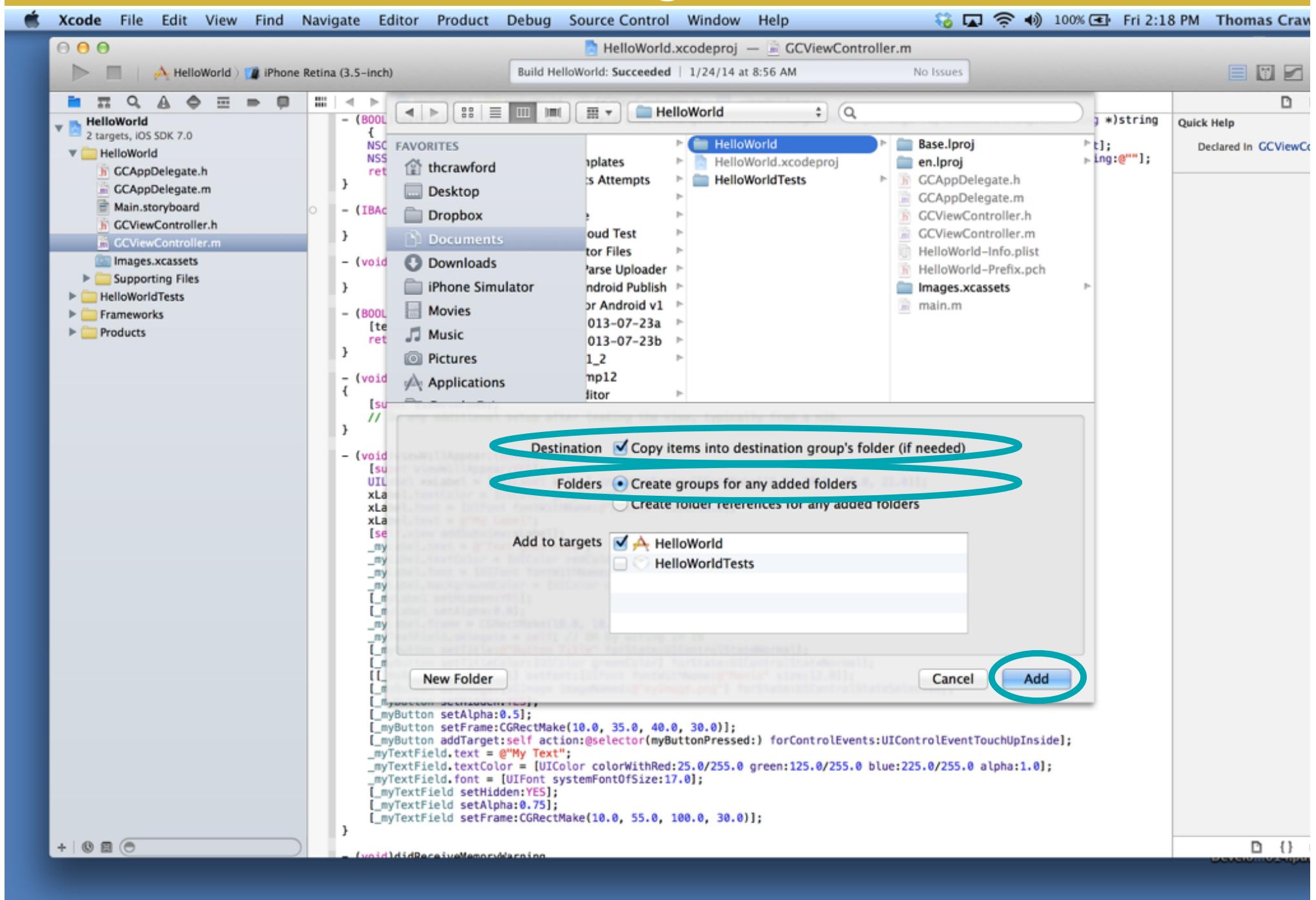
# Adding Files

There are **multiple** ways to add files  
There's only **one** that works consistently:

1. Place the files to be added into the project directory on your hard drive
2. File > Add Files to “yourproject”...
3. Select the files or folders to add
4. **Always, always, always** select “Copy items into destination’s group folder (if needed)”
5. Select “Create Groups for any added folders”
6. Click “Add”

Other ways seem easier, but don’t do it!

# Adding Files



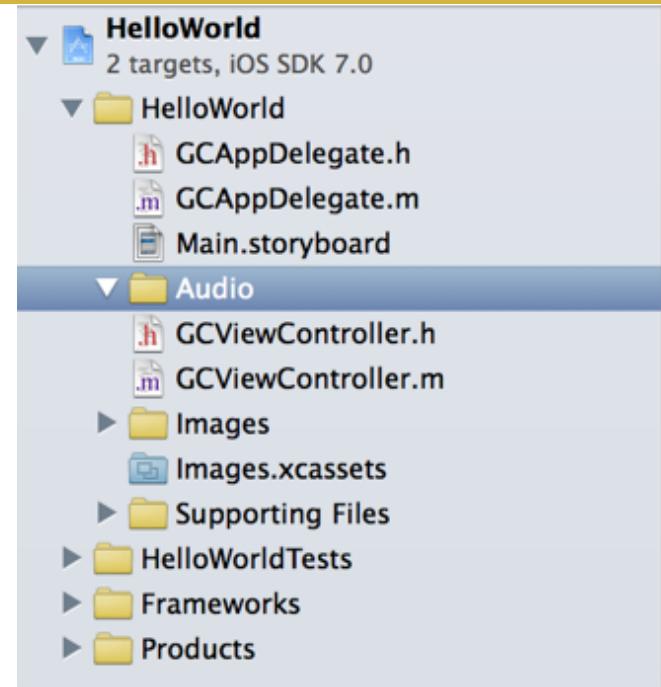
# File Groups in Xcode

Groups look like folders

They are **not**

They are just logical groupings of files and have nothing to do with the file structure on your hard drive

Organize them any way you want



# Random Numbers

# Random Numbers

```
int totalOptions = 451;  
int randomIndex = arc4random_uniform((uint32_t)totalOptions);
```

# Parsing CSV

# Parsing CSV

```
- (NSString *)readBundleFileToString:(NSString *)filename ofType:(NSString *)type {
    NSString *path = [[NSBundle mainBundle]
                      pathForResource:filename ofType:type];
    return [NSString stringWithContentsOfFile:path
                                         encoding:NSUTF8StringEncoding error:NULL];
}

- (NSArray *)convertCSVStringToArray:(NSString *)csvString {
    NSString *cleanString = [[csvString
                               componentsSeparatedByCharactersInSet:[NSCharacterSet
                                                               newlineCharacterSet]] componentsJoinedByString:@""];
    NSCharacterSet *set = [NSCharacterSet
                           characterSetWithCharactersInString:@",,"];
    return [cleanString
            componentsSeparatedByCharactersInSet:set];
}
```