

iOS DeCal : Lecture 1

Introduction, Xcode, and Swift

Overview : Today's Lecture

1. What is this course about?

2. Course Logistics

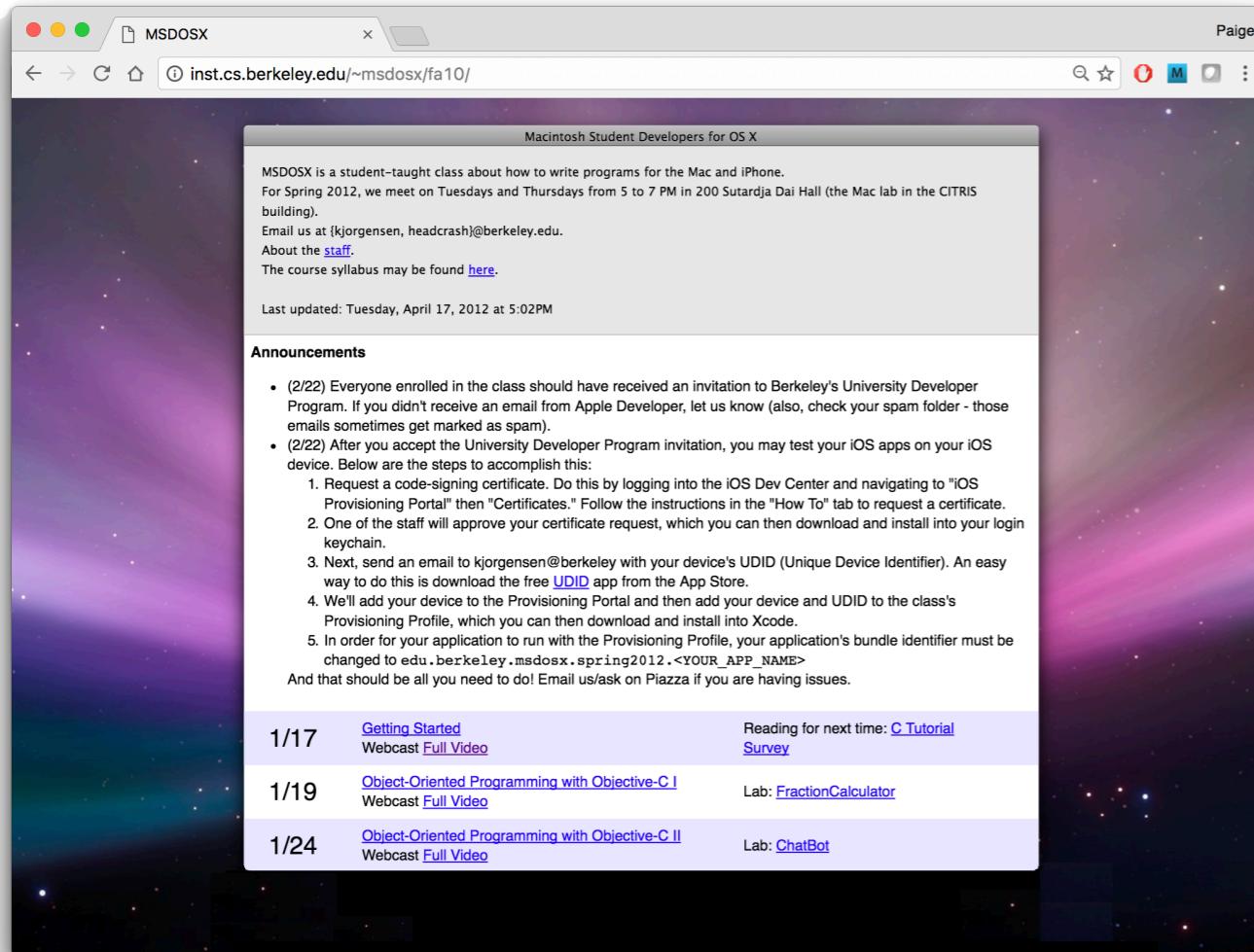
- Staff Introduction
- Lecture / lab format
- Lab Room Assignments

3. Introduction to Xcode (+ demo)

4. Swift 3 Overview

Course Background

Course History MSDOSX



2010 Macintosh
Student Developers

Macintosh Student Developers for OS X

Started by Dan Garcia in early 2000s

Originated from desire for **(the few)** passionate Mac users to program for their own devices

Course History MSDOSX



2010 Macintosh
Student Developers

Macintosh Student
Developers for OS X

Started by Dan Garcia
in early 2000s

Originated from desire
for **(the few)** passionate
Mac users to program
for their own devices

What will you learn in this course?



Swift 3
**(Programming
Language)**

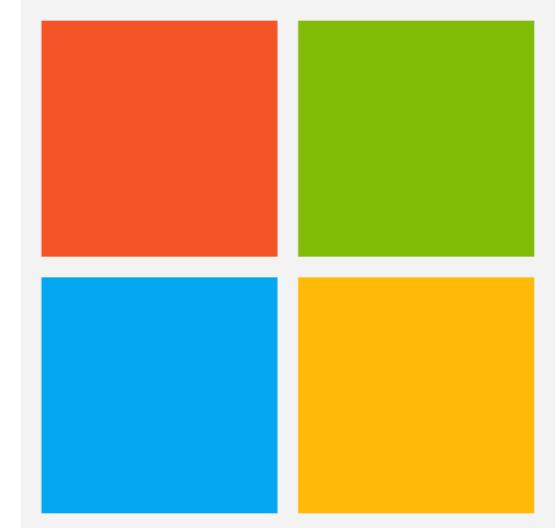


Xcode 8
(IDE)



**iOS Application
Development**

iOS Experience : Past Students



Course Logistics

Course Staff : Instructors



Akilesh Bapu



Paige Plander

Course Staff : Teaching Assistants



Anwar Baroudi
Head TA



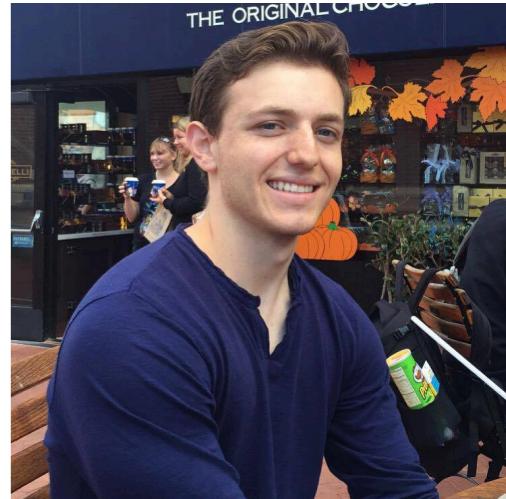
Sameer Suresh
Head TA



Chan Hee Park



Maya Reddy



Matt Turk



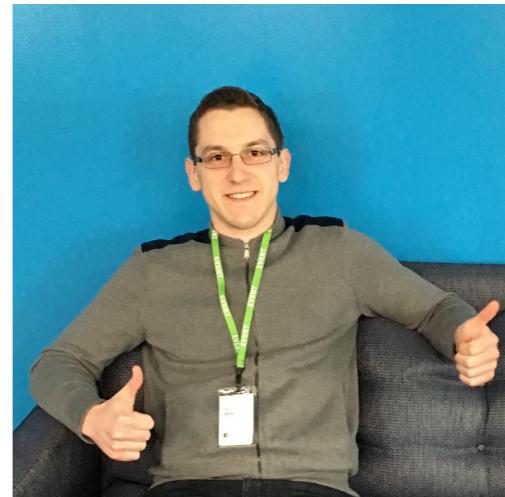
Chris Zielinski



Maaz Uddin



Nithi Narayanan



Gera Groshev



William Smith

Class Format : Lecture / Labs

Tuesdays : 6:30 - 8:00 PM (HP Auditorium)

Approximately 1 - 1.5 hour of lecture

Attendance check-in during lecture

Optional Office Hour immediately following lecture

Thursdays : 6:30 - 8:00 PM (Soda 310 **OR** Jacobs 220)

Approximately 1 - 1.5 hour lab

- Thursday labs due the following Tuesday, in case you don't finish during the lab period
- May work with a partner
- Submit lab via TA check-off (**recommended**) or Gradescope
- Labs will typically cover the last topic of lecture

Attendance check-in during lab

Work on iOS projects with your peers and instructor guidance!

Enrollment

**Rather than enrolling through CalCentral,
students will be automatically enrolled (either
into the class or onto waitlist)**

Since we are dropping students who didn't attend this first lecture, we are waiting to enroll students until after lecture.

E-mail us ASAP or talk to us after lecture if you have a conflict or want to drop.

Workload: Labs / Projects

Labs (11)

Approximately 1.5 hours

One for each week's topic (typically last topic of the day)

Projects (2)

Hangman

Snapchat Clone

Final Project

Come up with your own idea!

Work with up to 4 other students.

Lab Room Assignments : (Thursdays)

310 Soda - (Instructor : Paige)

Lastnames beginning with A - L

220 Jacobs - (Instructor : Akilesh)

Lastnames beginning with L - Z

**** Lab sections will be quite full - so we may need to move around some students**

Grading : Rough Breakdown

30% Projects

Graded on a scale from 1-10 (Rubrics will be available closer to the project due date)

35% Labs

Graded on a Pass/Fail Basis

35% Final Project

Graded out of 35 based on Rubric

Cheating

This is a DeCal.

We don't have any fancy algorithms in place like other courses.

We will know though.

iOS auto-graders are difficult to make so we'll be going through projects manually during grading. There's a pretty good chance we'll be able to find out if you did copy code.

Will be dealt with on a case-by-case basis.

You get out of it what you put in.

Leaving Lecture Early

We will be alternating when check-in forms are uploaded

Labs will primarily cover the LAST topic of lecture

Leaving early will make labs take more time

If you need to leave early, make a private post on Piazza and sit nearby the doors

Links / Contact Info

Course Website - updated regularly

iosdecal.com

Piazza - ask us questions here!

piazza.com/berkeley/spring2017/cs198s17

E-mail

iosdecalstaff@gmail.com

Attendance Check-in : Lecture

There will be a check-in every lecture via Google Form
You must check in with a different person in the class
each week (one form per pair).

Excused Absences: **Private Post Us on Piazza**
Conflicting exams only

Unexcused Absences
Students with 4+ Unexcused Absences will receive an
NP for the course

**** More info on Attendance Policy found on Piazza ****

Let's try it now!

Introduce yourself to another student or TA, and fill out the Google Form found on our course website (iosdecal.com).



Xcode!!

The IDE to Rule Them All

A screenshot of the Xcode interface showing the 'Choose a template for your new project:' screen. A green box highlights the top navigation bar, which includes tabs for iOS (selected), watchOS, tvOS, macOS, and Cross-platform, and a 'Filter' button. Below this, a section titled 'Application' shows five template options: Single View Application (selected), Game, Master-Detail Application, Page-Based Application, and Tabbed Application. Each template has a preview icon and a name. Below this section is another titled 'Framework & Library' with three items: Cocoa Touch, Cocoa Touch (Touch), and Metal Library.

Choose a template for your new project:

iOS watchOS tvOS macOS Cross-platform Filter

Application

Single View Application

Game

Master-Detail Application

Page-Based Application

Tabbed Application

Framework & Library

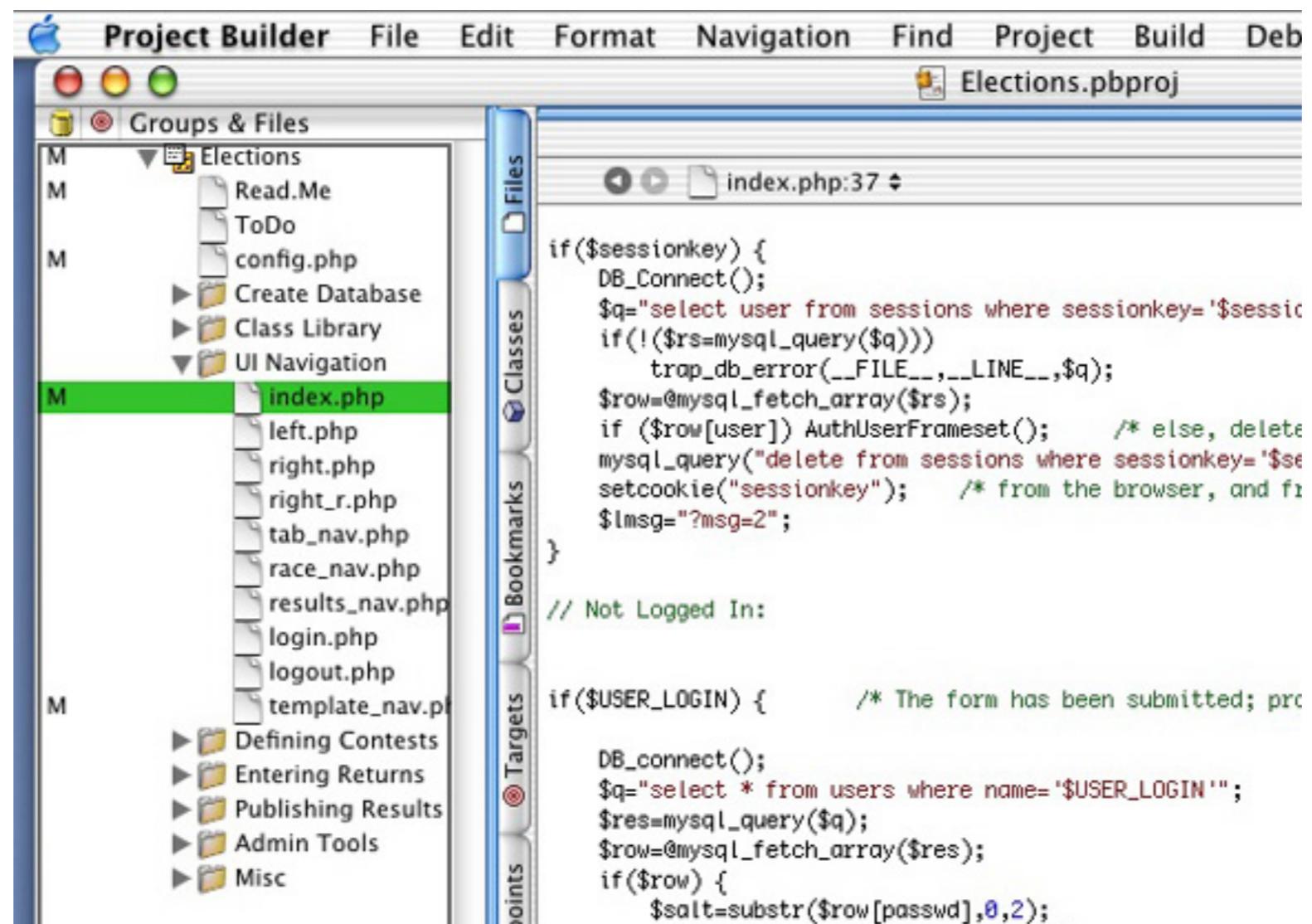
Cocoa Touch

Cocoa Touch Touch

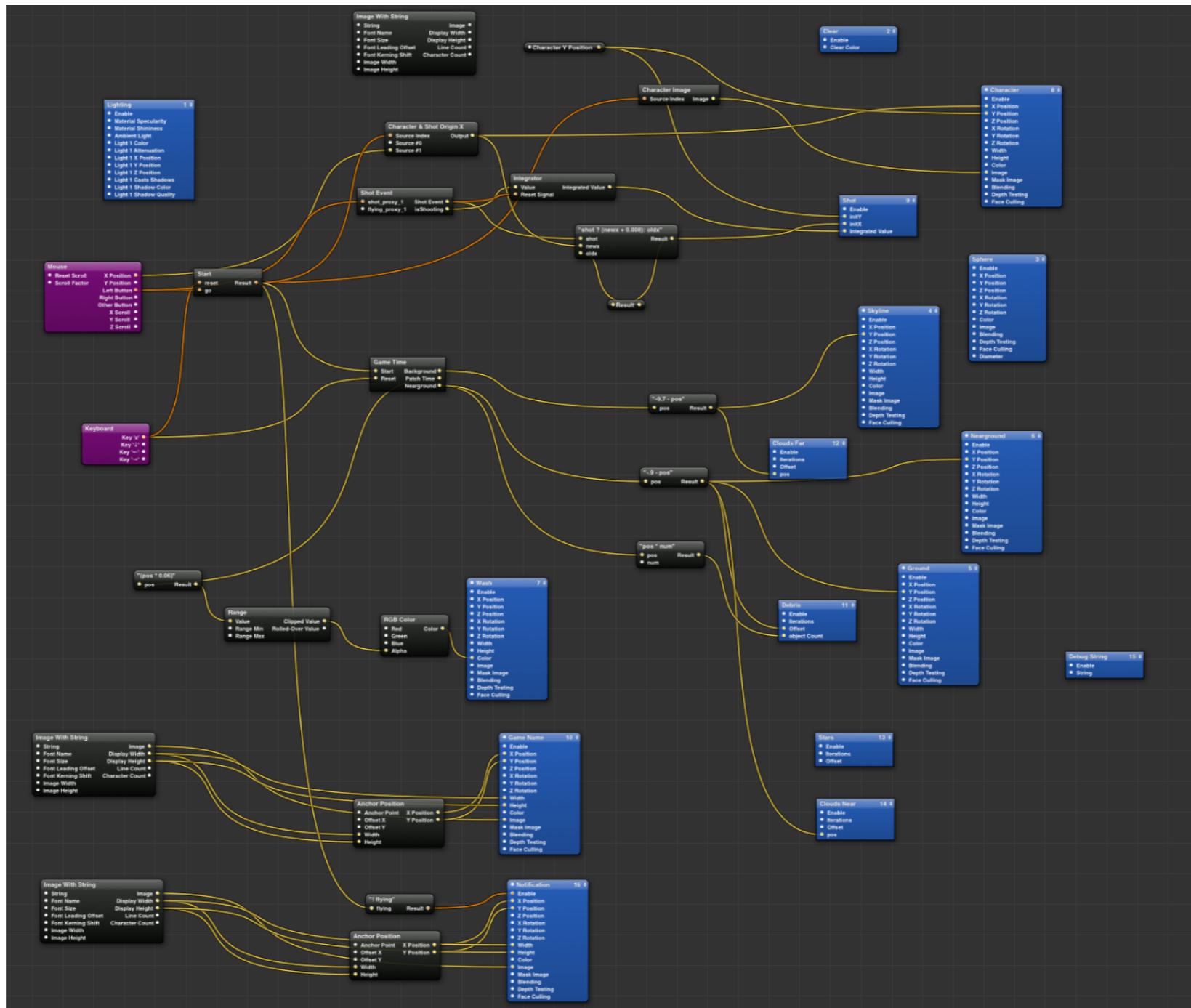
Metal Library

Xcode History

Project Builder



Xcode 2 - Visual Programming Language

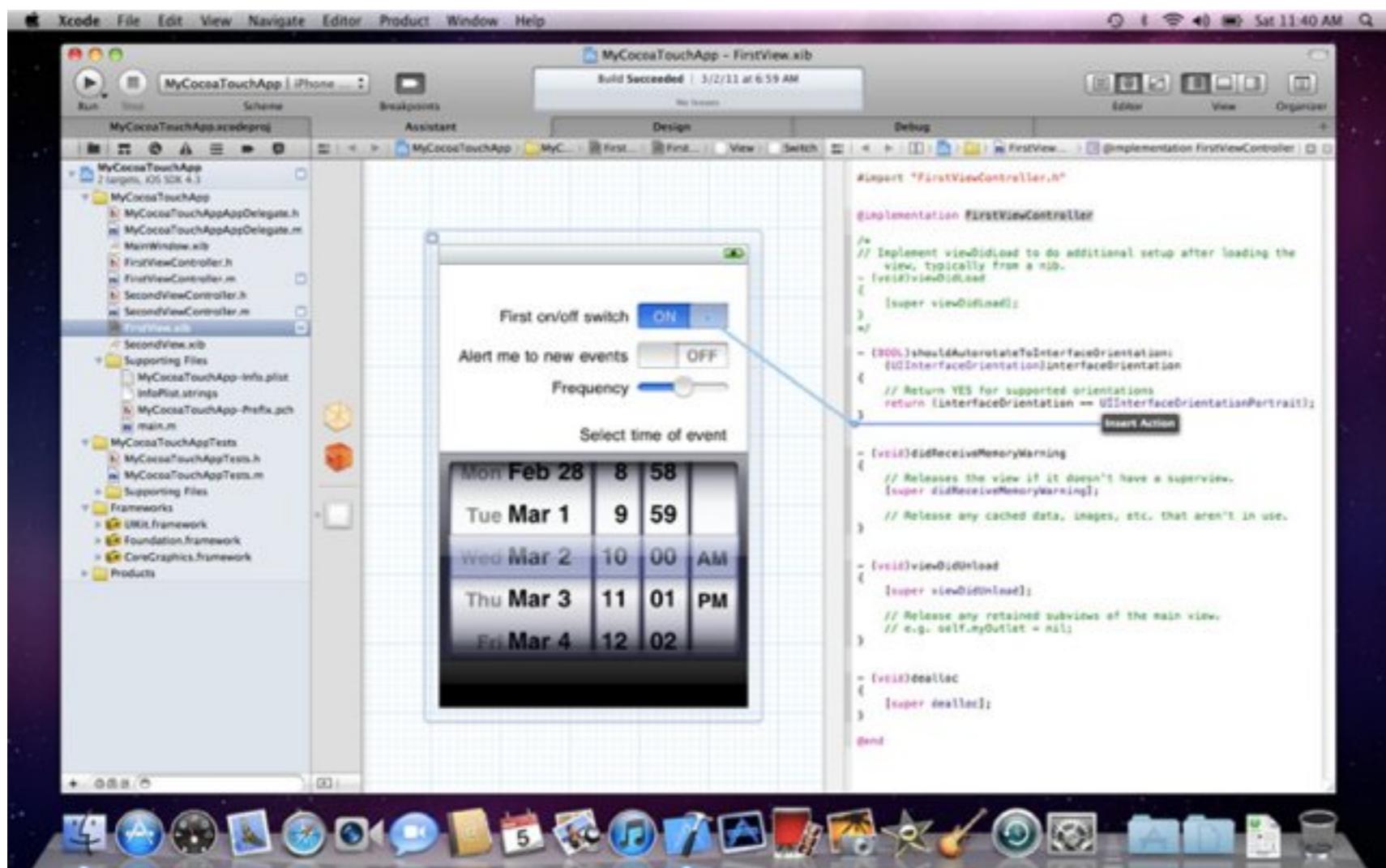


Xcode 3 - Beginnings of iOS Dev

The image displays three main components related to Xcode 3.1 development:

- Xcode Interface:** The top-left portion shows the Xcode IDE. It features a sidebar titled "iPhone Dev Center" with sections like Groups & Files, Overview, Action, Build and Go, Tasks, and Info. The Groups & Files panel lists frameworks such as CoreGraphics.framework, Foundation.framework, and UIKit.framework. The main area shows a file browser with files like RootViewController.m and RootViewController.h, and a code editor window showing the implementation of RootViewController.
- iPhone Simulator:** To the left of the Xcode interface, there is a virtual iPhone displaying a "Photo Albums" screen. The screen lists several albums: Photo Library (30), Hawaii (10), Slovenia (10), and Graduation Day (10). Each album has a small thumbnail preview.
- Welcome to Xcode 3.1:** The bottom-right portion is the "Welcome to Xcode" splash screen. It features a central illustration of a hammer resting on a blueprint. The text "Welcome to Xcode 3.1" is prominently displayed. Below the illustration, there are four cards with links:
 - Create your first Cocoa application**: Learn how easy it is to quickly create, build, and run your first Mac application.
 - Build your user interface**: Learn how Interface Builder works with Xcode to design your UI and wire your code to the visual controls. This card is highlighted with a red border.
 - Store your application data**: Learn how Xcode makes it easy to leverage Core Data to store your application's data.
 - Optimize your application**: Learn how to integrate Instruments into your Xcode workflow to analyze the performance of your application.

Xcode 4 - It All Comes Together (Literally)



- Brings IDE and Interface Builder together
- Seamless Interface for Creating iOS Apps
- Other than minor UI Changes, Xcode 8 (Todays version) still looks very similar to Xcode 4

Swift Playgrounds



- Think of it as a Swift Interpreter

A screenshot of the Swift Playgrounds application window. The title bar reads "swiftintro-1.playground — Edited". The window is divided into two main sections: the "Code Editor" on the left and the "Code Output" on the right. The Code Editor contains the following Swift code:

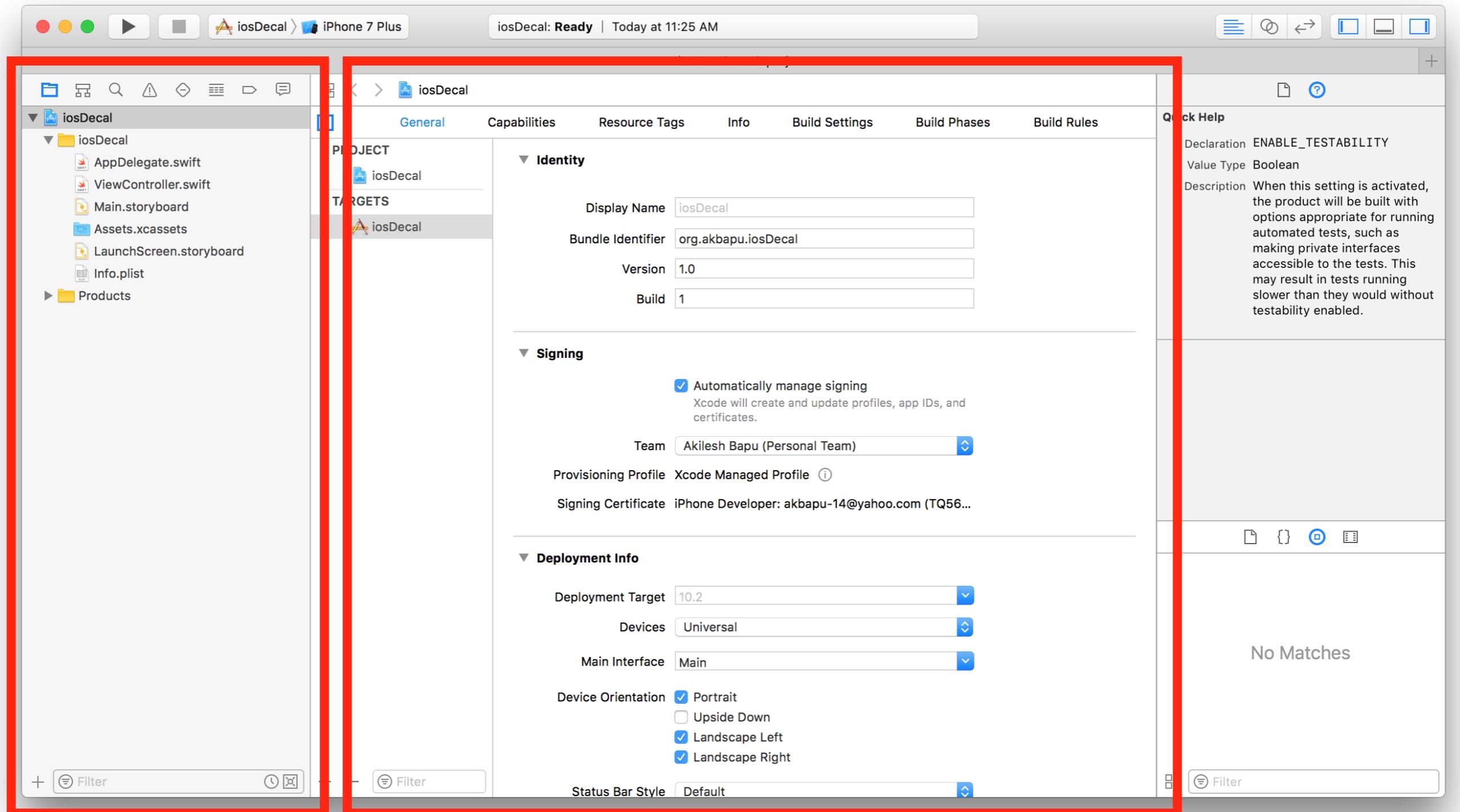
```
// Playground - noun: a place where people can play
import UIKit
var str = "Hello, playground"
```

The Code Output pane shows the result of the code execution: "Hello, playground".

Code Editor

Code Output

Xcode IDE Demo
Let's Create an App

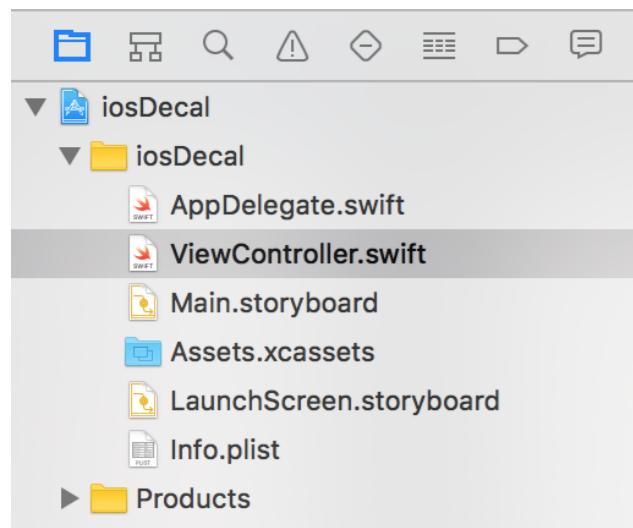


Project
Navigator

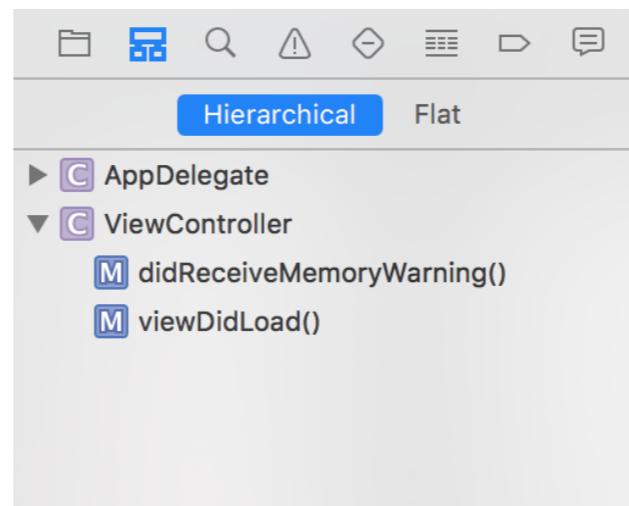
Project/Target Settings

Xcode Structure : Left Panel

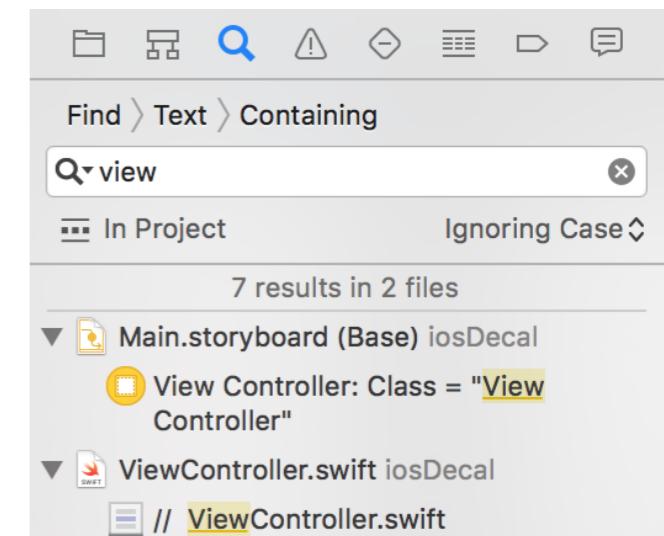
Project Navigator



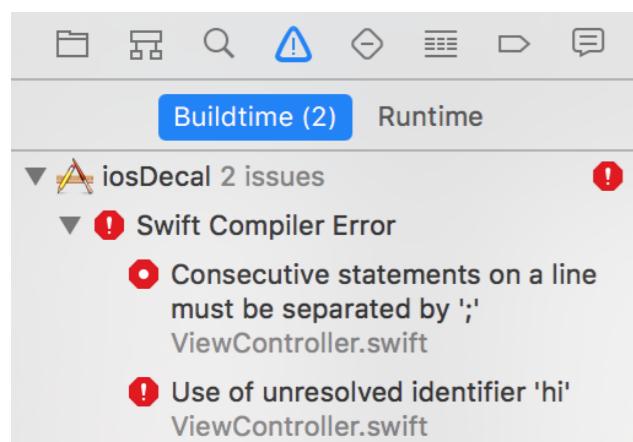
Symbol Navigator



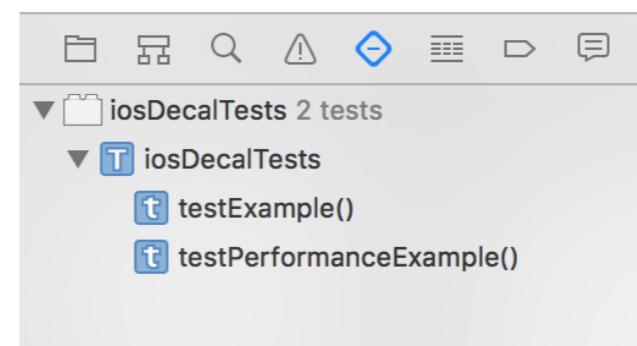
Find Navigator



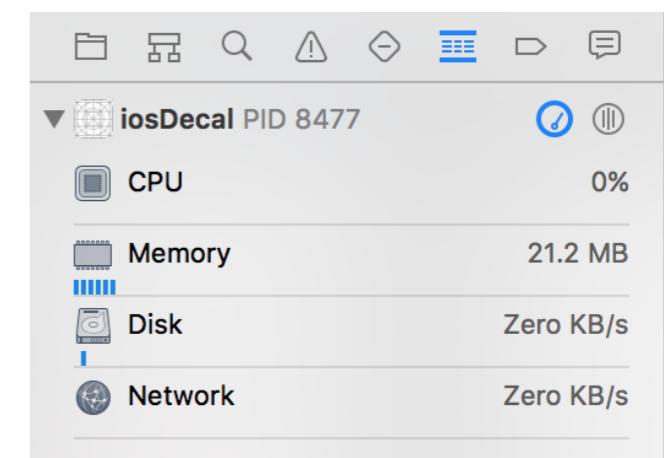
Issue Navigator



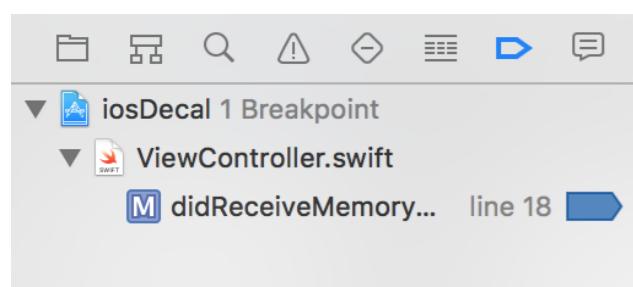
Test Navigator



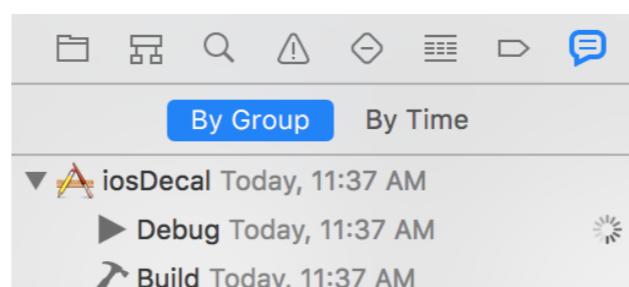
Debug Navigator



Breakpoint Navigator

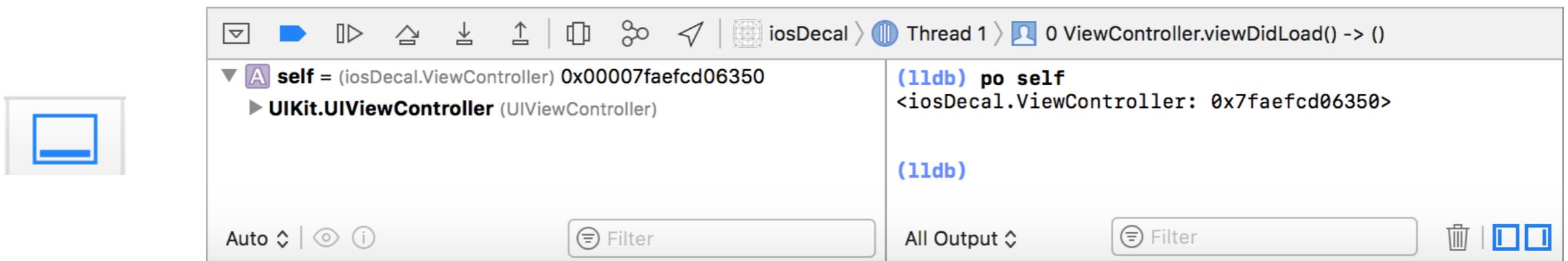


Report Navigator

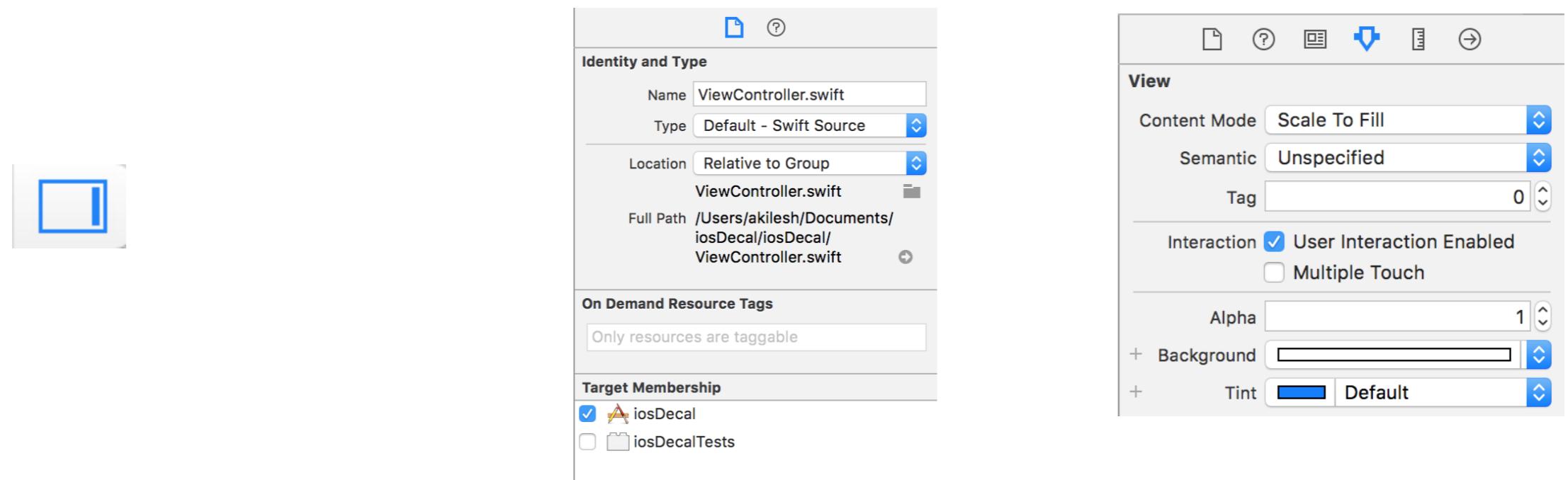


Xcode Navigator

Debug Area



Utilities Area



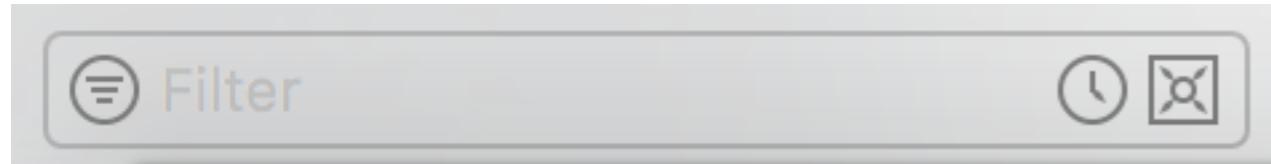
The Storyboard

The screenshot shows the Xcode Storyboard editor with the following details:

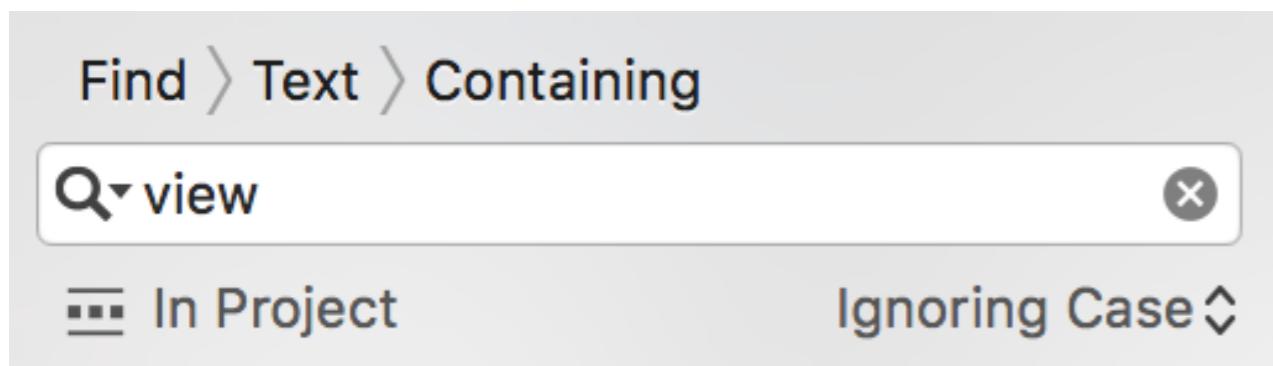
- Project Navigator:** Shows the project structure: iosDecal > MainStoryboard.storyboard.
- Document Outline:** Shows the View Controller Scene, View Controller, View, and a Button labeled "Hello World".
- Preview Area:** Displays a iPhone 7 simulator screen with the text "Hello World" centered.
- Attributes Inspector (Right Panel):**
 - Button Section:** Type: System, State Config: Default, Title: Plain, Title Text: Hello World, Font: System 36.0, Text Color: Default, Shadow Color: Default, Image: Default Image, Background: Default Background Image, Shadow Offset: 0, Reverses On Highlight: Unchecked, Shows Touch On Highlight: Unchecked, Highlighted Adjusts Image: Checked, Disabled Adjusts Image: Checked, Line Break: Truncate Middle.
 - Control Section:** Contains icons for File, Class, Object, and Document.
 - Descriptions:**
 - Button:** Intercepts touch events and sends an action message to a target object when it's tapped.
 - Item:** Bar Button Item - Represents an item on a UIToolbar or UINavigationItem object.
 - Fixed Space Bar Button Item:** Represents a fixed space item on a
- Bottom Bar:** Filter, View as: iPhone 7 (wC hR), zoom controls (69%), and other interface elements.

Xcode Tips and Tricks

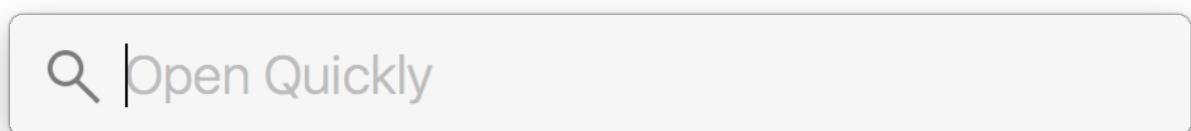
Search



Filter files by name, most recently edited, or whether it's source controlled



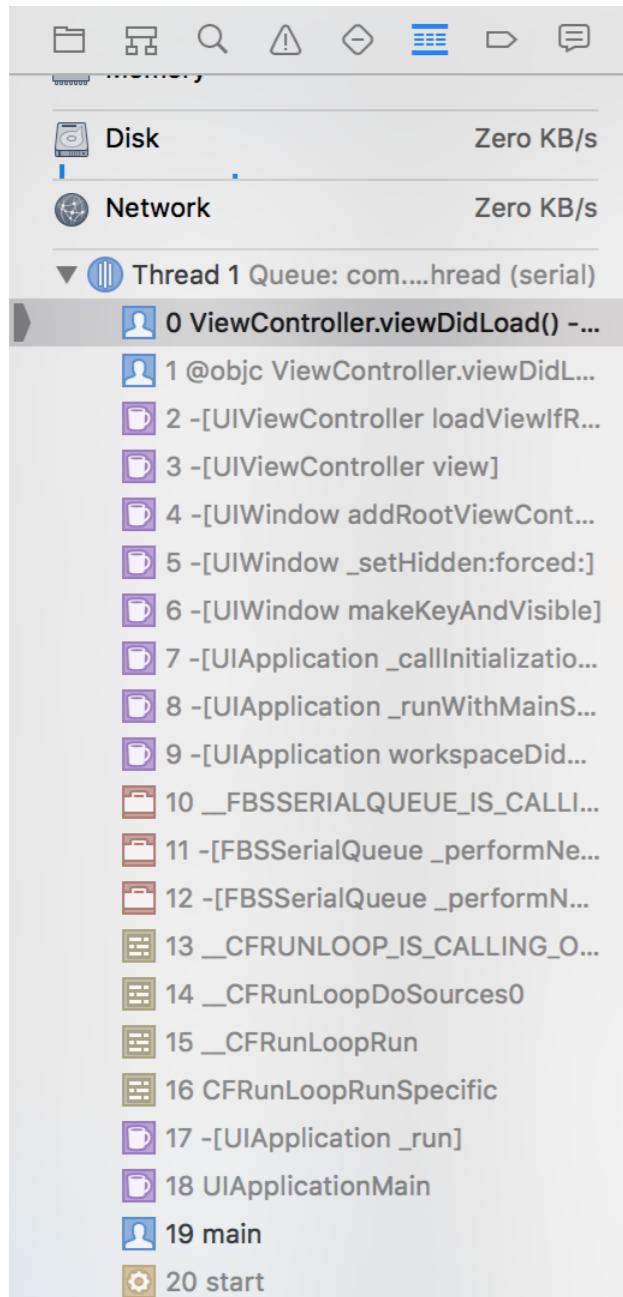
"Find" - Advanced search for strings in entire project (Can be SLOW)



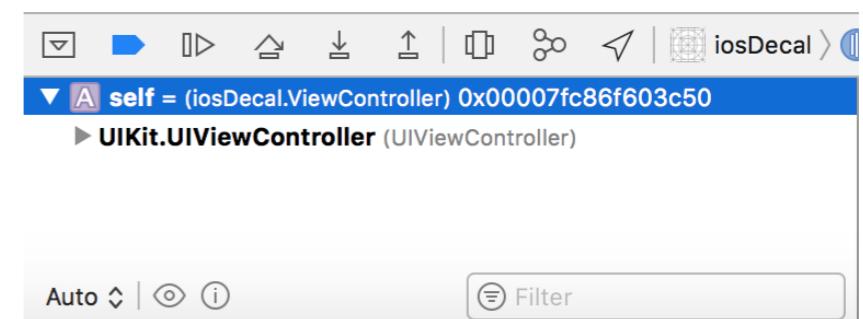
SHIFT-CMD-O - Jump to a certain file (SUPER HELPFUL and FAST)

Debugging

Call Hierarchy - Great if you step through code and end up at some random file and want to see how it's relevant



Currently Variables in Use - You can actually preview views

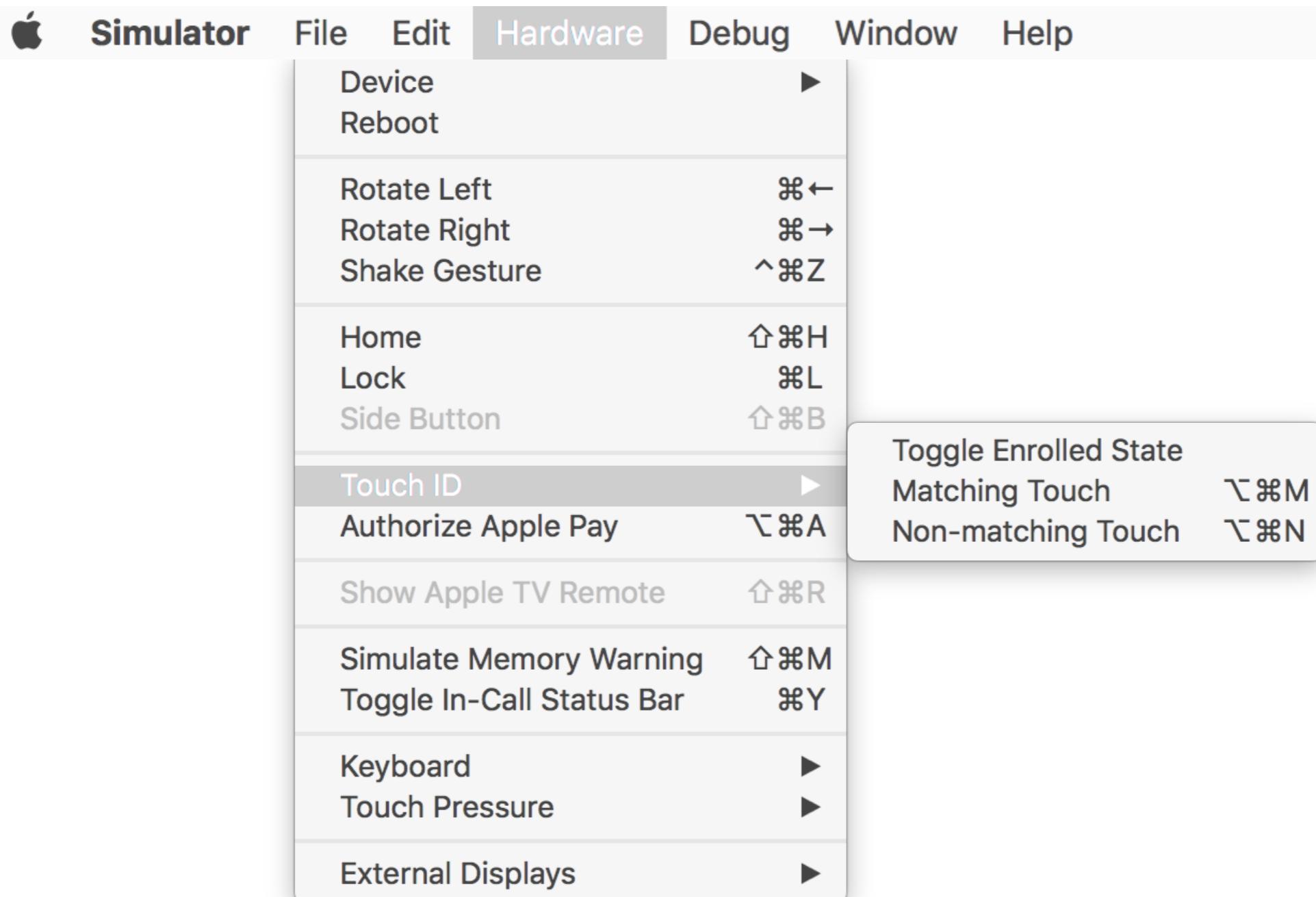


LLDB - Console Output +
"po" (RUN CODE AS THE APP IS
GOING)



Simulator Settings

Hardware and Debug Menus - Easily ignorable but amazingly helpful



Swift 3 Overview

Code Samples : functions

```
// Defining functions
func update(withNewData data: [String]) -> Bool {
    if data[0] == "Error" {
        return false
    }
    // ...
    return true
}
```

```
// Calling Functions
update(withNewData: ["iOS", "DeCal"])
```

Code Samples : functions

```
// Defining functions
func update(withNewData data: [String]) -> Bool {
    if data[0] == "Error" {
        return false
    }
    // ...
    return true
}
```

```
// Calling Functions
update(withNewData: ["iOS", "DeCal"])
```

Note: Internal Parameter (used in function) - data

External Parameter (used when calling function) - withNewData

Code Samples : Classes

```
class User { }

class Student: User {

    var enrolled: Bool
    let year: Int
    var favoriteDog: String?

    init(enrolled: Bool, year: Int) {
        self.enrolled = enrolled
        self.year = year
    }
}
```

Code Samples : Classes

```
class User { }
```

```
class Student: User {
```

```
    var enrolled: Bool
```

```
    let year: Int
```

```
    var favoriteDog: String?
```

```
    init(enrolled: Bool, year: Int) {
```

```
        self.enrolled = enrolled
```

```
        self.year = year
```

```
}
```

```
}
```

? → Optional Type



Optionals : Overview

A type that is logically allowed to have “no value”

- Able to be set to a value or nil

Properties of optional type are automatically initialized with a value of nil

“Unwrap” optionals with a “!” (Careful! If nil -> error)

```
var response: String? = "Hello World"  
print(response)
```

Optionals : Overview

A type that is logically allowed to have “no value”

- Able to be set to a value or nil

Properties of optional type are automatically initialized with a value of nil

“Unwrap” optionals with a “!” (Careful! If nil -> error)

```
var response: String? = "Hello World"  
print(response)
```

Console Output

```
Optional("Hello World")
```

Optionals : Overview

A type that is logically allowed to have “no value”

- Able to be set to a value or nil

Properties of optional type are automatically initialized with a value of nil

“Unwrap” optionals with a “!” (Careful! If nil -> error)

```
var response: String? = "Hello World"  
print(response!)
```

Console Output

“Hello World!”

Optionals : Overview

A type that is logically allowed to have “no value”

- Able to be set to a value or nil

Properties of optional type are automatically initialized with a value of nil

“Unwrap” optionals with a “!” (Careful! If nil -> error)

```
var response: String?  
print(response!)
```

Console Output

Optionals : Overview

A type that is logically allowed to have “no value”

- Able to be set to a value or nil

Properties of optional type are automatically initialized with a value of nil

“Unwrap” optionals with a “!” (Careful! If nil -> error)

```
var response: String?  
print(response!)
```

Console Output

```
fatal error: unexpectedly found nil while unwrapping an  
Optional value
```

Optionals : Usage

```
let dan = User()

func getWelcomeText(user: User) -> String? {
    if user.signedIn {
        return "Hello World!"
    }
    else {
        return nil
    }
}

if let response = getWelcomeText(user: dan) {
    print("The user responded: \(response)")
}
```

HWO : Swift 3 Tour (Reading)

Due this Thursday (before lab)

Next Lab : Xcode/Swift Tutorial Lab

Next Lecture : MVC and AutoLayout