iOS: Setup

Hello, World: iOS Edition

due by noon ET on Thu 4/7

Ingredients.

- Objective-C
- Xcode 4

Help.

Help is available throughout the week at `http://help.cs76.net/`! We'll do our best to respond within 24 hours. Be sure, though, to take advantage of lectures and sections as well as videos thereof!

Academic Honesty

All work that you do toward fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed by some project. Viewing, requesting, or copying another individual's work or lifting material from a book, magazine, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this course that you have submitted or will submit to another. Nor may you provide or make available your or other students' solutions to projects to individuals who take or may take this course (or CSCI S-76) in the future.

You are welcome to discuss the course's material with others in order to better understand it. You may even discuss problem sets with classmates, but you may not share code. You may also turn to the Web for instruction beyond the course's lectures and sections, for references, and for solutions to technical difficulties, but not for outright solutions to problems on projects. However, failure to cite (as with comments) the origin of any code or technique that you do discover outside of the course's lectures and sections (even while respecting these constraints) and then integrate into your own work may be considered academic dishonesty.

If in doubt as to the appropriateness of some discussion or action, contact the staff.

All forms of academic dishonesty are dealt with harshly.

Grades.

Your work on this problem set will be evaluated along one primary axis.

Correctness. To what extent is your code consistent with our specifications and free of bugs?

Required Reading.

☐ First curl up with *Learning Objective-C: A Primer*:

`http://developer.apple.com/library/mac/referencelibrary/GettingStarted/Learning_Objective-C_A_Primer/`

You'll find that it's a pretty quick read and hopefully whets your appetite for a bit more detail.

☐ It's time for more detail! Now curl up with *The Objective-C Programming Language*:

`http://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html`

This one's a few chapters, so be sure to click through to each by clicking **Next** in the bottom-right corner of most every page or by clicking through to each via the **Table of Contents** in the site's top-left corner.

Odds are you won't `retain` everything you read. (Pun intended.) But not to worry; it'll start to sink in once you get your hands dirty with code of your own.

☐ Next skim *Coding Guidelines for Cocoa*:

`http://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CodingGuidelines/CodingGuidelines.html`

Try to keep those guidelines in mind as you begin to write code of your own. We won't expect strict adherence to Apple's guidelines, so long as your own style is clean and consistent, but might as well familiarize yourself with some best practices.

☐ Now, for the sake of some review, read through the tutorial at:

`http://cocoadevcentral.com/d/learn_objectivec/`

If still hazy on some of its concepts, refer back to *The Objective-C Programming Language* as needed.

☐ Finally, spend some time reading

`http://www.otierney.net/objective-c.html`

particularly the code therein. Hopefully you're finding that you're developing an eye for Objective-C? If anything confuses, not to worry, reach out to classmates and staff via `help.cs76.net` with questions!

Incidentally, this particular tutorial covers Objective-C 1.0, not 2.0, so absent are topics like properties and fast enumeration.

Register as an Apple Developer.

☐ Phew, that was a lot of reading. Time to fill out some forms! Head to

`http://developer.apple.com/programs/register/`

if you're not already registered as an Apple Developer. Click **Get Started** when ready, and you should be prompted to **Create an Apple ID** or **Use an existing Apple ID**. Review the explanation beneath each option, then select the appropriate one and click **Continue**. Complete your personal profile on the page that appears, taking care to provide a valid email address, then click **Continue**. Complete your professional profile (with regard to your market and such) as you see fit, then click **Continue**. You'll then be prompted to agree to the **Registered Apple Developer Agreement**. Once you do, you'll receive a verification code via email. Input that code as prompted, then click **Continue**. You should now be registered as an Apple Developer!

If you plan to submit an app to Apple's App Store, whether during the semester or shortly thereafter, you might also want to sign up for the iOS Developer Program at

`http://developer.apple.com/programs/`

which costs $99/year or for the iOS Developer Enterprise Program at

`http://developer.apple.com/programs/ios/enterprise/`

which costs $299/year. However, know that you **do not need to sign up for either program** for this course. Because the course is part of the iOS University Program, you will be able to install apps that you write on your own iPad, iPhone, or iPod touch this semester for free. You won't be able to submit any apps to the App Store, though, unless you sign up for one of the paid programs. See `http://developer.apple.com/programs/which-program/` for more details.

Xcode.

From this point forward in the story, you'll need to be sure that you're on an Intel-based Mac running Mac OS X Snow Leopard version 10.6.6 or higher. To find out whether a particular Mac qualifies, select **About This Mac** from the menu. Hopefully you'll see **Version 10.6.6** (or higher) as well as some mention of **Intel** next to **Processor**. You'll also need at least 15GB of disk space available on your Mac's startup volume (*e.g.*, **Macintosh HD**); to check its available space, highlight the volume in the Finder, then select **Get Info** from the Finder's **File** menu.

If you don't own a Mac that meets these requirements and are distant from Cambridge, you will need to borrow or otherwise procure one. (Afraid these are Apple's requirements, not ours!) If you don't own a Mac that meets these requirements but are local to Cambridge, you're welcome to use the Mac lab at 53 Church Street in Harvard Square, the hours of which are listed at `http://lab.dce.harvard.edu/`; do bring proof of your enrollment in the course (*e.g.*, any paperwork you received from the school in the mail) in case asked for it by the User Assistants at the front desk.

If, at this very moment, using a Mac at 53 Church Street, where everything's already installed for you, go ahead and skip the following two checkboxes. Otherwise it's time to install Xcode!

☐ If you only signed up as an Apple Developer (for free) and neither of the paid programs, visit

`http://itunes.apple.com/us/app/xcode/id422352214?mt=12`

with Safari, even if not your preferred browser, then click **View in Mac App Store** at top-left. That button should trigger the Mac's own App Store to launch, at which point you can buy Xcode for $4.99. Do so and proceed to install it once downloaded.*

If you did sign up for the iOS Developer Program (for $99/year) or the iOS Developer Enterprise Program (for $299/year), head to

`http://developer.apple.com/xcode/`

and click **Log in** where prompted in order to download Xcode 4 for free. (Well, "free," seeing as you did just pay $99 or $299.) Proceed to install it once downloaded.*

No matter how you download Xcode 4, realize that the installer is over 4GB in size, so it might take some time!

If the installation appears to hang, see

`http://discussions.apple.com/thread.jspa?threadID=2777384`

for some tips. And if, after installing Xcode 4, you have trouble synching or backing up an iOS device in iTunes, see

`http://support.apple.com/kb/HT1747?viewlocale=en_US`

for some tips.

☐ Once Xcode 4 is installed, go ahead and launch it. (It can be found on your Mac's startup volume in `/Developer/Applications/`). Proceed to select **Preferences**... under the **Xcode** menu (to the right of the menu). Click the **Documentation** icon atop the window that appears. If you only see **Mac OS X 10.6 Core Library** and **Xcode 4.0 Developer Library**, click **Check and Install Now**, thereafter providing your Mac's password if prompted. You should then see **iOS 4.3 Library (Apple iPhone OS 4.3)** in gray; click **GET** to the right of that library, again providing your Mac's password if prompted. (No need to download **Mac OS X Legacy Library**). After Xcode is done getting the library, it should re-appear in black. Confirm as much by selecting **Xcode Help** under Xcode's **Help** menu, then click the eye icon at top-left. You should see **iOS 4.3 Library** among the

* If you already have an earlier version of Xcode installed, Xcode 4's installer will rename your existing `Developer` directory to `Developer-old`.

libraries that then appear to the left; click it, and documentation for the entire iOS Developer Library should appear to the right. Realize, though, that you can access that same documentation at `http://developer.apple.com/library/ios/`.

Hello, World.

☐ Alright, it's time to take Xcode for a spin. Go ahead and launch it, if not running already, and create a new project, as by clicking **Create a new Xcode project** in the splash screen that displays upon launch (if you didn't disable) or by selecting **New > New Project…** from Xcode's **File** menu. When prompted to choose a template for your new project, select **Application** under **Mac OS X** (not **iOS**), select **Command Line Tool**, then click **Next**. When prompted for a **Product Name**, input **MacApp** and then select **Foundation** (not **Core Foundation**) next to **Type**. Choose a location for the project when prompted, then click **Create**. (Best not to save anything in `/Developer`, lest you upgrade Xcode in the future.)

A window entitled **MacApp – MacApp.xcodeproj** should then appear. Go ahead and expand each of the triangles at left (except for **Foundation.framework**), and you should see `main.m`, `MacApp.1`, and `MacApp-Prefix.pch` among their contents. Click `MacApp.1` then hit **delete** on your keyboard; when prompted to permanently delete it, click **Delete**. (That file's just a template for a "man page" that you won't need for this project. See `http://en.wikipedia.org/wiki/Man_page` if curious.)

Now click `main.m`, and you should see its contents at right. Ha, turns out Xcode already wrote this program for you! But go ahead and change

```
@"Hello, World!"
```

to some other `NSString` of your choice. Then modify the comments atop the file so that they contain your full name and your Apple ID (*i.e.*, the email address with which you registered as an Apple Developer). Henceforth, be sure that `main.m` always contains at least those details for any project you write and submit.

Go ahead and click **Run** in Xcode's top-left corner (or hit ⌘-R on your keyboard), and you should find that Xcode's **Debug area** appears at bottom, among whose boldfaced output should be your `NSString`! You've just compiled and run `MacApp.app`, your very first app!

There's nothing wrong with leaving that **Debug area** open all of the time, but, so that you get a feel for Xcode's UI, go ahead and figure out how to hide it. (Hint: poke around Xcode's top-right corner.) In fact, poke around the rest of Xcode's UI, including its menu, to get a sense of its features. (If you start to feel a bit overwhelmed, simply close whatever you opened!) The interface is a bit complex, at least when you have everything showing, but not to worry, we'll point out its features as needed.

☐ Okay, that first app isn't going to impress any of your friends. Let's try a bit harder.

Create a new project in Xcode. (Remember how?) When prompted to choose a template for your new project, select **Application** under **iOS** (not **Mac OS X**), select **Window-based Application**, then click **Next**. (A **Window-based Application** is the simplest of the templates provided; we'll see some others before long.) Input **iPhoneApp** next to **Product Name**, input **edu.harvard.dce** next to **Company Identifier**, select **iPhone** next to **Device Family**, and leave both **Use Core Data** and **Include Unit Tests** unchecked. Then click **Next**. Choose a location for the project when prompted, then click **Create**.

A window entitled **iPhoneApp – iPhoneApp.xcodeproj** should then appear. As before, go ahead and expand each of the triangles at left (except for **UIKit.framework**, **Foundation.framework**, and **CoreGraphics.framework**), and you should see more files than last time.

Go ahead and click `main.m`, and its contents should appear to the right. As before, it's that file that will kick off this program. Notice that it calls `UIApplicationMain`. Hold down **option** on your keyboard, click the name of that function, and some documentation thereof should appear in a callout. (If you'd like even more detail, click the little book icon in that callout's top-right corner, and the entire UIKit Framework's documentation will appear inside of Organizer.) Apparently, `UIApplicationMain` creates "the application object and the application delegate." Hm, now where (if not what) are those…

Go ahead and click `MainWindow.xib`, and Xcode's "interface builder" should appear to the right. Odds are, to the left the "graph paper" you see, there'll be four icons (three cubes and one square). Down below those icons should be a small arrow within a rectangle; click the arrow to widen the icons' area. You should now see names next to those icons. Click **File's Owner** under **Placeholders**. Then, if it's not open already, open Xcode's **Utilities** by clicking the rightmost icon above **View** in Xcode's top-right corner. Below **View** are six smaller icons; click the third from the left to reveal Xcode's **Identity inspector**. (Incidentally, if at any point unsure what some icon means, hover over it with your cursor for a moment, and a tooltip should appear.) Assuming **File's Owner** is still highlighted, you should see that its class is `UIApplication`. Aha! Here's the "application" that `UIApplicationMain` promised to create.

Now click the rightmost icon beneath **View** to reveal Xcode's **Connections inspector**. Assuming **File's Owner** is still highlighted, you should see that your application's `delegate` is **I Phone App App Delegate**. (You might need to make the **Utilities** area wider to see all of those words.) Alright, now where is this supposed delegate?

Oh! Click **I Phone App App Delegate** under **Objects**, which itself is below **Placeholders**. Assuming you still have the **Connections inspector** revealed, you should that this delegate's `window` is **Window** (which itself is of type `UIWindow`, which just so happens to be a subclass of `UIView`). In other words, your delegate's default view is the iPhone's window itself.

Now what exactly is an **I Phone App App Delegate**? Well, assuming you still have it highlighted, reveal the **Identity inspector** again. (Remember how?) You should see that **I Phone App App Delegate** is of type `iPhoneAppAppDelegate`. (And therein lies the origin of

that object's name; Xcode has simply inserted spaces before any capitalized letters so that the object's name is, er, more readable.)

Feels like a quick recap might be in order: `main.m` calls `UIApplicationMain`, which creates an application (*i.e.*, `UIApplication`) object as per `MainWindow.xib`. That application object, meanwhile, has delegated control of the app to an object of type `iPhoneAppAppDelegate`. Because that delegate is wired to the iPhone's window (*i.e.*, `UIWindow`), it will have control of the screen.

So what is an `iPhoneAppAppDelegate` object exactly? Well, go ahead and click `iPhoneAppAppDelegate.h`. Interesting, not all that much there, but notice that it does have an "outlet" for a `UIWindow` (as we saw in `MainWindow.xib`). And it also conforms to the `UIApplicationDelegate` protocol, which is good, since that means a `UIApplication` object can delegate to it control of the app.

Now click `iPhoneAppAppDelegate.m`. Whoa, lots of instance methods defined in this one (most of which are declared in the `UIApplicationDelegate` protocol). But the most important one is probably `application:didFinishLaunchingWithOptions:`, since that one will be called first so that the app's `UIWindow` is made visible for the user to see.

Let's ensure that the user has something interesting to look at. Go ahead and click `MainWindow.xib` again, and open Xcode's **Utilities**, if not still open from before. Toward that area's bottom should be a library of **Objects**, the first of which is **Label**. (If not, you can select **Utilities > Object Library** under Xcode's **View** menu.) Drag a **Label** from that area to the center of the iPhone window depicted in Xcode's middle, atop the graph paper. (If you see no window, try clicking **Window** under **Objects** at left to make it appear.) Crosshairs should appear once you've aligned the **Label** perfectly. Assuming the **Label** is selected (as indicated by a rectangle of six dots around it), select the **Attributes inspector** among Xcode's **Utilities** at right. (We'll leave it to you to find that one.) Then specify some **Text**, a **Font**, and a **Text Color** (as well as any other attributes that you'd like) for your **Label**. Any changes you make should appear in the graph paper's window.

Once pleased with your **Label**, ensure that **iPhoneApp | iPhone 4.3 Simulator** is selected in the drop-down to the right of the **Run** button in Xcode's top-left corner. Then click **Run** (or hit ⌘-R on your keyboard). If all goes well, the iOS Simulator should launch with your app! If not, try to retrace your steps to determine what might have gone wrong. (Worst case, perhaps start over from the beginning!)

Incidentally, even though you didn't need to modify `main.m` this time, do remember to put your full name and your Apple ID atop the comments that file so that we know who you are.

What about those other files among this project's **Supporting Files**? Well, in `iPhoneApp-Info.plist` and `InfoPlist.strings` are some (default) properties for your app. In `iPhoneApp-Prefix.pch` are some (soon-to-be precompiled) headers that will be prepended to each of your source files. As for `iPhoneApp.app` under **Products**, that's your app. Or, at least,

it will be if you build your code for an actual device (which you don't need to yet); for now, it's probably red, which means you've only built your app for the iOS Simulator.

Phew, nicely done!

Looking Ahead: Student's Choice.

☐ It's time to propose your choice of iOS projects! Even if you're not yet sure what the platform can do (or how easily you can do it), propose a vision for an iOS app nonetheless, and we'll help you gauge its viability. You're welcome to target iPhones or iPads or both (or just the simulator thereof). As for the app's nature, the sky is the limit, so long as your teaching fellow approves.

You're welcome to propose re-implementing for iOS a project you previously wrote for Android (whether for your choice or ours). You're also welcome to propose implementing a project that you'd like to use or sell outside of the scope of the course, so long as you disclose as much in your proposal.

As before, this proposal is not binding; you can change your plans later, so long as your teaching fellow approves.

Submit your proposal before this project's deadline at:

`https://www.cs76.net/login/?form=ios-setup`

We'll then follow up via email with thumbs up or down!

How to Submit.

☐ First, open up each of your projects (*i.e.*, `MacApp` and `iPhoneApp`) in Xcode, select **Clean** from Xcode's **Product** menu, and then close them again.

Then create a ZIP file containing both of those projects' folders, and name the ZIP `########.zip`, where `#######` is your 8-digit Harvard ID (HUID), the same credential that you use to log into `help.cs76.net`.

Then head to `https://www.cs76.net/submit`, click the **login** link at top-right, click the link to your TF's dropboxes at top-left, click this project's own folder, click **Upload File**, and upload your ZIP file as prompted; no need to give it a title. Be sure not to click the wrong project's folder. You may re-submit in this same manner as many times as you'd like. Just take care to delete any prior submissions.

Be sure not to submit or re-submit after this project's deadline.