

iOS: Setup
Hello, World: iOS Edition

due by noon ET on Wed 4/4

*Note that submitting this project's Google form will take some time;
best not to wait until the last minute!*

Ingredients.

- Objective-C
- Xcode

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. 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.

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, select the appropriate one, click **Continue**, then follow the on-screen prompts.

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 Developer 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 Lion (Mac OS X 10.7.3 or later). To find out whether a particular Mac qualifies, select **About This Mac** from the  menu. Hopefully you'll see **Version 10.7.3** (or later) as well as some mention of **Intel** next to **Processor**.

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/id497799835?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 download Xcode 4.3.1 for free. 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 4.3.1 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, realize that the installer is over 4GB in size, so it might take some time!

- Once Xcode is installed, go ahead and launch it. (It can be found on your Mac's startup volume in `/Applications/`.[†]) Proceed to select **Preferences...** under the **Xcode** menu (to the right of the **Apple** menu). Click the **Documentation** icon atop the window that appears. Highlight **Command Line Tools**, then click **Install**, providing your Mac's password if prompted.

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**. On the screen that appears: input **MacApp** for **Product Name**; input **edu.harvard.extension** for **Company Identifier**; select **Foundation** (not **Core Foundation**) next to **Type**; then click **Next**. Choose a location for the project when prompted, check the box to create a local git repository if you'd like, then click **Create**.

A window entitled **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

* If you already have an earlier version of Xcode installed, Xcode 4.3.1's installer will allow you to retain or remove it.

† Prior versions of Xcode could be found in `/Developer/Applications/`.

page” that you won’t need for this project. See http://en.wikipedia.org/wiki/Main_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 **Single View Application**, then click **Next**. (A **Single View Application** is among the simplest of the templates provided.) On the screen that appears: input **iPhoneApp** for **Product Name**; input **edu.harvard.extension** for **Company Identifier**; leave **Class Prefix** blank (it’s okay if you see a placeholder of **XYZ** in gray); select **iPhone** next to **Device Family**; leave **Use Storyboards** unchecked; check **Use Automatic Reference Counting**; leave **Include Unit Tests** unchecked; then click **Next**. Choose a location for the project when prompted, check the box to create a local git repository if you’d like, then click **Create**.

A window entitled **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,” the latter of which is simply an object to which control of your application is “delegated.” How does it know

what kind of object to create for delegation? Via its fourth argument! Indeed, notice how the fourth argument to `UIApplicationMain` in `main.m` mentions `AppDelegate`, which just so happens to be a class declared in `AppDelegate.h` and defined in `AppDelegate.m` (both of which are provided for you via the **Single View Application** template).

Now take a peek at `AppDelegate.m`. Notice how it allocates a “view controller” in `application:didFinishLaunchingWithOptions:` and then initializes it with a nib (*i.e.*, `ViewController.xib`). Interesting!

Go ahead and click `ViewController.xib`, and Xcode’s “interface builder” should appear to the right. Hm, not that interesting after all; it’s just a blank view. Let’s change that!

In Xcode’s top-right corner, click the rightmost icon above **View** to show Xcode’s **Utilities** if not visible already. Toward that area’s bottom should be a library of **Objects** (assuming you have the little cube icon selected), the first of which is **Label**. Drag a **Label** from that area to the center of the view atop the graph paper in Xcode’s middle (*i.e.*, the iPhone-sized rectangle). Crosshairs should appear once you’ve aligned the **Label** perfectly. Assuming the **Label** is selected (as indicated by a rectangle of six squares around it), select the **Attributes inspector** among Xcode’s **Utilities** at right. (Hover over each of the icons below **View** in Xcode’s top-right corner to find that inspector.) 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 5.1 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 the other files that come with this template? Well, `ViewController.h` and `ViewController.m` (along with `ViewController.xib`) collectively govern the behavior of this application’s view controller. Since our only changes to the template were aesthetic, it sufficed to make them in the nib. In `iPhoneApp-Info.plist` and `InfoPlist.strings` are some (default) properties for your app. In `iPhoneApp-Prefix.pch` are some 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:

<http://goo.gl/OSnrZ>

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.