# CSCI E65g: Mobile Application Development Using Swift and iOS

## Fall 2018

## Assignment 0

### Issued: 09/04/2018 Due: 09/06/2018

#### Last revised: 2018-09-03: Updates for Fall 2018

**Purpose**: Get acquainted with all tools and course logistics, especially submitting work. Learn a bit about the development environment. Get the basics out of the way to clear the path for conceptual learning.

**Scoring**: None; these are just necessary prerequisites to submitting actual work.

**Reading and Videos**: The videos and Swift book are due by the start of 2nd class on Tuesday.

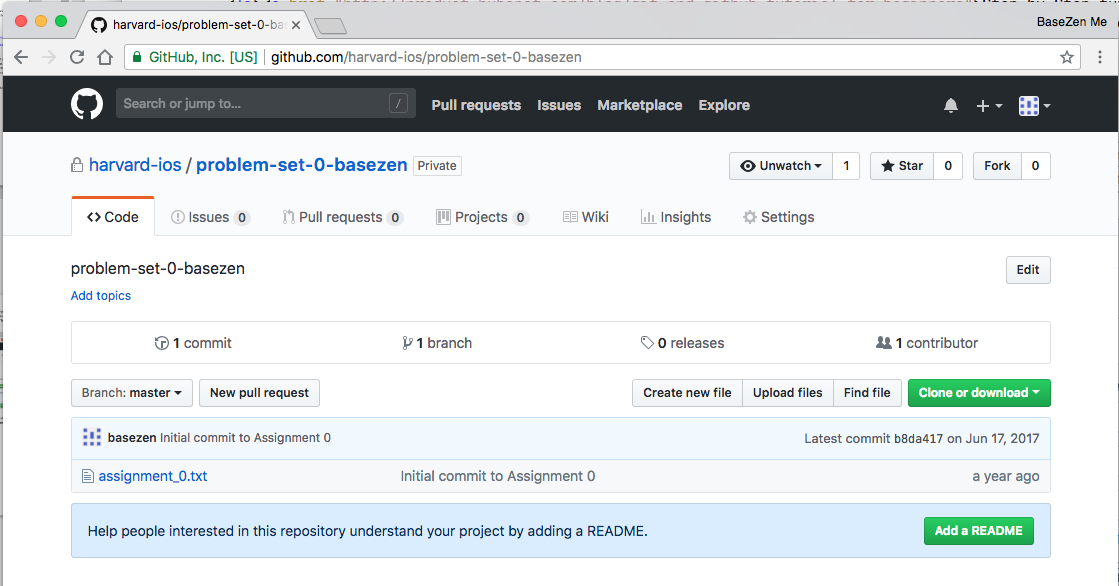
1. [Step-by-Step tutorial for Git](http://product.hubspot.com/blog/git-and-github-tutorial-for-beginners)
2. [Conceptual Introduction to Git](https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control)
3. [Swift iBook **The Swift Programming Language (Swift 4.1)**](https://itunes.apple.com/us/book/the-swift-programming-language-swift-4-1/id881256329)
4. [Official short intro video at WWDC 2014](https://www.youtube.com/watch?v=MO7Ta0DvEWA)
5. [Extended breakout session on Swift 1.0](https://www.youtube.com/watch?v=A0C6L4XmrZM)
6. [WWDC 2015 Swift 2.0 Intro](https://developer.apple.com/videos/play/wwdc2015/106/)
7. [WWDC 2016 Swift 3.0 Intro](https://developer.apple.com/videos/play/wwdc2016/402/)

**Format**: Except for the starter Xcode project, all answers should be put into a file called assignment\_0.*extension* at the top level of your repository (Example: **assignment\_0.rtf**). All output must be numbered with the corresponding problem number. Allowable formats (for all future textual work as well) are: Plain text (.txt), Rich Text (.rtf), or HTML (.html). Unless you already have a preferred text editor, the Mac OS built-in [Text Edit](https://support.apple.com/guide/textedit/welcome/mac) is fine.  
*We specifically do not allow proprietary or binary formats, such as Word (.docx), Pages (.pages), or PDF (.pdf).*

#### Problems

1. Create a free [GitHub (https://github.com)](https://gitbhub.com/) account using the e-mail address used for this course.  
   OUTPUT: The e-mail address and username chosen.

**NOTE** E-mail us at [bromberg@fas.harvard.edu](mailto:bromberg@fas.harvard.edu) and [benjenkinsv95@gmail.com](mailto:benjenkinsv95@gmail.com) with your GitHub username right away. The following GitHub-related steps will not work until we have added you to our GitHub classroom.

1. Create your personal Assignment 0 repository by following [this invitation link](https://classroom.github.com/assignment-invitations/b09637f79fea84a7888c83e081f4d7c2).  
   OUTPUT: The web address of your newly created repository.
2. You are now viewing the web interface to your Assignment 0 repository. (A separate repository will be created for each of your assignments.) Ensure it looks like this (except you won&t see any files yet as shown):  
     
   OUTPUT: None
3. Read Chapter 1 of the “Introduction to Git” reading but skip **Section 1.5 – Installing**, since Xcode will have installed git for you.  
   OUTPUT: In one or two sentences, assess your current experience level with git before, and how much this helped.
4. Open the built-in [Terminal](http://blog.teamtreehouse.com/introduction-to-the-mac-os-x-command-line) utility to get familiar with using it for essential tools, especially git. We recommend [pinning it to the dock (search page for **Options > Keep in Dock**)](https://support.apple.com/kb/ph25709?locale=en_US).
5. The command-line utility bash (a version of the Unix shell) lets you access a huge array of advanced functionality through typed commands, although documentation is sparse. For developers, it provides complete access to all features of git (and the ability to interact directly with a simplified Swift environment). Some customization makes it easier to handle.

Make the shell prompt more informative by defining the Primary Shell prompt variable PS1:

echo "export PS1='\u@\h:\w$ '" >> ~/.bashrc  
source ~/.bashrc

Successful result:

daniel@inspector-gadget:~$

* + The escape sequence \u shows your username
  + \h shows the computer name
  + \w shows the current working directory, a very important concept
  + ~ is Unix shorthand for “user’s home directory”.

Here we can see my username is daniel, the machine name is inspector-gadget, and my working directory is my home directory. Note that once we run (source) the .bashrc script, the shell prompt updates from bash$ to daniel@inspector-gadget:~$.  
OUTPUT: What’s the resulting shell prompt on your computer?

1. In Terminal at the bash prompt, create a master directory to store all of your work. By placing your work directory inside Desktop, you can navigate to it using its Finder icon.  
   mkdir ~/Desktop/e65g  
   OUTPUT: None
2. Enter the directory (technically, change the working directory of the shell) and notice the prompt changes.  
   cd ~/Desktop/e65g  
   List (ls) all files (-a) using the long format (-l) in the newly created directory: ls -al

OUTPUT: THe output of the ls command. The directory is empty yet some output appears. Find out what it signifies and explain briefly here.

1. Use the Assignment 0 repository web address obtained earlier to connect your computer to GitHub. (Technically, you are *cloning* your personal GitHub repository onto your computer. As you work, the GitHub (the *remote*) version of the repository will become out of date, so to communicate with the teaching staff you will need to synchronize upstream with a *push* command.) You’ll be asked for your GitHub username and password. git clone https://github.com/harvard-ios/assignment-0-*username*.git

OUTPUT: The feedback that Git gives from this command (the warnings are OK.)

1. The following steps use the four different areas of git that your content travels through: **working directory**, **staging area**, **local repository**, and **remote repository**. That seems like a lot of places to keep track of at first. Repetition will make it simple and obvious over time. *Your work is only truly safe (and visible for grading) when it has reached the final stage, the****Remote****, which is on GitHub cloud storage, that is, visible on the GitHub website.* All other copies are only on your own computer.
   * Using your favorite text editor, save the assignment text file you’re working on to this directory (~/Desktop/e65g/assignment-0-*github-username*), which is git's **working directory**. Confirm by listing the contents of the directory:

ls -l  
OUTPUT: The output of this command

* + Ask git to report the current status. It should notice the new file, but tell you in its own way that the file is currently unrecognized:  
    git status  
    OUTPUT: Output of this command
  + Add a special file called .gitignore in the same working directory. Don’t forget the leading period! This file tells git which files are temporary and maintained by the operating environment, and irrelevant to the repository. The details are unimportant, but briefly, the first line below is the hidden file used for file icon appearance in the Finder and the second is a directory that [stores appearance information](https://stackoverflow.com/questions/10956312/is-the-project-xcworkspace-file-important) in Xcode (such as exact window and pane sizing). These files change frequently and, if not ignored, will cause unnecessary notifications and repository activity. (Since this is a UI class, these files actually encapsulate an important concept: the **View State**! Take note.) It should have the following in plain text contents.
  + .DS\_Store

xcuserdata/

**As a rule, create this file for every assignment right away.**

* + Prepare a commit transaction by adding the files to the **staging area**.  
    git add assignment\_0.*extension*  
    git add .gitignore  
    OUTPUT: Output of git status after completing these commands
  + Complete the commit transaction, so the files become part of git (but currently, only in your **local repository**).

git commit assignment\_0.*extension* .gitignore -m 'Initial commit of Assignment 0 output'

OUTPUT: Output of this command and git status afterward

* + Synchronize your local repository to the **remote repository** with a push operation.  
    git push  
    OUTPUT: Output of this command and git status afterward
  + Reload the GitHub assignment repository page and ensure your files show up.  
    OUTPUT: None

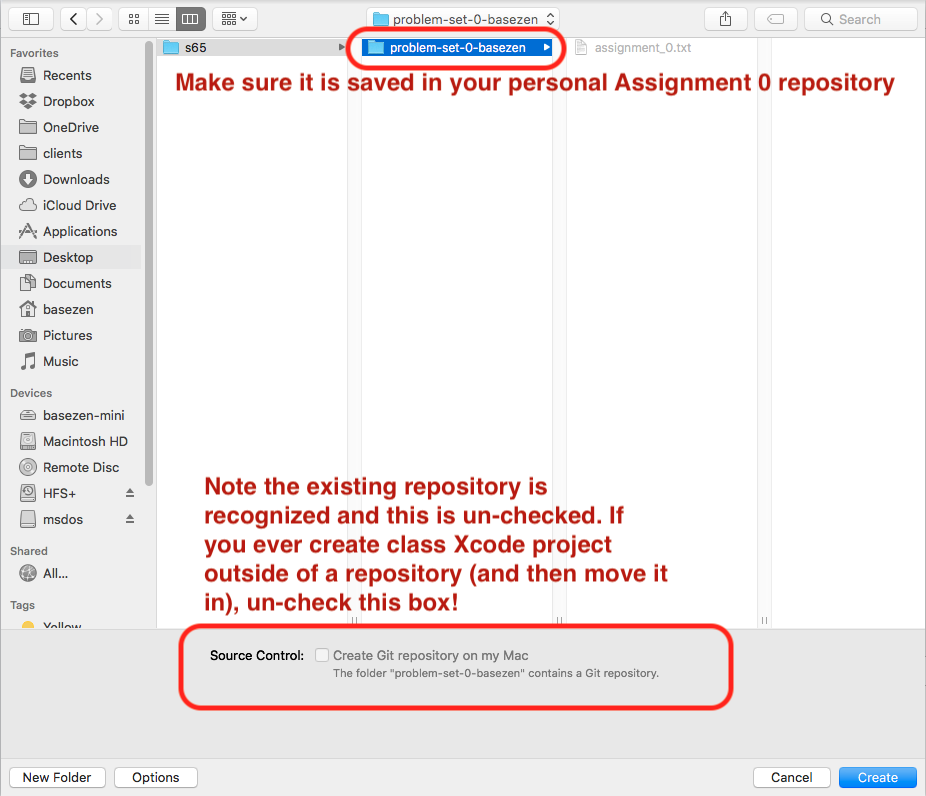
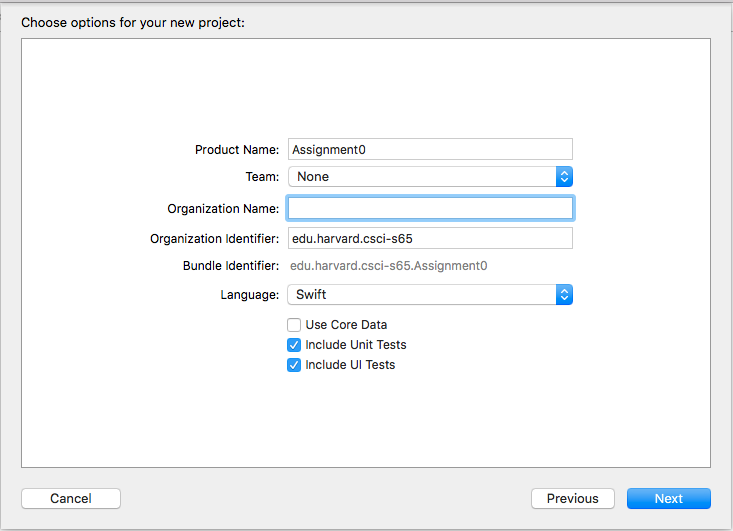
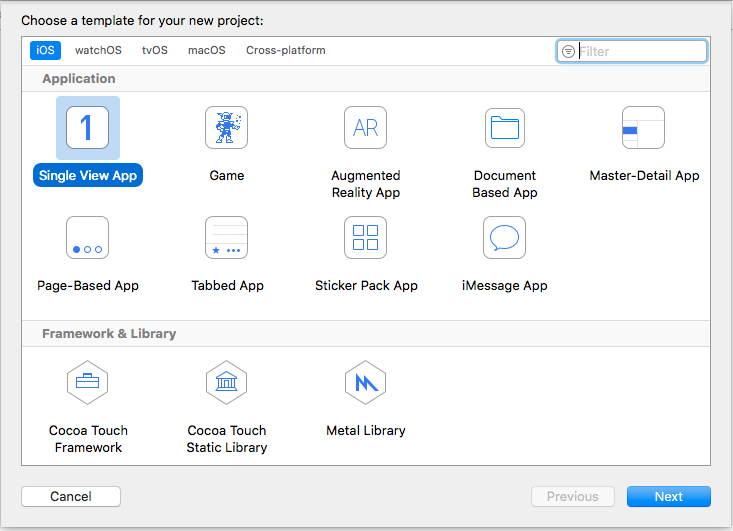
1. Install the shared course materials in a separate repository. **NOTE**: Never save your personal work or any other files here! This should be a STRICTLY READ-ONLY area. Using the same technique as above.

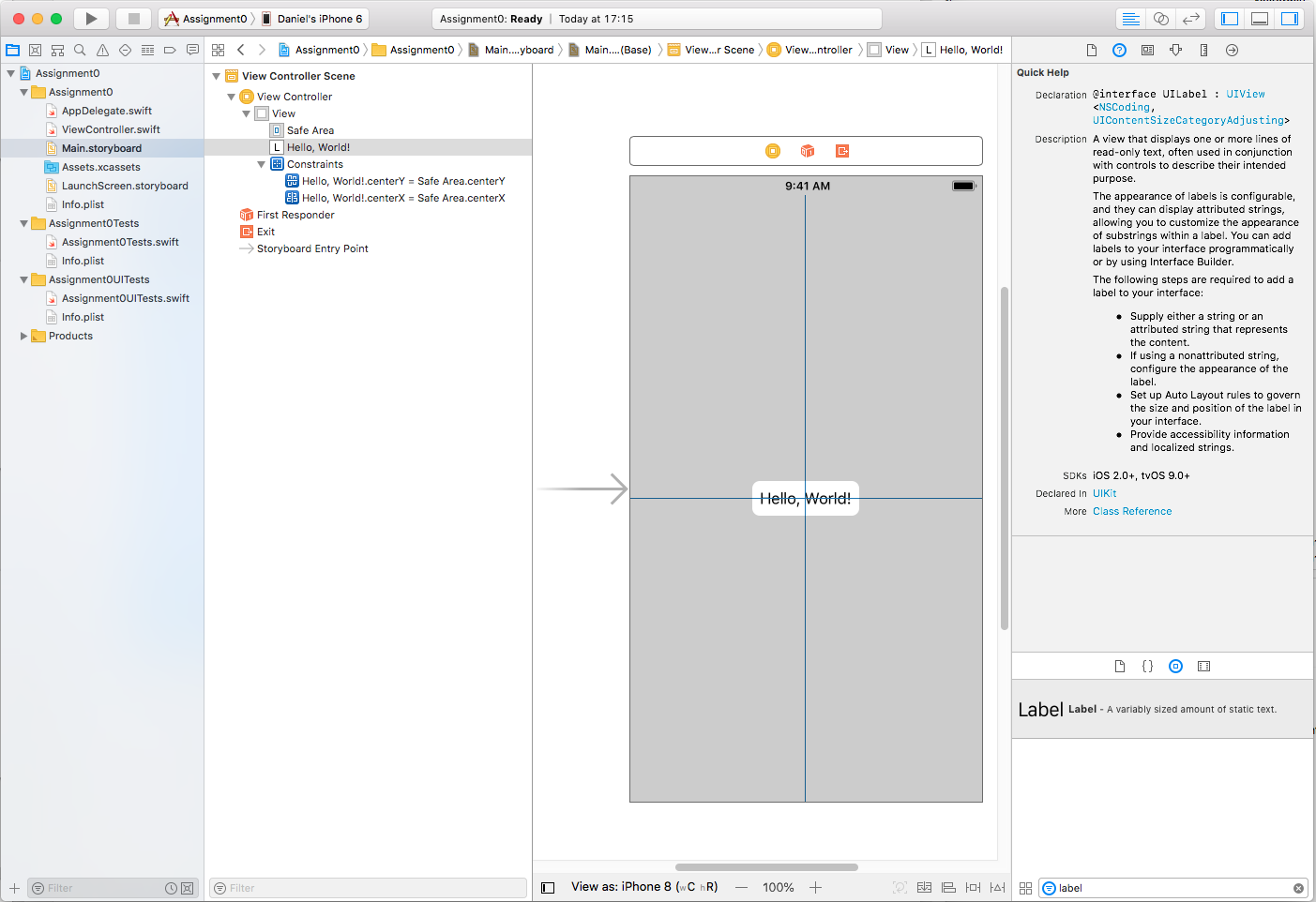
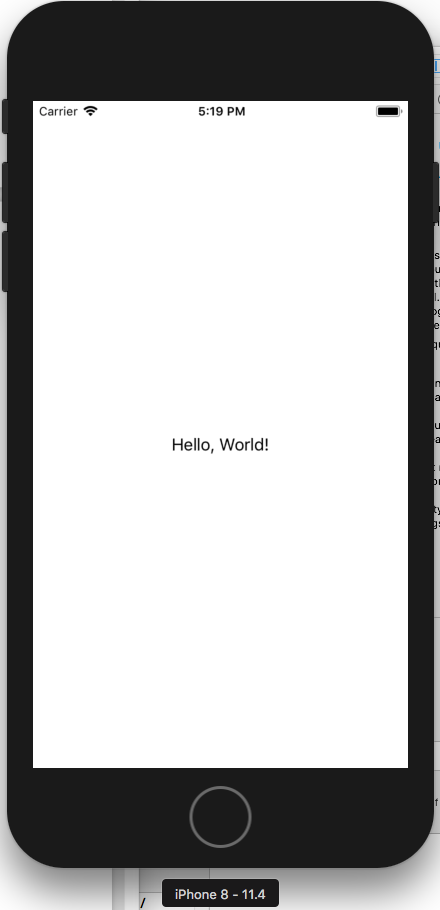
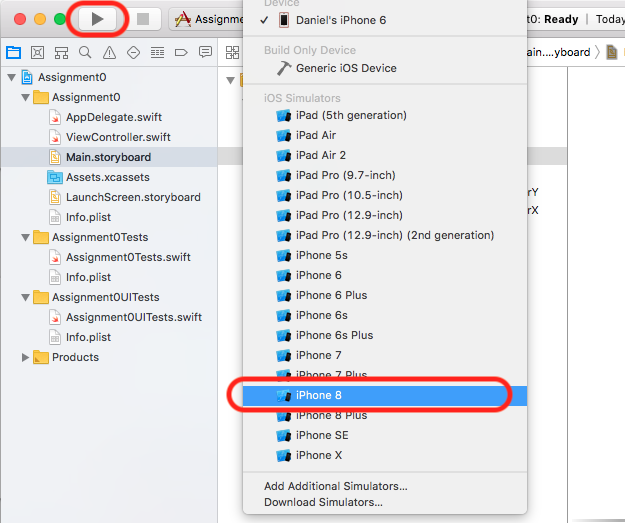
cd ~/Desktop/e65g  
git clone https://github.com/harvard-ios/csci-e65g-2018

Every time you sit down to study these materials, and just before every class, perform a git pull to ensure you’re seeing the latest version. Corrections and clarifications will come up all the time. If you ever manage to corrupt, delete, or accidentally change anything in this repository, it is always safe to delete it entirely (from Finder) and start over with the git clone operation. Just make sure to move any original work to the proper personal assignment working directory first.

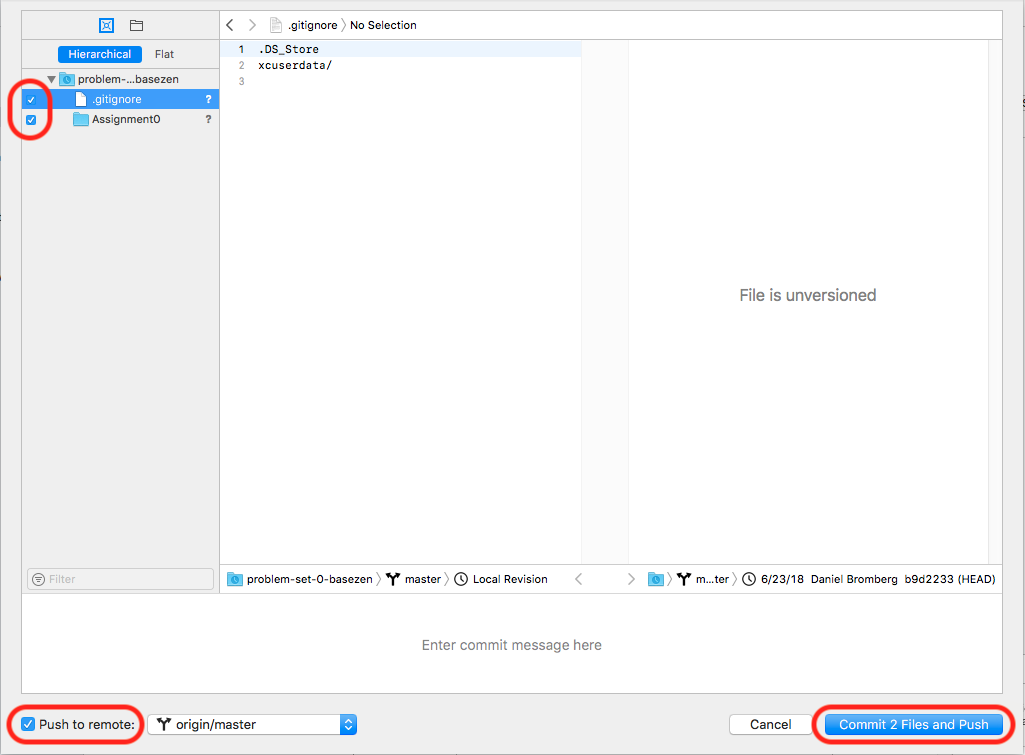
OUTPUT: Text output from clone operation and git status afterward

1. Create a simple Xcode project and push it to the repository.  
   OUTPUT: The project, committed to the repository and pushed.
   * Start a new Swift / iOS project in Xcode and save it in the same assignment-0 directory. Ensure the **Create Git repository** is *unchecked* when first creating the project, for every project in this class. Repository management is done as instructed as above, so any repositories Xcode creates here will conflict and make submission impossible.

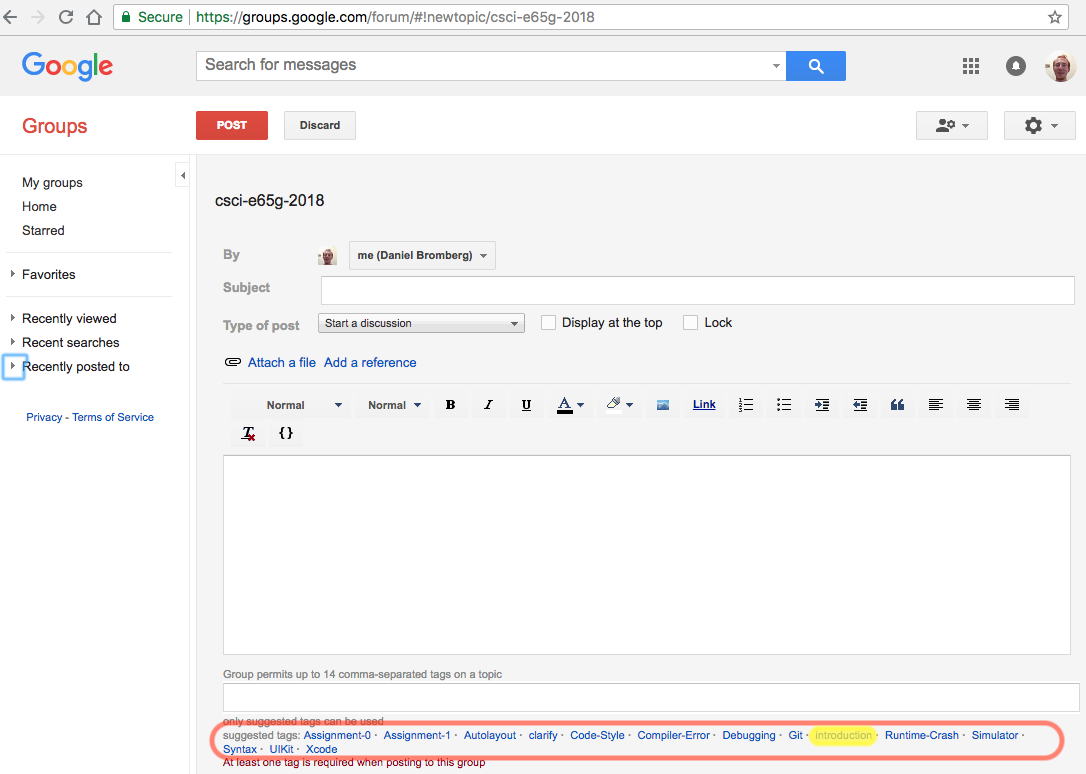
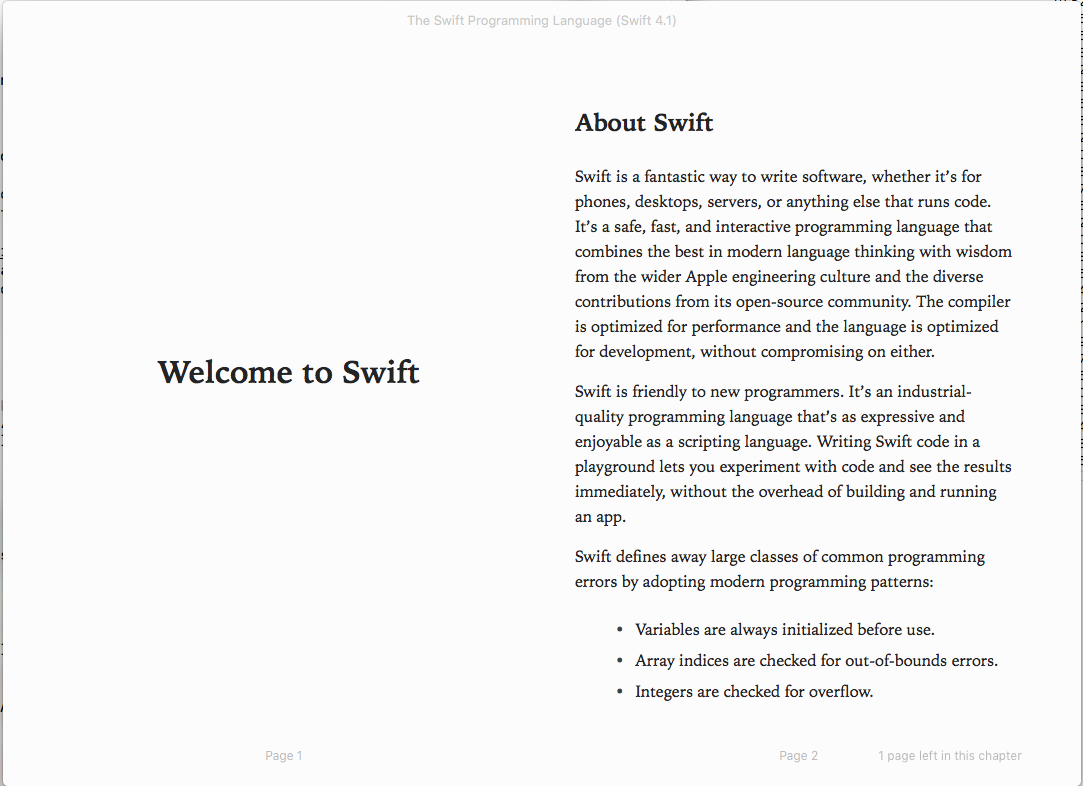


* + Navigate to the storyboard (**Main.storyboard** in the project navigator at left). Lay out (center X and Y) a single UILabel with some text.
  + Run the App in the iPhone 8 simulator.
  + Push the code to the repository, but this time using the Xcode UI rather than git command line tools. The important thing to realize here is that they are equivalent in functionality in the everyday commit-push cycle and you should be equally comfortable in both. Xcode provides some convenience but cannot deal with even minor exceptional conditions. You will always get the complete error feedback and have the necessary power user commands at the command line.)

Choose “Commit...” from the Xcode “Source Control” menu. Select the “Push to remote” box lower left to save a step. Confirm all files are checked at left, and click “Commit *count* Files and Push”.



* + Again return to the GitHub web page and ensure the project is present.

1. Record a short video or audio clip adding some personal details about your software background and academic goals relating to the class. All other students will be able to view this. It should be at least 15 seconds and at most one minute long. A QuickTime webcam capture is fine (a search will lead to several tutorials.) This is particularly important in distance education class to get to know one another even if we are not always physically present. Call it “self\_intro.*extension*” and also commit it to your repository. We will privately publish these videos and distribute a link in the introductions area. Take great care in using YouTube or other video management site as an intermediary; it may become publicly available immediately.
2. Familiarize yourself with the [the Google Discussion Group](https://groups.google.com/forum/" \l "!forum/csci-e65g-2018).
3. Introduce yourself in a **New Topic**. Be sure to add the **Introduction** tag. Attach the video or audio clip from above. Note the keyword tagging interface is very small and at the bottom. Every Topic must have at least one tag. Ideally, two: the scope (Assignment; Lecture; Section) and the technical area.
4. Install the [Swift iBook **The Swift Programming Language (Swift 4.1)**](https://itunes.apple.com/us/book/the-swift-programming-language-swift-4-1/id881256329). This is an excellent (and necessary) time to work out any forgotten Apple ID details. It will open in [iBooks](https://www.apple.com/ibooks/).Please note we are *not* using Xcode 10 or Swift 4.2 yet. They have not been officially released, though the process is very close.
5. Ensure you can begin viewing the WWDC 2014, 2015, and 2016 introductory videos. These are assigned viewing by the end of week 1 but not for this assignment. The WWDC 2017 and WWDC 2018 materials regarding Swift and Xcode are largely outside of the scope of this class, but have many useful advanced topics.
   * [Official short intro video at WWDC 2014](https://www.youtube.com/watch?v=MO7Ta0DvEWA)
   * [Extended breakout session on Swift 1.0](https://www.youtube.com/watch?v=A0C6L4XmrZM)
   * [WWDC 2015](https://developer.apple.com/videos/play/wwdc2015/106/)
   * [WWDC 2016](https://developer.apple.com/videos/play/wwdc2016/402/)
6. Make a statement of commitment, in your own words, to make frequent backups to your preferred backup method for your **working directory (which is your uncommitted** progress) and to **commit and push from local to remote repositories** after small incremental improvements or additions to your work. Include an udnerstanding that there are no extensions or leniency granted for any data disasters from not doing so.

OUTPUT: The written statement in any of the aforementioned text formats.

1. Make a statement, in your own words, that you have read, understood, and agree to the [academic honesty policy](https://harvard-ios.github.io/csci-e65g-2018/admin/syllabus-fall.html#academic-honesty), most especially citing Internet-derived knowledge and code.

OUTPUT: The written statement