Dependency Management with Carthage for iOS

What is Dependency Management?

When working in iOS, there are plenty of amazing third party libraries that we can use to save us time and energy in our mobile app development. Third party libraries can provide useful functions, methods, or templates and can assist in networking, parsing data, interfacing with Core Data, fetching web images, and many other common tasks. These third party libraries are known as dependencies. One way to obtain and keep these libraries up to date is through a dependency manager.

Most dependency managers will download your desired third party libraries, prepare them for your project, and obtain any other required dependencies. This is fantastic! Without the use of dependency managers, you can quickly find yourself tangled in a mess of unusable third party libraries. A good dependency manager will keep your libraries (and any dependencies they may have) up to date and functional.

There are plenty of useful libraries that can be found with a little searching on the internet. Many of these are kept on GitHub. For example, Alamofire is a fantastic library for networking; ObjectMapper helps to transfer model objects to and from JSON; and SugarRecord is great for leveraging Core Data in iOS applications. For our grand adventure into dependency managers, we will be using a library of functions and methods known as SwiftRandom, the dependency manager Carthage, and the Treehouse iOS Xcode project “FunFacts”. After using Carthage to download and prepare the SwiftRandom library, we will be using the random color method from SwiftRandom in place of the current random color method.

Note: Throughout the tutorial, you will be asked to “paste” lines of text from the tutorial for your use. Many times these lines of text are precise and can only be used as they are written in this tutorial. Although not recommended for beginners, you are welcome to manually type the text instead of using the “paste” commands.

What is Carthage?

There are several dependency managers that could be used in our Xcode project like Cocoa Pods, Xcode Maven, or XCode Gradle to name a few. Each dependency management software has its own set of benefits and drawbacks. One of best parts of Carthage is that, from the ground up, it is made to be ruthlessly simple. It was created by developers from GitHub, and was written entirely in Swift. In fact, it was the first dependency manager to officially support

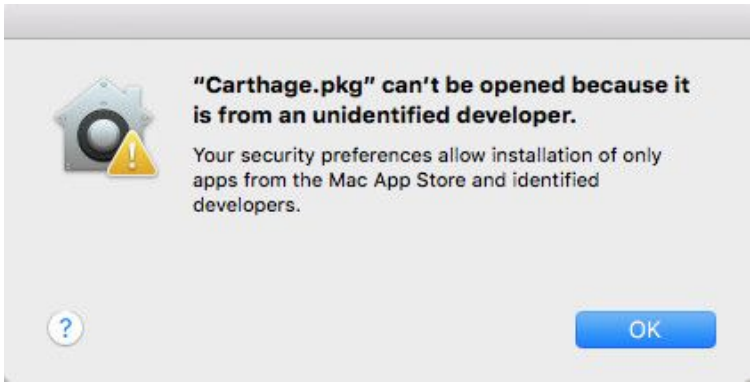
Swift. Once installed, Carthage can be used within your Xcode project in four simple steps, and we will get to those steps, once Carthage has been installed.

Installing Carthage

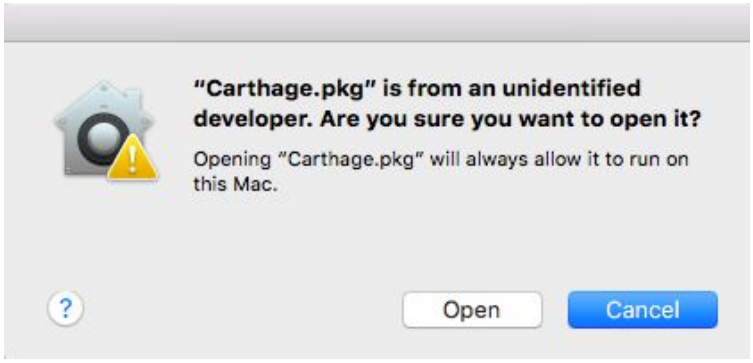
The “Carthage.pkg” installer can be downloaded from Github here. Once the download finishes, perform the following steps:

1. Double click “Carthage.pkg” to begin the installation.

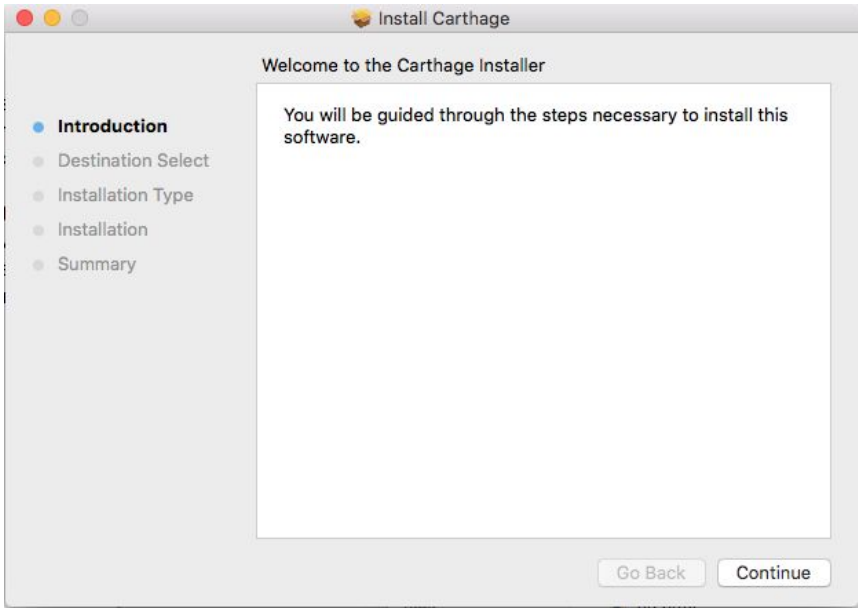
Note: If you received the following error:



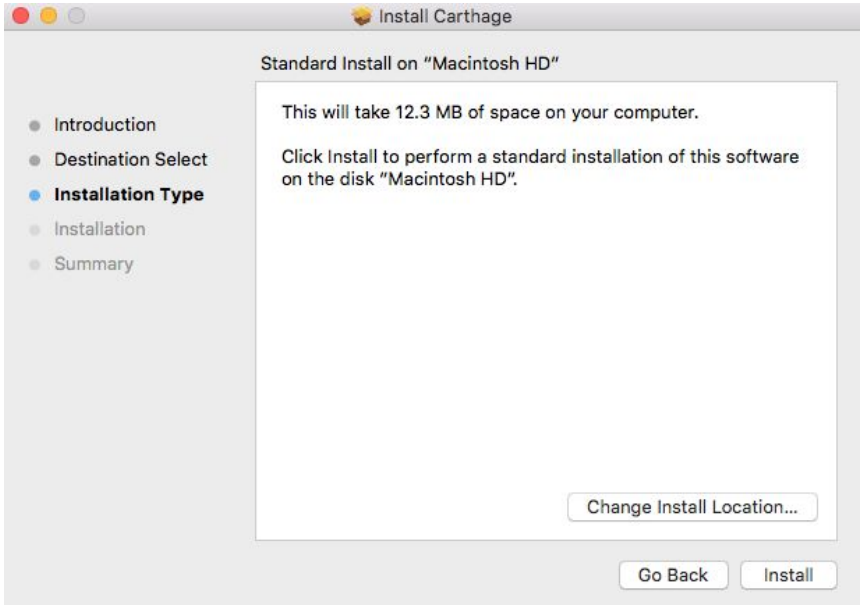
You can use Control-Click to select “Carthage.pkg”. Then choose “Open” from the context menu, and when the following notice appears, select “Open”:



2. Next, you should see the Carthage installer screen. Press Continue

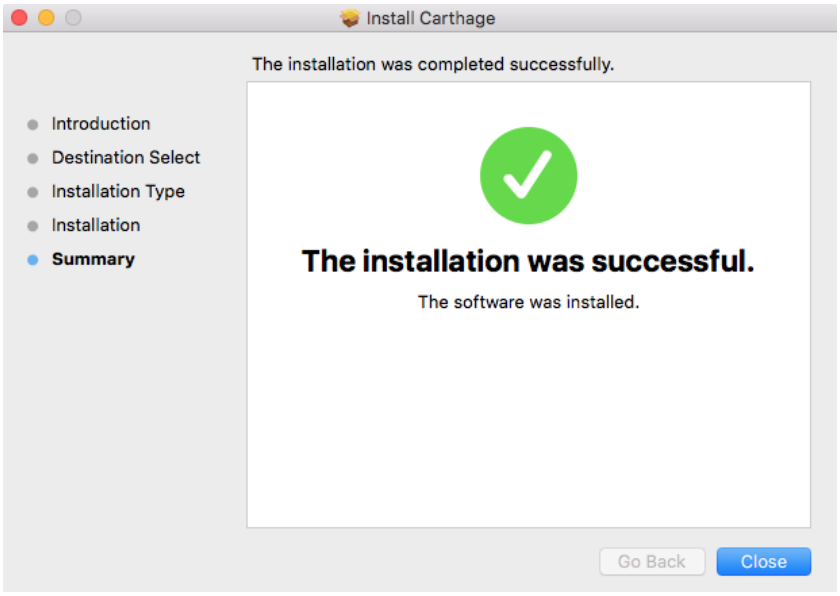


3. Select the destination drive to install Carthage. You will most likely want to install Carthage on your main Hard Drive.
4. Press “Continue” to go on to the next step.
5. Select “Install” to begin installing Carthage.



6. If you are asked to enter your username and password, do so and press “Install Software”.

CONGRATULATIONS! You have successfully installed Carthage!

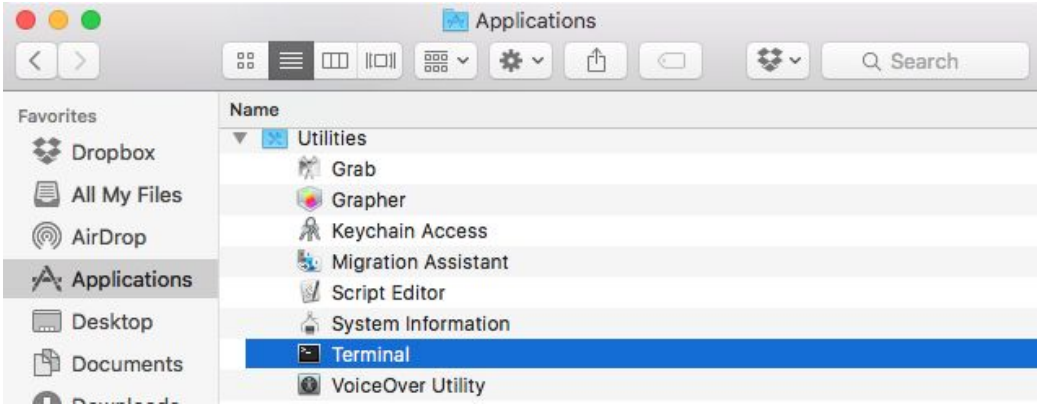


Verify Carthage Installation

Our next step is to verify that you have set-up carthage correctly by using Terminal*. To open Terminal, you will need to

1. Open a Finder window.
2. Select “Applications” under the “Favorites” folders on the left.
3. Select the “Utilities” folder.
4. Double click the “Terminal” icon to begin using Terminal.

*Terminal is a key tool that all developers should be familiar with. In fact, feel free to drag Terminal into your dock because we will be using again. If you would like to know more about Terminal, we have a fantastic course detailing Terminal basics, and if you don’t have time for a workshop, this cheat sheet of useful Terminal commands can also be great resource.



5. In the terminal window, paste

```
carthage version
```

You should see your version of Carthage listed. In the likely event that you see a version number other than “0.16.2”, that is fine and is expected. The basics of how Carthage operates should not change.

If there is no version displayed, take a deep breath, go back, and retry installing Carthage. Assuming that you now have Carthage installed, you can give yourself a pat on the back and know that you no longer have to go through these installation steps the next time that you want to use Carthage. Now we can begin setting up your project’s dependency.

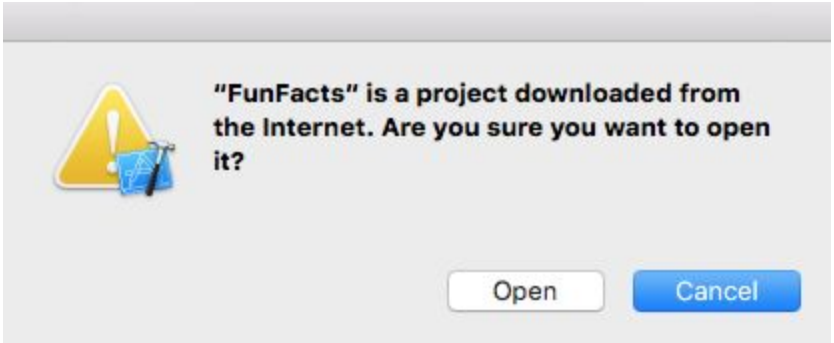
Downloading and Running The Project

1. Download the Fun Facts Project.
2. Unzip the zipped folder “SimpleiPhone-S5”.
3. Open the folder “SimpleiPhone-S5”.
4. Open the folder “FunFacts”.
5. Click “FunFacts.xcodeproj” to open the “FunFacts” Xcode project.

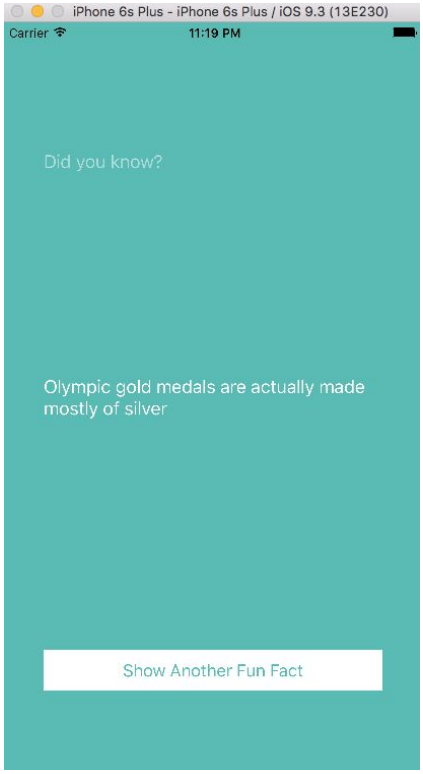
Note: If you see the following warning:

‘FunFacts’ is a project downloaded from the Internet. Are you sure you want to open it?

Go ahead and select “Open” to open the Xcode project.



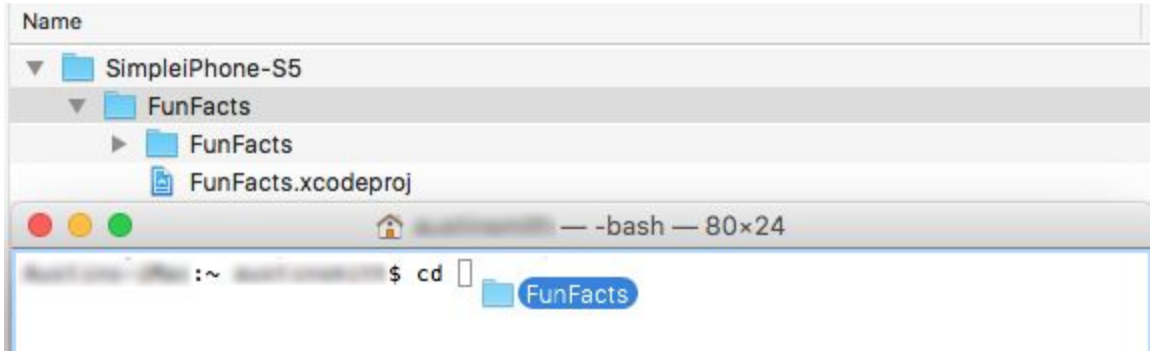
Once, the project is open, build and run the application to make sure that everything is compiling and running smoothly. Feel free to play around in the app to see how the app’s background changes colors and displays new fun facts every time the “Show Another Fun Fact” button is pressed.



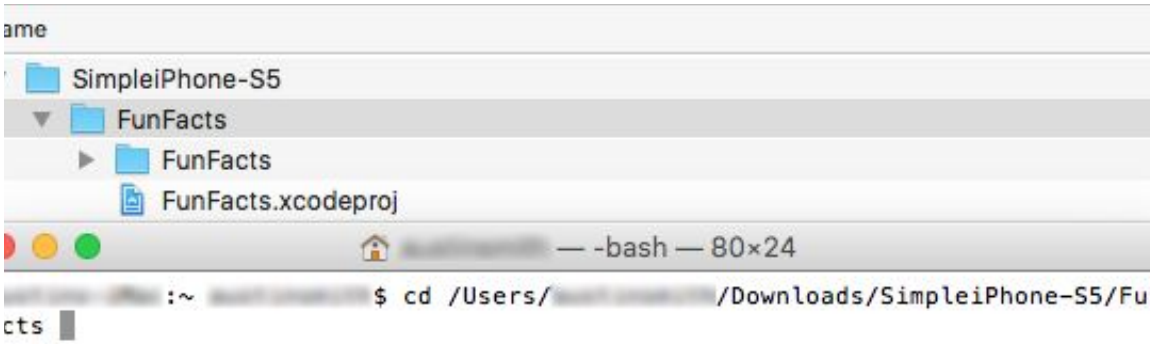
Excellent! Now, we need to install our SwiftRandom library with the help of Carthage! First, we will need to open Terminal again. If you need a refresher on how to open Terminal, feel free to reread the instructions given prior when we were installing Carthage.

Navigating to the Project within Terminal

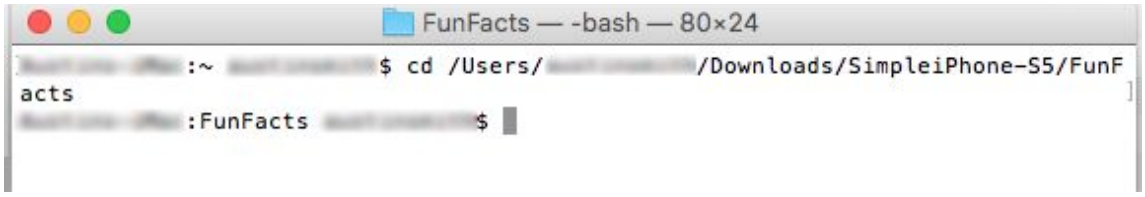
Next, using Terminal, let's go to the root directory of the FunFacts project. The root directory is the folder that contains "FunFacts.xcodeproj". You can navigate to this folder in Terminal by entering the letters "`cd`" **followed by the space character** (don't press "Enter" just yet). Then, drag the folder containing the "FunFacts" project from the Finder window into the terminal window.



Note: Be sure to check that there is, in fact, a space between the letters "cd" and the project directory. Also, make certain that you grabbed the correct "FunFacts" folder. The correct "FunFacts" folder is the one directly under "SimpleiPhone-S5".



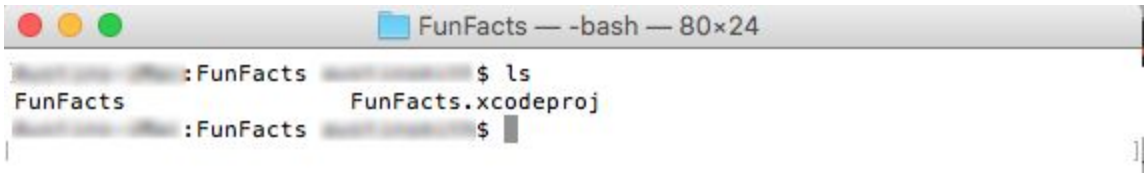
Now, press “Enter”. This command tells Terminal to change directory (cd) into the folder that has your Xcode project. Your terminal should look similar to the picture below.



It is very important that you are in the correct directory. As a final check, enter the following command into Terminal

```
ls
```

Press “Enter”. This command tells the terminal to show the files and folders at Terminal’s current location. You should see “FunFacts.xcodeproj” listed in the window.

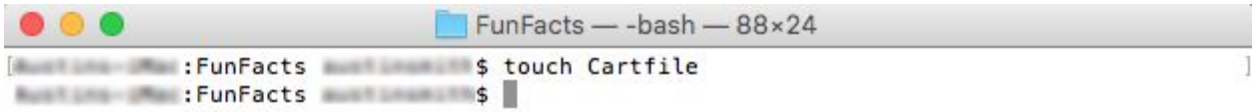


If you do not see “FunFacts.xcodeproj”, go back to the beginning of this section, and try again to navigate to the root directory. Make sure to pay careful attention to the folder you drag into Terminal. If you see “FunFacts.xcodeproj”, great job, you have correctly navigated to the project directory. I know it has taken us a while to get through the installation and setup of Carthage, but now we are ready to actually use Carthage and incorporate libraries into our project using those four steps I mentioned previously.

STEP 1) Create a Cartfile

We will create a Cartfile by pasting the following command into Terminal:

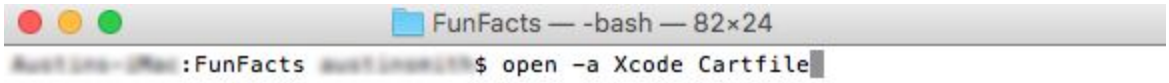
```
touch Cartfile
```



This command creates a file named “Cartfile” in the root directory of our project. The Cartfile will tell Carthage precise instructions for our desired libraries.

Next, we need to open the Cartfile with Xcode so that we can edit our Cartfile. We can open the Cartfile by pasting the following into Terminal:

```
open -a Xcode Cartfile
```

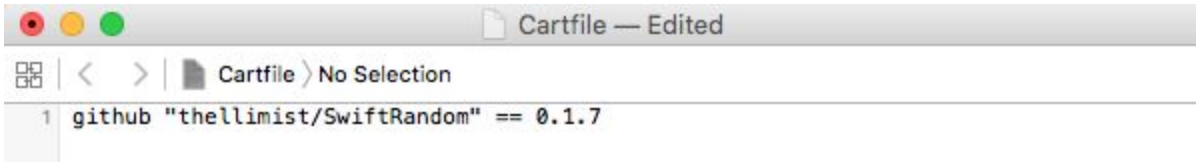


Because the Cartfile is the link between Carthage and our project it is is crucial to ensure that every line is written correctly. When going through this tutorial, it is most important that you understand the following concept about Carthage:

The Cartfile provides the instructions that tell Carthage where to find the desired library, the name of the library, and what version of the library to download.

Paste the following into your Cartfile:

```
github "thellimist/SwiftRandom" == 0.1.7
```



This Cartfile tells Carthage the following:

- **`Github`** instructs Carthage to look on Github for the library.
- **`thellimist/SwiftRandom`** indicates to download the “`SwiftRandom`” library from the user “`thellimist`”.
- **`== 0.1.7`** specifies to download version 0.1.7 of the SwiftRandom library.

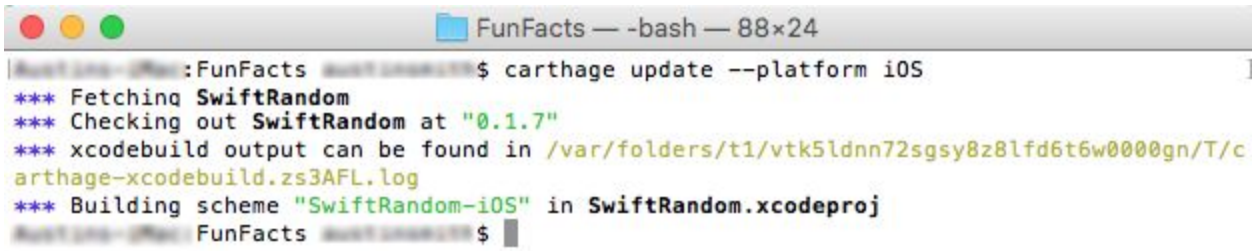
After pasting the text into the Cartfile, make sure to save the changes to the Cartfile.

STEP 2) Run Carthage

Carthage is run simply by pasting the following command into Terminal:

```
carthage update --platform iOS
```

Carthage will now read the Cartfile, retrieve the appropriate library, and build the library into a framework.

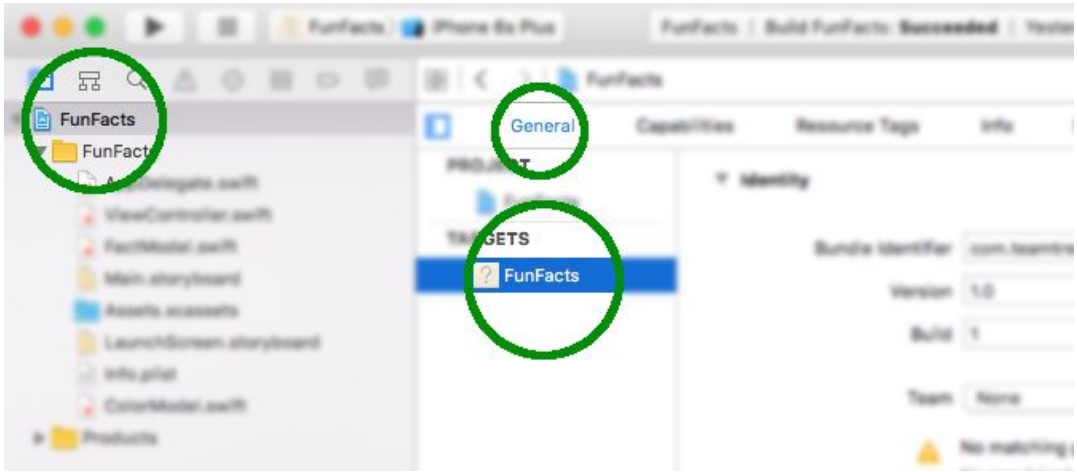


Note: if your terminal looks different from the screenshot, it is okay. Don't panic. Just make sure that the terminal displays "`Building scheme "SwiftRandom-iOS"`". Also, you might receive the "`Bad credentials`" warning. This warning indicates that you are not logged into Github, and the warning can be ignored. The SwiftRandom library will still be downloaded and built for you.

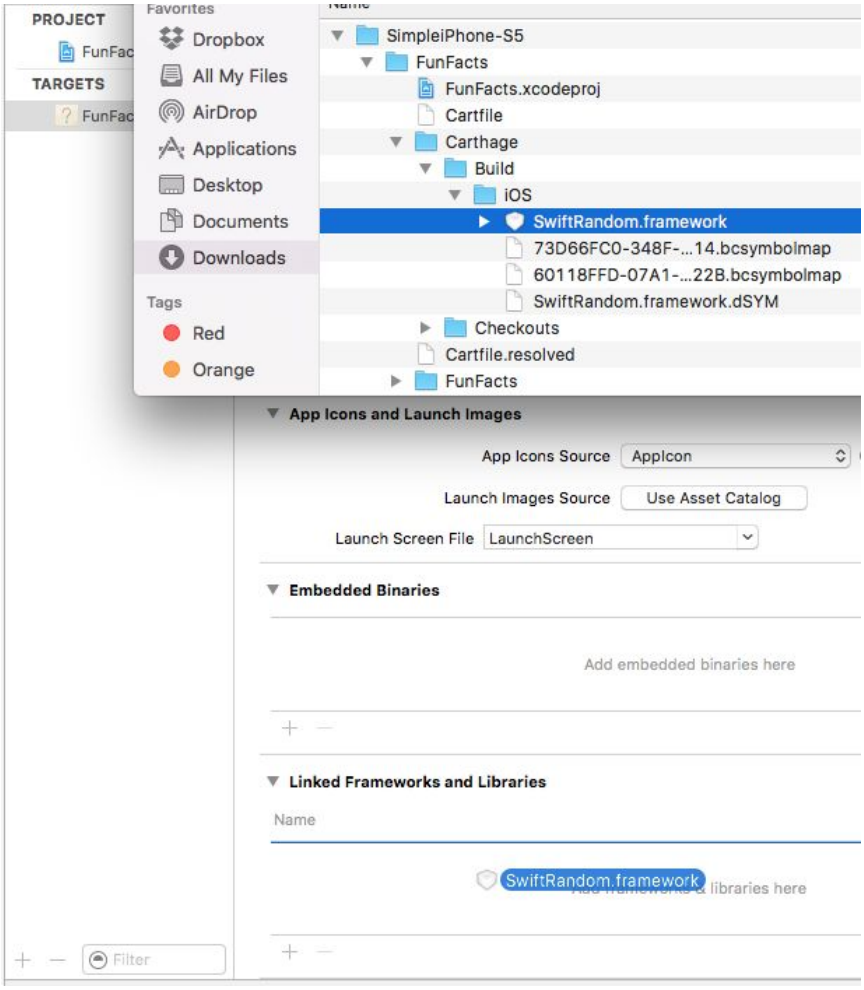
When Carthage has finished updating, you will have the SwiftRandom framework for use in your project!

STEP 3) Link Frameworks

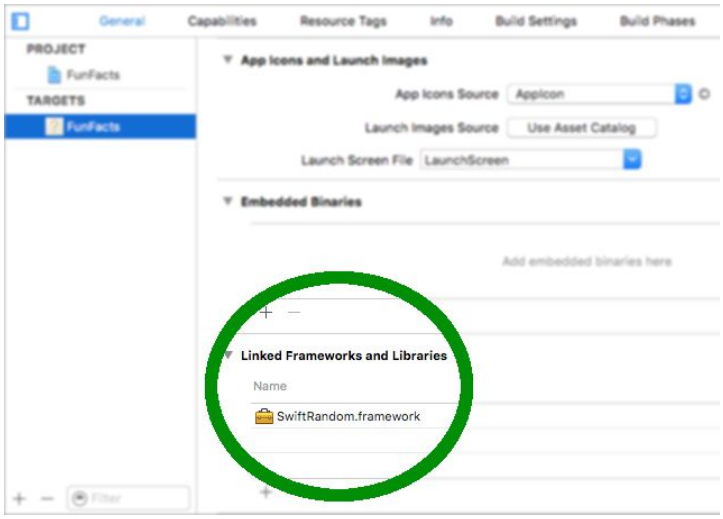
1. If the "FunFacts" project isn't open already, open it.
2. Click on "FunFacts" project that can be found in your project navigator on the left hand side.
3. Select the "FunFacts" Target.
4. Select the tab labeled "General".



5. In the “General” tab, scroll down to the bottom where you will see “Linked Frameworks and Libraries”. With the Xcode project window still available, open a Finder window and navigate to the “FunFacts” project directory. In the project directory, open the following folders: “Carthage/Build/iOS”. In the “iOS” folder, you should see “SwiftRandom.framework”. Drag the framework into the “Linked Frameworks and Libraries” section of the “FunFacts” project.



VIOLA, you have added the frameworks and are finished with part three! Before proceeding, make sure that your “Linked Frameworks and Libraries” section looks like the image below.

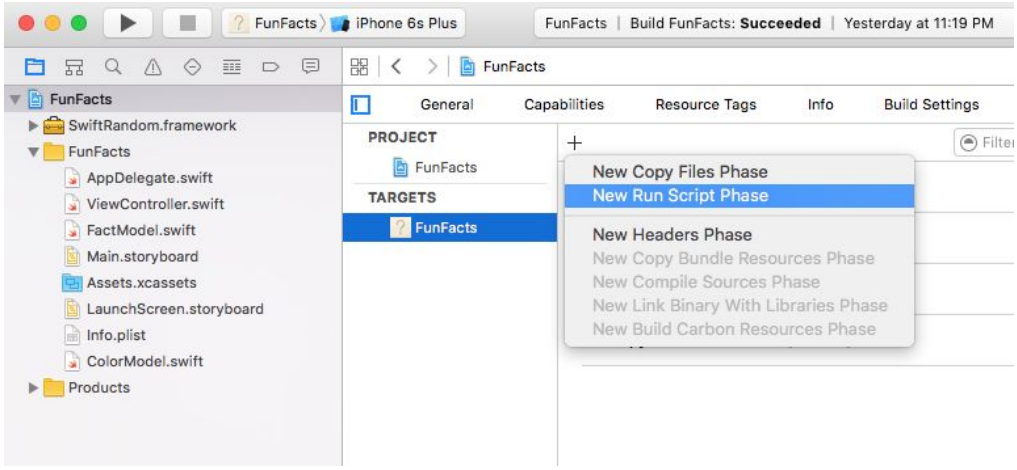


If the framework was not added, try dragging the framework into your project again.

STEP 4) Strip Architectures

You might think you are finished, and you would be if there was not a bug in the iOS app store that blocks universal frameworks from being submitted. In order to strip the architectures,

1. Click on the “Build Phases” tab.
2. Click on the “+” icon directly below the “Capabilities” tab (note: don’t select the “Capabilities” tab)
3. Select “New Run Script Phase”.

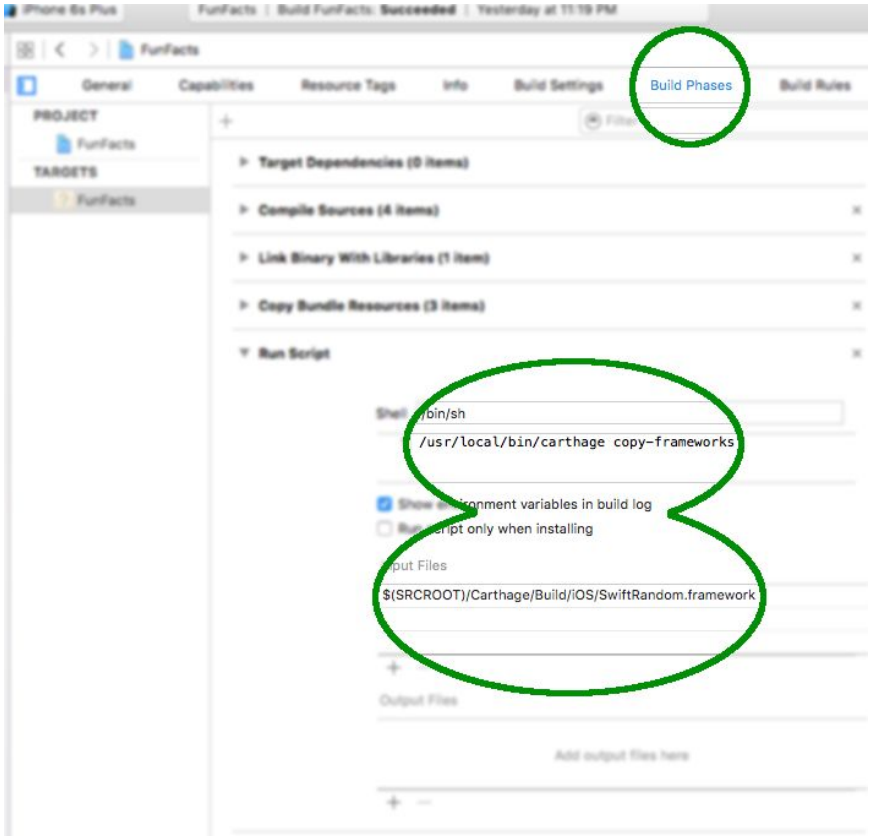


4. Click the enclosure triangle to the the left of “Run Script” in order to expand the window.
5. In the expanded window, find where it says, “`Type a script or drag a script file from your workspace to insert its path`”.
6. Paste the following script into the text box:

```
/usr/local/bin/carthage copy-frameworks
```

7. Click the “+” under “Input Files” (located halfway down the “Run Script” window).
8. Paste the following into the text box,

```
$(SRCROOT)/Carthage/Build/iOS/SwiftRandom.framework
```



With these updates, the SwiftRandom library should be ready for use. Build and run the application again so as to make sure everything is still working. If it runs, you have followed all four steps successfully, and have added a library to your project with Carthage. Congrats! Now, it’s time to use our new library!

Using SwiftRandom

Next using Xcode, go into “ViewController.swift”, and at the top of the file, directly under “`import UIKit`”, let’s import our SwiftRandom library by typing

```
Import SwiftRandom
```

Next, let’s replace the current getRandomColor() method that was defined previously with the new randomColor() method that is given to us by the SwiftRandom library. We can do this by commenting out the following line:

```
let randomColor = ColorModel().getRandomColor()
```

and adding this new line below:

```
let randomColor = Randoms.randomColor()
```

```
import UIKit
import SwiftRandom

class ViewController: UIViewController {

    @IBOutlet weak var funFactLabel: UILabel!

    @IBOutlet weak var funFactButton: UIButton!

    let factModel = FactModel()

    override func viewDidLoad() {
        super.viewDidLoad()
        funFactLabel.text = factModel.getRandomFact()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


    @IBAction func showFunFact() {
        //let randomColor = ColorModel().getRandomColor()
        let randomColor = Randoms.randomColor()
        view.backgroundColor = randomColor
        funFactButton.tintColor = randomColor
        funFactLabel.text = factModel.getRandomFact()
    }

}
```

Run and build the FunFacts app again, and you should see that the random colors for the app background are now delivered by the SwiftRandom library and not the ColorModel structure.

Note: In a real project, we would want to delete “ColorModel.swift” and remove the commented line of code, but since this is a tutorial, feel free to leave them both in the project to serve as a reminder of this tutorial.

Updating Your Libraries

In the future, you may find yourself wanting to update to a later version of the SwiftRandom library. To do so, you will go back to the Cartfile, open it, and either change the requested version number from 0.1.7 to your desired version, or you can replace the library’s version identifier from “ == ” to “ >= ”. For example, you can replace the following line in the Cartfile

```
github "thellimist/SwiftRandom" == 0.1.7
```

with

```
github "thellimist/SwiftRandom" >= 0.1.7
```

to indicate to Carthage that you are requesting **at least** version “0.1.7” of SwiftRandom. After you have saved your Cartfile, you will update your library by pasting the following update command into Terminal

```
carthage update --platform iOS
```

and pressing “Enter”. This command was the same command that we used to call Carthage the first time in this tutorial. Now, you will be ready with the newest version of SwiftRandom. It is that easy! For even more information on the Cartfile visit the Cartfile GitHub page.

Great job! Not only do you know how to add and update a library within your project, you also know that if you ever need to randomly generate a color, this library can be used to save yourself time and frustration! Also, feel free to look at the SwiftRandom GitHub Repository to find other random methods in the library. There are plenty of other fun methods like the random first name generator or the random conversation generator that you can use to fill those awkward online silences.

More Information

For even more awesome information on Carthage you can watch an in depth talk on what Carthage is, how to use it, and how it works. You can also do other tutorials to hone your craft or read more about Carthage on GitHub.