Swift 3 Migration Guide

We are inching closer to an official Swift 3 announcement, and we want you to be prepared for the transition! Thankfully, converting your existing Swift 2 projects into Swift 3 is easier than you might think. We've provided this abbreviated guide, but if you're interested in diving deeper, then check out Apple's in-depth migration guide on Swift.org.

Summary of Swift 3 Changes

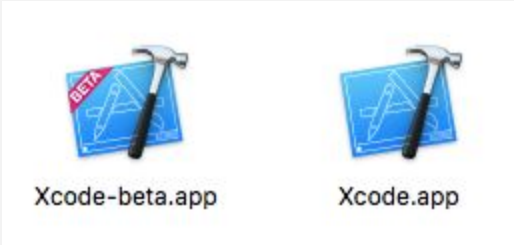
Before we start converting projects in Xcode, it is still a great idea to familiarize yourself with the changes present in Swift 3. Also, for the interested developer, it can be even more useful to study the proposals and rationale behind these changes which are available in Apple's Swift-Evolution repository. Here, we've curated a list of some of the largest changes we've seen in Swift 3:

- Swift-ify Objective-C APIs (SE-0005, SE-0006): The goal of these proposals is to make the standard library and other imported Objective-C feel more "Swifty". It includes guidelines like lowercasing enum cases which is something that came up often when converting our projects.
- A New Model for Collections and Indices (SE-0065): The Collection indexing model has changed dramatically in Swift 3. The most visible change is that indexes (`Index`) no longer have `successor()`, `predecessor()`, `advancedBy(_:)`, `advancedBy(_:limit:)`, or `distanceTo(_:)` methods. Instead this functionality now is part of the Collection itself. See the proposal's "Impact on Existing Code" section:
  - ```
    // Before:
    var i = c.index { $0 % 2 == 0 }
    let j = i.successor()
    print(c[j])
    ```
  - ```
    // After:
    var i = c.index { $0 % 2 == 0 } // No change in algorithm API.
    let j = c.index(after: i) // Advancing an index requires a collection instance.
    print(c[j]) // No change in subscripting.
    ```
- A new version of Foundation, tailored for Swift. On Swift 3.0, when you `import Foundation`, you get a custom-made version of the Foundation framework that feels a lot more "swiftier". Find out more about it in this wwdc session.
- Dropping the 'NS' Prefix (SE-0086): Remember the 'NS' prefix used by Foundation types? **Certain types are dropping the 'NS' prefix, but not every type**. On the proposal, see the "Detailed design" section for a listing of all types that are changing. The main differentiator for the names changing has to do with transitioning from reference types (classes) to value types (structs) in Swift. Sometimes this makes sense, but in other cases it does not. For example, if an existing 'NS' type is (1) tightly coupled with a delegate or if (2) it is important that the identity of the type can be determined based on a reference, then the 'NS' prefix will remain. Here's a summary of other reasons why 'NS' will be kept for certain classes:
  - Classes tightly coupled to Objective-C or specific platform will keep 'NS'.
  - Classes that already have a value-type equivalent will keep 'NS'. For example, `NSArray` already has `Array`, so `NSArray` is around to stay.

- Classes that will have a value-type in the near future will keep 'NS'.

Step 1: Setup Xcode 8

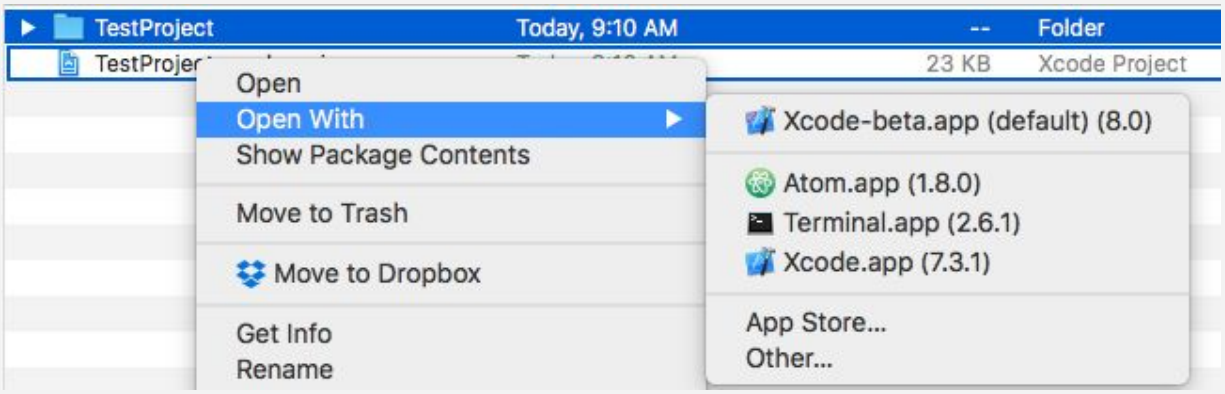
Before you can use Swift 3, you'll need the latest release of Xcode 8. Currently, that is Xcode 8 beta 3, but we can expect a stable release shortly after the announcement. All the downloads for Xcode are available on Apple's developer download page. Note, if you are worried about having multiple versions of Xcode installed on your machine, don't be afraid! Apple makes it easy by managing both versions as their own applications.



Different versions of Xcode are treated as their own separate applications.

There are just a few things you need to remember:

- When opening an Xcode project, you can use Finder to right-click and select the appropriate Xcode for your project.

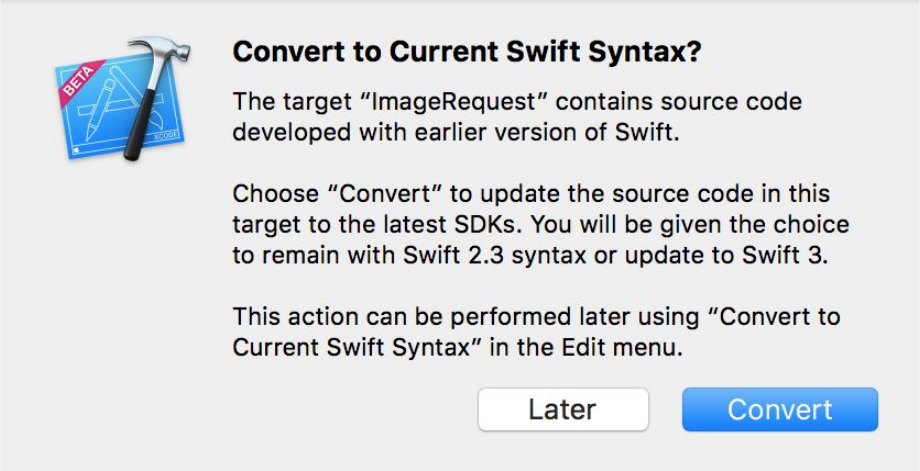


From Finder you can right-click and open an Xcode project using the appropriate version of Xcode for your project.

- After downloading a beta version of Xcode, it may automatically become the default application used to open Xcode projects.
- You can always remove a version of Xcode by going to your Applications and deleting it.

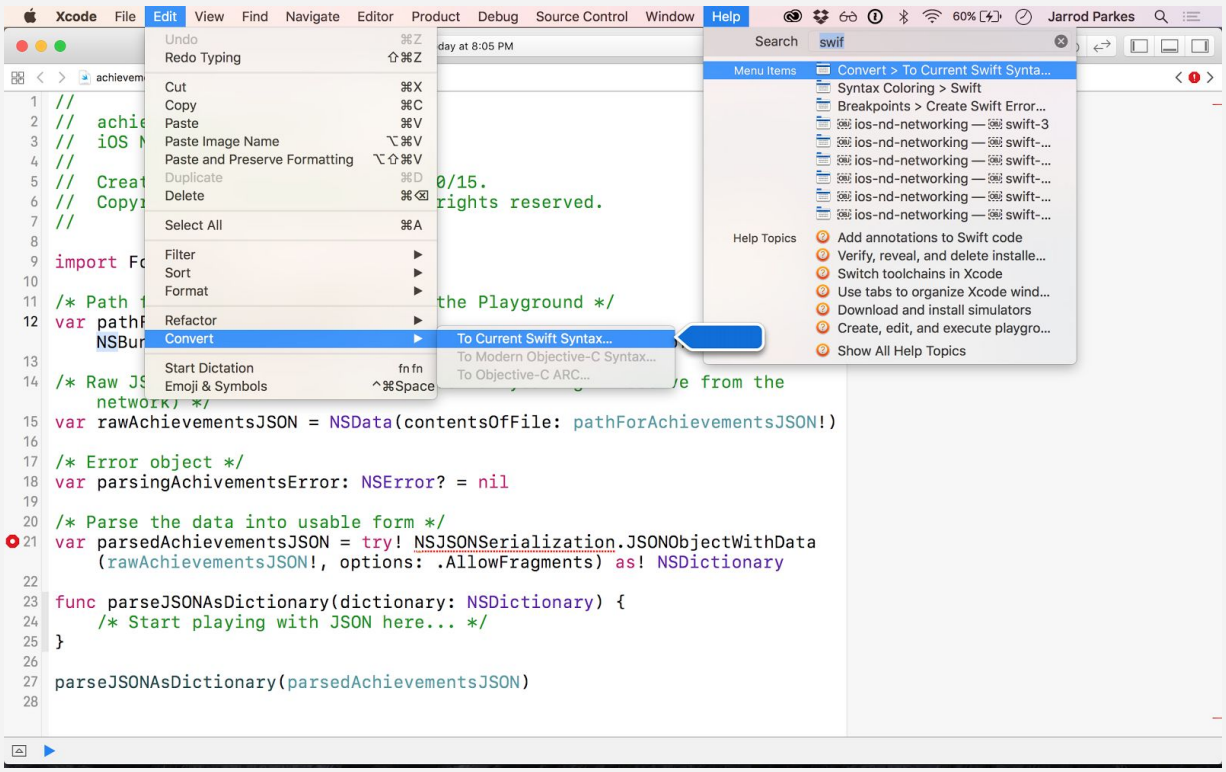
Step 2: Use the Conversion Wizard

Once you're up and running with Xcode 8, you're ready to open an existing project and migrate it to Swift 3. If Xcode detects that your project is using an older version of Swift, then you'll be automatically presented with the Swift conversion wizard:



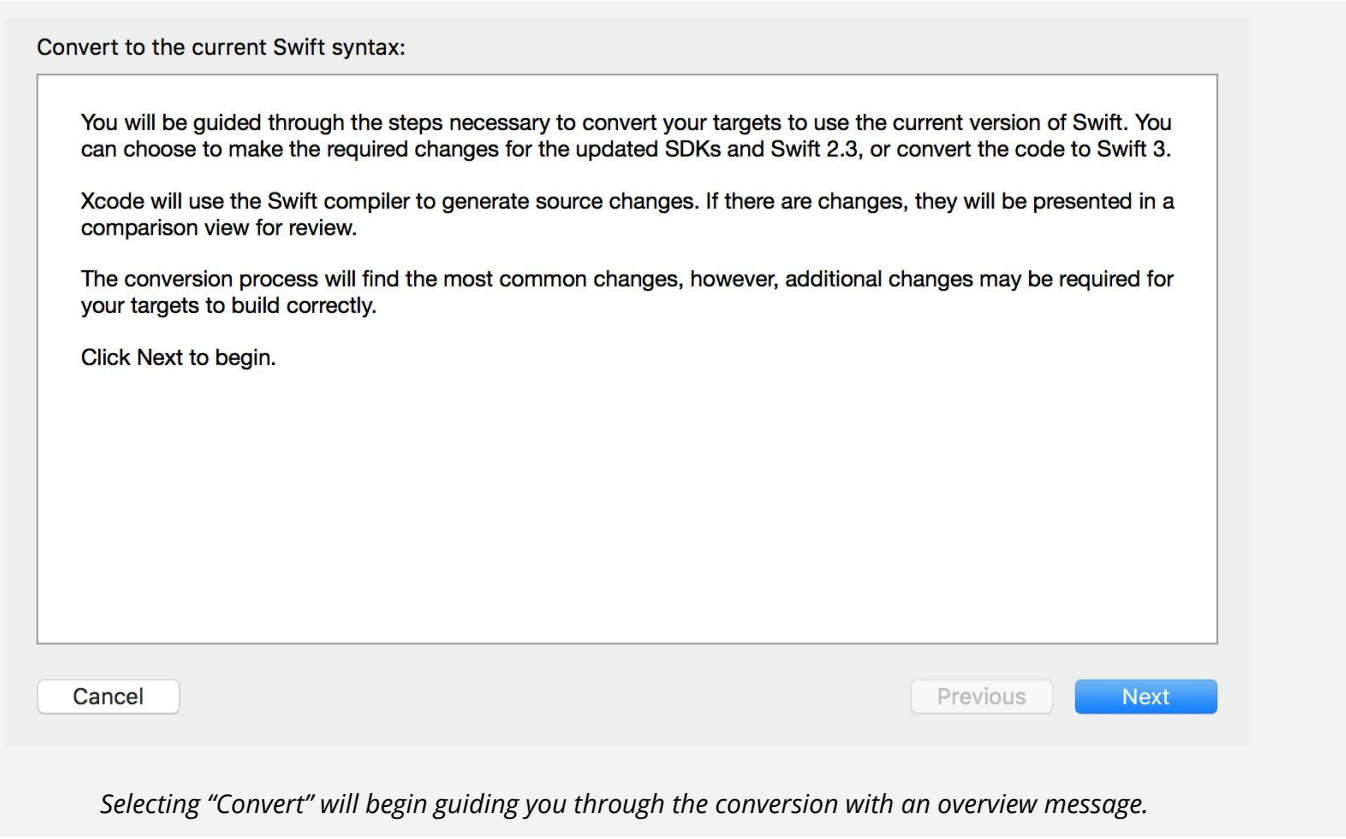
If Xcode 8 detects that your project uses an older version of Swift, then you will be automatically prompted to convert.

In other cases, you may have to manually activate the wizard using Edit → Convert → To Current Swift Syntax...



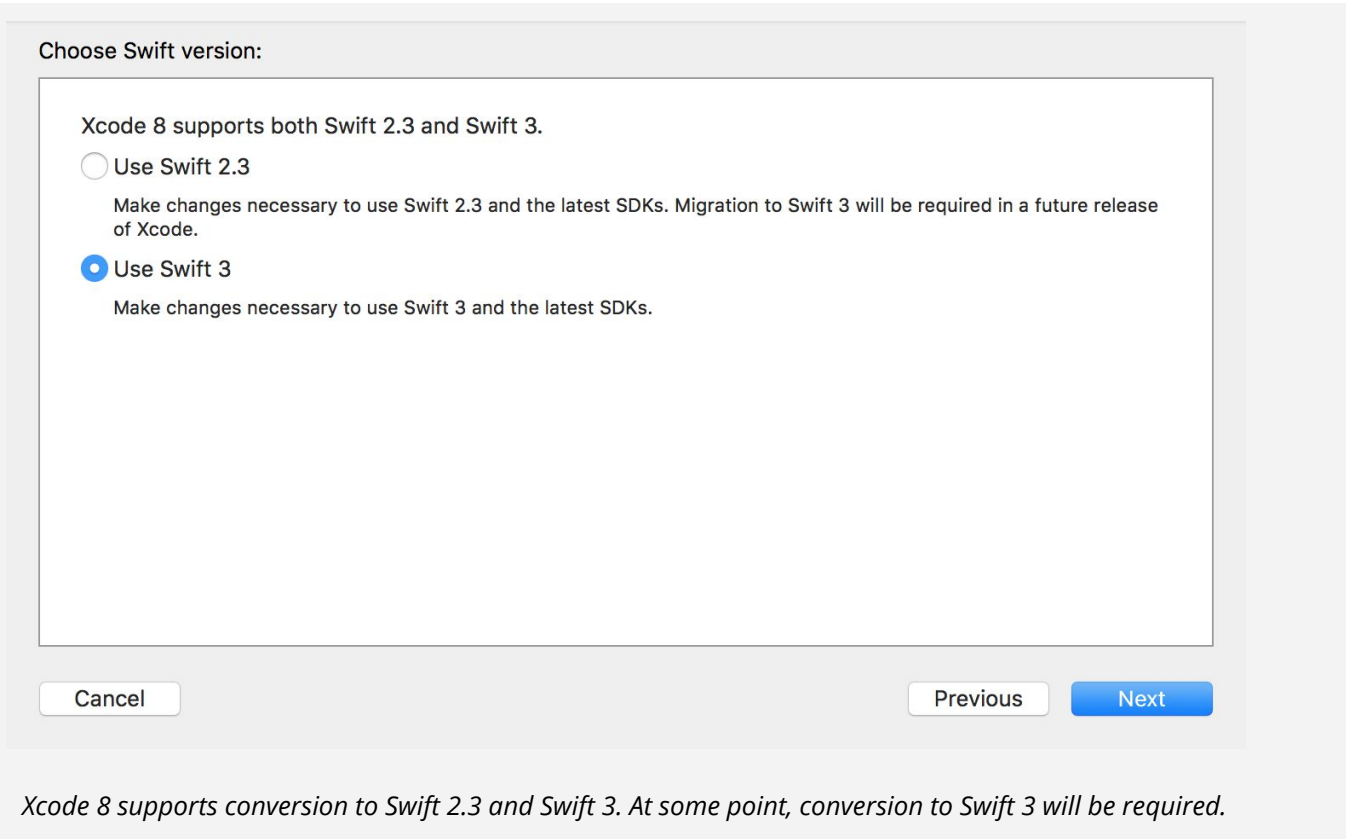
Use "Edit → Convert → To Current Swift Syntax..." to launch the Swift conversion wizard.

Selecting "Convert" will begin guiding you through the conversion with an overview message:



Selecting "Convert" will begin guiding you through the conversion with an overview message.

And, the next screen asks whether you'd like to convert to Swift 2.3 or Swift 3.

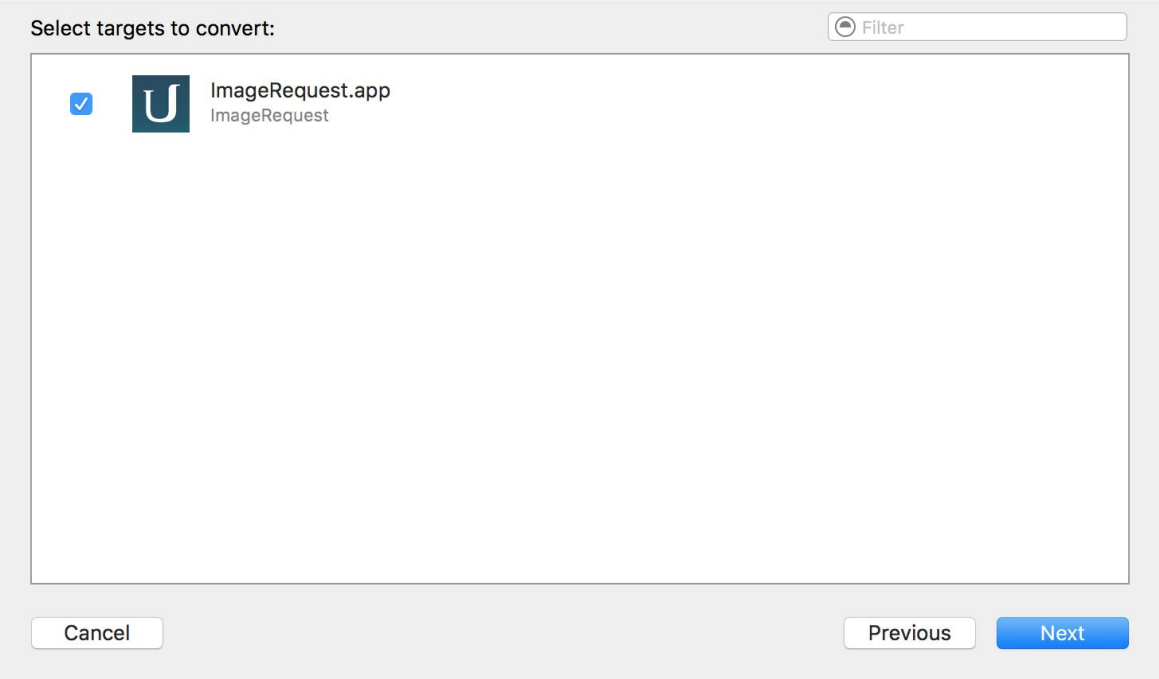


Xcode 8 supports conversion to Swift 2.3 and Swift 3. At some point, conversion to Swift 3 will be required.

We'll want to choose Swift 3; however, it is important to know about the Swift 2.3 option — here is the scoop from Apple's developer blog:

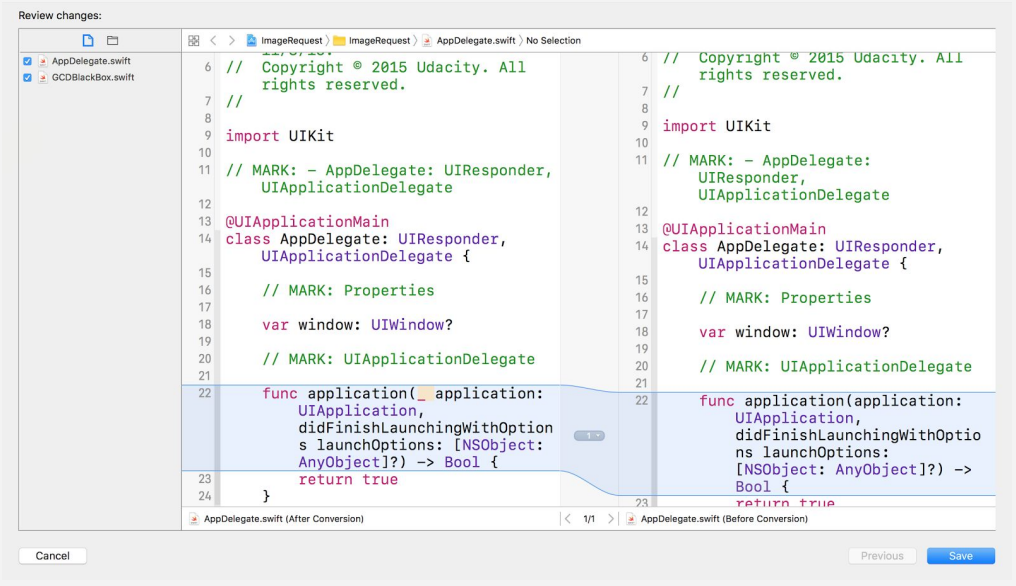
In addition to Swift 3, Xcode 8 supports development with Swift 2.3, a minor update to the Swift 2.2 language built to work with the new SDKs for macOS Sierra, iOS 10, tvOS 10, and watchOS 3. This is intended to allow developers to immediately move to these latest SDKs, even for projects that may be late in development with Swift 2.2 and not yet ready to move to Swift 3.

After selecting Swift 3, you'll be asked to select the targets you want to convert. This is a handy option if you have multiple targets, but for our simple example I'm selecting a target called "ImageRequest.app".



After selecting Swift 2.3 or Swift 3, you are asked to select which targets you'd like to convert.

Clicking next transitions you to overview screen where you can review the line-by-line changes to your project. Take some time to glance and see the changes.

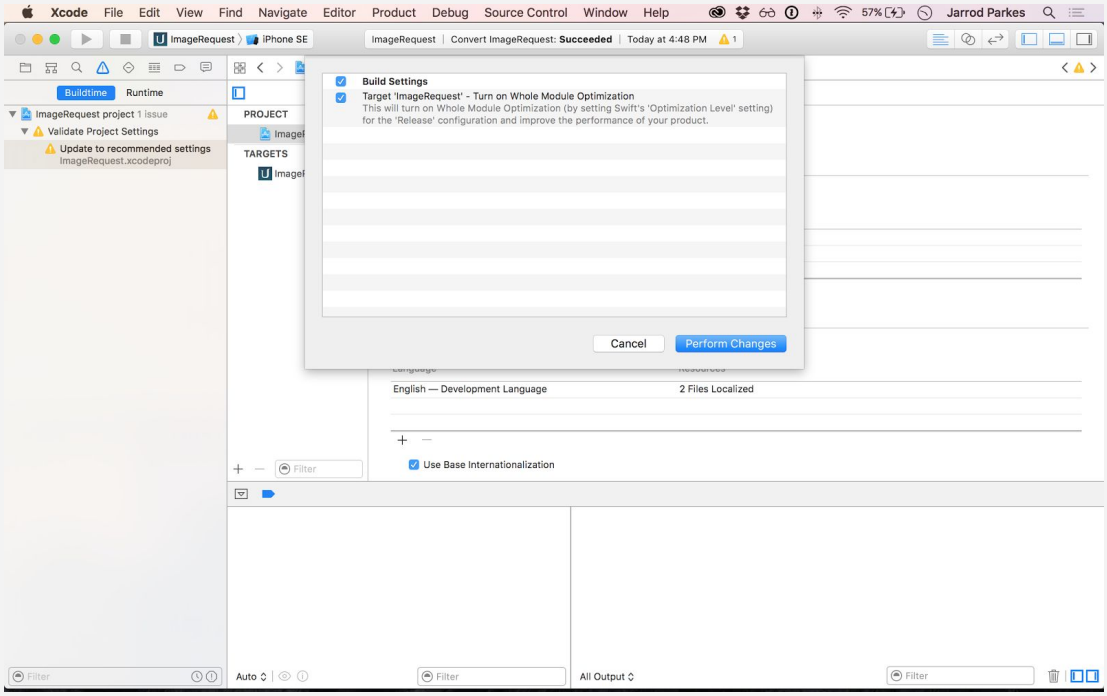


The Swift conversion wizard gives you the ability to review the line-by-line changes that will occur during the conversion process.

In the image above, we can see one such change which is more consistent label behavior for all function parameters. If you are interested, you can read more about how this particular change was integrated into Swift on GitHub.

If you like what you see, then go ahead and "Save" to perform the changes. During this process, Xcode will go line-by-line and make the changes that were previewed in the wizard.

After saving, you may still see a warning that recommends some optimization settings. For most cases, you are fine to leave these as the default.



After saving changes with the wizard, you may see a warning about optimization settings. In most cases, you can simply accept the recommend settings.

At this point, you should build and run the project before continuing forward. If you still get any warnings or errors, then you'll need to manually fix these before continuing.

Step 3: Fix Any Remaining Errors or Warnings

The conversion wizard isn't always perfect, so occasionally you'll have to tweak some things. Here is a short list of errors and warnings we still had to fix by hand:

- **Validate Project Settings — Update to recommend settings**: As mentioned above, this a quick fix. Just accept the recommended optimization changes.
- **Removal of the `where` clause**: If you have if-let statements that use a `where` clause, then `where` can be replaced with a comma (,).

- **Lowercase enumeration cases**: Sometimes the wizard misses this, but to follow Apple's guidelines, you should lowercase all enumeration cases.

There are certainly others we've missed on this list. If you encounter a new one, then let us know at ios-support@udacity.com and we'll get it added!