

Jim Rea

www.provue.com • @provuejim



Jim has been a Mac developer since 1984, is the founder of ProVUE Development, and is the author of OverVUE, Panorama, SiteWarrior, Surf Scout, Power Team. and now Panorama X.

Swiftly Scripting the Command Line

Agenda

- The Current Automation Landscape
- A Recent History
- Why Swift?
- Examples
- Getting Started







PyObjC

Objective-C

Python

Cocoa

Python Modules

macOS



PyObjC

-- OBJECTIVE-C --

```
result = [someObject someMethod:firstArg  
withFoo:foo andBar:bar];
```

-- PYTHON --

```
result =  
someObject.someMethod_withFoo_andBar_(firs  
tArg, foo, bar)
```



PyObjC



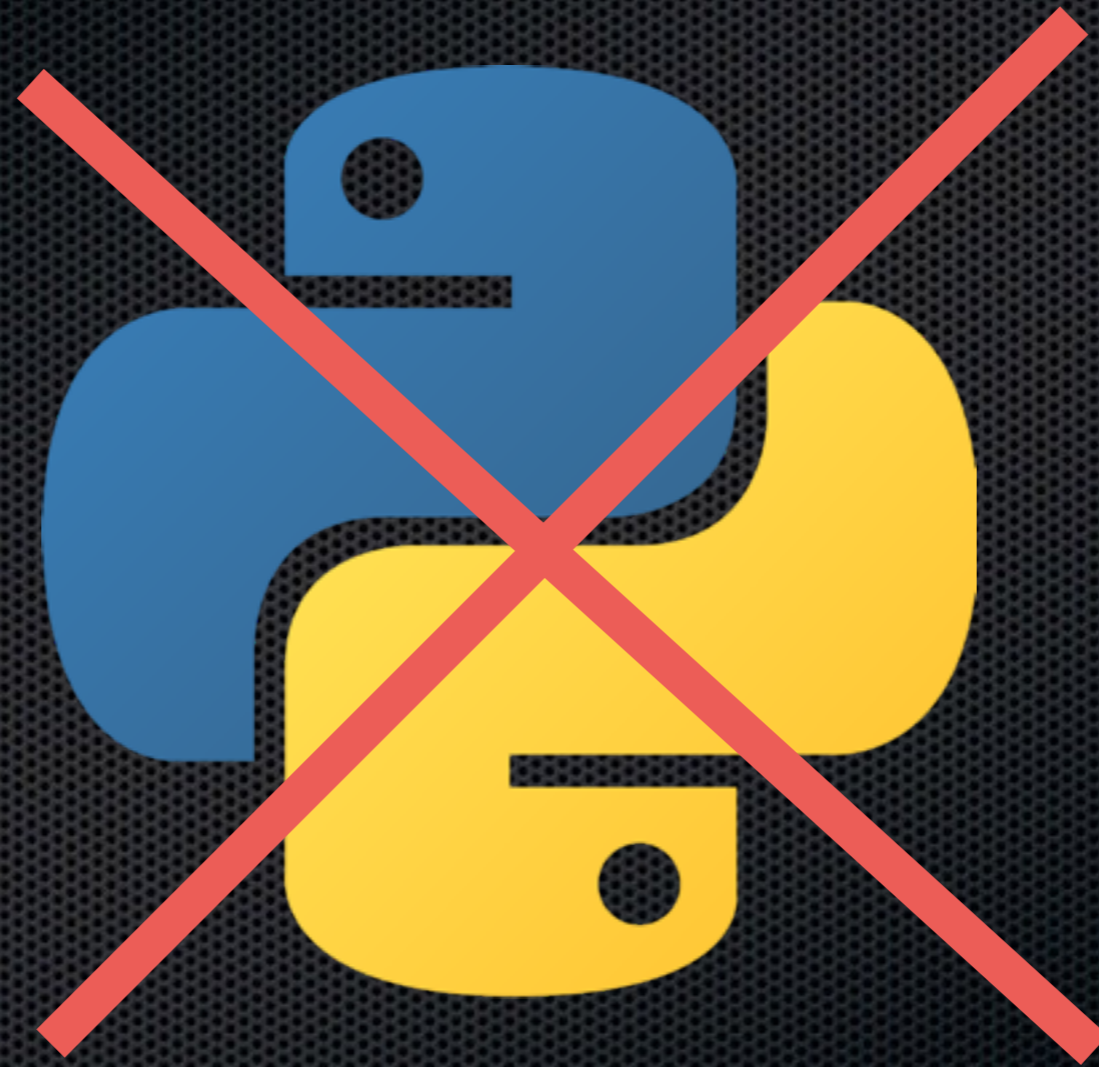
AutoPkgr

make-profile-pkg



**No Maintenance for
Python 2.7 past
January 1, 2020**
python.org









Swift Applications



- macOS apps
- iOS/watchOS/tvOS apps
- Libraries & Frameworks
- Server side web apps
- Scripting/automation



A Brief Swift History

- Modern successor to Objective-C
- Derives ideas from
 - Objective-C
 - Rust
 - Haskell
 - Ruby
 - Python
 - C#
 - CLU
 - and more...



A Brief Swift History

- July 2010 by Chris Lattner at Apple





A Brief Swift History

- <https://www.swiftcommunitypodcast.org>





A Swift History

- 2010 ○ Development starts by Chris Lattner (Apple)
- 2014 ○ Announced at WWDC
- Reached 1.0 with Xcode 6.0
- 2015 ○ Open Sourced swift.org
- 1.2
- 2.0 with Xcode 7.0 (El Capitan)
- 2016 ○ 3.0 with Xcode 8.0 (Sierra)



A Swift History

2016	○	3.0 with Xcode 8.0 (Sierra)
2017	○	4.0 with Xcode 9.0 (High Sierra)
2018	○	4.2 with Xcode 10.0 (Mojave)
2019	○	5.0 🙏



A Swift History

2014



1.0

(1.0 != 4.2)

2018



4.2

2019



5.0





A Swift History

2014 1.0

2018 4.2

2019 5.0 🙏

ABI Stability

Why Swift?





Apple's Goals with Swift

- Fast
- Modern
- Safe
- Interactive



Apple's Goals with Swift

- Fast
- Modern
- Safe
- Interactive



Interpreted vs. Compiled



- Bash
- Python
- JavaScript
- AppleScript
- C
- C++
- Objective-C
- Swift



Apple's Goals with Swift

- Fast
- Modern
- Safe
- Interactive



Modern Language Features

Objective-C

```
NSString * str = @"hello, ";
```

Swift

```
var str = "hello, "
```



Modern Language Features

Objective-C

```
NSString * str = @"hello, ";
```

Swift

```
var str = "hello, "
```



Modern Language Features

Objective-C

```
NSString * str = @"hello, ";
```

Swift

```
var str = "hello, "
```



Modern Language Features

Objective-C

```
NSString * str = @"hello, ";  
str = [str stringByAppendingString:@"world"]
```

Swift

```
var str = "hello, "  
str += " world"
```



Modern Language Features

Objective-C

```
NSString * str = @"hello, ";  
str = [str stringByAppendingString:@"world"]
```

Swift

```
var str = "hello, "  
str += " world"
```

Custom operator

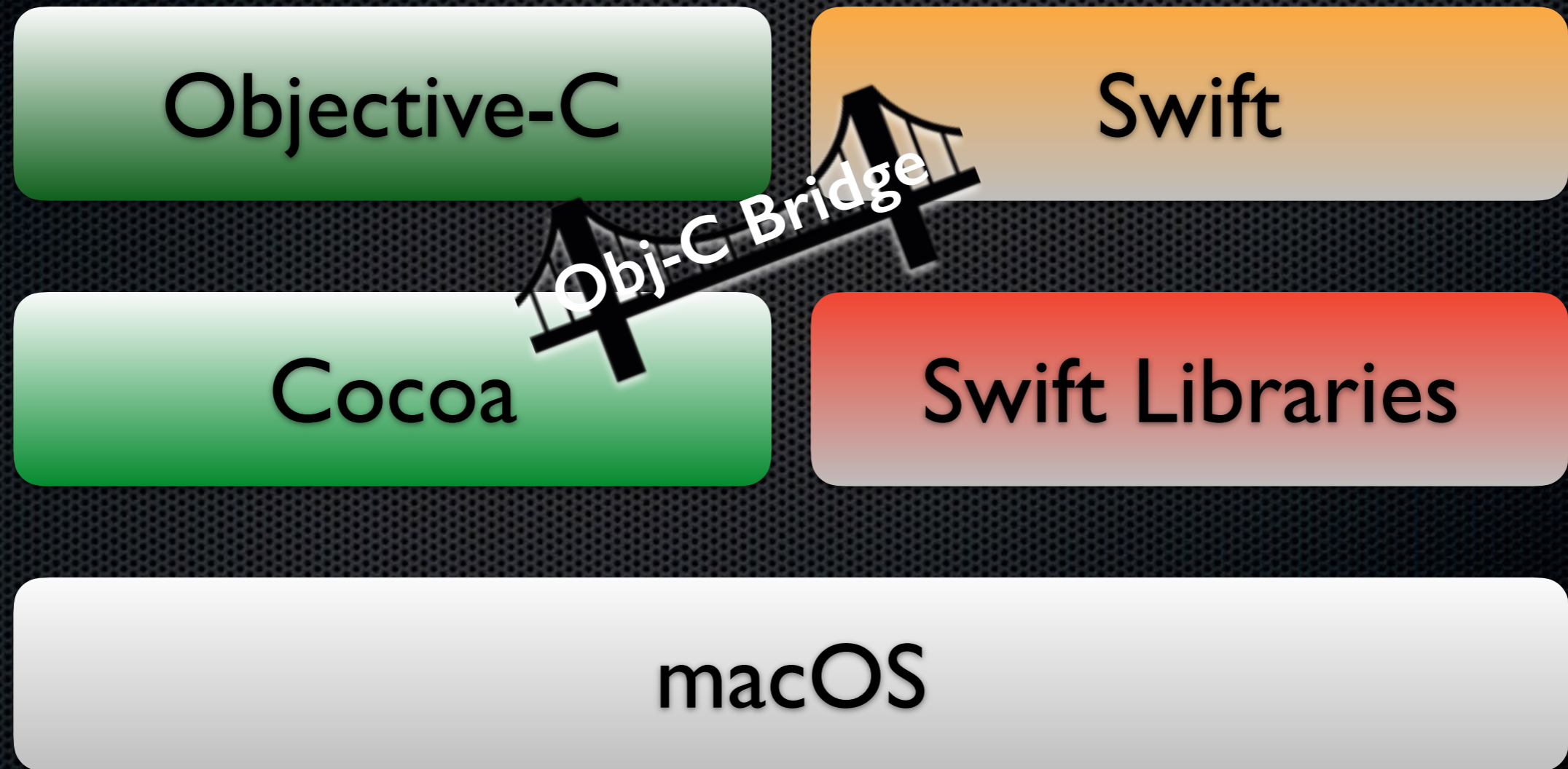


Modern Language Features

- Closures unified with function pointers
- Tuples and multiple return values
- Generics
- Fast and concise iteration over a range or collection
- Structs that support methods, extensions, and protocols
- Functional programming patterns, e.g., map and filter
- Powerful error handling built-in
- Advanced control flow with do, guard, defer, and repeat
- Enums
- String interpolation (v5)



Swift-ObjC





Swift-ObjC

Objective-C

```
UITableView *myTableView = [[UITableView alloc]  
initWithFrame:CGRectZero  
style:UITableViewStyleGrouped];
```

Swift

```
let myTableView = UITableView(frame: .zero,  
style: .grouped)
```



Apple's Goals with Swift

- Fast
- Modern
- Safe
- Interactive

“Objective-C without the C”

–Craig Federighi



Apple's Goals with Swift

- Fast
- Modern
- Safe
- Interactive



REPL

(Read Eval Print Loop)

```
Welcome to Apple Swift version 4.  
Type :help for assistance.  
  1> print("Hello, World!")  
Hello, World!
```



REPL


(Read Eval Print Loop)

```
Welcome to Apple Swift version 4.  
Type :help for assistance.  
  1> print("Hello, World!")  
Hello, World!  
  2> :script  
Python Interactive Interpreter. T  
>>> █
```



Swift Playgrounds

Learn serious code on your iPad.
In a seriously fun way.

Download the new Swift Playgrounds free 





Script Mode

MyScript.py

```
#!/usr/bin/python  
print("Hello, World!")
```



Script Mode

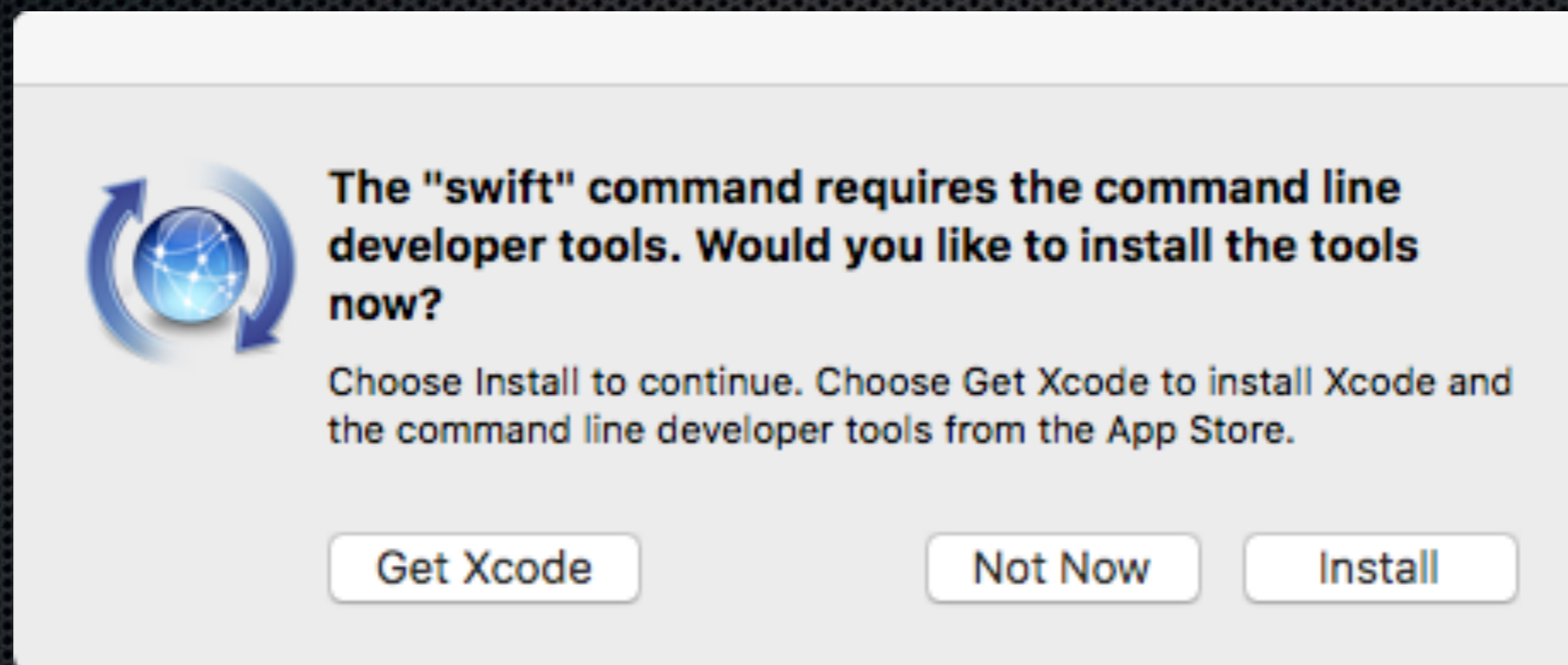
MyScript.swift

```
#!/usr/bin/swift  
print("Hello, World!")
```



Script Mode

MyScript.swift





Command Line Tools





Mac GUI Application

macOS System Frameworks

Cocoa Application - Application User Interface Responds to User Events, Manages App Behavior

App Kit Notification Center Game Center Sharing Full Screen Mode Cocoa Autolayout Popovers Software Configuration Accessibility Apple Script Spotlight

Media Plays, records, editing audiovisual media, Rendering 2D and 3D graphics

AV Foundation
Audio Playback, editing, Analysis & Recording

Core Animation
2D rendering & Animation
3D Transformations

Core Audio
Audio Services for recording, playback and synchronization

Core Image
Fast Image Processing
Uses GPU Based acceleration

Core Text
Handles Unicode Fonts & texts

Open AL
Delivers 3D Audio
High performance positional playbacks in games

Open GL
Portable 3D graphics apps & Games
Imaging functions & Effects

Quartz
OSX Graphics, Rendering support for 2D content
Event Routing & Cursor Management

Core Services - Fundamental Services for low level network communication, Automatic Reference Counting, Data Formatting, String Manipulation

Address Book
Centralized Database for contacts & groups

Core Foundation
declares C based programmatic interfaces
Data Types & Data Management

Quick Look
Enables Spotlight & finder to display thumbnail images

Security
User Authentication, Certificates & keys, Authorization, Keychain Services etc

Core Data
Data Model Management & Storage, Undo/Redo, Validation of property values

Foundation
Objective C Framework for Object Behavior, Internationalization, Data Types & Data Management

Social
Supports integration with Social Networking services

Webkit
Display HTML Content in apps. contains WebCore and JavaScript Core

Core OS - Related to hardware and networking. Interfaces for running high-performance computation tasks on CPU or GPU

Accelerate
Accelerate complex operations, improve performance using vector unit, Supports data parallelism, 3d Graphic imaging, image processing

Directory Services
Provides access to collected information about users, groups, computers, printers in a networked environment

Disk Arbitration
Notifies when local or remote volumes are mounted and unmounted

Open CL
Makes the high-performance parallel processing power of GPUs available to general purpose computing

System Configuration
Provides access to current network configuration information. Determines reachability of remote hosts. Notifies about change in network

Kernel & Device Drivers - Device drivers & BSD Libraries, low level components. Support for file system security, interprocess communications, device drivers

BSD
Provides basis for file systems and networking facilities, POSIX Thread support, BSD Sockets

File System
Supports multiple volume formats (NTFC, ExFAT, FAT etc) & File Protocols (AFP, NFS etc)

Mach
Protected Memory, Preemptive multitasking, Advanced Virtual Memory, Real Time Support

Networking
Supports network kernel extensions (NKEs). Create network modules, Configure protocol stacks, Monitor and modify network traffic



Command Line Tools

macOS System Frameworks

Cocoa Application - Application User Interface Responds to User Events, Manages App Behavior

App Kit

Notification Center

Game Center

Sharing

Full Screen Mode

Cocoa Autolayout

Popovers

Software Configuration

Accessibility

Apple Script

Spotlight

Media Plays, records, editing audiovisual media, Rendering 2D and 3D graphics

AV Foundation

Audio Playback, editing, Analysis & Recording

Core Animation

2D rendering & Animation
3D Transformations

Core Audio

Audio Services for recording, playback and synchronization

Core Image

Fast Image Processing
Uses GPU Based acceleration

Core Text

Handles Unicode Fonts & texts

Open AL

Delivers 3D Audio
High performance positional playbacks in games

Open GL

Portable 3D graphics apps & Games
Imaging functions & Effects

Quartz

OSX Graphics, Rendering support for 2D content
Event Routing & Cursor Management

Core Services - Fundamental Services for low level network communication, Automatic Reference Counting, Data Formatting, String Manipulation

Address Book

Centralized Database for contacts & groups

Core Foundation

declares C based programmatic interfaces
Data Types & Data Management

Quick Look

Enables Spotlight & finder to display thumbnail images

Security

User Authentication, Certificates & keys, Authorization, Keychain Services etc

Core Data

Data Model Management & Storage, Undo/Redo, Validation of property values

Foundation

Objective C Framework for Object Behavior, Internationalization, Data Types & Data Management

Social

Supports integration with Social Networking services

Webkit

Display HTML Content in apps. contains WebCore and JavaScript Core

Core OS - Related to hardware and networking. Interfaces for running high-performance computation tasks on CPU or GPU

Accelerate

Accelerate complex operations, improve performance using vector unit, Supports data parallelism, 3d Graphic imaging, image processing

Directory Services

Provides access to collected information about users, groups, computers, printers in a networked environment

Disk Arbitration

Notifies when local or remote volumes are mounted and unmounted

Open CL

Makes the high-performance parallel processing power of GPUs available to general purpose computing

System Configuration

Provides access to current network configuration information. Determines reachability of remote hosts. Notifies about change in network

Kernel & Device Drivers - Device drivers & BSD Libraries, low level components. Support for file system security, interprocess communications, device drivers

BSD

Provides basis for file systems and networking facilities, POSIX Thread support, BSD Sockets

File System

Supports multiple volume formats (NTFS, ExFAT, FAT etc) & File Protocols (AFP, NFS etc)

Mach

Protected Memory, Preemptive multitasking, Advanced Virtual Memory, Real Time Support

Networking

Supports network kernel extensions (NKEs). Create network modules, Configure protocol stacks, Monitor and modify network traffic

Python Notification Alert

```
#!/usr/bin/env python
from Foundation import NSUserNotification
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotificationDefaultSoundName
from optparse import OptionParser

def NotificationAlert(strTitle = 'Alert Title', strMessage = 'Message', bSound = True):
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(strTitle)
    notification.setInformativeText_(strMessage)
    notification.
    if bSound:
        notification.setSoundName_(NSUserNotificationDefaultSoundName)

    center = NSUserNotificationCenter.defaultUserNotificationCenter()
    center.deliverNotification_(notification)

def main():
    NotificationAlert('Example', 'Test', True)

if __name__ == '__main__':
    main()
```



Example
Test

Swift Notification Alert

```
import Foundation

func createNotification(title: String, name: String) {
    let notification = NSUserNotification()
    notification.title = title
    notification.subtitle = name
    NSUserNotificationCenter.default.scheduleNotification(notification)
}

createNotification(title:"Example", name:"Test")
```



Example
Test



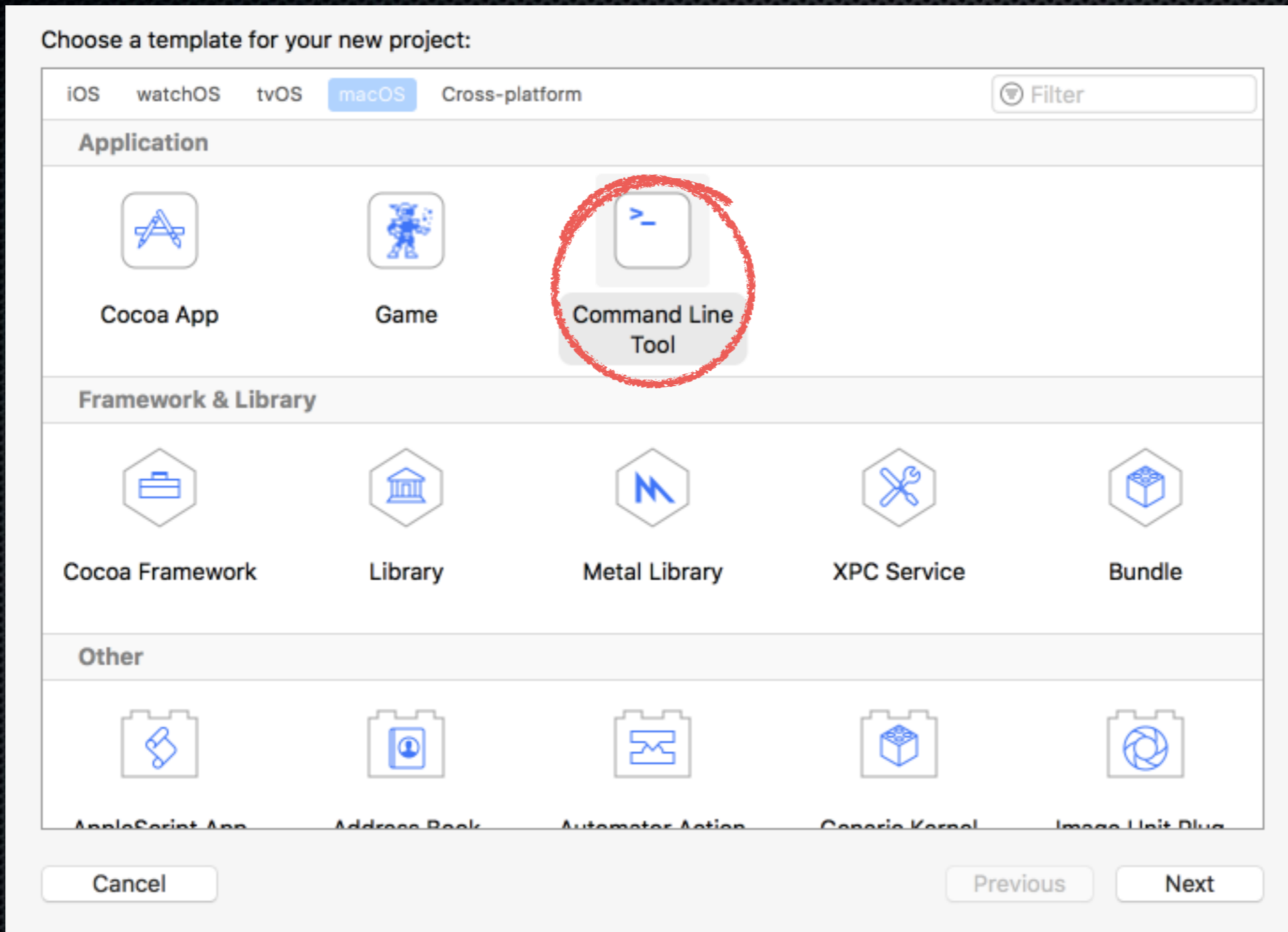
Xcode Advantages



- Excellent editor with interactive syntax checking and auto-completion
- Built in debugger
- Built in Git support
- Documentation included
- Private source code
- Code signing and notarization



Command Line Tools





Command Line Tools

Choose options for your new project:

Product Name: MacTech Demo

Team: ProVUE Development

Organization Name: Jim Rea

Organization Identifier: com.mactech

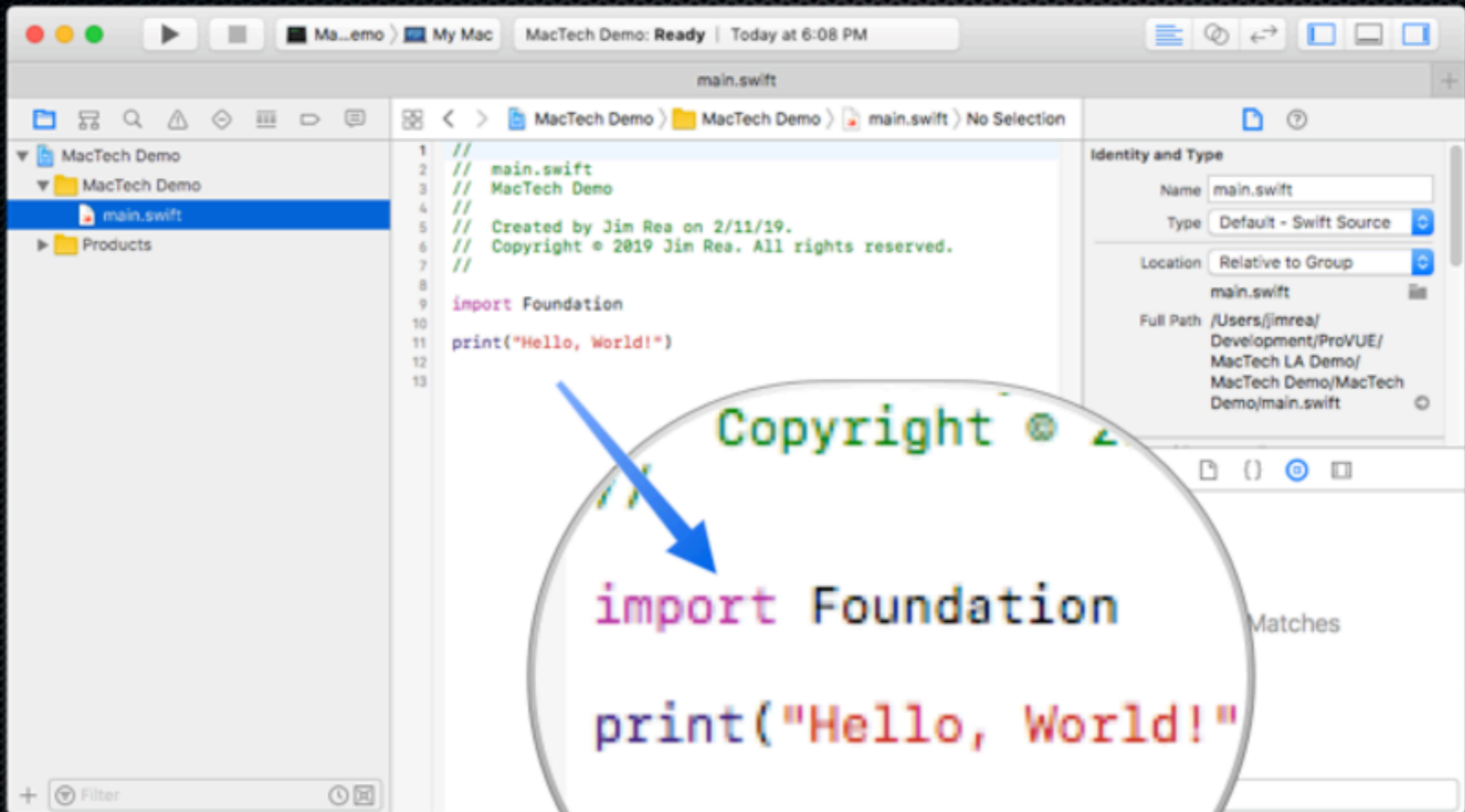
Bundle Identifier: com.mactech.MacTech-Demo

Language: Swift

Cancel Previous Next

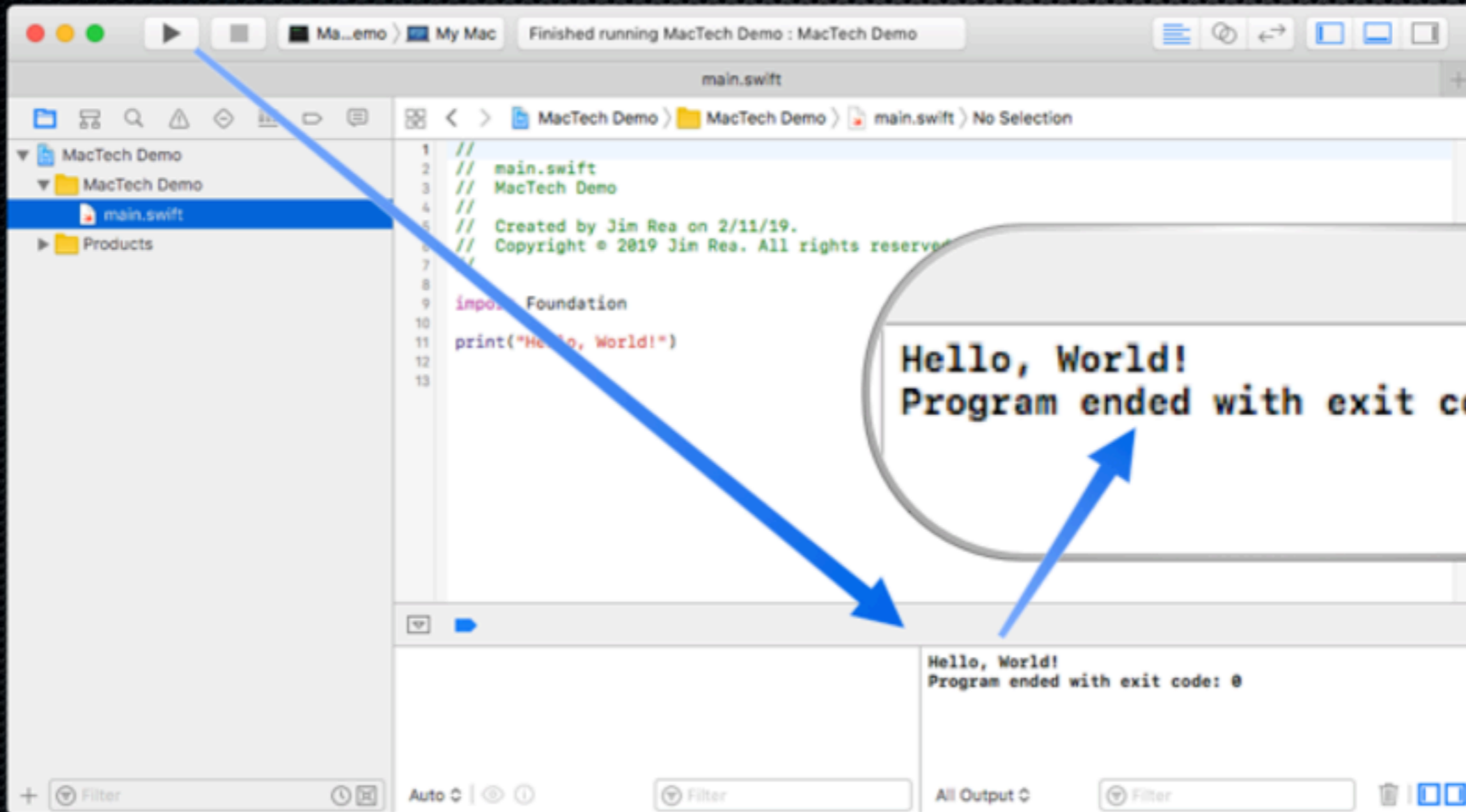


Command Line Tools





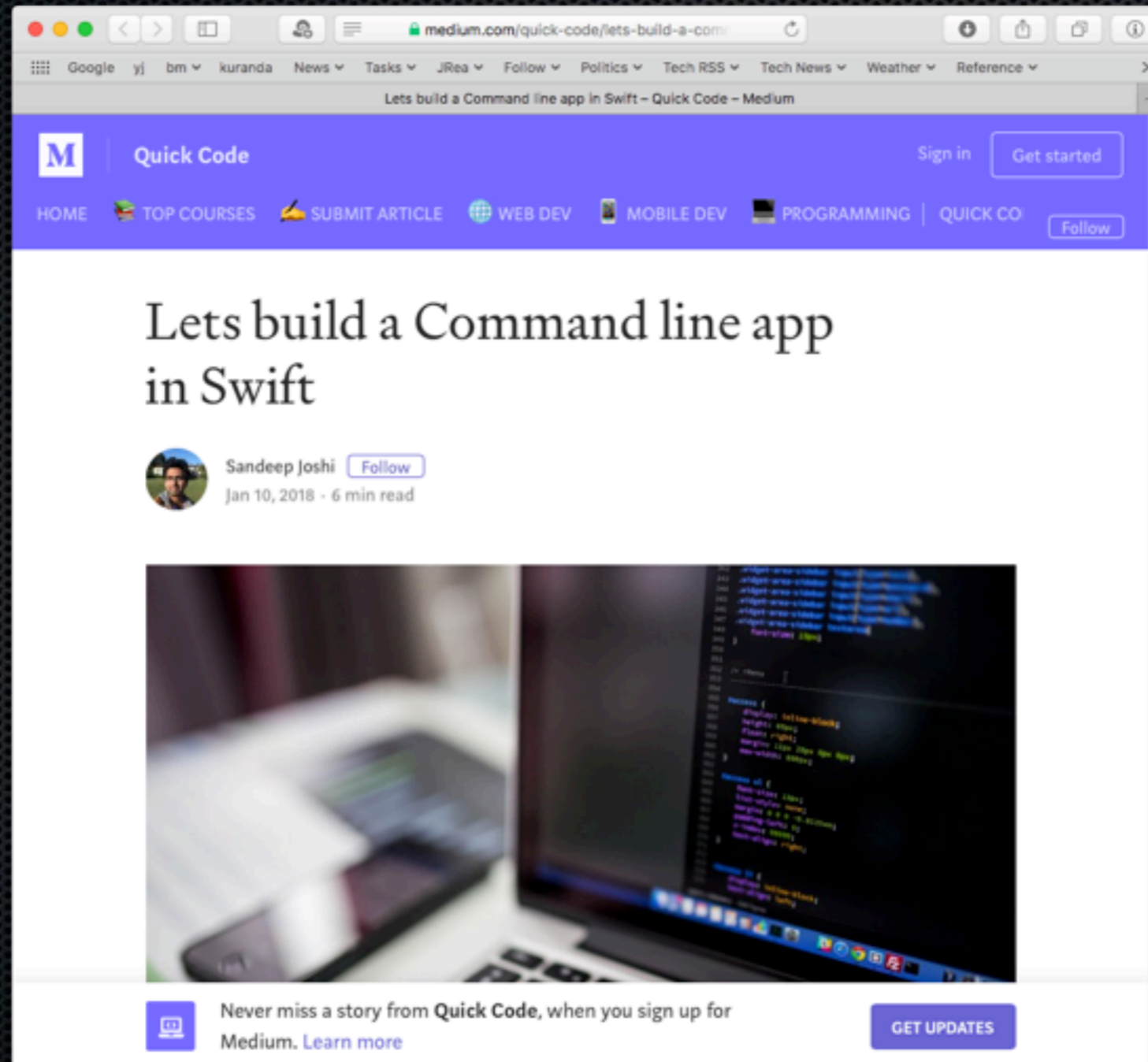
Command Line Tools





Command Line Tools

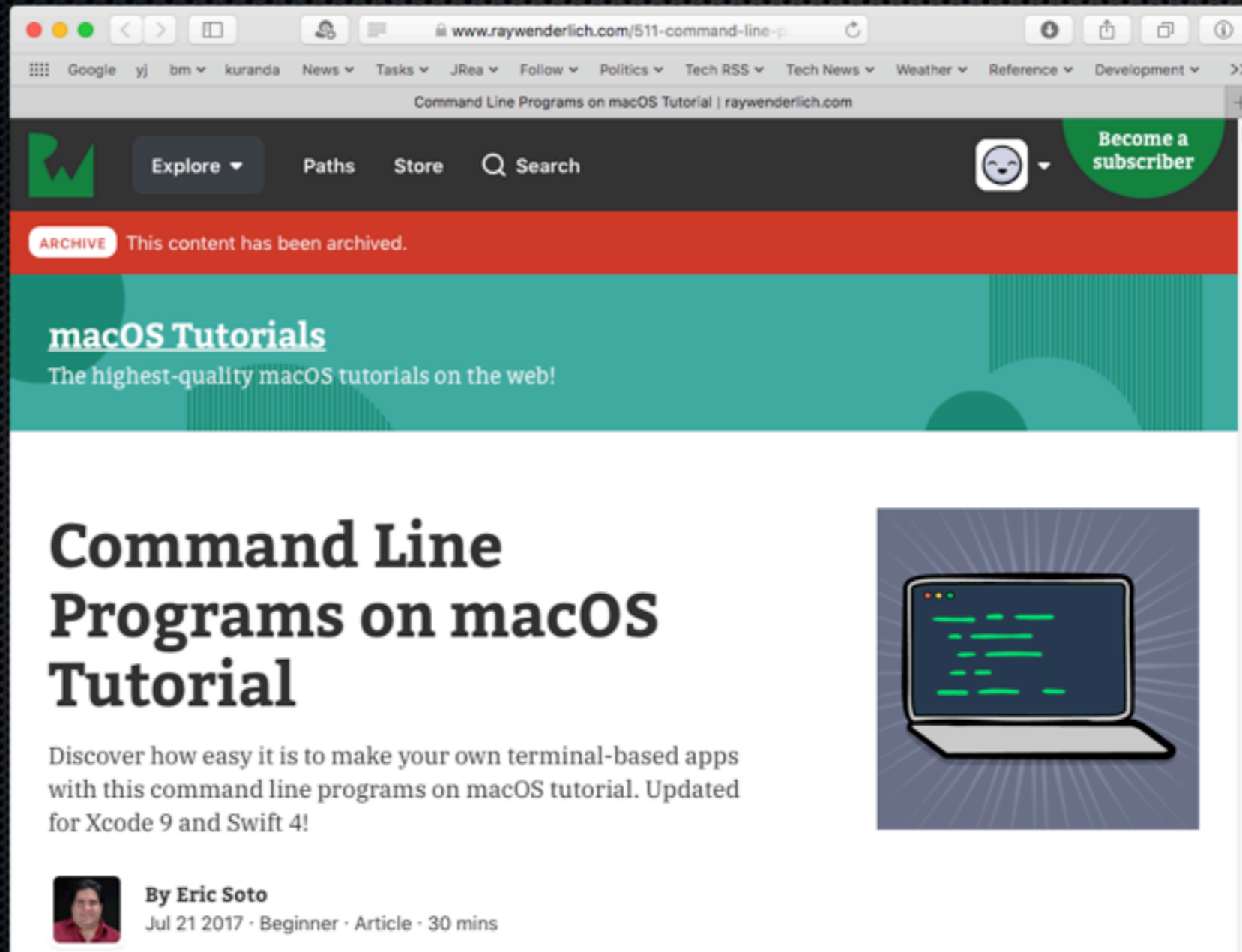
<https://medium.com/quick-code/lets-build-a-command-line-app-in-swift-328ce274f1cc>





Command Line Tools

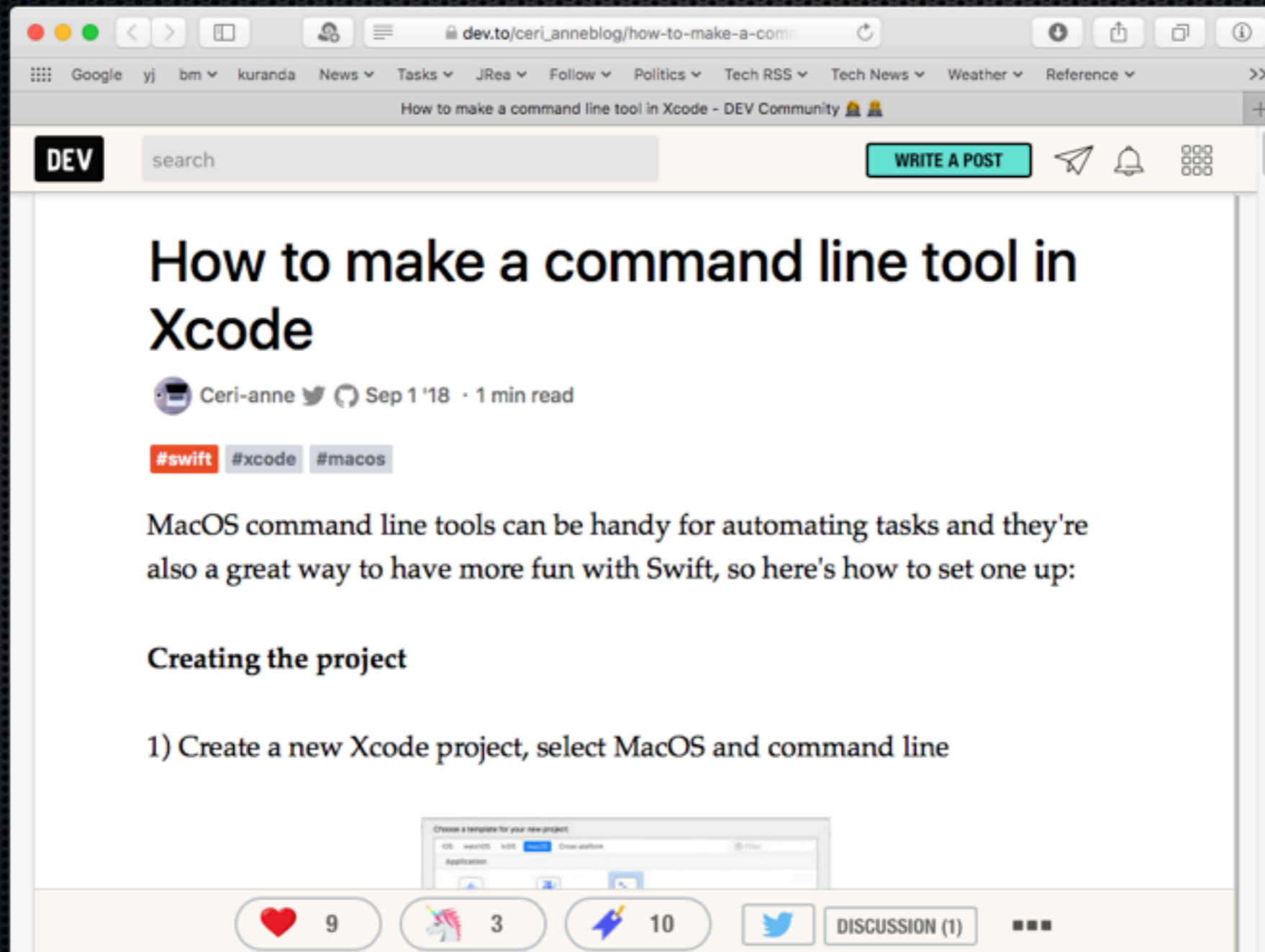
<https://www.raywenderlich.com/511-command-line-programs-on-macos-tutorial>





Command Line Tools

https://dev.to/ceri_anneblog/how-to-make-a-command-line-tool-in-xcode-2f81



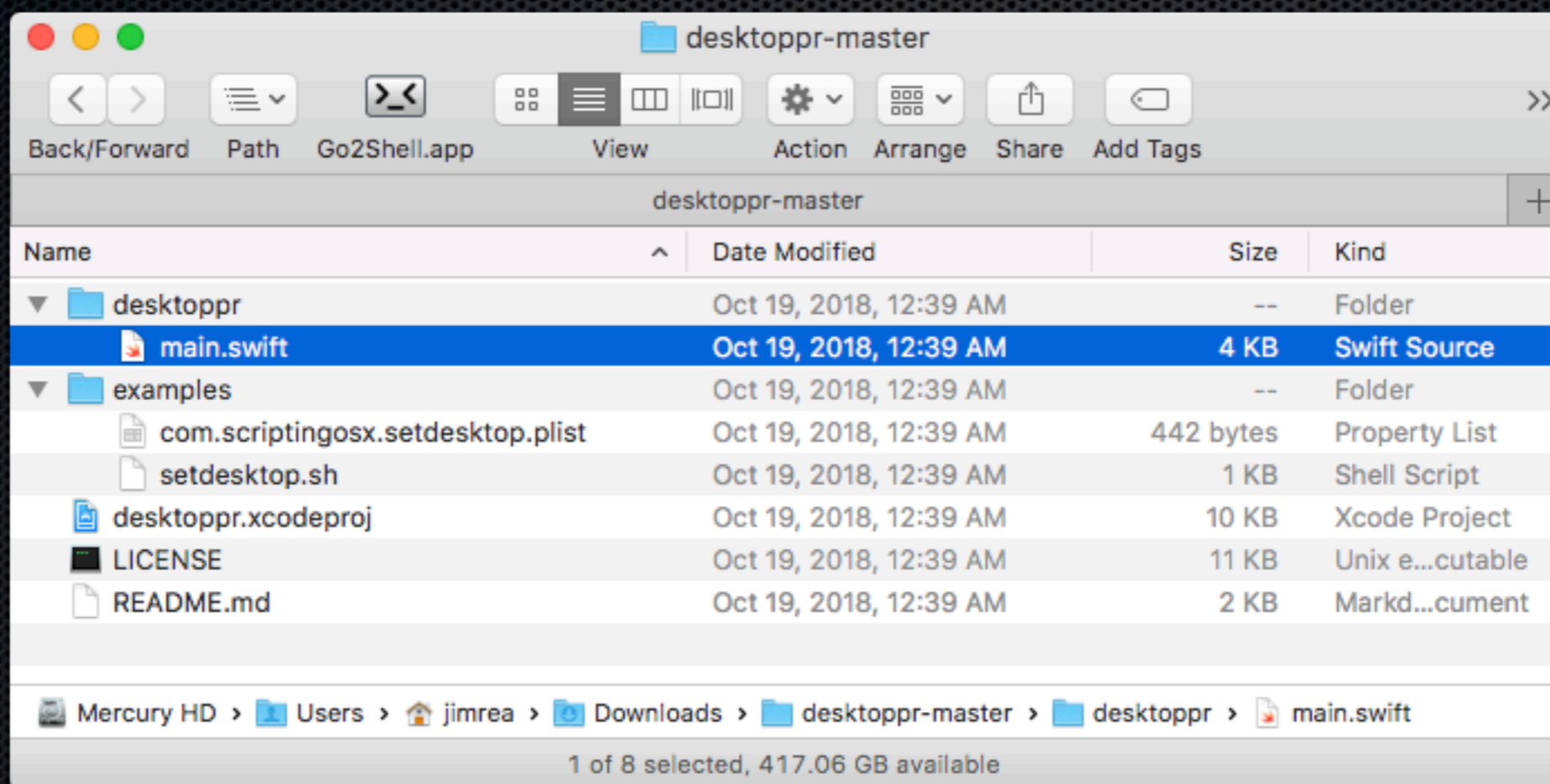
Swift Open Source Projects

- desktoppr
- DEPNotify
- SplashBuddy
- NoMAD & NoMADLogin-AD
- ProfileCreator
- replay

Swift Projects

desktoppr - command line tool to manage the desktop image

<https://scriptingosx.com/2018/09/managing-the-desktop-picture-on-macos/>



Swift Projects

DEPNotify - lightweight notification of DEP enrollment status

<https://gitlab.com/Mactroll/DEPNotify>



Swift Projects

SplashBuddy - onboarding splash screen for Jamf Pro

<https://github.com/Shufflepuck/SplashBuddy>

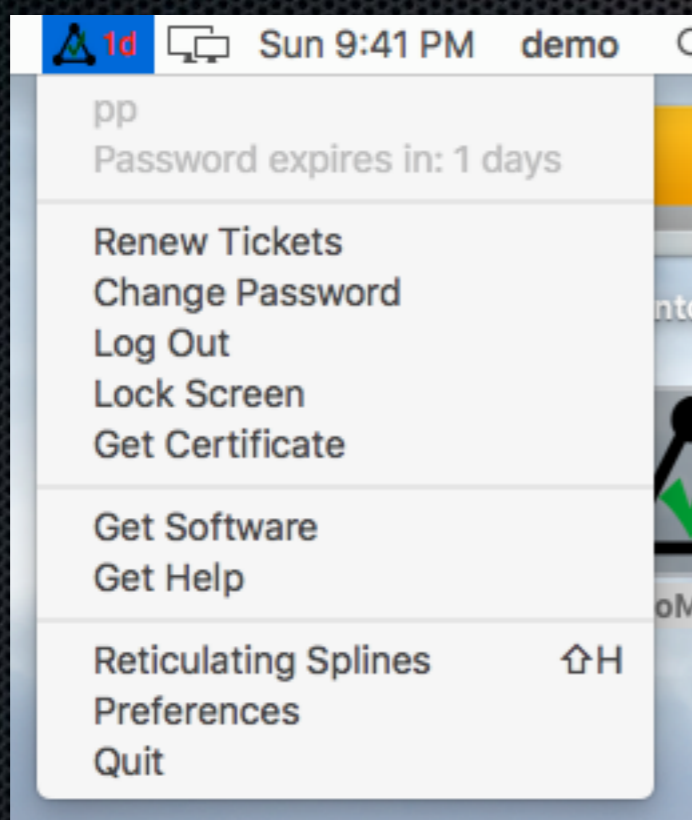


Swift Projects

NoMAD & NoMADLogin-AD - Active Directory enhancements

<https://gitlab.com/Mactroll/NoMAD>

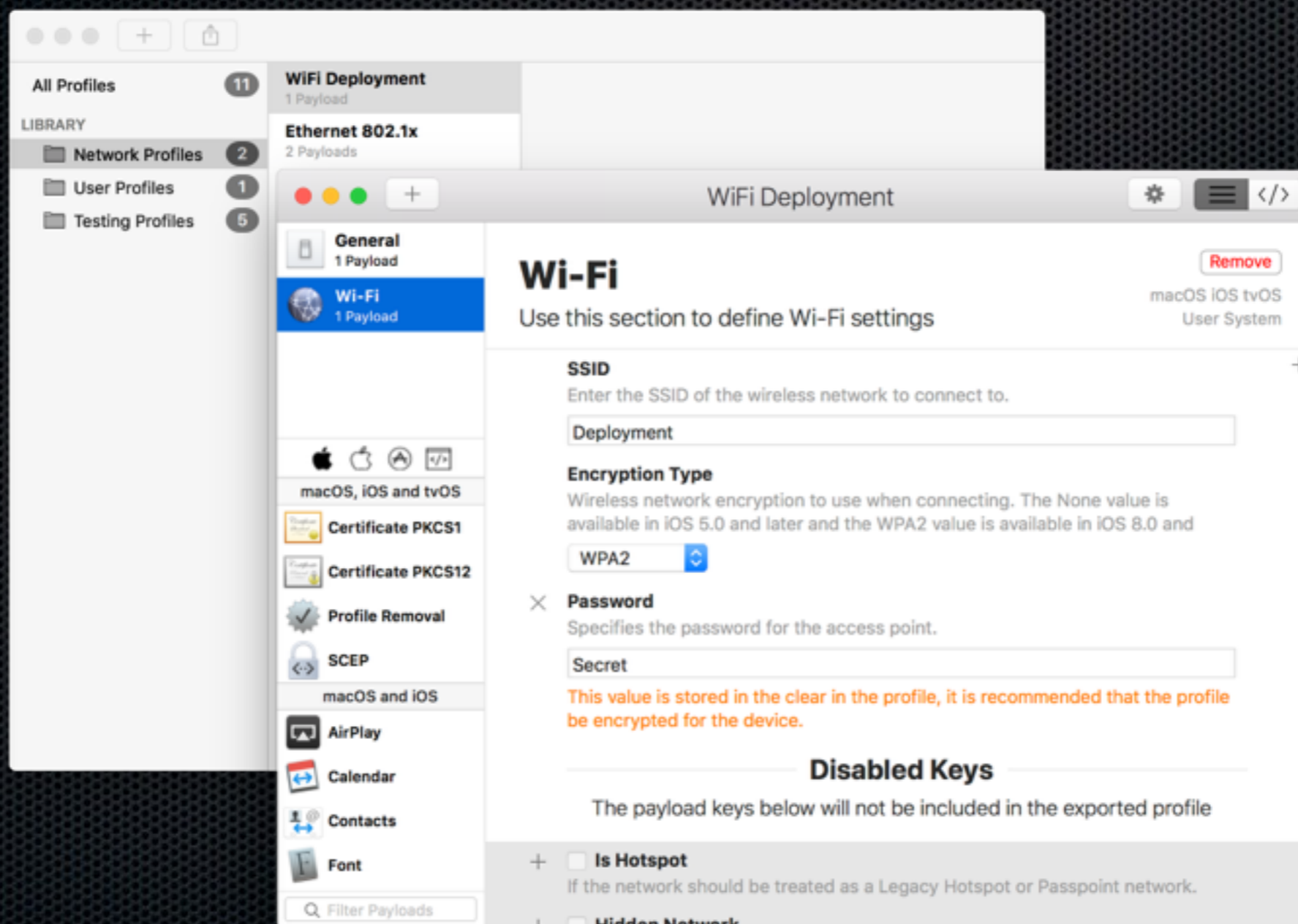
<https://gitlab.com/orchardandgrove-oss/NoMADLogin-AD>



Swift Projects

Profile Creator - Creates configuration profiles

<https://github.com/erikberglund/ProfileCreator>



Swift Projects

replay - converts AppleTV into a video kiosk

<https://github.com/vmware/replay-app-for-tvos>





Getting Started




macOS 10.14 Mojave



Xcode 10




Getting Started



The Swift Programming Language

Swift 4.2 Edition



Get

Get Sample

★★★★★ (1,501)

Published Jun 2, 2014

REQUIREMENTS

This book can only be viewed on an iOS device with Apple Books on iOS 12 or later, iBooks 1.5 or later and iOS 4.3.3 or later, or a Mac with iBooks 1.0 or later and OS X 10.9 or later.

MORE APPLE INC.

Collection

Alert Me

The Swift Programming Language (Swift 4.2)

Swift Programming Series >
Apple Inc. >

Details

Ratings and Reviews

Related

About the Book

Swift is a programming language for creating iOS, macOS, watchOS, and tvOS apps. Swift builds on the best of C and Objective-C, without the constraints of C compatibility. Swift adopts safe programming patterns and adds modern features to make programming easier, more flexible, and more fun. Swift's clean slate, backed by the mature and much-loved Cocoa and Cocoa Touch frameworks, is an opportunity to reimagine how software development works.

...

[more](#)

Screenshots

Apple Inc.

Arrays automatically grow as you add elements.

```
1 shoppingList.append("blue pants")
2 print(shoppingList)
```

To create an empty array or dictionary, use the initializer syntax.

```
1 let emptyArray = [String]()
2 let emptyDictionary = [String: Float]()
```

If type information can be inferred, you can write an empty array as [] and an empty dictionary as {}—but examples, when you set a new value for a variable or pass an argument to a function.

```
1 shoppingList = []
2 occupations = {}
```

The Swift Programming Language (Swift 4.2)

around the condition or loop variable are optional, braces around the body are required.

```
1 let individualScores = [75, 45, 90, 97, 100]
2 var teamScore = 0
3 for score in individualScores {
4     - if score > 50 {
5         teamScore += 3
6     }
7 }
8
9 print(teamScore)
10 // Prints "15"
```

In an if statement, the conditional must be a Boolean expression—this means that code such as if score < 100 is an error, not an implicit comparison to zero.

You can use if and let together to work with values that might be missing. These values are represented as optionals. An optional value either contains a value or contains nil to indicate that a value is missing. Write a

Apple Inc.

A while loop performs a set of statements until a condition becomes false. These kinds of loops are best used when the number of iterations is not known before the first iteration begins. Swift provides two kinds of while loops:

- while evaluates its condition at the start of each pass through the loop.
- repeat-while evaluates its condition at the end of each pass through the loop.

While


A while loop starts by evaluating a single condition. If the condition is true, a set of statements is repeated until the condition becomes false.

Here's the general form of a while loop:

```
while (condition) {
    statements
}
```

The Swift Programming Language (Swift 4.2)

This example (also known as the Fibonacci sequence) calculates the first 10 Fibonacci numbers.



The rules of the while loop are:

- The loop condition must be true.
- The loop body must contain at least one statement that changes the loop condition.
- Each time the loop body is executed, the loop condition is evaluated again.
- If the loop condition is false, the loop terminates.

Information

Language	English	Published	Jun 2, 2014
Genre	Programming	Updated	Sep 18, 2018
Publisher	Apple Inc.	Pages	500
Seller	Apple Inc.	Size	4.3 MB

MAGTECH
PRO EVENTS

65



Getting Started



- Everyone Can Code
<https://www.apple.com/everyone-can-code/>
- Stanford's Swift Course
<https://itunes.apple.com/us/course/developing-ios-11-apps-with-swift/id1309275316>



Everyone Can Code

- Julian Schiavo
<https://schiavo.me>



Questions?



Jim Rea

www.provue.com

@provuejim