SWIFT Programming Language

Paula Pamplona

## Introduction

Swift has recently entered the programming language world. Within around the past five to ten years. Our society has relying heaving on smartphone apps. Apple is one of the fastest growing in the mobile app department. Some developers at Apple decided to set a goal of creating the programming language: Swift to aid in mobile app developing for iOS. The idea of Swift began in 2010 by Chris Lattner along with Doug Gregor, John McCall, Ted Kremenek, and Joe Groff at Apple. Swift was inspired by Objective-C, Rust, Haskell, Python, Ruby, CLU, C#, and other programming languages. One of the main reasons Swift was created is for it to be a successor of Objective-C. Swift was officially introduced to the world in 2014 at Apples’ Worldwide Developers Conference. During this conference a prototype, very similar to the actual product, was released and available to some Apple developers. A 500 page manual called “The Swift Programming Language” was released and is currently available on iBook and the official Apple incorporation website. Very soon after the release of Swift in 2015, Swift became the most loved programming language according to the /stack overflow developer survey. In 2016, it placed second in the survey. Apple is a very protective over its product and ecosystem that Swift became a huge success.

## Versions

*Version Swift 1.0, 1.1, 1.2, & Xcode 6.0, 6.1, 6.3*

The first ever version of Swift was released on September 9th, 2014. It was also released alongside the Gold Masters of Xcode 6. This was the introduction of iOS and macOS app development. Swift is fully compatible with Objective-C and much faster than Objective-C. It is an object-oriented programming language with better improved philosophies that will be discussed in the next section. Swift came out with new features such as constants, variables, type interface, generic classes, function, closures, tuples, and dictionaries (Mindfire). Along with Apples’ release of their new programming language, they also came out with a new version of Xcode that helps developers to easily test their Swift code. A month later, in October, Swift version 1.1 was released version 1.0 with Xcode 6.1. Swift 1.2 was released April 8th, 2015 with Xcode 6.3.

*Version Swift 2.0, 2.1, 2.2, & Xcode 7.0, 7.1*

Swift 2.0 was announced at Apples’ Worldwide Developers Conference in June 2015. Pre-release versions of Swift 2.0 and Xcode 7 were available to developers. It wasn’t until September 15th, 2015 that Apple officially released Swift 2.0 and the Xcode 7 Gold Master build. Swift 2 updated to an open sourced version and more error handling safety features. This version made swift a protocol-oriented programming language. It improved the usage of interfaces, and methods and properties. It also updated some syntax features. Once again, a month later Swift 2.0 was updated to Swift 2.1 and Xcode released version 7.1. Apple released Swift 2.2 along with Xcode 7.3 on March 21st, 2016. Swift 2.2 made Swift available to Linux machine, rather than only macOS.

*Version Swift 3.0, 3.1, & Xcode 8*

Swift 3.0 was first announced before the release of Swift 2.2 on December 3rd, 2015. The version’s game plan was announced on GitHub and the actual release of Swift 3.0 and Xcode 8 was on September 13th, 2016. Swift 3 changed code and new requirement for programmers to label all function parameters. The purpose of this change is to keep code more concise. Swift changed naming techniques from UpperCamelCase to lowerCamelCase for classes, structs, enums, or properties. Swift 3 aided the implementation of C functions by making it easier to import C functions. Swift’s API guidelines were also improved to easily describe concepts. Apple released Swift 3.1 on March 27th, 2017.

*Version Swift 4.0, 4.2, & Xcode 9*

Swift 4.0 was announced at Apples’ Worldwide Developers Conference on June 15th, 2017 and was officially released on March 29th, 2020. Before the official release of this version. A beta version was released for testing. Swift 4 came with new and improved features for the APIs like strings, set, dictionary, archival, and serialization (Mindfire). The new protocol allows developers to simply code the serialization and deserialization of data types. Swift 4.2 was released on September 17th, 2018.

*Version Swift 5.0, 5.1, 5.2, 5.3, & Xcode 11, 12*

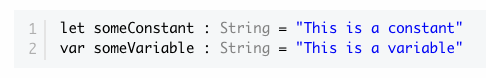
Swift 5.0 was released on March 25th, 2019. This release adds ABI Stability, ABI Stability enables applications created in different Swift versions to be compatible. Binary frameworks can now be distributed with the improved ABI Stability and requires module format stability, which stabilizes the framework’s interfaces. Some new features added in Swift 5 are dynamic callable types and improved enumeration handling. It also added some features to the Standard Library like vector types as well as enhanced string interpolation and support. Swift 5.1 and Xcode 11 were released on September 20th, 2019. Swift 5.2 released on March 24th, 2020. At the most recent Apples’ Worldwide Developers Conference, Apple announced the release of Swift 5.3. This version was officially released on September 16th, 2020 alongside Xcode 12.

# Philosophy

Swift is a parallel scripting language, meaning large data management passing through applications that increase the complexity for developers is easier to manage. Swift uses parallelism, which is used to aid the design process by reducing power supply and clock frequency. The designers of Swift followed a philosophy that focused less on the inheritance of objects and more on their behavior. Swift heavily focused on improving Objective-C structural methods. Swift improved code blocks, array implementation, Automatic Reference Counting (ARC), along with other features. Code initialization is one Swift’s important aspects the designers’ focused on. Swift requires the programmer to initialize all object and variables that are defined in order for the program to run without any compile time errors. All objects and variables must have a value. However, there is a way around this Swift designed, which is called an optional. An optional is one of Swift’s enums that can declares the variable can have a type or no type.

Along with Swift’s focus on variable and objects, it has a preference of constants instead of variables. Since Swift uses multithreaded environments, the use of constants rather than variables helps create safer code. It also leads to finer code optimization during compile time, because the compiler will know the value is not going to change. Figure 1 shows how to handle constants and variables. The keyword ‘let’ is used to define a constant and the keyword ‘var’ defines a variable.

**Figure 1. Difference between constants and variables**



To add to the other philosophies discussed, Swift also focused on branches and their required conditions. Swift requires all conditional statements, like if/else and switch/case to provide all conditions asked for. This helps reduce compile time error by covering all possible conditions. Swift wants to reduce compile time errors and has used these philosophies to create a less error prone language. The philosophies the designers’ used helped them achieve their goal of creating a “fast, modern, safe, and interactive programming language.

# Influences

Swift is still fairly new to the programming language world, but has greatly succeeded with the help of some matured languages. Some of these languages that influence Swift are the following: Python, C, and C-objective. There are many other languages that influenced Swift throughout its development.

## Python

Python and Swift may not seem very similar right off the bat, however they do share some things in common. Python’s easy to follow syntax influenced Swift’s syntax and they are both similar to English. Unlike many languages, adding a semicolon at the end of each line is not a problem developers have to deal with when developing in Python or Swift. Python however is used as more of a backend programming language, while Swift focuses on both frontend and backend.

## C

Most operators used in Swift are also used in C. The functionality of the equals sign ‘=’ is primary the same for both languages. ‘=’ is used for assigning variables and ‘==’ is used for comparison. ‘===’ is a new operator that is used to check if data elements refer to the same object. Control and conditional statements in C for the most part influenced Swift with some different features. When dealing with Swift and while and if statements, pattern matching and unwrapping optionals are something the programmer has to pay attention to in order to reduce compile time errors.

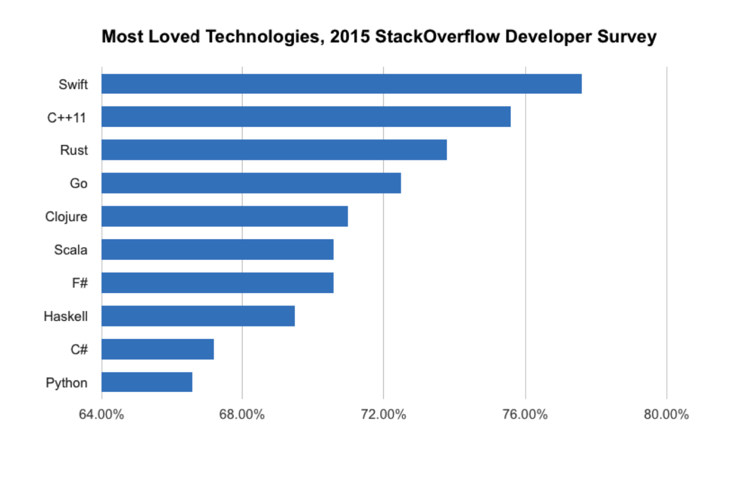
## C-Objective

There are more differences between Swift and C-Objective than similarities, but C-Objective did influence the creation of Swift. They both use the same basic types like Int, Float, and Double. Both also have class methods that are inherited and use ‘self’ is the class the method was called on. They both have similar for loop syntax and neither require semicolons after each line. Unlike C-Objective, Swift has error-handling.

# Criticism

It is not news that Apple is huge in the technology world. In 2016, it actually provided the world with more than 1 billion devices. “iOS is the world’s second most popular mobile platform” (AltexSoft Inc). As mentioned earlier, Swift won the Most Loved Technology survey in 2015 created by StackOverFlow. Figure 2 shows all the rating of most popular languages.

**Figure 2: StackOverFlow Survey Results in 2015**



Swift has been very successful since it release, but this does not mean it has not been criticized. Swift is very young and has a lot of improvements to make before considering it the “perfect language.” Swift has a smaller amount of available libraries and tools compared to older more improved languages. As Swift continues to come out with versions, programs created with the older versions are not compatible with new Swift versions. Swift makes drastic changes every release that it makes it hard to be compatible with its own older versions. According to another StackOverFlow survey, “only 6.5 percent of the 64,000 respondents use Swift” (AltexSoft Inc). However, the demand for Swift programming is growing immensely causing a talent gap. The problems with incompatible versions are still constantly being reported, which can hurt the growth of the language. Unfortunately Swift is only compatible with apps that are iOS7 or later, leading to more restrictions when developing Swift applications.

Swift has a lot of room to grow, but it is also dependent of the success and popularity of Apple, macOS, and iOS. Since Swift is mainly only used for Apple ecosystems, it’s competition with for largely popular languages and ecosystems like .NET, Java, and Python is hard to beat.

# Object-Oriented Programming

Object-Oriented Programming is an important paradigm to be a successful Swift programmer. Object-Oriented Programming, also known as OOP, is used to simplify code when using classes, for example. OOP consists of three main concepts: encapsulation, inheritance, and polymorphism.

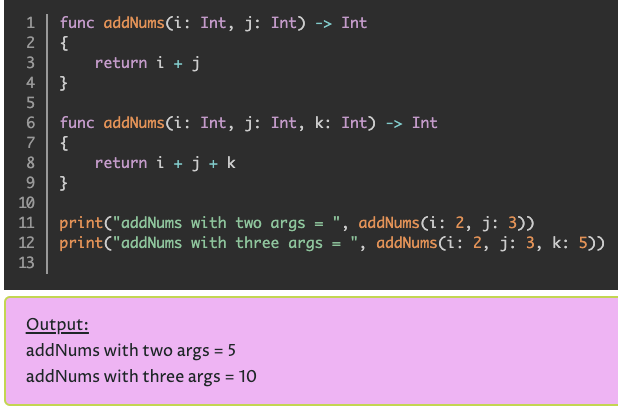
Encapsulation deals with the restriction of access to some data, variables, or fields. This helps keep a program consistent when needed and prevent unnecessary changes. Swift uses properties to encapsulate data. Swift’s encapsulation methods are not as strict as they are for C. It is not as strong, because of the relationship between objects. In C, there is a .c file which includes inaccessible encapsulated data. There will be times where inherited objects need access to their parent data and the inability to do this leads to complex code.

Inheritance allows objects to adopt the behavior and properties already existing. Swift does provide the mechanism used for inheritance. Swift and Objective-C provide a special mechanism called subtyping. Subtyping refers to being able to use the inherited type anywhere the parent or base type can be used.

Polymorphism “allows different data types to be handles using a uniform interface” (Anokhin). In other words, this allows inherited types to be used anywhere the base type can be used. In Swift, polymorphism has two different instances: compile time polymorphism and runtime polymorphism. Compile time polymorphism is basically method overloading, which means that functions can have the same name and perform different tasks depending on the number and type of parameters. Figure 3 shows an example of compile time polymorphism.

**Figure 3: Compile Time Polymorphism**

**https://www.cosmiclearn.com/swift/polymorphism.php**



Runtime polymorphism is the same as method overriding. Runtime polymorphism woks simultaneously with inheritance. Swift does not know which method to use until after compile time. The methodology is that a function can have a similar name and number of parameters, but the behavior of the function is based on the type of the parameter.

# REFERENCES

1. M. Rebouças, G. Pinto, F. Ebert, W. Torres, A. Serebrenik and F. Castor, "An Empirical Study on the Usage of the Swift Programming Language," 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, 2016, pp. 634-638, doi: 10.1109/SANER.2016.66.
2. Cleverism. 2016. SWIFT. (August 2016). Retrieved November 13, 2020 from https://www.cleverism.com/skills-and-tools/swift/
3. cosmiclearn.com. SWIFT POLYMORPHISM. Retrieved November 13, 2020 from https://www.cosmiclearn.com/swift/polymorphism.php
4. Dmytro Anokhin. 2017. Fundamental concepts of OOP in Swift and how to implement them in C. (November 2017). Retrieved November 13, 2020 from https://medium.com/@dmytro.anokhin/fundamental-concepts-of-oop-in-swift-and-how-to-implement-them-in-c-6581d405f23f
5. Katherine A. Klise and David L. Woodruff. 2019. Parallelism. (2019). Retrieved November 13, 2020 from https://www.sciencedirect.com/topics/computer-science/parallelism
6. Michael Musallam. 2014. An Introduction to Swift: Part 1. (July 2014). Retrieved November 13, 2020 from https://code.tutsplus.com/tutorials/an-introduction-to-swift-part-1--cms-21389
7. Michael Wilde, Mihael Hategan, Justin M. Wozniak, Ben Clifford, Daniel S. Katz, and Ian Foster. 2011. Swift: A language for distributed parallel scripting. (June 2011). Retrieved November 13, 2020 from https://www.sciencedirect.com/science/article/pii/S0167819111000524
8. Mindfire Solutions. 2017. The Evolution of Swift. (December 2017). Retrieved November 13, 2020 from https://medium.com/@mindfiresolutions.usa/the-evolution-of-swift-29e7a89f1a0c
9. Paris Buttfield-Addison. 2015. Swift is open source and why you should care. (December 2015). Retrieved November 13, 2020 from https://www.oreilly.com/content/swift-is-open-source-and-why-you-should-care/
10. Sonoo Jaiswal. History of Swift - Javatpoint. Retrieved November 13, 2020 from https://www.javatpoint.com/history-of-swift