Swift 4

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Swift 4 is a new programming language developed by Apple Inc for iOS and OS X development. Swift 4 adopts the best of C and Objective-C, without the constraints of C compatibility.

Swift 4 uses the same runtime as the existing Obj-C system on Mac OS and iOS, which enables Swift 4 programs to run on many existing iOS 6 and OS X 10.8 platforms.

Audience

This tutorial is designed for software programmers who would like to learn the basics of Swift 4 programming language from scratch. This tutorial will give you enough understanding on Swift 4 programming language from where you can take yourself to higher levels of expertise.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of Computer Programming terminologies and exposure to any programming language.

Execute Swift 4 Online

For most of the examples given in this tutorial, you will find a **Try it** option, so just use this option to execute your Swift 4 programs on the spot and enjoy your learning.

Try the following example using **Try it** option available at the top right corner of the following sample code box:

```
import Cocoa


/* My first program in Swift 4 */
var myString = "Hello, World!"
print(myString)
```

Disclaimer & Copyright

Table of Contents

1. Swift 4 – Overview

Swift 4 is a new programming language developed by Apple Inc for iOS and OS X development. Swift 4 adopts the best of C and Objective-C, without the constraints of C compatibility.

- Swift 4 makes use of safe programming patterns.
- Swift 4 provides modern programming features.
- Swift 4 provides Objective-C like syntax.
- Swift 4 is a fantastic way to write iOS and OS X apps.
- Swift 4 provides seamless access to existing Cocoa frameworks.
- Swift 4 unifies the procedural and object-oriented portions of the language.
- Swift 4 does not need a separate library import to support functionalities like input/output or string handling.

Swift 4 uses the same runtime as the existing Obj-C system on Mac OS and iOS, which enables Swift 4 programs to run on many existing iOS 6 and OS X 10.8 platforms.

Swift 4 comes with playground feature where Swift 4 programmers can write their code and execute it to see the results immediately.

The first public release of Swift was released in 2010. It took **Chris Lattner** almost 14 years to come up with the first official version, and later, it was supported by many other contributors. Swift 4 has been included in Xcode 6 beta.

Swift designers took ideas from various other popular languages such as Objective-C, Rust, Haskell, Ruby, Python, C#, and CLU.

2. Swift 4 – Environment

Try it Option Online

You really do not need to set up your own environment to start learning Swift 4 programming. Reason is very simple, we already have set up Swift 4 environment online, so that you can execute all the available examples online at the same time when you are doing your theory work. This gives you the confidence in what you are reading and in addition to that, you can verify the result with different options. Feel free to modify any example and execute it online.

Try the following example using the **Try it** option available at the top right corner of the following sample code box:

```
import Cocoa


/* My first program in Swift 4 */
var myString = "Hello, World!"


print(myString)
```

For most of the examples given in this tutorial, you will find a **Try it** option, so just make use of it and enjoy your learning.

Local Environment Setup

Swift 4 provides a Playground platform for learning purpose and we are going to setup the same. You need xCode software to start your Swift 4 coding in Playground. Once you are comfortable with the concepts of Swift 4, you can use xCode IDE for iSO/OS x application development.

To start with, we consider you already have an account at Apple Developer website. Once you are logged in, go to the following link:

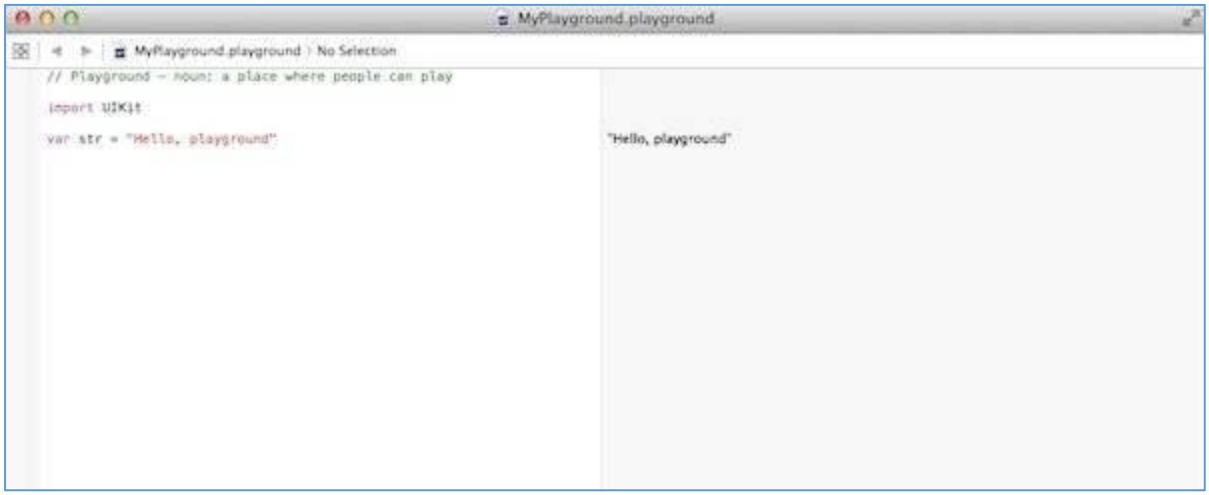
```
Download for Apple Developers
```

This will list down a number of software available as follows:

Now select xCode and download it by clicking on the given link near to disc image. After downloading the dmg file, you can install it by simply double-clicking on it and following the given instructions. Finally, follow the given instructions and drop xCode icon into the Application folder.

Now you have xCode installed on your machine. Next, open Xcode from the Application folder and proceed after accepting the terms and conditions. If everything is fine, you will get the following screen:

Select **Get started with a playground** option and enter a name for playground and select iOS as platform. Finally, you will get the Playground window as follows:



Following is the code taken from the default Swift 4 Playground Window.

```
import UIKit


var str = "Hello, playground"
```

If you create the same program for OS X program, then it will include **import Cocoa** and the program will look like as follows:

```
import Cocoa
var str = "Hello, playground"
```

When the above program gets loaded, it should display the following result in Playground result area (Right Hand Side).

```
Hello, playground
```

Congratulations, you have your Swift 4 programming environment ready and you can proceed with your learning vehicle "Tutorials Point".

3. Swift 4 – Basic Syntax

We have already seen a piece of Swift 4 program while setting up the environment. Let's start once again with the following **Hello, World!** program created for OS X playground, which includes **import Cocoa** as shown below:

```
import Cocoa

/* My first program in Swift 4 */
var myString = "Hello, World!"

print(myString)
```

If you create the same program for iOS playground, then it will include **import UIKit** and the program will look as follows:

```
import UIKit
var myString = "Hello, World!"
print(myString)
```

When we run the above program using an appropriate playground, we will get the following result.

```
Hello, World!
```

Let us now see the basic structure of a Swift 4 program, so that it will be easy for you to understand the basic building blocks of the Swift 4 programming language.

Import in Swift 4

You can use the **import** statement to import any Objective-C framework (or C library) directly into your Swift 4 program. For example, the above **import cocoa** statement makes all Cocoa libraries, APIs, and runtimes that form the development layer for all of OS X, available in Swift 4.

Cocoa is implemented in Objective-C, which is a superset of C, so it is easy to mix C and even C++ into your Swift 4 applications.

Tokens in Swift 4

A Swift 4 program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol. For example, the following Swift 4 statement consists of three tokens:

```
print("test!")
```

```
The individual tokens are:
print
(
   "test!"
)
```

Comments

Comments are like helping texts in your Swift 4 program. They are ignored by the compiler. Multi-line comments start with /* and terminate with the characters */ as shown below:

```
/* My first program in Swift 4 */
```

Multi-line comments can be nested in Swift 4. Following is a valid comment in Swift 4:

```
/* My first program in Swift 4 is Hello, World!
/* Where as second program is Hello, Swift 4! */ */
```

Single-line comments are written using // at the beginning of the comment.

```
// My first program in Swift 4
```

Semicolons

Swift 4 does not require you to type a semicolon (;) after each statement in your code, though it's optional; and if you use a semicolon, then the compiler does not complain about it.

However, if you are using multiple statements in the same line, then it is required to use a semicolon as a delimiter, otherwise the compiler will raise a syntax error. You can write the above Hello, World! program as follows:

```
import Cocoa
/* My first program in Swift 4 */
var myString = "Hello, World!"; print(myString)
```

Identifiers

A Swift 4 identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with an alphabet A to Z or a to z or an underscore _ followed by zero or more letters, underscores, and digits (0 to 9).

Swift 4 does not allow special characters such as @, $, and % within identifiers. Swift 4 is a **case sensitive** programming language. Thus, *Manpower* and *manpower* are two different identifiers in Swift 4. Here are some examples of acceptable identifiers:

```
Azad        zara     abc    move_name   a_123
myname50    _temp    j      a23b9       retVal
```

To use a reserved word as an identifier, you will need to put a backtick (`) before and after it. For example, **class** is not a valid identifier, but **`class`** is valid.

Keywords

The following keywords are reserved in Swift 4. These reserved words may not be used as constants or variables or any other identifier names, unless they're escaped with backticks:

Keywords used in declarations

Class	deinit	Enum	extension
Func	import	Init	internal
Let	operator	private	protocol
public	static	struct	subscript
typealias	var		

Keywords used in statements

break	case	continue	default
do	else	fallthrough	for
if	in	return	switch
where	while		

Keywords used in expressions and types

as	dynamicType	false	is
nil	self	Self	super
true	_COLUMN_	_FILE_	_FUNCTION_
LINE			

Keywords used in particular contexts

associativity	convenience	dynamic	didSet
final	get	infix	inout
lazy	left	mutating	none
nonmutating	optional	override	postfix

precedence	prefix	Protocol	required
right	set	Type	unowned
weak	willSet		

Whitespaces

A line containing only whitespace, possibly with a comment, is known as a blank line, and a Swift 4 compiler totally ignores it.

Whitespace is the term used in Swift 4 to describe blanks, tabs, newline characters, and comments. Whitespaces separate one part of a statement from another and enable the compiler to identify where one element in a statement, such as int, ends and the next element begins. Therefore, in the following statement:

```
var age
```

there must be at least one whitespace character (usually a space) between **var** and **age** for the compiler to be able to distinguish them. On the other hand, in the following statement:

```
int fruit = apples + oranges   //get the total fruits
```

no whitespace characters are necessary between fruit and =, or between = and apples, although you are free to include some for better readability.

Space on both side of a operator should be equal, for eg.

```
int fruit= apples +oranges   //is a wrong statement
int fruit = apples + oranges   //is a Correct statement
```

Literals

A literal is the source code representation of a value of an integer, floating-point number, or string type. The following are examples of literals:

```
92                 // Integer literal
4.24159            // Floating-point literal
"Hello, World!"    // String literal
```

Printing in Swift

To print anything in swift we have ' print ' keyword.

print has three different properties.

Items – Items to be printed

Separator – separator between items

Terminator – the value with which line should end, let's see a example and syntax of same.

```
print("Items to print", separator: "Value " , Terminator: "Value")
// E.g. of print statement.
print("Value one")
// prints "Value one \n" Adds, \n as terminator and " " as separator by
default.

print("Value one","Value two", separator: " Next Value" , terminator: " End")
//prints "Value one Next Value Value two End"
```

In the above code first print statement adds \n , newline Feed as terminator by default, where as in second print statement we've given " End " as terminator, hence it'll print "End " instead of \n.

We can give our custom separator and terminators according to our requirement.

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**