



Swift 4

tutorialspoint

SIMPLY EASY LEARNING

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

© Copyright 2017 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com.

Table of Contents

About the Tutorial.....	i
Audience	i
Prerequisites	i
Execute Swift 4 Online	i
Disclaimer & Copyright.....	i
Table of Contents	ii
1. SWIFT 4 – OVERVIEW	1
2. SWIFT 4 – ENVIRONMENT	2
Try it Option Online.....	2
Local Environment Setup.....	2
3. SWIFT 4 – BASIC SYNTAX	6
Import in Swift 4	6
Tokens in Swift 4	6
Comments.....	7
Semicolons.....	7
Identifiers.....	7
Keywords	8
Whitespaces.....	9
Literals	9
Printing in Swift.....	9
4. SWIFT 4 – DATA TYPES	11
Built-in Data Types	11
Bound Values	12
Type Aliases	12

Type Safety	13
Type Inference	13
5. SWIFT 4 – VARIABLES	15
Variable Declaration	15
Type Annotations	16
Naming Variables	16
Printing Variables	17
6. SWIFT 4 – OPTIONALS	18
Forced Unwrapping	18
Automatic Unwrapping	19
Optional Binding	20
7. SWIFT 4 – TUPLES	21
8. SWIFT 4 – CONSTANTS	22
Constants Declaration	22
Type Annotations	22
Naming Constants	23
Printing Constants	23
9. SWIFT 4 – LITERALS	24
Integer Literals	24
Floating-point Literals	24
String Literals	24
Boolean Literals	25
10. SWIFT 4 – OPERATORS	26
Arithmetic Operators	26
Comparison Operators	27

Logical Operators	27
Bitwise Operators	28
Assignment Operators.....	29
Range Operators	30
Misc Operators.....	31
Operators Precedence.....	31
11. SWIFT 4 – DECISION MAKING	33
if Statement	34
if-else Statement	35
if...else if...else Statement.....	36
Nested If Statements.....	38
Switch Statement	39
The ? : Operator	41
12. SWIFT 4 – LOOPS.....	42
for-in Loop	43
Swift 4 – while Loop	44
Swift 4 – repeat-while Loop.....	45
Loop Control Statements.....	47
Swift 4 – continue Statement	47
Swift 4 – break Statement	49
Swift 4 – Fallthrough Statement.....	50
13. SWIFT 4 – STRINGS	53
Create a String.....	53
Empty String.....	53
String Constants	54
String Interpolation	55

String Concatenation.....	55
String Length	56
String Comparison.....	56
String Iterating	56
Unicode Strings	57
String Functions & Operators	57
14. SWIFT 4 – CHARACTERS.....	60
Empty Character Variables	60
Accessing Characters from Strings.....	61
Concatenating Strings with Characters.....	61
15. SWIFT 4 – ARRAYS	62
Creating Arrays	62
Accessing Arrays.....	62
Modifying Arrays.....	63
Iterating Over an Array	64
Adding Two Arrays	65
The count Property	66
The empty Property	66
16. SWIFT 4 – SETS	68
Creating Sets	68
Accessing and modifying Sets.....	68
Iterating over a Set.....	69
Performing Set Operations.....	69
17. SWIFT 4 – DICTIONARIES	70
Creating Dictionary	70

Sequence Based Initialization	70
Filtering	71
Dictionary Grouping	71
Accessing Dictionaries	71
Modifying Dictionaries	72
Remove Key-Value Pairs.....	73
Iterating Over a Dictionary	74
Convert to Arrays	75
The count Property	76
The empty Property	76
18. SWIFT 4 – FUNCTIONS	77
Function Definition	77
Calling a Function	78
Parameters and Return Values	78
Functions without Parameters	79
Functions with Return Values.....	79
Functions without Return Values	80
Functions with Optional Return Types	80
Functions Local Vs External Parameter Names	81
External Parameter Names	82
Variadic Parameters.....	82
Constant, Variable, and I/O Parameters.....	83
Function Types & its Usage	84
Using Function Types	85
Function Types as Parameter Types & Return Types	85
Nested Functions	86

19. SWIFT 4 – CLOSURES	87
Expressions in Closures	88
Single Expression Implicit Returns.....	89
Known Type Closures	90
Declaring Shorthand Argument Names as Closures.....	90
Closures as Operator Functions.....	91
Closures as Trailers.....	91
Capturing Values and Reference Types	92
20. SWIFT 4 – ENUMERATIONS	94
Enumeration Functionality	94
Enumeration with Switch Statement.....	95
Difference between Associated Values and Raw Values.....	96
Enum with Associated Values.....	96
Enum with Raw Values.....	97
21. SWIFT 4 – STRUCTURES	98
Definition of a Structure.....	98
Accessing the Structure and its Properties	98
Best Usage Practices of Structures	100
22. SWIFT 4 – CLASSES	102
Class Identity Operators.....	104
23. SWIFT 4 – PROPERTIES	106
Stored Properties	106
Lazy Stored Property	107
Instance Variables	108
Computed Properties.....	108

Local and Global Variables	111
Type Properties.....	111
Querying and Setting Properties	112
24. SWIFT 4 – METHODS	113
Instance Methods	113
Local and External Parameter Names.....	114
External Parameter Name with # and _ Symbol	115
Self property in Methods	116
Modifying Value Types from Instance Methods	117
Self Property for Mutating Method	118
Type Methods	118
25. SWIFT 4 – SUBSCRIPTS.....	120
Subscript Declaration Syntax and its Usage	120
Options in Subscript	122
26. SWIFT 4 – INHERITANCE	124
Base Class.....	124
Subclass	125
Overriding	126
Methods Overriding	126
Property Overriding	127
Overriding Property Observers.....	128
Final Property to prevent Overriding.....	129
27. SWIFT 4 – INITIALIZATION	131
Initializer Role for Stored Properties	131
Setting Property Values by Default	132

Parameters Initialization	132
Local & External Parameters	133
Parameters without External Names.....	134
Optional Property Types	135
Modifying Constant Properties During Initialization.....	136
Default Initializers	137
Memberwise Initializers for Structure Types	138
Initializer Delegation for Value Types.....	138
Class Inheritance and Initialization.....	140
Initializer Inheritance and Overriding.....	142
Failable Initializer	143
Failable Initializers for Enumerations	144
Failable Initializers for Classes.....	145
Overriding a Failable Initializer.....	146
The init! Failable Initializer	147
Required Initializers	148
28. SWIFT 4 – DEINITIALIZATION	149
Deinitialization to Deallocate Memory Space.....	149
29. SWIFT 4 – ARC OVERVIEW	151
Functions of ARC	151
ARC Program	151
ARC Strong Reference Cycles Class Instances	152
ARC Weak and Unowned References	153
Strong Reference Cycles for Closures	155
Weak and Unowned References	156

30. SWIFT 4 – OPTIONAL CHAINING	158
Optional Chaining as an Alternative to Forced Unwrapping	158
Defining Model Classes for Optional Chaining & Accessing Properties	160
Calling Methods Through Optional Chaining	162
Accessing Subscripts through Optional Chaining	163
Accessing Subscripts of Optional Type	167
Linking Multiple Levels of Chaining	169
Chaining on Methods with Optional Return Values.....	173
31. SWIFT 4 – TYPE CASTING	175
Defining a Class Hierarchy	175
Type Checking	176
Downcasting	178
Typecasting:Any and Any Object	180
AnyObject	182
32. SWIFT 4 – EXTENSIONS.....	185
Computed Properties	185
Initializers	186
Methods.....	188
Mutating Instance Methods	188
Subscripts.....	189
Nested Types.....	190
33. SWIFT 4 – PROTOCOLS	192
Property and Method Requirements.....	192
Mutating Method Requirements.....	194
Initializer Requirements	195

Class Implementations of Protocol Initializer Requirements	196
Protocols as Types.....	197
Adding Protocol Conformance with an Extension.....	198
Protocol Inheritance.....	199
Class Only Protocols	201
Protocol Composition.....	202
Checking for Protocol Conformance	203
34. SWIFT 4 – GENERICS.....	205
Generic Functions: Type Parameters	205
Extending a Generic Type	207
Type Constraints	208
Associated Types.....	209
Where Clauses	211
35. SWIFT 4 – ACCESS CONTROL	213
Access Control for Function types	213
Access Control for Enumeration types.....	214
Access Control for SubClasses	215
Access Control for Constants, variables, properties and subscripts	215
Getters and Setters	216
Access Control for Initializers and Default Initializers	216
Access Control for Protocols.....	217
Access Control for Extensions.....	218
Access Control for Generics.....	218
Access Control for Type Aliases	219
Swift Encoding and Decoding	222

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 iOS/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:



The screenshot shows the Apple Developer website interface. At the top, there are navigation links for Technologies, Resources, Programs, Support, and Member Center, along with a search bar. The main heading is 'Downloads for Apple Developers'. Below this, there's a search bar and a list of categories on the left, including Applications (13), Developer Tools (216), iOS (16), OS X (72), OS X Server (0), and Safari (1). The main content area displays a list of downloads with columns for Description and Release Date. The 'Xcode 6.2' entry is highlighted in blue and includes a description and a download link with a disc image icon.

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.

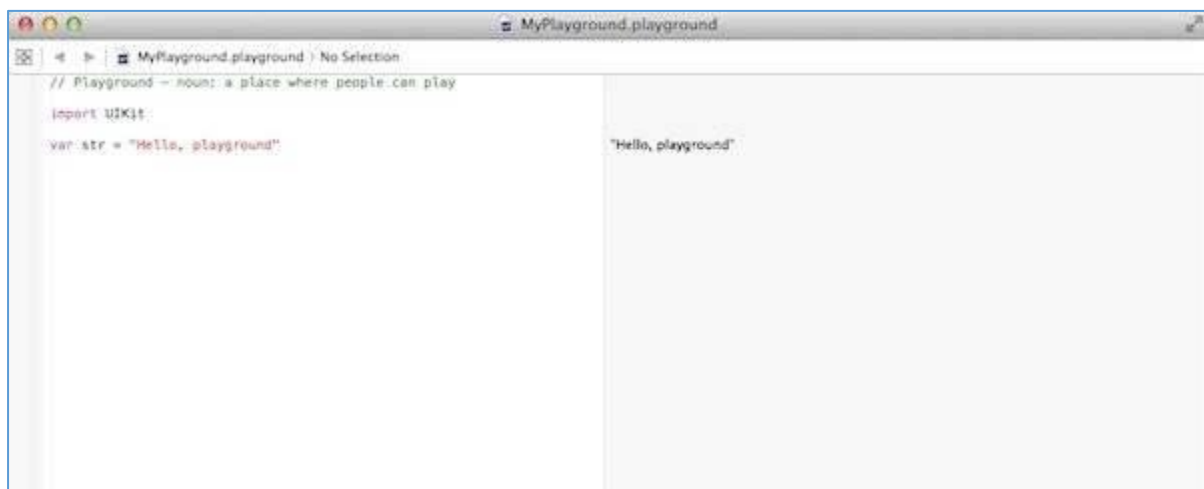


The screenshot shows the Xcode installation window. The window title is 'Xcode'. The main content area features the Xcode logo at the top. Below the logo, there is a diagram illustrating the installation process: a hammer icon labeled 'Xcode' is shown being dragged to a folder icon labeled 'Applications'. Below the diagram, the text reads 'Drag to install Xcode in your Applications 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 an 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>