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:

```swift
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>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About the Tutorial</td>
<td>i</td>
</tr>
<tr>
<td>Audience</td>
<td>i</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>i</td>
</tr>
<tr>
<td>Execute Swift 4 Online</td>
<td>i</td>
</tr>
<tr>
<td>Disclaimer &amp; Copyright</td>
<td>i</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>ii</td>
</tr>
<tr>
<td>1. SWIFT 4 – OVERVIEW</td>
<td>1</td>
</tr>
<tr>
<td>2. SWIFT 4 – ENVIRONMENT</td>
<td>2</td>
</tr>
<tr>
<td>Try it Option Online</td>
<td>2</td>
</tr>
<tr>
<td>Local Environment Setup</td>
<td>2</td>
</tr>
<tr>
<td>3. SWIFT 4 – BASIC SYNTAX</td>
<td>6</td>
</tr>
<tr>
<td>Import in Swift 4</td>
<td>6</td>
</tr>
<tr>
<td>Tokens in Swift 4</td>
<td>6</td>
</tr>
<tr>
<td>Comments</td>
<td>7</td>
</tr>
<tr>
<td>Semicolons</td>
<td>7</td>
</tr>
<tr>
<td>Identifiers</td>
<td>7</td>
</tr>
<tr>
<td>Keywords</td>
<td>8</td>
</tr>
<tr>
<td>Whitespaces</td>
<td>9</td>
</tr>
<tr>
<td>Literals</td>
<td>9</td>
</tr>
<tr>
<td>Printing in Swift</td>
<td>9</td>
</tr>
<tr>
<td>4. SWIFT 4 – DATA TYPES</td>
<td>11</td>
</tr>
<tr>
<td>Built-in Data Types</td>
<td>11</td>
</tr>
<tr>
<td>Bound Values</td>
<td>12</td>
</tr>
<tr>
<td>Type Aliases</td>
<td>12</td>
</tr>
</tbody>
</table>
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
Swift 4

14. SWIFT 4 – CHARACTERS .............................................................. 60
   String Concatenation ................................................................. 55
   String Length ................................................................................. 56
   String Comparison ........................................................................ 56
   String Iterating ............................................................................... 56
   Unicode Strings ............................................................................. 57
   String Functions & Operators ....................................................... 57

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

```swift
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:

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.

```swift
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:

```swift
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".
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:

```swift
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:

```swift
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:

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

```swift
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:

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

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

```swift
/* 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.

```swift
// 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:

```swift
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:
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**

<table>
<thead>
<tr>
<th>Class</th>
<th>deinit</th>
<th>Enum</th>
<th>extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Func</td>
<td>import</td>
<td>Init</td>
<td>internal</td>
</tr>
<tr>
<td>Let</td>
<td>operator</td>
<td>private</td>
<td>protocol</td>
</tr>
<tr>
<td>public</td>
<td>static</td>
<td>struct</td>
<td>subscript</td>
</tr>
<tr>
<td>typealias</td>
<td>var</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Keywords used in statements**

<table>
<thead>
<tr>
<th>break</th>
<th>case</th>
<th>continue</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>do</td>
<td>else</td>
<td>fallthrough</td>
<td>for</td>
</tr>
<tr>
<td>if</td>
<td>in</td>
<td>return</td>
<td>switch</td>
</tr>
<tr>
<td>where</td>
<td>while</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Keywords used in expressions and types**

<table>
<thead>
<tr>
<th>as</th>
<th>dynamicType</th>
<th>false</th>
<th>is</th>
</tr>
</thead>
<tbody>
<tr>
<td>nil</td>
<td>self</td>
<td>Self</td>
<td>super</td>
</tr>
<tr>
<td>true</td>
<td><em>COLUMN</em></td>
<td><em>FILE</em></td>
<td><em>FUNCTION</em></td>
</tr>
<tr>
<td><em>LINE</em></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Keywords used in particular contexts**

<table>
<thead>
<tr>
<th>associativity</th>
<th>convenience</th>
<th>dynamic</th>
<th>didSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>final</td>
<td>get</td>
<td>infix</td>
<td>inout</td>
</tr>
<tr>
<td>lazy</td>
<td>left</td>
<td>mutating</td>
<td>none</td>
</tr>
<tr>
<td>nonmutating</td>
<td>optional</td>
<td>override</td>
<td>postfix</td>
</tr>
</tbody>
</table>
Swift 4

<table>
<thead>
<tr>
<th>precedence</th>
<th>prefix</th>
<th>Protocol</th>
<th>required</th>
</tr>
</thead>
<tbody>
<tr>
<td>right</td>
<td>set</td>
<td>Type</td>
<td>unowned</td>
</tr>
<tr>
<td>weak</td>
<td>willSet</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

```swift
// 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