SWIFT CODING GUIDELINES

General Principles:

* Use camel case for classes, methods, variables, etc.

Note:*CamelCase (Camelcase, camelCase, camel case, camel caps or medial capitals) is the practice of writing compound words or phrases such that each word or abbreviation begins with a capital letter. Camel case may start with a capital or, especially in programming languages, with a lowercase letter*

* Its good to be both clear and brief but clarity should not suffer because of brevity.
* In general, don’t abbreviate names of things. Spell them out, even if they’re long. You may think an abbreviation is well-known, but it might not be, especially if the developer encountering your method or function name has a different cultural and linguistic background.
* Swift uses US English spelling to match with Apple’s API.

Preferred:

let color = “red”

Not Preferred:

let colour = “red”

Naming:

Use descriptive names with camel case for classes, methods, variables, etc. Class names should be capitalized, while method names and variables should start with a lower case letter.

Preferred:

Private let maximumWidgetCount = 100

Class WidgetContainer { //class name starts with upper case

var widgetButton: UIButton //variable name starts with lowecase

let sidgetHeightPercnetage = 0.85 //constant name starts with lower case

}

Not Preferred:

let MAX\_WIDGET\_COUNT = 100

class app\_widgetContainer {

var wBut: UIButton

let wHeightPct = 0.85

}

Spacing:

Indent using two spaces rather than using tabs to conserve space and prevent line wrapping.

Method braces and other braces (if/else/switch/while etc.) always open on the same line as the statement but close on a new line.

Note: You can re indent the same code by selecting the piece of cade you want to indent and then). Ctrl-I or (Editor\Structure\Re-Indent in the menu. Some of the Xcode template code will have 4-space tabs hard coded, so this is a good way to fix that.

Preferred:

if user.isHappy { //Braces opened on the same line

// Do something

} else { //Braces closed on a new line

// Do something else

}

Not Preferred:

if user.isHappy

{ //Do not open braces on a new line

// Do something

}

else {

// Do something else

}

There should be exactly one blank line between methods to aid in visual clarity and organization.

Whitespace within methods should separate functionality, but having too many sections in a method often means you should refractor into several methods.

Comment Lines:

Comment lines are used to explain why a piece of code does something.

Comment lines should be kept up to date or deleted.

Avoid block comments within the code as the code should be as self-documenting as possible.

Semicolons

Unlike many other languages, Swift does not require you to write a semicolon (;) after each statement in your code, although you can do so if you wish. However, semicolons are required if you want to write multiple separate statements on a single line.

Example:

Let number = 1; print(number)

//prints 1

Types:

Always use Swift’s native types when available.

Swift offers bridging to Objective-C so you can still use the full set of methods as needed.

Preferred:

let width = 120.0 //Double

let widthString = (width as NSNumber).stringValue //String

Not Preferred:

let width: NSNumber = 120.0 //NSNumber

let widthString: NSString = width.stringValue //NSString

Constants:

Constants are defined by using the let keyword.

Always use let when you are certain about the value and it will not change.

Note: It is a good practice to use let all the time and use var only when the compiler complains about it.

Variables:

Constants are defined by using th var keyword.

Always use var only when the compiler complains about using a constant.

Optionals:

Declare variables and function return types as optional with ? where a nil value is acceptable.

Use implicitly unwrapped types declared with ! only for instance variables that you know will be initialized later before use.

Type Inference:

Allow the compiler to infer the type of a constant or a variable , unless you need a specific type other than the default such as Int16.

Preferred:

let message = "Click the button"

let currentBounds = computeViewBounds()

var names = [String]()

Not Preferred:

let name : String = “John Morrison”

let names : [String] = []

Note: following descriptive is even more important than before.

Syntactic Sugar:

Prefer the shortcut versions of type declarations over the full generic syntax.

Preferred:

var names: [String]

var employees: [Int: String]

var phone: Int?

Not Preferred:

var names: Array<String>

var employees: Dictionary<Int, String>

var phone: Optional<Int>

Control flow:

Prefer the for-in style of for loop over the conventional for-condition-increment style.

Preferred:

for \_ in 0..<3 {

print("Hello three times")

}

Not Preferred:

for var i = 0; i < 3; i++ {

print("Hello three times")

}

Semicolons:

Unlike other languages Swift does not require semicolon after each statement in your code.

Semicolons are only required is you wish to combine multiple statements on a single line.

Preferred:

let swift = “It is a programming language”

Not Preferred:

let swift = “It is a programming language”;

Functions:

If the function declaration is short keep it on one line with the opening brace, For longer function declaration add a line break appropriately and add an extra indent.

Function with short declaration:

func rectangleArea(length: Int, breadth: Int) {

//function body goes here

}

Function with long declaration:

func printMathResult(mathFunction: (Int, Int) -> Int,

\_a: Int, \_b: Int {

//function body goes here

}

Classes, Structures and Enumerations:

Names of classes, structures and enumeration should always begin with uppercase letters.

Preferred:

Class NameOfTheClass {

//statements

}

struct NameOfTheStructure {

// statements

}

enum NameOfEnumeration {

// statements

}

Not Preferred :

Class nameOftheClass {

//statements

}

Same applies to structures and enumerations.