- Swift features
- Datatypes
- Constant (let) & Variables (var)
- Output statements
- Type Inference & Type Annotation
- Optionals(You use optionals in situations where a value may be absent.)
- Typecasting
- Conditional statements(if, if-else)
- Control statements (for-in while, repeat-while, switch)
- Functions
- Structures
- Enums
- typealias
- OOPs
- Strings
- Collection Classes (Arrays, Dictionaries & Sets)
- Access Specifiers
- Protocols
- Categories
- Closures
- Extensions
- Generics

For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com

SWIFT

Faster and Safer

-DNRED!

For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com

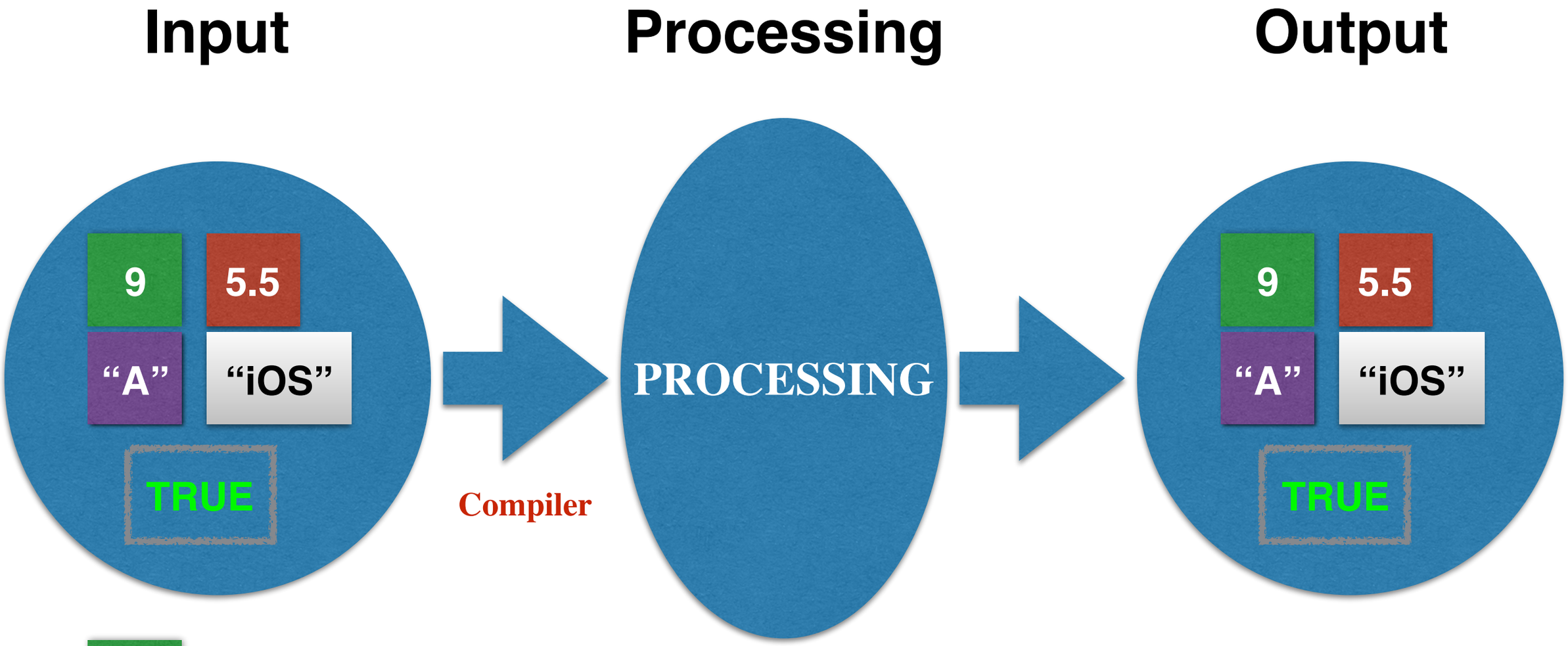
Swift Features

- Swift is a general purpose programming language for developing applications for iOS, MacOS X, tvOS and watchOS.
- Swift is type safe language
- High performance and faster
- Modern and Open Source
- Less code
- Swift can be used along with Objective-C. (Interoperability)

What is a Program?

Program is a set of instructions, which takes some input and produces some output in the form of Data.

The input taken by the program compiled by the Compiler and executed by the Runtime System.



9 — Whole values (Int type)

5.5 — Floating values (Float / Double type)

“A” — Single Character (Character type)

“iOS” — Multiple Characters (String type)

true — true/false (Bool type)

Data types

Based on the data / value, the data/values are classified into different types. Such as Int, Float, Double, Character, String, Bool etc..,

Declare the values before we use them as below

let a: Int = 10 (Here Int is a Datatype, a is Variable and 10 is value, let specifies a Not a constant)

The above statement conveys to the compiler that the variable ‘a’ stores an Int value (10).

DATA TYPE: Data type specifies to compiler that what kind of value the declared variable can store

Data types in Swift

Int (int) : stores integer values (10, 123, 597 etc)

Float (float): Stores floating point values

Double (double): Stores long precession float values

Character(char): Stores single character

Bool (BOOL): Stores true / false values

String (NSString*): Stores Textual data (“Hello”, “World”)

Array (NSArray*): Stores collections of similar or dissimilar objects or primitive datatypes and accessed through index(0 to n-1)

Dictionary (NSDictionary*): Stores Key - Value pairs

Set (NSSet): Stores objects or primitive datatypes as unordered collection of elements

Tuples: Stores group of values, those can be accessed through index (0 to n-1) or using specific identifiers

Any: Stores any kind of value / Object. (Obj-C: id)

Data types usage

```
Syntax: let/var variableName: Datatype = Value

let bRollNo: Int = 001
let bDistanceBetweenSunAndEarth: Int64 = 12345678912345

let bWeight: Float = 50.50
let bEarthWeight: Double = 999999999999.123456

let bInitial: Character = "S"
let bName: String = "Steve Jobs"

let bIsHuman: Bool = true
```

```
var aAnyObject: Any = 10 as Any
aAnyObject = "Any Object" as Any

// Collection of similar or dissimilar datatype elements
var smartPhones: Array = ["iPhone", "HTC", "Samsung", "Nokia",
"Windows"]

// Collection of Key-Value pairs of any similar or dissimilar type
elements
var yearOfIntroduce: Dictionary = ["iPhone": 2007, "Android": 2007,
"Windows":2000]
// Collection of similar or dissimilar non duplicate elements
var alphabets: Set = ["A","B","C","D","E","F"]
```

Constants and Variables

Variable declaration Syntax:

```
let/var variableName: Type = Value;
```

```
var aVariable: Int = 116;
```

Here Type is the data type of the variable.

Constant: The identifier whose value doesn't change. use let to create constants

```
let aConstant = 10;
```

Variable: The identifier whose value can be modified. Use var to create variables.

```
var aVariable = 100;
```

Note: You must specify the declared variable is constant(let) or variable(var)

For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com

Type Annotation & Type Inference

TYPE ANNOTATION:

The process of explicitly specifying the data type of declared Identifier (constant/variable)

```
let anInt: Int = 120;
let aFloat: Float = 12.12;
let aBool: Bool = true;
```

TYPE INFERENCE:

Type Inference is a process of “compiler identifying the variable datatype based on the value provided”.

```
let anInt = 120;

let aDouble = 12.12;
```

Compiler treats anInt as Int and aDouble as Double based on the input value.

Rules to declare variables:

i. Specify the declaring variable is variable or constant using var or let

ii.You must specify the Data type or initialise a value to the variable

```
let aVar: Int  or let aVar = 10;
```

iii.Declared variable must contain value before it is used.

```
let aVar: Int = 10; //Specifying and Initialising aVar with value 10
var bVar: Int // Specifying to compiler that bVar will have a value
var cVar = bVar  // ERROR: bVar must contain a value before it is used

var anyObj: AnyObject = 20;
anyObj = "anyObj was holding Int(20), now it is holding String”

let aConst: Int = 20;
//aConst = 50; // ERROR: constant values can not be modified
```

String Interpolation (String Formatting):

`print()` is an output statement in swift.

String Interpolation is the process of formatting the string by substituting the value with in the string.

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples.”
//  I have 3 apples.

let fruitSummary = "I have \(apples + oranges) pieces of fruits.”

// I have 8 pieces of fruits.
```

Tuples

TUPLE **groups multiple similar/dissimilar values together.**

```
var aTuple = (“John",18,"California")

print("My name is :\(aTuple.0). I am \(aTuple.1) years old and I am from \(aTuple.2)”)

var (bookName, price) = ("Swift", 199.99)

print("Book name is:\(bookName) and price is: \(price)”)

var bookInfo = (bookName: "Swift", price:199.99)

print("Book name is:\(bookInfo.bookName) and price is: \(bookInfo.price)")
```

Operators

Definition: Operator is a symbol which performs operations on Operands

Ex:

```
5 + 5 = 10
5 - 5 = 0
1000 / 100 == 10
```

```
let a = 10, b = 20
let sum = a + b;
```

Here,

+ is an operator

a and b are operands

Swift Operators

Arithmetic Operators (+, -, *, /, %)
Logical Operators (&&, ||, !, >, <, >=, <=, == and !=)
Assignment & Compound Operators (=, +=, -=, *=, /=, %=)
Conditional Operator (?: Ex: Condition? X : Y)

NEW:

Range Operators:
Nil Coalescing Operator (??)
Typecasting Operator

```swift
var a = 10
var b = 20
// Arithmetic Operators
print("Addition is: \(a+b)") // 30
print("Subtraction: \(a-b)") // -10
print("Multiply: \(a*b)") // 200
print("Division: \(b/a)") // 2
print("Modulus: \(b%a))") // 0,
  Ex: i) 5 % 2 = 1, ii) 5 % 3 = 2
```


For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com

Decision Making Statements (if, if-else, switch)

These statements are used to make decision based on the condition(s)/expression(s) result.

Syntax:

```
if expression
{
    // True block statements
}
```

Note: Curly braces are mandatory even for single statement

Ex:

```
let a = 10, b = 20
if a < b
{
   print("B is greater than A");
}
```

if-else

```
if expression
{
    // True block statements
}
else
{
    // False block statements
}
```

Ex:

```
int a = 10; b = 20;
if a > b{
    print("a is greater than b")
}
else
{
    print("b is greater than a")
}
```

```swift
var a = 10, b = 20

// Comparison Operators (>, <, >=, <=, ==, != )
if a > b // false
{
    print("a is greater than b")
}
if a < b // true
{
    print("a is less than b")
}
if a >= b // false
{
    print("a is greater than or equals to b")
}
if a <= b // true
{
    print("a is less than or equals to b")
}
if a == b // false
{
    print("a and b are equal")
}
if a != b // true
{
    print("a and b are not equal")
}
```

```
// Logical Operators (&&, ||, !)
```

	&&	||	!
T,T	TRUE	TRUE	-
T,F	FALSE	TRUE	-
F,F	FALSE	FALSE	-
T	-	-	FALSE
F	-	-	TRUE

```
if a >= 10 && a <= 100 // true
{
    print("a value is with in 10 - 100 range")
}

if a == 10 || b == 10 // true
{
    print("Either a or b equal to 10")
}

if !(a == 10) // false
{
    print("a is not equals to 10")
}
```

Compound & Conditional Operators

```
var a = 10
let b = 20

let aTemp = a // aTemp is 10
a += 20 // a = a + 20   // 30
a -= 20 // a = a - 20   // 10
a *= 20 // a = a * 20   // 200
a /= 20 // a = a / 20   // 10
a %= 20 // a = a % 20   // 10 % 20 = 10

// Conditional Operator (Condition? X : Y)
let result = a != 0 ? a : b // 10
```

Control Statements

Control statements controls the execution flow of the program.

Swift provides following control statements.

- **for-in**
- **while**
- **repeat-while**

Syntax:

```
for identifier in range(closed range or half opened range) or collection
{
    // Block of statements to execute for true expression
}
```

```swift
// CLOSED RANGE FOR LOOP: Includes 1st and last element also
for _ in 1...5
{
    print("Hello");
}
//  Hello Hello Hello  Hello  Hello


for i in 1...5
{
    print(“Hello\(i)“);
}
//  Hello1 Hello2 Hello3  Hello4  Hello5



// HALF OPEN RANGE FOR LOOP: Doesn't include last element
for i in 1..<5 // 1-4
{
    print(i);
}
/* 1  2  3  4  */
```

FAST ENUMERATION: Process of iterating collections in a quick way. It is faster than regular for-loop

```
let anArray = [1,2,3,4,5];
for item in anArray
{
    print(item);
}
// 1  2  3  4  5
let aDictionary: [String : String] = ["Key1": "Value1", "Key2" : "Value2"];
for item in aDictionary
{
    print(item)
}
/*
    ("Key1", "Value1")
    ("Key2", "Value2")
*/
// Through Tuples
for (index , value) in aDictionary
{
    print(index)
    print(value)
}
/*
    Key1
    Value1
    Key2
    Value2
*/
```

```swift
var i = 0;
while i < 5
{
    print(i+=1);
}
//0, 1, 2, 3, 4

var j = 0
repeat{
    print(j)
    j += 1;
}while j<5;
// 0,1,2,3,4

let a = 10;
let b = 15;
if a < b // true
{
    print("a is less than b")
}
else
{
  print("a is not less than b")
}
```

Switch

Switch is a conditional statement, which is used to choose 1 choice / option among multiple options.

Ex:
Which school to choose
Which phone to purchase

To choose one among multiple, we need to use multiple if statements. switch makes our life easier without using if, if else ladders

NOTE:

- **Swift Switch statements are auto break**
- **Default case is must**

Switch Statement

```
var age = 18
switch age {
case 0:
    print(“You are just born");
case 6...15:
    print(“Go to school”)
case 16...22:
    print(“College time")
case 18:
    print(“Time to Vote”)
case 23,24,25,26:
    print(“Marriage Time”)
case 27...60:
    print(“Family responsibilities”)
case 100:
    print(“Get ready to say Bye Bye")
default:
    print(“You are free bird");
}
//You are eligible for vote
```

Using fallthrough

fallthrough statement execute the next case irrespective of case matching

```
age = 22
switch age {
case 1...17:
    print("You are not allowed to Vote");
case 21...120:
    print("You are eligible having Liquor")
    fallthrough
case 18:
    print("You are eligible for vote")
default:
    print("Your's is unknown case");
}
//You are eligible having Liquor
//You are eligible for vote
```

```swift
let i = 100
switch i
{
case 1...25:
    print("Value \(i) is between 1 to 25");
case 26..<50:
    print("Value \(i) is between 26 to 49");
case 50...75:
    print("Value \(i) is between 50 to 75");
case 100, 200: // Multiple values separated by
coma
    print("Value is here");
default:
    print("Default case is must in Swift's switch
statement");
}

// Value is here
```

```swift
let char:Character = "C";
switch char
{
case "A"..."Z":
    print("Value is between A and Z");
case "a"..."z":
    print("Value is between a and z");
case "*":
    print("The value is : *");
case "$":
    print("The value is : $");
default:
    print("Input doesn't match any case");
}
// Value is between A and Z
```

Compound Operators

Compound operators performs arithmetic assignment operations.
There are 5 compound operators such as += , -=, *=, /= and %=

```
var sum = 0
for i in 1…10  {
    sum = sum + i
}
```

```
var sum = 0
for i in 1…10  {
    sum += i
}
```

```
var a = 20, b = 10
a += b     //==> a = a + b    ==>  30
a -= b     //==> a = a - b     ==> 10
a *= b     //==> a = a * b     ==> 200
a /= b     //==> a = a / b      ==> 2
a %= b     //==> a = a % b    ==> 0
```

Conditional Operator (?:)

Conditional operator is short form of if condition It contain success value and failure value. It is ternary operator.

Syntax:

```
var result = condition ? success Value : Failure Value

let value = a < 0 ? 0 : a
```

```
var gender = 0 // Assume 0 is female, 1 is male
var eligibilityAge = 0

if gender == 0 {
    eligibilityAge = 18
} else {
    eligibilityAge = 21
}

var eligibilityAge = (gender == 0) ? 18 : 21
```

Type Casting

Typecasting is a process of converting one datatype into another type.

```
DesiredDatatype(sourceDatatypeValue);
```

```
let aInt = Int(10.5)
```

Swift provides two keywords to achieve type casting **is** & **as (* discussed further)**

For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com

Values are never implicitly converted to required type. You must explicitly convert them to desired type.

```
let a = 10;
let b = 12.12;

let c = a + b; // can’t perform binary operation between Int and a Float.
let d = a + Int(b); // 22
```

```
var aFloat: Float = 12.12
var aDouble: Double = 11.11

print(aFloat + aDouble) // Error Float + Double
print(aFloat + Float(aDouble)) // 23.23
```

Swift Naming Conventions

File Names: Follow **Capital Camel Case**
Car.swift, CustomTableView.swift

Class Name : Follow **Capital Camel Case**
Ex: MyClass, SuperClass, Vehicle, Car

Variable Name: Follow **Lower Camel Case**
Ex: aInt, aCar, aLorry ….

Method Name: Follow **Lower Camel Case**

func addTwoNumbers()
func sayHello()
….

Optionals

Optional Variable: A variable which is capable of holding a value or nil. An Optional variable may or may not contains a value

When to Use: When you are not sure that a variable definitely contains a value.

```
let aConst: Int = 10
aConst = nil // Error: nil cannot be initialised to specified type Int
```

Syntax:

```
var/let variableName: Datatype?
```

```
var anOptional: Int?
anOptional = 10
anOptional = nil
```

Ex:

```
let aStr = "134"
let a: Int = Int(aStr)

let aStr = "134S"
let a: Int? = Int(aStr)
```

Optional Unwrapping

Optional Unwrapping: Optional unwrapping is a process of extracting underlying value of an Optional.

```
var firstName: String = "Steve"
var lastName: String? = nil
var fullName = firstName + lastName // ERROR:
Optional must be unwrapped before it is used
```

i. Force Unwrapping
ii. Implicitly Unwrapping

For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com

i. Force Unwrapping : Process of confirming that optional variable has a value by using exclamation(!)

```
var firstName: String = "Steve"
var lastName: String? = nil
var fullName = firstName + lastName! // ERROR: Optional variable lastName expression has no value

var firstName: String = "Steve"
var lastName: String? = "Jobs"
var fullName = firstName + lastName!
```

NOTE: Optional variable must contain a value before it is used.

i. Implicitly Unwrapping : Process of confirming that optional variable has a value by using !

```
var firstName: String = "Steve"
var lastName: String! // Specifying to compiler that lastName is an optional variable, that will be containing a value other than nil.
var fullName = firstName + lastName // ERROR: Optional variable lastName expression has no value
```

```
var firstName: String = "Steve"
var lastName: String!
lastName = "Jobs"
var fullName = firstName + lastName
```

NOTE: Optional variable must contain a value before it is used.

Optionals Binding

Optional Binding:
Other than forced unwrapping, optional binding is a simpler and recommended way to unwrap an optional. You use optional binding to check if the optional contains a value or not. If it does contain a value, unwrap it and put it into a temporary constant or variable.

Generally we need to make sure the optional variable has a other than nil

```
if (lastName != nil)
{
    var fullName = firstName + lastName!
}
```

Using Optional Binding:

```
if let ln = lastName
{
    var fullName = firstName + ln
}
```

nil-coalescing Operator(??)

nil-coalescing:
nil-coalescing operator is used to extract the wrapped value of an optional. If the optional doesn’t contain a value, it provides a default value.

Syntax:

```
let wrappedValue = aOptional ?? default value
```

```
let aOptional: Int? = 10
let actualValue = aOptional ?? 1
print(actualValue) // 10
```

```
let aOptional: Int?
let actualValue = aOptional ?? 1
print(actualValue) // 1
```

Optional Chaining

The process of querying(accessing), calling properties, subscripts and methods on an optional that may be 'nil' is defined as optional chaining.

Optional chaining return two values:

i. If the optional contains a 'value' then calling its related property, methods and subscripts returns values.

ii. If the optional contains a 'nil' value all its its related property, methods and subscripts returns nil.

Since multiple queries to methods, properties and subscripts are grouped together failure to one chain will affect the entire chain and results in 'nil' value.

```swift
class Person : Object
{
    var name: String = "SteveJobs"
    var nickName: String?
}

var aPerson = Person()
let nickName: Int? = aPerson.nickName?.characters.count // nil

let nickName: Int = aPerson.nickName!.characters.count // ERROR

aPerson.nickName = "Steve"
let nickName: Int? = aPerson.nickName?.characters.count
```

guard keyword

guard is a new conditional statement that requires execution of exit from the current block if the condition isn't met.

Any new optional bindings created in a guard statement's condition are available for the rest of the function or block, and the mandatory else must exit the current scope, by using return to leave a function, continue or break within a loop

Syntax:

```
guard condition else
{
   return / continue / break / exit(0)
}
statement 1
statement 2
statement 3
. . . . . .
. . . . . .
. . . . . .
```

```
class Person : NSObject
{
    var name: String = "SteveJobs"
    var nickName: String?
}

var aPerson = Person()

guard aPerson.nickName != nil else
{
    print("Person has no nick name”)
    return (in methods)|continue / throw /break (in
loops)
}
print("Person has nick name use it further")
```

Thank You

For Offline / Online Training, reach me@ iPhoneDev1990@gmail.com