

XCODE





Welcome to Xcode

Version 7.3.1 (7D1014)



Get started with a playground

Explore new ideas quickly and easily.



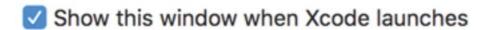
Create a new Xcode project

Start building a new iPhone, iPad or Mac application.



Check out an existing project

Start working on something from an SCM repository.



DEMO

- Si no tienen Xcode
 - https://swiftlang.ng.bluemix.net
 - http://iswift.org/playground
 - http://swiftstub.com/

Swift

LET VS VAR



let maximunNumberOfLoginAttempts = 10

var currentLoginAttempt = 0



LET VS VAR

```
if currentLoginAttemp < maximunNumberOfLoginAttempts {
    currentLoginAttempt += 1
    login()
}</pre>
```

maximunNumberOfLoginAttemps = 11

Cannot assign to value: 'maximunNumberOfLoginAttemps' is a 'let' constant

Cannot assign to value: 'maximunNumberOfLoginAttemps' is a 'let' constant

Fix-it Change 'let' to 'var' to make it mutable

PRINT

```
print(currentLoginAttempt)
// "0\n"
print("Intentos", currentLoginAttempt)
// "Intentos 0\n"
print("Intentos", currentLoginAttempt, "/", maximunNumberOfLoginAttempts)
// "Intentos 0 / 10\n"
print("1","2","3", separator: ",")
// "1,2,3\n"
print("no new line", terminator: "")
// "no new line"
print("2016","07","11", separator: "-", terminator: ";")
// "2016-07-11;"
```

ARRAY

```
let numbers = [9, 0, 1, 8, 2, 7, 3, 6, 4, 5]
// [9, 0, 1, 8, 2, 7, 3, 6, 4, 5]
numbers.count
// 10
numbers[0]
// 9
numbers_indexOf(1)
// 2
numbers contains (10)
// false
numbers_sort()
// [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers
// ???
```

ARRAY

```
var chars = ["S","I"]
// ["S", "I"]
chars.insert("W", atIndex: 1)
// ["S", "W", "I"]
chars append ("F")
// ["S", "W", "I", "F"]
chars = chars + ["T"]
// ["S", "W", "I", "F", "T"]
chars.sortInPlace()
// ["F", "I", "S", "T", "W"]
chars.joinWithSeparator("")
// FISTW
```

DICTIONARY

```
var dictionary = ["one": 1, "two": 2, "three": 3]
// ["one": 1, "three": 3, "two": 2]
dictionary["one"]
// 1
dictionary["four"] = 4
// 4
dictionary["five"]
// nil
```

- TIENEN PROPIEDADES PARA GUARDAR VALORES
- · PUEDEN TENER MÉTODOS QUE AGREGAN FUNCIONALIDAD
- · INICIALIZADORES CON VALORES/ESTADOS INICIALES
- PUEDEN SER EXTENDIDOS PARA AGREGAR FUNCIONALIDADES POR DEFECTO
- PUEDEN CONFORMAR PROTOCOLOS

CLASSES

- PERMITEN HERENCIA
- · CASTING
- · PUEDEN SER REFERENCIADAS DE MAS DE UN LUGAR A UNA SOLA INSTANCIA

```
struct Point {
    // structure definition goes here
}

class View {
    // class definition goes here
}
```

```
struct Point {
    var x = 0.0
    var y = 0.0
}

class View {
    var position = Point()
    var enable = true
}
```

```
struct Point {
    var x = 0.0
    var y = 0.0
}

class View {
    var position = Point(x: 100, y: 200)
    var enable = true
}
```

CLASSES

```
class View {
   var position = Point(x: 100, y: 200)
   var enable = true

   init(position: Point, enable: Bool) {
      self.position = position
      self.enable = enable
   }
}
let v = View(position: Point(x: 50, y: 200), enable: false)
```

PROPERTIES

```
class Window {
    var view = View()
let window = Window()
let view = window.view
view position = Point(x: 50, y: 100)
window.view.position.x
// 50
window.view.position.y
// 100
```

PROPERTIES

```
let window = Window()
let view = window.view
var point = view.position
point.x = -100
point y = -300
window.view.position.x
// ???
window.view.position.y
// ???
```

TYPES

TIPOS COMUNES

- > PRIMITIVOS: Int, String
- > CONTENEDORES: Array, Dictionary
- > DEFINIDOS POR EL USUARIO: Class, Structs

COMPILE TIME VS RUNTIME

STATIC TYPED

TYPE INFERCE

EXPLICIT VS IMPLICIT TYPE DECLARATION

STRONGLY TRYPED

OPTIONALS ?



SOME: HAY UN VALOR Y ES IGUAL A X NONE: NO HAY UN VALOR.

MIRAR DENTRO DE LA CAJA ! Y?

FORCED UNWRAPPING!

OPTIONAL CHAINING

OPTIONAL BINDING IF LET