Desarrollo en iOS

Día uno

Agenda

- El curso
- Instructores
- Formato
- Objetivos
- Swift!

El curso

- CocoaHeads Costa Rica
- Mucha demanda
- Pocas oportunidades
- Proveer herramientas

Instructores

- Esteban Torres
  - iOS Developer, Brewbot
  - Seis años de experiencia en iOS
  - @esttorhe
- Chris Jimenez
  - Senior Software Developer / Team Lead, SOIN
  - Ocho años de experiencia
  - @ChrisJimenezNat
- Leo Picado
  - Software Engineer, Gorilla Logic (iOS Developer wannabe)
  - Un año de experiencia
  - @leopyc

Formato

- Clase magistral
- Descanso
- Sesión práctica

Objetivos

- Salir de la zona de comfort
- Terminar con un app completo
- Aprender :)

Introducción

Desarrollo en iOS

- Primer iPhone 2006
- Objetive-C
- Manejo de memoria
- Punteros
- Círculo cerrado

Swift

- WWDC 2014
- Menor curva de aprendizaje
- Bootcamps
- Open Source
- Rápida adopción
- Lenguaje más popular en GitHub
    - Rust
    - Go
    - Ruby
    - PHP

Swift se usa en

- iOS
- OS X
- watchOS
- tvOS

Swift es

- Orientado a Objetos
- Propósito general
- Tipado fuerte
- Compilado

Pero antes

Lenguajes de Programación

- Propósito
- Tipado
- Implementación

Propósito

- General (GPL)
  - Muchos usos
  - No incluye entidades para un problema específico
  - PHP, JS
- Específicos al Dominio del Problema (DSL)
  - Contiene estructuras para un problema específico
  - Lenguajes para facturación, simulación de combate
  - HTML, SQL, Logo

Tipado

- Débil: las variables pueden tomar cualquier forma
  - JavaScript
  - PHP
- Fuerte: las variables tienen un solo tipo
  - Swift
  - Java

Implementación

- Compilado
  - Análisis
  - Síntesis
  - Generación de Código
- Intepretados
  - Analizado
  - Ejecutado

Swift es

- Orientado a Objetos
- Propósito general
- Tipado fuerte
- Compilado

Finalmente, Swift

Xcode

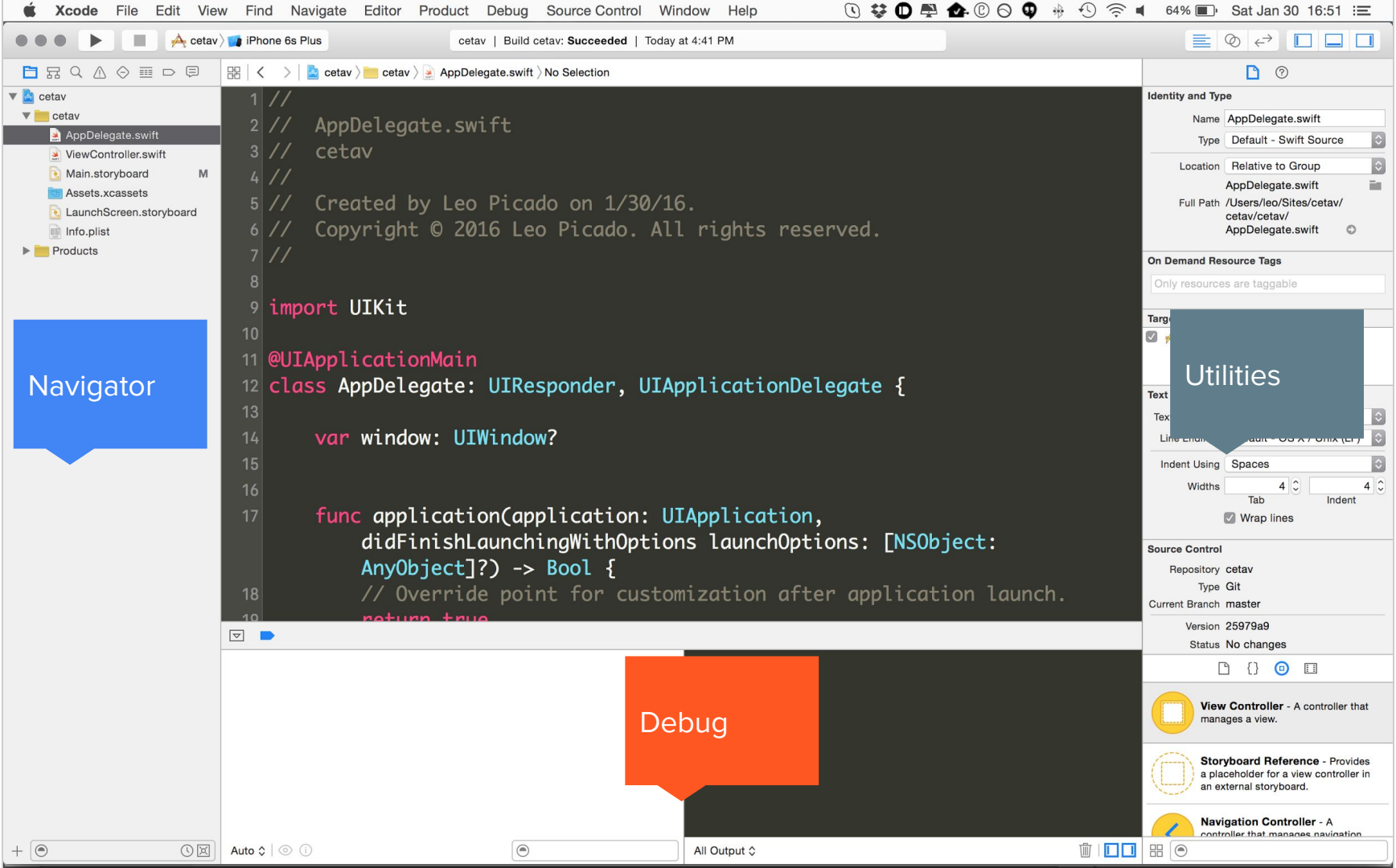
- IDE
- Apple
- C, Java, Python, Ruby, Objetive-C, Swift
- Integración con Git
- Gratis (mientras tenga MacOSX)
- Proyectos
- Espacios de trabajo

Proyectos en Xcode

- Nombre
- Organización
- Identificador
- Lenguaje
- Dispositivos
- Adicionales

Comandos básicos

- Compilar
  - Product > Build
  - ⌘ + B
- Ejecutar
  - Product > Run
  - ⌘ + C
- Limpiar
  - Product > Clean
  - ⇧ + ⌘ + K
- Buscar archivo
  - File > Open Quickly...
  - ⇧ + ⌘ + O


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
64% Sat Jan 30 16:51
cetav iPhone 6s Plus
cetav | Build cetav: Succeeded | Today at 4:41 PM
cetav
cetav
AppDelegate.swift
ViewController.swift
Main.storyboard M
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
cetav > cetav > AppDelegate.swift > No Selection
1 //
2 // AppDelegate.swift
3 // cetav
4 //
5 // Created by Leo Picado on 1/30/16.
6 // Copyright © 2016 Leo Picado. All rights reserved.
7 //
8
9 import UIKit
10
11 @UIApplicationMain
12 class AppDelegate: UIResponder, UIApplicationDelegate {
13
14 var window: UIWindow?
15
16
17 func application(application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [NSObject:
AnyObject]?) -> Bool {
18 // Override point for customization after application launch.
Navigator
Debug
Utilities
Identity and Type
Name AppDelegate.swift
Type Default - Swift Source
Location Relative to Group
AppDelegate.swift
Full Path /Users/leo/Sites/cetav/cetav/cetav/AppDelegate.swift
On Demand Resource Tags
Only resources are taggable
Indent Using Spaces
Widths 4 4
Tab Indent
Wrap lines
Source Control
Repository cetav
Type Git
Current Branch master
Version 25979a9
Status No changes
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.
Navigation Controller - A
Auto
All Output


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
63% Sat Jan 30 16:55
cetav iPhone 6s Plus
cetav | Build cetav: Succeeded | Today at 4:41 PM
cetav cetav AppDelegate.swift No Selection
Switches
//
// AppDelegate.swift
// cetav
//
// Created by Leo Picado on 1/30/16.
// Copyright © 2016 Leo Picado. All rights reserved.
//
import UIKit
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
var window: UIWindow?
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
// Override point for customization after application launch.
return true
}
func applicationWillResignActive(application: UIApplication) {
// Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
// Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates. Games should use this method to pause the game.
}

Paneles de Xcode

Navigator

- Lista de archivos
- Buscador de símbolos
- Errores y advertencias
- Breakpoints

Utilities

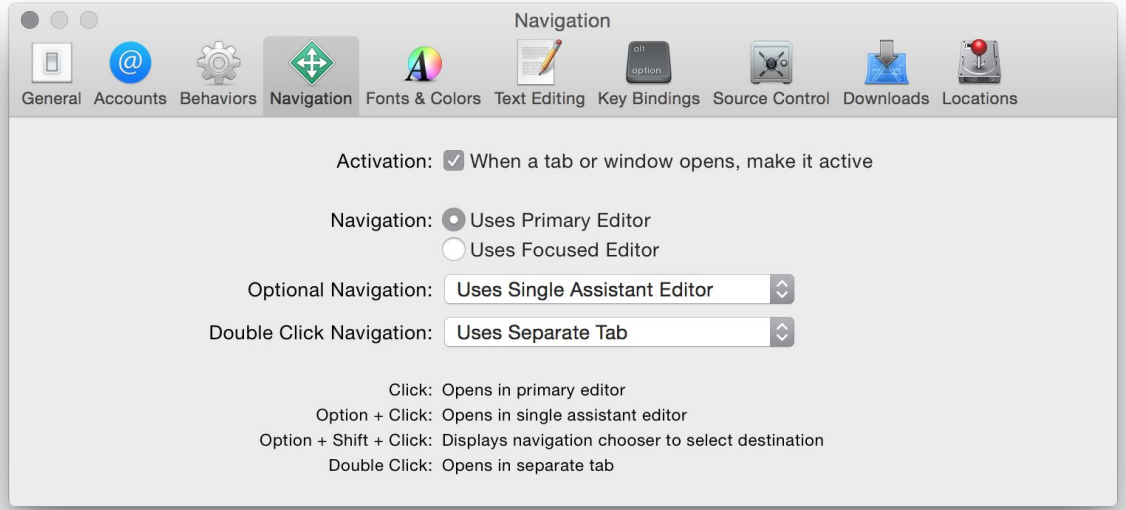
- Inspector
- Descripción
- Identidad
- Atributos
- Conexiones

Debug

- Variables
- Consola

Antes de seguir

- Xcode > Settings
- Navigation



Playgrounds

- Pruebas rápidas
- Compiladas y ejecutadas automáticamente
- Perfectos para aprender
- File > New > Playground

Hagamos un Nuevo Playground

- Xcode
- File > New > Playground

Operadores básicos

- + Suma
- - Resta
- * Multiplicación
- / División
- % Residuo
- = Asignación

Operadores de comparación

- == Igualdad
- != Desigualdad
- === Igualidad de indentidad
- > Mayor que
- < Menor que
- >= Mayor o igual
- <= Menor o igual
- ? Ternario

Más Operadores

- `&&` y lógico
- `||` or lógico
- `..` Rango cerrado
- `..<` Rango medio-abierto

Variables y Constantes

```
// (var|let) nombre:[tipo] = [valor]

// Variable: su valor cambiará dentro del programa
var cadenaDeTexto = "Hola Mundo!"

// Constante: su valor no cambiará
let numero = 4

let arregloDeTexto = ["uno", "dos", "tres", "catorce"]
```

Las anotaciones de tipos

El compilador infiere el tipo cuando se crea y asigna la variable

```
var cadenaDeTexto = "Hola Mundo!"
// Es igual a
var cadenaDeTexto:String = "Hola Mundo!"

let numero = 4
// Es igual a
let numero:Int = 4
```

Rompamos algunas cosas

```
// Válido
let constante = "Constante"
// Una constante no puede cambiar valor
constante = "nuevo valor"

// Válido
var numero = 5
// Conversión de tipo inválida
numero = "Cinco"
```

Estructuras de Control: `if`

```
let nombre: String

// No usamos ()'s
if algunaCondicion {
   nombre = "Ash"
} else {
   nombre = "Orta"
}

print(nombre)
```

Estructuras de Control: for

```
// No usamos ()'s
for var i = 0; i < 10; i++ {
   print("Hola: \(i)")
}
```

Estructuras de Control: for con rango

```
// El compilador crea el rango de 0 10
// Le asigna a i el valor del índice
for i in 0..<10 {
   print("Hola: \(i)")
}
```

Tipos de datos

- Texto: `Character`, `String` (
- Numerales: `Int`, `Float`, `Double`
- Booleano: `Boolean`
- Complejos
  - `Clases`
  - `Structs`
  - `Enums`

String: Operaciones Comunes 1/2

```
// Interpolaciones
let humano = "leo"
"Hola \(humano)."

// Reemplazos
let base = "Hola Humano"
base.stringByReplacingOccurrencesOfString("Hola", withString: "Adios")

// Verificar contenido
let profes = "Chris Esteban Leo"
profes.rangeOfString("Leo")
```

String: Operaciones Comunes 2/2

```
// Convertir a array usando " " como separador
let profesores = "Chris Esteban Leo"
profesores.componentsSeparatedByString(" ")

// Verificar contenido
let nombre = "leo"
"leo".isEmpty

// Remover caracteres en blanco
let nombreConEspacios = "  leo  "
nombreConEspacios.stringByTrimmingCharactersInSet(NSCharacterSet.
whitespaceAndNewlineCharacterSet())
```

Funciones

```
// Declaradas con func
// Función traducir:
// Recibe un parámetro: espanol de tipo String
// Retorna un String
func traducir(espanol: String) -> String {
   return "algo"
}
```

Funciones, parte dos

```
func traducir(espanol: String) -> String {
    // Switches
    switch espanol.lowercaseString {
    case "hola":
        return "Hello"
        // Sin break
    case "¿que hora es?":
        return "What time is it?"
    // Exhaustivos: cada posible escenario
    default:
        return ""
    }
}
```

El problema con nuestra función

- Poco contenido
- No falla cuando no maneja algo
- Debería fallar

Funciones, Parte Tres: El Ataque de los Opcionales

```
// Retornamos String si podemos
func traducir(espanol: String) -> String? {
    switch espanol.lowercaseString {
    case "hola":
        return "Hello"
    case "¿que hora es?":
        return "What time is it?"
    default:
        // Si no podemos, retornamos nil
        return nil
    }
}
```

String?

- Sabemos en que casos tenemos traducciones
- El resto de casos deben fallar y no retornar un String vacío (“”)
- `String != String?`

Consumiendo opcionales

```
let stringTraducido = traducir("hola")
print("Resultado \(stringTraducido)")
// "Resultado Optional("Hello")\n"
```

Opcionales

- Demuestran posible falta de valor
- Debemos extraer su valor
- Denotados como `Tipo + ?`

Obteniendo el valor de un opcional

```
// Si al llamar a traducir(“Hola”) se retorna un valor
if let stringTraducido = traducir("Hola") {
// Asígnelo a stringTraducido
   print("Traducido como '\(stringTraducido)'")
} else {
// Si se retorna nil
   print("Favor contactar a soporte técnico, la traducción fallo.")
}
```

Funciones: El Despertar de los Parámetros

```
// Función con más de un parámetro
// Dividido por comas
func saludo(nombre: String, dia: String) -> String {
   return "Hola \(nombre), hoy es \(dia)."
}

// Invocado con el nombre del parámetro
saludo("Esteban", dia: "Lunes")
```

Valores predeterminados

```swift
// Función con valores predeterminados
func segundoSaludo(nombre:String = "Chris", dia:String = "Martes") -> String {
   // Retornamos el llamado a la función anterior
   return saludo(nombre, dia: dia)
}

// Sin parámetros
segundoSaludo() // Hola Chris, hoy es Martes.

// Con solo un parámetro
segundoSaludo("Jose") // Hola Jose, hoy es Martes.

// Con ambos parámetros
segundoSaludo("Leo", dia: "Miércoles") // Hola Leo, hoy es Miércoles.
```

Colecciones

- Arrays: [ ]
  - Ordenados
- Diccionarios: { }
  - JavaScript Objetos
  - PHP: Array Asociativo
- Sets
  - Conjuntos sin orden

Arrays: 1/3

```
// Creamos un arreglo vacio, contiene Strings
var arreglo = [String]()

// Agregamos un elemento
arreglo.append("hola")

// El primer elemento es un Opcional
if let primeraEntradaPre = arreglo.first {
   print("Iniciamos con: \(primeraEntradaPre)")
} else {
   // En caso de no existir
   print("Arreglo vacio")
}
```

Arrays: 2/3

```
// Acceder directamente un elemento
print(arreglo[0])

// Iterando sobre los elementos
for entrada in arreglo {
   print(entrada)
}
```

Arrays: 3/3

```
// Revisamos si tiene contenido
if !arreglo.isEmpty {
   // Total entradas
   print("Count: \(arreglo.count)")
   // Removemos el primer elemento
   arreglo.removeAtIndex(0)
}

if let primeraEntradaPost = arreglo.first {
   print("Iniciamos con: \(primeraEntradaPost)")
} else {
   print("Arreglo vacio")
}
```

Diccionarios: 1/3

```
// Creamos un diccionario vacio, contendrá Strings como llaves: Ints como valores
var diccionario = [String:Int]()

// Agregamos un elemento
diccionario["hola"] = 1

// El primer elemento es Opcional
if let primeraEntradaPre = diccionario.first {
   print("Llave: \(primeraEntradaPre.0). Valor \(primeraEntradaPre.1)")
} else {
   // En caso de no existir
   print("Diccionario vacio")
}
```

Diccionarios: 2/3

```
// Acceder directamente un elemento
print(diccionario["hola"])

// Iterando sobre los elementos
for entrada in diccionario {
   print(entrada)
}

// Iterando sobre los elementos, separados como llave, valor
for (key, value) in diccionario {
   print("\(key) : \(value)")
}
```

Diccionarios: 3/3

```
if !diccionario.isEmpty {
   // Total entradas
   print("Count: \(diccionario.count)")
   // Removemos el único elemento
   diccionario.removeValueForKey("hola")
}

if let primeraEntradaPost = diccionario.first {
   print("Llave: \(primeraEntradaPost.0). Valor \(primeraEntradaPost.1)")
} else {
   print("Diccionario vacio")
}
```

Descanso

Ejercicios: 1 Variables/Constantes

1. Declare una variable llamada lenguaje y asignele el valor “Swift”
2. Declare una constante llamada postreFavorito y asignele el nombre de su postre favorito

Ejercicios: 2 Sintaxis y Convenciones 1/2

1. ¿Cuál de las siguientes se usa para declarar una constante?
   a. `const`
   b. `final`
   c. `let`
   d. `var`
2. Se debe siempre usar “let” a menos que ocupe cambiar o mutar el valor más adelante, en ese caso usamos “var”
   a. `Verdadero`
   b. `Falso`
3. Iniciamos los comentarios de una sola línea con los siguientes caracteres
   a. `(ノ°□°)ノ ┻━┻`
   b. `/`
   c. `\\`
   d. `//`

Ejercicios: 3 Sintaxis y Convenciones 2/2

1. ¿Cúal de los siguientes símbolos representa el operador de asignación?
   a. `=`
   b. `!`
   c. `?`
   d. `;`
2. ¿Cúal de los siguientes es un nombre válido para las variables, siguiendo la sintaxis y convenciones de código vistas en clase?
   a. `1Variable`
   b. `algunaVariable`
   c. `alguna Variable`
   d. `algunavariable`

Ejercicios: 4 Strings

1. Declare una constante llamada `nombre` y asignele un String con su nombre
2. Declare una constante llamada `saludo` y asígnele el valor de la interpolación de “Hola amigo “ y el valor almacenado en la constante `nombre`. El resultado final deberá ser algo como “Hola amigo Esteban.” Asegurese de usar un punto al final de su saludo
3. Declare una constante llamada `saludoFinal` y concatene el valor de saludo con el String “¿Cómo está?” El ejemplo anterior debe leerse como “Hola amigo Esteban. ¿Cómo está?”

Ejercicios: 5 Tipos 1/4

Vamos a trabajar en un ejemplo de un libro dentro del contexto de una librería

1. Declare una constante `titulo` y asígnele el string “Bailando con dragones”
2. Declare una variable `calificacion` y asígnele el valor 7.5
3. Declare una variable `disponible` y asignele el valor booleano de false

Ejercicios: 6 Tipos 2/4

Pretendamos que vamos a construir una calculadora

1. Declare dos constantes `primerValor` y `segundoValor` y asigneles algún valor que corresponda a un `Int`
2. Declare una constante llamada `producto` y asignele el valor de la multiplicación del `primerValor` y el `segundoValor`
3. Usando interpolación de strings, cree una constante llamada `resultado` que describa la operación que acabamos de realizar, si `primerValor` tenía el valor de 2 y `segundoValor` el valor de 4, la constante debe contener “El producto de multiplicar 2 por 4 es 8”.

Ejercicios: 7 Tipos 3/4

1. Los números de coma flotante en Swift se representan con el siguiente tipo
    a. Double
    b. Int
    c. Bool
    d. Float
    e. Double y Float
2. ¿Cúal de los siguientes no es un tipo de número?
    a. Int
    b. Float
    c. Double
    d. Bool
    e. Ninguno de los anteriores

Ejercicios: 8 Tipos 4/4

1. Falso o verdadero, un valor booleano se representa como Int con los siguientes valores: 1, 2
    a. Falso
    b. Verdadero
2. ¿Cuál de las siguientes afirmaciones es correcta?
    a. La concatenación le permite mezclar tipos distintos de datos en un String
    b. La interpolación le permite mezclar tipos distintos de datos en un String
3. El valor binario para true es
    a. 1
    b. 0

Ejercicios: 9 Operadores 1/6

1. Cree una constante `dividendo`, asígnele `200`
2. Cree una constante `divisor`, asígnele `5`
3. Usando el operador de residuo guarde el valor dado por `dividendo` y `divisor` en una constante llamada `resultado`
4. Cuando el residuo obtenido es 0, significa que el `dividendo` es un múltiplo del `divisor`. Compare el resultado contra 0 usando el operador de igualdad y asignele el resultado en una constante llamada `esMultiplo`

Ejercicios: 10 Operadores 2/6

1. Cree la constante `unaOperacion`, asígnele `20 + 400 % 10 / 2 - 15`
2. Cree la constante`otraOperacion`, asígnele `52 * 27 % 200 / 2 + 5`
3. Cree una nueva constante booleana `esMayor` y sin volver a calcular las operaciones verifique si el valor resultante de `unaOperacion` es mayor al de `otraOperacion`. Use el operador mayor o igual qué

Ejercicios: 11 Operadores 3/6

1. ¿Cúal es el valor de y? Si y está definida por 25 - 5 * 2 + 5
    a. 45
    b. 20
    c. -10
    d. 10
2. ¿Cúal es el valor de x? Si x está definida por 100 / 5 + 5
    a. 110
    b. 25
    c. 10
    d. 15
3. ¿Cúal es el valor de z? Si z está definida por 5 + 5 - 5 * 2
    a. 0
    b. 5
    c. 10

Ejercicios: 12 Operadores 4/6

1. ¿Cúal es el valor de a? Si a está definida por (2 + 2) * 2 + 2
    a. 16
    b. 10
    c. 8
2. ¿Cúal es el valor del residuo de 15 % 7?
    a. 1
    b. 8
    c. 2
    d. 0

Ejercicios: 13 Operadores 5/6

Vamos a pretender que creamos un juego, cada vez que un usuario completa un objetivo, obtiene un punto.

1. Cree una variable `marcadorInicial` y asígnele el valor de 8
2. Incremente en uno el valor de marcadorInicial y asigne el valor a una constante `marcadorTotal`
3. Declare una constante `juegoTerminado` y asígenele el resultado de la comparación de `marcadorTotal` con 10. Si el marcador es 10 el jugador ganó, si no lo es, perdió. Use el operador de desigualdad

Ejercicios: 14 Operadores 6/6

1. ¿Cúal es el valor de `gameOver` si se declara como `!false`?
   a. True
   b. False
2. ¿Cúal es el valor de `marcadorTotal` si:

```
let marcador = 100
let marcadorTotal = -marcador
```

   a. 100
   b. -100
   c. 99
   d. 0

Referencias

- Estadísticas de Swift en GitHub
- Curso de "Swift at Artsy"
- The Swift Programming Language
- Treehouse.com - Swift 2.0 Basics