Welcome to Swift

http://activ8conf.jonathanhooper.net

What is Swift?

- A statically typed, object oriented programming language
- Programming language for iOS, watchOS, macOS, and tvOS
- “Objective-C without the C”
- Open source

Where do I get it?

Xcode

Swift Playgrounds

*Linux Users: https://swift.org/download/#releases

Language Features

Constants and variables

```
// variable
var a = 5
// constant
let b = 6
```

```
// type annotation
let a: Int = 4
let c: String = "hello!"
```

Optionals

```
var optionalString: String? = "this could be nil. who knows? maybe?"
optionalString = nil  // no worries here

var nonoptionalString: String = "this cannot be nil"
nonoptionalString = nil  // this will explode
```

Optionals

```
var optionalInt: Int? = 5
optionalInt! + 5  // no worries. returns 10

optionalInt = nil
optionalInt! + 5  // this will explode.
```

Optionals

```
var unwrappedOptionalInt: Int! = 5
unwrappedOptionalInt + 5  // no worries. returns 10

unwrappedOptionalInt = nil
unwrappedOptionalInt + 5  // this will explode.
```

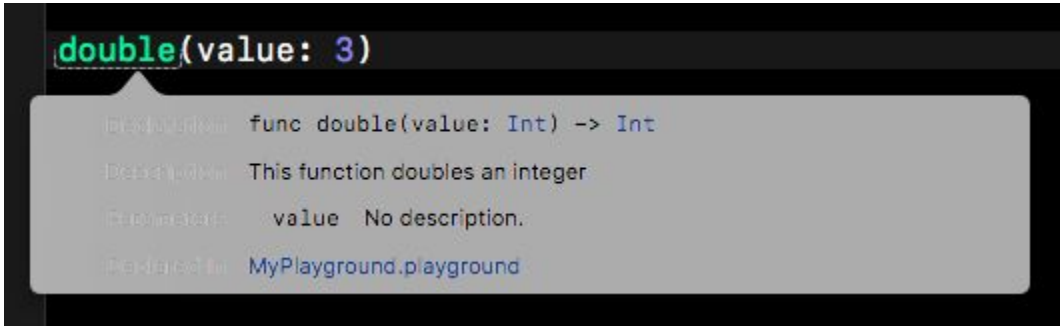
Comments

```
// This is a single line comment

/* This is a multiline comment.
   It spans many lines */
```

Documentation Comments

```
/**
 This function doubles an integer
*/
func double(value: Int) -> Int {
    return value * 2
}

/// This function halves an integer
func halve(value: Int) -> Int {
    return value / 2
}
```

Documentation Comments



Numbers

```
let integer: Int = 5
let double: Double = 5.0

let unsignedInteger: UInt = 3

let int64Bit: Int64 = 6
let int32Bit: Int32 = 7

let float: Float = 2.7
```

Strings

```
let helloWorld = "Hello, world!"
let characterH: Character = "H"

let mrBond = "James Bond"
let goodbyeMrBond =  "Goodbye, \(mrBond)"
```

Operators

Operators - Assignment

```
var a = 5  // a now equals 5
```

Operators - Arithmetic

```
1 + 2       // equals 3
3 - 4       // equals -1
5 * 6       // equals 30
7.0 / 8.0   // equals 0.875

"hello, " + "world"  // equals "hello, world"
```

Operators - Comparison

```
1 == 0   // false
2 != 1   // true
3 > 2    // true
4 < 3    // false
5 >= 4   // true
6 <= 5   // false
```

Operators - Logic

```
!true           // false
true || false   // true
true && false   // false
```

Even more operators

```
// The nil-coalescing operator
a ?? b
// The closed-range operator
1...5
// The half-open range operator
1..<5
// The ternary conditional operator
question ? "It's true!" : "It's false!"
// The remainder operator (different from a formal modulus operator, look it up!)
6 % 4
// The bitwise AND operator
0b11111100 & 0b00111111
```

Custom Operators

```
infix operator <!!>

extension Int {
  static func <!!>(a: Int, b: Int) -> Int {
    return (a + b) * 42
  }
}

4 <!!> 5  // equal 378, or (4 + 5) * 42
```

Collections

- Arrays
- Dictionaries
- Sets
- ...and more

Collections - Arrays

```
var arrayOfInts = [1, 2, 3]
var explicitArrayOfInts: [Int] = [4, 5, 6]
var emptyArrayOfInts = [Int]()

arrayOfInts[1] // returns 2
```

Collections - Arrays

```
var arrayOfInts = [1, 2, 3]
arrayOfInts.index(of: 3)    // returns 2
arrayOfInts.max()           // returns 3
arrayOfInts.min()           // returns 1
arrayOfInts.append(4)       // now equals [1, 2, 3, 4]
arrayOfInts.count           // returns 4
arrayOfInts.remove(at: 2)   // now equals [1, 2, 4]
arrayOfInts.first           // returns 1
arrayOfInts.last            // returns 4
```

Collections - Dictionaries

```
var numbers = [
  "one": 1,
  "two": 2,
  "three": 3,
]
var emptyDictionary = [String: Float]()
var explicitDictionary: [Character: String] = [
  "a": "ay",
  "b": "bee",
  "c": "see"
]

numbers["one"] // returns 1
```

Collections - Dictionaries

```
var numbers = [
  "one": 1,
  "two": 2,
  "three": 3,
]

numbers.count                                  // 3
numbers.isEmpty                                // false
numbers.removeValue(forKey: "two")             // key/value for "two" removed
numbers.updateValue(4, forKey: "three")        // numbers["three"] now equals 4
numbers.removeAll()                            // numbers now equals [:]
```

Control Flow

- If/Else
- Guard
- While loops
- ~~For loops~~
- For-in loops
- Switch Statements

If/Else

```
var number = 5

if number == 5 {
  // do something
}
```

If/Else

```
var number = 5

if number > 5 {
  // do something
} else {
  // do something else
}
```

If/Else

```
var number = 5

if number > 5 {
  // do something
} else if number == 5 {
  // do something else
} else {
  // do something... elser...?
}
```

If/Else

```
var optionalNumber: Int? = 5

if let optionalNumber = optionalNumber {
  // optionalNumber is now available as an unwrapped optional
   print(optionalNumber)
} else {
  // optionalNumber is nil
  print("optionalNumber is nil")
}
```

Guard

```
func fibonacci(at number: Int) -> Int {
  guard number > 1 else {
    return number
  }

  return fibonacci(at: number - 1) + fibonacci(at: number - 2)
}

fibonacci(at: 7)   // return 13
```

While loops

```
var counter = 0

while counter < 5 {
  print(counter)
  counter += 1
}
```

~~For loops~~

For-in loops

```
for number in [0, 1, 2, 3, 4] {
  print(number)
}

for number in 0..<5 {
  print(number)
}

for character in "01234".characters {
  print(character)
}
```

For-in loops

```
[0, 1, 2, 3, 4].forEach { number in
  print(number)
}

(0..<5).forEach { number in
  print(number)
}

"01234".characters.forEach { character in
  print(character)
}
```

Switch statements

```
var number = 3

switch number {
case 0:
    print("zero")
case 1:
    print("one")
case 2:
    print("two")
case 3:
    print("three")
default:
    print("Unrecognized number")
}
```

Functions

```
func sayHello() {
  print("Hello")
}
sayHello()  // prints "Hello"
```

Functions

```
func sayHello(name: String) {
  print("Hello, " + name)
}
sayHello(name: "Chuck Norris")   // prints "Hello, Chuck Norris"
```

Functions

```
func sayHello(to name: String) {
  print("Hello, " + name)
}
sayHello(to: "Bruce Lee")  // prints "Hello, Bruce Lee"
```

Functions

```
func sayHello(_ name: String) {
  print("Hello, " + name)
}
sayHello("Ip Man")  // prints "Hello, Ip Man"
```

Functions

```
func helloString(for name: String) -> String {
  return "Hello, " + name
}
let string = helloString(for: "Kung Fury")
print(string)   // prints "Hello, Kung Fury"
```

Enums

```
enum PrimateType {
  case Monkey
  case Gorilla
  case Human
}
```

Enums

```
var primateType = PrimateType.Gorilla

switch primateType {
case .Monkey:
    print("ooh ooh ahh ahh")
case .Gorilla:
    print("*gorilla noise*")
case .Human:
    print("Lorem Ipsum")
}
```

Structs

```
struct Primate {
  let name: String
  let type: PrimateType

  func makeANoise() {
    switch type {
    case .Monkey:
      print("ooh ooh ahh ahh")
    case .Gorilla:
      print("*gorilla noise*")
    case .Human:
      print("Lorem Ipsum")
    }
  }
}
```

Structs

```
let harambe = Primate(name: "Harambe", type: .Gorilla)
print(harambe.name)    // prints Harambe
harambe.makeANoise()   // prints *gorilla noise*
```

Classes

```swift
class Primate {
  let name: String
  let type: PrimateType

  init(name: String, type: PrimateType) {
    self.name = name
    self.type = type
  }

  func makeANoise() {
    switch type {
    case .Monkey:
      print("ooh ooh ahh ahh")
    case .Gorilla:
      print("*gorilla noise*")
    case .Human:
      print("Lorem Ipsum")
    }
  }
}
```

Classes

```
let batman = Primate(name: "Christian Bale", type: .Human)
print(batman.name)    // prints Christian Bale
batman.makeANoise()   // prints Lorem Ipsum
```

Structs

- Define properties to store data
- Define methods to add functionality
- Define initializers

Classes

- Define properties to store data
- Define methods to add functionality
- Define initializers
- Inheritance
- Type casting
- Deinitialization

Classes & Structs - self keyword

```
struct Primate {
  // ...

  func firstName() -> String? {
    return self.name.components(separatedBy: " ").first
  }
}

let batman = Primate(name: "Christian Bale", type: .Human)

if let name = batman.firstName() {
  print(name)  // prints Christian
} else {
  print("Batman doesn't have a name")
}
```

Demo

Where to go from here?

- Swift documentation: https://swift.org/documentation
- Apple's iOS documentation: https://developer.apple.com/develop/
- Stanford OCW - CS 193P: http://web.stanford.edu/class/cs193p
- Ray Wenderlich: https://www.raywenderlich.com/
- NSScreencast: http://nsscreencast.com/

Questions?