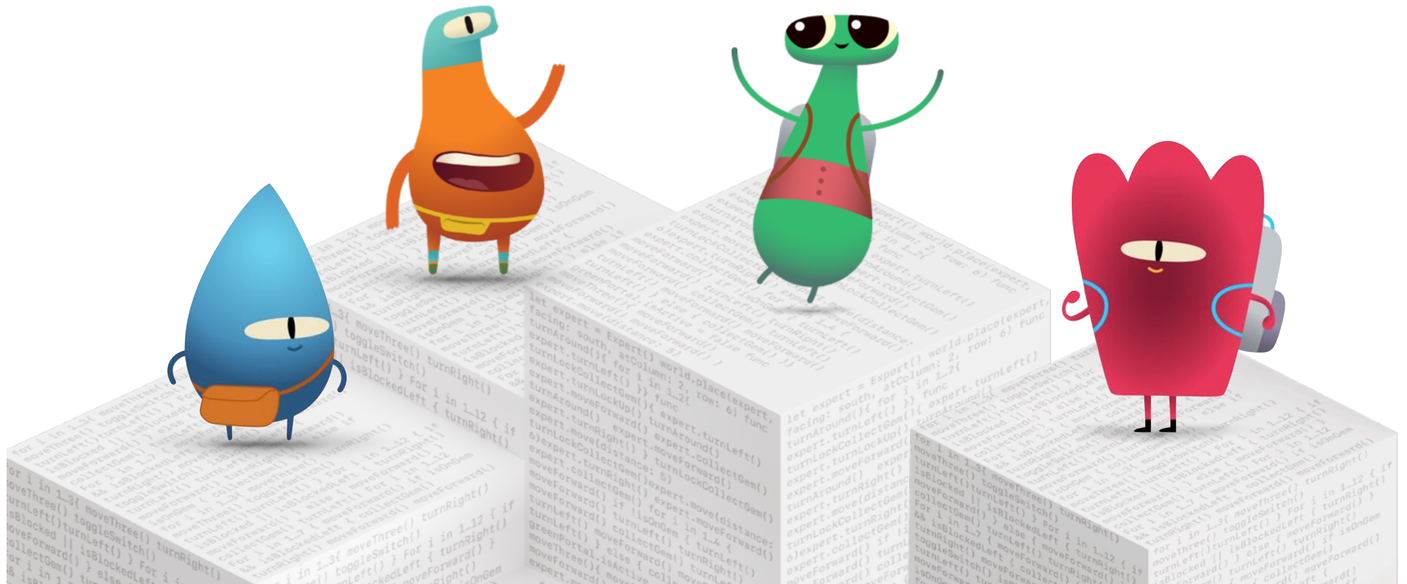
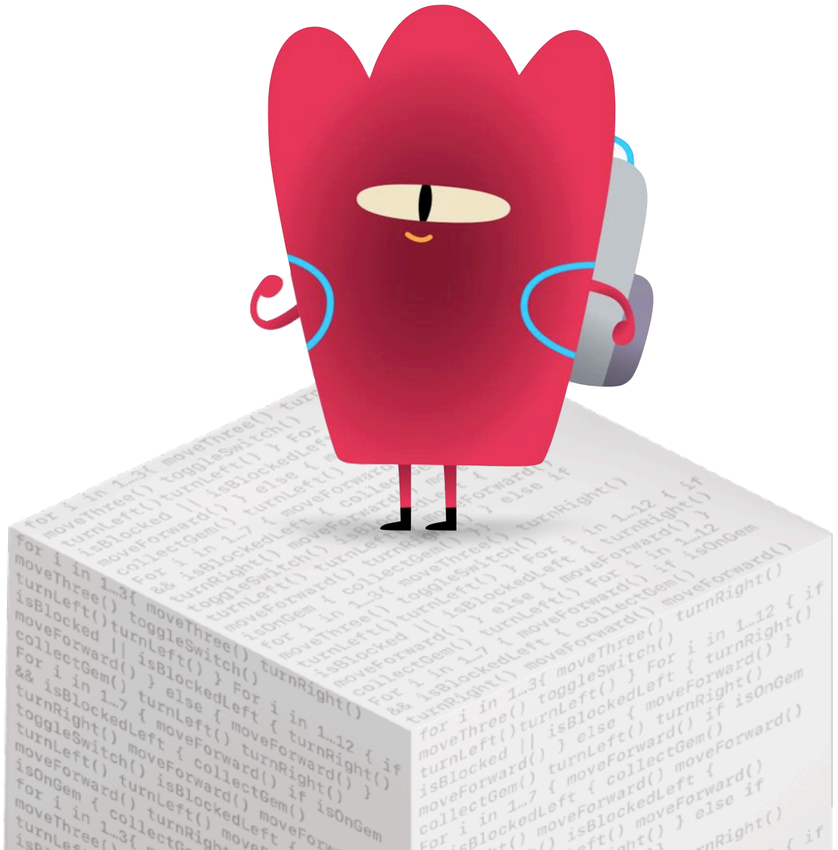
Swift Coding Club

Special Session



Session 2

More Swift Fundamentals



Let's Recap

More Swift Fundamentals

let vs var

Let's Recap

More Swift Fundamentals

Which one is the correct naming practice?

HelloWorld

helloWorld

helloworld

Let's Recap

More Swift Fundamentals

What represents full numbers

Double

Char

String

Int

Float

Let's Recap

More Swift Fundamentals

What represents decimals

Double

Char

String

Int

Float

Let's Recap

More Swift Fundamentals

What's the difference?

Double

15 decimal places accuracy

Float

6 decimal place accuracy

Let's Recap

More Swift Fundamentals

Then, what type does swift default if we create this variable

```
var number = 15.1234
```

Double

Float

Let's Recap

More Swift Fundamentals

How do we separate large digits of number?

```
var largeUglyNumber = 1000000000
```

Let's Recap

More Swift Fundamentals

Underscores!

```
var largeUglyNumber = 1_000_000_000
```

Let's Recap

More Swift Fundamentals

Then, how to do inline operations

Adding 13 to 5

```
var number = 13
```

Let's Recap

More Swift Fundamentals

Then, how to do inline operations

Adding 13 to 5

```
var number = 13
number += 5
```

Let's Recap

More Swift Fundamentals

Tell me what each logical operators do

==

>

<=

&&

!=

||

>=

<

!

Let's Recap

More Swift Fundamentals

What will this statement print out?

```
var statement1 = true
var statement2 = false
```

```
if statement1 && statement2 {
   print("hello")
} else  {
   print("huh")
}
```

Let's Recap

More Swift Fundamentals

What will this statement print out?

```
var statement1 = true
var statement2 = false
```

```
if statement1 && statement2 {
   print("hello")
} else  {
   print("huh")
}
```

Let's Recap

More Swift Fundamentals

What will this statement print out?

```
var statement1 = true
var statement2 = false
```

```
if statement1 == !statement2 {
   print("hello")
} else  {
   print("huh")
}
```

Let's Recap

More Swift Fundamentals

What will this statement print out?

```
var statement1 = true
var statement2 = false
```

```
if statement1 == !statement2 {
   print("hello")
} else  {
   print("huh")
}
```

Structures

Structures

```
struct Person {
  var name: String
}
```

Capitalize type names

Use lowercase for property names

Structures

Accessing property values

```
struct Person {
  var name: String
}

let person = Person(name: "Jasmine")
print(person.name)
```

```
Jasmine
```

Structures
Adding functionality

```
struct Person {
  var name: String

 func sayHello() {
    print("Hello there! My name is \(name)!")
  }

}

let person = Person(name: "Jasmine")
person.sayHello()
```

```
Hello there! My name is Jasmine!
```

Instances

```
struct Shirt {
  var size: String
  var color: String
}

let myShirt = Shirt(size: "XL", color: "blue")

let yourShirt = Shirt(size: "M", color: "red")
```

```
struct Car {
  var make: String
  var year: Int
  var color: String

  func startEngine() {...}

  func drive() {...}

  func park() {...}

  func steer(direction: Direction) {...}
}

let firstCar = Car(make: "Honda", year: 2010, color: "blue")
let secondCar = Car(make: "Ford", year: 2013, color: "black")

firstCar.startEngine()
firstCar.drive()
```

Initializers

```
let string = String.init() // ""
let integer = Int.init() // 0
let bool = Bool.init() // false
```

Initializers

```
var string = String() // ""
var integer = Int() // 0
var bool = Bool() // false
```

Initializers

Default values

```
struct Odometer {
  var count: Int = 0
}

let odometer = Odometer()
print(odometer.count)
```

```
0
```

Initializers

Memberwise initializers

```
let odometer = Odometer(count: 27000)
print(odometer.count)
```

```
27000
```

Initializers

Memberwise initializers

```
struct Person {
  var name: String
}
```

Initializers

Memberwise initializers

```
struct Person {
  var name: String

  func sayHello() {
    print("Hello there!")
  }
}

let person = Person(name: "Jasmine") // Memberwise initializer
```

Initializers

Memberwise initializers

```
struct BankAccount {
  var accountNumber: Int
  var balance: Double = 0
}

let newAccount = BankAccount(accountNumber: 123, balance: 0)
let transferredAccount = BankAccount(accountNumber: 123)
```

```swift
struct Shirt {
  let size: String
  let color: String
}

let myShirt = Shirt(size: "XL", color: "blue") // Memberwise initializer

struct Car {
  let make: String
  let year: Int
  let color: String
}

let firstCar = Car(make: "Honda", year: 2010, color: "blue") // Memberwise initializer

struct Bird {
  let canFly: Bool = true
  let weight: Double
  let color: String
}

let seagull = Bird(weight: 1.5, color: "white") // Memberwise initializer
let ostrich = Bird(canFly: false, weight: 200.0, color: "white") // Memberwise initializer
```

Initializers

Custom initializers

```
struct Temperature {
  var celsius: Double
}

let temperature = Temperature(celsius: 30.0)
```

```
let fahrenheitValue = 98.6
let celsiusValue = (fahrenheitValue - 32) / 1.8

let newTemperature = Temperature(celsius: celsiusValue)
```

```
struct Temperature {
  var celsius: Double

  init(celsius: Double) {
    self.celsius = celsius
  }

  init(fahrenheit: Double) {
    celsius = (fahrenheit - 32) / 1.8
  }
}

let currentTemperature = Temperature(celsius: 18.5)
let boiling = Temperature(fahrenheit: 212.0)

print(currentTemperature.celsius)
print(boiling.celsius)

18.5
100.0
```

Instance methods

```
struct Size {
  var width: Double
  var height: Double

  func area() -> Double {
    width * height
  }
}

var someSize = Size(width: 10.0, height: 5.5)

let area = someSize.area() // Area is assigned a value of 55.0
```

Mutating methods

```
struct Odometer {
  var count: Int = 0 // Assigns a default value to the 'count' property.
}
```

Need to

- Increment the mileage
- Reset the mileage

```
struct Odometer {
  var count: Int = 0 // Assigns a default value to the 'count' property.

  mutating func increment() {
    count += 1
  }

  mutating func increment(by amount: Int) {
    count += amount
  }

  mutating func reset() {
    count = 0
  }
}

var odometer = Odometer() // odometer.count defaults to 0
odometer.increment() // odometer.count is incremented to 1
odometer.increment(by: 15) // odometer.count is incremented to 16
odometer.reset() // odometer.count is reset to 0
```

Computed properties

```
struct Temperature {
  let celsius: Double
  let fahrenheit: Double
  let kelvin: Double
}

let temperature = Temperature(celsius: 0, fahrenheit: 32, kelvin: 273.15)
```

```swift
struct Temperature {
  var celsius: Double
  var fahrenheit: Double
  var kelvin: Double

  init(celsius: Double) {
    self.celsius = celsius
    fahrenheit = celsius * 1.8 + 32
    kelvin = celsius + 273.15
  }

  init(fahrenheit: Double) {
    self.fahrenheit = fahrenheit
    celsius = (fahrenheit - 32) / 1.8
    kelvin = celsius + 273.15
  }

  init(kelvin: Double) {
    self.kelvin = kelvin
    celsius = kelvin - 273.15
    fahrenheit = celsius * 1.8 + 32
  }
}

let currentTemperature = Temperature(celsius: 18.5)
let boiling = Temperature(fahrenheit: 212.0)
let freezing = Temperature(kelvin: 273.15)
```

```
struct Temperature {
  var celsius: Double

  var fahrenheit: Double {
    celsius * 1.8 + 32
  }
}

let currentTemperature = Temperature(celsius: 0.0)
print(currentTemperature.fahrenheit)

32.0
```

Property observers

```
struct StepCounter {
    var totalSteps: Int = 0 {
        willSet {
            print("About to set totalSteps to \(newValue)")
        }
        didSet {
            if totalSteps > oldValue  {
                print("Added \(totalSteps - oldValue) steps")
            }
        }
    }
}
```

Property observers

```
var stepCounter = StepCounter()
stepCounter.totalSteps = 40
stepCounter.totalSteps = 100
```

```
About to set totalSteps to 40
Added 40 steps
About to set totalSteps to 100
Added 60 steps
```

Type properties and methods

```
struct Temperature {
  static var boilingPoint = 100.0

  static func convertedFromFahrenheit(_ temperatureInFahrenheit: Double) -> Double {
   (((temperatureInFahrenheit - 32) * 5) / 9)
  }

}

let boilingPoint = Temperature.boilingPoint

let currentTemperature = Temperature.convertedFromFahrenheit(99)

let positiveNumber = abs(-4.14)
```

Copying

```
var someSize = Size(width: 250, height: 1000)
var anotherSize = someSize

someSize.width = 500

print(someSize.width)
print(anotherSize.width)
```

```
500
250
```

self

```
struct Car {
  var color: Color

  var description: String {
   "This is a \(self.color) car."
  }
}
```

self

When not required

Not required when property or method names exist on the current object

```
struct Car {
  var color: Color

  var description: String {
   "This is a \(color) car."
  }
}
```

self

When required

```
struct Temperature {
  var celsius: Double

  init(celsius: Double) {
    self.celsius = celsius
  }
}
```

Classes and Inheritance

```swift
class Person {
  let name: String

  init(name: String) {
    self.name = name
  }

  func sayHello() {
    print("Hello there!")
  }
}

let person = Person(name: "Jasmine")
print(person.name)
person.sayHello()

Jasmine
Hello there!
```

Inheritance

Base class	`Vehicle`
Subclass	`TandemBicycle`
Superclass	`Bicycle`

Inheritance

Defining a base class

```
class Vehicle {
   var currentSpeed = 0.0

   var description: String {
     "traveling at \(currentSpeed) miles per hour"
   }

   func makeNoise() {
      // do nothing - an arbitrary vehicle doesn't necessarily make a noise
   }
}

let someVehicle = Vehicle()
print("Vehicle: \(someVehicle.description)")
```

```
Vehicle: traveling at 0.0 miles per hour
```

Inheritance

Create a subclass

```
class SomeSubclass: SomeSuperclass {
    // subclass definition goes here
}
```

```
class Bicycle: Vehicle {
    var hasBasket = false
}
```

Inheritance

Create a subclass

```
class Bicycle: Vehicle {
    var hasBasket = false
}

let bicycle = Bicycle()
bicycle.hasBasket = true

bicycle.currentSpeed = 15.0
print("Bicycle: \(bicycle.description)")
```

```
Bicycle: traveling at 15.0 miles per hour
```

Inheritance

Create a subclass

```
class Tandem: Bicycle {
    var currentNumberOfPassengers = 0
}
```

Inheritance

Create a subclass

```
class Tandem: Bicycle {
    var currentNumberOfPassengers = 0
}

let tandem = Tandem()
tandem.hasBasket = true
tandem.currentNumberOfPassengers = 2
tandem.currentSpeed = 22.0
print("Tandem: \(tandem.description)")
```

```
Tandem: traveling at 22.0 miles per hour
```

Inheritance

Override methods and properties

```
class Train: Vehicle {
    override func makeNoise() {
        print("Choo Choo!")
    }
}

let train = Train()
train.makeNoise()
```

```
Choo Choo!
```

Inheritance

Override methods and properties

```
class Car: Vehicle {
    var gear = 1
    override var description: String {
        super.description + " in gear \(gear)"
    }
}
```

Inheritance

Override methods and properties

```
class Car: Vehicle {
    var gear = 1
    override var description: String {
        super.description + " in gear \(gear)"
    }
}

let car = Car()
car.currentSpeed = 25.0
car.gear = 3
print("Car: \(car.description)")
```

```
Car: traveling at 25.0 miles per hour in gear 3
```

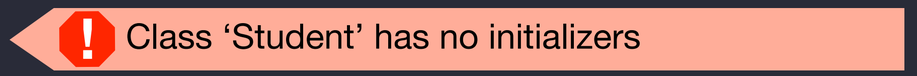
Inheritance

Override initializer

```
class Person {
  let name: String

  init(name: String) {
    self.name = name
  }
}

class Student: Person {
  var favoriteSubject: String
}
```



Inheritance

Override initializer

```
class Person {
  let name: String

  init(name: String) {
    self.name = name
  }
}

class Student: Person {
  var favoriteSubject: String
  init(name: String, favoriteSubject: String) {
    self.favoriteSubject = favoriteSubject
    super.init(name: name)
  }
}
```

References

- When you create an instance of a class:
  - Swift returns the address of that instance
  - The returned address is assigned to the variable
- When you assign the address of an instance to multiple variables:
  - Each variable contains the same address
  - Update one instance, and all variables refer to the updated instance

```swift
class Person {
  let name: String
  var age: Int

  init(name: String, age: Int) {
    self.name = name
    self.age = age
  }
}

var jack = Person(name: "Jack", age: 24)
var myFriend = jack

jack.age += 1

print(jack.age)
print(myFriend.age)
```

```
25
25
```

```swift
struct Person {
  let name: String
  var age: Int
}

var jack = Person(name: "Jack", age: 24)
var myFriend = jack

jack.age += 1

print(jack.age)
print(myFriend.age)
```

```
25
24
```

Memberwise initializers

- Swift does not create memberwise initializers for classes
- Common practice is for developers to create their own for their defined classes

Class or structure?

- Start new types as structures
- Use a class:
  - When you’re working with a framework that uses classes
  - When you want to refer to the same instance of a type in multiple places
  - When you want to model inheritance

Collections

Collection types

Array

Dictionary

Arrays

Defining

```
[value1, value2, value3]
```

```
var names: [String] = ["Anne", "Gary", "Keith"]
```

Arrays

Defining

```
[value1, value2, value3]
```

```
var names = ["Anne", "Gary", "Keith"]
```

Arrays
Defining

```
var numbers = [1, -3, 50, 72, -95, 115]
```

Arrays
Defining

```
var numbers: [Int8] = [1, -3, 50, 72, -95, 115]
```

Arrays
contains

```
let numbers = [4, 5, 6]
if numbers.contains(5) {
  print("There is a 5")
}
```

```
There is a 5
```

Array types

```
var myArray: [Int] = []
```

```
var myArray: Array<Int> = []
```

```
var myArray = [Int]()
```

Working with arrays

repeating

```
var myArray = [Int](repeating: 0, count: 100)

let count = myArray.count

if myArray.isEmpty { }
```

Working with arrays

Accessing or setting a specific item

```
var names = ["Anne", "Gary", "Keith"]
let firstName = names[0]
print(firstName)
```

```
Anne
```

```
names[1] = "Paul"
print(names)
```

```
["Anne", "Paul", "Keith"]
```

Working with arrays

Appending

```
var names = ["Amy"]
names.append("Joe")
names += ["Keith", "Jane"]
print(names)
```

```
["Amy", "Joe", "Keith", "Jane"]
```

Working with arrays

Inserting

```
var names = ["Amy", "Brad", "Chelsea", "Dan"]
names.insert("Bob", at: 0)
print(names)
```

```
["Bob", "Amy", "Brad", "Chelsea", "Dan"]
```

Working with arrays

Removing

```
var names = ["Amy", "Brad", "Chelsea", "Dan"]
let chelsea = names.remove(at:2)
let dan = names.removeLast()
print(names)
```

```
["Amy", "Brad"]
```

```
names.removeAll()
print(names)
```

```
[]
```

Working with arrays

```
var myNewArray = firstArray + secondArray
```

Working with arrays

Arrays within arrays

```
let array1 = [1,2,3]
let array2 = [4,5,6]
let containerArray = [array1, array2]
let firstArray = containerArray[0]
let firstElement = containerArray[0][0]
print(containerArray)
print(firstArray)
print(firstElement)
```

```
[[1, 2, 3], [4, 5, 6]]
[1, 2, 3]
1
```

Dictionaries

```
[key1 : value1, key2: value2, key3: value3]
```

```
var scores = ["Richard": 500, "Luke": 400, "Cheryl": 800]
```

```
var myDictionary = [String: Int]()
var myDictionary = Dictionary<String, Int>()
var myDictionary: [String: Int] = [:]
```

Add/remove/modify a dictionary

Adding or modifying

```
var scores = ["Richard": 500, "Luke": 400, "Cheryl": 800]

scores["Oli"] = 399

let oldValue = scores.updateValue(100, forKey: "Richard")
```

Add/remove/modify a dictionary

Adding or modifying

```
var scores = ["Richard": 500, "Luke": 400, "Cheryl": 800]

scores["Oli"] = 399

if let oldValue = scores.updateValue(100, forKey: "Richard") {
  print("Richard's old value was \(oldValue)")
}
```

```
Richard's old value was 500
```

Add/remove/modify a dictionary

Removing

```
var scores = ["Richard": 100, "Luke": 400, "Cheryl": 800]
scores["Richard"] = nil
print(scores)

if let removedValue = scores.removeValue(forKey: "Luke") {
  print("Luke's score was \(removedValue) before he stopped playing")
}
print(scores)
```

```
["Cheryl": 800, "Luke": 400]
Luke's score was 400 before he stopped playing
["Cheryl": 800]
```

Accessing a dictionary

```
var scores = ["Richard": 500, "Luke": 400, "Cheryl": 800]

let players = Array(scores.keys) //[“Richard", "Luke", "Cheryl"]
let points = Array(scores.values) //[500, 400, 800]

if let lukesScore = scores["Luke"] {
  print(lukesScore)
}
```

```
400
```

```
if let henrysScore = scores["Henry"] {
  print(henrysScore)
}
```

Swift Coding Club

Inspire, Design, Code