CODEPATH*ORG

An Intro to Swift

IOS101

Unit #1

Welcome to Unit 01: Introduction to Swift

1. **Turn on** your camera

2. **Rename** yourself with your pod number and full name:

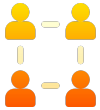
   6 - Montero Hill

3. **Answer** the question below in the chat:

   What is your first memory of interacting with technology?

As a community, we will...

Be present, physically and mentally



Actively build an inclusive community and space

Practice curiosity and adopt a learning mindset

Ask for help when we need it

Keep our cameras on

Agenda

1. Introductions
2. Course Logistics
3. Swift Basics
4. Closures
5. Lab

00

Welcome to CodePath!

CODEPATH*ORG

Meet the Team!

Hello!

- **Andros Slowley**
  - Career
    - Meta - Software Engineer
    - Previously Strava
  - Tech Stack
    - Swift/Objective-C
    - C++
  - Passions
    - Traveler
      - 39 Countries
      - What should be my 40th?
    - Health & Wellness
  - Favorite Quote

Don't fear failure. — Not failure, but low aim, is the crime. In great attempts it is glorious even to fail.

Hi!

Charlie Hieger
CodePath iOS Instructor

“I’ll play it first and tell you what it is later.” – Miles Davis

Education

🎸 Berklee College of Music, Boston MA

iOS Engineering

📲 Velos, San Francisco - Senior iOS Engineer

👨‍💻 CodePath iOS Curriculum Developer + Instructor

Interests

🎧 Music Production + Songwriting

🥾 Exploring nature in the PNW

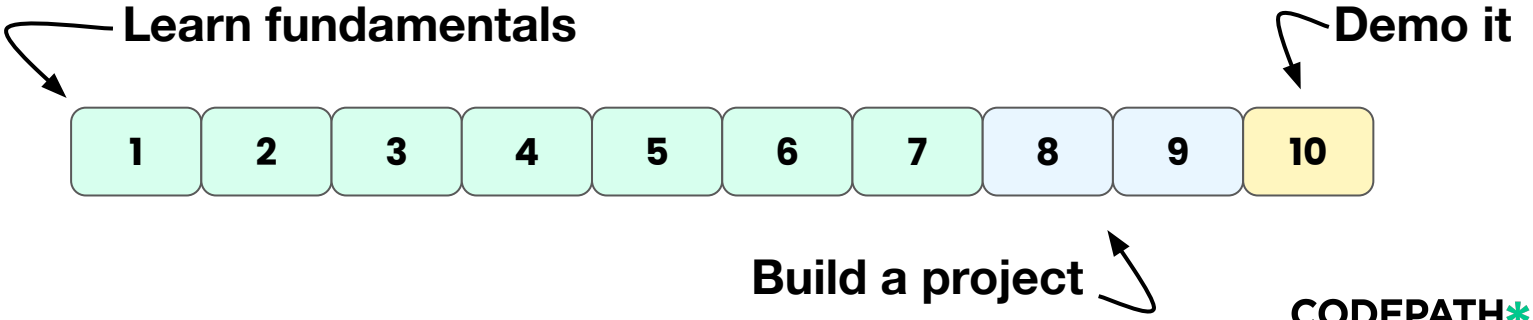
Connect: linkedin.com/in/cnah/

CODEPATH*ORG

Quick Reminder: Class Policies and Structure

Course Logistics

- 10 weeks of iOS!
  - 7 weeks of building technical foundation
  - 2 weeks of building your own app!
  - Final week finishes with demo day
- You should have a solid understanding of how iOS apps are built by the end of this course



Reminder:Classroom Policy

* Attendance at all sessions is mandatory.
* All students are allowed up to **2 absences** throughout the entire program.
    - Students do *not* need to submit a request to use these absences.
    - Students will be marked as late if they arrive after the session starts. **Being significantly late will count as an absence.**
* Each week will you submit a **Project** assignment. All assignments **are due at 11:59 PM PST the day before class (This is 2:59 AM for the east coast)**
    - **You are allowed 2 assignment extensions (Only up to 1 per assignment)**
    - **Do everything in Xcode 16.0 or higher!**

Class Structure

Your class will follow this general structure: Up to an 1 hour of **Direct Instruction** and 1 hour for **Breakout Rooms**

Direct Instruction

- Instructors will present unit slides which can be found in the course portal
- Students are expected to attend and be present
- TFs will be present and support students by answering questions in the chat or Slack channel

Breakout Rooms

- Students will join their pods in their breakout rooms
- Students are encouraged to work together to complete assignments, activities, projects or labs
- TFs will join breakout rooms and support students in their pods

Swift Basics

Swift

* A programming language for iOS, macOS, and any other OS in the Apple ecosystem.
* Originally iOS apps were written in Objective-C, Swift was developed by Apple as a proprietary language for iOS design
* A Type-Safe language

Coding is Like Writing a Sentence!

* In programming, we create "objects" and give them names and actions, just like in a sentence!

Person = Object

* Like a character in a story. Has traits and actions

Talk = Verb

* What the person does (an action)

Name = Noun

* Describes the person (a trait)

Pseudo Code

```
person {
    name = "Dros"
    function talk() {
        "hello everyone"
    }
}
```

Declaring Variables and Constants

	Python	Java	Swift
variable	`num_students = 25`	`int numStudents = 25`	`var numberOfStudents = 25`
constant	`GRAVITY = 9.8`	`final int GRAVITY = 9.8`	`let gravity = 9.8`

- The value of a **constant** cannot change once it has been declared, whereas the value of a **variable** can.
- You can declare multiple constants or variables on a row separated by commas
- If you don't plan to change a value, store it as a constant!

Naming Conventions

Promote Clear Usage

- Include all words needed to avoid ambiguity
- Omit needless words
- Name variables, parameters, and types according to their roles (rather than their type constraints)
- Avoid abbreviations

```
var kartView0
var kartView1
var kartView2

func didDoubleTapKart(_ sender:
            UITapGestureRecognizer) {
 }

Func didPinchKart(_ sender:
            UIPinchGestureRecognizer) {
}
```

Case Conventions

- Names of types and protocols are `UpperCamelCase`. Everything else is `lowerCamelCase`.

Type Annotations

Variable and constant types can be inferred, however you may also provide a type annotation when they are declared.

```
var welcomeMessage: String
welcomeMessage = "Welcome to iOS101!"
```

Declaration

Initialization

Collection Types

Swift supports Arrays, Sets, and Dictionaries

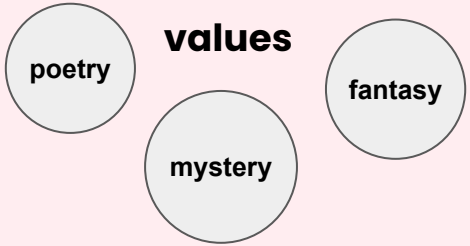
- The mutability of a collection is dependent upon whether it is defined as a variable or a constant

- Collections are generic - they can hold any type - but all elements must be of the same type.

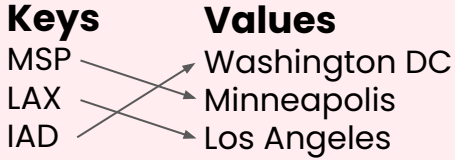
Arrays

index	value
0	Lab1-Mario Kart Pt1
1	Lab2-Mario Kart Pt2
2	Lab3-Weather App Pt1
3	Lab4-Weather App Pt2

Sets



Dictionaries



Arrays

- Create an array of a specified type by using initializer syntax
- Elements can be accessed and modified using the array's methods and properties or subscript syntax

```
// Creates a new array to contain String values
var groceryList: [String] = []

// Adds an item
groceryList.append("eggs")

// Append another array
groceryList += ["cheese", "apples"]

// Accesss and modify a value
groceryList[2] = "tomatoes"

// Prints the number of items in the list
print(groceryList.count)

// Prints the valid indices
print(groceryList.indices)
```

Check for Understanding

What is stored in **`familyVehicles`** at the end of this program?

```
var familyVehicles: [String] = ["Sedan"]


familyVehicles.append("Van")

familyVehicles.append("Sedan")

familyVehicles[0] = "SUV"

familyVehicles.remove(at: 1)

familyVehicles += ["Sedan"]
```

Functions

Function declaration keyword

Parameter format:
`parameterName: Type`

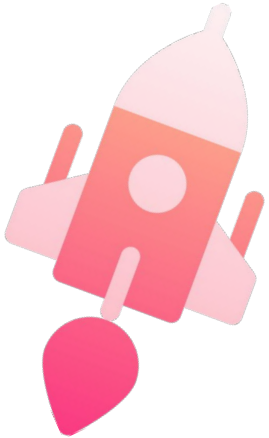
Return format:
`Type`

```
func findHypotenuse(side1: Double, side2: Double) -> Double {
    let sidesSquared = side1 * side1 + side2 * side2
    return sqrt(sidesSquared)
}



let hypotenuse = findHypotenuse(side1: 8.0, side2: 6.0)
```

Instructor Demo

Let's create a program that:

- Uses constants and variables
- Stores data in an array
- Manipulates an array
- Uses a function, conditional, and loop

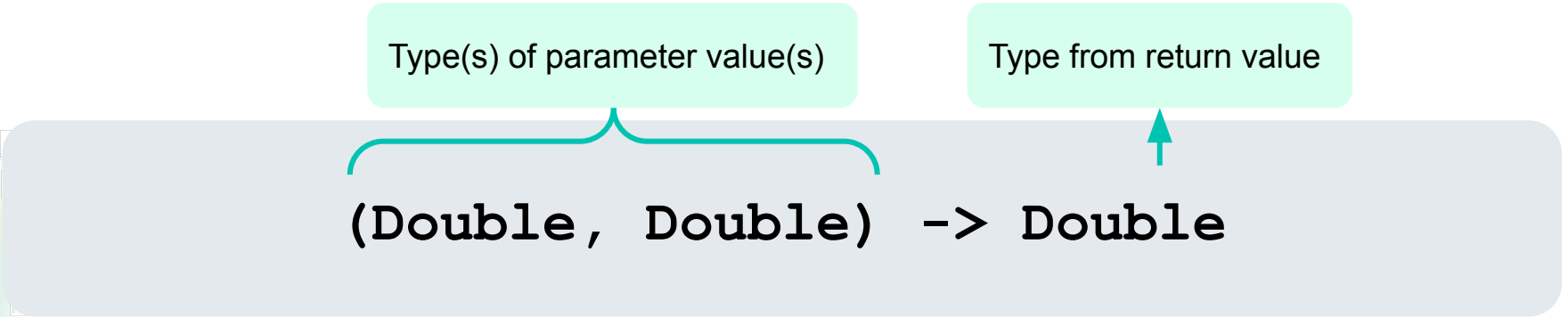


Swift Basics Questions?

Closures

Review: Functions have types

```
func findHypotenuse(side1: Double, side2: Double) -> Double {
    let sidesSquared = side1 * side1 + side2 * side2
    return sqrt(sidesSquared)
}
```



Check for Understanding

What are the types of the three functions on the right?

```
func add(num1: Int, num2: Int) -> Int {
    return(num1 + num2)
}


func greet(name: String){
    print("Hello:" + name)
}


func generator() -> Int {
    Return Int.random(0...10)
}
```

What is a Closure?

Closures are arbitrary blocks of code which can be passed and executed at some point in time.

- They are like an anonymous function in JavaScript or Python

```
// Function

func saysHello() {
    print("Hello")
}

// Call the function
saysHello()
```

```
// Closure

var greeting = {
    print("Hi There!")
}

// Call the closure
greeting()
```

Closures

Closures are arbitrary blocks of code which can be passed and executed at some point in time. They can:

* Closures can be assigned to variables
* Closures capture (or enclose) variables from the surrounding context

```
{ (<Parameters>) -> <Return Type> in

    <Statements>

}
```

Closures:

Closures can be assigned to variables.

```
var multiplier = { (a: Int, b: Int) -> Int in
    return a * b
}
```

```
(Int, Int) -> Int
```

Closures:

Closures capture *or enclose* variables from the surrounding scope

```
func incrementBy(n: Int = 1) -> () -> Int {
  var count: Int = 0
   func counter() -> Int {
      count += n
      return count
  }
    return counter
}

let byFive = incrementBy(n: 5)
print( byFive() )                    //5
print( byFive() )                    //10
```

The closure counter() captures the value of count

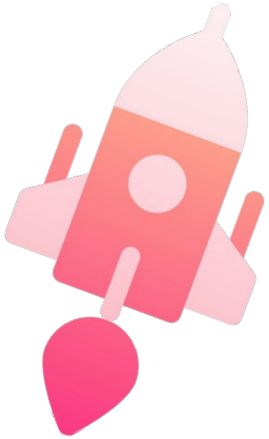
Closures

```
func prismVolume(
    height:Double,
    dim1:Double,
    dim2:Double,
    area:(Double,Double) -> Double))
    -> Double {
        return area(dim1,dim2) *
        height
    }
}
print (volume(height: 2,dim1: 10,
dim2: 12, area:rectangleArea))
print (volume(height: 2,dim1: 10,
dim2: 12, area:rightTrianglelArea))
```

```
let rectangleArea = {
(length:Double,width:Double) ->
Double in
    return length * width
}
let  rightTriangleArea = {
    (length,width) -> Double in
    return length * width / 2.0
}
```

Instructor Demo

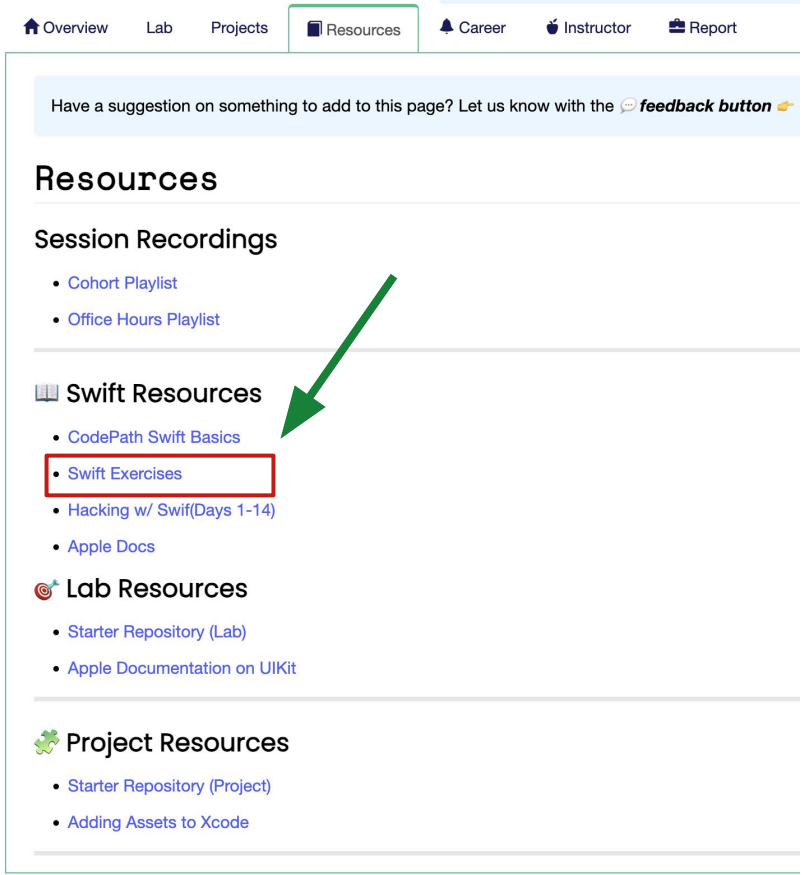
Let’s look at how we may use closures



Closures Questions?

Swift Resources

- A great resource for students while they're learning iOS development is the CodePath iOS Guide.
- For Swift guidance check out The Basics
- Check out the `Resources` tab for more
- **IMPORTANT:** Complete Swift Exercises to help w/ Swift mastery (Optional, but extremely helpful)



Lab

Lab: Mario Kart

* Double tap on a kart to animate forward
* Pan a cart to move it
* Pinch the card to scale

Breakout Rooms

75 minutes

- **Navigate** to the **Lab** tab in the course portal
- **Complete** the Lab!

Questions?
Post on Slack and tag:

@[handle]

Wrap Up

Project Preview: Wordle

* Navigate and make changes in an XCode project
* Learn basic Swift syntax and features
* Learn how to use Apple documentation