

# MOBILE DEVELOPMENT

## LESSON 03

### INTRODUCTION TO SWIFT

Arthur Ariel Sabintsev  
Lead Mobile Architect, [ID.me](https://id.me)

---

**GETTING STARTED**

---

**SOURCETREE**

---

**GETTING STARTED**

---

# LESSON 02 REVIEW

---

## GETTING STARTED

---

# WHAT DID WE LEARN IN LESSON 02?

- Label everything in the Xcode window
- Storyboards vs. Xibs
  - Review of Storyboards
  - Review of Xibs (pronounced *nibs*)
- Learn about View Controllers (High-level)
  - Add multiple View Controllers to the storyboard
  - Link multiple View Controllers together with segues
- Learn about Navigation Controllers

---

**GETTING STARTED**

---

# **HOMEWORK REVIEW**

---

## INTRO TO SWIFT

---

### QUESTIONS

- What are the benefits of using xibs over storyboards?
- What are the benefits of using storyboards over xibs?
- Give a real world example when you would use a nib over a storyboards and vice-versa.
- What are Segues?
- What is a navigation controller?
- How do you use a navigation controller?
  
- Let's do one more thing with Interface Builder: TextFields!

---

**GETTING STARTED**

---

# LEARNING OBJECTIVES

## INTRO TO SWIFT

---

# LEARNING OBJECTIVES

- Nomenclature
- State of Mind
- Swift and Playgrounds
  - Playgrounds Demo #1: Fundamental Data Types
  - Playgrounds Demo #2: Printing to the Console
  - Playgrounds Demo #3: Operators
  - Playgrounds Demo #4: Control Flow
- In-class assignment
- Homework



---

**INTRO TO SWIFT**

---

# NOMENCLATURE

# NOMENCLATURE (PT. 1)

*Source Code:* A collection of computer instructions written using some human-readable computer language.

# NOMENCLATURE (PT. 2)

*Syntax:* The set of rules that defines the combinations of symbols that are written inside your source code files.

---

**INTRO TO SWIFT**

---

# STATE OF MIND

## INTRO TO SWIFT

---

# PROGRAMMING IS LIKE COOKING

- When you cook, you want to do one step at a time, to make sure you don't make a mistake.
- When you program, you want to do many steps at once, to maximize efficiency.
  - After all, isn't that what computers are for?

*Programming is like hyper-efficient cooking*

---

## INTRO TO SWIFT

---

# PROGRAMMING IS LIKE COOKING

- Xcode is your kitchen.
- The Swift programming language is your cabinet full of simple ingredients.
- Your source code is the recipe, which you make from scratch!
- When you cook, you want to do one step at a time, to make sure you don't make a mistake.
- When you program, you want to do many steps at once.
  - Programming is like hyper-efficient cooking

---

**INTRO TO SWIFT**

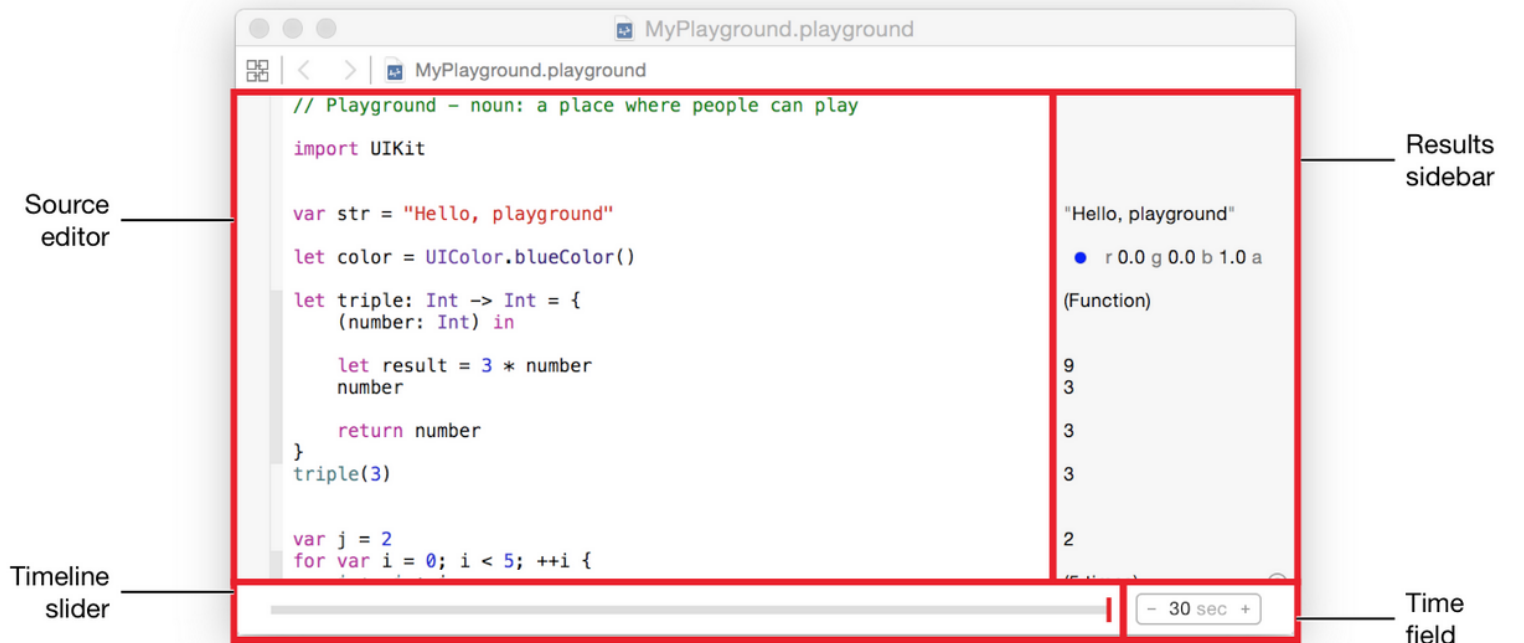
---

# **SWIFT AND PLAYGROUNDS**

# INTRO TO SWIFT

## PLAYGROUNDS

- *Source editor*: Type in swift code and view inline quick look views.
- *Results sidebar*: See the results of evaluating code in the source editor.
- *Timeline slider*: Playback the evaluation of the playground updating any results views.
- *Time field*: Set the total seconds that the playground executes each run.



More info here: [https://developer.apple.com/library/ios/recipes/Playground\\_Help/Chapters/AboutPlaygrounds.html](https://developer.apple.com/library/ios/recipes/Playground_Help/Chapters/AboutPlaygrounds.html)



# INTRO TO SWIFT

---

## DEMO 1 REVIEW: FUNDAMENTAL DATA TYPES

- › Comments on one line (`// Your comment goes here`)
- › Comments on multiple lines (`/* Your comment goes here */`)
- › Constants (`let`)
- › Variables (`var`)
- › Type (e.g., `String`, `Bool`, `Int`, `Float`, etc.)
  - › The type of a constant or variable can be inferred from its initialized value
    - › `let number = 5` // Inferred to be a constant with type `Int`
    - › `let letter = "A"` // Inferred to be a variable with type `String`

## INTRO TO SWIFT

---

# DEMO 1 REVIEW: MUTABILITY VS IMMUTABILITY

Constants (**let**) are **immutable**, meaning that when you set a value to a constant, it stays, well, constant.

Variables (**var**) are **mutable**, meaning that you can change it's value.

## INTRO TO SWIFT

---

# DEMO 2 REVIEW: PRINTLN AND STRING INTERPOLATION (PT. 1)

- Use `println()` whenever you want to print something into the console.
  - `println("Hello")`
- Use String Literal syntax `\()` whenever you want to print out the value of a variable or constant inside of another string.
  - `println("Hello, \(name)")` // Where name is a variable or constant

## INTRO TO SWIFT

---

# DEMO 2 REVIEW: PRINTLN AND STRING INTERPOLATION (PT. 2)

```
//1: Print the following statement  
println("My name is Arthur")
```

```
//2a: Declare a constant of type String with your name  
let name = "Arthur"
```

```
//2b: Print your name  
println(name)
```

```
//3a: Print the string declared name inside of another string  
println("My name is \(name)")
```

```
//3b:  
// \() is known as String literal. It prints out the value of a  
// constant or variable that is lodged between the parenthesis.
```

## DEMO 3 REVIEW: BASIC OPERATORS

- Assignment Operator: `=`
- Arithmetic Operators
  - Addition: `+`
  - Subtraction: `-`
  - Multiplication: `*`
  - Division: `/`
- Remainder Operator (modulo): `%`
- Increment Operator: `++`
- Decrement Operator: `--`
- Comparison Operators
  - Is Equal To: `==`
  - Is Not Equal To: `!=`
  - Greater Than: `>`
  - Greater Than or Equal To: `>=`
  - Less Than: `<`
  - Less Than or Equal To: `<=`

# INTRO TO SWIFT

---

## DEMO 3 REVIEW: CONTROL FLOW (IF STATEMENTS)

- › Conditional statements allow you to execute different pieces of code under certain conditions.

```
// Define a constant named 'volume', and set its initial value to
11.
let volume = 11;

// Print out a statement that depends on the value of volume.
if volume >= 5 && volume <= 10 {
    println("The volume is between 5 and 10")
} else if volume == 11 {
    println("The volume has been turned up to 11!")
} else {
    println("The volume is some other number that is less than 5
        or greater than 11.")
}
```

## INTRO TO SWIFT

---

# DEMO 4 REVIEW: CONTROL FLOW (WHILE LOOPS)

- Evaluates a condition while its true.

```
// Declare a variable and set its initial value to zero.  
var num = 0  
/*  
Perform an action and increment num by one after each action until  
the condition is satisfied.  
*/  
while num < 100 {  
    println(num)  
    num = num + 1  
}
```

### DEMO 4 REVIEW: CONTROL FLOW (FOR-IN LOOPS)

- For-In loops with closed range operator (...)
- Used when iterating over a collection of data.

```
// Prints 1 to 10
for i in 1...10 {
    println(i)
}
```



### DEMO 4 REVIEW: CONTROL FLOW (FOR-IN LOOPS)

- For-In loops with half-open range operator (...)
- Used when iterating over a collection of data.

```
// Prints 1 to 9
for i in 1..10 {
    println(i)
}
```

# GETTING STARTED

---



## EXERCISE

### **KEY OBJECTIVE(S)**

---

Complete the Lesson03.playground file.

### **TIMING**

---

*45 min* 1. Code with partner

*15 min* 2. Debrief

### **DELIVERABLE**

---

To the best of your ability, complete the provided playground file.  
If you hit a question you don't feel comfortable with, ask an instructor.

---

**INTRO TO SWIFT**

---

# **HOMEWORK**

---

## GETTING STARTED

---

# HOMEWORK

- Homework 1 (Lessons 1-2)
  - Link: HW 1 for Lessons 1-2
  - **Due: Sunday, June 14, 2015 at 6pm.**
- At your own pace, read the following:
  - Chapter 3 in the Gitbook:
    - Link: Chapter 3 in MOB Gitbook
  - **The Basics** Chapter in Apple's Swift book
    - Link: The Basics in the Official Swift Book
  - (BONUS) **Control Flow** chapter in Apple's Swift book
    - Link: Control Flow in the Official Swift Book