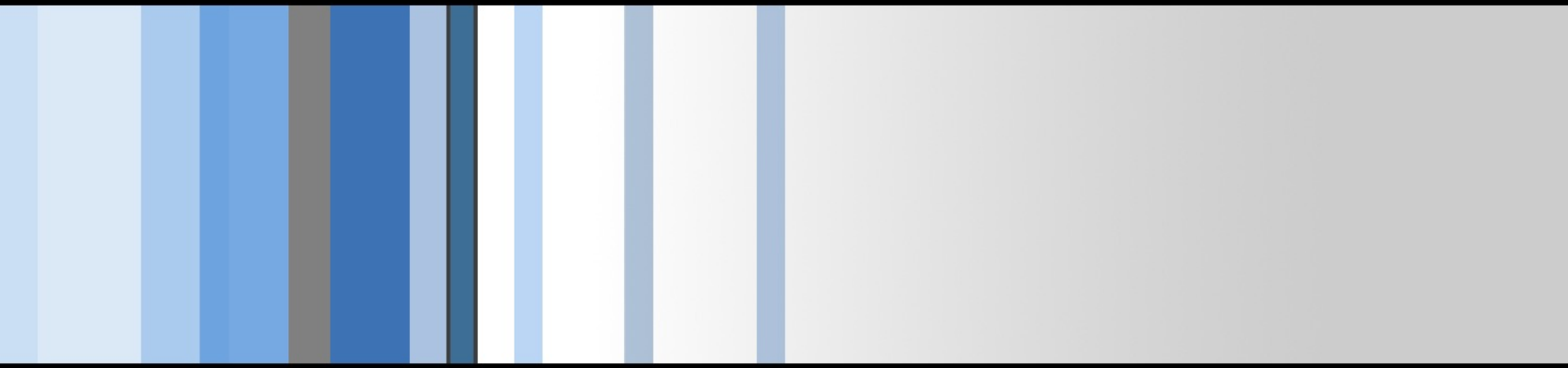
Class 1

# CSC 496: iOS App Development
# Introduction

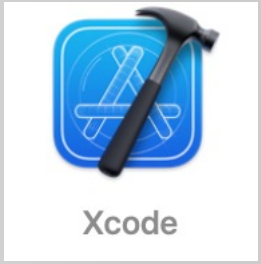
Si Chen (schen@wcupa.edu)



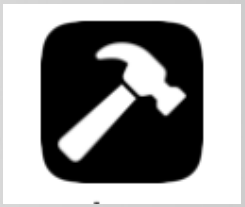
# Introduction

- You'll start by focusing on iOS development tools, basic programming concepts, and industry best practices. Building on this foundation, you'll work through practical exercises, creating apps from scratch, and building the mindset of an app developer.

# Getting Started with App Development

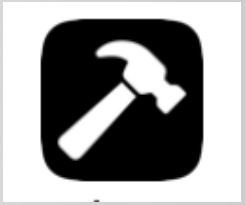



- Two type of lessons
  - Swift Fundamental (Usually on Tuesday)
    - Basic syntax
    - Some new concepts
  - App Design and  Development (Usually on Thursday)
    - SDK
    - Building specific features for iOS apps
    - Mini-project, in-class labs



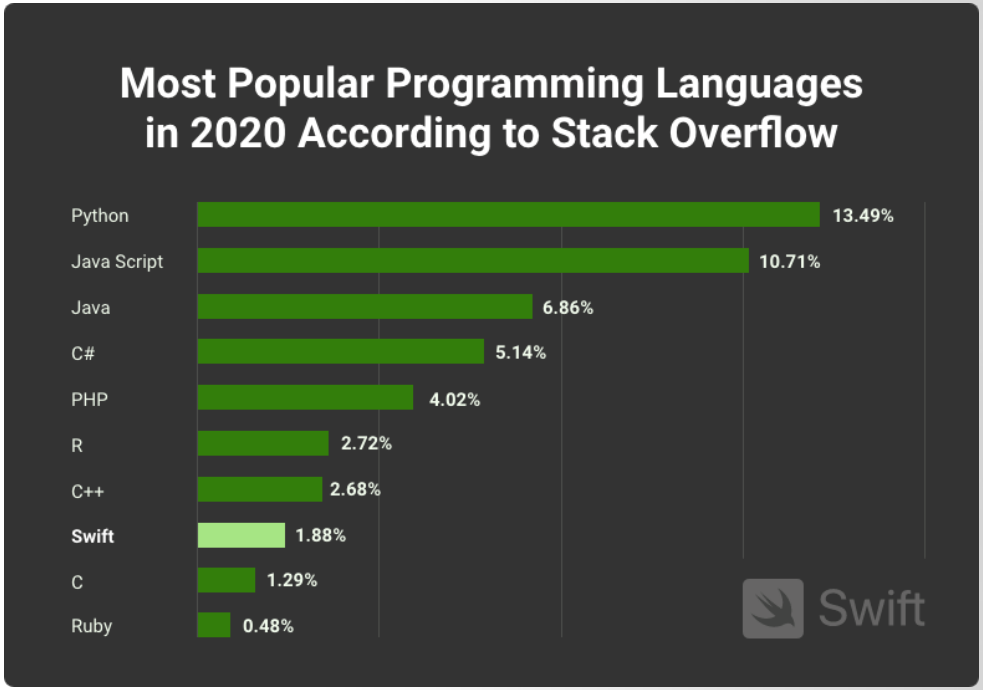
# Getting Started with App Development

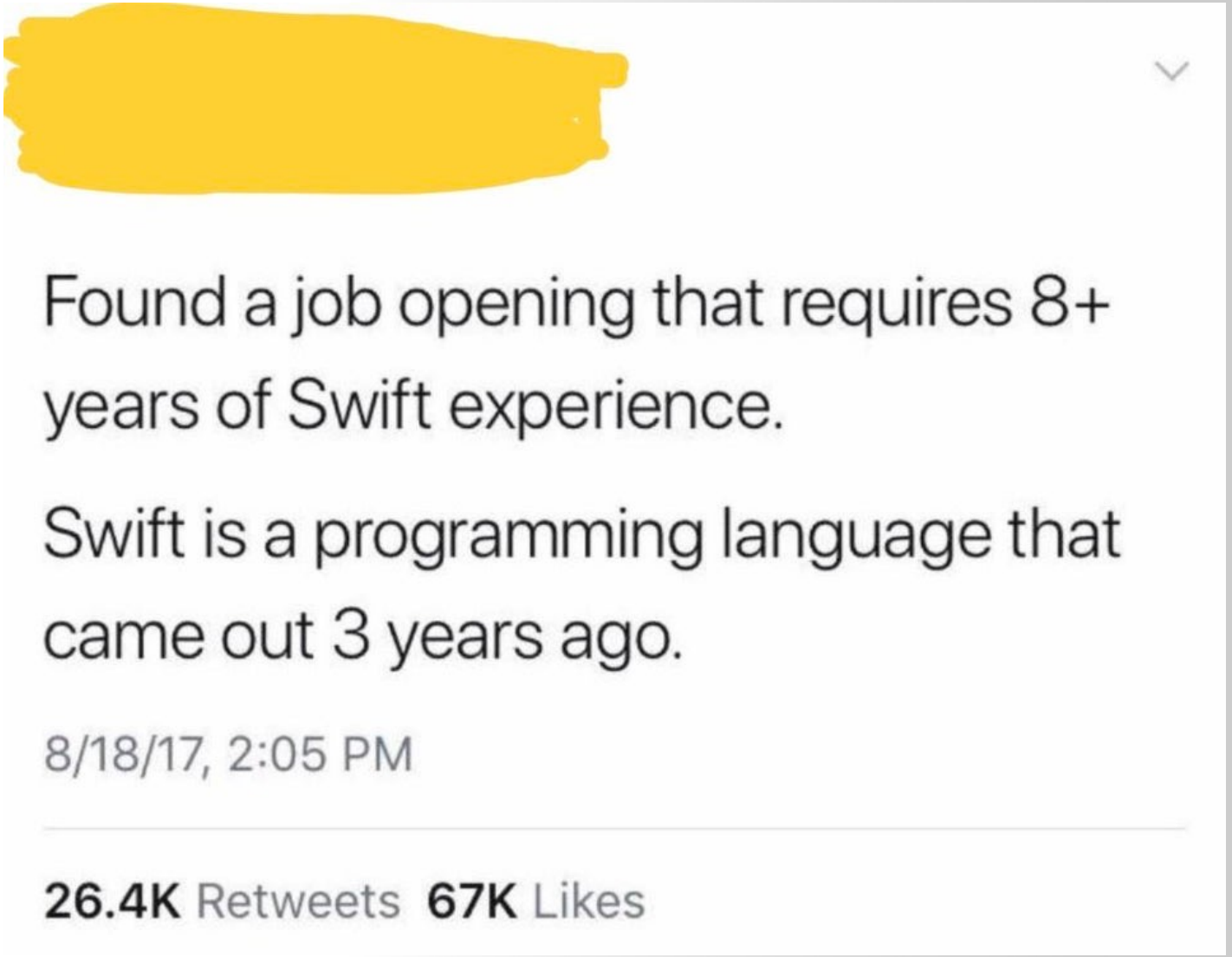
- Two type of lessons
  - **Swift Fundamental (Usually on Tuesday)**
    - **Basic syntax**
    - **Some new concepts (closures, types…)**
  - App Design and  Development (Usually on Thursday)
    - SDK
    - Building specific features for iOS apps
    - Mini-project, in-class labs



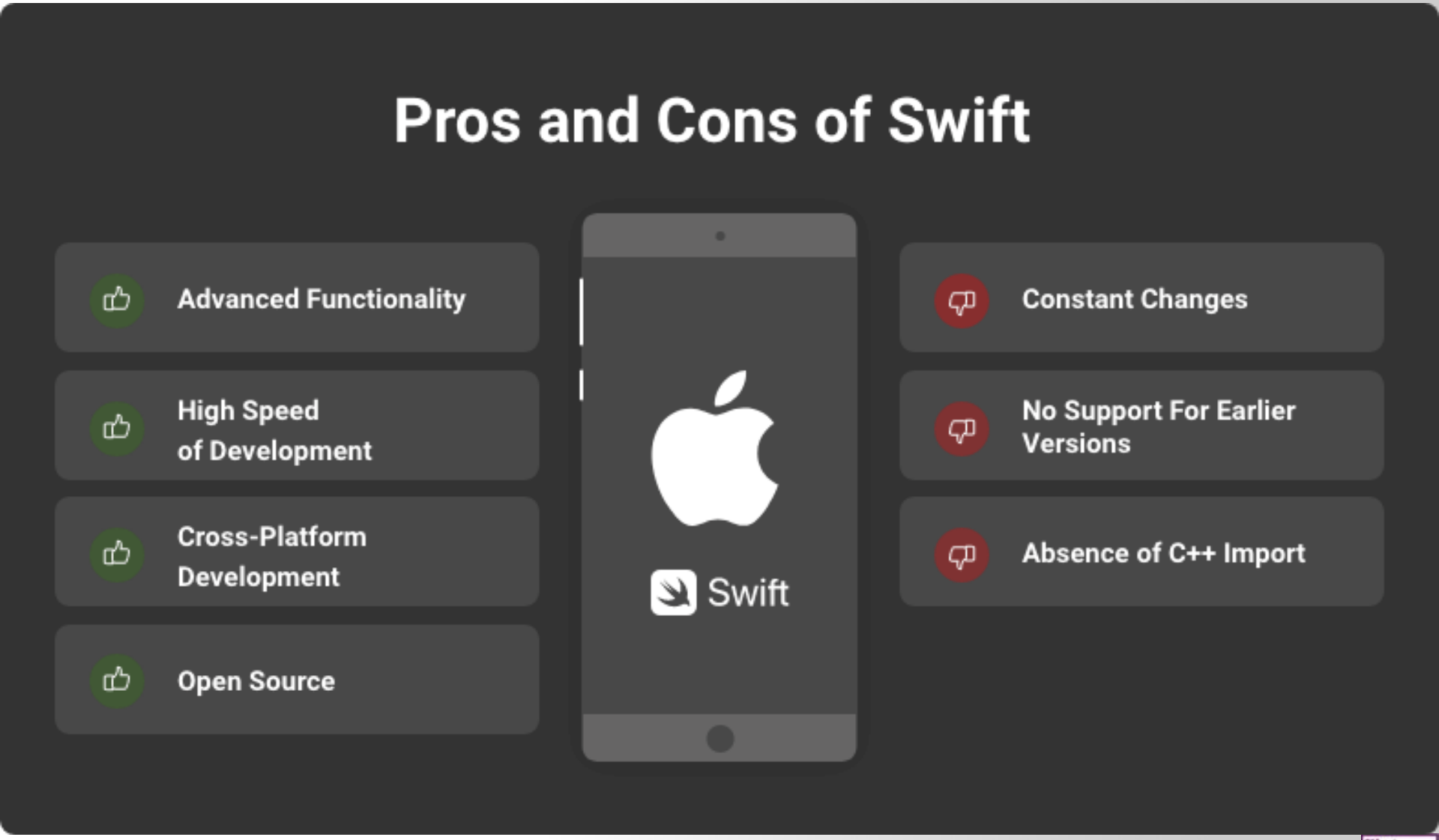
# A Little History about Swift

- At the Apple Worldwide Developers Conference 2014, Apple introduced ***Swift*** as a modern language for writing apps for iOS and macOS. Apple now has new platforms, including watchOS and tvOS, that also use Swift as the primary programming language.
- Since the 1990s, most developers have written applications for Apple platforms in ***Objective-C***, a language built on top of the ***C*** programming language. ***Objective-C*** is more than 30 years old, and ***C*** is more than 40 years old.
- As you learn ***Swift***, you may see the influence of its ***C*** and ***Objective-C*** heritage.

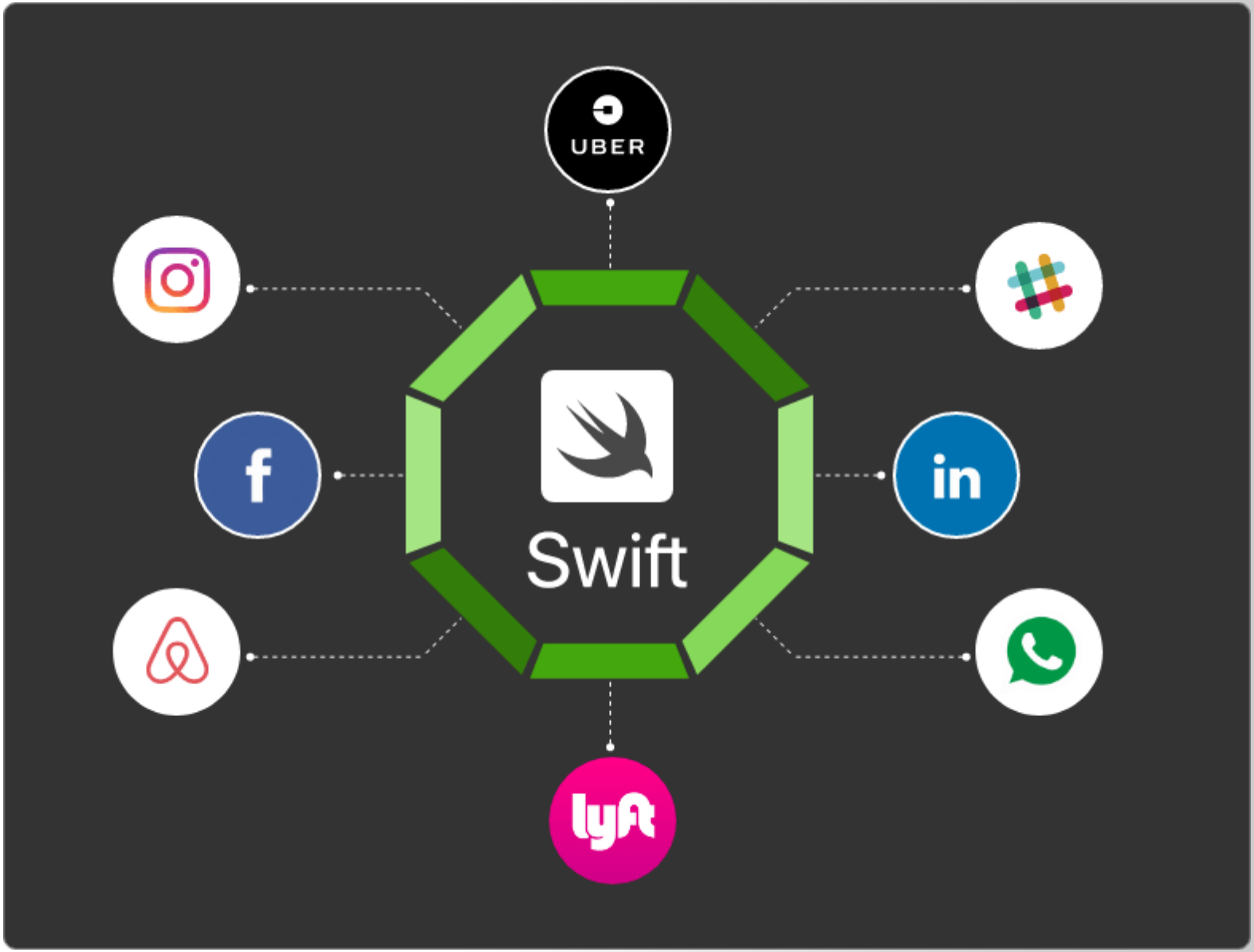



Found a job opening that requires 8+ years of Swift experience.
Swift is a programming language that came out 3 years ago.
8/18/17, 2:05 PM
26.4K Retweets 67K Likes

# Pros and Cons of Swift



# Popular Products Built with Swift



# Functional Programming Features

- Swift is a function-based programming language. It has *higher-order functions, functions as values, nested functions, closures, anonymous functions*, and other functional programming attributes.

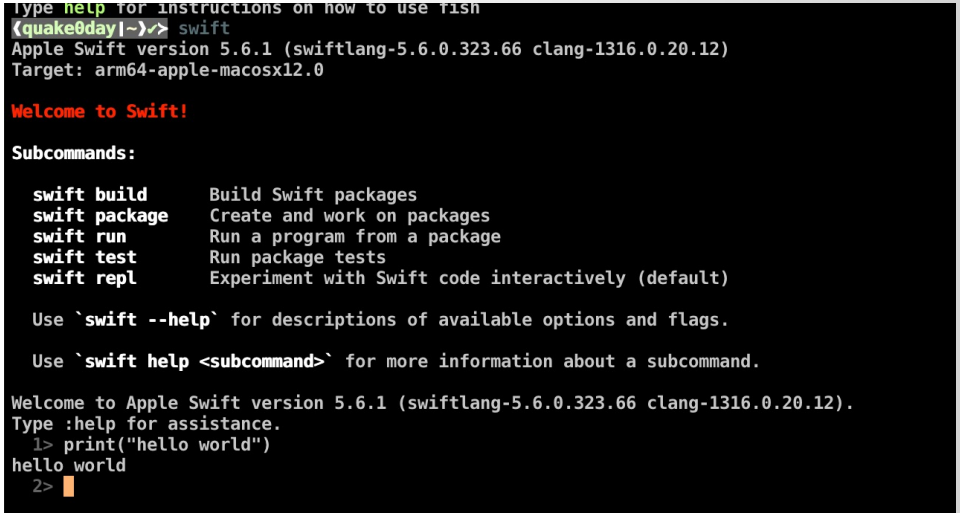
# Hello World

- Swift code is written in plain text files with a .swift file extension. Each line in the file represents a statement, and a program is made up of one or more statements. These are the instructions you wish your app to run.
- In Swift, the default file is called **main.swift**. Any Swift code included in the **main.swift** file will be executed from top to bottom.

```
print("Hello, world!")
```

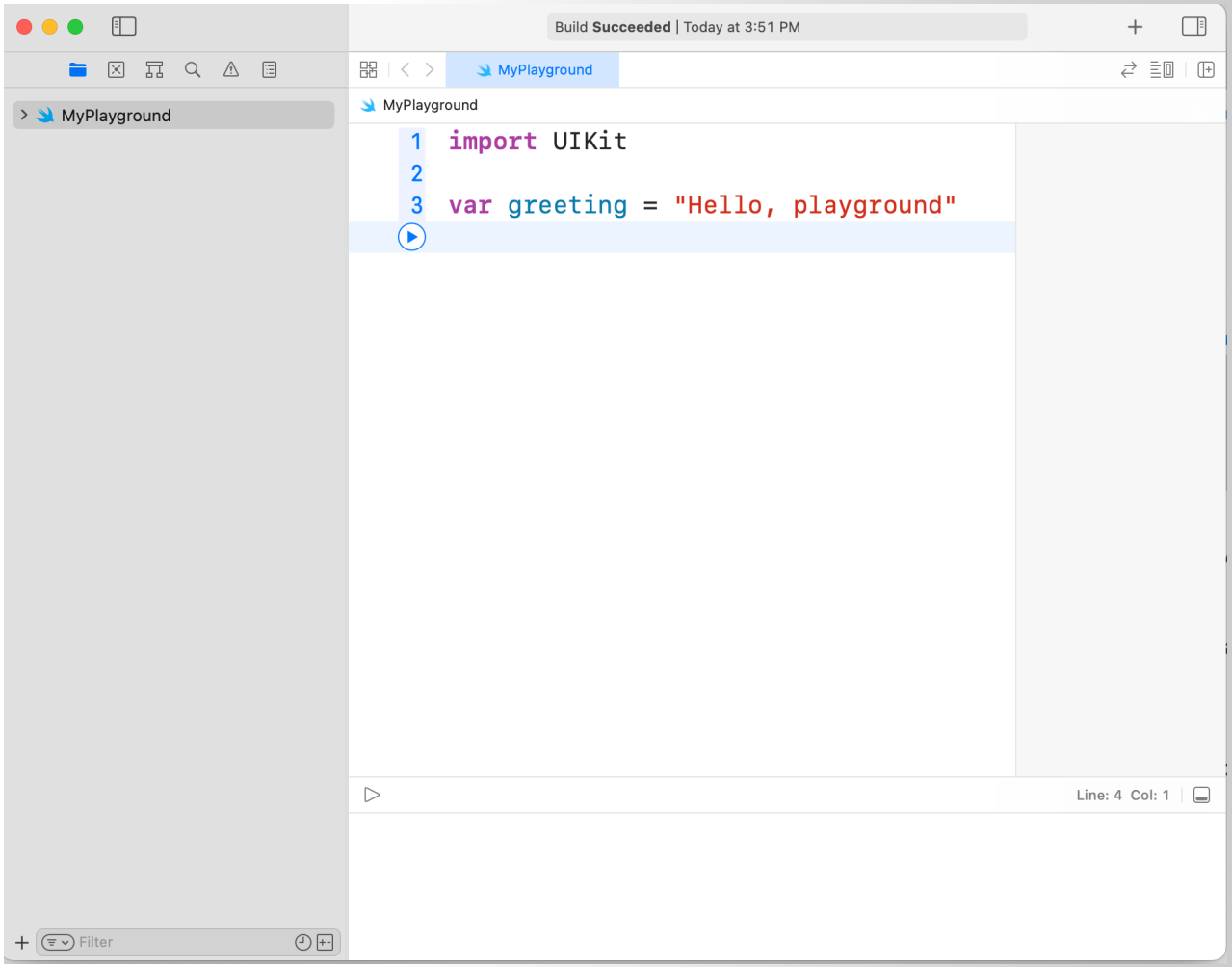
# Terminal and Swift REPL

- **macOS** comes with a console app called **Terminal**, and **Swift** comes with a tool called a **REPL**, which stands for *Read, Eval, Print Loop*. The REPL allows you to enter simple commands, evaluate them, and print the result.
- Use the Swift REPL in the console to write your first “Hello, world!” program.
  - Open the Terminal application on your Mac. You can search “Terminal” in Spotlight or find the application in the system Applications/Utilities folder.
  - Enter the Swift REPL by typing swift and pressing Return.
  - Type the command print(”Hello, world!”) and press Return to execute it.

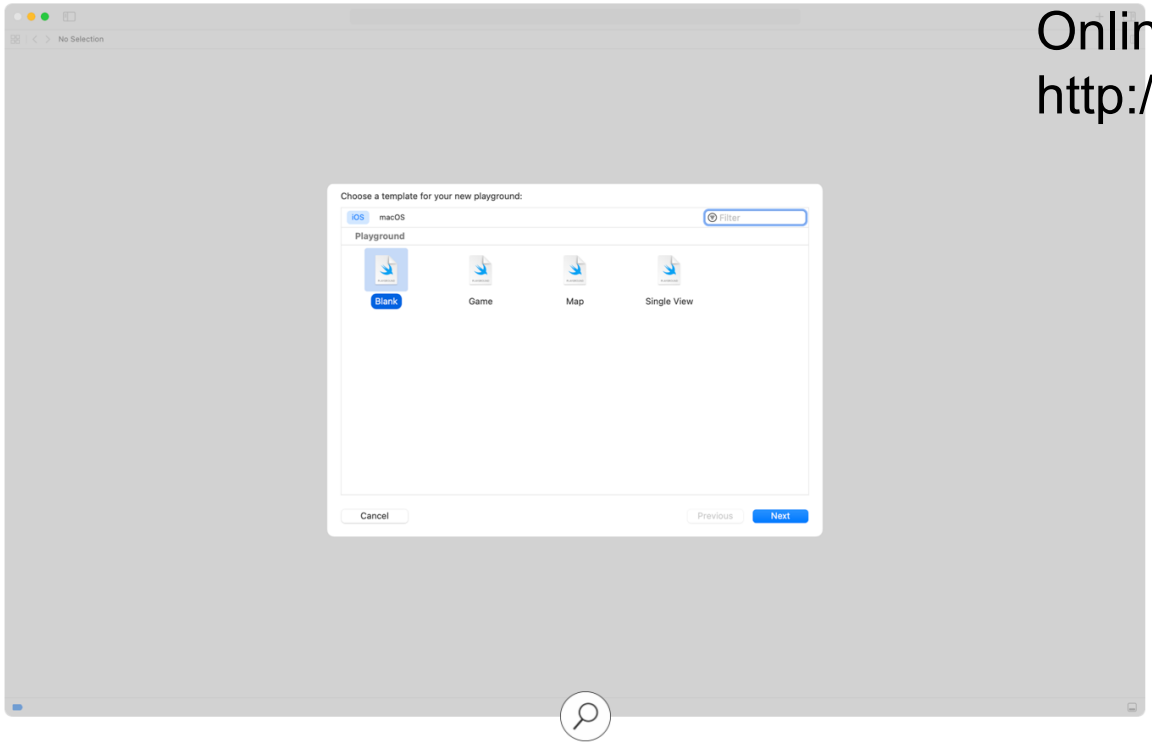


# Playground

- **Playground**: a special Xcode document type that runs Swift code in a simple format with easily visible results.



# Swift Playground




Online:
http://online.swiftplayground.run/

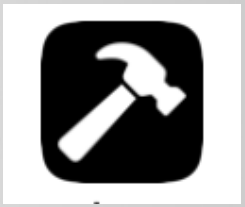
Now go and build your first "Hello, world!" using an Xcode playground.

1. Open Xcode.
2. Create a new playground file by choosing **File > New > Playground** from the menu bar.
3. Choose the iOS platform and the Blank template.
4. Name your playground "Hello, world!" and save it to your course resources folder.

# What about the UI??


# Getting Started with App Development

- Two type of lessons
  - Swift Fundamental (Usually on Tuesday)
    - Basic syntax
    - Some new concepts (closures, types…)
  - **App Design and  Development (Usually on Thursday)**
    - **SDK**
    - **Building specific features for iOS apps**
    - **Mini-project, in-class labs**



# SwiftUI

SwiftUI helps you build great-looking apps across all Apple platforms with the power of Swift — and surprisingly little code. You can bring even better experiences to everyone, on any Apple device, using just one set of tools and APIs.

```
8  import SwiftUI
9
10 @main
11 struct myfristapp111App: App {
12     var body: some Scene {
13         WindowGroup {
14             ContentView()
15         }
16     }
17 }
```

# Other concepts


- Add high-performance 2D content with smooth animations to your app, or create a game with a high-level set of 2D game-based tools.

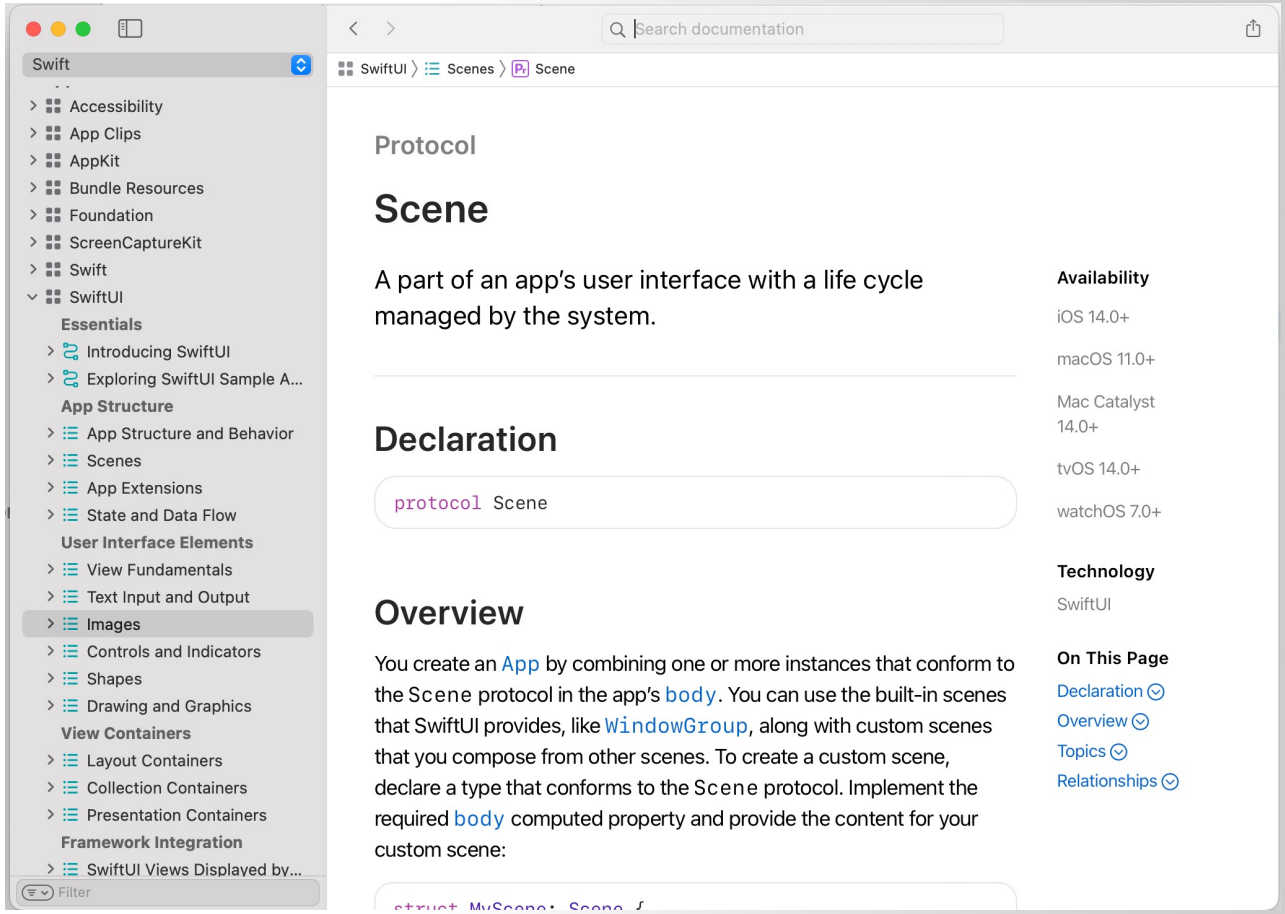
# AVFoundation

- AVFoundation combines several major technology areas that together encompass a wide range of tasks for inspecting, playing, capturing, and processing audiovisual media on Apple platforms.

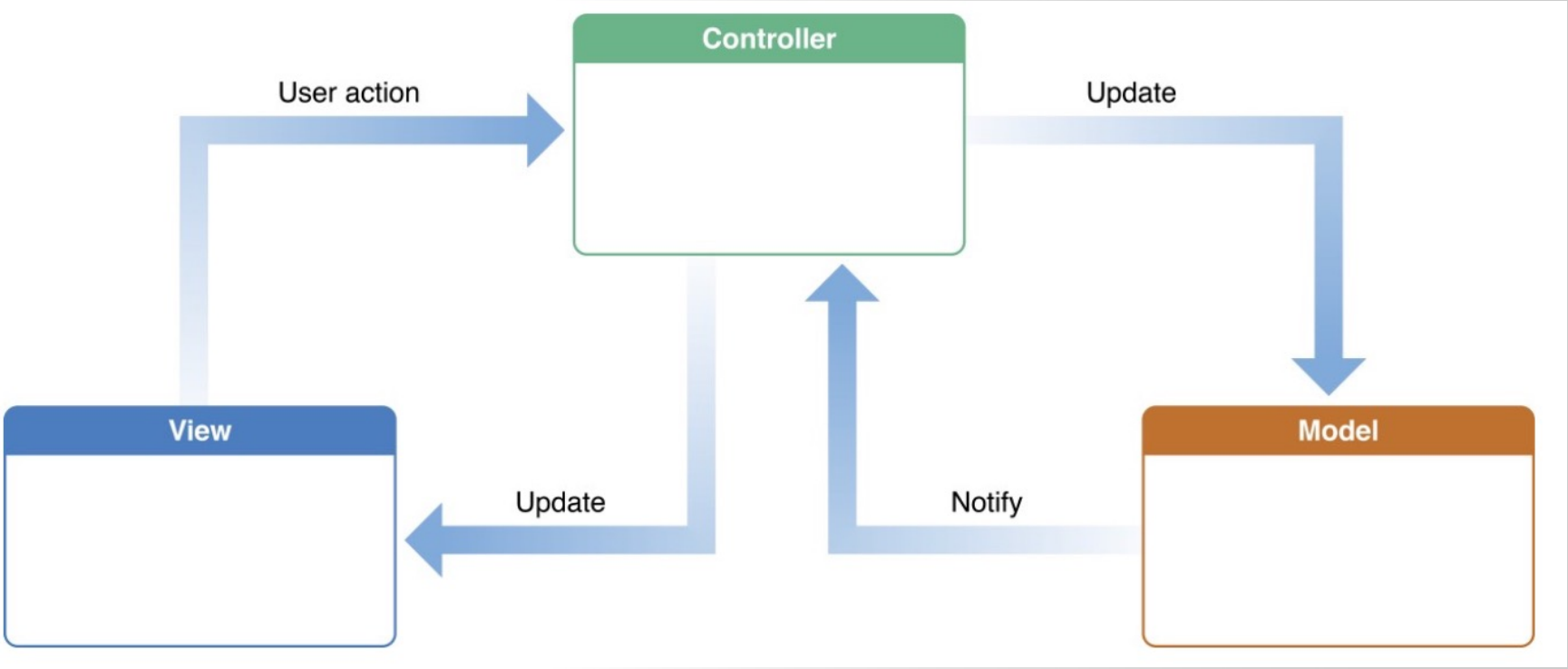
# GameKit


- Enable players to interact with friends, compare leaderboard ranks, earn achievements, and participate in multiplayer games.

# Developer Documentation



# Model-View-Controller



# Constants, Variables, and Data Types

- Constants and variables must be declared before they're used. You declare constants with the **let** keyword and variables with the **var** keyword.

```
let maximumNumberOfLoginAttempts = 10
var currentLoginAttempt = 0
```

- You can declare multiple constants or multiple variables on a single line, separated by commas:

```
var x = 0.0, y = 0.0, z = 0.0
```

# Type Annotations

- You can provide a ***type*** *annotation* when you declare a constant or variable, to be clear about the kind of values the constant or variable can store.
- Write a type annotation by placing a colon after the constant or variable name, followed by a space, followed by the name of the type to use.

```
var welcomeMessage: String
```

# Type Annotations

- You can define multiple related variables of the same type on a single line, separated by commas, with a single type annotation after the final variable name:

```
var red, green, blue: Double
```

# Printing Constants and Variables

- Swift uses *string interpolation* to include the name of a constant or variable as a placeholder in a longer string, and to prompt Swift to replace it with the current value of that constant or variable.
- ***Wrap the name in parentheses and escape it with a backslash before the opening parenthesis:***

```
print("The current value of friendlyWelcome is \(friendlyWelcome)")
// Prints "The current value of friendlyWelcome is Bonjour!"
```

# Comments

- Comments in Swift are very similar to comments in Java.

```
// This is a comment.
```

```
/* This is also a comment
but is written over multiple lines. */
```

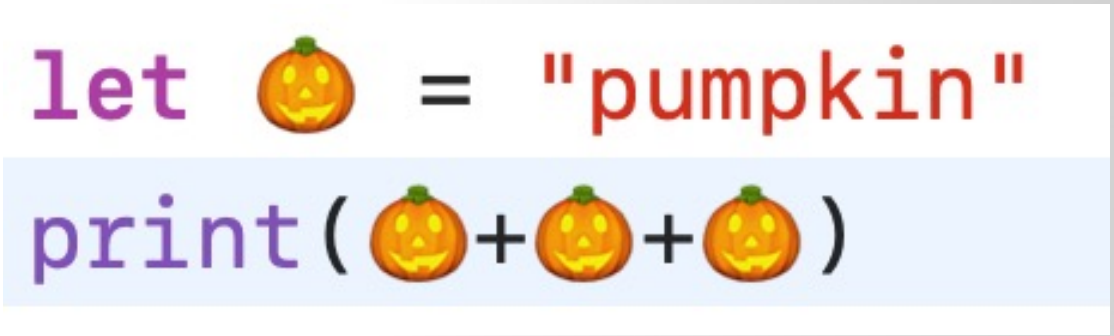
```
/* This is the start of the first multiline comment.
 /* This is the second, nested multiline comment. */
This is the end of the first multiline comment. */
```

# Semicolons

- Unlike many other languages, Swift **doesn't require** you to write a semicolon (;) after each statement in your code, although you can do so if you wish.
- However, semicolons *are* required if you want to write multiple separate statements on a single line:

```
let cat = "🐱"; print(cat)
// Prints "🐱"
```

# Exercise




**http://online.swiftplayground.run/**

# Most Common Types in Swift

| Name | Type | Purpose | Example |
|---|---|---|---|
| Integer | `Int` | Represents whole numbers, or integers | `4` |
| Double | `Double` | Represents numbers requiring decimal points, or real numbers | `13.45` |
| Boolean | `Bool` | Represents `true` or `false` values | `true` |
| String | `String` | Represents text | `"Once upon a time..."` |

# Type Inference

- You may have noticed that you **don't have to specify the type of value when you declare a constant or variable**. This is called **type inference**. Swift uses type inference to make assumptions about the type based on the value assigned to the constant or variable.

```
let cityName = "San Francisco"
// "San Francisco" is obviously a `String`, so the compiler
automatically assigns the type of cityName to a `String`.

let pi = 3.1415927
// 3.1415927 is a number with decimal points, so the compiler
automatically assigns the type `pi` to a `Double`.
```

```
let cityName: String = "San Francisco"

let pi: Double = 3.1415927
```

# Create your own type

```
struct Car {
  var make: String
  var model: String
  var year: Int
}
```

# Exercise

- Defines a new structure called **Resolution**, to describe a pixel-based display resolution.
- This structure has **two** stored properties called **width** and **height**. Stored properties are constants or variables that are bundled up and stored as part of the structure or class. These two properties are inferred to be of type **Int** by setting them to an **initial integer value of 0**.

```
1  import UIKit
2
3  struct Car{
4      var make:String
5      var model:String
6      var year:Int
7  }
8
9  var c = Car(make: "BMW",
       model: "X3", year:
       2022)
10 print(c.make)
```

# Q & A