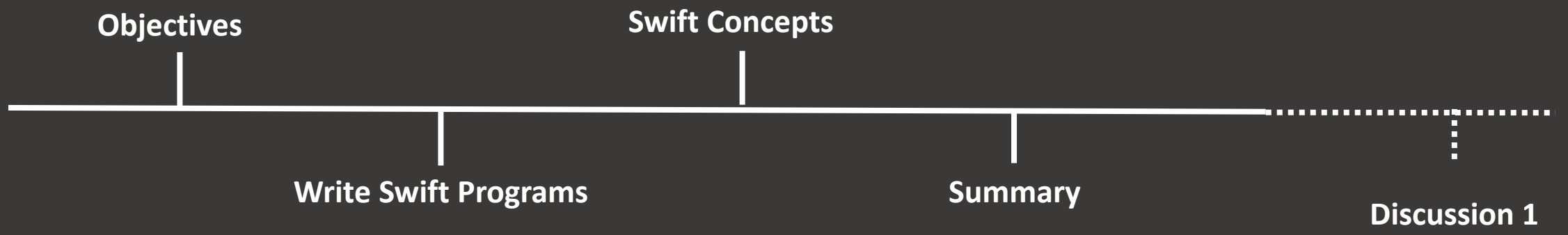
Mobile Applications Development 2 (MAD2)

Diploma in IT

Teaching Team:
Mr Ng Poh Oon
Mr Charles Keck

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY

Chapter 1
Intro to Swift 5.x



Objectives

To be able to understand:

- Swift Language
  - Swift data types
  - Swift operators
  - Swift Control structures
  - Swift Arrays
  - Swift Functions
  - Swift Enumerations
  - Swift Optionals

Introduction to Swift 5.x

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY

MAD2 Oct 2019

Introduction to Swift 5.x

- Swift is a general-purpose programming language built using a modern approach to safety, performance, and software design patterns.
  - Safe: Eliminate undefined behavior, and catch developer mistakes before software is in production.
  - Fast: Comparable to other languages in terms of performance. Performance must be predictable and consistent.
  - Expressive: Modern language.
- *Swift is a powerful and intuitive programming language for macOS, iOS, iPadOS, watchOS and tvOS. Writing Swift code is interactive and fun, the syntax is concise yet expressive.*

Introduction to Swift 5.x

- Designed for Safety
  - Swift eliminates entire classes of unsafe code
  - Variables - initialized before use
  - Arrays and integers - checked for overflow
  - Memory – managed (memory guards against mistakes)
  - By default Swift objects can never be nil

https://developer.apple.com/swift/#safety

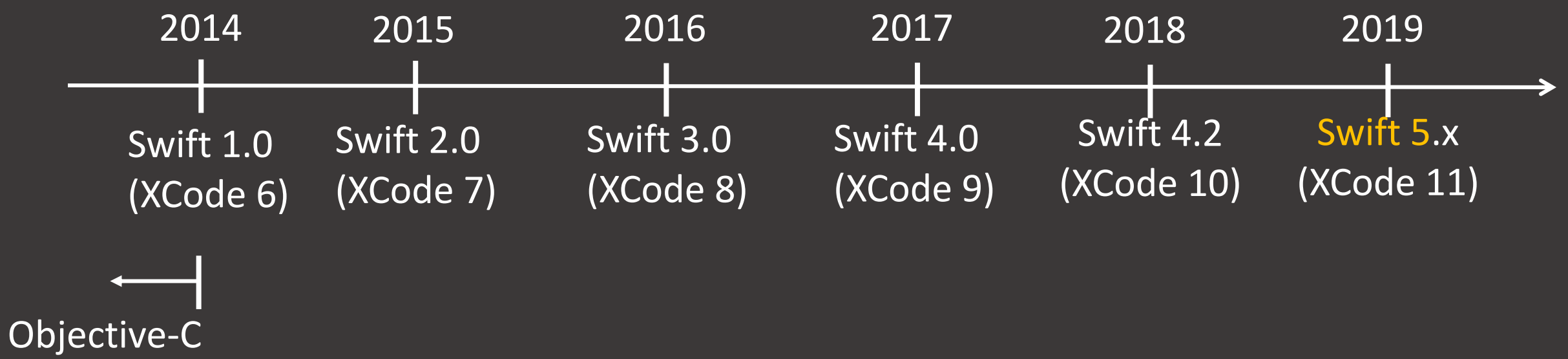
Introduction to Swift 5.x

- Fast and Powerful
  - high-performance LLVM compiler technology
  - Syntax and standard library tuned for performance
- Source and Binary Compatibility
  - Swift 4 code compatible to new compiler

https://developer.apple.com/swift/#fast

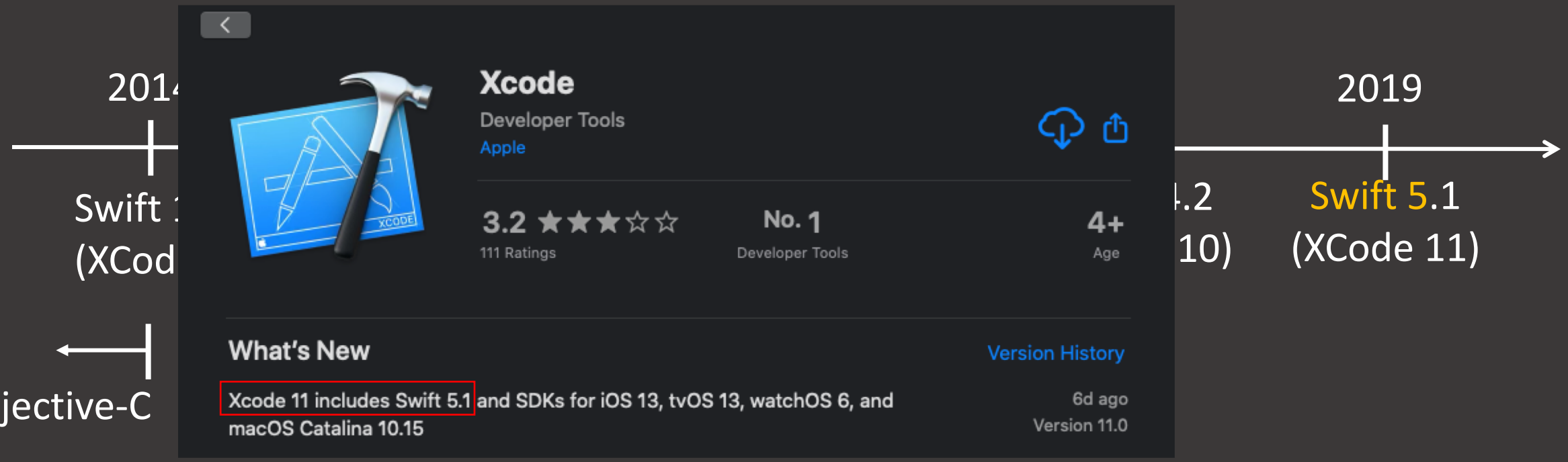
Introduction to Swift 5.x

- Evolution



Introduction to Swift 5.1

- Evolution



Introduction to Swift 5.x

new

- iPhone 11, 11 Pro and 11 Pro Max ➔ iOS 13

Introduction to Swift 5.x (iPhone 11)

- 19 Sep 2019 (Thurs) Orchard Road store in Singapore

https://www.macrumors.com/2019/09/19/iphone-11-lines-apple-stores/

Write Swift Programs

MAD2 Oct 2019

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY

Write Swift Programs

- REPL
  - MacOS and Linux

Using the REPL

If you run the `swift` command without any other arguments, you'll launch the REPL, an interactive shell that will read, evaluate, and print the results of any Swift code you enter.

```
CK-MacBook-Pro:~ charles$ swift
Welcome to Apple Swift version 5.1 (swiftlang-1100.0.270.6 clang-1100.0.32.1).
Type :help for assistance.
  1>
```

- XCode (Integrated Development Environment)
  - macOS, iOS, iPadOS, watchOS and tvOS
  - Playground

Swift Concepts

https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html

MAD2 Oct 2019

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY

Swift Data Types

Type	Description
`Char`	A character
`Int`	An integer (1, 14, -2, 124 and etc.), depending on platform Int32/Int64
`Float`	A floating point number (1.4, 68.3, -273.15 & etc), 64 bits
`Double`	Double precision float, 32 bits
`UInt8`	Only positive integer values from 0 - 255
`Bool`	Yes or No

`Types are CAPITALISED!`

Example:

```
let someString = "Welcome to MAD2!"
```

Like Python and C#, Swift is case-sensitive

Swift Data Types

- Constant and variable names can contain almost any character, including Unicode characters:

```
let π = 3.14159
let 🐶🐮 = "dogcow"
```

- You can change the value of an existing variable to another value of a compatible type.

```
var friendlyWelcome = "Hello!"
friendlyWelcome = "Bonjour!"
// friendlyWelcome is now "Bonjour!"
```

- You can print the current value of a constant or variable with the print( ) function:

```
print(friendlyWelcome)
```

Swift Operators

Operator	Description
+	Addition
–	Subtraction
/	Division
*	Multiplication
%	Modulo (remainder function)
!=	Not equal
!	Boolean Not
&&	Boolean And
||	Boolean Or

Swift Operators

Examples:

```
var num = 5
num += 1  //num now equals to 6
var remainder = num % 4  //remainder equals to 2
```

Swift Strings

- String
  - To represent an array of Unicode characters
  - Unicode provides a unique number for any character, regardless of platform, language

```
var emptyString = "" or String()
if emptyString.isEmpty {
       print ("Nothing to see here")
}
```

- String Mutability
  - The content can be edited/changed after creation

```
var variableString = "Horse"
variableString += " and carriage"
// variableString is now "Horse and carriage"
```

Working with Swift Characters

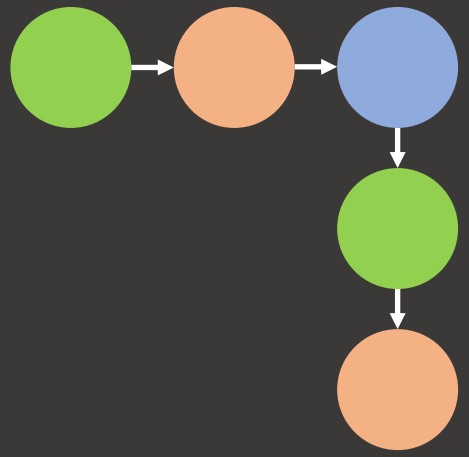
- You can access the individual Character values for a String by iterating over its characters property with a for-in loop:

```
for character in "Dog!🐶".characters {
    print(character)
}
// D
// o
// g
// !
// 🐶
```

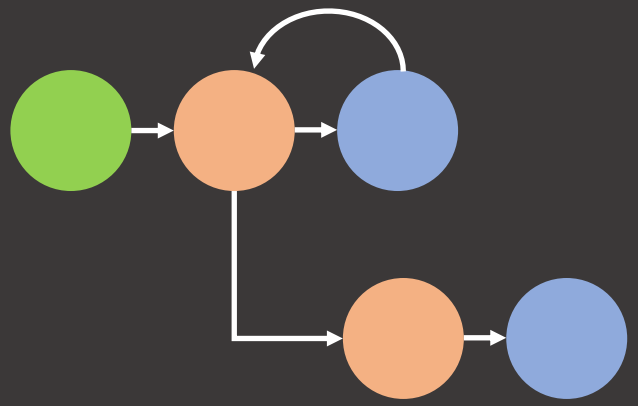
What about special characters " or \?

Swift Control Structures

- Sequence



- Conditional
  - if
  - switch



- Loop
  - For-in
  - while
  - repeat while

If Selection Statements

```
if (amount < balance) {
   balance = balance – amount
}
```

```
if (amount < balance) {
   balance = balance – amount
}
else {
   balance = balance + amount
}
```

```
if (amount < balance) {
   balance = balance – amount
}
else if (amount > balance) {
   balance = balance + amount
}
else {
   balance = balance * 2
}
```

Switch Statements

```
switch ( num ) {
   case 0:
     mark = 10
   case 1:
     mark = 20
   case 2:
     mark = 30
   default:
     mark = 100
}
```

Explicit break is missing?

Switch Statements

```
let someCharacter: Character = "e"
switch someCharacter {
case "a", "e", "i", "o", "u":
    print("\(someCharacter) is a vowel")
case "b", "c", "d", "f", "g", "h", "j", "k", "l", "m", "n", "p",
     "q", "r", "s", "t", "v", "w", "x", "y", "z":
    print("\(someCharacter) is a consonant")
default:
    print("\(someCharacter) is not a vowel or a consonant")
}
```

Loops

```
for index in 1...5 {
    print("\(index) times 5 is \(index * 5)")
}
// 1 times 5 is 5
:
:
// 5 times 5 is 25
```

```
var num = 0

while ( num < 10 )
{
    :
    :
    :

    num += 1
}
```

Loops

```
var num = 0

repeat {

    num += 1
}
while num < 10
```

Swift Arrays

- Numeric array

```
var num:[Int] = [1,2,3,4,5]
var marks: [Float] = [0.0, 1.1]
marks.append(2.2)
    OR
marks += [2.2]
```

- Retrieve a value from the array by using *subscript syntax:*

```
var firstItem = marks[0]
```

- You can use subscript syntax to change an existing value at a given index:

```
marks[0] = 1.2
```

Swift Functions

- Functions are self-contained chunks of code that perform a specific task. You give a function a name that identifies what it does, and this name is used to “call” the function to perform its task when needed.
- Every function in Swift has a type, consisting of the function’s parameter types and return type.

Functions without parameters or multiple parameters?

```
func sayHello(personName: String) -> String {
        let greeting = "Hello, " + personName + "!"
        return greeting
}
```

```
print(sayHello(personName: "Ah Boy"))
```

Swift Functions

- Functions with Multiple parameters

```
func sayHello(personName: String, alreadyGreeted: Bool) -> String {
      if alreadyGreeted {
                  return sayHelloAgain(personName)
      } else {
                return sayHello(personName)
    }
```

```
print(sayHello("Ah Boy", alreadyGreeted: true))
```

Swift Closures

- Closures are self-contained blocks of functionality that can be passed around and used in your code
- E.g. Sort Method

```
let names = ["Chris", "Alex", "Ewa", "Barry", "Daniella"]

func backwards(s1: String, _ s2: String) -> Bool {
      return s1 > s2
}

var reversed = names.sorted(backwards)

// reversed is equal to ["Ewa", "Daniella", "Chris", "Barry", "Alex"]
```

Swift Eumerations

An enumeration defines a common type for a group of related values and enables you to work with those values in a type-safe way within your code.

```
enum CompassPoint {
       case North
       case South
       case East
       case West
}

var directionToHead = CompassPoint.East

enum Planet {
    case Mercury, Venus, Earth, Mars, Jupiter,
       Saturn, Uranus, Neptune }

var smallestPlanet = Planet.Mercury
```

Swift Optionals

An optional in Swift is a type that can hold either a value or no value. Optionals are written by appending a ? to any type

```
var middlename: String?
```

```
var red: String = "Red"
red = nil // error: nil cannot be assigned to type 'String'
```

Swift Optionals (forced Unwrapped)

By appending an exclamation mark at the end of an optional, the optional is forced unwrapped.

```
var message : String?

message = "Hello"

print("\(message)") // Optional("Hello")\n


// forced Unwrapped
print("\(message!") // Hello\n
```

Summary

To be able to understand:

- Swift Language
  - Swift data types
  - Swift operators
  - Swift Control structures
  - Swift Arrays
  - Swift Functions
  - Swift Enumerations
  - Swift Optionals