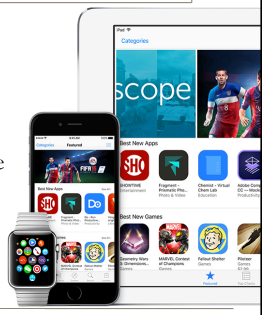
CSC 471 / 371 Mobile Application Development for iOS

Prof. Xiaoping Jia
School of Computing, CDM
DePaul University
xjia@cdm.depaul.edu
@DePaulSWEng

Outline

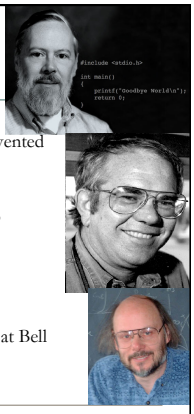
- A little history:
  from Objective-C to Swift
- Xcode Playground
- A primer on Swift programming language, the basics
  - Data types
  - Control statements



Objective-C Programming Language

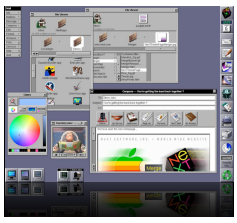
Objective-C

- Invention of C
  - Dennis Ritchie at AT&T (Bell Labs) invented C early 1970
- Extension of Objective-C
  - Brad Cox designed Objective C in 1983
    - Inspired by SmallTalk
    - Layered on top of C language
    - Added objects
- Contemporary to C++
  - Invented in 1979, by Bjarne Stroustrup at Bell Labs
  - Originally called C with classes



NeXTSTEP and NeXT Cube

- First major licensee
  - 1985 – Steve Jobs, NeXT Computer
  - Used to develop UI for NeXTSTEP OS (and later, OpenStep)
- Objective C became the basis for the operating system for the NeXT cube



From NeXTSTEP to Mac OS X

- Apple acquired NeXT Inc.
- Mac OS X 10.0
  - March, 2001, Initial release of OS X
- Based on NeXTSTEP
  - Mac Toolbox APIs
  - 'Mac-like' UI tweaks
  - HFS+ file system
  - Mac OS 9 compatibility environment
  - Quartz rendering engine
  - Objective-C UI layer rebranded 'Cocoa'

iPhone & iOS

- iPhone
  - Released June 29, 2007
- iPhone OS
  - Port of Mac OS X
  - Shares same developer toolset
  - Developer frameworks adapted and scaled down for mobile device
- iPhone OS 2.0b2
  - March, 2008. Initial release of iPhone OS SDK

DePaul University 7

Objective-C Language

- Strict superset of C
  - Mix C with Objective C
  - Or even C++ with Objective C (usually referred to as Objective C++)
- A very simple language, but some new syntax
- Single inheritance, classes inherit from one and only one superclass
- Protocols define behavior that cross classes
- Dynamic runtime
- Loosely typed

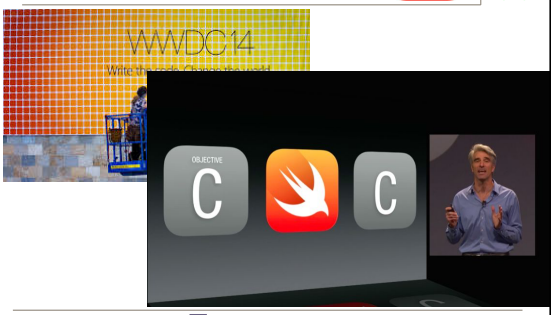
DePaul University 8

Objective-C 2.0

- Released in October 2007
- New features
  - garbage collection (OS X only)
  - declared and synthesized properties
  - dot syntax
  - fast enumeration
  - etc.
- Clang/LLVM compiler
  - Automated Reference Counting (ARC)
  - Performance improvements

DePaul University 9

The Birth of Swift
WWDC 2014, June 2, 2014



DePaul University 10

Swift Version History

- Swift beta: June 2, 2014 (Xcode 6.0 beta)
  - Announced at WWDC 2014
- Swift 1.0: September 9, 2014 (Xcode 6.0)
- Swift 1.1: October 22, 2014 (Xcode 6.1)
- Swift 1.2: April 8, 2015 (Xcode 6.3)
- Swift 2.0: September 21, 2015 (Xcode 7.0)
  - Announced at WWDC 2015, June 8, 2015
- Swift 2.1: October 20, 2015 (Xcode 7.1) — Official stable release
- Swift 2.2: December 3, 2015 — Development branch
  - Open source (OSX, Linux): Swift.org

DePaul University 11

Hello, Swift!

Swift Programming Language

- Modern design and rich features
  - Fast, safe, and powerful
- Syntactic style similar to Java, C++, C#
  - Natural, concise, and readable
- Evolved from Objective-C
  - Not backward compatible, but *interoperable*
  - All Objective-C frameworks are available in Swift

DePaul University 13

Hello, World!

- Let's start in Java

```
public class Hello {

    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }

}
```

DePaul University 14

Hello, World!

- … and simplify a little more. We have Swift

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

DePaul University 15

Hello, World!

- Oh, *one more thing*, semi-colon is optional

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

DePaul University 16

Hello, World!

- We can say hello in Chinese. Just as easy.

```
print("你好, 世界!")
```

DePaul University 17

Hello, World!

- Or in Emoji!

```
print("👋🌏!")
```

DePaul University 18

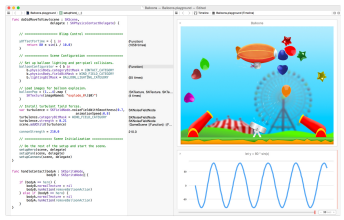
Running Swift Programs

- Playground
  - Interactive execution environment
  - Learning, experimenting, prototyping
  - Support GUI for iOS and OS X
- REPL (Read-Eval-Print-Loop)
  - Command-line, no GUI support
  - Available in Xcode console or OS X Terminals
- Xcode project
  - Full-blown projects for iOS or OS X

The Playground

The Xcode Playground

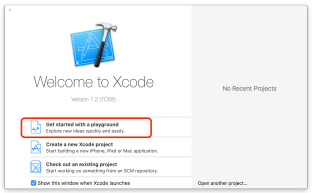
- Introduced in Xcode 6.0
- An interactive execution environment for Swift
  - Learning
  - Exploring
  - Visualizing



Launch Playground

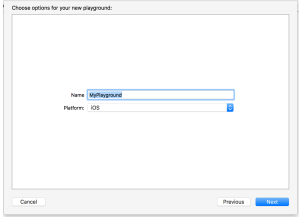
- Launch Xcode,
  - Select "*Get started with a playground*"
- If Xcode is already running
  - From the Xcode menu bar

File | New | Playground …



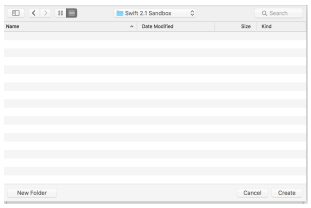
Launch Playground

- Choose a name
- Choose a platform:
  - "iOS", "OS X", or "tvOS"
  - Choose "iOS"
  - You may choose any of the three, for simple programs without UI
- Click "Next"

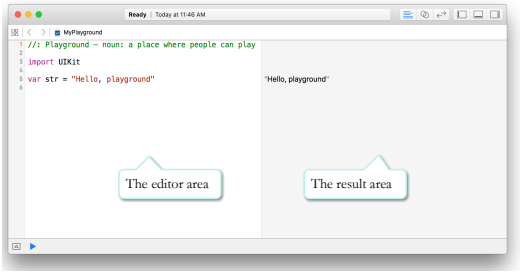


Launch Playground

- Choose a folder to save the playground file
  - Playground files are saved with the .playground suffix
- Click "Create"



New Playground



Let's Play

- The starting template

```
// Playground - noun: a place where people can play

import UIKit

var str = "Hello, playground"
```

A comment.

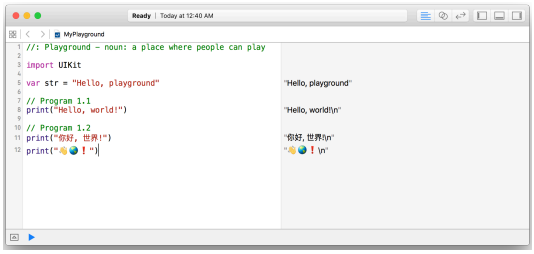
Import the **UIKit** framework (for iOS). Includes the **Foundation** framework

A variable declaration.

The Hello World Program

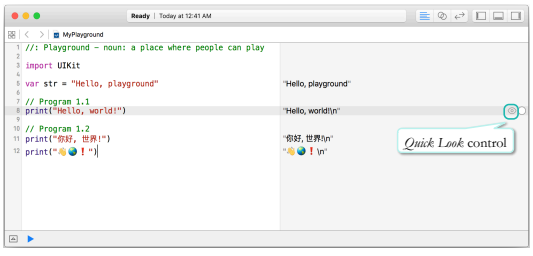
```
import UIKit

var str = "Hello, playground"

// Program 1.1
print("Hello, world!")

// Program 1.2
print("你好, 世界!")
print("👋🌏❗")
```


"Hello, World!" In Playground



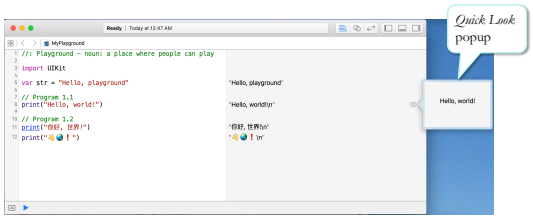
Quick Look

- Left: "Quick Look" control



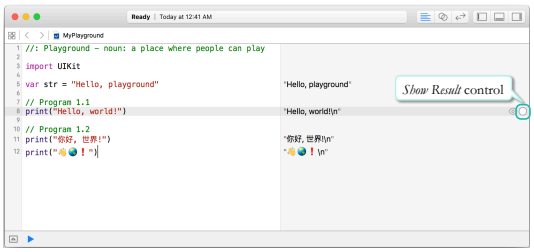
Quick Look Popup

- A popup displaying the results



Show Inline Results

- Right: "Show Result"



DePaul University

Inline Results

- Click on the "Show Result" control

DePaul University 32

The Debug Area

DePaul University 33

Swift Code in the Playground

- You can edit and run Swift code in a Playground
- Swift code in a Playground may contain
  - Constant and variable declarations
  - Statements
  - Values and expressions
  - Function declarations, and
  - Class declarations, etc.

DePaul University 34

Introduction to Swift

Whitespace and Comments

- (Most) Whitespace and comments are ignored
- Single-line comments

```
// This is a single-line comment
```

  - Same as C++, Java
- Multi-line comments

```
/* This is a long multi-line comment …
   It spans across several lines. */
```

  - Same as C, C++, Java
- In Swift, multi-line comments *can be nested.*

```
/* Outer comment begins …
   /* Nested comment … */
   … Outer comment ends */
```

DePaul University 36

Basic Data Types

- Swift provides a rich set of basic data types, include
  `Int` `Double` `String` `Character` `Bool`
- Basic data types are *structs*
  - First class citizens (compared to objects) without the overhead
  - Similar syntax and capabilities as classes
  - Values of basic types can be used anywhere objects are expected
    - No conversion to objects necessary
    - No boxing and unboxing necessary
- Basic data types are *value types*
  - Simpler semantics and memory management. Not *ARC managed*

Stay tuned for *structs* and *classes*

Stay tuned for full discussions on *value types* and *ARC*

The Integer Types

- `Int` `Uint`
  - *Signed* and *unsigned* integers
  - Either 32 or 64 bit integer, based on the platform
- Other integer types (with specific sizes)
  `Int8` `Int16` `Int32` `Int64`
  `UInt8` `UInt16` `UInt32` `UInt64`
- Literals (_ are for readability only, ignored)
  `123` `2014` `-32` `1_000_000`
  `0b0111_1011` `0o173` `0x7B`
- Operators
  `+` `-` `*` `/` `%` `<` `>` `<=` `>=` `==` `!=`

Decimal by default

Binary, octal, and hex-decimal values (equal to decimal 123)

The Floating-Point Types

- `Double` `Float`
  - 64 and 32 bit floating-point numbers
- Literals
  `3.1415927` `-100.0`
  `1.2345e2` `1E-10`
- Operators
  `+` `-` `*` `/` `%` `<` `>` `<=` `>=` `==` `!=`

The String and Character Type

- `Character`
  - A single Unicode character. Full Unicode support
- `String`
  - A sequence of Unicode character
- Literals
  `"Hello, World!"` `"A"` `"t"`
- Escape sequences
  `\n` `\r` `\t` `\"` `\'` `\\` `\0`
- Operators
  `+` String concatenation
  `<` `>` `<=` `>=` `==` `!=` **Compare string** *values*

Single character string and character literals are indistinguishable.

The Boolean Type

- `Bool`
  - Boolean values
  - Different and incompatible with `Int`
- Literals
  `true` `false`
- Operators
  `&&` `||` `!` `==` `!=`

Declarations

- Constant declaration
  `let` *Identifier* `:` *Type* (optional) `=` *Initial Value*
- Variable declaration
  `var` *Identifier* `:` *Type* (optional) `=` *Initial Value* (optional)
- Constants are immutable. Variables are mutable.

Example: Declarations

```
let distanceToMoon = 384_400        // km
let earthGravityAcceleration = 9.8  // m/s/s
let languageName = "Swift"
let swiftIsAwesome = true

var total = 100
var velocity = 30.5
var statusMessage = "Success"
var isComplete = false
```

- Types are omitted, since they can be inferred from the initial values

Types of Constants & Variables

- Constants and variables
  - Declaration before use
  - Initialization before reference
- Types are optional in declarations
  - Types can be inferred from initial value
- Static typing
  - Types are determined at the time of declaration
    - Explicitly declared or *inferred*
  - Types *cannot* be changed after declaration

Explicit Type Declaration

- Explicit type declarations are necessary sometimes
  - when the inferred type is different from the intended type

```
let currencySymbol : Character = "$"

let ten : Int32 = 10
var itemCount : UInt = 0

let length : Float = 100.0
var totalAmount : Double = 0
```


Variable Declarations

```
var total : Int
var velocity : Double
var statusMessage : String
var isComplete : Bool

total = 100
velocity = 30.5
statusMessage = "Success"
isComplete = false
```


Mixed-Type Expressions

- All common arithmetic and comparison operators are supported
- Mixed-type operations are *not* allowed
  - Binary operations on values of the same type only
- No implicit type conversion, a.k.a. coercion
- Type conversion must be explicit

Type Name (*Expression*)

- Numeric literals are convertible to compatible types

String Interpolation

- A string literal may contain interpolated expressions of any type

  \(*Expression*)

- Each interpolated expression is evaluated at runtime and replace by its value (converted to string)

```
let a = 3, b = 5
let result = "\(a) times \(b) is \(a * b)"
```

  - Result: "3 time 5 is 15"

Control Statements

Control Statements

- All common control statements are supported
  - The *if* statement, and the *if-else* statement
  - The *while* loop
  - The *repeat-while* loop (formerly the *do-while* loop)
  - The *for* loop
  - The *for-in* loop, a.k.a. *for-each* loop, **new**
  - The *switch* statement, **enhanced**

Stay tuned

Conditional Statements

```
var x = 7.5, y = 3.0
if y != 0 {
    print("\(x)/\(y) is \(x/y)")
} else {
    print("\(x)/\(y) is undefined")
}
```

Parentheses around the condition are *optional*

Braces for the branches are *required*.

- Similar rules on parentheses and braces for *while*, *repeat-while*, *for*, and *for-in* loops

For-In Loop

```
for Variable in Collection {
    Statements
}
```

- The *Collection* can be
  - an *array*,
  - a *set*,
  - a *dictionary*, or
  - a *range*.

Range Expression

- Define a range of consecutive integers
  - *Start index* and *end index* must be integers
  - *Start index* ≤ *end index*
- *Closed* range

Expression `...` *Expression*

From the *start index* up to and include the *end index*.

- *Half-open* range

Expression `..<` *Expression*

From the *start index* up to the *end index* - 1.

Examples of For-In Loop

```
let N = 100
var sum = 0
for i in 1 ... N {
    sum += i
}
print("sum = \(sum)")
```

Calculate the sum of 1 + 2 + … + 100

```
var powerOfTwo = 1
for _ in 1 ... 10 {
    powerOfTwo *= 2
}
print(powerOfTwo)
```

Anonymous loop variable _

Array Literals

- An *array* is an ordered collection of values of a uniform type
  - Not the same as arrays in C++ and Java
  - Can be variable size
    - Can grow and shrink as needed
  - Behave like a *List* in Java
- Array literals

[$Expression_1$, $Expression_2$, ...]

DePaul University 55

Example: Linear Search

```
let cities = [
  "London", "Berlin", "Madrid", "Rome", "Paris", … ,
  "Marseille", "Amsterdam", "Valencia", "Zagreb" ]
let searches = [ "Rome", "Venice", "Barcelona", "Sevilla" ]
for city in searches {
    var found = false
    for c in cities {
        if c == city {
            found = true
            break
        }
    }
    print(city, "is", found ? "found" : "not found")
}
```

DePaul University 56

Next …

- More Swift
- Functions
- Classes and objects

✧ Xcode, iOS, WatchOS are trademarks of Apple Inc.

DePaul University 57