# Swift 简介

极客时间

扫码试看/订阅

《Swift核心技术与实战》视频课程

# 内容综述-基础语法和特性

- Swift 简介
- 基本数据类型
- 运算符和表达式
- 流程控制
- 集合类
- 函数和闭包
- 面向对象编程
- 泛型

# 内容综述-多线程

- 多线程
  - 理论与实践
  - 玩转 GCD
  - 基础设施和编程范式

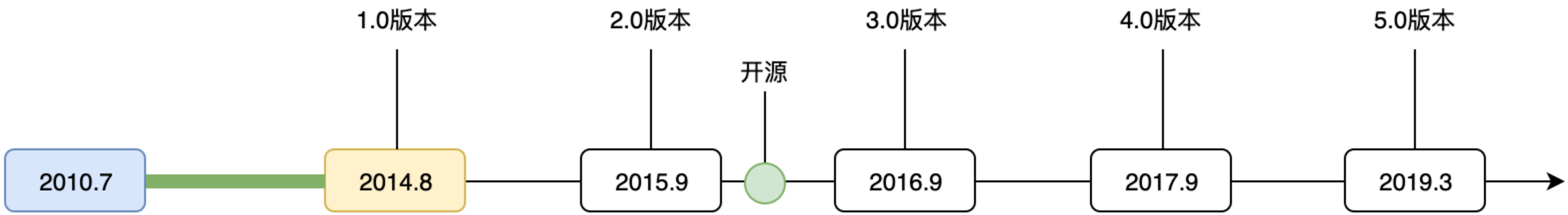
# 内容综述-其他话题

- 其他话题
  - 常见第三方库、SwiftUI
  - Swift 和 Objective-C 混编
  - Swift 运行时
  - ……

# 内容综述-实战

- 项目实战

# Swift 发展历史




- 从 2014 年至今，已经有 15 个版本发布，其中 5 个大版本，10 个小版本
- 与之对比的是 Objective-C 从80年代至今，只有两个版本
- 2015 年 12 月 Swift 正式开源，目前 Swift 可以应用到多个领域，甚至连 TensorFlow 也有 Swift 语言版本

# Swift 2

- Error handling 增强
- guard 语法
- 协议支持扩展
- ……

# Swift 3

- 新的 GCD 和 Core Graphics
- NS 前缀从老的 Foundation 类型中移除
- 内联序列函数 sequence
- 新增 fileprivate 和 open 两个权限控制
- 移除了诸多弃用的特性，比如 ++、- -（两个 - ）运算符等
- ……

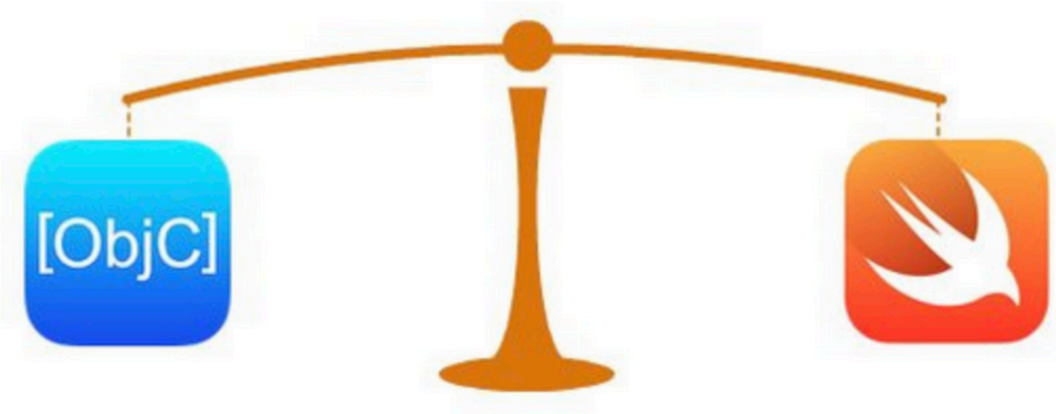
# Swift 4

- extension 中可以访问 private 的属性
- 类型和协议的组合类型
- Associated Type 可以追加 Where 约束语句
- 新的 Key Paths 语法
- 下标支持泛型
- 字符串增强
- ……

# Swift 5

- ABI 稳定
- Raw strings
- 标准库新增 Result
- 定义了与 Python 或 Ruby 等脚本语言互操作的动态可调用类型
- ……

# Swift VS Objective-C



# 编程范式

- Swift 可以面向协议编程、函数式编程、面向对象编程。
- Objective-C 以面向对象编程为主，当然你可以引入类似ReactiveCocoa的类库来进行函数式编程。

# 类型安全

- Swift 是一门类型安全的语言。鼓励程序员在代码中清楚明确值的类型。如果代码中使用一个字符串 String，那么你不能错误地传递一个整型 Int 给它。因为 Swift 是类型安全的，它会在代码编译的时候做类型检查，并且把所有不匹配的类型作为一个错误标记出来。这样使得程序员在开发中尽可能早地发现和修正错误。

- 而 Objective-C 则不然，你声明一个 NSString 变量，仍然可以传一个 NSNumber 给它，尽管编译器会抱怨，但是你仍然可以作为 NSNumber 来使用它。

# 值类型增强

- 在 Swift 中，典型的有 struct、enum 以及 tuple 都是值类型。而平时使用的 Int、Double、Float、String、Array、Dictionary、Set 其实都是用结构体实现的，也是值类型。

- Objective-C中，NSNumber、NSString以及集合类对象都是指针类型。

# 枚举增强

- Swift 的枚举可以使用整型、浮点型、字符串等，还能拥有属性和方法，甚至支持泛型、协议、扩展等等。
- Objective-C 里面的枚举则鸡肋很多。

# 泛型

- Swift 中支持泛型，也支持泛型的类型约束等特性。
- 苹果推出了 Swift 2.0 版本，为了让开发者从 Objective-C 更好地过渡到 Swift 上，苹果也为 Objective-C 带来了 Generics 泛型支持，不过 Objective-C 的泛型约束也仅停留在编译器警告阶段。

# 协议和扩展

- Swift 对协议的支持更加丰富，配合扩展（extension）、泛型、关联类型等可以实现面向协议编程，从而大大提高代码的灵活性。同时，Swift 中的 protocol 还可以用于值类型，如结构体和枚举。

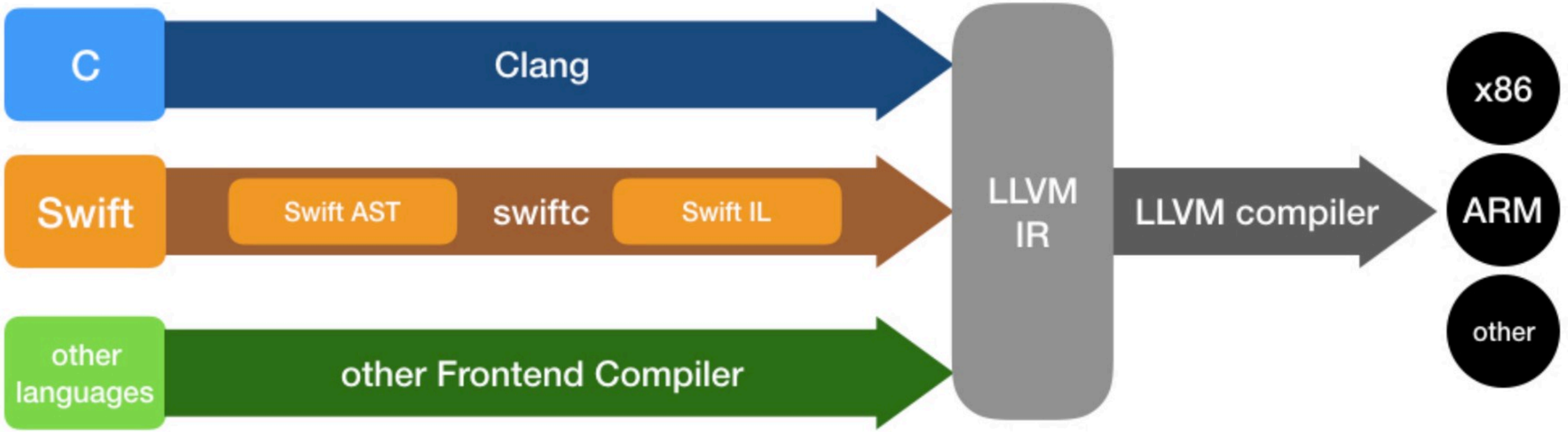
- Objective-C 的协议缺乏强约束，提供的 optional 特性往往成为很多问题的来源，而如果放弃 optional 又会让实现代价过大。

# 函数和闭包

- Swift 函数是一等公民，可以直接定义函数类型变量，可以作为其他函数参数传递，可以作为函数返回值返回。
- Objective-C 里面函数仍然是次等公民，需要 selector 封装或者使用block才能模拟 Swift 中类似的效果。

# Swift 命令行编译

# 编译过程


C
Clang
Swift
Swift AST
swiftc
Swift IL
other languages
other Frontend Compiler
LLVM IR
LLVM compiler
x86
ARM
other

# 编译过程


noahs.swift

noahs.swift 〉 main()

```
func main() {

    let 🌍 = "🐶🐴🐱🐭🐹🐰🐸🐯🐨🐻🐷🐽🐮🐗🐵🐒🐴🐑🐘🐼🐧🐦🐤🐥🐣🐔🐍🐢🐛🐝🐜
        🐌🐞🐙🐚🐠🐟🐬🐳🐋🐄🐏🐀🐃🐅🐇🐉🐎🐐🐓🐕🐖🐁🐂🐲🐡🐊🐫🐪🐆🐈🐩"

    var 🏭: [String] = []

    for 💕 in 🌍 .characters {
        🏭.append(String(💕) + String(💕))
    }

    print(🏭)

}

main()
```

# swiftc

- swiftc -o main.out main.swift
- Swift Abstract Syntax Tree (AST)　swiftc main.swift -dump-ast
- Swift Intermediate Language (SIL)　swiftc main.swift -emit-sil
- LLVM Intermediate Representation (LLVM IR)　swiftc main.swift -emit-ir
- Assembly Language　swiftc main.swift -emit-assembly

# swiftc

# Swift REPL

# REPL

- Xcode 6.1引入了另外一种以交互式的方式来体验 Swift 的方法。
- Read Eval PrintLoop，简称 REPL

```
➜  ~ swift
Welcome to Apple Swift version 5.0.1 (swiftlang-1001.0.82.4 clang-1001.0.46.4).
Type :help for assistance.
  1>
```

# REPL

```
➜  ~ swift
Welcome to Apple Swift version 5.0.1 (swiftlang-1001.0.82.4 clang-1001.0.46.4).
Type :help for assistance.
  1> let a = "100"
a: String = "100"
  2> Int(a)
$R0: Int? = 100
  3> let b = $R0 + 50
error: repl.swift:3:13: error: binary operator '+' cannot be applied to operands
 of type 'Int?' and 'Int'
let b = $R0 + 50
        ~~~ ^ ~~

repl.swift:3:13: note: expected an argument list of type '(Int, Int)'
let b = $R0 + 50
            ^


  3> let b = $R0! + 50
b: Int = 150
  4>
```

# REPL

```
➜  ~ swift
Welcome to Apple Swift version 5.0.1 (swiftlang-1001.0.82.4 clang-1001.0.46.4).
Type :help for assistance.
  1> func addTwoNumbers(num1:Int, num2:Int) -> Int {
  2.     return num1 + num2
  3. }
  4. let sum = addTwoNumbers(num1:100, num2:50)
sum: Int = 150
  5>
```

# REPL

- 退出 :quit
- 帮助 :help
- 将光标移动到当前行的开始处 Control+A
- 将光标移动到当前行的结束处 Control+E

```
The complete set of LLDB debugging commands are also available as described
below.  Commands must be prefixed with a colon at the REPL prompt (:quit for
example.)  Typing just a colon followed by return will switch to the LLDB
prompt.
Debugger commands:
  apropos           -- List debugger commands related to a word or subject.
  breakpoint        -- Commands for operating on breakpoints (see 'help b' for
                       shorthand.)
  bugreport         -- Commands for creating domain-specific bug reports.
  command           -- Commands for managing custom LLDB commands.
  disassemble       -- Disassemble specified instructions in the current
                       target.  Defaults to the current function for the
                       current thread and stack frame.
  expression        -- Evaluate an expression on the current thread.  Displays
                       any returned value with LLDB's default formatting.
  frame             -- Commands for selecting and examing the current thread's
                       stack frames.
  gdb-remote        -- Connect to a process via remote GDB server.  If no host
                       is specifed, localhost is assumed.
  gui               -- Switch into the curses based GUI mode.
  help              -- Show a list of all debugger commands, or give details
                       about a specific command.
  kdp-remote        -- Connect to a process via remote KDP server.  If no UDP
                       port is specified, port 41139 is assumed.
  language          -- Commands specific to a source language.
```

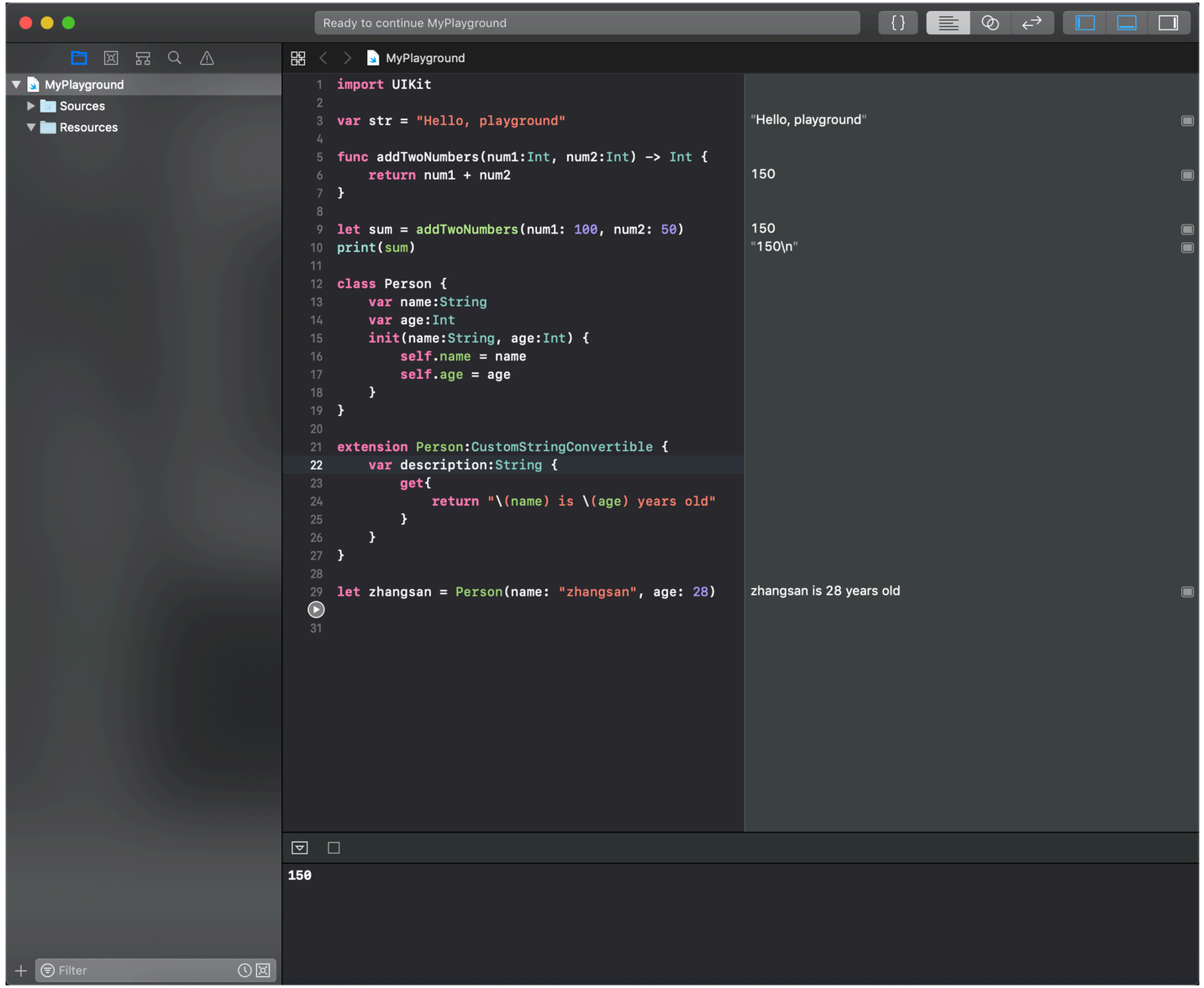
# REPL

# Playground

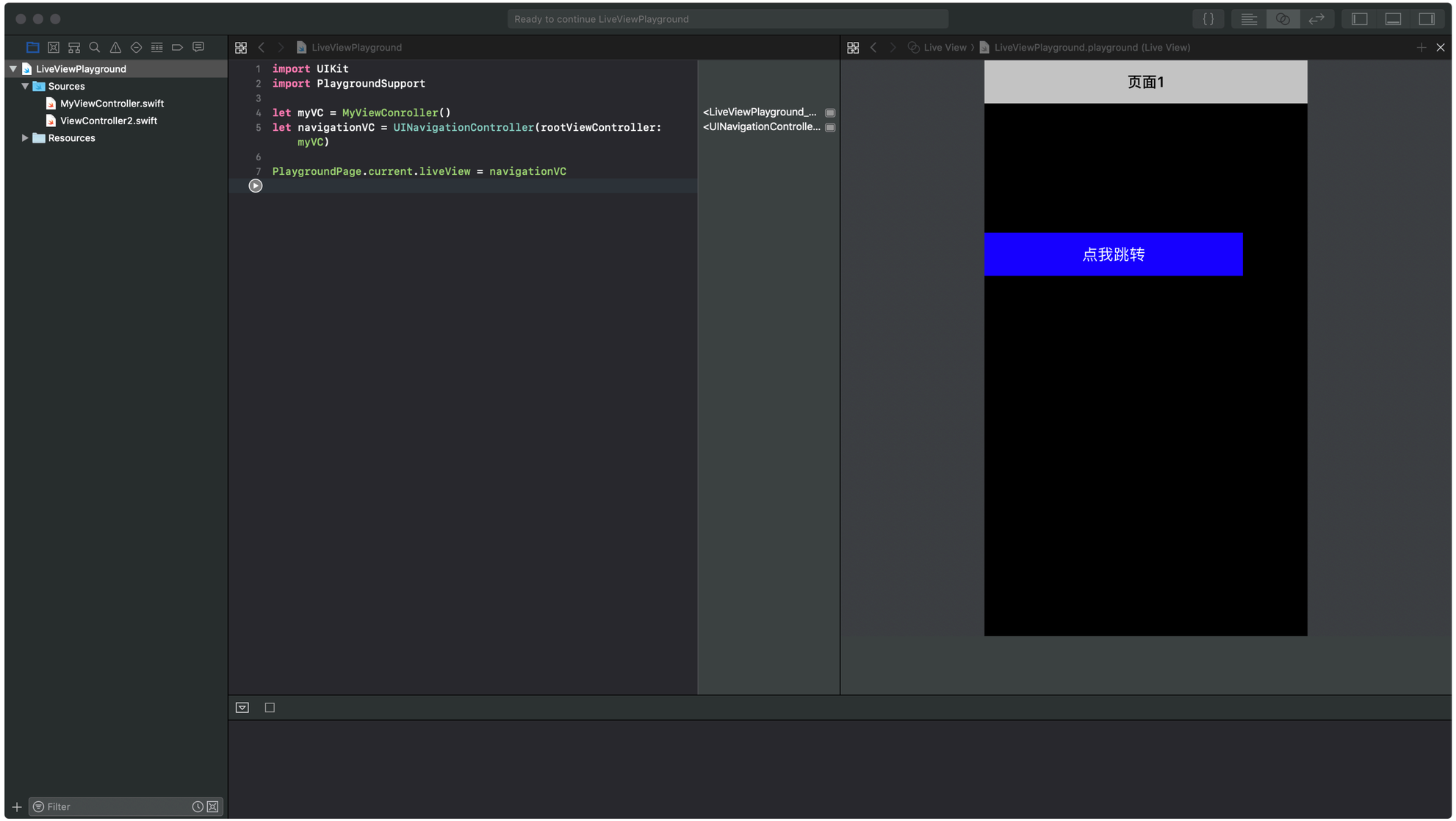
# Playground

- Swift Playground 首次公布于WWDC2016
- 最开始是为了让人人都能愉快的学习 Swift 编程
- 但发展至今, 这个工具越来越强大
- iPad APP Playgrounds

# Playground


Ready to continue MyPlayground
MyPlayground
MyPlayground
Sources
Resources
import UIKit
var str = "Hello, playground"
"Hello, playground"
func addTwoNumbers(num1:Int, num2:Int) -> Int {
return num1 + num2
}
150
let sum = addTwoNumbers(num1: 100, num2: 50)
print(sum)
150
"150\n"
class Person {
var name:String
var age:Int
init(name:String, age:Int) {
self.name = name
self.age = age
}
}
extension Person:CustomStringConvertible {
var description:String {
get{
return "\(name) is \(age) years old"
}
}
}
let zhangsan = Person(name: "zhangsan", age: 28)
zhangsan is 28 years old
150
Filter

# Playground-LiveView



极客时间

扫码试看/订阅

《Swift核心技术与实战》视频课程