

# Introduction to iOS Development

Session 101

Alex Telek

Clarence Ji

KCL Tech iOS Engineers

# What You Will Learn



iOS 9



OS X 10.11



watchOS 2



tvOS 9



Swift 2.0



Xcode 7



# iOS Sessions

Every week - Tuesday @ 6:30pm, JCMB B.16

Start with basics (iOS, OS X, watchOS, tvOS, Swift, Xcode)

UI

Custom Views and Animations

Concurrency and Multithreading

3D Touch

Foundation, UIKit, CoreLocation, CoreImage, CoreData, MapKit, CloudKit, WatchConnectivity, ClockKit....

Quickly becomes challenging

Learn by doing, not by listening!

Slack #ios-programming



# WWDC16

World Wide Developer Conference in San Francisco

300 Student Scholarship Recipients

## 1. Create an App (Developer Account)

- Development Projects
- Educational background
- Professional background
- Technical skills
- Interest

## 2. Judging

- Technical accomplishment
- Creativity of ideas expressed in the app
- Technical/work experience



Deadline: April, 2016



# Xcode 7



Swift 2.0

watchOS 2

tvOS 9

OS X

Live design

Visual debugging

Performance testing

UI Testing

Code Coverage

Address Sanitizer





# iOS 9



3D Touch

Slide Over

Split View

Picture in Picture

App Thinning

Safari View Controller

Search API



# watchOS 2



Digital Crown

Audio

Animation APIs

Taptic Engine

Core Motion

ClockKit

Time Travel

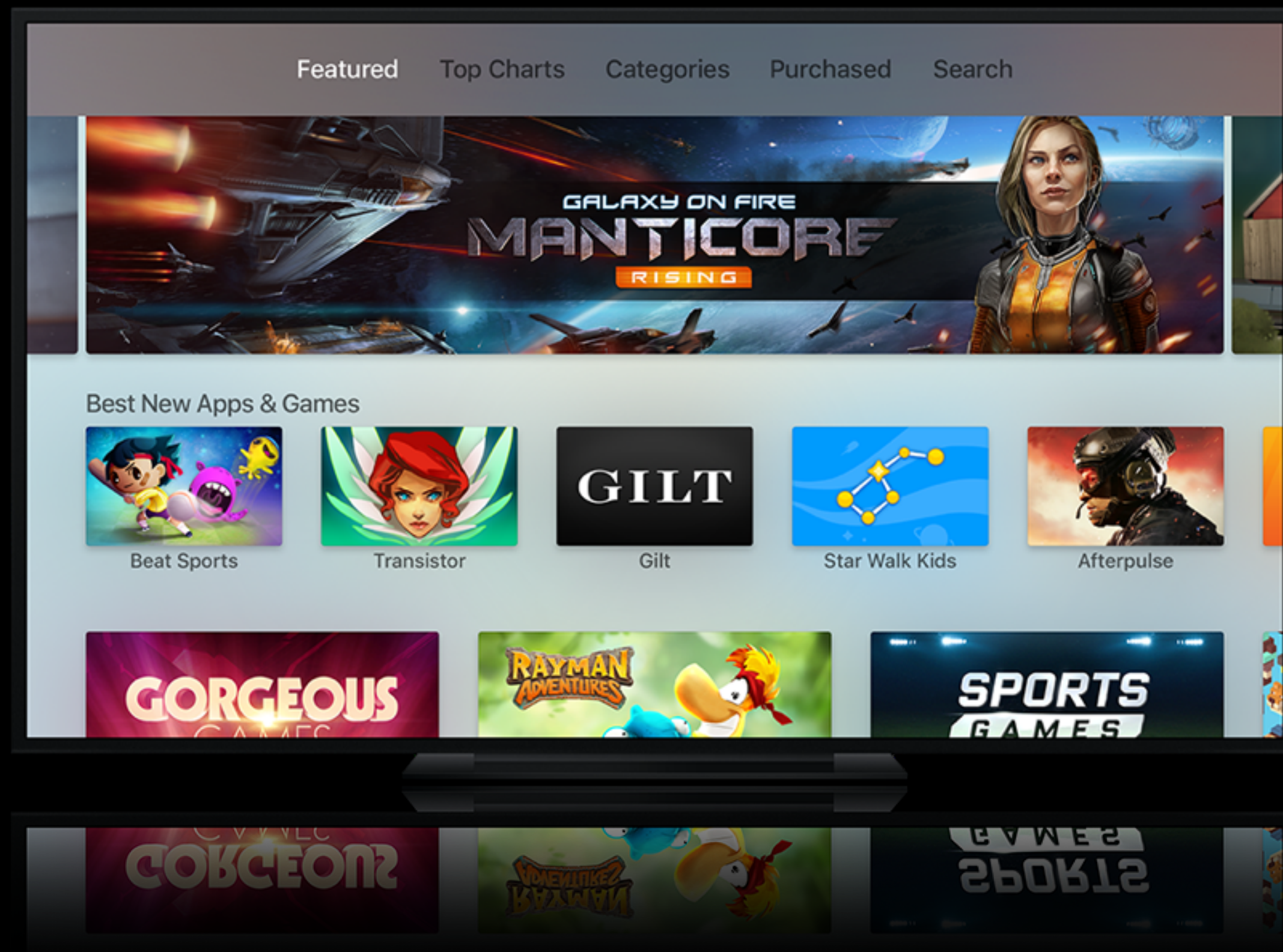
WatchConnectivity

NSURLSession





# tvOS



Most Frameworks

Javascript + TVML

Remote

Siri





*Demo*

Xcode 7

# Swift 2.0

SAFE

MODERN

POWER



```
#include <stdio.h>

int main()
{
    printf("Hello, KCLTech\n");
    return 0;
}
```



# Variables

```
var languageName: String = "Swift"
```





# Constants and Variables

SAFE

```
let languageName: String = "Swift"  
var version: Double = 1.0  
let introducedIn: Int = 2014  
let isAwesome: Bool = true
```



# Type Inference

SAFE

```
let languageName := String "Swift"  
var version := Double = 1.0  
let introducedIn := Int 2014  
let isAwesome := Boolean = true
```



# Unicode Names

```
let languageName = "Swift"  
var version = 1.0  
let introducedIn = 2014  
let isAwesome = true  
let  $\pi$  = 3.14159265  
let 🐶🐮 = "dogcow"
```



# Combining Strings and Characters

```
let dog = "dog"  
let cow = "cow"  
let dogCow = dog + cow  
// dogCow is "dogcow"
```





# Building Complex Strings

POWER

```
let a = 3
```

```
let b = 5
```

```
// "3 times 5 is 15"
```

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



# String Mutability

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

```
let constantString = "Horse"  
constantString += " and carriage"  
//error
```



# Array and Dictionary

```
var names = ["Anna", "Brian", "Jack"]
```

```
var numberOfLegs = ["ant": 6, "snake": 0]
```



# Typed Collections

```
var names = ["Anna", "Brian", "Jack"]
```





# Typed Collections

```
var names = ["Anna", "Brian", "Jack", 42]
```



# Typed Collections

```
var names = ["Anna", "Brian", "Jack", true]
```



# Typed Collections

```
var names: Int[] = ["Anna", "Brian", "Jack"]
```

```
var ages = Int[]()
```

```
let values = Int[](count: 5, repeatedValue: 1)
```



# Typed Collections

SAFE

```
var names = ["Anna", "Brian", "Jack"]  
// an array of String values
```

```
var numberOfLegs = ["ant": 6, "snake": 0]  
// a Dictionary with String keys and Int values
```





# Loops

```
while hungry {  
    eatCake()  
}
```

```
for var i = 0; i < 10; i++ {  
    eat(i)  
}
```

```
var index = 0  
repeat {  
    index++  
} while index < 5
```



# For-In: Strings and Characters

POWER

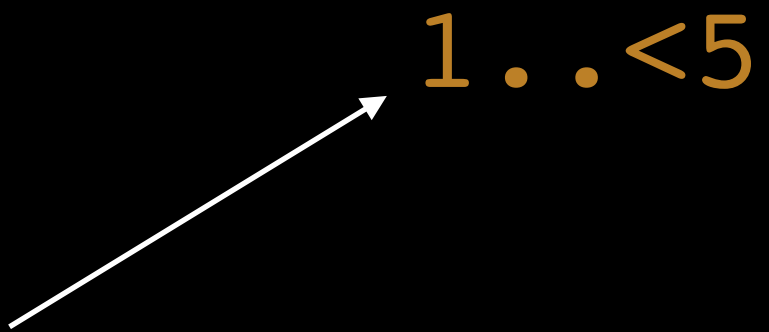
```
for character in "🐭🐭🐭🐭🐭" {  
    print(character)  
}
```

🐭  
🐭  
🐭  
🐭  
🐭



# For-In: Ranges

```
for number in 1...5 {  
    print("\(number) times 4 is \(number*4)")  
}
```



```
1 times 4 is 4  
2 times 4 is 8  
3 times 4 is 12  
4 times 4 is 16  
5 times 4 is 20
```



# For-In: Arrays

POWER

```
for name in ["Anna", "Brian", "Jack"] {  
    print("Hello, \(name)!")  
}
```

```
Hello, Anna!  
Hello, Brian!  
Hello, Jack!
```



# For-In: Dictionaries

```
var numberOfLegs = ["ant": 6, "snake": 0, "cheetah": 4]
```

```
for (animal, leg) in numberOfLegs {  
    print("\(animal)s have \(leg) legs")  
}
```

```
ants have 6 legs  
snakes have 0 legs  
cheetahs have 4 legs
```



# If Statements

```
if legCount == 0 {  
    print("It slides")  
} else {  
    print("It walks")  
}
```





# More Complex If Statements

```
if legCount == 0 {  
    print("It slides")  
} else if legCount == 1 {  
    print("It hops")  
} else {  
    print("It walks")  
}
```



# Switch

```
switch legCount {  
    case 0:  
        print("It slides")  
  
    case 1:  
        print("It slides")  
  
    default:  
        print("It slides")  
}
```



# Switch

```
switch legCount {  
    case 0:  
        print("It slides")  
  
    case 1,3,5,7,9:  
        print("It hops")  
  
    case 2,4,6,8,10:  
        print("It walks")  
  
    default:  
        print("No idea")  
}
```



# Switch

```
switch textField {  
    case userNameTextField:  
        print("You tapped the username text field")  
  
    case passwordTextField:  
        print("You tapped the password text field")  
  
    default:  
        print("You tapped some other object")  
}
```



# Switch

```
let point = (1, -1)

switch point {
  case let (x, y) where x == y:
    print("x equals with y")

  case let (x, y) where x == -y:
    print("x is the abs of y")

  case let (x, y):
    print("x and y are two coordinates")
}
```



# Functions

```
func sayHello() {  
    print("Hello!")  
}
```

```
sayHello()
```

Hello



# Functions with Parameters

```
func sayHello(name: String) {  
    print("Hello \(name)!")  
}
```

```
sayHello("WWDC")
```

Hello, WWDC!



# Functions with Parameters

```
func sayHello(name: String, age: Int) {  
    print("Hello \(name), \(age)!")  
}
```

```
sayHello("Alex", age: 21)
```

```
func sayHello(name: String, _ age: Int) {  
    print("Hello \(name), \(age)!")  
}
```

```
sayHello("Alex", 21)
```





# Functions with Parameters

```
func addListOfParams(params: String...) {  
    // Do something with params  
}
```

```
addListOfParams("Name", "Alex", "Age", "21")
```



# Returning Values

```
func sayHello(name: String) -> String {  
    return "Hello " + name  
}
```

```
let greeting = sayHello("WWDC")
```

```
println(greeting)
```

Hello, WWDC



# Returning Multiple Values

MODERN

```
func refreshWebPage() -> (Int, String) {  
    //...try to refresh...  
  
    return (200, "Success")  
}
```



# Functions in General

```
func functionName(variableName: ParamType) -> ReturnType {  
    return Variable as ReturnType  
}
```



*Demo*

Swift in Playground

