The platform - iOS

- Mobile OS.
- Developed by Apple
- First released in 2007
- Main version-iOS5.X (2011),iOS7.X(2013),iOS8.X(2014)
- Current version iOS 9.3(Beta)
- iOS Version Status 2016-03-09

All Platforms:

9.X	77.5%
8.X	11.5%
7.X	6.6%
6.X	2.9%
5.X	1.5%
4.X	0.1%

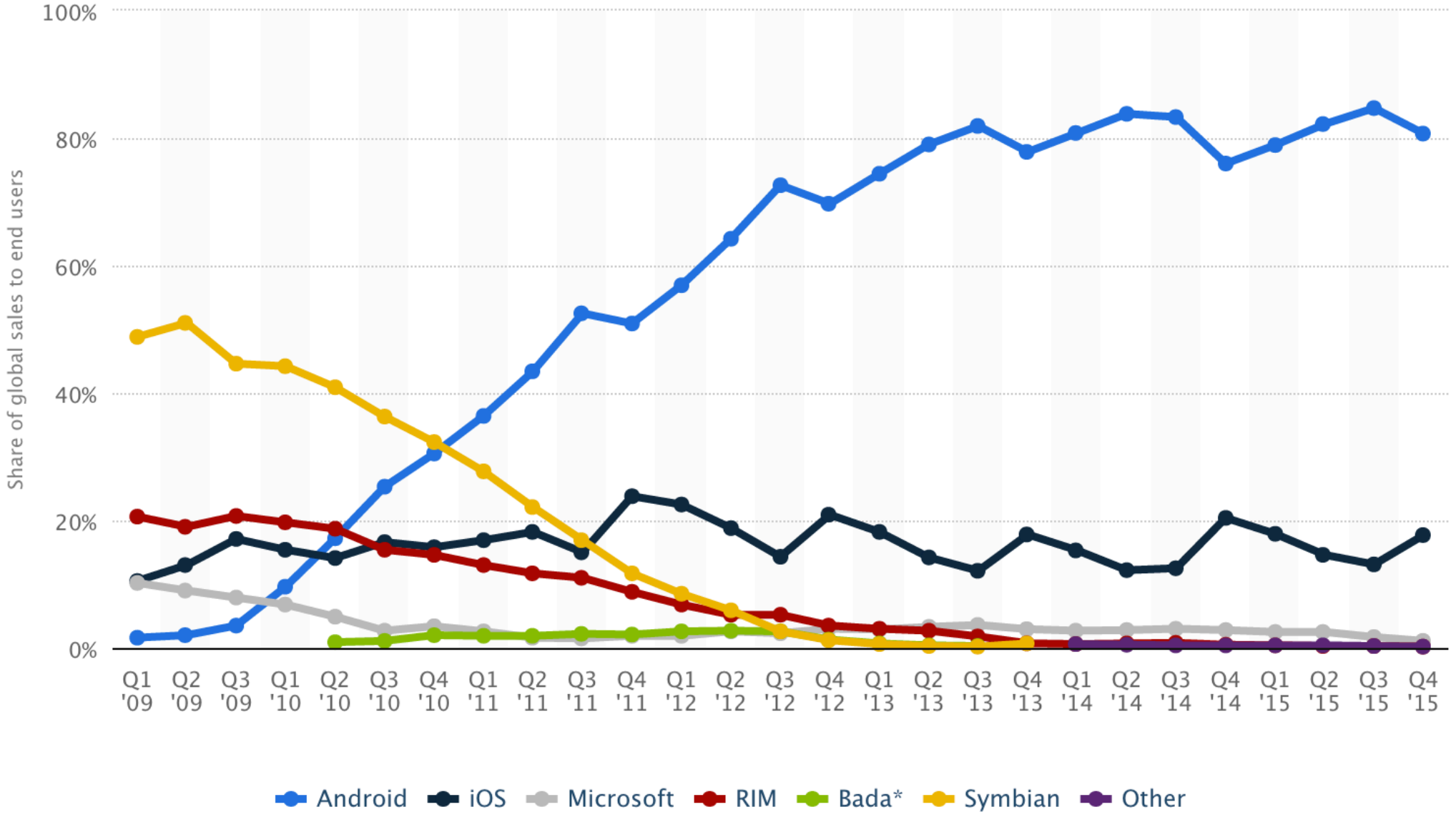
iOS Devices

iphone

ipad & ipad pro

Apple
watch

Global market share



Smartphone OS Sales Share (%)

Germany	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	71.0	74.2	3.2
iOS	18.7	19.3	0.6
Windows	8.9	5.9	-3.0
Other	1.4	0.6	-0.8

GB	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	51.5	52.6	1.1
iOS	39.9	38.6	-1.3
Windows	7.5	8.6	1.1
Other	1.1	0.2	-0.9

France	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	65.3	71.8	6.5
iOS	20.2	19.3	-0.9
Windows	13.0	7.8	-5.2
Other	1.5	1.1	-0.4

Italy	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	66.6	78.1	11.5
iOS	18.3	14.4	-3.9
Windows	13.2	7.2	-6.0
Other	1.9	0.3	-1.6

Spain	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	86.7	87.8	1.1
iOS	10.4	11.4	1.0
Windows	2.5	0.8	-1.7
Other	0.4	0.0	-0.4

USA	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	51.9	58.2	6.3
iOS	42.8	39.1	-3.7
Windows	4.3	2.6	-1.7
Other	1.0	0.1	-0.9

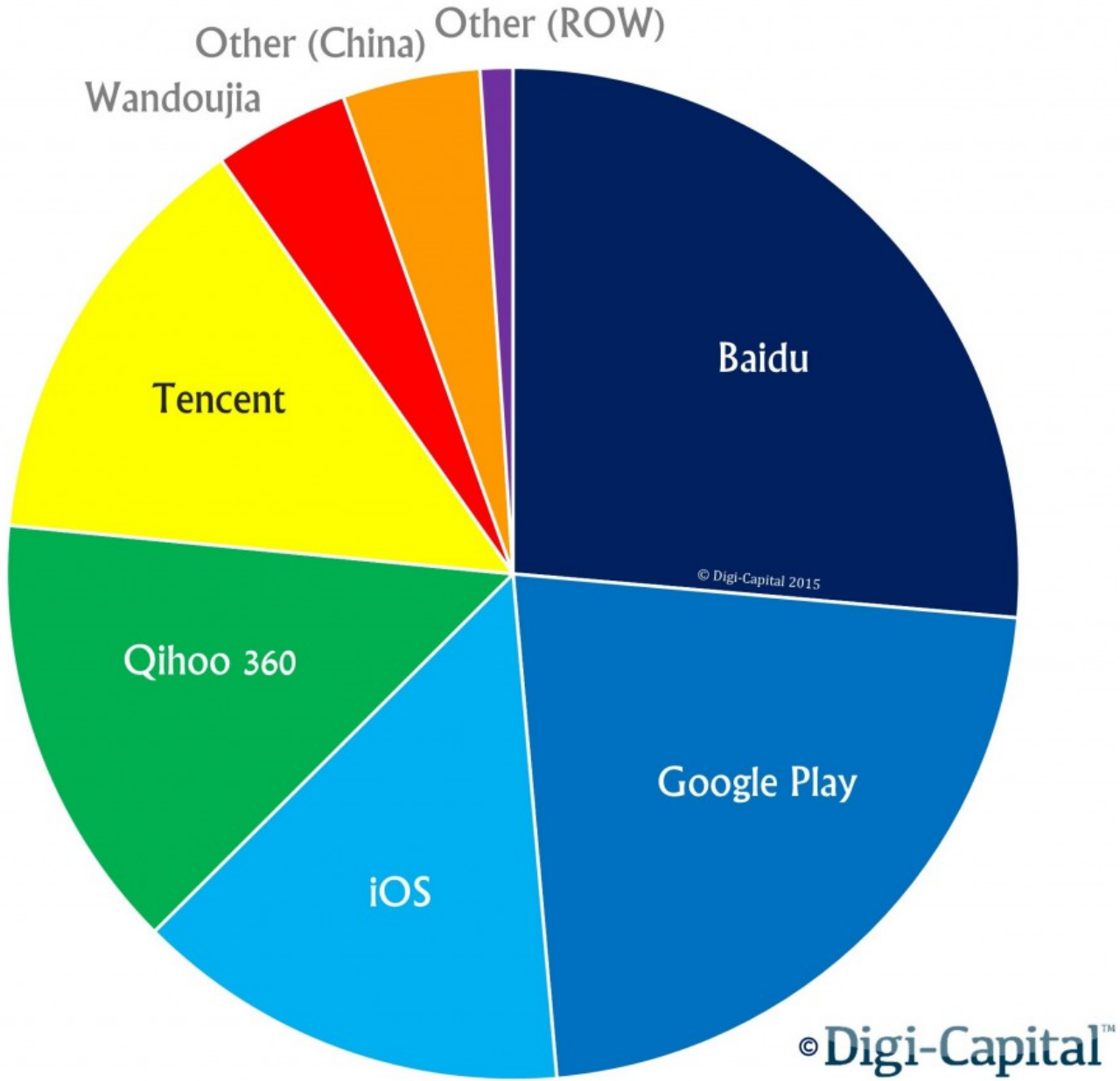
China	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	74.2	73.9	-0.3
iOS	23.9	25.0	1.1
Windows	1.0	0.9	-0.1
Other	0.9	0.2	-0.7

Australia	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	46.9	52.6	5.7
iOS	42.4	41.2	-1.2
Windows	8.7	5.4	-3.3
Other	2.0	0.8	-1.2

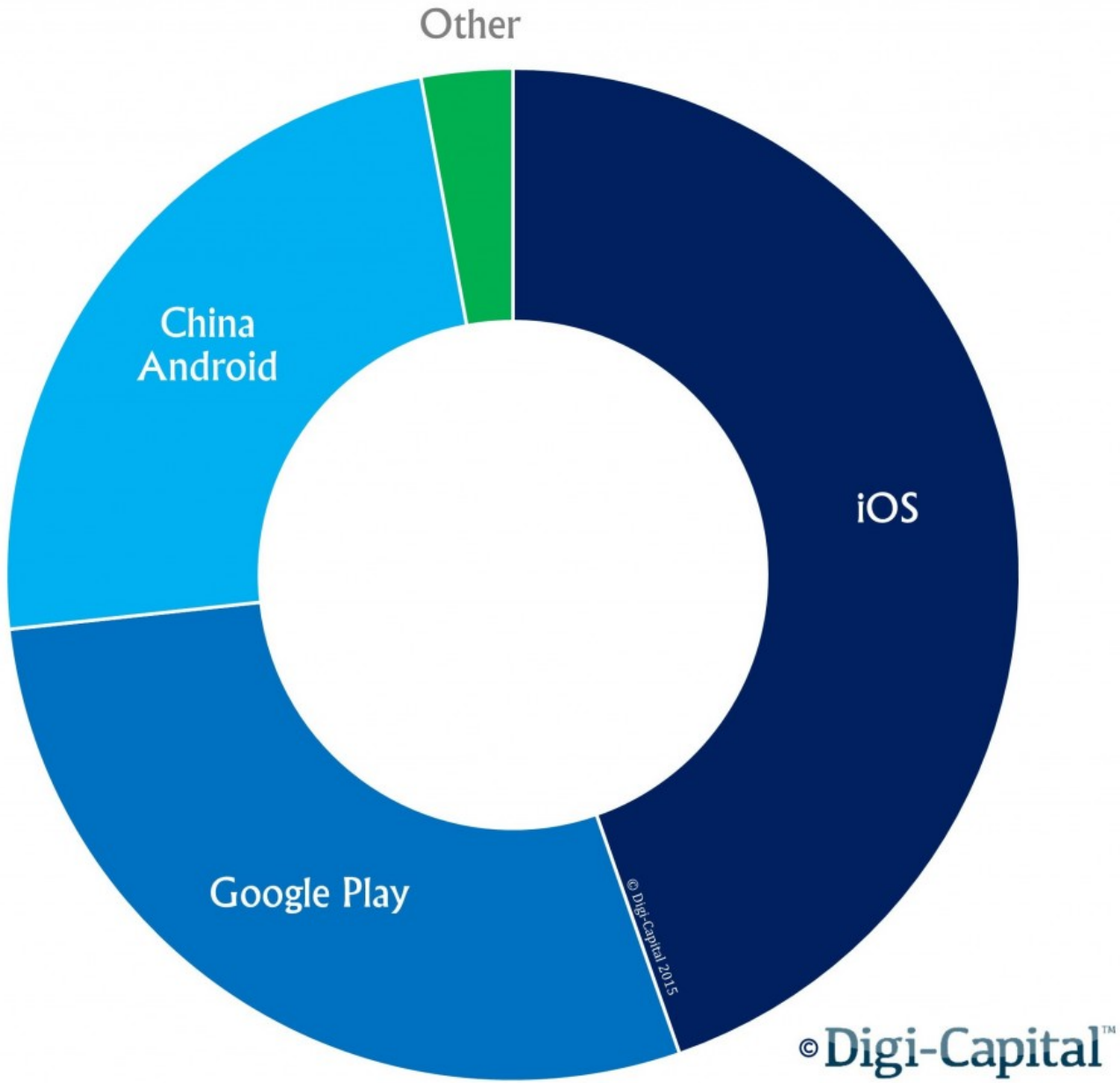
Japan	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	45.4	48.7	3.3
iOS	52.4	50.3	-2.1
Windows	0.2	0.5	0.3
Other	2.0	0.5	-1.5

EU5	3 m/e Jan 15	3 m/e Jan 16	% pt. Change
Android	67.1	72.8	5.7
iOS	22.0	20.3	-1.7
Windows	9.6	6.4	-3.3
Other	1.3	0.5	-0.8

Global app stores download volume share 2014



Global app stores revenue value share 2014



©Digi-Capital™

Pros and cons

- +Highest revenue for mobile OS
- +Little fragmentation-- iPhone iPad iWatch
- +Runs on high-end devices
- +Big developer community and excellent support
- +Many open-source libraries available
- −Strictly controlled by Apple
- −Development only possible in Mac OS.

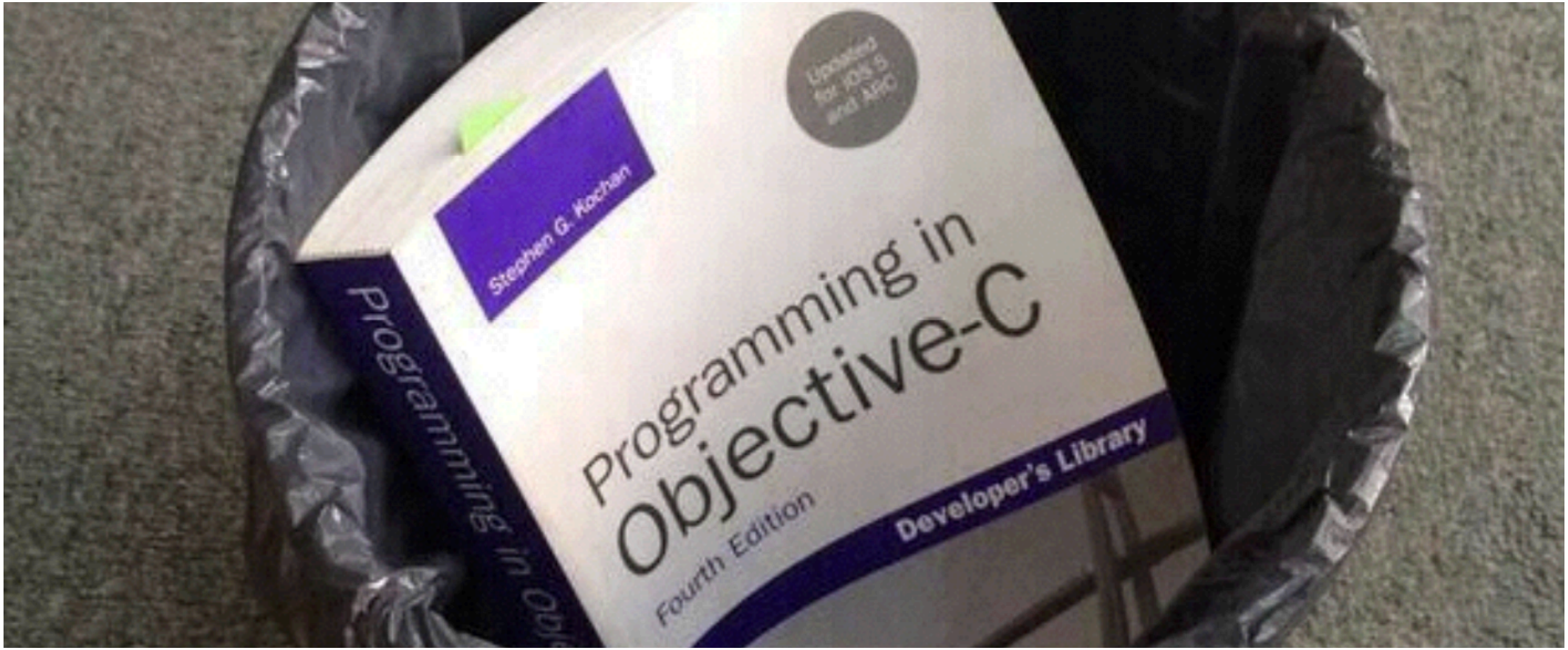
Native Apps vs. Hybrid Apps vs. HTML 5 Apps

- Native Apps: Mobile apps built using Objective-C,Swift (for iOS), Java (for Android), or .NET (for Windows), for a particular platform. The code is written for a specific platform and won't work for other platforms.
- Hybrid Apps: App built using HTML5, CSS3 and JavaScript, put into a native wrapper that enables it to access the device's hardware. Built using tools like Xamarin, PhoneGap,Appcelerator Titanium RhoMobile, Telerik AppBuilder, Appmethod, AppGyver, RAD Studio
- HTML5/Web Apps: Also built using HTML5, CSS3 and JavaScript. Similar to mobile sites, opens in mobile browsers. The user needs Internet to access the apps and such apps cannot be sold on apps stores. Also, apps cannot access on board hardware and software in mobiles.

	Native	Hybrid	HTML/Web
Cross-Platform/ Code Reusability	No	Yes	Yes
Development & Maintenance Cost	High	Low	Low
User Interface	High	Medium	Low
App Store Distribution	Yes	Yes	No
Offline Access	Yes	Yes	No
Native API Access	Complete	Partial-to-High	Low
Native Performance	Complete	Partial	Low

https://dzone.com/articles/native-apps-vs-hybrid-apps-vs-html-5-apps-what-wor

Swift (programming language)


Updated for iOS 5 and ARC
Stephen G. Kochan
Programming in Objective-C
Fourth Edition
Developer's Library
Programming in Objective-C

Swift Programming Language

Swift was introduced at Apple's 2014 Worldwide Developers Conference (WWDC)

version 2.2 was made open source and made available under the Apache License 2.0 on December 3, 2015, for Apple's platforms and Linux.

Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks and the large body of existing Objective-C code written for Apple products.

Swift is a multi-paradigm, compiled programming language created for iOS, OS X, watchOS, tvOS and Linux development by Apple Inc.

变量与常量

Swift使用var声明变量，let声明常量。

```
var myVariable = 42
myVariable = 50
let myConstant = 42
```

类型推导

Swift支持类型推导（Type Inference），所以上面的代码不需指定类型，如果需要指定类型：

```
let explicitDouble : Double = 70
```

Swift不支持隐式类型转换（Implicitly casting），所以下面的代码需要显式类型转换（Explicitly casting）：

```
let label = "The width is "
let width = 94
let widthLabel = label + String(width)
```

字符串格式化

Swift使用\(item)的形式进行字符串格式化：

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples."
let appleSummary = "I have \(apples + oranges) pieces of fruit."
```

字符串格式化

Swift使用\(item)的形式进行字符串格式化：

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples."
let appleSummary = "I have \(apples + oranges) pieces of fruit."
```

数组和字典

Swift使用[]操作符声明数组（array）和字典（dictionary）：

```
var shoppingList = ["catfish", "water", "tulips", "blue paint"]
shoppingList[1] = "bottle of water"

var occupations = [
    "Malcolm": "Captain",
    "Kaylee": "Mechanic",
]
occupations["Jayne"] = "Public Relations"
```

一般使用初始化器（initializer）语法创建空数组和空字典：

```
let emptyArray = String[]()
let emptyDictionary = Dictionary<String, Float>()
```

如果类型信息已知，则可以使用[]声明空数组，使用[:]声明空字典。

控制流

概览

Swift的条件语句包含if和switch，循环语句包含for-in、for、while和do-while，循环/判断条件不需要括号，但循环/判断体（body）必需括号：

```
let individualScores = [75, 43, 103, 87, 12]
var teamScore = 0
for score in individualScores {
    if score > 50 {
        teamScore += 3
    } else {
        teamScore += 1
    }
}
```

可空类型

结合if和let，可以方便的处理可空变量（nullable variable）。对于空值，需要在类型声明后添加?显式标明该类型可空。

```
var optionalString: String? = "Hello"
optionalString == nil

var optionalName: String? = "John Appleseed"
var gretting = "Hello!"
if let name = optionalName {
    gretting = "Hello, \(name)"
}
```

灵活的switch

Swift中的switch支持各种各样的比较操作：

```
let vegetable = "red pepper"
switch vegetable {
case "celery":
    let vegetableComment = "Add some raisins and make ants on a log."
case "cucumber", "watercress":
    let vegetableComment = "That would make a good tea sandwich."
case let x where x.hasSuffix("pepper"):
    let vegetableComment = "Is it a spicy \(x)?"
default:
    let vegetableComment = "Everything tastes good in soup."
}
```

其它循环

for-in除了遍历数组也可以用来遍历字典：

```
let interestingNumbers = [
    "Prime": [2, 3, 5, 7, 11, 13],
    "Fibonacci": [1, 1, 2, 3, 5, 8],
    "Square": [1, 4, 9, 16, 25],
]
var largest = 0
for (kind, numbers) in interestingNumbers {
    for number in numbers {
        if number > largest {
            largest = number
        }
    }
}
largest
```

while循环和do-while循环：

```
var n = 2
while n < 100 {
    n = n * 2
}
n

var m = 2
do {
    m = m * 2
} while m < 100
m
```

Swift支持传统的for循环，此外也可以通过结合..（生成一个区间）和for-in实现同样的逻辑。

```
var firstForLoop = 0
for i in 0..3 {
    firstForLoop += i
}
firstForLoop

var secondForLoop = 0
for var i = 0; i < 3; ++i {
    secondForLoop += 1
}
secondForLoop
```

注意：Swift除了..还有...：..生成前闭后开的区间，而...生成前闭后闭的区间。

函数和闭包

函数

Swift使用func关键字声明函数：

```
01. func greet(name: String, day: String) -> String {
02.     return "Hello \(name), today is \(day)."
03. }
04. greet("Bob", "Tuesday")
```

通过元组（Tuple）返回多个值：

```
01. func getGasPrices() -> (Double, Double, Double) {
02.     return (3.59, 3.69, 3.79)
03. }
04. getGasPrices()
```

支持带有变长参数的函数：

```
01. func sumOf(numbers: Int...) -> Int {
02.     var sum = 0
03.     for number in numbers {
04.         sum += number
05.     }
06.     return sum
07. }
08. sumOf()
09. sumOf(42, 597, 12)
```

函数也可以嵌套函数：

```
func returnFifteen() -> Int {
    var y = 10
    func add() {
        y += 5
    }
    add()
    return y
}
returnFifteen()
```

作为头等对象，函数既可以作为返回值，也可以作为参数传递：

```
func makeIncrementer() -> (Int -> Int) {
    func addOne(number: Int) -> Int {
        return 1 + number
    }
    return addOne
}
var increment = makeIncrementer()
increment(7)
```

```
func hasAnyMatches(list: Int[], condition: Int -> Bool) -> Bool {
    for item in list {
        if condition(item) {
            return true
        }
    }
    return false
}
func lessThanTen(number: Int) -> Bool {
    return number < 10
}
var numbers = [20, 19, 7, 12]
hasAnyMatches(numbers, lessThanTen)
```

类和对象

创建和使用类

Swift使用class创建一个类，类可以包含字段和方法：

```
class Shape {
    var numberOfSides = 0
    func simpleDescription() -> String {
        return "A shape with \(numberOfSides) sides."
    }
}
```

创建Shape类的实例，并调用其字段和方法。

```
var shape = Shape()
shape.numberOfSides = 7
var shapeDescription = shape.simpleDescription()
```

通过init构建对象，既可以使用self显式引用成员字段（name），也可以隐式引用（numberOfSides）。

```
class NamedShape {
    var numberOfSides: Int = 0
    var name: String

    init(name: String) {
        self.name = name
    }

    func simpleDescription() -> String {
        return "A shape with \(numberOfSides) sides."
    }
}
```

使用deinit进行清理工作。

扩展

扩展用于在已有的类型上增加新的功能（比如新的方法或属性），Swift使用extension声明扩展：

```
extension Int: ExampleProtocol {
    var simpleDescription: String {
        return "The number \(self)"
    }
    mutating func adjust() {
        self += 42
    }
}
7.simpleDescription
```

泛型（generics）

Swift使用<>来声明泛型函数或泛型类型：

```
func repeat<ItemType>(item: ItemType, times: Int) -> ItemType[] {
    var result = ItemType[]()
    for i in 0..times {
        result += item
    }
    return result
}
repeat("knock", 4)
```

1.属性（Property）、可空值（Nullable type）语法和泛型（Generic Type）语法源自C#。
2.格式风格与Go相仿（没有句末的分号，判断条件不需要括号）。
3.Python风格的当前实例引用语法（使用self）和列表字典声明语法。
4.Haskell风格的区间声明语法（比如1..3，1...3）。
5.协议和扩展源自Objective-C（自家产品随便用）。
6.枚举类型很像Java（可以拥有成员或方法）。
7.class和struct的概念和C#极其相似。

注意这里不是说Swift是抄袭——实际上编程语言能玩的花样基本就这些，况且Swift选的都是在我看来相当不错的特性。

而且，这个大杂烩有一个好处——就是任何其它编程语言的开发者都不会觉得Swift很陌生——这一点很重要。

Quick Terminology: Storyboard

- Storyboards help you graphically lay out your app before you code it.
- It makes it easy to see the “flow” of your app
- You are advised to use Storyboards going forward with you iOS programming adventures
- If you have tinkered with iOS in the past, you might be asking about the xib/nibs. They are still there, however, Storyboards offer similar functionality and make it easier to visualize your views.

Quick Terminology: MVC

- Model-View-Controller (MVC)
- MVC is the paradigm of iOS programming
- Model: Holds data, should know nothing of the interface
- View: Code for getting data in/out of a view. Deals with items like buttons, lists, tables, etc
- Controller: Keeps the Model objects and View objects in sync

Apple developer program

- Apple developer account is free
- Apple developer program is not free- 99$/year
- Registration done from https://developer.apple.com/programs/enroll
- iOS debugging

CocoaPods

- $ sudo gem install cocoapods --$ sudo gem update —system
- $ pod search AFNetworking
- $ vim Podfile
- $ pod install
- open .xcworkspace file with Xcode
- Swift package manager

Autolayout

Hello World– demo project

To be continued…