

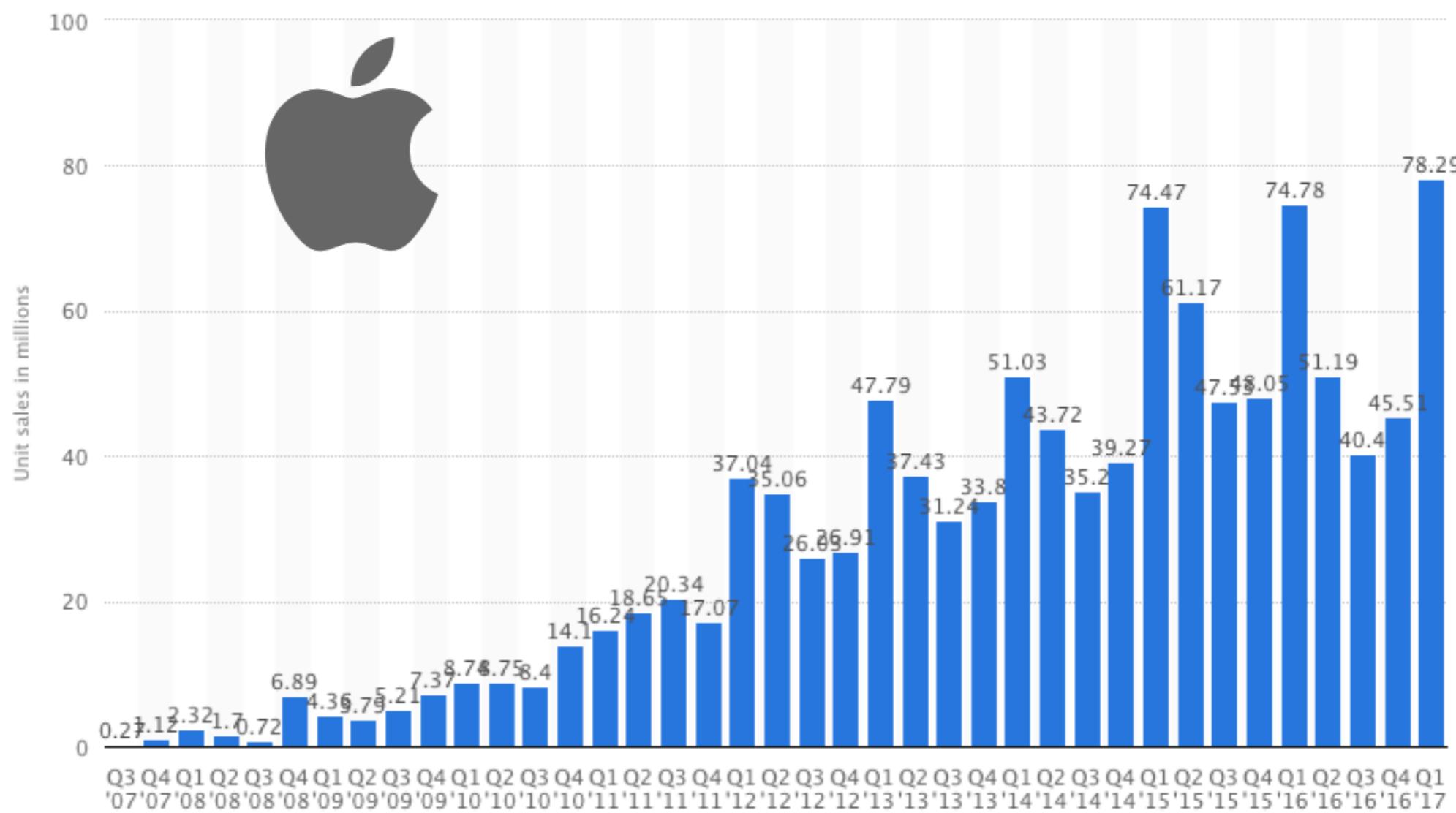
Mobile Development (ST2MOB)

iOS



Nicolas Sicard - 2017

There are now more than 1 billion active Apple devices in use around the world (January 26th, 2016)





Mobile platform

Multiple devices:

iPhone, iPad, iPod, AppleTV, AppleWatch



Family of OSes:
iOS, tvOS, watchOS

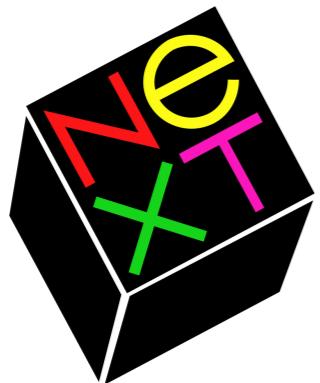
watchOS

tvOS





Evolution



90'

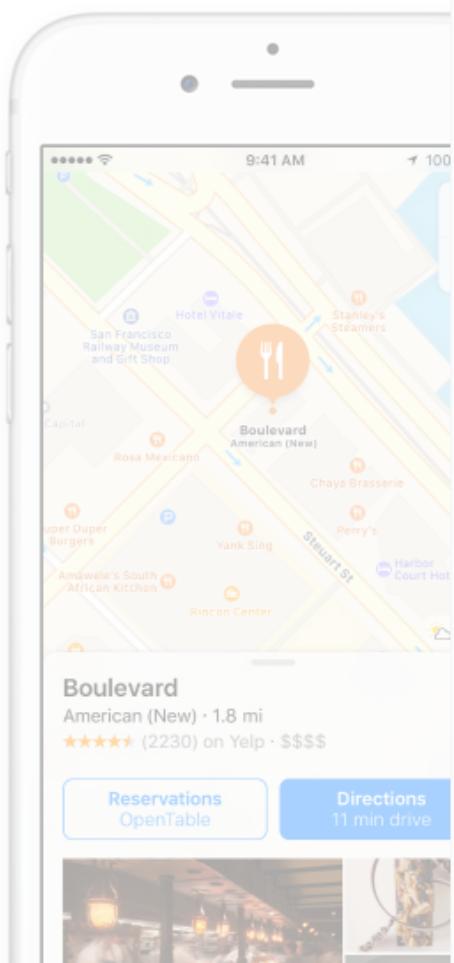
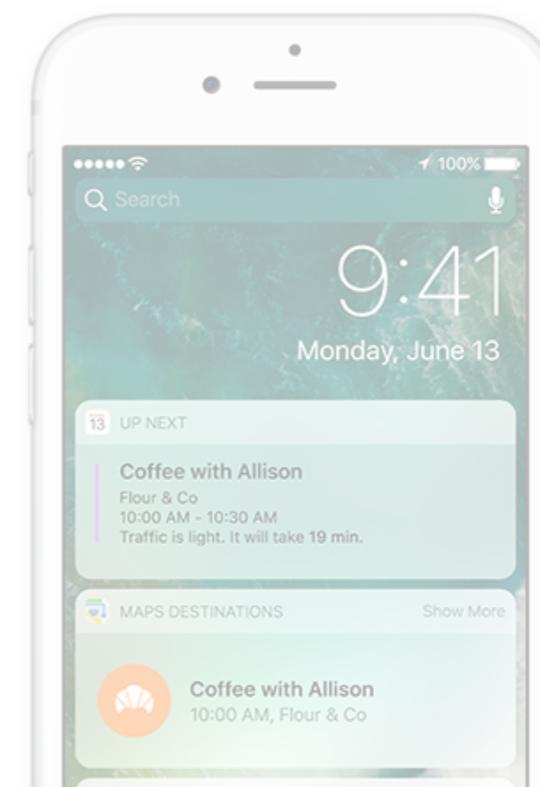
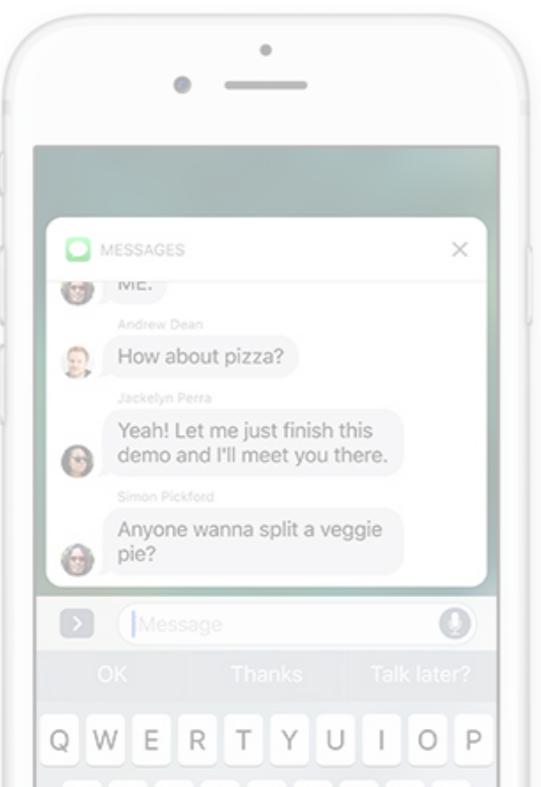
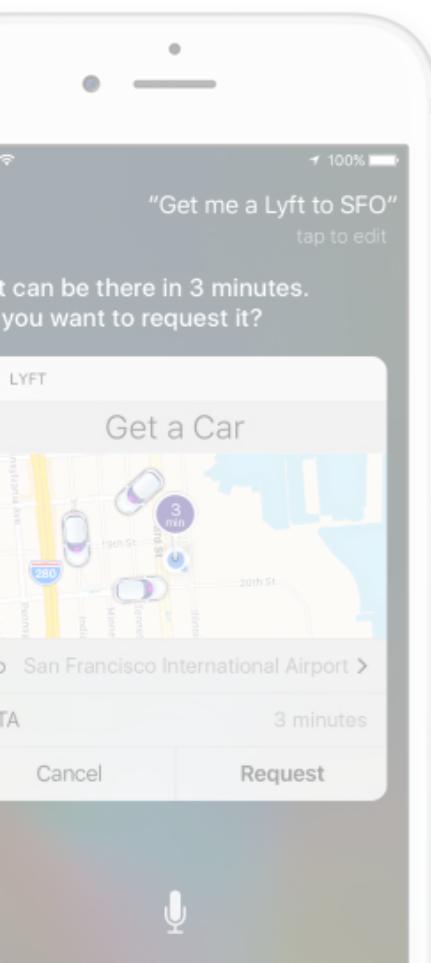


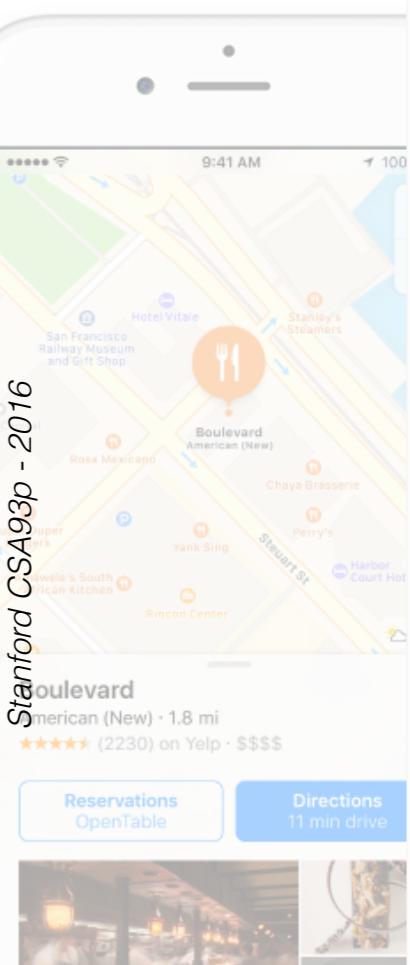
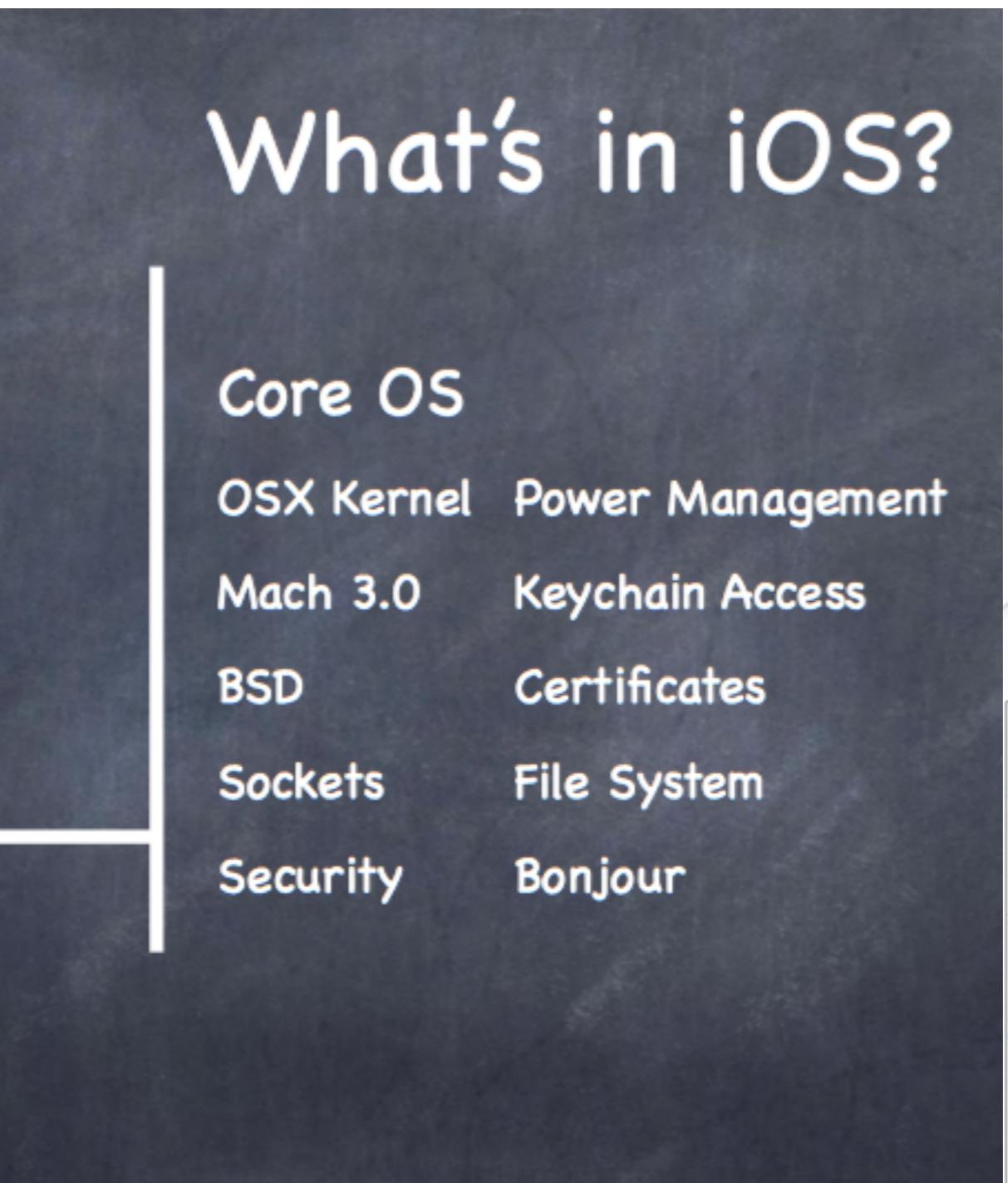
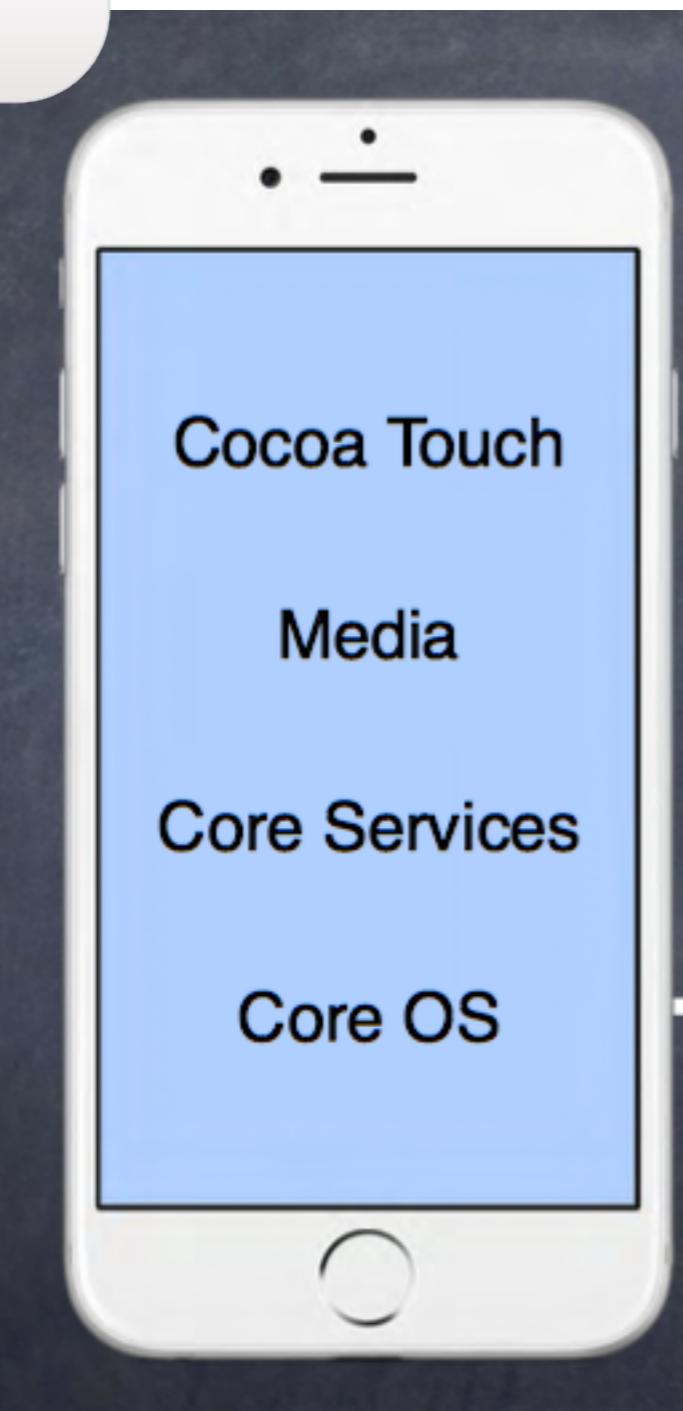
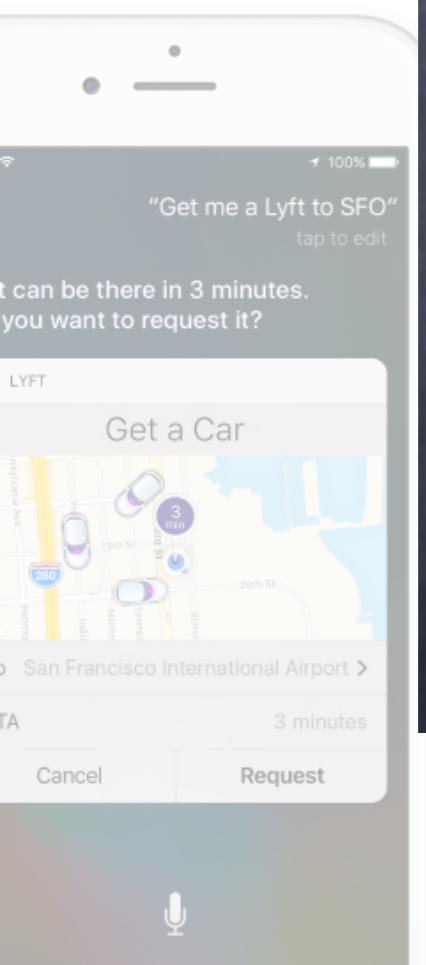


January, 2007
version 1.0

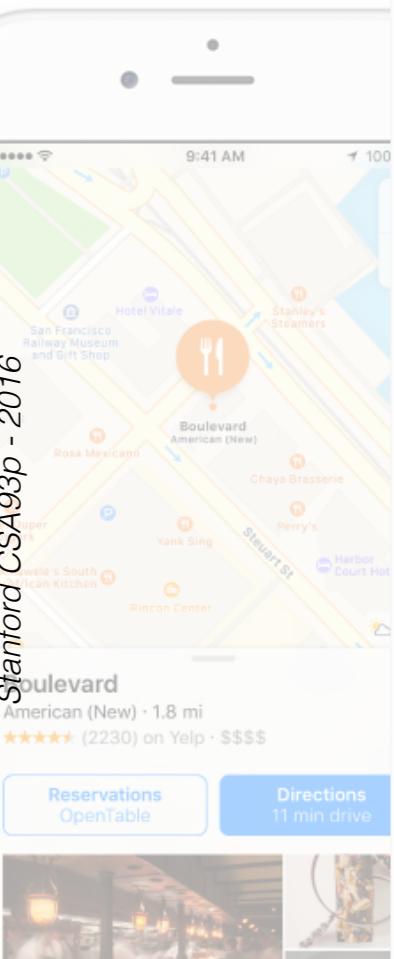
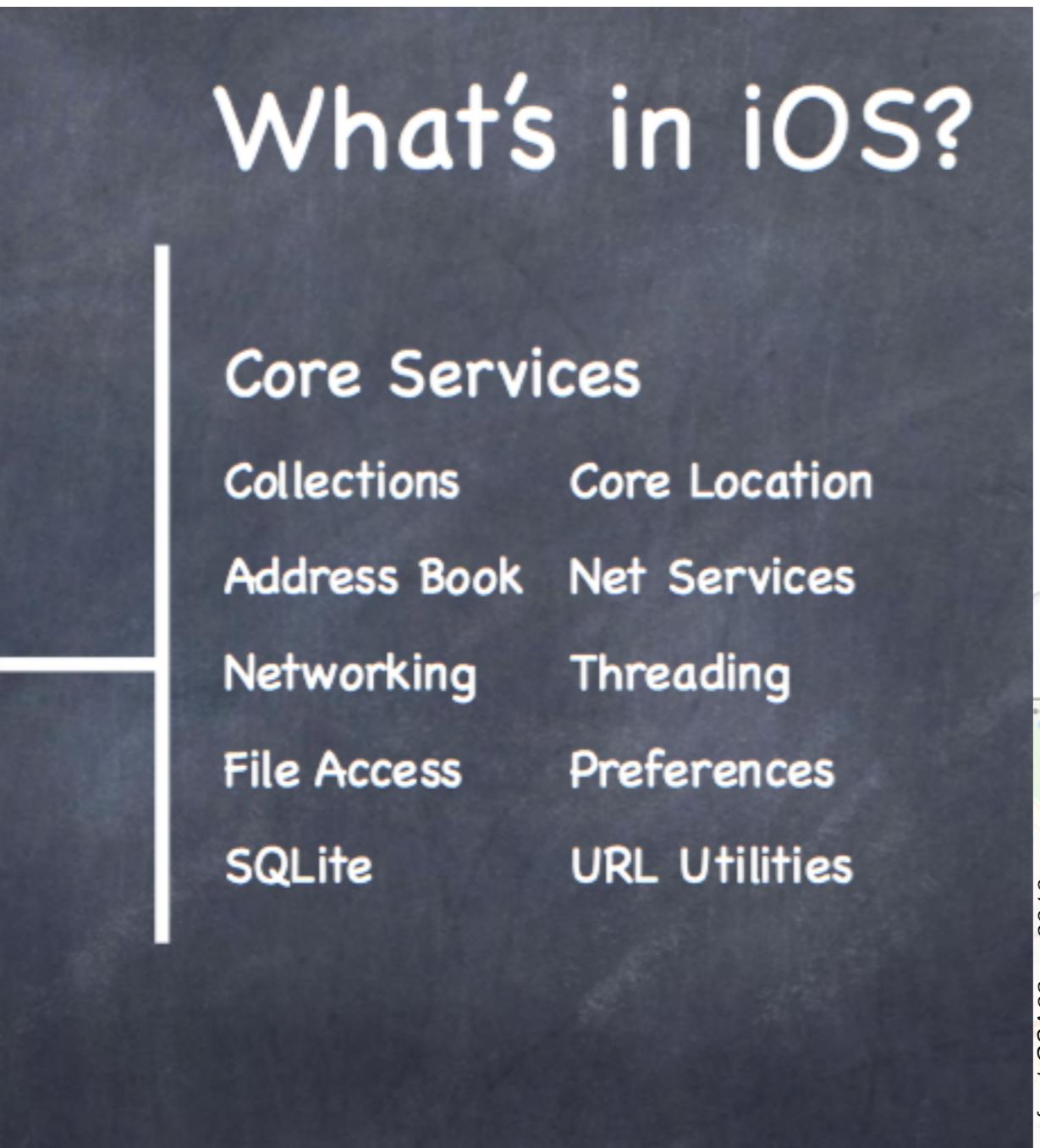
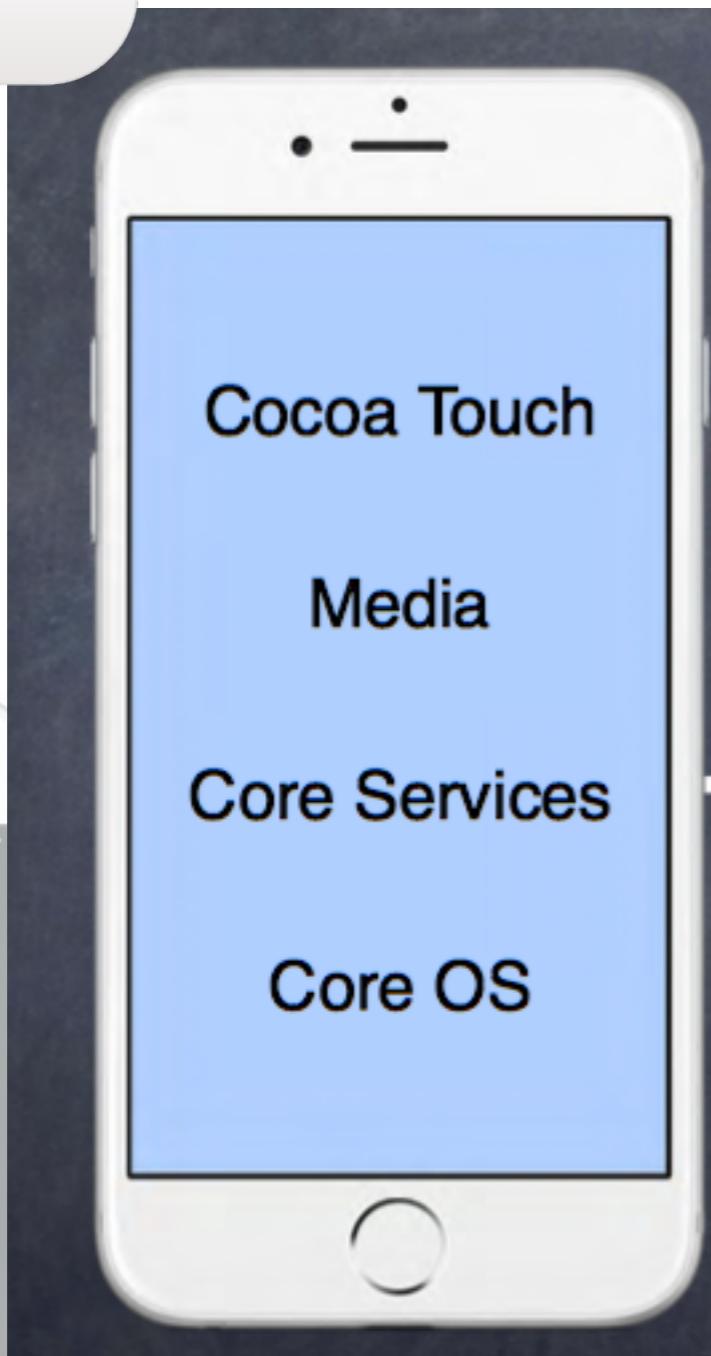


October, 2016
version 10.0

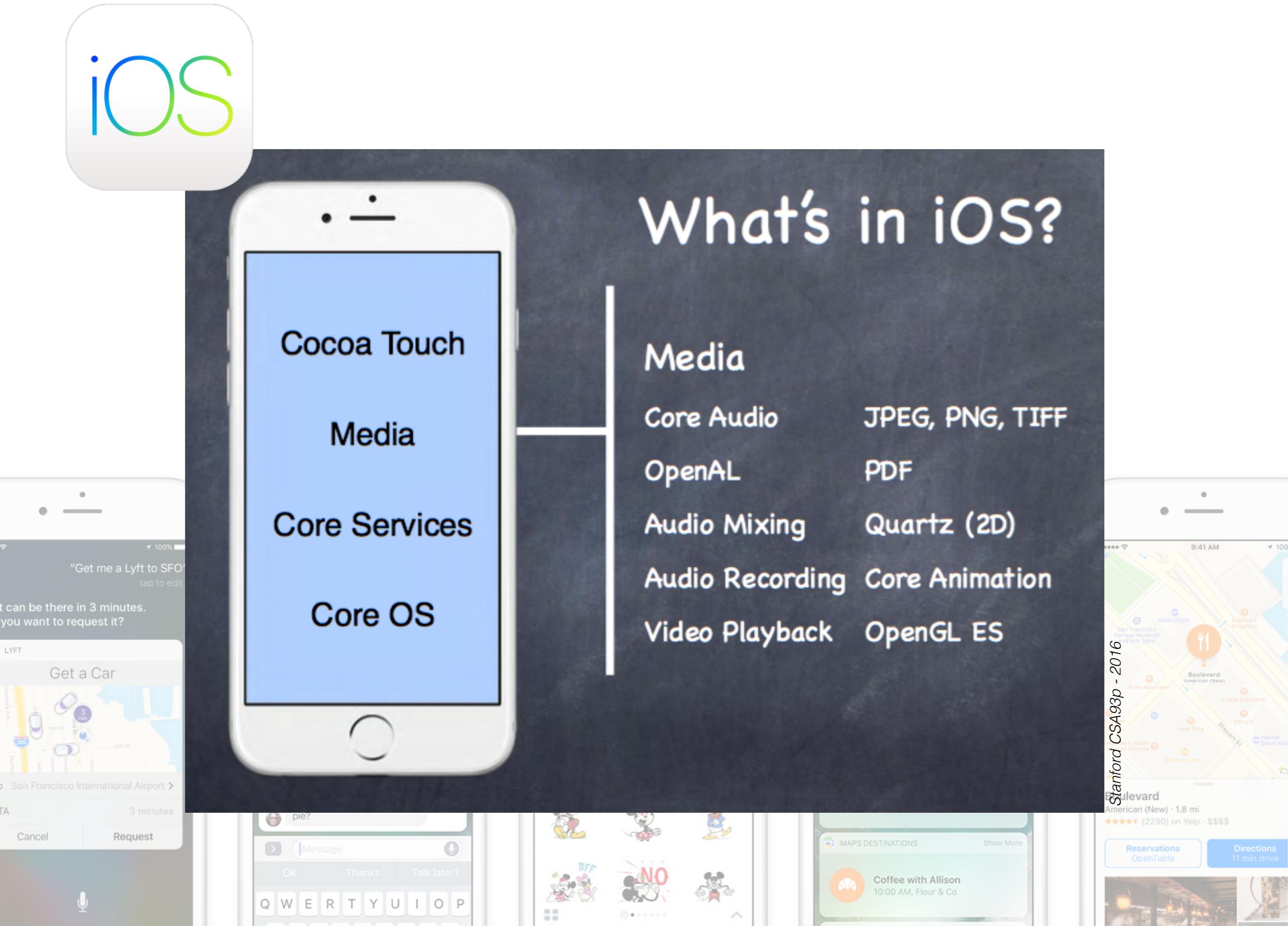




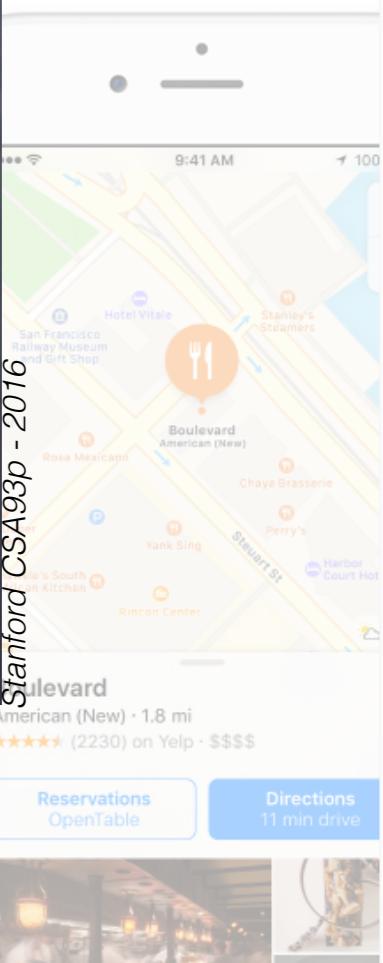
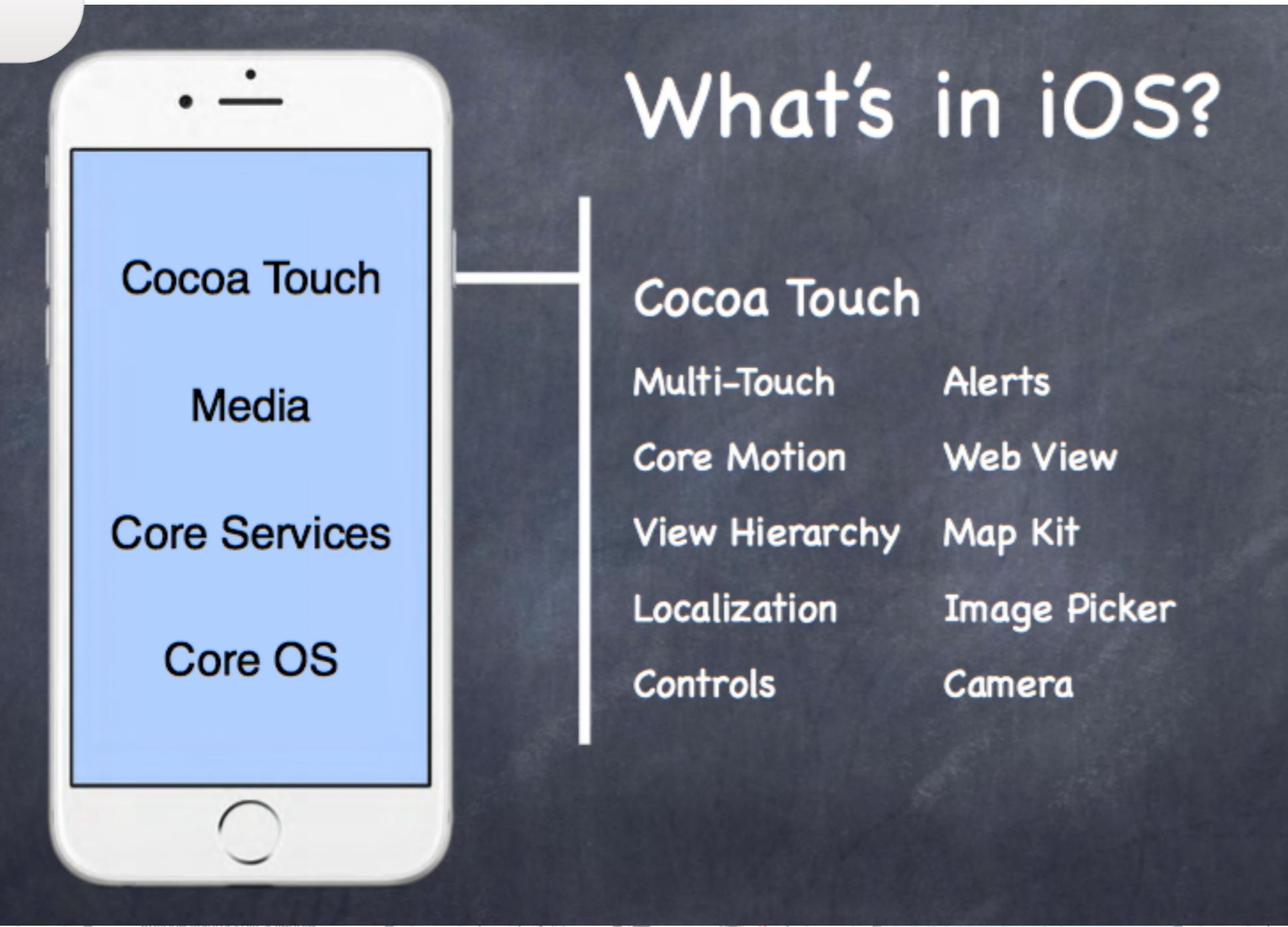
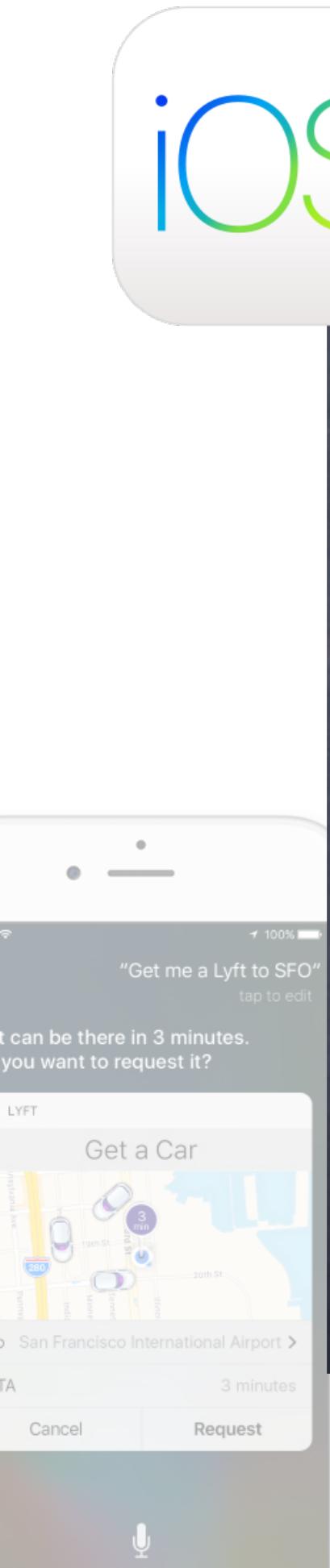
iOS



iOS



iOS



Stanford CSA93p - 2016

ios

Platform Components

- Tools



Xcode 7/8



Instruments

- Language(s)

```
let value = formatter.numberFromString(display.text!)?.doubleValue
```



Foundation

Core Data



UIKit

Core Motion
Map Kit

- Frameworks

MVC

- Design Strategy



CS193p
Spring 2016



Swift

Safe, fast, expressive language, designed to replace Objective-C and more...

Development started in 2010 (Chris Lattner), already 14th in the TIOBE index today

Version 1.0 in September, 2014, open-source in late 2015 (Linux port), v3.0 in September, 2016

The goal of the Swift project is to create the best available language for uses ranging from systems programming, to mobile and desktop apps, scaling up to cloud services. Most importantly, Swift is designed to make writing and maintaining correct programs easier for the developer.



Swift

Many changes between v1.0 and v3.0 (v3.0 is a stable version)

<https://developer.apple.com/swift/>

<https://swift.org>

*The Swift Programming Language book : [https://
itunes.apple.com/fr/book/swift-programming-
language/id881256329?mt=11](https://itunes.apple.com/fr/book/swift-programming-language/id881256329?mt=11)*



The Swift
Programming
Language

Swift 3.0.1 Edition



Swift Playground



The image shows a Swift Playground interface. On the left, the code editor displays the contents of `MyPlayground.playground`. The code defines a `GameScene` class that initializes a green sprout-like sprite at position (250, 250) and performs a sinusoidal oscillation on its `update` method. The right side of the interface shows the resulting output: a green sprout character on a black background, and a timeline graph illustrating the oscillating value of `i`.

```
// Playground - noun: a place where people can play
// Think as below as your Main class, basically the Stage

// this imports higher level APIs like Starling
import SpriteKit
import XCPlayground

// our main logic inside this class
class GameScene: SKScene {

    // properties initialization
    // note that the spriteNode property below is not initialized
    // we initialize it through the init initializer below
    var spriteNode: SKSpriteNode
    var i = 0.0

    // this is our initializer, called once when the scene is created
    // we do our initialization/setup here
    init(size: CGSize){

        // let's grab an image, like [Embed] in AS3, results in image data like
        // BitmapData
        // let is to declare a constant, var a variable
        // note that we don't type things, you actually can to resolve
        // ambiguity sometimes
        // but it is inferred by default and does not cause performance issues
        // to not statically type
        let sprite = NSImage(contentsOfFile:"/Users/timbert/Documents/
            Adium.png")

        // let's create a bitmap, like Bitmap in AS3
        let myTexture = SKTexture(image: sprite)

        // let's wrap it inside a node
        spriteNode = SKSpriteNode(texture: myTexture)

        // we position it, we could scale it, etc.
        spriteNode.position = CGPoint (x: 250, y: 250)

        // we complete the initialization by initializing the superclass
        super.init(size: size)
    }

    // this gets triggered automatically when the scene is presented by the view
    // similar to Event.ADDED_TO_STAGE
    override func didMoveToView(view: SKView) {

        // let's add it to the display list
        self.addChild(spriteNode)
    }

    // we override update, which is like an Event.ENTER_FRAME or advanceTime in
    // Starling
    override func update(currentTime: CFTimeInterval) {
        i += 0.1
        // oscillation with sin, like Math.sin
    }
}
```

MyPlayground.playgroundEdited

Timeline MyPlayground.playground (Timeline)

GameScene spriteNode SKSpriteNode i 0.0

40 SEC +

bytearray.org



Swift@IBM

Server side swift :

<https://developer.ibm.com/swift/>

<https://developer.ibm.com/swift/blogs/>

Bluemix (swift online sandbox)

<https://swiftlang.ng.bluemix.net/#/repl>

<https://developer.ibm.com/swift/2015/12/03/introducing-the-ibm-swift-sandbox/>



Swift@IBM

<https://swiftlang.ng.bluemix.net/#/repl>

IBM Swift Sandbox BETA

About What's New? Feedback f t Sign Up Sign In

```
1 // Write some awesome Swift code, or import libraries like
2 "Foundation",
3 // "Dispatch", or "Glibc"
4
5 print("Hello world!")
6 let a = 5
7 class Toto {
8     let value = 7
9     func display(message: String) {
10         for i in 1...value {
11             print("[\((i))] \(message)")
12     }
13 }
14
15 let myToto = Toto()
16 myToto.display(message: "Hello")
17
18
```

Swift Ver. 3.0.2 (Release)
Platform: Linux (x86_64)

Hello world!
[1] Hello
[2] Hello
[3] Hello
[4] Hello
[5] Hello
[6] Hello
[7] Hello

New Load Code Save Code ➤ Settings

References

<https://developer.apple.com>

<http://www.weheartswift.com/>

<https://learnxinyminutes.com/docs/swift/>

<https://www.raywenderlich.com/>

<http://stackoverflow.com/>

<http://appventure.me/>

<http://swiftyyeti.com/generics/>

<http://codewithchris.com/learn-swift-from-objective-c/>

<http://dean.cafelembas.com/>

<https://github.com/raywenderlich/swift-style-guide>

ST2MOB

(iOS)

1 Introduction session : 3h30 (today)

3 x 3h30 Lab sessions : 10h30 (during feb. and march)

Course based on the CS193p Spring 2016 Stanford online course (available on iTunesU) :

<http://web.stanford.edu/class/cs193p/cgi-bin/drupal/>

<https://itunes.apple.com/us/course/developing-ios-9-apps-swift/id1104579961>

Developing iOS 9 Apps with Swift

[View More from This Institution](#)

by Stanford

To subscribe to an iTunes U course, click View in iTunes.

[View in iTunes](#)

Category: Computer Science

Language: English

Customer Ratings

★★★★★ 122 Ratings

Course Description

Updated for iOS 9 and Swift. Tools and APIs required to build applications for the iPhone and iPad platforms using the iOS SDK. User interface design for mobile devices and unique user interactions using multi-touch technologies. Object-oriented design using model-view-controller paradigm, memory management, Swift programming language. Other topics include: animation, mobile device power management, multi-threading, networking and performance considerations.

Prerequisites: C language and object-oriented programming experience exceeding [Programming Abstractions](#) level, and completion of [Programming Paradigms](#).

Recommended: UNIX, graphics, databases.

[...More](#)

Name	Description	Time	Price	
1 1. Course Overview and Intr...	💻 --	1:17:45	Free	View in iTunes ▶
2 Lecture 1 Slides	📄 --	--	Free	View in iTunes ▶
3 Reading 1: Intro to Swift	📄 --	--	Free	View in iTunes ▶

ST2MOB

(iOS)

- Online materials : videos, slides and assignments
- Instructions will be emailed ahead of each session
- You should watch the video(s) ahead of lab sessions
- MacBooks will be available during lab sessions (bring your own if you have one!)
- Evaluations : final exam (60%) and project (40%)
 - final exam (Android **and** iOS)
 - one project (Android **or** iOS)

Demo

➊ Calculator

All this stuff can be very abstract until you see it in action.

We'll start getting comfortable with Swift and Xcode ~~7~~⁸ by building something right away.

Two part demo starting today, finishing on Wednesday.

➋ Today's topics in the demo ...

Creating a Project in Xcode ~~7~~⁸

Building a UI

The iOS Simulator

`print` (outputting to the console using `\()` notation)

Defining a class in Swift, including how to specify instance variables and methods

Connecting properties (instance variables) from our Swift code to the UI (outlets)

Connecting UI elements to invoke methods in our Swift code (actions)

Accessing iOS documentation from our code

Optionals (?), unwrapping implicitly by declaring with `!`, and unwrapping explicitly with `! and if let`)



Videos (Stanford online course) :

1 - Course overview and Introduction to iOS, Xcode and Swift

2 - Applying MVC