INTRODUCTION TO

IOS DEVELOPMENT

ALICIA M.F. KEY
HTTP://AKEY7.COM
MARCH 5 AND 6, 2014. DENVER & BOULDER, COLORADO

PURPOSE OF TALK

- Aiming at a wide audience.
- Evolution of Objective-C, NeXT, Apple, and iOS.
- How mobile differs from desktop web development.
- Objective-C the language, and how it relates to C. Frameworks provide functionality. Design patterns give it structure. Compilation, linking and runtime make it fly.
- Memory management uses reference counting and no garbage collection.
- Model-View-Controller on iOS
- Building an iOS Hello World app with Xcode

WHAT I WILL COVER

- iOS development with Objective-C and Xcode targeted for iOS devices only.

WHAT I WILL NOT COVER

- Other languages on iOS (Appcelerator Titanium, Ruby Motion, Xamarin)
- Cross-platform development (PhoneGap)
- Web apps for mobile (HTML5/CSS3/JavaScript)
- Alternative IDEs (full command line, AppCode)

ABOUT ME

- Hardware and software development:
    - Digital electronics, analog electronics, C, Java, Ruby (pre-RoR), Objective-C, JavaScript
- Other: physical chemistry and biochemistry, computational science
- Advocate for systems thinking

iOS is a big development world

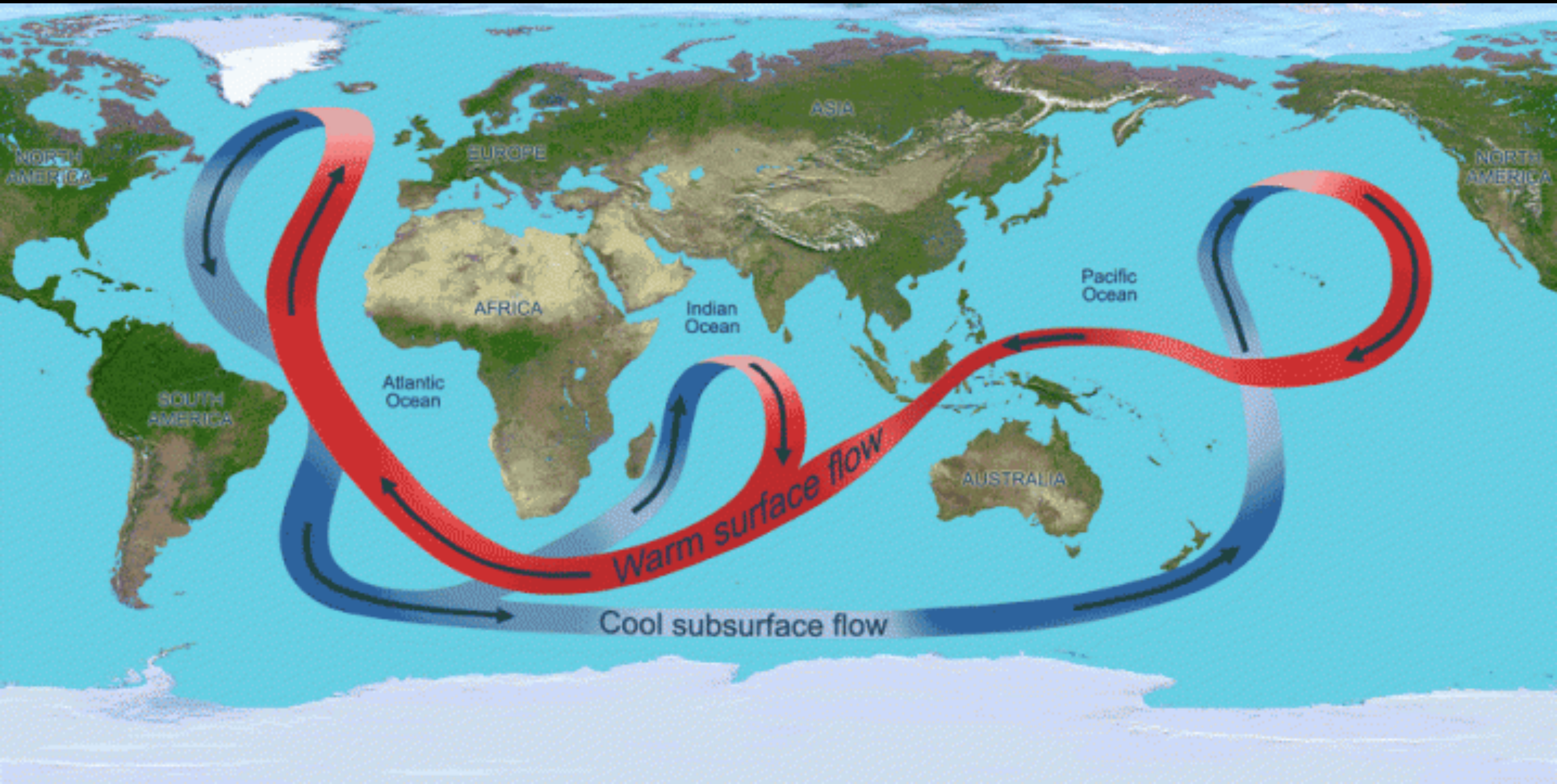
IOS IS DEEP

- Full-blown operating system
- Low-level to very high level programming
- Full suite of application services
  - CocoaTouch, Media, Core Services, Core OS layers.
  - Core Data, file system, SQLite available for data storage.
  - Hardware-specific performance optimization (which is very important)
  - Low-level programming (Accelerate framework, raw C language availability)

Epipelagic
200 m
650 ft
Mesopelagic
1000 m
3300 ft
Bathypelagic
4000 m
13000 ft
Abyssopelagic
Hadopelagic

KEY CONCEPTS WILL...

SHOW YOU THE SYSTEM



ASSUMPTIONS OF AUDIENCE

- Three types:
    - Mobile development who don’t have jobs doing it.
    - Software developers getting into mobile of their own volition
    - Those who have been thrown in off the deep end and told to “just make it work”, not necessarily voluntarily.

HOW MANY OF YOU HAVE…

- Done C development?
- Languages that use non-bytecode compilers?
- Know Ruby the language, disconnected from the Rails web framework?
- Know JavaScript and its design patterns outside of the various JavaScript frameworks?
- Basic computer graphics (RGBA color space, graphics context, geometric primitives?)
- Electronics: voltage, amps, mAh, watts, RF?
- Embedded hardware?

PLATFORM EVOLUTION

IOS / OBJC HISTORY

- Early 1980s: Objective-C invented by combining ***C*** and ***Smalltalk***.
- Mid 1980s: NeXT computer started by Steve Jobs. The NeXTSTEP operating system used Objective-C.
- Late 1990s: Apple acquires NeXT and NeXTSTEP
- Early 2000s: OS X released as an evolution from NeXTSTEP
- Late 2000s: iOS created for iPhone
- 2013: iOS 7 released.

SYSTEMS THINKING

- Crucial for mobile development
- Hardware: Industrial design and electronics
- Software: Low-level, higher level implementation
- User interface, user experience
- How does something fit into the ecosystem of a user's life?

MOBILE VS DESKTOP DIFFERENCES

USER EXPERIENCE

- A mobile application is not simply a shrunken desktop application.
- A user dives in and does quick tasks on an iPhone
- Their attention span with an iPad may be longer.
- Information architecture is different. Splitting content into multiple easy-to-dip-into and read screens is vital.
- How do you use it? What apps do you love and hate?

MOBILE VS DESKTOP DIFFERENCES

HARDWARE DIFFERENCES

- New concepts to people who have traditionally focused 100% on software running on hardware connected to a great WiFi network or Ethernet jack and tied into the AC power grid 24/7.
- In contrast…
- Energy and battery are limited
- Data connectivity is limited and uses valuable battery energy. Don't hit the network unless you have to.
- ***HARDWARE OPTIMIZATION IS IMPORTANT.***

USEFUL C TO KNOW

- Standard printf() string formatting
- **typedefs**
- ***pointers** and **&addresses**
- **structs**
- dynamic memory allocation concepts
- compile and link process

OBJECTIVE-C LANGUAGE

COMPILATION AND RUNTIME

- Steps for an Objective-C program to run:
  - Source files (.h and .m) are compiled
  - The linker runs to connect these source files together with external libraries and frameworks.
  - The iOS Objective-C runtime executes the program.
  - Provisioning profiles, developer certificates, private keys to authorize execution.
- This leads to complexities in Xcode: targets, build phases, build settings, schemes.

MEMORY MANAGEMENT 1

- As your program runs, objects go into and out use. Clearing out unused objects (deallocating) is critical so you don't run out of finite RAM.

- In the strictest sense, your program is responsible for clearing unused memory. Recall malloc() and free() in C.

- Many languages use garbage collection to do this for you. Ruby and Java are examples.

- iOS Objective-C does not use garbage collection for performance reasons.

OBJECTIVE-C LANGUAGE

MEMORY MANAGEMENT 2

- The iOS Objective-C runtime uses reference counting to handle deallocation.

- When an object is instantiated, or another object asserts ownerships (retains), the reference count is incremented by 1. When an object releases ownership, the reference count is decremented by 1

- When the retain count is zero, the object is deallocated.

- This used to be entirely manual.

MEMORY MANAGEMENT 3

- Automatic Reference Counting (ARC)
- ARC allows you to define an object graph with ownership relations among objects.
- Once the graph is defined, the compiler creates code to manage the graph for you.
- This means you are less likely to accidentally make a memory leak by not decrementing a retain count. But retain cycles are still possible if you don’t set your graph up correctly.
- Whiteboard time.

OBJECTIVE-C LANGUAGE

MEMORY MANAGEMENT 4

- Other memory management systems outside of ARC.
  - @autorelase pool
  - Core Foundation, CFRelease() and CFRetain()
  - C memory management—local non-object variables, malloc(), free(), byte-aligned memory allocation, etc
- All are more or less independent.

SHOW AND TELL

HELLO MARS

- Storyboards and segues
- NS* classes
  - All *classes* descend ultimately from NSObject
  - Object-oriented utility functions
- UI* classes (User Interface)
- CG* functions and data structures (CoreGraphics drawing)

OBJECTIVE-C LANGUAGE

DESIGN PATTERNS

- iOS and OS X applications are built around well-known design patterns. With some give or take, major ones are:
  - *Strategy*: Delegates and data sources
  - *Command*: target-action, NSInvocation
  - *Observer*: NSNotificationCenter, key-value-observing
- Loose coupling.
- Blocks are important (similar to Ruby blocks

LEARNING RESOURCES

MY 3 FAVORITE BOOKS

- Neuberg: iOS 7 Programming Fundamentals
- Neuberg: Programming iOS 7
- Napier, Kumar: iOS 6 Programming: Pushing the Limits (iOS 7 version available, but very different)
- Not instant gratification—but it shows you the *why* and not just *how*, which IMO is more valuable.

MY #1 LEARNING RESOURCES

APPLE REFERENCES

- I love the Apple resources—they come straight from the source.
    - Google "$CLASSNAME reference" to bypass tutorial blogs.
- WWDC videos: 2011, 2012, 2013.
- Apple iOS Technology Overview: About the iOS Technologies