

Introductions



CS 442: Mobile App Development
Michael Saelee <lee@iit.edu>

Michael (Sae) Lee

- lee@iit.edu
- moss.cs.iit.edu
- Office: SB 226A



Agenda

- Syllabus & Administrivia
- Course overview



Android section!

- All regular semester CS 442 sections
 - Taught by industry instructor
- Ideal if you don't have a Mac (and aren't interested in iOS development)
- Same high level content



§ Syllabus



Prerequisites

- “substantial” programming experience
 - previously, C was advantageous; no more!
- data structures (CS 331)
- systems programming (CS 351)
- databases (CS 425)



Prerequisites

- you need a Mac! (or regular access to one)
- essentials:
 - yes, there is a right click
 - command (⌘) for control
 - single menu bar



Online resources

1. Course website

moss.cs.iit.edu/cs442

- static information

- syllabus, lecture calendar, assignments, slides, links to reading material

CS 442: Mobile App Development

Announcements

- Welcome to the Summer 2016 edition of CS 442: Mobile App Development

Calendar

May 23

- Slides: [Syllabus & Overview](#)

May 25

- Swift language intro
- Reading: [The Swift Programming Language](#)

June 1

- Anatomy of an App
- Reading: [App Programming Guide for iOS](#)

June 6

- iOS API overview
- Reading: [Start Developing iOS Apps](#)

Machine Problems

Machine problems will be listed below as they are assigned and made available:

- TBA

Project deliverables:

1. Informal proposal (5%) due: June 1
2. Elevator pitch (5%) due: June 8
3. Requirements document (10%) due: June 13
4. Paper prototype (10%) due: June 22
5. Early prototype (20%) due: June 22
6. Final prototype (40%) due: July 2
7. Final presentation (10%) due: July 2

Resources

Administrivia

- Instructor: Michael Lee
 - E-mail: lee@iit.edu
 - Office: SB 226A
- [Course syllabus](#)

Development Tools

- [Xcode](#)
- [Instruments](#)

Documentation & API References

- [iOS Developer Library](#)

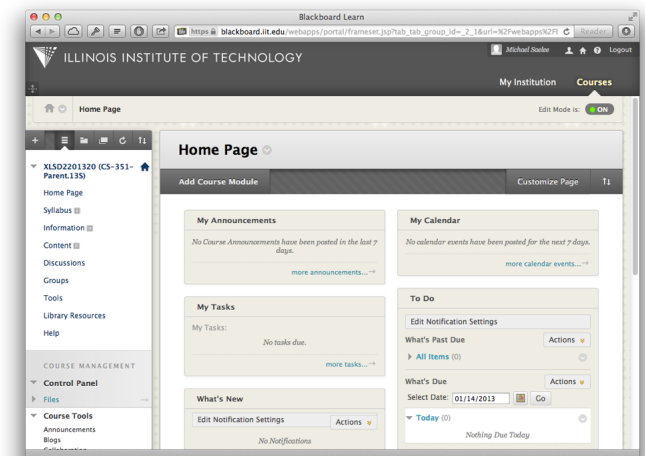
Last updated: Mon May 23 17:42:17 2016



Online resources

2. Blackboard

- *only for grade reporting!*



Textbooks

- None!
 - Plenty of slides, screencasts, sample code
 - developer.apple.com/ios is a great resource



Grading

- Breakdown: 50% assignments, 50% project
- No exams!
- ~6 programming assignments



Assignment grading: checkmark system

✓+ | ✓ | ✓- | 0



← $\sqrt{+}$ $\sqrt{-}$ →

| | | | | | | | | | |
|----------------------------|-----|----|----|----|---------|----|----|----|----|
| Pluses/ Minuses | +4 | +3 | +2 | +1 | ± 0 | -1 | -2 | -3 | -4 |
| Letter Grade | A+ | A | A- | B+ | B | B- | C+ | C | C- |
| Score (%) | 100 | 95 | 92 | 88 | 85 | 82 | 78 | 75 | 72 |

$$0 = 3 \times (\sqrt{-})$$



✓+ =

- no warnings, bugs, or crashes
- good coding style & organization
- “suggested extras” in most assignments



Project = substantial iOS app

- solo or pair work (ideally in pairs!)
- initial proposal deadline: **June 1st**
- deliverables scattered across semester



No:

- iMessage clone
- *XXX* reference (e.g., Matlab reference)
- 100% static / read-only apps



project deliverables:

- elevator pitch
- requirements analysis
- paper prototype / mockup
- intermediate on-device prototype
- final demo & presentation



§ Class & Topics Overview





iOS Development



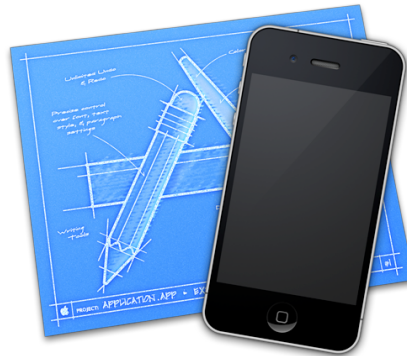
required:

- Intel Mac (Hackintosh?) & OS X 10.10+
- iOS developer account for on-device testing
 - free university program invites coming
 - \$99 for App store deployment

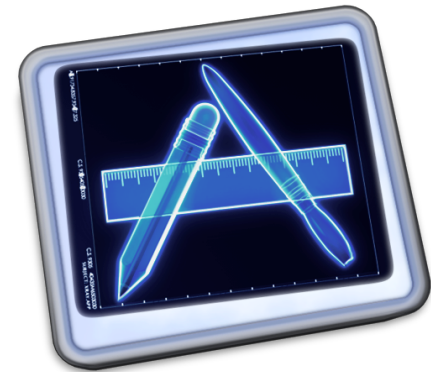




Xcode



iOS Simulator



Instruments

Development Tools

(old) language: Objective-C

- OO, dynamically typed superset of C
 - open source runtime and compiler
- Fairly small language specification



new (preferred) language: Swift!

- multi-paradigm, strongly typed, first-class functions, type inference (& more)
- shares runtime with ObjC platform
 - fully cross-compatible (language/libraries)

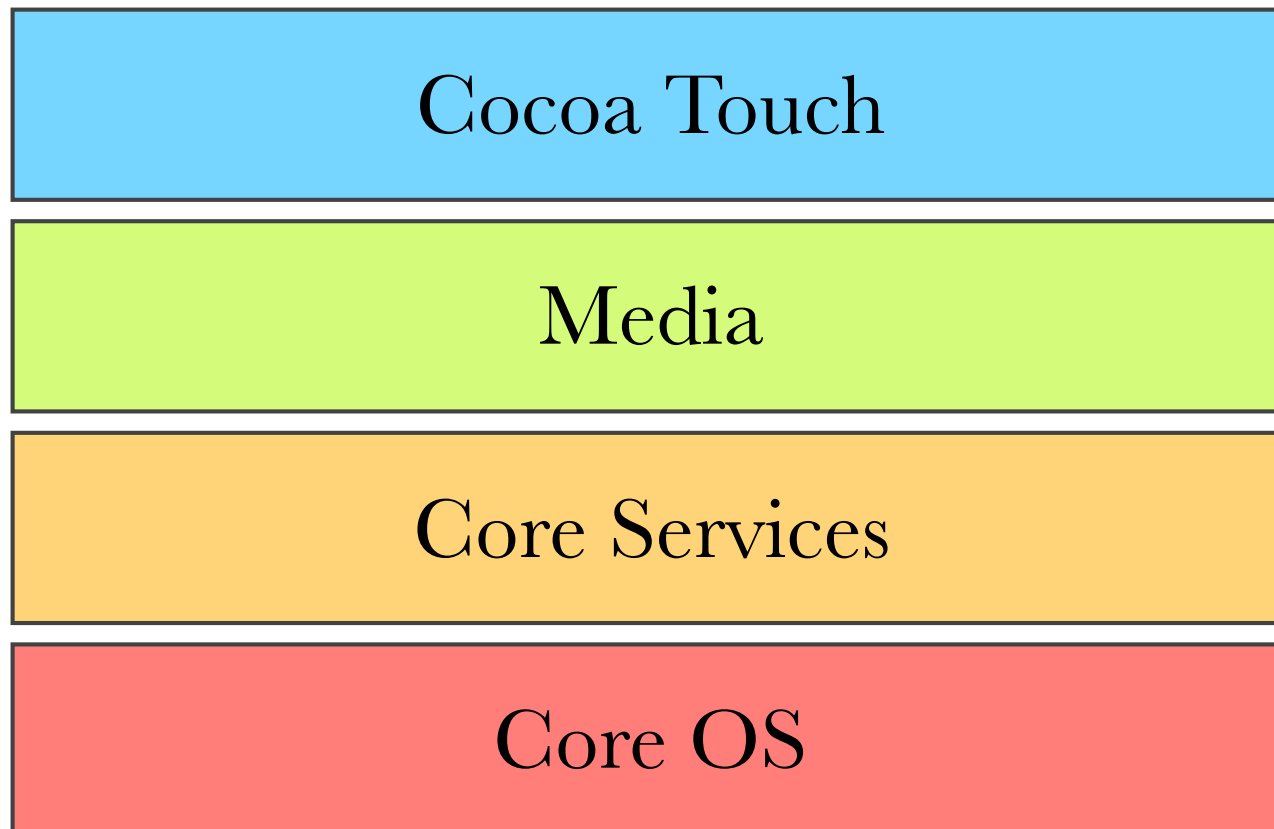


iOS API inherits a lot from the OS X platform

- NeXTSTEP ancestry
- Low level APIs (e.g., data structures, threading, networking)
- “Core” APIs: graphics, animation, etc.



Apple-provided frameworks fall into different layers of the iOS *architectural stack*



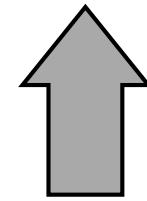


Cocoa Touch

Media

Core Services

Core OS



*object-oriented,
more abstract,
less code needed*



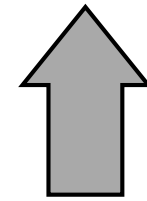


Cocoa Touch

Media

Core Services

Core OS



*less flexible,
less fine-tunable,
more overhead*



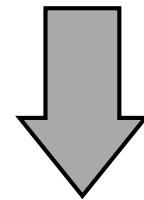
Cocoa Touch

Media

Core Services

Core OS

*may be procedural,
more granular,
exposes hardware*



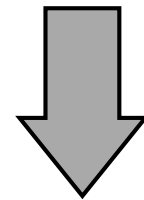
Cocoa Touch

Media

Core Services

Core OS

*complex APIs,
more details ...
... more code!*



Core OS

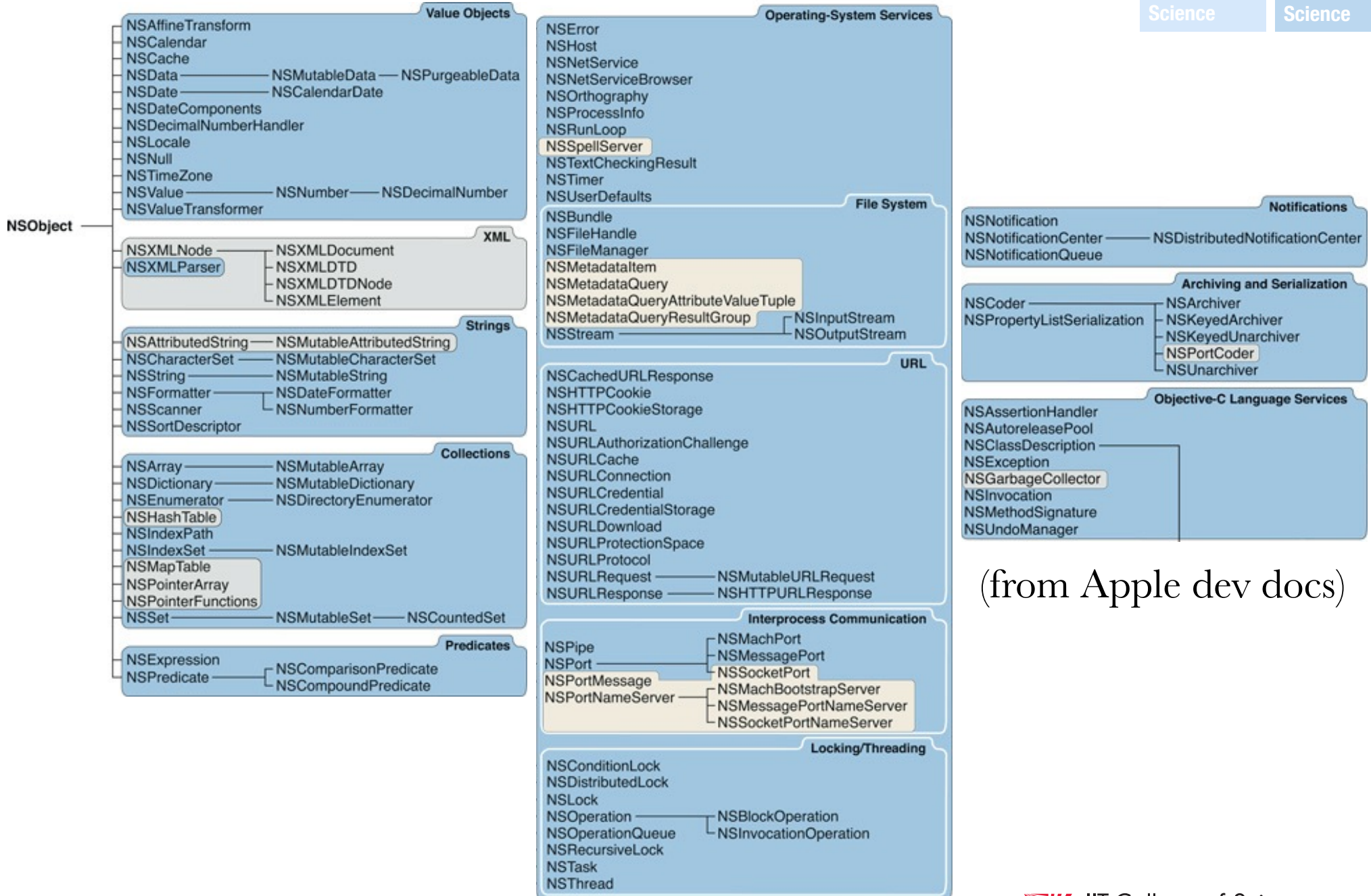
- Unlikely to use directly, but used by other layers of iOS stack
- e.g., Security, Bluetooth and System APIs (POSIX / Unix)



Core Services

- “Core” system services for all iOS apps
- Infrastructure: iCloud, In-App Purchase, Newsstand, Social, etc.
- Hardware: Location, Motion, Telephony
- Data structures/management: Core data, **Foundation** framework





(from Apple dev docs)

Media

- Graphics, Audio, Video APIs
- Core Graphics/Animation/Image/etc.
 - e.g., custom 2D drawing and rendering
- OpenGL ES
 - hardware accelerated 2D/3D graphics



Cocoa Touch

- High level app infrastructure
 - e.g., touch-events, on-screen interface elements, transitions, gestures
- Built-in controllers (e.g., map, photopicker)
- Key framework: **UIKit**



Cocoa Touch

Media

Core Services

Core OS



Typically many ways to accomplish a given task!
(i.e., with frameworks at different levels)



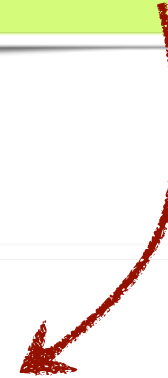
UIKit



```
// clear with white rectangle
[[UIColor whiteColor] set];
UIRectFill(self.bounds);

// load and draw image at (0,0)
[[UIImage imageNamed:@"image.png"] drawAtPoint:CGPointMake(0, 0)];
```

Core Graphics



```
// get current graphics context to draw into
CGContextRef context = UIGraphicsGetCurrentContext();

// clear with white rectangle
CGContextSetRGBFillColor(context, 1.0, 1.0, 1.0, 1.0);
CGContextFillRect(context, self.bounds);

// load image from file
NSString* imageFileName = [[[NSBundle mainBundle] resourcePath
                           stringByAppendingPathComponent:@"image.png"];
CGDataProviderRef provider = CGDataProviderCreateWithFilename([imageFileName UTF8String]);
CGImageRef image = CGImageCreateWithPNGDataProvider(provider,
                                                    NULL,
                                                    true,
                                                    kCGRenderingIntentDefault);

CGDataProviderRelease(provider);

// draw image at (0,0)
CGContextDrawImage(context,
                  CGRectMake(0, 0, CGImageGetWidth(image), CGImageGetHeight(image)),
                  image);
CGImageRelease(image);
```

academic value?

- not just APIs
- focus on design techniques & best practices



broader concerns:

- software design patterns
- testing (functionality, performance)
- prototyping workflow
- version control

