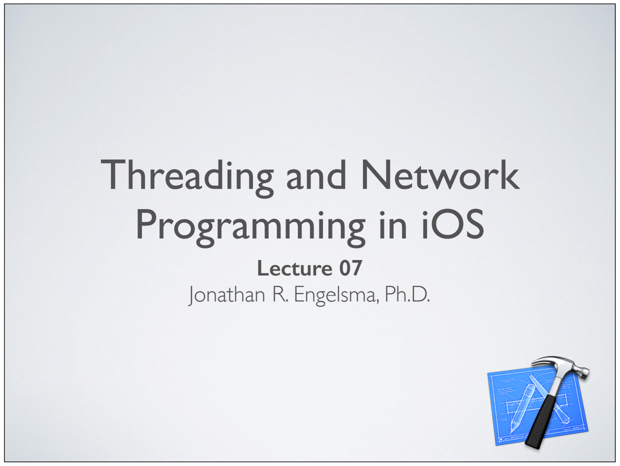
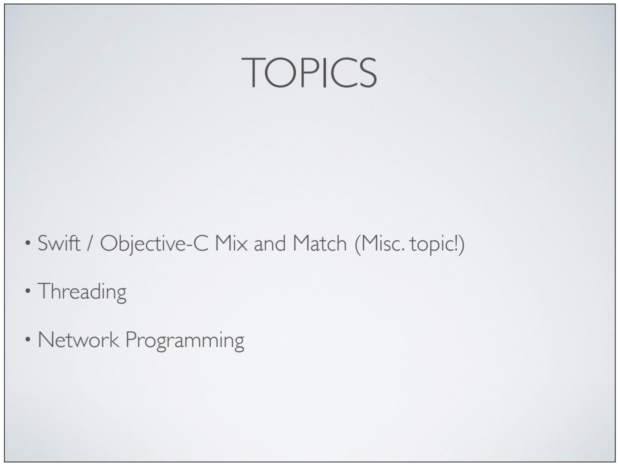


Threading and Network Programming in iOS

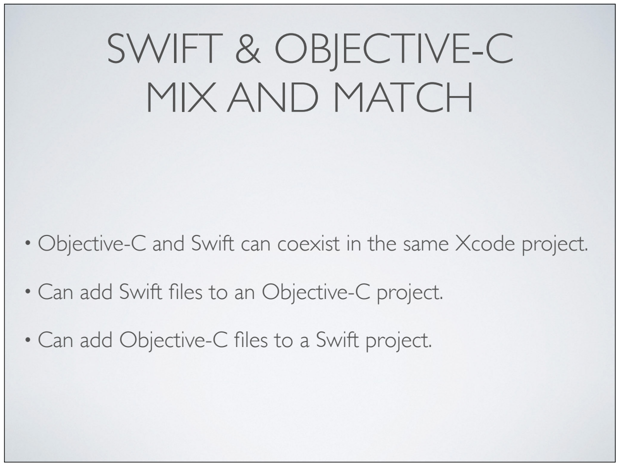
Lecture 07

Jonathan R. Engelsma, Ph.D.



TOPICS

- Swift / Objective-C Mix and Match (Misc. topic!)
- Threading
- Network Programming

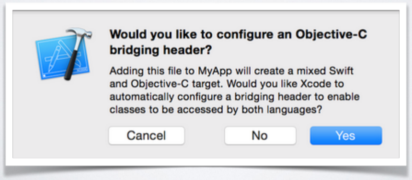


SWIFT & OBJECTIVE-C MIX AND MATCH

- Objective-C and Swift can coexist in the same Xcode project.
- Can add Swift files to an Objective-C project.
- Can add Objective-C files to a Swift project.

OBJECTIVE-C TO SWIFT

- Simply drag the Objective-C files into your Swift project.
- You will be prompted to configured a bridging header. (click Yes)
- Add #imports for every Objective-C header you need.



Would you like to configure an Objective-C bridging header?

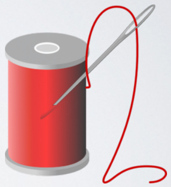
Adding this file to MyApp will create a mixed Swift and Objective-C target. Would you like Xcode to automatically configure a bridging header to enable classes to be accessed by both languages?

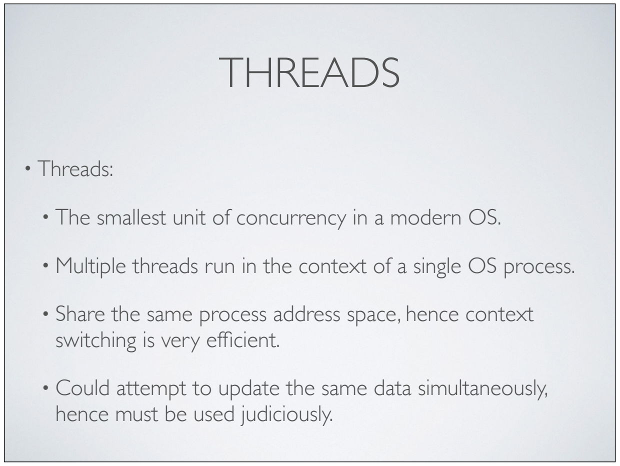
SWIFT TO OBJECTIVE-C

- Simply drag your Swift files into your Objective-C project.
- Xcode generates header files: ModuleName-Swift.h.
- Import these generated headers in your Objective-C code where visibility is needed.

THREADING

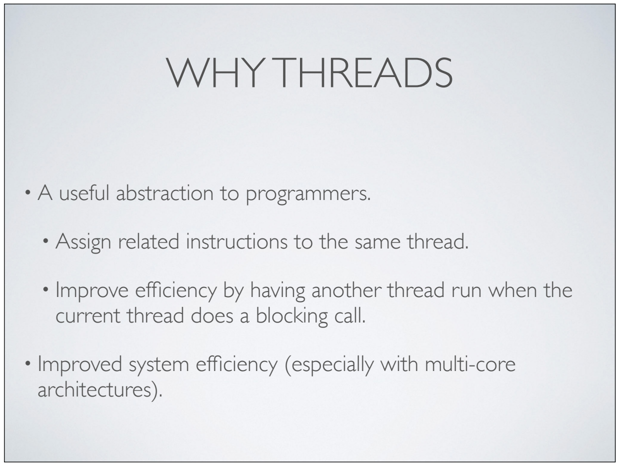
- What is a thread?
  - *"The smallest sequence of programmed instructions that can be managed independently by an operating system scheduler".* wikipedia.com





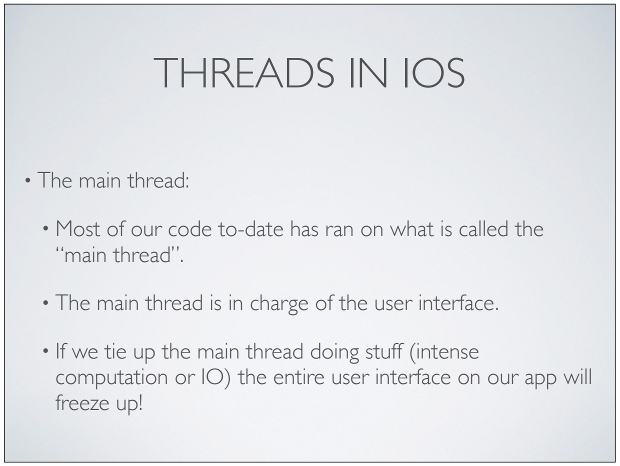
THREADS

- Threads:
  - The smallest unit of concurrency in a modern OS.
  - Multiple threads run in the context of a single OS process.
  - Share the same process address space, hence context switching is very efficient.
  - Could attempt to update the same data simultaneously, hence must be used judiciously.



WHY THREADS

- A useful abstraction to programmers.
  - Assign related instructions to the same thread.
  - Improve efficiency by having another thread run when the current thread does a blocking call.
- Improved system efficiency (especially with multi-core architectures).

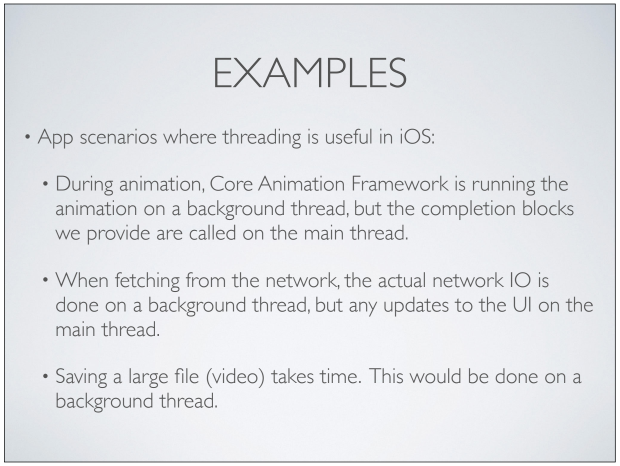


THREADS IN IOS

- The main thread:
  - Most of our code to-date has ran on what is called the "main thread".
  - The main thread is in charge of the user interface.
  - If we tie up the main thread doing stuff (intense computation or IO) the entire user interface on our app will freeze up!

MAIN THREAD

- Main thread runs code that looks roughly like this:
  1. Process the next event that happens on the UI (e.g. somebody pressed a button or scrolls a few, etc.)
  2. A handler method in our code (e.g. IBAction) gets invoked by the main thread to handle the event.
  3. Goto Step 1 above.



EXAMPLES

- App scenarios where threading is useful in iOS:
  - During animation, Core Animation Framework is running the animation on a background thread, but the completion blocks we provide are called on the main thread.
  - When fetching from the network, the actual network IO is done on a background thread, but any updates to the UI on the main thread.
  - Saving a large file (video) takes time. This would be done on a background thread.

THREADING IN IOS

- Most of the iOS frameworks hide threading from us.
- In situations we need to thread, we have several options:
  - NSThreads
  - NSOperations
  - Grand Central Dispatch (GCD)

NSTHREAD

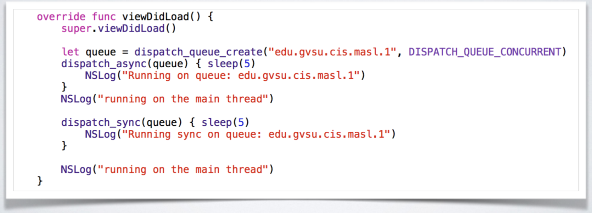
- Gives developed fine-grained control over underlying thread model.
- Will be used very rarely, e.g. only likely time is when you are working with real-time apps.
- In most cases higher level NSOperations or GCD will more than suffice.

NSOPERATION

- NSOperation encapsulates a task, and let's platform worry about the threading.
  - describe an operation.
  - add the operation to a NSOperationQueue
  - arranged to be notified when it completes.

GRAND CENTRAL DISPATCH

- System handles all details of threading in a multi-threaded / multicore situation:



```
override func viewDidLoad() {
    super.viewDidLoad()

    let queue = dispatch_queue_create("edu.gvsu.cis.masl.1", DISPATCH_QUEUE_CONCURRENT)
    dispatch_async(queue) { sleep(5)
        NSLog("Running on queue: edu.gvsu.cis.masl.1")
    }
    NSLog("running on the main thread")

    dispatch_sync(queue) { sleep(5)
        NSLog("Running sync on queue: edu.gvsu.cis.masl.1")
    }

    NSLog("running on the main thread")
}
```

NSOPERATION VS GCD

- NSOperations are implemented on top of GCD
- Adding dependencies among tasks can be tricky on GCD
- Canceling or suspending blocks in GCD requires more work.
- NSOperation adds a bit of overhead but makes it easy to add dependencies among tasks and to cancel/suspend.

NETWORK PROGRAMMING

- Observe that...
  - The mobile phone was inherently a "social" platform
  - First truly "personal" computer
  - Its form factor (small, battery operated) + pervasive network connectivity is what makes it a really interesting computing platform.

NETWORK PROGRAMMING

- Fact: most interesting mobile apps access the network, for example:
  - integration with social media portals
  - access information relevant to the mobile user's current location.
  - multiplayer game might sync with a game server.
  - Flashlight app might display ads pulled from an ad server!

CHALLENGES

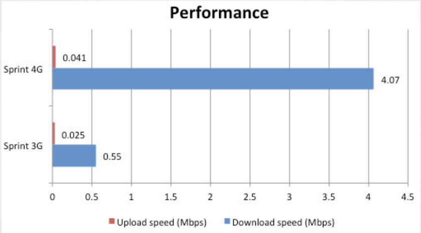
- Accessing the network from a mobile device poses a number of challenges that the app developer must be aware of:
  - Bandwidth/latency limitations
  - Intermittent service
  - Battery drain
  - Security/Privacy

BANDWIDTH/LATENCY LIMITATIONS

- bandwidth: the amount of data that can be moved across a communication channel in a given time period. (aka throughput) usually measured in kilobits or megabits per second.
  - impacts what our mobile apps can or cannot do...
- latency: the amount of time it takes for a packet of data to get from point A to point B.
  - impacts the usability of our mobile apps

BANDWIDTH CHALLENGES

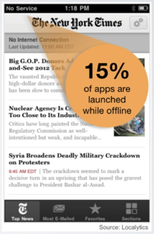
- Lack of...
- Handling the variability...



http://www.computerworld.com/s/article/9201098/3G_vs._4G_Real_world_speed_tests

INTERMITTENT SERVICE

- Key consideration in the native app vs. mobile web app decision
  - native mobile apps can still be used when there is no network connectivity!
  - this happens a LOT more than you might think... 15% of all app launches according to Localytics.



Syria Broadens Deadly Military Crackdown on Protesters

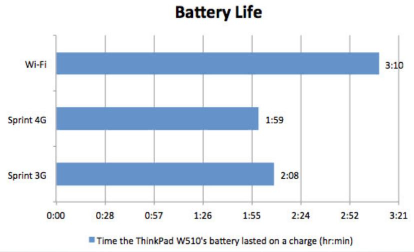
9:45 AM EDT | The crackdown seemed to mark a decisive turn in an uprising that has posed the gravest challenge to President Bashar al-Assad.

Source: Localytics

http://www.localytics.com/blog/2011/15-percent-of-mobile-app-usage-is-offine/

BATTERY DRAIN CHALLENGE

- Powering radio(s) for communication consumes more battery

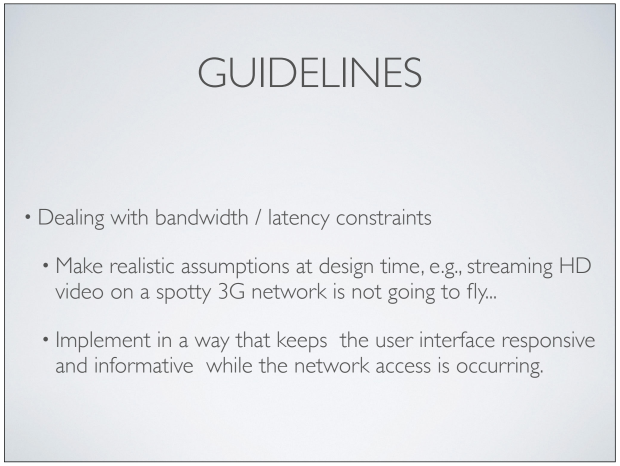


http://www.computerworld.com/s/article/9201098/3G_vs._4G_Real_world_speed_tests

SECURITY / PRIVACY

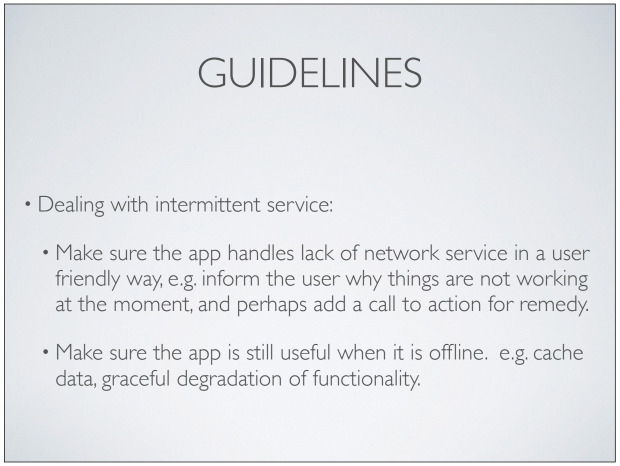
- The perception is that Android has a bigger share of the problems due to the fact Google Play Store is not curated.
- However, iOS has its problems as well:
  - The Apple "LocationGate" debacle.
  - SSL vulnerability
  - Early Random PRNG vulnerability

http://www.scmagazine.com/researcher-finds-easier-way-to-exploit-ios-7-kernel-vulnerabilities/article/338390/



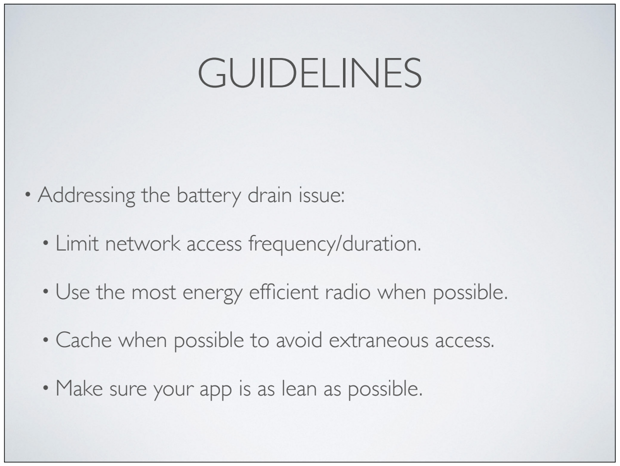
GUIDELINES

- Dealing with bandwidth / latency constraints
  - Make realistic assumptions at design time, e.g., streaming HD video on a spotty 3G network is not going to fly...
  - Implement in a way that keeps the user interface responsive and informative while the network access is occurring.



GUIDELINES

- Dealing with intermittent service:
  - Make sure the app handles lack of network service in a user friendly way, e.g. inform the user why things are not working at the moment, and perhaps add a call to action for remedy.
  - Make sure the app is still useful when it is offline. e.g. cache data, graceful degradation of functionality.



GUIDELINES

- Addressing the battery drain issue:
  - Limit network access frequency/duration.
  - Use the most energy efficient radio when possible.
  - Cache when possible to avoid extraneous access.
  - Make sure your app is as lean as possible.

GUIDELINES

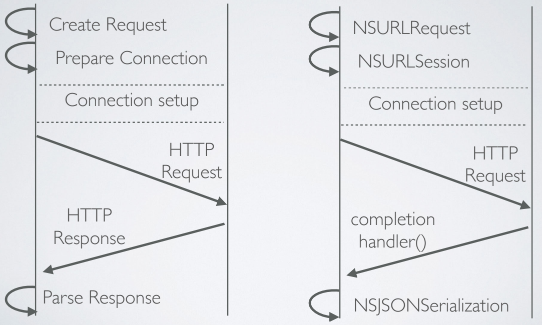
- Avoiding security / privacy issues:
  - Have a written privacy policy available within the app and/or online.
  - Present meaningful user choices.
  - Minimize data collection and limit retention.
  - Education.
  - Practice privacy / security by design.

http://www.futureofprivacy.org/2011/05/19/statement-from-cdt-and-fpf-on-the-development-of-app-privacy-guidelines/

ACCESSING THE NETWORK

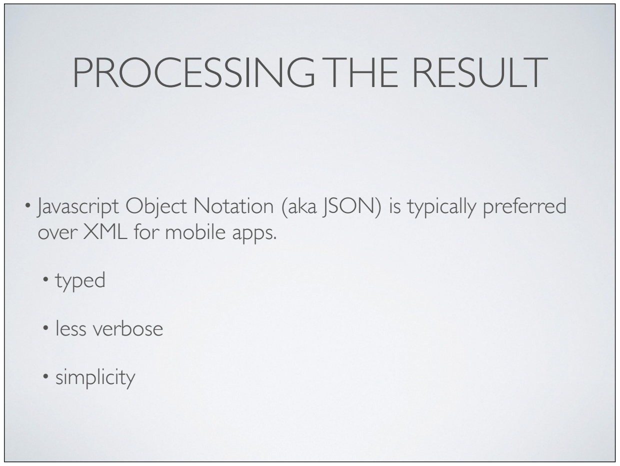
- Most mobile apps will utilize web services to retrieve and store network-based data.
- Hence, HTTP is the protocol that will be used.
- Simple text-based request/response protocol.

HTTP IN IOS



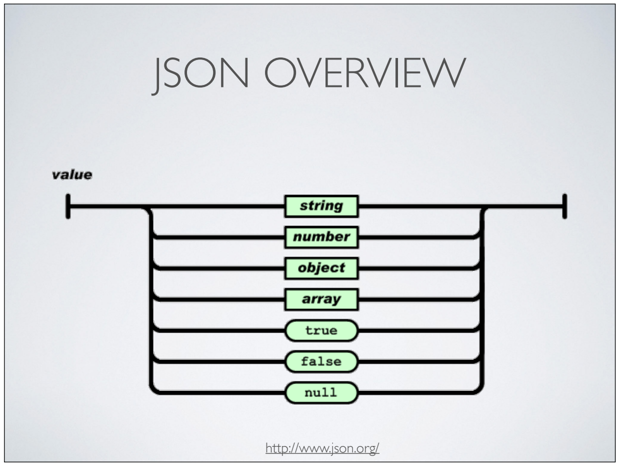
Generic HTTP

iOS HTTP



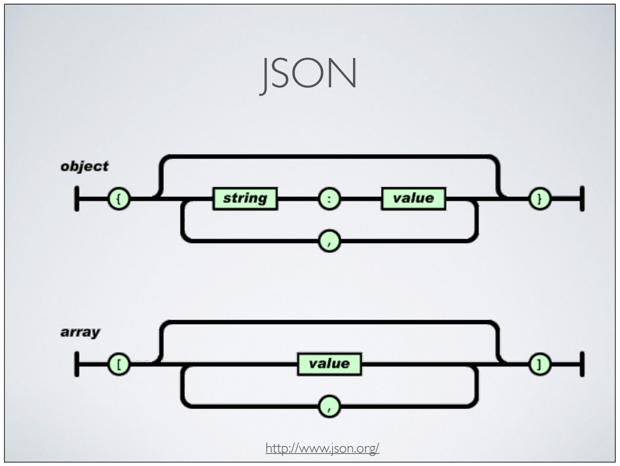
PROCESSING THE RESULT

- Javascript Object Notation (aka JSON) is typically preferred over XML for mobile apps.
  - typed
  - less verbose
  - simplicity



JSON OVERVIEW

http://www.json.org/



JSON

http://www.json.org/

JSON EXAMPLE

```
{
   "toptracks":{
      "track":[
         {
            "name":"Ho Hey",
            "duration":"163",
            "listeners":"646",
            "mbid":"1536cd90-024b-46ff-8f0b-073fabc8e795",
            "url":"http:\/\/www.last.fm\/music\/The+Lumineers\/_\/Ho+Hey",
            "streamable":{
               "#text":"1",
               "fulltrack":"0"
            },
            "artist":{
               "name":"The Lumineers",
               "mbid":"bfcb6630-9b31-4e63-b11f-7b0363be72b5",
               "url":"http:\/\/www.last.fm\/music\/The+Lumineers"
            },
            "image":[
               {
                  "#text":"http:\/\/userserve-ak.last.fm\/serve\/34s\/85356953.png",
                  "size":"small"
               },
               {
                  "#text":"http:\/\/userserve-ak.last.fm\/serve\/126\/85356953.png",
                  "size":"large"
               }
            ],
            "@attr":{
               "rank":"2"
            }
         }
      ]
   }
}
```

READING ASSIGNMENT

- Chapter 24, 25: Programming iOS 8 (by Neuburg)

TOP TRACKS APP DEMO