Announcements

• Lab 3 is due tonight by 11:59 PM

• We will present our final projects pitches on Wednesday July 10th

• Lab 4 is due on Wednesday July 17th

Project Descriptions

• Team Member’s names
  – “Groups” of 1 – 2 students

• 1 Paragraph description of the project

• Project Name

• Due by Tuesday July 10th at 11:59 PM
Final Project Point Distribution and Due Dates

- The final project accounts for 30% of your final grade
  - Final project score is out of 100 points

- Final Project Group Description – 5 points
  - Due on Tuesday July 9th by 11:59 PM

- Project Proposal Presentation – 15 points
  - Submit as PPT, Keynote, or Google slides
  - Due on Wednesday July 10th by 5:30 PM

- Project Update Presentation – 10 points
  - Provide a brief description of what has been accomplished in the email
  - Submit code to demonstrate the accomplishments
  - Due on Wednesday July 24th by 5:30 PM

- Final Project Code - 70 points
  - Due on Wednesday July 31st by 11:40 AM

- Submit all portions of the final project to cse438ta@gmail.com

- Late submissions will result in a 0 for that portion of the final project

---

Final Project Proposal Presentations

- Each group will have up to 10 minutes to present their app

- If someone from your group is not here to present your project you will receive a 0 that portion of the project
Requirements for Project Proposal Presentation

• Motivate the need for the app

• Explain the pieces involved in creating the app

• Show a distribution of work for each team member

• Provide a detailed timeline with tasks and deliverable dates
  – What will be accomplished by the update presentation?
    • Will you demo a skeleton GUI by the second update meeting?
  – What will you show at the final presentation?

• Include a wireframe for your entire application
  – Each “screen” of your app should be included
  – Consider using Keynote, PowerPoint, fluidui.com or ninjamock.com

Wireframes
Today’s Topics

- WKWebKit
- Threading

Web Content in iOS
Displaying Web Content

- Web content can be displayed with WKWebView
  - Introduced in iOS 8, part of WKWebKit Framework
    - Replaces UIWebView
- Content can be
  - local HTML string
  - local raw data + MIME type
  - remote URL
- Leverages WebKit
  - full WK functionality not currently exposed
  - simple API for loading & navigating
  - delegate for some control
  - Same JavaScript engine that powers Safari

WKWebView

- WKWebView subclass, configure in Storyboard or in code
  - Feed it data to display

  func loadHTMLString(_ string: String, baseURL:URL?) -> WKNavigation?

  func load(_ data: Data,
            mimeType: MIMEType, String,
            characterEncodingName: String,
            baseURL: URL?) -> WKNavigation?

- Or give it a URL request
  func load(_ request: URLRequest) -> WKNavigation

  WKNavigation
  - Object that contains information for tracking the loading progress of a webpage

- What’s this URLRequest?
  - Encapsulates a URL to load and caching policy for fetched data
  - Older versions of iOS used an NSURL and NSURLRequest
WKWebView

• Properties and actions you’d expect from a web view
  
isLoading: Bool
  canGoBack: Bool
  canGoForward: Bool
  reload()
  stopLoading()
  goBack()
  goForward()

• A couple others that are handy
  estimatedProgress: Double
  evaluateJavaScript(_:completionHandler:)

WKNavigationDelegate

• Callbacks for load progress
  webView(_: didCommit: ) //called when content starts arriving
  webView(_: didFinish: ) //called when navigation is complete

• Error handling
  webView(_: didFail: withError: )

• Navigation Loading Policy
  //Decides whether to allow or cancel a navigation
  webView(_: decidePolicyFor: decisionHandler: )
Demo
WKWebView

Multithreading in iOS

- Work done with Queues
- Functions (closures) are assigned as units of work to the queues
- Queues execute on a CPU thread
- Queues are either serial or concurrent
- Queues are synchronous or asynchronous
Types of Queues

- **Main Queue**
  - Special serial queue where all UI-Activity happens
  - Non-UI actions should take place on background queue
    - Important to do this to free up main queue

- **Global Queues**
  - Four queues shared by the system with different priority levels

- **Custom Queues**
  - User generated queues with custom attributes (name, priority level, etc)

Multithreading (from CS193P)

- **Executing a function on another queue**
  ```swift
  let someQueue = DispatchQueue(label: "name")
  someQueue.async {
  /* do work here */
  }
  ```

- **The main queue (serial queue)**
  - DispatchQueue.main

- All UI work done on main queue
- All time intensive code or synchronous (blocking) done on another queue
- Swift 3 introduced global queues with different priorities

  ```swift
  DispatchQueue.global(qos: .userInitiated).async {
  // do non-UI stuff that may take time
  DispatchQueue.main.async {
  // Call UI functions with with results from other queue
  }
  }
Multithreading (from CS193P)

• Specifying QOS for queues
  – userInteractive //quick and high priority
  – userInitiated //high priority, may take some time
  – utility //long running
  – background //user not concerned (prefetching)

  let queue1 = DispatchQueue(label: "low priority" qos: "DispatchQueue.background")
  let queue2 = DispatchQueue(label: "high priority" qos: "DispatchQueue.userInteractive")

QoS Demo
Multithreading (CS 193P)

• Multithreaded iOS API
  – Many iOS APIs execute on a queue other than the main queue
  – These APIs typically provide a closure as an argument, which is called upon completion of the method
  – If you want to update the UI, you will need to dispatch back to the main queue

```swift
DispatchQueue.global.async {
    DispatchQueue.main.async {
        // Call UI functions with results from other queue
    }
}
```

Multithreading DEMO
More on Concurrent Programming

• **Grand Central Dispatch (GCD)**

• **GCD Tutorial with Examples**
  - https://www.raywenderlich.com/148513/grand-central-dispatch-tutorial-swift-3-part-1