iOS Dev Accelerator
Week 2 Day 2

- Closures
- NSURLSession

Closures

- Before we can talk about closures, we need to talk about functions
- Functions are first class citizens in Swift (you can have functions be parameters to another function, and you can return a function from another function)
- functions have a type, and the type is defined by the parameters + return type.
- String has a method toInt() that takes a String and returns an int. So that function's type is (String) -> Int.

Closures

- So functions are actually closures that don't do any capturing.
- Nested functions are closures that can capture values from their enclosing function
- Closure expressions are what inline blocks are to Obj-C and can capture values from their surrounding context.

Gosh darn Syntax

- The syntax has many ways to optimize, which is cool but makes it hard to grasp at first
- Closure expression syntax has the following general form: {(parameters) -> return type in statements
- the in keyword is the start of the closure's body
- please refer to fuckingclosuresyntax.com (or its family friendly twin goshdarnclosuresyntax.com)

Refining the syntax

- Implicit Returns from Single-Expression Closures (if the closure only does one expression, you don't need the keyword return)
- Shorthand argument names. You can use $0, $1,$2, to refer to the parameters in the closures body, and drop the argument list entirely. Thanks Dre.

Trailing Closures

- If a closure is the function's final argument, you can write it as a trailing closure instead.
- It is written outside and after the parens of the function.
- Very completion blocky
- Best practice is to use this when you cant fit the closure on a single line.

Capturing Values

- Closures capture variables from the context they are defined in.
- variables that aren't modified are copied, variables that are modified are referenced and kept alive.

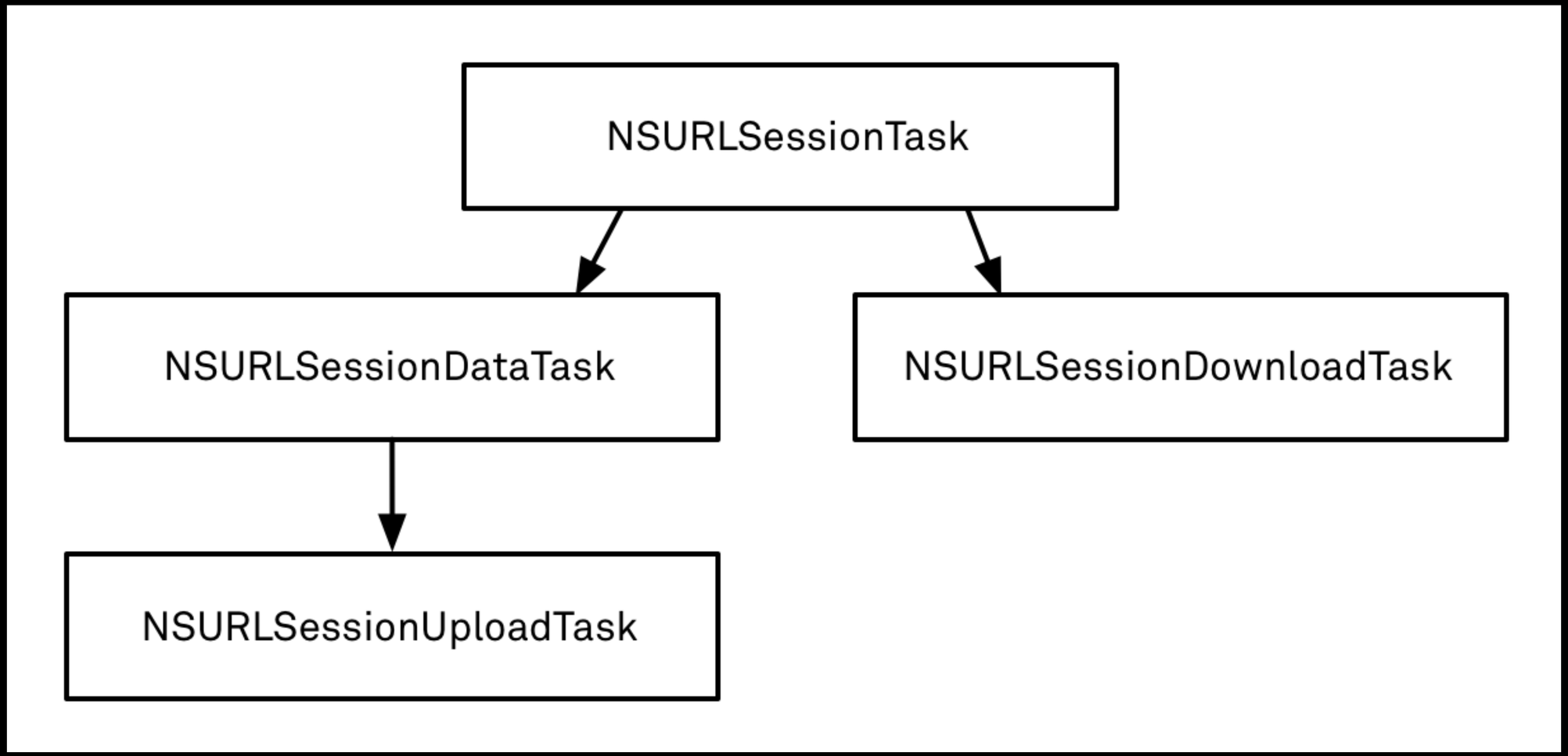
Closures & Callbacks

- Closures help us create callbacks in our application workflow
- “a callback is a piece of executable code that is passed as an argument to other code, which is expected to call back (execute) the argument at some convenient time”
- Since we never know exactly how long a network call is going to take, we will pass in a callback with our methods to our network controller, and these callbacks will be called at a later time once the network operation is complete.
- This is an asynchronous callback.

NSURLSession

- The NSURLSession class and related classes provide an API for making HTTP requests.
- NSURLSession is highly asynchronous.
- NSURLSession has two ways to use it, with completion handlers (closures) or delegation.
- We will focus on the closure way for now.

NSURLSession



NSURLSession Setup

- NSURLSession is initialized with a NSURLSessionConfiguration.
- NSURLSessionConfiguration has 3 types:
    1. Default session - disk based cache
    2. Ephemeral session - no disk based caching, everything in memory only
    3. Background session - similar to default, except a separate process handles all data transfers
- Configurations have many properties for customization. For example you can set the maximum number of connections per host, timeout intervals, cellular access or wifi only, and http header fields.
- We will start with ephemeral and not worry about caching.

NSURLSession Setup

```
  var configuration =
NSURLSessionConfiguration.ephemeralSessionConfigur
ation()

    self.urlSession = NSURLSession(configuration:
configuration)
```

NSURLSession

- All http requests made with NSURLSession are considered ‘Tasks’. Think of them as little minions for your session (or don’t).
- A task must run on a session (NSURLSession class)
- Three types of tasks:
    1. Data tasks - receive and send data using NSData objects in memory
    2. Upload tasks - send files w/ background support
    3. Download tasks - retrieve files w/ background support

NSURLSession Data Task

```
var request = NSMutableURLRequest(URL: NSURL(string: "https://api.github.com/search/
repositories?q=\(string)"))
        request.HTTPMethod = "GET"

        let repoDataTask = self.urlSession.dataTaskWithRequest(request, completionHandler:
{(data, response, error) in

            if error {
                //do something for general error
            }
          else {
          //do switch statement on response code and do something with the data
          }
            })
        repoDataTask.resume()
```

NSURLSession Upload Task

```
let url = NSURL.URLWithString("http://example.com/upload")
let request = NSURLRequest(URL: url)
let dataToUpload = NSData(contentsOfFile: "/path/to/my/file.ext")

let session = NSURLSession.sharedSession()
let uploadTask = session.uploadTaskWithRequest(request, fromData: dataToUpload) {
    (data, response, error) -> Void in
    if error {
        // handle error
    } else {
        // handle response
    }
}
uploadTask.resume()
```

NSURLSession Download Task

```
let url = NSURL(string: "http://example.com/file.zip")
let request = NSURLRequest(URL: url)
let session = NSURLSession.sharedSession()
let downloadTask = session.downloadTaskWithRequest(request) {
    (url, response, error) -> Void in
    if error {
        //handle error
    } else {
        // handle response data
    }
}
downloadTask.resume()
```

iOS Background Modes