THE FUTURE OF ASYNC APIS

@sebreh

Citrix

ASYNC: AT THE CORE OF APP DEVELOPMENT

- UI events
- Application lifecycle
- Networking
- IO

ASYNC ON IOS

- Delegates
- Notifications
- Blocks (oh praise the lord)
- KVO


like some kind of caveman

WHY IS THIS A PROBLEM?

EXAMPLE: NSURLSESSIONTASK

```
[[[NSURLSession sharedSession] dataTaskWithURL:[NSURL URLWithString:@"https://api.podio.com/user"]
                             completionHandler:^(NSData *data,
                                                 NSURLResponse *response,
                                                 NSError *error) {

  NSLog(@"Done.");
}] resume];
```

EXAMPLE: NSURLSESSIONTASK

But what if I want to track progress?

`-countOfBytesReceived` on `NSURLSessionTask` does not officially support KVO.

EXAMPLE: NSURLSESSIONTASK

Discussion

To be notified when this value changes, implement the `URLSession:dataTask:didReceiveData:` delegate method (for data and upload tasks) or the `URLSession:downloadTask:didWriteData:totalBytesWritten:totalBytesExpectedToWrite:` method (for download tasks).

Ok, on the task delegate, right? Nope, on the session delegate!

EXAMPLE: NSURLSESSIONTASK

Ok, so lets implement it on the session delegate.

```
#pragma mark - NSURLSessionDataDelegate

- (void)URLSession:(NSURLSession *)session
          dataTask:(NSURLSessionDataTask *)dataTask
    didReceiveData:(NSData *)data {
  ...
}

#pragma mark - NSURLSessionDownloadDelegate

- (void)URLSession:(NSURLSession *)session
      downloadTask:(NSURLSessionDownloadTask *)downloadTask
      didWriteData:(int64_t)bytesWritten
      totalBytesWritten:(int64_t)totalBytesWritten
      totalBytesExpectedToWrite:(int64_t)totalBytesExpectedToWrite {
  ...
}
```

EXAMPLE: NSURLSESSIONTASK

What now? My completion handler is not called anymore?

```
[[[NSURLSession sharedSession] dataTaskWithURL:[NSURL URLWithString:@"https://api.podio.com/user"]
                             completionHandler:^(NSData *data,
                                                 NSURLResponse *response,
                                                 NSError *error) {
  NSLog(@"I am not called anymore.");
}] resume];
```

EXAMPLE: NSURLSESSIONTASK

```
- (void)URLSession:(NSURLSession *)session
              task:(NSURLSessionTask *)task
              didCompleteWithError:(NSError *)error {
  ...
}

#pragma mark - NSURLSessionDataDelegate

- (void)URLSession:(NSURLSession *)session
          dataTask:(NSURLSessionDataTask *)dataTask
    didReceiveData:(NSData *)data {
  ...
}

- (void)URLSession:(NSURLSession *)session
              task:(NSURLSessionTask *)task
   didSendBodyData:(int64_t)bytesSent
    totalBytesSent:(int64_t)totalBytesSent
    totalBytesExpectedToSend:(int64_t)totalBytesExpectedToSend {
  ...
}

#pragma mark - NSURLSessionDownloadDelegate

- (void)URLSession:(NSURLSession *)session
      downloadTask:(NSURLSessionDownloadTask *)downloadTask
      didWriteData:(int64_t)bytesWritten
      totalBytesWritten:(int64_t)totalBytesWritten
      totalBytesExpectedToWrite:(int64_t)totalBytesExpectedToWrite {
  ...
}

- (void)URLSession:(NSURLSession *)session
      downloadTask:(NSURLSessionDownloadTask *)downloadTask
      didFinishDownloadingToURL:(NSURL *)location {
  ...
}
```

OTHER CHALLENGES

- State keeping as a side effect of imperative code
- How do we deal with sequential or parallel tasks?
- How do we cancel a task?

ASYNC BEHAVIOR IS HARD, BUT CLUNKY APIS MAKE IT A LOT WORSE

GOAL: DEAL WITH ASYNC WORK THROUGH A HIGHER LEVEL OF ABSTRACTION

- Abstract away the technical details
- Avoid manual synchronization

MEANWHILE, IN THE REAL WORLD...

JAVASCRIPT: PROMISES

```
promise.then(function(result) {
  console.log(result); // "Stuff worked!"
}, function(err) {
  console.log(err); // Error: "It broke"
});
```

(source: http://www.html5rocks.com/en/tutorials/es6/promises/)

SCALA (AND JAVA): FUTURES

```
import scala.concurrent._
import ExecutionContext.Implicits.global

val session = socialNetwork.createSessionFor("user", credentials)
val f: Future[List[Friend]] = future {
  session.getFriends()
}
```

(source: http://docs.scala-lang.org/overviews/core/futures.html)

.NET: ASYNC/AWAIT

```
async Task<int> AccessTheWebAsync() {
    HttpClient client = new HttpClient();

    Task<string> getStringTask = client.GetStringAsync("http://msdn.microsoft.com");

    DoIndependentWork();

    string urlContents = await getStringTask;

    return urlContents.Length;
}
```

(source: https://msdn.microsoft.com/en-us/library/hh191443.aspx)

.NET: ASYNC/AWAIT

- Write async code as synchronous code
- Even supports exception handling in async scope

FUTURES: A SIMPLE ABSTRACTION OF AN EVENTUAL VALUE

- A single value at some point in time
- The power of combinators (map, then, when)

Examples: Forbind (see next talk!), PromiseKit

TASKS: A MORE SPECIFIC FUTURE

- Evolution of a future - represent a unit of work
  - Cancellable
  - Progress updates

Examples: .NET Task, PKTAsyncTask (from PodioKit)

REPLACING NSURLSESSIONTASK WITH PKTASYNCTASK

```
PKTAsyncTask *task1 = [[NSURLSession sharedSession]
  pkt_dataTaskWithURL:[NSURL URLWithString:@"https://api.podio.com/user"]];
PKTAsyncTask *task2 = [[NSURLSession sharedSession]
  pkt_dataTaskWithURL:[NSURL URLWithString:@"https://api.podio.com/profile"]];

[task1 onProgress:^(float progress) {
  NSLog(@"Progress update for task 1: %.2f", progress);
}];

[[PKTAsyncTask when:@[task1, task2]] onSuccess:^(NSArray *results) {
  NSLog(@"Both tasks completed.");
}];
```

THE JACKHAMMER: RX AND REACTIVECOCOA

- Like a system of pipes where data flows - model any async behavior
  - Signals a.k.a. "Observables" (the pipes)
  - Combinators (the joints)
  - Declarative

(https://github.com/ReactiveCocoa/ReactiveCocoa)

RX IN OTHER LANGUAGES

RxJava, RxJS, Rx.NET, RxScala, RxClojure, RxCpp,
Rx.rb, RxPY, RxGroovy, RxJRuby

(http://reactivex.io)

LET'S GIVE OUR FRAMEWORKS GREAT ASYNC APIS

FURTHER READING

- Async/await in .NET (https://msdn.microsoft.com/en-us/library/hh191443.aspx)
- http://reactivex.io
- ReactiveCocoa (https://github.com/ReactiveCocoa/ReactiveCocoa)
- ReactiveCocoa 3.0 Changelog (https://github.com/ReactiveCocoa/ReactiveCocoa/blob/swift-development/CHANGELOG.md)

THANKS