Networking

Matthias Tretter, @myell0w

UIWebView

- A full internet browser in a UIView
  Can use it not only to take your users to websites, but to display PDFs, for example.
- Built on WebKit
  An open source HTML rendering framework (started by Apple).
- Supports JavaScript
  But limited to 5 seconds or 10 megabytes of memory allocation.
- Property to control whether page is scaled to fit the UIView
  `@property (nonatomic) BOOL scalesPagesToFit;`
  If `YES`, then page will be squished or stretched to fit the width of the UIView.
  If `NO`, the page will be its natural size and the user cannot zoom inside it.
- Property to get the scroll view it uses
  `@property (nonatomic, readonly, strong) UIScrollView *scrollView;`
  Can now set properties in the scroll view to control behavior.

UIWebView

- Three ways to load up HTML …

```
- (void)loadRequest:(NSURLRequest *)request;
- (void)loadHTMLString:(NSString *)string baseURL:(NSURL *)baseURL;
- (void)loadData:(NSData *)data
        MIMEType:(NSString *)MIMEtype
textEncodingName:(NSString *)encodingName
         baseURL:(NSURL *)baseURL;
```

- We'll talk about `NSURLRequest` in a moment
- Base URL is the "environment" to load resources out of

  i.e., it's the base URL for relative URL's in the data or HTML string.
- MIME type says how to interpret the passed-in data

  Multimedia Internet Mail Extension

  Standard way to denote file types (like PDF).

  Think "e-mail attachments" (that's where the name MIME comes from).

UIWebView

- NSURLRequest

```
+ (NSURLRequest *)requestWithURL:(NSURL *)url;
+ (NSURLRequest *)requestWithURL:(NSURL *)url
                     cachePolicy:(NSURLRequestCachePolicy)policy
                 timeoutInterval:(NSTimeInterval)timeoutInterval;
```

- NSURL

  Basically like an `NSString`, but enforced to be "well-formed."

  Examples: `file://...` or `http://...`

  In fact, it is the recommended way to specify a file name in the iOS API.

```
+ (NSURL *)urlWithString:(NSString *)urlString;
+ (NSURL *)fileURLWithPath:(NSString *)path isDirectory:(BOOL)isDirectory;
```

- NSURLRequestCachePolicy

  Ignore local cache; ignore caches on the internet; use expired caches;
  use cache only (don't go out onto the internet); use cache only if validated

Synchronous Loading

- Blocks (main) thread, UI can't update
- not cancel-able
- Don't use, just for demonstration

NSString:

```
- (id)initWithContentsOfURL:(NSURL *)url
                   encoding:(NSStringEncoding)enc
                      error:(NSError **)error
```

NSData:

```
- (id)initWithContentsOfURL:(NSURL *)url
```

Synchronous Loading

```
NSString *address = @“http://www.some.url“;
NSURL *URL = [NSURL URLWithString:address];
NSError *error = nil;

NSString *s = [[NSString alloc] initWithContentsOfURL:URL
                                             encoding:NSUTF8StringEncoding
                                                error:&error];

// s contains content of file at URL
```

Blocks

- What is a block?

  A block of code (i.e. a sequence of statements inside {}).
  Usually included "in-line" with the calling of method that is going to use the block of code.
  Very smart about local variables, referenced objects, etc.

- What does it look like?

  Here's an example of calling a method that takes a block as an argument.

```
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@"value for key %@ is %@", key, value);
    if ([@"ENOUGH" isEqualToString:key]) {
        *stop = YES;
    }
}];
```

  This NSLog()s every key and value in aDictionary (but stops if the key is ENOUGH).

- Blocks start with the magical character caret ^

  Then it has (optional) arguments in parentheses, then {, then code, then }.

Blocks

- Can use local variables declared before the block inside the block

```
double stopValue = 53.5;
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@"value for key %@ is %@", key, value);
    if ([@"ENOUGH" isEqualToString:key] || ([value doubleValue] == stopValue)) {
        *stop = YES;
    }
}];
```

- But they are read only!

```
BOOL stoppedEarly = NO;
double stopValue = 53.5;
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@"value for key %@ is %@", key, value);
    if ([@"ENOUGH" isEqualToString:key] || ([value doubleValue] == stopValue)) {
        *stop = YES;
        stoppedEarly = YES;  // ILLEGAL
    }
}];
```

Blocks

- Unless you mark the local variable as __block

```
__block BOOL stoppedEarly = NO;
double stopValue = 53.5;
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@"value for key %@ is %@", key, value);
    if ([@"ENOUGH" isEqualToString:key] || ([value doubleValue] == stopValue)) {
        *stop = YES;
        stoppedEarly = YES;  // this is legal now
    }
}];
if (stoppedEarly) NSLog(@"I stopped logging dictionary values early!");
```

- Or if the "variable" is an instance variable

  But we only access instance variables (e.g. _display) in setters and getters.
  So this is of minimal value to us.

Blocks

- So what about objects which are messaged inside the block?

```
NSString *stopKey = [@“Enough” uppercaseString];
__block BOOL stoppedEarly = NO;
double stopValue = 53.5;
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@“value for key %@ is %@”, key, value);
    if ([stopKey isEqualToString:key] || ([value doubleValue] == stopValue)) {
        *stop = YES;
        stoppedEarly = YES;  // this is legal now
    }
}];
if (stoppedEarly) NSLog(@“I stopped logging dictionary values early!”);
```

stopKey will essentially have a strong pointer to it until the block goes out of scope or the block itself leaves the heap (i.e. no one points strongly to the block anymore).

Why does that matter?

Asynchronous Loading

- Doesn`t block (main) thread
- Cancel-able
- calls back via delegation or completion-block
- The way to go™

NSURLConnection

- An NSURLConnection object provides support to perform the loading of a URL request
- delegate-based or block-based

```
+ (void)sendAsynchronousRequest:(NSURLRequest *)request
                          queue:(NSOperationQueue *)queue
              completionHandler:(void(^)(NSURLResponse*,NSData*,NSError*))handler


- (id)initWithRequest:(NSURLRequest *)request
             delegate:(id<NSURLConnectionDelegate>)delegate
- (void)start
- (void)cancel
```

Asynchronous Loading

```
NSString *address = @“http://www.some.url“;
NSURL *URL = [NSURL URLWithString:address];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];

[NSURLConnection sendAsynchronousRequest:request
                                   queue:[NSOperationQueue mainQueue]
       completionHandler:^(NSURLResponse *r, NSData *d, NSError *e) {

    NSString *s = [NSString alloc] initWithData:d
                                       encoding:NSUTF8StringEncoding];

    // s contains value of file at address
}];
```

Demo

- Add UIWebView to display website
- Load list of websites from web
- UIRefreshControl to reload
- UITableView animation
- iPad: UISplitViewController