Mobile Application Development

Aileen Pierce

IOS AND JSON

JSON

- JavaScript Object Notation(JSON) is a language independent data format used to store and exchange data
- Supported by every major modern programming language including JavaScript, Swift, and Java
- JSON is built on two structures
  – A collection of name/value pairs
  – An ordered list of values

JSON

- Name/value pairs are stored as an object, record, struct, dictionary, hash table, keyed list, or associative array in various languages
  - An object in in curly brackets { }
  - Each name is followed by a colon :
  - Name/value pairs are separated by a comma ,
- Ordered list of values are stored as an array, vector, list, or sequence in various languages
  - An array is in square brackets [ ]
  - Values are separated by a comma ,

JSON Sample

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

JSON and iOS

- In iOS you can download a JSON file from a server using **`NSURLSession.sharedSession().dataTaskWithURL(NSURL, completionHandler: (NSData?, NSURLResponse?, NSError?)->Void)-> NSURLSessionDataTask`**
- Completion block
  - Success: the data parameter will hold the data downloaded, the error parameter will be nil
  - Fail: the error parameter will hold the error, the data parameter will be nil
  - The response is a NSHTTPURLResponse object
- After you create the task, you must start it by calling its resume method

JSON and iOS

- The HTTP status code is stored in the NSHTTPURLResponse statusCode property
  - 200 is OK
- Once the JSON has been downloaded successfully we are ready to parse the data
- We will use **`dispatch_async(queue, block)`** to parse the JSON asynchronously
  - Queue will be the main queue **`dispatch_get_main_queue()`**
  - Block will parse the JSON

JSON and iOS

- In iOS you can create a Foundation object (type AnyObject) from a JSON data object using

```
NSJSONSerialization.JSONObjectWithData(data,
options:NSJSONReadingOptions.AllowFragments)
```

  - Data: JSON data object (NSData)
  - NSJSONReadingOptions
    - NSJSONReadingMutableContainers
      - arrays and dictionaries are created as mutable objects
    - NSJSONReadingMutableLeaves
      - leaf strings in the JSON object graph are created as instances of NSMutableString
    - NSJSONReadingAllowFragments
      - the parser should allow top-level objects that are not an instance of NSArray or NSDictionary
  - This method could throw an error

JSON and iOS

- We can cast the returned object as an NSDictionary storing the key/value pairs
- You can use the keys and grab all the values and store them as an Array
- Iterate through the array and grab the data you want
- Create your own data model to store the data

Swift 2

- Swift 2 introduced a new guard statement
  - Like an if statement, the guard statement evaluates a boolean expression
  - Guard statements are run if the test condition is false
  - Lets you handle false conditions early
  - Always has an else clause that MUST transfer control out of the code block
  - Keeps the code that handles a violated condition next to the test condition
  - The code that is typically run is kept in the main flow and not in an else statement

Swift 2

- Avoids nested if statements (pyramids of doom)

```
if firstName != "" {
     if lastName != "" {
          if address != "" {
               // do great code
          }
     }
}
```

Swift 2

- The code that is typically run is kept in the main flow and not in an else statement

```
guard age > 18
     else { return false }
// main code
```

- Any variables defined in a guard statement remain in scope after the guard finishes.

```
guard let unwrappedName = name
     else { print(”Provide a name.")
          return
     }
print(unwrappedName)
```

Early Exit

- You can transfer control with an early exit
  - Continue
    - Used in loops to skip that iteration and go to the next iteration of the loop
  - Break
    - Used in loops or switch statements to exit completely out of the loop or switch and go on to the rest of the function
  - Return
    - Exits out of the current scope. In functions this will return control to where the function was called
  - Throw
    - Used to throw (return) an error

Error Handling

- Swift 2 introduced a new process for error handling
- A function can be defined to throw an error to indicate something unexpected happened

```
func canThrowErrors() throws -> String
```

- A `throw` statement returns an error and immediately transfers program control back to where the function was called

Error Handling

- Use the **`try`** keyword when calling a function that throws an error
- Use a do-catch statement to handle errors. If an error is thrown in the do clause, it is sent to the catch clause

```
do {
    try canThrowErrors()
    //code if no error
} catch {
    print("Error: \(error)")
}
```

- If a catch clause doesn't have a pattern, the clause matches any error and binds the error to a local constant named error