GENERAL ASSEMBLY

MOBILE DEVELOPMENT

NETWORKING – APIS AND PARSING

William Martin
Head of Product, Floored

LEARNING OBJECTIVES

- Describe a REST API and why they're important.
- Work with a REST API
- Parse JSON (and XML) data
- Incorporate REST+JSON into an app

APIS

WHAT IS AN API?

- An Application Programming Interface (or API) is typically a collection of protocols and functionality used to build software applications.
- In the networking world, we often talk about web services and applications providing APIs for other services and apps to consume.
- For example, Meetup.com provides an API that enables other applications to use meetup's data and functionality.
- Let's take a look! http://www.meetup.com/meetup_api/

WHAT IS AN API?

A good API will show:

- Available requests and the information they provide/require.
- Sample responses from those requests.
- Login / authentication procedure.
- If you're lucky, a way to test the API on their site without code.

Not all APIs are created equal. *Run some sample queries on Meetup's API*.

"RESTFUL" APIS

- So-called "RESTful" APIs are APIs that conform to the REST (REpresentational State Transfer) architecture style.
- https://en.wikipedia.org/wiki/Representational_state_transfer
- The style acts like a protocol, so we can have some expectations of what to expect whenever we encounter a "REST API."

"RESTFUL" APIS

Most important for us are the so-called RESTful "verbs," which come from the HTTP protocol.

The verbs both limit the scope of operations we can perform in an API and standardizes what those operations mean. This makes it easier to understand how to use the API:

- GET – List a collection (a set of entries), retrieve info about a single entry
- POST – Create an entry in a collection
- PUT – Replace a collection, replace an entry
- DELETE – Delete a collection, delete an entry

REQUESTING DATA FROM A "REST API"

Try this URL in your browser:

https://api.meetup.com/2/cities

Find the description of that request in the Meetup API documentation.

- http://www.meetup.com/meetup_api/

REQUESTING DATA FROM A "REST API"

You've just made a GET request to the Meetup API, and it returned a response consisting of a JSON-formatted list of the available cities in Meetup's database.

Neat! But how do we consume this response in an iOS app?

REVIEW OF DATA

WHAT ARE DATA?

- What are data?

83, 80, 82, 87, 86, 88, 78, 87, 88, 89, 85, 85, 88, 81, 83, 87, 90, 88, 95, 88, 96, 94, 93, 92, 89, 92, 86, 88, 97, 92, 89

WHAT ARE DATA?

- What are data?

83, 80, 82, 87, 86, 88, 78, 87, [illegible] 85, 88, 81, 83, 87, 90, 88, 95, 88, 96, 94, 93, 92, 89, 92, 86, 88, 97, 92, 89

What does it mean?!

- Data are pieces of information devoid of meaning.

WHAT ARE DATA?

- What are data?

83, 80, 82, 87, 86, 88, 78, 87, [illegible] 85, 88, 81, 83, 87, 90, 88, 95, 88, 96, 94, 93, 92, 89, 92, 86, 88, 97, 92, 89

What does it mean?!

- Data are pieces of information devoid of meaning.
- Once we know they are average daily temperatures measured in Central Park in May of 2015, now it has meaning.

WHAT ARE DATA?

- But for us, there's more to it than just the presence of meaning.
- When we retrieve data from the Internet, it needs some sort of recognizable structure, like the syntax for our Swift data structures (Arrays, Dictionaries, etc.).
- Without a consistent structure or syntax, Swift won't know where the numbers are. All it sees are a bunch of characters, like a big String.
- *Enter data formats…*

DATA FORMATS

DATA FORMATS

- Data needs "formatting" in order to portray structure.
- That is, if we have a list of numbers, like:
  83, 80, 82, 87, 86, 88, 78, 87, 88, 89, 85, 85, 88, 81, 83, 87, 90, 88, 95, 88, 96, 94, 93, 92, 89, 92, 86, 88, 97, 92, 89
- We use commas to delineate where the numbers are. If we hadn't used any formatting rules, we may have gotten:
  83808287868878878889858588818387908895889694939289928688979289
- Which doesn't really help anyone.
- There are two overall categories: binary and text.

DATA FORMATS

- **Binary formats** are machine-readable, but not human-readable.
  - Have the advantage of being very compact.
  - Generally impossible to use without knowing the rules of the format first (e.g. a specification or class).

- **Text formats** are both machine-readable and human-readable.
  - They take up more space.
  - But one can generally look at the raw data and figure out how to use it.

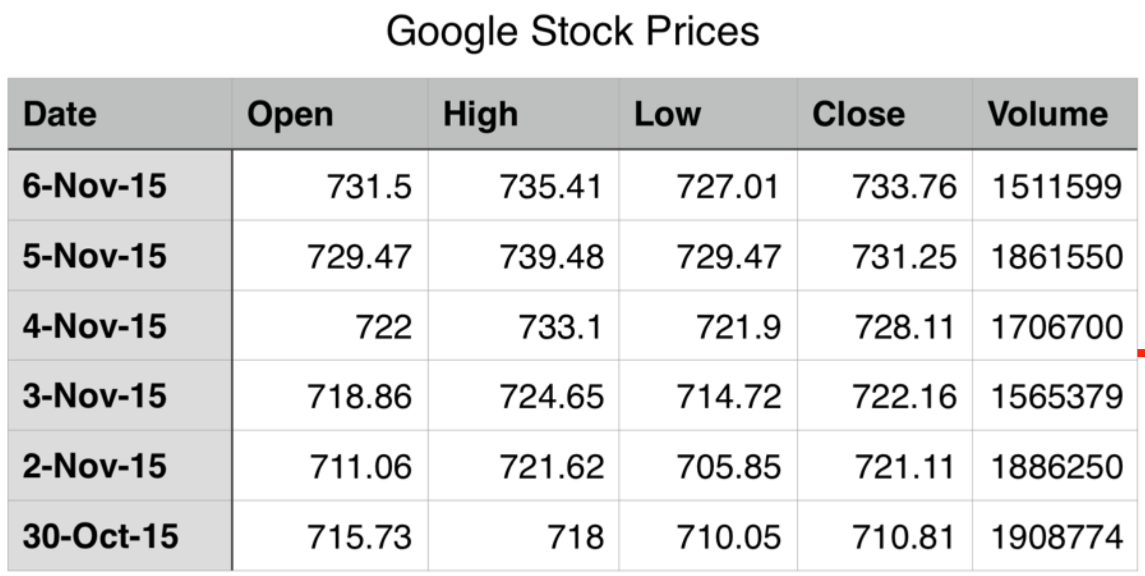
DATA FORMATS

- Different files types use formatting to contain data in a structured way for transferring data between applications and different computers.
- One of the simplest formats for containing tables of data is the comma-separated value format, or CSV.

DATA FORMATS : CSV

- The CSV file format defines a "syntax" that can take pieces of data and make them distinct and readable by a computer.
- More specifically, it describes tabular data, like a two-dimensional grid of numbers, where rows are groups of values of particular types and meanings, and columns show all the values of a particular parameter.
- Like a spreadsheet.

DATA FORMATS : CSV



Google Stock Prices

Date	Open	High	Low	Close	Volume
6-Nov-15	731.5	735.41	727.01	733.76	1511599
5-Nov-15	729.47	739.48	729.47	731.25	1861550
4-Nov-15	722	733.1	721.9	728.11	1706700
3-Nov-15	718.86	724.65	714.72	722.16	1565379
2-Nov-15	711.06	721.62	705.85	721.11	1886250
30-Oct-15	715.73	718	710.05	710.81	1908774

In a spreadsheet

```
Date,Open,High,Low,Close,Volume
6-Nov-15,731.50,735.41,727.01,733.76,1511599
5-Nov-15,729.47,739.48,729.47,731.25,1861550
4-Nov-15,722.00,733.10,721.90,728.11,1706700
3-Nov-15,718.86,724.65,714.72,722.16,1565379
2-Nov-15,711.06,721.62,705.85,721.11,1886250
30-Oct-15,715.73,718.00,710.05,710.81,1908774
```

CSV formatted version

https://www.google.com/finance/historical?q=NASDAQ%3AGOOG&ei=c8w_VojQHMSheeWcpJAC

PARSING

- While it may seem obvious where the values are in such a file (numbers are the continuous chunks of digits separated by commas), it's not trivial to turn this into a Swift data structure that we can use in an app.
- After all, what we get when reading a file is a String.
- We can convert `"110.0"` to a Double by using `Double("110.0")`, but what about `"110.0, 132.5"`?
- That comma makes things difficult.
- *What do we do?*

PARSING

We need to "parse" the data.

- Parsing is a process of converting the raw string representation of data into a data structure in a given language.
- The data format describes "rules" for finding values and how they relate to each other.
- It's the responsibility of a "parser," a special routine (function, class, application, etc.) to perform this conversion.

HOW DOES PARSING WORK?

The entire file is a single String, so split by the newline characters…

```
Date,Open,High,Low,Close,Volume\n
6-Nov-15,731.50,735.41,727.01,733.76,1511599\n
5-Nov-15,729.47,739.48,729.47,731.25,1861550\n
4-Nov-15,722.00,733.10,721.90,728.11,1706700\n
3-Nov-15,718.86,724.65,714.72,722.16,1565379\n
2-Nov-15,711.06,721.62,705.85,721.11,1886250\n
30-Oct-15,715.73,718.00,710.05,710.81,1908774\n
```

HOW DOES PARSING WORK?

Once you have a list of Strings for rows, split those by commas…

```
Date,Open,High,Low,Close,Volume
6-Nov-15,731.50,735.41,727.01,733.76,1511599
5-Nov-15,729.47,739.48,729.47,731.25,1861550
4-Nov-15,722.00,733.10,721.90,728.11,1706700
3-Nov-15,718.86,724.65,714.72,722.16,1565379
2-Nov-15,711.06,721.62,705.85,721.11,1886250
30-Oct-15,715.73,718.00,710.05,710.81,1908774
```

HOW DOES PARSING WORK?

Look at each entry and convert it to the appropriate type.

```
Date,Open,High,Low,Close,Volume
6-Nov-15,731.50,735.41,727.01,733.76,1511599
5-Nov-15,729.47,739.48,729.47,731.25,1861550
4-Nov-15,722.00,733.10,721.90,728.11,1706700
3-Nov-15,718.86,724.65,714.72,722.16,1565379
2-Nov-15,711.06,721.62,705.85,721.11,1886250
30-Oct-15,715.73,718.00,710.05,710.81,1908774
```

```
  NSDate   Double   Double   Double   Double    Int
```

HOW DOES PARSING WORK?

A more general approach iterates over individual characters…

```
Date,Open,High,Low,Close,Volume
6-Nov-15,731.50,735.41,727.01,733.76,151
5-Nov-15,729.47,739.48,729.47,731.25,186
4-Nov-15,722.00,733.10,721.90,728.11,170
3-Nov-15,718.86,724.65,714.72,722.16,156
```

HOW DOES PARSING WORK?

A more general approach iterates over individual characters…

```
Date,Open,High,Low,Close,Volume
6-Nov-15,731.50,735.41,727.01,733.76,15
5-Nov-15,729.47,739.48,729.47,731.25,18
4-Nov-15,722.00,733.10,721.90,728.11,17
3-Nov-15,718.86,724.65,714.72,722.16,15
```

HOW DOES PARSING WORK?

A more general approach iterates over individual characters…

```
Date,Open,High,Low,Close,Volume
6-Nov-15,731.50,735.41,727.01,733.76,151
5-Nov-15,729.47,739.48,729.47,731.25,186
4-Nov-15,722.00,733.10,721.90,728.11,170
3-Nov-15,718.86,724.65,714.72,722.16,156
2-Nov-15,711.06,721.62,705.85,721.11,1
```

HOW DOES PARSING WORK?

A more general approach iterates over individual characters…

```
Date,Open,High,Low,Close,Volume
6-Nov-15,731.50,735.41,727.01,733.76,151
5-Nov-15,729.47,739.48,729.47,731.25,186
4-Nov-15,722.00,733.10,721.90,728.11,170
3-Nov-15,718.86,724.65,714.72,722.16,156
```

HOW DOES PARSING WORK?

…until you reach special ones.

```
Date,Open,High,Low,Close,Volume
6-No                                  5,151
5-Nov                                 5,186
4-Nov 15,722.00,733.10,721.90,728.11,170
3-Nov-15,718.86,724.65,714.72,722.16,156
```

A comma! New value found!

Which is "`Date`". Hmm, this doesn't look like a number, date, or anything special, so assume it's a String.

HOW DOES PARSING WORK?

Iterate over characters until you reach special structural ones.

```
Date,Open,High,Low,Close,Volume\n
6-Nov-15,73
5-Nov-15,72
4-Nov-15,722.00,733.10,721.90,728.11,17
3-Nov-15,718.86,724.65,714.72,722.16,15
```

Newline! End of a complete entry!

This is the header row, so it's not all that interesting.
But at least we now know the names of all the fields.

HOW DOES PARSING WORK?

Iterate over characters until you reach special structural ones.

```
Date,Open,High,Low,Close,Volume\n
6-Nov-15,731.50,735.41,727.01,733.76,151
                        ,729.47,731.25,186
4-Nov-15,722.00,733.10,721.90,728.11,170
3-Nov-15,718.86,724.65,714.72,722.16,156
```

Do the same for each line of data.
This time, we find dates and numbers.

HOW DOES PARSING WORK?

A good parser will return a convenient data structure.

```
Date,Open,High,Low,Close,Volume
6-Nov-15,731.50,735.41,727.01,733.76,1511599
5-Nov-15,729.47,739.48,729.47,731.25,1861550
4-Nov-15,722.00,733.10,721.90,728.11,1706700
3-Nov-15,718.86,724.65,714.72,722.16,1565379
2-Nov-15,711.06,721.62,705.85,721.11,1886250
30-Oct-15,715.73,718.00,710.05,710.81,1908774
```

→

```
[
    {"Date" : NSDate(6-Nov-15),
     "Open" : 731.50,
     "High" : 735.41,
     "Low" : 727.01,
     "Close" : 733.76,
     "Volume" : 1511599},
    {"Date": NSDate(5-Nov-15),
     "Open" : 729.47,
```

HOW DOES PARSING WORK?

Writing a parser can be a pain.

Although it's a popular exercise in CS programs to write one (when studying programming languages, for instance), we won't be… *ever*…

Luckily, open source developers (those sneaky people) usually jump on such opportunities to write parsers in various languages for popular formats.

So we usually don't have to write them ourselves (unless we encounter a rare format for which no parser has been written).

JSON

JSON

- JSON is a data-interchange format.
    - It's a standardized text format that serves as a way to exchange data.
    - It stands for JavaScript Object Notation.
    - It takes its syntax from JavaScript objects.
    - http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

JSON

- Since most programming languages have similar data structures, JSON can be used to represent data structures in those that share those structures.
- JSON supports this nicely by being able to represent:
    - Arrays
    - Dictionaries (called "Objects" in JavaScript)
    - Strings
    - Numbers
    - Boolean values ("true" and "false")
    - Nil value ("null")

JSON

- Unlike HTML, this format doesn't imply anything about formatting natively.
- That is, where the "<h1>" tag in HTML is a major header, typically the largest standard text drawn on a webpage; there is no conventional equivalent in JSON. JSON is very generic and is specific only to the data it carries.
- Note: HTML is a special subset of XML.

JSON

- JSON makes it easy to read a data structure, but nothing about the JSON standard dictates anything about the actual content it carries.
- It only describes a standard method for structuring data, but it doesn't have anything that ensures all structures are alike.
- For example, as we'll discover, blog posts from different websites may have different structures, even though the information seems identical.

JSON EXAMPLES

A list of strings:

```
["hello", "doctor", "name", "continue", "yesterday", "tomorrow"]
```

A list of numbers:

```
[3.14, 2.718, 1.618, 6.28]
```

JSON EXAMPLES

A simple dictionary:

```
{
    "name" : "Toshi",
    "species" : "canis lupus familiaris",
    "breed" : "Shiba Inu",
    "age" : "3 yrs"
}
```

Note that spacing doesn't matter.

JSON EXAMPLES

A more complex dictionary:

```
{
    "name" : "Toshi",
    "species" : "canis lupus familiaris",
    "breed" : "Shiba Inu",
    "age" : "3 yrs",
    "checkups" : [{"date":"2015-10-10", "vet":"Lily Doloroso"},
                  {"date":"2015-04-02", "vet":"Lily Doloroso"},
                  {"date":"2014-11-01", "vet":"Lily Doloroso"}]
}
```

JSON EXAMPLES

From https://reddit.com/.json

```
{
  "kind": "Listing",
  "data": {
    "modhash": "",
    "children": [
      {
        "kind": "t3",
        "data": {
          "title": "This is cosplay done right",
          "banned_by": null,
          "media_embed": {
            "content": "&lt;iframe class=\"embedly-embed\" src=\"//cdn.embedly.com/
```

PARSING JSON IN IOS

So how do we "parse" JSON in iOS?

- There are native classes that do this for us, but the problem is that they're terrible to work with.
- So we're going to use a 3rd-party class called *SwiftyJSON*.
- Find it at http://github.com/SwiftyJSON/SwiftyJSON/.
- Download and add the single file, SwiftyJSON.swift, to your project.

PARSING JSON IN IOS

```
func onCompletion(data:NSData?, response:NSURLResponse?, error:NSError?) {
    if let _data = data {
        let json = JSON(data:_data)
        let posts = json["data"]["children"].array!
        for post in posts {
            let title = post["data"]["title"].string!
            print(title)
        }
    }
}

if let url = NSURL(string: "https://reddit.com/.json") {
    let sessionMgr = NSURLSession.sharedSession()
    sessionMgr.dataTaskWithURL(url, completionHandler: onCompletion)
    task.resume()
}
```

PARSING JSON IN IOS

Our statements `json["data"]["children"].array!` mimic the JSON structure:

```
{
  "kind": "Listing",
  "data": {
    "modhash": "",
    "children": [
      {
        "kind": "t3",
        "data": {
          "title": "This is cosplay done right",
          "banned_by": null,
          "media_embed": {
            "content": "&lt;iframe class=\"embedly-embed\" src=\"//cdn.embedly.com/
```

This is a single Reddit post.

PARSING JSON IN IOS

So, the routines we need for converting JSON into Swift are API-specific.

We have to examine the response structure of each API and write code that parses that data.

Rarely will two APIs be alike.

Best practice: Once you have a routine that works for parsing the API response data, then build a data model around it so you can work with it easily in your app.

JSON CODE PRACTICE

XML

XML

Another common data format we'll encounter in APIs is XML, or "eXtensible Markup Language."

It looks a lot like HTML, but it doesn't describe the content of webpages by default; it's much more general than that.

XML

XML files use elements and attributes to structure data. For example:

```
<dog>
    <name>Toshi</name>
    <species>canis lupus familiaris</species>
    <breed>Shiba Inu</breed>
    <age units="years">3</age>
</dog>
```

XML

Elements

```xml
<dog>
    <name>Toshi</name>
    <species>canis lupus familiaris</species>
    <breed>Shiba Inu</breed>
    <age units="years">3</age>
</dog>
```

Attributes

XML

XML tends to be more verbose (thus bigger files and more data to transfer) and harder to write manually than JSON.

The two formats aren't easily transformable from one to another. (JSON doesn't have attributes, for example.)

RSS

A lot of data comes in XML form, however. Especially RSS ("Rich Site Summary" or "Really Simple Syndication").

http://www.whatisrss.com/

The RSS standard provides a consistent structure for blog posts so we don't have to rewrite a parsing routine for every API.

Example: https://www.reddit.com/.rss

PARSING XML

To parse XML (and RSS), I recommend:

https://github.com/drmohundro/SWXMLHash

It's similar to SwiftyJSON, having a slightly different syntax:

```
let xml = SWXMLHash.parse(data)
let posts = xml["rss"]["channel"]["item"].all

for post in posts {
    let title = post["title"].element!.text!
    print(title)
}
```