INTRO TO APP DEVELOPMENT

DAY 2: API, JAVASCRIPT, TECH STACKS, AND TEAM STRUCTURES

AGENDA

- What is an API? REST?
- Set up Cloudant accounts
- Using Cloudant as our Database
- Access API via Javascript
- Web Stacks
- Development Cycles

API

- What is an API?

  Application Programming Interface

- REST API (REpresentational State Transfer)
- HTTP verbs: GET, POST, PUT, UPDATE, DELETE
- New York Times API: (sign up for an account)

  http://developer.nytimes.com/

REST STANDS FOR "REPRESENTATIONAL STATE TRANSFER"

- An "architectural style" for the web
- The formal REST constraints are:
  - Client–server - This separation of concerns means that, for example, clients are not concerned with data storage, which remains internal to each server, so that the portability of client code is improved. Servers are not concerned with the user interface or user state, so that servers can be simpler and more scalable.
  - Stateless - no client context being stored on the server between requests
  - Cacheable - Responses must therefore, implicitly or explicitly, define themselves as cacheable, or not, to prevent clients from reusing stale or inappropriate data in response to further requests.
  - Layered system - such that lower layers provide functions and services that support the functions and services of higher layers.
  - Uniform interface

-wikipedia

UNIFORM INTERFACE

- The uniform interface constraint is fundamental to the design of any REST service. The uniform interface simplifies and decouples the architecture, which enables each part to evolve independently. The four constraints for this uniform interface are:

1. Identification of resources
2. Manipulation of resources through these representations
3. Self-descriptive messages
4. Hypermedia as the engine of application state (HATEOAS)

-wikipedia

UNIFORM INTERFACE

- **Identification of resources**

  Individual resources are identified in requests, for example using URIs in web-based REST systems. The resources themselves are conceptually separate from the representations that are returned to the client. For example, the server may send data from its database as HTML, XML or JSON, none of which are the server's internal representation.

- **Manipulation of resources through these representations**

  When a client holds a representation of a resource, including any metadata attached, it has enough information to modify or delete the resource.

- **Self-descriptive messages**

  Each message includes enough information to describe how to process the message. For example, which parser to invoke may be specified by an Internet media type (previously known as a MIME type).

- **Hypermedia as the engine of application state (HATEOAS)**

  Clients make state transitions only through actions that are dynamically identified within hypermedia by the server (e.g., by hyperlinks within hypertext). Except for simple fixed entry points to the application, a client does not assume that any particular action is available for any particular resources beyond those described in representations previously received from the server.

-wikipedia

WEB SERVICE APIS THAT ADHERE TO THE REST ARCHITECTURAL CONSTRAINTS ARE CALLED **RESTFUL APIS**. HTTP-BASED RESTFUL APIS ARE DEFINED WITH THE FOLLOWING ASPECTS:

1. base URI, such as http://example.com/resources/
2. an Internet media type for the data. This is often JSON but can be any other valid Internet media type (e.g., XML, Atom, microformats, application/vnd.collection+json, etc.)
3. standard HTTP methods (e.g., OPTIONS, GET, PUT, POST, or DELETE)
4. hypertext links to reference state
5. hypertext links to reference-related resources

-wikipedia

JSON – JAVASCRIPT OBJECT NOTATION

- Why use JSON formatting?

```
{
    key: “value”,
    another_key: “string”,
    a: true,
    b: 12345,
}
```

USING THE CONSOLE

index.html

```
<!DOCTYPE html>

<html>

<head>

  <meta http-equiv="Content-Type" content="text/html;
  charset=utf-8" />

    <title>Class 2</title>

    <script type="text/javascript" src="scripts/main.js">

    </script>

</head>

<body>

     <h1>Class 2</h1>

</body>

</html>
```

scripts/main.js

```
console.log("1 hello");
```

ADDING JQUERY CDN

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Class 2 Example Page</title>
</head>
<body>
    <h1 id=“myText">Class 2</h1>
    <script src=“https://code.jquery.com/jquery-1.12.0.min.js">
    </script>
    <script type="text/javascript" src=“scripts/main.js">
    </script>
</body>
</html>
```

scripts/main.js

```
console.log(“2 hello”);

var myText = $(‘#myText’);
```

THERE ARE MANY DIFFERENT TYPES OF API'S

JQUERY API

HTTP://API.JQUERY.COM/

- .text()
- .get()
- .post()

NY TIMES CONGRESS

HTTP://DEVELOPER.NYTIMES.COM/

- http://api.nytimes.com/svc/politics/{version}/us/legislative/congress/{congress-number}/{chamber}/members[.response-format]?[optional-params]&api-key={your-API-key}

NEW YORK TIMES CONGRESS API

http://api.nytimes.com/svc/politics/{version}/us/legislative/congress/{congress-number}/{chamber}/members[.response-format]?[optional-params]&api-key={your-API-key}

EXAMPLE JQUERY GET

```
var textA = $('#textA');

$.get( "http://example.com", function( data ) {

  textA.html( data );

  alert("Load was performed.");

});
```

USE JQUERY .GET() API

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>
    Class 2 Example Page
  </title>
</head>
<body>
  <h1 id=“myText">
    Class 2
  </h1>
  <div id="textA"></div>
  <script src=“https://
  code.jquery.com/
  jquery-1.12.0.min.js">
  </script>
  <script type="text/
  javascript" src=“scripts/
  main.js”>
```

scripts/main.js

```
console.log('3 hello');

var myText = $('#myText').text("hello");

var textA = $('#textA');


//this is an example of how you use jQuery .get() API

$.get( "http://example.com", function( data ) {

  textA.html( data );

});


console.log("you will get an error here since 'http://
example.com' will not allow access");
```

USING THE ENDPOINT

```
var membersEndpoint = “http://api.nytimes.com/svc/
politics/{version}/us/legislative/congress/
{congress-number}/{chamber}/members[.response-
format]?[optional-params]&api-key={your-API-key}";
```

```
$.get( membersEndpoint, function( data ) {
  textA.html( data );
  alert("Load was performed.");
});
```

SPLIT UP YOUR VARIABLES

main.js

```
1  console.log("4 hello");
2  var myText = $('#myText').text("hello class");
3
4  var textA = $('#textA');
5
6  var version = 1;
7  var congressNumber = 2;
8  var chamber = 3;
9  var responseFormat = 4;
10 var optionalParams = 5;
11 var APIKey= 6;
12
13 var membersEndpoint = 'http://api.nytimes.com/svc/politics/'+
14                        version +'/us/legislative/congress/' +
15                        congressNumber + '/' +
16                        chamber +'/members'+
17                        responseFormat + '?[' +
18                        optionalParams +']&api-key=' +
19                        APIKey;
20
21 $.get( membersEndpoint, function( data ) {
22   textA.html( data.copyright );
23   console.log(data);
24 });
25
26 console.log("you will get an error here since your variables are not set");
27 console.log("error in this line: http://api.nytimes.
       com/svc/politics/1/us/legislative/congress/2/3/members4?[5]&api-key=6 ");
```

FINAL RESULT

main.js

```
1  console.log("5 main");
2  var textA = $('#textA');
3
4  var version = 'v3';
5  var congressNumber = 113;
6  var chamber = 'senate';
7  var responseFormat = '.json';
8  var optionalParams = 'CA';
9  var APIKey= '15abcdefgd4ebefe64fc4f930b97a4c5:1:71588093';
10 //not a real api key
11
12 var membersEndpoint = 'http://api.nytimes.com/svc/politics/'+
13                        version +'/us/legislative/congress/' +
14                        congressNumber + '/' +
15                        chamber +'/members'+
16                        responseFormat + '?[' +
17                        optionalParams +']&api-key=' +
18                        APIKey;
19
20 $.get( membersEndpoint, function( response ) {
21     textA.html( response.copyright );
22     console.log(response);
23 });
24
```

FINAL RESULT — WITH DONE()

main.js

```
1  console.log("6 main");
2  var textA = $('#textA');
3
4  var version = 'v3';
5  var congressNumber = 113;
6  var chamber = 'senate';
7  var responseFormat = '.json';
8  var optionalParams = 'CA';
9  var APIKey= '15abcdefgd4ebefe64fc4f930b97a4c5:1:71588093';
10 //not a real api
11
12 var membersEndpoint = 'http://api.nytimes.com/svc/politics/'+
13                        version +'/us/legislative/congress/' +
14                        congressNumber + '/' +
15                        chamber +'/members'+
16                        responseFormat + '?[' +
17                        optionalParams +']&api-key=' +
18                        APIKey;
19
20 $.get( membersEndpoint, function( response ) {
21     textA.html( response.copyright );
22 }).done(function(response) {
23     console.log('the whole response:', response);
24     var insideTheResponse = response.results[0].members[3].twitter_account;
25     console.log('one part of the response;', insideTheResponse);
26     textA.html(insideTheResponse);
27 });
28
```

CLOUDANT.COM

- Database as a Service (Service)
- $50 limit per month for free service
  - depending on Heavy Usage/Light Usage/Data Storage
- Create a database
- Create a document
- Upload pictures into documents with names

CLOUDANT

- /_all_dbs
- /{db} or /baseball
- /{db}/{_id} or /baseball/sfgiants
- _id
- rev

ACCESS CLOUDANT API VIA JAVASCRIPT

main.js

```
1  console.log('cloudant main');
2
3  var textB = $('#textB');
4  var cloudantEndpoint = 'https://michelle.cloudant.com/exampledb/mydocument';
5
6  $.ajax({
7    url: cloudantEndpoint,
8    type: 'GET',
9    dataType: 'text',
10   xhrFields: {
11     withCredentials: true
12   },
13   success: function (response) {
14     // look at the response
15     console.log('data', typeof response);
16
17     //turn response into an object
18     var obj = JSON.parse(response);
19
20     //look at the object
21     console.log('object', obj);
22
23     //access the objects through keys and values
24     textB.text(obj.spiritanimal);
25   }
26 });
```

WEB STACKS

- MAMP, LAMP, WAMP - Mac, Apache Server, MySQL, PHP
- MEAN - MongoDB, ExpressJS, AngularJS, NodeJS

DEVELOPMENT TERMINOLOGY

- Agile Methodology - able to move quickly and easily / characterized by the division of tasks into short phases of work and frequent reassessment and adaptation of plans: Contrasted with waterfall.
- Scrum Methodology - part of agile: Scrum is a simple set of roles, responsibilities, and meetings that never change. By removing unnecessary unpredictability, we're better able to cope with the necessary unpredictability of continuous discovery and learning.
- Lean 1 Eliminate waste 2 Amplify learning 3 Decide as late as possible 4 Deliver as fast as possible 5 Empower the team 6 Build integrity in 7 See the whole
- Ticketing/ Scrum board
- Waterfall