Spatial Data Capture and Analysis

Interactive Visualisation 3: Node.JS

Steven Gray

1. Introduction to Databases
2. Introduction to SQL
3. Advanced SQL
4. Data Handling and Cleaning
5. Contextualising Data
6. Clustering and Regression
7. Interactive Viz 1: HTML + CSS
8. Interactive Viz 2: Javascript
9. **Server Side Coding: Node.JS**
10. Real-time data visualisation

Recap

A bit of JavaScript

```
var jedi = {
    name: "Yoda",
    age: 899,
    talk: function () { console.log("may the force be with you"); },
    father:  0,
    child: undefined
};

console.log(jedi.name);
console.log(jedi.age);
jedi.talk();

if(jedi.father == true && child != undefined){
   console.log(jedi.child.name + ", I am your father!");
}
```

Recap

A little bit of jQuery

```
$("button").click(function(){
        var div=$("div");
        div.animate({height: '300px', opacity: '0.4'}, "slow");
        div.animate({width: '300px', opacity: '0.8'}, "slow");
        div.animate({height: '100px', opacity: '0.4'}, "slow");
        div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

jQuery.animate can chain events to make your site amazing

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_animation

This week

API’s, sharing data, and NodeJS

What’s an API

Quick Guide

APPLICATION PROGRAMING INTERFACE

Allows applications (and machines) to pass data between each other

What's an API

Application Programming Interfaces

Send Data

Pass some variables via a URL

Get the Data back

Server send results back

Server

Application

http://dev.spatialdatacapture.org:8870/data/**:lat/:lon/:radius**

http://dev.spatialdatacapture.org:8870/data/**51.514756/-0.104345/50**

What’s an API

Behind the scenes and a bit of a recap

http://dev.spatialdatacapture.org:8870/data/:lat/:lon/:radius

http://api.nytimes.com/svc/books/{version}/lists

http://content.guardianapis.com/search?q=debates

https://graph.facebook.com/v2.2/{album-id}

Various Applications give you as a developer access to data via an API

What's an API

Behind the scenes and a bit of a recap

http://api.nytimes.com/svc/books/{version}/lists

Books API: Best Sellers

With the Best Sellers request type, you can get data from all New York Times best-seller lists, including rank history for specific best sellers. To learn more about the Book Reviews request type, go here.

Note: In this document, curly braces { } indicate required items. Square brackets [ ] indicate optional items or placeholders.

The Best Sellers Request Type at a Glance	
Base URI	`http://api.nytimes.com/svc/books/{version}/lists`
Scope	New York Times best-seller lists from June 2008 to present (one-week delay)
HTTP method	GET
Response formats	JSON (`.json`, default), XML (`.xml`), serialized PHP (`.sphp`), JSONP (`.jsonp`)
Quick links	Requests | Responses | Examples | Errors | Feedback

To retrieve best-seller lists, you must sign up for an API key for the Books API. Usage is limited to 5,000 requests per day (rate limits are subject to change). Please read and agree to the API Terms of Use and the Attribution Guidelines before you proceed.

Requests

The Best Sellers service uses a RESTful style. Five request types are available: get a best-seller list, search best-seller lists, get the history of a best seller, get an overview of all of the best-seller lists for a given week and get the names of Times best-seller lists.

For general tips on creating a request URI, see Constructing a Request.

COMMON PARAMETERS

These parameters are used in all request types. See below for URI structures and additional parameters for each type of request.

Name and Description	Required?	Value and Notes
version **The API version**	Yes	v2 (in URI path)
api-key **Your registered API key**	Yes	Alphanumeric (in query string) For more information, see Requesting a Key.
response-format **Sets the representation data format**	No	`.json` | `.xml` | `.sphp` | `.jsonp` (extension) To return XML, serialized PHP or JSONP, add the appropriate extension to your request URI. JSON is the default format; you do not need to add the `.json` extension to get a JSON response. To

http://developer.nytimes.com/docs/books_api/Books_API_Best_Sellers

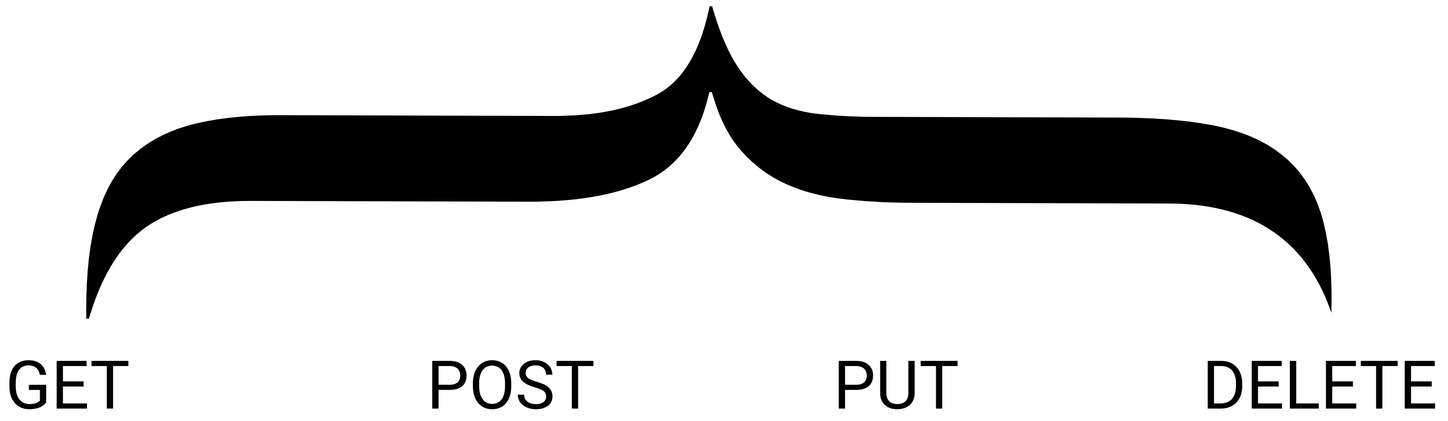
All API's have Documentation

If you are creating an API, then provide documentation!

What’s an API

Methods and Verbs

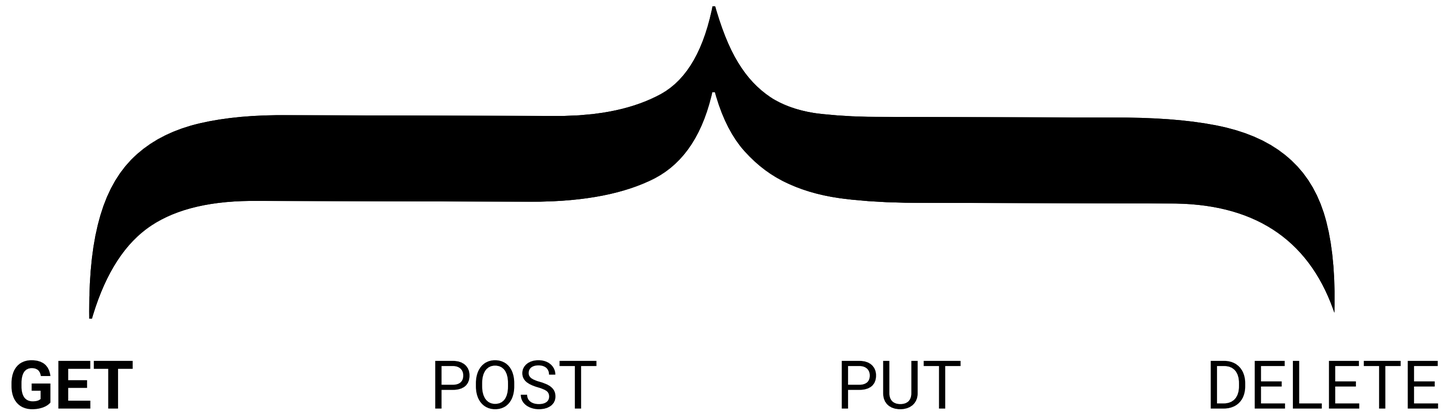
http://dev.spatialdatacapture.org:8870/data/:lat/:lon/:radius



What's an API

Methods or Verbs

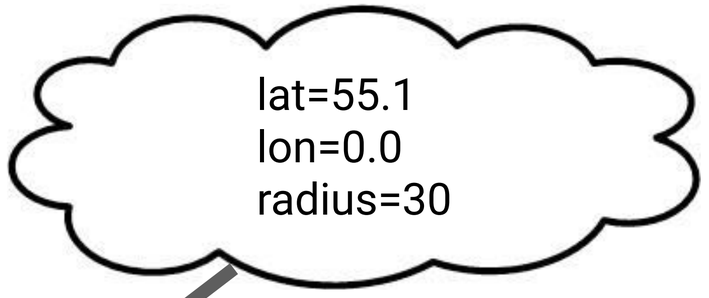
http://dev.spatialdatacapture.org:8870/data/:lat/:lon/:radius



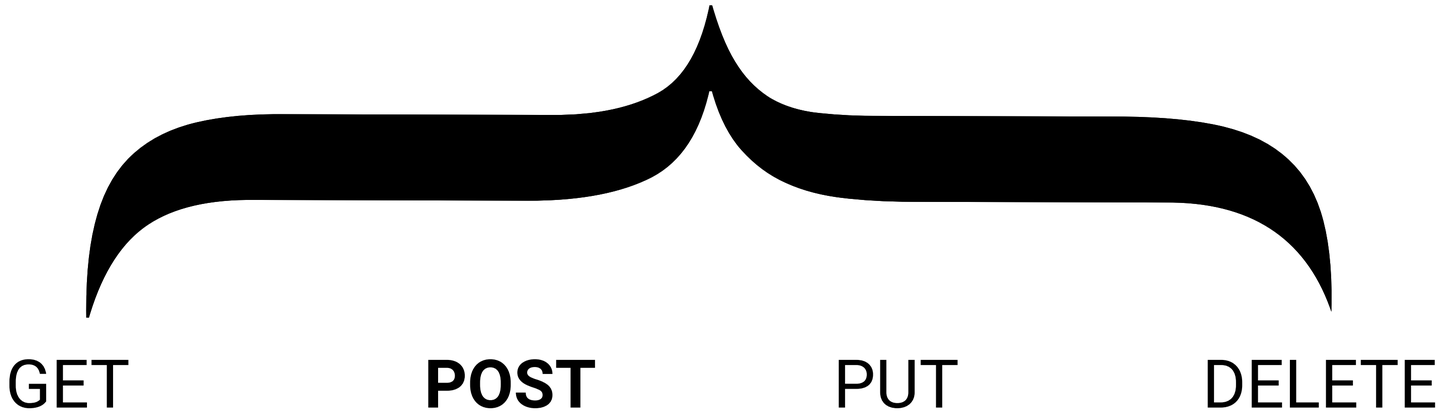
GET data without passing any data (or through request)

What's an API

Methods or Verbs



http://dev.spatialdatacapture.org:8870/data

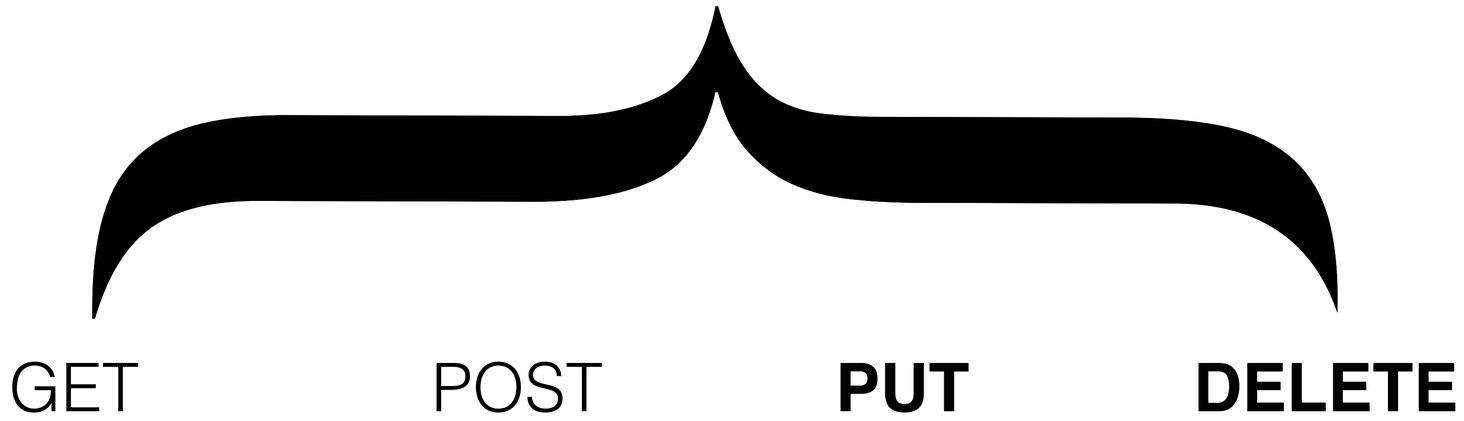


POST data before getting data

What’s an API

Methods or Verbs

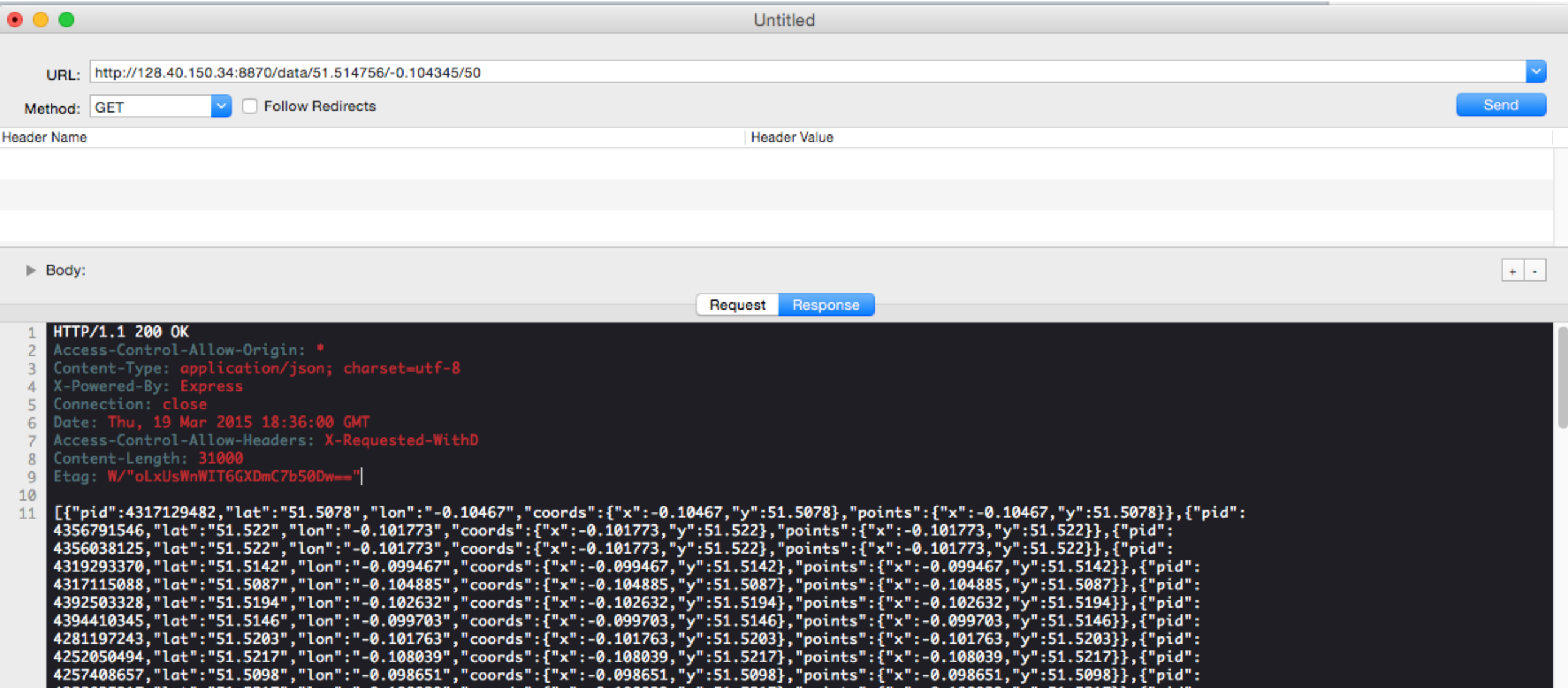
http://dev.spatialdatacapture.org:8870/data/uploadData



Special Verbs to trigger specific commands on servers

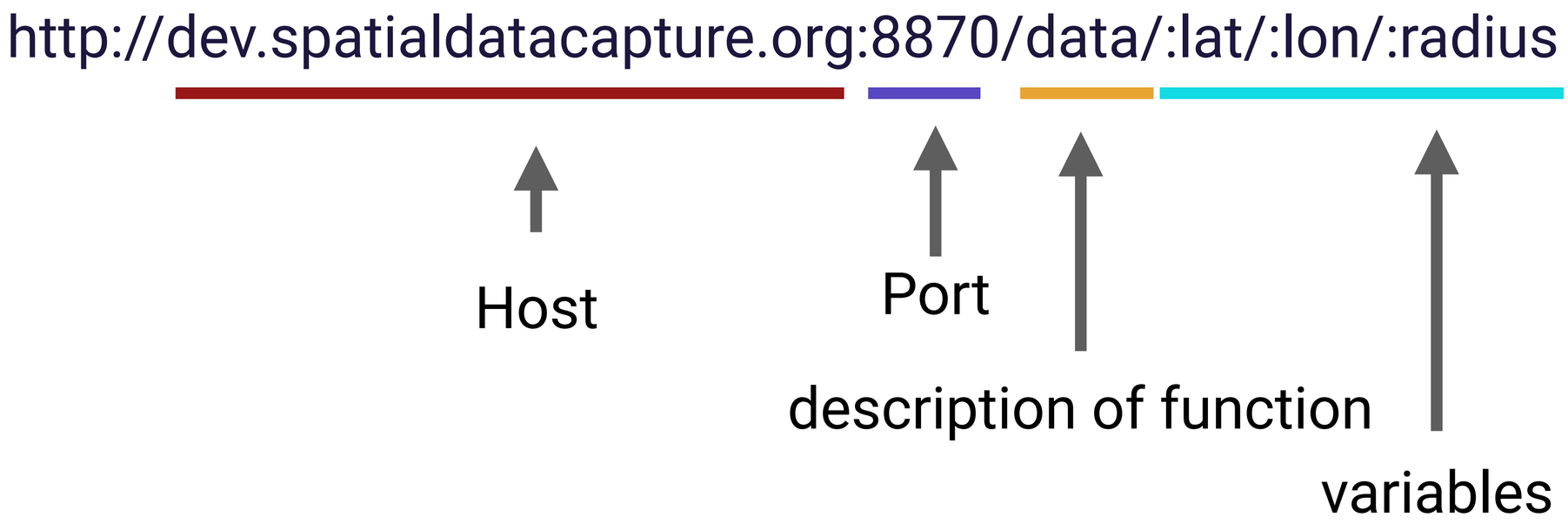
What's an API

Data is returned from the API



What’s an API

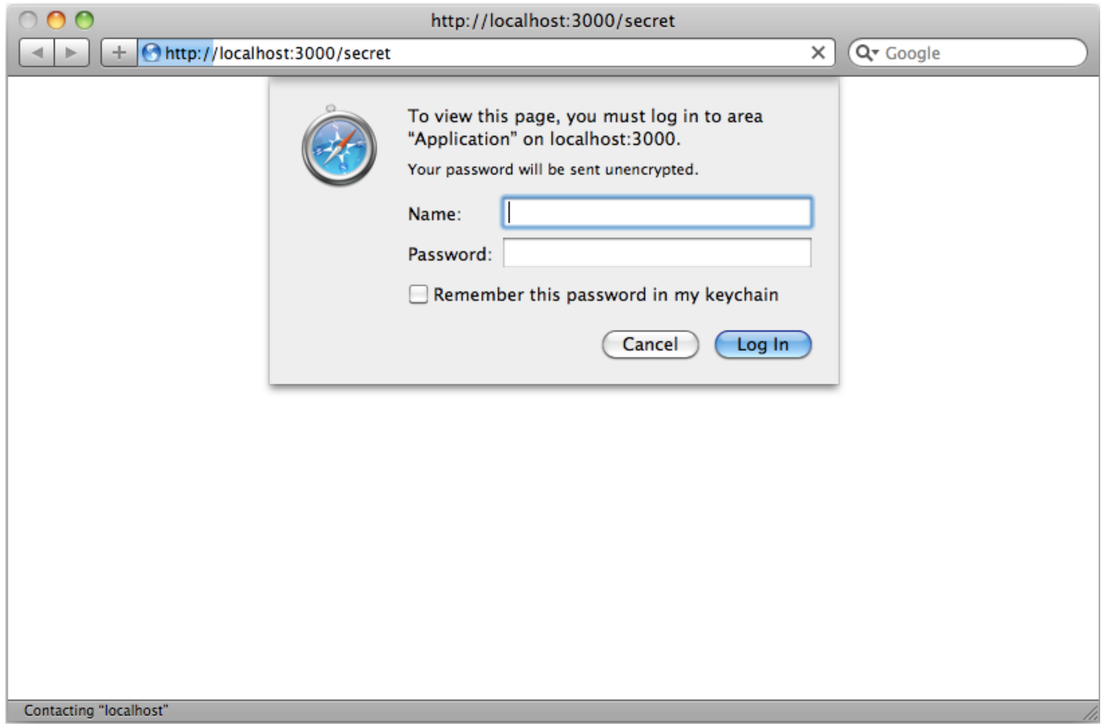
URL’s and their Significance



Breaking down a GET requests to an API’s

What's an API

Authentication for API's



Basic Authentication

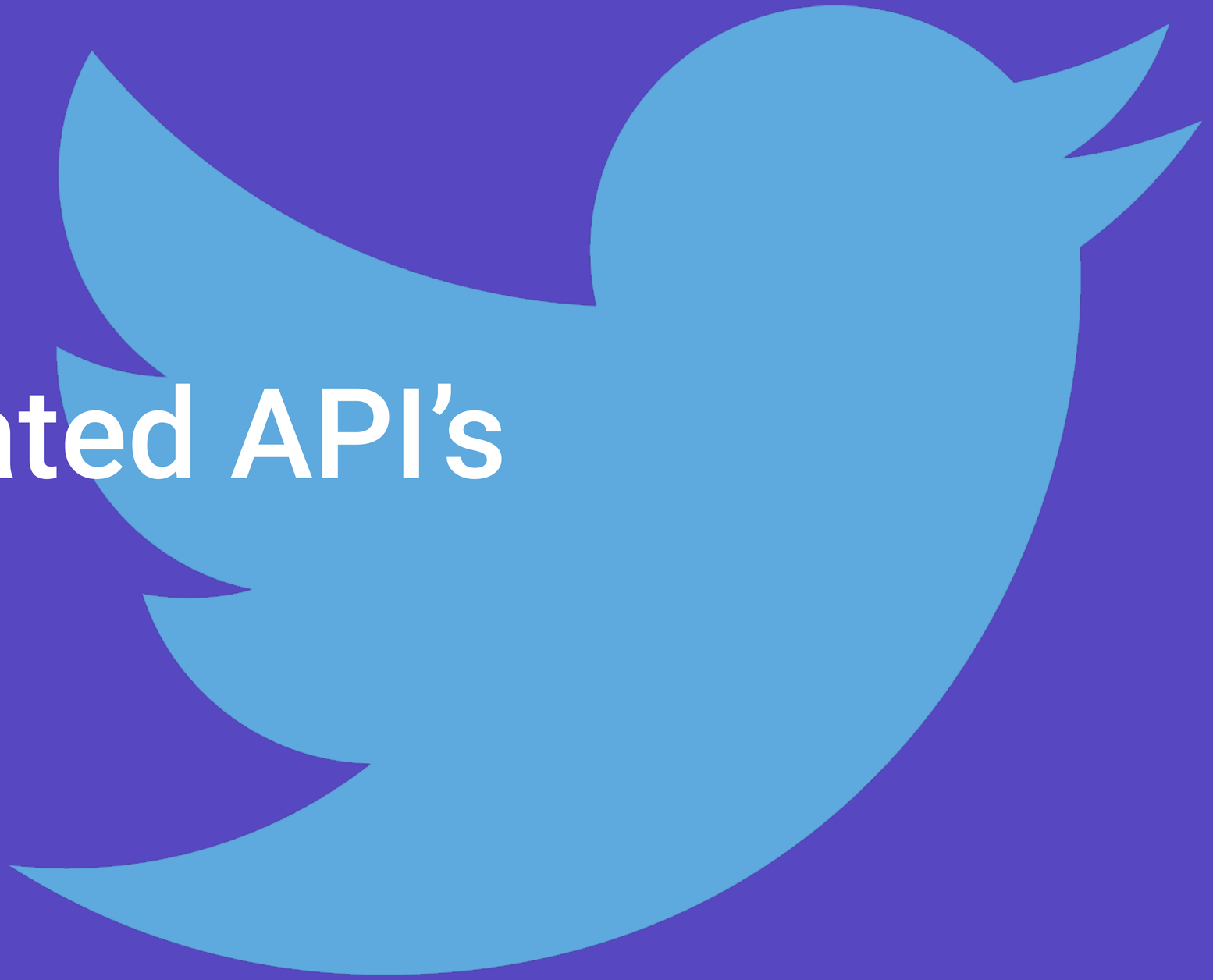
Username and Password Combination

OAuth 1.0 and 2.0

Token Based Authentication

Authenticated API’s

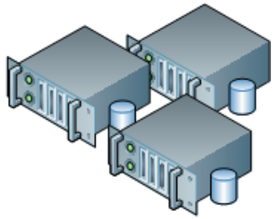
A real world example



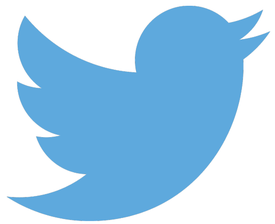
What's an API

Real world example of using Twitter's API

User Profiles



Tweet Database



http://dev.twitter.com/doc

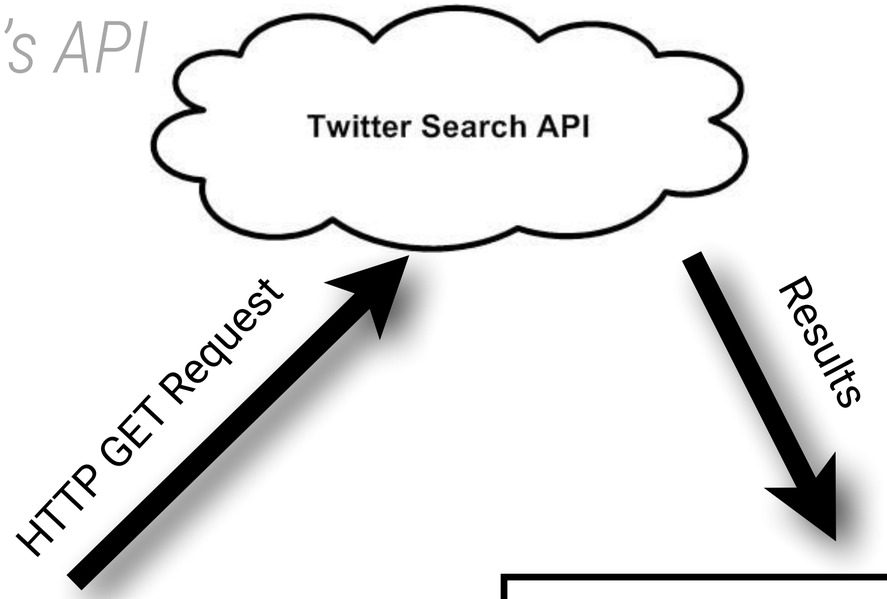
What's an API

Real world example of using Twitter's API

Returns tweets that:

- match specified query
- located in Lat/Lon box
- Historical Tweets
- User Profile Information
- Return Public Timeline

Service is rate limited



Your Application

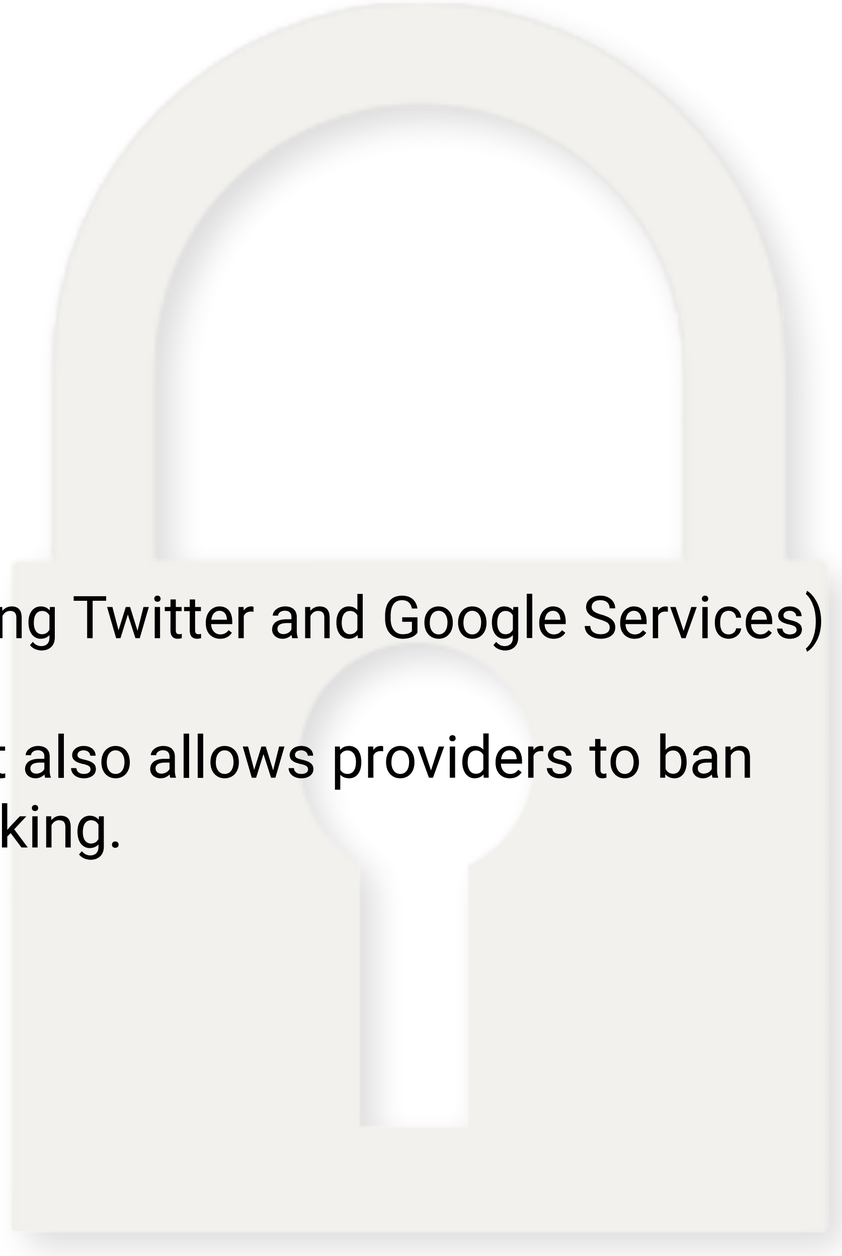

```
{"results":[
      {"text":"@twitterapi is great!",
      "to_user_id":396524,
      "to_user":"TwitterAPI",
      "from_user":"jkoum",
      "metadata":
      {
       "result_type":"popular",
       "recent_retweets": 100

      },
      "id":1478555574,
      "from_user_id":1833773,
      "iso_language_code":"nl",
       ....}
}
```


JSON or Atom

What’s an API

Authentication of Twitter API

2 Methods of Authentication
  -- Basic HTTP Authentication (Deprecated)
  -- oAuth 2.0 (By Standard)



Most modern day web services use oAuth (including Twitter and Google Services)

Keeps information safe from the wrong people but also allows providers to ban people if they are using too many resources or making.

Open Authentication (OAuth)

Real world example of using Twitter's API

Application

Request a Session Token

Pass a ID and a Secret

Asks Login with ID

Server gives a Pin number for session

Server

https://api.twitter.com/oauth/authenticate?oauth_token=Z6eEdO8MOmk394WozF5oKyuAv855l4Mlqo7hhlSLik

Open Authentication (OAuth)

Real world example of using Twitter's API

Application

Login is successful

:)

Sends App Tokens

These need to be saved!

Server

https://api.twitter.com/oauth/request_token

Open Authentication (OAuth)

Real world example of using Twitter's API

Application

Request using tokens

Server checks if it's valid

Data Returned

:)

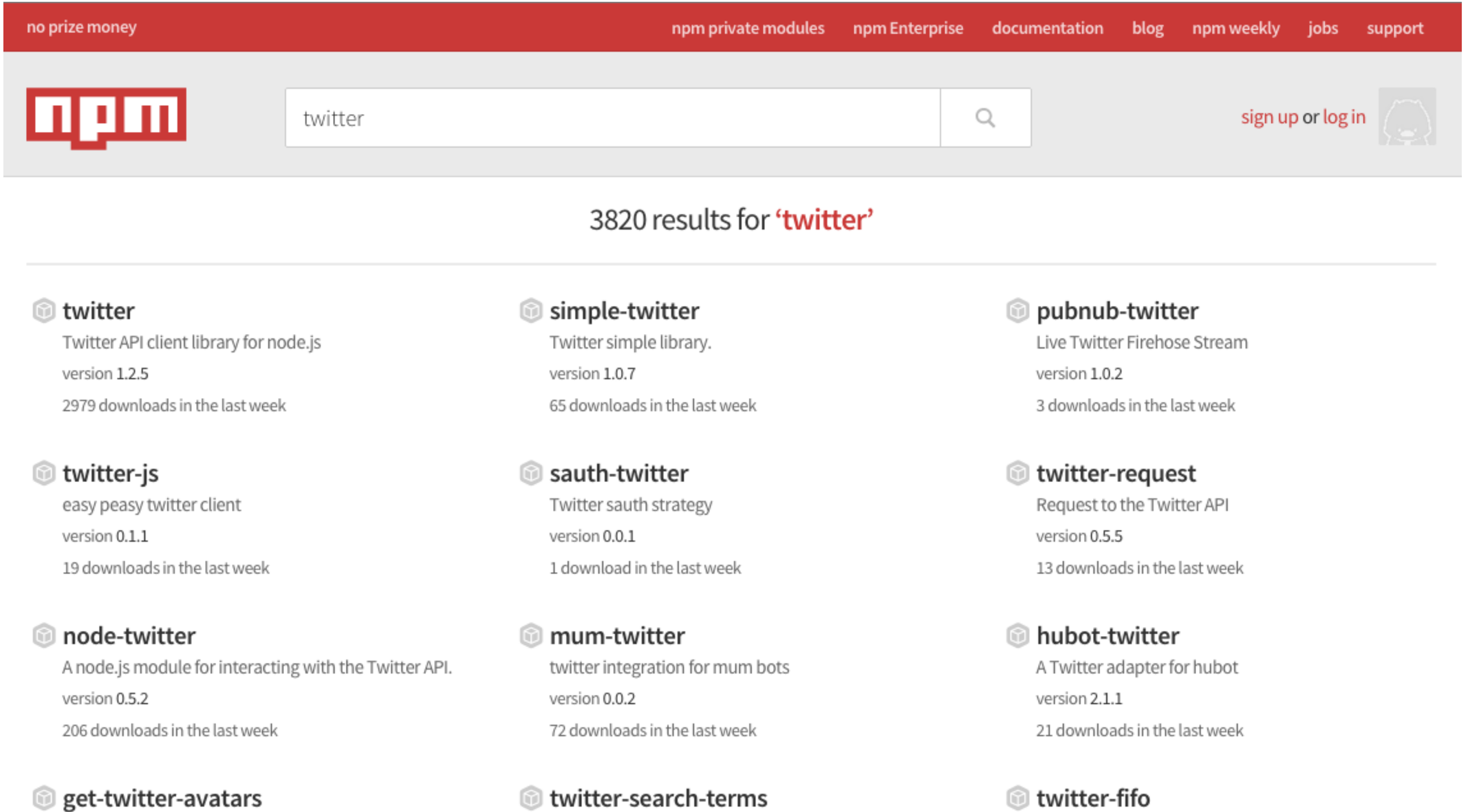
Server

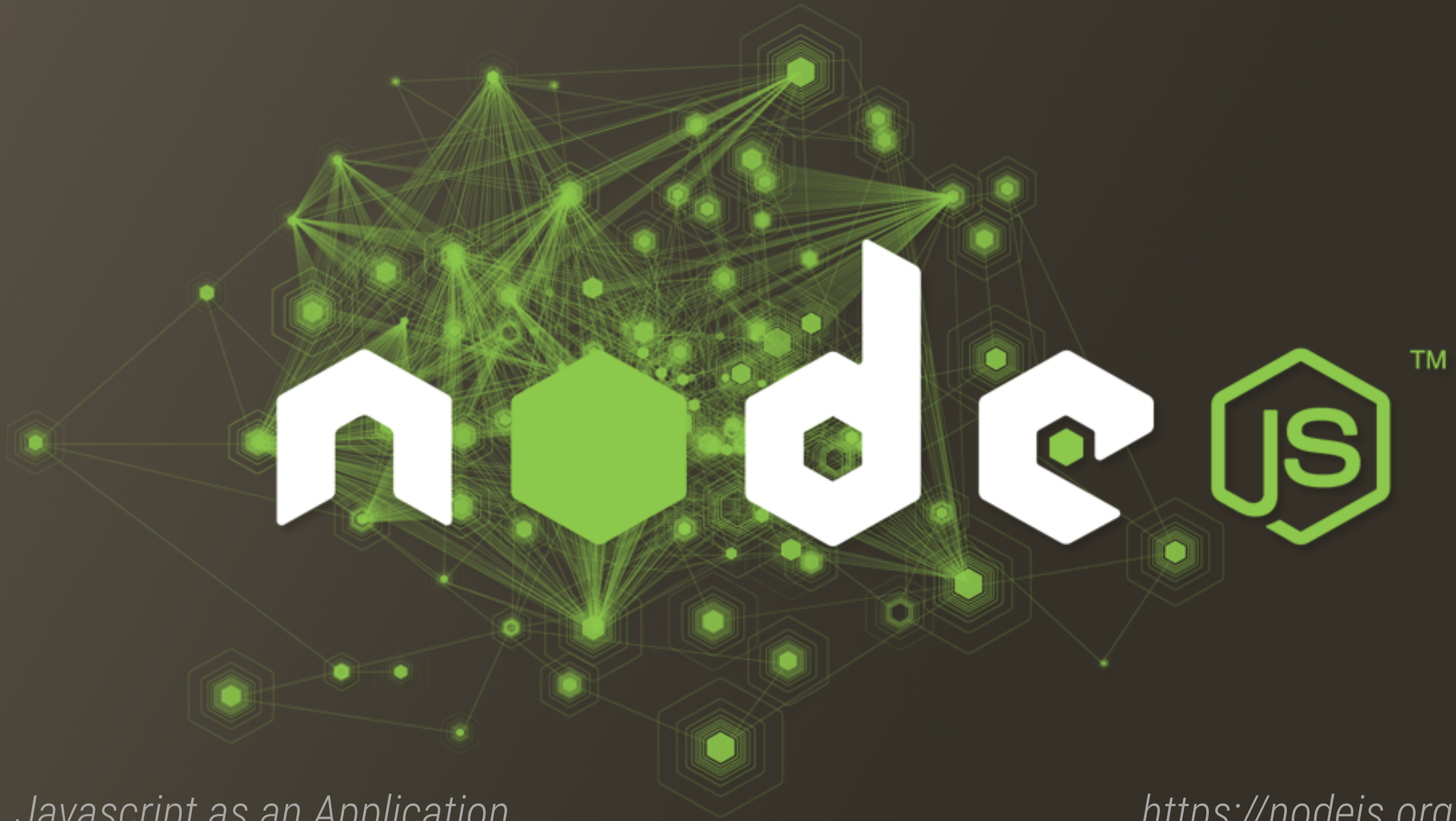
https://api.twitter.com/1.1/search/tweets.json?q=london&since_id=24012619984051000&max_id=25012619984051814 5&result_type=mixed&count=100

'Authorization: OAuth oauth_consumer_key="XjZipyQVHhjkasdhfkajs65vwTvA", oauth_nonce="db273a234a2b7f95235de5d62cc711c56db8e3"

Open Authentication (OAuth)

If in doubt use a library!





Javascript as an Application

https://nodejs.org

Introduction to Node.JS

Javascript as an Application

Scalable, Fast Programming Langauge

- Built to host lightweight applications for modern computing
- Asynchronous nature makes it perfect for web applications
- Launched in 2009

Open Source, Cross Platform

- Lots of Libraries (Packages) to use
- Extremely Portable
- Uses same JavaScript engine as Chrome (V8) - but won't run in browser
- File Extension is **.js**

Note: Also known as io.js (https://iojs.org)

Same as Node but a community driven effort to update node and make it truly open source

Introduction to Node.JS

Javascript coding on the server

Your machine

Request Web Application

http://www.stevenjamesgray.com

Server

NodeJS Application

Introduction to Node.JS

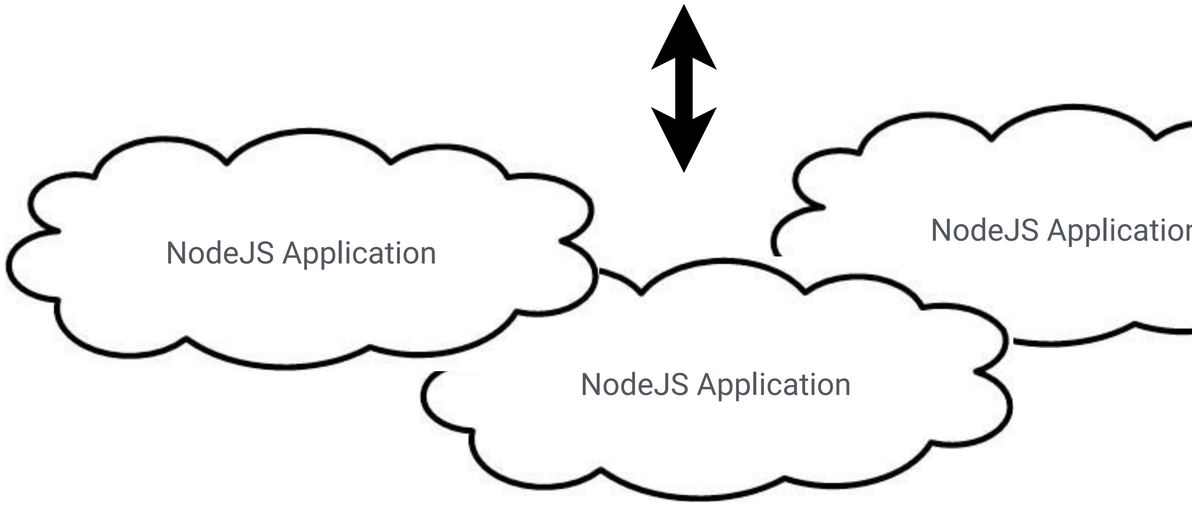
Javascript coding on the server

Your machine

Request Web Application

http://www.stevenjamesgray.com

Server



Introduction to Node.JS

Example of a Node.JS script


```
// displayServer.js

// NodeJS Socket IO Server to serve data to the data table display unit
// Author:   Steven Gray
// Date:     28/04/2014
// Version:  1.0

var table_id = "a43d";
var port = 8080
var io = require('socket.io').listen(port);
var address,os = require('os'),ifaces = os.networkInterfaces();
var Firebase = require('firebase');
io.set('log level', 0);

var currentRunningID = {id:"", duration: ""};
var currentRunningVizList = undefined;

// Iterate over interfaces ...
for (var dev in ifaces) {
    var iface = ifaces[dev].filter(function(details) {
        return details.family === 'IPv4' && details.internal === false;
    });
    if(iface.length > 0) address = iface[0].address;
}

console.log("Starting Display Server on "+ address + ":" + port)
console.log("Connected to Firebase");
```

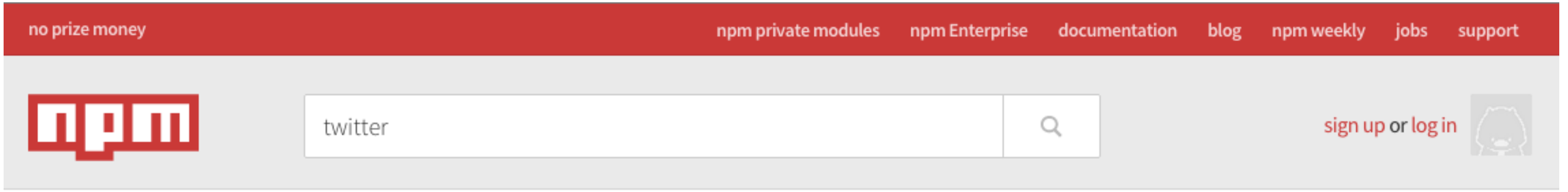
Same Syntax as JavaScript

Only difference is that this is a system wide application and not running in the browser.

Introduction to Node.JS

Package Management - External Libraries

https://www.npmjs.com



3820 results for 'twitter'

twitter
Twitter API client library for node.js
version 1.2.5
2979 downloads in the last week

simple-twitter
Twitter simple library.
version 1.0.7
65 downloads in the last week

pubnub-twitter
Live Twitter Firehose Stream
version 1.0.2
3 downloads in the last week

twitter-js
easy peasy twitter client
version 0.1.1
19 downloads in the last week

sauth-twitter
Twitter sauth strategy
version 0.0.1
1 download in the last week

twitter-request
Request to the Twitter API
version 0.5.5
13 downloads in the last week

node-twitter
A node.js module for interacting with the Twitter API.
version 0.5.2
206 downloads in the last week

mum-twitter
twitter integration for mum bots
version 0.0.2
72 downloads in the last week

hubot-twitter
A Twitter adapter for hubot
version 2.1.1
21 downloads in the last week

Introduction to Node.JS

Installing Packages for Node.JS

Command Line

```
npm install mysql
```

Node Package Manager

NPM installs packages into your current folder

Your scripts (apps) must be in the same folder to use that package

Introduction to Node.JS

Running Node Scripts

Command Line

```
node first.js
```

You must be in the folder to execute your script

Introduction to Node.JS

Using Packages inside you application


```
var mysql = require('mysql');

// MySQL Connection Variables
var connection = mysql.createConnection({
  host     : 'dev.spatialdatacapture.org',
  user     : '',
  password : '',
  database : ''
});
```

Load the library into a variable and then use that variable as commands

Don’t assign anything else to that variable or you will lose the reference to the library (package)

Introduction to Node.JS

Building an API with Node.JS

http://expressjs.com

npm install express

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World!')
})

var server = app.listen(3000, function () {

  var host = server.address().address
  var port = server.address().port

  console.log('Example app listening at http://%s:%s', host, port)

})
```

ExpressJS - A web server written in Node.JS

Can host a full website but we will use Express to build a API server like we used last week.

Introduction to Node.JS

Building an API with Node.JS


```
// respond with "Hello World!" on the homepage
app.get('/', function (req, res) {
  res.send('Hello World!');
})

// accept POST request on the homepage
app.post('/data', function (req, res) {
  res.send('Got a POST request');
})

// accept PUT request at /user
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user');
})

// accept DELETE request at /user
app.delete('/deleteUser', function (req, res) {
  res.send('Got a DELETE request at /user');
})
```

http://localhost:8080/

http://localhost:8080/data

http://localhost:8080/user

http://localhost:8080/deleteUser

Defining your endpoints

Your API will carry out the code that is between each block when a browser requests it.

Understanding how it works

Looking at last Weeks API Server Code


```
var portNumber = 8870;

var mysql = require('mysql');

// MySQL Connection Variables
var connection = mysql.createConnection({
  host     : 'dev.spatialdatacapture.org',
  user     : '',
  password : '',
  database : ''
});

connection.connect();

//  Setup the Express Server
var express = require('express')
var app = express()
app.set('view engine', 'ejs');

// Provides the static folders we have added in the project to the web server.
app.use(express.static(__dirname + '/js'));
app.use(express.static(__dirname + '/css'));
app.use(express.static(__dirname + '/images'));

// Default API Endpoint - return the index.ejs file in the views folder
app.get('/', function(req, res) {
    return res.render('index');
})
```

```
//  API EndPoint to get data from specific area - /data/51.1/0.0/30
app.get('/data/:lat/:lon/:radius', function (req, res) {

      // Allows data to be downloaded from the server with security concerns
      res.header("Access-Control-Allow-Origin", "*");
      res.header("Access-Control-Allow-Headers", "X-Requested-WithD");
      // If all the variables are provided connect to the database
      if(req.params.lat != "" && req.params.lon != "" && req.params.radius != ""){

                // Parse the values from the URL into numbers for the query
                var lat = parseFloat(req.params.lat);
                var lon = parseFloat(req.params.lon);
                var radius = parseFloat(req.params.radius);


                // SQL Statement to run
                var sql = "SELECT * FROM photo_locations WHERE DISTANCE(points, POINT("+lon+","+lat+") ) <= " + radius;

                // Log it on the screen for debugging
                console.log(sql);

                // Run the SQL Query
                connection.query(sql, function(err, rows, fields) {
                        if (err) console.log("Err:" + err);
                        if(rows != undefined){
                                // If we have data that comes bag send it to the user.
                                res.send(rows);
                        }else{
                                res.send("");
                        }
                });
        }else{
                // If all the URL variables are not passed send an empty string to the user
                res.send("");
        }
});
```

```
// API Endpoint to get data for specific photograph from database - /data/photoDescription/1234567
app.get('/data/photoDescription/:pid', function (req, res) {
      res.header("Access-Control-Allow-Origin", "*");
      res.header("Access-Control-Allow-Headers", "X-Requested-WithD");
      if(req.params.pid != ""){
                var pid = parseInt(req.params.pid);

                var sql = "SELECT * FROM metadata as a INNER JOIN photos as b ON a.pid = b.pid where a.pid = " + pid;

                console.log(sql);
                connection.query(sql, function(err, rows, fields) {
                        if (err) console.log("Err:" + err);
                        if(rows != undefined){
                                res.send(rows);
                        }else{
                                res.send("");
                        }
                });
        }else{
                res.send("");
        }
});

// Setup the server and print a string to the screen when server is ready
var server = app.listen(portNumber, function () {
  var host = server.address().address;
  var port = server.address().port;
  console.log('App listening at http://%s:%s', host, port);
})
```

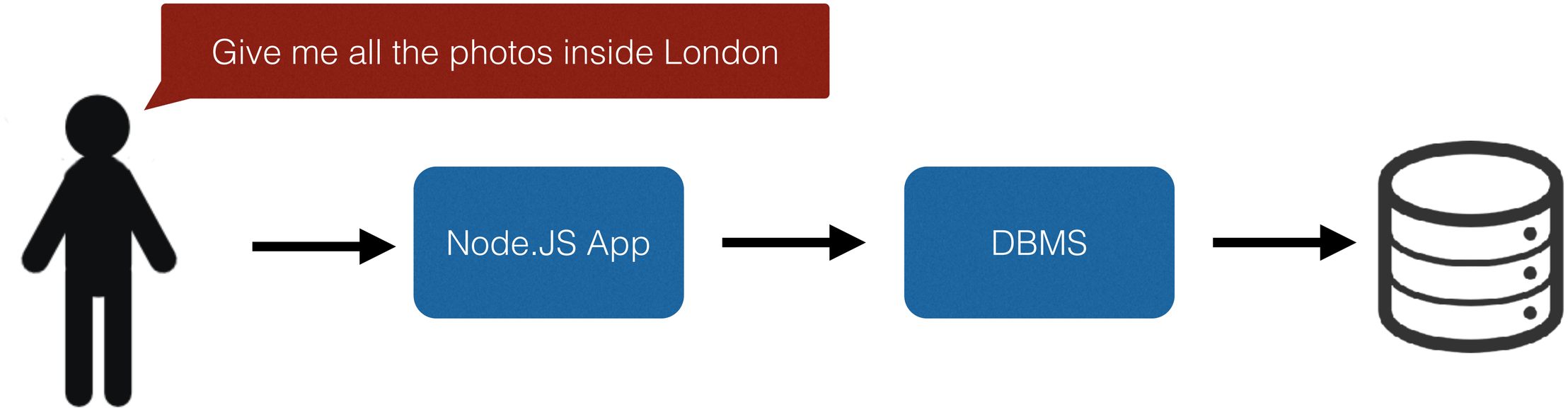
Building an API

Behind the scenes

Remember this from Week 1

Building an API

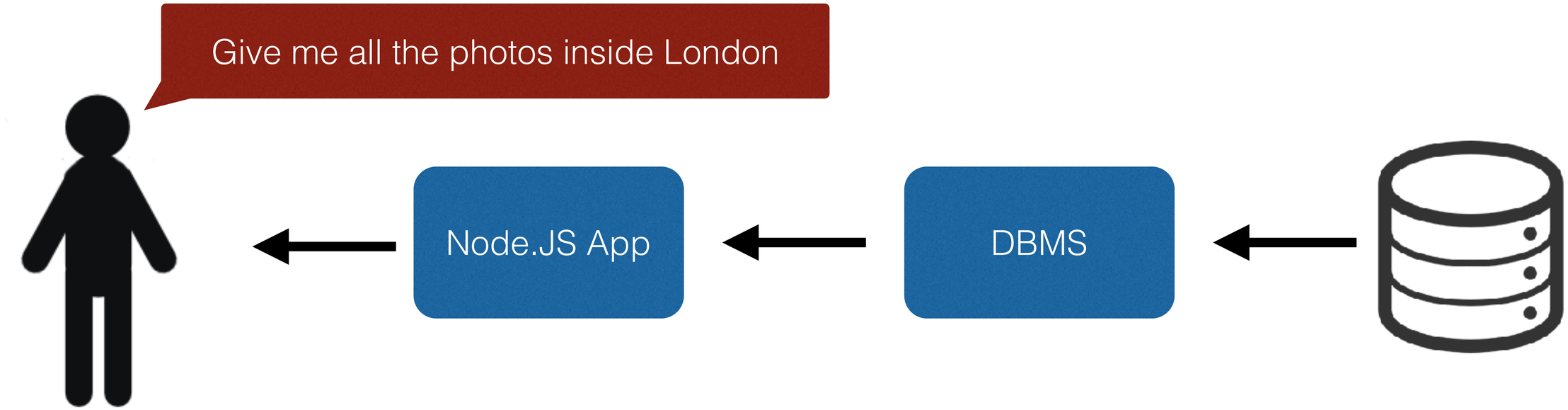
Behind the scenes



Node Application makes the call for us to Database and returns the data

Building an API

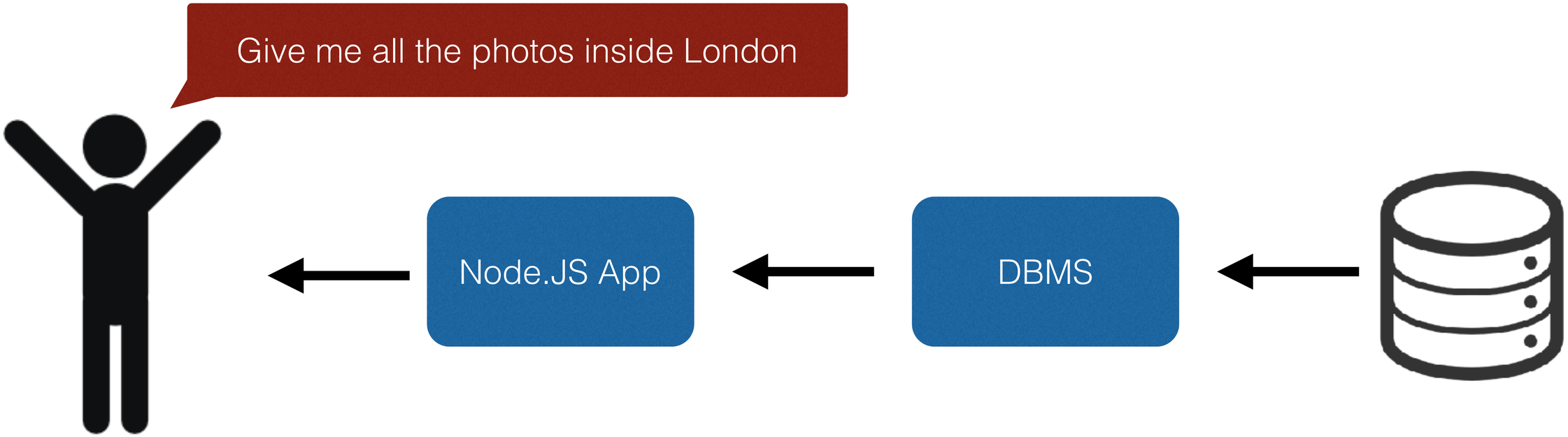
Behind the scenes



Node Application makes the call for us to Database and returns the data

Building an API

Behind the scenes



Node Application makes the call for us to Database and returns the data

A Warning about SQL and API's

SQL Injection

```
var myDevice = 'iPhone';
var sql = "SELECT * FROM metadata WHERE device = '" + myDevice + "'";
connection.query(sql, function(err, rows, fields) {
    …
});
```

What happens when I pass a different variable type through an API end point?

(e.g a number)

A Warning about SQL and API's

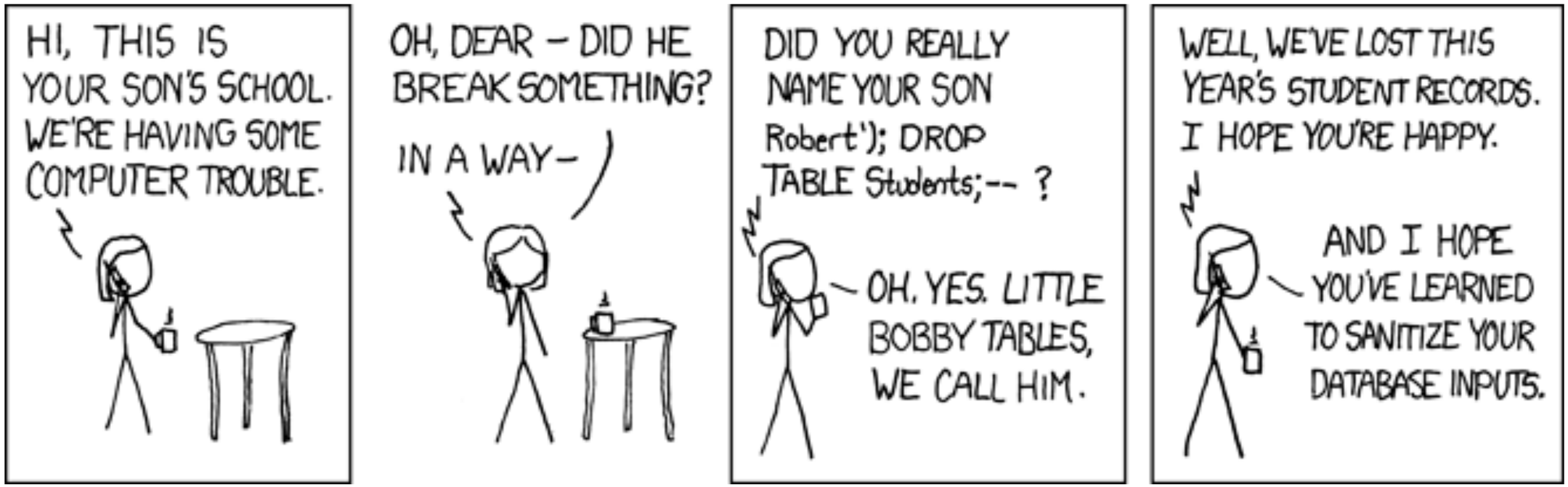
SQL Injection

```
var myDevice = 'iPhone';
var sql = "SELECT * FROM metadata WHERE device = '" + myDevice + "'";
connection.query(sql, function(err, rows, fields) {
    …
});
```

What happens If I pass an SQL command instead of a device?

A Warning about SQL and API's

SQL Injection



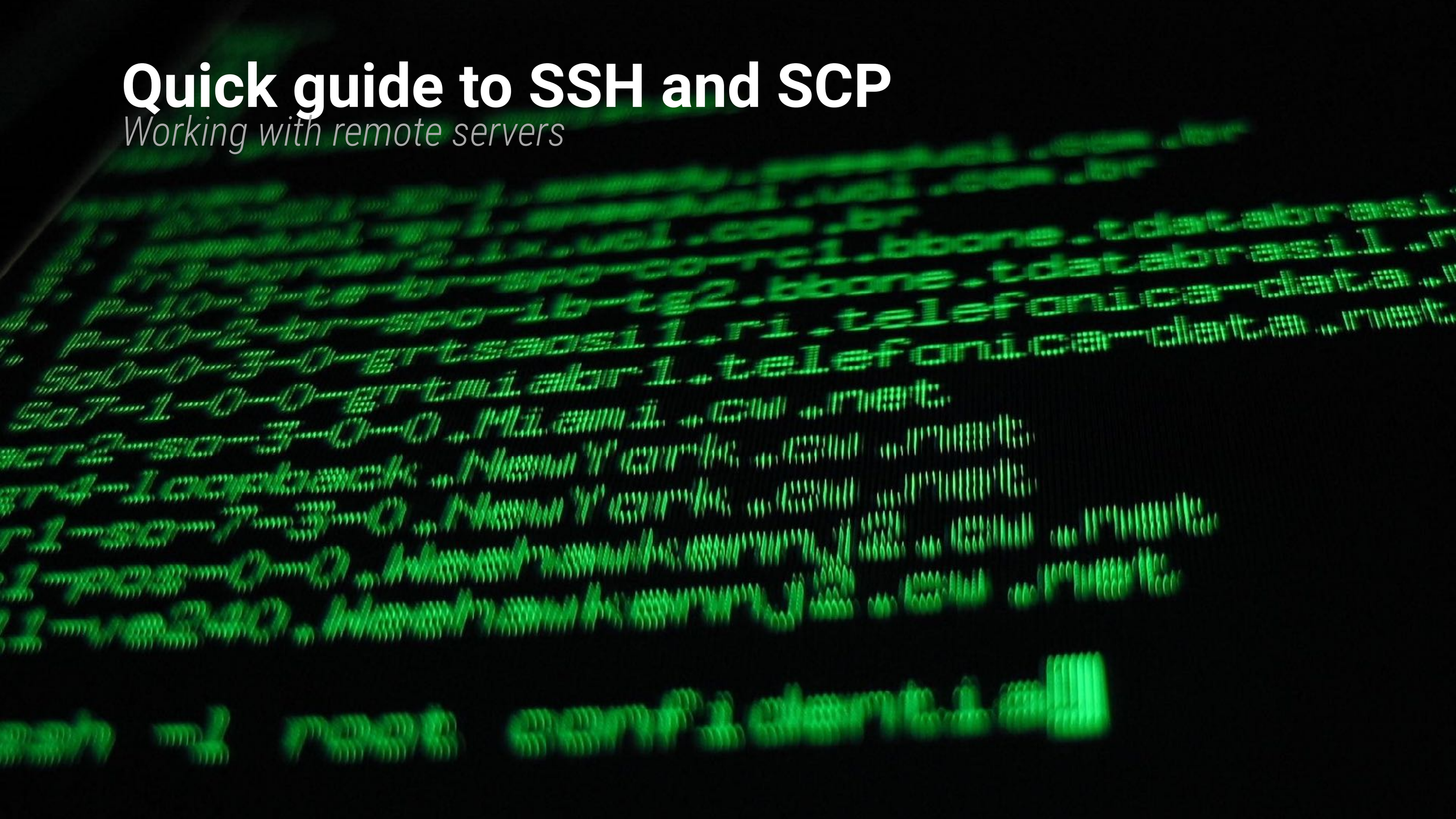
Sanitise all variables before running a query

```
// API Endpoint to get data for specific photograph from database - /data/photoDescription/1234567
app.get('/data/photoDescription/:pid', function (req, res) {
      res.header("Access-Control-Allow-Origin", "*");
      res.header("Access-Control-Allow-Headers", "X-Requested-WithD");
      if(req.params.pid != ""){
                var pid = parseInt(req.params.pid);

                var sql = "SELECT * FROM metadata as a INNER JOIN photos as b ON a.pid = b.pid where a.pid = " + pid;

                console.log(sql);
                connection.query(sql, function(err, rows, fields) {
                        if (err) console.log("Err:" + err);
                        if(rows != undefined){
                                res.send(rows);
                        }else{
                                res.send("");
                        }
                });
        }else{
                res.send("");
        }
});

// Setup the server and print a string to the screen when server is ready
var server = app.listen(portNumber, function () {
  var host = server.address().address;
  var port = server.address().port;
  console.log('App listening at http://%s:%s', host, port);
})
```

```
// API Endpoint to get data for specific photograph from database - /data/photoDescription/1234567
app.get('/data/photoDescription/:pid', function (req, res) {
      res.header("Access-Control-Allow-Origin", "*");
      res.header("Access-Control-Allow-Headers", "X-Requested-WithD");
      if(req.params.pid != ""){
                var pid = parseInt(req.params.pid);

                var sql = "SELECT * FROM metadata as a INNER JOIN photos as b ON a.pid = b.pid where a.pid = " + pid;

                console.log(sql);
                connection.query(sql, function(err, rows, fields) {
                        if (err) console.log("Err:" + err);
                        if(rows != undefined){
                                res.send(rows);
                        }else{
                                res.send("");
                        }
                });
        }else{
                res.send("");
        }
});

// Setup the server and print a string to the screen when server is ready
var server = app.listen(portNumber, function () {
  var host = server.address().address;
  var port = server.address().port;
  console.log('App listening at http://%s:%s', host, port);
})
```

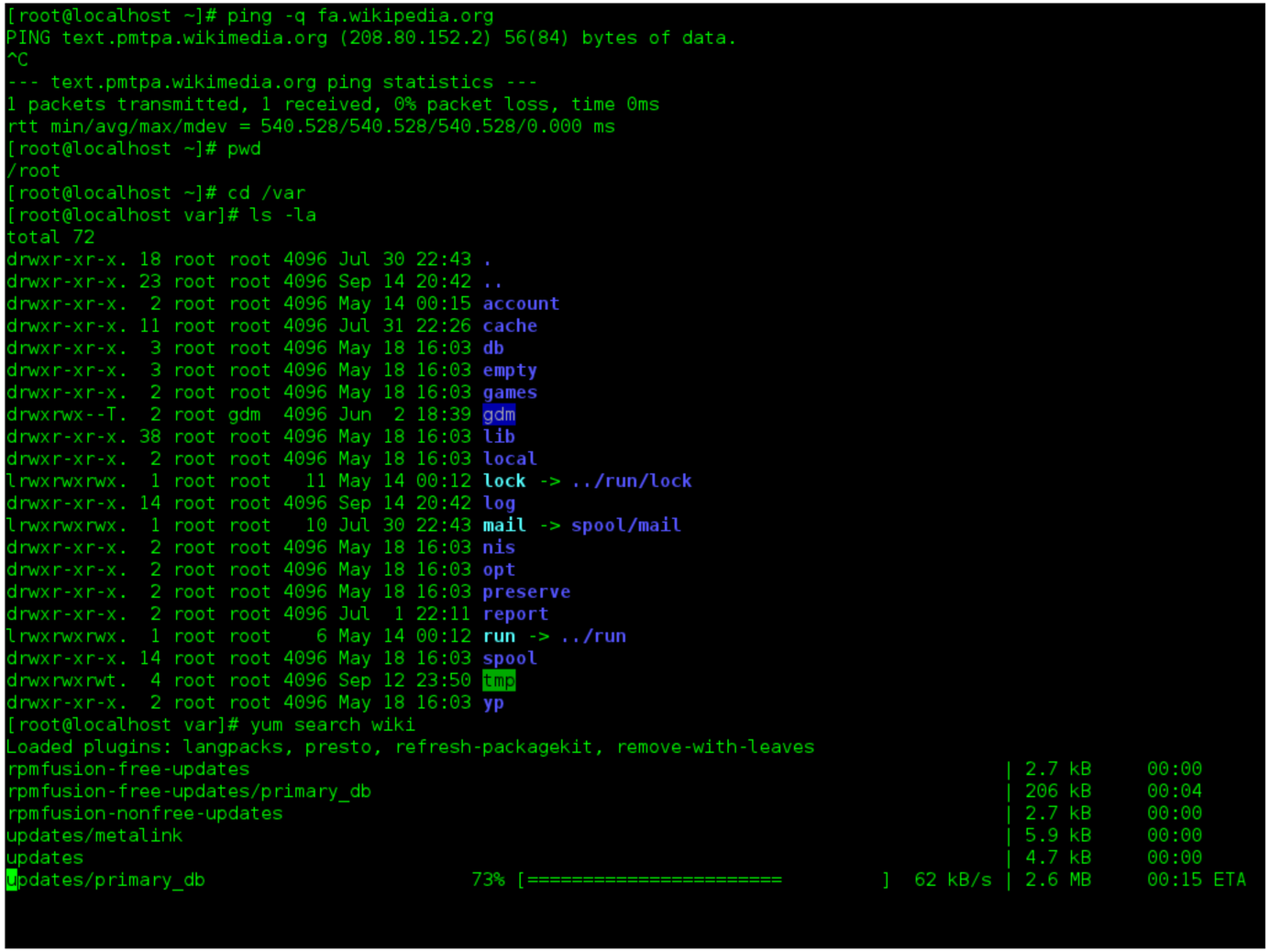
Quick guide to SSH and SCP

Working with remote servers



Quick guide to SSH and SCP

Working with remote servers



Control Remote Computers with a GUI

— Or Servers

Everything that can be done via commands

Example Connection

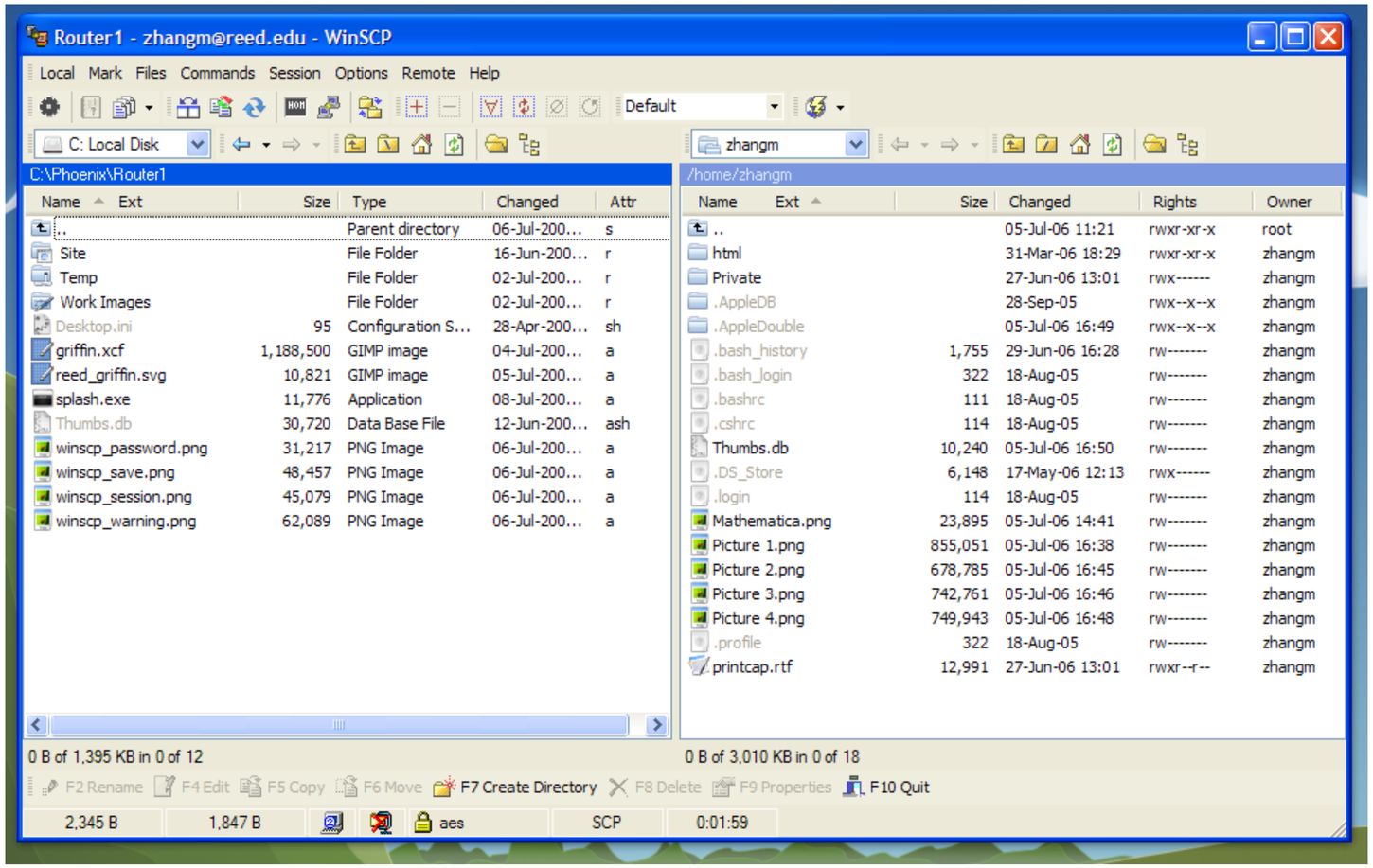
ssh ucfnsjg@dev.spatialdatacapture.org

Example Commands

```
touch foo     - Creates a file called foo
rm foo        - Deletes a file called foo
mkdir bar     - Creates a folder called bar
rmdir foo     - Removes a folder called foo
cd bar        - Change to directory bar
cd ..         - Go back 1 directory
ls .          - Prints all files in current directory
nano bar.js   - Opens (or creates if it doesn't exist)
                a file called bar.txt in a text editor
```

Quick guide to SSH and SCP

Working with remote servers



SFTP - Secure File Transfer Protocol (Uses SCP to transfer files)

WinSCP or CyberDuck

Workshop: Week 9

Building your own data API and extended the Interactive Map

- Extend the Interactive Viewer for Flickr Photos
  - Capture user interactions and pass to the API
  - Draw Markers on the page that respond to user input
- Build your own API
  - Learn how to maintain and share data with an open API
  - Develop SQL skills from Week 1 - 3
  - Build new API endpoints to get camera type from Database

Next Week

More JavaScript practise, graphs and animations

10 Real-time data visualisation

Learn how to use timers to animate elements and use jQuery to draw interactive elements and graphs that update in realtime.

Questions?

steven.gray@ucl.ac.uk
@frogo
020 3108 3886

Before you start

https://gist.github.com/sjg/ef74829dfa05388471c5

https://goo.gl/A10UYD

Run on your MySQL Table ‘photo_locations’
and edit with your username in the SQL