Csci 4131
JSON Wrap-up
Node.js intro
AJAX Revisited

Lecture 15, October 24th

Fall 2018

Dr. Dan Challou

Logistics

- HW 4 is due this coming Friday, October 26th at 2:00pm. After that time, assignments will be accepted with penalty though EARLY Saturday Morning (2:00am), 10/27

  After 2:00am 10/27 – HW 4 submissions

  will not be accepted.

- Exam 1 is next Wednesday 10/31; In this classroom at the usual class time. Open Book, Open notes. No Electronics (Computers, Phones, Smart Watches, Google Googles, Alexa, …, etc.)

Logistics, continued

- **An Important Note on Exam I:**

If you don't notify me BEFORE the exam that you can't make it AND you don't have a university sanctioned excuse AND you are not here for the exam, you get a zero on the Exam

Note, you can notify me and all TA's of any issue in this regard via the Class Email:

csci4131f18_help@umn.edu

Slides Describing Exam Scope, Overview will be posted with Today's Lecture 15 Materials (October 24th)

- File name:

Csci4131Exam1_Info.pdf

Reading and Tutorials: JSON, Ajax, Node.js

- JSON
  - Sebesta – Chapters 10, Section 3.3
  - https://www.w3schools.com/js/js_json_intro.asp
  - https://www.json.org/
- AJAX
  - Sebesta – Chapter 10
  - https://www.w3schools.com/xml/ajax_intro.asp
- Node.js
  - https://www.w3schools.com/nodejs/default.asp
  - https://www.tutorialspoint.com/nodejs/
  - https://nodejs.org/en/docs/guides

Node.js training videos on Lynda

- You can access Lynda via the following link:
  http://lynda.umn.edu

Use your x.500 id and password to sign in.

The following videos are most helpful:

- 1. Node.js Essential Training (6h 22m - Detailed Node.js video)
- 2. Building a Website with Node.js and Express.js (3h 16m - Focusses on Express.js and Node.js)
- 3. Learning Node.js (1h 57m)

Last Time

- HTTP Response codes and exercise (which we reviewed last class)
- JSON
- Intro to AJAX

Questions?

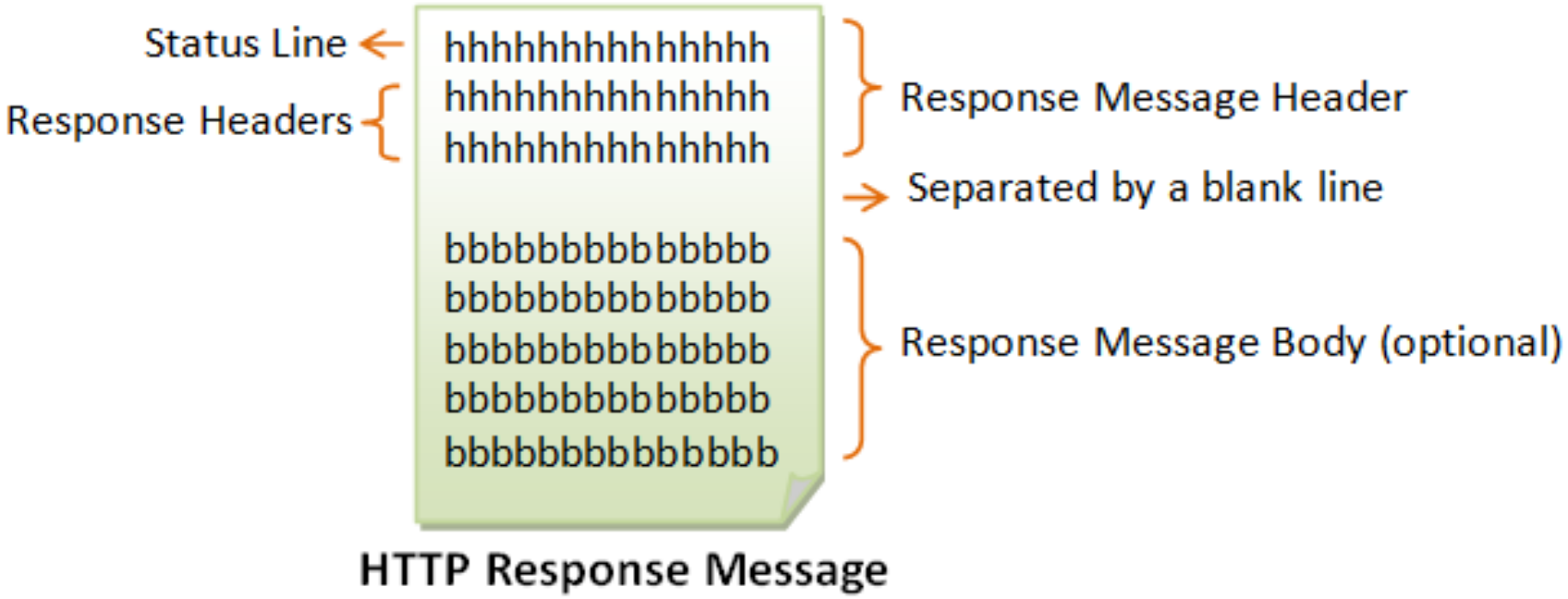
Today

- JSON Wrap-up
- Intro to Node.js
- AJAX revisited

Homework 4 notes

- When composing a response message containing an image or audio file
- The response message header should be encoded in utf-8
- BUT
- The response message body should be sent back in binary (so don’t encode it in utf-8)
- Helpful Python Code for Reading in a binary file: **contents = open(fname, 'rb')**

HTTP Response Message (From the Server)



Revew Exercise 1 From 10/22 (Lecture 14)

An HTTP 1.1. Compliant Python webserver is running on the host computer:

`csel-kh1262-11.cselabs.umn.edu.`

The server was executed (i.e., run) from the `/webserver` folder (directory) (which has world executable permissions) and is listening on port `9004`. The webserver pecontents of `/webserver` folder (directory) are as follows:

```
-rw------- 1 x500user CSEL-student 3275 Oct 3 07:25 server.py
-rw-r--r-- 1 x500user CSEL-student 1261 Oct 4 17:42 schedule.html
-rw-r--r-- 1 x500user CSEL-student  368 Oct 4 21:40 main.js
-rw-r--r-- 1 x500user CSEL-student 2561 Oct 4 17:42 my schedule.html
```

What HTTP 1.1 message response code will be sent to the client/browser after the following requests are sent by the client/browser:

```
GET /webserver/schedule.html HTTP/1.1
Host: csel-kh1262-11.cselabs.umn.edu
Accept: */*

DELETE /webserver/main.js HTTP/1.1
Host: csel-Kh1262-11.cselabs.umn.edu
Accept: */*

HEAD /webserver/my schedule.html HTTP/1.1
Host: csel-kh1262-11.cselabs.umn.edu
Accept: */*
```

Exercise 1: JSON – Submit via paper (with name and x.500 id) or electronically via Moodle (Lecture 15 Exercise Link)

Submit at end of class or by 4pm TODAY via Moodle

1. Create an HTML page with a **div** element. The div element should have an id named: **locations**
2. Add the JavaScript necessary to do the following:
3. Store the following TEXT in a JavaScript Variable in a JSON format:
4. "lat1": "44.95045", "lon1": "-93.345002"
5. "lat2": "44.95045", "lon2": "-93.345002“
6. Convert it to the text to a JSON object using JSON parse
7. Next, write JavaScript necessary to display the latitudes (**lat)** and longitudes (**lon**) in a list on the div element with the id named: **locations**

- Example: jsonexer1.html

(Hint – look at the json examples from last lecture)

Node.js:

(info obtained from:
https://www.w3schools.com/nodejs/nodejs_intro.asp
https://www.w3schools.com/nodejs/nodejs_get_started.asp
https://www.w3schools.com/nodejs/nodejs_modules.asp

- Node.js is an open source **server** framework
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript to implement and augment **server** functionality

Node.js can:

- generate dynamic page content
- create, open, read, write, delete, and close files on the server
- can collect form data
- can add, delete, modify data in your database

A Node.js file contains:

- tasks that will be executed when triggered by certain events
  - A typical event is someone trying to access a port on the server
  - Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

Node.js handles a file request as follows

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

- Thus node.js is single threaded, non-blocking, and asynchronous
- Here is how Php handles a file request:
  1. Sends the task to the computer's file system.
  2. Waits while the file system opens and reads the file.
  3. Returns the content to the client.
  4. Ready to handle the next request.
- So, for this task, PHP (and ASP) operate synchronously (and block).

Like Python, Node.js has lots of libraries (called modules) that you will want to include in your application

- For example:
  - http module, used to create a server

Example of a node.js file

```
var http = require('http');
http.createServer(function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/html'});
   res.end('Hello World!');
}).listen(8080);
```

Assuming it is in the file: myfirst.js

You run it from the command line by typing:
node myfirst.js

Then fire up your browser, and in the address bar type:
http:/localhost:8080

And, you will get the response: Hello World – rendered in your browser

Next Time

- Node.js intro revisted
- Ajax revisited
- HW 5