Csci 4131 Internet Programming

Lecture 15, March 13th
Spring 2024

Instructor: Dr. Dan Challou

Logistics (Csci 4131, Lecture 15, March 13th)

- Zybooks HW 7 due Sunday 3/17 (***topics are key to doing HW 5 successfully !!!***)
- Homework 4 due next Friday 3/22
- Homework 5 will be out next week – Using Fetch or AJAX, JSON, Node.js

Readings/Tutorials: Node.js, JSON, Fetch, AJAX – For HW 5!

Node.js References and Tutorials:
Your zyBook
https://www.w3schools.com/nodejs/
https://codeburst.io/the-only-nodejs-introduction-youll-ever-need-d969a47ef219
Video intro: https://www.youtube.com/watch?v=TlB_eWDSMt4

JSON References / Tutorials:
Your zyBook
https://www.w3schools.com/js/js_json_intro.asp
https://www.w3schools.com/js/js_json.asp
www.json.org
Optional: Chapter 10.3.3 Sebesta

FETCH References / Tutorials:
Your Zybook
https://www.w3schools.com/js/js_api_fetch.asp
https://javascript.info/fetch

AJAX References / Tutorials:
Your Zybook
https://www.w3schools.com/xml/ajax_intro.asp
Optional: Sebesta, Chapter 10

Questions ?

Agenda

- More on HW 4
- JavaScript Object Notation (JSON)
- Node.js Revisited
- AJAX and Fetch

Questions?

More on HW4 – see HW2 server

JavaScript Object Notation - Revisited

- JavaScript Object Notation (JSON)
- References
  - Your Zybook
  - https://www.w3schools.com/js/js_json_intro.asp
  - https://www.w3schools.com/js/js_json.asp
  - www.json.org
  - Optional: Chapter 10.3.3 Sebesta

JSON

- Lightweight data interchange and storage format
- Self-documenting – human readable and writeable
- It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999.
- JSON is a text format that is completely language independent **BUT**
- It uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

Why JSON

- Pile's o data stored out there on the internet/www in various formats
  - Text files
  - CSV files
  - XML files
  - JSON
    - JSON is Compact, Readable, easy to transport – so it has become the storage format of choice (NoSQL databases – mongo, etc)

- JSON is built using one or both of the following two structures:
  - A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array
  - An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence

JSON Values Can Be:

- A number (integer or floating point)
- A string (in double quotes)
- A Boolean (true or false)
- An array (in square brackets)
- An object (in curly braces)
- null

JSON Objects / JSON Arrays

- JSON objects are written inside curly braces.
- Just like in JavaScript, objects can contain multiple name/values pairs:
  - e.g., {"firstName":"John", "lastName":"Doe"}

- JSON arrays are written inside square brackets.

- As in JavaScript, an array can contain multiple objects:

-
  ```
  {"employees":[
     {"firstName":"John", "lastName":"Doe"},
     {"firstName":"Anna", "lastName":"Smith"},
     {"firstName":"Peter", "lastName":"Jones"}
  ]}
  ```
- The object "employees" is an JavaScript object with a value that is an array containing three objects. Each object in the array is a record of a person (with a first name and a last name).

Have you used JSON in this Course Before?

Where? (Please Share!!!)

JSON Uses JavaScript Syntax

Example:

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

The first entry in the JavaScript object array can be accessed as follows:

```
                employees[0].firstName + " " + employees[0].lastName;
```

The content returned will be:

John Doe

Data in the array can be modified as follows:

```
employees[0].firstName = "Gilbert";
var employees = [
   {"firstName":“Gilbert", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter", "lastName": "Jones"}
];
```

Creating JSON Objects from a string

The following creates a JavaScript string containing JSON syntax:

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

The JavaScript function **JSON.parse(*text*)** can be used to convert text in a JSON format into a JavaScript object:

```
var obj = JSON.parse(text); // creates a json object from the string text and
                            //  associates the object with the identifier obj
```

Question – THINK /PAIR / SHARE – 2 minutes:

alert(obj.employees[1].lastName) **// what is shown in the alert box?**

Creating a string from a JSON object

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
var obj = JSON.parse(text);
```

// THINK / PAIR / SHARE – 3.5 minutes

alert(JSON.stringify(obj)); **// What is shown in the alert box???**

Example – JSON to JavaScript Objects

jsonex1.html

An HTML and JAVASCRIPT example that starts with text stored in JSON notation

Converts the text to a JavaScript Object

Displays the contents of the Object

Example – JSON to JavaScript Objects

```
<!DOCTYPE html>
<html>
<body>

<h2>JSON Object Creation in JavaScript</h2>

<p id="demo"></p>

<script>
var text = '{"name":"President Biden","streetaddress":"1600 Pennsylvania Ave.","phone":"202 4561414"}'

var obj = JSON.parse(text);


document.getElementById("demo").innerHTML =
obj.name + "<br>" +
obj.streetaddress + "<br>" +
obj.phone;
</script>

</body>
</html>
```

jsonex1.html

Example 2: JSON to JavaScript Arrays

[jsonex2.html]

An HTML and JAVASCRIPT example that starts with text stored in JavaScript Array notation

Converts the text to a JavaScript Array

Displays the contents of the Array

Example 2: JSON to JavaScript Arrays

```
<!DOCTYPE html>
<html>
<body>

<h2>JSON Array Creation in JavaScript</h2>
        <p id="result"></p>

        <script>
        var nums = '["200","400","600","800"]';
        var anarray = JSON.parse(nums);

        var sum = 0;
        for (i = 0; i < anarray.length; i++) {
                sum += parseInt(anarray[i]);
        }
        document.getElementById("result").innerHTML = sum;
        </script>
</body>
</html>
```

jsonex2.html

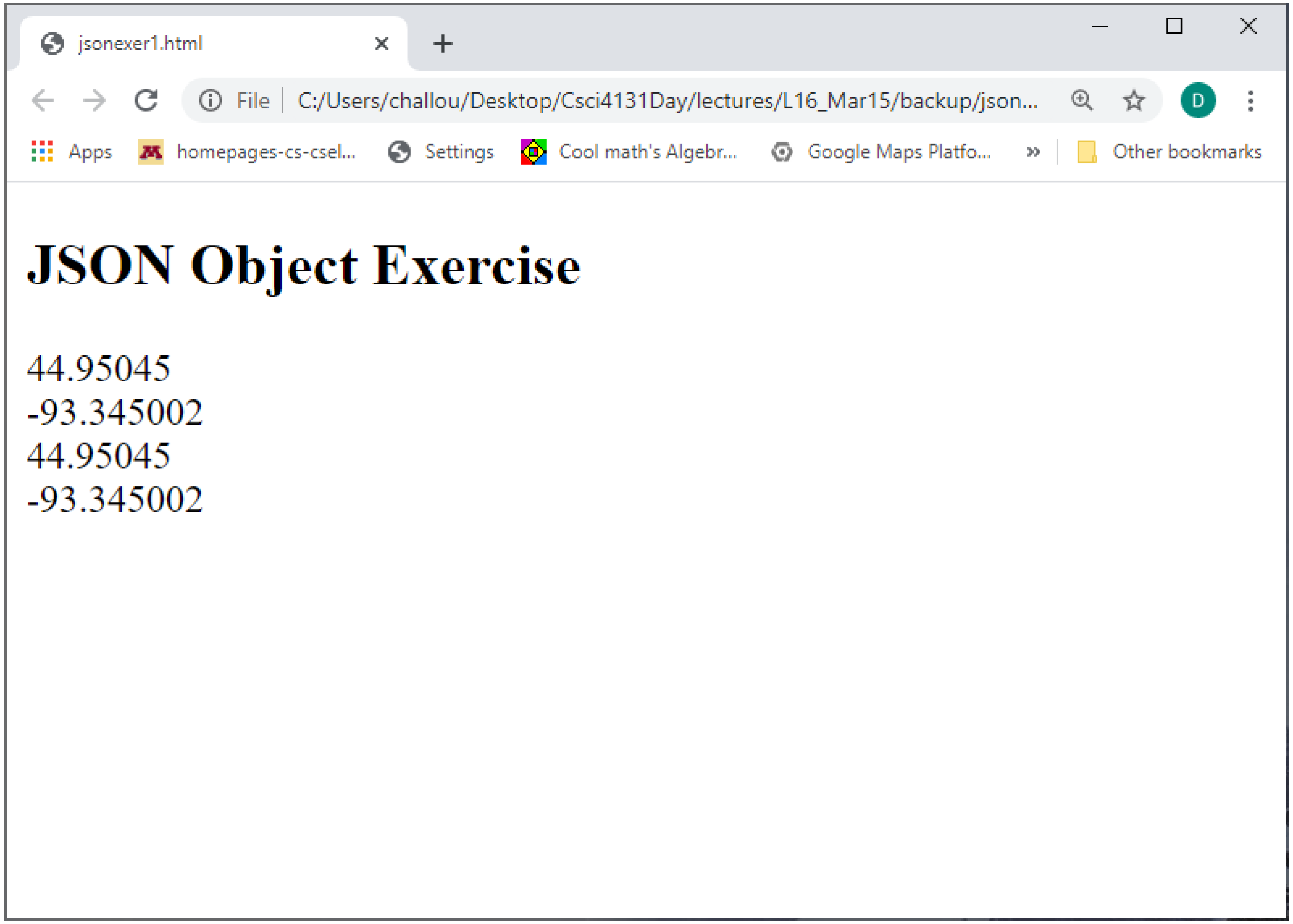
Exercise 1: JSON – Submit via Canvas Lecture 15 Exercise 1 Submission Item in the Week 8 Module

Think/Pair - submit

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 the text to a JSON object using **JSON.parse(thing_to_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**
8. Convert the JSON object **obj** back to a string format using **JSON.stringify(thing_to_stringify)** and display the result in an alert box

Example: jsonexer1.html

Possible output from Exercise 1



JSON Object Exercise

```
44.95045
-93.345002
44.95045
-93.345002
```

NODE.js revisited

- **Code – along activity – files index.html located in week 8 module with lecture 8 materials**

1. Log into a CSE Labs machine (using Vole or Putty)

2. Download the files: **SimpleFs.js**, and **index.html** from the week 8 module on Canvas

3. BUT **edit** SimpleFs.js so it runs on a different port as follows:

    i. Set the port to: 9 OR 8 + Three digits from your x.500 id

    So, for example, for my x.500 id: chal0001, I would run on port 8001
    e.g., % node SimpleFs.js

4 . Then, in your browser address bar, request the file: **index.html**

For example, for port 8001, type: http://localhost:8001/index.js

And, the sentence:

A simple Web Page

should be rendered in your browser!

AJAX, and its newer version fetch

- Enable web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. ***This means that it is possible to update parts of a web page, without reloading the whole page.***
- Web pages that do not use AJAX reload the entire page if any content on the page changes

AJAX – Based on Internet Standards

- Uses a combination of:
  - XMLHttpRequest object (to exchange data asynchronously with a server)
  - JavaScript/DOM (to display/interact with the information)
  - CSS (to style the data)
  - XML (often used as the format for transferring data) – but can be JSON or just plain text

Who uses AJAX?

- Google (Gmail, Maps and Suggest)
- Facebook (tabs)
- Youtube

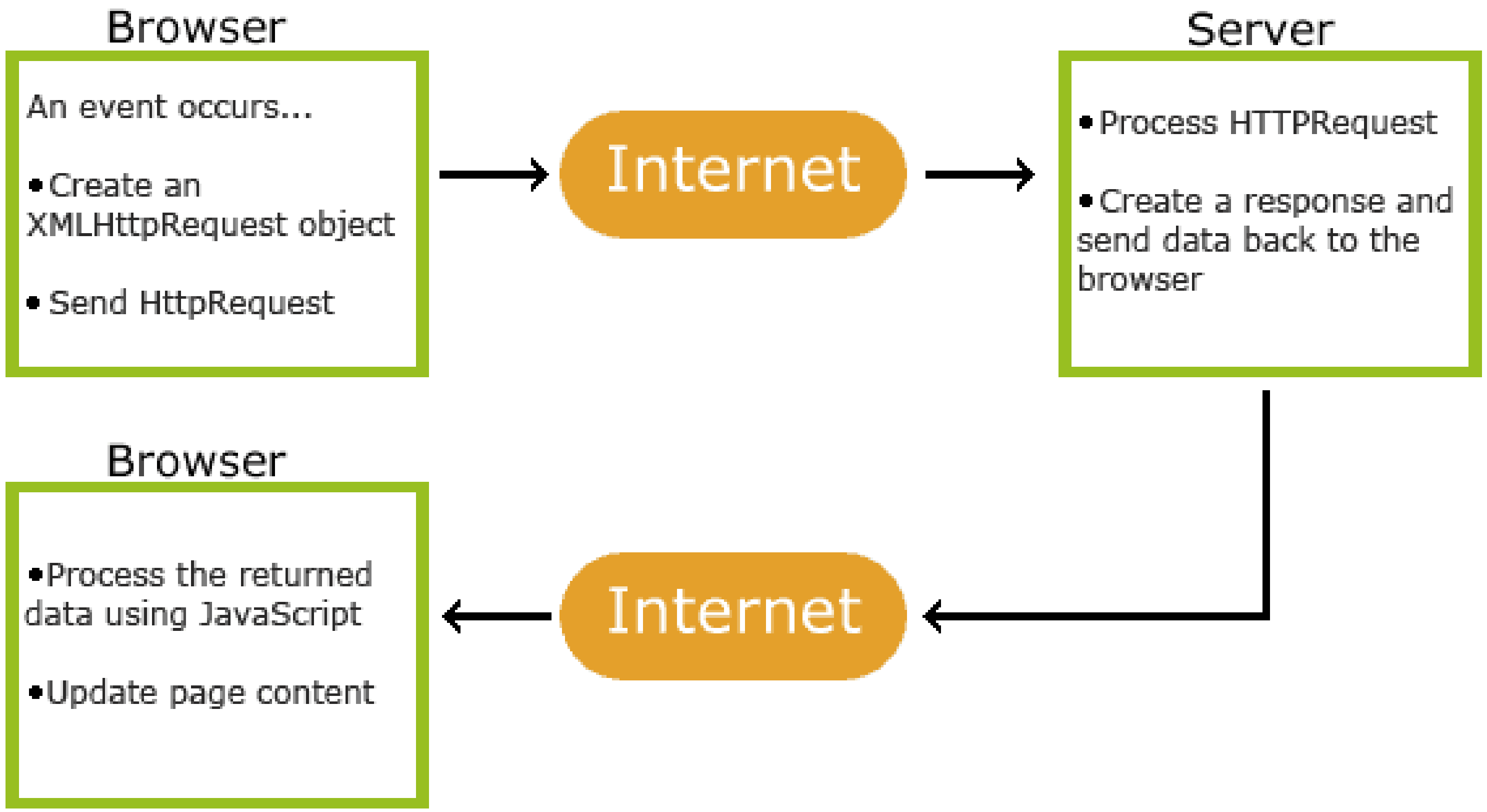
- Source: http://www.w3schools.com/php/php_ajax_intro.asp

The name AJAX (or AJAJ) is a bit of a misnomer

- Asynchronous JavaScript can be used to retrieve data stored in various formats including:
  - Text
  - Images
  - JSON (in string form)
  - XML
  - ???

How Do AJAX (and Fetch) Work? (How do they Get HTML, CSS, JAVASCRIPT, JSON, XML FILES FROM SERVER)?

Step 0 – user requests webpage from server, and server
Returns page, browser renders page
Step 1, before – Ajax/Fetch enabled web page obtained from Server


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Browser
• Process the returned data using JavaScript
• Update page content
Internet

Source: http://www.w3schools.com/php/php_ajax_intro.asp

The XMLHttpRequest Object

- This is the backbone of AJAX
- The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Creating an XMLHttpRequest Object

- Syntax for creating an XMLHttpRequest object:

    variable=new XMLHttpRequest();

Key Event for : The ***onreadystatechange*** Event

- **When an AJAX request to a server is sent,**
  we want our webpage (which sent the AJAX request)
  to perform some actions based on the response.

- **The** *onreadystatechange* **event is triggered every time the** *readyState* **changes.**

- **The** *readyState* **property holds the status of the XMLHttpRequest.**

- **We attach a callback function to the** *onreadystatechange* **event,**
  which will execute each time the server sends a response

Source:http://www.w3schools.com/ajax/ajax_xmlhttprequest_onreadystatechange.asp

Three Important Properties of the onreadystatechange event:

When status == 200, and state =4, we have obtained the response from our initial request

Property	Description
onreadystate change	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

Source:http://www.w3schools.com/ajax/ajax_xmlhttprequest_onreadystatechange.asp

Example of AJAX in Action – reading a text file

https://www-users.cs.umn.edu/~challou/simpleAJAXex.html

Next Time

- Node.js revisited
- JSON Revisited
- Introduction to Fecth,AJAX