Csci 4131 Internet Programming

Lecture 16, March 18th
Spring 2024

Instructor: Dr. Dan Challou

Logistics (Csci 4131, Lecture 16, March 18th)

- Homework 4 due next Friday 3/22
- Zybooks HW 8 due Sunday 3/24 (***topics are key to doing HW 5 and 6 successfully !!!***)
- **Exam 2 next Wednesday 3/27** – emphasis on topics covered since the last exam in Week 5
- Homework 5 will be out this 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 ?

Technologies needed for HW 5

- All the stuff from the first ½ of the course plus:
- Node.js (You will use it to construct a HTTP server)
- JSON
- FETCH
  - OR
- AJAX

Agenda

- Last Time
  - Node.js revisited
  - JavaScript Object Notation (JSON)
- Today
  - JSON wrapped up
  - Final 2 cents on HW4
  - AJAX and Fetch
  - Node.js revisited

Final HW-4 related items

File Permissions

- All directories should have permissions: 755
  - For example: **`chmod 755 myDirectory`**
  - myDirectory will have permissions: rwxr-xr-x
- All files except 1 should have permissions: 644
  - For example: **`chmod 644 myFile.html`** (or .png, .jpg, .mp3, .mp4, etc)
  - myFile.html will have permissions: rw-r--r--
- The file you use to test for permissions, should have permissions: 640
  - For example: **`chmod 640 private.html`** should have permissions: rw-r-----

To test for permission denied (403), you will need to test on a machine with a Unix or Linux Os (Mac, PC running Unix or CSE Labs Machines)

Favicon Errors:

- FaviconErr.pdf

Sketch of Python for building an HTTP response message to a get Request to return a text/html/css/js or binary file

```
After checking for calculator and redirect in the method **do_Get**
In either **do_Get** or **handle_request**
        Check to see if file exists (if not, return a 404)
        If file found check if other permissions set (if not, return a 403)
        otherwise, in handle request
                Find the mime type using file extension (suffix) from file name being requested
                mime = get_file_mime_type(file_extension)
                If  should_return_binary(file_extension) == false
                        return read text file + content type based on file extension (mime type)
                else
                        return read binary file + content type based on file extension (mime type)
```

Ins and Outs of HW4 server returning redirect request as a response

The following is sent from MyServer.html to your server running on port 9001
OR
typed in Browser Address Bar
http://localhost:9001/redirect?query_string=Gopher+Hockey&provider=YouTube

Think/Pair/Share – ***What is the request line of the HTTP message that arrives at the Server?*** (**2 minutes**)

Message from Server to Browser

```
HTTP/1.1 307 Redirect
Location: https://www.youtube.com/results?search_query=Gopher+Hockey
Content-Length: 0
```

And, time to move on…

Review Lecture 15, Exercise 1: JSON

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

AJAX

- Not a cleaning product that is stronger than dirt ***(does anyone even get this reference? if so, please nod your head discreetly!)***
- AJAX = Asynchronous JavaScript and XML
- Enables the implementation of more efficient web pages

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

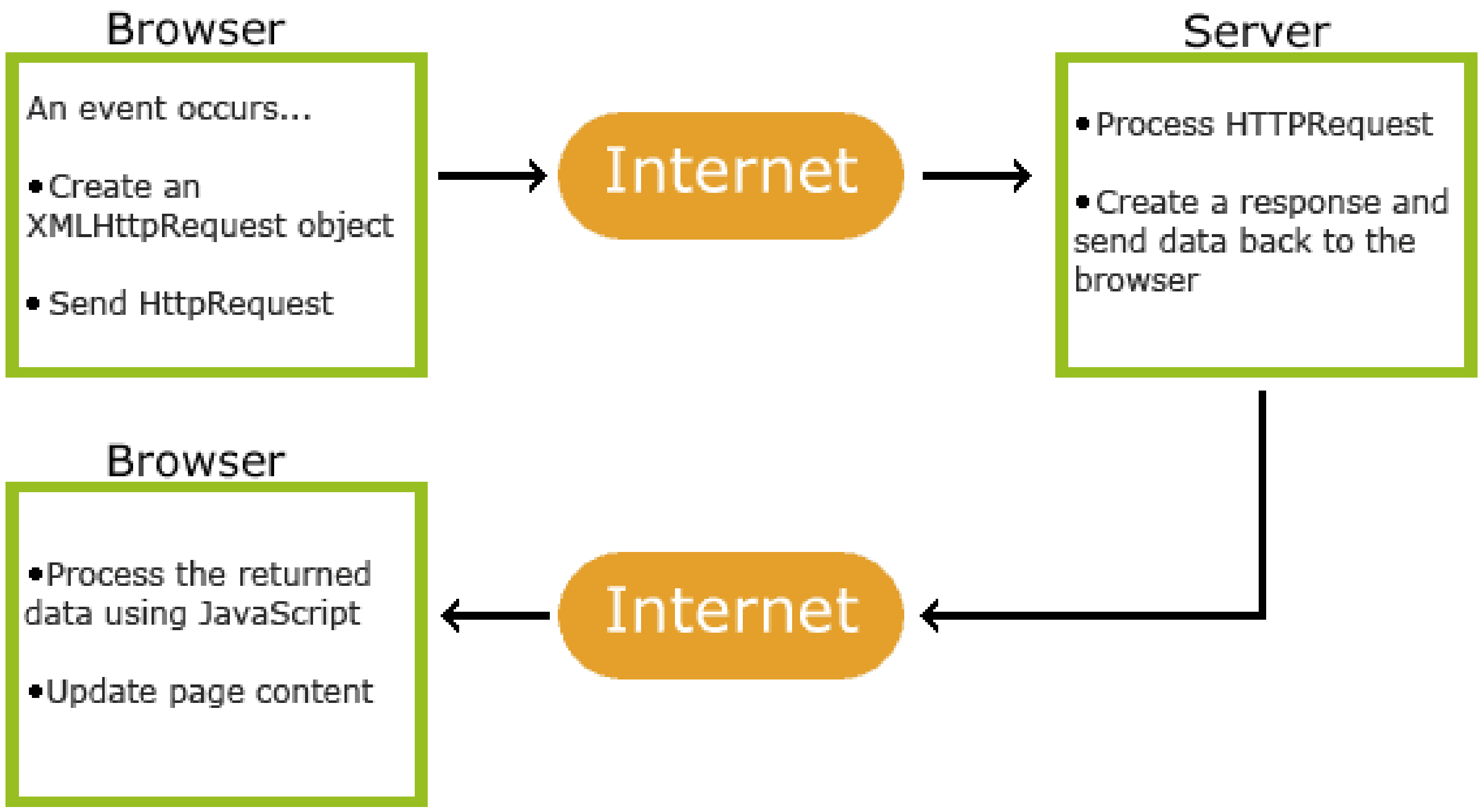
- Source: http://www.w3schools.com/php/php_ajax_intro.asp

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 Does AJAX Work? (How doe it 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

Note, the example returns a string

- And strings can contain JSON objects / arrays (or an object with an array or …)

AJAX Adheres to A “Same Origin” Policy

- https://en.wikipedia.org/wiki/Same-origin_policy

- **Why is that important?**
- **When might a Same Origin Policy be a problem?**

- **(Think/Pair/Share – 3 minutes)**

Here is another example (AJAX returns JSON string, which is parsed into a JSON Object and then used in the JavaScript Code)

- https://www-users.cs.umn.edu/~chal0006/JSON/JSONreqex.html

Fetch – newer alternative to Ajax

-
  ```
  async function getText(url) {
    let response = await fetch(url);
    console.log("Status is: " + response.status);
    let myText = await response.text();
    alert(myText);
  }
  ```

Alternate Form

```
fetch(url)
        .then(response => {console.log("Status is: " +
                                                response.status);
                return response.text();
                })
        .then(myText=>{console.log(text);})
        .catch(error=> console.log("Request failed",error));
```

Comparison

- AJAX vs Fetch

- Download files:
  - **`JSONreqex.html`** and **`FetchJsonLat.html`**

  From week 9, Lecture 16 materials on Canvas

Lets review an example of how **node.js** and **AJAX/Fetch** work together

- Download files: **`StudentFileServerAF.js`**
  and locations.txt from week 9 lecture 16
  materials on Canvas

  – We'll review them and run them – and you can too (if you have download the files above, and the files **`JSONreqex.html`** and **`FetchJsonLat.html`**
  From the week 9, Lecture 16 materials on Canvas)
  onto a computer where you can run node.js
  (on your computer or remotely on a cse-labs machine (via
  vole or ssh)

Next Time

- Node.js revisited
- Ajax / Fetch Revisited
- Introduction to RDBMS?